Package: rtemis (via r-universe)

October 27, 2024

Version 0.98.1

Title Machine Learning and Visualization

Date 2024-10-15

Description Advanced Machine Learning and Visualization. Unsupervised Learning (Clustering, Decomposition), Supervised Learning (Classification, Regression), Cross-Decomposition, Bagging, Boosting, Meta-models. Static and interactive graphics.

License GPL (>= 3)

URL https://rtemis.org

BugReports https://github.com/egenn/rtemis/issues

ByteCompile yes

Depends R (>= 4.1.0)

Imports grDevices, graphics, stats, methods, parallel, utils, data.table, R6, future, htmltools

Suggests ada, arm, arrow, bartMachine, base64enc, bit64, dbscan, car, cluster, colorspace, C50, data.tree, DBI, dendextend (>= 0.18.0), DiagrammeR, DiagrammeRsvg, doParallel, dplyr, duckdb, earth, EMCluster, evtree, e1071, fansi, fastICA, flexclust, FNN, fpc, fs, future.apply, gbm, geojsonio, geosphere, ggplot2, glmnet, GPArotation, grid, gsubfn, hal9001, haven, heatmaply, hopach, htmlwidgets, h2o, ica, igraph, imager, inTrees, JuliaCall, jsonlite, keras, kernlab, knitr, lars, lavaan, leaflet, leaps, lightgbm, matrixStats, MASS, mda, meanShiftR, mgcv, mlbench, mice, missForest, missRanger, networkD3, nlme, nnet, NMF, np, nsprcomp, openxlsx, partykit, pbapply, plotly, polspline, pROC, progress, progressr, psych, PMA, png, plotrix, plyr, pmml, PRROC, pvclust, qrnn, randomForest, randomForestSRC, ranger, RDRToolbox, reactable, RColorBrewer, rmarkdown, ROCR, rpart, rpart.plot, rstudioapi, Rtsne, rsvg, R.utils, scales (>= 0.2.5), seqinr, sgd, sparklyr, sparseLDA, splines2, spls, survival, tensorflow, tgp, threejs, testthat (>= 3.0.0), usmap, uwot, vegan, visNetwork, vroom, xgboost

Encoding UTF-8

Config/testthat/edition 3

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

LazyData true

Repository https://ictml-project.r-universe.dev

RemoteUrl https://github.com/egenn/rtemis

RemoteRef HEAD

RemoteSha 6e47fc7fe6a03dd8c25ca0a48519b829dc4af01e

Contents

rtemis-package 11
any_constant 13
as.data.tree.linadleaves 13
as.data.tree.rpart 14
as.data.tree.shyoptleaves 14
auc 15
auc_pairs 16
bacc 16
betas.lihad 17
bias_variance 18
binmat2vec 19
bold 19
boost 20
bootstrap 23
brier_score 23
calibrate 24
calibrate_cv 25
catrange 26
catsize 27
checkpoint_earlystop 27
check_data 29
check_files 30
chill 31
class_error 31
class_imbalance 32
clean_colnames 33
clean_names 33
clust 34
coef.lihad 34
col2grayscale 35
col2hex 35
colMax 36
colorAdjust 36

colorGrad . . . 37
colorGrad.x . . . 39
colorgradient.x . . . 40
colorMix . . . 41
colorOp . . . 41
color_fade . . . 42
color_invertRGB . . . 43
color_mean . . . 43
color_order . . . 44
color_separate . . . 45
color_sqdist . . . 45
cols2list . . . 46
create_config . . . 46
crules . . . 47
c_CMeans . . . 48
c_DBSCAN . . . 49
c_EMC . . . 50
c_H2OKMeans . . . 51
c_HARDCL . . . 52
c_HOPACH . . . 53
c_KMeans . . . 54
c_MeanShift . . . 55
c_NGAS . . . 56
c_PAM . . . 57
c_PAMK . . . 58
c_SPEC . . . 59
dat2bsplinemat . . . 60
dat2poly . . . 61
date2factor . . . 62
date2ym . . . 63
date2yq . . . 63
ddb_collect . . . 64
ddb_data . . . 65
ddSci . . . 66
decom . . . 67
dependency_check . . . 68
desaturate . . . 68
describe . . . 69
df_movecolumn . . . 69
distillTreeRules . . . 70
dplot3_addtree . . . 71
dplot3_bar . . . 72
dplot3_box . . . 75
dplot3_calibration . . . 80
dplot3_cart . . . 82
dplot3_conf . . . 84
dplot3_fit . . . 86
dplot3_graphd3 . . . 86

dplot3_graphjs . . . 87
dplot3_heatmap . . . 90
dplot3_leaflet . . . 93
dplot3_linad . . . 95
dplot3_pie . . . 97
dplot3_protein . . . 99
dplot3_pvals . . . 105
dplot3_spectrogram . . . 106
dplot3_table . . . 109
dplot3_ts . . . 110
dplot3_varimp . . . 113
dplot3_volcano . . . 114
dplot3_x . . . 118
dplot3_xt . . . 122
dplot3_xy . . . 126
dplot3_xyz . . . 132
drange . . . 136
dt_check_unique . . . 136
dt_describe . . . 137
dt_get_column_attr . . . 138
dt_get_duplicates . . . 138
dt_get_factor_levels . . . 139
dt_index_attr . . . 139
dt_inspect_type . . . 140
dt_keybin_reshape . . . 140
dt_merge . . . 141
dt_names_by_attr . . . 142
dt_names_by_class . . . 143
dt_Nuniqueperfeat . . . 143
dt_pctmatch . . . 144
dt_pctmissing . . . 144
dt_set_autotypes . . . 145
dt_set_cleanfactorlevels . . . 145
dt_set_clean_all . . . 146
dt_set_logical2factor . . . 146
d_H2OAE . . . 147
d_H2OGLRM . . . 150
d_ICA . . . 152
d_Isomap . . . 153
d_KPCA . . . 154
d_LLE . . . 155
d_MDS . . . 156
d_NMF . . . 157
d_PCA . . . 158
d_SPCA . . . 159
d_SVD . . . 161
d_TSNE . . . 162
d_UMAP . . . 163

earlystop 164
expand.boost 165
explain 167
f1 167
factoryze 168
factor_harmonize 169
factor_NA2missing 170
fct_describe 170
format.call 171
formatLightRules 172
formatRules 172
fwhm2sigma 173
get-names 174
getnames 174
getnamesandtypes 175
get_loaded_pkg_version 175
get_mode 176
get_rules 177
get_vars_from_rules 177
ggtheme_dark 178
ggtheme_light 179
glmLite 180
gmean 182
gp 182
graph_node_metrics 183
gridCheck 184
gtTable 184
htest 185
inspect_type 187
invlogit 187
is_constant 188
is_discrete 189
kfold 189
labelify 190
lincoef 191
list2csv 193
logistic 193
logit 194
logloss 194
loocv 195
lotri2edgeList 195
lsapply 196
make_key 196
massGAM 197
massGLAM 198
massGLM 200
massUni 201
matchcases 202

mergelongtreatment . . . 203
meta_mod . . . 204
mgetnames . . . 206
mhist . . . 206
mlegend . . . 208
mod_error . . . 209
mplot3_adsr . . . 210
mplot3_bar . . . 213
mplot3_box . . . 215
mplot3_conf . . . 218
mplot3_confbin . . . 221
mplot3_decision . . . 223
mplot3_fit . . . 224
mplot3_fret . . . 225
mplot3_graph . . . 226
mplot3_harmonograph . . . 229
mplot3_heatmap . . . 230
mplot3_img . . . 234
mplot3_laterality . . . 236
mplot3_lolli . . . 238
mplot3_missing . . . 240
mplot3_mosaic . . . 241
mplot3_pr . . . 243
mplot3_prp . . . 245
mplot3_res . . . 245
mplot3_roc . . . 246
mplot3_surv . . . 248
mplot3_survfit . . . 250
mplot3_varimp . . . 253
mplot3_x . . . 254
mplot3_xy . . . 259
mplot3_xym . . . 266
mplot_AGGTEobj . . . 268
mplot_hsv . . . 269
mplot_raster . . . 270
mse . . . 272
multigplot . . . 272
nCr . . . 273
oddsratio . . . 273
oddsratiotable . . . 274
oneHot . . . 274
onehot2factor . . . 276
palettize . . . 277
permute . . . 277
pfread . . . 278
plot.massGAM . . . 279
plot.massGLM . . . 280
plot.resample . . . 281

plot.rtModCVCalibration . . . 281
plot.rtTest . . . 283
plotly.heat . . . 283
precision . . . 284
predict.addtree . . . 285
predict.boost . . . 285
predict.cartLite . . . 286
predict.cartLiteBoostTV . . . 287
predict.glmLite . . . 288
predict.glmLiteBoostTV . . . 288
predict.hytboost . . . 289
predict.hytboostnow . . . 290
predict.hytreenow . . . 290
predict.hytreew . . . 291
predict.LightRuleFit . . . 292
predict.lihad . . . 293
predict.linadleaves . . . 294
predict.nlareg . . . 295
predict.nullmod . . . 296
predict.rtBSplines . . . 296
predict.rtModCVCalibration . . . 297
predict.rtTLS . . . 297
predict.rulefit . . . 298
preprocess . . . 298
preprocess_ . . . 302
present . . . 304
present_gridsearch . . . 306
previewcolor . . . 306
print.addtree . . . 308
print.boost . . . 309
print.cartLiteBoostTV . . . 309
print.CheckData . . . 310
print.class_error . . . 310
print.glmLiteBoostTV . . . 311
print.gridSearch . . . 311
print.hytboost . . . 312
print.hytboostnow . . . 312
print.lihad . . . 313
print.linadleaves . . . 313
print.massGAM . . . 314
print.massGLM . . . 314
print.regError . . . 315
print.resample . . . 315
print.rtBiasVariance . . . 316
print.rtDecom . . . 316
print.rtTLS . . . 317
print.surv_error . . . 317
prune.addtree . . . 318

psd . . . 318
qstat . . . 319
read . . . 319
read_config . . . 321
recycle . . . 322
reg_error . . . 322
relu . . . 323
resample . . . 324
reverseLevels . . . 325
revfactorlevels . . . 326
rfVarSelect . . . 326
rnormmat . . . 327
rowMax . . . 328
rsd . . . 328
rsq . . . 329
rstudio_theme_rtemis . . . 329
rtClust-methods . . . 330
rtemis_palette . . . 330
rtInitProjectDir . . . 331
rtlayout . . . 331
rtMeta-methods . . . 332
rtMod-methods . . . 332
rtModBag-methods . . . 334
rtModClass-class . . . 335
rtModCV-methods . . . 338
rtModLite-methods . . . 339
rtModLog-class . . . 340
rtModLogger-class . . . 341
rtpalette . . . 343
rtPalettes . . . 344
rtROC . . . 351
rtset . . . 352
rtversion . . . 352
rtXDecom-class . . . 352
rt_reactable . . . 354
rt_save . . . 355
ruleDist . . . 356
rules2medmod . . . 357
runifmat . . . 357
savePMML . . . 358
se . . . 359
selectiter . . . 359
select_clust . . . 360
select_decom . . . 360
select_learn . . . 361
sensitivity . . . 362
seql . . . 362
setdiffsym . . . 363

setup.bag.resample . . . 363
setup.color . . . 364
setup.cv.resample . . . 365
setup.decompose . . . 366
setup.earlystop . . . 366
setup.GBM . . . 367
setup.grid.resample . . . 368
setup.LightRuleFit . . . 369
setup.LIHAD . . . 370
setup.lincoef . . . 371
setup.MARS . . . 372
setup.meta.resample . . . 373
setup.preprocess . . . 373
setup.Ranger . . . 376
setup.resample . . . 377
sge_submit . . . 377
sigmoid . . . 379
size . . . 379
softmax . . . 380
softplus . . . 380
sortedlines . . . 380
sparsernorm . . . 381
sparseVectorSummary . . . 382
sparsify . . . 382
specificity . . . 383
stderror . . . 383
strat.boot . . . 384
strat.sub . . . 385
strata2factor . . . 385
summarize . . . 386
summary.massGAM . . . 387
summary.massGLM . . . 387
surv_error . . . 388
svd1 . . . 388
synth_multimodal . . . 389
synth_reg_data . . . 391
s_AdaBoost . . . 392
s_AddTree . . . 394
s_BART . . . 397
s_BayesGLM . . . 400
s_BRUTO . . . 403
s_C50 . . . 405
s_CART . . . 407
s_CTree . . . 410
s_EVTree . . . 413
s_GAM . . . 415
s_GBM . . . 417
s_GLM . . . 421

s_GLMNET . . . 424
s_GLMTree . . . 427
s_GLS . . . 430
s_H2ODL . . . 432
s_H2OGBM . . . 435
s_H2ORF . . . 438
s_HAL . . . 441
s_KNN . . . 443
s_LDA . . . 445
s_LightCART . . . 447
s_LightGBM . . . 450
s_LightRF . . . 455
s_LightRuleFit . . . 458
s_LIHAD . . . 461
s_LIHADBoost . . . 463
s_LINAD . . . 466
s_LINOA . . . 469
s_LM . . . 472
s_LMTree . . . 475
s_LOESS . . . 477
s_LOGISTIC . . . 479
s_MARS . . . 479
s_MLRF . . . 482
s_MULTINOM . . . 485
s_NBayes . . . 486
s_NLA . . . 487
s_NLS . . . 489
s_NW . . . 492
s_POLY . . . 494
s_PolyMARS . . . 494
s_PPR . . . 497
s_PSurv . . . 499
s_QDA . . . 501
s_QRNN . . . 503
s_Ranger . . . 505
s_RF . . . 509
s_RFSRC . . . 513
s_RLM . . . 515
s_RuleFit . . . 516
s_SDA . . . 518
s_SGD . . . 521
s_SPLS . . . 524
s_SVM . . . 527
s_TFN . . . 529
s_TLS . . . 533
s_XGBoost . . . 534
s_XRF . . . 539
table1 . . . 543

themes 544
theme_black 545
timeProc 558
tohtml 559
train_cv 559
tunable 563
typeset 563
uci_heart_failure 564
uniprot_get 565
uniquevalsperfeat 565
winsorize 566
xlsx2list 567
xselect_decom 568
xtdescribe 569
x_CCA 569
zip2longlat 571
zipdist 572
%BC% 572

Index **573**

`rtemis-package` **rtemis**: *Machine Learning and Visualization*

Description

Advanced Machine Learning made easy, efficient, reproducible

Online Documentation and Vignettes

`https://rtemis.org`

System Setup

There are some options you can define in your .Rprofile (usually found in your home directory), so you do not have to define each time you execute a function.

rt.theme General plotting theme; set to e.g. "whiteigrid" or "darkgraygrid"

rt.palette Name of default palette to use in plots. See options by running `rtpalette()`

rt.font Font family to use in plots.

rt.cores Number of cores to use. By default, rtemis will use available cores reported by future::availableCores(). In shared systems, you should limit this as appropriate.

future.plan Default plan to use for parallel processing.

Visualization

Static graphics are handled using the `mplot3` family. Dynamic graphics are handled using the `dplot3` family.

Supervised Learning

Functions for Regression and Classification begin with `s_*`. Run select_learn to get a list of available algorithms The documentation of each supervised learning function indicates in brackets, after the title whether the function supports classification, regression, and survival analysis `[C, R, S]`

Clustering

Functions for Clustering begin with `c_*`. Run select_clust to get a list of available algorithms

Decomposition

Functions for Decomposition and Dimensionality reduction begin with `d_*`. Run select_decom to get a list of available algorithms

Cross-Decomposition

Functions for Cross-Decomposition begin with `x_*`. Run xselect_decom to get a list of available algorithms

Meta-Modeling

Meta models are trained using `meta*` functions.

Notes

Function documentation includes input type (e.g. "String", "Integer", "Float"/"Numeric", etc) and range in interval notation where applicable. For example, Float: [0, 1)" means floats between 0 and 1 including 0, but excluding 1

For all classification models, the outcome should be provided as a factor, with the first level of the factor being the 'positive' class, if applicable. A character vector supplied as outcome will be converted to factors, where by default the levels are set alphabetically and therefore the positive class may not be set correctly.

Author(s)

Maintainer: E.D. Gennatas `<gennatas@gmail.com>` (ORCID)

See Also

Useful links:

- `https://rtemis.org`
- Report bugs at `https://github.com/egenn/rtemis/issues`

any_constant *Check for constant columns*

Description

Checks if any column of a data frame have zero variance

Usage

```
any_constant(x)
```

Arguments

`x`	Input Data Frame

Author(s)

E.D. Gennatas

as.data.tree.linadleaves *Convert `linadleaves` to `data.tree` object*

Description

Convert `linadleaves` to `data.tree` object

Usage

```
as.data.tree.linadleaves(object)
```

Arguments

`object`	`linadleaves` object

`as.data.tree.rpart` *Convert* `rpart` *rules to* `data.tree` *object*

Description

Convert an `rpart` object to a `data.tree` object, which can be plotted with dplot3_cart

Usage

```
as.data.tree.rpart(object, verbose = FALSE)
```

Arguments

`object`	`rpart` object
`verbose`	Logical: If TRUE, print messages to console

Value

`data.tree` object

Author(s)

E.D. Gennatas

`as.data.tree.shyoptleaves` *Convert* `shyoptleaves` *to* `data.tree` *object*

Description

Convert `shyoptleaves` to `data.tree` object

Usage

```
as.data.tree.shyoptleaves(object)
```

Arguments

`object`	`shyoptleaves` object

auc — *Area under the ROC Curve*

Description

Get the Area under the ROC curve to assess classifier performance.

Usage

```
auc(
  preds,
  labels,
  method = c("pROC", "ROCR", "auc_pairs"),
  verbose = FALSE,
  trace = 0
)
```

Arguments

`preds`	Numeric, Vector: Probabilities or model scores (e.g. c(.32, .75, .63), etc)
`labels`	True labels of outcomes (e.g. c(0, 1, 1))
`method`	Character: "pROC", "auc_pairs", or "ROCR": Method to use. Will use `pROC::roc`, auc_pairs, `ROCR::performance`, respectively.
`verbose`	Logical: If TRUE, print messages to output
`trace`	Integer: If > 0, print more messages to output

Details

Important Note: We assume that true labels are a factor where the first level is the "positive" case, a.k.a. the event. All methods used here, "pROC", "auc_pairs", "ROCR", have been setup to expect this. This goes against the default setting for both "pROC" and "ROCR", which will not give an AUC less than .5 because they will reorder levels. We don't want this because you can have a classifier perform worse than .5 and it can be very confusing if levels are reordered automatically and different functions give you different AUC.

Author(s)

EDG

Examples

```
## Not run:
preds <- c(0.7, 0.55, 0.45, 0.25, 0.6, 0.7, 0.2)
labels <- factor(c("a", "a", "a", "b", "b", "b", "b"))
auc(preds, labels, method = "ROCR")
auc(preds, labels, method = "pROC")
auc(preds, labels, method = "auc_pairs")
```

```
## End(Not run)
```

---

auc_pairs *Area under the Curve by pairwise concordance*

Description

Get the Area under the ROC curve to assess classifier performance using pairwise concordance

Usage

```
auc_pairs(estimated.score, true.labels, verbose = TRUE)
```

Arguments

estimated.score	Float, Vector: Probabilities or model scores (e.g. c(.32, .75, .63), etc)
true.labels	True labels of outcomes (e.g. c(0, 1, 1))
verbose	Logical: If TRUE, print messages to output

Details

The first level of `true.labels` must be the positive class, and high numbers in `estimated.score` should correspond to the positive class.

Examples

```
## Not run:
true.labels <- factor(c("a", "a", "a", "b", "b", "b", "b"))
estimated.score <- c(0.7, 0.55, 0.45, 0.25, 0.6, 0.7, 0.2)
auc_pairs(estimated.score, true.labels, verbose = TRUE)

## End(Not run)
```

---

bacc *Balanced Accuracy*

Description

Balanced Accuracy of a binary classifier

Usage

```
bacc(true, predicted, harmonize = FALSE, verbosity = 1)
```

Arguments

`true`	True labels
`predicted`	Estimated labels
`harmonize`	Logical: passed to sensitivity and specificity, which use factor_harmonize. Default = FALSE
`verbosity`	Integer: If > 0, print messages to console.

Details

BAcc = .5 * (Sensitivity + Specificity)

`betas.lihad` *Extract coefficients from Additive Tree leaves*

Description

Extract coefficients from Additive Tree leaves

Usage

```
betas.lihad(object, newdata, verbose = FALSE, trace = 0)
```

Arguments

`object`	`lihad` object
`newdata`	matrix/data.frame of features
`verbose`	Logical: If TRUE, print output to console
`trace`	Integer 0:2 Increase verbosity

Author(s)

E.D. Gennatas

bias_variance *Bias-Variance Decomposition*

Description

Bias-Variance Decomposition

Usage

```
bias_variance(
  x,
  y,
  mod,
  res1_train.p = 0.7,
  params = list(),
  resample.params = setup.resample(n.resamples = 100),
  seed = NULL,
  verbose = TRUE,
  res.verbose = FALSE,
  ...
)
```

Arguments

`x`	Predictors
`y`	Outcome
`mod`	Character: rtemis learner
`res1_train.p`	Numeric: Proportion of cases to use for training
`params`	List of `mod` parameters
`resample.params`	Output of setup.resample
`seed`	Integer: Seed for initial train/test split
`verbose`	Logical: If TRUE, print messages to console
`res.verbose`	Logical: passed to the learning function
`...`	Additional arguments passed to `resLearn`

Author(s)

E.D. Gennatas

`binmat2vec` *Binary matrix times character vector*

Description

Binary matrix times character vector

Usage

```
binmat2vec(x, labels = colnames(x))
```

Arguments

`x`	A binary matrix or data.frame
`labels`	Character vector length equal to `ncol(x)`

Value

a character vector

`bold` *String formatting utilities*

Description

String formatting utilities

Usage

```
bold(...)

italic(...)

underline(...)

hilite(..., col = "69;1", bold = TRUE)

hilitebig(x)

red(..., bold = FALSE)

green(..., bold = FALSE)

orange(..., bold = FALSE)
```

```
cyan(..., bold = FALSE)

magenta(..., bold = FALSE)

gray(..., bold = FALSE, sep = " ")

reset(...)
```

Arguments

...	Character objects to format
bold	Logical: If TRUE, use bold font
x	Numeric: Input
sep	Character: Separator

---

boost *Boost an* **rtemis** *learner for regression*

Description

Train an ensemble using boosting of any learner

Usage

```
boost(
  x,
  y = NULL,
  x.valid = NULL,
  y.valid = NULL,
  x.test = NULL,
  y.test = NULL,
  mod = "cart",
  resid = NULL,
  boost.obj = NULL,
  mod.params = list(),
  case.p = 1,
  weights = NULL,
  learning.rate = 0.1,
  earlystop.params = setup.earlystop(window = 30, window_decrease_pct_min = 0.01),
  earlystop.using = "train",
  tolerance = 0,
  tolerance.valid = 1e-05,
  max.iter = 10,
  init = NULL,
  x.name = NULL,
  y.name = NULL,
```

```
    question = NULL,
    base.verbose = FALSE,
    verbose = TRUE,
    trace = 0,
    print.progress.every = 5,
    print.error.plot = "final",
    prefix = NULL,
    plot.theme = rtTheme,
    plot.fitted = NULL,
    plot.predicted = NULL,
    print.plot = FALSE,
    print.base.plot = FALSE,
    plot.type = "l",
    outdir = NULL,
    ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.valid	Data.frame; optional: Validation data
y.valid	Float, vector; optional: Validation outcome
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
mod	Character: Algorithm to train base learners, for options, see select_learn. Default = "cart"
resid	Float, vector, length = length(y): Residuals to work on. Do not change unless you know what you're doing. Default = NULL, for regular boosting
boost.obj	(Internal use)
mod.params	Named list of arguments for mod
case.p	Float (0, 1]: Train each iteration using this perceent of cases. Default = 1, i.e. use all cases
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ifw, therefore set weights = NULL if using ifw. Note: If weight are provided, ifw is not used. Leave NULL if setting ifw = TRUE.
learning.rate	Float (0, 1] Learning rate for the additive steps
earlystop.params	List with early stopping parameters. Set using setup.earlystop
earlystop.using	Character: "train" or "valid". For the latter, requires x.valid
tolerance	Float: If training error <= this value, training stops
tolerance.valid	Float: If validation error <= this value, training stops

max.iter	Integer: Maximum number of iterations (additive steps) to perform. Default = 10
init	Float: Initial value for prediction. Default = mean(y)
x.name	Character: Name for feature set
y.name	Character: Name for outcome
question	Character: the question you are attempting to answer with this model, in plain language.
base.verbose	Logical: verbose argument passed to learner
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If > 0, print diagnostic info to console
print.progress.every	Integer: Print progress over this many iterations
print.error.plot	String or Integer: "final" plots a training and validation (if available) error curve at the end of training. If integer, plot training and validation error curve every this many iterations during training. "none" for no plot.
prefix	Internal
plot.theme	Character: "zero", "dark", "box", "darkbox"
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted.
print.base.plot	Logical: Passed to print.plot argument of base learner, i.e. if TRUE, print error plot for each base learner
plot.type	Character: "l" or "p". Plot using lines or points.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
...	Additional parameters to be passed to learner define by mod

Details

If learning.rate is set to 0, a nullmod will be created

Author(s)

E.D. Gennatas

`bootstrap` *Bootstrap Resampling*

Description

Bootstrap Resampling

Usage

```
bootstrap(x, n.resamples = 10, seed = NULL)
```

Arguments

`x`	Input vector
`n.resamples`	Integer: Number of resamples to make. Default = 10
`seed`	Integer: If provided, set seed for reproducibility. Default = NULL

Author(s)

E.D. Gennatas

`brier_score` *Brier Score*

Description

Calculate the Brier Score for classification:

Usage

```
brier_score(true, estimated.prob)
```

Arguments

`true`	Numeric vector, 0, 1: True labels
`estimated.prob`	Numeric vector, [0, 1]: Estimated probabilities

Details

$$BS = \frac{1}{N} \sum_{i=1}^{N} (y_i - p_i)^2$$

Author(s)

E.D. Gennatas

calibrate *Calibrate predicted probabilities using GAM*

Description

Calibrate predicted probabilities using a generalized additive model (GAM).

Usage

```
calibrate(
  true.labels,
  predicted.prob,
  pos.class = NULL,
  mod = c("gam", "glm"),
  k = 5,
  verbose = TRUE
)
```

Arguments

true.labels	Factor with true class labels
predicted.prob	Numeric vector with predicted probabilities
pos.class	Integer: Index of the positive class
mod	Character: Model to use for calibration. Either "gam" or "glm"
k	Integer: GAM degrees of freedom
verbose	Logical: If TRUE, print messages to the console

Value

mod: fitted GAM model. Use `mod$fitted.values` to get calibrated input probabilities; use `predict(mod, newdata = newdata, type = "response")` to calibrate other estimated probabilities.

Author(s)

EDG

Examples

```
## Not run:
data(segment_logistic, package = "probably")

# Plot the calibration curve of the original predictions
dplot3_calibration(
  true.labels = segment_logistic$Class,
  predicted.prob = segment_logistic$.pred_poor,
  n_windows = 10,
  pos.class = 2
```

```
)

# Plot the calibration curve of the calibrated predictions
dplot3_calibration(
  true.labels = segment_logistic$Class,
  predicted.prob = calibrate(
    segment_logistic$Class,
    segment_logistic$.pred_poor
  )$fitted.values,
  n_windows = 10,
  pos.class = 2
)

## End(Not run)
```

---

calibrate_cv *Calibrate cross-validated model*

Description

Calibrate cross-validated model trained using train_cv

Usage

```
calibrate_cv(
  mod,
  alg = "gam",
  learn.params = list(),
  resample.params = setup.resample(resampler = "kfold", n.resamples = 5, seed = NULL),
  which.repeat = 1,
  verbosity = 1,
  debug = FALSE
)
```

Arguments

mod	rtModCV object returned by train_cv
alg	Character: "gam" or "glm", algorithm to use for calibration
learn.params	List: List of parameters to pass to the learning algorithm
resample.params	List of parameters to pass to the resampling algorithm. Build using setup.resample
which.repeat	Integer: Which repeat to use for calibration
verbosity	Integer: 0: silent, > 0: print messages
debug	Logical: If TRUE, run without parallel processing, to allow better debugging.

Details

This is a work in progress to be potentially incorporated into train_cv You start by training a cross-validated model using train_cv, then this function can be used to calibrate the model. In order to use all available data, each outer resample from the input `mod` is resampled (using 5-fold CV by default) to train and test calibration models. This allows using the original label-based metrics of `mod` and also extract calibration metrics based on the same data, after aggregating the test set predictions of the calibration models.

Value

List: Calibrated models, test-set labels, test set performance metrics, estimated probabilities (uncalibrated), calibrated probabilities,

Author(s)

E.D. Gennatas

`catrange` *Print range of continuous variable*

Description

Print range of continuous variable

Usage

```
catrange(x, ddSci = TRUE, decimal.places = 1, na.rm = TRUE)
```

Arguments

`x`	Numeric vector
`ddSci`	Logical: If TRUE, use ddSci or range. Default = TRUE
`decimal.places`	Integer: Number of decimal place to use if `ddSci` = `TRUE`. Default = 1
`na.rm`	Logical: passed to `base::range`

Author(s)

E.D. Gennatas

catsize — *Print Size*

Description

Get `NCOL(x)` and `NROW{x}`

Usage

```
catsize(x, name = NULL, verbose = TRUE, newline = TRUE)
```

Arguments

`x`	R object (usually that inherits from matrix or data.frame)
`name`	Character: Name of input object
`verbose`	Logical: If TRUE, print NROW and NCOL to console.
`newline`	Logical: If TRUE, end with new line character.

Value

vector of NROW, NCOL invisibly

Author(s)

E.D. Gennatas

Examples

```
catsize(iris)
```

checkpoint_earlystop — *Early stopping check*

Description

Returns list with relative variance over n.steps, absolute.threshold, last value, and logical "stop", if conditions are met and training should stop. The final stop decision is: `check.thresh | (check.rthresh & check.rvar)` if `combine.relative.thresholds = "AND"` or `check.thresh | (check.rthresh | check.rvar)` if `combine.relative.thresholds = "OR"`

Usage

```
checkpoint_earlystop(
  x,
  absolute.threshold = NA,
  relative.threshold = NA,
  minimize = TRUE,
  relativeVariance.threshold = NA,
  n.steps = 10,
  combine.relative.thresholds = "AND",
  min.steps = 50,
  na.response = c("stop", "continue"),
  verbose = TRUE
)
```

Arguments

`x`	Float, vector: Input - this would normally be the loss at each iteration
`absolute.threshold`	Float: If set and the last value of `x` is less than or equal to this (if `minimize = TRUE`) or greater than or equal to this (if `minimize = FALSE`), then return `stop = TRUE`. See output under Value. Default = NA
`relative.threshold`	Float: If set, checks if the relative change from the first to last value of `x` exceeds this number. i.e. if set to .9 and `minimize = TRUE`, if there is a 90 percent drop from `x[1]` to `x[length(x)]`, then the function returns `stop = TRUE`. If `minimize = FALSE`, then checks if there is a 90 percent increase, accordingly.
`minimize`	Logical: See `absolute.threshold`. Default = TRUE
`relativeVariance.threshold`	Float: If relative variance over last `n.steps` is less than or equal to this, return `stop = TRUE`. See output under Value
`n.steps`	Integer; > 1: Calculate relative variance over this many last values of `x`
`combine.relative.thresholds`	Character: "AND" or "OR": How to combine the criteria `relative.threshold` and `relativeVariance.threshold`. Default = "AND", which means both must be TRUE to stop. The scenario is you might want to check relastiveVariance threshold only after a certain amount of learning has taken place, which you can't predict with `min.steps` but would rather quantify with `relative.threshold`.
`min.steps`	Integer: Do not calculate relativeVariance unless `x` is at least this length
`na.response`	Character: "stop" or "continue": what should happen if the last value of `x` is `NA`
`verbose`	Logical: If TRUE, print messages to console

Value

List with the following items:

- `last.value` Float: Last value of `x`

- `relativeVariance` Float: relative variance of last `n.steps`
- `check.thresh` Logical: TRUE, if absolute threshold was reached
- `check.rvar` Logical: TRUE, if relative variance threshold was reached
- `stop` Logical: TRUE, if either criterion was met - absolute threshold or relativeVariance.threshold

Author(s)

E.D. Gennatas

`check_data` *Check Data*

Description

Check Data

Usage

```
check_data(
  x,
  name = NULL,
  get_duplicates = TRUE,
  get_na_case_pct = FALSE,
  get_na_feature_pct = FALSE
)
```

Arguments

`x`	data.frame, data.table or similar structure
`name`	Character: Name of dataset
`get_duplicates`	Logical: If TRUE, check for duplicate cases
`get_na_case_pct`	Logical: If TRUE, calculate percent of NA values per case
`get_na_feature_pct`	Logical: If TRUE, calculate percent of NA values per feature

Author(s)

E.D. Gennatas

Examples

```
## Not run:
n <- 1000
x <- rnormmat(n, 50, return.df = TRUE)
x$char1 <- sample(letters, n, TRUE)
x$char2 <- sample(letters, n, TRUE)
x$fct <- factor(sample(letters, n, TRUE))
x <- rbind(x, x[1, ])
x$const <- 99L
x[sample(nrow(x), 20), 3] <- NA
x[sample(nrow(x), 20), 10] <- NA
x$fct[30:35] <- NA
check_data(x)

## End(Not run)
```

---

check_files *Check file(s) exist*

Description

Check file(s) exist

Usage

```
check_files(paths, verbose = TRUE, pad = 0)
```

Arguments

paths	Character vector of paths
verbose	Logical: If TRUE, print messages to console
pad	Integer: Number of spaces to pad to the left

Author(s)

E.D. Gennatas

`chill` *Chill*

Description

Relax. Use Ctrl-C to exit (but try to stay relaxed)

Usage

```
chill(sleep = 0.5, text = NULL, max = 1000)
```

Arguments

`sleep`	Float: Time in seconds between drawings. Default = .5
`text`	Character: Text to display
`max`	Integer: Max times to repeat. Default = 1000

`class_error` *Classification Error*

Description

Calculates Classification Metrics

Usage

```
class_error(
  true,
  estimated,
  estimated.prob = NULL,
  calc.auc = TRUE,
  calc.brier = TRUE,
  auc.method = c("pROC", "ROCR", "auc_pairs"),
  trace = 0
)
```

Arguments

`true`	Factor: True labels
`estimated`	Factor: Estimated values
`estimated.prob`	Numeric vector: Estimated probabilities
`calc.auc`	Logical: If TRUE, calculate AUC. May be slow in very large datasets.
`calc.brier`	Logical: If TRUE, calculate Brier Score
`auc.method`	Character: "pROC", "ROCR", "auc_pairs": Method to use, passed to auc.
`trace`	Integer: If > 0, print diagnostic messages. Default = 0

Details

Note that auc.method = "pROC" is the only one that will output an AUC even if one or more estimated probabilities are NA.

Value

S3 object of type "class_error"

Author(s)

E.D. Gennatas

Examples

```
## Not run:
true <- factor(c("a", "a", "a", "b", "b", "b", "b", "b", "b", "b"))
estimated <- factor(c("a", "a", "b", "b", "a", "a", "b", "b", "a", "a"))
estimated.prob <- c(0.7, 0.55, 0.45, 0.25, 0.6, 0.7, 0.2, .37, .57, .61)

class_error(true, estimated, estimated.prob, auc.method = "pROC")
class_error(true, estimated, estimated.prob, auc.method = "ROCR")
class_error(true, estimated, estimated.prob, auc.method = "auc_pairs")

## End(Not run)
```

---

class_imbalance *Class Imbalance*

Description

Calculate class imbalance as given by:

$$I = K \cdot \sum_{i=1}^{K} (n_i/N - 1/K)^2$$

where K is the number of classes, and n_i is the number of instances of class i

Usage

```
class_imbalance(x)
```

Arguments

x	Vector, factor: Labels of outcome. If x has more than 1 column, the last one will be used

Author(s)

E.D. Gennatas

clean_colnames *Clean column names*

Description

Clean column names by replacing all spaces and punctuation with a single underscore

Usage

```
clean_colnames(x)
```

Arguments

x	Character, vector

Author(s)

E.D. Gennatas

Examples

```
clean_colnames(iris)
```

clean_names *Clean names*

Description

Clean character vector by replacing all symbols and sequences of symbols with single underscores, ensuring no name begins or ends with a symbol

Usage

```
clean_names(x, prefix_digits = "V_")
```

Arguments

x	Character vector
prefix_digits	Character: prefix to add to names beginning with a digit. Set to NA to skip

Author(s)

E.D. Gennatas

Examples

```
x <- c("Patient ID", "_Date-of-Birth", "SBP (mmHg)")
x
clean_names(x)
```

`clust` *Clustering with* **rtemis**

Description

Convenience function to perform any **rtemis** clustering

Usage

```
clust(x, clust = "kmeans", x.test = NULL, verbose = TRUE, ...)
```

Arguments

`x`	Numeric matrix / data frame: Input data
`clust`	Character: Decomposition algorithm name, e.g. "nmf" (case-insensitive)
`x.test`	Numeric matrix / Data frame: Testing set data if supported by `clust`
`verbose`	Logical: if TRUE, print messages to screen
`...`	Additional arguments to be passed to clusterer `clust`

Value

`rtClust` object

Author(s)

E.D. Gennatas

`coef.lihad` *Extract coefficients from Hybrid Additive Tree leaves*

Description

Extract coefficients from Hybrid Additive Tree leaves

Usage

```
## S3 method for class 'lihad'
coef(object, newdata, verbose = FALSE, trace = 0, ...)
```

Arguments

`object`	`lihad` object
`newdata`	matrix/data.frame of features
`verbose`	Logical: If TRUE, print output to console
`trace`	Integer 0:2 Increase verbosity
`...`	Not used

Author(s)

E.D. Gennatas

`col2grayscale` *Color to Grayscale*

Description

Convert a color to grayscale

Usage

```
col2grayscale(x, what = c("color", "decimal"))
```

Arguments

`x`	Color to convert to grayscale
`what`	Character: "color" returns a hexadecimal color, "decimal" returns a decimal between 0 and 1

Details

Uses the NTSC grayscale conversion: 0.299 * R + 0.587 * G + 0.114 * B

Examples

```
col2grayscale("red")
col2grayscale("red", "dec")
```

`col2hex` *Convert R color to hexadecimal code*

Description

Convert a color that R understands into the corresponding hexadecimal code

Usage

```
col2hex(color)
```

Arguments

`color`	Color(s) that R understands

Author(s)

E.D. Gennatas

Examples

```
col2hex(c("gray50", "skyblue"))
```

---

colMax *Collapse data.frame to vector by getting column max*

Description

Collapse data.frame to vector by getting column max

Usage

```
colMax(x, na.rm = TRUE)
```

Arguments

`x`	Matrix or Data frame input
`na.rm`	Logical: passed to `max`, If TRUE, ignore NA values, otherwise if NA is present in any column, NA will be returned.

Author(s)

E.D. Gennatas

---

colorAdjust *Adjust HSV Color*

Description

Modify alpha, hue, saturation and value (HSV) of a color

Usage

```
colorAdjust(color, alpha = NULL, hue = 0, sat = 0, val = 0)
```

Arguments

`color`	Input color. Any format that grDevices::col2rgb() recognizes
`alpha`	Numeric: Scale alpha by this amount. Future: replace with absolute setting
`hue`	Float: How much hue to add to `color`
`sat`	Float: How much saturation to add to `color`
`val`	Float: How much to increase value of `color` by

Value

Adjusted color

Author(s)

E.D. Gennatas

---

colorGrad *Color Gradient*

Description

Create a gradient of colors and optionally a colorbar

Usage

```
colorGrad(
  n = 21,
  colors = NULL,
  space = c("rgb", "Lab"),
  lo = "#18A3AC",
  lomid = NULL,
  mid = NULL,
  midhi = NULL,
  hi = "#F48024",
  preview = FALSE,
  colorbar = FALSE,
  cb.n = 21,
  cb.mar = c(1, 1, 1, 1),
  cb.add = FALSE,
  cb.add.mar = c(5, 0, 2, 5),
  cb.axis.pos = 1.1,
  cb.axis.las = 1,
  cb.axis.hadj = 0,
  cb.cex = 6,
  bar.min = -1,
  bar.mid = 0,
  bar.max = 1,
  cex = 1.2,
  filename = NULL,
  pdf.width = 3,
  pdf.height = 7,
  theme = getOption("rt.theme", "light"),
  bg = NULL,
  col.text = NULL,
  plotlycb = FALSE,
  plotly.width = 80,
```

```
  plotly.height = 500,
  rtrn.plotly = FALSE,
  margins = c(0, 0, 0, 0),
  pad = 0,
  par.reset = TRUE
)
```

Arguments

n	Integer: How many distinct colors you want. If not odd, converted to `n + 1` Defaults to 21
colors	Character: Acts as a shortcut to defining `lo`, `mid`, etc for a number of defaults: "french", "penn", "grnblkred",
space	Character: Which colorspace to use. Option: "rgb", or "Lab". Default = "rgb". Recommendation: If `mid` is "white" or "black" (default), use "rgb", otherwise "Lab"
lo	Color for low end
lomid	Color for low-mid
mid	Color for middle of the range or "mean", which will result in `colorOp(c(lo, hi), "mean")`. If `mid = NA`, then only `lo` and `hi` are used to create the color gradient.
midhi	Color for middle-high
hi	Color for high end
preview	Logical: Plot the colors horizontally
colorbar	Logical: Create a vertical colorbar
cb.n	Integer: How many steps you would like in the colorbar
cb.mar	Vector, length 4: Colorbar margins. Default: c(1, 1, 1, 1)
cb.add	Logical: If TRUE, colorbar will be added to existing plot
cb.add.mar	Vector: Margins for colorbar (See `par("mar")`)
cb.axis.pos	Float: Position of axis (See `axis("pos")`)
cb.axis.las	Integer 0,1,2,3: Style of axis labels. 0: Always parallel to the axis, 1: Horizontal, 2: Perpendicular, 3: Vertical. Default = 1
cb.axis.hadj	Float: Adjustment parallel to the reading direction (See `par("adj")`)
cb.cex	FLoat: Character expansion factor for colorbar (See `par("cex")`)
bar.min	Numeric: Lowest value in colorbar
bar.mid	Numeric: Middle value in colorbar
bar.max	Numeric: Max value in colorbar
cex	Float: Character expansion for axis
filename	String (Optional: Path to file to save colorbar
pdf.width	Float: Width for PDF output. Default = 3
pdf.height	Float: Height for PDF output. Default = 7

theme	Character: "light", "dark"
bg	Color: Background color
col.text	Color: Colorbar text color
plotlycb	Logical: Create colorbar using plotly (instead of base R graphics)
plotly.width	Float: Width for plotly colorbar. Default = 80
plotly.height	Float: Height for plotly colorbar. Default = 500
rtrn.plotly	Logical: If TRUE, return plotly object
margins	Vector: Plotly margins. Default = c(0, 0, 0, 0)
pad	Float: Padding for plotly. Default = 0
par.reset	Logical: If TRUE (Default), reset par settings after running

Details

It is best to provide an odd number, so that there is always an equal number of colors on either side of the midpoint. For example, if you want a gradient from -1 to 1 or equivalent, an n = 11, will give 5 colors on either side of 0, each representing a 20\

colors can be defined as a sequence of 3-letter color abbreviations of 2, 3, 4, or 5 colors which will correspond to values: {"lo","hi"}; {"lo", "mid", "hi"}; {"lo", "mid", "midhi", "hi"}, and {"lo", "lo-mid", "mid", "midhi", "hi"}, respectively. For example, try colorGrad(21, "blugrnblkredyel", colorbar = TRUE) 3-letter color abbreviations: wht: white; blk: black; red; grn: green; blu: blue; yel: yellow; rng: orange; prl: purple

Value

Invisible vector of hexadecimal colors / plotly object if rtrn.plotly = TRUE

Author(s)

E.D. Gennatas

---

colorGrad.x *Color gradient for continuous variable*

Description

Color gradient for continuous variable

Usage

```
colorGrad.x(x, color = c("gray20", "#18A3AC"), space = "Lab")
```

Arguments

x	Float, vector
color	Color, vector, length 2
space	Character: "rgb" or "Lab", Default = "Lab"

Author(s)

E.D. Gennatas

---

colorgradient.x *Color gradient for continuous variable*

Description

Color gradient for continuous variable

Usage

```
colorgradient.x(
  x,
  symmetric = FALSE,
  lo.col = "#0290EE",
  mid.col = "#1A1A1A",
  hi.col = "#FFBD4F",
  space = "Lab"
)
```

Arguments

x	Float, vector
symmetric	Logical: If TRUE, make symmetric gradient between -max(abs(x)) and max(abs(x))
lo.col	Low color
mid.col	Middle color
hi.col	High color
space	Character: "rgb" or "Lab". Default = "Lab"

Author(s)

E.D. Gennatas

Examples

```
## Not run:
x <- seq(-10, 10, length.out = 51)
previewcolor(colorgradient.x(x))
x <- sort(rnorm(40))
previewcolor(colorgradient.x(x, mid.col = "white"))
# Notice how most values are near zero therefore almost white

## End(Not run)
```

colorMix — *Create an alternating sequence of graded colors*

Description

Create an alternating sequence of graded colors

Usage

```
colorMix(color, n = 4)
```

Arguments

`color`	List: List of two or more elements, each containing two colors. A gradient will be created from the first to the second color of each element
`n`	Integer: Number of steps in each gradient.

Author(s)

E.D. Gennatas

Examples

```
color <- list(blue = c("#82afd3", "#000f3a"),
              gray = c("gray10", "gray85"))
previewcolor(desaturate(colorMix(color, 6), .3))

color <- list(blue = c("#82afd3", "#57000a"),
              gray = c("gray10", "gray85"))
previewcolor(desaturate(colorMix(color, 6), .3))

color <- list(blue = c("#82afd3", "#000f3a"),
              purple = c("#23001f", "#c480c1"))
previewcolor(desaturate(colorMix(color, 5), .3))
```

colorOp — *Simple Color Operations*

Description

Invert a color or calculate the mean of two colors in HSV or RGB space. This may be useful in creating colors for plots

Usage

```
colorOp(col, fn = c("invert", "mean"), space = c("HSV", "RGB"))
```

Arguments

`col`	Input color(s)
`fn`	Character: "invert", "mean": Function to perform
`space`	Character: "HSV", "RGB": Colorspace to operate in - for averaging only

Details

The average of two colors in RGB space will often pass through gray, which is likely undesirable. Averaging in HSV space, better for most applications.

Value

Color

Author(s)

E.D. Gennatas

`color_fade` *Fade color towards target*

Description

Fade color towards target

Usage

```
color_fade(x, to = "#000000", pct = 0.5)
```

Arguments

`x`	Color source
`to`	Target color
`pct`	Numeric (0, 1) fraction of the distance in RGBA space between `x` and `to` to move. e.g. .5 gets the mean RGBA value of the two

Value

Color in hex notation

Author(s)

E.D. Gennatas

color_invertRGB — *Invert Color in RGB space*

Description

Invert Color in RGB space

Usage

```
color_invertRGB(x)
```

Arguments

x	Color, vector

Value

Inverted colors using hexadecimal notation #RRGGBBAA

Author(s)

E.D. Gennatas

Examples

```
## Not run:
cols <- c("red", "green", "blue")
previewcolor(cols)
cols |>
  color_invertRGB() |>
  previewcolor()

## End(Not run)
```

color_mean — *Average colors*

Description

Average colors

Usage

```
color_mean(x, space = c("RGB", "HSV"))
```

Arguments

x	Color vector
space	Character: RGB or HSV; space to average in

Author(s)

E.D. Gennatas

Examples

```
## Not run:
color_mean(c("red", "blue")) |> previewcolor()
color_mean(c("red", "blue"), "HSV") |> previewcolor()

## End(Not run)
```

`color_order` *Order colors*

Description

Order colors by RGB distance

Usage

```
color_order(x, start_with = 1, order_by = c("similarity", "dissimilarity"))
```

Arguments

x	Vector of colors
start_with	Integer: Which color to output in first position
order_by	Character: "similarity" or "dissimilarity"

Author(s)

E.D. Gennatas

color_separate *Separate colors*

Description

Separate colors by RGB distance

Usage

```
color_separate(x, start_with = 1)
```

Arguments

x	Vector of colors
start_with	Integer: Which color to output in first position

Details

Starting with the first color defined by `start_with`, the next color is chosen to be max distance from all preceding colors

Author(s)

E.D. Gennatas

color_sqdist *Squared Color Distance*

Description

Get the squared RGB distance between two colors

Usage

```
color_sqdist(x, y)
```

Arguments

x	Color
y	Color

Author(s)

E.D. Gennatas

Examples

```
color_sqdist("red", "green")
color_sqdist("#16A0AC", "#FA6E1E")
```

`cols2list` *Convert data frame columns to list elements*

Description

Convenience function to create a list out of data frame columns

Usage

```
cols2list(x)
```

Arguments

x	Input: Will be coerced to data.frame, then each column will become an element of a list

Author(s)

E.D. Gennatas

`create_config` *Create rtemis configuration file*

Description

Defines a complete predictive modeling pipeline and saves it as a JSON file.

Usage

```
create_config(
  data_path,
  target = NULL,
  binclass_posid = 1,
  alg = "lightgbm",
  train.params = NULL,
  inner.resampling = setup.resample(resampler = "cv", n.resamples = 5),
  outer.resampling = setup.resample(resampler = "cv", n.resamples = 10),
  config.path = "rtemis-config.json",
  model.outdir = NULL,
  allow.overwrite = FALSE,
  verbose = TRUE
)
```

Arguments

`data_path`	Character: Path to data file. Can be any file recognized by read, commonly CSV, Excel, or RDS.
`target`	Character: Name of the target variable in the data. If not specified, the last column of `data` will be used.
`alg`	Character: Algorithm to use. Any of `select_learn()`.
`train.params`	List: Parameters for the training algorithm.
`inner.resampling`	List: Resampling method for the inner loop, i.e. hyperparameter tuning, a.k.a. model selection. Set using setup.resample
`outer.resampling`	List: Resampling method for the outer loop, i.e. testing. Set using setup.resample
`config.path`	Character: Path to save configuration file.
`model.outdir`	Character: Directory to save trained model and associated files. If NULL, the directory of `config.path` will be used.

Value

config as list, invisibly.

Author(s)

EDG

`crules` *Combine rules*

Description

Combine rules

Usage

```
crules(...)
```

Arguments

`...`	Character: Rules

Author(s)

E.D. Gennatas

c_CMeans *Fuzzy C-means Clustering*

Description

Perform fuzzy C-means clustering using e1071::cmeans

Usage

```
c_CMeans(
  x,
  k = 2,
  iter.max = 100,
  dist = "euclidean",
  method = "cmeans",
  m = 2,
  rate.par = NULL,
  weights = 1,
  control = list(),
  verbose = TRUE,
  ...
)
```

Arguments

x	Input data
k	Integer: Number of clusters to get. Default = 2
iter.max	Integer: Maximum number of iterations. Default = 100
dist	Character: Distance measure to use: 'euclidean' or 'manhattan'. Default = "euclidean"
method	Character: "cmeans" - fuzzy c-means clustering; "ufcl": on-line update. Default = "cmeans"
m	Float (>1): Degree of fuzzification. Default = 2
rate.par	Float (0, 1): Learning rate for the online variant. (Default = .3)
weights	Float (>0): Case weights
control	List of control parameters. See e1071::cmeans
verbose	Logical: If TRUE, print messages to console
...	Additional parameters to be passed to e1071::cmeans

Value

rtClust object

Author(s)

E.D. Gennatas

See Also

Other Clustering: c_DBSCAN(), c_EMC(), c_H2OKMeans(), c_HARDCL(), c_HOPACH(), c_KMeans(), c_MeanShift(), c_NGAS(), c_PAM(), c_PAMK(), c_SPEC()

---

c_DBSCAN *Density-based spatial clustering of applications with noise*

Description

Perform DBSCAN clustering

Usage

```
c_DBSCAN(
  x,
  x.test = NULL,
  eps = 1,
  minPts = NCOL(x) + 1,
  weights = NULL,
  borderPoints = TRUE,
  search = c("kdtree", "linear", "dist"),
  verbose = TRUE,
  ...
)
```

Arguments

x	Input matrix / data.frame
x.test	Testing set matrix / data.frame
eps	Numeric: Radius of the epsilon neighborhood
minPts	Integer: Number of minimum points required in the eps neighborhood for core points (including the point itself).
weights	Numeric vector: Data points' weights. Needed for weighted clustering.
borderPoints	Logical: If TRUE, assign border points to clusters, otherwise they are considered noise
search	Character: "kdtree", "linear" or "dist": nearest neighbor search strategy
verbose	Logical: If TRUE, print messages to screen
...	Additional parameters to be passed to `flexclust::cclust`

Details

See dbscan::dbscan for info on how to choose eps and minPts

Author(s)

Efstathios D. Gennatas

See Also

Other Clustering: c_CMeans(), c_EMC(), c_H2OKMeans(), c_HARDCL(), c_HOPACH(), c_KMeans(), c_MeanShift(), c_NGAS(), c_PAM(), c_PAMK(), c_SPEC()

---

c_EMC *Expectation Maximization Clustering*

Description

Perform clustering by EM using EMCluster::emcluster

Usage

```
c_EMC(
  x,
  x.test = NULL,
  k = 2,
  lab = NULL,
  EMC = EMCluster::.EMC,
  verbose = TRUE,
  ...
)
```

Arguments

x	Input matrix / data.frame
x.test	Testing set matrix / data.frame
k	Integer: Number of clusters to get
lab	Vector, length NROW(x): Labels for semi-supervised clustering
EMC	List of control parameters for EMCluster::emcluster. Default = EMCluster::.EMC
verbose	Logical: If TRUE, print messages to screen
...	Additional parameters to be passed to EMCluster::emcluster

Details

First, EMCluster::simple.init(x, nclass = k) is run, followed by EMCluster::emcluster(x, emobj = emobj, assign.class = TRUE, ...)

This can be very slow.

Author(s)

E.D. Gennatas

See Also

Other Clustering: c_CMeans(), c_DBSCAN(), c_H2OKMeans(), c_HARDCL(), c_HOPACH(), c_KMeans(), c_MeanShift(), c_NGAS(), c_PAM(), c_PAMK(), c_SPEC()

---

c_H2OKMeans *K-Means Clustering with H2O*

Description

Perfomr K-Means clustering using h2o::h2o.kmeans

Usage

```
c_H2OKMeans(
  x,
  x.test = NULL,
  k = 2,
  estimate.k = FALSE,
  nfolds = 0,
  max.iterations = 10,
  ip = "localhost",
  port = 54321,
  n.cores = rtCores,
  seed = -1,
  init = c("Furthest", "Random", "PlusPlus", "User"),
 categorical.encoding = c("AUTO", "Enum", "OneHotInternal", "OneHotExplicit", "Binary",
    "Eigen", "LabelEncoder", "SortByResponse", "EnumLimited"),
  verbose = TRUE,
  ...
)
```

Arguments

x	Input matrix / data.frame
x.test	Testing set matrix / data.frame
k	Integer: Number of clusters to get
estimate.k	Logical: if TRUE, estimate k up to a maximum set by the k argument
nfolds	Integer: Number of cross-validation folds
max.iterations	Integer: Maximum number of iterations
ip	Character: IP address of H2O server. Default = "localhost"
port	Integer: Port number of H2O server. Default = 54321

n.cores	Integer: Number of cores to use
seed	Integer: Seed for H2O's random number generator. Default = -1 (time-based ranodm number)
init	Character: Initialization mode: "Furthest", "Random", "PlusPlus", "User". Default = "Furthest"
categorical.encoding	Character: How to encode categorical variables: "AUTO", "Enum", "OneHotInternal", "OneHotExplicit", "Binary", "Eigen", "LabelEncoder", "SortByResponse", "EnumLimited". Default = "AUTO"
verbose	Logical: If TRUE, print messages to screen
...	Additional arguments to pass to `h2p::h2o.kmeans`

Details

Check out the H2O Flow at `[ip]:[port]`, Default IP:port is "localhost:54321" e.g. if running on localhost, point your web browser to `localhost:54321` For additional information, see help on `h2o::h2o.kmeans`

Value

`rtMod` object

Author(s)

E.D. Gennatas

See Also

Other Clustering: `c_CMeans()`, `c_DBSCAN()`, `c_EMC()`, `c_HARDCL()`, `c_HOPACH()`, `c_KMeans()`, `c_MeanShift()`, `c_NGAS()`, `c_PAM()`, `c_PAMK()`, `c_SPEC()`

---

c_HARDCL *Clustering by Hard Competitive Learning*

Description

Perform clustering by Hard Competitive Learning using `flexclust::cclust`

Usage

```
c_HARDCL(x, x.test = NULL, k = 2, dist = "euclidean", verbose = TRUE, ...)
```

Arguments

x	Input matrix / data.frame
x.test	Optional test set data
k	Integer: Number of clusters to get
dist	Character: Distance measure to use: 'euclidean' or 'manhattan'
verbose	Logical: If TRUE, print messages to console
...	Additional parameters to be passed to `flexclust::cclust`

Author(s)

E.D. Gennatas

See Also

Other Clustering: `c_CMeans()`, `c_DBSCAN()`, `c_EMC()`, `c_H2OKMeans()`, `c_HOPACH()`, `c_KMeans()`, `c_MeanShift()`, `c_NGAS()`, `c_PAM()`, `c_PAMK()`, `c_SPEC()`

---

c_HOPACH — *Hierarchical Ordered Partitioning and Collapsing Hybrid*

Description

Perform HOPACH clustering using hopach::hopach

Usage

```
c_HOPACH(
  x,
  dmat = NULL,
  metric = c("cosangle", "abscosangle", "euclid", "abseuclid", "cor", "abscor"),
  k = 15,
  kmax = 9,
  khigh = 9,
  trace = 0,
  verbose = TRUE,
  ...
)
```

Arguments

x	Input matrix / data.frame
dmat	Matrix (numeric, no missing values) or `hdist` object of pairwise distances. If NULL, it is computed based on `metric`
metric	Character: Dissimilarity metric to be used. Options: "cosangle", "abscosangle", "euclid", "abseuclid", "cor", "abscor"

k	Integer, (0:15]: Maximum number of levels
kmax	Integer, [1:9]: Maximum number of children at each node in the tree
khigh	Integer, [1:9]: Maximum number of children at each nod ein the tree when computing the the Mean/Median Split Silhouette. Usually same as kmax
trace	Integer: If trace > 0, print messages during HOPACH run. Default = 0
verbose	Logical: If TRUE, print messages to console
...	Additional parameters to pass to cluster::hopach

Author(s)

E.D. Gennatas

See Also

Other Clustering: c_CMeans(), c_DBSCAN(), c_EMC(), c_H2OKMeans(), c_HARDCL(), c_KMeans(), c_MeanShift(), c_NGAS(), c_PAM(), c_PAMK(), c_SPEC()

---

c_KMeans *K-means Clustering*

Description

Perform K-means clustering using flexclust::cclust

Usage

```
c_KMeans(x, x.test = NULL, k = 2, dist = "euclidean", verbose = TRUE, ...)
```

Arguments

x	Input matrix / data.frame
x.test	Testing set matrix / data.frame
k	Integer: Number of clusters to get
dist	Character: Distance measure to use: 'euclidean' or 'manhattan'
verbose	Logical: If TRUE, print messages to screen
...	Additional parameters to pass to flexclust::cclust

Author(s)

E.D. Gennatas

See Also

Other Clustering: c_CMeans(), c_DBSCAN(), c_EMC(), c_H2OKMeans(), c_HARDCL(), c_HOPACH(), c_MeanShift(), c_NGAS(), c_PAM(), c_PAMK(), c_SPEC()

c_MeanShift *Mean Shift Clustering*

Description

Perform Mean Shift clustering using meanShiftR::meanShift

Usage

```
c_MeanShift(
  x,
  nNeighbors = NROW(x),
  algorithm = c("LINEAR", "KDTREE"),
  kernelType = c("NORMAL", "EPANECHNIKOV", "BIWEIGHT"),
  bandwidth = rep(1, NCOL(x)),
  alpha = 0,
  iterations = 10,
  epsilon = 1e-08,
  epsilonCluster = 1e-04,
  parameters = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

x	Input matrix
nNeighbors	Integer: Number of neighbors to consider for kernel density estimate
algorithm	Character: "LINEAR" or "KDTREE"
kernelType	Character: "NORMAL", "EPANECHNIKOV", "BIWEIGHT"
bandwidth	Numeric vector, length = ncol(x): Use in kernel density estimation for steepest ascent classification.
alpha	Numeric: A scalar tuning parameter for normal kernels. When this parameter is set to zero, the mean shift algorithm will operate as usual. When this parameter is set to one, the mean shift algorithm will be approximated through Newton's Method. When set to a value between zero and one, a generalization of Newton's Method and mean shift will be used instead providing a means to balance convergence speed with stability.
iterations	Integer: Number of iterations to perform
epsilon	Numeric: used to determine when to terminate the iteration of an individual query point. If the distance between the query point at iteration i and i+1 is less than epsilon, then iteration ceases on this point.
epsilonCluster	Numeric: Used to determine the minimum distance between distinct clusters. This distance is applied after all iterations have finished and in order of the rows of queryData.

`parameters`	A scalar or vector of paramters used by the specific algorithm. There are no optional parameters for the "LINEAR" method, "KDTREE" supports optional parameters for the maximum number of points to store in a leaf node and the maximum value for the quadratic form in the normal kernel, ignoring the constant value -0.5.
`verbose`	Logical: If TRUE, print messages to console
`...`	Additional parameters to be passed to `flexclust::cclust`

Author(s)

E.D. Gennatas

See Also

Other Clustering: `c_CMeans()`, `c_DBSCAN()`, `c_EMC()`, `c_H2OKMeans()`, `c_HARDCL()`, `c_HOPACH()`, `c_KMeans()`, `c_NGAS()`, `c_PAM()`, `c_PAMK()`, `c_SPEC()`

---

c_NGAS — *Neural Gas Clustering*

Description

Perform Neural Gas clustering using `flexclust::cclust`

Usage

```
c_NGAS(x, x.test = NULL, k = 2, dist = "euclidean", verbose = TRUE, ...)
```

Arguments

`x`	Input matrix / data.frame
`x.test`	Testing set matrix / data.frame
`k`	Integer: Number of clusters to get
`dist`	Character: Distance measure to use: 'euclidean' or 'manhattan'
`verbose`	Logical: If TRUE, print messages to screen
`...`	Additional parameters to be passed to `flexclust::cclust`

Value

`rtClust` object

Author(s)

E.D. Gennatas

See Also

Other Clustering: c_CMeans(), c_DBSCAN(), c_EMC(), c_H2OKMeans(), c_HARDCL(), c_HOPACH(), c_KMeans(), c_MeanShift(), c_PAM(), c_PAMK(), c_SPEC()

---

c_PAM *Partitioning Around Medoids*

Description

Perform PAM clustering using `cluster::pam`

Usage

```
c_PAM(
  x,
  k = 2,
  diss = FALSE,
  metric = "euclidean",
  do.swap = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

`x`	Input matrix / data.frame
`k`	Integer: Number of clusters to get
`diss`	Logical: If TRUE, `x` should be a `dist` or dissimilarity matrix. Otherwise, `x` should be a matrix of cases by features. Default = FALSE
`metric`	Character: Dissimilarity metric to be used. Options: 'euclidean', 'manhattan'
`do.swap`	Logical: If TRUE, perform the swap phase (See `cluster::pam`), as in the original PAM algorithm. This is computationally intensive and can be skipped. Default = TRUE
`verbose`	Logical: If TRUE, print messages to screen
`...`	Additional parameters to be passed to `cluster::pam`

Author(s)

E.D. Gennatas

See Also

Other Clustering: c_CMeans(), c_DBSCAN(), c_EMC(), c_H2OKMeans(), c_HARDCL(), c_HOPACH(), c_KMeans(), c_MeanShift(), c_NGAS(), c_PAMK(), c_SPEC()

c_PAMK *Partitioning Around Medoids with k Estimation*

Description

Estimate PAM clustering solution and optimal k using `fpc::pamk`

Usage

```
c_PAMK(
  x,
  krange = 2:10,
  criterion = "asw",
  usepam = ifelse(nrow(x) < 2000, TRUE, FALSE),
  scaling = TRUE,
  diss = inherits(data, "dist"),
  metric = "euclidean",
  do.swap = TRUE,
  trace = 0,
  verbose = TRUE,
  ...
)
```

Arguments

`x`	Input matrix / data.frame
`krange`	Integer vector: Range of k values to try
`criterion`	Character: Criterion to use for selecting k: "asw", "multiasw" or "ch". See `fpc::pamk`
`usepam`	Logical: If TRUE, use `cluster::pam`, otherwise use `cluster::clara`.
`scaling`	Logical or Numeric vector: If TRUE, scale input. If numeric vector of length equal to number of features, the features are divided by the corresponding value.
`diss`	Logical: If TRUE, treat `x` as a dissimilarity matrix, otherwise as a matrix of cases by features. Default = TRUE, if x inherits from class `dist`, FALSE otherwise.
`metric`	Character: Dissimilarity metric to be used. Options: 'euclidean', 'manhattan'
`do.swap`	Logical: If TRUE, perform the swap phase. See `fpc::pam` for more info
`trace`	Integer [0, 3]: Trace level for `fpc::pamk`
`verbose`	Logical: If TRUE, print messages to console
`...`	Additional parameters to be passed to `fpc::pamk` and/or `cluster::pam`

Value

`rtClust` object

Author(s)

E.D. Gennatas

See Also

Other Clustering: c_CMeans(), c_DBSCAN(), c_EMC(), c_H2OKMeans(), c_HARDCL(), c_HOPACH(), c_KMeans(), c_MeanShift(), c_NGAS(), c_PAM(), c_SPEC()

---

c_SPEC *Spectral Clustering*

Description

Perform Spectral Clustering using `kernlab::specc`

Usage

```
c_SPEC(
  x,
  k = 2,
  kernel = "rbfdot",
  kpar = "automatic",
  nystrom.red = FALSE,
  nystrom.sample = dim(x)[1]/6,
  iterations = 200,
  mod.sample = 0.75,
  na.action = na.omit,
  verbose = TRUE,
  ...
)
```

Arguments

`x`	Input matrix / data.frame
`k`	Integer: Number of clusters to get
`kernel`	Character: Kernel to use: "rbfdot", "polydot", "vanilladot", tanhdot", "laplacedot", "besseldot", "anovadot", "splinedot", "stringdot"
`kpar`	String OR List: "automatic", "local" OR list with: sigma (for "rbfdor", "laplacedot"); degree, scale, offset (for "polydot"); scale, offset (for "tanhdot"); sigma, order, degree (for "besseldot"); sigma, degree (for "anovadot"); length, lambda, normalized (for "stringdot")
`nystrom.red`	Logical: if TRUE, use nystrom method to calculate eigenvectors (Default = FALSE)
`nystrom.sample`	Integer: Number of points to use for estimating the eigenvalues when `nystrom.red` = TRUE Default = `dim(x)[1]/6`

`iterations`	Integer: Number of iterations allowed
`mod.sample`	Float (0, 1): Proportion of data to use when estimating sigma. Default = .75
`na.action`	Function: Action to perform on NA (Default = `na.omit`)
`verbose`	Logical: If TRUE, print messages to screen
`...`	Additional parameters to be passed to `flexclust::cclust`

Author(s)

E.D. Gennatas

See Also

Other Clustering: `c_CMeans()`, `c_DBSCAN()`, `c_EMC()`, `c_H2OKMeans()`, `c_HARDCL()`, `c_HOPACH()`, `c_KMeans()`, `c_MeanShift()`, `c_NGAS()`, `c_PAM()`, `c_PAMK()`

`dat2bsplinemat` *B-Spline matrix from dataset*

Description

Convert a dataset to its b-spline basis set

Usage

```
dat2bsplinemat(
  x,
  df = NULL,
  knots = NULL,
  degree = 3L,
  intercept = FALSE,
  Boundary.knots = range(x, na.rm = TRUE),
  return.deriv = FALSE,
  as.data.frame = TRUE
)
```

Arguments

`x`	data.frame: Input
`df`	Integer: Degrees of freedom. See `splines::bSpline`
`knots`	Float, vector: Internal breakpoints. See `splines::bSpline`
`degree`	Integer (>0): Degree of the piecewise polynomial. See `splines::bSpline`
`intercept`	Logical: If TRUE, an intercept is included. Default = FALSE
`Boundary.knots`	Float, vector (length = 2): Boundary points to anchor the spline basis.
`return.deriv`	Logical: If TRUE, return list containing a data frame with the splines and another data frame with their derivatives
`as.data.frame`	Logical: If TRUE, return data.frame, otherwise matrix. Default = TRUE See `splines::bSpline`

Value

If `return.deriv=F`, a data frame where each original feature is replaced with its basis set or a list, otherwise a list containing a data frame with splines and a data frame with their derivatives

Author(s)

E.D. Gennatas

`dat2poly` *Create n-degree polynomial from data frame*

Description

This is a convenience function that will take each column of the input and calculate 1:degree powers and concatenate into a data.frame of dimensions `n * (degree * p)` given an `n * p` input

Usage

```
dat2poly(
  dat,
  method = c("simple", "poly"),
  degree = 2,
  raw = FALSE,
  as.data.frame = TRUE
)
```

Arguments

`dat`	Numeric, matrix / data.frame: Input
`method`	Character: "simple", "poly". "simple": raise each column of `dat` up to `degree`. "poly": use `stats::poly` with arguments `degree` and `raw`
`degree`	Integer: degree of polynomials to create. Default = 2
`raw`	Logical: If TRUE, create simple polynomial, not orthogonalized. Default = FALSE
`as.data.frame`	Logical: If TRUE, return data.frame. Default = TRUE

Author(s)

E.D. Gennatas

date2factor *Date to factor time bin*

Description

Convert Date to time bin factor.

Usage

```
date2factor(
  x,
  time_bin = c("year", "quarter", "month", "day"),
  make_bins = c("range", "present"),
  bin_range = range(x, na.rm = TRUE),
  ordered = FALSE
)
```

Arguments

x	Date vector
time_bin	Character: "year", "quarter", "month", or "day"
make_bins	Character: "range" or "preseent". If "range" the factor levels will include all time periods define by time_bin within bin_range. This means factor levels can be empty. Otherwise, if "present", factor levels only include time periods present in data.
bin_range	Date, vector, length 2: Range of dates to make levels for. Defaults to range of input dates x
ordered	Logical: If TRUE, factor output is ordered. Default = FALSE

Details

Order of levels will be chronological (important e.g. for plotting) Additionally, can output ordered factor with ordered = TRUE

Value

factor of time periods

Author(s)

E.D. Gennatas

Examples

```
## Not run:
library(data.table)
startDate <- as.Date("2018-01-01")
endDate <- as.Date("2020-12-31")
time <- sample(seq(startDate, endDate, length.out = 100))
date2factor(time)
date2factor(time, "quarter")
date2factor(time, "month")
date2factor(time, "day")
# range vs present
x <- sample(seq(as.Date("2018-01-01"), as.Date("2021-01-01"), by = 1), 10)
date2factor(x, time_bin = "quarter", make_bins = "present")
date2factor(x, time_bin = "quarter", make_bins = "range")

## End(Not run)
```

---

date2ym *Date to year-month factor*

Description

Date to year-month factor

Usage

```
date2ym(x, ordered = FALSE)
```

Arguments

x	Date vector
ordered	Logical: If TRUE, return ordered factor. Default = FALSE

Author(s)

E.D. Gennatas

---

date2yq *Date to year-quarter factor*

Description

Date to year-quarter factor

Usage

```
date2yq(x, ordered = FALSE)
```

Arguments

`x`	Date vector
`ordered`	Logical: If TRUE, return ordered factor. Default = FALSE

Author(s)

E.D. Gennatas

`ddb_collect` *Collect a lazy-read duckdb table*

Description

Collect a table read with `ddb_data(x, collect = FALSE)`

Usage

```
ddb_collect(sql, progress = TRUE, returnobj = c("data.frame", "data.table"))
```

Arguments

`sql`	Character: DuckDB SQL query, usually output of ddb_data with `collect = FALSE`
`progress`	Logical: If TRUE, show progress bar
`returnobj`	Character: data.frame or data.table: class of object to return

Author(s)

E.D. Gennatas

Examples

```
## Not run:
sql <- ddb_data("/Data/iris.csv", collect = FALSE)
ir <- ddb_ollect(sql)

## End(Not run)
```

ddb_data *Read CSV using DuckDB*

Description

Lazy-read a CSV file, optionally filter rows, remove duplicates, clean column names, convert character to factor, and collect.

Usage

```
ddb_data(
  filename,
  datadir = NULL,
  sep = ",",
  header = TRUE,
  quotechar = "",
  ignore_errors = TRUE,
  make_unique = TRUE,
  select_columns = NULL,
  filter_column = NULL,
  filter_vals = NULL,
  character2factor = FALSE,
  collect = TRUE,
  progress = TRUE,
  returnobj = c("data.table", "data.frame"),
  data.table.key = NULL,
  clean_colnames = TRUE,
  verbose = TRUE
)
```

Arguments

filename	Character: file name; either full path or just the file name, if datadir is also provided
datadir	Character: Optional path if filename is not full path
sep	Character: Field delimiter/separator
header	Logical: If TRUE, first line will be read as column names
quotechar	Character: Quote character
ignore_errors	Logical: If TRUE, ignore parsing errors (sometimes it's either this or no data, so)
make_unique	Logical: If TRUE, keep only unique rows
select_columns	Character vector: Column names to select
filter_column	Character: Name of column to filter on, e.g. "ID"
filter_vals	Numeric or Character vector: Values in filter_column to keep.

character2factor	Logical: If TRUE, convert character columns to factors
collect	Logical: If TRUE, collect data and return structure class as defined by returnobj
progress	Logical: If TRUE, print progress (no indication this works)
returnobj	Character: "data.frame" or "data.table" object class to return. If "data.table", data.frame object returned from DBI::dbGetQuery is passed to data.table::setDT; will add to execution time if very large, but then that's when you need a data.table
data.table.key	Character: If set, this correspond to a column name in the dataset. This column will be set as key in the data.table output
clean_colnames	Logical: If TRUE, clean colnames with clean_colnames
verbose	Logical: If TRUE, print messages to console

Author(s)

E.D. Gennatas

Examples

```
## Not run:
ir <- ddb_data("/Data/massive_dataset.csv",
  filter_column = "ID",
  filter_vals = 8001:9999
)

## End(Not run)
```

---

ddSci *Format Numbers for Printing*

Description

2 Decimal places, otherwise scientific notation

Usage

```
ddSci(x, decimal.places = 2, hi = 1e+06, asNumeric = FALSE)
```

Arguments

x	Vector of numbers
decimal.places	Integer: Return this many decimal places. Default = 2
hi	Float: Threshold at or above which scientific notation is used. Default = 1e06
asNumeric	Logical: If TRUE, convert to numeric before returning. Default = FALSE. This will not force all numbers to print 2 decimal places. For example: 1.2035 becomes "1.20" if asNumeric = FALSE, but 1.2 otherwise This can be helpful if you want to be able to use the output as numbers / not just for printing.

Details

Numbers will be formatted to 2 decimal places, unless this results in 0.00 (e.g. if input was .0032), in which case they will be converted to scientific notation with 2 significant figures. `ddSci` will return `0.00` if the input is exactly zero. This function can be used to format numbers in plots, on the console, in logs, etc.

Value

Formatted number

Author(s)

E.D. Gennatas

Examples

```
x <- .34876549
ddSci(x)
# "0.35"
x <- .00000000457823
ddSci(x)
# "4.6e-09"
```

---

decom — *Matrix Decomposition with* **rtemis**

Description

Convenience function to perform any **rtemis** decomposition

Usage

```
decom(x, decom = "PCA", verbose = TRUE, ...)
```

Arguments

`x`	Numeric matrix / data frame: Input data
`decom`	Character: Decomposer name. See]linkselect_decom.
`verbose`	Logical: if TRUE, print messages to console
`...`	Additional arguments to be passed to `decom`

Details

`decom` returns an R6 class object `rtDecom`

Value

`rtDecom` object

Author(s)

E.D. Gennatas

`dependency_check` **rtemis** *internal: Dependencies check*

Description

Checks if dependencies can be loaded; names missing dependencies if not.

Usage

```
dependency_check(..., verbose = FALSE)
```

Arguments

`...`	List or vector of strings defining namespaces to be checked
`verbose`	Logical. If TRUE, print messages to consolde. Note: An error will always printed if dependencies are missing. Setting this to FALSE stops it from printing "Dependencies check passed".

Author(s)

E.D. Gennatas

`desaturate` *Pastelify a color (make a color more pastel)*

Description

Lower a color's saturation by a given percent in the HSV color system

Usage

```
desaturate(color, s = 0.3)
```

Arguments

`color`	Color, vector: Color(s) to operate on
`s`	Float: Decrease saturation by this fraction. Default = .3, which means if saturation of given color is 1, it will become .7

Value

List of adjusted colors

Author(s)

E.D. Gennatas

Examples

```
color <- c("red", "green", "blue")
color.p <- desaturate(color)
```

---

describe — *Describe generic*

Description

Describe generic

Usage

```
describe(object, ...)
```

Arguments

`object`	object to describe
`...`	Additional arguments passed to `describe`

Author(s)

E.D. Gennatas

---

df_movecolumn — *Move data frame column*

Description

Move data frame column

Usage

```
df_movecolumn(x, from, to = ncol(x))
```

Arguments

`x`	data.frame
`from`	String or Integer: Define which column holds the vector you want to move
`to`	Integer: Define which column number you want the vector to be moved to. Default = `ncol(x)` i.e. the last column.

Author(s)

E.D. Gennatas

Examples

```
mtcars_hp <- df_movecolumn(mtcars, "hp")
```

`distillTreeRules` *Distill rules from trained RF and GBM learners*

Description

Extract rules from RF or GBM model, prune, and remove unnecessary rules using `inTrees`

Usage

```
distillTreeRules(
  mod,
  x,
  y = NULL,
  n.trees = NULL,
  maxdepth = 100,
  maxDecay = 0.05,
  typeDecay = 2,
  verbose = TRUE
)
```

Arguments

`mod`	A trained RF or GBM model
`x`	The training set features
`y`	The training set outcomes. If NULL, assumed to be last column of `x`
`n.trees`	Integer: Number of trees to extract
`maxdepth`	Integer: Max depth to consider
`maxDecay`	Float: See `inTree=es::pruneRule`
`typeDecay`	Integer: See `inTreees::pruneRule`
`verbose`	Logical: If TRUE, print messages to output

Details

Models must be trained with s_RF or s_GBM

Author(s)

E.D. Gennatas

dplot3_addtree *Plot AddTree trees*

Description

Plot AddTree trees trained with s_AddTree using `data.tree::plot.Node`

Usage

```
dplot3_addtree(
  addtree,
  col.positive = "#F48024DD",
  col.negative = "#18A3ACDD",
  node.col = "#666666",
  node.shape = "none",
  node.labels = TRUE,
  node.labels.pct.pos = NULL,
  pos.name = NULL,
  edge.col = "#999999",
  layout = "dot",
  rankdir = "TB",
  splines = "polyline",
  fontname = "helvetica",
  bg.color = "#ffffff",
  overlap = "false",
  prune = NULL,
  prune.empty.leaves = TRUE,
  remove.bad.parents = FALSE
)
```

Arguments

`addtree`	Additive Tree object created by s_AddTree
`col.positive`	Color for outcome positive.
`col.negative`	Color for negative outcome.
`node.col`	Color for non-terminal leaves.
`node.shape`	Character: Node shape, passed to `data.tree::SetNodeStyle`
`node.labels`	Logical: If `TRUE`, show node labels.
`node.labels.pct.pos`	Logical: If `TRUE`, show % positive cases in node labels.
`pos.name`	Character: Name for "positive" outcome.
`edge.col`	Color for edges.
`layout`	Character: Passed to `data.tree::SetGraphStyle`
`rankdir`	Character: Passed to `data.tree::SetGraphStyle`

splines	Character: Passed to data.tree::SetGraphStyle
fontname	Character: Passed to data.tree::SetGraphStyle
bg.color	Background color.
overlap	Character: Passed to data.tree::SetGraphStyle
prune	Logical: If TRUE, prune AddTree.
prune.empty.leaves	Logical: If TRUE, prune empty leaves.
remove.bad.parents	Logical: If TRUE, remove nodes with no siblings but children and give their children to their parent.

Details

Edge info and styles have been removed because of problems with DiagrammeR

Author(s)

E.D. Gennatas

---

dplot3_bar *Interactive Barplots*

Description

Draw interactive barplots using plotly

Usage

```
dplot3_bar(
  x,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  col = NULL,
  alpha = 1,
  horizontal = FALSE,
  theme = rtTheme,
  palette = rtPalette,
  barmode = c("group", "relative", "stack", "overlay"),
  group.names = NULL,
  order.by.val = FALSE,
  ylim = NULL,
  hovernames = NULL,
  feature.names = NULL,
  font.size = 16,
  annotate = FALSE,
```

```
    annotate.col = theme$labs.col,
    legend = NULL,
    legend.col = NULL,
    legend.xy = c(1, 1),
    legend.orientation = "v",
    legend.xanchor = "left",
    legend.yanchor = "auto",
    hline = NULL,
    hline.col = NULL,
    hline.width = 1,
    hline.dash = "solid",
    hline.annotate = NULL,
    hline.annotation.x = 1,
    margin = list(b = 65, l = 65, t = 50, r = 10, pad = 0),
    automargin.x = TRUE,
    automargin.y = TRUE,
    padding = 0,
    displayModeBar = TRUE,
    modeBar.file.format = "svg",
    filename = NULL,
    file.width = 500,
    file.height = 500,
    file.scale = 1,
    trace = 0,
    ...
)
```

Arguments

x	vector (possibly named), matrix, or data.frame: If matrix or data.frame, rows are groups (can be 1 row), columns are features
main	Character: Main plot title.
xlab	Character: x-axis label.
ylab	Character: y-axis label.
col	Color, vector: Color for bars. Default NULL, which will draw colors from `palette`
alpha	Float (0, 1]: Transparency for bar colors. Default = .8
horizontal	Logical: If TRUE, plot bars horizontally
theme	List or Character: Either the output of a `theme_*()` function or the name of a theme. Use `themes()` to get available theme names. Theme functions are of the form `theme_<name>`.
palette	Character: Name of **rtemis** palette to use. Default = "rtCol1". Only used if `col = NULL`
barmode	Character: Type of bar plot to make: "group", "relative", "stack", "overlay". Default = "group". Use "relative" for stacked bars, wich handles negative values correctly, unlike "stack", as of writing.

group.names	Character, vector, length = NROW(x): Group names. Default = NULL, which uses rownames(x)
order.by.val	Logical: If TRUE, order bars by increasing value. Only use for single group data. Default = NULL
ylim	Float, vector, length 2: y-axis limits.
hovernames	Character, vector: Optional character vector to show on hover over each bar.
feature.names	Character, vector, length = NCOL(x): Feature names. Default = NULL, which uses colnames(x)
font.size	Float: Font size for all labels. Default = 16
annotate	Logical: If TRUE, annotate stacked bars
annotate.col	Color for annotations
legend	Logical: If TRUE, draw legend. Default = NULL, and will be turned on if there is more than one feature present
legend.col	Color: Legend text color. Default = NULL, determined by theme
legend.xy	Numeric, vector, length 2: x and y for plotly's legend
legend.orientation	"v" or "h" for vertical or horizontal
legend.xanchor	Character: Legend's x anchor: "left", "center", "right", "auto"
legend.yanchor	Character: Legend's y anchor: "top", "middle", "bottom", "auto"
hline	Float: If defined, draw a horizontal line at this y value.
hline.col	Color for hline. Default = "#ff0000" (red)
hline.width	Float: Width for hline. Default = 1
hline.dash	Character: Type of line to draw: "solid", "dot", "dash", "longdash", "dashdot", or "longdashdot"
hline.annotate	Character: Text of horizontal line annotation if hline is set
hline.annotation.x	Numeric: x position to place annotation with paper as reference. 0: to the left of the plot area; 1: to the right of the plot area
margin	Named list: plot margins.
automargin.x	Logical: If TRUE, automatically set x-axis amrgins
automargin.y	Logical: If TRUE, automatically set y-axis amrgins
padding	Integer: N pixels to pad plot.
displayModeBar	Logical: If TRUE, show plotly's modebar
modeBar.file.format	Character: "svg", "png", "jpeg", "pdf" / any output file type supported by plotly and your system
filename	Character: Path to file to save static plot. Default = NULL
file.width	Integer: File width in pixels for when filename is set.
file.height	Integer: File height in pixels for when filename is set.
file.scale	Numeric: If saving to file, scale plot by this number
trace	Integer: The height the number the more diagnostic info is printed to the console
...	Additional arguments passed to theme

Author(s)

E.D. Gennatas

Examples

```
## Not run:
dplot3_bar(VADeaths, legend.xy = c(0, 1))
dplot3_bar(VADeaths, legend.xy = c(1, 1), legend.xanchor = "left")
# simple individual bars
a <- c(4, 7, 2)
dplot3_bar(a)
# if input is a data.frame, each row is a group and each column is a feature
b <- data.frame(x = c(3, 5, 7), y = c(2, 1, 8), z = c(4, 5, 2))
rownames(b) <- c("Jen", "Ben", "Ren")
dplot3_bar(b)
# stacked
dplot3_bar(b, barmode = "stack")

## End(Not run)
```

---

dplot3_box *Interactive Boxplots & Violin plots*

Description

Draw interactive boxplots or violin plots using **plotly**

Usage

```
dplot3_box(
  x,
  time = NULL,
  time.bin = c("year", "quarter", "month", "day"),
  type = c("box", "violin"),
  group = NULL,
  x.transform = c("none", "scale", "minmax"),
  main = NULL,
  xlab = "",
  ylab = NULL,
  col = NULL,
  alpha = 0.6,
  bg = NULL,
  plot.bg = NULL,
  theme = rtTheme,
  palette = rtPalette,
  boxpoints = "outliers",
  quartilemethod = "linear",
  xlim = NULL,
```

```
ylim = NULL,
violin.box = TRUE,
orientation = "v",
annotate_n = FALSE,
annotate_n_y = 1,
annotate_mean = FALSE,
annotate_meansd = FALSE,
annotate_meansd_y = 1,
annotate.col = theme$labs.col,
xnames = NULL,
group.lines = FALSE,
group.lines.dash = "dot",
group.lines.col = NULL,
group.lines.alpha = 0.5,
labelify = TRUE,
order.by.fn = NULL,
font.size = 16,
ylab.standoff = 18,
legend = NULL,
legend.col = NULL,
legend.xy = NULL,
legend.orientation = "v",
legend.xanchor = "auto",
legend.yanchor = "auto",
xaxis.type = "category",
cataxis_tickangle = "auto",
margin = list(b = 65, l = 65, t = 50, r = 12, pad = 0),
automargin.x = TRUE,
automargin.y = TRUE,
boxgroupgap = NULL,
hovertext = NULL,
show_n = FALSE,
pvals = NULL,
htest = "none",
htest.compare = 0,
htest.y = NULL,
htest.annotate = TRUE,
htest.annotate.x = 0,
htest.annotate.y = -0.065,
htest.star.col = theme$labs.col,
htest.bracket.col = theme$labs.col,
starbracket.pad = c(0.04, 0.05, 0.09),
use.plotly.group = FALSE,
width = NULL,
height = NULL,
displayModeBar = TRUE,
modeBar.file.format = "svg",
filename = NULL,
```

```
  file.width = 500,
  file.height = 500,
  file.scale = 1,
  ...
)
```

Arguments

`x`	Vector or List of vectors: Input
`time`	Date or date-time vector
`time.bin`	Character: "year", "quarter", "month", or "day". Period to bin by
`type`	Character: "box" or "violin"
`group`	Factor to group by
`x.transform`	Character: "none", "scale", or "minmax" to use raw values, scaled and centered values or min-max normalized to 0-1, respectively. Transform is applied to each variable before grouping, so that groups are comparable
`main`	Character: Plot title.
`xlab`	Character: x-axis label.
`ylab`	Character: y-axis label.
`col`	Color, vector: Color for boxes. If NULL, which will draw colors from `palette`
`alpha`	Float (0, 1]: Transparency for box colors.
`bg`	Color: Background color. Default = "white"
`plot.bg`	Color: Background color for plot area.
`theme`	Character: Theme to use: Run `themes()` for available themes
`palette`	Character: Name of **rtemis** palette to use. Default = "rtCol1". Only used if `col = NULL`
`boxpoints`	Character or FALSE: "all", "suspectedoutliers", "outliers" See https://plotly.com/r/box-plots/#choosing-the-algorithm-for-computing-quartiles
`quartilemethod`	Character: "linear", "exclusive", "inclusive"
`xlim`	Numeric vector: x-axis limits
`ylim`	Numeric vector: y-axis limits
`violin.box`	Logical: If TRUE and type is "violin" show box within violin plot
`orientation`	Character: "v" or "h" for vertical, horizontal
`annotate_n`	Logical: If TRUE, annotate with N in each box
`annotate_n_y`	Numeric: y position for `annotate_n`
`annotate_mean`	Logical: If TRUE, annotate with mean of each box
`annotate_meansd`	Logical: If TRUE, annotate with mean (SD) of each box
`annotate_meansd_y`	Numeric: y position for `annotate_meansd`
`annotate.col`	Color for annotations

`xnames`	Character, vector, length = NROW(x): x-axis names. Default = NULL, which tries to set names appropriately
`group.lines`	Logical: If TRUE, add separating lines between groups of boxplots
`group.lines.dash`	Character: "solid", "dot", "dash", "longdash", "dashdot", or "longdashdot"
`group.lines.col`	Color for `group.lines`
`group.lines.alpha`	Numeric: transparency for `group.lines.col`
`labelify`	Logical: If TRUE, labelify x names
`order.by.fn`	Function: If defined, order boxes by increasing value of this function (e.g. median).
`font.size`	Float: Font size for all labels.
`ylab.standoff`	Numeric: Standoff for y-axis label
`legend`	Logical: If TRUE, draw legend. Default = TRUE
`legend.col`	Color: Legend text color. Default = NULL, determined by the theme
`legend.xy`	Float, vector, length 2: Relative x, y position for legend.
`legend.orientation`	"v" or "h" for vertical, horizontal
`legend.xanchor`	Character: Legend's x anchor: "left", "center", "right", "auto"
`legend.yanchor`	Character: Legend's y anchor: "top", "middle", "bottom", "auto"
`xaxis.type`	Character: "linear", "log", "date", "category", "multicategory"
`cataxis_tickangle`	Numeric: Angle for categorical axis tick labels
`margin`	Named list: plot margins. Default = `list(b = 65, l = 65, t = 50, r = 10, pad = 0)`
`automargin.x`	Logical: If TRUE, automatically set x-axis amrgins
`automargin.y`	Logical: If TRUE, automatically set y-axis amrgins
`boxgroupgap`	Numeric: Sets the gap (in plot fraction) between boxes of the same location coordinate
`hovertext`	Character vector: Text to show on hover for each data point
`show_n`	Logical: If TRUE, show N in each box
`pvals`	Numeric vector: Precomputed p-values. Should correspond to each box. Bypasses `htest` and `htest.compare`. Requires `group` to be set
`htest`	Character: e.g. "t.test", "wilcox.test" to compare each box to the *first* box. If grouped, compare within each group to the first box. If p-value of test is less than `htest.thresh`, add asterisk above/ to the side of each box
`htest.compare`	Integer: 0: Compare all distributions against the first one; 2: Compare every second box to the one before it. Requires `group` to be set
`htest.y`	Numeric: y coordinate for `htest` annotation
`htest.annotate`	Logical: if TRUE, include htest annotation

`htest.annotate.x`
: Numeric: x-axis paper coordinate for htest annotation

`htest.annotate.y`
: Numeric: y-axis paper coordinate for htest annotation

`htest.star.col` Color for htest annotation stars

`htest.bracket.col`
: Color for htest annotation brackets

`starbracket.pad`
: Numeric: Padding for htest annotation brackets

`use.plotly.group`
: If TRUE, use plotly's `group` arg to group boxes.

`width` Numeric: Force plot size to this width. Default = NULL, i.e. fill available space

`height` Numeric: Force plot size to this height. Default = NULL, i.e. fill available space

`displayModeBar` Logical: If TRUE, show plotly's modebar

`modeBar.file.format`
: Character: "svg", "png", "jpeg", "pdf"

`filename` Character: Path to file to save static plot.

`file.width` Integer: File width in pixels for when `filename` is set.

`file.height` Integer: File height in pixels for when `filename` is set.

`file.scale` Numeric: If saving to file, scale plot by this number

`...` Additional arguments passed to theme

Details

For multiple box plots, the recommendation is:

- `x=dat[, columnindex]` for multiple variables of a data.frame
- `x=list(a=..., b=..., etc.)` for multiple variables of potentially different length
- `x=split(var, group)` for one variable with multiple groups: group names appear below boxplots
- `x=dat[, columnindex], group = factor` for grouping multiple variables: group names appear in legend

If `orientation == "h"`, `xlab` is applied to y-axis and vice versa. Similarly, `x.axist.type` applies to y-axis - this defaults to "category" and would not normally need changing.

Author(s)

E.D. Gennatas

Examples

```
## Not run:
# A.1 Box plot of 4 variables
dplot3_box(iris[, 1:4])
# A.2 Grouped Box plot
dplot3_box(iris[, 1:4], group = iris$Species)
dplot3_box(iris[, 1:4], group = iris$Species, annotate_n = TRUE)
# B. Boxplot binned by time periods
# Synthetic data with an instantenous shift in distributions
set.seed(2021)
dat1 <- data.frame(alpha = rnorm(200, 0), beta = rnorm(200, 2), gamma = rnorm(200, 3))
dat2 <- data.frame(alpha = rnorm(200, 5), beta = rnorm(200, 8), gamma = rnorm(200, -3))
x <- rbind(dat1, dat2)
startDate <- as.Date("2019-12-04")
endDate <- as.Date("2021-03-31")
time <- seq(startDate, endDate, length.out = 400)
dplot3_box(x[, 1], time, "year", ylab = "alpha")
dplot3_box(x, time, "year", legend.xy = c(0, 1))
dplot3_box(x, time, "quarter", legend.xy = c(0, 1))
dplot3_box(x, time, "month",
  legend.orientation = "h",
  legend.xy = c(0, 1),
  legend.yanchor = "bottom"
)
# (Note how the boxplots widen when the period includes data from both dat1 and dat2)

## End(Not run)
```

`dplot3_calibration` *Draw calibration plot*

Description

Draw calibration plot

Usage

```
dplot3_calibration(
  true.labels,
  predicted.prob,
  n.bins = 10,
  bin.method = c("quantile", "equidistant"),
  pos.class = NULL,
  main = NULL,
  subtitle = NULL,
  xlab = "Mean predicted probability",
  ylab = "Empirical risk",
  show.marginal.x = TRUE,
```

```
  marginal.x.y = -0.02,
  marginal.col = NULL,
  marginal.size = 10,
  mode = "markers+lines",
  show.brier = TRUE,
  theme = rtTheme,
  filename = NULL,
  ...
)
```

Arguments

true.labels	Factor or list of factors with true class labels
predicted.prob	Numeric vector or list of numeric vectors with predicted probabilities
n.bins	Integer: Number of windows to split the data into
bin.method	Character: "quantile" or "equidistant": Method to bin the estimated probabilities.
pos.class	Integer: Index of the positive class
main	Character: Main title
subtitle	Character: Subtitle, placed bottom right of plot
xlab	Character: x-axis label
ylab	Character: y-axis label
show.marginal.x	Logical: Add marginal plot of distribution of estimated probabilities
marginal.x.y	Numeric: Y position of marginal markers on x-axis
marginal.col	Color for marginal markers
marginal.size	Numeric: Size of marginal markers
mode	Character: "lines", "markers", "lines+markers": How to plot.
show.brier	Logical: If TRUE, add Brier scores to trace names.
theme	List or Character: Either the output of a `theme_*()` function or the name of a theme. Use `themes()` to get available theme names. Theme functions are of the form `theme_<name>`.
filename	Character: Path to save output.
...	Additional arguments passed to dplot3_xy

Author(s)

EDG

Examples

```
## Not run:
data(segment_logistic, package = "probably")

# Plot the calibration curve of the original predictions
dplot3_calibration(
  true.labels = segment_logistic$Class,
  predicted.prob = segment_logistic$.pred_poor,
  n.bins = 10,
  pos.class = 2
)

# Plot the calibration curve of the calibrated predictions
dplot3_calibration(
  true.labels = segment_logistic$Class,
  predicted.prob = calibrate(
    segment_logistic$Class,
    segment_logistic$.pred_poor
  )$fitted.values,
  n.bins = 10,
  pos.class = 2
)

## End(Not run)
```

---

dplot3_cart *Plot* `rpart` *decision trees*

Description

Plot `rpart` decision trees using `data.tree::plot.Node`

Usage

```
dplot3_cart(
  object,
  col.positive = "#F48024DD",
  col.negative = "#18A3ACDD",
  col.lo = "#80ffff",
  col.mid = "gray20",
  col.hi = "#F4A0FF",
  node.col = "#666666",
  node.shape = "none",
  node.labels = TRUE,
  node.cond = TRUE,
  node.prob = TRUE,
  node.estimate = NULL,
  node.n = TRUE,
```

```
  edge.col = "#999999",
  edge.width = 2,
  edge.labels = FALSE,
  arrowhead = "vee",
  layout = "dot",
  drop.leaves = FALSE,
  rankdir = "TB",
  splines = "polyline",
  fontname = "helvetica",
  bg.color = "white",
  overlap = "false",
  prune = FALSE,
  rpart.cp = NULL,
  verbose = TRUE
)
```

Arguments

`object`	Either `rpart` object or `rtMod` object trained with s_CART
`col.positive`	Color for outcome positive.
`col.negative`	Color for negative outcome.
`col.lo`	Low color for estimated outcome
`col.mid`	Middle color for estimated outcome
`col.hi`	High color for estimated outcome
`node.col`	Color for non-terminal leaves.
`node.shape`	Shape of node. Default = "none"
`node.labels`	Logical: If TRUE, print the node labels.
`node.cond`	Logical: If TRUE, print the splitting condition inside each node.
`node.prob`	Logical: If TRUE, print the probability estimate for the first class of the outcome inside each node.
`node.estimate`	Logical: If TRUE, print the estimated outcome level inside each node.
`node.n`	Logical: If TRUE, print the number of cases (from training data) that matched this condition
`edge.col`	Color for edges.
`edge.width`	Width of edges.
`edge.labels`	Logical: If TRUE, print the splitting condition on the edge.
`arrowhead`	Character: Arrowhead shape.
`layout`	Character: Passed to `data.tree::SetGraphStyle`
`drop.leaves`	Logical: If TRUE, position leaves at the bottom of the plot.
`rankdir`	Character: Passed to `data.tree::SetGraphStyle`
`splines`	Character: Passed to `data.tree::SetGraphStyle`
`fontname`	Character: Passed to `data.tree::SetGraphStyle`

bg.color	Background color.
overlap	Character: Passed to data.tree::SetGraphStyle
prune	Logical: If TRUE, prune tree using rpart::prune.rpart
rpart.cp	Numeric: Complexity parameter for pruning. If NULL, no pruning is performed.
verbose	Logical: If TRUE, print messages.

Details

If you want to show split conditions as edge labels (edge.labels = TRUE), it is recommened to set rankdir = "LR" and node.cond = FALSE. Edge labels in graphviz are shown to the right of the edge when rankdir = "TB" and above when rankdir = "LR".

Author(s)

E.D. Gennatas

---

dplot3_conf *Plot confusion matrix*

Description

Plot confusion matrix

Usage

```
dplot3_conf(
  x,
  true.col = "#72CDF4",
  false.col = "#FEB2E0",
  pos.class = rtenv$binclasspos,
  font.size = 18,
  main = NULL,
  main.y = 1,
  main.yanchor = "bottom",
  theme = rtTheme,
  margin = list(l = 20, r = 5, b = 5, t = 20),
  filename = NULL,
  file.width = 500,
  file.height = 500,
  file.scale = 1,
  ...
)
```

Arguments

x	Confusion matrix where rows are the reference and columns are the estimated classes or rtemis `class_error` object produced by mod_error
true.col	Color for true positives & true negatives
false.col	Color for false positives & false negatives
pos.class	Integer: Index of factor level to treat as the positive class
font.size	Integer: font size
main	Character: plot title
main.y	Numeric: y position of the title
main.yanchor	Character: y anchor of the title
theme	List or Character: Either the output of a `theme_*()` function or the name of a theme. Use `themes()` to get available theme names. Theme functions are of the form `theme_<name>`.
margin	List: Plot margins
filename	Character: Path to file to save static plot.
file.width	Integer: File width in pixels for when `filename` is set.
file.height	Integer: File height in pixels for when `filename` is set.
file.scale	Numeric: If saving to file, scale plot by this number
...	Additional arguments passed to theme function.

Value

A plotly object

Author(s)

EDG

Examples

```
## Not run:
true <- factor(c("a", "a", "a", "a", "b", "b", "b", "b", "b", "b", "b", "b"))
predicted <- factor(c("a", "a", "b", "a", "b", "b", "a", "a", "b", "b", "a", "a"))
predicted.prob <- c(0.7, 0.55, 0.45, 0.62, 0.41, 0.32, 0.59, .63, .32, .21, .52, .58)
error <- mod_error(true, predicted, predicted.prob)
dplot3_conf(error)

## End(Not run)
```

dplot3_fit *True vs. Predicted Plot*

Description

A dplot3_xy wrapper for plotting true vs. predicted values

Usage

```
dplot3_fit(x, y, fit = "gam", se.fit = TRUE, ...)
```

Arguments

x	Numeric, vector/data.frame/list: True values. If y is NULL and NCOL(x) > 1, first two columns used as x and y, respectively
y	Numeric, vector/data.frame/list: Predicted values
fit	Character: **rtemis** model to calculate y ~ x fit. Options: see select_learn() Can also be Logical, which will give a GAM fit if TRUE. If you specify "NLA", the activation function should be passed as a string.
se.fit	Logical: If TRUE, draw the standard error of the fit
...	Additional arguments passed to dplot3_xy

Author(s)

EDG

Examples

```
## Not run:
x <- rnorm(500)
y <- x + rnorm(500)
dplot3_fit(x, y)

## End(Not run)
```

dplot3_graphd3 *Plot graph using* **networkD3**

Description

Plot graph using **networkD3**

Usage

```
dplot3_graphd3(
  net,
  groups = NULL,
  color.scale = NULL,
  edge.col = NULL,
  node.col = NULL,
  node.alpha = 0.5,
  edge.alpha = 0.33,
  zoom = TRUE,
  legend = FALSE,
  palette = rtPalette,
  theme = rtTheme,
  ...
)
```

Arguments

net	**igraph** network
groups	Vector, length n nodes indicating group/cluster/community membership of nodes in net
color.scale	D3 colorscale (e.g. networkD3::JS("d3.scaleOrdinal(d3.schemeCategory20b);"))
edge.col	Color for edges
node.col	Color for nodes
node.alpha	Float [0, 1]: Node opacity. Default = .5
edge.alpha	Float [0, 1]: Edge opacity. Default = .33
zoom	Logical: If TRUE, graph is zoomable. Default = TRUE
legend	Logical: If TRUE, display legend for groups
palette	Vector of colors, or Character defining a builtin palette - get options with rtpalette()
theme	rtemis theme to use
...	Additional arguments to pass to networkD3

Author(s)

E.D. Gennatas

---

dplot3_graphjs *Plot network using* **threejs::graphjs**

Description

Interactive plotting of an **igraph** net using **threejs**

Usage

```
dplot3_graphjs(
  net,
  vertex.size = 1,
  vertex.col = NULL,
  vertex.label.col = NULL,
  vertex.label.alpha = 0.66,
  vertex.frame.col = NA,
  vertex.label = NULL,
  vertex.shape = "circle",
  edge.col = NULL,
  edge.alpha = 0.5,
  edge.curved = 0.35,
  edge.width = 2,
  layout = c("fr", "dh", "drl", "gem", "graphopt", "kk", "lgl", "mds", "sugiyama"),
  coords = NULL,
  layout_params = list(),
  cluster = NULL,
  groups = NULL,
  cluster_params = list(),
  cluster_mark_groups = TRUE,
  cluster_color_vertices = FALSE,
  main = "",
  theme = rtTheme,
  theme_extra_args = list(),
  palette = rtPalette,
  mar = rep(0, 4),
  par.reset = TRUE,
  filename = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

`net` igraph network

`vertex.size` Numeric: Vertex size

`vertex.col` Color for vertices

`vertex.label.col` Color for vertex labels

`vertex.label.alpha` Numeric: transparency for `vertex.label.col`

`vertex.frame.col` Color for vertex border (frame)

`vertex.label` Character vector: Vertex labels. Default = NULL, which will keep existing names in `net` if any. Set to NA to avoid printing vertex labels

`vertex.shape`	Character, vector, length 1 or N nodes: Vertex shape. See `graphjs("vertex.shape")`. Default = "circle"
`edge.col`	Color for edges
`edge.alpha`	Numeric: Transparency for edges
`edge.curved`	Numeric: Curvature of edges. Default = .35
`edge.width`	Numeric: Edge thickness
`layout`	Character: one of: "fr", "dh", "drl", "gem", "graphopt", "kk", "lgl", "mds", "sugiyama", corresponding to all the available layouts in **igraph**
`coords`	Output of precomputed **igraph** layout. If provided, `layout` is ignored
`layout_params`	List of parameters to pass to `layout` function
`cluster`	Character: one of: "edge_betweenness", "fast_greedy", "infomap", "label_prop", "leading_eigen", "louvain", "optimal", "spinglass", "walktrap", corresponding to all the available **igraph** clustering functions
`groups`	Output of precomputed **igraph** clustering. If provided, `cluster` is ignored
`cluster_params`	List of parameters to pass to `cluster` function
`cluster_mark_groups`	Logical: If TRUE, draw polygons to indicate clusters, if `groups` or `cluster` defined
`cluster_color_vertices`	Logical: If TRUE, color vertices by cluster membership
`main`	Character: main title
`theme`	**rtemis** theme to use
`theme_extra_args`	List of extra arguments to pass to the theme function defined by `theme`. This argument is used when the extra args (...) are passed the plotting function, in this case `igraph::plot.igraph` and not to the theme function
`palette`	Color vector or name of rtemis palette
`mar`	Numeric vector, length 4: `par`'s margin argument
`par.reset`	Logical: If TRUE, reset par before exiting. Default = TRUE
`filename`	Character: If provided, save plot to this filepath
`verbose`	Logical, If TRUE, print messages to console. Default = TRUE
`...`	Extra arguments to pass to `igraph::plot.igraph()`

Author(s)

E.D. Gennatas

dplot3_heatmap *Interactive Heatmaps*

Description

Draw interactive heatmaps using heatmaply

Usage

```
dplot3_heatmap(
  x,
  Rowv = TRUE,
  Colv = TRUE,
  cluster = FALSE,
  symm = FALSE,
  cellnote = NULL,
  colorGrad.n = 101,
  colors = NULL,
  space = "rgb",
  lo = "#18A3AC",
  lomid = NULL,
  mid = NULL,
  midhi = NULL,
  hi = "#F48024",
  k_row = 1,
  k_col = 1,
  grid.gap = 0,
  limits = NULL,
  margins = NULL,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  key.title = NULL,
  showticklabels = NULL,
  colorbar_len = 0.7,
  plot_method = "plotly",
  theme = rtTheme,
  row_side_colors = NULL,
  row_side_palette = NULL,
  col_side_colors = NULL,
  col_side_palette = NULL,
  font.size = NULL,
  padding = 0,
  displayModeBar = TRUE,
  modeBar.file.format = "svg",
  filename = NULL,
  file.width = 500,
```

```
    file.height = 500,
    file.scale = 1,
    ...
  )
```

Arguments

`x`	Input matrix
`Rowv`	Logical or dendrogram. If Logical: Compute dendrogram and reorder rows. Defaults to FALSE If dendrogram: use as is, without reordering See more at `heatmaply::heatmaply("Rowv")`
`Colv`	Logical or dendrogram. If Logical: Compute dendrogram and reorder columns. Defaults to FALSE If dendrogram: use as is, without reordering See more at `heatmaply::heatmaply("Colv")`
`cluster`	Logical: If TRUE, set `Rowv` and `Colv` to TRUE
`symm`	Logical: If TRUE, treat `x` symmetrically - `x` must be a square matrix. Default = FALSE
`cellnote`	Matrix with values to be displayed on hover. Defaults to `ddSci(x)`
`colorGrad.n`	Integer: Number of distinct colors to generate using colorGrad. Default = 101
`colors`	Character: Acts as a shortcut to defining `lo`, `mid`, etc for a number of defaults: "french", "penn", "grnblkred",
`space`	Character: Which colorspace to use. Option: "rgb", or "Lab". Default = "rgb". Recommendation: If `mid` is "white" or "black" (default), use "rgb", otherwise "Lab"
`lo`	Color for low end
`lomid`	Color for low-mid
`mid`	Color for middle of the range or "mean", which will result in `colorOp(c(lo, hi), "mean")`. If `mid = NA`, then only `lo` and `hi` are used to create the color gradient.
`midhi`	Color for middle-high
`hi`	Color for high end
`k_row`	Integer: Number of desired number of groups by which to color dendrogram branches in the rows. Default = NA (determined automatically). See `heatmaply::heatmaply("k_row")`
`k_col`	Integer: Number of desired number of groups by which to color dendrogram branches in the columns. Default = NA (determined automatically). See `heatmaply::heatmaply("k_col`
`grid.gap`	Integer: Space between cells. Default = 0 (no space)
`limits`	Float, length 2: Determine color range. Default = NULL, which automatically centers values around 0
`margins`	Float, length 4: Heatmap margins.
`main`	Character: Plot title
`xlab`	Character: x-axis label
`ylab`	Character: y-axis label

`key.title`	Character: Title for the color key.
`showticklabels`	Logical: If TRUE, show tick labels.
`colorbar_len`	Numeric: Length of the colorbar.
`plot_method`	Character: Update February 2021: "ggplot" causes R session to hang on MacOS but "plotly" seems to work
`theme`	Character: "light", "dark"
`row_side_colors`	Data frame: Column names will be label names, cells should be label colors. See `heatmaply::heatmaply("row_side_colors")`
`row_side_palette`	Color palette function: See `heatmaply::heatmaply("row_side_palette")`
`col_side_colors`	Data frame: Column names will be label names, cells
`col_side_palette`	Color palette function: See `heatmaply::heatmaply("col_side_palette")`
`font.size`	Numeric: Font size
`padding`	Numeric: Padding between cells
`displayModeBar`	Logical: If TRUE, display the plotly mode bar
`modeBar.file.format`	Character: File format for image exports from the mode bar
`filename`	String (Optional: Path to file to save colorbar
`file.width`	Numeric: Width of exported image
`file.height`	Numeric: Height of exported image
`file.scale`	Numeric: Scale of exported image
`...`	Additional arguments to be passed to `heatmaply::heatmaply`

Author(s)

E.D. Gennatas

Examples

```
## Not run:
x <- rnormmat(200, 20)
xcor <- cor(x)
dplot3_heatmap(xcor)

## End(Not run)
```

dplot3_leaflet *Plot interactive choropleth map using* **leaflet**

Description

Plot interactive choropleth map using **leaflet**

Usage

```
dplot3_leaflet(
  fips,
  values,
  names = NULL,
  fillOpacity = 1,
  palette = NULL,
  color.mapping = c("Numeric", "Bin"),
  col.lo = "#0290EE",
  col.hi = "#FE4AA3",
  col.na = "#303030",
  col.highlight = "#FE8A4F",
  col.interpolate = c("linear", "spline"),
  col.bins = 21,
  domain = NULL,
  weight = 0.5,
  color = "black",
  alpha = 1,
  bg.tile.provider = leaflet::providers$Stamen.TonerBackground,
  bg.tile.alpha = 0.67,
  fg.tile.provider = leaflet::providers$Stamen.TonerLabels,
  legend.position = c("topright", "bottomright", "bottomleft", "topleft"),
  legend.alpha = 0.8,
  legend.title = NULL,
  init.lng = -98.5418083333333,
  init.lat = 39.2074138888889,
  init.zoom = 3,
  stroke = TRUE
)
```

Arguments

fips	Character vector of FIPS codes. (If numeric, it will be appropriately zero-padded)
values	Values to map to fips
names	Character vector: Optional county names to appear on hover along values
fillOpacity	Float: Opacity for fill colors. Default = 1
palette	Character: Color palette to use

`color.mapping`	Character: "Numeric" or "Bin"
`col.lo`	Overlay color mapped to lowest value
`col.hi`	Overaly color mapped to highest value
`col.na`	Color mappes to NA values
`col.highlight`	Hover border color. Default = "#FE8A4F" (orange)
`col.interpolate`	Character: "linear" or "spline"
`col.bins`	Integer: Number of color bins to create if `color.mapping = "Bin"`. Default = 21
`domain`	Limits for mapping colors to values. Default = NULL and set to range
`weight`	Float: Weight of county border lines. Default = .5
`color`	Color of county border lines. Default = "black"
`alpha`	Float: Overaly transparency. Default = 1
`bg.tile.provider`	Background tile (below overlay colors), one of `leaflet::providers`
`bg.tile.alpha`	Float: Background tile transparency. Default = .67
`fg.tile.provider`	Foreground tile (above overlay colors), one of `leaflet::providers`
`legend.position`	Character: One of: "topright", "bottomright", "bottomleft", "topleft". Default = "topright"
`legend.alpha`	Float: Legend box transparency. Default = .8
`legend.title`	Character: Defaults to name of `values` variable.
`init.lng`	Float: Center map around this longitude (in decimal form). Default = -98.54180833333334 (US geographic center)
`init.lat`	Float: Center map around this latitude (in decimal form). Default = 39.207413888888894 (US geographic center)
`init.zoom`	Integer: Initial zoom level (depends on device, i.e. window, size). Default = 3
`stroke`	Logical: If TRUE, draw polygon borders. Default = TRUE

Author(s)

E.D. Gennatas

Examples

```
## Not run:
fips <- c(06075, 42101)
population <- c(874961, 1579000)
names <- c("SF", "Philly")
dplot3_leaflet(fips, supervals, names)

## End(Not run)
```

dplot3_linad *Plot a Linear Additive Tree trained by s_LINAD using* visNetwork

Description

Plot a Linear Additive Tree trained by s_LINAD using *visNetwork*

Usage

```
dplot3_linad(
  x,
  main = NULL,
  bg = "#FFFFFF",
  shape = "box",
  nodelabels = TRUE,
  ncases.inlabels = TRUE,
  rules.on.edges = FALSE,
  top = NULL,
  root.col = "#202020",
  node.col = "#5a5a5a",
  leaf.col = "#178CCB",
  edge.col = "#848484",
  edge.width = 4,
  arrow.scale = 0.7,
  arrow.middle = FALSE,
  col.highlight = "#FE4AA3",
  node.font.col = NULL,
  edge.font.col = "#000000",
  sort.coefs = FALSE,
  height = NULL,
  width = NULL,
  levelSeparation = 100,
  tree.font.size = 22,
  edgethickness.by.ncases = FALSE,
  font.family = "Lato",
  uselog = FALSE,
  tooltip.coefs = TRUE,
  tooltip.delay = 50,
  table.font.size = "16px",
  table.dat.padding = "0px",
  table.lo.col = "#0290EE",
  table.hi.col = "#FE4AA3",
  dragNodes = FALSE,
  zoomView = FALSE,
  nodeSpacing = 150,
  blockShifting = TRUE,
  edgeMinimization = TRUE,
```

```
  parentCentralization = TRUE,
  direction = "UD",
  trace = 0
)
```

Arguments

`x`	`rtMod` object trained using s_LINAD
`main`	Character: Title.
`bg`	Background color.
`shape`	Character: Node shape; one of: "square", "triangle", "box", "circle", "dot", "star", "ellipse", "database", "text", "diamond".
`nodelabels`	Logical: If TRUE, inlcude node labels.
`ncases.inlabels`	Logical: If TRUE, include number of cases with the node labels.
`rules.on.edges`	Logical: If TRUE, display rules on edges instead of nodes.
`top`	Integer: If not NULL, only show the top `top` coefficients.
`root.col`	Color for root node.
`node.col`	Color for nodes.
`leaf.col`	Color for leaf nodes.
`edge.col`	Color for edges.
`edge.width`	Numeric: Width for edges.
`arrow.scale`	Numeric: Scale factor for arrows.
`arrow.middle`	Logical: If TRUE, draw arrows in the middle of edges.
`col.highlight`	Color for surrounding edges when node is selected.
`node.font.col`	Color for node labels. Default varies by `shape`, black or white depending if `visNetwork` draws labels on node or underneath
`edge.font.col`	Color for edge labels.
`sort.coefs`	Logical: If TRUE, sort each coefs table.
`height`	Numeric: Height for `visNetwork`. Default = NULL, i.e. auto
`width`	Numeric: Width for `visNetwork`. Default = NULL, i.e. auto
`levelSeparation`	Numeric: N of pixels to separate tree levels.
`tree.font.size`	Integer: Font size for tree labels. Default = 22
`edgethickness.by.ncases`	Logical: If TRUE, scale edge thickness by number of cases with weight = 1
`font.family`	Character: Font to use throughout. Default = 'Helvetica Neue', because otherwise it may fail on a number of external viewers.
`uselog`	Logical: If TRUE, use log10 scale for coefficient colors.
`tooltip.coefs`	Logical: If TRUE, show html coefficient tables on hover over nodes. This was placed here before a custom html table creation function was made to replace some impossibly slow alternatives.

`tooltip.delay` Numeric: Delay (in milliseconds) on mouse over before showing tooltip.

`table.font.size`
Character: Font size for html coefficient on-hover tables.

`table.dat.padding`
Ignore, has no visible effect. Otherwise, Character: html table padding.

`table.lo.col` Color for lowest coefficient values (negative)

`table.hi.col` Color for highest coefficient values (positive).

`dragNodes` Logical: If TRUE, allow dragging nodes.

`zoomView` Logical: If TRUE, allow zooming.

`nodeSpacing` Numeric: Spacing between nodes.

`blockShifting` Logical: If TRUE, allow block shifting.

`edgeMinimization`
Logical: If TRUE, minimize edge length.

`parentCentralization`
Logical: If TRUE, centralize parent nodes.

`direction` Character: Direction of tree. One of: "UD", "DU", "LR", "RL".

`trace` Integer: If > 0, print info to console (not particularly informative).

Author(s)

E.D. Gennatas

`dplot3_pie` *Interactive Pie Chart*

Description

Draw interactive pie charts using `plotly`

Usage

```
dplot3_pie(
  x,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  col = NULL,
  alpha = 0.8,
  bg = NULL,
  plot.bg = NULL,
  theme = getOption("rt.theme", "black"),
  palette = rtPalette,
  category.names = NULL,
  textinfo = "label+percent",
```

```
  font.size = 16,
  labs.col = NULL,
  legend = TRUE,
  legend.col = NULL,
  sep.col = NULL,
  margin = list(b = 50, l = 50, t = 50, r = 20),
  padding = 0,
  displayModeBar = TRUE,
  modeBar.file.format = "svg",
  filename = NULL,
  file.width = 500,
  file.height = 500,
  file.scale = 1,
  ...
)
```

Arguments

x	data.frame: Input: Either a) 1 numeric column with categories defined by rownames, or b) two columns, the first is category names, the second numeric or c) a numeric vector with categories defined using the `category.names` argument
main	Character: Plot title. Default = NULL, which results in colnames(x)[1],
xlab	Character: x-axis label.
ylab	Character: y-axis label.
col	Color, vector: Color for bars. Default NULL, which will draw colors from `palette`
alpha	Float (0, 1]: Transparency for bar colors. Default = .8
bg	Background color
plot.bg	Plot background color
theme	Character: "light", "dark". Default = `getOption("rt.theme", "light")`
palette	Character: Name of **rtemis** palette to use. Default = "rtCol1". Only used if `col = NULL`
category.names	Character, vector, length = NROW(x): Category names. Default = NULL, which uses either `rownames(x)`, or the first column of `x` if `ncol(x) = 2`
textinfo	Character: Info to show over each slince: "label", "percent", "label+percent" Default = "label+percent"
font.size	Float: Font size for all labels. Default = 16
labs.col	Color of labels
legend	Logical: If TRUE, draw legend. Default = NULL, and will be turned on if there is more than one feature present
legend.col	Color: Legend text color. Default = NULL, determined by theme
sep.col	Separator color
margin	Named list: plot margins.

padding	Integer: N pixels to pad plot.
displayModeBar	Logical: If TRUE, show plotly's modebar
modeBar.file.format	Character: "svg", "png", "jpeg", "pdf" / any output file type supported by plotly and your system
filename	Character: Path to file to save static plot. Default = NULL
file.width	Integer: File width in pixels for when `filename` is set.
file.height	Integer: File height in pixels for when `filename` is set.
file.scale	Numeric: If saving to file, scale plot by this number
...	Additional arguments passed to theme

Author(s)

E.D. Gennatas

Examples

```
## Not run:
dplot3_pie(VADeaths[, 1, drop = F])

## End(Not run)
```

`dplot3_protein` *Plot the amino acid sequence with annotations*

Description

Plot the amino acid sequence with annotations

Usage

```
dplot3_protein(
  x,
  site = NULL,
  region = NULL,
  ptm = NULL,
  clv = NULL,
  variant = NULL,
  disease.variants = NULL,
  n.per.row = NULL,
  main = NULL,
  main.xy = c(0.055, 0.975),
  main.xref = "paper",
  main.yref = "paper",
  main.xanchor = "middle",
  main.yanchor = "top",
```

```
layout = c("simple", "grid", "1curve", "2curve"),
show.markers = TRUE,
show.labels = TRUE,
font.size = 18,
label.col = NULL,
scatter.mode = "markers+lines",
marker.size = 28,
marker.col = NULL,
marker.alpha = 1,
marker.symbol = "circle",
line.col = NULL,
line.alpha = 1,
line.width = 2,
show.full.names = TRUE,
region.scatter.mode = "markers+lines",
region.style = 3,
region.marker.size = marker.size,
region.marker.alpha = 0.6,
region.marker.symbol = "circle",
region.line.dash = "solid",
region.line.shape = "line",
region.line.smoothing = 1,
region.line.width = 1,
region.line.alpha = 0.6,
theme = rtTheme,
region.palette = rtPalette,
region.outline.only = FALSE,
region.outline.pad = 2,
region.pad = 0.35,
region.fill.alpha = 0.1666666,
region.fill.shape = "line",
region.fill.smoothing = 1,
bpadcx = 0.5,
bpadcy = 0.5,
site.marker.size = marker.size,
site.marker.symbol = marker.symbol,
site.marker.alpha = 1,
site.border.width = 1.5,
site.palette = rtPalette,
variant.col = "#FA6E1E",
disease.variant.col = "#E266AE",
showlegend.ptm = TRUE,
ptm.col = NULL,
ptm.symbol = "circle",
ptm.offset = 0.12,
ptm.pad = 0.35,
ptm.marker.size = marker.size/4.5,
clv.col = NULL,
```

```
  clv.symbol = "triangle-down",
  clv.offset = 0.12,
  clv.pad = 0.35,
  clv.marker.size = marker.size/4,
  annotate.position.every = 10,
  annotate.position.alpha = 0.5,
  annotate.position.ay = -0.4 * marker.size,
  position.font.size = font.size - 6,
  legend.xy = c(0.97, 0.954),
  legend.xanchor = "left",
  legend.yanchor = "top",
  legend.orientation = "v",
  legend.col = NULL,
  legend.bg = "#FFFFFF00",
  legend.border.col = "#FFFFFF00",
  legend.borderwidth = 0,
  legend.group.gap = 0,
  margin = list(b = 0, l = 0, t = 0, r = 0, pad = 0),
  showgrid.x = FALSE,
  showgrid.y = FALSE,
  automargin.x = TRUE,
  automargin.y = TRUE,
  xaxis.autorange = TRUE,
  yaxis.autorange = "reversed",
  scaleanchor.y = "x",
  scaleratio.y = 1,
  hoverlabel.align = "left",
  displayModeBar = TRUE,
  modeBar.file.format = "svg",
  scrollZoom = TRUE,
  filename = NULL,
  file.width = 1320,
  file.height = 990,
  file.scale = 1,
  width = NULL,
  height = NULL,
  verbosity = 1,
  ...
)
```

Arguments

`x`	Character vector: amino acid sequence (1-letter abbreviations) OR `a3` object OR Character: path to JSON file OR Character: UniProt accession number
`site`	Named list of lists with indices of sites. These will be highlighted by coloring the border of markers
`region`	Named list of lists with indices of regions. These will be highlighted by coloring the markers and lines of regions using the `palette` colors

`ptm`	List of post-translational modifications
`clv`	List of cleavage sites
`variant`	List of variant information
`disease.variants`	List of disease variant information
`n.per.row`	Integer: Number of amino acids to show per row
`main`	Character: Main title
`main.xy`	Numeric vector, length 2: x and y coordinates for title. e.g. if `main.xref` and `main.yref` are `"paper"`: `c(0.055, .975)` is top left, `c(.5, .975)` is top and middle
`main.xref`	Character: xref for title
`main.yref`	Character: yref for title
`main.xanchor`	Character: xanchor for title
`main.yanchor`	Character: yanchor for title
`layout`	Character: "1curve", "grid": type of layout to use
`show.markers`	Logical: If TRUE, show amino acid markers
`show.labels`	Logical: If TRUE, annotate amino acids with elements
`font.size`	Integer: Font size for labels
`label.col`	Color for labels
`scatter.mode`	Character: Mode for scatter plot
`marker.size`	Integer: Size of markers
`marker.col`	Color for markers
`marker.alpha`	Numeric: Alpha for markers
`marker.symbol`	Character: Symbol for markers
`line.col`	Color for lines
`line.alpha`	Numeric: Alpha for lines
`line.width`	Numeric: Width for lines
`show.full.names`	Logical: If TRUE, show full names of amino acids
`region.scatter.mode`	Character: Mode for scatter plot
`region.style`	Integer: Style for regions
`region.marker.size`	Integer: Size of region markers
`region.marker.alpha`	Numeric: Alpha for region markers
`region.marker.symbol`	Character: Symbol for region markers
`region.line.dash`	Character: Dash for region lines

region.line.shape
: Character: Shape for region lines

region.line.smoothing
: Numeric: Smoothing for region lines

region.line.width
: Numeric: Width for region lines

region.line.alpha
: Numeric: Alpha for region lines

theme
: Character: Theme to use: Run `themes()` for available themes

region.palette
: Named list of colors for regions

region.outline.only
: Logical: If TRUE, only show outline of regions

region.outline.pad
: Numeric: Padding for region outline

region.pad
: Numeric: Padding for region

region.fill.alpha
: Numeric: Alpha for region fill

region.fill.shape
: Character: Shape for region fill

region.fill.smoothing
: Numeric: Smoothing for region fill

bpadcx
: Numeric: Padding for region border

bpadcy
: Numeric: Padding for region border

site.marker.size
: Integer: Size of site markers

site.marker.symbol
: Character: Symbol for site markers

site.marker.alpha
: Numeric: Alpha for site markers

site.border.width
: Numeric: Width for site borders

site.palette
: Named list of colors for sites

variant.col
: Color for variants

disease.variant.col
: Color for disease variants

showlegend.ptm
: Logical: If TRUE, show legend for PTMs

ptm.col
: Named list of colors for PTMs

ptm.symbol
: Character: Symbol for PTMs

ptm.offset
: Numeric: Offset for PTMs

ptm.pad
: Numeric: Padding for PTMs

ptm.marker.size
: Integer: Size of PTM markers

clv.col
: Color for cleavage site annotations

clv.symbol
: Character: Symbol for cleavage site annotations

clv.offset
: Numeric: Offset for cleavage site annotations

clv.pad
: Numeric: Padding for cleavage site annotations

clv.marker.size
: Integer: Size of cleavage site annotation markers

annotate.position.every
: Integer: Annotate every nth position

annotate.position.alpha
: Numeric: Alpha for position annotations

annotate.position.ay
: Numeric: Y offset for position annotations

position.font.size
: Integer: Font size for position annotations

legend.xy
: Numeric vector, length 2: x and y coordinates for legend

legend.xanchor
: Character: xanchor for legend

legend.yanchor
: Character: yanchor for legend

legend.orientation
: Character: Orientation for legend

legend.col
: Color for legend

legend.bg
: Color for legend background

legend.border.col
: Color for legend border

legend.borderwidth
: Numeric: Width for legend border

legend.group.gap
: Numeric: Gap between legend groups

margin
: List: Margin settings

showgrid.x
: Logical: If TRUE, show x grid

showgrid.y
: Logical: If TRUE, show y grid

automargin.x
: Logical: If TRUE, use automatic margin for x axis

automargin.y
: Logical: If TRUE, use automatic margin for y axis

xaxis.autorange
: Logical: If TRUE, use automatic range for x axis

yaxis.autorange
: Character: If TRUE, use automatic range for y axis

scaleanchor.y
: Character: Scale anchor for y axis

scaleratio.y
: Numeric: Scale ratio for y axis

hoverlabel.align
: Character: Alignment for hover label

displayModeBar
: Logical: If TRUE, display mode bar

`modeBar.file.format`
Character: File format for mode bar

`scrollZoom`	Logical: If TRUE, enable scroll zoom
`filename`	Character: File name to save plot
`file.width`	Integer: Width for saved file
`file.height`	Integer: Height for saved file
`file.scale`	Numeric: Scale for saved file
`width`	Integer: Width for plot
`height`	Integer: Height for plot
`verbosity`	Integer: If > 0, print messages to console. If > 1, print trace messages
`...`	Additional arguments to pass to the theme function

Value

A plotly object

Author(s)

E.D. Gennatas

Examples

```
## Not run:
tau <- seqinr::read.fasta("https://rest.uniprot.org/uniprotkb/P10636.fasta",
  seqtype = "AA"
)
dplot3_protein(as.character(tau[[1]]))

# or directly using the UniProt accession number:
dplot3_protein("P10636")

## End(Not run)
```

`dplot3_pvals` *Barplot p-values using dplot3_bar*

Description

Plot 1 - p-values as a barplot

Usage

```
dplot3_pvals(
  x,
  xnames = NULL,
  yname = NULL,
  p.adjust.method = "none",
  pval.hline = 0.05,
  hline.col = "#FE4AA3",
  hline.dash = "dash",
  ...
)
```

Arguments

`x`	Float, vector: p-values
`xnames`	Character, vector: feature names
`yname`	Character: outcome name
`p.adjust.method`	Character: method for p.adjust. Default = "none"
`pval.hline`	Float: Significance level at which to plot horizontal line. Default = .05
`hline.col`	Color for `pval.hline`. Default = "#FE4AA3"
`hline.dash`	Character: type of line to draw. Default = "dash"
`...`	Additional arguments passed to dplot3_bar

Author(s)

E.D. Gennatas

`dplot3_spectrogram` *Interactive Spectrogram*

Description

Draw interactive spectrograms using `plotly`

Usage

```
dplot3_spectrogram(
  x,
  y,
  z,
  colorGrad.n = 101,
  colors = NULL,
  xlab = "Time",
  ylab = "Frequency",
```

```
  zlab = "Power",
  hover.xlab = xlab,
  hover.ylab = ylab,
  hover.zlab = zlab,
  zmin = NULL,
  zmax = NULL,
  zauto = TRUE,
  hoverlabel.align = "right",
  colorscale = "Jet",
  colorbar.y = 0.5,
  colorbar.yanchor = "middle",
  colorbar.xpad = 0,
  colorbar.ypad = 0,
  colorbar.len = 0.75,
  colorbar.title.side = "bottom",
  showgrid = FALSE,
  space = "rgb",
  lo = "#18A3AC",
  lomid = NULL,
  mid = NULL,
  midhi = NULL,
  hi = "#F48024",
  grid.gap = 0,
  limits = NULL,
  main = NULL,
  key.title = NULL,
  showticklabels = NULL,
  theme = rtTheme,
  font.size = NULL,
  padding = 0,
  displayModeBar = TRUE,
  modeBar.file.format = "svg",
  filename = NULL,
  file.width = 500,
  file.height = 500,
  file.scale = 1,
  ...
)
```

Arguments

x	Numeric: Time
y	Numeric: Frequency
z	Numeric: Power
colorGrad.n	Integer: Number of distinct colors to generate using colorGrad. Default = 101
colors	Character: Acts as a shortcut to defining lo, mid, etc for a number of defaults: "french", "penn", "grnblkred",

`xlab`	Character: x-axis label
`ylab`	Character: y-axis label
`zlab`	Character: z-axis label
`hover.xlab`	Character: x-axis label for hover
`hover.ylab`	Character: y-axis label for hover
`hover.zlab`	Character: z-axis label for hover
`zmin`	Numeric: Minimum value for color scale
`zmax`	Numeric: Maximum value for color scale
`zauto`	Logical: If TRUE, automatically set zmin and zmax
`hoverlabel.align`	Character: Alignment of hover labels
`colorscale`	Character: Color scale. Default = "Jet"
`colorbar.y`	Numeric: Y position of colorbar
`colorbar.yanchor`	Character: Y anchor of colorbar
`colorbar.xpad`	Numeric: X padding of colorbar
`colorbar.ypad`	Numeric: Y padding of colorbar
`colorbar.len`	Numeric: Length of colorbar
`colorbar.title.side`	Character: Side of colorbar title
`showgrid`	Logical: If TRUE, show grid
`space`	Character: Which colorspace to use. Option: "rgb", or "Lab". Default = "rgb". Recommendation: If `mid` is "white" or "black" (default), use "rgb", otherwise "Lab"
`lo`	Color for low end
`lomid`	Color for low-mid
`mid`	Color for middle of the range or "mean", which will result in `colorOp(c(lo, hi), "mean")`. If `mid = NA`, then only `lo` and `hi` are used to create the color gradient.
`midhi`	Color for middle-high
`hi`	Color for high end
`grid.gap`	Integer: Space between cells. Default = 0 (no space)
`limits`	Numeric, length 2: Determine color range. Default = NULL, which automatically centers values around 0
`main`	Character: Plot title
`key.title`	Character: Title of the key
`showticklabels`	Logical: If TRUE, show tick labels
`theme`	Character: "light", "dark"
`font.size`	Numeric: Font size

`padding`	Numeric: Padding between cells
`displayModeBar`	Logical: If TRUE, display the plotly mode bar
`modeBar.file.format`	Character: File format for image exports from the mode bar
`filename`	String (Optional: Path to file to save colorbar
`file.width`	Numeric: Width of exported image
`file.height`	Numeric: Height of exported image
`file.scale`	Numeric: Scale of exported image
`...`	Additional arguments to be passed to `heatmaply::heatmaply`

Details

To set custom colors, use a minimum of `lo` and `hi`, optionnaly also `lomid`, `mid`, `midhi` colors and set `colorscale = NULL`.

Author(s)

E.D. Gennatas

`dplot3_table` *Simple HTML table*

Description

Draw an html table using `plotly`

Usage

```
dplot3_table(
  x,
  .ddSci = TRUE,
  main = NULL,
  main.col = "black",
  main.x = 0,
  main.xanchor = "auto",
  fill.col = "#18A3AC",
  table.bg = "white",
  bg = "white",
  line.col = "white",
  lwd = 1,
  header.font.col = "white",
  table.font.col = "gray20",
  font.size = 14,
  font.family = "Helvetica Neue",
  margin = list(l = 0, r = 5, t = 30, b = 0, pad = 0)
)
```

Arguments

x	data.frame: Table to draw
.ddSci	Logical: If TRUE, apply ddSci to numeric columns.
main	Character: Table tile.
main.col	Color: Title color.
main.x	Float [0, 1]: Align title: 0: left, .5: center, 1: right.
main.xanchor	Character: "auto", "left", "right": plotly's layout xanchor for title. Default = "auto"
fill.col	Color: Used to fill header with column names and first column with row names.
table.bg	Color: Table background.
bg	Color: Background.
line.col	Color: Line color.
lwd	Float: Line width. Default = 1
header.font.col	Color: Header font color.
table.font.col	Color: Table font color.
font.size	Integer: Font size.
font.family	Character: Font family.
margin	List: plotly's margins.

Author(s)

E.D. Gennatas

---

dplot3_ts *Interactive Timeseries Plots*

Description

Draw interactive timeseries plots using `plotly`

Usage

```
dplot3_ts(
  x,
  time,
  window = 7L,
  group = NULL,
  roll.fn = c("mean", "median", "max", "none"),
  roll.col = NULL,
  roll.alpha = 1,
  roll.lwd = 2,
```

```
    roll.name = NULL,
    alpha = NULL,
    align = "center",
    group.names = NULL,
    xlab = "Time",
    n.xticks = 12,
    scatter.type = "scatter",
    legend = TRUE,
    x.showspikes = TRUE,
    y.showspikes = FALSE,
    spikedash = "solid",
    spikemode = "across",
    spikesnap = "hovered data",
    spikecolor = NULL,
    spikethickness = 1,
    displayModeBar = TRUE,
    modeBar.file.format = "svg",
    theme = rtTheme,
    palette = rtPalette,
    filename = NULL,
    file.width = 500,
    file.height = 500,
    file.scale = 1,
    ...
)
```

Arguments

`x`	Numeric vector of values to plot or list of vectors
`time`	Numeric or Date vector of time corresponding to values of `x`
`window`	Integer: apply `roll.fn` over this many units of time
`group`	Factor defining groups
`roll.fn`	Character: "mean", "median", "max", or "sum": Function to apply on rolling windows of `x`
`roll.col`	Color for rolling line
`roll.alpha`	Numeric: transparency for rolling line
`roll.lwd`	Numeric: width of rolling line
`roll.name`	Rolling function name (for annotation)
`alpha`	Numeric [0, 1]: Transparency
`align`	Character: "center", "right", or "left"
`group.names`	Character vector of group names
`xlab`	Character: x-axis label
`n.xticks`	Integer: number of x-axis ticks to use (approximately)
`scatter.type`	Character: "scatter" or "lines"

legend	Logical: If TRUE, show legend
x.showspikes	Logical: If TRUE, show x-axis spikes on hover
y.showspikes	Logical: If TRUE, show y-axis spikes on hover
spikedash	Character: dash type string ("solid", "dot", "dash", "longdash", "dashdot", or "longdashdot") or a dash length list in px (eg "5px,10px,2px,2px")
spikemode	Character: If "toaxis", spike line is drawn from the data point to the axis the series is plotted on. If "across", the line is drawn across the entire plot area, and supercedes "toaxis". If "marker", then a marker dot is drawn on the axis the series is plotted on
spikesnap	Character: "data", "cursor", "hovered data". Determines whether spikelines are stuck to the cursor or to the closest datapoints.
spikecolor	Color for spike lines
spikethickness	Numeric: spike line thickness
displayModeBar	Logical: If TRUE, display plotly's modebar
modeBar.file.format	Character: modeBar image export file format
theme	Character: theme name or list of theme parameters
palette	Character: palette name, or list of colors
filename	Character: Path to filename to save plot
file.width	Numeric: image export width
file.height	Numeric: image export height
file.scale	Numeric: image export scale
...	Additional arguments to be passed to dplot3_xy

Author(s)

E.D. Gennatas

Examples

```
## Not run:
time <- sample(seq(as.Date("2020-03-01"), as.Date("2020-09-23"), length.out = 140))
x1 <- rnorm(140)
x2 <- rnorm(140, 1, 1.2)
# Single timeseries
dplot3_ts(x1, time)
# Multiple timeseries input as list
dplot3_ts(list(Alpha = x1, Beta = x2), time)
# Multiple timeseries grouped by group, different lengths
time1 <- sample(seq(as.Date("2020-03-01"), as.Date("2020-07-23"), length.out = 100))
time2 <- sample(seq(as.Date("2020-05-01"), as.Date("2020-09-23"), length.out = 140))
time <- c(time1, time2)
x <- c(rnorm(100), rnorm(140, 1, 1.5))
group <- c(rep("Alpha", 100), rep("Beta", 140))
dplot3_ts(x, time, 7, group)
```

```
## End(Not run)
```

---

dplot3_varimp *Interactive Variable Importance Plot*

Description

Plot variable importance using `plotly`

Usage

```
dplot3_varimp(
  x,
  names = NULL,
  main = NULL,
  xlab = "Variable Importance",
  ylab = "",
  plot.top = 1,
  labelify = TRUE,
  col = NULL,
  alpha = 1,
  palette = NULL,
  mar = NULL,
  font.size = 16,
  axis.font.size = 14,
  theme = rtTheme,
  showlegend = TRUE,
  ...
)
```

Arguments

`x`	Vector, numeric: Input
`names`	Vector, string: Names of features
`main`	Character: main title
`xlab`	Character: x-axis label
`ylab`	Character: y-axis label
`plot.top`	Float or Integer: If <= 1, plot this percent highest absolute values, otherwise plot this many top values. i.e.: `plot.top = .2` will print the top 20% highest values, and `plot.top = 20` will plot the top 20 highest values
`labelify`	Logical: If TRUE convert `names(x)` using labelify. Default = TRUE
`col`	Vector, colors: Single value, or multiple values to define bar (feature) color(s)
`alpha`	Numeric: Transparency

`palette`	Character: Name of **rtemis** palette to use.
`mar`	Vector, numeric, length 4: Plot margins in pixels (NOT inches). Default = c(50, 110, 50, 50)
`font.size`	Integer: Overall font size to use (essentially for the title at this point). Default = 14
`axis.font.size`	Integer: Font size to use for axis labels and tick labels (Seems not to be in same scale as `font.size` for some reason. Experiment!)
`theme`	Output of an rtemis theme function (list of parameters) or theme name. Use `themes()` to print available themes.
`showlegend`	Logical: If TRUE, show legend
`...`	Additional arguments passed to theme

Details

A simple `plotly` wrapper to plot horizontal barplots, sorted by value, which can be used to visualize variable importance, model coefficients, etc.

Author(s)

E.D. Gennatas

Examples

```
# made-up data
x <- rnorm(10)
names(x) <- paste0("Feature_", seq(x))
dplot3_varimp(x)
```

`dplot3_volcano` *Volcano Plot*

Description

Volcano Plot

Usage

```
dplot3_volcano(
  x,
  pvals,
  xnames = NULL,
  group = NULL,
  x.thresh = 0,
  p.thresh = 0.05,
  p.transform = function(x) -log10(x),
 p.adjust.method = c("holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr",
```

```
    "none"),
  legend = NULL,
  legend.lo = NULL,
  legend.hi = NULL,
  label.lo = "Low",
  label.hi = "High",
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  margin = list(b = 65, l = 65, t = 50, r = 10, pad = 0),
  xlim = NULL,
  ylim = NULL,
  alpha = NULL,
  hline = NULL,
  hline.col = NULL,
  hline.width = 1,
  hline.dash = "solid",
  hline.annotate = NULL,
  hline.annotation.x = 1,
  annotate = TRUE,
  annotate.col = theme$labs.col,
  theme = rtTheme,
  font.size = 16,
  palette = NULL,
  legend.x.lo = NULL,
  legend.x.hi = NULL,
  legend.y = 0.97,
  annotate.n = 7,
  ax.lo = NULL,
  ay.lo = NULL,
  ax.hi = NULL,
  ay.hi = NULL,
  annotate.alpha = 0.7,
  hovertext = NULL,
  displayModeBar = FALSE,
  filename = NULL,
  file.width = 500,
  file.height = 500,
  file.scale = 1,
  verbose = TRUE,
  ...
)
```

Arguments

x	Numeric vector: Input values, e.g. log2 fold change, coefficients, etc.
pvals	Numeric vector: p-values
xnames	Character vector: x names

`group`	Factor: Used to color code points. If NULL, significant points below `x.thresh`, non-significant points, and significant points above `x.thresh` will be plotted with the first, second and third color fo `palette`
`x.thresh`	Numeric x-axis threshold separating low from high
`p.thresh`	Numeric: p-value threshold of significance. Default = .05
`p.transform`	function. Default = `\(x) -log10(x)`
`p.adjust.method`	Character: p-value adjustment method. "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none" Default = "holm". Use "none" for raw p-values.
`legend`	Logical: If TRUE, show legend. Will default to FALSE, if `group = NULL`, otherwise to TRUE
`legend.lo`	Character: Legend to annotate significant points below the `x.thresh`
`legend.hi`	Character: Legend to annotate significant points above the `x.thresh`
`label.lo`	Character: label for low values
`label.hi`	Character: label for high values
`main`	Character: Main plot title.
`xlab`	Character: x-axis label
`ylab`	Character: y-axis label
`margin`	Named list of plot margins. Default = `list(b = 65, l = 65, t = 50, r = 10, pad = 0)`
`xlim`	Numeric vector, length 2: x-axis limits
`ylim`	Numeric vector, length 2: y-axis limits
`alpha`	Numeric: point transparency
`hline`	Numeric: If defined, draw a horizontal line at this y value.
`hline.col`	Color for `hline`. Default = "#ff0000" (red)
`hline.width`	Numeric: Width for `hline`. Default = 1
`hline.dash`	Character: Type of line to draw: "solid", "dot", "dash", "longdash", "dashdot", or "longdashdot"
`hline.annotate`	Character: Text of horizontal line annotation if `hline` is set
`hline.annotation.x`	Numeric: x position to place annotation with paper as reference. 0: to the left of the plot area; 1: to the right of the plot area
`annotate`	Logical: If TRUE, annotate significant points
`annotate.col`	Color for annotations
`theme`	List or Character: Either the output of a `theme_*()` function or the name of a theme. Use `themes()` to get available theme names. Theme functions are of the form `theme_<name>`.
`font.size`	Float: Font size for all labels. Default = 16
`palette`	Character: Name of **rtemis** palette to use. Default = "rtCol1". Only used if `col = NULL`

legend.x.lo	Numeric: x position of legend.lo
legend.x.hi	Numeric: x position of legend.hi
legend.y	Numeric: y position for legend.lo and legend.hi
annotate.n	Integer: Number of significant points to annotate
ax.lo	Numeric: Sets the x component of the arrow tail about the arrow head for significant points below x.thresh
ay.lo	Numeric: Sets the y component of the arrow tail about the arrow head for significant points below x.thresh
ax.hi	Numeric: Sets the x component of the arrow tail about the arrow head for significant points above x.thresh
ay.hi	Numeric: Sets the y component of the arrow tail about the arrow head for significant points above x.thresh
annotate.alpha	Numeric: Transparency for annotations
hovertext	List of character vectors with hovertext to include for each group of markers
displayModeBar	Logical: If TRUE, show plotly's modebar
filename	Character: Path to file to save static plot. Default = NULL
file.width	Integer: File width in pixels for when filename is set.
file.height	Integer: File height in pixels for when filename is set.
file.scale	Numeric: If saving to file, scale plot by this number
verbose	Logical: If TRUE, print messages to console
...	Additional parameters passed to dplot3_xy

Author(s)

E.D. Gennatas

Examples

```
## Not run:
set.seed(2019)
x <- rnormmat(500, 500)
y <- x[, 3] + x[, 5] - x[, 9] + x[, 15] + rnorm(500)
mod <- massGLM(y, x)
dplot3_volcano(mod$summary$`Coefficient y`, mod$summary$`p_value y`)

## End(Not run)
```

dplot3_x *Interactive Univariate Plots*

Description

Draw interactive univariate plots using plotly

Usage

```
dplot3_x(
  x,
  type = c("density", "histogram"),
  mode = c("overlap", "ridge"),
  group = NULL,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  col = NULL,
  alpha = 0.75,
  plot.bg = NULL,
  theme = rtTheme,
  palette = rtPalette,
  axes.square = FALSE,
  group.names = NULL,
  font.size = 16,
  font.alpha = 0.8,
  legend = NULL,
  legend.xy = c(0, 1),
  legend.col = NULL,
  legend.bg = "#FFFFFF00",
  legend.border.col = "#FFFFFF00",
  bargap = 0.05,
  vline = NULL,
  vline.col = theme$fg,
  vline.width = 1,
  vline.dash = "dot",
  text = NULL,
  text.x = 1,
  text.xref = "paper",
  text.xanchor = "left",
  text.y = 1,
  text.yref = "paper",
  text.yanchor = "top",
  text.col = theme$fg,
  margin = list(b = 65, l = 65, t = 50, r = 10, pad = 0),
  automargin.x = TRUE,
  automargin.y = TRUE,
```

```
  zerolines = FALSE,
  density.kernel = "gaussian",
  density.bw = "SJ",
  histnorm = c("", "density", "percent", "probability", "probability density"),
  histfunc = c("count", "sum", "avg", "min", "max"),
  hist.n.bins = 20,
  barmode = "overlay",
  ridge.sharex = TRUE,
  ridge.y.labs = FALSE,
  ridge.order.on.mean = TRUE,
  displayModeBar = TRUE,
  modeBar.file.format = "svg",
  width = NULL,
  height = NULL,
  filename = NULL,
  file.width = 500,
  file.height = 500,
  file.scale = 1,
  ...
)
```

Arguments

x	Numeric, vector / data.frame /list: Input. If not a vector, each column of or each element
type	Character: "density" or "histogram"
mode	Character: "overlap", "ridge". How to plot different groups; on the same axes ("overlap"), or on separate plots with the same x-axis ("ridge")
group	Vector: Will be converted to factor; levels define group members. Default = NULL
main	Character: Main plot title.
xlab	Character: x-axis label.
ylab	Character: y-axis label.
col	Color, vector: Color for bars. Default NULL, which will draw colors from palette
alpha	Float (0, 1]: Transparency for bar colors. Default = .8
plot.bg	Color: Background color for plot area
theme	List or Character: Either the output of a `theme_*()` function or the name of a theme. Use `themes()` to get available theme names. Theme functions are of the form `theme_<name>`.
palette	Character: Name of **rtemis** palette to use. Default = "rtCol1". Only used if `col = NULL`
axes.square	Logical: If TRUE: draw a square plot to fill the graphic device. Default = FALSE. Note: If TRUE, the device size at time of call is captured and height and width are set so as to draw the largest square available. This means that resizing the device window will not automatically resize the plot.

group.names	Character, vector, length = NROW(x): Group names. Default = NULL, which uses rownames(x)
font.size	Float: Font size for all labels. Default = 16
font.alpha	Float: Alpha transparency for font.
legend	Logical: If TRUE, draw legend. Default = NULL, which will be set to TRUE if x is a list of more than 1 element
legend.xy	Float, vector, length 2: Relative x, y position for legend. Default = c(0, 1), which places the legend top left within the plot area. Set to NULL to place legend top right beside the plot area
legend.col	Color: Legend text color. Default = NULL, determined by theme
legend.bg	Color: Background color for legend
legend.border.col	Color: Border color for legend
bargap	Float: The gap between adjacent histogram bars in plot fraction.
vline	Float, vector: If defined, draw a vertical line at this x value(s). Default = NULL
vline.col	Color for vline. Default = theme$fg
vline.width	Float: Width for vline. Default = 1
vline.dash	Character: Type of line to draw: "solid", "dot", "dash", "longdash", "dashdot", or "longdashdot"
text	Character: If defined, add this text over the plot
text.x	Float: x-coordinate for text
text.xref	Character: "x": text.x refers to plot's x-axis; "paper": text.x refers to plotting area from 0-1
text.xanchor	Character: "auto", "left", "center", "right"
text.y	Float: y-coordinate for text
text.yref	Character: "y": text.y refers to plot's y-axis; "paper": text.y refers to plotting area from 0-1
text.yanchor	Character: "auto", "top", "middle", "bottom"
text.col	Color for text. Default = theme$fg
margin	Named list: plot margins.
automargin.x	Logical: If TRUE, automatically set x-axis amrgins
automargin.y	Logical: If TRUE, automatically set y-axis amrgins
zerolines	Logical: If TRUE, draw lines at y = 0.
density.kernel	Character: Kernel to use for density estimation.
density.bw	Character: Bandwidth to use for density estimation.
histnorm	Character: NULL, "percent", "probability", "density", "probability density"
histfunc	Character: "count", "sum", "avg", "min", "max".
hist.n.bins	Integer: Number of bins to use if type = "histogram".

barmode	Character: Type of bar plot to make: "group", "relative", "stack", "overlay". Default = "group". Use "relative" for stacked bars, wich handles negative values correctly, unlike "stack", as of writing.
ridge.sharex	Logical: If TRUE, draw single x-axis when `mode = "ridge"`.
ridge.y.labs	Lofical: If TRUE, show individual y labs when `mode = "ridge"`.
ridge.order.on.mean	Logical: If TRUE, order groups by mean value when `mode = "ridge"`. Turn to FALSE, if, for example, groups are ordered by date or similar.
displayModeBar	Logical: If TRUE, show plotly's modebar
modeBar.file.format	Character: "svg", "png", "jpeg", "pdf" / any output file type supported by plotly and your system
width	Float: Force plot size to this width. Default = NULL, i.e. fill available space
height	Float: Force plot size to this height. Default = NULL, i.e. fill available space
filename	Character: Path to file to save static plot.
file.width	Integer: File width in pixels for when `filename` is set.
file.height	Integer: File height in pixels for when `filename` is set.
file.scale	Numeric: If saving to file, scale plot by this number
...	Additional arguments passed to theme function.

Details

If input is data.frame, non-numeric variables will be removed

Value

A plotly object

Author(s)

E.D. Gennatas

Examples

```
## Not run:
dplot3_x(iris)
dplot3_x(split(iris$Sepal.Length, iris$Species), xlab = "Sepal Length")

## End(Not run)
```

dplot3_xt *Plot timeseries data*

Description

Plot timeseries data

Usage

```
dplot3_xt(
  x,
  y = NULL,
  x2 = NULL,
  y2 = NULL,
  which.xy = NULL,
  which.xy2 = NULL,
  shade.bin = NULL,
  shade.interval = NULL,
  shade.col = NULL,
  shade.x = NULL,
  shade.name = "",
  shade.showlegend = FALSE,
  ynames = NULL,
  y2names = NULL,
  xlab = NULL,
  ylab = NULL,
  y2lab = NULL,
  xunits = NULL,
  yunits = NULL,
  y2units = NULL,
  yunits.col = NULL,
  y2units.col = NULL,
  zt = NULL,
  show.zt = TRUE,
  show.zt.every = NULL,
  zt.nticks = 18L,
  main = NULL,
  main.y = 1,
  main.yanchor = "bottom",
  x.nticks = 0,
  y.nticks = 0,
  show.rangeslider = NULL,
  slider.start = NULL,
  slider.end = NULL,
  theme = rtTheme,
  palette = rtpalette(rtPalette),
  font.size = 16,
```

```
    yfill = "none",
    y2fill = "none",
    fill.alpha = 0.2,
    yline.width = 2,
    y2line.width = 2,
    x.showspikes = TRUE,
    spike.dash = "solid",
    spike.col = NULL,
    x.spike.thickness = -2,
    tickfont.size = 16,
    x.tickmode = "auto",
    x.tickvals = NULL,
    x.ticktext = NULL,
    x.tickangle = NULL,
    legend.x = 0,
    legend.y = 1.1,
    legend.xanchor = "left",
    legend.yanchor = "top",
    legend.orientation = "h",
    margin = list(l = 75, r = 75, b = 75, t = 75),
    x.standoff = 20L,
    y.standoff = 20L,
    y2.standoff = 20L,
    hovermode = "x",
    displayModeBar = TRUE,
    modeBar.file.format = "svg",
    scrollZoom = TRUE,
    filename = NULL,
    file.width = 960,
    file.height = 500,
    file.scale = 1,
    ...
  )
```

Arguments

x	Datetime vector or list of vectors OR object of class xt. If xt, x2, y, y2, xunits, yunits, and y2units will be extracted from the object and the corresponding arguments will be ignored.
y	Numeric vector or named list of vectors: y-axis data.
x2	Datetime vector or list of vectors, optional: must be provided if y2 does not correspond to values in x. A single x-axis will be drawn for all values in x and x2.
y2	Numeric vector, optional: If provided, a second y-axis will be added to the right side of the plot
which.xy	Integer vector: Indices of x and y to plot. If not provided, will select up to the first two x-y traces.

which.xy2 Integer vector: Indices of x2 and y2 to plot. If not provided, will select up to the first two x2-y2 traces.

shade.bin Integer vector 0, 1: Time points in x to shade on the plot. For example, if there are 10 time points in x, and you want to shade time points 3 to 7, shade.bin = c(0, 0, 1, 1, 1, 1, 1, 0, 0, 0). Only set shade.bin or shade.interval, not both.

shade.interval List of numeric vectors: Intervals to shade on the plot. Only set shade.bin or shade.interval, not both.

shade.col Color: Color to shade intervals.

shade.x Numeric vector: x-values to use for shading.

shade.name Character: Name for shaded intervals.

shade.showlegend
Logical: If TRUE, show legend for shaded intervals.

ynames Character vector, optional: Names for each vector in y.

y2names Character vector, optional: Names for each vector in y2.

xlab Character: x-axis label.

ylab Character: y-axis label.

y2lab Character: y2-axis label.

xunits Character: x-axis units.

yunits Character: y-axis units.

y2units Character: y2-axis units.

yunits.col Color for y-axis units.

y2units.col Color for y2-axis units.

zt Numeric vector: Zeitgeber time. If provided, will be shown on the x-axis instead of x. To be used only with a single x vector and no x2.

show.zt Logical: If TRUE, show zt on x-axis, if zt is provided.

show.zt.every Integer: Show zt every show.zt.every ticks. If NULL, will be calculated to be x.nticks +/- 1 if x.nticks is not 0, otherwise 12 +/- 1.

zt.nticks Integer: Number of zt ticks to show. Only used if show.zt.every is NULL. The actual number of ticks shown will depend on the periodicity of zt, so that zt = 0 is always included.

main Character: Main title.

main.y Numeric: Y position of main title.

main.yanchor Character: "top", "middle", "bottom".

x.nticks Integer: Number of ticks on x-axis.

y.nticks Integer: Number of ticks on y-axis.

show.rangeslider
Logical: If TRUE, show a range slider.

slider.start Numeric: Start of range slider.

slider.end Numeric: End of range slider.

theme	Character or list: Name of theme or list of plot parameters.
palette	Color list: will be used to draw each vector in y and y2, in order.
font.size	Numeric: Font size for text.
yfill	Character: Fill type for y-axis: "none", "tozeroy", "tonexty"
y2fill	Character: Fill type for y2-axis: "none", "tozeroy", "tonexty"
fill.alpha	Numeric: Fill opacity for y-axis.
yline.width	Numeric: Line width for y-axis lines.
y2line.width	Numeric: Line width for y2-axis lines.
x.showspikes	Logical: If TRUE, show spikes on x-axis.
spike.dash	Character: Dash type for spikes: "solid", "dot", "dash", "longdash", "dashdot", "longdashdot".
spike.col	Color for spikes.
x.spike.thickness	Numeric: Thickness of spikes. -2 avoids drawing border around spikes.
tickfont.size	Numeric: Font size for tick labels.
x.tickmode	Character: "auto", "linear", "array".
x.tickvals	Numeric vector: Tick positions.
x.ticktext	Character vector: Tick labels.
x.tickangle	Numeric: Angle of tick labels.
legend.x	Numeric: X position of legend.
legend.y	Numeric: Y position of legend.
legend.xanchor	Character: "left", "center", "right".
legend.yanchor	Character: "top", "middle", "bottom".
legend.orientation	Character: "v" for vertical, "h" for horizontal.
margin	Named list with 4 numeric values: "l", "r", "t", "b" for left, right, top, bottom margins.
x.standoff	Numeric: Distance from x-axis to x-axis label.
y.standoff	Numeric: Distance from y-axis to y-axis label.
y2.standoff	Numeric: Distance from y2-axis to y2-axis label.
hovermode	Character: "closest", "x", "x unified"
displayModeBar	Logical: If TRUE, display plotly mode bar.
modeBar.file.format	Character: "png", "svg", "jpeg", "webp", "pdf": file format for mode bar image export.
scrollZoom	Logical: If TRUE, enable zooming by scrolling.
filename	Character: Path to file to save static plot.
file.width	Integer: File width in pixels for when filename is set.
file.height	Integer: File height in pixels for when filename is set.
file.scale	Numeric: If saving to file, scale plot by this number
...	Additional theme arguments.

Details

We are switching to palette being a color vector instead of the name of a built-in palette.

Value

A plotly object

Author(s)

EDG

---

dplot3_xy *Interactive Scatter Plots*

Description

Draw interactive scatter plots using plotly

Usage

```
dplot3_xy(
  x,
  y = NULL,
  fit = NULL,
  se.fit = FALSE,
  se.times = 1.96,
  include.fit.name = TRUE,
  cluster = NULL,
  cluster.params = list(k = 2),
  group = NULL,
  formula = NULL,
  rsq = TRUE,
  mode = "markers",
  order.on.x = NULL,
  main = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  col = NULL,
  alpha = NULL,
  theme = rtTheme,
  palette = rtPalette,
  axes.square = FALSE,
  group.names = NULL,
  font.size = 16,
  marker.col = NULL,
  marker.size = 8,
```

```
symbol = "circle",
fit.col = NULL,
fit.alpha = 0.8,
fit.lwd = 2.5,
se.col = NULL,
se.alpha = 0.4,
scatter.type = "scatter",
show.marginal.x = FALSE,
show.marginal.y = FALSE,
marginal.x = x,
marginal.y = y,
marginal.x.y = NULL,
marginal.y.x = NULL,
marginal.col = NULL,
marginal.alpha = 0.333,
marginal.size = 10,
legend = NULL,
legend.xy = c(0, 0.98),
legend.xanchor = "left",
legend.yanchor = "auto",
legend.orientation = "v",
legend.col = NULL,
legend.bg = "#FFFFFF00",
legend.border.col = "#FFFFFF00",
legend.borderwidth = 0,
legend.group.gap = 0,
x.showspikes = FALSE,
y.showspikes = FALSE,
spikedash = "solid",
spikemode = "across",
spikesnap = "hovered data",
spikecolor = NULL,
spikethickness = 1,
margin = list(b = 65, l = 65, t = 50, r = 10, pad = 0),
main.y = 1,
main.yanchor = "bottom",
subtitle.x = 0.02,
subtitle.y = 0.99,
subtitle.xref = "paper",
subtitle.yref = "paper",
subtitle.xanchor = "left",
subtitle.yanchor = "top",
automargin.x = TRUE,
automargin.y = TRUE,
xlim = NULL,
ylim = NULL,
axes.equal = FALSE,
diagonal = FALSE,
```

```
    diagonal.col = NULL,
    diagonal.alpha = 0.2,
    fit.params = list(),
    vline = NULL,
    vline.col = theme$fg,
    vline.width = 1,
    vline.dash = "dot",
    hline = NULL,
    hline.col = theme$fg,
    hline.width = 1,
    hline.dash = "dot",
    hovertext = NULL,
    width = NULL,
    height = NULL,
    displayModeBar = TRUE,
    modeBar.file.format = "svg",
    scrollZoom = TRUE,
    filename = NULL,
    file.width = 500,
    file.height = 500,
    file.scale = 1,
    trace = 0,
    ...
)
```

Arguments

- `x` — Numeric, vector/data.frame/list: x-axis data. If `y` is NULL and `NCOL(x)` > 1, first two columns used as `x` and `y`, respectively
- `y` — Numeric, vector/data.frame/list: y-axis data
- `fit` — Character: **rtemis** model to calculate `y ~ x` fit. Options: see `select_learn()` Can also be Logical, which will give a GAM fit if TRUE. If you specify "NLA", the activation function should be passed as a string.
- `se.fit` — Logical: If TRUE, draw the standard error of the fit
- `se.times` — Draw polygon or lines at +/- `se.times` * standard error of fit. Defaults to 1.96, which corresponds to 95 percent confidence interval
- `include.fit.name` — Logical: If TRUE, include fit name in legend.
- `cluster` — Character: Clusterer name. Will cluster `data.frame(x, y)` and pass result to `group`. Run select_clust for options
- `cluster.params` — List: Names list of parameters to pass to the `cluster` function
- `group` — Vector: Will be converted to factor; levels define group members. Default = NULL
- `formula` — Formula: Provide a formula to be solved using s_NLS. If provided, `fit` is forced to `'nls'`. e.g. y ~ b * m ^ x for a power curve. Note: `nls` is powerful but is prone to errors and warnings. Use single letters for parameter names, no numbers.

rsq	Logical: If TRUE, print R-squared values in legend if fit is set
mode	Character, vector: "markers", "lines", "markers+lines".
order.on.x	Logical: If TRUE, order x and y on x. Default = NULL, which becomes TRUE if mode includes lines.
main	Character: Main plot title.
subtitle	Character: Subtitle
xlab	Character: x-axis label.
ylab	Character: y-axis label.
col	Color for markers. Default=NULL, which will draw colors from palette
alpha	Float (0, 1]: Transparency for bar colors. Default = .8
theme	List or Character: Either the output of a theme_*() function or the name of a theme. Use themes() to get available theme names. Theme functions are of the form theme_<name>.
palette	Character: Name of **rtemis** palette to use. Default = "rtCol1". Only used if col = NULL
axes.square	Logical: If TRUE: draw a square plot to fill the graphic device. Default = FALSE. Note: If TRUE, the device size at time of call is captured and height and width are set so as to draw the largest square available. This means that resizing the device window will not automatically resize the plot.
group.names	Character, vector, length = NROW(x): Group names. Default = NULL, which uses rownames(x)
font.size	Float: Font size for all labels. Default = 16
marker.col	Color for marker
marker.size	Numeric: Marker size.
symbol	Character: Marker symbol.
fit.col	Color: Color of the fit line.
fit.alpha	Float [0, 1]: Transparency for fit line
fit.lwd	Float: Fit line width
se.col	Color for se.fit
se.alpha	Alpha for se.fit
scatter.type	Character: "scatter", "scattergl", "scatter3d", "scatterternary", "scatterpolar", "scattermapbox",
show.marginal.x	Logical: If TRUE, add marginal distribution line markers on x-axis
show.marginal.y	Logical: If TRUE, add marginal distribution line markers on y-axis
marginal.x	Numeric: Data whose distribution will be shown on x-axis. Only specify if different from x
marginal.y	Numeric: Data whose distribution will be shown on y-axis. Only specify if different from y

`marginal.x.y`	Numeric: Y position of marginal markers on x-axis
`marginal.y.x`	Numeric: X position of marginal markers on y-axis
`marginal.col`	Color for marginal markers
`marginal.alpha`	Numeric: Alpha for marginal markers
`marginal.size`	Numeric: Size of marginal markers
`legend`	Logical: If TRUE, draw legend. Default = NULL, which will be set to TRUE if there are more than 1 groups, or `fit` is set
`legend.xy`	Numeric, vector, length 2: x and y for plotly's legend
`legend.xanchor`	Character: Legend's x anchor: "left", "center", "right", "auto"
`legend.yanchor`	Character: Legend's y anchor: "top", "middle", "bottom", "auto"
`legend.orientation`	"v" or "h" for vertical or horizontal
`legend.col`	Color: Legend text color. Default = NULL, determined by theme
`legend.bg`	Color: Background color for legend
`legend.border.col`	Color: Border color for legend
`legend.borderwidth`	Numeric: Border width for legend
`legend.group.gap`	Numeric: Gap between legend groups
`x.showspikes`	Logical: If TRUE, show spikes on x-axis
`y.showspikes`	Logical: If TRUE, show spikes on y-axis
`spikedash`	Character: Dash type for spikes
`spikemode`	Character: "across", "toaxis", "marker", or any combination of those joined by +, e.g. "toaxis+across+marker"
`spikesnap`	Character: "data", "cursor", "hovered data"
`spikecolor`	Color for spikes
`spikethickness`	Numeric: Thickness of spikes
`margin`	Named list: plot margins.
`main.y`	Numeric: Y position of main title
`main.yanchor`	Character: "top", "middle", "bottom"
`subtitle.x`	Numeric: X position of subtitle relative to paper
`subtitle.y`	Numeric: Y position of subtitle relative to paper
`subtitle.xref`	Character: "paper", "x", "y"
`subtitle.yref`	Character: "paper", "x", "y"
`subtitle.xanchor`	Character: "left", "center", "right"
`subtitle.yanchor`	Character: "top", "middle", "bottom"
`automargin.x`	Logical: If TRUE, automatically set x-axis amrgins

automargin.y	Logical: If TRUE, automatically set y-axis amrgins
xlim	Float vector, length 2: x-axis limits
ylim	Float, vector, length 2: y-axis limits.
axes.equal	Logical: Should axes be equal? Defaults to FALSE
diagonal	Logical: If TRUE, draw diagonal line.
diagonal.col	Color: Color for diagonal.
diagonal.alpha	Float: Alpha for diagonal
fit.params	List: Arguments for learner defined by fit. Default = NULL, i.e. use default learner parameters
vline	Float, vector: If defined, draw a vertical line at this x value(s). Default = NULL
vline.col	Color for vline. Default = theme$fg
vline.width	Float: Width for vline. Default = 1
vline.dash	Character: Type of line to draw: "solid", "dot", "dash", "longdash", "dashdot", or "longdashdot"
hline	Float: If defined, draw a horizontal line at this y value.
hline.col	Color for hline. Default = "#ff0000" (red)
hline.width	Float: Width for hline. Default = 1
hline.dash	Character: Type of line to draw: "solid", "dot", "dash", "longdash", "dashdot", or "longdashdot"
hovertext	List of character vectors with hovertext to include for each group of markers
width	Numeric: Force plot size to this width. Default = NULL, i.e. fill available space
height	Numeric: Force plot size to this height. Default = NULL, i.e. fill available space
displayModeBar	Logical: If TRUE, show plotly's modebar
modeBar.file.format	Character: "svg", "png", "jpeg", "pdf" / any output file type supported by plotly and your system
scrollZoom	Logical: If TRUE, enable scroll zoom
filename	Character: Path to file to save static plot. Default = NULL
file.width	Integer: File width in pixels for when filename is set.
file.height	Integer: File height in pixels for when filename is set.
file.scale	Numeric: If saving to file, scale plot by this number
trace	Integer: The height the number the more diagnostic info is printed to the console
...	Additional arguments passed to theme

Details

use theme$tick.labels.col for both tick color and tick label color - this may change

Author(s)

E.D. Gennatas

Examples

```
## Not run:
dplot3_xy(iris$Sepal.Length, iris$Petal.Length,
  fit = "gam", se.fit = TRUE, group = iris$Species
)

## End(Not run)
```

---

dplot3_xyz *Interactive 3D Plots*

Description

Draw interactive 3D plots using `plotly`

Usage

```
dplot3_xyz(
  x,
  y = NULL,
  z = NULL,
  fit = NULL,
  cluster = NULL,
  cluster.params = list(k = 2),
  group = NULL,
  formula = NULL,
  rsq = TRUE,
  mode = "markers",
  order.on.x = NULL,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  zlab = NULL,
  col = NULL,
  alpha = 0.8,
  bg = NULL,
  plot.bg = NULL,
  theme = rtTheme,
  palette = rtPalette,
  axes.square = FALSE,
  group.names = NULL,
  font.size = 16,
  marker.col = NULL,
  marker.size = 8,
  fit.col = NULL,
  fit.alpha = 0.7,
```

```
  fit.lwd = 2.5,
  tick.font.size = 12,
  spike.col = NULL,
  legend = NULL,
  legend.xy = c(0, 1),
  legend.xanchor = "left",
  legend.yanchor = "auto",
  legend.orientation = "v",
  legend.col = NULL,
  legend.bg = "#FFFFFF00",
  legend.border.col = "#FFFFFF00",
  legend.borderwidth = 0,
  legend.group.gap = 0,
  margin = list(t = 30, b = 0, l = 0, r = 0),
  fit.params = list(),
  width = NULL,
  height = NULL,
  padding = 0,
  displayModeBar = TRUE,
  modeBar.file.format = "svg",
  trace = 0,
  filename = NULL,
  file.width = 500,
  file.height = 500,
  file.scale = 1,
  ...
)
```

Arguments

`x`	Numeric, vector/data.frame/list: x-axis data. If `y` is NULL and `NCOL(x) > 1`, first two columns used as `x` and `y`, respectively
`y`	Numeric, vector/data.frame/list: y-axis data
`z`	Numeric, vector/data.frame/list: z-axis data
`fit`	Character: **rtemis** model to calculate `y ~ x` fit. Options: see `select_learn()` Can also be Logical, which will give a GAM fit if TRUE. If you specify "NLA", the activation function should be passed as a string.
`cluster`	Character: Clusterer name. Will cluster `data.frame(x, y)` and pass result to `group`. Run select_clust for options
`cluster.params`	List: Names list of parameters to pass to the `cluster` function
`group`	Vector: Will be converted to factor; levels define group members. Default = NULL
`formula`	Formula: Provide a formula to be solved using s_NLS. If provided, `fit` is forced to `'nls'`. e.g. y ~ b * m ^ x for a power curve. Note: `nls` is powerful but is prone to errors and warnings. Use single letters for parameter names, no numbers.
`rsq`	Logical: If TRUE, print R-squared values in legend if `fit` is set

mode	Character, vector: "markers", "lines", "markers+lines".
order.on.x	Logical: If TRUE, order x and y on x. Default = NULL, which becomes TRUE if mode includes lines.
main	Character: Main plot title.
xlab	Character: x-axis label.
ylab	Character: y-axis label.
zlab	Character: z-axis label
col	Color for markers. Default=NULL, which will draw colors from palette
alpha	Float (0, 1]: Transparency for bar colors. Default = .8
bg	Background color
plot.bg	Plot background color
theme	List or Character: Either the output of a theme_*() function or the name of a theme. Use themes() to get available theme names. Theme functions are of the form theme_<name>.
palette	Character: Name of **rtemis** palette to use. Default = "rtCol1". Only used if col = NULL
axes.square	Logical: If TRUE: draw a square plot to fill the graphic device. Default = FALSE. Note: If TRUE, the device size at time of call is captured and height and width are set so as to draw the largest square available. This means that resizing the device window will not automatically resize the plot.
group.names	Character, vector, length = NROW(x): Group names. Default = NULL, which uses rownames(x)
font.size	Float: Font size for all labels. Default = 16
marker.col	Color for marker
marker.size	Numeric: Marker size.
fit.col	Color: Color of the fit line.
fit.alpha	Float [0, 1]: Transparency for fit line
fit.lwd	Float: Fit line width
tick.font.size	Numeric: Tick font size
spike.col	Spike lines color
legend	Logical: If TRUE, draw legend. Default = NULL, which will be set to TRUE if there are more than 1 groups, or fit is set
legend.xy	Numeric, vector, length 2: x and y for plotly's legend
legend.xanchor	Character: Legend's x anchor: "left", "center", "right", "auto"
legend.yanchor	Character: Legend's y anchor: "top", "middle", "bottom", "auto"
legend.orientation	"v" or "h" for vertical or horizontal
legend.col	Color: Legend text color. Default = NULL, determined by theme
legend.bg	Color: Background color for legend

`legend.border.col`
: Color: Border color for legend

`legend.borderwidth`
: Numeric: Border width for legend

`legend.group.gap`
: Numeric: Gap between legend groups

`margin`
: Numeric, named list: Margins for top, bottom, left, right. Default = `list(t = 30, b = 0, l = 0, r = 0`

`fit.params`
: List: Arguments for learner defined by `fit`. Default = NULL, i.e. use default learner parameters

`width`
: Numeric: Force plot size to this width. Default = NULL, i.e. fill available space

`height`
: Numeric: Force plot size to this height. Default = NULL, i.e. fill available space

`padding`
: Numeric: Graph padding.

`displayModeBar`
: Logical: If TRUE, show plotly's modebar

`modeBar.file.format`
: Character: "svg", "png", "jpeg", "pdf" / any output file type supported by plotly and your system

`trace`
: Integer: The height the number the more diagnostic info is printed to the console

`filename`
: Character: Path to file to save static plot. Default = NULL

`file.width`
: Integer: File width in pixels for when `filename` is set.

`file.height`
: Integer: File height in pixels for when `filename` is set.

`file.scale`
: Numeric: If saving to file, scale plot by this number

`...`
: Additional arguments passed to theme

Details

Note that `dplot3_xyz` uses the theme's `plot.bg` as `grid.col`

Author(s)

E.D. Gennatas

Examples

```
## Not run:
dplot3_xyz(iris, group = iris$Species, theme = "darkgrid")

## End(Not run)
```

drange *Set Dynamic Range*

Description

rtemis preproc: Adjusts the dynamic range of a vector or matrix input. By default normalizes to 0-1 range.

Usage

```
drange(x, lo = 0, hi = 1, byCol = TRUE)
```

Arguments

x	Numeric vector or matrix / data frame: Input
lo	Target range minimum. Defaults to 0
hi	Target range maximum. Defaults to 1
byCol	Logical: If TRUE: if x is matrix, drange each column separately

Author(s)

E.D. Gennatas

Examples

```
x <- runif(20, -10, 10)
x <- drange(x)
```

dt_check_unique *Check if all levels in a column are unique*

Description

Check if all levels in a column are unique

Usage

```
dt_check_unique(x, on)
```

Arguments

x	data.frame or data.table
on	Integer or character: column to check

Author(s)

E.D. Gennatas

dt_describe *Describe data.table*

Description

Describe data.table

Usage

```
dt_describe(x)
```

Arguments

x	data.table

Examples

```
## Not run:
origin <- as.POSIXct("2022-01-01 00:00:00", tz = "America/Los_Angeles")
x <- data.table(
  ID = paste0("ID", 1:10),
  V1 = rnorm(10),
  V2 = rnorm(10, 20, 3),
  V1_datetime = as.POSIXct(
    seq(
      1, 1e7,
      length.out = 10
    ),
    origin = origin
  ),
  V2_datetime = as.POSIXct(
    seq(
      1, 1e7,
      length.out = 10
    ),
    origin = origin
  ),
  C1 = sample(c("alpha", "beta", "gamma"), 10, TRUE),
  F1 = factor(sample(c("delta", "epsilon", "zeta"), 10, TRUE))
)

## End(Not run)
```

dt_get_column_attr *Tabulate column attributes*

Description

Tabulate column attributes

Usage

```
dt_get_column_attr(x, attr = "source", useNA = "always")
```

Arguments

`x`	data.table
`attr`	Character: Attribute to get
`useNA`	Character: Passed to `table`

Author(s)

E.D. Gennatas

dt_get_duplicates *Get index of duplicate values*

Description

Get index of duplicate values

Usage

```
dt_get_duplicates(x, on)
```

Arguments

`x`	data.frame or data.table
`on`	Integer or character: column to check

Author(s)

E.D. Gennatas

dt_get_factor_levels *Get factor levels from data.table*

Description

Get factor levels from data.table

Usage

```
dt_get_factor_levels(dat)
```

Arguments

dat	data.table Note: If dat is not a data.table, it will be converted in place, i.e. the original object will be modified.

Value

Named list of factor levels. Names correspond to column names.

dt_index_attr *Index columns by attribute name & value*

Description

Index columns by attribute name & value

Usage

```
dt_index_attr(x, name, value)
```

Arguments

x	data.frame or compatible
name	Character: Name of attribute
value	Character: Value of attribute

Author(s)

E.D. Gennatas

dt_inspect_type *Inspect column types*

Description

Will attempt to identify columns that should be numeric but are either character or factor by running inspect_type on each column.

Usage

```
dt_inspect_type(x, cols = NULL, verbose = TRUE)
```

Arguments

`x`	data.table
`cols`	Character vector: columns to inspect.
`verbose`	Logical: If TRUE, print messages to console.

Author(s)

E.D. Gennatas

dt_keybin_reshape *Long to wide key-value reshaping*

Description

Reshape a long format `data.table` using key-value pairs with `data.table::dcast`

Usage

```
dt_keybin_reshape(
  x,
  id_name,
  key_name,
  positive = 1,
  negative = 0,
  xname = NULL,
  verbose = TRUE
)
```

Arguments

x	A data.table object
id_name	Character: Name of column in x that defines the IDs identifying individual rows
key_name	Character: Name of column in x that holds the key
positive	Numeric or Character: Used to fill id ~ key combination present in the long format input x
negative	Numeric or Character: Used to fill id ~ key combination NOT present in the long format input x
xname	Character: Name of x to be used in messages
verbose	Logical: If TRUE, print messages to the console

Author(s)

E.D. Gennatas

Examples

```
## Not run:
x <- data.table(
  ID = rep(1:3, each = 2),
  Dx = c("A", "C", "B", "C", "D", "A")
)
dt_keybin_reshape(x, id_name = "ID", key_name = "Dx")

## End(Not run)
```

---

dt_merge — *Merge data.tables*

Description

Merge data.tables

Usage

```
dt_merge(
  left,
  right,
  on = NULL,
  left_on = NULL,
  right_on = NULL,
  how = "left",
  left_name = NULL,
  right_name = NULL,
  left_suffix = NULL,
  right_suffix = NULL,
```

```
    verbose = TRUE,
    ...
)
```

Arguments

left	data.table
right	data.table
on	Character: Name of column to join on
left_on	Character: Name of column on left table
right_on	Character: Name of column on right table
how	Character: Type of join: "inner", "left", "right", "outer".
left_name	Character: Name of left table
right_name	Character: Name of right table
left_suffix	Character: If provided, add this suffix to all left column names, excluding on/left_on
right_suffix	Character: If provided, add this suffix to all right column names, excluding on/right_on
verbose	Logical: If TRUE, print messages to console
...	Additional arguments to be passed to `data.table::merge`

Author(s)

E.D. Gennatas

---

dt_names_by_attr *List column names by attribute*

Description

List column names by attribute

Usage

```
dt_names_by_attr(x, which, exact = TRUE, sorted = TRUE)
```

Arguments

x	data.table
which	Character: name of attribute
exact	Logical: If TRUE, use exact matching
sorted	Logical: If TRUE, sort the output

Author(s)

E.D. Gennatas

dt_names_by_class *List column names by class*

Description

List column names by class

Usage

```
dt_names_by_class(x, sorted = TRUE, item.format = hilite, maxlength = 24)
```

Arguments

x	data.table
sorted	Logical: If TRUE, sort the output
item.format	Function: Function to format each item
maxlength	Integer: Maximum number of items to print

Author(s)

E.D. Gennatas

dt_Nuniqueperfeat *Number of unique values per feature*

Description

Number of unique values per feature

Usage

```
dt_Nuniqueperfeat(x, excludeNA = FALSE)
```

Arguments

x	data.table
excludeNA	Logical: If TRUE, exclude NA values

Author(s)

E.D. Gennatas

`dt_pctmatch` *Get N and percent match of values between two columns of two data.tables*

Description

Get N and percent match of values between two columns of two data.tables

Usage

```
dt_pctmatch(x, y, on = NULL, left_on = NULL, right_on = NULL, verbose = TRUE)
```

Arguments

x	data.table
y	data.table
on	Integer or character: column to read in x and y, if it is the same
left_on	Integer or character: column to read in x
right_on	Integer or character: column to read in y
verbose	Logical: If TRUE, print messages to console

Author(s)

E.D. Gennatas

`dt_pctmissing` *Get percent of missing values from every column*

Description

Get percent of missing values from every column

Usage

```
dt_pctmissing(x, verbose = TRUE)
```

Arguments

x	data.frame or data.table
verbose	Logical: If TRUE, print messages to console

Author(s)

E.D. Gennatas

dt_set_autotypes *Set column types automatically*

Description

This function inspects a data.table and attempts to identify columns that should be numeric but have been read in as character, because one or more fields contain non-numeric characters

Usage

```
dt_set_autotypes(x, cols = NULL, verbose = TRUE)
```

Arguments

`x`	data.table
`cols`	Character vector: columns to work on. If not defined, will work on all columns
`verbose`	Logical: If TRUE, print messages to console

Author(s)

E.D. Gennatas

dt_set_cleanfactorlevels *Clean factor levels of data.table in-place*

Description

Finds all factors in a data.table and cleans factor levels to include only underscore symbols

Usage

```
dt_set_cleanfactorlevels(x, prefix_digits = NA)
```

Arguments

`x`	data.table
`prefix_digits`	Character: If not NA, add this prefix to all factor levels that are numbers

Author(s)

E.D. Gennatas

Examples

```
## Not run:
x <- as.data.table(iris)
levels(x$Species) <- c("setosa:iris", "versicolor$iris", "virginica iris")
dt_set_cleanfactorlevels(x)
x

## End(Not run)
```

`dt_set_clean_all` *Clean column names and factor levels in-place*

Description

Clean column names and factor levels in-place

Usage

```
dt_set_clean_all(x, prefix_digits = NA)
```

Arguments

`x`	data.table
`prefix_digits`	Character: prefix to add to names beginning with a digit. Set to NA to skip Note: If x is not a data.table, it will be converted in place, i.e. the original object will be modified.

Author(s)

E.D. Gennatas

`dt_set_logical2factor` *Convert data.table logical columns to factor with custom labels in-place*

Description

Convert data.table logical columns to factor with custom labels in-place

Usage

```
dt_set_logical2factor(
  x,
  cols = NULL,
  labels = c("False", "True"),
  maintain_attributes = TRUE,
  fillNA = NULL
)
```

Arguments

- `x` data.table
- `cols` Integer or character: columns to convert, if NULL, operates on all logical columns
- `labels` Character: labels for factor levels
- `maintain_attributes` Logical: If TRUE, maintain column attributes
- `fillNA` Character: If not NULL, fill NA values with this constant

Author(s)

E.D. Gennatas

Examples

```
## Not run:
library(data.table)
x <- data.table(a = 1:5, b = c(T, F, F, F, T))
x
dt_set_logical2factor(x)
x
z <- data.table(alpha = 1:5, beta = c(T, F, T, NA, T), gamma = c(F, F, T, F, NA))
z
# You can usee fillNA to fill NA values with a constant
dt_set_logical2factor(z, cols = "beta", labels = c("No", "Yes"), fillNA = "No")
z
w <- data.table(mango = 1:5, banana = c(F, F, T, T, F))
w
dt_set_logical2factor(w, cols = 2, labels = c("Ugh", "Huh"))
w
# Column attributes are maintained by default:
z <- data.table(alpha = 1:5, beta = c(T, F, T, NA, T), gamma = c(F, F, T, F, NA))
for (i in seq_along(z)) setattr(z[[i]], "source", "Guava")
str(z)
dt_set_logical2factor(z, cols = "beta", labels = c("No", "Yes"))
str(z)

## End(Not run)
```

---

d_H2OAE *Autoencoder using H2O*

Description

Train an Autoencoder using `h2o::h2o.deeplearning` Check out the H2O Flow at `[ip]:[port]`, Default IP:port is "localhost:54321" e.g. if running on localhost, point your web browser to `localhost:54321`

Usage

```
d_H2OAE(
  x,
  x.test = NULL,
  x.valid = NULL,
  ip = "localhost",
  port = 54321,
  n.hidden.nodes = c(ncol(x), 3, ncol(x)),
  extract.layer = ceiling(length(n.hidden.nodes)/2),
  epochs = 5000,
  activation = "Tanh",
  loss = "Automatic",
  input.dropout.ratio = 0,
  hidden.dropout.ratios = rep(0, length(n.hidden.nodes)),
  learning.rate = 0.005,
  learning.rate.annealing = 1e-06,
  l1 = 0,
  l2 = 0,
  stopping.rounds = 50,
  stopping.metric = "AUTO",
  scale = TRUE,
  center = TRUE,
  n.cores = rtCores,
  verbose = TRUE,
  save.mod = FALSE,
  outdir = NULL,
  ...
)
```

Arguments

`x`	Vector / Matrix / Data Frame: Training set Predictors
`x.test`	Vector / Matrix / Data Frame: Testing set Predictors
`x.valid`	Vector / Matrix / Data Frame: Validation set Predictors
`ip`	Character: IP address of H2O server. Default = "localhost"
`port`	Integer: Port number for server. Default = 54321
`n.hidden.nodes`	Integer vector of length equal to the number of hidden layers you wish to create
`extract.layer`	Integer: Which layer to extract. For regular autoencoder, this is the middle layer. Default = `ceiling(length(n.hidden.nodes)/2)`
`epochs`	Integer: How many times to iterate through the dataset. Default = 5000
`activation`	Character: Activation function to use: "Tanh" (Default), "TanhWithDropout", "Rectifier", "RectifierWithDropout", "Maxout", "MaxoutWithDropout"
`loss`	Character: "Automatic" (Default), "CrossEntropy", "Quadratic", "Huber", "Absolute"
`input.dropout.ratio`	Float (0, 1): Dropout ratio for inputs

hidden.dropout.ratios
: Vector, Float (0, 2): Dropout ratios for hidden layers

learning.rate
: Float: Learning rate. Default = .005

learning.rate.annealing
: Float: Learning rate annealing. Default = 1e-06

l1
: Float (0, 1): L1 regularization (introduces sparseness; i.e. sets many weights to 0; reduces variance, increases generalizability)

l2
: Float (0, 1): L2 regularization (prevents very large absolute weights; reduces variance, increases generalizability)

stopping.rounds
: Integer: Stop if simple moving average of length stopping.rounds of the stopping.metric does not improve. Set to 0 to disable. Default = 50

stopping.metric
: Character: Stopping metric to use: "AUTO", "deviance", "logloss", "MSE", "RMSE", "MAE", "RMSLE", "AUC", "lift_top_group", "misclassification", "mean_per_class_error". Default = "AUTO" ("logloss" for Classification, "deviance" for Regression)

scale
: Logical: If TRUE, scale input before training autoencoder. Default = TRUE

center
: Logical: If TRUE, center input before training autoencoder. Default = TRUE

n.cores
: Integer: Number of cores to use

verbose
: Logical: If TRUE, print summary to screen.

save.mod
: Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)

outdir
: Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE

...
: Additional arguments to pass to h2p::h2o.deeplearning

Value

rtDecom object

Author(s)

E.D. Gennatas

See Also

decom

Other Decomposition: d_H2OGLRM(), d_ICA(), d_Isomap(), d_KPCA(), d_LLE(), d_MDS(), d_NMF(), d_PCA(), d_SPCA(), d_SVD(), d_TSNE(), d_UMAP()

Other Deep Learning: s_H2ODL(), s_TFN()

d_H2OGLRM *Generalized Low-Rank Models (GLRM) on H2O*

Description

Perform GLRM decomposition using `h2o::h2o.glrm` Given Input matrix `A`: `A(m x n) = X(m x k) \%*\% Y(k x n)`

Usage

```
d_H2OGLRM(
  x,
  x.test = NULL,
  x.valid = NULL,
  k = 3,
  ip = "localhost",
  port = 54321,
  transform = "NONE",
  loss = "Quadratic",
  regularization.x = "None",
  regularization.y = "None",
  gamma.x = 0,
  gamma.y = 0,
  max_iterations = 1000,
  max_updates = 2 * max_iterations,
  init_step_size = 1,
  min_step_size = 1e-04,
  seed = -1,
  init = "PlusPlus",
  svd.method = "Randomized",
  verbose = TRUE,
  print.plot = TRUE,
  plot.theme = rtTheme,
  n.cores = rtCores,
  ...
)
```

Arguments

`x`	Input data
`x.test`	Optional test set. Will be projected on to NMF basis
`x.valid`	Optional validation set
`k`	Integer: Rank of decomposition
`ip`	Character: IP address of H2O server. Default = "localhost"
`port`	Integer: Port number for server. Default = 54321
`transform`	Character: Transformation of input prior to decomposition

loss	Character: Numeric loss function: "Quadratic", "Absolute", "Huber", "Poisson", "Hinge", "Logistic", "Periodic". Default = "Quadratic"
regularization.x	Character: Regularization function for X matrix: "None", "Quadratic", "L2", "L1", "NonNegative", "OneSparse", "UnitOneSparse", "Simplex". Default = "None"
regularization.y	Character: Regularization function for Y matrix: "None", "Quadratic", "L2", "L1", "NonNegative", "OneSparse", "UnitOneSparse", "Simplex". Default = "None"
gamma.x	Float: Regularization weight on X matrix. Default = 0
gamma.y	Float: Regularization weight on Y matrix. Default = 0
max_iterations	Integer: Maximum number of iterations. Default = 1000
max_updates	Integer: Maximum number of iterations. Default = 2 * max_iterations
init_step_size	Float: Initial step size. Default = 1
min_step_size	Float: Minimum step size. Default = .0001
seed	Integer: Seed for random number generator. Default = -1 (time-based)
init	Character: Initialization mode: "Random", "SVD", "PlusPlus", "User". Default = "PlusPlus"
svd.method	Character: SVD method for initialization: "GramSVD", "Power", "Randomized". Default = "Randomized"
verbose	Logical: If TRUE, print console messages
print.plot	Logical: If TRUE, print objective score against iteration number
plot.theme	Character: Theme to pass to mplot3_xy if print.plot = TRUE
n.cores	Integer: Number of cores to use
...	Additional parameters to be passed to h2o::h2o.glrm

Details

Learn more about GLRM from the H2O tutorial https://github.com/h2oai/h2o-tutorials/blob/master/tutorials/glrm/glrm-tutorial.md

Value

rtDecom object

Author(s)

E.D. Gennatas

See Also

Other Decomposition: d_H2OAE(), d_ICA(), d_Isomap(), d_KPCA(), d_LLE(), d_MDS(), d_NMF(), d_PCA(), d_SPCA(), d_SVD(), d_TSNE(), d_UMAP()

d_ICA *Independent Component Analysis*

Description

Perform ICA decomposition using the fastICA algorithm in `fastICA::fastICA` or `ica::fastica`

Usage

```
d_ICA(
  x,
  k = 3,
  package = c("fastICA", "ica"),
  alg.type = "parallel",
  maxit = 100,
  scale = TRUE,
  center = TRUE,
  verbose = TRUE,
  trace = 0,
  ...
)
```

Arguments

`x`	Input data
`k`	Integer vector of length 1 or greater. Rank of decomposition
`package`	Character: Which package to use for ICA. "fastICA" will use `fastICA::fastICA`, "ica" will use `ica::fastica`. Default = "fastICA". Note: only `fastICA` works with k = 1
`alg.type`	Character: For package = `"fastICA"`, "parallel" or "deflation".
`maxit`	Integer: Maximum N of iterations
`scale`	Logical: If TRUE, scale input data before decomposition.
`center`	Logical: If TRUE, also center input data if `scale` is `TRUE`.
`verbose`	Logical: If TRUE, print messages to screen. Default = TRUE
`trace`	Integer: If > 0, print messages during ICA run. Default = 0
`...`	Additional parameters to be passed to `fastICA::fastICA` or `ica::icafast`

Details

Project scaled variables to ICA components. Input must be n by p, where n represents number of cases, and p represents number of features. fastICA will be applied to the transpose of the n x p matrix. fastICA will fail if there are any NA values or constant features: remove them using preprocess

Value

rtDecom object

Author(s)

E.D. Gennatas

See Also

Other Decomposition: d_H2OAE(), d_H2OGLRM(), d_Isomap(), d_KPCA(), d_LLE(), d_MDS(), d_NMF(), d_PCA(), d_SPCA(), d_SVD(), d_TSNE(), d_UMAP()

---

d_Isomap *Isomap*

Description

Perform ISOMAP decomposition using vegan::isomap

Usage

```
d_Isomap(
  x,
  k = 2,
  dist.method = "euclidean",
  nsd = 0,
  path = c("shortest", "extended"),
  center = TRUE,
  scale = TRUE,
  verbose = TRUE,
  n.cores = rtCores,
  ...
)
```

Arguments

x	Input data
k	Integer vector of length 1 or greater. Rank of decomposition
dist.method	Character: Distance calculation method. See vegan::vegdist
nsd	Integer: Number of shortest dissimilarities retained
path	Character: The path argument of vegan::isomap
center	Logical: If TRUE, center data prior to decomposition. Default = TRUE
scale	Logical: If TRUE, scale data prior to decomposition. Default = TRUE
verbose	Logical: If TRUE, print messages to output
n.cores	Integer: Number of cores to use
...	Additional parameters to be passed to vegan::isomap

Details

Project scaled variables to ISOMAP components Input must be n by p, where n represents number of cases, and p represents number of features. ISOMAP will be applied to the transpose of the n x p matrix.

Value

rtDecom object

Author(s)

E.D. Gennatas

See Also

Other Decomposition: d_H2OAE(), d_H2OGLRM(), d_ICA(), d_KPCA(), d_LLE(), d_MDS(), d_NMF(), d_PCA(), d_SPCA(), d_SVD(), d_TSNE(), d_UMAP()

---

d_KPCA — *Kernel Principal Component Analysis*

Description

Perform kernel PCA decomposition using kernlab::kpca

Usage

```
d_KPCA(
  x,
  x.test = NULL,
  k = 2,
  th = 1e-04,
  kernel = "rbfdot",
  kpar = NULL,
  center = TRUE,
  scale = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

x	Input data
x.test	Optional test set. Will be projected on to KPCA basis
k	Integer vector of length 1 or greater. N of components to return If set to 0, th determines eigenvalue below which PCs are ignored

th	Threshold for eigenvalue below which PCs are ignored if k is set to 0
kernel	Character: Type of kernel to use. See kernlab::kpca
kpar	List of hyperparameters: See kernlab::kpca("kpar")
center	Logical: If TRUE, center data prior to decomposition. Default = TRUE
scale	Logical: If TRUE, scale data prior to decomposition. Default = TRUE
verbose	Logical: If TRUE, print messages to screen. Default = TRUE
...	Additional parameters to be passed to fastKPCA::fastKPCA

Details

Project scaled variables to KPCA components. Input must be n by p, where n represents number of cases, and p represents number of features. KPCA will be applied to the transpose of the n x p matrix.

Value

rtDecom object

Author(s)

E.D. Gennatas

See Also

Other Decomposition: d_H2OAE(), d_H2OGLRM(), d_ICA(), d_Isomap(), d_LLE(), d_MDS(), d_NMF(), d_PCA(), d_SPCA(), d_SVD(), d_TSNE(), d_UMAP()

---

d_LLE *Locally Linear Embedding*

Description

Perform LLE decomposition using RDRToolbox::lle

Usage

```
d_LLE(x, k = 2, nn = 6, verbose = TRUE)
```

Arguments

x	Input data
k	Integer: dimensionality of the embedding
nn	Integer: Number of neighbors.
verbose	Logical: If TRUE, print messages to screen. Default = TRUE

Details

Project scaled variables to LLE components Input must be n by p, where n represents number of cases, and p represents number of features.

Value

`rtDecom` object

Author(s)

E.D. Gennatas

See Also

Other Decomposition: `d_H2OAE()`, `d_H2OGLRM()`, `d_ICA()`, `d_Isomap()`, `d_KPCA()`, `d_MDS()`, `d_NMF()`, `d_PCA()`, `d_SPCA()`, `d_SVD()`, `d_TSNE()`, `d_UMAP()`

`d_MDS` *Multidimensional Scaling*

Description

Perform MDS decomposition using `stats:cmdscale`

Usage

```
d_MDS(
  x,
  k = 2,
  dist.method = c("euclidean", "maximum", "manhattan", "canberra", "binary", "minkowski"),
  eig = FALSE,
  add = FALSE,
  x.ret = FALSE,
  scale = TRUE,
  center = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

`x`	Input data
`k`	Integer vector of length 1 or greater. Rank of decomposition
`dist.method`	Character: method to use to calculate distance. See `stats::dist("method")`
`eig`	Logical: If TRUE, return eigenvalues. Default = FALSE

`add`	Logical: If TRUE, an additive constant `c*` will be computed and added to the non-diagonal dissimilarities, which makes the Euclidean. Default = FALSE
`x.ret`	Logical: If TRUE, return the doubly centered symmetric distance matrix. Default = FALSE
`scale`	Logical: If TRUE, scale input data before decomposition. Default = TRUE
`center`	Logical: If TRUE, also center input data if `scale` is `TRUE`. Default = TRUE
`verbose`	Logical: If TRUE, print messages to screen. Default = TRUE
`...`	Additional parameters to be passed to `svd`

Details

Project scaled variables to MDS components. Input must be n by p, where n represents number of cases, and p represents number of features. fastMDS will be applied to the transpose of the n x p matrix. fastMDS will fail if there are any NA values or constant features: remove them using preprocess

Value

`rtDecom` object

Author(s)

E.D. Gennatas

See Also

Other Decomposition: `d_H2OAE()`, `d_H2OGLRM()`, `d_ICA()`, `d_Isomap()`, `d_KPCA()`, `d_LLE()`, `d_NMF()`, `d_PCA()`, `d_SPCA()`, `d_SVD()`, `d_TSNE()`, `d_UMAP()`

---

d_NMF *Non-negative Matrix Factorization (NMF)*

Description

Perform NMF decomposition using `NMF::nmf`

Usage

```
d_NMF(
  x,
  x.test = NULL,
  k = 2,
  method = "brunet",
  nrun = 30,
  scale = TRUE,
  center = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

x	Input data
x.test	Optional test set. Will be projected on to NMF basis
k	Integer vector of length 1 or greater. Rank of decomposition
method	NMF method. Defaults to "brunet". See NMF::nmf
nrun	Integer: Number of runs to perform
scale	Logical: If TRUE, scale input data before projecting
center	Logical: If TRUE, also center input data if scale is TRUE
verbose	Logical: If TRUE, print messages to screen. Default = TRUE
...	Additional parameters to be passed to NMF::nmf

Details

Project scaled variables to NMF bases. Input must be n by p, where n represents number of cases, and p represents number of features. NMF will be applied to the transpose of the n x p matrix.

Value

rtDecom object

Author(s)

E.D. Gennatas

See Also

Other Decomposition: d_H2OAE(), d_H2OGLRM(), d_ICA(), d_Isomap(), d_KPCA(), d_LLE(), d_MDS(), d_PCA(), d_SPCA(), d_SVD(), d_TSNE(), d_UMAP()

---

d_PCA *Principal Component Analysis*

Description

Perform PCA decomposition using stats::prcomp

Usage

```
d_PCA(
  x,
  x.test = NULL,
  k = NULL,
  scale = TRUE,
  center = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

x	Input matrix
x.test	Optional test set. Will be projected on to PCA basis
k	Integer: Number of right singular vectors to compute (svd's nv)
scale	Logical: If TRUE, scale input data before doing SVD
center	Logical: If TRUE, also center input data if scale is TRUE
verbose	Logical: If TRUE, print messages to screen. Default = TRUE
...	Additional parameters to be passed to PCA::PCA

Details

Same solution as d_SVD. d_PCA runs prcomp, which has useful summary output

Value

rtDecom object

Author(s)

E.D. Gennatas

See Also

Other Decomposition: d_H2OAE(), d_H2OGLRM(), d_ICA(), d_Isomap(), d_KPCA(), d_LLE(), d_MDS(), d_NMF(), d_SPCA(), d_SVD(), d_TSNE(), d_UMAP()

---

d_SPCA *Sparse Principal Component Analysis*

Description

Perform sparse and/or non-negative PCA or cumulative PCA decomposition using nsprcomp::nsprcomp or nsprcomp::nscumcomp respectively

Usage

```
d_SPCA(
  x,
  x.test = NULL,
  k = 1,
  nz = floor(0.5 * NCOL(x)),
  nneg = FALSE,
  gamma = 0,
  method = c("cumulative", "vanilla"),
  scale = TRUE,
```

```
  center = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

x	Input matrix
x.test	Optional test set. Will be projected on to SPCA basis
k	Integer vector of length 1 or greater. N of components to return If set to 0, th determines eigenvalue below which PCs are ignored
nz	Integer: Upper bound on non-zero loadings. See `nsprcomp::nscumcomp("k")`
nneg	Logical: If TRUE, calculate non-negative loadings only. Default = FALSE
gamma	Float (>0): Penalty on the divergence from otrhonormality of the pseudo-rotation matrix. Default = 0, i.e. no penalty. May need to increase with collinear features.
method	Character: "cumulative" or "vanilla" sparse PCA. Default = "cumulative"
scale	Logical: If TRUE, scale input data before projecting. Default = TRUE
center	Logical: If TRUE, also center input data if `scale` is `TRUE`. Default = FALSE
verbose	Logical: If TRUE, print messages to screen. Default = TRUE
...	Additional parameters to be passed to `fastSPCA::fastSPCA`

Details

Project scaled variables to sparse and/or non-negative PCA components. Input must be n by p, where n represents number of cases, and p represents number of features. SPCA will be applied to the transpose of the n x p matrix.

Value

`rtDecom` object

Author(s)

E.D. Gennatas

See Also

Other Decomposition: `d_H2OAE()`, `d_H2OGLRM()`, `d_ICA()`, `d_Isomap()`, `d_KPCA()`, `d_LLE()`, `d_MDS()`, `d_NMF()`, `d_PCA()`, `d_SVD()`, `d_TSNE()`, `d_UMAP()`

d_SVD *Singular Value Decomposition*

Description

Perform SVD decomposition using `base::svd`

Usage

```
d_SVD(
  x,
  x.test = NULL,
  k = 2,
  nu = 0,
  scale = TRUE,
  center = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

`x`	Input matrix
`x.test`	Optional test set matrix. Will be projected on to SVD bases
`k`	Integer: Number of right singular vectors to compute (`svd`'s `nv`)
`nu`	Integer: Number of left singular vectors to compute
`scale`	Logical: If TRUE, scale input data before doing SVD. Default = TRUE
`center`	Logical: If TRUE, also center input data if `scale` is `TRUE`. Default = TRUE
`verbose`	Logical: If TRUE, print messages to screen. Default = TRUE
`...`	Additional parameters to be passed to `svd`

Details

Same solution as d_PCA

Value

`rtDecom` object

Author(s)

E.D. Gennatas

See Also

Other Decomposition: `d_H2OAE()`, `d_H2OGLRM()`, `d_ICA()`, `d_Isomap()`, `d_KPCA()`, `d_LLE()`, `d_MDS()`, `d_NMF()`, `d_PCA()`, `d_SPCA()`, `d_TSNE()`, `d_UMAP()`

d_TSNE *t-distributed Stochastic Neighbor Embedding*

Description

Perform t-SNE decomposition using Rtsne::Rtsne

Usage

```
d_TSNE(
  x,
  k = 3,
  initial.dims = 50,
  perplexity = 15,
  theta = 0,
  check.duplicates = TRUE,
  pca = TRUE,
  max.iter = 1000,
  scale = FALSE,
  center = FALSE,
  is.distance = FALSE,
  verbose = TRUE,
  outdir = "./",
  ...
)
```

Arguments

x	Input matrix
k	Integer. Number of t-SNE components required
initial.dims	Integer: Number of dimensions to retain in initial PCA. Default = 50
perplexity	Numeric: Perplexity parameter
theta	Float: 0.0: exact TSNE. Increase for higher speed, lower accuracy. Default = 0
check.duplicates	Logical: If TRUE, Checks whether duplicates are present. Best to set test manually
pca	Logical: If TRUE, perform initial PCA step. Default = TRUE
max.iter	Integer: Maximum number of iterations. Default = 1000
scale	Logical: If TRUE, scale before running t-SNE using base::scale. Default = FALSE
center	Logical: If TRUE, and scale = TRUE, also center. Default = FALSE
is.distance	Logical: If TRUE, x should be a distance matrix. Default = FALSE
verbose	Logical: If TRUE, print messages to output
outdir	Path to output directory
...	Options for Rtsne::Rtsne

Value

rtDecom object

Author(s)

E.D. Gennatas

See Also

Other Decomposition: d_H2OAE(), d_H2OGLRM(), d_ICA(), d_Isomap(), d_KPCA(), d_LLE(), d_MDS(), d_NMF(), d_PCA(), d_SPCA(), d_SVD(), d_UMAP()

---

d_UMAP *Uniform Manifold Approximation and Projection (UMAP)*

Description

Perform UMAP decomposition using uwot::umap

Usage

```
d_UMAP(
  x,
  x.test = NULL,
  k = 2,
  n.neighbors = 15,
  init = "spectral",
  metric = c("euclidean", "cosine", "manhattan", "hamming", "categorical"),
  epochs = NULL,
  learning.rate = 1,
  scale = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

x	Input matrix
x.test	Optional test set matrix. Will be projected on to UMAP bases
k	Integer: Number of projections
n.neighbors	Integer: Number of keighbors
init	Character: Initialization type. See uwot::umap "init"
metric	Character: Distance metric to use: "euclidean", "cosine", "manhattan", "hamming", "categorical". Default = "euclidean"
epochs	Integer: Number of epochs

learning.rate	Float: Learning rate. Default = 1
scale	Logical: If TRUE, scale input data before doing UMAP. Default = TRUE
verbose	Logical: If TRUE, print messages to screen. Default = TRUE
...	Additional parameters to be passed to uwot::umap

Details

Updated 2023-12-09: See GitHub issue and related comment

Value

rtDecom object

Author(s)

E.D. Gennatas

See Also

Other Decomposition: d_H2OAE(), d_H2OGLRM(), d_ICA(), d_Isomap(), d_KPCA(), d_LLE(), d_MDS(), d_NMF(), d_PCA(), d_SPCA(), d_SVD(), d_TSNE()

---

earlystop *Early stopping*

Description

Check loss vector for early stopping criteria: - either total percent decrease from starting error (e.g. if predictions started at expectation) - or minimum percent decrease (relative to the first value of the vector) over a window of last n steps

Usage

```
earlystop(
  x,
  window = 10,
  window_decrease_pct_min = 0.01,
  total_decrease_pct_max = NULL,
  verbose = TRUE
)
```

Arguments

- `x` Numeric vector: loss at each iteration
- `window` Integer: Number of steps to consider
- `window_decrease_pct_min` Float: Stop if improvement is less than this percent over last `window` steps
- `total_decrease_pct_max` Float: Stop if improvement from first to last step exceeds this percent. If defined, overrides `window_decrease_pct_min`
- `verbose` Logical: If TRUE, print messages to console. Default = TRUE

Details

If the first loss value was set to be the loss when yhat = mean(y) (e.g. in boosting), then `total_decrease_pct_max` corresponds to R-squared and `window_decrease_pct_min` to percent R-squared improvement over `window` last steps.

Author(s)

E.D. Gennatas

`expand.boost` *Expand boosting series*

Description

Expand a boost object by adding more iterations

Usage

```
expand.boost(
  object,
  x,
  y = NULL,
  x.valid = NULL,
  y.valid = NULL,
  x.test = NULL,
  y.test = NULL,
  mod = NULL,
  resid = NULL,
  mod.params = NULL,
  max.iter = 10,
  learning.rate = NULL,
  case.p = 1,
  prefix = NULL,
  verbose = TRUE,
  trace = 0,
```

```
  print.error.plot = "final",
  print.plot = FALSE
)
```

Arguments

object	boost object
x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.valid	Data.frame; optional: Validation data
y.valid	Float, vector; optional: Validation outcome
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
mod	Character: Algorithm to train base learners, for options, see select_learn. Default = "cart"
resid	Float, vector, length = length(y): Residuals to work on. Do not change unless you know what you're doing. Default = NULL, for regular boosting
mod.params	Named list of arguments for mod
max.iter	Integer: Maximum number of iterations (additive steps) to perform. Default = 10
learning.rate	Float (0, 1] Learning rate for the additive steps
case.p	Float (0, 1]: Train each iteration using this perceent of cases. Default = 1, i.e. use all cases
prefix	Internal
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If > 0, print diagnostic info to console
print.error.plot	String or Integer: "final" plots a training and validation (if available) error curve at the end of training. If integer, plot training and validation error curve every this many iterations during training. "none" for no plot.
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted.

Author(s)

E.D. Gennatas

`explain` *Explain individual-level model predictions*

Description

Explain individual-level model predictions

Usage

```
explain(mod, x, digits = 2, top = NULL, trace = 0)
```

Arguments

`mod`	`rtMod` object.
`x`	Single-row data.frame of predictors.
`digits`	Integer: Number of digits to round coefficients to.
`top`	Integer: Number of top rules to show by absolute coefficient.
`trace`	Integer: If > 0, print more messages to output.

Author(s)

ED Gennatas

`f1` *F1 score*

Description

Calculate the F1 score for classification:

Usage

```
f1(precision, recall)
```

Arguments

`precision`	Float [0, 1]: Precision a.k.a. Positive Predictive Value
`recall`	Float [0, 1]: Recall a.k.a. Sensitivity

Details

$$F1 = 2 \frac{Recall \cdot Precision}{Recall + Precision}$$

Author(s)

E.D. Gennatas

factoryze *Factor Analysis*

Description

Perform parallel analysis, factor analysis, bifactor analysis and hierarchical clustering

Usage

```
factoryze(
  x,
  n.factors = NULL,
  method = "minres",
  rotation = "oblimin",
  scores = "regression",
  cor = "cor",
  fa.n.iter = 100,
  omega.method = "minres",
  omega.rotation = c("oblimin", "simplimax", "promax", "cluster", "target"),
  omega.n.iter = 1,
  x.name = NULL,
  print.plot = TRUE,
  do.pa = TRUE,
  do.fa = TRUE,
  do.bifactor = TRUE,
  do.hclust = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

x	Data. Will be coerced to data frame
n.factors	Integer: If NULL, will be estimated using parallel analysis
method	Character: Factor analysis method: "minres": minimum residual (OLS), "wls": weighted least squares (WLS); "gls": generalized weighted least squares (GLS); "pa": principal factor solution; "ml": maximum likelihood; "minchi": minimize the sample size weighted chi square when treating pairwise correlations with different number of subjects per pair; "minrank": minimum rank factor analysis. Default = "minres"
rotation	Character: Rotation methods. No rotation: "none"; Orthogonal: "varimax", "quartimax", "bentlerT", "equamax", "varimin", "geominT", "bifactor"; Oblique: "promax", "oblimin", "simplimax", "bentlerQ, "geominQ", "biquartimin", "cluster". Default = "oblimin"

scores	Character: Factor score estimation method. Options: "regression", "Thurstone": simple regression, "tenBerge": correlation-preserving, "Anderson", "Barlett". Default = "regression"
cor	Character: Correlation method: "cor": Pearson correlation, "cov": Covariance, "tet": tetrachoric, "poly": polychoric, "mixed": mixed cor for a mixture of tetrachorics, polychorics, Pearsons, biserials, and polyserials, "Yuleb": Yulebonett, "Yuleq" and "YuleY": Yule coefficients
fa.n.iter	Integer: Number of iterations for factor analysis. Default = 100
omega.method	Character: Factor analysis method for the bifactor analysis. Same options as `method` Default = "minres"
omega.rotation	Character: Rotation method for bifactor analysis: "oblimin", "simplimax", "promax", "cluster", "target". Default = "oblimin"
omega.n.iter	Integer: Number of iterations for bifactor analysis. Default = 1
x.name	Character: Name your dataset. Used for plotting
print.plot	Logical: If TRUE, print plots along the way. Default = TRUE
do.pa	Logical: If TRUE, perform parallel analysis. Default = TRUE
do.fa	Logical: If TRUE, perform factor analysis. Default = TRUE
do.bifactor	Logical: If TRUE, perform bifactor analysis. Default = TRUE
do.hclust	Logical: If TRUE, perform hierarchical cluster analysis. Default = TRUE
verbose	Logical: If TRUE, print messages to output. Default = TRUE
...	Additional arguments to pass to `psych::fa`

Details

Consult `psych::fa` for more information on the parameters

Author(s)

E.D. Gennatas

`factor_harmonize` *Factor harmonize*

Description

Factor harmonize

Usage

```
factor_harmonize(reference, x, verbosity = 1)
```

Arguments

reference	Reference factor
x	Input factor
verbosity	Integer: If > 0, print messages to console.

factor_NA2missing *Factor NA to "missing" level*

Description

Set NA values of a factor vector to a new level indicating missingness

Usage

```
factor_NA2missing(x, na_level_name = "missing")
```

Arguments

x	Factor
na_level_name	Character: Name of new level to create that will be assigned to all current NA values. Default = "missing"

Author(s)

E.D. Gennatas

Examples

```
x <- factor(sample(letters[1:3], 100, TRUE))
x[sample(1:100, 10)] <- NA
xm <- factor_NA2missing(x)
```

fct_describe *Decribe factor*

Description

Outputs a single character with names and counts of each level of the input factor

Usage

```
fct_describe(x, max_n = 5, return_ordered = TRUE)
```

Arguments

x	factor
max_n	Integer: Return counts for up to this many levels
return_ordered	Logical: If TRUE, return levels ordered by count, otherwise return in level order

Value

Character with level counts

Author(s)

E.D. Gennatas

Examples

```
## Not run:
# Small number of levels
fct_describe(iris$Species)

# Large number of levels: show top n by count
x <- factor(sample(letters, 1000, TRUE))
fct_describe(x)
fct_describe(x, 3)

## End(Not run)
```

`format.call` *Format method for* `call` *objects*

Description

Format method for `call` objects

Usage

```
## S3 method for class 'call'
format(x, as.html = FALSE, class = "rtcode", ...)
```

Arguments

`x`	`call` object
`as.html`	Logical: If TRUE, output HTML span element
`class`	Character: CSS class to assign to span containing code
`...`	Not used

Author(s)

E.D. Gennatas

Examples

```
## Not run:
irmod <- elevate(iris,
  mod = "cart",
  maxdepth = 2:3,
  n.resamples = 9,
  train.p = .85
)
format(irmod$call) |> cat()

## End(Not run)
```

`formatLightRules` *Format LightRuleFit rules*

Description

Converts R-executable logical expressions to a more human-friendly format

Usage

```
formatLightRules(x, space.after.comma = FALSE, decimal.places = NULL)
```

Arguments

`x`	Vector, string: Logical expressions
`space.after.comma`	Logical: If TRUE, place spaces after commas. Default = false
`decimal.places`	Integer: Limit all floats (numbers of the form 9.9) to this many decimal places

Author(s)

E.D. Gennatas

`formatRules` *Format rules*

Description

Converts R-executable logical expressions to a more human-friendly format

Usage

```
formatRules(x, space.after.comma = FALSE, decimal.places = NULL)
```

Arguments

`x`	Vector, string: Logical expressions
`space.after.comma`	Logical: If TRUE, place spaces after commas. Default = false
`decimal.places`	Integer: Limit all floats (numbers of the form 9.9) to this many decimal places

Author(s)

E.D. Gennatas

---

fwhm2sigma — *FWHM to Sigma*

Description

Convert Full width at half maximum values to sigma

Usage

```
fwhm2sigma(fwhm)
```

Arguments

`fwhm`	FWHM value

Value

sigma

Author(s)

E.D. Gennatas

Examples

```
fwhm2sigma(8)
# FWHM of 8 is equivalent to sigma = 3.397287
```

get-names	*Get factor/numeric/logical/character names from data.frame/data.table*

Description

Get factor/numeric/logical/character names from data.frame/data.table

Usage

```
getfactornames(x)
```

Arguments

x	data.frame or data.table (or data.frame-compatible object)

Author(s)

E.D. Gennatas

getnames	*Get names by string matching*

Description

Get names by string matching

Usage

```
getnames(
  x,
  pattern = NULL,
  starts_with = NULL,
  ends_with = NULL,
  ignore.case = TRUE
)

getnumericnames(x)

getlogicalnames(x)

getcharacternames(x)

getdatenames(x)
```

Arguments

`x`	object with `names()` method
`pattern`	Character: pattern to match anywhere in names of x
`starts_with`	Character: pattern to match in the beginning of names of x
`ends_with`	Character: pattern to match at the end of names of x
`ignore.case`	Logical: If TRUE, well, ignore case. Default = TRUE

Author(s)

E.D. Gennatas

`getnamesandtypes` *Get data.frame names and types*

Description

Get data.frame names and types

Usage

```
getnamesandtypes(x)
```

Arguments

`x`	data.frame / data.table or similar

Value

character vector of column names with attribute "type" holding the class of each column

---

`get_loaded_pkg_version` *Get version of all loaded packages (namespaces)*

Description

Get version of all loaded packages (namespaces)

Usage

```
get_loaded_pkg_version()
```

Value

Data frame with columns "Package_Name" and "Version"

Author(s)

E.D. Gennatas

`get_mode` *Get the mode of a factor or integer*

Description

Returns the mode of a factor or integer

Usage

```
get_mode(x, na.exclude = TRUE, getlast = TRUE, retain.class = TRUE)
```

Arguments

`x`	Vector, factor or integer: Input data
`na.exclude`	Logical: If TRUE, exclude NAs
`getlast`	Logical: If TRUE, get
`retain.class`	Logical: If TRUE, output is always same class as input

Value

The mode of `x`

Author(s)

E.D. Gennatas

Examples

```
x <- c(9, 3, 4, 4, 0, 2, 2, NA)
get_mode(x)
x <- c(9, 3, 2, 2, 0, 4, 4, NA)
get_mode(x)
get_mode(x, getlast = FALSE)
```

get_rules *Get RuleFit rules*

Description

Get rules generated by s_RuleFit or s_LightRuleFit

Usage

```
get_rules(
  mod,
  formatted = FALSE,
  collapse = TRUE,
  collapse.keep.names = FALSE,
  collapse.unique = TRUE
)
```

Arguments

`mod`	Model created by s_RuleFit or s_LightRuleFit
`formatted`	Logical: If TRUE, return human-readable rules, otherwise return R-parsable rules
`collapse`	Logical: If TRUE, collapse all rules to a single character vector
`collapse.keep.names`	Logical: If TRUE, keep names when collapsing (will be able to tell which run each rule came from). However, has no effect if `collapse.unique = TRUE`, as `unique()` removes names.
`collapse.unique`	Logical: If TRUE and `collapse = TRUE`, will return only unique rules

Author(s)

ED Gennatas

get_vars_from_rules *Extract variable names from rules*

Description

Extract variable names from rules

Usage

```
get_vars_from_rules(rules, unique = FALSE)
```

Arguments

`rules`	Character vector: Rules.
`unique`	Logical: If TRUE, return only unique variables.

Value

Character vector: Variable names.

Author(s)

E.D. Gennatas

`ggtheme_dark` **rtemis** `ggplot2` *dark theme*

Description

rtemis `ggplot2` dark theme

Usage

```
ggtheme_dark(
  base_size = 14,
  base_family = "Helvetica Neue",
  base_line_size = base_size/22,
  base_rect_size = base_size/22,
  axis.text.size.rel = 1,
  legend.key.fill = NA,
  legend.text.size.rel = 1,
  legend.position = "right",
  strip.background.fill = "gray25"
)
```

Arguments

`base_size`	Float: Base font size. Default = 14
`base_family`	Character: Font family. Default = "Helvetica Neue"
`base_line_size`	Float: Line size. Default = base_size/22
`base_rect_size`	Float: Size for rect elements. Default = base_size/22
`axis.text.size.rel`	Float: Relative size for axis text. Default = 1
`legend.key.fill`	Color: Fill color for legend. Default = NA (no color)
`legend.text.size.rel`	Float: Relative size for legend text. Default = 1

legend.position
Character: Legend position, "top", "bottom", "right", "left" Default = "right"

strip.background.fill
Color: Fill color from facet labels. Default = "gray25"

Author(s)

E.D. Gennatas

Examples

```
## Not run:
(p <- ggplot(iris, aes(Sepal.Length, Petal.Length, color = Species)) +
  geom_point() +
  ggtheme_light())

## End(Not run)
```

---

ggtheme_light *rtemis ggplot2 light theme*

Description

rtemis ggplot2 light theme

Usage

```
ggtheme_light(
  base_size = 14,
  base_family = "Helvetica Neue",
  base_line_size = base_size/22,
  base_rect_size = base_size/22,
  axis.text.size.rel = 1,
  legend.key.fill = NA,
  legend.text.size.rel = 1,
  legend.position = "right",
  strip.background.fill = "grey85"
)
```

Arguments

base_size Float: Base font size. Default = 14

base_family Character: Font family. Default = "Helvetica Neue"

base_line_size Float: Line size. Default = base_size/22

base_rect_size Float: Size for rect elements. Default = base_size/22

axis.text.size.rel
Float: Relative size for axis text. Default = 1

legend.key.fill
: Color: Fill color for legend. Default = NA (no color)

legend.text.size.rel
: Float: Relative size for legend text. Default = 1

legend.position
: Character: Legend position, "top", "bottom", "right", "left" Default = "right"

strip.background.fill
: Color: Fill color from facet labels. Default = "grey85"

Author(s)

E.D. Gennatas

Examples

```
## Not run:
(p <- ggplot(iris, aes(Sepal.Length, Petal.Length, color = Species)) +
  geom_point() +
  ggtheme_light())

## End(Not run)
```

---

glmLite *Bare bones decision tree derived from* rpart

Description

A super-stripped down decision tree for when space and performance are critical

Usage

```
glmLite(
  x,
  y,
  weights = NULL,
  method = c("glmnet", "cv.glmnet", "lm.ridge", "allSubsets", "forwardStepwise",
    "backwardStepwise", "glm", "sgd", "solve"),
  alpha = 0,
  lambda = 0.01,
  lambda.seq = NULL,
  cv.glmnet.nfolds = 5,
  which.cv.glmnet.lambda = c("lambda.min", "lambda.1se"),
  nbest = 1,
  nvmax = 8,
  sgd.model = "glm",
  sgd.model.control = list(lambda1 = 0, lambda2 = 0),
  sgd.control = list(method = "ai-sgd"),
```

```
  save.fitted = FALSE,
  ...
)
```

Arguments

x
: Feature matrix or data.frame. Will be coerced to data.frame for method = "allSubsets", "forwardStepwise", or "backwardStepwise"

y
: Outcome

weights
: Float, vector: Case weights

method
: Character: Method to use:
  - "glm": uses `stats::lm.wfit`
  - "glmnet": uses `glmnet::glmnet`
  - "cv.glmnet": uses `glmnet:cv.glmnet`
  - "lm.ridge": uses `MASS::lm.ridge`
  - "allsubsets": uses `leaps::regsubsets` with `method = "exhaustive"`
  - "forwardStepwise": uses `leaps::regsubsets` with `method = "forward"`
  - "backwardStepwise": uses `leaps::regsubsets` with `method = "backward"`
  - "sgd": uses `sgd::sgd`
  - "solve": uses `base::solve`
  - "none": fits no model and returns all zeroes, for programming convenience in special cases

alpha
: Float: `alpha` for method = `glmnet` or `cv.glmnet`.

lambda
: Float: The lambda value for `glmnet`, `cv.glmnet`, `lm.ridge` Note: For `glmnet` and `cv.glmnet`, this is the lambda used for prediction. Training uses `lambda.seq`.

lambda.seq
: Float, vector: lambda sequence for `glmnet` and `cv.glmnet`.

cv.glmnet.nfolds
: Integer: Number of folds for `cv.glmnet`

which.cv.glmnet.lambda
: Character: Whitch lambda to pick from cv.glmnet: "lambda.min": Lambda that gives minimum cross-validated error;

nbest
: Integer: For `method = "allSubsets"`, number of subsets of each size to record. Default = 1

nvmax
: Integer: For `method = "allSubsets"`, maximum number of subsets to examine.

sgd.model
: Character: Model to use for `method = "sgd"`.

sgd.model.control
: List: `model.control` list to pass to `sgd::sgd`

sgd.control
: List: `sgd.control` list to pass to `sgd::sgd` "lambda.1se": Largest lambda such that error is within 1 s.e. of the minimum.

save.fitted
: Logical: If TRUE, save fitted values in output. Default = FALSE

...
: Additional arguments to pass to lincoef

Author(s)

E.D. Gennatas

gmean *Geometric mean*

Description

Geometric mean

Usage

```
gmean(x)
```

Arguments

x	Numeric vector

Author(s)

E.D. Gennatas

Examples

```
x <- c(1, 3, 5)
mean(x)
gmean(x)
# same as, but a little faster than:
exp(mean(log(x)))
```

gp *Bayesian Gaussian Processes [R]*

Description

Fit a gaussian process

Usage

```
gp(
  x,
  y,
  new.x = NULL,
  x.name = "x",
  y.name = "y",
  print.plot = TRUE,
  lwd = 3,
  cex = 1.2,
  par.reset = TRUE,
  ...
)
```

Arguments

x	Numeric vector or matrix of features, i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
new.x	(Optional) Numeric vector or matrix of new set of features Must have same set of columns as x
x.name	Character: Name for feature set
y.name	Character: Name for outcome
print.plot	Logical: if TRUE, draw plot when done
lwd	Line width for plotting
cex	Character expansion factor for plotting
par.reset	Logical. Reset par to its original state
...	Additional arguments to be passed to tgp::bgp

Author(s)

E.D. Gennatas

---

graph_node_metrics *Node-wise (i.e. vertex-wise) graph metrics*

Description

Node-wise (i.e. vertex-wise) graph metrics

Usage

```
graph_node_metrics(x, verbose = TRUE)
```

Arguments

x	**igraph** network
verbose	Logical: If TRUE, print messages to console

Author(s)

E.D. Gennatas

Examples

```
## Not run:
datcor <- cor(rnormmat(20, 20, seed = 2021))
datcor[sample(seq(datcor), 250)] <- 0
x <- igraph::graph_from_adjacency_matrix(adjmatrix = datcor,
                                         mode = "lower",
                                         weighted = TRUE,
                                         diag = FALSE)

graph_node_metrics(x)

## End(Not run)
```

`gridCheck` **rtemis** *internal: Grid check*

Description

Checks if grid search needs to be performed. All tunable parameters should be passed to this function, individually or as a list. If any argument has more than one assigned values, the function returns TRUE, otherwise FALSE. This can be used to check whether `gridSearchLearn` must be run.

Usage

```
gridCheck(...)
```

Arguments

`...`	Parameters; will be converted to a list

Details

The idea is that if you know which parameter values you want to use, you define them directly e.g. `alpha = 0, lambda = .2`. If you don't know, you enter the set of values to be tested, e.g. `alpha = c(0, .5, 1), lambda = seq(.1, 1, .1)`.

`gtTable` *Greater-than Table*

Description

Compare vectors element-wise, and tabulate N times each vector is greater than the others

Usage

```
gtTable(x = list(), x.name = NULL, na.rm = TRUE, verbose = TRUE)
```

Arguments

x	List of vectors of same length
x.name	Character: Name of measure being compared
na.rm	Passed to sum to handle missing values
verbose	Logical: If TRUE, write output to console

Author(s)

E.D. Gennatas

---

htest *Basic Bivariate Hypothesis Testing and Plotting*

Description

Basic Bivariate Hypothesis Testing and Plotting

Usage

```
htest(
  y,
  group = NULL,
  x = NULL,
  yname = NULL,
  groupname = NULL,
  xname = NULL,
  test = c("t.test", "wilcox.test", "aov", "kruskal.test", "chisq.test", "fisher.test",
    "cor.test", "pearson", "kendall", "spearman", "ks"),
  print.plot = TRUE,
  plot.args = list(),
  theme = rtTheme,
  verbose = TRUE,
  ...
)
```

Arguments

y	Float, vector: Outcome of interest
group	Factor: Groups to compare
x	Float, vector: Second outcome for correlation tests
yname	Character: y variable name
groupname	Character: group variable name
xname	Character: x variable name
test	Character: Test to use; one of:

- Continuous outcome by group: "t.test", "wilcox.test", "aov", "kruskal.test"
- Categorical outcome by group: "chisq.test", "fisher.test", "cor.test"
- Two continuous variables: "pearson", "kendall", "spearman"

`print.plot` Logical: If TRUE, print plot. Default = TRUE

`plot.args` List of arguments to pass to plotting function

`theme` Character: Run `themes()` for available themes

`verbose` Logical: If TRUE, print messages to console. Default = TRUE

`...` Additional arguments to pass to test call

Author(s)

E.D. Gennatas

Examples

```
## Not run:
# t.test, wilcoxon
y <- c(rnorm(200, 2, 1.2), rnorm(300, 2.5, 1.4))
group <- c(rep(1, 200), rep(2, 300))

ht_ttest <- htest(y, group, test = "t.test")
ht_wilcoxon <- htest(y, group, test = "wilcox.test")

# aov, kruskal
y <- c(rnorm(200, 2, 1.2), rnorm(300, 2.5, 1.4), rnorm(100, 2.3, 1.1))
group <- c(rep(1, 200), rep(2, 300), rep(3, 100))

ht_aov <- htest(y, group, test = "aov")
ht_kruskal <- htest(y, group, test = "kruskal.test")

# chisq, fisher
y <- c(sample(c(1, 2), 100, T, c(.7, .3)), sample(c(1, 2), 100, T, c(.35, .65)))
group <- c(rep(1, 100), rep(2, 100))
ht_chisq <- htest(y, group, test = "chisq")
ht_fisher <- htest(y, group, test = "fisher")

# cor.test
x <- rnorm(300)
y <- x * .3 + rnorm(300)
ht_pearson <- htest(x = x, y = y, test = "pearson")
ht_kendall <- htest(x = x, y = y, test = "kendall")
ht_kendall <- htest(x = x, y = y, test = "spearman")

## End(Not run)
```

inspect_type *Inspect character and factor vector*

Description

Checks character or factor vector to determine whether it might be best to convert to numeric.

Usage

```
inspect_type(x, xname = NULL, verbose = TRUE, thresh = 0.5, na.omit = TRUE)
```

Arguments

x	Character or factor vector.
xname	Character: Name of input vector x.
verbose	Logical: If TRUE, print messages to console.
thresh	Numeric: Threshold for determining whether to convert to numeric.
na.omit	Logical: If TRUE, remove NA values before checking.

Author(s)

E.D. Gennatas

Examples

```
## Not run:
x <- c("3", "5", "undefined", "21", "4", NA)
inspect_type(x)
z <- c("mango", "banana", "tangerine", NA)
inspect_type(z)

## End(Not run)
```

invlogit *Inverse Logit*

Description

Inverse Logit

Usage

```
invlogit(x)
```

Arguments

x	Float: Input data

Value

The inverse logit of the input

Author(s)

E.D. Gennatas

---

is_constant *Check if vector is constant*

Description

Check if vector is constant

Usage

```
is_constant(x, skip_missing = FALSE)
```

Arguments

x	Vector: Input
skip_missing	Logical: If TRUE, skip NA values before testing

Author(s)

E.D. Gennatas

Examples

```
## Not run:
x <- rep(9, 1000000)
is_constant(x)
x[10] <- NA
is_constant(x)
is_constant(x, skip_missing = TRUE)

## End(Not run)
```

`is_discrete` *Check if variable is discrete (factor or integer)*

Description

Check if variable is discrete (factor or integer)

Usage

```
is_discrete(x)
```

Arguments

x	Input

Author(s)

E.D. Gennatas

`kfold` *K-fold Resampling*

Description

K-fold Resampling

Usage

```
kfold(
  x,
  k = 10,
  stratify.var = NULL,
  strat.n.bins = 4,
  seed = NULL,
  verbosity = TRUE
)
```

Arguments

x	Input Vector
k	Integer: Number of folds. Default = 10
stratify.var	Numeric vector (optional): Variable used for stratification.
strat.n.bins	Integer: Number of groups to use for stratification for `resampler = "strat.sub" / "strat.boot"`
seed	Integer: (Optional) Set seed for random number generator, in order to make output reproducible. See `?base::set.seed`
verbosity	Logical: If TRUE, print messages to console

Author(s)

E.D. Gennatas

---

labelify *Format text for label printing*

Description

Format text for label printing

Usage

```
labelify(
  x,
  underscoresToSpaces = TRUE,
  dotsToSpaces = TRUE,
  toLower = FALSE,
  toTitleCase = TRUE,
  capitalize.strings = c("id"),
  stringsToSpaces = c("\\$", "`")
)
```

Arguments

x	Character: Input
underscoresToSpaces	Logical: If TRUE, convert underscores to spaces.
dotsToSpaces	Logical: If TRUE, convert dots to spaces.
toLower	Logical: If TRUE, convert to lowercase (precedes `toTitleCase`). Default = FALSE (Good for getting all-caps words converted to title case, bad for abbreviations you want to keep all-caps)
toTitleCase	Logical: If TRUE, convert to Title Case. Default = TRUE (This does not change all-caps words, set `toLower` to TRUE if desired)
capitalize.strings	Character, vector: Always capitalize these strings, if present. Default = `"id"`
stringsToSpaces	Character, vector: Replace these strings with spaces. Escape as needed for `gsub`. Default = `"\\$"`, which formats common input of the type `data.frame$variable`

Author(s)

E.D. Gennatas

lincoef *Linear Model Coefficients*

Description

Get linear model coefficients

Usage

```
lincoef(
  x,
  y,
  weights = NULL,
  method = "glmnet",
  type = c("Regression", "Classification", "Survival"),
  learning.rate = 1,
  alpha = 1,
  lambda = 0.05,
  lambda.seq = NULL,
  cv.glmnet.nfolds = 5,
  which.cv.glmnet.lambda = c("lambda.min", "lambda.1se"),
  nbest = 1,
  nvmax = 8,
  sgd.model = "glm",
  sgd.model.control = list(lambda1 = 0, lambda2 = 0),
  sgd.control = list(method = "ai-sgd"),
  trace = 0
)
```

Arguments

`x`	Feature matrix or data.frame. Will be coerced to data.frame for method = "allSubsets", "forwardStepwise", or "backwardStepwise"
`y`	Outcome
`weights`	Float, vector: Case weights
`method`	Character: Method to use: • "glm": uses `stats::lm.wfit` • "glmnet": uses `glmnet::glmnet` • "cv.glmnet": uses `glmnet:cv.glmnet` • "lm.ridge": uses `MASS::lm.ridge` • "allsubsets": uses `leaps::regsubsets` with `method = "exhaustive"` • "forwardStepwise": uses `leaps::regsubsets` with `method = "forward"` • "backwardStepwise": uses `leaps::regsubsets` with `method = "backward"` • "sgd": uses `sgd::sgd` • "solve": uses `base::solve`

- "none": fits no model and returns all zeroes, for programming convenience in special cases

`type` Character: "Regression", "Classification", or "Survival"

`learning.rate` Numeric: Coefficients will be multiplied by this number

`alpha` Float: `alpha` for method = `glmnet` or `cv.glmnet`.

`lambda` Float: The lambda value for `glmnet`, `cv.glmnet`, `lm.ridge` Note: For `glmnet` and `cv.glmnet`, this is the lambda used for prediction. Training uses `lambda.seq`.

`lambda.seq` Float, vector: lambda sequence for `glmnet` and `cv.glmnet`.

`cv.glmnet.nfolds` Integer: Number of folds for `cv.glmnet`

`which.cv.glmnet.lambda` Character: Whitch lambda to pick from cv.glmnet: "lambda.min": Lambda that gives minimum cross-validated error;

`nbest` Integer: For `method = "allSubsets"`, number of subsets of each size to record. Default = 1

`nvmax` Integer: For `method = "allSubsets"`, maximum number of subsets to examine.

`sgd.model` Character: Model to use for `method = "sgd"`.

`sgd.model.control` List: `model.control` list to pass to `sgd::sgd`

`sgd.control` List: `sgd.control` list to pass to `sgd::sgd` "lambda.1se": Largest lambda such that error is within 1 s.e. of the minimum.

`trace` Integer: If set to zero, all warnings are ignored

Details

This function minimizes checks for speed. It doesn't check dimensionality of `x`. Only use methods "glm", "sgd", or "solve" if there is only one feature in `x`.

Value

Named numeric vector of linear coefficients

Author(s)

E.D. Gennatas

list2csv	*Write list elements to CSV files*

Description

Write list elements to CSV files

Usage

```
list2csv(x, outdir)
```

Arguments

x	List containing R objects to be written to CSV (e.g. data.frames, matrices, etc.)
outdir	Character: Path to output directory

Author(s)

E.D. Gennatas

logistic	*Logistic function*

Description

Logistic function

Usage

```
logistic(x, x0 = 0, L = 1, k = 1)
```

Arguments

x	Float: Input
x0	x-value of the midpoint.
L	maximum value.
k	steepness of the curve.

logit	*Logit transform*

Description

Logit transform

Usage

```
logit(x)
```

Arguments

x	Float [0, 1] Input

logloss	*Log Loss for a binary classifier*

Description

Log Loss for a binary classifier

Usage

```
logloss(true, estimated.prob)
```

Arguments

true	Factor: True labels. First level is the positive case
estimated.prob	Float, vector: Estimated probabilities

Author(s)

E.D. Gennatas

loocv — *Leave-one-out Resampling*

Description

Leave-one-out Resampling

Usage

```
loocv(x)
```

Arguments

x	Input vector

Author(s)

E.D. Gennatas

lotri2edgeList — *Connectivity Matrix to Edge List*

Description

Turn the lower triangle of a connectivity matrix (e.g. correlation matrix or similar) to an edge list of the form: Source, Target, Weight

Usage

```
lotri2edgeList(A, filename = NULL, verbose = TRUE)
```

Arguments

A	Square matrix
filename	Character: Path for csv file. Defaults to "conmat2edgelist.csv"
verbose	Logical: If TRUE, print messages to console

Details

The output can be read, for example, into gephi

Author(s)

E.D. Gennatas

lsapply lsapply

Description

lsapply

Usage

```
lsapply(X, FUN, ..., outnames = NULL, simplify = FALSE)
```

Arguments

X	a vector (atomic or list) or an expression object. Other objects (including classed objects) will be coerced by base::as.list.
FUN	the function to be applied to each element of X: see 'Details'. In the case of functions like +, %*%, the function name must be backquoted or quoted.
...	optional arguments to FUN.
outnames	Character vector: Optional names to apply to output
simplify	logical or character string; should the result be simplified to a vector, matrix or higher dimensional array if possible? For sapply it must be named and not abbreviated. The default value, TRUE, returns a vector or matrix if appropriate, whereas if simplify = "array" the result may be an array of "rank" (=length(dim(.))) one higher than the result of FUN(X[[i]]).

make_key *Make key from data.table id - description columns*

Description

Make key from data.table id - description columns

Usage

```
make_key(x, code_name, description_name, filename = NULL)
```

Arguments

x	Input data.table
code_name	Character: Name of column name that holds codes
description_name	Character: Name of column that holds descriptions
filename	Character: Path to file to save CSV with key

Author(s)

E.D. Gennatas

massGAM *Mass-univariate GAM Analysis*

Description

Fits a GAM for each of multiple outcomes using a fixed set of features (many y's, one X).

Usage

```
massGAM(
  x,
  y,
  covariates = NULL,
  x.name = NULL,
  y.name = NULL,
  k = NULL,
  family = gaussian(),
  weights = NULL,
  method = "REML",
  n.cores = rtCores,
  save.mods = FALSE,
  save.summary = TRUE,
  print.plots = FALSE,
  outdir = NULL,
  save.plots = FALSE,
  new.x.breaks = 9
)
```

Arguments

`x`	Numeric vector or matrix / data frame of features i.e. independent variables
`y`	Numeric matrix / data frame: Outcomes
`covariates`	Numeric matrix / data.frame of additional covariates
`x.name`	Character: Name of the predictor
`y.name`	Character, vector: Names of the outcomes
`k`	Integer: Basis dimension for smoothing spline
`family`	`family` argument for `mgcv::gam`
`weights`	Vector, numeric: Weights for GAM
`method`	Estimation method for GAM
`n.cores`	Integer. Number of cores to use
`save.mods`	Logical. Should models be saved
`save.summary`	Logical. Should model summary be saved
`print.plots`	Logical Should plots be shown

outdir	Path to save output
save.plots	Logical. Should plots be saved
new.x.breaks	Integer. Number of splits in the range of x to form vector of features for estimation of fitted values

Details

NA in the input will be kept as NA in the results, maintaining n of cases.

Author(s)

E.D. Gennatas

---

massGLAM *Mass-univariate GLM Analysis*

Description

Run a mass-univariate analysis with either: a) single outome (y) and multiple predictors (x), one at a time, with an optional common set of covariates in each model - "massx" b) multiple different outcomes (y) with a fixed set of predictors (x) - "massy" Therefore, the term mass-univariate refers to looking at one variable of interest (with potential covariates of no interest) at a time

Usage

```
massGLAM(
  x,
  y,
  scale.x = FALSE,
  scale.y = FALSE,
  mod = c("glm", "gam"),
  type = NULL,
  xnames = NULL,
  ynames = NULL,
  spline.index = NULL,
  gam.k = 6,
  save.mods = TRUE,
  print.plot = FALSE,
  include_anova_pvals = NA,
  verbose = TRUE,
  trace = 0,
  n.cores = 1
)
```

Arguments

x	Matrix / data frame of features
y	Matrix / data frame of outcomes
scale.x	Logical: If TRUE, scale and center x
scale.y	Logical: If TRUE, scale and center y
mod	Character: "glm" or "gam".
type	Character: "massx" or "massy". Default = NULL, where if (NCOL(x) > NCOL(y)) "massx" else "massy"
xnames	Character vector: names of x feature(s)
ynames	Character vector: names of y feature(s)
spline.index	Integer vector: indices of features to fit splines for.
gam.k	Integer: The dimension of the spline basis.
save.mods	Logical: If TRUE, save models. Default = TRUE
print.plot	Logical: If TRUE, print plot. Default = FALSE (best to choose which p-values you want to plot directly)
include_anova_pvals	Logical: If TRUE, include ANOVA p-values, generated by glm2table (internal function)
verbose	Logical: If TRUE, print messages during run
trace	Integer: If > 0, print more verbose output to console.
n.cores	Integer: Number of cores to use. (Testing only, do not change from 1)

Author(s)

E.D. Gennatas

Examples

```
## Not run:
# Common usage is "reversed":
# x: outcome of interest as first column, optional covariates
# in the other columns
# y: features whose association with x we want to study
set.seed(2022)
features <- rnormmat(500, 40)
outcome <- features[, 3] - features[, 5] + features[, 14] + rnorm(500)
massmod <- massGLAM(outcome, features)
plot(massmod)
plot(massmod, what = "coef")
plot(massmod, what = "volcano")

## End(Not run)
```

`massGLM` *Mass-univariate GLM Analysis*

Description

Run a mass-univariate analysis with either: a) single outome (y) and multiple predictors (x), one at a time, with an optional common set of covariates in each model - "massx" b) multiple different outcomes (y) with a fixed set of predictors (x) - "massy" Therefore, the term mass-univariate refers to looking at one variable of interest (with potential covariates of no interest) at a time

Usage

```
massGLM(
  x,
  y,
  scale.x = FALSE,
  scale.y = FALSE,
  type = NULL,
  xnames = NULL,
  ynames = NULL,
  coerce.y.numeric = FALSE,
  save.mods = FALSE,
  print.plot = FALSE,
  include_anova_pvals = NA,
  verbose = TRUE,
  trace = 0
)
```

Arguments

`x`	Matrix / data frame of features
`y`	Matrix / data frame of outcomes
`scale.x`	Logical: If TRUE, scale and center `x`
`scale.y`	Logical: If TRUE, scale and center `y`
`type`	Character: "massx" or "massy". Default = NULL, where if (NCOL(x) > NCOL(y)) "massx" else "massy"
`xnames`	Character vector: names of `x` feature(s)
`ynames`	Character vector: names of `y` feature(s)
`coerce.y.numeric`	Logical: If `TRUE`, coerce `y` to numeric
`save.mods`	Logical: If TRUE, save models.
`print.plot`	Logical: If TRUE, print plot.
`include_anova_pvals`	Logical: If TRUE, include ANOVA p-values, (generated by `glm2table`)
`verbose`	Logical: If TRUE, print messages during run
`trace`	Integer: If > 0, print more verbose output to console.

Author(s)

E.D. Gennatas

Examples

```
## Not run:
# Common usage is "reversed":
# x: outcome of interest as first column, optional covariates
# in the other columns
# y: features whose association with x we want to study
set.seed(2022)
features <- rnormmat(500, 40)
outcome <- features[, 3] - features[, 5] + features[, 14] + rnorm(500)
massmod <- massGLM(outcome, features)
plot(massmod)
plot(massmod, what = "coef")
plot(massmod, what = "volcano")

## End(Not run)
```

`massUni` *Mass-univariate Analysis*

Description

Run a mass-univariate analysis: same features (predictors) on multiple outcomes

Usage

```
massUni(
  x,
  y,
  mod = "gam",
  save.mods = FALSE,
  verbose = TRUE,
  n.cores = rtCores,
  ...
)
```

Arguments

`x`	Matrix / data frame of features
`y`	Matrix / data frame of outcomes
`mod`	**rtemis** algorithm to use. Options: run `select_learn()`
`save.mods`	Logical: If TRUE, save fitted models
`verbose`	Logical: If TRUE, print messages during run

`n.cores`	Integer: Number of cores to use
`...`	Arguments to be passed to `mod`

Author(s)

E.D. Gennatas

`matchcases` *Match cases by covariates*

Description

Find one or more cases from a `pool` data.frame that match cases in a target data.frame. Match exactly and/or by distance (sum of squared distances).

Usage

```
matchcases(
  target,
  pool,
  n.matches = 1,
  target.id = NULL,
  pool.id = NULL,
  exactmatch.factors = TRUE,
  exactmatch.cols = NULL,
  distmatch.cols = NULL,
  norepeats = TRUE,
  ignore.na = FALSE,
  verbose = TRUE
)
```

Arguments

`target`	data.frame you are matching against
`pool`	data.frame you are looking for matches from
`n.matches`	Integer: Number of matches to return
`target.id`	Character: Column name in `target` that holds unique cases IDs. Default = NULL, in which case integer case numbers will be used
`pool.id`	Character: Same as `target.id` for `pool`
`exactmatch.factors`	Logical: If TRUE, selected cases will have to exactly match factors available in `target`
`exactmatch.cols`	Character: Names of columns that should be matched exactly
`distmatch.cols`	Character: Names of columns that should be distance-matched

norepeats	Logical: If TRUE, cases in pool can only be chosen once.
ignore.na	Logical: If TRUE, ignore NA values during exact matching.
verbose	Logical: If TRUE, print messages to console. Default = TRUE

Author(s)

E.D. Gennatas

Examples

```
set.seed(2021)
cases <- data.frame(
  PID = paste0("PID", seq(4)),
  Sex = factor(c(1, 1, 0, 0)),
  Handedness = factor(c(1, 1, 0, 1)),
  Age = c(21, 27, 39, 24),
  Var = c(.7, .8, .9, .6),
  Varx = rnorm(4)
)
controls <- data.frame(
  CID = paste0("CID", seq(50)),
  Sex = factor(sample(c(0, 1), 50, TRUE)),
  Handedness = factor(sample(c(0, 1), 50, TRUE, c(.1, .9))),
  Age = sample(16:42, 50, TRUE),
  Var = rnorm(50),
  Vary = rnorm(50)
)

mc <- matchcases(cases, controls, 2, "PID", "CID")
```

mergelongtreatment *Merge panel data treatment and outcome data*

Description

Merge long format treatment and outcome data from multiple sources with possibly hierarchical matching IDs using **data.table**

Usage

```
mergelongtreatment(
  x,
  group_varnames,
  time_varname = "Date",
  start_date,
  end_date,
  interval_days = 14,
  verbose = TRUE,
  trace = 1
)
```

Arguments

x	Named list: Long form datasets to merge. Will be converted to `data.table`
group_varnames	Vector, character: Variable names to merge by, in order. If first is present on a given pair of datasets, merge on that, otherwise try the next in line.
time_varname	Character: Name of column that should be present in all datasets containing time information. Default = "Date"
start_date	Date or characcter: Start date for final dataset in format "YYYY-MM-DD"
end_date	Date or character: End dat for final dataset in format "YYYY-MM-DD"
interval_days	Integer: Starting with `start_date` create timepoints every this many days. Default = 14
verbose	Logical: If TRUE, print messages to console. Default = TRUE
trace	Integer: If > 0 print additional info to console. Default = 1

Value

Merged **data.table**

---

meta_mod *Meta Models for Regression (Model Stacking)*

Description

Train a meta model from the output of base learners trained using different learners (algorithms)

Usage

```
meta_mod(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  base.mods = c("mars", "ranger"),
  base.params = vector("list", length(base.mods)),
  base.resample.params = setup.resample(resampler = "kfold", n.resamples = 4),
  meta.mod = "gam",
  meta.params = list(),
  x.name = NULL,
  y.name = NULL,
  save.base.res = TRUE,
  save.base.full = FALSE,
  col = NULL,
  se.lty = 3,
  print.base.plot = FALSE,
  print.plot = TRUE,
```

```
    plot.fitted = NULL,
    plot.predicted = NULL,
    plot.theme = rtTheme,
    question = NULL,
    verbose.base.res.mods = FALSE,
    verbose.base.mods = FALSE,
    verbose = TRUE,
    trace = 0,
    base.n.cores = 1,
    n.cores = rtCores,
    save.mod = FALSE,
    outdir = NULL,
    ...
)
```

Arguments

`x`	Numeric vector or matrix of features, i.e. independent variables
`y`	Numeric vector of outcome, i.e. dependent variable
`x.test`	(Optional) Numeric vector or matrix of validation set features must have set of columns as `x`
`y.test`	(Optional) Numeric vector of validation set outcomes
`base.mods`	Character vector: Two or more base learners. Options: select_learn
`base.params`	List of length equal to N of `base.mods`. Each element should be a list of arguments to pass to the corresponding base mod
`meta.mod`	String. Meta learner. Options: select_learn
`x.name`	Character: Name for predictor set. (What kind of data is it?)
`y.name`	Character: Name for outcome
`se.lty`	How to plot standard errors. If a number, it corresponds to par("lty") line types and is plotted with lines(). If "solid", a transparent polygon is plotted using polygon()
`resampler`	String. Resampling method to use. Options: "bootstrap", "kfold", "strat.boot", "strat.sub"

Details

This is included mainly for educational purposes.

- Train a set of base learners on resamples of the training set x
- Train a meta learner to map bases' validation set predictions to outcomes
- Train base learners on full training set x
- Use the meta learner to predict test set outcome y.test from testing set (x.test)

Author(s)

E.D. Gennatas

mgetnames *Get names by string matching multiple patterns*

Description

Get names by string matching multiple patterns

Usage

```
mgetnames(
  x,
  pattern = NULL,
  starts_with = NULL,
  ends_with = NULL,
  ignore.case = TRUE,
  return.index = FALSE
)
```

Arguments

`x`	Character vector or object with `names()` method
`pattern`	Character vector: pattern(s) to match anywhere in names of x
`starts_with`	Character: pattern to match in the beginning of names of x
`ends_with`	Character: pattern to match at the end of names of x
`ignore.case`	Logical: If TRUE, well, ignore case. Default = TRUE
`return.index`	Logical: If TRUE, return integer index of matches instead of names

Value

Character vector of matched names or integer index

Author(s)

E.D. Gennatas

mhist *Histograms*

Description

Draws a histogram using lines.

Usage

```
mhist(
  x,
  breaks = "Sturges",
  measure = c("density", "counts"),
  lwd = 3,
  xlim = NULL,
  ylim = NULL,
  plot.axes = FALSE,
  xaxis = TRUE,
  yaxis = TRUE,
  xaxis.line = 0,
  yaxis.line = 0,
  xlab = NULL,
  ylab = measure,
  xaxs = "r",
  yaxs = "r",
  box = FALSE,
  grid = FALSE,
  col = pennCol$lighterBlue,
  horiz = FALSE,
  main = "",
  add = FALSE,
  ...
)
```

Arguments

`x`	Input vector
`breaks`	See `hist("breaks")` Default = "Sturges"
`measure`	Character: "density"(Default), "counts"
`lwd`	Float: Line width
`xlim`	Vector, length 2: x-axis limits
`ylim`	Vector, length 2: y-axis limits
`plot.axes`	Logical: If TRUE, draws plot axes. Separate from `xaxis` and `yaxis`
`xaxis`	Logical: If TRUE, draws x-axis
`yaxis`	Logical: If TRUE, draws y-axis
`xaxis.line`	Float: Number of lines into the margin to position `xaxis`. See `axis("line")`
`yaxis.line`	Float: Number of lines into the margin to position `yaxis`. See `axis("line")`
`xlab`	Character: x-axis label
`ylab`	Character: y-axis label
`xaxs`	Character: 'r' (Default): Extends x-axis range by 4 percent at each end, 'i': Does not extend x-axis range

yaxs	Character: 'r' (Default): Extends y-axis range by 4 percent at each end, 'i': Does not extend y-axis range
box	Logical: If TRUE, draws a box around plot
grid	Logical: If TRUE, draws a grid
col	Color to use for histogram lines
horiz	Logical: If TRUE, switches x and y axes. Important: Provide all other arguments as if for a non-rotated plot - i.e. `xlab` will become the y-axis label
main	Character: Main title
add	Logical: If TRUE, add histogram to existing plot (Caution: make sure the axes line up!)
...	Additional arguments to be passed to `graphics::plot`

Details

Using `horiz = TRUE`, you can draw vertical histograms (as used by `mplot3_xym`)

Author(s)

E.D. Gennatas

`mlegend` *Add legend to* `mplot3` *plot*

Description

Add legend to `mplot3` plot

Usage

```
mlegend(
  lims,
  title = NULL,
  group.names,
  title.col = "black",
  col = rtpalette("rtCol1"),
  horiz.pad = 0.04,
  footer = NULL,
  font = 1,
  font.family = "Helvetica Neue"
)
```

Arguments

`lims`	List with plot limits in the form list(xlim = xlim, ylim = ylim) e.g. as returned by mplot3_xy
`title`	Character: Legend title
`group.names`	Character: group names
`title.col`	Title color
`col`	Color vector
`horiz.pad`	Numeric: Proportion of plot width to pad by
`footer`	Character: Footer annotation
`font`	1 or 2 for regular and bold
`font.family`	Character: Font family to use

Author(s)

E.D. Gennatas

`mod_error` *Error Metrics for Supervised Learning*

Description

Calculate error metrics for pair of vector, e.g. true and estimated values from a model

Usage

```
mod_error(
  true,
  estimated,
  estimated.prob = NULL,
  type = NULL,
  rho = FALSE,
  tau = FALSE,
  na.rm = TRUE,
  verbosity = 0
)
```

Arguments

`true`	Vector: True values
`estimated`	Vector: Estimated values
`estimated.prob`	Vector: Estimated probabilities for Classification, if available.
`type`	Character: "Regression", "Classification", or "Survival". If not provided, will be set to Regression if y is numeric.

rho	Logical: If TRUE, calculate Spearman's rho.
tau	Logical: If TRUE, calculate Kendall's tau. This can be slow for long vectors
na.rm	Logical: Passed to mean and range functions.
verbosity	Integer: If > 0, print messages to console.

Details

In regression, NRMSE = RMSE / range(observed)

Value

Object of class mod_error

Author(s)

E.D. Gennatas

---

mplot3_adsr *mplot3: ADSR Plot*

Description

Plot Attack Decay Sustain Release Envelope Generator using mplot3_xy

Usage

```
mplot3_adsr(
  Attack = 300,
  Decay = 160,
  Sustain = 40,
  Release = 500,
  Value = 80,
  I = 200,
  O = 1800,
  lty = 1,
  lwd = 4,
  main = "ADSR Envelope",
  main.line = 1.6,
  main.col = "white",
  Attack.col = "#44A6AC",
  Decay.col = "#F4A362",
  Sustain.col = "#3574A7",
  Release.col = "#C23A70",
  draw.poly = FALSE,
  poly.alpha = 0.15,
  draw.verticals = TRUE,
  v.lty = 1,
```

```
  v.lwd = 0.8,
  arrow.code = 2,
  arrow.length = 0.09,
  grid = FALSE,
  grid.lty = 1,
  grid.lwd = 0.4,
  grid.col = NULL,
  zerolines.col = "gray50",
  theme = "darkgray",
  labs.col = "gray70",
  tick.col = "gray70",
  on.col = "gray70",
  off.col = "gray70",
  pty = "m",
  mar = c(3, 3, 3.2, 0.5),
  xaxs = "i",
  yaxs = "i",
  par.reset = TRUE,
  ...
)
```

Arguments

`Attack`	Numeric: Attack time (in milliseconds)
`Decay`	Numeric: Decay time (in milliseconds)
`Sustain`	Numeric: Sustain Level (percent)
`Release`	Numeric: Release time (in milliseconds)
`Value`	Numeric: Value (percent)
`I`	Numeric: Note on time (in milliseconds)
`O`	Numeric: Note off time (in milliseconds)
`lty`	Integer: Line type
`lwd`	Numeric: Line width
`main`	Character: Main title
`main.line`	Numeric: Main title line height
`main.col`	Main title color
`Attack.col`	Attack color
`Decay.col`	Decay color
`Sustain.col`	Sustain color
`Release.col`	Release color
`draw.poly`	Logical: If TRUE, draw polygons for each segment
`poly.alpha`	Numeric: Polygon alpha
`draw.verticals`	Logical: If TRUE, draw vertical lines
`v.lty`	Integer: Vertical line type

v.lwd	Numeric: Vertical line width
arrow.code	Integer: Arrow code
arrow.length	Numeric: Arrow length
grid	Logical: If TRUE, draw grid
grid.lty	Integer: Grid line type
grid.lwd	Numeric: Grid line width
grid.col	Grid line color
zerolines.col	Color for zero lines
theme	Character: "light" or "dark" (Default)
labs.col	Color for axis labels
tick.col	Color for axis ticks
on.col	Color for "on" line
off.col	Color for "off" line
pty	Character: "s" gives a square plot; "m" gives a plot that fills graphics device size. Default = "m" (See `par("pty")`)
mar	Float, vector, length 4: Margins; see `par("mar")`
xaxs	Character: "r": Extend plot x-axis limits by 4% on either end; "i": Use exact x-axis limits.
yaxs	Character: as `xaxs` for the y-axis.
par.reset	Logical: If TRUE, reset `par` setting before exiting.
...	Additional arguments to pass to mplot3_xy

Details

Learn more: (https://en.wikipedia.org/wiki/Synthesizer#Attack_Decay_Sustain_Release_.28ADSR.29_envelope "ADSR Wikipedia")

Author(s)

E.D. Gennatas

Examples

```
## Not run:
mplot3_adsr()

## End(Not run)
```

mplot3_bar *mplot3: Barplot*

Description

Draw barplots

Usage

```
mplot3_bar(
  x,
  error = NULL,
  col = NULL,
  error.col = "white",
  error.lwd = 2,
  alpha = 1,
  beside = TRUE,
  border = NA,
  width = 1,
  space = NULL,
  xlim = NULL,
  ylim = NULL,
  xlab = NULL,
  ylab = NULL,
  main = NULL,
  las = 1.5,
  xnames = NULL,
  xnames.srt = 0,
  xnames.adj = ifelse(xnames.srt == 0, 0.5, 1),
  xnames.line = 0.5,
  xnames.font = 1,
  xnames.cex = 1,
  xnames.y.pad = 0.08,
  xnames.at = NULL,
  color.bygroup = FALSE,
  group.legend = NULL,
  legend.x = NULL,
  legend.y = NULL,
  group.names = NULL,
  legend.font = 1,
  bartoplabels = NULL,
  bartoplabels.line = 0,
  bartoplabels.font = 1,
  mar = c(2.5, 3, 2, 1),
  pty = "m",
  barplot.axes = FALSE,
  yaxis = TRUE,
```

```
    ylim.pad = 0.04,
    theme = rtTheme,
    palette = rtPalette,
    autolabel = letters,
    par.reset = TRUE,
    pdf.width = 6,
    pdf.height = 6,
    filename = NULL,
    ...
)
```

Arguments

`x`	Vector or Matrix: If Vector, each value will be drawn as a bar. If Matrix, each column is a vector, so multiple columns signify a different group. e.g. Columns could be months and rows could be N days sunshine, N days rainfall, N days snow, etc.
`error`	Vector or Matrix: If Vector, each value will be drawn as an error bar. If Matrix, each column is a vector, so multiple columns signify a different group.
`col`	Vector of colors to use
`alpha`	Float: Alpha to be applied to `col`
`border`	Color if you wish to draw border around bars, NA for no borders (Default)
`space`	Float: Space left free on either side of the bars, as a fraction of bar width. A single number or a vector, one value per bar. If `x` is a matrix, space can be length 2 vector, signifying space between bars within group and between groups. Default = c(0, 1) if x is matrix and `beside = TRUE`, otherwise Default = .2
`xlim`	Float vector, length 2: x-axis limits
`ylim`	Float vector, length 2: y-axis limits
`xlab`	Character: x-axis label
`ylab`	Character: y-axis label
`main`	Character: Plot title
`color.bygroup`	Logical: If TRUE, and input is a matrix, each group's bars will be given the same color, otherwise bars across groups will be given the same sequence of colors. Default = FALSE
`group.legend`	Logical: If TRUE, place `group.names` in a legend
`group.names`	(Optional) If multiple groups are plotted, use these names if `group.title = TRUE`
`mar`	Float, vector, length 4: Margins; see `par("mar")`
`pty`	Character: "s" gives a square plot; "m" gives a plot that fills graphics device size. Default = "m" (See `par("pty")`)
`theme`	Character: Run `themes()` for available themes
`palette`	Vector of colors, or Character defining a builtin palette - get options with `rtpalette()`
`autolabel`	Character vector to be used to generate autolabels when using rtlayout with `autolabel = TRUE`.

`par.reset`	Logical: If TRUE, reset `par` setting before exiting.
`pdf.width`	Float: Width in inches for pdf output (if `filename` is set).
`pdf.height`	Float: Height in inches for pdf output.
`filename`	Character: Path to file to save plot. Default = NULL
`...`	Additional arguments to `graphics::barplot`
`legend`	Logical: If TRUE, and input is matrix, draw legend for each case. Note: you may need to adjust `mar` and `legend.inset` if you want to place the legend outside the plot (can use e.g.`legend.inset = c(-.5, 0)`)

Author(s)

E.D. Gennatas

`mplot3_box` `mplot3`*: Boxplot*

Description

Draw boxplots of a vector (single box), data.frame (one box per column) or list (one box per element - good for variable of different length)

Usage

```
mplot3_box(
  x,
  col = NULL,
  alpha = 0.66,
  border = NULL,
  border.alpha = 1,
  group.spacing = 0.25,
  xlim = NULL,
  ylim = NULL,
  xlab = NULL,
  ylab = NULL,
  boxwex = NULL,
  staplewex = 0.5,
  horizontal = FALSE,
  main = NULL,
  groupnames = NULL,
  xnames = NULL,
  xnames.at = NULL,
  xnames.y = NULL,
  xnames.font = 1,
  xnames.adj = NULL,
  xnames.pos = NULL,
  xnames.srt = NULL,
```

```
  order.by.fn = NULL,
  legend = FALSE,
  legend.names = NULL,
  legend.position = "topright",
  legend.inset = c(0, 0),
  mar = NULL,
  oma = rep(0, 4),
  pty = "m",
  yaxis = TRUE,
  ylim.pad = 0,
  theme = rtTheme,
  labelify = TRUE,
  autolabel = letters,
  na.rm = TRUE,
  palette = rtPalette,
  par.reset = TRUE,
  pdf.width = 6,
  pdf.height = 6,
  filename = NULL,
  ...
)
```

Arguments

x	Vector, data.frame or list: Each data.frame column or list element will be drawn as a box
col	Vector of colors to use
alpha	Numeric: col transparency
border	Color for lines around boxes
border.alpha	Numeric: border transparency
group.spacing	Numeric: Spacing between groups of boxes (when input is data.frame or list)
xlim	Float vector, length 2: x-axis limits
ylim	Float vector, length 2: y-axis limits
xlab	Character: x-axis label
ylab	Character: y-axis label
boxwex	Numeric: Scale factor for box width. Default = .5
staplewex	Numeric: max and min line ("staple") width proportional to box. Default = .5
horizontal	Logical: If TRUE, draw horizontal boxplot(s).
main	Character: Plot title
groupnames	Character vector: Group names
xnames	Character vector: Names for individual boxes
xnames.at	Numeric: Position of xnames

order.by.fn	Character: "mean", "median" or any function that outputs a single number: E stimate function on each vector and order boxes (when input is data.frame or list) by ascending order. Default = NULL, i.e. no reordering
mar	Float, vector, length 4: Margins; see `par("mar")`
oma	Float, vector, length 4: Outer margins; see `par("oma")`
pty	Character: "s" gives a square plot; "m" gives a plot that fills graphics device size. Default = "m" (See `par("pty")`)
theme	Character: Run `themes()` for available themes
autolabel	Character vector to be used to generate autolabels when using rtlayout with `autolabel = TRUE`.
na.rm	Logical: If TRUE, remove NA values, otherwise function will give error. Default = TRUE
palette	Vector of colors, or Character defining a builtin palette - get options with `rtpalette()`
par.reset	Logical: If TRUE, reset `par` setting before exiting.
pdf.width	Float: Width in inches for pdf output (if `filename` is set).
pdf.height	Float: Height in inches for pdf output.
filename	Character: Path to file to save plot. Default = NULL
...	Additional arguments to `graphics::boxplot`

Details

Note that argument `xnames` refers to the x-axis labels below each box. If not specified, these are inferred from the input when possible. Argument `xlab` is a single label for the x-axis as per usual and often omitted if `xnames` suffice.

Author(s)

E.D. Gennatas

Examples

```
## Not run:
## vector
x <- rnorm(500)
mplot3_box(x)

## data.frame - each column one boxplot
x <- data.frame(alpha = rnorm(50), beta = rnorm(50), gamma = rnorm(50))
mplot3_box(x)

## list of vectors - allows different length vectors
x <- list(alpha = rnorm(50),
          beta = rnorm(80, 4, 1.5),
          gamma = rnorm(30, -3, .5))
mplot3_box(x)

## grouped boxplots: input a list of lists. outer list: groups; inner lists: matched data vectors
```

```
x <- list(Cases = list(Weight = rnorm(50), Temperature = rnorm(45, 1)),
          Controls = list(Weight = rnorm(80), Temperature = rnorm(72)))
mplot3_box(x)

## End(Not run)
```

---

mplot3_conf *Plot confusion matrix*

Description

Plots confusion matrix and classification metrics

Usage

```
mplot3_conf(
  object,
  main = "auto",
  xlab = "Reference",
  ylab = "Predicted",
  plot.metrics = TRUE,
  mod.name = NULL,
  oma = c(0, 0, 0, 0),
  dim.main = NULL,
  dim.lab = 1,
  dim.in = 4,
  dim.out = -1,
  font.in = 2,
  font.out = 1,
  cex.main = 1.2,
  cex.in = 1.2,
  cex.lab = 1.2,
  cex.lab2 = 1.2,
  cex.lab3 = 1,
  cex.out = 1,
  col.main = "auto",
  col.lab = "auto",
  col.text.out = "auto",
  col.bg = "auto",
  col.bg.out1 = "auto",
  col.bg.out2 = "auto",
  col.text.hi = "auto",
  col.text.lo = "auto",
  show.ba = TRUE,
  theme = getOption("rt.theme", "white"),
  mid.col = "auto",
  hi.color.pos = "#18A3AC",
```

```
  hi.color.neg = "#C23A70",
  autolabel = letters,
  par.reset = TRUE,
  pdf.width = 7,
  pdf.height = 7,
  filename = NULL,
  ...
)
```

Arguments

object	Either a classification rtMod, or a table/matrix/data.frame confusion matrix where rows are the reference classes and columns are the predicted classes.
main	Character: Plot title.
xlab	Character: x-axis label.
ylab	Character: y-axis label.
plot.metrics	Logical: If TRUE, draw classification metrics next to confusion matrix.
mod.name	Character: Name of the algorithm used to make predictions. If NULL, will look for object$mod.name.
oma	Numeric, vector, length 4: Outer margins.
dim.main	Numeric: Height for title.
dim.lab	Numeric: Height for labels.
dim.in	Numeric: Height/Width for confusion matrix cells.
dim.out	Numeric: Height for metrics cells. Default = -1, which autoadjusts depending on number of output classes.
font.in	Integer: The font parameter for confusion matrix cells.
font.out	Integer: The font parameter for metrics cells.
cex.main	Numeric: The cex parameter for the main title.
cex.in	Numeric: The cex parameter for confusion matrix cells.
cex.lab	Numeric: The cex parameter for first line of label cells.
cex.lab2	Numeric: The cex parameter for second line of label cells.
cex.lab3	Numeric: The cex parameter for classification metrics.
cex.out	Numeric: The cex parameter for metrics cells.
col.main	Color for title. Default = "auto", determined by theme.
col.lab	Color for labels. Default = "auto", determined by theme.
col.text.out	Color for metrics cells' text. Default = "auto", determined by theme.
col.bg	Color for background. Default = "auto", determined by theme.
col.bg.out1	Color for metrics cells' background (row1). Default = "auto", determined by theme.
col.bg.out2	Color for metrics cells' background (row2). Default = "auto", determined by theme.

col.text.hi	Color for high confusion matrix values. Default = "auto", determined by theme.
col.text.lo	Color for low confusion matrix values. Default = "auto", determined by theme.
show.ba	Logical: If TRUE, show Balanced Accuracy at bottom right corner.
theme	Character: "light", or "dark". Set to options("rt.theme"), if set, otherwise "light"
mid.col	Color: The mid color for the confusion matrix. Default = "auto", determined by theme.
hi.color.pos	Color: The hi color for correct classification.
hi.color.neg	Color: The hi color for missclassification.
autolabel	Character vector to be used to generate autolabels when using rtlayout with autolabel = TRUE.
par.reset	Logical: If TRUE, reset par before exit.
pdf.width	Numeric: PDF width, if filename is set.
pdf.height	Numeric: PDF height, if filename is set.
filename	Character: If specified, save plot to this path.
...	Additional arguments passed to theme.

Details

This function uses its multiple cex args instead of the theme's cex parameter

Value

List of metrics, invisibly

Author(s)

E.D. Gennatas

Examples

```
## Not run:
true <- c("alpha", "alpha", "alpha", "alpha", "beta", "beta", "beta", "beta")
predicted <- c("alpha", "alpha", "alpha", "beta", "beta", "alpha", "alpha", "beta")
mplot3_conf(table(predicted, true))

## End(Not run)
```

`mplot3_confbin` *Plot extended confusion matrix for binary classification*

Description

Plots an extended confusion matrix using mplot3_img

Usage

```
mplot3_confbin(
  object,
  main = NULL,
  xlab = "True",
  ylab = "Estimated",
  mod.name = NULL,
  mar = c(4, 5, 4, 3),
  dim.lab = 1,
  dim.in = 4,
  dim.out = 2,
  font.in = 2,
  font.out = 2,
  cex.in = 1.2,
  cex.lab = 1.2,
  cex.lab2 = 1,
  cex.out = 1,
  col.text.out = "white",
  col.bg.out = "gray50",
  theme = "light",
  mid.color = NULL,
  hi.color.pos = "#18A3AC",
  hi.color.neg = "#716FB2",
  par.reset = TRUE,
  pdf.width = 8.7,
  pdf.height = 8.7,
  filename = NULL,
  ...
)
```

Arguments

`object`	Either 1. a classification `rtMod`, b. a `caret::confusionMatrix` object, or c. a matrix / data.frame / table
`main`	Character: Plot title
`xlab`	Character: x-axis label
`ylab`	Character: y-axis label

mod.name	Character: Name of the algorithm used to make predictions. If NULL, will look for object$mod.name. Default = NULL
mar	Numeric, vector, length 4: Overall margins
dim.lab	Float: Height for labels
dim.in	Float: Width and height for confusion matrix cells
dim.out	Float: Height for metrics cells
font.in	Integer: The font parameter for confusion matrix cells
font.out	Integer: The font parameter for metrics cells
cex.in	Float: The cex parameter for confusion matrix cells
cex.lab	Float: The cex parameter for first line of label cells
cex.lab2	Float: The cex parameter for second line of label cells
cex.out	Float: The cex parameter for metrics cells
col.text.out	Color for metrics cells' text
col.bg.out	Color for metrics cells' background
theme	Character: "light", or "dark"
mid.color	Color: The mid color for the confusion matrix. Default = "white" for theme = "light", "black" for "dark"
hi.color.pos	Color: The hi color for correct classification.
hi.color.neg	Color: The hi color for missclassification
par.reset	Logical: If TRUE, reset par before exit. Default = TRUE
pdf.width	Float: PDF width, if filename is set
pdf.height	Float: PDF height, if filename is set
filename	Character: If specified, save plot to this path. Default = NULL
...	Not used

Value

List of metrics, invisibly

Author(s)

E.D. Gennatas

mplot3_decision   mplot3: *Decision boundaries*

Description

Plot classification decision boundaries of rtemis models

Usage

```
mplot3_decision(
  rtmod,
  data,
  vars = c(1, 2),
  dots.per.axis = 100,
  bg.cex = 0.5,
  bg.alpha = 0.4,
  bg.pch = 15,
  par.reset = TRUE,
  theme = "white",
  col = c("#18A3AC", "#F48024"),
  contour.col = "black",
  contour.lwd = 0.1,
  point.pch = c(3, 4),
  point.alpha = 1
)
```

Arguments

rtmod	rtemics trained model
data	Matrix / data frame of features; last column is class
vars	Integer vector, length 2: Index of features (columns of x) to use to draw decision boundaries. Default = c(1, 2)
dots.per.axis	Integer: Draw a grid with this many dots on each axis. Default = 100
bg.cex	Float: Point cex for background / decision surface. Default = .5
bg.alpha	Float: Point alpha for background / decision surface. Default = .2
bg.pch	Integer vector: pch for background / decision surface. Default = c(3, 4)
par.reset	Logical: If TRUE, reset par before exiting. Default = TRUE
theme	Character: Theme for mplot3_xy, "light" or "dark". Default = "light'
col	Color vector for classes. Default = ucsfPalette
contour.col	Color for decision boundary. Default = "black"
contour.lwd	Float: Line width for decision boundary. Default = .3
point.pch	Integer: pch for data points. Default = c(3, 4)
point.alpha	Float: Alpha for data points. Default = 1

Details

If data has more than 2 variables, any variable not selected using `vars` will be fixed to their mean Underlying model (e.g. `randomForest`, `rpart`, etc) must support standard R predict format for classification: `predict(model, newdata, type = "class")`

Value

Predicted labels for background grid (invisibly)

Author(s)

E.D. Gennatas

Examples

```
## Not run:
dat <- as.data.frame(mlbench::mlbench.2dnormals(200))
mod.cart <- s_CART(dat)
mod.rf <- s_RF(dat)
mplot3_decision(mod.cart, dat)
mplot3_decision(mod.rf, dat)

## End(Not run)
```

---

mplot3_fit *True vs. Fitted plot*

Description

An `mplot3_xy` wrapper with defaults for plotting a learner's performance

Usage

```
mplot3_fit(
  x,
  y,
  fit = "lm",
  se.fit = TRUE,
  fit.error = TRUE,
  axes.equal = TRUE,
  diagonal = TRUE,
  theme = rtTheme,
  marker.col = NULL,
  fit.col = NULL,
  pty = "s",
  fit.legend = FALSE,
  mar = NULL,
  ...
)
```

Arguments

x	Numeric vector: True values
y	Numeric vector: Predicted values
fit	Character: **rtemis** model to calculate `y ~ x` fit. Options: see `select_learn()` Can also be Logical, which will give a GAM fit if TRUE. If you specify "NLA", the activation function should be passed as a string.
se.fit	Logical: If TRUE, draw the standard error of the fit
fit.error	Logical: If TRUE: draw fit error annotation. Default = NULL, which results in TRUE, if fit is set
axes.equal	Logical: Should axes be equal? Defaults to FALSE
diagonal	Logical: If TRUE, draw diagonal line.
theme	Character: Run `themes()` for available themes
marker.col	Color for marker
fit.col	Color: Color of the fit line.
pty	Character: "s" for square plot, "m" to fill device. Default = "s"
fit.legend	Logical: If TRUE, show fit legend
mar	Float, vector, length 4: Margins; see `par("mar")`
...	Additional argument to be passed to mplot3_conf (classification) or mplot3_xy (regression)

Author(s)

EDG

---

mplot3_fret — mplot3: *Guitar Fretboard*

Description

Draw color-coded notes on a guitar fretboard for standard E-A-D-G-B-e tuning

Usage

```
mplot3_fret(
  theme = rtTheme,
  useSharps = FALSE,
  strings.col = "auto",
  frets.col = "auto",
  inlays = TRUE,
  inlays.col = "auto",
  inlays.cex = 2,
  par.reset = TRUE,
  ...
)
```

Arguments

theme	Character: "light" or "dark"
useSharps	Logical: If TRUE, draw sharp instead of flat notes. Default = FALSE
strings.col	Color for strings
frets.col	Color for frets
inlays	Logical: Draw fretboard inlays. Default = TRUE
inlays.col	Color for inlays
inlays.cex	Numeric: Character expansion factor for inlays. Default = 2
par.reset	Logical: If TRUE, reset par before exit
...	Additional arguments to theme

Details

Plot is very wide and short. Adjust plot window accordingly. Practice every day.

Author(s)

E.D. Gennatas

---

mplot3_graph *Plot* igraph *networks*

Description

Plot igraph networks

Usage

```
mplot3_graph(
  net,
  vertex.size = 12,
  vertex.col = NULL,
  vertex.alpha = 0.33,
  vertex.label.col = NULL,
  vertex.label.alpha = 0.66,
  vertex.frame.col = NA,
  vertex.label = NULL,
  vertex.shape = "circle",
  edge.col = NULL,
  edge.alpha = 0.2,
  edge.curved = 0.35,
  edge.width = 2,
  layout = c("fr", "dh", "drl", "gem", "graphopt", "kk", "lgl", "mds", "sugiyama"),
  coords = NULL,
```

```
  layout_params = list(),
  cluster = NULL,
  groups = NULL,
  cluster_params = list(),
  cluster_mark_groups = TRUE,
  mark.col = NULL,
  mark.alpha = 0.3,
  mark.border = NULL,
  mark.border.alpha = 1,
  cluster_color_vertices = FALSE,
  theme = rtTheme,
  theme_extra_args = list(),
  palette = rtPalette,
  mar = rep(0, 4),
  par.reset = TRUE,
  filename = NULL,
  pdf.width = 6,
  pdf.height = 6,
  verbose = TRUE,
  ...
)
```

Arguments

net	igraph network
vertex.size	Numeric: Vertex size
vertex.col	Color for vertices
vertex.alpha	Numeric: Transparency for vertex.col
vertex.label.col	Color for vertex labels
vertex.label.alpha	Numeric: Transparency for vertex.label.col
vertex.frame.col	Color for vertex border (frame)
vertex.label	Character vector: Vertex labels. Default = NULL, which will keep existing names in net if any. Set to NA to avoid printing vertex labels
vertex.shape	Character: Vertex shape. See igraph::plot.igraph("vertex.shape"). Default = "circle"
edge.col	Color for edges
edge.alpha	Numeric: Transparency for edges. Default = .2
edge.curved	Numeric: Curvature of edges. Default = .35
edge.width	Numeric: Edge thickness
layout	Character: one of: "fr", "dh", "drl", "gem", "graphopt", "kk", "lgl", "mds", "sugiyama", corresponding to all the available layouts in **igraph**
coords	Output of precomputed **igraph** layout. If provided, layout is ignored

layout_params List of parameters to pass to layout function

cluster Character: one of: "edge_betweenness", "fast_greedy", "infomap", "label_prop", "leading_eigen", "louvain", "optimal", "spinglass", "walktrap", corresponding to all the available **igraph** clustering functions

groups Output of precomputed **igraph** clustering. If provided, cluster is ignored

cluster_params List of parameters to pass to cluster function

cluster_mark_groups
Logical: If TRUE, draw polygons to indicate clusters, if groups or cluster defined

mark.col Colors, one per group for polygon surrounding cluster. Note: You won't know the number of groups unless they are precomputed. The colors will be recycled as needed.

mark.alpha Float [0, 1]: Transparency for mark.col.

mark.border Colors, similar to mark.col for border

mark.border.alpha
Float [0, 1]: Transparency for mark.border.

cluster_color_vertices
Logical: If TRUE, color vertices by cluster membership.

theme **rtemis** theme to use

theme_extra_args
List of extra arguments to pass to the theme function defined by theme. This argument is used when the extra args (...) are passed the plotting function (in this case igraph::plot.igraph) and not to the theme function

palette Vector of colors, or Character defining a builtin palette - get options with rtpalette()

mar Numeric vector, length 4: par's margin argument

par.reset Logical: If TRUE, reset par before exiting.

filename Character: If provided, save plot to this filepath

pdf.width Float: Width in inches for pdf output (if filename is set).

pdf.height Float: Height in inches for pdf output.

verbose Logical, If TRUE, print messages to console.

... Extra arguments to pass to igraph::plot.igraph()

Author(s)

E.D. Gennatas

mplot3_harmonograph *Plot a harmonograph*

Description

Plot a harmonograph

Usage

```
mplot3_harmonograph(
  steps = seq(1, 500, by = 0.01),
  seed = NULL,
  col = "white",
  alpha = 0.2,
  bg = "black",
  lwd = 1,
  text = NULL,
  text.side = 1,
  text.line = -1,
  text.adj = 0,
  text.padj = 0,
  text.col = NULL,
  mar = c(0, 0, 0, 0),
  oma = c(0, 0, 0, 0),
  xlim = NULL,
  ylim = NULL,
  new = FALSE,
  par.reset = TRUE
)
```

Arguments

`steps`	Float, vector
`seed`	Integer
`col`	Line color. Default = "white"
`alpha`	Alpha for line color `col`. Default = .2
`bg`	Color for background. Default = "black"
`lwd`	Float: Line width
`text`	Character: Text you want printed along with the harmonograph. Default = NULL
`text.side`	Integer 1, 2, 3, 4: `side` argument for `mtext`
`text.line`	Float: `line` argument for `mtext`
`text.adj`	Float: `adj` argument for `mtext`
`text.padj`	Float: `padj` argument for `mtext`

text.col	Color: Text color. Default is same as col
mar	Float vector, length 4: Plot margins. (par's mar argument)
oma	Float vector, length 4: Outer margins. (par's oma argument)
xlim	Float vector, length 2: x-axis limits
ylim	Float vector, length 2: y-axis limits
new	Logical. If TRUE, do not clear plot before drawing
par.reset	Logical. If TRUE, reset par before exit

Details

Unless you define a seed, each graph will be random. Try different seeds if you want to reproduce your graphs. Some seeds to try: 9, 17, 26, 202, 208, ...

Author(s)

E.D. Gennatas

---

mplot3_heatmap   mplot3 *Heatmap (*image*; modified* heatmap*)*

Description

Customized heatmap with optional colorbar

Usage

```
mplot3_heatmap(
  x,
  colorGrad.n = 101,
  colorGrad.col = NULL,
  lo = "#18A3AC",
  lomid = NULL,
  mid = NULL,
  midhi = NULL,
  hi = "#F48024",
  space = "rgb",
  theme = getOption("rt.theme", "white"),
  colorbar = TRUE,
  cb.n = 21,
  cb.title = NULL,
  cb.cex = NULL,
  cb.title.cex = 1,
  cb.mar = NULL,
  Rowv = TRUE,
  Colv = TRUE,
```

```
  distfun = dist,
  hclustfun = hclust,
  reorderfun = function(d, w) reorder(d, w),
  add.expr,
  symm = FALSE,
  revC = identical(Colv, "Rowv"),
  scale = "none",
  na.rm = TRUE,
  margins = NULL,
  group.columns = NULL,
  group.legend = !is.null(group.columns),
  column.palette = rtPalette,
  group.rows = NULL,
  row.palette = rtPalette,
  ColSideColors,
  RowSideColors,
  cexRow = 0.2 + 1/log10(nr),
  cexCol = 0.2 + 1/log10(nc),
  labRow = NULL,
  labCol = NULL,
  labCol.las = NULL,
  main = "",
  main.adj = 0,
  main.line = NA,
  xlab = NULL,
  ylab = NULL,
  xlab.line = NULL,
  ylab.line = NULL,
  keep.dendro = FALSE,
  trace = 0,
  zlim = NULL,
  autorange = TRUE,
  autolabel = letters,
  filename = NULL,
  par.reset = TRUE,
  pdf.width = 7,
  pdf.height = 7,
  ...
)
```

Arguments

x	Input matrix
colorGrad.n	Integer: Number of distinct colors to generate using colorGrad. Default = 101
colorGrad.col	Character: the colors argument of colorGrad: Character: Acts as a shortcut to defining lo, mid, etc for a number of defaults: "french", "penn", "grnblkred"
lo	Color for low end
lomid	Color for low-mid

`mid`	Color for middle of the range or "mean", which will result in colorOp(c(lo, hi), "mean"). If mid = NA, then only lo and hi are used to create the color gradient.
`midhi`	Color for middle-high
`hi`	Color for high end
`space`	Character: Which colorspace to use. Option: "rgb", or "Lab". Default = "rgb". Recommendation: If mid is "white" or "black" (default), use "rgb", otherwise "Lab"
`theme`	Character: Defaults to option "rt.theme", if set, otherwise "light"
`colorbar`	Logical: If TRUE, plot colorbar next to heatmap. Default = TRUE
`cb.n`	Integer: Number of steps in colorbar. Default = 21, which gives 10 above and 10 below midline. If midline is zero, this corresponds to 10 percent increments / decrements
`cb.title`	Character: Title for the colorbar. Default = NULL
`cb.cex`	Float: Character expansion (`cex`) for colobar. Default = 1
`cb.title.cex`	Float: `cex` for colorbar title. Default = 1
`cb.mar`	Float, vector, length 4: Margins for colorbar. (passed to colorGrad's `cb.add.mar`). Default set automatically
`Rowv`	Logical OR a dendrogram OR integer vector that determines index for reordering OR NA to suppress. Default = TRUE
`Colv`	See `Rowv`
`distfun`	Function: used to compute the distance/dissimilarity matrix between rows and columns. Default = `dist`
`hclustfun`	Function: used to determined hierarchical clustering when `Rowv` or `Colv` are not dendrograms. Default = `hclust` (Should take as argument a result of distfun and return an object to which `as.dendrogram` can be applied)
`reorderfun`	Function (d, w): function of dendrogram and weights that determines reordering of row and column dendrograms. Default uses `reorder.dendrogram`
`add.expr`	Expression: will be evaluated after the call to `image`. Can be used to add components to the plot
`symm`	Logical: If TRUE, treat `x` symmetrically. Can only be used if `x` is square
`revC`	Logical: If TRUE, reverse column order for plotting. Default = TRUE, if Rowv and Colv are identical
`scale`	Character: "row", "column", or "none". Determines whether values are centered and scaled in either the row or column direction. Default = "none"
`na.rm`	Logical: If TRUE, NAs are removed. Default = TRUE
`margins`	Float, vector, length 2: bottom and right side margins. Automatically determined by length of variable names
`ColSideColors`	Color, vector, length = ncol(x): Colors for a horizontal side bar to annotate columns of `x`
`RowSideColors`	Color, vector, length = nrow(x): Like `ColSideColors`, but for rows
`cexRow`	Float: `cex.axis` for rows

cexCol	Float: cex.axis for columns
labRow	Character, vector: Row labels to use. Default = rownames(x)
labCol	Character, vector: Column labels to use. Default = colnames(x)
labCol.las	Integer 0:3: par's las argument. Default set by length of labCol
main	Character: Plot title
main.adj	Float: par's adj argument for title
main.line	Float: title's line argument
xlab	Character: x-axis label
ylab	Character: y-axis label
xlab.line	Float: mtext's line argument for x-axis label
ylab.line	Float: mtext's line argument for y-axis label
keep.dendro	Logical: If TRUE, dedrogram is returned invisibly. Default = FALSE
trace	Integer: If > 0, print diagnostic messages to console. Default = 0
zlim	Float, vector, length 2: Passed to graphics::image. Default = +/- max(abs(x)) if autorange = TRUE, otherwise = range(x).
autorange	Logical: See zlim
filename	Character: If provided, save heatmap to file. Default = NULL
par.reset	Logical: If TRUE, reset par before exit. Default = TRUE
pdf.width	Float: Width of PDF output, if filename is set
pdf.height	Float: Height of PDF output, if filename is set
...	Additional arguments passed to graphics::image

Details

The main difference from the original stats::heatmap is the addition of a colorbar on the side. This is achieved with colorGrad. Other differences:

- Dendrograms are not drawn by default. Set Rowv = T and Colv = T to get them.
- Column labels are only drawn perpendicular to the x-axis if any one is longer than two characters. Otherwise, the arguments are the same as in stats::heatmap

Author(s)

E.D. Gennatas modified from original stats::heatmap by Andy Liaw, R. Gentleman, M. Maechler, W. Huber

Examples

```
## Not run:
x <- rnormmat(200, 20)
xcor <- cor(x)
mplot3_heatmap(xcor)

## End(Not run)
```

mplot3_img *Draw image (False color 2D)*

Description

Draw a bitmap from a matrix of values.

Usage

```
mplot3_img(
  z,
  as.mat = TRUE,
  col = NULL,
  xnames = NULL,
  xnames.y = 0,
  ynames = NULL,
  main = NULL,
  main.adj = 0,
  x.axis.side = 3,
  y.axis.side = 2,
  x.axis.line = -0.5,
  y.axis.line = -0.5,
  x.axis.las = 0,
  y.axis.las = 1,
  x.tick.labs.adj = NULL,
  y.tick.labs.adj = NULL,
  x.axis.font = 1,
  y.axis.font = 1,
  xlab = NULL,
  ylab = NULL,
  xlab.adj = 0.5,
  ylab.adj = 0.5,
  xlab.line = 1.7,
  ylab.line = 1.7,
  xlab.padj = 0,
  ylab.padj = 0,
  xlab.side = 1,
  ylab.side = 2,
  main.col = NULL,
  axlab.col = NULL,
  axes.col = NULL,
  labs.col = NULL,
  tick.col = NULL,
  cell.lab.hi.col = NULL,
  cell.lab.lo.col = NULL,
  cex = 1.2,
  cex.ax = NULL,
```

```
  cex.x = NULL,
  cex.y = NULL,
  zlim = NULL,
  autorange = TRUE,
  pty = "m",
  mar = NULL,
  asp = NULL,
  ann = FALSE,
  axes = FALSE,
  cell.labs = NULL,
  cell.labs.col = NULL,
  cell.labs.autocol = TRUE,
  bg = NULL,
  theme = getOption("rt.theme", "white"),
  autolabel = letters,
  filename = NULL,
  file.width = NULL,
  file.height = NULL,
  par.reset = TRUE,
  ...
)
```

Arguments

z	Input matrix
as.mat	Logical: If FALSE, rows and columns of z correspond to x and y coordinates accoridngly. This is the `image` default. If TRUE (default), resulting image's cells will be in the same order as values appear when you print z in the console. This is `t(apply(z, 2, rev))`. In this case, you can think of z as a table of values you want to pictures with colors. For example, you can convert a correlation table to a figure. In this case, you might want to add `cell.labs` with the values. Consider first using ddSci.
col	Colors to use. Defaults to `colorGrad(100)`
cell.labs	Matrix of same dimensions as z (Optional): Will be printed as strings over cells
cell.labs.col	Color for `cell.labs`. If NULL, the upper and lower quartiles will be set to "white", the rest "black".
bg	Background color
filename	String (Optional): Path to file where image should be saved. R-supported extensions: ".pdf", ".jpeg", ".png", ".tiff".
file.width	Output Width in inches
file.height	Output height in inches
par.reset	Logical: If TRUE, par will be reset to original settings before exit. Default = TRUE
...	Additional arguments to be passed to `graphics::image`

Details

This is also a good way to plot a large heatmap. This function calls `image` which is a lot faster than drawing heatmaps

Author(s)

E.D. Gennatas

`mplot3_laterality` *Laterality scatter plot*

Description

Laterality scatter plot

Usage

```
mplot3_laterality(
  x,
  regionnames,
  main = NULL,
  ylab = "Left to Right",
  summary.fn = "median",
  summary.lty = 1,
  summary.lwd = 2.5,
  summary.col = NULL,
  arrowhead.length = 0.075,
  deltas = TRUE,
  line.col = theme$fg,
  line.alpha = 0.25,
  lty = 1,
  lwd = 0.3,
  ylim = NULL,
  theme = rtTheme,
  labelify = TRUE,
  autolabel = letters,
  mar = NULL,
  oma = rep(0, 4),
  pty = "m",
  palette = rtPalette,
  par.reset = TRUE,
  pdf.width = 6,
  pdf.height = 6,
  filename = NULL,
  ...
)
```

Arguments

x	data.frame or data.table which includes columns with ROI names ending in "_L" or "_R"
regionnames	Character, vector: Regions to plot. For example, if regionnames contains "Ant_Insula", x must contain columns Ant_Insula_L and Ant_Insula_R
main	Character: Plot title
ylab	Character: y-axis label
summary.fn	Character: Name of function to summarize left and right values. Default = "median"
summary.lty	Integer: line type for summary arrows
summary.lwd	Float: line width for summary arrows
summary.col	Color for summary arrows
arrowhead.length	Float: arrowhead length in inches. Default = .075
deltas	Logical, If TRUE, show summary statistics. Default = TRUE
line.col	Color for individual cases' lines
line.alpha	Float: transparency for individual lines
lty	Integer: Line type for individual lines. Default = 1
lwd	Float: Line width for individual lines. Default = .3
ylim	Float, vector, length 2: y-axis limits
theme	Character: Run themes() for available themes
labelify	Logical: If TRUE, labelify regionnames
autolabel	Character vector to be used to generate autolabels when using rtlayout with autolabel = TRUE.
mar	Float, vector, length 4: Margins; see par("mar")
oma	Float, vector, length 4: Outer margins; see par("oma")
pty	Character: "s" gives a square plot; "m" gives a plot that fills graphics device size. Default = "m" (See par("pty"))
palette	Vector of colors, or Character defining a builtin palette - get options with rtpalette()
par.reset	Logical: If TRUE, reset par setting before exiting.
pdf.width	Float: Width in inches for pdf output (if filename is set).
pdf.height	Float: Height in inches for pdf output.
filename	Character: Path to file to save plot. Default = NULL
...	Additional arguments to be passed to theme function

Author(s)

E.D. Gennatas

mplot3_lolli   mplot3 *Lollipop Plot*

Description

mplot3 Lollipop Plot

Usage

```
mplot3_lolli(
  x,
  order.on.x = TRUE,
  plot.top = 1,
  orientation = c("horizontal", "vertical"),
  xnames = NULL,
  points = TRUE,
  segments = TRUE,
  main = NULL,
  col = NULL,
  cex = 1.2,
  matching.segment.col = FALSE,
  segment.alpha = 0.333,
  lty = 3,
  lwd = 2,
  theme = rtTheme,
  palette = rtPalette,
  autolabel = letters,
  par.reset = TRUE,
  pdf.width = 6,
  pdf.height = 6,
  mar = c(2.5, 3, 2, 1),
  pty = "m",
  pch = 16,
  x.axis.at = NULL,
  y.axis.at = NULL,
  xlab = NULL,
  ylab = NULL,
  label.las = 1,
  label.padj = 0.5,
  xaxs = "r",
  yaxs = "r",
  xlab.adj = 0.5,
  ylab.adj = 0.5,
  filename = NULL,
  ...
)
```

Arguments

x	Float, vector: Input data
order.on.x	Logical: If TRUE, order by value of x. Default = TRUE
plot.top	Float or Integer: If <= 1, plot this percent highest absolute values, otherwise plot this many top values. i.e.: plot.top = .2 will print the top 20% highest values, and plot.top = 20 will plot the top 20 highest values
xnames	Character, vector: Names of x
main	Character: Main title
col	Color, vector: Lollipop color
cex	Float: Character expansion factor for points. Default = 1.2
matching.segment.col	Logical: If TRUE, color line segments using col, i.e. same as
segment.alpha	Float: Transparency for line segments. Default = .5 points. Default = FALSE, in which case they are colored with theme$fg
lty	Integer: Line type for segment segments. See par("lty") Default = 1
lwd	Float: Width for segment segments. See par("lty") Default = 1
theme	Character: Run themes() for available themes
palette	Vector of colors, or Character defining a builtin palette - get options with rtpalette()
autolabel	Character vector to be used to generate autolabels when using rtlayout with autolabel = TRUE.
par.reset	Logical: If TRUE, reset par setting before exiting.
pdf.width	Float: Width in inches for pdf output (if filename is set).
pdf.height	Float: Height in inches for pdf output.
mar	Float, vector, length 4: Margins; see par("mar")
pty	Character: "s" gives a square plot; "m" gives a plot that fills graphics device size. Default = "m" (See par("pty"))
pch	Integer: Point character.
x.axis.at	Float, vector: x coordinates to place tick marks. Default = NULL, determined by graphics::axis aautomatically
y.axis.at	As x.axis.at for y-axis
xlab	Character: x-axis label
ylab	Character: y-axis label
xaxs	Character: "r": Extend plot x-axis limits by 4% on either end; "i": Use exact x-axis limits.
yaxs	Character: as xaxs for the y-axis.
xlab.adj	Float: adj for xlab (See par("adj"))
ylab.adj	Float: adj for ylab (See par("adj"))
filename	Character: Path to file to save plot. Default = NULL
...	Additional arguments to be passed to theme function

Author(s)

E.D. Gennatas

Examples

```
## Not run:
x <- rnorm(12)
mplot3_lolli(x)
# a "rounded" barplot
mplot3_lolli(x, segments = T, points = F,
             lty = 1, matching.segment.col = T,
             lwd = 10, segment.alpha = 1)

## End(Not run)
```

---

mplot3_missing *Plot missingness*

Description

Plot missingness

Usage

```
mplot3_missing(
  x,
  feat.names = NULL,
  case.names = NULL,
  main = NULL,
  col.missing = "#FE4AA3",
  show = c("percent", "total"),
  names.srt = 90,
  case.names.x = 0.25,
  case.names.every = NULL,
  theme = rtTheme,
  alpha = 1,
  mar = c(3, 3.5, 5.5, 1),
  oma = c(0.5, 0.5, 0.5, 0.5),
  par.reset = TRUE,
  ...
)
```

Arguments

x	Data matrix or data.frame
feat.names	Character: Feature names. Defaults to `colnames(x)`
case.names	Character: Case names. Defaults to `rownames(x)`

main	Character: Main title
col.missing	Color for missing cases.
show	Character: "percent" or "total". Show percent missing or total missing per column on the x-axis
names.srt	Numeric: Angle of feature names in degrees.
case.names.x	Numeric: x position of case names
case.names.every	Numeric: Show case names every this many cases
theme	Character: Run `themes()` for available themes
alpha	Numeric: Multiply theme's `fg` color by this amount
mar	Float, vector, length 4: Margins; see `par("mar")`
oma	Float, vector, length 4: Outer margins; see `par("oma")`
par.reset	Logical: If TRUE, reset `par` setting before exiting.
...	Additional arguments to be passed to theme function

Examples

```
## Not run:
dat <- iris
dat[c(1, 5, 17:20, 110, 115, 140), 1] <-
dat[c(12, 15, 55, 73, 100:103), 2] <-
dat[sample(1:150, 25), 4] <- NA
mplot_missing(dat)

## End(Not run)
```

`mplot3_mosaic` *Mosaic plot*

Description

Plots a mosaic plot using `graphics::mosaicplot`

Usage

```
mplot3_mosaic(
  x,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  border = FALSE,
  theme = rtTheme,
  theme.args = list(),
  palette = rtPalette,
```

```
  mar = NULL,
  oma = rep(0, 4),
  par.reset = TRUE,
  new = FALSE,
  autolabel = letters,
  filename = NULL,
  pdf.width = 5,
  pdf.height = 5,
  ...
)
```

Arguments

`x`	contingency table, e.g. output of `table()`
`main`	Character: Main title
`xlab`	Character: x-axis label
`ylab`	Character: y-axis label
`border`	Color vector for cell borders or FALSE to turn off. Default = FALSE
`theme`	Character: Run `themes()` for available themes
`theme.args`	List of arguments to pass to `theme`. Optional, same args can be passed to theme function
`palette`	Vector of colors, or Character defining a builtin palette - get options with `rtpalette()`
`new`	Logical: If TRUE, add plot to existing plot. See `par("new")`
`filename`	Character: Path to file to save plot. Default = NULL
`pdf.width`	Float: Width in inches for PDF output, if `filename` is defined
`pdf.height`	Float: Height in inches for PDF output, if `filename` is defined

Author(s)

E.D. Gennatas

Examples

```
## Not run:
party <- as.table(rbind(c(762, 327, 468), c(484, 239, 477)))
dimnames(party) <- list(gender = c("F", "M"),
                        party = c("Democrat","Independent", "Republican"))
mplot3_mosaic(party)

## End(Not run)
```

mplot3_pr *mplot3 Precision Recall curves*

Description

Plot Precision Recall curve for a binary classifier

Usage

```
mplot3_pr(
  prob,
  labels,
  f1 = FALSE,
  main = "",
  col = NULL,
  cex = 1.2,
  lwd = 2.5,
  diagonal = FALSE,
  hline.lty = 1,
  hline.lwd = 1,
  hline.col = "red",
  diagonal.lwd = 2.5,
  diagonal.lty = 3,
  group.legend = FALSE,
  annotation = TRUE,
  annotation.side = 3,
  annotation.col = col,
  annot.line = NULL,
  annot.adj = 1,
  annot.font = 1,
  mar = c(2.5, 3, 2.5, 1),
  theme = rtTheme,
  palette = rtPalette,
  par.reset = TRUE,
  verbose = TRUE,
  filename = NULL,
  pdf.width = 5,
  pdf.height = 5
)
```

Arguments

prob	Vector, Float [0, 1]: Predicted probabilities (i.e. c(.1, .8, .2, .9))
labels	Vector, Integer 0, 1: True labels (i.e. c(0, 1, 0, 1))
f1	Logical: If TRUE, annotate the point of maximal F1 score.
main	Character: Plot title.

col	Color, vector: Colors to use for ROC curve(s)
cex	Float: Character expansion factor.
lwd	Float: Line width.
diagonal	Logical: If TRUE, draw diagonal.
hline.lty	Integer: Line type for horizontal line(s)
hline.lwd	Float: Width for horizontal line(s)
hline.col	Color for horizontal line(s)
diagonal.lwd	Float: Line width for diagonal.
diagonal.lty	Integer: Line type for diagonal.
group.legend	Logical
annotation	Character: Add annotation at the bottom right of the plot
annotation.side	Integer: Side of plot to place annotation.
annotation.col	Color: Color of annotation.
annot.line	Numeric: Line number for annotation.
annot.adj	Numeric: Adjustment for annotation.
annot.font	Integer: Font for annotation.
mar	Float, vector, length 4: Margins; see `par("mar")`
theme	Character: Run `themes()` for available themes
palette	Vector of colors, or Character defining a builtin palette - get options with `rtpalette()`
par.reset	Logical: If TRUE, reset `par` setting before exiting.
verbose	Logical: If TRUE, print messages to console.
filename	Path to file: If supplied, plot will be printed to file
pdf.width	Float: Width in inches for pdf output (if `filename` is set).
pdf.height	Float: Height in inches for pdf output.

Value

List with Precision, Recall, and Threshold values, invisibly

Author(s)

E.D. Gennatas

`mplot3_prp` *Plot CART Decision Tree*

Description

Plot output of a regression or classification tree created using `rpart` A wrapper for `rpart.plot::rpart.plot`

Usage

```
mplot3_prp(
  object,
  type = 0,
  extra = "auto",
  branch.lty = 1,
  under = FALSE,
  fallen.leaves = TRUE,
  palette = NULL,
  filename = NULL,
  pdf.width = 7,
  pdf.height = 5,
  ...
)
```

Arguments

`object`	Output of s_CART
`palette`	Color vector

`mplot3_res` `mplot3` *Plot* `resample`

Description

Visualizes resampling output using mplot3_img

Usage

```
mplot3_res(res, col = NULL, mar = NULL, theme = rtTheme, ...)
```

Arguments

`res`	rtemis resample object
`col`	Color vector
`mar`	Numeric vector: image margins
`theme`	rtemis theme
`...`	Additional theme arguments

Details

For resampling with no replacement where each case may be selected 0 or 1 time, 0 is white and 1 is teal For resampling with replacement, 0 is white, 1 is blue, 2 is teal

Author(s)

E.D. Gennatas

Examples

```
## Not run:
x <- rnorm(500)
res <- resample(x)
mplot3_res(res)

## End(Not run)
```

---

mplot3_roc   mplot3 *ROC curves*

Description

Plot ROC curve for a binary classifier

Usage

```
mplot3_roc(
  prob,
  labels,
  method = c("pROC", "rt"),
  type = "TPR.FPR",
  balanced.accuracy = FALSE,
  main = "",
  col = NULL,
  alpha = 1,
  cex = 1.2,
  lwd = 2.5,
  diagonal = TRUE,
  diagonal.lwd = 1,
  diagonal.lty = 1,
  diagonal.col = "red",
  group.legend = FALSE,
  annotation = TRUE,
  annotation.col = col,
  annot.line = NULL,
  annot.adj = 1,
  annot.font = 1,
```

```
  pty = "s",
  mar = c(2.5, 3, 2, 1),
  theme = rtTheme,
  palette = rtPalette,
  verbose = TRUE,
  par.reset = TRUE,
  filename = NULL,
  pdf.width = 5,
  pdf.height = 5
)
```

Arguments

prob	Numeric vector or list of numeric vectors [0, 1]: Predicted probabilities (e.g. c(.1, .8, .2, .9))
labels	Integer vector or list of integer vectors 0, 1: True labels (e.g. c(0, 1, 0, 1))
method	Character: "rt" or "pROC" will use rtROC and `pROC::roc` respectively to get points of the ROC.
type	Character: "TPR.FPR" or "Sens.Spec". Only changes the x and y labels. True positive rate vs. False positive rate and Sensitivity vs. Specificity.
balanced.accuracy	Logical: If TRUE, annotate the point of maximal Balanced Accuracy.
main	Character: Plot title.
col	Color, vector: Colors to use for ROC curve(s)
alpha	Numeric: Alpha transparency for lines
cex	Float: Character expansion factor.
lwd	Float: Line width.
diagonal	Logical: If TRUE, draw diagonal.
diagonal.lwd	Float: Line width for diagonal.
diagonal.lty	Integer: Line type for diagonal.
diagonal.col	Color: Color for diagonal.
group.legend	Logical: If TRUE, print group legend
annotation	Character: Add annotation at the bottom right of the plot
annotation.col	Color: Color for annotation.
annot.line	Numeric: Line position for annotation.
annot.adj	Numeric: Text adjustment for annotation.
annot.font	Integer: Font for annotation.
pty	Character: "s" gives a square plot; "m" gives a plot that fills graphics device size. Default = "m" (See `par("pty")`)
mar	Float, vector, length 4: Margins; see `par("mar")`
theme	Character: Run `themes()` for available themes
palette	Vector of colors, or Character defining a builtin palette - get options with `rtpalette()`

verbose	Logical: If TRUE, print messages to console.
par.reset	Logical: If TRUE, reset par setting before exiting.
filename	Path to file: If supplied, plot will be printed to file
pdf.width	Float: Width in inches for pdf output (if filename is set).
pdf.height	Float: Height in inches for pdf output.

Author(s)

E.D. Gennatas

---

mplot3_surv *mplot3: Survival Plots*

Description

Plots survival step functions using mplot3_xy

Usage

```
mplot3_surv(
  x,
  lty = 1,
  lwd = 2,
  alpha = 1,
  col = NULL,
  mark.censored = TRUE,
  normalize.time = FALSE,
  cex = 1.2,
  xlab = NULL,
  ylab = "Survival",
  main = "Kaplan-Meier curve",
  theme = rtTheme,
  palette = rtPalette,
  plot.error = FALSE,
  error.lty = 2,
  error.alpha = 0.5,
  group.legend = NULL,
  group.title = "",
  group.names = NULL,
  group.side = 3,
  group.adj = 0.98,
  group.padj = 2,
  group.at = NA,
  par.reset = TRUE,
  ...
)
```

Arguments

x	Survival object / list of Survival objects created using `survival::Surv`
lty	Integer: Line type. Default = 1. See `par("lty")`
lwd	Float: Line width. Default = 2
alpha	Float: Alpha for lines. Default = 1
normalize.time	Logical: If TRUE, convert each input's time to 0-1 range. This is useful when survival estimates are not provided in original time scale. Default = FALSE.
xlab	Character: x-axis label
ylab	Character: y-axis label
main	Character: Plot title
theme	Character: Run `themes()` for available themes
palette	Vector of colors, or Character defining a builtin palette - get options with `rtpalette()`
group.legend	Logical: If TRUE, place `group.names` in a legend
group.title	Character: Group title, shown above group names. e.g. if group names are c("San Francisco", "Philadelphia"), `group.title` can be "City"
group.names	(Optional) If multiple groups are plotted, use these names if `group.title = TRUE`
group.side	Integer: Side to show group legend
group.adj	Float: `adj` for group legend. See `mtext("adj")`
group.padj	Float: `padj` for group legend See `mtext("padj")`
group.at	Float: location for group legend. See `mtext("at")`
par.reset	Logical: If TRUE, reset `par` setting before exiting.
...	Additional arguments to pass to mplot3_xy

Author(s)

E.D. Gennatas

Examples

```
## Not run:
library(survival)
mplot3_surv(Surv(time = lung$time, event = lung$status))

## End(Not run)
```

mplot3_survfit — mplot3: *Plot* survfit *objects*

Description

Plots survival step functions using mplot3_xy

Usage

```
mplot3_survfit(
  x,
  lty = 1,
  lwd = 1.5,
  alpha = 1,
  col = NULL,
  plot.median = FALSE,
  group.median = FALSE,
  median.lty = 3,
  median.lwd = 2,
  median.col = theme$fg,
  median.alpha = 0.5,
  censor.mark = TRUE,
  censor.col = NULL,
  censor.alpha = 0.4,
  censor.pch = "I",
  censor.cex = 0.8,
  mark.censored = FALSE,
  nrisk.table = FALSE,
  nrisk.pos = "below",
  nrisk.spacing = 0.9,
  table.font = 1,
  time.at = NULL,
  time.by = NULL,
  xlim = NULL,
  ylim = NULL,
  xlab = "Time",
  ylab = "Survival",
  main = "",
  theme = rtTheme,
  palette = rtPalette,
  plot.error = FALSE,
  error.alpha = 0.33,
  autonames = TRUE,
  group.legend = NULL,
  group.legend.type = c("legend", "mtext"),
  group.names = NULL,
  group.title = NULL,
```

```
  group.line = NULL,
  group.side = NULL,
  legend.x = NULL,
  mar = c(2.5, 3, 2, 1),
  oma = NULL,
  par.reset = TRUE,
  pdf.width = 6,
  pdf.height = 6,
  filename = NULL,
  ...
)
```

Arguments

x	survfit object (output of `survival::survfit`)
lty	Integer: Line type. See `par("lty")`
lwd	Float: Line width.
alpha	Float: Alpha for lines.
col	Color, vector: Color(s) to use for survival curves and annotations. If NULL, taken from `palette`
plot.median	Logical: If TRUE, draw lines at 50 percent median survival.
group.median	Logical: If TRUE, include median survival times with group legend
median.lty	Integer: Median survival line type
median.lwd	Float: Median line width.
median.col	Color for median survival lines
median.alpha	Float, (0, 1): Transparency for median survival lines.
censor.mark	Logical: If TRUE, mark each censored case.
censor.col	Color to mark censored cases if `censor.mark = TRUE`
censor.alpha	Transparency for `censor.col`.
censor.pch	Character: Point character for censored marks.
censor.cex	Float: Character expansion factor for censor marks.
mark.censored	Logical: This is an alternative to `censor.mark` which whill mark censored cases using the same color as the survival curve. It can be harder to distinguish the censoring marks from the curve itself, therefore not preferred.
nrisk.table	Logical: If TRUE, print Number at risk table.
nrisk.pos	Character: "above" or "below": where to place `nrisk.table`
nrisk.spacing	Float: Determines spacing between `nrisk.table` rows.
table.font	Integer: 1: regular font, 2: bold.
time.at	Float, vector: x-axis positions to place tickmarks and labels as well as n at risk values if `nrisk.table = TRUE`
time.by	Float: Divide time by this amount to determine placing of tickmarks
xlim	Float, vector, length 2: x-axis limits

ylim	Float, vector, length 2: y-axis limits
xlab	Character: x-axis label
ylab	Character: y-axis label
main	Character: main title
theme	Character: Run `themes()` for available themes
palette	Vector of colors, or Character defining a builtin palette - get options with `rtpalette()`
autonames	Logical: If TRUE, extract grouping variable names and level labels from `x` and use for legend. It is best to give informative level labels, like female, male instead of 0, 1 when using this.
group.legend	Logical: If TRUE, include group legend
group.names	Character, vector: Group names to use. If NULL, extracted from `x`
group.title	Character: Group legend title
group.line	Float, vector: Lines to print group legend using `mtext`
group.side	Integer: Side to print group legend. Default is determined by survival curves, to avoid overlap of legend with curves.
mar	Float, vector, length 4: Margins. See `par("mar")`
oma	Float, vector, length 4: Outer margins. See `par("oma")`
par.reset	Logical: If TRUE, reset par to initial values before exit
...	Additional arguments to pass to theme

Author(s)

E.D. Gennatas

Examples

```
## Not run:
# Get the lung dataset
data(cancer, package = "survival")
sf1 <- survival::survfit(survival::Surv(time, status) ~ 1, data = lung)
mplot3_survfit(sf1)
sf2 <- survival::survfit(survival::Surv(time, status) ~ sex, data = lung)
mplot3_survfit(sf2)
# with N at risk table
mplot3_survfit(sf2, nrisk.table = TRUE)

## End(Not run)
```

mplot3_varimp   mplot3: *Variable Importance*

Description

Draw horizontal barplots for variable importance

Usage

```
mplot3_varimp(
  x,
  error = NULL,
  names = NULL,
  names.pad = 0.02,
  plot.top = 1,
  labelify = TRUE,
  col = NULL,
  palette = rtPalette,
  alpha = 1,
  error.col = theme$fg,
  error.lwd = 2,
  beside = TRUE,
  border = NA,
  width = 1,
  space = 0.75,
  xlim = NULL,
  ylim = NULL,
  xlab = "Variable Importance",
  xlab.line = 1.3,
  ylab = NULL,
  ylab.line = 1.5,
  main = NULL,
  names.arg = NULL,
  axisnames = FALSE,
  sidelabels = NULL,
  mar = NULL,
  pty = "m",
  barplot.axes = FALSE,
  xaxis = TRUE,
  x.axis.padj = -1.2,
  tck = -0.015,
  theme = rtTheme,
  zerolines = FALSE,
  par.reset = TRUE,
  autolabel = letters,
  pdf.width = NULL,
  pdf.height = NULL,
```

```
    trace = 0,
    filename = NULL,
    ...
)
```

Arguments

x	Vector, numeric: Input
error	Vector, numeric; length = length(x): Plot error bars with given error.
names	Vector, string; optional: Names of variables in x
plot.top	Float or Integer: If <= 1, plot this percent highest absolute values, otherwise plot this many top values. i.e.: plot.top = .2 will print the top 20% highest values, and plot.top = 20 will plot the top 20 highest values
labelify	Logical: If TRUE convert names(x) using labelify. Default = TRUE
col	Colors: Gradient to use for barplot fill.
alpha	Float (0, 1): Alpha for col
error.col	Color: For error bars
trace	Integer: If trace > 0 prints out the automatically set mar (so you can adjust if needed) names provided

Details

"NA" values in input are set to zero.

Value

Position of bar centers (invisibly)

Author(s)

E.D. Gennatas

---

mplot3_x mplot3*: Univariate plots: index, histogram, density, QQ-line*

Description

Draw plots of 1-dimensional data: index, histogram, density, and Q-Q plots.

Usage

```
mplot3_x(
  x,
  type = c("density", "histogram", "hd", "lhist", "index", "ts", "qqline"),
  group = NULL,
  data = NULL,
  xlab = NULL,
  ylab = NULL,
  main = NULL,
  xlim = NULL,
  ylim = NULL,
  index.ypad = 0.1,
  axes.swap = FALSE,
  axes.col = NULL,
  tick.col = NULL,
  cex = 1.2,
  col = NULL,
  alpha = 0.75,
  index.type = c("p", "l"),
  hist.breaks = "Sturges",
  hist.type = c("bars", "lines"),
  hist.probability = FALSE,
  hist.lwd = 3,
  density.line = FALSE,
  density.shade = TRUE,
  density.legend.side = 3,
  density.legend.adj = 0.98,
  density.bw = "nrd0",
  density.kernel = "gaussian",
  density.params = list(na.rm = na.rm),
  qqline.col = "#18A3AC",
  qqline.alpha = 1,
  pch = 16,
  point.col = NULL,
  point.cex = 1,
  point.bg.col = NULL,
  point.alpha = 0.66,
  hline = NULL,
  vline = NULL,
  diagonal = FALSE,
  grid = FALSE,
  grid.col = NULL,
  grid.alpha = 0.5,
  grid.lty = 3,
  grid.lwd = 1,
  annotation = NULL,
  annot.col = NULL,
  group.legend = NULL,
```

```
    group.title = "",
    group.names = NULL,
    group.side = 3,
    group.adj = 0.02,
    group.at = NA,
    text.xy = NULL,
    text.x = NULL,
    text.y = NULL,
    text.xy.cex = 1,
    text.xy.col = "white",
    line.col = "#008E00",
    x.axis.padj = -1.1,
    y.axis.padj = 0.9,
    labs.col = NULL,
    lab.adj = 0.5,
    density.avg = ifelse(type == "density", TRUE, FALSE),
    density.avg.fn = c("median", "mean"),
    density.avg.line = FALSE,
    density.avg.lwd = 1.5,
    density.avg.lty = 3,
    hline.col = "black",
    hline.lwd = 1,
    hline.lty = 1,
    vline.col = "black",
    vline.lwd = 1,
    vline.lty = 1,
    lty = 1,
    lwd = 2,
    qqline.lwd = lwd,
    density.lwd = lwd,
    theme = rtTheme,
    palette = rtPalette,
    pty = "m",
    mar = NULL,
    oma = rep(0, 4),
    xaxs = "r",
    yaxs = "r",
    autolabel = letters,
    new = FALSE,
    alpha.off = FALSE,
    na.rm = TRUE,
    par.reset = TRUE,
    filename = NULL,
    pdf.width = 6,
    pdf.height = 6,
    ...
  )
```

Arguments

x	Numeric vector or list of vectors, one for each group. If data is provided, x is name of variable in data
type	Character: "density", "histogram", "hd" (histogram bars & density lines), "lhist" (line histogram like mhist; same as type = "hist", hist.type = "lines"), "index", "ts", "qqline" Case-insensitive and supports partial matching: e.g. mplot3_x(x, "H") gives histogram
group	Vector denoting group membership. Will be converted to factor. If data is provided, group is name of variable if data
data	Optional data frame containing x data
xlab	Character: x-axis label
ylab	Character: y-axis label
main	Character: Plot title
xlim	Float vector, length 2: x-axis limits
ylim	Float vector, length 2: y-axis limits
index.ypad	Float: Expand ylim by this much for plot type "index" Default = .1 (Stops points being cut off)
index.type	Character: "p" for points (Default), "l" for lines (timeseries)
hist.breaks	See histogram("breaks")
hist.type	Character: "bars" or "lines". Default = "bars"
hist.lwd	Float: Line width for type = "histogram"; hist.type = "lines"
density.line	Logical: If TRUE, draw line for type = "density". Default = FALSE
density.shade	Logical: If TRUE, draw shaded polygon for type = "density". Default = TRUE
qqline.col	Color for Q-Q line
qqline.alpha	Float: Alpha for Q-Q line
pch	Integer: Point character.
point.cex	Float: Character expansion for points.
point.bg.col	Color: point background
hline	Vector: y-value(s) for horizontal lines.
vline	Vector: x-value(s) for vertical lines.
diagonal	Logical: If TRUE, draw diagonal line.
annotation	Character: Add annotation at the bottom right of the plot
group.legend	Logical: If TRUE, include legend with group names
group.title	Character: Title above group names
group.names	(Optional) If multiple groups are plotted, use these names if group.title = TRUE
group.side	Integer: Side to show group legend
group.adj	Float: adj for group legend. See mtext("adj")

`group.at`	Float: location for group legend. See `mtext("at")`
`line.col`	Color for lines
`labs.col`	Color for labels
`lab.adj`	Adjust the axes labels. 0 = left adjust; 1 = right adjust; .5 = center (Default)
`density.avg`	Logical: If TRUE, print mean of `x` along plot. Default = TRUE, for `type = "density"`
`density.avg.fn`	Character: "median" or "mean". Function to use if `density.avg = TRUE`. Default = "median"
`density.avg.line`	Logical: If TRUE, draw vertical lines at the density average x-value
`density.avg.lwd`	Float: Line width for `density.avg.line`. Default = 1.5
`density.avg.lty`	Integer: Line type for `density.avg.line`. Default = 3
`hline.col`	Color for horizontal line(s)
`hline.lwd`	Float: Width for horizontal line(s)
`hline.lty`	Integer: Line type for horizontal line(s)
`vline.col`	Color for vertical lines
`vline.lwd`	Float: Width for vertical lines
`vline.lty`	Integer: Line type for vertical lines
`lty`	Integer: Line type. See `par("lty")`
`lwd`	Integer: Line width. Used for `type = "ts" or "density"`
`theme`	Character: Run `themes()` for available themes
`palette`	Vector of colors, or Character defining a builtin palette - get options with `rtpalette()`
`pty`	Character: "s" gives a square plot; "m" gives a plot that fills graphics device size. Default = "m" (See `par("pty")`)
`mar`	Float, vector, length 4: Margins; see `par("mar")`
`oma`	Float, vector, length 4: Outer margins; see `par("oma")`
`xaxs`	Character: "r": Extend plot x-axis limits by 4% on either end; "i": Use exact x-axis limits.
`yaxs`	Character: as `xaxs` for the y-axis.
`autolabel`	Character vector to be used to generate autolabels when using rtlayout with `autolabel = TRUE`.
`new`	Logical: If TRUE, add plot to existing plot. See `par("new")`
`na.rm`	Logical: Will be passed to all functions that support it. If set to FALSE, input containing NA values will result in error, depending on the `type`
`par.reset`	Logical: If TRUE, reset `par` setting before exiting.
`filename`	Path to file: If supplied, plot will be printed to file
`pdf.width`	Float: Width in inches for pdf output (if `filename` is set).
`pdf.height`	Float: Height in inches for pdf output.
`...`	Additional arguments to be passed to theme function

Details

You can group data either by supplying x as a list where each element contains one vector per group, or as a data frame where each column represents group, or by providing a `group` variable, which will be converted to factor. For bivariate plots, see mplot3_xy and mplot3_xym. For heatmaps, see mplot3_heatmap To plot histograms of multiple groups, it's best to use `hist.type = "lines"`, which will use mhist and space apart the breaks for each group

Value

Invisibly returns the output of `density`, `hist`, `qqnorm`, or NULL

Author(s)

E.D. Gennatas

See Also

mplot3_xy, mplot3_xym, mplot3_heatmap

Examples

```
## Not run:
mplot3_x(iris)
mplot3_x(split(iris$Sepal.Length, iris$Species), xlab = "Sepal Length")

## End(Not run)
```

`mplot3_xy`   `mplot3`: *XY Scatter and line plots*

Description

Plot points and lines with optional fits and standard error bands

Usage

```
mplot3_xy(
  x,
  y = NULL,
  fit = NULL,
  formula = NULL,
  se.fit = FALSE,
  fit.params = NULL,
  error.x = NULL,
  error.y = NULL,
  cluster = NULL,
  cluster.params = list(),
```

```
data = NULL,
type = "p",
group = NULL,
xlab = NULL,
ylab = NULL,
main = NULL,
xlim = NULL,
ylim = NULL,
xpd = TRUE,
xaxs = "r",
yaxs = "r",
log = "",
rsq = NULL,
rsq.pval = FALSE,
rsq.side = 1,
rsq.adj = 0.98,
rsq.col = NULL,
rsq.line = NULL,
fit.error = FALSE,
fit.error.side = 1,
fit.error.padj = NA,
xaxp = NULL,
yaxp = NULL,
scatter = TRUE,
axes.equal = FALSE,
pty = "m",
annotation = NULL,
annotation.col = NULL,
x.axis.at = NULL,
x.axis.labs = TRUE,
y.axis.at = NULL,
y.axis.labs = TRUE,
xlab.adj = 0.5,
ylab.adj = 0.5,
mar = NULL,
oma = rep(0, 4),
point.cex = 1,
point.bg.col = NULL,
pch = ifelse(is.null(point.bg.col), 16, 21),
line.col = NULL,
line.alpha = 0.66,
lwd = 1,
lty = 1,
marker.col = NULL,
marker.alpha = NULL,
error.x.col = NULL,
error.y.col = NULL,
error.x.lty = 1,
```

```
        error.y.lty = 1,
        error.x.lwd = 1,
        error.y.lwd = 1,
        error.arrow.code = 3,
        fit.col = NULL,
        fit.lwd = 2.5,
        fit.alpha = 1,
        fit.legend = ifelse(is.null(fit), FALSE, TRUE),
        se.lty = "poly",
        se.lwd = 1,
        se.col = NULL,
        se.alpha = 0.5,
        se.times = 1.96,
        se.border = FALSE,
        se.density = NULL,
        hline = NULL,
        hline.col = NULL,
        hline.lwd = 1.5,
        hline.lty = 3,
        vline = NULL,
        vline.lwd = 1.5,
        vline.col = "blue",
        vline.lty = 3,
        diagonal = FALSE,
        diagonal.inv = FALSE,
        diagonal.lwd = 1.5,
        diagonal.lty = 1,
        diagonal.col = "gray50",
        diagonal.alpha = 1,
        group.legend = NULL,
        group.title = NULL,
        group.names = NULL,
        group.side = 3,
        group.adj = 0.02,
        group.padj = 2,
        group.at = NA,
        fit.legend.col = NULL,
        fit.legend.side = 3,
        fit.legend.adj = 0.02,
        fit.legend.padj = 2,
        fit.legend.at = NA,
        rm.na = TRUE,
        theme = rtTheme,
        palette = rtPalette,
        order.on.x = NULL,
        autolabel = letters,
        new = FALSE,
        par.reset = TRUE,
```

```
  return.lims = FALSE,
  pdf.width = 6,
  pdf.height = 6,
  trace = 0,
  filename = NULL,
  ...
)
```

Arguments

`x`	Numeric vector or list of vectors for x-axis. If `data` is provided, name of variable, unquoted.
`y`	Numeric vector of list of vectors for y-axis If `data` is provided, name of variable, unquoted.
`fit`	Character: **rtemis** model to calculate `y ~ x` fit. Options: see `select_learn()` Can also be Logical, which will give a GAM fit if TRUE. If you specify "NLA", the activation function should be passed as a string.
`formula`	Formula: Provide a formula to be solved using s_NLS. If provided, `fit` is forced to `'nls'`. e.g. y ~ b * m ^ x for a power curve. Note: `nls` is powerful but is prone to errors and warnings. Use single letters for parameter names, no numbers.
`se.fit`	Logical: If TRUE, draw the standard error of the fit
`fit.params`	List: Arguments for learner defined by `fit`. Default = NULL, i.e. use default learner parameters
`error.x`	Vector, float: Error in `x` (e.g. standard deviation) will be plotted as bars around point
`error.y`	Vector, float: Error in `y` (e.g. standard deviation) will be plotted as bars around point
`cluster`	Character: Clusterer name. Will cluster `data.frame(x, y)` and pass result to `group`. Run select_clust for options
`cluster.params`	List: Names list of parameters to pass to the `cluster` function
`data`	(Optional) data frame, where `x` and `y` are defined
`type`	Character: "p" for points, "l" for lines, "s" for steps. Default = "p". If `x` and/or `y` contains multiple vectors, `type` can be a vector, e.g. `c("p", "l", "l")` will give a set of points and two sets of lines. Otherwise, `type` is recycled to length of x
`group`	Vector: will be converted to factor. If `data` is provided, name of variable, unquoted.
`xlab`	Character: x-axis label
`ylab`	Character: y-axis label
`main`	Character: Plot title
`xlim`	Float vector, length 2: x-axis limits
`ylim`	Float vector, length 2: y-axis limits
`xpd`	Logical or NA: FALSE: plotting clipped to plot region; TRUE: plotting clipped to figure region; NA: plotting clipped to device region.

`xaxs`	Character: "r": Extend plot x-axis limits by 4% on either end; "i": Use exact x-axis limits.
`yaxs`	Character: as `xaxs` for the y-axis.
`log`	Character: "x", "y", or "xy", defines if either or both axes should be log-transformed.
`rsq`	Logical: If TRUE, add legend with R-squared (if fit is not NULL)
`rsq.pval`	Logical: If TRUE, add legend with R-squared and its p-value, if fit is not NULL
`rsq.side`	Integer: [1:4] Where to place the `rsq` annotation. Default = 1 (i.e. bottom)
`rsq.adj`	Float: Adjust `rsq` annotation. See `mtext "adj"`
`rsq.col`	Color: Color for `rsq` annotation. Default = NULL, which results in `fit.col`
`rsq.line`	Numeric: Passed to `mtext "line"` to place R-squared annotation.
`fit.error`	Logical: If TRUE: draw fit error annotation. Default = NULL, which results in TRUE, if fit is set
`fit.error.side`	Integer [1:4]: Which side to draw `fit.error` on.
`fit.error.padj`	Float: See `mtext:padg` Default = NA
`xaxp`	See `par("xaxp")`
`yaxp`	See `par("yaxp")`
`scatter`	Logical: If TRUE, plot (x, y) scatter points.
`axes.equal`	Logical: Should axes be equal? Defaults to FALSE
`pty`	Character: "s" gives a square plot; "m" gives a plot that fills graphics device size. Default = "m" (See `par("pty")`)
`annotation`	Character: Add annotation at the bottom right of the plot
`annotation.col`	Color for annotation
`x.axis.at`	Float, vector: x coordinates to place tick marks. Default = NULL, determined by `graphics::axis` aautomatically
`x.axis.labs`	See `axis("labels")`
`y.axis.at`	As `x.axis.at` for y-axis
`y.axis.labs`	See `axis("labels")`
`xlab.adj`	Float: `adj` for `xlab` (See `par("adj")`)
`ylab.adj`	Float: `adj` for `ylab` (See `par("adj")`)
`mar`	Float, vector, length 4: Margins; see `par("mar")`
`oma`	Float, vector, length 4: Outer margins; see `par("oma")`
`point.cex`	Float: Character expansion for points.
`point.bg.col`	Color: point background
`pch`	Integer: Point character.
`line.col`	Color for lines
`line.alpha`	Float [0, 1]: Transparency for lines
`lwd`	Float: Line width
`lty`	Integer: Line type. See `par("lty")`

marker.col	Color for marker
marker.alpha	Float [0, 1]: Transparency for markers
error.x.col	Color for x-axis error bars
error.y.col	Color for y-axis error bars
error.x.lty	Integer: line type for x-axis error bars
error.y.lty	Integer: line type for y-axis error bars
error.x.lwd	Float: Line width for x-axis error bars
error.y.lwd	Float: Line width for y-axis error bars
error.arrow.code	Integer: Type of arrow to draw for error bars. See arrows("code")
fit.col	Color: Color of the fit line.
fit.lwd	Float: Fit line width
fit.alpha	Float [0, 1]: Transparency for fit line
fit.legend	Logical: If TRUE, show fit legend
se.lty	How to draw the se.fit "poly" draws a polygon around the fit line, otherwise an integer defines the lty (line type) for lines to be drawn
se.lwd	Float: Line width for standard error bounds
se.col	Color for se.fit
se.alpha	Alpha for se.fit
se.times	Draw polygon or lines at +/- se.times * standard error of fit. Defaults to 1.96, which corresponds to 95 percent confidence interval
se.border	Define border of polygon for se.fit. See border in graphics::polygon
se.density	Density of shading line of polygon for se.fit. See density in graphics::polygon
hline	Vector: y-value(s) for horizontal lines.
hline.col	Color for horizontal line(s)
hline.lwd	Float: Width for horizontal line(s)
hline.lty	Integer: Line type for horizontal line(s)
vline	Vector: x-value(s) for vertical lines.
vline.lwd	Float: Width for vertical lines
vline.col	Color for vertical lines
vline.lty	Integer: Line type for vertical lines
diagonal	Logical: If TRUE, draw diagonal line.
diagonal.inv	Logical: If TRUE, draw inverse diagonal line. Will use diagonal.lwd, diagonal.lty, diagonal.col, diagonal.alpha (Note: it only makes sense to use only one of diagonal or diagonal.inv)
diagonal.lwd	Float: Line width for diagonal.
diagonal.lty	Integer: Line type for diagonal.
diagonal.col	Color: Color for diagonal.

diagonal.alpha	Float: Alpha for diagonal
group.legend	Logical: If TRUE, place group.names in a legend
group.title	Character: Group title, shown above group names. e.g. if group names are c("San Francisco", "Philadelphia"), group.title can be "City"
group.names	(Optional) If multiple groups are plotted, use these names if group.title = TRUE
group.side	Integer: Side to show group legend
group.adj	Float: adj for group legend. See mtext("adj")
group.padj	Float: padj for group legend See mtext("padj")
group.at	Float: location for group legend. See mtext("at")
fit.legend.col	Color for fit legend
fit.legend.side	Integer: Side for fit legend
fit.legend.adj	Float: adj for fit legend
fit.legend.padj	Float: padj for fit legend
fit.legend.at	Float: location for fit legend. See mtext("at")
rm.na	Logical: If TRUE, remove all NA values pairwise between x and y. Set to FALSE if you know your data has no missing values.
theme	Character: Run themes() for available themes
palette	Vector of colors, or Character defining a builtin palette - get options with rtpalette()
order.on.x	Logical: If TRUE, order (x, y) by increasing x. Default = NULL: will be set to TRUE if fit is set, otherwise FALSE
autolabel	Character vector to be used to generate autolabels when using rtlayout with autolabel = TRUE.
new	Logical: If TRUE, add plot to existing plot. See par("new")
par.reset	Logical: If TRUE, reset par setting before exiting.
return.lims	Logical: If TRUE, return xlim and ylim.
pdf.width	Float: Width in inches for pdf output (if filename is set).
pdf.height	Float: Height in inches for pdf output.
trace	Integer: If > 0, pass verbose = TRUE to the cluster and fit functions, if used.
filename	Character: Path to file to save plot. Default = NULL
...	Additional arguments to be passed to theme function

Author(s)

E.D. Gennatas

Examples

```
## Not run:
set.seed(1999)
x <- rnorm(500)
ycu <- x^3 + 12 + rnorm(500)
mplot3_xy(x, ycu)
mplot3_xy(x, ycu, fit = "gam")
ysq <- x^2 + 3 + rnorm(500)
mplot3_xy(x, list(squared = ysq, cubed = ycu), fit = "gam")

## End(Not run)
```

---

mplot3_xym *Scatter plot with marginal density and/or histogram*

Description

Draw a scatter plot with fit line and marginal density and/or histogram

Usage

```
mplot3_xym(
  x,
  y,
  margin = c("histogram", "density", "both"),
  fit = "gam",
  se.fit = TRUE,
  xlim = NULL,
  ylim = NULL,
  col = "#18A3AC",
  density.alpha = 0.66,
  hist.breaks = 30,
  hist.alpha = 0.66,
  hist.space = 0.05,
  hist.lwd = 3,
  lwd = 4,
  main = NULL,
  main.adj = 0,
  axes.density = FALSE,
  pty = "m",
  mar = c(3, 3, 0, 0),
  margin.mar = 0.2,
  xaxs = "r",
  yaxs = "r",
  theme = rtTheme,
  par.reset = TRUE,
  widths = NULL,
```

```
  heights = NULL,
  filename = NULL,
  pdf.width = 7,
  pdf.height = 7,
  ...
)
```

Arguments

x	Numeric vector: x-axis data
y	Numeric vector: y-axis data
margin	Character: "density", "histogram", or "both". Type of marginal plots to draw.
fit	Character: Algorithm to use to draw y ~ x.
se.fit	Logical: If TRUE: plot +/- 2 * Standard Error of fit
xlim	Float vector, length 2: x-axis limits
ylim	Float vector, length 2: y-axis limits
col	Color for marginal plots
density.alpha	Numeric: Alpha for density plots
hist.breaks	Integer: Number of histogram breaks
hist.alpha	Numeric: Alpha for barplots
hist.space	Numeric: Space between bars in barplots
hist.lwd	Numeric: Line width for barplots
lwd	Numeric: Line width
main	Character: Main title
main.adj	Numeric: Main title adjustment
axes.density	Logical: If TRUE, plot margin plot axes for density (debugging only)
pty	Character: "s" gives a square plot; "m" gives a plot that fills graphics device size. Default = "m" (See `par("pty")`)
mar	Float, vector, length 4: Margins; see `par("mar")`
margin.mar	Numeric: Margin for marginal plots
xaxs	Character: "r": Extend plot x-axis limits by 4% on either end; "i": Use exact x-axis limits.
yaxs	Character: as `xaxs` for the y-axis.
theme	Character: Run `themes()` for available themes
par.reset	Logical: Resest `par` to original settings
filename	Character: Path to file to save plot. Default = NULL
pdf.width	Float: Width in inches for pdf output (if `filename` is set).
pdf.height	Float: Height in inches for pdf output.
...	Additional arguments to passed to mplot3_xy

Details

To make wide plot, change `widths`: e.g. widths = c(7, 1)

Author(s)

E.D. Gennatas

Examples

```
## Not run:
x <- rnorm(500)
y <- x^3 + 12 + rnorm(500)
mplot3_xym(x, y)

## End(Not run)
```

---

mplot_AGGTEobj *Plot AGGTEobj object*

Description

Plot AGGTEobj object from the **did** package.

Usage

```
mplot_AGGTEobj(
  x,
  x.factor = 1,
  y.factor = 1,
  error = c("se", "95%ci"),
  main = "Average Effect by Length of Exposure",
  legend.title = "",
  group.names = c("Pre", "Post"),
  xlab = NULL,
  ylab = NULL,
  mar = c(2.5, 3.5, 2, 7),
  theme = rtTheme,
  font.family = "Helvetica",
  col = c("#EC1848", "#18A3AC"),
  filename = NULL,
  file.width = 6.5,
  file.height = 5.5,
  par.reset = TRUE,
  ...
)
```

Arguments

x	AGGTEobj object
main	Character: Plot title
group.names	(Optional) If multiple groups are plotted, use these names if group.title = TRUE
xlab	Character: x-axis label
ylab	Character: y-axis label
mar	Float, vector, length 4: Margins; see par("mar")
theme	Character: Run themes() for available themes
filename	Character: Path to file to save plot. Default = NULL
par.reset	Logical: If TRUE, reset par setting before exiting.
...	Additional arguments to be passed to theme function

Author(s)

E.D. Gennatas

---

mplot_hsv — *Plot HSV color range*

Description

Plot HSV color range

Usage

```
mplot_hsv(
  h.steps = seq(0, 1, 0.025),
  s.steps = seq(0, 1, 0.05),
  v = 1,
  alpha = 1,
  pch = 16,
  bg = "black",
  axes = TRUE,
  pty = "s",
  cex = 2,
  mar = c(3, 3, 2, 0.5),
  lab.col = NULL,
  type = c("radial", "square"),
  line.col = "gray50",
  show.grid = TRUE,
  show.radial.grid = FALSE,
  show.grid.labels = 1,
  cex.axis = 1,
```

```
  cex.lab = 1,
  par.reset = TRUE
)
```

Arguments

`h.steps`	Float, vector: Hue values to plot. Default = `seq(0, 1, .0125)`
`s.steps`	Float, vector: Saturation values to plot. Default = same as `h.steps`
`v`	Float: Value.
`alpha`	Float: Alpha.
`pch`	Integer: pch plot parameter. Default = 15 (square)
`bg`	Color: Background color. Default = "black"
`axes`	Logical: for `type = "square"`: If TRUE, draw axes.
`pty`	Character: for `type = "square"`: "s", "r", par's pty argument. Default = "s" (square plot)
`cex`	Float: `par/plot`'s cex argument.
`mar`	Float, vector: for `type = "square"`: `par`'s mar argument.
`lab.col`	Color: Color for axes and labels. Defaults to inverse of bg, i.e. white if bg is black
`type`	Character: "square" for square plot, "radial" for radial plot.
`show.grid`	Logical: if TRUE, show grid then type is "radial"
`par.reset`	Logical: If TRUE, reset `par` before exit

Author(s)

E.D. Gennatas

Examples

```
## Not run:
mplot_hsv()

## End(Not run)
```

`mplot_raster` *Plot Array as Raster Image*

Description

Plots 2D (grayscale) or 3D (color) array as Raster Image

Usage

```
mplot_raster(
  x,
  max.value = max(x),
  mar = NULL,
  main = NULL,
  main.line = 0,
  main.side = 3,
  main.col = "#ffffff",
  main.adj = 0,
  main.font = 2,
  mono = FALSE,
  mono.fn = mean,
  bg = "gray10",
  par.set = TRUE,
  par.reset = TRUE,
  verbose = TRUE
)
```

Arguments

x	Array, 2D or 3D: Input describing grayscale or color image in RGB space
mono	Logical: If TRUE, plot as grayscale using mono.fn to convert RGB to grayscale. Default = FALSE
mono.fn	Function: Apply this function to the array to convert to 2D for grayscale plotting. Default = mean
bg	Color: Background color (around the plotted image when window proportions do not match image). Default = "gray10"
par.reset	Logical: If TRUE, reset par settings before exiting. Default = TRUE
verbose	Logical: If TRUE, print messages to console. Default = TRUE

Author(s)

E.D. Gennatas

Examples

```
## Not run:
img <- imager::load.image("https://www.r-project.org/logo/Rlogo.png")
mplot_raster(img)

## End(Not run)
```

mse *Error functions*

Description

Convenience functions for calculating loss. These can be passed as arguments to learners that support custom loss functions.

Usage

```
mse(x, y, na.rm = TRUE)

msew(x, y, weights = rep(1, length(y)), na.rm = TRUE)

rmse(x, y, na.rm = TRUE)

mae(x, y, na.rm = TRUE)
```

Arguments

x	Vector of True values
y	Vector of Estimated values
na.rm	Logical: If TRUE, remove NA values before computation. Default = TRUE
weights	Float, vector: Case weights

multigplot *Multipanel* **ggplot2** *plots*

Description

Plot a panel of **gplot2** plots

Usage

```
multigplot(plots = NULL, nrows = NULL, byrow = TRUE)
```

Arguments

plots	List of ggplot2 plots
nrows	Integer: number of rows for panel arrangement. Defaults to number of rows required to plot 2 plots per row
byrow	Logical: If TRUE, draw plots in order provided by row, otherwise by column. Default = TRUE

Author(s)

E.D. Gennatas

nCr — *n Choose r*

Description

Calculate number of combinations

Usage

```
nCr(n, r)
```

Arguments

`n`	Integer: Total number of items
`r`	Integer: Number of items in each combination

Details

In plain language: You have `n` items. How many different cobinations of `r` items can you make?

Value

Integer: Number of combinations

Author(s)

E.D. Gennatas

oddsratio — *Calculate odds ratio for a 2x2 contingency table*

Description

Calculate odds ratio for a 2x2 contingency table

Usage

```
oddsratio(x, verbose = TRUE)
```

Arguments

`x`	2x2 contingency table (created with `table(x, y)`, where `x` and `y` are factors with the first level being the control / unaffected / negative)
`verbose`	Logical: If TRUE, print messages to console

`oddsratiotable` *Odds ratio table from logistic regression*

Description

Odds ratio table from logistic regression

Usage

```
oddsratiotable(x, confint.method = c("default", "profilelikelihood"))
```

Arguments

x	glm object fit with `family = binomial`
confint.method	"default" or "profilelikelihood"

Value

matrix with 4 columns: OR, 2.5% & 97.5% CI, p_val

Author(s)

E.D. Gennatas

Examples

```
## Not run:
ir2 <- iris[51:150, ]
ir2$Species <- factor(ir2$Species)
ir.fit <- glm(Species ~ ., data = ir2, family = binomial)
oddsratiotable(ir.fit)

## End(Not run)
```

`oneHot` *One hot encoding*

Description

One hot encode a vector or factors in a data.frame

Usage

```
oneHot(x, xname = NULL, verbose = FALSE)

## Default S3 method:
oneHot(x, xname = NULL, verbose = TRUE)

## S3 method for class 'data.frame'
oneHot(x, xname = NULL, verbose = TRUE)

## S3 method for class 'data.table'
oneHot(x, xname = NULL, verbose = TRUE)

dt_set_oneHot(x, xname = NULL, verbose = TRUE)
```

Arguments

x	Vector or data.frame
xname	Character: Variable name
verbose	Logical: If TRUE, print messages to console

Details

A vector input will be one-hot encoded regardless of type by looking at all unique values. With data.frame input, only column of type factor will be one-hot encoded. This function is used by preprocess. `oneHot.data.table` operates on a copy of its input. `oneHot_` performs one-hot encoding in-place.

Value

For vector input, a one-hot-encoded matrix, for data.frame frame input, an expanded data.frame where all factors are one-hot encoded

Author(s)

E.D. Gennatas

Examples

```
## Not run:
iris_oh <- oneHot(iris)
# factor with only one unique value but 2 levels:
vf <- factor(rep("alpha", 20), levels = c("alpha", "beta"))
vf_onehot <- oneHot(vf)

## End(Not run)
oneHot(iris) |> head()
ir <- data.table::as.data.table(iris)
ir_oh <- oneHot(ir)
ir_oh
```

```
ir <- data.table::as.data.table(iris)
# dt_set_oneHot operates in-place; therefore no assignment is used:
dt_set_oneHot(ir)
ir
```

---

onehot2factor *Convert one-hot encoded matrix to factor*

Description

Convert one-hot encoded matrix to factor

Usage

```
onehot2factor(x, labels = colnames(x))
```

Arguments

x	one-hot encoded matrix or data.frame
labels	Character vector of level names. Default = `colnames(x)`

Details

If input has a single column, it will be converted to factor and returned

Author(s)

E.D. Gennatas

Examples

```
## Not run:
x <- data.frame(matrix(F, 10, 3))
colnames(x) <- c("Dx1", "Dx2", "Dx3")
x$Dx1[1:3] <- x$Dx2[4:6] <- x$Dx3[7:10] <- T
onehot2factor(x)

## End(Not run)
```

palettize *Palettize colors*

Description

Filter and order a set of colors to produce a palette suitable for multicolor plots

Usage

```
palettize(
  x,
  grayscale_hicut = 0.8,
  start_with = "#16A0AC",
  order_by = c("separation", "dissimilarity", "similarity")
)
```

Arguments

x	Color vector
grayscale_hicut	Numeric: exclude colors whose grayscale equivalent is greater than this value
start_with	Integer or color: For integer, start with this color out of x, otherwise find color x closer to this color and place it first
order_by	Character: "separation", "dissimilarity", "similarity"

Author(s)

E.D. Gennatas

permute *Create permutations*

Description

Creates all possible permutations

Usage

```
permute(n)
```

Arguments

n	Integer: Length of elements to permute

Details

n higher than 10 will take a while, or may run out of memory in systems with limited RAM

Value

Matrix where each row is a different permutation

`pfread` *fread delimited file in parts*

Description

fread delimited file in parts

Usage

```
pfread(
  x,
  part_nrows,
  nrows = NULL,
  header = TRUE,
  sep = "auto",
  verbose = TRUE,
  stringsAsFactors = TRUE,
  ...
)
```

Arguments

`x`	Character: Path to delimited file
`part_nrows`	Integer: Number of rows to read in each part
`nrows`	Integer: Number of rows in the file
`header`	Logical: If TRUE, the file is assumed to include a header row
`sep`	Character: Delimiter
`verbose`	Logical: If TRUE, print messages to console
`stringsAsFactors`	Logical: If TRUE, characters will be converted to factors
`...`	Additional arguments to pass to `data.table::fread()`

Author(s)

E.D. Gennatas

plot.massGAM *Plot* massGAM *object*

Description

Plots a massGAM object using dplot3_bar

Usage

```
## S3 method for class 'massGAM'
plot(
  x,
  predictor = NULL,
  main = NULL,
  what = "pvals",
  p.adjust.method = "none",
  p.transform = function(x) -log10(x),
  show = c("all", "signif"),
  pval.hline = c(0.05, 0.001),
  hline.col = NULL,
  hline.dash = "dash",
  hline.annotate = as.character(pval.hline),
  ylim = NULL,
  xlab = NULL,
  ylab = NULL,
  group = NULL,
  grouped.nonsig.alpha = 0.5,
  order.by.group = TRUE,
  palette = rtPalette,
  col.sig = "#43A4AC",
  col.ns = "#7f7f7f",
  theme = rtTheme,
  alpha = NULL,
  margin = NULL,
  displayModeBar = FALSE,
  trace = 0,
  filename = NULL,
  ...
)
```

Arguments

x	massGAM object
xlab	Character: x-axis label for volcano plot

Author(s)

E.D. Gennatas

---

plot.massGLM *Plot* massGLM *object*

Description

Plots a massGLM object using dplot3_bar

Usage

```
## S3 method for class 'massGLM'
plot(
  x,
  predictor = NULL,
  main = NULL,
  what = c("volcano", "coefs", "pvals"),
  p.adjust.method = "holm",
  p.transform = function(x) -log10(x),
  show = c("all", "signif"),
  xnames = NULL,
  pval.hline = c(0.05, 0.001),
  hline.col = "#ffffff",
  hline.dash = "dash",
  hline.annotate = as.character(pval.hline),
  ylim = NULL,
  xlab = NULL,
  ylab = NULL,
  group = NULL,
  col.neg = "#43A4AC",
  col.pos = "#FA9860",
  col.ns = "#7f7f7f",
  theme = rtTheme,
  alpha = NULL,
  volcano.annotate = TRUE,
  volcano.annotate.n = 7,
  volcano.hline = NULL,
  volcano.hline.dash = "dot",
  volcano.hline.annotate = NULL,
  volcano.p.transform = function(x) -log10(x),
  margin = NULL,
  displayModeBar = TRUE,
  trace = 0,
  filename = NULL,
  ...
)
```

Arguments

`x`	`massGLM` object
`what`	Character: "adjusted" or "raw" p-values to plot
`xlab`	Character: x-axis label for volcano plot

Author(s)

E.D. Gennatas

`plot.resample`   `plot` *method for* `resample` *object*

Description

Run mplot3_res on a resample object

Usage

```
## S3 method for class 'resample'
plot(x, col = NULL, ...)
```

Arguments

`x`	Vector; numeric or factor: Outcome used for resampling
`col`	Vector, color
`...`	Additional arguments passed to mplot3_res

Author(s)

E.D. Gennatas

`plot.rtModCVCalibration`   *Plot* `rtModCVCalibration` *object*

Description

Plot Calibration plots and Brier score boxplots for `rtModCVCalibration` object

Usage

```
## S3 method for class 'rtModCVCalibration'
plot(
  x,
  what = c("calibration", "brier"),
  type = c("aggregate.all", "aggregate.by.resample"),
  bin.method = c("quantile", "equidistant"),
  filename = NULL,
  ...
)
```

Arguments

`x`	`rtModCVCalibration` object
`what`	Character: "calibration" or "brier"
`type`	Character: "aggregate.all" or "aggregate.by.resample"
`bin.method`	Character: "quantile" or "equidistant"
`filename`	Character: Path to save plot as pdf
`...`	Additional arguments

Details

For calibration plots, `type = "aggregate.all"` is likely the more informative one. It shows calibrations curves before and after calibration by aggregating across all outer test sets. The `type = "aggregate.by.resample"` option shows the calibration curves after calibration for each outer resample.

For Brier boxplots, `type = "aggregate.all"` shows 1 score per outer resample prior to calibration and multiple (equal n.resamples used in calibrate_cv) brier scores per outer resample after calibration. This is for diagnostic purposes mainly. For presentation, the `type = "aggregate.by.resample"` option shows the mean Brier score per outer resample prior to calibration and after calibration and makes more sense. The uncalibrated estimates could be resampled using the calibration model resamples to produce a more comparable boxplot of Brier scores when using the `type = "aggregate.all"` option, but that seems artifactual.

More options are certainly possibly, e.g. an "aggregate.none" that would show all calibration resamples for all outer ersamples, and can be added in the future if needed.

Value

plotly object

Author(s)

E.D. Gennatas

`plot.rtTest` *Plot* `rtTest` *object*

Description

Plot `rtTest` object

Usage

```
## S3 method for class 'rtTest'
plot(
  x,
  main = NULL,
  mar = NULL,
  uni.type = c("density", "histogram", "hd"),
  boxplot.xlab = FALSE,
  theme = rtTheme,
  par.reset = TRUE,
  ...
)
```

Arguments

`x`	`rtTest` object
`main`	Character: Main title
`theme`	Character: Run `themes()` for available themes

Author(s)

E.D. Gennatas

`plotly.heat` *Heatmap with* `plotly`

Description

Draw a heatmap using `plotly`

Usage

```
plotly.heat(
  z,
  x = NULL,
  y = NULL,
  title = NULL,
  col = penn.heat(21),
  xlab = NULL,
  ylab = NULL,
  zlab = NULL,
  transpose = TRUE
)
```

Arguments

z	Input matrix
x, y	Vectors for x, y axes
title	Plot title
col	Set of colors to make gradient from
xlab	x-axis label
ylab	y-axis label
zlab	z value label
transpose	Logical: If TRUE, transpose matrix

Author(s)

E.D. Gennatas

---

precision *Precision (aka PPV)*

Description

The first factor level is considered the positive case.

Usage

```
precision(true, estimated, harmonize = FALSE, verbosity = 1)
```

Arguments

true	Factor: True labels
estimated	Factor: Estimated labels
harmonize	Logical: If TRUE, run factor_harmonize first
verbosity	Integer: If > 0, print messages to console.

`predict.addtree` *Predict Method for MediBoost Model*

Description

Obtains predictions from a trained MediBoost model

Usage

```
## S3 method for class 'addtree'
predict(object, newdata, verbose = FALSE, ...)
```

Arguments

`object`	A trained model of class "addtree"
`newdata`	Optional: a matrix / data.frame of features with which to predict
`verbose`	Logical: If TRUE, print messages to output
`...`	Not used

Author(s)

E.D. Gennatas

`predict.boost` *Predict method for* `boost` *object*

Description

Predict method for `boost` object

Usage

```
## S3 method for class 'boost'
predict(
  object,
  newdata = NULL,
  n.feat = NCOL(newdata),
  n.iter = NULL,
  as.matrix = FALSE,
  verbose = FALSE,
  n.cores = rtCores,
  ...
)
```

Arguments

`object`	boost object
`newdata`	data.frame: New data to predict on
`n.feat`	Integer: Number of features to use from `newdata`
`n.iter`	Integer: Number of iterations to use
`as.matrix`	Logical: If TRUE, return matrix of predictions for each iteration, otherwise return vector
`verbose`	Logical: If TRUE, print messages to console
`n.cores`	Integer: Number of cores to use
`...`	Not used

Author(s)

E.D. Gennatas

`predict.cartLite` *Predict method for* `cartLite` *object*

Description

Predict method for `cartLite` object

Usage

```
## S3 method for class 'cartLite'
predict(object, newdata, verbose = FALSE, ...)
```

Arguments

`object`	`cartLite` object
`newdata`	Data frame of predictors
`verbose`	Logical: If TRUE, print messages to console.
`...`	Unused

Author(s)

E.D. Gennatas

predict.cartLiteBoostTV

Predict method for `cartLiteBoostTV` *object*

Description

Predict method for `cartLiteBoostTV` object

Usage

```
## S3 method for class 'cartLiteBoostTV'
predict(
  object,
  newdata = NULL,
  n.feat = NCOL(newdata),
  n.iter = NULL,
  as.matrix = FALSE,
  verbose = FALSE,
  n.cores = rtCores,
  ...
)
```

Arguments

`object`	`cartLiteBoostTV` object
`newdata`	Set of predictors
`n.feat`	Integer: N of features to use. Default = NCOL(newdata)
`n.iter`	Integer: N of iterations to predict from. Default = (all available)
`as.matrix`	Logical: If TRUE, return predictions from each iterations. Default = FALSE
`verbose`	Logical: If TRUE, print messages to console. Default = FALSE
`n.cores`	Integer: Number of cores to use. Default = `rtCores`
`...`	Unused

Author(s)

E.D. Gennatas

predict.glmLite *Predict method for* `glmLite` *object*

Description

Predict method for `glmLite` object

Usage

```
## S3 method for class 'glmLite'
predict(object, newdata, verbose = FALSE, ...)
```

Arguments

`object`	glmLite object
`newdata`	Data frame of predictors
`verbose`	Logical: If TRUE, print messages to console. Default = FALSE
`...`	Unused

Author(s)

E.D. Gennatas

predict.glmLiteBoostTV *Predict method for* `glmLiteBoostTV` *object*

Description

Predict method for `glmLiteBoostTV` object

Usage

```
## S3 method for class 'glmLiteBoostTV'
predict(
  object,
  newdata = NULL,
  n.feat = NCOL(newdata),
  n.iter = NULL,
  as.matrix = FALSE,
  verbose = FALSE,
  n.cores = rtCores,
  ...
)
```

Arguments

`object`	`glmLiteBoostTV` object
`newdata`	Set of predictors
`n.feat`	Integer: N of features to use.
`n.iter`	Integer: N of iterations to predict from.
`as.matrix`	Logical: If TRUE, return predictions from each iteration.
`verbose`	Logical: If TRUE, print messages to console.
`n.cores`	Integer: Number of cores to use.
`...`	Unused

Author(s)

E.D. Gennatas

`predict.hytboost` *Predict method for* `hytboost` *object*

Description

Predict method for `hytboost` object

Usage

```
## S3 method for class 'hytboost'
predict(
  object,
  newdata = NULL,
  n.iter = NULL,
  fixed.cxr = NULL,
  as.matrix = FALSE,
  n.cores = 1,
  verbose = FALSE,
  ...
)
```

Arguments

`object`	`hytboost` object
`newdata`	data.frame of predictors
`n.iter`	Integer: Use the first so many trees for prediction
`fixed.cxr`	(internal use) Matrix: Cases by rules to use instead of matching cases to rules using `newdata`
`as.matrix`	Logical: If TRUE, output
`n.cores`	Integer: Number of cores to use
`verbose`	Logical: If TRUE, print messages to console
`...`	Not used

Author(s)

E.D. Gennatas

---

predict.hytboostnow *Predict method for* hytboostnow *object*

Description

Predict method for hytboostnow object

Usage

```
## S3 method for class 'hytboostnow'
predict(
  object,
  newdata = NULL,
  n.feat = NCOL(newdata),
  n.iter = NULL,
  fixed.cxr = NULL,
  as.matrix = FALSE,
  n.cores = 1,
  verbose = FALSE,
  ...
)
```

Arguments

object	hytboostnow object

Author(s)

E.D. Gennatas

---

predict.hytreenow *Predict method for* hytreeLite *object*

Description

Predict method for hytreeLite object

Usage

```
## S3 method for class 'hytreenow'
predict(
  object,
  newdata,
  n.feat = NCOL(newdata),
  fixed.cxr = NULL,
  cxr.newdata = NULL,
  cxr = FALSE,
  cxrcoef = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

`object`	`hytreenow`
`newdata`	Data frame of predictors
`n.feat`	(internal use) Integer: Use first `n.feat` columns of newdata to predict. Defaults to all
`fixed.cxr`	(internal use) Matrix: Cases by rules to use instead of matching cases to rules using `newdata`
`cxr.newdata`	(internal use) Data frame: Use these values to match cases by rules
`cxr`	Logical: If TRUE, return list which includes cases-by-rules matrix along with predicted values
`cxrcoef`	Logical: If TRUE, return cases-by-rules * coefficients matrix along with predicted values
`verbose`	Logical: If TRUE, print messages to console
`...`	Not used

Author(s)

E.D. Gennatas

`predict.hytreew` *Predict method for* `hytreew` *object*

Description

Predict method for `hytreew` object

Usage

```
## S3 method for class 'hytreew'
predict(
  object,
  newdata,
  n.feat = NCOL(newdata),
  fixed.cxr = NULL,
  cxr.newdata = NULL,
  cxr = FALSE,
  cxrcoef = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

`object`	`hytreew`
`newdata`	Data frame of predictors
`n.feat`	(internal use) Integer: Use first `n.feat` columns of newdata to predict. Defaults to all
`fixed.cxr`	(internal use) Matrix: Cases by rules to use instead of matching cases to rules using `newdata`
`cxr.newdata`	(internal use) Data frame: Use these values to match cases by rules
`cxr`	Logical: If TRUE, return list which includes cases-by-rules matrix along with predicted values
`cxrcoef`	Logical: If TRUE, return cases-by-rules * coefficients matrix along with predicted values
`verbose`	Logical: If TRUE, print messages to console
`...`	Not used

Author(s)

E.D. Gennatas

`predict.LightRuleFit` `predict` *method for* `LightRuleFit` *object*

Description

`predict` method for `LightRuleFit` object

Usage

```
## S3 method for class 'LightRuleFit'
predict(
  object,
  newdata = NULL,
  return.cases.by.rules = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

`object`	`LightRuleFit` object
`newdata`	Feature matrix / data.frame: will be converted to `data.table`
`return.cases.by.rules`	Logical: If TRUE, return cases by rules matrix
`verbose`	Logical: If TRUE, print messages during execution. Default = TRUE
`...`	Ignored

Value

Vector of estimated values

---

predict.lihad — *Predict method for* `lihad` *object*

Description

Predict method for `lihad` object

Usage

```
## S3 method for class 'lihad'
predict(
  object,
  newdata = NULL,
  learning.rate = NULL,
  n.feat = NULL,
  verbose = FALSE,
  cxrcoef = FALSE,
  ...
)
```

Arguments

`object`	an `rtMod` trained with s_LIHAD or an `lihad` object
`newdata`	data frame of predictor features
`learning.rate`	Float: learning rate if `object` was `lihad`
`n.feat`	Integer: internal use only
`verbose`	Logical: If TRUE, print messages to console.
`cxrcoef`	Logical: If TRUE, return matrix of cases by coefficients along with predictions.
`...`	Not used

Author(s)

E.D. Gennatas

`predict.linadleaves` *Predict method for* `linadleaves` *object*

Description

Predict method for `linadleaves` object

Usage

```
## S3 method for class 'linadleaves'
predict(
  object,
  newdata,
  type = c("response", "probability", "all", "step"),
  n.leaves = NULL,
  fixed.cxr = NULL,
  cxr.newdata = NULL,
  cxr = FALSE,
  cxrcoef = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

`object`	`shytreeRaw`
`newdata`	Data frame of predictors
`type`	Character: "response", "probability", "all", "step"
`n.leaves`	Integer: Use the first `n.leaves` of the tree for prediction
`fixed.cxr`	(internal use) Matrix: Cases by rules to use instead of matching cases to rules using `newdata`

`cxr.newdata` (internal use) Data frame: Use these values to match cases by rules

`cxr` Logical: If TRUE, return list which includes cases-by-rules matrix along with predicted values

`cxrcoef` Logical: If TRUE, return cases-by-rules * coefficients matrix along with predicted values

`verbose` Logical: If TRUE, print messages to console

`...` Not used

Author(s)

E.D. Gennatas

`predict.nlareg` *Predict method for* `nlareg` *object*

Description

Predict method for `nlareg` object

Usage

```
## S3 method for class 'nlareg'
predict(object, newdata, ...)
```

Arguments

`object` nlareg object

`newdata` Data frame of predictors

`...` Unused

Author(s)

E.D. Gennatas

predict.nullmod *rtemis internal: predict for an object of class* `nullmod`

Description

rtemis internal: predict for an object of class `nullmod`

Usage

```
## S3 method for class 'nullmod'
predict(object, newdata = NULL, ...)
```

Arguments

`object`	Object of class `nullmod`
`newdata`	Not used
`...`	Not used

predict.rtBSplines *Predict S3 method for* `rtBSplines`

Description

Predict S3 method for `rtBSplines`

Usage

```
## S3 method for class 'rtBSplines'
predict(object, newdata = NULL, ...)
```

Arguments

`object`	`rtBSplines` object created by dat2bsplinemat
`newdata`	`data.frame` of new data.
`...`	Not used.

Author(s)

E.D. Gennatas

predict.rtModCVCalibration — *Predict using calibrated model*

Description

Predict using a calibrated model returned by calibrate_cv

Usage

```
## S3 method for class 'rtModCVCalibration'
predict(object, mod, newdata, ...)
```

Arguments

`object`	`rtModCVCalibration` object returned by calibrate_cv
`mod`	`rtModCV` object returned by train_cv
`newdata`	Data frame: New data to predict on
`...`	Additional arguments - Use to define `which.repeat`, which should be an integer defining which repeat to use for prediction. Defaults to 1 if not specified.

Author(s)

EDG

predict.rtTLS — `predict.rtTLS`: `predict` *method for* `rtTLS` *object*

Description

`predict.rtTLS`: `predict` method for `rtTLS` object

Usage

```
## S3 method for class 'rtTLS'
predict(object, newdata, ...)
```

Arguments

`object`	`rtTLS` object created by s_TLS
`newdata`	`data.frame` of new data.
`...`	Not used.

predict.rulefit — `predict` *method for* `rulefit` *object*

Description

`predict` method for `rulefit` object

Usage

```
## S3 method for class 'rulefit'
predict(object, newdata = NULL, verbose = TRUE, ...)
```

Arguments

`object`	`rulefit` object
`newdata`	Feature matrix / data.frame: will be converted to `data.table`
`verbose`	Logical: If TRUE, print messages during execution. Default = TRUE
`...`	Ignored

Value

Vector of estimated values

preprocess — *Data preprocessing*

Description

Prepare data for analysis and visualization

Usage

```
preprocess(
  x,
  completeCases = FALSE,
  removeCases.thres = NULL,
  removeFeatures.thres = NULL,
  missingness = FALSE,
  impute = FALSE,
  impute.type = c("missRanger", "micePMM", "meanMode"),
  impute.missRanger.params = list(pmm.k = 3, maxiter = 10, num.trees = 500),
  impute.discrete = get_mode,
  impute.numeric = mean,
  integer2factor = FALSE,
  integer2numeric = FALSE,
```

```
  logical2factor = FALSE,
  logical2numeric = FALSE,
  numeric2factor = FALSE,
  numeric2factor.levels = NULL,
  numeric.cut.n = 0,
  numeric.cut.labels = FALSE,
  numeric.quant.n = 0,
  numeric.quant.NAonly = FALSE,
  len2factor = 0,
  character2factor = FALSE,
  factorNA2missing = FALSE,
  factorNA2missing.level = "missing",
  factor2integer = FALSE,
  factor2integer_startat0 = TRUE,
  scale = FALSE,
  center = scale,
  removeConstants = FALSE,
  removeConstants.skipMissing = TRUE,
  removeDuplicates = FALSE,
  oneHot = FALSE,
  exclude = NULL,
  xname = NULL,
  verbose = TRUE
)
```

Arguments

`x` data.frame to be preprocessed

`completeCases` Logical: If TRUE, only retain complete cases (no missing data). Default = FALSE

`removeCases.thres`
Float (0, 1): Remove cases with >= to this fraction of missing features.

`removeFeatures.thres`
Float (0, 1): Remove features with missing values in >= to this fraction of cases.

`missingness` Logical: If TRUE, generate new boolean columns for each feature with missing values, indicating which cases were missing data.

`impute` Logical: If TRUE, impute missing cases. See `impute.discrete` and `impute.numeric` for how

`impute.type` Character: How to impute data: "missRanger" and "missForest" use the packages of the same name to impute by iterative random forest regression. "rfImpute" uses `randomForest::rfImpute` (see its documentation), "meanMode" will use mean and mode by default or any custom function defined in `impute.discrete` and `impute.numeric`. Default = "missRanger" (which is much faster than "missForest"). "missForest" is included for compatibility with older pipelines.

`impute.missRanger.params`
Named list with elements "pmm.k" and "maxiter", which are passed to `missRanger::missRanger`. `pmm.k` greater than 0 results in predictive mean matching. Default `pmm.k = 3`

maxiter = 10 num.trees = 500. Reduce num.trees for faster imputation especially in large datasets. Set pmm.k = 0 to disable predictive mean matching to missForest::missForest

impute.discrete
: Function that returns single value: How to impute discrete variables for impute.type = "meanMode". Default = get_mode

impute.numeric
: Function that returns single value: How to impute continuous variables for impute.type = "meanMode". Default = mean

integer2factor
: Logical: If TRUE, convert all integers to factors. This includes bit64::integer64 columns

integer2numeric
: Logical: If TRUE, convert all integers to numeric (will only work if integer2factor = FALSE) This includes bit64::integer64 columns

logical2factor
: Logical: If TRUE, convert all logical variables to factors

logical2numeric
: Logical: If TRUE, convert all logical variables to numeric

numeric2factor
: Logical: If TRUE, convert all numeric variables to factors

numeric2factor.levels
: Character vector: Optional - will be passed to levels arg of factor() if numeric2factor = TRUE (For advanced/ specific use cases; need to know unique values of numeric vector(s) and given all numeric vars have same unique values)

numeric.cut.n
: Integer: If > 0, convert all numeric variables to factors by binning using base::cut with breaks equal to this number

numeric.cut.labels
: Logical: The labels argument of base::cut

numeric.quant.n
: Integer: If > 0, convert all numeric variables to factors by binning using base::cut with breaks equal to this number of quantiles produced using stats::quantile

numeric.quant.NAonly
: Logical: If TRUE, only bin numeric variables with missing values

len2factor
: Integer (>=2): Convert all variables with less than or equal to this number of unique values to factors. Default = NULL. For example, if binary variables are encoded with 1, 2, you could use len2factor = 2 to convert them to factors.

character2factor
: Logical: If TRUE, convert all character variables to factors

factorNA2missing
: Logical: If TRUE, make NA values in factors be of level factorNA2missing.level. In many cases this is the preferred way to handle missing data in categorical variables. Note that since this step is performed before imputation, you can use this option to handle missing data in categorical variables and impute numeric variables in the same preprocess call.

factorNA2missing.level
: Character: Name of level if factorNA2missing = TRUE. Default = "missing"

factor2integer
: Logical: If TRUE, convert all factors to integers

factor2integer_startat0
: Logical: If TRUE, start integer coding at 0

scale
: Logical: If TRUE, scale columns of x

center
: Logical: If TRUE, center columns of x. Note that by default it is the same as scale

removeConstants
: Logical: If TRUE, remove constant columns.

removeConstants.skipMissing
: Logical: If TRUE, skip missing values, before checking if feature is constant

removeDuplicates
: Logical: If TRUE, remove duplicate cases.

oneHot
: Logical: If TRUE, convert all factors using one-hot encoding

exclude
: Integer, vector: Exclude these columns from preprocessing.

xname
: Character: Name of x for messages

verbose
: Logical: If TRUE, write messages to console.

Details

Order of operations (reflected by order of arguments in usage):

- keep complete cases only
- remove constants
- remove duplicates
- remove cases by missingness threshold
- remove features by missingness threshold
- integer to factor
- integer to numeric
- logical to factor
- logical to numeric
- numeric to factor
- cut numeric to n bins
- cut numeric to n quantiles
- numeric with less than N unique values to factor
- character to factor
- factor NA to named level
- add missingness column
- impute
- scale and/or center
- one-hot encoding

Author(s)

E.D. Gennatas

preprocess_ *Data preprocessing (in-place)*

Description

Prepare data for analysis and visualization

Usage

```
preprocess_(
  x,
  removeFeatures.thres = NULL,
  missingness = FALSE,
  integer2factor = FALSE,
  integer2numeric = FALSE,
  logical2factor = FALSE,
  logical2numeric = FALSE,
  numeric2factor = FALSE,
  numeric2factor.levels = NULL,
  len2factor = 0,
  character2factor = FALSE,
  factorNA2missing = FALSE,
  factorNA2missing.level = "missing",
  scale = FALSE,
  center = scale,
  removeConstants = FALSE,
  oneHot = FALSE,
  exclude = NULL,
  verbose = TRUE
)
```

Arguments

`x`	data.frame or data.table to be preprocessed. If data.frame, will be converted to data.table in-place of missing features.
`removeFeatures.thres`	Float (0, 1): Remove features with missing values in >= to this fraction of cases.
`missingness`	Logical: If TRUE, generate new boolean columns for each feature with missing values, indicating which cases were missing data.
`integer2factor`	Logical: If TRUE, convert all integers to factors
`integer2numeric`	Logical: If TRUE, convert all integers to numeric (will only work if `integer2factor = FALSE`)
`logical2factor`	Logical: If TRUE, convert all logical variables to factors
`logical2numeric`	Logical: If TRUE, convert all logical variables to numeric

`numeric2factor` Logical: If TRUE, convert all numeric variables to factors

`numeric2factor.levels`
Character vector: Optional - If `numeric2factor = TRUE`, use these levels for all numeric variables.

`len2factor` Integer (>=2): Convert all numeric variables with less than or equal to this number of unique values to factors. For example, if binary variables are encoded with 1, 2, you could use `len2factor = 2` to convert them to factors. If race is encoded with 6 integers, you can use 6.

`character2factor`
Logical: If TRUE, convert all character variables to factors

`factorNA2missing`
Logical: If TRUE, make NA values in factors be of level `factorNA2missing.level`. In many cases this is the preferred way to handle missing data in categorical variables. Note that since this step is performed before imputation, you can use this option to handle missing data in categorical variables and impute numeric variables in the same `preprocess` call.

`factorNA2missing.level`
Character: Name of level if `factorNA2missing = TRUE`.

`scale` Logical: If TRUE, scale columns of `x`

`center` Logical: If TRUE, center columns of `x`

`removeConstants`
Logical: If TRUE, remove constant columns.

`oneHot` Logical: If TRUE, convert all factors using one-hot encoding

`exclude` Integer, vector: Exclude these columns from preprocessing.

`verbose` Logical: If TRUE, write messages to console.

Details

This function (ending in "_") performs operations **in-place** and returns the preprocessed data.table silently (e.g. for piping). Note that imputation is not currently supported - use preprocess for imputation.

Order of operations is the same as the order of arguments in usage:

- keep complete cases only
- remove duplicates
- remove cases by missingness threshold
- remove features by missingness threshold
- integer to factor
- integer to numeric
- logical to factor
- logical to numeric
- numeric to factor
- numeric with less than N unique values to factor

- character to factor
- factor NA to named level
- add missingness column
- scale and/or center
- remove constants
- one-hot encoding

Author(s)

E.D. Gennatas

Examples

```
## Not run:
x <- data.table(a = sample(c(1:3), 30, T),
b = rnorm(30, 12),
c = rnorm(30, 200),
d = sample(c(21:22), 30, T),
e = rnorm(30, -100),
f = rnorm(30, 950),
g = rnorm(30),
h = rnorm(30))
## add duplicates
x <- rbind(x, x[c(1, 3), ])
## add constant
x[, z := 99]
preprocess_(x)

## End(Not run)
```

`present` *Present elevate models*

Description

Plot training and testing performance boxplots of multiple 'rtModCV" objects created by train_cv using dplot3_box

Usage

```
present(
  ...,
  mod.names = NULL,
  which.repeat = 1,
  metric = NULL,
  plot.train = TRUE,
  plot.test = TRUE,
```

```
    boxpoints = "all",
    annotate_meansd = TRUE,
    main = NULL,
    ylim = NULL,
    htest = "none",
    htest.annotate.y = NULL,
    col = NULL,
    theme = rtTheme,
    margin = list(b = 65, l = 100, t = 60, r = 18, pad = 0),
    subplot.margin = 0.0666,
    filename = NULL,
    file.width = 500,
    file.height = 550,
    file.scale = 1
)
```

Arguments

`...`	rtModCV objects created with train_cv
`mod.names`	Character: Names of models being plotted.
`which.repeat`	Integer: which `rtModCV` repeat to plot.
`metric`	Character: which metric to plot.
`plot.train`	Logical: If TRUE, plot training performance.
`plot.test`	Logical: If TRUE, plot testing performance.
`boxpoints`	Character or FALSE: "all", "suspectedoutliers", "outliers" See `https://plotly.com/r/box-plots/#choosing-the-algorithm-for-computing-quartiles`
`annotate_meansd`	Logical: If TRUE, annotate with mean (SD) of each box
`main`	Character: Plot title.
`ylim`	Numeric vector: y-axis limits
`htest`	Character: e.g. "t.test", "wilcox.test" to compare each box to the *first* box. If grouped, compare within each group to the first box. If p-value of test is less than `htest.thresh`, add asterisk above/ to the side of each box
`htest.annotate.y`	Numeric: y-axis paper coordinate for htest annotation
`col`	Color, vector: Color for boxes. If NULL, which will draw colors from `palette`
`theme`	Character: Theme to use: Run `themes()` for available themes
`margin`	Named list: plot margins. Default = `list(b = 65, l = 65, t = 50, r = 10, pad = 0)`
`subplot.margin`	Numeric: margin between subplots.
`filename`	Character: Path to file to save static plot.
`file.width`	Integer: File width in pixels for when `filename` is set.
`file.height`	Integer: File height in pixels for when `filename` is set.
`file.scale`	Numeric: If saving to file, scale plot by this number

Author(s)

E.D. Gennatas

`present_gridsearch` *Present gridsearch results*

Description

Present gridsearch results

Usage

```
present_gridsearch(x, rtModCV.repeat = 1, ...)
```

Arguments

`x`	rtMod or rtModCV objects
`rtModCV.repeat`	Integer: Which repeat to use, when x is rtModCV object
`...`	Additional arguments to pass to `knitr::kable` used to print the best tune table

Author(s)

E.D. Gennatas

Examples

```
## Not run:
mod <- s_CART(iris, cp = c(0, .1), maxdepth = c(3, 5))
mod_10ss <- elevate(iris, mod = "cart", cp = c(0, .1), maxdepth = c(3, 5))
present_gridsearch(mod)
present_gridsearch(mod_10ss)

## End(Not run)
```

`previewcolor` *Preview color v2.0*

Description

Preview one or multiple colors using little rhombi with their little labels up top

Usage

```
previewcolor(
  x,
  main = NULL,
  bg = "#333333",
  main.col = "#b3b3b3",
  main.x = 0.7,
  main.y = 0.2,
  main.adj = 0,
  main.cex = 0.9,
  main.font = 1,
  width = NULL,
  xlim = NULL,
  ylim = c(0, 2.2),
  asp = 1,
  labels.y = 1.55,
  label.cex = NULL,
  mar = c(0, 0, 0, 1),
  par.reset = TRUE,
  filename = NULL,
  pdf.width = 8,
  pdf.height = 2.5
)
```

Arguments

`x`	Color, vector: One or more colors that R understands
`main`	Character: Title. Default = NULL, which results in `deparse(substitute(x))`
`bg`	Background color.
`main.col`	Color: Title color
`main.x`	Float: x coordinate for `main`.
`main.y`	Float: y coordinate for `main`.
`main.adj`	Float: `adj` argument to mtext for `main`.
`main.cex`	Float: character expansion factor for `main`. Default = .9
`main.font`	Integer, 1 or 2: Weight of `main` 1: regular, 2: bold. Default = 2
`width`	Float: Plot width. Default = NULL, i.e. set automatically
`xlim`	Vector, length 2: x-axis limits. Default = NULL, i.e. set automatically
`ylim`	Vector, length 2: y-axis limits.
`asp`	Float: Plot aspect ratio.
`labels.y`	Float: y coord for labels. Default = 1.55 (rhombi are fixed and range y .5 - 1.5)
`label.cex`	Float: Character expansion for labels. Default = NULL, and is calculated automatically based on length of `x`
`mar`	Numeric vector, length 4: margin size.

par.reset	Logical: If TRUE, reset par settings on exit.
filename	Character: Path to save plot as PDF.
pdf.width	Numeric: Width of PDF in inches.
pdf.height	Numeric: Height of PDF in inches.

Value

Nothing, prints plot

Examples

```
colors <- colorgradient.x(seq(-5, 5))
previewcolor(colors)
```

---

print.addtree *Print method for* addtree *object created using* s_AddTree

---

Description

Print method for addtree object created using s_AddTree

Usage

```
## S3 method for class 'addtree'
print(x, ...)
```

Arguments

x	rtMod object created using s_AddTree
...	Not used

Author(s)

E.D. Gennatas

print.boost *Print method for boost object*

Description

Print method for boost object

Usage

```
## S3 method for class 'boost'
print(x, ...)
```

Arguments

x	boost object
...	Not used

Author(s)

E.D. Gennatas

print.cartLiteBoostTV *Print method for cartLiteBoostTV object*

Description

Print method for cartLiteBoostTV object

Usage

```
## S3 method for class 'cartLiteBoostTV'
print(x, ...)
```

Arguments

x	`cartLiteBoostTV` object
...	Additional arguments

Author(s)

E.D. Gennatas

print.CheckData *Print* CheckData *object*

Description

Print CheckData object

Usage

```
## S3 method for class 'CheckData'
print(
  x,
  type = c("plaintext", "html"),
  name = NULL,
  check_integers = FALSE,
 css = list(font.family = "Helvetica", color = "#fff", background.color = "#242424"),
  ...
)
```

Arguments

x	CheckData object.
type	Character: Output type: "plaintext" or "html".
name	Character: Dataset name.
check_integers	Logical: If TRUE and there are integer features, prints a message to consider converting to factors.
css	List with font.family, color, and background.color elements.
...	Not used.

Author(s)

E.D. Gennatas

print.class_error *Print class_error*

Description

Print class_error

Usage

```
## S3 method for class 'class_error'
print(x, decimal.places = 4, ...)
```

Arguments

x	Object of type class_error
decimal.places	Integer: Number of decimal places to print
...	Not used

Author(s)

E.D. Gennatas

---

print.glmLiteBoostTV — *Print method for* `glmLiteBoostTV` *object*

Description

Print method for `glmLiteBoostTV` object

Usage

```
## S3 method for class 'glmLiteBoostTV'
print(x, ...)
```

Arguments

x	`glmLiteBoostTV` object
...	Not used

Author(s)

E.D. Gennatas

---

print.gridSearch — `print` *method for* `gridSearch` *object*

Description

`print` method for `gridSearch` object

Usage

```
## S3 method for class 'gridSearch'
print(x, ...)
```

Arguments

x	Object of class `gridSearch` created by `gridSearchLearn`
...	Unused

Author(s)

E.D. Gennatas

`print.hytboost` *Print method for* `hytboost` *object*

Description

Print method for `hytboost` object

Usage

```
## S3 method for class 'hytboost'
print(x, ...)
```

Arguments

`x`	`hytboost` object
`...`	Not used

Author(s)

E.D. Gennatas

`print.hytboostnow` *Print method for* `boost` *object*

Description

Print method for `boost` object

Usage

```
## S3 method for class 'hytboostnow'
print(x, ...)
```

Arguments

`x`	`hytboostnow` object
`...`	Not used

Author(s)

E.D. Gennatas

print.lihad *Print method for* `lihad` *object*

Description

Print method for `lihad` object

Usage

```
## S3 method for class 'lihad'
print(x, ...)
```

Arguments

`x`	`lihad` object
`...`	Not used

Author(s)

E.D. Gennatas

print.linadleaves *Print method for* `linadleaves` *object*

Description

Print method for `linadleaves` object

Usage

```
## S3 method for class 'linadleaves'
print(x, ...)
```

Arguments

`x`	`linadleaves` object
`...`	Not used

Author(s)

E.D. Gennatas

print.massGAM — `print`*massGAM object*

Description

`print`massGAM object

Usage

```
## S3 method for class 'massGAM'
print(x, ...)
```

Arguments

`x`	massGAM object
`...`	Not used

Author(s)

E.D. Gennatas

print.massGLM — `print`*massGLM object*

Description

`print`massGLM object

Usage

```
## S3 method for class 'massGLM'
print(x, ...)
```

Arguments

`x`	massGLM object
`...`	Not used

Author(s)

E.D. Gennatas

print.regError — *Print* regError *object*

Description

print regError object

Usage

```
## S3 method for class 'regError'
print(x, ...)
```

Arguments

x	regError object
...	Not used

Author(s)

E.D. Gennatas

print.resample — print *method for resample object*

Description

Print resample information

Usage

```
## S3 method for class 'resample'
print(x, ...)
```

Arguments

x	resample object
...	Not used

Author(s)

E.D. Gennatas

print.rtBiasVariance — *Print method for bias_variance*

Description

Print method for bias_variance

Usage

```
## S3 method for class 'rtBiasVariance'
print(x, ...)
```

Arguments

x	Output of bias_variance
...	Not used

Author(s)

E.D. Gennatas

print.rtDecom — *print.rtDecom: print method for rtDecom object*

Description

print.rtDecom: print method for rtDecom object

Usage

```
## S3 method for class 'rtDecom'
print(x, ...)
```

Arguments

x	rtDecom object
...	Not used

print.rtTLS — print.rtTLS: print method for rtTLS object

Description

print.rtTLS: print method for rtTLS object

Usage

```
## S3 method for class 'rtTLS'
print(x, ...)
```

Arguments

x	rtTLS object created by s_TLS
...	Not used.

print.surv_error — Print surv_error

Description

Print surv_error

Usage

```
## S3 method for class 'surv_error'
print(x, decimal.places = 4, ...)
```

Arguments

x	Object of type surv_error
decimal.places	Integer: Number of decimal places to print. Default = 4
...	Not used

Author(s)

E.D. Gennatas

prune.addtree — *Prune AddTree tree*

Description

Prune an AddTree tree in Node format using `data.tree` to remove sister nodes with same class estimate.

Usage

```
prune.addtree(
  addtree,
  prune.empty.leaves = TRUE,
  remove.bad.parents = TRUE,
  verbose = TRUE
)
```

Arguments

`addtree`	rtMod trained with s_AddTree
`prune.empty.leaves`	Logical: If TRUE, remove leaves with 0 cases.
`remove.bad.parents`	Logical: If TRUE, remove nodes with no siblings but children and give their children to their parent.
`verbose`	Logical: If TRUE, print messages to console.

Author(s)

E.D. Gennatas

psd — *Population Standard Deviation*

Description

Estimate the population standard deviation:

$$sqrt(mean(x^2) - mean(x)^2)$$

Usage

```
psd(x)
```

Arguments

`x`	Numeric vector

Details

This will be particularly useful when the machines finally collect data on all humans. Caution is advised, however, as you never know how many may be hiding underground.

Value

Population standard deviation

Author(s)

E.D. Gennatas

`qstat` *SGE qstat*

Description

Run SGE qstat

Usage

```
qstat()
```

Details

alias for `system("qstat")`

`read` *Read tabular data from a variety of formats*

Description

Read data and optionally clean column names, keep unique rows, and convert characters to factors

Usage

```
read(
  filename,
  datadir = NULL,
  make.unique = TRUE,
  character2factor = FALSE,
  clean.colnames = TRUE,
  delim.reader = c("data.table", "vroom", "duckdb", "arrow"),
  xlsx.sheet = 1,
  sep = NULL,
  quote = "\"",
  na.strings = c(""),
  output = c("data.table", "default"),
  attr = NULL,
  value = NULL,
  verbose = TRUE,
  fread_verbose = FALSE,
  timed = verbose,
  ...
)
```

Arguments

`filename`	Character: filename or full path if `datadir = NULL`
`datadir`	Character: Optional path to directory where `filename` is located. If not specified, `filename` must be the full path.
`make.unique`	Logical: If TRUE, keep unique rows only
`character2factor`	Logical: If TRUE, convert character variables to factors
`clean.colnames`	Logical: If TRUE, clean columns names using clean_colnames
`delim.reader`	Character: package to use for reading delimited data
`xlsx.sheet`	Integer or character: Name or number of XLSX sheet to read
`sep`	Single character: field separator. If `delim.reader = "fread"` and `sep = NULL`, this defaults to "auto", otherwise defaults to ","
`quote`	Single character: quote character
`na.strings`	Character vector: Strings to be interpreted as NA values. For `delim.reader = "duckdb"`, this must be a single string.
`output`	Character: "default" or "data.table", If default, return the delim.reader's default data structure, otherwise convert to data.table
`attr`	Character: Attribute to set (Optional)
`value`	Character: Value to set (if `attr` is not NULL)
`verbose`	Logical: If TRUE, print messages to console
`fread_verbose`	Logical: Passed to `data.table::fread`
`timed`	Logical: If TRUE, time the process and print to console

`...`	Additional parameters to pass to `data.table::fread`, `arrow::read_delim_arrow()`, `vroom::vroom()`, or `openxlsx::read.xlsx()`

Details

`read` is a convenience function to read:

- **RDS** files using `readRDS()`
- **Parquet** files using `arrow::read_parquet()`
- **XLSX** files using `readxl::read_excel()`
- **DTA** files from Stata using `haven::read_dta()`
- **Delimited** files using `data.table:fread()`, `arrow:read_delim_arrow()`, `vroom::vroom()`, `duckdb::duckdb_read_csv()`
- **FASTA** files using `seqinr::read.fasta()`

Author(s)

E.D. Gennatas

Examples

```
## Not run:
datadir <- "~/icloud/Data"
dat <- read("iris.csv", datadir)

## End(Not run)
```

`read_config` *Read rtemis configuration file*

Description

Reads rtemis configuration file.

Usage

```
read_config(config.path, verbose = TRUE)
```

Arguments

`file`	Character: Path to configuration file created by create_config.

Value

List.

Author(s)

EDG

recycle *Recycle values of vector to match length of target*

Description

Recycle values of vector to match length of target

Usage

```
recycle(x, target)
```

Arguments

x	Vector to be recycled
target	Object whose length defines target length

Author(s)

E.D. Gennatas

reg_error *Regression Error Metrics*

Description

Calculate error metrics for regression

Usage

```
reg_error(
  x,
  y,
  rho = FALSE,
  tau = FALSE,
  pct.red = FALSE,
  na.rm = FALSE,
  verbosity = 0
)
```

Arguments

`x`	Numeric vector: True values
`y`	Numeric vector: Predicted values
`rho`	Logical: If TRUE, calculate Spearman's rho
`tau`	Logical: If TRUE, calculate Kendall's tau
`pct.red`	Logical: If TRUE, calculate percent reduction in error
`na.rm`	Logical: If TRUE, remove NA values before computation
`verbosity`	Integer: If > 0, print messages to console

Value

Object of class `regError`

Author(s)

E.D. Gennatas

`relu` *ReLU - Rectified Linear Unit*

Description

ReLU - Rectified Linear Unit

Usage

```
relu(x)
```

Arguments

`x`	Numeric: Input

resample *Resampling methods*

Description

Create resamples of your data, e.g. for model building or validation. "bootstrap" gives the standard bootstrap, i.e. random sampling with replacement, using bootstrap, "strat.sub" creates stratified subsamples using strat.sub, while "strat.boot" uses strat.boot which runs strat.sub and then randomly duplicates some of the training cases to reach original length of input (default) or length defined by `target.length`.

Usage

```
resample(
  y,
  n.resamples = 10,
  resampler = c("strat.sub", "strat.boot", "kfold", "bootstrap", "loocv"),
  index = NULL,
  group = NULL,
  stratify.var = y,
  train.p = 0.75,
  strat.n.bins = 4,
  target.length = NROW(y),
  id.strat = NULL,
  rtset = NULL,
  seed = NULL,
  verbosity = TRUE
)
```

Arguments

`y`	Vector or data.frame: Usually the outcome; `NROW(y)` defines sample size
`n.resamples`	Integer: Number of training/testing sets required
`resampler`	Character: Type of resampling to perform: "bootstrap", "kfold", "strat.boot", "strat.sub".
`index`	List where each element is a vector of training set indices. Use this for manual/pre-defined train/test splits
`group`	Integer, vector, length = `length(y)`: Integer vector, where numbers define fold membership. e.g. for 10-fold on a dataset with 1000 cases, you could use `group = rep(1:10, each = 100)`
`stratify.var`	Numeric vector (optional): Variable used for stratification.
`train.p`	Float (0, 1): Fraction of cases to assign to traininig set for `resampler = "strat.sub"`
`strat.n.bins`	Integer: Number of groups to use for stratification for `resampler = "strat.sub" / "strat.boot"`
`target.length`	Integer: Number of cases for training set for `resampler = "strat.boot"`.

`id.strat`	Vector of IDs which may be replicated: resampling should force replicates of each ID to only appear in the training or testing.
`rtset`	List: Output of an setup.resample (or named list with same structure). NOTE: Overrides all other arguments. Default = NULL
`seed`	Integer: (Optional) Set seed for random number generator, in order to make output reproducible. See `?base::set.seed`
`verbosity`	Logical: If TRUE, print messages to console

Details

`resample` is used by multiple **rtemis** learners, `gridSearchLearn`, and train_cv. Note that option 'kfold', which uses kfold results in resamples of slightly different length for y of small length, so avoid all operations which rely on equal-length vectors. For example, you can't place resamples in a data.frame, but must use a list instead.

Author(s)

E.D. Gennatas

See Also

train_cv

Examples

```
y <- rnorm(200)
# 10-fold (stratified)
res <- resample(y, 10, "kfold")
# 25 stratified subsamples
res <- resample(y, 25, "strat.sub")
# 100 stratified bootstraps
res <- resample(y, 100, "strat.boot")
```

`reverseLevels` *Reverse factor levels*

Description

Reverse the order of a factor's levels

Usage

```
reverseLevels(x)
```

Arguments

`x`	Factor

Author(s)

E.D. Gennatas

`revfactorlevels` *Reverse factor level order*

Description

Reverse factor level order

Usage

```
revfactorlevels(x)
```

Arguments

`x`	factor

Author(s)

E.D. Gennatas

`rfVarSelect` *Variable Selection by Random Forest*

Description

Select important variables from a set of features based on RF-estimated variable importance

Usage

```
rfVarSelect(x, y, p = 0.2, print.plot = TRUE, verbose = TRUE)
```

Arguments

`x`	Predictors
`y`	outcome
`p`	Float (0, 1): Fraction of variables in x to select. `p * ncol(x)`. May help to set to a fraction twice what you expect to be the true fraction of useful variables, to reduce false negatives at the expense of false positives which can be dealt by an appropriate learning algorithm.
`print.plot`	Logical: If TRUE, print plot of variable importance
`verbose`	Logical: If TRUE, print messages to console.

Details

Please note that this function is included for academic and exploratory purposes. It may be best to rely on each supervised learning algorithm's own variable selection approach.

Author(s)

E.D. Gennatas

---

rnormmat *Random Normal Matrix*

Description

Create a matrix or data frame of defined dimensions, whose columns are random normal vectors

Usage

```
rnormmat(
  nrow = 10,
  ncol = 10,
  mean = 0,
  sd = 1,
  return.df = FALSE,
  seed = NULL
)
```

Arguments

nrow	Integer: Number of rows. Default = 10
ncol	Integer: Number of columns. Default = 10
mean	Float: Mean. Default = 0
sd	Float: Standard deviation. Default = 1
return.df	Logical: If TRUE, return data.frame, otherwise matrix. Default = TRUE
seed	Integer: Set seed for `rnorm`. Default = NULL

Author(s)

E.D. Gennatas

rowMax — *Collapse data.frame to vector by getting row max*

Description

Collapse data.frame to vector by getting row max

Usage

```
rowMax(x, na.rm = TRUE)
```

Arguments

x	Input vector
na.rm	Logical. If TRUE, missing values are not considered.

Author(s)

E.D. Gennatas

rsd — *Coefficient of Variation (Relative standard deviation)*

Description

Calculates the coefficient of variation, also known as relative standard deviation, which is given by

$$sd(x)/mean(x)$$

Usage

```
rsd(x, as.percentage = TRUE, na.rm = TRUE, adjust = FALSE, adjust.lo = 1)
```

Arguments

x	Numeric: Input
as.percentage	Logical: If TRUE, multiply by 100
na.rm	Logical: If TRUE, remove missing values before computation
adjust	Logical: If TRUE, if `x` contains values < `adjust.lo`, x will be shifted up by adding its minimum
adjust.lo	Float: Threshold to be used if `adjust = TRUE`

Details

This is not meaningful if mean is close to 0. For such cases, set `adjust = TRUE`. This will add `min(x)` to x

Examples

```
## Not run:
mplot3_x(sapply(1:100, function(x) cov(rnorm(100))), 'd', xlab = 'rnorm(100) x 100 times')
# cov of rnorm without adjustment is all over the place
mplot3_x(sapply(1:100, function(x) cov(rnorm(100), adjust = T)), 'd',
xlab = 'rnorm(100) x 100 times')
# COV after shifting above 1 is what you probably want

## End(Not run)
```

`rsq` *R-squared*

Description

R-squared

Usage

```
rsq(x, y)
```

Arguments

`x`	Float, vector: True values
`y`	Float, vector: Estimated values

Author(s)

E.D. Gennatas

`rstudio_theme_rtemis` *Apply rtemis theme for RStudio*

Description

Apply the rtemis RStudio theme, an adaptation of the rscodeio theme (https://github.com/anthonynorth/rscodeio) Recommended to use the Fira Code font with the theme (https://fonts.google.com/specimen/Fira+Code?query=fira+code)

Usage

```
rstudio_theme_rtemis(theme = "dark")
```

Arguments

`theme`	Character: "dark" or "light"

Author(s)

E.D. Gennatas

rtClust-methods *rtClust S3 methods*

Description

S3 methods for `rtClust` class.

Usage

```
## S3 method for class 'rtClust'
print(x, ...)
```

Arguments

x	`rtClust` object
...	Not used

rtemis_palette *Access rtemis palette colors*

Description

Allows you to get `n` colors of a defined palette, useful for passing to other functions, like ggplot

Usage

```
rtemis_palette(n, palette = rtPalette)
```

Arguments

n	Integer: Number of colors to output
palette	Character: Palette to use. See available options with `rtpalette()`. Default = `rtPalette`

Author(s)

E.D. Gennatas

Examples

```
rtemis_palette(3)
```

`rtInitProjectDir` *Initialize Project Directory*

Description

Initializes Directory Structure: "R", "Data", "Results"

Usage

```
rtInitProjectDir(verbose = TRUE)
```

Arguments

`verbose`	Logical, If TRUE, print messages to console

Author(s)

E.D. Gennatas

`rtlayout` *Create multipanel plots with the* `mplot3` *family*

Description

Set layout for drawing multiple plots in the same view

Usage

```
rtlayout(
  nrows = NULL,
  ncols = NULL,
  byrow = FALSE,
  autolabel = FALSE,
  pdf.width = NULL,
  pdf.height = NULL,
  filename = NULL
)
```

Arguments

`nrows`	Integer: N of rows
`ncols`	Integer: N of columns
`byrow`	Logical: If TRUE, draw add plots by row Default = FALSE
`autolabel`	Logical: If TRUE, place letter labels on the top left corner of each figure. Default = FALSE

`pdf.width`	Float: Width of PDF to save, if `filename` is provided. Default = `ncols * 4.5`
`pdf.height`	Float: Height of PDF to save, if `filename` is provided. Default = `nrows * 4.5`
`filename`	String, optional: Save multiplot to file. Default = NULL

Author(s)

E.D. Gennatas

`rtMeta-methods` *rtMeta S3 methods*

Description

S3 methods for `rtMeta` class that differ from those of the `rtMod` superclass

Usage

```
## S3 method for class 'rtMeta'
predict(object, newdata, fn = median, ...)
```

Arguments

`object`	`rtMeta` object
`newdata`	Testing set features
`fn`	Function to average predictions
`...`	Additional arguments passed to `predict(object)`

`rtMod-methods` `rtMod` *S3 methods*

Description

S3 methods for `rtMod` class. Excludes functions `print` and `plot` defined within the `rtMod` class itself.

Get coefficients or relative variable importance for `rtMod` object

Usage

```
## S3 method for class 'rtMod'
print(x, ...)

## S3 method for class 'rtMod'
fitted(object, ...)

## S3 method for class 'rtMod'
predict(
  object,
  newdata,
  classification.output = c("prob", "class"),
  trace = 0,
  verbose = TRUE,
  ...
)

## S3 method for class 'rtMod'
residuals(object, ...)

## S3 method for class 'rtMod'
plot(x, estimate = NULL, theme = rtTheme, filename = NULL, ...)

## S3 method for class 'rtMod'
summary(
  object,
  plots = TRUE,
  cex = 1,
  fit.true.line = "lm",
  resid.fit.line = "gam",
  fit.legend = TRUE,
  se.fit = TRUE,
  single.fig = TRUE,
  summary = TRUE,
  theme = rtTheme,
  title.col = NULL,
  ...
)

## S3 method for class 'rtMod'
coef(object, verbose = TRUE, ...)

## S3 method for class 'rtModLite'
predict(object, newdata, ...)
```

Arguments

x	`rtMod` object

...	Additional argument passed to `predict(object)`
`object`	`rtModLite` object
`newdata`	Testing set features
`classification.output`	Character: "prob" or "class" for classification models
`trace`	Integer: Set trace level
`verbose`	Logical: If TRUE, output messages to console
`estimate`	Character: "fitted" or "predicted"
`theme`	Character: theme to use. Options: "box", "darkbox", "light", "dark"
`filename`	Character: Path to file to save plot
`plots`	Logical: If TRUE, print plots. Default = TRUE
`cex`	Float: Character expansion factor
`fit.true.line`	**rtemis** algorithm to use for fitted vs. true line Options: `select_learn()`
`resid.fit.line`	**rtemis** algorithm to use for residuals vs. fitted line. Options: `select_learn()`
`fit.legend`	Logical: If TRUE, print fit legend. Default = TRUE
`se.fit`	Logical: If TRUE, plot 2 * standard error bands. Default = TRUE
`single.fig`	Logical: If TRUE, draw all plots in a single figure. Default = TRUE
`summary`	Logical: If TRUE, print summary. Default = TRUE
`title.col`	Color for main title

Author(s)

E.D. Gennatas

`rtModBag-methods` *rtModBag S3 methods*

Description

S3 methods for `rtModBag` class that differ from those of the `rtMod` superclass

Usage

```
## S3 method for class 'rtModBag'
predict(object, newdata, aggr.fn = NULL, n.cores = 1, verbose = FALSE, ...)
```

Arguments

`object`	`rtModBag` object
`newdata`	Testing set features
`aggr.fn`	Character: Function to aggregate models' prediction. If NULL, defaults to "median"
`n.cores`	Integer: Number of cores to use
`verbose`	Logical: If TRUE, print messages to console.
...	Not used

`rtModClass-class` **rtemis** *Classification Model Class*

Description

rtemis Classification Model Class

rtemis Classification Model Class

Details

R6 Class for **rtemis** Classification Models

Super class

`rtemis::rtMod` -> `rtModClass`

Public fields

`fitted.prob` Training set probability estimates

`predicted.prob` Testing set probability estimates

Methods

Public methods:

- `rtModClass$new()`
- `rtModClass$plotROC()`
- `rtModClass$plotROCfitted()`
- `rtModClass$plotROCpredicted()`
- `rtModClass$plotPR()`
- `rtModClass$plotPRfitted()`
- `rtModClass$plotPRpredicted()`
- `rtModClass$clone()`

Method `new()`**:** Initialize `rtModClass` object

Usage:

```
rtModClass$new(
  mod.name = character(),
  y.train = numeric(),
  y.test = numeric(),
  x.name = character(),
  y.name = character(),
  xnames = character(),
  mod = list(),
  type = character(),
  gridsearch = NULL,
```

```
  parameters = list(),
  fitted = numeric(),
  fitted.prob = numeric(),
  se.fit = numeric(),
  error.train = list(),
  predicted = NULL,
  predicted.prob = NULL,
  se.prediction = NULL,
  error.test = NULL,
  varimp = NULL,
  question = character(),
  extra = list()
)
```

Arguments:

`mod.name` Character: Algorithm name

`y.train` Training set output

`y.test` Testing set output

`x.name` Character: Feature set name

`y.name` Character: Output name

`xnames` Character vector: Feature names

`mod` Trained model

`type` Character: Type of model (Regression, Classification, Survival)

`gridsearch` Grid search output

`parameters` List of training parameters

`fitted` Fitted values (training set predictions)

`fitted.prob` Training set probability estimates

`se.fit` Standard error of the fit

`error.train` Training set error

`predicted` Predicted values (Testing set predictions)

`predicted.prob` Testing set probability estimates

`se.prediction` Testting set standard error

`error.test` Testing set error

`varimp` Variable importance

`question` Question the model is trying to answer

`extra` List of extra model info

`sessionInfo` R session info at time of training

Method `plotROC()`**:** plot ROC. Uses testing set if available, otherwise training

Usage:

```
rtModClass$plotROC(theme = rtTheme, filename = NULL, ...)
```

Arguments:

`theme` Theme to pass to plotting function

`filename` Character: Path to file to save plot

`...` Extra arguments to pass to plotting function

Method `plotROCfitted()`**:** Plot training set ROC

Usage:

```
rtModClass$plotROCfitted(
  main = "ROC Training",
  theme = rtTheme,
  filename = NULL,
  ...
)
```

Arguments:

`main` Character: Main title

`theme` Theme to pass to plotting function

`filename` Character: Path to file to save plot

`...` Extra arguments to pass to plotting function

Method `plotROCpredicted()`**:** plot testing set ROC

Usage:

```
rtModClass$plotROCpredicted(
  main = "ROC Testing",
  theme = rtTheme,
  filename = NULL,
  ...
)
```

Arguments:

`main` Character: Main title

`theme` Theme to pass to plotting function

`filename` Character: Path to file to save plot

`...` Extra arguments to pass to plotting function

Method `plotPR()`**:** plot Precision-Recall curve. Uses testing set if available, otherwise training

Usage:

```
rtModClass$plotPR(theme = rtTheme, filename = NULL, ...)
```

Arguments:

`theme` Theme to pass to plotting function

`filename` Character: Path to file to save plot

`...` Extra arguments to pass to plotting function

Method `plotPRfitted()`**:** Plot training set Precision-Recall curve.

Usage:

```
rtModClass$plotPRfitted(
  main = "P-R Training",
  theme = rtTheme,
  filename = NULL,
  ...
)
```

Arguments:

`main` Character: Main title

`theme` Theme to pass to plotting function

`filename` Character: Path to file to save plot

`...` Extra arguments to pass to plotting function

Method `plotPRpredicted()`**:** plot testing set Precision-Recall curve.

Usage:

```
rtModClass$plotPRpredicted(
  main = "P-R Testing",
  theme = rtTheme,
  filename = NULL,
  ...
)
```

Arguments:

`main` Character: Main title

`theme` Theme to pass to plotting function

`filename` Character: Path to file to save plot

`...` Extra arguments to pass to plotting function

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
rtModClass$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`rtModCV-methods` *S3 methods for* `rtModCV` *class that differ from those of the* `rtMod` *superclass*

Description

S3 methods for `rtModCV` class that differ from those of the `rtMod` superclass

`plot.rtModCV`: `plot` method for `rtModCV` object

`summary.rtModCV`: `summary` method for `rtModCV` object

`predict.rtModCV`: `predict` method for `rtModCV` object

`describe` method for `rtModCV` object

Usage

```
## S3 method for class 'rtModCV'
plot(x, ...)

## S3 method for class 'rtModCV'
summary(object, ...)

## S3 method for class 'rtModCV'
predict(
  object,
  newdata,
  which.repeat = 1,
  classification.output = c("prob", "class"),
  output = c("array", "avg"),
  ...
)

## S3 method for class 'rtModCV'
describe(object, ...)
```

Arguments

`x`	`rtModCV` object
`...`	Not used
`object`	`rtModCV` object
`newdata`	Set of predictors to use
`which.repeat`	Integer: Which repeat to use for prediction
`classification.output`	Character: "prob" or "class" for classification models If "class" and `output` is "avg", the mode of the predictions is returned.
`output`	Character: "matrix" or "avg". Produce either a matrix with predictions of each model in different columns, or the mean/mode of the predictions across models
`n.cores`	Integer: Number of cores to use

`rtModLite-methods` *rtModLite S3 methods*

Description

S3 methods for `rtModLite` class.

Usage

```
## S3 method for class 'rtModLite'
print(x, ...)
```

Arguments

x	`rtModLite` object
...	Not used

`rtModLog-class` **rtemis** *Supervised Model Log Class*

Description

rtemis Supervised Model Log Class

rtemis Supervised Model Log Class

Public fields

`mod.name` Learner algorithm name

`parameters` List of hyperparameters used when building model

`error.train` Training error

`error.test` Testing error

`sessionInfo` The output of `sessionInfo()` at the time the model was trained

Methods

Public methods:

- `rtModLog$new()`
- `rtModLog$print()`
- `rtModLog$clone()`

Method `new()`**:** Initialize `rtModLog` object

Usage:

```
rtModLog$new(
  mod.name = character(),
  parameters = list(),
  error.train = list(),
  error.test = NULL
)
```

Arguments:

`mod.name` Learner algorithm name

`parameters` List of hyperparameters used when building model

`error.train` Training error

`error.test` Testing error

Method `print()`**:** Print method for `rtModLog` object

Usage:

```
rtModLog$print()
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
rtModLog$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Author(s)

E.D. Gennatas

`rtModLogger-class` **rtemis** *model logger*

Description

rtemis model logger

rtemis model logger

Details

R6 class to save trained models' parameters and performance. Keep your experiment results tidy in one place, with an option to write out to a multi-sheet Excel file.

Public fields

`mods` List of trained models

Methods

Public methods:

- `rtModLogger$new()`
- `rtModLogger$print()`
- `rtModLogger$add()`
- `rtModLogger$summarize()`
- `rtModLogger$summary()`
- `rtModLogger$tabulate()`
- `rtModLogger$plot()`
- `rtModLogger$clone()`

Method `new()`: Initialize `rtModLogger` object

Usage:

```
rtModLogger$new(mods = list())
```

Arguments:

`mods` List of trained models

Method `print()`**:** Print method for `rtModLogger` object

Usage:

```
rtModLogger$print()
```

Method `add()`**:** Add model to logger

Usage:

```
rtModLogger$add(mod, verbose = TRUE)
```

Arguments:

`mod` Model to add

`verbose` Logical: If TRUE, print messages to console

Method `summarize()`**:** Summary method for `rtModLogger`

Usage:

```
rtModLogger$summarize(
  class.metric = "Balanced Accuracy",
  reg.metric = "Rsq",
  surv.metric = "Coherence",
  decimal.places = 3,
  print.metric = FALSE
)
```

Arguments:

`class.metric` Character: Metric to use for Classification models

`reg.metric` Character: Metric to use for Regression models

`surv.metric` Character: Metric to use for Survival models

`decimal.places` Integer: Number of decimal places to display

`print.metric` Logical: If TRUE, print metric name

Method `summary()`**:** Summary method for `rtModLogger`

Usage:

```
rtModLogger$summary(
  class.metric = "Balanced Accuracy",
  reg.metric = "Rsq",
  surv.metric = "Coherence"
)
```

Arguments:

`class.metric` Character: Metric to use for Classification models

`reg.metric` Character: Metric to use for Regression models

`surv.metric` Character: Metric to use for Survival models

Method `tabulate()`**:** Tabulate models' parameters and performance

Usage:

```
rtModLogger$tabulate(filename = NULL)
```

Arguments:

`filename` Character: Path to file to save parameters and performance - will be saved as .xlsx file with multiple sheets

Method `plot()`**:** Plot method for `rtModLogger`

Usage:

```
rtModLogger$plot(
  names = NULL,
  col = unlist(rtpalette(rtPalette)),
  mar = NULL,
  ...
)
```

Arguments:

`names` Character: Model names

`col` Colors to use

`mar` Float, vector: plot margins

`...` Additional arguments to pass to plotting function

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
rtModLogger$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Author(s)

E.D. Gennatas

`rtpalette` **rtemis** *Color Palettes*

Description

`rtpalette()` prints names of available color palettes Each palette is a named list of hexadecimal color definitions which can be used with any graphics function. `rtpalette(palette_name)` returns a list of colors for a given palette.

Usage

```
rtpalette(palette = NULL, verbose = TRUE)
```

Arguments

palette	Character: Name of palette to return. Default = NULL: available palette names are printed and no palette is returned
verbose	Logical: If TRUE, print messages to console

Value

A list of available palettes, invisibly

Examples

```
rtpalette("imperial")
```

---

rtPalettes *UCSF Colors*

Description

ucsfCol: UCSF color palette (https://identity.ucsf.edu/brand-guide/color)

ucsfPalette: Subset of ucsfCol

ucdCol: UC Davis color palette (https://marketingtoolbox.ucdavis.edu/visual-identity/color.html)

berkeleyCol: Berkeley color palette (https://brand.berkeley.edu/colors/)

ucscCol: UC Santa Cruz color palette (https://communications.ucsc.edu/visual-design/color/)

ucmercedCol: UC Merced color palette (https://publicrelations.ucmerced.edu/color-guidelines)

ucsbCol: UC Santa Barbara color palette (https://www.ucsb.edu/visual-identity/color)

uclaCol: UCLA color palette (http://brand.ucla.edu/identity/colors)

ucrCol: UC Riverside color palette (https://brand.ucr.edu/ucr-colors)

uciCol: UCI color palette (https://communications.uci.edu/campus-resources/graphic-standards/colors.php)

ucsdCol: UC San Diego color palette (https://ucpa.ucsd.edu/brand/elements/color-palette/)

californiaCol: University of California color palette (http://brand.universityofcalifornia.edu/guidelines/color.html#!primar colors)

stanfordCol: Stanford color palette (https://identity.stanford.edu/color.html#digital-color)

csuCol: California State University color palette (https://www2.calstate.edu/csu-system/csu-branding-standards/Documents/CSU-Brand-Guidelines-8-2018.pdf)

calpolyCol: Cal Poly color palette (https://universitymarketing.calpoly.edu/brand-guidelines/colors/)

caltechCol: Caltech color palette (http://identity.caltech.edu/colors)

scrippsCol: Scripps Research color palette

pennCol: Penn color palette (http://www.upenn.edu/about/styleguide-color-type)

cmuCol: Carnegie Mellon color palette (https://www.cmu.edu/marcom/brand-standards/web-standards.html#colors)

mitCol: MIT color palette (http://web.mit.edu/graphicidentity/colors.html) princetonCol: Princeton color palette (https://communications.princeton.edu/guides-tools/logo-graphic-identity)

columbiaCol: Columbia color palette (https://visualidentity.columbia.edu/content/web-0)

brownCol: Brown color palette (https://www.brown.edu/university-identity/sites/university-identity/files/Brown_Visual_Ider 07-22.pdf)

yaleCol: Yale color palette (https://yaleidentity.yale.edu/web)

cornellCol: Yale color palette (https://brand.cornell.edu/design-center/colors/

hmsCol: Harvard Medical School color palette (https://identityguide.hms.harvard.edu/color)

dartmouthCol: Dartmouth color palette (https://communications.dartmouth.edu/visual-identity/design-elements/color-palette#web%20palette)

usfCol: USF color palette (https://myusf.usfca.edu/marketing-communications/resources/graphics-resources/brand-standards/color-palette) Color conversions performed using https://www.pantone.com/color-finder/

uwCol: University of Washington color palette (http://www.washington.edu/brand/graphic-elements/primary-color-palette/)

jhuCol: Johns Hopkins University color palette (https://brand.jhu.edu/color/)

nyuCol: NYU color palette (https://www.nyu.edu/employees/resources-and-services/media-and-communications/styleguide/website/graphic-visual-design.html)

washuCol: WashU color palette (https://marcomm.wustl.edu/resources/branding-logo-toolkit/color-palettes/)

chicagoCol: University of Chicago color palette (https://news.uchicago.edu/sites/default/files/attachments/_uchicago.identit

texasCol: Penn State color palette (https://brand.psu.edu/design-essentials.html#color)

sfsuCol: SF State color palette (https://logo.sfsu.edu/color-system)

illinoisCol: University of Illinois color palette (https://www.uillinois.edu/OUR/brand/color_palettes)

umdCol: University of Maryland color palette (https://osc.umd.edu/licensing-trademarks/brand-standards/logos/#color)

msuCol: MSU color palette (https://brand.msu.edu/visual/color-palette)

michiganCol: Michigan color palette (https://brand.umich.edu/design-resources/colors/)

iowaCol: University of Iowa color palette (https://brand.uiowa.edu/color)

texasCol: University of Texas color palette (https://brand.utexas.edu/identity/color/)

emoryCol: Emory color palette (https://brand.emory.edu/color.html)

techCol: Georgia Tech color palette (http://www.licensing.gatech.edu/super-block/239)

vanderbiltCol: Vanderbilt color palette (https://www.vanderbilt.edu/communications/brand/color.php)

jeffersonCol: Jefferson color palette (http://creative.jefferson.edu/downloads/Jefferson-Brand-Guidelines.pdf)

hawaiiCol: University of Hawaii color palette (https://www.hawaii.edu/offices/eaur/graphicsstandards.pdf)

nihCol: NIH color palette (https://www.nlm.nih.gov/about/nlm_logo_guidelines_030414_508.pdf)

imperialCol: Imperial College London colour palette (https://www.imperial.ac.uk/brand-style-guide/visual-identity/brand-colours/)

uclCol: UCL colour palette (https://www.ucl.ac.uk/cam/brand/guidelines/colour) oxfordCol: Oxford University colour palette (https://www.ox.ac.uk/sites/files/oxford/media_wysiwyg/Oxford%20Blue%20

nhsCol: NHS colour palette (https://www.england.nhs.uk/nhsidentity/identity-guidelines/colours/)

ubcCol: UBC color palette (http://assets.brand.ubc.ca/downloads/ubc_colour_guide.pdf)

torontoCol: U Toronto color palette (https://trademarks.utoronto.ca/colors-fonts/)

mcgillCol: McGill color palette (https://www.mcgill.ca/visual-identity/visual-identity-guide)

ethCol: ETH color palette (https://ethz.ch/services/en/service/communication/corporate-design/colour.html)

rwthCol: RWTH Aachen color palette (http://www9.rwth-aachen.de/global/show_document.asp?id=aaaaaaaaaadpbhq)

mozillaCol: Mozilla design colors (https://mozilla.design/mozilla/color/)

firefoxCol: Firefox design colors (https://mozilla.design/firefox/color/)

appleCol: Apple Human Interface Guidelines color palette (https://developer.apple.com/design/human-interface-guidelines/ios/visual-design/color/)

googleCol: Google brand palette (https://brandpalettes.com/google-colors/)

amazonCol: Amazon brand palette (https://images-na.ssl-images-amazon.com/images/G/01/AdvertisingSite/pdfs/AmazonBr

microsoftCol: Microsoft brand palette (https://brandcolors.net/b/microsoft)

Usage

```
ucsfLegacyCol

ucsfPalette

ucdCol

berkeleyCol

ucscCol

ucmercedCol

ucsbCol

uclaCol

ucrColor

uciCol

ucsdCol

californiaCol

stanfordCol

csuCol
```

calpolyCol

caltechCol

scrippsCol

pennCol

pennPalette

pennLightPalette

cmuCol

mitCol

princetonCol

columbiaCol

brownCol

yaleCol

cornellCol

hmsCol

dartmouthCol

usfCol

uwCol

jhuCol

nyuCol

washuCol

chicagoCol

pennstateCol

sfsuCol

illinoisCol

umdCol

msuCol

michiganCol

iowaCol

texasCol

emoryCol

techCol

vanderbiltCol

jeffersonCol

hawaiiCol

nihCol

imperialCol

uclCol

oxfordCol

nhsCol

ubcCol

torontoCol

mcgillCol

ethCol

rwthCol

mozillaCol

firefoxCol

appleCol

googleCol

`amazonCol`

`microsoftCol`

Format

An object of class `list` of length 13.

An object of class `list` of length 8.

An object of class `list` of length 18.

An object of class `list` of length 18.

An object of class `list` of length 9.

An object of class `list` of length 7.

An object of class `list` of length 10.

An object of class `list` of length 13.

An object of class `list` of length 3.

An object of class `list` of length 8.

An object of class `list` of length 11.

An object of class `list` of length 17.

An object of class `list` of length 27.

An object of class `list` of length 3.

An object of class `list` of length 18.

An object of class `list` of length 19.

An object of class `list` of length 6.

An object of class `list` of length 30.

An object of class `list` of length 11.

An object of class `list` of length 5.

An object of class `list` of length 8.

An object of class `list` of length 3.

An object of class `list` of length 2.

An object of class `list` of length 19.

An object of class `list` of length 8.

An object of class `list` of length 10.

An object of class `list` of length 14.

An object of class `list` of length 14.

An object of class `list` of length 14.

An object of class `list` of length 3.

An object of class `list` of length 3.

An object of class `list` of length 21.

An object of class `list` of length 17.

An object of class `list` of length 18.

An object of class `list` of length 25.

An object of class `list` of length 20.

An object of class `list` of length 9.

An object of class `list` of length 15.

An object of class `list` of length 3.

An object of class `list` of length 6.

An object of class `list` of length 15.

An object of class `list` of length 9.

An object of class `list` of length 10.

An object of class `list` of length 22.

An object of class `list` of length 3.

An object of class `list` of length 9.

An object of class `list` of length 8.

An object of class `list` of length 11.

An object of class `list` of length 2.

An object of class `list` of length 26.

An object of class `list` of length 23.

An object of class `list` of length 24.

An object of class `list` of length 21.

An object of class `list` of length 6.

An object of class `list` of length 2.

An object of class `list` of length 27.

An object of class `list` of length 10.

An object of class `list` of length 60.

An object of class `list` of length 8.

An object of class `list` of length 8.

An object of class `list` of length 8.

An object of class `list` of length 4.

An object of class `list` of length 2.

An object of class `list` of length 4.

rtROC *Build an ROC curve*

Description

Calculate the points of an ROC curve and the AUC

Usage

```
rtROC(
  true.labels,
  predicted.probabilities,
  thresholds = NULL,
  plot = FALSE,
  theme = rtTheme,
  verbose = TRUE
)
```

Arguments

`true.labels`	Factor with true labels
`predicted.probabilities`	Numeric vector of predicted probabilities / estimated score
`thresholds`	Numeric vector of thresholds to consider
`plot`	Logical: If TRUE, print plot
`theme`	rtemis theme to use
`verbose`	Logical: If TRUE, print messages to console

Details

`true.labels` should be a factor (will be coerced to one) where the first level is the "positive" case. `predicted.probabilities` should be a vector of floats 0, 1 where `[0, .5)` corresponds to the first level and `[.5, 1]` corresponds to the second level. predicted.probabilities

Author(s)

E.D. Gennatas

`rtset` **rtemis** *default-setting functions*

Description

These functions output lists of default settings for different **rtemis** functions. This removes the need of passing named lists of arguments, and provides autocompletion, making it easier to setup functions without having to refer to the manual.

Author(s)

E.D. Gennatas

`rtversion` *Get rtemis and OS version info*

Description

Get rtemis and OS version info

Usage

```
rtversion()
```

`rtXDecom-class` *R6 class for* **rtemis** *cross-decompositions*

Description

R6 class for **rtemis** cross-decompositions

R6 class for **rtemis** cross-decompositions

Details

rtemis cross-decomposition R6 object

Public fields

`xdecom.name` Character: Name of cross-decomposition algorithm

`k` Integer: Number of projections

`xnames` Character vector: Column names of x

`znames` Character vector: Column names of z

`xdecom` Cross-decomposition model output

`xprojections.train` x data training set projections

`xprojections.test` x data test set data projections

`zprojections.train` z data training set projections

`zprojections.test` z data test set projections

`parameters` Cross-decomposition parameters

`extra` List: Algorithm-specific output

Methods

Public methods:

- `rtXDecom$new()`
- `rtXDecom$print()`
- `rtXDecom$clone()`

Method `new()`**:**

Usage:

```
rtXDecom$new(
  xdecom.name = character(),
  k = integer(),
  xnames = character(),
  znames = character(),
  xdecom = list(),
  xprojections.train = numeric(),
  xprojections.test = numeric(),
  zprojections.train = numeric(),
  zprojections.test = numeric(),
  parameters = list(),
  extra = list()
)
```

Arguments:

`xdecom.name` Character: Name of cross-decomposition algorithm

`k` Integer: Number of projections

`xnames` Character vector: Column names of x

`znames` Character vector: Column names of z

`xdecom` Cross-decomposition model output

`xprojections.train` x data training set projections

`xprojections.test` x data test set data projections

`zprojections.train` z data training set projections
`zprojections.test` z data test set projections
`parameters` Cross-decomposition parameters
`extra` List: Algorithm-specific output

Method `print()`**:** Print method for `rtXDecom` objects

Usage:

```
rtXDecom$print()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
rtXDecom$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Author(s)

E.D. Gennatas

`rt_reactable` *View table using reactable*

Description

View table using reactable

Usage

```
rt_reactable(
  x,
  datatypes = NULL,
  lightsout = TRUE,
  bg = "#121212",
  pagination = TRUE,
  searchable = TRUE,
  bordered = TRUE,
  ...
)
```

Arguments

`x`	data.frame, data.table or similar
`datatypes`	Character vector: Data types of columns in x, e.g. `c("numeric", "factor", "character")`
`lightsout`	Logical: If TRUE, use dark theme.

`bg`	Background color.
`pagination`	Logical: If TRUE, paginate table.
`searchable`	Logical: If TRUE, add search box.
`bordered`	Logical: If TRUE, add border.
`...`	Additional arguments passed to `reactable::reactable`

Author(s)

E D Gennatas

`rt_save` *Write **rtemis** model to RDS file*

Description

Write **rtemis** model to RDS file

Usage

```
rt_save(rtmod, outdir, file.prefix = "s_", verbose = TRUE)
```

Arguments

`rtmod`	**rtemis** model
`outdir`	Path to output directory
`file.prefix`	Character: Prefix for filename
`verbose`	Logical: If TRUE, print messages to output

Author(s)

E.D. Gennatas

ruleDist *Rule distance*

Description

Calculate pairwise distance among a set of rules or between two sets of rules, where each rule defines a subpopulation

Usage

```
ruleDist(
  x,
  rules1,
  rules2 = NULL,
  print.plot = TRUE,
  plot.type = c("static", "interactive"),
  heat.lo = "black",
  heat.mid = NA,
  heat.hi = "#F48024",
  verbose = TRUE
)
```

Arguments

`x`	Data frame / matrix: Input features (cases by features)
`rules1`	Character, vector: Rules as combination of conditions on the features of `x`
`rules2`	String, vector, Optional: Rules as combination of conditions on the features of `x`
`print.plot`	Logical: If TRUE, plot heatmap for calculated distance
`plot.type`	Character: "static", "interactive": type of graphics to use, base or plotly, respectively. Default = "static"
`heat.lo`	Color: Heatmap low color. Default = "black"
`heat.mid`	Color: Heatmap mid color. Default = NA (i.e. create gradient from `heat.lo` to `heat.hi`)
`heat.hi`	Colo: Heatmap hi colo. Default = "#F48024" (orange)
`verbose`	Logical: If TRUE, print console messages. Default = TRUE

Details

If only rules1 is provided, computes pairwise distance among rules1, otherwise computes pairwise distance between rules1 and rules2

Author(s)

E.D. Gennatas

rules2medmod — *Convert rules from cutoffs to median/mode and range*

Description

Convert rules from cutoffs to `median (range)` and `mode (range)` format

Usage

```
rules2medmod(rules, x, .ddSci = TRUE, verbose = TRUE, trace = 0)
```

Arguments

`rules`	Character, vector: Input rules
`x`	Data frame: Data to evaluate rules
`.ddSci`	Logical: If TRUE, format all continuous variables using ddSci, which will give either 2 decimal places, or scientific notation if two decimal places result in 0.00
`verbose`	Logical: If TRUE, print messages to console.
`trace`	Integer: If greater than zero, print progress

Author(s)

E.D. Gennatas

runifmat — *Random Uniform Matrix*

Description

Create a matrix or data frame of defined dimensions, whose columns are random uniform vectors

Usage

```
runifmat(
  nrow = 10,
  ncol = 10,
  min = 0,
  max = 1,
  return.df = FALSE,
  seed = NULL
)
```

Arguments

nrow	Integer: Number of rows.
ncol	Integer: Number of columns.
min	Float: Min.
max	Float: Max.
return.df	Logical: If TRUE, return data.frame, otherwise matrix.
seed	Integer: Set seed for rnorm.

Author(s)

E.D. Gennatas

---

savePMML — *Save rtemis model to PMML file*

Description

Save rtemis model to PMML file

Usage

```
savePMML(
  x,
  filename,
  transforms = NULL,
  model_name = NULL,
  model_version = NULL,
  description = NULL,
  copyright = NULL,
  ...
)
```

Arguments

x	rtemis model
filename	Character: path to file
transforms	List of PMML transformations
model_name	Character: name of the model
model_version	Character: version of the model
description	Character: description of the model
copyright	Character: copyright information
...	Additional arguments passed to pmml::pmml()

Author(s)

E.D. Gennatas

`se` *Extract standard error of fit from rtemis model*

Description

Returns mod$se.fit

Usage

```
se(x)
```

Arguments

`x`	An `rtMod` object

Value

Standard error of fitted values of `mod`

Author(s)

E.D. Gennatas

`selectiter` *Select N of learning iterations based on loss*

Description

Select N of learning iterations based on loss

Usage

```
selectiter(
  loss.valid,
  loss.train,
  smooth = TRUE,
  plot = FALSE,
  verbose = FALSE
)
```

Arguments

`loss.valid`	Float, vector: Validation loss. Can be NULL
`loss.train`	Float, vector: Training loss
`smooth`	Logical: If TRUE, smooth loss before finding minimum.
`plot`	Logical: If TRUE, plot loss curve.
`verbose`	Logical: If TRUE, print messages to console.

Author(s)

E.D. Gennatas

`select_clust` *Select* **rtemis** *Clusterer*

Description

Accepts clusterer name (supports abbreviations) and returns **rtemis** function name or the function itself. If run with no parameters, prints list of available algorithms.

Usage

```
select_clust(clust, fn = FALSE, desc = FALSE)
```

Arguments

`clust`	Character: Clustering algorithm name. Case insensitive, supports partial matching. e.g. "hop" for HOPACH
`fn`	Logical: If TRUE, return function, otherwise name of function.
`desc`	Logical: If TRUE, return full name of algorithm `clust`

Value

Name of function (Default) or function (`fn=TRUE`) or full name of algorithm (`desc=TRUE`)

Author(s)

E.D. Gennatas

`select_decom` *Select* **rtemis** *Decomposer*

Description

Accepts decomposer name (supports abbreviations) and returns **rtemis** function name or the function itself. If run with no parameters, prints list of available algorithms.

Usage

```
select_decom(decom, fn = FALSE, desc = FALSE)
```

Arguments

`decom`	Character: Decomposition name. Case insensitive. e.g. "iso" for isomap
`fn`	Logical: If TRUE, return function, otherwise name of function. Defaults to FALSE
`desc`	Logical: If TRUE, return full name of algorithm `decom`

Value

Function or name of function (see param `fn`) or full name of algorithm (`desc`)

Author(s)

E.D. Gennatas

`select_learn` *Select* **rtemis** *Learner*

Description

Accepts learner name (supports abbreviations) and returns **rtemis** function name or the function itself. If run with no parameters, prints list of available algorithms.

Usage

```
select_learn(alg, fn = FALSE, name = FALSE, desc = FALSE)
```

Arguments

`alg`	Character: Model name. Case insensitive. e.g. "XGB" for xgboost
`fn`	Logical: If TRUE, return function, otherwise name of function. Defaults to FALSE
`name`	Logical: If TRUE, return canonical name of algorithm `alg`
`desc`	Logical: If TRUE, return full name / description of algorithm `alg`

Value

function or name of function (see param `fn`) or short algorithm name (`name = TRUE`) or full algorithm name (`desc = TRUE`)

Author(s)

E.D. Gennatas

sensitivity *Sensitivity*

Description

The first factor level is considered the positive case.

Usage

```
sensitivity(true, estimated, harmonize = FALSE, verbosity = 1)
```

Arguments

`true`	True labels
`estimated`	Estimated labels
`harmonize`	Logical: If TRUE, run factor_harmonize first
`verbosity`	Integer: If > 0, print messages to console.

seql *Sequence generation with automatic cycling*

Description

Sequence generation with automatic cycling

Usage

```
seql(x, target)
```

Arguments

`x`	R object of some `length`
`target`	R object of some `length`

Author(s)

E.D. Gennatas

Examples

```
color <- c("red", "blue")
target <- 1:5
color[seql(color, target)]
# "red"  "blue" "red"  "blue" "red"
color <- c("red", "green", "blue", "yellow", "orange")
target <- 1:3
color[seql(color, target)]
# "red"   "green" "blue"
```

setdiffsym — *Symmetric Set Difference*

Description

Symmetric Set Difference

Usage

```
setdiffsym(x, y)
```

Arguments

x	vector
y	vector of same type as x

Author(s)

E.D. Gennatas

Examples

```
setdiff(1:10, 1:5)
setdiff(1:5, 1:10)
setdiffsym(1:10, 1:5)
setdiffsym(1:5, 1:10)
```

setup.bag.resample — *Set resample parameters for rtMod bagging*

Description

Set resample parameters for `rtMod` bagging

Usage

```
setup.bag.resample(
  resampler = "strat.sub",
  n.resamples = 10,
  stratify.var = NULL,
  train.p = 0.75,
  strat.n.bins = 4,
  target.length = NULL,
  verbosity = 1
)
```

Arguments

resampler	Character: Type of resampling to perform: "bootstrap", "kfold", "strat.boot", "strat.sub".
n.resamples	Integer: Number of training/testing sets required
stratify.var	Numeric vector (optional): Variable used for stratification.
train.p	Float (0, 1): Fraction of cases to assign to traininig set for `resampler = "strat.sub"`
strat.n.bins	Integer: Number of groups to use for stratification for `resampler = "strat.sub"` / `"strat.boot"`
target.length	Integer: Number of cases for training set for `resampler = "strat.boot"`.
verbosity	Logical: If TRUE, print messages to console

---

setup.color *Set colorGrad parameters*

Description

Set colorGrad parameters

Usage

```
setup.color(
  n = 101,
  colors = NULL,
  space = "rgb",
  lo = "#01256E",
  lomid = NULL,
  mid = "white",
  midhi = NULL,
  hi = "#95001A",
  colorbar = FALSE,
  cb.mar = c(1, 1, 1, 1),
  ...
)
```

Arguments

n	Integer: How many distinct colors you want. If not odd, converted to `n + 1` Defaults to 21
colors	Character: Acts as a shortcut to defining `lo`, `mid`, etc for a number of defaults: "french", "penn", "grnblkred",
space	Character: Which colorspace to use. Option: "rgb", or "Lab". Default = "rgb". Recommendation: If `mid` is "white" or "black" (default), use "rgb", otherwise "Lab"
lo	Color for low end

lomid	Color for low-mid
mid	Color for middle of the range or "mean", which will result in colorOp(c(lo, hi), "mean"). If mid = NA, then only lo and hi are used to create the color gradient.
midhi	Color for middle-high
hi	Color for high end
colorbar	Logical: Create a vertical colorbar
cb.mar	Vector, length 4: Colorbar margins. Default: c(1, 1, 1, 1)
...	Additional arguments

---

setup.cv.resample *setup.cv.resample: resample defaults for cross-validation*

Description

setup.cv.resample: resample defaults for cross-validation

Usage

```
setup.cv.resample(
  resampler = "strat.sub",
  n.resamples = 10,
  stratify.var = NULL,
  train.p = 0.8,
  strat.n.bins = 4,
  target.length = NULL,
  id.strat = NULL,
  verbosity = 1
)
```

Arguments

resampler	Character: Type of resampling to perform: "bootstrap", "kfold", "strat.boot", "strat.sub".
n.resamples	Integer: Number of training/testing sets required
stratify.var	Numeric vector (optional): Variable used for stratification.
train.p	Float (0, 1): Fraction of cases to assign to traininig set for resampler = "strat.sub"
strat.n.bins	Integer: Number of groups to use for stratification for resampler = "strat.sub" / "strat.boot"
target.length	Integer: Number of cases for training set for resampler = "strat.boot".
id.strat	Vector of IDs which may be replicated: resampling should force replicates of each ID to only appear in the training or testing.
verbosity	Logical: If TRUE, print messages to console

setup.decompose	*Set decomposition parameters for train_cv* `.decompose` *argument*

Description

Set decomposition parameters for train_cv `.decompose` argument

Usage

```
setup.decompose(decom = "ICA", k = 2, ...)
```

Arguments

`decom`	Character: Name of decomposer to use.
`k`	Integer: Number of dimensions to project to.
`...`	Additional arguments to be passed to decomposer

setup.earlystop	*Set earlystop parameters*

Description

Set earlystop parameters

Usage

```
setup.earlystop(
  window = 150,
  window_decrease_pct_min = 0.01,
  total_decrease_pct_max = NULL
)
```

Arguments

`window`	Integer: Number of steps to consider
`window_decrease_pct_min`	Float: Stop if improvement is less than this percent over last `window` steps
`total_decrease_pct_max`	Float: Stop if improvement from first to last step exceeds this percent. If defined, overrides `window_decrease_pct_min`

`setup.GBM` *Set s_GBM parameters*

Description

Set s_GBM parameters

Usage

```
setup.GBM(
  interaction.depth = 2,
  shrinkage = 0.001,
  max.trees = 5000,
  min.trees = 100,
  bag.fraction = 0.9,
  n.minobsinnode = 5,
  grid.resample.params = setup.resample("kfold", 5),
  ifw = TRUE,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  ...
)
```

Arguments

`interaction.depth`	[gS] Integer: Interaction depth.
`shrinkage`	[gS] Float: Shrinkage (learning rate).
`max.trees`	Integer: Maximum number of trees to fit
`min.trees`	Integer: Minimum number of trees to fit.
`bag.fraction`	[gS] Float (0, 1): Fraction of cases to use to train each tree. Helps avoid overfitting.
`n.minobsinnode`	[gS] Integer: Minimum number of observation allowed in node.
`grid.resample.params`	List: Output of setup.resample defining grid search parameters.
`ifw`	Logical: If TRUE, apply inverse frequency weighting (for Classification only). Note: If `weights` are provided, `ifw` is not used.
`upsample`	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
`downsample`	Logical: If TRUE, downsample majority class to match size of minority class

`resample.seed`	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
`...`	Additional arguments

`setup.grid.resample` *Set resample parameters for* `gridSearchLearn`

Description

Set resample parameters for `gridSearchLearn`

Usage

```
setup.grid.resample(
  resampler = "kfold",
  n.resamples = 5,
  stratify.var = NULL,
  train.p = 0.75,
  strat.n.bins = 4,
  target.length = NULL,
  verbosity = 1
)
```

Arguments

`resampler`	Character: Type of resampling to perform: "bootstrap", "kfold", "strat.boot", "strat.sub".
`n.resamples`	Integer: Number of training/testing sets required
`stratify.var`	Numeric vector (optional): Variable used for stratification.
`train.p`	Float (0, 1): Fraction of cases to assign to traininig set for `resampler = "strat.sub"`
`strat.n.bins`	Integer: Number of groups to use for stratification for `resampler = "strat.sub" / "strat.boot"`
`target.length`	Integer: Number of cases for training set for `resampler = "strat.boot"`.
`verbosity`	Logical: If TRUE, print messages to console

setup.LightRuleFit *Set s_LightRuleFit parameters*

Description

Sets parameters for the GBM and GLMNET (LASSO) steps of s_LightRuleFit

Usage

```
setup.LightRuleFit(
  n_trees = 200,
  num_leaves = 32L,
  max_depth = 3,
  learning_rate = 0.1,
  subsample = 0.666,
  subsample_freq = 1L,
  lambda_l1 = 0,
  lambda_l2 = 0,
  objective = NULL,
  extra.lgbm.params = NULL,
  lightgbm.ifw = TRUE,
  lightgbm.resample.params = setup.resample("kfold", 5),
  glmnet.ifw = TRUE,
  importance = FALSE,
  alpha = 1,
  lambda = NULL,
  glmnet.resample.params = setup.resample("kfold", 5)
)
```

Arguments

`num_leaves`	Integer: [gS] Maximum tree leaves for base learners.
`max_depth`	Integer: [gS] Maximum tree depth for base learners, <=0 means no limit.
`learning_rate`	Numeric: [gS] Boosting learning rate
`subsample`	Numeric: [gS] Subsample ratio of the training set.
`subsample_freq`	Integer: Subsample every this many iterations
`lambda_l1`	Numeric: [gS] L1 regularization term
`lambda_l2`	Numeric: [gS] L2 regularization term
`objective`	(Default = NULL)
`lightgbm.ifw`	Logical: Passed to s_LightGBM's `ifw` argument
`glmnet.ifw`	Logical: Passed to s_GLMNET's `ifw` argument
`importance`	Logical: If `TRUE`, calculate variable importance
`alpha`	[gS] Float [0, 1]: The elasticnet mixing parameter: `a = 0` is the ridge penalty, `a = 1` is the lasso penalty
`lambda`	[gS] Float vector: Best left to NULL, `cv.glmnet` will compute its own lambda sequence

Author(s)

ED Gennatas

---

setup.LIHAD *Set s_LIHAD parameters*

Description

Set s_LIHAD parameters

Usage

```
setup.LIHAD(
  max.depth = 2,
  learning.rate = 1,
  lincoef.params = setup.lincoef("glmnet"),
  alpha = 0,
  lambda = 0.1,
  minobsinnode = 2,
  minobsinnode.lin = 20,
  ...
)
```

Arguments

max.depth	[gS] Integer: Max depth of additive tree. Default = 3
learning.rate	[gS] Float (0, 1): Learning rate.
lincoef.params	Named List: Output of setup.lincoef
alpha	[gS] Float: `lincoef` alpha Overrides `lincoef.params` alpha
lambda	[gS] Float: `lincoef` lambda. Overrides `lincoef.params` lambda
minobsinnode	[gS] Integer: Minimum N observations needed in node, before considering splitting
minobsinnode.lin	Integer: Minimum N observations needed in node in order to train linear model.
...	Additional arguments

`setup.lincoef` *Set lincoef parameters*

Description

Set lincoef parameters

Usage

```
setup.lincoef(
  method = c("glmnet", "cv.glmnet", "lm.ridge", "allSubsets", "forwardStepwise",
    "backwardStepwise", "glm", "sgd", "solve"),
  alpha = 0,
  lambda = 0.01,
  lambda.seq = NULL,
  cv.glmnet.nfolds = 5,
  which.cv.glmnet.lambda = c("lambda.min", "lambda.1se"),
  nbest = 1,
  nvmax = 8,
  sgd.model = "glm",
  sgd.model.control = list(lambda1 = 0, lambda2 = 0),
  sgd.control = list(method = "ai-sgd")
)
```

Arguments

`method` Character: Method to use:

- "glm": uses `stats::lm.wfit`
- "glmnet": uses `glmnet::glmnet`
- "cv.glmnet": uses `glmnet:cv.glmnet`
- "lm.ridge": uses `MASS::lm.ridge`
- "allsubsets": uses `leaps::regsubsets` with `method` = `"exhaustive"`
- "forwardStepwise": uses `leaps::regsubsets` with `method` = `"forward"`
- "backwardStepwise": uses `leaps::regsubsets` with `method` = `"backward"`
- "sgd": uses `sgd::sgd`
- "solve": uses `base::solve`
- "none": fits no model and returns all zeroes, for programming convenience in special cases

`alpha` Float: `alpha` for method = `glmnet` or `cv.glmnet`.

`lambda` Float: The lambda value for `glmnet`, `cv.glmnet`, `lm.ridge` Note: For `glmnet` and `cv.glmnet`, this is the lambda used for prediction. Training uses `lambda.seq`.

`lambda.seq` Float, vector: lambda sequence for `glmnet` and `cv.glmnet`.

`cv.glmnet.nfolds` Integer: Number of folds for `cv.glmnet`

`which.cv.glmnet.lambda`
: Character: Whitch lambda to pick from cv.glmnet: "lambda.min": Lambda that gives minimum cross-validated error;

`nbest`
: Integer: For `method = "allSubsets"`, number of subsets of each size to record. Default = 1

`nvmax`
: Integer: For `method = "allSubsets"`, maximum number of subsets to examine.

`sgd.model`
: Character: Model to use for `method = "sgd"`.

`sgd.model.control`
: List: `model.control` list to pass to `sgd::sgd`

`sgd.control`
: List: `sgd.control` list to pass to `sgd::sgd` "lambda.1se": Largest lambda such that error is within 1 s.e. of the minimum.

`setup.MARS` *Set s_MARS parameters*

Description

Set s_MARS parameters

Usage

```
setup.MARS(
  hidden = 1,
  activation = NULL,
  learning.rate = 0.8,
  momentum = 0.5,
  learningrate_scale = 1,
  output = NULL,
  numepochs = 100,
  batchsize = NULL,
  hidden_dropout = 0,
  visible_dropout = 0,
  ...
)
```

Arguments

`...`
: Additional parameters to pass to `earth::earth`

`setup.meta.resample` *Set resample parameters for meta model training*

Description

Set resample parameters for meta model training

Usage

```
setup.meta.resample(
  resampler = "strat.sub",
  n.resamples = 4,
  stratify.var = NULL,
  train.p = 0.75,
  strat.n.bins = 4,
  target.length = NULL,
  verbosity = TRUE
)
```

Arguments

`resampler`	Character: Type of resampling to perform: "bootstrap", "kfold", "strat.boot", "strat.sub".
`n.resamples`	Integer: Number of training/testing sets required
`stratify.var`	Numeric vector (optional): Variable used for stratification.
`train.p`	Float (0, 1): Fraction of cases to assign to traininig set for `resampler = "strat.sub"`
`strat.n.bins`	Integer: Number of groups to use for stratification for `resampler = "strat.sub" / "strat.boot"`
`target.length`	Integer: Number of cases for training set for `resampler = "strat.boot"`.
`verbosity`	Logical: If TRUE, print messages to console

`setup.preprocess` *Set preprocess parameters for train_cv* `.preprocess` *argument*

Description

Set preprocess parameters for train_cv `.preprocess` argument

Usage

```
setup.preprocess(
  completeCases = FALSE,
  removeCases.thres = NULL,
  removeFeatures.thres = NULL,
  impute = FALSE,
  impute.type = "missRanger",
  impute.missRanger.params = list(pmm.k = 0, maxiter = 10),
  impute.discrete = get_mode,
  impute.numeric = mean,
  integer2factor = FALSE,
  integer2numeric = FALSE,
  logical2factor = FALSE,
  logical2numeric = FALSE,
  numeric2factor = FALSE,
  numeric2factor.levels = NULL,
  numeric.cut.n = 0,
  numeric.cut.labels = FALSE,
  numeric.quant.n = 0,
  character2factor = FALSE,
  scale = FALSE,
  center = FALSE,
  removeConstants = TRUE,
  oneHot = FALSE,
  exclude = NULL
)
```

Arguments

`completeCases` Logical: If TRUE, only retain complete cases (no missing data). Default = FALSE

`removeCases.thres` Float (0, 1): Remove cases with >= to this fraction of missing features.

`removeFeatures.thres` Float (0, 1): Remove features with missing values in >= to this fraction of cases.

`impute` Logical: If TRUE, impute missing cases. See `impute.discrete` and `impute.numeric` for how

`impute.type` Character: How to impute data: "missRanger" and "missForest" use the packages of the same name to impute by iterative random forest regression. "rfImpute" uses `randomForest::rfImpute` (see its documentation), "meanMode" will use mean and mode by default or any custom function defined in `impute.discrete` and `impute.numeric`. Default = "missRanger" (which is much faster than "missForest"). "missForest" is included for compatibility with older pipelines.

`impute.missRanger.params` Named list with elements "pmm.k" and "maxiter", which are passed to `missRanger::missRanger`. `pmm.k` greater than 0 results in predictive mean matching. Default `pmm.k = 3` maxiter = 10 num.trees = 500. Reduce num.trees for faster imputation especially in large datasets. Set pmm.k = 0 to disable predictive mean matching to missForest::missForest

impute.discrete
: Function that returns single value: How to impute discrete variables for impute.type = "meanMode". Default = get_mode

impute.numeric
: Function that returns single value: How to impute continuous variables for impute.type = "meanMode". Default = mean

integer2factor
: Logical: If TRUE, convert all integers to factors. This includes bit64::integer64 columns

integer2numeric
: Logical: If TRUE, convert all integers to numeric (will only work if integer2factor = FALSE) This includes bit64::integer64 columns

logical2factor
: Logical: If TRUE, convert all logical variables to factors

logical2numeric
: Logical: If TRUE, convert all logical variables to numeric

numeric2factor
: Logical: If TRUE, convert all numeric variables to factors

numeric2factor.levels
: Character vector: Optional - will be passed to levels arg of factor() if numeric2factor = TRUE (For advanced/ specific use cases; need to know unique values of numeric vector(s) and given all numeric vars have same unique values)

numeric.cut.n
: Integer: If > 0, convert all numeric variables to factors by binning using base::cut with breaks equal to this number

numeric.cut.labels
: Logical: The labels argument of base::cut

numeric.quant.n
: Integer: If > 0, convert all numeric variables to factors by binning using base::cut with breaks equal to this number of quantiles produced using stats::quantile

character2factor
: Logical: If TRUE, convert all character variables to factors

scale
: Logical: If TRUE, scale columns of x

center
: Logical: If TRUE, center columns of x. Note that by default it is the same as scale

removeConstants
: Logical: If TRUE, remove constant columns.

oneHot
: Logical: If TRUE, convert all factors using one-hot encoding

exclude
: Integer, vector: Exclude these columns from preprocessing.

`setup.Ranger` *Set s_Ranger parameters*

Description

Set s_Ranger parameters

Usage

```
setup.Ranger(
  n.trees = 1000,
  min.node.size = 1,
  mtry = NULL,
  grid.resample.params = setup.resample("kfold", 5),
  ifw = TRUE,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  ...
)
```

Arguments

`n.trees`	Integer: Number of trees to grow. Default = 1000
`min.node.size`	[gS] Integer: Minimum node size
`mtry`	[gS] Integer: Number of features sampled randomly at each split. Defaults to square root of n of features for classification, and a third of n of features for regression.
`grid.resample.params`	List: Output of setup.resample defining grid search parameters.
`ifw`	Logical: If TRUE, apply inverse frequency weighting (for Classification only). Note: If `weights` are provided, `ifw` is not used.
`upsample`	Logical: If TRUE, upsample training set cases not belonging in majority outcome group
`downsample`	Logical: If TRUE, downsample majority class to match size of minority class
`resample.seed`	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
`...`	Additional arguments to be passed to `ranger::ranger`

`setup.resample` *Set resample settings*

Description

Set resample settings

Usage

```
setup.resample(
  resampler = c("strat.sub", "strat.boot", "kfold", "bootstrap", "loocv"),
  n.resamples = 10,
  stratify.var = NULL,
  train.p = 0.8,
  strat.n.bins = 4,
  target.length = NULL,
  id.strat = NULL,
  seed = NULL
)
```

Arguments

`resampler`	Character: Type of resampling to perform: "bootstrap", "kfold", "strat.boot", "strat.sub".
`n.resamples`	Integer: Number of training/testing sets required
`stratify.var`	Numeric vector (optional): Variable used for stratification.
`train.p`	Float (0, 1): Fraction of cases to assign to traininig set for `resampler = "strat.sub"`
`strat.n.bins`	Integer: Number of groups to use for stratification for `resampler = "strat.sub" / "strat.boot"`
`target.length`	Integer: Number of cases for training set for `resampler = "strat.boot"`.
`id.strat`	Vector of IDs which may be replicated: resampling should force replicates of each ID to only appear in the training or testing.
`seed`	Integer: (Optional) Set seed for random number generator, in order to make output reproducible. See `?base::set.seed`

`sge_submit` *Submit expression to SGE grid*

Description

Submit expression to SGE grid

Usage

```
sge_submit(
  expr,
  obj_names = NULL,
  packages = NULL,
  queue = NULL,
  n_threads = 4,
  sge_out = file.path(getwd(), "./sge_out"),
  sge_error = sge_out,
  sge_env = "#! /usr/bin/env bash",
  sge_opts = "#$ -cwd",
  R_command = NULL,
  system_command = NULL,
  h_rt = "00:25:00",
  mem_free = NULL,
  temp_dir = file.path(getwd(), ".sge_tempdir"),
  verbose = TRUE,
  trace = 1
)
```

Arguments

`expr`	R expression
`obj_names`	Character vector: Names of objects to copy to cluster R session
`packages`	Character vector: Names of packages to load in cluster R session
`queue`	Character: Name of SGE queue to submit to
`n_threads`	Integer: Number of threads to request from scheduler
`sge_out`	Character: Path to directory to write standard out message files
`sge_error`	Character: Path to directory to write error message files
`sge_env`	Character: Shell environment for script to be submitted to SGE
`sge_opts`	Character: SGE options that will be written in shell script. Default = "#$ -cwd"
`R_command`	Character: Optional R command(s) to run at the beginning of the R script
`system_command`	Character: system command to be run by shell script before executing R code. For example a command that export the R executable to use
`h_rt`	Character: Max time to request. Default = "00:25:00", i.e. 25 minutes
`mem_free`	Character: Amount of memory to request from the scheduler
`temp_dir`	Character: Temporary directory that is accessible to all execution nodes. Default = `file.path(getwd(), ".sge_tempdir")` You can use `tempdir()` if all execution nodes have access to the same filesystem as the submit node.
`verbose`	Logical: If TRUE, print messages to console. Default = TRUE
`trace`	Integer: If > 0 print diagnostic messages to console.

Author(s)

E.D. Gennatas

`sigmoid` *Sigmoid function*

Description

Sigmoid function

Usage

```
sigmoid(x)
```

Arguments

x	Vector, float: Input

`size` *Size of matrix or vector*

Description

Return the size of a matrix or vector as (Nrows, Ncolumns) Are you tired of getting NULL when you run dim() on a vector?

Usage

```
size(x)
```

Arguments

x	Vector or matrix input

Value

Integer vector of length 2: c(Nrow, Ncols)

Author(s)

E.D. Gennatas

Examples

```
x <- rnorm(20)
size(x)
# 20  1

x <- matrix(rnorm(100), 20, 5)
size(x)
# 20  5
```

`softmax` *Softmax function*

Description

Softmax function

Usage

```
softmax(x)
```

Arguments

x	Vector, Float: Input

`softplus` *Softplus function*

Description

Softplus function:

$$log(1 + e^x)$$

Usage

```
softplus(x)
```

Arguments

x	Vector, Float: Input

`sortedlines` *lines, but sorted*

Description

lines, but sorted

Usage

```
sortedlines(x, y, col = "red", ...)
```

Arguments

x	Input vector
y	Input vector
col	Line color. Default = "red"
...	Extra params to pass to `lines`

Author(s)

E.D. Gennatas

---

sparsernorm — *Sparse rnorm*

Description

A sparse version of `stats::rnorm` Outputs a vector where a fraction of values are zeros (determined by `sparseness`) and the rest are drawn from a random normal distribution using `stats::rnorm`

Usage

```
sparsernorm(n, sparseness = 0.1, mean = 0, sd = 1)
```

Arguments

n	Integer: Length of output vector
sparseness	Float (0, 1): Fraction of required nonzero elements, i.e. output will have `round(sparseness * n) nonzer` If `sparseness = 0`, a vector of zeros length `n` is returned, if `sparseness = 1`, `rnorm(n, mean, sd)` is returned. Default = 0.1
mean	Float: Target mean of nonzero elements, passed to `stats::rnorm` Default = 0
sd	Float: Target sd of nonzero elements, passed to `stats::rnorm` Default = 1

Author(s)

E.D. Gennatas

sparseVectorSummary *Sparseness and pairwise correlation of vectors*

Description

Get sparseness measure on a matrix of vectors

Usage

```
sparseVectorSummary(vectors, y = NULL)
```

Arguments

vectors	Matrix of column vectors
y	Optional numeric vector.

sparsify *Sparsify a vector*

Description

Keep top x% of values of a vector

Usage

```
sparsify(x, sparseness)
```

Arguments

x	Input vector
sparseness	Percent of values of x to keep. The rest will be set to zero.

Author(s)

E.D. Gennatas

specificity — *Specificity*

Description

The first factor level is considered the positive case.

Usage

```
specificity(true, estimated, harmonize = FALSE, verbosity = 1)
```

Arguments

`true`	True labels
`estimated`	Estimated labels
`harmonize`	Logical: If TRUE, run factor_harmonize first
`verbosity`	Integer: If > 0, print messages to console.

stderror — *Standard Error of the Mean*

Description

Calculate the standard error of the mean, which is equal to the standard deviation divided by the square root of the sample size. NA values are automatically removed

Usage

```
stderror(x)
```

Arguments

`x`	Vector, numeric: Input data

Author(s)

E.D. Gennatas

strat.boot *Stratified Bootstrap Resampling*

Description

Stratified Bootstrap Resampling

Usage

```
strat.boot(
  x,
  n.resamples = 10,
  train.p = 0.75,
  stratify.var = NULL,
  strat.n.bins = 4,
  target.length = NULL,
  seed = NULL,
  verbosity = TRUE
)
```

Arguments

`x`	Input vector
`n.resamples`	Integer: Number of training/testing sets required
`train.p`	Float (0, 1): Fraction of cases to assign to traininig set for `resampler = "strat.sub"`
`stratify.var`	Numeric vector (optional): Variable used for stratification.
`strat.n.bins`	Integer: Number of groups to use for stratification for `resampler = "strat.sub"` / `"strat.boot"`
`target.length`	Integer: Number of cases for training set for `resampler = "strat.boot"`.
`seed`	Integer: (Optional) Set seed for random number generator, in order to make output reproducible. See `?base::set.seed`
`verbosity`	Logical: If TRUE, print messages to console

Author(s)

E.D. Gennatas

strat.sub — *Resample using Stratified Subsamples*

Description

Resample using Stratified Subsamples

Usage

```
strat.sub(
  x,
  n.resamples = 10,
  train.p = 0.75,
  stratify.var = NULL,
  strat.n.bins = 4,
  seed = NULL,
  verbosity = TRUE
)
```

Arguments

`x`	Input vector
`n.resamples`	Integer: Number of training/testing sets required
`train.p`	Float (0, 1): Fraction of cases to assign to traininig set for `resampler = "strat.sub"`
`stratify.var`	Numeric vector (optional): Variable used for stratification.
`strat.n.bins`	Integer: Number of groups to use for stratification for `resampler = "strat.sub"` / `"strat.boot"`
`seed`	Integer: (Optional) Set seed for random number generator, in order to make output reproducible. See `?base::set.seed`
`verbosity`	Logical: If TRUE, print messages to console

Author(s)

E.D. Gennatas

strata2factor — *Convert `survfit` object's strata to a factor*

Description

Convert `survfit` object's strata to a factor

Usage

```
strata2factor(x)
```

Arguments

x	survfit object

Value

factor

Author(s)

E.D. Gennatas

---

summarize — *Summarize numeric variables*

Description

Summarize numeric variables

Usage

```
summarize(
  x,
  varname,
  group_by = NULL,
  type = c("all", "median-range", "mean-sd"),
  na.rm = TRUE
)
```

Arguments

x	data.frame or data.table (will be coerced to data.table)
varname	Character, vector: Variable name(s) to summarize. Must be column names in x of type numeric.
group_by	Character, vector: Variable name(s) of factors to group by. Must be column names in x. Default = NULL
type	Character: "all", "median-range" or "mean-sd". Default = "all", which returns Mean, SD, Median, Range, NA (number of NA values)
na.rm	Logical: Passed to median and mean. Default = TRUE

Value

data.table with summary

Author(s)

E.D. Gennatas

`summary.massGAM` — `massGAM` *object summary*

Description

`massGAM` object summary

Usage

```
## S3 method for class 'massGAM'
summary(object, ...)
```

Arguments

`object`	A `massGAM` object created by massGLAM
`...`	Not used

Author(s)

E.D. Gennatas

`summary.massGLM` — `massGLM` *object summary*

Description

`massGLM` object summary

Usage

```
## S3 method for class 'massGLM'
summary(object, ...)
```

Arguments

`object`	An object created by massGLM
`...`	Not used

Author(s)

E.D. Gennatas

`surv_error` *Survival Analysis Metrics*

Description

Survival Analysis Metrics

Usage

```
surv_error(true, estimated)
```

Arguments

`true`	Vector, numeric: True survival times
`estimated`	Vector, numeric: Estimated survival times

Author(s)

E.D. Gennatas

`svd1` `rtemis-internals` *Project Variables to First Eigenvector*

Description

Convenience function for SVD k = 1

Usage

```
svd1(x, x.test = NULL)
```

Arguments

`x`	Input matrix / data frame
`x.test`	Optional test matrix / data frame

Author(s)

E.D. Gennatas

synth_multimodal *Create "Multimodal" Synthetic Data*

Description

Create "Multimodal" Synthetic Data using squares and arctangents

Usage

```
synth_multimodal(
  n.cases = 10000,
  init.fn = "runifmat",
  init.fn.params = list(min = -10, max = 10),
  n.groups = 4,
  n.feat.per.group = round(seq(10, 300, length.out = n.groups)),
  contrib.p = 0.33,
  linear.p = 0.66,
  square.p = 0.1,
  atan.p = 0.1,
  pair.multiply.p = 0.05,
  pair.square.p = 0.05,
  pair.atan.p = 0.05,
  verbose = TRUE,
  seed = NULL,
  filename = NULL
)
```

Arguments

`n.cases`	Integer: Number of cases to create. Default = 10000
`init.fn`	Character: "runifmat" or "rnormmat". Use the respective functions to generate features as random uniform and random normal variables, respectively. Default = "runifmat"
`init.fn.params`	Named list with arguments "min", "max" for "runifmat" and "mean", "sd" for "rnormmat". Default = `list(min = -10, max = 10)`
`n.groups`	Integer: Number of feature groups / modalities to create. Default = 4
`n.feat.per.group`	Integer, vector, length `n.groups`: Number of features per group to create. Default = `c(50, 100, 200, 300)`
`contrib.p`	Float (0, 1]: Ratio of features contributing to outcome per group. Default = .33, i.e. a third of the features in each group will be used to produce the outcome y
`linear.p`	Float [0, 1]: Ratio of contributing features to be included linearly. Default = .1, i.e. .1 of .33 of features in each group will be included
`square.p`	Float [0, 1]: Ratio of contributing features to be squared. Default = .1, i.e. .1 of .33 of features in each group will be squared

atan.p Float [0, 1]: Ratio of contributing features whose atan will be used. These will be selected from the features that were NOT sampled for squaring. Default = .1, i.e. .1 of .33 of features in each group will be transformed using atan, but given these features were not already picked to be squared (see square.p)

pair.multiply.p Float [0, 1] Ratio of features will be divided into pairs and multiplied. Default = .05

pair.square.p Float [0, 1] Ratio of features which will be divided into pairs, multiplied and squared.

pair.atan.p Float [0, 1] Ratio of features which will be divided into pairs, multiplied and transformed using atan.

verbose Logical: If TRUE, print messages to console.

seed Integer: If set, pass to set.seed for reproducibility

filename Character: Path to file to save output.

Details

There are no checks yet for compatibility among inputs and certain combinations may not work.

Value

List with elements x, y, index.square, index.atan, index.pair.square

Author(s)

E.D. Gennatas

Examples

```
xmm <- synth_multimodal(
  n.cases = 10000,
  init.fn = "runifmat",
  init.fn.params = list(min = -10, max = 10),
  n.groups = 5,
  n.feat.per.group = c(20, 50, 100, 200, 300),
  contrib.p = .33,
  linear.p = .66,
  square.p = .1,
  atan.p = .1,
  pair.multiply.p = .1,
  pair.square.p = .1,
  pair.atan.p = .1,
  seed = 2019
)
```

synth_reg_data *Synthesize Simple Regression Data*

Description

Synthesize Simple Regression Data

Usage

```
synth_reg_data(
  nrow = 500,
  ncol = 50,
  noise.sd.factor = 1,
  resample.params = setup.resample(),
  seed = NULL,
  verbose = FALSE
)
```

Arguments

`nrow`	Integer: Number of rows. Default = 500
`ncol`	Integer: Number of columns. Default = 50
`noise.sd.factor`	Numeric: Add rnorm(nrow, sd = noise.sd.factor * sd(y)). Default = 2
`resample.params`	Output of setup.resample defining training/testing split. The first resulting resample will be used to create `dat.train` and `dat.test` output; all resample output under `resamples`
`seed`	Integer: Seed for random number generator. Default = NULL
`verbose`	Logical: If TRUE, print messages to console. Default = FALSE

Value

List with elements `dat, dat.train, dat.test, resamples, w, seed`

Author(s)

E.D. Gennatas

s_AdaBoost *Adaboost Binary Classifier C*

Description

Train an Adaboost Classifier using ada::ada

Usage

```
s_AdaBoost(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  loss = "exponential",
  type = "discrete",
  iter = 50,
  nu = 0.1,
  bag.frac = 0.5,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  x.name = NULL,
  y.name = NULL,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  trace = 0,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
loss	Character: "exponential" (Default), "logistic"
type	Character: "discrete", "real", "gentle"

iter	Integer: Number of boosting iterations to perform. Default = 50
nu	Float: Shrinkage parameter for boosting. Default = .1
bag.frac	Float (0, 1]: Sampling fraction for out-of-bag samples
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
x.name	Character: Name for feature set
y.name	Character: Name for outcome
print.plot	Logical: if TRUE, produce plot using `mplot3` Takes precedence over `plot.fitted` and `plot.predicted`.
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires `x.test` and `y.test`
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If higher than 0, will print more information to the console.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if `save.mod` is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in `outdir` `save.mod` is TRUE by default if an `outdir` is defined. If set to TRUE, and no `outdir` is defined, outdir defaults to `paste0("./s.", mod.name)`
...	Additional arguments

Details

`ada::ada` does not support case weights

Value

`rtMod` object

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: s_AddTree(), s_BART(), s_BRUTO(), s_BayesGLM(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GAM(), s_GBM(), s_GLM(), s_GLMNET(), s_GLMTree(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_HAL(), s_KNN(), s_LDA(), s_LM(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MARS(), s_MLRF(), s_NBayes(), s_NLA(), s_NLS(), s_NW(), s_PPR(), s_PolyMARS(), s_QDA(), s_QRNN(), s_RF(), s_RFSRC(), s_Ranger(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBoost(), s_XRF()

Other Tree-based methods: s_AddTree(), s_BART(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GBM(), s_GLMTree(), s_H2OGBM(), s_H2ORF(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MLRF(), s_RF(), s_RFSRC(), s_Ranger(), s_XGBoost(), s_XRF()

Other Ensembles: s_GBM(), s_RF(), s_Ranger()

---

s_AddTree — *Additive Tree: Tree-Structured Boosting C*

Description

Train an Additive Tree model

Usage

```
s_AddTree(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  weights = NULL,
  update = c("exponential", "polynomial"),
  min.update = ifelse(update == "polynomial", 0.035, 1000),
  min.hessian = 0.001,
  min.membership = 1,
  steps.past.min.membership = 0,
  gamma = 0.8,
  max.depth = 30,
  learning.rate = 0.1,
  ifw = TRUE,
  ifw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  imetrics = TRUE,
  grid.resample.params = setup.resample("kfold", 5),
```

```
  metric = "Balanced Accuracy",
  maximize = TRUE,
  rpart.params = NULL,
  match.rules = TRUE,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  prune.verbose = FALSE,
  trace = 1,
  grid.verbose = verbose,
  outdir = NULL,
  save.rpart = FALSE,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  n.cores = rtCores
)
```

Arguments

`x`	Numeric vector or matrix / data frame of features i.e. independent variables
`y`	Numeric vector of outcome, i.e. dependent variable
`x.test`	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in `x`
`y.test`	Numeric vector of testing set outcome
`x.name`	Character: Name for feature set
`y.name`	Character: Name for outcome
`weights`	Numeric vector: Weights for cases. For classification, `weights` takes precedence over `ifw`, therefore set `weights = NULL` if using `ifw`. Note: If `weight` are provided, `ifw` is not used. Leave NULL if setting `ifw = TRUE`.
`update`	Character: "exponential" or "polynomial". Type of weight update. Default = "exponential"
`min.update`	Float: Minimum update for gradient step
`min.hessian`	[gS] Float: Minimum second derivative to continue splitting. Default = .001
`min.membership`	Integer: Minimum number of cases in a node. Default = 1
`steps.past.min.membership`	Integer: N steps to make past `min.membership` - For testing. Default = 0
`gamma`	[gS] Float: acceleration factor = lambda/(1 + lambda). Default = .8
`max.depth`	[gS] Integer: maximum depth of the tree. Default = 30
`learning.rate`	[gS] learning rate for the Newton Raphson step that updates the function values of the node
`ifw`	Logical: If TRUE, apply inverse frequency weighting (for Classification only). Note: If `weights` are provided, `ifw` is not used.

`ifw.type`	Integer 0, 1, 2 1: class.weights as in 0, divided by min(class.weights) 2: class.weights as in 0, divided by max(class.weights)
`upsample`	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
`downsample`	Logical: If TRUE, downsample majority class to match size of minority class
`resample.seed`	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
`imetrics`	Logical: If TRUE, save interpretability metrics, i.e. N total nodes in tree and depth, in output. Default = TRUE
`grid.resample.params`	List: Output of setup.resample defining grid search parameters.
`metric`	Character: Metric to minimize, or maximize if `maximize = TRUE` during grid search. Default = NULL, which results in "Balanced Accuracy" for Classification, "MSE" for Regression, and "Coherence" for Survival Analysis.
`maximize`	Logical: If TRUE, `metric` will be maximized if grid search is run.
`rpart.params`	List: `rpart` parameters, passed to `rpart::rpart("parms")`
`match.rules`	Logical: If TRUE, match cases to rules to get statistics per node, i.e. what percent of cases match each rule. If available, these are used by dplot3_addtree when plotting. Default = TRUE
`print.plot`	Logical: if TRUE, produce plot using `mplot3` Takes precedence over `plot.fitted` and `plot.predicted`.
`plot.fitted`	Logical: if TRUE, plot True (y) vs Fitted
`plot.predicted`	Logical: if TRUE, plot True (y.test) vs Predicted. Requires `x.test` and `y.test`
`plot.theme`	Character: "zero", "dark", "box", "darkbox"
`question`	Character: the question you are attempting to answer with this model, in plain language.
`verbose`	Logical: If TRUE, print summary to screen.
`prune.verbose`	Logical: If TRUE, prune tree.
`trace`	Integer: 0, 1, 2. The higher the number, the more verbose the output.
`grid.verbose`	Logical: Passed to `gridSearchLearn`
`outdir`	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if `save.mod` is TRUE
`save.rpart`	Logical: passed to `addtree`
`save.mod`	Logical: If TRUE, save all output to an RDS file in `outdir` `save.mod` is TRUE by default if an `outdir` is defined. If set to TRUE, and no `outdir` is defined, outdir defaults to `paste0("./s.", mod.name)`
`n.cores`	Integer: Number of cores to use.

Details

This function is for binary classification. The outcome must be a factor with two levels, the first level is the 'positive' class. Ensure there are no missing values in the data and that variables are either numeric (including integers) or factors. Use preprocess as needed to impute and convert characters to factors.

Factor levels should not contain the "/" character (it is used to separate conditions in the addtree object)

[gS] Indicates that more than one value can be supplied, which will result in grid search using internal resampling lambda = gamma/(1 - gamma)

Value

Object of class `rtMod`

Author(s)

E.D. Gennatas

References

Jose Marcio Luna, Efstathios D Gennatas, Lyle H Ungar, Eric Eaton, Eric S Diffenderfer, Shane T Jensen, Charles B Simone, Jerome H Friedman, Timothy D Solberg, Gilmer Valdes Building more accurate decision trees with the additive tree Proc Natl Acad Sci U S A. 2019 Oct 1;116(40):19887-19893. doi: 10.1073/pnas.1816748116

See Also

Other Supervised Learning: `s_AdaBoost()`, `s_BART()`, `s_BRUTO()`, `s_BayesGLM()`, `s_C50()`, `s_CART()`, `s_CTree()`, `s_EVTree()`, `s_GAM()`, `s_GBM()`, `s_GLM()`, `s_GLMNET()`, `s_GLMTree()`, `s_GLS()`, `s_H2ODL()`, `s_H2OGBM()`, `s_H2ORF()`, `s_HAL()`, `s_KNN()`, `s_LDA()`, `s_LM()`, `s_LMTree()`, `s_LightCART()`, `s_LightGBM()`, `s_MARS()`, `s_MLRF()`, `s_NBayes()`, `s_NLA()`, `s_NLS()`, `s_NW()`, `s_PPR()`, `s_PolyMARS()`, `s_QDA()`, `s_QRNN()`, `s_RF()`, `s_RFSRC()`, `s_Ranger()`, `s_SDA()`, `s_SGD()`, `s_SPLS()`, `s_SVM()`, `s_TFN()`, `s_XGBoost()`, `s_XRF()`

Other Tree-based methods: `s_AdaBoost()`, `s_BART()`, `s_C50()`, `s_CART()`, `s_CTree()`, `s_EVTree()`, `s_GBM()`, `s_GLMTree()`, `s_H2OGBM()`, `s_H2ORF()`, `s_LMTree()`, `s_LightCART()`, `s_LightGBM()`, `s_MLRF()`, `s_RF()`, `s_RFSRC()`, `s_Ranger()`, `s_XGBoost()`, `s_XRF()`

Other Interpretable models: `s_C50()`, `s_CART()`, `s_GLM()`, `s_GLMNET()`, `s_GLMTree()`, `s_LMTree()`

`s_BART` *Bayesian Additive Regression Trees (C, R)*

Description

Trains a Bayesian Additive Regression Tree (BART) model using package `bartMachine`

Usage

```
s_BART(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  n.trees = c(100, 200),
  k_cvs = c(2, 3),
  nu_q_cvs = list(c(3, 0.9), c(10, 0.75)),
  k_folds = 5,
  n.burnin = 250,
  n.iter = 1000,
  n.cores = rtCores,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  trace = 0,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  java.mem.size = 12,
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class

resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted.
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If higher than 0, will print more information to the console.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: if TRUE, sets bartMachine's serialize to TRUE and saves model to outdir
...	Additional arguments to be passed to bartMachine::bartMachine

Details

Be warned this can take a very long time to train. If you are having trouble with rJava in Rstudio on macOS, see: https://support.rstudio.com/hc/en-us/community/posts/203663956/comments/249073727 bartMachine does not support case weights

Value

Object of class **rtemis**

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: s_AdaBoost(), s_AddTree(), s_BRUTO(), s_BayesGLM(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GAM(), s_GBM(), s_GLM(), s_GLMNET(), s_GLMTree(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_HAL(), s_KNN(), s_LDA(), s_LM(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MARS(), s_MLRF(), s_NBayes(), s_NLA(), s_NLS(), s_NW(), s_PPR(), s_PolyMARS(), s_QDA(), s_QRNN(), s_RF(), s_RFSRC(), s_Ranger(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBoost(), s_XRF()

Other Tree-based methods: s_AdaBoost(), s_AddTree(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GBM(), s_GLMTree(), s_H2OGBM(), s_H2ORF(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MLRF(), s_RF(), s_RFSRC(), s_Ranger(), s_XGBoost(), s_XRF()

s_BayesGLM *Bayesian GLM*

Description

Train a bayesian GLM using arm::bayesglm

Usage

```
s_BayesGLM(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  family = NULL,
  prior.mean = 0,
  prior.scale = NULL,
  prior.df = 1,
  prior.mean.for.intercept = 0,
  prior.scale.for.intercept = NULL,
  prior.df.for.intercept = 1,
  min.prior.scale = 1e-12,
  scaled = TRUE,
  keep.order = TRUE,
  drop.baseline = TRUE,
  maxit = 100,
  x.name = NULL,
  y.name = NULL,
  weights = NULL,
  ifw = TRUE,
  ifw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  metric = NULL,
  maximize = NULL,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  grid.verbose = verbose,
  verbose = TRUE,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

`x`	Numeric vector or matrix / data frame of features i.e. independent variables
`y`	Numeric vector of outcome, i.e. dependent variable
`x.test`	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in `x`
`y.test`	Numeric vector of testing set outcome
`family`	Character or function for the error distribution and link function to be used. See `arm::bayesglm` for details.
`prior.mean`	Numeric, vector: Prior mean for the coefficients. If scalar, it will be replicated to length N features.
`prior.scale`	Numeric, vector: Prior scale for the coefficients. Default = NULL, which results in 2.5 for logit, 2.5*1.6 for probit. If scalar, it will be replicated to length N features.
`prior.df`	Numeric: Prior degrees of freedom for the coefficients. Set to 1 for t distribution; set to Inf for normal prior distribution. If scalar, it will be replicated to length N features.
`prior.mean.for.intercept`	Numeric: Prior mean for the intercept.
`prior.scale.for.intercept`	Numeric: Default = NULL, which results in 10 for a logit model, and 10*1.6 for probit model.
`prior.df.for.intercept`	Numeric: Prior df for the intercept.
`min.prior.scale`	Numeric: Minimum prior scale for the coefficients.
`scaled`	Logical: If TRUE, the scale for the prior distributions are: For feature with single value, use `prior.scale`, for predictor with two values, use `prior.scale/range(x)`, for more than two values, use `prior.scale/(2*sd(x))`. If response is gaussian, `prior.scale` is multiplied by `2 * sd(y)`. Default = TRUE
`keep.order`	Logical: If TRUE, the feature positions are maintained, otherwise they are reordered: main effects, interactions, second-order, third-order, etc.
`drop.baseline`	Logical: If TRUE, drop the base level of factor features.
`maxit`	Integer: Maximum number of iterations
`x.name`	Character: Name for feature set
`y.name`	Character: Name for outcome
`weights`	Numeric vector: Weights for cases. For classification, `weights` takes precedence over `ifw`, therefore set `weights = NULL` if using `ifw`. Note: If `weight` are provided, `ifw` is not used. Leave NULL if setting `ifw = TRUE`.
`ifw`	Logical: If TRUE, apply inverse frequency weighting (for Classification only). Note: If `weights` are provided, `ifw` is not used.
`ifw.type`	Integer 0, 1, 2 1: class.weights as in 0, divided by min(class.weights) 2: class.weights as in 0, divided by max(class.weights)

upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
metric	Character: Metric to minimize, or maximize if maximize = TRUE during grid search. Default = NULL, which results in "Balanced Accuracy" for Classification, "MSE" for Regression, and "Coherence" for Survival Analysis.
maximize	Logical: If TRUE, metric will be maximized if grid search is run.
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted.
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
grid.verbose	Logical: Passed to gridSearchLearn
verbose	Logical: If TRUE, print summary to screen.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional parameters to pass to arm::bayesglm

Author(s)

E.D. Gennatas

See Also

Other Supervised Learning: s_AdaBoost(), s_AddTree(), s_BART(), s_BRUTO(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GAM(), s_GBM(), s_GLM(), s_GLMNET(), s_GLMTree(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_HAL(), s_KNN(), s_LDA(), s_LM(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MARS(), s_MLRF(), s_NBayes(), s_NLA(), s_NLS(), s_NW(), s_PPR(), s_PolyMARS(), s_QDA(), s_QRNN(), s_RF(), s_RFSRC(), s_Ranger(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBoost(), s_XRF()

s_BRUTO *Projection Pursuit Regression (BRUTO) [R]*

Description

Trains a BRUTO model and validates it

Usage

```
s_BRUTO(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  grid.resample.params = setup.grid.resample(),
  weights = NULL,
  weights.col = NULL,
  dfmax = 6,
  cost = 2,
  maxit.select = 20,
  maxit.backfit = 20,
  thresh = 1e-04,
  start.linear = TRUE,
  n.cores = rtCores,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  trace = 0,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

`x`	Numeric vector or matrix / data frame of features i.e. independent variables
`y`	Numeric vector of outcome, i.e. dependent variable
`x.test`	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in `x`
`y.test`	Numeric vector of testing set outcome
`x.name`	Character: Name for feature set

y.name	Character: Name for outcome
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ifw, therefore set weights = NULL if using ifw. Note: If weight are provided, ifw is not used. Leave NULL if setting ifw = TRUE.
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted.
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If higher than 0, will print more information to the console.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments to be passed to mda::bruto

Value

Object of class **rtemis**

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: s_AdaBoost(), s_AddTree(), s_BART(), s_BayesGLM(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GAM(), s_GBM(), s_GLM(), s_GLMNET(), s_GLMTree(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_HAL(), s_KNN(), s_LDA(), s_LM(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MARS(), s_MLRF(), s_NBayes(), s_NLA(), s_NLS(), s_NW(), s_PPR(), s_PolyMARS(), s_QDA(), s_QRNN(), s_RF(), s_RFSRC(), s_Ranger(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBoost(), s_XRF()

s_C50 — *C5.0 Decision Trees and Rule-Based Models C*

Description

Train a C5.0 decision tree using `C50::C5.0`

Usage

```
s_C50(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  trials = 10,
  rules = FALSE,
  weights = NULL,
  ifw = TRUE,
  ifw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  control = C50::C5.0Control(),
  costs = NULL,
  x.name = NULL,
  y.name = NULL,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  trace = 0,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

`x`	Numeric vector or matrix / data frame of features i.e. independent variables
`y`	Numeric vector of outcome, i.e. dependent variable
`x.test`	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in `x`
`y.test`	Numeric vector of testing set outcome
`trials`	Integer [1, 100]: Number of boosting iterations

rules	Logical: If TRUE, decompose the tree to a rule-based model
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ifw, therefore set weights = NULL if using ifw. Note: If weight are provided, ifw is not used. Leave NULL if setting ifw = TRUE.
ifw	Logical: If TRUE, apply inverse frequency weighting (for Classification only). Note: If weights are provided, ifw is not used.
ifw.type	Integer 0, 1, 2 1: class.weights as in 0, divided by min(class.weights) 2: class.weights as in 0, divided by max(class.weights)
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
control	List: output of `C50::C5.0Control()`
costs	Matrix: Cost matrix. See `C50::C5.0`
x.name	Character: Name for feature set
y.name	Character: Name for outcome
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted.
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If higher than 0, will print more information to the console.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to `paste0("./s.", mod.name)`
...	Additional arguments

Value

rtMod object

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: s_AdaBoost(), s_AddTree(), s_BART(), s_BRUTO(), s_BayesGLM(), s_CART(), s_CTree(), s_EVTree(), s_GAM(), s_GBM(), s_GLM(), s_GLMNET(), s_GLMTree(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_HAL(), s_KNN(), s_LDA(), s_LM(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MARS(), s_MLRF(), s_NBayes(), s_NLA(), s_NLS(), s_NW(), s_PPR(), s_PolyMARS(), s_QDA(), s_QRNN(), s_RF(), s_RFSRC(), s_Ranger(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBoost(), s_XRF()

Other Tree-based methods: s_AdaBoost(), s_AddTree(), s_BART(), s_CART(), s_CTree(), s_EVTree(), s_GBM(), s_GLMTree(), s_H2OGBM(), s_H2ORF(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MLRF(), s_RF(), s_RFSRC(), s_Ranger(), s_XGBoost(), s_XRF()

Other Interpretable models: s_AddTree(), s_CART(), s_GLM(), s_GLMNET(), s_GLMTree(), s_LMTree()

---

s_CART *Classification and Regression Trees [C, R, S]*

Description

Train a CART for regression or classification using rpart

Usage

```
s_CART(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  weights = NULL,
  ifw = TRUE,
  ifw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  method = "auto",
  parms = NULL,
  minsplit = 2,
  minbucket = round(minsplit/3),
  cp = 0.01,
  maxdepth = 20,
  maxcompete = 0,
  maxsurrogate = 0,
  usesurrogate = 2,
  surrogatestyle = 0,
```

```
  xval = 0,
  cost = NULL,
  model = TRUE,
  prune.cp = NULL,
  use.prune.rpart.rt = TRUE,
  return.unpruned = FALSE,
  grid.resample.params = setup.resample("kfold", 5),
  gridsearch.type = c("exhaustive", "randomized"),
  gridsearch.randomized.p = 0.1,
  save.gridrun = FALSE,
  metric = NULL,
  maximize = NULL,
  n.cores = rtCores,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  grid.verbose = verbose,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE)
)
```

Arguments

`x`	Numeric vector or matrix / data frame of features i.e. independent variables
`y`	Numeric vector of outcome, i.e. dependent variable
`x.test`	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in `x`
`y.test`	Numeric vector of testing set outcome
`x.name`	Character: Name for feature set
`y.name`	Character: Name for outcome
`weights`	Numeric vector: Weights for cases. For classification, `weights` takes precedence over `ifw`, therefore set `weights = NULL` if using `ifw`. Note: If `weight` are provided, `ifw` is not used. Leave NULL if setting `ifw = TRUE`.
`ifw`	Logical: If TRUE, apply inverse frequency weighting (for Classification only). Note: If `weights` are provided, `ifw` is not used.
`ifw.type`	Integer 0, 1, 2 1: class.weights as in 0, divided by min(class.weights) 2: class.weights as in 0, divided by max(class.weights)
`upsample`	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
`downsample`	Logical: If TRUE, downsample majority class to match size of minority class

resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
method	Character: "auto", "anova", "poisson", "class" or "exp".
parms	List of additional parameters for the splitting function. See rpart::rpart("parms")
minsplit	[gS] Integer: Minimum number of cases that must belong in a node before considering a split.
minbucket	[gS] Integer: Minimum number of cases allowed in a child node.
cp	[gS] Float: Complexity threshold for allowing a split.
maxdepth	[gS] Integer: Maximum depth of tree.
maxcompete	Integer: The number of competitor splits saved in the output
maxsurrogate	Integer: The number of surrogate splits retained in the output (See rpart::rpart.control).
usesurrogate	See rpart::rpart.control
surrogatestyle	See rpart::rpart.control
xval	Integer: Number of cross-validations
cost	Vector, Float (> 0): One for each variable in the model. See rpart::rpart("cost")
model	Logical: If TRUE, keep a copy of the model.
prune.cp	[gS] Numeric: Complexity for cost-complexity pruning after tree is built
use.prune.rpart.rt	(Testing only, do not change)
return.unpruned	Logical: If TRUE and prune.cp is set, return unpruned tree under extra in rtMod.
grid.resample.params	List: Output of setup.resample defining grid search parameters.
gridsearch.type	Character: Type of grid search to perform: "exhaustive" or "randomized".
gridsearch.randomized.p	Float (0, 1): If gridsearch.type = "randomized", randomly test this proportion of combinations.
save.gridrun	Logical: If TRUE, save grid search models.
metric	Character: Metric to minimize, or maximize if maximize = TRUE during grid search. Default = NULL, which results in "Balanced Accuracy" for Classification, "MSE" for Regression, and "Coherence" for Survival Analysis.
maximize	Logical: If TRUE, metric will be maximized if grid search is run.
n.cores	Integer: Number of cores to use.
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted.
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"

question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
grid.verbose	Logical: Passed to gridSearchLearn
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)

Details

[gS] indicates grid search will be performed automatically if more than one value is passed

Value

Object of class rtMod

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: s_AdaBoost(), s_AddTree(), s_BART(), s_BRUTO(), s_BayesGLM(), s_C50(), s_CTree(), s_EVTree(), s_GAM(), s_GBM(), s_GLM(), s_GLMNET(), s_GLMTree(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_HAL(), s_KNN(), s_LDA(), s_LM(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MARS(), s_MLRF(), s_NBayes(), s_NLA(), s_NLS(), s_NW(), s_PPR(), s_PolyMARS(), s_QDA(), s_QRNN(), s_RF(), s_RFSRC(), s_Ranger(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBoost(), s_XRF()

Other Tree-based methods: s_AdaBoost(), s_AddTree(), s_BART(), s_C50(), s_CTree(), s_EVTree(), s_GBM(), s_GLMTree(), s_H2OGBM(), s_H2ORF(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MLRF(), s_RF(), s_RFSRC(), s_Ranger(), s_XGBoost(), s_XRF()

Other Interpretable models: s_AddTree(), s_C50(), s_GLM(), s_GLMNET(), s_GLMTree(), s_LMTree()

---

s_CTree *Conditional Inference Trees [C, R, S]*

Description

Train a conditional inference tree using partykit::ctree

Usage

```
s_CTree(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  weights = NULL,
  control = partykit::ctree_control(),
  ifw = TRUE,
  ifw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  x.name = NULL,
  y.name = NULL,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ifw, therefore set weights = NULL if using ifw. Note: If weight are provided, ifw is not used. Leave NULL if setting ifw = TRUE.
control	List of parameters for the CTree algorithms. Set using partykit::ctree_control
ifw	Logical: If TRUE, apply inverse frequency weighting (for Classification only). Note: If weights are provided, ifw is not used.
ifw.type	Integer 0, 1, 2 1: class.weights as in 0, divided by min(class.weights) 2: class.weights as in 0, divided by max(class.weights)
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class

resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
x.name	Character: Name for feature set
y.name	Character: Name for outcome
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted.
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments

Value

rtMod object

Author(s)

E.D. Gennatas

See Also

train_cv

Other Supervised Learning: s_AdaBoost(), s_AddTree(), s_BART(), s_BRUTO(), s_BayesGLM(), s_C50(), s_CART(), s_EVTree(), s_GAM(), s_GBM(), s_GLM(), s_GLMNET(), s_GLMTree(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_HAL(), s_KNN(), s_LDA(), s_LM(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MARS(), s_MLRF(), s_NBayes(), s_NLA(), s_NLS(), s_NW(), s_PPR(), s_PolyMARS(), s_QDA(), s_QRNN(), s_RF(), s_RFSRC(), s_Ranger(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBoost(), s_XRF()

Other Tree-based methods: s_AdaBoost(), s_AddTree(), s_BART(), s_C50(), s_CART(), s_EVTree(), s_GBM(), s_GLMTree(), s_H2OGBM(), s_H2ORF(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MLRF(), s_RF(), s_RFSRC(), s_Ranger(), s_XGBoost(), s_XRF()

s_EVTree *Evolutionary Learning of Globally Optimal Trees (C, R)*

Description

Train a EVTree for regression or classification using evtree

Usage

```
s_EVTree(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  weights = NULL,
  ifw = TRUE,
  ifw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  control = evtree::evtree.control(),
  na.action = na.exclude,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome

weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ifw, therefore set weights = NULL if using ifw. Note: If weight are provided, ifw is not used. Leave NULL if setting ifw = TRUE.
ifw	Logical: If TRUE, apply inverse frequency weighting (for Classification only). Note: If weights are provided, ifw is not used.
ifw.type	Integer 0, 1, 2 1: class.weights as in 0, divided by min(class.weights) 2: class.weights as in 0, divided by max(class.weights)
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
control	Passed to evtree::evtree
na.action	How to handle missing values. See ?na.fail
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted.
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments to be passed to evtree::evtree

Value

Object of class rtMod

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: s_AdaBoost(), s_AddTree(), s_BART(), s_BRUTO(), s_BayesGLM(), s_C50(), s_CART(), s_CTree(), s_GAM(), s_GBM(), s_GLM(), s_GLMNET(), s_GLMTree(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_HAL(), s_KNN(), s_LDA(), s_LM(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MARS(), s_MLRF(), s_NBayes(), s_NLA(), s_NLS(), s_NW(), s_PPR(), s_PolyMARS(), s_QDA(), s_QRNN(), s_RF(), s_RFSRC(), s_Ranger(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBoost(), s_XRF()

Other Tree-based methods: s_AdaBoost(), s_AddTree(), s_BART(), s_C50(), s_CART(), s_CTree(), s_GBM(), s_GLMTree(), s_H2OGBM(), s_H2ORF(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MLRF(), s_RF(), s_RFSRC(), s_Ranger(), s_XGBoost(), s_XRF()

---

s_GAM *Generalized Additive Model (GAM) (C, R)*

Description

Trains a GAM using mgcv::gam and validates it. Input will be used to create a formula of the form:

$$y = s(x_1, k) + s(x_2, k) + ... + s(x_n, k)$$

Usage

```
s_GAM(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  k = 6,
  family = NULL,
  weights = NULL,
  ifw = TRUE,
  ifw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  method = "REML",
  select = FALSE,
  removeMissingLevels = TRUE,
  spline.index = NULL,
  verbose = TRUE,
  trace = 0,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  na.action = na.exclude,
  question = NULL,
  n.cores = rtCores,
```

```
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
k	Integer. Number of bases for smoothing spline
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ifw, therefore set weights = NULL if using ifw. Note: If weight are provided, ifw is not used. Leave NULL if setting ifw = TRUE.
ifw	Logical: If TRUE, apply inverse frequency weighting (for Classification only). Note: If weights are provided, ifw is not used.
ifw.type	Integer 0, 1, 2 1: class.weights as in 0, divided by min(class.weights) 2: class.weights as in 0, divided by max(class.weights)
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
method	Character: "auto", "anova", "poisson", "class" or "exp".
select	Logical: Passed to mgcv::gam's select argument to allow for each term to be penalized to zero.
verbose	Logical: If TRUE, print summary to screen.
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted.
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
n.cores	Integer: Number of cores to use.

outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if `save.mod` is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in `outdir` `save.mod` is TRUE by default if an `outdir` is defined. If set to TRUE, and no `outdir` is defined, outdir defaults to `paste0("./s.", mod.name)`
...	Additional arguments to be passed to `mgcv::gam`

Value

rtMod

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: `s_AdaBoost()`, `s_AddTree()`, `s_BART()`, `s_BRUTO()`, `s_BayesGLM()`, `s_C50()`, `s_CART()`, `s_CTree()`, `s_EVTree()`, `s_GBM()`, `s_GLM()`, `s_GLMNET()`, `s_GLMTree()`, `s_GLS()`, `s_H2ODL()`, `s_H2OGBM()`, `s_H2ORF()`, `s_HAL()`, `s_KNN()`, `s_LDA()`, `s_LM()`, `s_LMTree()`, `s_LightCART()`, `s_LightGBM()`, `s_MARS()`, `s_MLRF()`, `s_NBayes()`, `s_NLA()`, `s_NLS()`, `s_NW()`, `s_PPR()`, `s_PolyMARS()`, `s_QDA()`, `s_QRNN()`, `s_RF()`, `s_RFSRC()`, `s_Ranger()`, `s_SDA()`, `s_SGD()`, `s_SPLS()`, `s_SVM()`, `s_TFN()`, `s_XGBoost()`, `s_XRF()`

---

s_GBM *Gradient Boosting Machine [C, R, S]*

Description

Train a GBM model using `gbm::gbm.fit`

Usage

```
s_GBM(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  weights = NULL,
  ifw = TRUE,
  ifw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  distribution = NULL,
  interaction.depth = 2,
```

```
  shrinkage = 0.01,
  bag.fraction = 0.9,
  n.minobsinnode = 5,
  n.trees = 2000,
  max.trees = 5000,
  force.n.trees = NULL,
  gbm.select.smooth = FALSE,
  n.new.trees = 500,
  min.trees = 50,
  failsafe.trees = 500,
  imetrics = FALSE,
  .gs = FALSE,
  grid.resample.params = setup.resample("kfold", 5),
  gridsearch.type = "exhaustive",
  metric = NULL,
  maximize = NULL,
  plot.tune.error = FALSE,
  n.cores = rtCores,
  relInf = TRUE,
  varImp = FALSE,
  offset = NULL,
  var.monotone = NULL,
  keep.data = TRUE,
  var.names = NULL,
  response.name = "y",
  checkmods = FALSE,
  group = NULL,
  plot.perf = FALSE,
  plot.res = ifelse(!is.null(outdir), TRUE, FALSE),
  plot.fitted = NULL,
  plot.predicted = NULL,
  print.plot = FALSE,
  plot.theme = rtTheme,
  x.name = NULL,
  y.name = NULL,
  question = NULL,
  verbose = TRUE,
  trace = 0,
  grid.verbose = verbose,
  gbm.fit.verbose = FALSE,
  outdir = NULL,
  save.gridrun = FALSE,
  save.res = FALSE,
  save.res.mod = FALSE,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE)
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ifw, therefore set weights = NULL if using ifw. Note: If weight are provided, ifw is not used. Leave NULL if setting ifw = TRUE.
ifw	Logical: If TRUE, apply inverse frequency weighting (for Classification only). Note: If weights are provided, ifw is not used.
ifw.type	Integer 0, 1, 2 1: class.weights as in 0, divided by min(class.weights) 2: class.weights as in 0, divided by max(class.weights)
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
distribution	Character: Distribution of the response variable. See gbm::gbm
interaction.depth	[gS] Integer: Interaction depth.
shrinkage	[gS] Float: Shrinkage (learning rate).
bag.fraction	[gS] Float (0, 1): Fraction of cases to use to train each tree. Helps avoid overfitting.
n.minobsinnode	[gS] Integer: Minimum number of observation allowed in node.
n.trees	Integer: Initial number of trees to fit
max.trees	Integer: Maximum number of trees to fit
force.n.trees	Integer: If specified, use this number of trees instead of tuning number of trees
gbm.select.smooth	Logical: If TRUE, smooth the validation error curve.
n.new.trees	Integer: Number of new trees to train if stopping criteria have not been met.
min.trees	Integer: Minimum number of trees to fit.
failsafe.trees	Integer: If tuning fails to find n.trees, use this number instead.
imetrics	Logical: If TRUE, save extra$imetrics with n.trees, depth, and n.nodes.
.gs	Internal use only
grid.resample.params	List: Output of setup.resample defining grid search parameters.
gridsearch.type	Character: Type of grid search to perform: "exhaustive" or "randomized".

metric	Character: Metric to minimize, or maximize if maximize = TRUE during grid search. Default = NULL, which results in "Balanced Accuracy" for Classification, "MSE" for Regression, and "Coherence" for Survival Analysis.
maximize	Logical: If TRUE, metric will be maximized if grid search is run.
plot.tune.error	Logical: If TRUE, plot the tuning error curve.
n.cores	Integer: Number of cores to use.
relInf	Logical: If TRUE (Default), estimate variables' relative influence.
varImp	Logical: If TRUE, estimate variable importance by permutation (as in random forests; noted as experimental in gbm). Takes longer than (default) relative influence. The two measures are highly correlated.
offset	Numeric vector of offset values, passed to gbm::gbm.fit
var.monotone	Integer vector with values 0, 1, -1 and length = N features. Used to define monotonicity constraints. 0: no constraint, 1: increasing, -1: decreasing.
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted.
plot.theme	Character: "zero", "dark", "box", "darkbox"
x.name	Character: Name for feature set
y.name	Character: Name for outcome
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
grid.verbose	Logical: Passed to gridSearchLearn
outdir	Character: If defined, save log, 'plot.all' plots (see above) and RDS file of complete output
save.gridrun	Logical: If TRUE, save grid search models.
save.res.mod	Logical: If TRUE, save gbm model for each grid run. For diagnostic purposes only: Object size adds up quickly
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)

Details

Early stopping is implemented by fitting n.trees initially, checking the optionally smoothed validation error curve, and adding n.new.trees if needed, until error does not reduce or max.trees is reached. [gS] in the argument description indicates that a vector of values can be passed, in which case grid search will be performed automatically using the resampling scheme defined by grid.resample.params.

This function includes a workaround for when gbm.fit fails. If an error is detected, gbm.fit is rerun until successful and the procedure continues normally

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: s_AdaBoost(), s_AddTree(), s_BART(), s_BRUTO(), s_BayesGLM(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GAM(), s_GLM(), s_GLMNET(), s_GLMTree(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_HAL(), s_KNN(), s_LDA(), s_LM(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MARS(), s_MLRF(), s_NBayes(), s_NLA(), s_NLS(), s_NW(), s_PPR(), s_PolyMARS(), s_QDA(), s_QRNN(), s_RF(), s_RFSRC(), s_Ranger(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBoost(), s_XRF()

Other Tree-based methods: s_AdaBoost(), s_AddTree(), s_BART(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GLMTree(), s_H2OGBM(), s_H2ORF(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MLRF(), s_RF(), s_RFSRC(), s_Ranger(), s_XGBoost(), s_XRF()

Other Ensembles: s_AdaBoost(), s_RF(), s_Ranger()

---

s_GLM — *Generalized Linear Model (C, R)*

Description

Train a Generalized Linear Model for Regression or Classification (i.e. Logistic Regression) using `stats::glm`. If outcome `y` has more than two classes, Multinomial Logistic Regression is performed using `nnet::multinom`

Usage

```
s_GLM(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  family = NULL,
  interactions = NULL,
  class.method = NULL,
  weights = NULL,
  ifw = TRUE,
  ifw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  intercept = TRUE,
  polynomial = FALSE,
```

```
  poly.d = 3,
  poly.raw = FALSE,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  na.action = na.exclude,
  removeMissingLevels = TRUE,
  question = NULL,
  verbose = TRUE,
  trace = 0,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

`x`	Numeric vector or matrix / data frame of features i.e. independent variables
`y`	Numeric vector of outcome, i.e. dependent variable
`x.test`	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in `x`
`y.test`	Numeric vector of testing set outcome
`x.name`	Character: Name for feature set
`y.name`	Character: Name for outcome
`family`	Error distribution and link function. See `stats::family`
`interactions`	List of character pairs denoting column names in `x` that should be entered as interaction terms in the GLM formula
`class.method`	Character: Define "logistic" or "multinom" for classification. The only purpose of this is so you can try `nnet::multinom` instead of glm for binary classification
`weights`	Numeric vector: Weights for cases. For classification, `weights` takes precedence over `ifw`, therefore set `weights = NULL` if using `ifw`. Note: If `weight` are provided, `ifw` is not used. Leave NULL if setting `ifw = TRUE`.
`ifw`	Logical: If TRUE, apply inverse frequency weighting (for Classification only). Note: If `weights` are provided, `ifw` is not used.
`ifw.type`	Integer 0, 1, 2 1: class.weights as in 0, divided by min(class.weights) 2: class.weights as in 0, divided by max(class.weights)
`upsample`	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
`downsample`	Logical: If TRUE, downsample majority class to match size of minority class
`resample.seed`	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)

`intercept`	Logical: If TRUE, fit an intercept term.
`polynomial`	Logical: if TRUE, run lm on `poly(x, poly.d)` (creates orthogonal polynomials)
`poly.d`	Integer: degree of polynomial.
`poly.raw`	Logical: if TRUE, use raw polynomials. Default, which should not really be changed is FALSE
`print.plot`	Logical: if TRUE, produce plot using `mplot3` Takes precedence over `plot.fitted` and `plot.predicted`.
`plot.fitted`	Logical: if TRUE, plot True (y) vs Fitted
`plot.predicted`	Logical: if TRUE, plot True (y.test) vs Predicted. Requires `x.test` and `y.test`
`plot.theme`	Character: "zero", "dark", "box", "darkbox"
`na.action`	How to handle missing values. See `?na.fail`
`removeMissingLevels`	Logical: If TRUE, finds factors in `x.test` that contain levels not present in `x` and substitutes with NA. This would result in error otherwise and no predictions would be made, ending `s_GLM` prematurely
`question`	Character: the question you are attempting to answer with this model, in plain language.
`verbose`	Logical: If TRUE, print summary to screen.
`trace`	Integer: If higher than 0, will print more information to the console.
`outdir`	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if `save.mod` is TRUE
`save.mod`	Logical: If TRUE, save all output to an RDS file in `outdir` `save.mod` is TRUE by default if an `outdir` is defined. If set to TRUE, and no `outdir` is defined, outdir defaults to `paste0("./s.", mod.name)`
`...`	Additional arguments

Details

A common problem with `glm` arises when the testing set containts a predictor with more levels than those in the same predictor in the training set, resulting in error. This can happen when training on resamples of a data set, especially after stratifying against a different outcome, and results in error and no prediction. `s_GLM` automatically finds such cases and substitutes levels present in `x.test` and not in `x` with NA.

Value

`rtMod`

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: s_AdaBoost(), s_AddTree(), s_BART(), s_BRUTO(), s_BayesGLM(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GAM(), s_GBM(), s_GLMNET(), s_GLMTree(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_HAL(), s_KNN(), s_LDA(), s_LM(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MARS(), s_MLRF(), s_NBayes(), s_NLA(), s_NLS(), s_NW(), s_PPR(), s_PolyMARS(), s_QDA(), s_QRNN(), s_RF(), s_RFSRC(), s_Ranger(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBoost(), s_XRF()

Other Interpretable models: s_AddTree(), s_C50(), s_CART(), s_GLMNET(), s_GLMTree(), s_LMTree()

Examples

```
x <- rnorm(100)
y <- .6 * x + 12 + rnorm(100) / 2
mod <- s_GLM(x, y)
```

---

s_GLMNET — *GLM with Elastic Net Regularization [C, R, S]*

Description

Train an elastic net model

Usage

```
s_GLMNET(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  grid.resample.params = setup.resample("kfold", 5),
  gridsearch.type = c("exhaustive", "randomized"),
  gridsearch.randomized.p = 0.1,
  intercept = TRUE,
  nway.interactions = 0,
  family = NULL,
  alpha = seq(0, 1, 0.2),
  lambda = NULL,
  nlambda = 100,
  which.cv.lambda = c("lambda.1se", "lambda.min"),
  penalty.factor = NULL,
  weights = NULL,
  ifw = TRUE,
  ifw.type = 2,
```

```
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  res.summary.fn = mean,
  metric = NULL,
  maximize = NULL,
  .gs = FALSE,
  n.cores = rtCores,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

`x`	Numeric vector or matrix / data frame of features i.e. independent variables
`y`	Numeric vector of outcome, i.e. dependent variable
`x.test`	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in `x`
`y.test`	Numeric vector of testing set outcome
`x.name`	Character: Name for feature set
`y.name`	Character: Name for outcome
`grid.resample.params`	List: Output of setup.resample defining grid search parameters.
`gridsearch.type`	Character: Type of grid search to perform: "exhaustive" or "randomized".
`gridsearch.randomized.p`	Float (0, 1): If `gridsearch.type = "randomized"`, randomly test this proportion of combinations.
`intercept`	Logical: If TRUE, include intercept in the model.
`nway.interactions`	Integer: Number of n-way interactions to include in the model.
`family`	Error distribution and link function. See `stats::family`
`alpha`	[gS] Float [0, 1]: The elasticnet mixing parameter: `a = 0` is the ridge penalty, `a = 1` is the lasso penalty
`lambda`	[gS] Float vector: Best left to NULL, `cv.glmnet` will compute its own lambda sequence
`nlambda`	Integer: Number of lambda values to compute

which.cv.lambda
: Character: Which lambda to use for prediction: "lambda.1se" or "lambda.min"

penalty.factor
: Float vector: Multiply the penalty for each coefficient by the values in this vector. This is most useful for specifying different penalties for different groups of variables

weights
: Numeric vector: Weights for cases. For classification, `weights` takes precedence over `ifw`, therefore set `weights = NULL` if using `ifw`. Note: If `weight` are provided, `ifw` is not used. Leave NULL if setting `ifw = TRUE`.

ifw
: Logical: If TRUE, apply inverse frequency weighting (for Classification only). Note: If `weights` are provided, `ifw` is not used.

ifw.type
: Integer 0, 1, 2 1: class.weights as in 0, divided by min(class.weights) 2: class.weights as in 0, divided by max(class.weights)

upsample
: Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness

downsample
: Logical: If TRUE, downsample majority class to match size of minority class

resample.seed
: Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)

res.summary.fn
: Function: Used to average resample runs.

metric
: Character: Metric to minimize, or maximize if `maximize = TRUE` during grid search. Default = NULL, which results in "Balanced Accuracy" for Classification, "MSE" for Regression, and "Coherence" for Survival Analysis.

maximize
: Logical: If TRUE, `metric` will be maximized if grid search is run.

.gs
: (Internal use only)

n.cores
: Integer: Number of cores to use.

print.plot
: Logical: if TRUE, produce plot using `mplot3` Takes precedence over `plot.fitted` and `plot.predicted`.

plot.fitted
: Logical: if TRUE, plot True (y) vs Fitted

plot.predicted
: Logical: if TRUE, plot True (y.test) vs Predicted. Requires `x.test` and `y.test`

plot.theme
: Character: "zero", "dark", "box", "darkbox"

question
: Character: the question you are attempting to answer with this model, in plain language.

verbose
: Logical: If TRUE, print summary to screen.

outdir
: Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if `save.mod` is TRUE

save.mod
: Logical: If TRUE, save all output to an RDS file in `outdir` `save.mod` is TRUE by default if an `outdir` is defined. If set to TRUE, and no `outdir` is defined, outdir defaults to `paste0("./s.", mod.name)`

...
: Additional arguments

Details

s_GLMNET runs glmnet::cv.glmnet for each value of alpha, for each resample in grid.resample.params. Mean values for min.lambda and MSE (Regression) or Accuracy (Classification) are aggregated for each alpha and resample combination
\[gS\] Indicates tunable hyperparameters: If more than a single value is provided, grid search will be automatically performed

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: s_AdaBoost(), s_AddTree(), s_BART(), s_BRUTO(), s_BayesGLM(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GAM(), s_GBM(), s_GLM(), s_GLMTree(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_HAL(), s_KNN(), s_LDA(), s_LM(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MARS(), s_MLRF(), s_NBayes(), s_NLA(), s_NLS(), s_NW(), s_PPR(), s_PolyMARS(), s_QDA(), s_QRNN(), s_RF(), s_RFSRC(), s_Ranger(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBoost(), s_XRF()

Other Interpretable models: s_AddTree(), s_C50(), s_CART(), s_GLM(), s_GLMTree(), s_LMTree()

---

s_GLMTree *Generalized Linear Model Tree [R]*

Description

Train a GLMTree for regression or classification using partykit::glmtree

Usage

```
s_GLMTree(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  weights = NULL,
  alpha = 0.05,
  bonferroni = TRUE,
  minsize = NULL,
  maxdepth = Inf,
  prune = NULL,
  minsplit = minsize,
  minbucket = minsize,
```

```
  epsilon = 1e-08,
  maxit = 25,
  ifw = TRUE,
  ifw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  na.action = na.exclude,
  grid.resample.params = setup.resample("kfold", 5),
  gridsearch.type = c("exhaustive", "randomized"),
  gridsearch.randomized.p = 0.1,
  metric = NULL,
  maximize = NULL,
  n.cores = rtCores,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  grid.verbose = verbose,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

`x`	Numeric vector or matrix / data frame of features i.e. independent variables
`y`	Numeric vector of outcome, i.e. dependent variable
`x.test`	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in `x`
`y.test`	Numeric vector of testing set outcome
`x.name`	Character: Name for feature set
`y.name`	Character: Name for outcome
`weights`	Numeric vector: Weights for cases. For classification, `weights` takes precedence over `ifw`, therefore set `weights = NULL` if using `ifw`. Note: If `weight` are provided, `ifw` is not used. Leave NULL if setting `ifw = TRUE`.
`maxdepth`	[gS] Integer: Maximum depth of tree.
`minsplit`	[gS] Integer: Minimum number of cases that must belong in a node before considering a split.
`minbucket`	[gS] Integer: Minimum number of cases allowed in a child node.
`ifw`	Logical: If TRUE, apply inverse frequency weighting (for Classification only). Note: If `weights` are provided, `ifw` is not used.
`ifw.type`	Integer 0, 1, 2 1: class.weights as in 0, divided by min(class.weights) 2: class.weights as in 0, divided by max(class.weights)

upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
grid.resample.params	List: Output of setup.resample defining grid search parameters.
gridsearch.type	Character: Type of grid search to perform: "exhaustive" or "randomized".
gridsearch.randomized.p	Float (0, 1): If `gridsearch.type = "randomized"`, randomly test this proportion of combinations.
metric	Character: Metric to minimize, or maximize if `maximize = TRUE` during grid search. Default = NULL, which results in "Balanced Accuracy" for Classification, "MSE" for Regression, and "Coherence" for Survival Analysis.
maximize	Logical: If TRUE, `metric` will be maximized if grid search is run.
n.cores	Integer: Number of cores to use.
print.plot	Logical: if TRUE, produce plot using `mplot3` Takes precedence over `plot.fitted` and `plot.predicted`.
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires `x.test` and `y.test`
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
grid.verbose	Logical: Passed to `gridSearchLearn`
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if `save.mod` is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in `outdir` `save.mod` is TRUE by default if an `outdir` is defined. If set to TRUE, and no `outdir` is defined, outdir defaults to `paste0("./s.", mod.name)`
...	Additional arguments passed to `partykit::mob_control`

Value

Object of class `rtMod`

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: s_AdaBoost(), s_AddTree(), s_BART(), s_BRUTO(), s_BayesGLM(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GAM(), s_GBM(), s_GLM(), s_GLMNET(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_HAL(), s_KNN(), s_LDA(), s_LM(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MARS(), s_MLRF(), s_NBayes(), s_NLA(), s_NLS(), s_NW(), s_PPR(), s_PolyMARS(), s_QDA(), s_QRNN(), s_RF(), s_RFSRC(), s_Ranger(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBoost(), s_XRF()

Other Tree-based methods: s_AdaBoost(), s_AddTree(), s_BART(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GBM(), s_H2OGBM(), s_H2ORF(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MLRF(), s_RF(), s_RFSRC(), s_Ranger(), s_XGBoost(), s_XRF()

Other Interpretable models: s_AddTree(), s_C50(), s_CART(), s_GLM(), s_GLMNET(), s_LMTree()

---

s_GLS *Generalized Least Squares [R]*

Description

Train a Generalized Least Squares regression model using nlme::gls

Usage

```
s_GLS(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  interactions = FALSE,
  nway.interactions = 0,
  covariate = NULL,
  weights = NULL,
  intercept = TRUE,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  na.action = na.exclude,
  question = NULL,
  verbose = TRUE,
  trace = 0,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
interactions	List of character pairs denoting column names in x that should be entered as interaction terms in the GLM formula
nway.interactions	Integer: Include n-way interactions. This integer defines the n in: formula = y ~^n
covariate	Character: Name of column. Will include interactions between all features this variable.
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ifw, therefore set weights = NULL if using ifw. Note: If weight are provided, ifw is not used. Leave NULL if setting ifw = TRUE.
intercept	Logical: If TRUE, fit an intercept term.
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted.
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
na.action	How to handle missing values. See ?na.fail
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If higher than 0, will print more information to the console.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments

Value

rtMod

Author(s)

E.D. Gennatas

See Also

Other Supervised Learning: s_AdaBoost(), s_AddTree(), s_BART(), s_BRUTO(), s_BayesGLM(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GAM(), s_GBM(), s_GLM(), s_GLMNET(), s_GLMTree(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_HAL(), s_KNN(), s_LDA(), s_LM(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MARS(), s_MLRF(), s_NBayes(), s_NLA(), s_NLS(), s_NW(), s_PPR(), s_PolyMARS(), s_QDA(), s_QRNN(), s_RF(), s_RFSRC(), s_Ranger(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBoost(), s_XRF()

---

s_H2ODL *Deep Learning on H2O (C, R)*

Description

Trains a Deep Neural Net using H2O (http://www.h2o.ai) Check out the H2O Flow at [ip]:[port], Default IP:port is "localhost:54321" e.g. if running on localhost, point your web browser to localhost:54321

Usage

```
s_H2ODL(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.valid = NULL,
  y.valid = NULL,
  x.name = NULL,
  y.name = NULL,
  ip = "localhost",
  port = 54321,
  n.hidden.nodes = c(20, 20),
  epochs = 1000,
  activation = "Rectifier",
  mini.batch.size = 1,
  learning.rate = 0.005,
  adaptive.rate = TRUE,
  rho = 0.99,
  epsilon = 1e-08,
  rate.annealing = 1e-06,
  rate.decay = 1,
  momentum.start = 0,
  momentum.ramp = 1e+06,
  momentum.stable = 0,
  nesterov.accelerated.gradient = TRUE,
  input.dropout.ratio = 0,
  hidden.dropout.ratios = NULL,
  l1 = 0,
```

```
  l2 = 0,
  max.w2 = 3.4028235e+38,
  nfolds = 0,
  initial.biases = NULL,
  initial.weights = NULL,
  loss = "Automatic",
  distribution = "AUTO",
  stopping.rounds = 5,
  stopping.metric = "AUTO",
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  na.action = na.fail,
  n.cores = rtCores,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  trace = 0,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

`x`	Vector / Matrix / Data Frame: Training set Predictors
`y`	Vector: Training set outcome
`x.test`	Vector / Matrix / Data Frame: Testing set Predictors
`y.test`	Vector: Testing set outcome
`x.valid`	Vector / Matrix / Data Frame: Validation set Predictors
`y.valid`	Vector: Validation set outcome
`x.name`	Character: Name for feature set
`y.name`	Character: Name for outcome
`ip`	Character: IP address of H2O server. Default = "localhost"
`port`	Integer: Port number for server. Default = 54321
`n.hidden.nodes`	Integer vector of length equal to the number of hidden layers you wish to create
`epochs`	Integer: How many times to iterate through the dataset. Default = 1000
`activation`	Character: Activation function to use: "Tanh", "TanhWithDropout", "Rectifier", "RectifierWithDropout", "Maxout", "MaxoutWithDropout". Default = "Rectifier"
`learning.rate`	Float: Learning rate to use for training. Default = .005
`adaptive.rate`	Logical: If TRUE, use adaptive learning rate. Default = TRUE

`rate.annealing`	Float: Learning rate annealing: rate / (1 + rate_annealing * samples). Default = 1e-6
`input.dropout.ratio`	Float (0, 1): Dropout ratio for inputs
`hidden.dropout.ratios`	Vector, Float (0, 2): Dropout ratios for hidden layers
`l1`	Float (0, 1): L1 regularization (introduces sparseness; i.e. sets many weights to 0; reduces variance, increases generalizability)
`l2`	Float (0, 1): L2 regularization (prevents very large absolute weights; reduces variance, increases generalizability)
`upsample`	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
`downsample`	Logical: If TRUE, downsample majority class to match size of minority class
`resample.seed`	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
`na.action`	How to handle missing values. See `?na.fail`
`n.cores`	Integer: Number of cores to use
`print.plot`	Logical: if TRUE, produce plot using `mplot3` Takes precedence over `plot.fitted` and `plot.predicted`.
`plot.fitted`	Logical: if TRUE, plot True (y) vs Fitted
`plot.predicted`	Logical: if TRUE, plot True (y.test) vs Predicted. Requires `x.test` and `y.test`
`plot.theme`	Character: "zero", "dark", "box", "darkbox"
`question`	Character: the question you are attempting to answer with this model, in plain language.
`verbose`	Logical: If TRUE, print summary to screen.
`trace`	Integer: If higher than 0, will print more information to the console.
`outdir`	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if `save.mod` is TRUE
`save.mod`	Logical: If TRUE, save all output to an RDS file in `outdir` `save.mod` is TRUE by default if an `outdir` is defined. If set to TRUE, and no `outdir` is defined, outdir defaults to `paste0("./s.", mod.name)`
`...`	Additional parameters to pass to `h2o::h2o.deeplearning`

Details

x & y form the training set. x.test & y.test form the testing set used only to test model generalizability. x.valid & y.valid form the validation set used to monitor training progress

Value

`rtMod` object

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: s_AdaBoost(), s_AddTree(), s_BART(), s_BRUTO(), s_BayesGLM(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GAM(), s_GBM(), s_GLM(), s_GLMNET(), s_GLMTree(), s_GLS(), s_H2OGBM(), s_H2ORF(), s_HAL(), s_KNN(), s_LDA(), s_LM(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MARS(), s_MLRF(), s_NBayes(), s_NLA(), s_NLS(), s_NW(), s_PPR(), s_PolyMARS(), s_QDA(), s_QRNN(), s_RF(), s_RFSRC(), s_Ranger(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBoost(), s_XRF()

Other Deep Learning: d_H2OAE(), s_TFN()

---

s_H2OGBM — *Gradient Boosting Machine on H2O (C, R)*

Description

Trains a Gradient Boosting Machine using H2O (http://www.h2o.ai)

Usage

```
s_H2OGBM(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  ip = "localhost",
  port = 54321,
  h2o.init = TRUE,
  gs.h2o.init = FALSE,
  h2o.shutdown.at.end = TRUE,
  grid.resample.params = setup.resample("kfold", 5),
  metric = NULL,
  maximize = NULL,
  n.trees = 10000,
  force.n.trees = NULL,
  max.depth = 5,
  n.stopping.rounds = 50,
  stopping.metric = "AUTO",
  p.col.sample = 1,
  p.row.sample = 0.9,
  minobsinnode = 5,
```

```
  min.split.improvement = 1e-05,
  quantile.alpha = 0.5,
  learning.rate = 0.01,
  learning.rate.annealing = 1,
  weights = NULL,
  ifw = TRUE,
  ifw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  na.action = na.fail,
  grid.n.cores = 1,
  n.cores = rtCores,
  imetrics = FALSE,
  .gs = FALSE,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  trace = 0,
  grid.verbose = verbose,
  save.mod = FALSE,
  outdir = NULL,
  ...
)
```

Arguments

`x`	Numeric vector or matrix / data frame of features i.e. independent variables
`y`	Numeric vector of outcome, i.e. dependent variable
`x.test`	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in `x`
`y.test`	Numeric vector of testing set outcome
`x.name`	Character: Name for feature set
`y.name`	Character: Name for outcome
`ip`	Character: IP address of H2O server. Default = "localhost"
`port`	Integer: Port number for server. Default = 54321
`h2o.shutdown.at.end`	Logical: If TRUE, run `h2o.shutdown(prompt = FALSE)` after training is complete.
`n.trees`	Integer: Number of trees to grow. Maximum number of trees if `n.stopping.rounds > 0`
`max.depth`	[gS] Integer: Depth of trees to grow

n.stopping.rounds
: Integer: If > 0, stop training if stopping.metric does not improve for this many rounds

stopping.metric
: Character: "AUTO" (Default), "deviance", "logloss", "MSE", "RMSE", "MAE", "RMSLE", "AUC", "lift_top_group", "misclassification", "mean_per_class_error"

p.col.sample
: [gS]

p.row.sample
: [gS]

minobsinnode
: [gS]

learning.rate
: [gS]

learning.rate.annealing
: [gS]

weights
: Numeric vector: Weights for cases. For classification, weights takes precedence over ifw, therefore set weights = NULL if using ifw. Note: If weight are provided, ifw is not used. Leave NULL if setting ifw = TRUE.

ifw
: Logical: If TRUE, apply inverse frequency weighting (for Classification only). Note: If weights are provided, ifw is not used.

ifw.type
: Integer 0, 1, 2 1: class.weights as in 0, divided by min(class.weights) 2: class.weights as in 0, divided by max(class.weights)

upsample
: Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness

downsample
: Logical: If TRUE, downsample majority class to match size of minority class

resample.seed
: Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)

na.action
: How to handle missing values. See ?na.fail

n.cores
: Integer: Number of cores to use

.gs
: Internal use only

print.plot
: Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted.

plot.fitted
: Logical: if TRUE, plot True (y) vs Fitted

plot.predicted
: Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test

plot.theme
: Character: "zero", "dark", "box", "darkbox"

question
: Character: the question you are attempting to answer with this model, in plain language.

verbose
: Logical: If TRUE, print summary to screen.

trace
: Integer: If higher than 0, will print more information to the console.

save.mod
: Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)

outdir
: Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE

...
: Additional arguments

Details

[gS] denotes tunable hyperparameters Warning: If you get an HTTP 500 error at random, use `h2o.shutdown()` to shutdown the server. It will be restarted when `s_H2OGBM` is called

Value

`rtMod` object

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: `s_AdaBoost()`, `s_AddTree()`, `s_BART()`, `s_BRUTO()`, `s_BayesGLM()`, `s_C50()`, `s_CART()`, `s_CTree()`, `s_EVTree()`, `s_GAM()`, `s_GBM()`, `s_GLM()`, `s_GLMNET()`, `s_GLMTree()`, `s_GLS()`, `s_H2ODL()`, `s_H2ORF()`, `s_HAL()`, `s_KNN()`, `s_LDA()`, `s_LM()`, `s_LMTree()`, `s_LightCART()`, `s_LightGBM()`, `s_MARS()`, `s_MLRF()`, `s_NBayes()`, `s_NLA()`, `s_NLS()`, `s_NW()`, `s_PPR()`, `s_PolyMARS()`, `s_QDA()`, `s_QRNN()`, `s_RF()`, `s_RFSRC()`, `s_Ranger()`, `s_SDA()`, `s_SGD()`, `s_SPLS()`, `s_SVM()`, `s_TFN()`, `s_XGBoost()`, `s_XRF()`

Other Tree-based methods: `s_AdaBoost()`, `s_AddTree()`, `s_BART()`, `s_C50()`, `s_CART()`, `s_CTree()`, `s_EVTree()`, `s_GBM()`, `s_GLMTree()`, `s_H2ORF()`, `s_LMTree()`, `s_LightCART()`, `s_LightGBM()`, `s_MLRF()`, `s_RF()`, `s_RFSRC()`, `s_Ranger()`, `s_XGBoost()`, `s_XRF()`

---

s_H2ORF *Random Forest on H2O (C, R)*

Description

Trains a Random Forest model using H2O (http://www.h2o.ai)

Usage

```
s_H2ORF(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.valid = NULL,
  y.valid = NULL,
  x.name = NULL,
  y.name = NULL,
  ip = "localhost",
  port = 54321,
  n.trees = 500,
```

```
  max.depth = 20,
  n.stopping.rounds = 0,
  mtry = -1,
  nfolds = 0,
  weights = NULL,
  balance.classes = TRUE,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  na.action = na.fail,
  h2o.shutdown.at.end = TRUE,
  n.cores = rtCores,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  trace = 0,
  save.mod = FALSE,
  outdir = NULL,
  ...
)
```

Arguments

x Training set features

y Training set outcome

x.test Testing set features (Used to evaluate model performance)

y.test Testing set outcome

x.valid Validation set features (Used to build model / tune hyperparameters)

y.valid Validation set outcome

x.name Character: Name for feature set

y.name Character: Name for outcome

ip Character: IP address of H2O server. Default = "localhost"

port Integer: Port to connect to at ip

n.trees Integer: Number of trees to grow

max.depth Integer: Maximum tree depth

n.stopping.rounds
Integer: Early stopping if simple moving average of this many rounds does not improve. Set to 0 to disable early stopping.

mtry Integer: Number of variables randomly sampled and considered for splitting at each round. If set to -1, defaults to sqrt(N_features) for classification and N_features/3 for regression.

nfolds
: Integer: Number of folds for K-fold CV used by `h2o.randomForest`. Set to 0 to disable (included for experimentation only, use train_cv for outer resampling)

weights
: Numeric vector: Weights for cases. For classification, `weights` takes precedence over `ifw`, therefore set `weights = NULL` if using `ifw`. Note: If `weight` are provided, `ifw` is not used. Leave NULL if setting `ifw = TRUE`.

balance.classes
: Logical: If TRUE, `h2o.randomForest` will over/undersample to balance data. (included for experimentation only)

upsample
: Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness

downsample
: Logical: If TRUE, downsample majority class to match size of minority class

resample.seed
: Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)

na.action
: How to handle missing values. See `?na.fail`

h2o.shutdown.at.end
: Logical: If TRUE, run `h2o.shutdown(prompt = FALSE)` after training is complete.

n.cores
: Integer: Number of cores to use

print.plot
: Logical: if TRUE, produce plot using `mplot3` Takes precedence over `plot.fitted` and `plot.predicted`.

plot.fitted
: Logical: if TRUE, plot True (y) vs Fitted

plot.predicted
: Logical: if TRUE, plot True (y.test) vs Predicted. Requires `x.test` and `y.test`

plot.theme
: Character: "zero", "dark", "box", "darkbox"

question
: Character: the question you are attempting to answer with this model, in plain language.

verbose
: Logical: If TRUE, print summary to screen.

trace
: Integer: If higher than 0, will print more information to the console.

save.mod
: Logical: If TRUE, save all output to an RDS file in `outdir` `save.mod` is TRUE by default if an `outdir` is defined. If set to TRUE, and no `outdir` is defined, outdir defaults to `paste0("./s.", mod.name)`

outdir
: Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if `save.mod` is TRUE

...
: Additional parameters to pass to `h2o::h2o.randomForest`

Value

`rtMod` object

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: s_AdaBoost(), s_AddTree(), s_BART(), s_BRUTO(), s_BayesGLM(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GAM(), s_GBM(), s_GLM(), s_GLMNET(), s_GLMTree(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_HAL(), s_KNN(), s_LDA(), s_LM(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MARS(), s_MLRF(), s_NBayes(), s_NLA(), s_NLS(), s_NW(), s_PPR(), s_PolyMARS(), s_QDA(), s_QRNN(), s_RF(), s_RFSRC(), s_Ranger(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBoost(), s_XRF()

Other Tree-based methods: s_AdaBoost(), s_AddTree(), s_BART(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GBM(), s_GLMTree(), s_H2OGBM(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MLRF(), s_RF(), s_RFSRC(), s_Ranger(), s_XGBoost(), s_XRF()

---

s_HAL *Highly Adaptive LASSO [C, R, S]*

Description

Train a HAL model

Usage

```
s_HAL(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  family = NULL,
  max.degree = ifelse(ncol(x) >= 20, 2, 3),
  lambda = NULL,
  x.name = NULL,
  y.name = NULL,
  grid.resample.params = setup.resample("kfold", 5),
  gridsearch.type = c("exhaustive", "randomized"),
  gridsearch.randomized.p = 0.1,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  metric = NULL,
  maximize = NULL,
  .gs = FALSE,
  n.cores = rtCores,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
```

```
    verbose = TRUE,
    outdir = NULL,
    save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
    ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
family	Error distribution and link function. See stats::family
max.degree	Integer: The highest order of interaction terms to generate basis functions for.
lambda	Float vector: hal9001::fit_hal lambda
x.name	Character: Name for feature set
y.name	Character: Name for outcome
grid.resample.params	List: Output of setup.resample defining grid search parameters.
gridsearch.type	Character: Type of grid search to perform: "exhaustive" or "randomized".
gridsearch.randomized.p	Float (0, 1): If gridsearch.type = "randomized", randomly test this proportion of combinations.
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
metric	Character: Metric to minimize, or maximize if maximize = TRUE during grid search. Default = NULL, which results in "Balanced Accuracy" for Classification, "MSE" for Regression, and "Coherence" for Survival Analysis.
maximize	Logical: If TRUE, metric will be maximized if grid search is run.
.gs	Internal use only
n.cores	Integer: Number of cores to use.
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted.
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test

plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if `save.mod` is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in `outdir` `save.mod` is TRUE by default if an `outdir` is defined. If set to TRUE, and no `outdir` is defined, outdir defaults to `paste0("./s.", mod.name)`
...	Additional arguments

Details

\[gS\] Indicates tunable hyperparameters: If more than a single value is provided, grid search will be automatically performed

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: `s_AdaBoost()`, `s_AddTree()`, `s_BART()`, `s_BRUTO()`, `s_BayesGLM()`, `s_C50()`, `s_CART()`, `s_CTree()`, `s_EVTree()`, `s_GAM()`, `s_GBM()`, `s_GLM()`, `s_GLMNET()`, `s_GLMTree()`, `s_GLS()`, `s_H2ODL()`, `s_H2OGBM()`, `s_H2ORF()`, `s_KNN()`, `s_LDA()`, `s_LM()`, `s_LMTree()`, `s_LightCART()`, `s_LightGBM()`, `s_MARS()`, `s_MLRF()`, `s_NBayes()`, `s_NLA()`, `s_NLS()`, `s_NW()`, `s_PPR()`, `s_PolyMARS()`, `s_QDA()`, `s_QRNN()`, `s_RF()`, `s_RFSRC()`, `s_Ranger()`, `s_SDA()`, `s_SGD()`, `s_SPLS()`, `s_SVM()`, `s_TFN()`, `s_XGBoost()`, `s_XRF()`

---

s_KNN *k-Nearest Neighbors Classification and Regression (C, R)*

Description

Train a k-Nearest Neighbors learner for regression or classification using `FNN`

Usage

```
s_KNN(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
```

```
  k = 3,
  algorithm = "kd_tree",
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE)
)
```

Arguments

`x`	Numeric vector or matrix / data frame of features i.e. independent variables
`y`	Numeric vector of outcome, i.e. dependent variable
`x.test`	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in `x`
`y.test`	Numeric vector of testing set outcome
`x.name`	Character: Name for feature set
`y.name`	Character: Name for outcome
`k`	Integer: Number of neighbors considered
`algorithm`	Character: Algorithm to use. Options: "kd_tree", "cover_tree", "brute"
`print.plot`	Logical: if TRUE, produce plot using `mplot3` Takes precedence over `plot.fitted` and `plot.predicted`.
`plot.fitted`	Logical: if TRUE, plot True (y) vs Fitted
`plot.predicted`	Logical: if TRUE, plot True (y.test) vs Predicted. Requires `x.test` and `y.test`
`plot.theme`	Character: "zero", "dark", "box", "darkbox"
`question`	Character: the question you are attempting to answer with this model, in plain language.
`verbose`	Logical: If TRUE, print summary to screen.
`outdir`	Optional. Path to directory to save output
`save.mod`	Logical: If TRUE, save all output to an RDS file in `outdir` `save.mod` is TRUE by default if an `outdir` is defined. If set to TRUE, and no `outdir` is defined, outdir defaults to `paste0("./s.", mod.name)`

Value

Object of class `rtMod`

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: s_AdaBoost(), s_AddTree(), s_BART(), s_BRUTO(), s_BayesGLM(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GAM(), s_GBM(), s_GLM(), s_GLMNET(), s_GLMTree(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_HAL(), s_LDA(), s_LM(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MARS(), s_MLRF(), s_NBayes(), s_NLA(), s_NLS(), s_NW(), s_PPR(), s_PolyMARS(), s_QDA(), s_QRNN(), s_RF(), s_RFSRC(), s_Ranger(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBoost(), s_XRF()

---

s_LDA *Linear Discriminant Analysis*

Description

Train an LDA Classifier using MASS::lda

Usage

```
s_LDA(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  prior = NULL,
  method = "moment",
  nu = NULL,
  upsample = TRUE,
  downsample = FALSE,
  resample.seed = NULL,
  x.name = NULL,
  y.name = NULL,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable

x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
prior	Numeric: Prior probabilities of class membership
method	"moment" for standard estimators of the mean and variance, "mle" for MLEs, "mve" to use cov.mve, or "t" for robust estimates based on a t distribution
nu	Integer: Degrees of freedom for method = "t"
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
x.name	Character: Name for feature set
y.name	Character: Name for outcome
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted.
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments passed to MASS::lda

Details

Note: LDA requires all predictors to be numeric. The variable importance output ("varimp") is the vector of coefficients for LD1

Value

rtMod object

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: s_AdaBoost(), s_AddTree(), s_BART(), s_BRUTO(), s_BayesGLM(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GAM(), s_GBM(), s_GLM(), s_GLMNET(), s_GLMTree(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_HAL(), s_KNN(), s_LM(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MARS(), s_MLRF(), s_NBayes(), s_NLA(), s_NLS(), s_NW(), s_PPR(), s_PolyMARS(), s_QDA(), s_QRNN(), s_RF(), s_RFSRC(), s_Ranger(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBoost(), s_XRF()

---

s_LightCART *LightCART Classification and Regression (C, R)*

Description

Train a single decision tree using LightGBM.

Usage

```
s_LightCART(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  weights = NULL,
  ifw = TRUE,
  ifw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  objective = NULL,
  num_leaves = 32L,
  max_depth = -1L,
  lambda_l1 = 0,
  lambda_l2 = 0,
  max_cat_threshold = 32L,
  min_data_per_group = 32L,
  linear_tree = FALSE,
  .gs = FALSE,
  grid.resample.params = setup.resample("kfold", 5),
  gridsearch.type = "exhaustive",
  metric = NULL,
  maximize = NULL,
  importance = TRUE,
  print.plot = FALSE,
```

```
    plot.fitted = NULL,
    plot.predicted = NULL,
    plot.theme = rtTheme,
    question = NULL,
    verbose = TRUE,
    grid.verbose = FALSE,
    lightgbm_verbose = -1,
    save.gridrun = FALSE,
    n.cores = 1,
    n_threads = 0,
    force_col_wise = FALSE,
    force_row_wise = FALSE,
    outdir = NULL,
    save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
    ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ifw, therefore set weights = NULL if using ifw. Note: If weight are provided, ifw is not used. Leave NULL if setting ifw = TRUE.
ifw	Logical: If TRUE, apply inverse frequency weighting (for Classification only). Note: If weights are provided, ifw is not used.
ifw.type	Integer 0, 1, 2 1: class.weights as in 0, divided by min(class.weights) 2: class.weights as in 0, divided by max(class.weights)
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
objective	(Default = NULL)
num_leaves	Integer: [gS] Maximum tree leaves for base learners.
max_depth	Integer: [gS] Maximum tree depth for base learners, <=0 means no limit.
lambda_l1	Numeric: [gS] L1 regularization term

`lambda_l2`	Numeric: [gS] L2 regularization term
`max_cat_threshold`	Integer: Max number of splits to consider for categorical variable
`min_data_per_group`	Integer: Minimum number of observations per categorical group
`linear_tree`	Logical: [gS] If `TRUE`, use linear trees
`.gs`	(Internal use only)
`grid.resample.params`	List: Output of setup.resample defining grid search parameters.
`gridsearch.type`	Character: Type of grid search to perform: "exhaustive" or "randomized".
`metric`	Character: Metric to minimize, or maximize if `maximize = TRUE` during grid search. Default = NULL, which results in "Balanced Accuracy" for Classification, "MSE" for Regression, and "Coherence" for Survival Analysis.
`maximize`	Logical: If TRUE, `metric` will be maximized if grid search is run.
`importance`	Logical: If `TRUE`, calculate variable importance
`print.plot`	Logical: if TRUE, produce plot using `mplot3` Takes precedence over `plot.fitted` and `plot.predicted`.
`plot.fitted`	Logical: if TRUE, plot True (y) vs Fitted
`plot.predicted`	Logical: if TRUE, plot True (y.test) vs Predicted. Requires `x.test` and `y.test`
`plot.theme`	Character: "zero", "dark", "box", "darkbox"
`question`	Character: the question you are attempting to answer with this model, in plain language.
`verbose`	Logical: If TRUE, print summary to screen.
`grid.verbose`	Logical: Passed to `gridSearchLearn`
`lightgbm_verbose`	Integer: Passed to `lightgbm::train`, `< 0`: Fatal, `0`: Error (Warning), `1`: Info, `> 1`: Debug
`save.gridrun`	Logical: If `TRUE`, save all grid search models
`n.cores`	Integer: Number of cores to use.
`n_threads`	Integer: Number of threads for lightgbm using OpenMP. Only parallelize resamples using `n.cores` or the lightgbm execution using this setting.
`force_col_wise`	Logical: If `TRUE`, force column-wise histogram building (See https://lightgbm.readthedocs.io/en/latest/Par
`force_row_wise`	Logical: If `TRUE`, force row-wise histogram building (See https://lightgbm.readthedocs.io/en/latest/Parame
`outdir`	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if `save.mod` is TRUE
`save.mod`	Logical: If TRUE, save all output to an RDS file in `outdir` `save.mod` is TRUE by default if an `outdir` is defined. If set to TRUE, and no `outdir` is defined, outdir defaults to `paste0("./s.", mod.name)`
`...`	Extra arguments appended to `lgb.train`'s `params`.

Details

[gS]: indicates parameter will be autotuned by grid search if multiple values are passed. LightGBM trains trees leaf-wise (best-first) rather than depth-wise. For categorical variables, convert to integer and indicate to lgb they are categorical, so that they are not treated as numeric.

Value

rtMod object

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: s_AdaBoost(), s_AddTree(), s_BART(), s_BRUTO(), s_BayesGLM(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GAM(), s_GBM(), s_GLM(), s_GLMNET(), s_GLMTree(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_HAL(), s_KNN(), s_LDA(), s_LM(), s_LMTree(), s_LightGBM(), s_MARS(), s_MLRF(), s_NBayes(), s_NLA(), s_NLS(), s_NW(), s_PPR(), s_PolyMARS(), s_QDA(), s_QRNN(), s_RF(), s_RFSRC(), s_Ranger(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBoost(), s_XRF()

Other Tree-based methods: s_AdaBoost(), s_AddTree(), s_BART(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GBM(), s_GLMTree(), s_H2OGBM(), s_H2ORF(), s_LMTree(), s_LightGBM(), s_MLRF(), s_RF(), s_RFSRC(), s_Ranger(), s_XGBoost(), s_XRF()

Examples

```
## Not run:
x <- rnormmat(500, 10)
y <- x[, 3] + .5 * x[, 5]^2 + rnorm(500)
dat <- data.frame(x, y)
mod <- s_LightGBM(dat)

## End(Not run)
```

---

s_LightGBM *LightGBM Classification and Regression (C, R)*

Description

Tune hyperparameters using grid search and resampling, train a final model, and validate it

Usage

```
s_LightGBM(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  weights = NULL,
  ifw = TRUE,
  ifw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  boosting = "gbdt",
  objective = NULL,
  max_nrounds = 1000L,
  force_nrounds = NULL,
  early_stopping_rounds = 10L,
  nrounds_default = 100L,
  num_leaves = 32L,
  max_depth = -1L,
  learning_rate = 0.01,
  subsample = 0.8,
  subsample_freq = 1L,
  lambda_l1 = 0,
  lambda_l2 = 0,
  max_cat_threshold = 32L,
  min_data_per_group = 32L,
  linear_tree = FALSE,
  tree_learner = "serial",
  .gs = FALSE,
  grid.resample.params = setup.resample("kfold", 5),
  gridsearch.type = "exhaustive",
  metric = NULL,
  maximize = NULL,
  importance = TRUE,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  grid.verbose = FALSE,
  lightgbm_verbose = -1,
  save.gridrun = FALSE,
  n.cores = 1,
  n_threads = 0,
```

```
    force_col_wise = FALSE,
    force_row_wise = FALSE,
    outdir = NULL,
    save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
    ...
)
```

Arguments

`x`	Numeric vector or matrix / data frame of features i.e. independent variables
`y`	Numeric vector of outcome, i.e. dependent variable
`x.test`	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in `x`
`y.test`	Numeric vector of testing set outcome
`x.name`	Character: Name for feature set
`y.name`	Character: Name for outcome
`weights`	Numeric vector: Weights for cases. For classification, `weights` takes precedence over `ifw`, therefore set `weights = NULL` if using `ifw`. Note: If `weight` are provided, `ifw` is not used. Leave NULL if setting `ifw = TRUE`.
`ifw`	Logical: If TRUE, apply inverse frequency weighting (for Classification only). Note: If `weights` are provided, `ifw` is not used.
`ifw.type`	Integer 0, 1, 2 1: class.weights as in 0, divided by min(class.weights) 2: class.weights as in 0, divided by max(class.weights)
`upsample`	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
`downsample`	Logical: If TRUE, downsample majority class to match size of minority class
`resample.seed`	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
`boosting`	Character: [gS] "gbdt", "rf", "dart", "goss"
`objective`	(Default = NULL)
`max_nrounds`	Integer: Maximum number of rounds to run. Can be set to a high number as early stopping will limit nrounds by monitoring inner CV error
`force_nrounds`	Integer: Number of rounds to run if not estimating optimal number by CV
`early_stopping_rounds`	Integer: Training on resamples of `x` (tuning) will stop if performance does not improve for this many rounds
`nrounds_default`	Integer: Default number of rounds to run if cross-validation fails - likely will never be used
`num_leaves`	Integer: [gS] Maximum tree leaves for base learners.
`max_depth`	Integer: [gS] Maximum tree depth for base learners, <=0 means no limit.

learning_rate Numeric: [gS] Boosting learning rate

subsample Numeric: [gS] Subsample ratio of the training set.

subsample_freq Integer: Subsample every this many iterations

lambda_l1 Numeric: [gS] L1 regularization term

lambda_l2 Numeric: [gS] L2 regularization term

max_cat_threshold
Integer: Max number of splits to consider for categorical variable

min_data_per_group
Integer: Minimum number of observations per categorical group

linear_tree Logical: [gS] If TRUE, use linear trees

tree_learner Character: [gS] "serial", "feature", "data", "voting"

.gs (Internal use only)

grid.resample.params
List: Output of setup.resample defining grid search parameters.

gridsearch.type
Character: Type of grid search to perform: "exhaustive" or "randomized".

metric Character: Metric to minimize, or maximize if maximize = TRUE during grid search. Default = NULL, which results in "Balanced Accuracy" for Classification, "MSE" for Regression, and "Coherence" for Survival Analysis.

maximize Logical: If TRUE, metric will be maximized if grid search is run.

importance Logical: If TRUE, calculate variable importance

print.plot Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted.

plot.fitted Logical: if TRUE, plot True (y) vs Fitted

plot.predicted Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test

plot.theme Character: "zero", "dark", "box", "darkbox"

question Character: the question you are attempting to answer with this model, in plain language.

verbose Logical: If TRUE, print summary to screen.

grid.verbose Logical: Passed to gridSearchLearn

lightgbm_verbose
Integer: Passed to lightgbm::train, < 0: Fatal, 0: Error (Warning), 1: Info, > 1: Debug

save.gridrun Logical: If TRUE, save all grid search models

n.cores Integer: Number of cores to use.

n_threads Integer: Number of threads for lightgbm using OpenMP. Only parallelize resamples using n.cores or the lightgbm execution using this setting.

force_col_wise Logical: If TRUE, force column-wise histogram building (See https://lightgbm.readthedocs.io/en/latest/Par

force_row_wise Logical: If TRUE, force row-wise histogram building (See https://lightgbm.readthedocs.io/en/latest/Parame

outdir Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE

save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Extra arguments appended to lgb.train's params.

Details

[gS]: indicates parameter will be autotuned by grid search if multiple values are passed. LightGBM trains trees leaf-wise (best-first) rather than depth-wise. For categorical variables, convert to integer and indicate to lgb they are categorical, so that they are not treated as numeric.

Value

rtMod object

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: s_AdaBoost(), s_AddTree(), s_BART(), s_BRUTO(), s_BayesGLM(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GAM(), s_GBM(), s_GLM(), s_GLMNET(), s_GLMTree(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_HAL(), s_KNN(), s_LDA(), s_LM(), s_LMTree(), s_LightCART(), s_MARS(), s_MLRF(), s_NBayes(), s_NLA(), s_NLS(), s_NW(), s_PPR(), s_PolyMARS(), s_QDA(), s_QRNN(), s_RF(), s_RFSRC(), s_Ranger(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBoost(), s_XRF()

Other Tree-based methods: s_AdaBoost(), s_AddTree(), s_BART(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GBM(), s_GLMTree(), s_H2OGBM(), s_H2ORF(), s_LMTree(), s_LightCART(), s_MLRF(), s_RF(), s_RFSRC(), s_Ranger(), s_XGBoost(), s_XRF()

Examples

```
## Not run:
x <- rnormmat(500, 10)
y <- x[, 3] + .5 * x[, 5]^2 + rnorm(500)
dat <- data.frame(x, y)
mod <- s_LightGBM(dat)

## End(Not run)
```

s_LightRF *Random Forest using LightGBM*

Description

Random Forest using LightGBM

Usage

```
s_LightRF(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  weights = NULL,
  ifw = TRUE,
  ifw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  objective = NULL,
  nrounds = 500L,
  early_stopping_rounds = -1L,
  num_leaves = 4096L,
  max_depth = -1L,
  learning_rate = 1,
  subsample = 0.623,
  subsample_freq = 1L,
  lambda_l1 = 0,
  lambda_l2 = 0,
  max_cat_threshold = 32L,
  min_data_per_group = 32L,
  linear_tree = FALSE,
  tree_learner = "data_parallel",
  grid.resample.params = setup.resample("kfold", 5),
  gridsearch.type = "exhaustive",
  metric = NULL,
  maximize = NULL,
  importance = TRUE,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
```

```
  grid.verbose = FALSE,
  lightgbm_verbose = -1,
  save.gridrun = FALSE,
  n.cores = 1,
  n_threads = rtCores,
  force_col_wise = FALSE,
  force_row_wise = FALSE,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  .gs = FALSE,
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ifw, therefore set weights = NULL if using ifw. Note: If weight are provided, ifw is not used. Leave NULL if setting ifw = TRUE.
ifw	Logical: If TRUE, apply inverse frequency weighting (for Classification only). Note: If weights are provided, ifw is not used.
ifw.type	Integer 0, 1, 2 1: class.weights as in 0, divided by min(class.weights) 2: class.weights as in 0, divided by max(class.weights)
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
objective	(Default = NULL)
nrounds	Integer: Number of trees to grow
early_stopping_rounds	Integer: Training on resamples of x (tuning) will stop if performance does not improve for this many rounds
num_leaves	Integer: [gS] Maximum tree leaves for base learners.
max_depth	Integer: [gS] Maximum tree depth for base learners, <=0 means no limit.
learning_rate	Numeric: [gS] Boosting learning rate

subsample: Numeric: [gS] Subsample ratio of the training set.

subsample_freq: Integer: Subsample every this many iterations

lambda_l1: Numeric: [gS] L1 regularization term

lambda_l2: Numeric: [gS] L2 regularization term

max_cat_threshold: Integer: Max number of splits to consider for categorical variable

min_data_per_group: Integer: Minimum number of observations per categorical group

linear_tree: Logical: [gS] If TRUE, use linear trees

tree_learner: Character: [gS] "serial", "feature", "data", "voting"

grid.resample.params: List: Output of setup.resample defining grid search parameters.

gridsearch.type: Character: Type of grid search to perform: "exhaustive" or "randomized".

metric: Character: Metric to minimize, or maximize if maximize = TRUE during grid search. Default = NULL, which results in "Balanced Accuracy" for Classification, "MSE" for Regression, and "Coherence" for Survival Analysis.

maximize: Logical: If TRUE, metric will be maximized if grid search is run.

importance: Logical: If TRUE, calculate variable importance

print.plot: Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted.

plot.fitted: Logical: if TRUE, plot True (y) vs Fitted

plot.predicted: Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test

plot.theme: Character: "zero", "dark", "box", "darkbox"

question: Character: the question you are attempting to answer with this model, in plain language.

verbose: Logical: If TRUE, print summary to screen.

grid.verbose: Logical: Passed to gridSearchLearn

lightgbm_verbose: Integer: Passed to lightgbm::train, < 0: Fatal, 0: Error (Warning), 1: Info, > 1: Debug

save.gridrun: Logical: If TRUE, save all grid search models

n.cores: Integer: Number of cores to use.

n_threads: Integer: Number of threads for lightgbm using OpenMP. Only parallelize resamples using n.cores or the lightgbm execution using this setting.

force_col_wise: Logical: If TRUE, force column-wise histogram building (See https://lightgbm.readthedocs.io/en/latest/Par

force_row_wise: Logical: If TRUE, force row-wise histogram building (See https://lightgbm.readthedocs.io/en/latest/Parame

outdir: Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE

save.mod: Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)

.gs: (Internal use only)

...: Extra arguments appended to lgb.train's params.

Author(s)

ED Gennatas

Examples

```
## Not run:
x <- rnormmat(500, 10)
y <- x[, 3] + .5 * x[, 5]^2 + rnorm(500)
dat <- data.frame(x, y)
mod <- s_LightRF(dat)

## End(Not run)
```

---

s_LightRuleFit *RuleFit with LightGBM (C, R)*

Description

Train a LightGBM gradient boosting model, extract rules, and fit using LASSO

Usage

```
s_LightRuleFit(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  lgbm.mod = NULL,
  n_trees = 200,
  num_leaves = 32L,
  max_depth = 4,
  learning_rate = 0.1,
  subsample = 0.666,
  subsample_freq = 1L,
  lambda_l1 = 0,
  lambda_l2 = 0,
  objective = NULL,
  importance = FALSE,
  lgbm.ifw = TRUE,
 lgbm.grid.resample.params = setup.resample(resampler = "kfold", n.resamples = 5),
  glmnet.ifw = TRUE,
  alpha = 1,
  lambda = NULL,
 glmnet.grid.resample.params = setup.resample(resampler = "kfold", n.resamples = 5),
  grid.resample.params = setup.resample("kfold", 5),
  gridsearch.type = "exhaustive",
  metric = NULL,
```

```
    maximize = NULL,
    grid.verbose = FALSE,
    save.gridrun = FALSE,
    weights = NULL,
    empirical_risk = TRUE,
    cases_by_rules = NULL,
    save_cases_by_rules = FALSE,
    x.name = NULL,
    y.name = NULL,
    n.cores = rtCores,
    question = NULL,
    print.plot = FALSE,
    plot.fitted = NULL,
    plot.predicted = NULL,
    plot.theme = rtTheme,
    outdir = NULL,
    save.mod = if (!is.null(outdir)) TRUE else FALSE,
    verbose = TRUE,
    trace = 0,
    .gs = FALSE
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
lgbm.mod	rtMod object created by s_LightGBM. If provided, the gradient boosting step is skipped.
num_leaves	Integer: [gS] Maximum tree leaves for base learners.
max_depth	Integer: [gS] Maximum tree depth for base learners, <=0 means no limit.
learning_rate	Numeric: [gS] Boosting learning rate
subsample	Numeric: [gS] Subsample ratio of the training set.
subsample_freq	Integer: Subsample every this many iterations
lambda_l1	Numeric: [gS] L1 regularization term
lambda_l2	Numeric: [gS] L2 regularization term
objective	(Default = NULL)
importance	Logical: If TRUE, calculate variable importance
alpha	[gS] Float [0, 1]: The elasticnet mixing parameter: a = 0 is the ridge penalty, a = 1 is the lasso penalty
lambda	[gS] Float vector: Best left to NULL, cv.glmnet will compute its own lambda sequence

grid.resample.params
: List: Output of setup.resample defining grid search parameters.

gridsearch.type
: Character: Type of grid search to perform: "exhaustive" or "randomized".

metric
: Character: Metric to minimize, or maximize if maximize = TRUE during grid search. Default = NULL, which results in "Balanced Accuracy" for Classification, "MSE" for Regression, and "Coherence" for Survival Analysis.

maximize
: Logical: If TRUE, metric will be maximized if grid search is run.

grid.verbose
: Logical: Passed to gridSearchLearn

save.gridrun
: Logical: If TRUE, save all grid search models

weights
: Numeric vector: Weights for cases. For classification, weights takes precedence over ifw, therefore set weights = NULL if using ifw. Note: If weight are provided, ifw is not used. Leave NULL if setting ifw = TRUE.

empirical_risk
: Logical: If TRUE, calculate empirical risk

cases_by_rules
: Matrix of cases by rules from a previoue rulefit run. If provided, the GBM step is skipped.

save_cases_by_rules
: Logical: If TRUE, save cases_by_rules to object

x.name
: Character: Name for feature set

y.name
: Character: Name for outcome

n.cores
: Integer: Number of cores to use

question
: Character: the question you are attempting to answer with this model, in plain language.

print.plot
: Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted.

plot.fitted
: Logical: if TRUE, plot True (y) vs Fitted

plot.predicted
: Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test

plot.theme
: Character: "zero", "dark", "box", "darkbox"

outdir
: Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE

save.mod
: Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)

verbose
: Logical: If TRUE, print summary to screen.

trace
: Integer: Verbosity level

.gs
: (Internal use only)

Details

Based on "Predictive Learning via Rule Ensembles" by Friedman and Popescu http://statweb.stanford.edu/~jhf/ftp/RuleFit.pdf

Value

rtMod object

Author(s)

E.D. Gennatas

References

Friedman JH, Popescu BE, "Predictive Learning via Rule Ensembles", http://statweb.stanford.edu/~jhf/ftp/RuleFit.pdf

---

s_LIHAD *The Linear Hard Hybrid Tree: Hard Additive Tree (no gamma) with Linear Nodes [R]*

Description

Train a Linear Hard Hybrid Tree for Regression

Usage

```
s_LIHAD(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  max.depth = 3,
  alpha = 0,
  lambda = 0.1,
  lincoef.params = setup.lincoef("glmnet"),
  minobsinnode = 2,
  minobsinnode.lin = 10,
  learning.rate = 1,
  part.minsplit = 2,
  part.xval = 0,
  part.max.depth = 1,
  part.cp = 0,
  weights = NULL,
  metric = "MSE",
  maximize = FALSE,
  grid.resample.params = setup.grid.resample(),
  keep.x = FALSE,
  simplify = TRUE,
  cxrcoef = FALSE,
  n.cores = rtCores,
  verbose = TRUE,
  verbose.predict = FALSE,
  trace = 0,
  x.name = NULL,
  y.name = NULL,
  question = NULL,
```

```
  outdir = NULL,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  save.mod = FALSE
)
```

Arguments

`x`	Numeric vector or matrix / data frame of features i.e. independent variables
`y`	Numeric vector of outcome, i.e. dependent variable
`x.test`	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in `x`
`y.test`	Numeric vector of testing set outcome
`max.depth`	[gS] Integer: Max depth of additive tree. Default = 3
`alpha`	[gS] Float: `lincoef` alpha Overrides `lincoef.params` alpha
`lambda`	[gS] Float: `lincoef` lambda. Overrides `lincoef.params` lambda
`lincoef.params`	Named List: Output of setup.lincoef
`minobsinnode`	[gS] Integer: Minimum N observations needed in node, before considering splitting
`minobsinnode.lin`	Integer: Minimum N observations needed in node in order to train linear model.
`learning.rate`	[gS] Float (0, 1): Learning rate.
`part.max.depth`	Integer: Max depth for each tree model within the additive tree
`part.cp`	[gS] Float: Minimum complexity needed to allow split by `rpart`.
`weights`	Numeric vector: Weights for cases. For classification, `weights` takes precedence over `ifw`, therefore set `weights = NULL` if using `ifw`. Note: If `weight` are provided, `ifw` is not used. Leave NULL if setting `ifw = TRUE`.
`cxrcoef`	Logical: Passed to predict.lihad, if TRUE, returns cases by coefficients matrix
`verbose`	Logical: If TRUE, print summary to screen.
`trace`	Integer: If higher than 0, will print more information to the console.
`x.name`	Character: Name for feature set
`y.name`	Character: Name for outcome
`question`	Character: the question you are attempting to answer with this model, in plain language.
`outdir`	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if `save.mod` is TRUE
`print.plot`	Logical: if TRUE, produce plot using `mplot3` Takes precedence over `plot.fitted` and `plot.predicted`.
`plot.fitted`	Logical: if TRUE, plot True (y) vs Fitted
`plot.predicted`	Logical: if TRUE, plot True (y.test) vs Predicted. Requires `x.test` and `y.test`

plot.theme	Character: "zero", "dark", "box", "darkbox"
save.mod	Logical: If TRUE, save all output to an RDS file in `outdir` `save.mod` is TRUE by default if an `outdir` is defined. If set to TRUE, and no `outdir` is defined, outdir defaults to `paste0("./s.", mod.name)`

Details

The Hybrid Tree grows a tree using a sequence of regularized linear models and tree stumps. Use s_LINAD for the standard Linear Additive Tree Algorithm, which grows branches stepwise and includes all observations weighted by gamma

Grid searched parameters: max.depth, alpha, lambda, minobsinnode, learning.rate, part.cp

Author(s)

E.D. Gennatas

`s_LIHADBoost` *Boosting of Linear Hard Additive Trees [R]*

Description

Boost a Linear Hard Additive Tree (i.e. LIHAD, i.e. LINAD with hard splits)

Usage

```
s_LIHADBoost(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  resid = NULL,
  boost.obj = NULL,
  learning.rate = 0.5,
  case.p = 1,
  max.depth = 5,
  gamma = 0.1,
  alpha = 0,
  lambda = 1,
  lambda.seq = NULL,
  minobsinnode = 2,
  minobsinnode.lin = 10,
  shrinkage = 1,
  part.minsplit = 2,
  part.xval = 0,
  part.max.depth = 1,
  part.cp = 0,
  part.minbucket = 5,
```

```
  lin.type = c("glmnet", "cv.glmnet", "lm.ridge", "allSubsets", "forwardStepwise",
    "backwardStepwise", "glm", "sgd", "solve", "none"),
  cv.glmnet.nfolds = 5,
  which.cv.glmnet.lambda = "lambda.min",
  max.iter = 10,
  tune.n.iter = TRUE,
  earlystop.params = setup.earlystop(),
  lookback = TRUE,
  init = NULL,
  .gs = FALSE,
  grid.resample.params = setup.resample("kfold", 5),
  gridsearch.type = "exhaustive",
  metric = NULL,
  maximize = NULL,
  cxrcoef = FALSE,
  print.progress.every = 5,
  print.error.plot = "final",
  x.name = NULL,
  y.name = NULL,
  question = NULL,
  base.verbose = FALSE,
  verbose = TRUE,
  grid.verbose = FALSE,
  trace = 0,
  prefix = NULL,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  print.plot = FALSE,
  print.base.plot = FALSE,
  print.tune.plot = TRUE,
  plot.type = "l",
  save.gridrun = FALSE,
  outdir = NULL,
  n.cores = rtCores,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

`x`	Numeric vector or matrix / data frame of features i.e. independent variables
`y`	Numeric vector of outcome, i.e. dependent variable
`x.test`	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in `x`
`y.test`	Numeric vector of testing set outcome
`learning.rate`	Float (0, 1] Learning rate for the additive steps

max.iter	Integer: Maximum number of iterations (additive steps) to perform. Default = 10
init	Float: Initial value for prediction. Default = mean(y)
print.error.plot	String or Integer: "final" plots a training and validation (if available) error curve at the end of training. If integer, plot training and validation error curve every this many iterations during training
x.name	Character: Name for feature set
y.name	Character: Name for outcome
question	Character: the question you are attempting to answer with this model, in plain language.
base.verbose	Logical: verbose argument passed to learner
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If > 0, print diagnostic info to console
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted.
print.base.plot	Logical: Passed to print.plot argument of base learner, i.e. if TRUE, print error plot for each base learner
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional parameters to be passed to learner

Details

By default, early stopping works by checking training loss.

Author(s)

E.D. Gennatas

s_LINAD *Linear Additive Tree (C, R)*

Description

Train a Linear Additive Tree for Regression or Binary Classification

Usage

```
s_LINAD(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  weights = NULL,
  max.leaves = 20,
  lookback = TRUE,
  force.max.leaves = NULL,
  learning.rate = 0.5,
  ifw = TRUE,
  ifw.type = 1,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  leaf.model = c("line", "spline"),
  gamlearner = "gamsel",
  gam.params = list(),
  nvmax = 3,
  gamma = 0.5,
  gamma.on.lin = FALSE,
 lin.type = c("glmnet", "forwardStepwise", "cv.glmnet", "lm.ridge", "allSubsets",
    "backwardStepwise", "glm", "solve", "none"),
  first.lin.type = "cv.glmnet",
  first.lin.learning.rate = 1,
  first.lin.alpha = 1,
  first.lin.lambda = NULL,
  cv.glmnet.nfolds = 5,
  which.cv.glmnet.lambda = "lambda.min",
  alpha = 1,
  lambda = 0.05,
  lambda.seq = NULL,
  minobsinnode.lin = 10,
  part.minsplit = 2,
  part.xval = 0,
  part.max.depth = 1,
  part.cp = 0,
  part.minbucket = 1,
```

```
    .rho = TRUE,
    rho.max = 1000,
    init = NULL,
    metric = "auto",
    maximize = NULL,
    grid.resample.params = setup.resample("kfold", 5),
    gridsearch.type = "exhaustive",
    save.gridrun = FALSE,
    select.leaves.smooth = FALSE,
    cluster = FALSE,
    keep.x = FALSE,
    simplify = TRUE,
    cxrcoef = FALSE,
    n.cores = rtCores,
    .preprocess = NULL,
    verbose = TRUE,
    grid.verbose = FALSE,
    plot.tuning = FALSE,
    verbose.predict = FALSE,
    trace = 1,
    x.name = NULL,
    y.name = NULL,
    question = NULL,
    outdir = NULL,
    print.plot = FALSE,
    plot.fitted = NULL,
    plot.predicted = NULL,
    plot.theme = rtTheme,
    save.mod = FALSE,
    .gs = FALSE
  )
```

Arguments

`x`	Numeric vector or matrix / data frame of features i.e. independent variables
`y`	Numeric vector of outcome, i.e. dependent variable
`x.test`	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in `x`
`y.test`	Numeric vector of testing set outcome
`weights`	Numeric vector: Weights for cases. For classification, `weights` takes precedence over `ifw`, therefore set `weights = NULL` if using `ifw`. Note: If `weight` are provided, `ifw` is not used. Leave NULL if setting `ifw = TRUE`.
`max.leaves`	Integer: Maximum number of terminal nodes to grow. Setting this to a value > 1, triggers cross-validation to find best number of leaves. To force a given number of leaves and not cross-validate, set `force.max.leaves` to any (integer) value.
`lookback`	Logical: If TRUE, use validation error to decide best number of leaves to use.

`force.max.leaves`	Integer: If set, `max.leaves` is ignored and the tree will attempt to reach this number of leaves, without performing tuning number of leaves.
`learning.rate`	[gS] Numeric: learning rate for steps after initial linear model
`ifw`	Logical: If TRUE, apply inverse frequency weighting (for Classification only). Note: If `weights` are provided, `ifw` is not used.
`ifw.type`	Integer 0, 1, 2 1: class.weights as in 0, divided by min(class.weights) 2: class.weights as in 0, divided by max(class.weights)
`upsample`	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
`downsample`	Logical: If TRUE, downsample majority class to match size of minority class
`resample.seed`	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
`nvmax`	[gS] Integer: Number of max features to use for lin.type "allSubsets", "forwardStepwise", or "backwardStepwise". If values greater than n of features in `x` are provided, they will be excluded
`gamma`	[gS] Numeric: Soft weighting parameter. Weights of cases that do not belong to node get multiplied by this amount
`lin.type`	Character: One of "glmnet", "forwardStepwise", "cv.glmnet", "lm.ridge", "allSubsets", "backwardStepwise", "glm", "solve", or "none" to not fit linear models See lincoef for more
`first.lin.type`	Character: same options as `lin.type`, the first linear model to fit on the root node.
`first.lin.alpha`	Numeric: alpha for the first linear model, if `first.lin.type` is "glmnet" or "cv.glmnet"
`lambda`	[gS] Numeric: lambda value for lin.type `glmnet`, `cv.glmnet`, `lm.ridge`
`minobsinnode.lin`	[gS] Integer: Minimum number of observation needed to fit linear model
`part.minsplit`	[gS] Integer: Minimum number of observations in node to consider splitting
`part.max.depth`	Integer: Max depth for each tree model within the additive tree
`part.cp`	[gS] Numeric: Split must decrease complexity but at least this much to be considered
`part.minbucket`	[gS] Integer: Minimum number of observations allowed in child node to allow splitting
`init`	Initial value. Default = `mean(y)`
`verbose`	Logical: If TRUE, print summary to screen.
`plot.tuning`	Logical: If TRUE, plot validation error during gridsearch
`trace`	Integer: If higher than 0, will print more information to the console.
`x.name`	Character: Name for feature set

`y.name`	Character: Name for outcome
`question`	Character: the question you are attempting to answer with this model, in plain language.
`outdir`	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if `save.mod` is TRUE
`print.plot`	Logical: if TRUE, produce plot using `mplot3` Takes precedence over `plot.fitted` and `plot.predicted`.
`plot.fitted`	Logical: if TRUE, plot True (y) vs Fitted
`plot.predicted`	Logical: if TRUE, plot True (y.test) vs Predicted. Requires `x.test` and `y.test`
`plot.theme`	Character: "zero", "dark", "box", "darkbox"
`save.mod`	Logical: If TRUE, save all output to an RDS file in `outdir` `save.mod` is TRUE by default if an `outdir` is defined. If set to TRUE, and no `outdir` is defined, outdir defaults to `paste0("./s.", mod.name)`
`.gs`	internal use only

Details

The Linear Additive Tree trains a tree using a sequence of regularized linear models and splits. We specify an upper threshold of leaves using `max.leaves` instead of directly defining a number, because depending on the other parameters and the datasets, splitting may stop early.

[gS] indicates tunable hyperparameters that can accept a vector of possible values

Author(s)

E.D. Gennatas

`s_LINOA` *Linear Optimized Additive Tree (C, R)*

Description

Train a Linear Optimized Additive Tree

Usage

```
s_LINOA(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  weights = NULL,
  ifw = TRUE,
  ifw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
```

```
  resample.seed = NULL,
  max.leaves = 8,
  learning.rate = 0.5,
  select.leaves.smooth = TRUE,
  force.max.leaves = NULL,
  lookback = TRUE,
  gamma = 0,
  n.quantiles = 20,
  minobsinnode = NULL,
  minbucket = NULL,
  lin.type = c("forwardStepwise", "glmnet", "cv.glmnet", "lm.ridge", "allSubsets",
    "backwardStepwise", "glm", "solve", "none"),
  alpha = 1,
  lambda = 0.05,
  lambda.seq = NULL,
  cv.glmnet.nfolds = 5,
  which.cv.glmnet.lambda = "lambda.min",
  nbest = 1,
  nvmax = 3,
  .rho = TRUE,
  rho.max = 1000,
  init = NULL,
  metric = "auto",
  maximize = NULL,
  grid.resample.params = setup.resample("kfold", 5),
  gridsearch.type = "exhaustive",
  save.gridrun = FALSE,
  grid.verbose = verbose,
  keep.x = FALSE,
  simplify = TRUE,
  cxrcoef = FALSE,
  n.cores = rtCores,
  splitline.cores = 1,
  .preprocess = NULL,
  plot.tuning = TRUE,
  verbose.predict = FALSE,
  x.name = NULL,
  y.name = NULL,
  question = NULL,
  outdir = NULL,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  save.mod = FALSE,
  .gs = FALSE,
  verbose = TRUE,
  trace = 1
```

```
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in `x`
y.test	Numeric vector of testing set outcome
weights	Numeric vector: Weights for cases. For classification, `weights` takes precedence over `ifw`, therefore set `weights = NULL` if using `ifw`. Note: If `weight` are provided, `ifw` is not used. Leave NULL if setting `ifw = TRUE`.
ifw	Logical: If TRUE, apply inverse frequency weighting (for Classification only). Note: If `weights` are provided, `ifw` is not used.
ifw.type	Integer 0, 1, 2 1: class.weights as in 0, divided by min(class.weights) 2: class.weights as in 0, divided by max(class.weights)
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
max.leaves	Integer: Maximum number of terminal nodes to grow
lookback	Logical: If TRUE, check validation error to decide when to stop growing tree. Default = FALSE
minobsinnode	Integer: Minimum N observations needed in node, before considering splitting
lambda	Float: lambda parameter for `MASS::lm.ridge` Default = .01
nvmax	[gS] Integer: Number of max features to use for lin.type "allSubsets", "forwardStepwise", or "backwardStepwise". If values greater than n of features in `x` are provided, they will be excluded
init	Initial value. Default = `mean(y)`
plot.tuning	Logical: If TRUE, plot validation error during gridsearch
x.name	Character: Name for feature set
y.name	Character: Name for outcome
question	Character: the question you are attempting to answer with this model, in plain language.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if `save.mod` is TRUE
print.plot	Logical: if TRUE, produce plot using `mplot3` Takes precedence over `plot.fitted` and `plot.predicted`.
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted

plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
.gs	internal use only
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If higher than 0, will print more information to the console.

Details

The Linear Optimized Additive Tree grows a tree by finding splits that minimize loss after linear models are fit on each child. We specify an upper threshold of leaves using max.leaves instead of directly defining a number, because depending on the other parameters and the datasets, splitting may stop early.

Author(s)

E.D. Gennatas

---

s_LM *Linear model*

Description

Fit a linear model and validate it. Options include base lm(), robust linear model using MASS:rlm(), generalized least squares using nlme::gls, or polynomial regression using stats::poly to transform features

Usage

```
s_LM(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  weights = NULL,
  ifw = TRUE,
  ifw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  intercept = TRUE,
  robust = FALSE,
```

```
  gls = FALSE,
  polynomial = FALSE,
  poly.d = 3,
  poly.raw = FALSE,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  na.action = na.exclude,
  question = NULL,
  verbose = TRUE,
  trace = 0,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

`x`	Numeric vector or matrix / data frame of features i.e. independent variables
`y`	Numeric vector of outcome, i.e. dependent variable
`x.test`	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in `x`
`y.test`	Numeric vector of testing set outcome
`x.name`	Character: Name for feature set
`y.name`	Character: Name for outcome
`weights`	Numeric vector: Weights for cases. For classification, `weights` takes precedence over `ifw`, therefore set `weights = NULL` if using `ifw`. Note: If `weight` are provided, `ifw` is not used. Leave NULL if setting `ifw = TRUE`.
`ifw`	Logical: If TRUE, apply inverse frequency weighting (for Classification only). Note: If `weights` are provided, `ifw` is not used.
`ifw.type`	Integer 0, 1, 2 1: class.weights as in 0, divided by min(class.weights) 2: class.weights as in 0, divided by max(class.weights)
`upsample`	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
`downsample`	Logical: If TRUE, downsample majority class to match size of minority class
`resample.seed`	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
`intercept`	Logical: If TRUE, fit an intercept term.
`robust`	Logical: if TRUE, use `MASS::rlm()` instead of base `lm()`
`gls`	Logical: if TRUE, use `nlme::gls`
`polynomial`	Logical: if TRUE, run lm on `poly(x, poly.d)` (creates orthogonal polynomials)

poly.d	Integer: degree of polynomial
poly.raw	Logical: if TRUE, use raw polynomials. Default, which should not really be changed is FALSE
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted.
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
na.action	How to handle missing values. See ?na.fail
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If higher than 0, will print more information to the console.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical. If TRUE, save all output as RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments to be passed to MASS::rlm if robust = TRUE or MASS::lm.gls if gls = TRUE

Details

GLS can be useful in place of a standard linear model, when there is correlation among the residuals and/or they have unequal variances. Warning: nlme's implementation is buggy, and predict will not work because of environment problems, which means it fails to get predicted values if x.test is provided. robut = TRUE trains a robust linear model using MASS::rlm. gls = TRUE trains a generalized least squares model using nlme::gls.

Value

rtMod

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: s_AdaBoost(), s_AddTree(), s_BART(), s_BRUTO(), s_BayesGLM(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GAM(), s_GBM(), s_GLM(), s_GLMNET(), s_GLMTree(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_HAL(), s_KNN(), s_LDA(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MARS(), s_MLRF(), s_NBayes(), s_NLA(), s_NLS(), s_NW(), s_PPR(), s_PolyMARS(), s_QDA(), s_QRNN(), s_RF(), s_RFSRC(), s_Ranger(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBoost(), s_XRF()

Examples

```
x <- rnorm(100)
y <- .6 * x + 12 + rnorm(100) / 2
mod <- s_LM(x, y)
```

---

s_LMTree *Linear Model Tree [R]*

Description

Train a LMTree for regression or classification using `partykit::lmtree`

Usage

```
s_LMTree(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  weights = NULL,
  offset = NULL,
  ifw = TRUE,
  ifw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  na.action = na.exclude,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

`x`	Numeric vector or matrix / data frame of features i.e. independent variables
`y`	Numeric vector of outcome, i.e. dependent variable
`x.test`	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in `x`

`y.test`	Numeric vector of testing set outcome
`x.name`	Character: Name for feature set
`y.name`	Character: Name for outcome
`weights`	Numeric vector: Weights for cases. For classification, `weights` takes precedence over `ifw`, therefore set `weights = NULL` if using `ifw`. Note: If `weight` are provided, `ifw` is not used. Leave NULL if setting `ifw = TRUE`.
`offset`	Numeric vector of a priori known offsets
`ifw`	Logical: If TRUE, apply inverse frequency weighting (for Classification only). Note: If `weights` are provided, `ifw` is not used.
`ifw.type`	Integer 0, 1, 2 1: class.weights as in 0, divided by min(class.weights) 2: class.weights as in 0, divided by max(class.weights)
`upsample`	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
`downsample`	Logical: If TRUE, downsample majority class to match size of minority class
`resample.seed`	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
`na.action`	Character: How to handle missing values. See `?model.frame`
`print.plot`	Logical: if TRUE, produce plot using `mplot3` Takes precedence over `plot.fitted` and `plot.predicted`.
`plot.fitted`	Logical: if TRUE, plot True (y) vs Fitted
`plot.predicted`	Logical: if TRUE, plot True (y.test) vs Predicted. Requires `x.test` and `y.test`
`plot.theme`	Character: "zero", "dark", "box", "darkbox"
`question`	Character: the question you are attempting to answer with this model, in plain language.
`verbose`	Logical: If TRUE, print summary to screen.
`outdir`	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if `save.mod` is TRUE
`save.mod`	Logical: If TRUE, save all output to an RDS file in `outdir` `save.mod` is TRUE by default if an `outdir` is defined. If set to TRUE, and no `outdir` is defined, outdir defaults to `paste0("./s.", mod.name)`
`...`	Additional arguments passed to `partykit::mob_control`

Value

Object of class `rtMod`

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: s_AdaBoost(), s_AddTree(), s_BART(), s_BRUTO(), s_BayesGLM(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GAM(), s_GBM(), s_GLM(), s_GLMNET(), s_GLMTree(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_HAL(), s_KNN(), s_LDA(), s_LM(), s_LightCART(), s_LightGBM(), s_MARS(), s_MLRF(), s_NBayes(), s_NLA(), s_NLS(), s_NW(), s_PPR(), s_PolyMARS(), s_QDA(), s_QRNN(), s_RF(), s_RFSRC(), s_Ranger(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBoost(), s_XRF()

Other Tree-based methods: s_AdaBoost(), s_AddTree(), s_BART(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GBM(), s_GLMTree(), s_H2OGBM(), s_H2ORF(), s_LightCART(), s_LightGBM(), s_MLRF(), s_RF(), s_RFSRC(), s_Ranger(), s_XGBoost(), s_XRF()

Other Interpretable models: s_AddTree(), s_C50(), s_CART(), s_GLM(), s_GLMNET(), s_GLMTree()

---

s_LOESS *Local Polynomial Regression (LOESS) [R]*

Description

Fits a LOESS curve or surface

Usage

```
s_LOESS(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  trace = 0,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable

x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted.
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If higher than 0, will print more information to the console.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments to loess

Details

A maximum of 4 features are allowed in this implementation (stats::loess) The main use for this algorithm would be fitting curves in bivariate plots, where GAM or similar is preferable anyway. It is included in **rtemis** mainly for academic purposes - not for building predictive models.

Value

Object of class **rtemis**

Author(s)

E.D. Gennatas

See Also

train_cv

s_LOGISTIC — *Logistic Regression*

Description

Convenience alias for `s_GLM(family = binomial(link = "logit"))`.

Usage

```
s_LOGISTIC(
  x,
  y,
  x.test = NULL,
  y.test = NULL,
  family = binomial(link = "logit"),
  ...
)
```

Arguments

`x`	Numeric vector or matrix / data frame of features i.e. independent variables
`y`	Numeric vector of outcome, i.e. dependent variable
`x.test`	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in `x`
`y.test`	Numeric vector of testing set outcome
`family`	Error distribution and link function. See `stats::family`
`...`	Additional arguments

s_MARS — *Multivariate adaptive regression splines (MARS) (C, R)*

Description

Trains a MARS model using `earth::earth`. [gS] in Arguments description indicates that hyperparameter will be tuned if more than one value are provided For more info on algorithm hyperparameters, see `?earth::earth`

Usage

```
s_MARS(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  grid.resample.params = setup.grid.resample(),
  weights = NULL,
  ifw = TRUE,
  ifw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  glm = NULL,
  degree = 2,
  penalty = 3,
  nk = NULL,
  thresh = 0,
  minspan = 0,
  endspan = 0,
  newvar.penalty = 0,
  fast.k = 2,
  fast.beta = 1,
  linpreds = FALSE,
  pmethod = "forward",
  nprune = NULL,
  nfold = 4,
  ncross = 1,
  stratify = TRUE,
  wp = NULL,
  na.action = na.fail,
  metric = NULL,
  maximize = NULL,
  n.cores = rtCores,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  trace = 0,
  save.mod = FALSE,
  outdir = NULL,
  ...
)
```

Arguments

x	Numeric vector or matrix of features, i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	(Optional) Numeric vector or matrix of validation set features must have set of columns as x
y.test	(Optional) Numeric vector of validation set outcomes
x.name	Character: Name for feature set
y.name	Character: Name for outcome
grid.resample.params	List: Output of setup.resample defining grid search parameters.
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ifw, therefore set weights = NULL if using ifw. Note: If weight are provided, ifw is not used. Leave NULL if setting ifw = TRUE.
ifw	Logical: If TRUE, apply inverse frequency weighting (for Classification only). Note: If weights are provided, ifw is not used.
ifw.type	Integer 0, 1, 2 1: class.weights as in 0, divided by min(class.weights) 2: class.weights as in 0, divided by max(class.weights)
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
glm	List of parameters to pass to glm
degree	[gS] Integer: Maximum degree of interaction. Default = 2
penalty	[gS] Float: GCV penalty per knot. 0 penalizes only terms, not knots. -1 means no penalty. Default = 3
nk	[gS] Integer: Maximum number of terms created by the forward pass. See earth::earth
thresh	[gS] Numeric: Forward stepping threshold. Forward pass terminates if RSq reduction is less than this.
minspan	Numeric: Minimum span of the basis functions. Default = 0
pmethod	[gS] Character: Pruning method: "backward", "none", "exhaustive", "forward", "seqrep", "cv". Default = "forward"
nprune	[gS] Integer: Max N of terms (incl. intercept) in the pruned model
na.action	How to handle missing values. See ?na.fail
metric	Character: Metric to minimize, or maximize if maximize = TRUE during grid search. Default = NULL, which results in "Balanced Accuracy" for Classification, "MSE" for Regression, and "Coherence" for Survival Analysis.
maximize	Logical: If TRUE, metric will be maximized if grid search is run.

n.cores	Integer: Number of cores to use.
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted.
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If higher than 0, will print more information to the console.
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
...	Additional parameters to pass to earth::earth

Value

Object of class rtMod

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: s_AdaBoost(), s_AddTree(), s_BART(), s_BRUTO(), s_BayesGLM(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GAM(), s_GBM(), s_GLM(), s_GLMNET(), s_GLMTree(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_HAL(), s_KNN(), s_LDA(), s_LM(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MLRF(), s_NBayes(), s_NLA(), s_NLS(), s_NW(), s_PPR(), s_PolyMARS(), s_QDA(), s_QRNN(), s_RF(), s_RFSRC(), s_Ranger(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBoost(), s_XRF()

---

s_MLRF *Spark MLlib Random Forest (C, R)*

Description

Train an MLlib Random Forest model on Spark

Usage

```
s_MLRF(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  n.trees = 500L,
  max.depth = 30L,
  subsampling.rate = 1,
  min.instances.per.node = 1,
  feature.subset.strategy = "auto",
  max.bins = 32L,
  x.name = NULL,
  y.name = NULL,
  spark.master = "local",
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  trace = 0,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	vector, matrix or dataframe of training set features
y	vector of outcomes
x.test	vector, matrix or dataframe of testing set features
y.test	vector of testing set outcomes
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
n.trees	Integer: Number of trees to train
max.depth	Integer: Max depth of each tree

`subsampling.rate`
: Numeric: Fraction of cases to use for training each tree

`min.instances.per.node`
: Integer: Min N of cases per node.

`feature.subset.strategy`
: Character: The number of features to consider for splits at each tree node. Supported options: "auto" (choose automatically for task: If numTrees == 1, set to "all." If numTrees > 1 (forest), set to "sqrt" for classification and to "onethird" for regression), "all" (use all features), "onethird" (use 1/3 of the features), "sqrt" (use sqrt(number of features)), "log2" (use log2(number of features)), "n": (when n is in the range (0, 1.0], use n * number of features. When n is in the range (1, number of features), use n features). Default is "auto".

`max.bins`
: Integer. Max N of bins used for discretizing continuous features and for choosing how to split on features at each node. More bins give higher granularity.

`x.name`
: Character: Name for feature set

`y.name`
: Character: Name for outcome

`spark.master`
: Spark cluster URL or "local"

`print.plot`
: Logical: if TRUE, produce plot using `mplot3` Takes precedence over `plot.fitted` and `plot.predicted`.

`plot.fitted`
: Logical: if TRUE, plot True (y) vs Fitted

`plot.predicted`
: Logical: if TRUE, plot True (y.test) vs Predicted. Requires `x.test` and `y.test`

`plot.theme`
: Character: "zero", "dark", "box", "darkbox"

`question`
: Character: the question you are attempting to answer with this model, in plain language.

`verbose`
: Logical: If TRUE, print summary to screen.

`trace`
: Integer: If higher than 0, will print more information to the console.

`outdir`
: Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if `save.mod` is TRUE

`save.mod`
: Logical: If TRUE, save all output to an RDS file in `outdir` `save.mod` is TRUE by default if an `outdir` is defined. If set to TRUE, and no `outdir` is defined, outdir defaults to `paste0("./s.", mod.name)`

`...`
: Additional arguments

Details

The overhead incurred by Spark means this is best used for larged datasets on a Spark cluster.

See also: Spark MLLib documentation

Value

`rtMod` object

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: s_AdaBoost(), s_AddTree(), s_BART(), s_BRUTO(), s_BayesGLM(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GAM(), s_GBM(), s_GLM(), s_GLMNET(), s_GLMTree(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_HAL(), s_KNN(), s_LDA(), s_LM(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MARS(), s_NBayes(), s_NLA(), s_NLS(), s_NW(), s_PPR(), s_PolyMARS(), s_QDA(), s_QRNN(), s_RF(), s_RFSRC(), s_Ranger(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBoost(), s_XRF()

Other Tree-based methods: s_AdaBoost(), s_AddTree(), s_BART(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GBM(), s_GLMTree(), s_H2OGBM(), s_H2ORF(), s_LMTree(), s_LightCART(), s_LightGBM(), s_RF(), s_RFSRC(), s_Ranger(), s_XGBoost(), s_XRF()

---

s_MULTINOM *Multinomial Logistic Regression*

Description

Convenience alias for `s_GLM(class.method = "multinom")`.

Usage

```
s_MULTINOM(x, y, x.test = NULL, y.test = NULL, class.method = "multinom", ...)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
class.method	Character: Define "logistic" or "multinom" for classification. The only purpose of this is so you can try nnet::multinom instead of glm for binary classification
...	Additional arguments

s_NBayes *Naive Bayes Classifier C*

Description

Train a Naive Bayes Classifier using e1071::naiveBayes

Usage

```
s_NBayes(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  laplace = 0,
  x.name = NULL,
  y.name = NULL,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
laplace	Float (>0): Laplace smoothing. Default = 0 (no smoothing). This only affects categorical features
x.name	Character: Name for feature set
y.name	Character: Name for outcome
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted.
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"

question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if `save.mod` is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in `outdir` `save.mod` is TRUE by default if an `outdir` is defined. If set to TRUE, and no `outdir` is defined, outdir defaults to `paste0("./s.", mod.name)`
...	Additional arguments

Details

The `laplace` argument only affects categorical predictors

Value

`rtMod` object

Author(s)

E.D. Gennatas

See Also

Other Supervised Learning: `s_AdaBoost()`, `s_AddTree()`, `s_BART()`, `s_BRUTO()`, `s_BayesGLM()`, `s_C50()`, `s_CART()`, `s_CTree()`, `s_EVTree()`, `s_GAM()`, `s_GBM()`, `s_GLM()`, `s_GLMNET()`, `s_GLMTree()`, `s_GLS()`, `s_H2ODL()`, `s_H2OGBM()`, `s_H2ORF()`, `s_HAL()`, `s_KNN()`, `s_LDA()`, `s_LM()`, `s_LMTree()`, `s_LightCART()`, `s_LightGBM()`, `s_MARS()`, `s_MLRF()`, `s_NLA()`, `s_NLS()`, `s_NW()`, `s_PPR()`, `s_PolyMARS()`, `s_QDA()`, `s_QRNN()`, `s_RF()`, `s_RFSRC()`, `s_Ranger()`, `s_SDA()`, `s_SGD()`, `s_SPLS()`, `s_SVM()`, `s_TFN()`, `s_XGBoost()`, `s_XRF()`

`s_NLA` *NonLinear Activation unit Regression (NLA) [R]*

Description

Train an equivalent of a 1 hidden unit neural network with a defined nonlinear activation function using `optim`

Usage

```
s_NLA(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  activation = softplus,
```

```
  b_o = mean(y),
  W_o = 1,
  b_h = 0,
  W_h = 0.01,
  optim.method = "BFGS",
  control = list(),
  x.name = NULL,
  y.name = NULL,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  trace = 0,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
activation	Function: Activation function to use. Default = softplus
b_o	Float, vector (length y): Output bias. Defaults to mean(y)
W_o	Float: Output weight. Defaults to 1
b_h	Float: Hidden layer bias. Defaults to 0
W_h	Float, vector (length NCOL(x)): Hidden layer weights. Defaults to 0
optim.method	Character: Optimization method to use: "Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN", "Brent". See stats::optim for more details. Default = "BFGS"
control	List: Control parameters passed to stats::optim
x.name	Character: Name for feature set
y.name	Character: Name for outcome
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted.
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"

question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If > 0, print model summary.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if `save.mod` is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in `outdir` `save.mod` is TRUE by default if an `outdir` is defined. If set to TRUE, and no `outdir` is defined, outdir defaults to `paste0("./s.", mod.name)`
...	Additional arguments to be passed to `sigreg`

Details

Since we are using `optim`, results will be sensitive to the combination of optimizer method (See `optim::method` for details), initialization values, and activation function.

Value

Object of class **rtemis**

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: `s_AdaBoost()`, `s_AddTree()`, `s_BART()`, `s_BRUTO()`, `s_BayesGLM()`, `s_C50()`, `s_CART()`, `s_CTree()`, `s_EVTree()`, `s_GAM()`, `s_GBM()`, `s_GLM()`, `s_GLMNET()`, `s_GLMTree()`, `s_GLS()`, `s_H2ODL()`, `s_H2OGBM()`, `s_H2ORF()`, `s_HAL()`, `s_KNN()`, `s_LDA()`, `s_LM()`, `s_LMTree()`, `s_LightCART()`, `s_LightGBM()`, `s_MARS()`, `s_MLRF()`, `s_NBayes()`, `s_NLS()`, `s_NW()`, `s_PPR()`, `s_PolyMARS()`, `s_QDA()`, `s_QRNN()`, `s_RF()`, `s_RFSRC()`, `s_Ranger()`, `s_SDA()`, `s_SGD()`, `s_SPLS()`, `s_SVM()`, `s_TFN()`, `s_XGBoost()`, `s_XRF()`

`s_NLS` *Nonlinear Least Squares (NLS) [R]*

Description

Build a NLS model

Usage

```
s_NLS(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  formula = NULL,
  weights = NULL,
  start = NULL,
  control = nls.control(maxiter = 200),
  .type = NULL,
  default.start = 0.1,
  algorithm = "default",
  nls.trace = FALSE,
  x.name = NULL,
  y.name = NULL,
  save.func = TRUE,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  verbosity = 0,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

`x`	Numeric vector or matrix / data frame of features i.e. independent variables
`y`	Numeric vector of outcome, i.e. dependent variable
`x.test`	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in `x`
`y.test`	Numeric vector of testing set outcome
`formula`	Formula for the model. If NULL, a model is built with all predictors.
`weights`	Numeric vector: Weights for cases. For classification, `weights` takes precedence over `ifw`, therefore set `weights = NULL` if using `ifw`. Note: If `weight` are provided, `ifw` is not used. Leave NULL if setting `ifw = TRUE`.
`start`	List of starting values for the parameters in the model.
`control`	Control parameters for `nls` created by `nls.control`
`.type`	Type of model to build. If NULL, a linear model is built. If "sig", a sigmoid model is built.
`default.start`	Numeric: Default starting value for all parameters

algorithm	Character: Algorithm to use for nls. See nls for details.
nls.trace	Logical: If TRUE, trace information is printed during the optimization process.
x.name	Character: Name for feature set
y.name	Character: Name for outcome
save.func	Logical: If TRUE, save model as character string
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted.
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
verbosity	Integer: If > 0, print model summary
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments to be passed to nls

Value

Object of class **rtemis**

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: s_AdaBoost(), s_AddTree(), s_BART(), s_BRUTO(), s_BayesGLM(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GAM(), s_GBM(), s_GLM(), s_GLMNET(), s_GLMTree(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_HAL(), s_KNN(), s_LDA(), s_LM(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MARS(), s_MLRF(), s_NBayes(), s_NLA(), s_NW(), s_PPR(), s_PolyMARS(), s_QDA(), s_QRNN(), s_RF(), s_RFSRC(), s_Ranger(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBoost(), s_XRF()

s_NW *Nadaraya-Watson kernel regression [R]*

Description

Computes a kernel regression estimate using np::npreg()

Usage

```
s_NW(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  bw = NULL,
  plot.bw = FALSE,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  trace = 0,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
bw	Bandwidth as calculate by np::npregbw. Default = NULL, in which case np::npregbw will be run
plot.bw	Logical. Plot bandwidth selector results
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted.

plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If higher than 0, will print more information to the console.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional parameters to be passed to npreg

Details

np::npreg allows inputs with mixed data types. NW automatically models interactions, like PPR, but the latter is a lot faster

Value

Object of class **rtemis**

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: s_AdaBoost(), s_AddTree(), s_BART(), s_BRUTO(), s_BayesGLM(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GAM(), s_GBM(), s_GLM(), s_GLMNET(), s_GLMTree(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_HAL(), s_KNN(), s_LDA(), s_LM(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MARS(), s_MLRF(), s_NBayes(), s_NLA(), s_NLS(), s_PPR(), s_PolyMARS(), s_QDA(), s_QRNN(), s_RF(), s_RFSRC(), s_Ranger(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBoost(), s_XRF()

Examples

```
## Not run:
x <- rnorm(100)
y <- .6 * x + 12 + rnorm(100)
mod <- s_NW(x, y)

## End(Not run)
```

s_POLY *Polynomial Regression*

Description

Convenience alias for `s_GLM(polynomial = T)`. Substitutes all features with `poly(x, poly.d)`

Usage

```
s_POLY(x, y, x.test = NULL, y.test = NULL, poly.d = 3, poly.raw = FALSE, ...)
```

Arguments

`x`	Numeric vector or matrix / data frame of features i.e. independent variables
`y`	Numeric vector of outcome, i.e. dependent variable
`x.test`	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in `x`
`y.test`	Numeric vector of testing set outcome
`poly.d`	Integer: degree of polynomial(s) to use
`poly.raw`	Logical: if TRUE, use raw polynomials. Defaults to FALSE, resulting in orthogonal polynomials. See `stats::poly`
`...`	Additional arguments

s_PolyMARS *Multivariate adaptive polynomial spline regression (POLYMARS) (C, R)*

Description

Trains a POLYMARS model using `polspline::polymars` and validates it

Usage

```
s_PolyMARS(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  grid.resample.params = setup.grid.resample(),
  weights = NULL,
  ifw = TRUE,
```

```
  ifw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  maxsize = ceiling(min(6 * (nrow(x)^{
     1/3
 }), nrow(x)/4, 100)),
  n.cores = rtCores,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  trace = 0,
  save.mod = FALSE,
  outdir = NULL,
  ...
)
```

Arguments

- `x` Numeric vector or matrix of features, i.e. independent variables
- `y` Numeric vector of outcome, i.e. dependent variable
- `x.test` (Optional) Numeric vector or matrix of validation set features must have set of columns as `x`
- `y.test` (Optional) Numeric vector of validation set outcomes
- `x.name` Character: Name for feature set
- `y.name` Character: Name for outcome
- `grid.resample.params` List: Output of setup.resample defining grid search parameters.
- `weights` Numeric vector: Weights for cases. For classification, `weights` takes precedence over `ifw`, therefore set `weights = NULL` if using `ifw`. Note: If `weight` are provided, `ifw` is not used. Leave NULL if setting `ifw = TRUE`.
- `ifw` Logical: If TRUE, apply inverse frequency weighting (for Classification only). Note: If `weights` are provided, `ifw` is not used.
- `ifw.type` Integer 0, 1, 2 1: class.weights as in 0, divided by min(class.weights) 2: class.weights as in 0, divided by max(class.weights)
- `upsample` Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
- `downsample` Logical: If TRUE, downsample majority class to match size of minority class
- `resample.seed` Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)

maxsize	Integer: Maximum number of basis functions to use
n.cores	Integer: Number of cores to use.
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted.
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If > 0, print summary of model
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
...	Additional parameters to pass to polspline::polymars

Value

Object of class rtMod

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: s_AdaBoost(), s_AddTree(), s_BART(), s_BRUTO(), s_BayesGLM(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GAM(), s_GBM(), s_GLM(), s_GLMNET(), s_GLMTree(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_HAL(), s_KNN(), s_LDA(), s_LM(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MARS(), s_MLRF(), s_NBayes(), s_NLA(), s_NLS(), s_NW(), s_PPR(), s_QDA(), s_QRNN(), s_RF(), s_RFSRC(), s_Ranger(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBoost(), s_XRF()

s_PPR *Projection Pursuit Regression (PPR) [R]*

Description

Train a Projection Pursuit Regression model

Usage

```
s_PPR(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  grid.resample.params = setup.grid.resample(),
  gridsearch.type = c("exhaustive", "randomized"),
  gridsearch.randomized.p = 0.1,
  weights = NULL,
  nterms = NULL,
  max.terms = nterms,
  optlevel = 3,
  sm.method = "spline",
  bass = 0,
  span = 0,
  df = 5,
  gcvpen = 1,
  metric = "MSE",
  maximize = FALSE,
  n.cores = rtCores,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  trace = 0,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable

`x.test`	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in `x`
`y.test`	Numeric vector of testing set outcome
`x.name`	Character: Name for feature set
`y.name`	Character: Name for outcome
`grid.resample.params`	List: Output of setup.resample defining grid search parameters.
`gridsearch.type`	Character: Type of grid search to perform: "exhaustive" or "randomized".
`gridsearch.randomized.p`	Float (0, 1): If `gridsearch.type = "randomized"`, randomly test this proportion of combinations.
`weights`	Numeric vector: Weights for cases. For classification, `weights` takes precedence over `ifw`, therefore set `weights = NULL` if using `ifw`. Note: If `weight` are provided, `ifw` is not used. Leave NULL if setting `ifw = TRUE`.
`nterms`	[gS] Integer: number of terms to include in the final model
`max.terms`	Integer: maximum number of terms to consider in the model
`optlevel`	[gS] Integer [0, 3]: optimization level (Default = 3). See Details in `stats::ppr`
`sm.method`	[gS] Character: "supsmu", "spline", or "gcvspline". Smoothing method. Default = "spline"
`bass`	[gS] Numeric [0, 10]: for `sm.method = "supsmu"`: larger values result in greater smoother. See stats::ppr
`span`	[gS] Numeric [0, 1]: for `sm.method = "supsmu"`: 0 (Default) results in automatic span selection by local crossvalidation. See stats::ppr
`df`	[gS] Numeric: for `sm.method = "spline"`: Specify smoothness of each ridge term. See stats::ppr
`gcvpen`	[gs] Numeric: for `sm.method = "gcvspline"`: Penalty used in the GCV selection for each degree of freedom used. Higher values result in greater smoothing. See stats::ppr.
`metric`	Character: Metric to minimize, or maximize if `maximize = TRUE` during grid search. Default = NULL, which results in "Balanced Accuracy" for Classification, "MSE" for Regression, and "Coherence" for Survival Analysis.
`maximize`	Logical: If TRUE, `metric` will be maximized if grid search is run.
`n.cores`	Integer: Number of cores to use.
`print.plot`	Logical: if TRUE, produce plot using `mplot3` Takes precedence over `plot.fitted` and `plot.predicted`.
`plot.fitted`	Logical: if TRUE, plot True (y) vs Fitted
`plot.predicted`	Logical: if TRUE, plot True (y.test) vs Predicted. Requires `x.test` and `y.test`
`plot.theme`	Character: "zero", "dark", "box", "darkbox"
`question`	Character: the question you are attempting to answer with this model, in plain language.

verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If greater than 0, print additional information to console
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if `save.mod` is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in `outdir` `save.mod` is TRUE by default if an `outdir` is defined. If set to TRUE, and no `outdir` is defined, outdir defaults to `paste0("./s.", mod.name)`
...	Additional arguments to be passed to `ppr`

Details

[gS]: If more than one value is passed, parameter tuning via grid search will be performed on resamples of the training set prior to training model on full training set Interactions: PPR automatically models interactions, no need to specify them

Value

Object of class `rtMod`

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: `s_AdaBoost()`, `s_AddTree()`, `s_BART()`, `s_BRUTO()`, `s_BayesGLM()`, `s_C50()`, `s_CART()`, `s_CTree()`, `s_EVTree()`, `s_GAM()`, `s_GBM()`, `s_GLM()`, `s_GLMNET()`, `s_GLMTree()`, `s_GLS()`, `s_H2ODL()`, `s_H2OGBM()`, `s_H2ORF()`, `s_HAL()`, `s_KNN()`, `s_LDA()`, `s_LM()`, `s_LMTree()`, `s_LightCART()`, `s_LightGBM()`, `s_MARS()`, `s_MLRF()`, `s_NBayes()`, `s_NLA()`, `s_NLS()`, `s_NW()`, `s_PolyMARS()`, `s_QDA()`, `s_QRNN()`, `s_RF()`, `s_RFSRC()`, `s_Ranger()`, `s_SDA()`, `s_SGD()`, `s_SPLS()`, `s_SVM()`, `s_TFN()`, `s_XGBoost()`, `s_XRF()`

---

s_PSurv *Parametric Survival Regression [S]*

Description

Fit a parametric survival regression model using `survival::survreg`

Usage

```
s_PSurv(
  x,
  y,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  weights = NULL,
  dist = "weibull",
  control = survival::survreg.control(),
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  trace = 0,
  save.mod = FALSE,
  outdir = NULL,
  ...
)
```

Arguments

x	Numeric vector or matrix of features, i.e. independent variables
y	Object of class "Surv" created using survival::Surv
x.test	(Optional) Numeric vector or matrix of testing set features must have set of columns as x
y.test	(Optional) Object of class "Surv" created using survival::Surv
x.name	Character: Name for feature set
y.name	Character: Name for outcome
weights	Float: Vector of case weights
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted.
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)

outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if `save.mod` is TRUE
...	Additional parameters to pass to `survival::survreg`

Value

Object of class `rtMod`

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

`s_QDA` *Quadratic Discriminant Analysis C*

Description

Train a QDA Classifier using `MASS::qda`

Usage

```
s_QDA(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  prior = NULL,
  method = "moment",
  nu = NULL,
  x.name = NULL,
  y.name = NULL,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
prior	Numeric, vector (length = N classes of outcome variable): Prior probabilities
method	Character: "moment", "mle", "mve", or "t". See MASS::qda
nu	Integer: Degrees of freedom for methdo "t"
x.name	Character: Name for feature set
y.name	Character: Name for outcome
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted.
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments

Value

rtMod object

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: s_AdaBoost(), s_AddTree(), s_BART(), s_BRUTO(), s_BayesGLM(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GAM(), s_GBM(), s_GLM(), s_GLMNET(), s_GLMTree(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_HAL(), s_KNN(), s_LDA(), s_LM(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MARS(), s_MLRF(), s_NBayes(), s_NLA(), s_NLS(), s_NW(), s_PPR(), s_PolyMARS(), s_QRNN(), s_RF(), s_RFSRC(), s_Ranger(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBoost(), s_XRF()

---

s_QRNN — *Quantile Regression Neural Network [R]*

Description

Train an ensemble of Neural Networks to perform Quantile Regression using qrnn

Usage

```
s_QRNN(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  n.hidden = 1,
  tau = 0.5,
  n.ensemble = 5,
  iter.max = 5000,
  n.trials = 5,
  bag = TRUE,
  lower = -Inf,
  eps.seq = 2^(-8:-32),
  Th = qrnn::sigmoid,
  Th.prime = qrnn::sigmoid.prime,
  penalty = 0,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
n.hidden	Integer: Number of hidden nodes.
tau	Numeric: tau-quantile.
n.ensemble	Integer: Number of NNs to train.
iter.max	Integer: Max N of iteration of the optimization algorithm.
n.trials	Integer: N of trials. Used to avoid local minima.
bag	Logical: If TRUE, use bagging.
lower	Numeric: Left censoring point.
eps.seq	Numeric: sequence of eps values for the finite smoothing algorithm.
Th	Function: hidden layer transfer function; use qrnn::sigmoid, qrnn::elu, or qrnn::softplus for a nonlinear model and qrnn::linear for a linear model.
Th.prime	Function: derivative of hidden layer transfer function.
penalty	Numeric: weight penalty for weight decay regularization.
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted.
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments to be passed to qrnn::qrnn.fit.

Details

For more details on hyperparameters, see qrnn::qrnn.fit

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: s_AdaBoost(), s_AddTree(), s_BART(), s_BRUTO(), s_BayesGLM(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GAM(), s_GBM(), s_GLM(), s_GLMNET(), s_GLMTree(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_HAL(), s_KNN(), s_LDA(), s_LM(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MARS(), s_MLRF(), s_NBayes(), s_NLA(), s_NLS(), s_NW(), s_PPR(), s_PolyMARS(), s_QDA(), s_RF(), s_RFSRC(), s_Ranger(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBoost(), s_XRF()

---

s_Ranger *Random Forest Classification and Regression (C, R)*

Description

Train a Random Forest for regression or classification using ranger

Usage

```
s_Ranger(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  n.trees = 1000,
  weights = NULL,
  ifw = TRUE,
  ifw.type = 2,
  ifw.case.weights = TRUE,
  ifw.class.weights = FALSE,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  autotune = FALSE,
  classwt = NULL,
  n.trees.try = 500,
  stepFactor = 2,
  mtry = NULL,
  mtryStart = NULL,
  inbag.resample = NULL,
  stratify.on.y = FALSE,
  grid.resample.params = setup.resample("kfold", 5),
  gridsearch.type = c("exhaustive", "randomized"),
  gridsearch.randomized.p = 0.1,
  metric = NULL,
```

```
  maximize = NULL,
  probability = NULL,
  importance = "impurity",
  local.importance = FALSE,
  replace = TRUE,
  min.node.size = NULL,
  splitrule = NULL,
  strata = NULL,
  sampsize = if (replace) nrow(x) else ceiling(0.632 * nrow(x)),
  tune.do.trace = FALSE,
  imetrics = FALSE,
  n.cores = rtCores,
  print.tune.plot = FALSE,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  grid.verbose = verbose,
  verbose = TRUE,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

`x`	Numeric vector or matrix / data frame of features i.e. independent variables
`y`	Numeric vector of outcome, i.e. dependent variable
`x.test`	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in `x`
`y.test`	Numeric vector of testing set outcome
`x.name`	Character: Name for feature set
`y.name`	Character: Name for outcome
`n.trees`	Integer: Number of trees to grow. Default = 1000
`weights`	Numeric vector: Weights for cases. For classification, `weights` takes precedence over `ifw`, therefore set `weights = NULL` if using `ifw`. Note: If `weight` are provided, `ifw` is not used. Leave NULL if setting `ifw = TRUE`.
`ifw`	Logical: If TRUE, apply inverse frequency weighting (for Classification only). Note: If `weights` are provided, `ifw` is not used.
`ifw.type`	Integer 0, 1, 2 1: class.weights as in 0, divided by min(class.weights) 2: class.weights as in 0, divided by max(class.weights)
`ifw.case.weights`	Logical: If TRUE, define ranger's `case.weights` using IPW. Default = TRUE Note: Cannot use case.weights together with `stratify.on.y` or `inbag.resample`

ifw.class.weights
: Logical: If TRUE, define ranger's `class.weights` using IPW. Default = FALSE

upsample
: Logical: If TRUE, upsample training set cases not belonging in majority outcome group

downsample
: Logical: If TRUE, downsample majority class to match size of minority class

resample.seed
: Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)

autotune
: Logical: If TRUE, use `randomForest::tuneRF` to determine `mtry`

classwt
: Vector, Float: Priors of the classes for `randomForest::tuneRF` if `autotune = TRUE`. For classification only; need not add up to 1

n.trees.try
: Integer: Number of trees to train for tuning, if `autotune = TRUE`

stepFactor
: Float: If `autotune = TRUE`, at each tuning iteration, `mtry` is multiplied or divided by this value. Default = 1.5

mtry
: [gS] Integer: Number of features sampled randomly at each split. Defaults to square root of n of features for classification, and a third of n of features for regression.

mtryStart
: Integer: If `autotune = TRUE`, start at this value for `mtry`

inbag.resample
: List, length `n.tree`: Output of setup.resample to define resamples used for each tree. Default = NULL

stratify.on.y
: Logical: If TRUE, overrides `inbag.resample` to use stratified bootstraps for each tree. This can help improve test set performance in imbalanced datasets. Default = FALSE. Note: Cannot be used with `ifw.case.weights`

grid.resample.params
: List: Output of setup.resample defining grid search parameters.

gridsearch.type
: Character: Type of grid search to perform: "exhaustive" or "randomized".

gridsearch.randomized.p
: Float (0, 1): If `gridsearch.type = "randomized"`, randomly test this proportion of combinations.

metric
: Character: Metric to minimize, or maximize if `maximize = TRUE` during grid search. Default = NULL, which results in "Balanced Accuracy" for Classification, "MSE" for Regression, and "Coherence" for Survival Analysis.

maximize
: Logical: If TRUE, `metric` will be maximized if grid search is run.

probability
: Logical: If TRUE, grow a probability forest. See `ranger::ranger`. Default = FALSE

importance
: Character: "none", "impurity", "impurity_corrected", or "permutation" Default = "impurity"

local.importance
: Logical: If TRUE, return local importance values. Only applicable if `importance` is set to "permutation".

replace
: Logical: If TRUE, sample cases with replacement during training.

min.node.size
: [gS] Integer: Minimum node size

`splitrule`	Character: For classification: "gini" (Default) or "extratrees"; For regression: "variance" (Default), "extratrees" or "maxstat". For survival "logrank" (Default), "extratrees", "C" or "maxstat".
`strata`	Vector, Factor: Will be used for stratified sampling
`sampsize`	Integer: Size of sample to draw. In Classification, if `strata` is defined, this can be a vector of the same length, in which case, corresponding values determine how many cases are drawn from the strata.
`tune.do.trace`	Same as `do.trace` but for tuning, when `autotune = TRUE`
`imetrics`	Logical: If TRUE, calculate interpretability metrics (N of trees and N of nodes) and save under the `extra` field of `rtMod`
`n.cores`	Integer: Number of cores to use.
`print.tune.plot`	Logical: passed to `randomForest::tuneRF`.
`print.plot`	Logical: if TRUE, produce plot using `mplot3` Takes precedence over `plot.fitted` and `plot.predicted`.
`plot.fitted`	Logical: if TRUE, plot True (y) vs Fitted
`plot.predicted`	Logical: if TRUE, plot True (y.test) vs Predicted. Requires `x.test` and `y.test`
`plot.theme`	Character: "zero", "dark", "box", "darkbox"
`question`	Character: the question you are attempting to answer with this model, in plain language.
`grid.verbose`	Logical: Passed to `gridSearchLearn`
`verbose`	Logical: If TRUE, print summary to screen.
`outdir`	String, Optional: Path to directory to save output
`save.mod`	Logical: If TRUE, save all output to an RDS file in `outdir` `save.mod` is TRUE by default if an `outdir` is defined. If set to TRUE, and no `outdir` is defined, outdir defaults to `paste0("./s.", mod.name)`
`...`	Additional arguments to be passed to `ranger::ranger`

Details

You should cconsider, or try, setting mtry to NCOL(x), especially for small number of features. By default mtry is set to NCOL(x) for NCOL(x) <= 20. For imbalanced datasets, setting stratify.on.y = TRUE should improve performance. If `autotune = TRUE`, `randomForest::tuneRF` will be run to determine best `mtry` value. [gS]: indicated parameter will be tuned by grid search if more than one value is passed

See Tech Report comparing balanced (ifw.case.weights = TRUE) and weighted (ifw.class.weights = TRUE) Random Forests.

Value

`rtMod` object

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: `s_AdaBoost()`, `s_AddTree()`, `s_BART()`, `s_BRUTO()`, `s_BayesGLM()`, `s_C50()`, `s_CART()`, `s_CTree()`, `s_EVTree()`, `s_GAM()`, `s_GBM()`, `s_GLM()`, `s_GLMNET()`, `s_GLMTree()`, `s_GLS()`, `s_H2ODL()`, `s_H2OGBM()`, `s_H2ORF()`, `s_HAL()`, `s_KNN()`, `s_LDA()`, `s_LM()`, `s_LMTree()`, `s_LightCART()`, `s_LightGBM()`, `s_MARS()`, `s_MLRF()`, `s_NBayes()`, `s_NLA()`, `s_NLS()`, `s_NW()`, `s_PPR()`, `s_PolyMARS()`, `s_QDA()`, `s_QRNN()`, `s_RF()`, `s_RFSRC()`, `s_SDA()`, `s_SGD()`, `s_SPLS()`, `s_SVM()`, `s_TFN()`, `s_XGBoost()`, `s_XRF()`

Other Tree-based methods: `s_AdaBoost()`, `s_AddTree()`, `s_BART()`, `s_C50()`, `s_CART()`, `s_CTree()`, `s_EVTree()`, `s_GBM()`, `s_GLMTree()`, `s_H2OGBM()`, `s_H2ORF()`, `s_LMTree()`, `s_LightCART()`, `s_LightGBM()`, `s_MLRF()`, `s_RF()`, `s_RFSRC()`, `s_XGBoost()`, `s_XRF()`

Other Ensembles: `s_AdaBoost()`, `s_GBM()`, `s_RF()`

`s_RF` *Random Forest Classification and Regression (C, R)*

Description

Train a Random Forest for regression or classification using `randomForest`

Usage

```
s_RF(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  n.trees = 1000,
  autotune = FALSE,
  n.trees.try = 1000,
  stepFactor = 1.5,
  mtry = NULL,
  nodesize = NULL,
  maxnodes = NULL,
  mtryStart = mtry,
  grid.resample.params = setup.resample("kfold", 5),
  metric = NULL,
  maximize = NULL,
  classwt = NULL,
  ifw = TRUE,
  ifw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
```

```
  resample.seed = NULL,
  importance = TRUE,
  proximity = FALSE,
  replace = TRUE,
  strata = NULL,
  sampsize = if (replace) nrow(x) else ceiling(0.632 * nrow(x)),
  sampsize.ratio = NULL,
  do.trace = NULL,
  tune.do.trace = FALSE,
  imetrics = FALSE,
  n.cores = rtCores,
  print.tune.plot = FALSE,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  proximity.tsne = FALSE,
  discard.forest = FALSE,
  tsne.perplexity = 5,
  plot.tsne.train = FALSE,
  plot.tsne.test = FALSE,
  question = NULL,
  verbose = TRUE,
  grid.verbose = verbose,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

`x`	Numeric vector or matrix / data frame of features i.e. independent variables
`y`	Numeric vector of outcome, i.e. dependent variable
`x.test`	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in `x`
`y.test`	Numeric vector of testing set outcome
`x.name`	Character: Name for feature set
`y.name`	Character: Name for outcome
`n.trees`	Integer: Number of trees to grow. Default = 1000
`autotune`	Logical: If TRUE, use `randomForest::tuneRF` to determine `mtry`
`n.trees.try`	Integer: Number of trees to train for tuning, if `autotune = TRUE`
`stepFactor`	Float: If `autotune = TRUE`, at each tuning iteration, `mtry` is multiplied or divided by this value. Default = 1.5
`mtry`	[gS] Integer: Number of features sampled randomly at each split
`nodesize`	[gS]: Integer: Minimum size of terminal nodes. Default = 5 (Regression); 1 (Classification)

`maxnodes`	[gS]: Integer: Maximum number of terminal nodes in a tree. Default = NULL; trees grown to maximum possible
`mtryStart`	Integer: If `autotune = TRUE`, start at this value for `mtry`
`grid.resample.params`	List: Output of setup.resample defining grid search parameters.
`metric`	Character: Metric to minimize, or maximize if `maximize = TRUE` during grid search. Default = NULL, which results in "Balanced Accuracy" for Classification, "MSE" for Regression, and "Coherence" for Survival Analysis.
`maximize`	Logical: If TRUE, `metric` will be maximized if grid search is run.
`classwt`	Vector, Float: Priors of the classes for classification only. Need not add up to 1
`ifw`	Logical: If TRUE, apply inverse frequency weighting (for Classification only). Note: If `weights` are provided, `ifw` is not used.
`ifw.type`	Integer 0, 1, 2 1: class.weights as in 0, divided by min(class.weights) 2: class.weights as in 0, divided by max(class.weights)
`upsample`	Logical: If TRUE, upsample training set cases not belonging in majority outcome group
`downsample`	Logical: If TRUE, downsample majority class to match size of minority class
`resample.seed`	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
`importance`	Logical: If TRUE, estimate variable relative importance.
`proximity`	Logical: If TRUE, calculate proximity measure among cases.
`replace`	Logical: If TRUE, sample cases with replacement during training.
`strata`	Vector, Factor: Will be used for stratified sampling
`sampsize`	Integer: Size of sample to draw. In Classification, if `strata` is defined, this can be a vector of the same length, in which case, corresponding values determine how many cases are drawn from the strata.
`sampsize.ratio`	Float (0, 1): Heuristic of sorts to increase sensitivity in unbalanced cases. Sample with replacement from minority case to create bootstraps of length N cases. Select `(sampsize.ratio * N minority cases)` cases from majority class.
`do.trace`	Logical or integer: If TRUE, `randomForest` will outpout information while it is running. If an integer, `randomForest` will report progress every this many trees. Default = `n.trees/10` if `verbose = TRUE`
`tune.do.trace`	Same as `do.trace` but for tuning, when `autotune = TRUE`
`imetrics`	Logical: If TRUE, calculate interpretability metrics (N of trees and N of nodes) and save under the `extra` field of `rtMod`
`n.cores`	Integer: Number of cores to use.
`print.tune.plot`	Logical: passed to `randomForest::tuneRF`.
`print.plot`	Logical: if TRUE, produce plot using `mplot3` Takes precedence over `plot.fitted` and `plot.predicted`.
`plot.fitted`	Logical: if TRUE, plot True (y) vs Fitted

plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
proximity.tsne	Logical: If TRUE, perform t-SNE on proximity matrix. Will be saved under 'extra' field of rtMod. Default = FALSE
discard.forest	Logical: If TRUE, remove forest from rtMod object to save space. Default = FALSE
tsne.perplexity	Numeric: Perplexity parameter for Rtsne::Rtsne
plot.tsne.train	Logical: If TRUE, plot training set tSNE projections
plot.tsne.test	Logical: If TRUE, plot testing set tSNE projections
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
grid.verbose	Logical: Passed to gridSearchLearn
outdir	String, Optional: Path to directory to save output
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments to be passed to randomForest::randomForest

Details

If autotue = TRUE, randomForest::tuneRF will be run to determine best mtry value.

Value

rtMod object

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: s_AdaBoost(), s_AddTree(), s_BART(), s_BRUTO(), s_BayesGLM(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GAM(), s_GBM(), s_GLM(), s_GLMNET(), s_GLMTree(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_HAL(), s_KNN(), s_LDA(), s_LM(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MARS(), s_MLRF(), s_NBayes(), s_NLA(), s_NLS(), s_NW(), s_PPR(), s_PolyMARS(), s_QDA(), s_QRNN(), s_RFSRC(), s_Ranger(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBoost(), s_XRF()

Other Tree-based methods: s_AdaBoost(), s_AddTree(), s_BART(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GBM(), s_GLMTree(), s_H2OGBM(), s_H2ORF(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MLRF(), s_RFSRC(), s_Ranger(), s_XGBoost(), s_XRF()

Other Ensembles: s_AdaBoost(), s_GBM(), s_Ranger()

s_RFSRC *Random Forest for Classification, Regression, and Survival [C, R, S]*

Description

Train a Random Forest for Regression, Classification, or Survival Regression using randomForestSRC

Usage

```
s_RFSRC(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  n.trees = 1000,
  weights = NULL,
  ifw = TRUE,
  ifw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  bootstrap = "by.root",
  mtry = NULL,
  importance = TRUE,
  proximity = TRUE,
  nodesize = if (!is.null(y) && !is.factor(y)) 5 else 1,
  nodedepth = NULL,
  na.action = "na.impute",
  trace = FALSE,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix of features, i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable

`x.test`	(Optional) Numeric vector or matrix of validation set features must have set of columns as `x`
`y.test`	(Optional) Numeric vector of validation set outcomes
`x.name`	Character: Name for feature set
`y.name`	Character: Name for outcome
`n.trees`	Integer: Number of trees to grow. The more the merrier.
`weights`	Numeric vector: Weights for cases. For classification, `weights` takes precedence over `ifw`, therefore set `weights = NULL` if using `ifw`. Note: If `weight` are provided, `ifw` is not used. Leave NULL if setting `ifw = TRUE`.
`ifw`	Logical: If TRUE, apply inverse frequency weighting (for Classification only). Note: If `weights` are provided, `ifw` is not used.
`ifw.type`	Integer 0, 1, 2 1: class.weights as in 0, divided by min(class.weights) 2: class.weights as in 0, divided by max(class.weights)
`upsample`	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
`downsample`	Logical: If TRUE, downsample majority class to match size of minority class
`resample.seed`	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
`bootstrap`	Character:
`mtry`	Integer: Number of features sampled randomly at each split
`importance`	Logical: If TRUE, calculate variable importance.
`proximity`	Character or Logical: "inbag", "oob", "all", TRUE, or FALSE; passed to `randomForestSRC::rfsrc`
`nodesize`	Integer: Minimum size of terminal nodes.
`nodedepth`	Integer: Maximum tree depth.
`na.action`	Character: How to handle missing values.
`trace`	Integer: Number of seconds between messages to the console.
`print.plot`	Logical: if TRUE, produce plot using `mplot3` Takes precedence over `plot.fitted` and `plot.predicted`.
`plot.fitted`	Logical: if TRUE, plot True (y) vs Fitted
`plot.predicted`	Logical: if TRUE, plot True (y.test) vs Predicted. Requires `x.test` and `y.test`
`plot.theme`	Character: "zero", "dark", "box", "darkbox"
`question`	Character: the question you are attempting to answer with this model, in plain language.
`verbose`	Logical: If TRUE, print summary to screen.
`outdir`	Optional. Path to directory to save output
`save.mod`	Logical: If TRUE, save all output to an RDS file in `outdir` `save.mod` is TRUE by default if an `outdir` is defined. If set to TRUE, and no `outdir` is defined, outdir defaults to `paste0("./s.", mod.name)`
`...`	Additional arguments to be passed to `randomForestSRC::rfsrc`

Details

For Survival Regression, y must be an object of type `Surv`, created using `survival::Surv(time, status)` `mtry` is the only tunable parameter, but it usually only makes a small difference and is often not tuned.

Value

Object of class `rtMod`

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: `s_AdaBoost()`, `s_AddTree()`, `s_BART()`, `s_BRUTO()`, `s_BayesGLM()`, `s_C50()`, `s_CART()`, `s_CTree()`, `s_EVTree()`, `s_GAM()`, `s_GBM()`, `s_GLM()`, `s_GLMNET()`, `s_GLMTree()`, `s_GLS()`, `s_H2ODL()`, `s_H2OGBM()`, `s_H2ORF()`, `s_HAL()`, `s_KNN()`, `s_LDA()`, `s_LM()`, `s_LMTree()`, `s_LightCART()`, `s_LightGBM()`, `s_MARS()`, `s_MLRF()`, `s_NBayes()`, `s_NLA()`, `s_NLS()`, `s_NW()`, `s_PPR()`, `s_PolyMARS()`, `s_QDA()`, `s_QRNN()`, `s_RF()`, `s_Ranger()`, `s_SDA()`, `s_SGD()`, `s_SPLS()`, `s_SVM()`, `s_TFN()`, `s_XGBoost()`, `s_XRF()`

Other Tree-based methods: `s_AdaBoost()`, `s_AddTree()`, `s_BART()`, `s_C50()`, `s_CART()`, `s_CTree()`, `s_EVTree()`, `s_GBM()`, `s_GLMTree()`, `s_H2OGBM()`, `s_H2ORF()`, `s_LMTree()`, `s_LightCART()`, `s_LightGBM()`, `s_MLRF()`, `s_RF()`, `s_Ranger()`, `s_XGBoost()`, `s_XRF()`

`s_RLM` *Robust linear model*

Description

Convenience alias for `s_LM(robust = T)`. Uses `MASS::rlm`

Usage

```
s_RLM(x, y, x.test = NULL, y.test = NULL, ...)
```

Arguments

`x`	Numeric vector or matrix / data frame of features i.e. independent variables
`y`	Numeric vector of outcome, i.e. dependent variable
`x.test`	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in `x`
`y.test`	Numeric vector of testing set outcome
`...`	Additional parameters to be passed to `MASS::rlm`

s_RuleFit *Rulefit [C, R]*

Description

Train a gradient boosting model, extract rules, and fit using LASSO

Usage

```
s_RuleFit(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  gbm.params = list(list(n.trees = 300, bag.fraction = 1, shrinkage = 0.1,
    interaction.depth = 3, ifw = TRUE)),
  meta.alpha = 1,
  meta.lambda = NULL,
  meta.extra.params = list(ifw = TRUE),
  cases.by.rules = NULL,
  x.name = NULL,
  y.name = NULL,
  n.cores = rtCores,
  question = NULL,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  outdir = NULL,
  save.mod = if (!is.null(outdir)) TRUE else FALSE,
  verbose = TRUE
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
gbm.params	List of named lists: A list, each element of which is a named list of parameters for s_GBM. i.e. If you want to train a single GBM model, this could be: gbm.params = list(list(n.trees = 300, bag.fraction = 1, shrinkage = .1, interaction.depth = 3, ifw = TRUE)) if you wanted to train 2 GBM models, this could be: gbm.params = list(list(n.trees = 300, bag.fraction = 1, shrinkage = .1, interaction.depth = 3, ifw = TRUE), list(n.trees

= 500, bag.fraction = 1, shrinkage = .1, interaction.depth = 3, ifw = TRUE))

meta.alpha	Float [0, 1]: alpha for s_GLMNET
meta.lambda	Float: lambda for s_GLMNET. Default = NULL (will be determined automatically by crossvalidation)
meta.extra.params	Named list: Parameters for s_GLMNET for the feature selection step
cases.by.rules	Matrix of cases by rules from a previoue rulefit run. If provided, the GBM step is skipped. Default = NULL
x.name	Character: Name for feature set
y.name	Character: Name for outcome
n.cores	Integer: Number of cores to use
question	Character: the question you are attempting to answer with this model, in plain language.
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted.
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
outdir	Character: If defined, save log, 'plot.all' plots (see above) and RDS file of complete output
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
verbose	Logical: If TRUE, print summary to screen.

Details

Based on "Predictive Learning via Rule Ensembles" by Friedman and Popescu http://statweb.stanford.edu/~jhf/ftp/RuleFit.pdf

Value

rtMod object

Author(s)

E.D. Gennatas

References

Friedman JH, Popescu BE, "Predictive Learning via Rule Ensembles", http://statweb.stanford.edu/~jhf/ftp/RuleFit.pdf

s_SDA *Sparse Linear Discriminant Analysis*

Description

Train an SDA Classifier using `sparseLDA::sda`

Usage

```
s_SDA(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  lambda = 1e-06,
  stop = NULL,
  maxIte = 100,
  Q = NULL,
  tol = 1e-06,
  .preprocess = setup.preprocess(scale = TRUE, center = TRUE),
  upsample = TRUE,
  downsample = FALSE,
  resample.seed = NULL,
  x.name = NULL,
  y.name = NULL,
  grid.resample.params = setup.resample("kfold", 5),
  gridsearch.type = c("exhaustive", "randomized"),
  gridsearch.randomized.p = 0.1,
  metric = NULL,
  maximize = NULL,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  grid.verbose = verbose,
  trace = 0,
  outdir = NULL,
  n.cores = rtCores,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE)
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable

`x.test`	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in `x`
`y.test`	Numeric vector of testing set outcome
`lambda`	L2-norm weight for elastic net regression
`stop`	If STOP is negative, its absolute value corresponds to the desired number of variables. If STOP is positive, it corresponds to an upper bound on the L1-norm of the b coefficients. There is a one to one correspondence between stop and t. The default is -p (-the number of variables).
`maxIte`	Integer: Maximum number of iterations
`Q`	Integer: Number of components
`tol`	Numeric: Tolerance for change in RSS, which is the stopping criterion
`.preprocess`	List of preprocessing parameters. Scaling and centering is enabled by default, because it is crucial for algorithm to learn.
`upsample`	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
`downsample`	Logical: If TRUE, downsample majority class to match size of minority class
`resample.seed`	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
`x.name`	Character: Name for feature set
`y.name`	Character: Name for outcome
`grid.resample.params`	List: Output of setup.resample defining grid search parameters.
`gridsearch.type`	Character: Type of grid search to perform: "exhaustive" or "randomized".
`gridsearch.randomized.p`	Float (0, 1): If `gridsearch.type = "randomized"`, randomly test this proportion of combinations.
`metric`	Character: Metric to minimize, or maximize if `maximize = TRUE` during grid search. Default = NULL, which results in "Balanced Accuracy" for Classification, "MSE" for Regression, and "Coherence" for Survival Analysis.
`maximize`	Logical: If TRUE, `metric` will be maximized if grid search is run.
`print.plot`	Logical: if TRUE, produce plot using `mplot3` Takes precedence over `plot.fitted` and `plot.predicted`.
`plot.fitted`	Logical: if TRUE, plot True (y) vs Fitted
`plot.predicted`	Logical: if TRUE, plot True (y.test) vs Predicted. Requires `x.test` and `y.test`
`plot.theme`	Character: "zero", "dark", "box", "darkbox"
`question`	Character: the question you are attempting to answer with this model, in plain language.
`verbose`	Logical: If TRUE, print summary to screen.

grid.verbose	Logical: Passed to gridSearchLearn
trace	Integer: passed to sparseLDA::sda
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
n.cores	Integer: Number of cores to use.
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)

Value

rtMod object

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: s_AdaBoost(), s_AddTree(), s_BART(), s_BRUTO(), s_BayesGLM(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GAM(), s_GBM(), s_GLM(), s_GLMNET(), s_GLMTree(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_HAL(), s_KNN(), s_LDA(), s_LM(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MARS(), s_MLRF(), s_NBayes(), s_NLA(), s_NLS(), s_NW(), s_PPR(), s_PolyMARS(), s_QDA(), s_QRNN(), s_RF(), s_RFSRC(), s_Ranger(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBoost(), s_XRF()

Examples

```
## Not run:
datc2 <- iris[51:150, ]
datc2$Species <- factor(datc2$Species)
resc2 <- resample(datc2)
datc2_train <- datc2[resc2$Subsample_1, ]
datc2_test <- datc2[-resc2$Subsample_1, ]
# Without scaling or centering, fails to learn
mod_c2 <- s_SDA(datc2_train, datc2_test, .preprocess = NULL)
# Learns fine with default settings (scaling & centering)
mod_c2 <- s_SDA(datc2_train, datc2_test)

## End(Not run)
```

s_SGD *Stochastic Gradient Descent (SGD) (C, R)*

Description

Train a model by Stochastic Gradient Descent using sgd::sgd

Usage

```
s_SGD(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  model = NULL,
  model.control = list(lambda1 = 0, lambda2 = 0),
  sgd.control = list(method = "ai-sgd"),
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
model	character specifying the model to be used: "lm" (linear model), "glm" (generalized linear model), "cox" (Cox proportional hazards model), "gmm" (generalized method of moments), "m" (M-estimation). See 'Details'.

`model.control` a list of parameters for controlling the model.

- `family` ("glm") a description of the error distribution and link function to be used in the model. This can be a character string naming a family function, a family function or the result of a call to a family function. (See `family` for details of family functions.)
- `rank` ("glm") logical. Should the rank of the design matrix be checked?
- `fn` ("gmm") a function $g(\theta, x)$ which returns a k-vector corresponding to the k moment conditions. It is a required argument if `gr` not specified.
- `gr` ("gmm") a function to return the gradient. If unspecified, a finite-difference approximation will be used.
- `nparams` ("gmm") number of model parameters. This is automatically determined for other models.
- `type` ("gmm") character specifying the generalized method of moments procedure: `"twostep"` (Hansen, 1982), `"iterative"` (Hansen et al., 1996). Defaults to `"iterative"`.
- `wmatrix` ("gmm") weighting matrix to be used in the loss function. Defaults to the identity matrix.
- `loss` ("m") character specifying the loss function to be used in the estimating equation. Default is the Huber loss.
- `lambda1` L1 regularization parameter. Default is 0.
- `lambda2` L2 regularization parameter. Default is 0.

`sgd.control` an optional list of parameters for controlling the estimation.

- `method` character specifying the method to be used: `"sgd"`, `"implicit"`, `"asgd"`, `"ai-sgd"`, `"momentum"`, `"nesterov"`. Default is `"ai-sgd"`. See 'Details'.
- `lr` character specifying the learning rate to be used: `"one-dim"`, `"one-dim-eigen"`, `"d-dim"`, `"adagrad"`, `"rmsprop"`. Default is `"one-dim"`. See 'Details'.
- `lr.control` vector of scalar hyperparameters one can set dependent on the learning rate. For hyperparameters aimed to be left as default, specify `NA` in the corresponding entries. See 'Details'.
- `start` starting values for the parameter estimates. Default is random initialization around zero.
- `size` number of SGD estimates to store for diagnostic purposes (distributed log-uniformly over total number of iterations)
- `reltol` relative convergence tolerance. The algorithm stops if it is unable to change the relative mean squared difference in the parameters by more than the amount. Default is `1e-05`.
- `npasses` the maximum number of passes over the data. Default is 3.
- `pass` logical. Should `tol` be ignored and run the algorithm for all of `npasses`?
- `shuffle` logical. Should the algorithm shuffle the data set including for each pass?
- `verbose` logical. Should the algorithm print progress?

`upsample` Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness

downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted.
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments to be passed to sgd.control

Details

From sgd::sgd: "Models: The Cox model assumes that the survival data is ordered when passed in, i.e., such that the risk set of an observation i is all data points after it."

Value

Object of class **rtemis**

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: s_AdaBoost(), s_AddTree(), s_BART(), s_BRUTO(), s_BayesGLM(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GAM(), s_GBM(), s_GLM(), s_GLMNET(), s_GLMTree(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_HAL(), s_KNN(), s_LDA(), s_LM(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MARS(), s_MLRF(), s_NBayes(), s_NLA(), s_NLS(), s_NW(), s_PPR(), s_PolyMARS(), s_QDA(), s_QRNN(), s_RF(), s_RFSRC(), s_Ranger(), s_SDA(), s_SPLS(), s_SVM(), s_TFN(), s_XGBoost(), s_XRF()

s_SPLS *Sparse Partial Least Squares Regression (C, R)*

Description

Train an SPLS model using `spls::spls` (Regression) and `spls::splsda` (Classification)

Usage

```
s_SPLS(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  upsample = TRUE,
  downsample = FALSE,
  resample.seed = NULL,
  k = 2,
  eta = 0.5,
  kappa = 0.5,
  select = "pls2",
  fit = "simpls",
  scale.x = TRUE,
  scale.y = TRUE,
  maxstep = 100,
  classifier = c("lda", "logistic"),
  grid.resample.params = setup.resample("kfold", 5),
  gridsearch.type = c("exhaustive", "randomized"),
  gridsearch.randomized.p = 0.1,
  metric = NULL,
  maximize = NULL,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  trace = 0,
  grid.verbose = verbose,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  n.cores = rtCores,
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
k	[gS] Integer: Number of components to estimate.
eta	[gS] Float [0, 1): Thresholding parameter.
kappa	[gS] Float [0, .5]: Only relevant for multivariate responses: controls effect of concavity of objective function.
select	[gS] Character: "pls2", "simpls". PLS algorithm for variable selection.
fit	[gS] Character: "kernelpls", "widekernelpls", "simpls", "oscorespls". Algorithm for model fitting.
scale.x	Logical: if TRUE, scale features by dividing each column by its sample standard deviation
scale.y	Logical: if TRUE, scale outcomes by dividing each column by its sample standard deviation
maxstep	[gS] Integer: Maximum number of iteration when fitting direction vectors.
classifier	Character: Classifier used by spls::splsda "lda" or "logistic":
grid.resample.params	List: Output of setup.resample defining grid search parameters.
gridsearch.type	Character: Type of grid search to perform: "exhaustive" or "randomized".
gridsearch.randomized.p	Float (0, 1): If gridsearch.type = "randomized", randomly test this proportion of combinations.
metric	Character: Metric to minimize, or maximize if maximize = TRUE during grid search. Default = NULL, which results in "Balanced Accuracy" for Classification, "MSE" for Regression, and "Coherence" for Survival Analysis.
maximize	Logical: If TRUE, metric will be maximized if grid search is run.
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted.

plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
trace	If > 0 print diagnostic messages
grid.verbose	Logical: Passed to gridSearchLearn
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
n.cores	Integer: Number of cores to be used by gridSearchLearn
...	Additional parameters to be passed to npreg

Details

[gS] denotes argument can be passed as a vector of values, which will trigger a grid search using gridSearchLearn np::npreg allows inputs with mixed data types.

Value

Object of class **rtemis**

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: s_AdaBoost(), s_AddTree(), s_BART(), s_BRUTO(), s_BayesGLM(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GAM(), s_GBM(), s_GLM(), s_GLMNET(), s_GLMTree(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_HAL(), s_KNN(), s_LDA(), s_LM(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MARS(), s_MLRF(), s_NBayes(), s_NLA(), s_NLS(), s_NW(), s_PPR(), s_PolyMARS(), s_QDA(), s_QRNN(), s_RF(), s_RFSRC(), s_Ranger(), s_SDA(), s_SGD(), s_SVM(), s_TFN(), s_XGBoost(), s_XRF()

Examples

```
## Not run:
x <- rnorm(100)
y <- .6 * x + 12 + rnorm(100)
mod <- s_SPLS(x, y)

## End(Not run)
```

s_SVM *Support Vector Machines (C, R)*

Description

Train an SVM learner using `e1071::svm`

Usage

```
s_SVM(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  grid.resample.params = setup.resample("kfold", 5),
  gridsearch.type = c("exhaustive", "randomized"),
  gridsearch.randomized.p = 0.1,
  class.weights = NULL,
  ifw = TRUE,
  ifw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  kernel = "radial",
  degree = 3,
  gamma = NULL,
  coef0 = 0,
  cost = 1,
  probability = TRUE,
  metric = NULL,
  maximize = NULL,
  plot.fitted = NULL,
  plot.predicted = NULL,
  print.plot = FALSE,
  plot.theme = rtTheme,
  n.cores = rtCores,
  question = NULL,
  verbose = TRUE,
  grid.verbose = verbose,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

`x`	Numeric vector or matrix / data frame of features i.e. independent variables
`y`	Numeric vector of outcome, i.e. dependent variable
`x.test`	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in `x`
`y.test`	Numeric vector of testing set outcome
`x.name`	Character: Name for feature set
`y.name`	Character: Name for outcome
`grid.resample.params`	List: Output of setup.resample defining grid search parameters.
`gridsearch.type`	Character: Type of grid search to perform: "exhaustive" or "randomized".
`gridsearch.randomized.p`	Float (0, 1): If `gridsearch.type = "randomized"`, randomly test this proportion of combinations.
`class.weights`	Float, length = n levels of outcome: Weights for each outcome class.For classification, `class.weights` takes precedence over `ifw`, therefore set `class.weights = NULL` if using `ifw`.
`ifw`	Logical: If TRUE, apply inverse frequency weighting (for Classification only). Note: If `weights` are provided, `ifw` is not used.
`ifw.type`	Integer 0, 1, 2 1: class.weights as in 0, divided by min(class.weights) 2: class.weights as in 0, divided by max(class.weights)
`upsample`	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
`downsample`	Logical: If TRUE, downsample majority class to match size of minority class
`resample.seed`	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
`kernel`	Character: "linear", "polynomial", "radial", "sigmoid"
`degree`	[gS] Integer: Degree for `kernel = "polynomial"`.
`gamma`	[gS] Float: Parameter used in all kernels except `linear`
`coef0`	[gS] Float: Parameter used by kernels `polynomial` and `sigmoid`
`cost`	[gS] Float: Cost of constraints violation; the C constant of the regularization term in the Lagrange formulation.
`probability`	Logical: If TRUE, model allows probability estimates
`metric`	Character: Metric to minimize, or maximize if `maximize = TRUE` during grid search. Default = NULL, which results in "Balanced Accuracy" for Classification, "MSE" for Regression, and "Coherence" for Survival Analysis.
`maximize`	Logical: If TRUE, `metric` will be maximized if grid search is run.
`plot.fitted`	Logical: if TRUE, plot True (y) vs Fitted

plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted.
plot.theme	Character: "zero", "dark", "box", "darkbox"
n.cores	Integer: Number of cores to use.
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
grid.verbose	Logical: Passed to gridSearchLearn
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments to be passed to e1071::svm

Details

[gS] denotes parameters that will be tuned by cross-validation if more than one value is passed. Regarding SVM tuning, the following guide from the LIBSVM authors can be useful: http://www.csie.ntu.edu.tw/~cjlin/papers/g
They suggest searching for cost = 2 ^ seq(-5, 15, 2) and gamma = 2 ^ seq(-15, 3, 2)

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: s_AdaBoost(), s_AddTree(), s_BART(), s_BRUTO(), s_BayesGLM(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GAM(), s_GBM(), s_GLM(), s_GLMNET(), s_GLMTree(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_HAL(), s_KNN(), s_LDA(), s_LM(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MARS(), s_MLRF(), s_NBayes(), s_NLA(), s_NLS(), s_NW(), s_PPR(), s_PolyMARS(), s_QDA(), s_QRNN(), s_RF(), s_RFSRC(), s_Ranger(), s_SDA(), s_SGD(), s_SPLS(), s_TFN(), s_XGBoost(), s_XRF()

---

s_TFN *Feedforward Neural Network with* **tensorflow** *(C, R)*

Description

Train an Feedforward Neural Network using **keras** and **tensorflow**

Usage

```
s_TFN(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  class.weights = NULL,
  ifw = TRUE,
  ifw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  net = NULL,
  n.hidden.nodes = NULL,
  initializer = c("glorot_uniform", "glorot_normal", "he_uniform", "he_normal",
    "lecun_uniform", "lecun_normal", "random_uniform", "random_normal",
   "variance_scaling", "truncated_normal", "orthogonal", "zeros", "ones", "constant"),
  initializer.seed = NULL,
  dropout = 0,
 activation = c("relu", "selu", "elu", "sigmoid", "hard_sigmoid", "tanh", "exponential",
    "linear", "softmax", "softplus", "softsign"),
  kernel_l1 = 0.1,
  kernel_l2 = 0,
  activation_l1 = 0,
  activation_l2 = 0,
  batch.normalization = TRUE,
  output = NULL,
  loss = NULL,
 optimizer = c("rmsprop", "adadelta", "adagrad", "adam", "adamax", "nadam", "sgd"),
  learning.rate = NULL,
  metric = NULL,
  epochs = 100,
  batch.size = NULL,
  validation.split = 0.2,
  callback = keras::callback_early_stopping(patience = 150),
  scale = TRUE,
  x.name = NULL,
  y.name = NULL,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x Numeric vector or matrix / data frame of features i.e. independent variables

y Numeric vector of outcome, i.e. dependent variable

x.test Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x

y.test Numeric vector of testing set outcome

class.weights Numeric vector: Class weights for training.

ifw Logical: If TRUE, apply inverse frequency weighting (for Classification only). Note: If weights are provided, ifw is not used.

ifw.type Integer 0, 1, 2 1: class.weights as in 0, divided by min(class.weights) 2: class.weights as in 0, divided by max(class.weights)

upsample Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness

downsample Logical: If TRUE, downsample majority class to match size of minority class

resample.seed Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)

net Pre-defined keras network to be trained (optional)

n.hidden.nodes Integer vector: Length must be equal to the number of hidden layers you wish to create. Can be zero, in which case you get a linear model. Default = N of features, i.e. NCOL(x)

initializer Character: Initializer to use for each layer: "glorot_uniform", "glorot_normal", "he_uniform", "he_normal", "cun_uniform", "lecun_normal", "random_uniform", "random_normal", "variance_scaling", "truncated_normal", "orthogonal", "zeros", "ones", "constant". Glorot is also known as Xavier initialization.

initializer.seed
Integer: Seed to use for each initializer for reproducibility.

dropout Floar, vector, (0, 1): Probability of dropping nodes. Can be a vector of length equal to N of layers, otherwise will be recycled. Default = 0

activation String vector: Activation type to use: "relu", "selu", "elu", "sigmoid", "hard_sigmoid", "tanh", "exponential", "linear", "softmax", "softplus", "softsign". Defaults to "relu" for Classification and "tanh" for Regression

kernel_l1 Float: l1 penalty on weights.

kernel_l2 Float: l2 penalty on weights.

activation_l1 Float: l1 penalty on layer output.

activation_l2 Float: l2 penalty on layer output.

batch.normalization
Logical: If TRUE, batch normalize after each hidden layer.

output Character: Activation to use for output layer. Can be any as in activation. Default = "linear" for Regression, "sigmoid" for binary classification, "softmax" for multiclass

`loss`	Character: Loss to use: Default = "mean_squared_error" for regression, "binary_crossentropy" for binary classification, "sparse_categorical_crossentropy" for multiclass
`optimizer`	Character: Optimization to use: "rmsprop", "adadelta", "adagrad", "adam", "adamax", "nadam", "sgd". Default = "rmsprop"
`learning.rate`	Float: learning rate. Defaults depend on `optimizer` used and are: `rmsprop = .01, adadelta = 1, adagr`
`metric`	Character: Metric used for evaluation during train. Default = "mse" for regression, "accuracy" for classification.
`epochs`	Integer: Number of epochs. Default = 100
`batch.size`	Integer: Batch size. Default = N of cases
`validation.split`	Float (0, 1): proportion of training data to use for validation. Default = .2
`callback`	Function to be called by keras during fitting. Default = `keras::callback_early_stopping(patience = 150)` for early stopping.
`scale`	Logical: If TRUE, scale featues before training. column means and standard deviation will be saved in `rtMod$extra` field to allow scaling ahead of prediction on new data
`x.name`	Character: Name for feature set
`y.name`	Character: Name for outcome
`print.plot`	Logical: if TRUE, produce plot using `mplot3` Takes precedence over `plot.fitted` and `plot.predicted`.
`plot.fitted`	Logical: if TRUE, plot True (y) vs Fitted
`plot.predicted`	Logical: if TRUE, plot True (y.test) vs Predicted. Requires `x.test` and `y.test`
`plot.theme`	Character: "zero", "dark", "box", "darkbox"
`question`	Character: the question you are attempting to answer with this model, in plain language.
`verbose`	Logical: If TRUE, print summary to screen.
`outdir`	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if `save.mod` is TRUE
`save.mod`	Logical: If TRUE, save all output to an RDS file in `outdir` `save.mod` is TRUE by default if an `outdir` is defined. If set to TRUE, and no `outdir` is defined, outdir defaults to `paste0("./s.", mod.name)`
`...`	Additional parameters

Details

For more information on arguments and hyperparameters, see (https://keras.rstudio.com/) and (https://keras.io/) It is important to define network structure and adjust hyperparameters based on your problem. You cannot expect defaults to work on any given dataset.

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: s_AdaBoost(), s_AddTree(), s_BART(), s_BRUTO(), s_BayesGLM(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GAM(), s_GBM(), s_GLM(), s_GLMNET(), s_GLMTree(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_HAL(), s_KNN(), s_LDA(), s_LM(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MARS(), s_MLRF(), s_NBayes(), s_NLA(), s_NLS(), s_NW(), s_PPR(), s_PolyMARS(), s_QDA(), s_QRNN(), s_RF(), s_RFSRC(), s_Ranger(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_XGBoost(), s_XRF()

Other Deep Learning: d_H2OAE(), s_H2ODL()

---

s_TLS *Total Least Squares Regression [R]*

Description

A minimal function to perform total least squares regression

Usage

```
s_TLS(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = "x",
  y.name = "y",
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set

y.name	Character: Name for outcome
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted.
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments

Details

The main differences between a linear model and TLS is that the latter assumes error in the features as well as the outcome. The solution is essentially the projection on the first principal axis.

Author(s)

E.D. Gennatas

---

s_XGBoost *XGBoost Classification and Regression (C, R)*

Description

Tune hyperparameters using grid search and resampling, train a final model, and validate it

Usage

```
s_XGBoost(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  booster = c("gbtree", "gblinear", "dart"),
  missing = NA,
  nrounds = 1000L,
  force.nrounds = NULL,
```

```
    weights = NULL,
    ifw = TRUE,
    ifw.type = 2,
    upsample = FALSE,
    downsample = FALSE,
    resample.seed = NULL,
    obj = NULL,
    feval = NULL,
    xgb.verbose = NULL,
    print_every_n = 100L,
    early_stopping_rounds = 50L,
    eta = 0.01,
    gamma = 0,
    max_depth = 2,
    min_child_weight = 5,
    max_delta_step = 0,
    subsample = 0.75,
    colsample_bytree = 1,
    colsample_bylevel = 1,
    lambda = 0,
    alpha = 0,
    tree_method = "auto",
    sketch_eps = 0.03,
    num_parallel_tree = 1,
    base_score = NULL,
    objective = NULL,
    sample_type = "uniform",
    normalize_type = "forest",
    rate_drop = 0,
    one_drop = 0,
    skip_drop = 0,
    grid.resample.params = setup.resample("kfold", 5),
    gridsearch.type = "exhaustive",
    metric = NULL,
    maximize = NULL,
    importance = NULL,
    print.plot = FALSE,
    plot.fitted = NULL,
    plot.predicted = NULL,
    plot.theme = rtTheme,
    question = NULL,
    verbose = TRUE,
    grid.verbose = FALSE,
    trace = 0,
    save.gridrun = FALSE,
    n.cores = 1,
    nthread = rtCores,
    outdir = NULL,
```

```
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  .gs = FALSE,
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
booster	Character: "gbtree", "gblinear": Booster to use.
missing	String or Numeric: Which values to consider as missing.
nrounds	Integer: Maximum number of rounds to run. Can be set to a high number as early stopping will limit nrounds by monitoring inner CV error
force.nrounds	Integer: Number of rounds to run if not estimating optimal number by CV
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ifw, therefore set weights = NULL if using ifw. Note: If weight are provided, ifw is not used. Leave NULL if setting ifw = TRUE.
ifw	Logical: If TRUE, apply inverse frequency weighting (for Classification only). Note: If weights are provided, ifw is not used.
ifw.type	Integer 0, 1, 2 1: class.weights as in 0, divided by min(class.weights) 2: class.weights as in 0, divided by max(class.weights)
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
obj	Function: Custom objective function. See ?xgboost::xgboost
feval	Function: Custom evaluation function. See ?xgboost::xgboost
xgb.verbose	Integer: Verbose level for XGB learners used for tuning.
print_every_n	Integer: Print evaluation metrics every this many iterations
early_stopping_rounds	Integer: Training on resamples of x.train (tuning) will stop if performance does not improve for this many rounds
eta	[gS] Numeric (0, 1): Learning rate.
gamma	[gS] Numeric: Minimum loss reduction required to make further partition

max_depth
: [gS] Integer: Maximum tree depth.

min_child_weight
: [gS] Numeric: Minimum sum of instance weight needed in a child.

max_delta_step
: [gS] Numeric: Maximum delta step we allow each leaf output to be. O means no constraint. 1-10 may help control the update, especially with imbalanced outcomes.

subsample
: [gS] Numeric: subsample ratio of the training instance

colsample_bytree
: [gS] Numeric: subsample ratio of columns when constructing each tree

colsample_bylevel
: [gS] Numeric

lambda
: [gS] L2 regularization on weights

alpha
: [gS] L1 regularization on weights

tree_method
: [gS] XGBoost tree construction algorithm

sketch_eps
: [gS] Numeric (0, 1):

num_parallel_tree
: Integer: N of trees to grow in parallel: Results in Random Forest -like algorithm. (Default = 1; i.e. regular boosting)

base_score
: Numeric: The mean outcome response.

objective
: (Default = NULL)

sample_type
: Character: Type of sampling algorithm for `dart` booster "uniform": dropped trees are selected uniformly. "weighted": dropped trees are selected in proportion to weight.

normalize_type
: Character.

rate_drop
: [gS] Numeric: Dropout rate for `dart` booster.

one_drop
: [gS] Integer 0, 1: When this flag is enabled, at least one tree is always dropped during the dropout.

skip_drop
: [gS] Numeric [0, 1]: Probability of skipping the dropout procedure during a boosting iteration. If a dropout is skipped, new trees are added in the same manner as gbtree. Non-zero `skip_drop` has higher priority than `rate_drop` or `one_drop`.

grid.resample.params
: List: Output of setup.resample defining grid search parameters.

gridsearch.type
: Character: Type of grid search to perform: "exhaustive" or "randomized".

metric
: Character: Metric to minimize, or maximize if `maximize = TRUE` during grid search. Default = NULL, which results in "Balanced Accuracy" for Classification, "MSE" for Regression, and "Coherence" for Survival Analysis.

maximize
: Logical: If TRUE, `metric` will be maximized if grid search is run.

importance
: Logical: If TRUE, calculate variable importance.

print.plot
: Logical: if TRUE, produce plot using `mplot3` Takes precedence over `plot.fitted` and `plot.predicted`.

plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
grid.verbose	Logical: Passed to gridSearchLearn
trace	Integer: If > 0, print parameter values to console.
save.gridrun	Logical: If TRUE, save grid search models.
n.cores	Integer: Number of cores to use.
nthread	Integer: Number of threads for xgboost using OpenMP. Only parallelize resamples using n.cores or the xgboost execution using this setting. At the moment of writing, parallelization via this parameter causes a linear booster to fail most of the times. Therefore, default is rtCores for 'gbtree', 1 for 'gblinear'
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
.gs	Internal use only
...	Additional arguments passed to xgboost::xgb.train

Details

[gS]: indicates parameter will be autotuned by grid search if multiple values are passed. Learn more about XGBoost's parameters here: http://xgboost.readthedocs.io/en/latest/parameter.html

Value

rtMod object

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: s_AdaBoost(), s_AddTree(), s_BART(), s_BRUTO(), s_BayesGLM(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GAM(), s_GBM(), s_GLM(), s_GLMNET(), s_GLMTree(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_HAL(), s_KNN(), s_LDA(), s_LM(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MARS(), s_MLRF(), s_NBayes(), s_NLA(), s_NLS(), s_NW(), s_PPR(), s_PolyMARS(), s_QDA(), s_QRNN(), s_RF(), s_RFSRC(), s_Ranger(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XRF()

Other Tree-based methods: s_AdaBoost(), s_AddTree(), s_BART(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GBM(), s_GLMTree(), s_H2OGBM(), s_H2ORF(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MLRF(), s_RF(), s_RFSRC(), s_Ranger(), s_XRF()

---

s_XRF *XGBoost Random Forest Classification and Regression (C, R)*

Description

Tune hyperparameters using grid search and resampling, train a final model, and validate it

Usage

```
s_XRF(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  num_parallel_tree = 1000,
  booster = c("gbtree", "gblinear", "dart"),
  missing = NA,
  nrounds = 1,
  weights = NULL,
  ifw = TRUE,
  ifw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  obj = NULL,
  feval = NULL,
  xgb.verbose = NULL,
  print_every_n = 100L,
  early_stopping_rounds = 50L,
  eta = 1,
  gamma = 0,
  max_depth = 12,
  min_child_weight = 1,
  max_delta_step = 0,
  subsample = 0.75,
  colsample_bytree = 1,
  colsample_bylevel = 1,
  lambda = 0,
  alpha = 0,
  tree_method = "auto",
  sketch_eps = 0.03,
```

```
  base_score = NULL,
  objective = NULL,
  sample_type = "uniform",
  normalize_type = "forest",
  rate_drop = 0,
  one_drop = 0,
  skip_drop = 0,
  .gs = FALSE,
  grid.resample.params = setup.resample("kfold", 5),
  gridsearch.type = "exhaustive",
  metric = NULL,
  maximize = NULL,
  importance = TRUE,
  print.plot = FALSE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  grid.verbose = FALSE,
  trace = 0,
  save.gridrun = FALSE,
  n.cores = 1,
  nthread = rtCores,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

`x`	Numeric vector or matrix / data frame of features i.e. independent variables
`y`	Numeric vector of outcome, i.e. dependent variable
`x.test`	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in `x`
`y.test`	Numeric vector of testing set outcome
`x.name`	Character: Name for feature set
`y.name`	Character: Name for outcome
`num_parallel_tree`	Integer: Number of trees to grow
`booster`	Character: Booster to use. Options: "gbtree", "gblinear"
`missing`	String or Numeric: Which values to consider as missing. Default = NA
`nrounds`	Integer: Maximum number of rounds to run. Can be set to a high number as early stopping will limit nrounds by monitoring inner CV error
`weights`	Numeric vector: Weights for cases. For classification, `weights` takes precedence over `ifw`, therefore set `weights = NULL` if using `ifw`. Note: If `weight` are provided, `ifw` is not used. Leave NULL if setting `ifw = TRUE`.

`ifw` Logical: If TRUE, apply inverse frequency weighting (for Classification only). Note: If `weights` are provided, `ifw` is not used.

`ifw.type` Integer 0, 1, 2 1: class.weights as in 0, divided by min(class.weights) 2: class.weights as in 0, divided by max(class.weights)

`upsample` Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness

`downsample` Logical: If TRUE, downsample majority class to match size of minority class

`resample.seed` Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)

`obj` Function: Custom objective function. See `?xgboost::xgboost`

`feval` Function: Custom evaluation function. See `?xgboost::xgboost`

`xgb.verbose` Integer: Verbose level for XGB learners used for tuning.

`print_every_n` Integer: Print evaluation metrics every this many iterations

`early_stopping_rounds`
Integer: Training on resamples of `x.train` (tuning) will stop if performance does not improve for this many rounds

`eta` [gS] Numeric (0, 1): Learning rate.

`gamma` [gS] Numeric: Minimum loss reduction required to make further partition

`max_depth` [gS] Integer: Maximum tree depth.

`min_child_weight`
[gS] Numeric: Minimum sum of instance weight needed in a child.

`max_delta_step` [gS] Numeric: Maximum delta step we allow each leaf output to be. O means no constraint. 1-10 may help control the update, especially with imbalanced outcomes.

`subsample` [gS] Numeric: subsample ratio of the training instance

`colsample_bytree`
[gS] Numeric: subsample ratio of columns when constructing each tree

`colsample_bylevel`
[gS] Numeric

`lambda` [gS] L2 regularization on weights

`alpha` [gS] L1 regularization on weights

`tree_method` [gS] XGBoost tree construction algorithm

`sketch_eps` [gS] Numeric (0, 1):

`base_score` Numeric: The mean outcome response (Defaults to mean)

`objective` (Default = NULL)

`sample_type` Character. Default = "uniform"

`normalize_type` Character. Default = "forest"

`rate_drop` [gS] Numeric: Dropout rate for `dart` booster.

one_drop	[gS] Integer 0, 1: When this flag is enabled, at least one tree is always dropped during the dropout.
skip_drop	[gS] Numeric [0, 1]: Probability of skipping the dropout procedure during a boosting iteration. If a dropout is skipped, new trees are added in the same manner as gbtree. Non-zero skip_drop has higher priority than rate_drop or one_drop.
.gs	Internal use only
grid.resample.params	List: Output of setup.resample defining grid search parameters.
gridsearch.type	Character: Type of grid search to perform: "exhaustive" or "randomized".
metric	Character: Metric to minimize, or maximize if maximize = TRUE during grid search. Default = NULL, which results in "Balanced Accuracy" for Classification, "MSE" for Regression, and "Coherence" for Survival Analysis.
maximize	Logical: If TRUE, metric will be maximized if grid search is run.
importance	Logical: If TRUE, calculate variable importance.
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted.
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
grid.verbose	Logical: Passed to gridSearchLearn
trace	Integer: If higher than 0, will print more information to the console.
save.gridrun	Logical: If TRUE, save grid search models.
n.cores	Integer: Number of cores to use.
nthread	Integer: Number of threads for xgboost using OpenMP. Only parallelize resamples using n.cores or the xgboost execution using this setting. At the moment of writing, parallelization via this parameter causes a linear booster to fail most of the times. Therefore, default is rtCores for 'gbtree', 1 for 'gblinear'
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments

Details

[gS]: indicates parameter will be autotuned by grid search if multiple values are passed. Learn more about XGBoost's parameters here: http://xgboost.readthedocs.io/en/latest/parameter.html

Value

rtMod object

Author(s)

E.D. Gennatas

See Also

train_cv for external cross-validation

Other Supervised Learning: s_AdaBoost(), s_AddTree(), s_BART(), s_BRUTO(), s_BayesGLM(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GAM(), s_GBM(), s_GLM(), s_GLMNET(), s_GLMTree(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_HAL(), s_KNN(), s_LDA(), s_LM(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MARS(), s_MLRF(), s_NBayes(), s_NLA(), s_NLS(), s_NW(), s_PPR(), s_PolyMARS(), s_QDA(), s_QRNN(), s_RF(), s_RFSRC(), s_Ranger(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBoost()

Other Tree-based methods: s_AdaBoost(), s_AddTree(), s_BART(), s_C50(), s_CART(), s_CTree(), s_EVTree(), s_GBM(), s_GLMTree(), s_H2OGBM(), s_H2ORF(), s_LMTree(), s_LightCART(), s_LightGBM(), s_MLRF(), s_RF(), s_RFSRC(), s_Ranger(), s_XGBoost()

---

table1 *Table 1*

Description

Build Table 1. Subject characteristics

Usage

```
table1(
  x,
  summaryFn1 = mean,
  summaryFn2 = sd,
  summaryFn1.extraArgs = list(na.rm = TRUE),
  summaryFn2.extraArgs = list(na.rm = TRUE),
  labelify = TRUE,
  verbose = TRUE,
  filename = NULL
)
```

Arguments

x	data.frame or matrix: Input data, cases by features
summaryFn1	Function: Summary function 1. Default = mean. See Details
summaryFn2	Function: Summary function 2. Default = sd. See Details

summaryFn1.extraArgs
: List: Extra arguments for `summaryFn1`.

summaryFn2.extraArgs
: List: Extra arguments for `summaryFn2`.

labelify
: Logical: If TRUE, apply labelify to column names of `x`

verbose
: Logical: If TRUE, print messages to console.

filename
: Character: Path to output CSV file to save table.

Details

The output will look like "summaryFn1 (summaryFn2)". Using defaults this will be "mean (sd)"

Value

A data.frame, invisibly, with two columns: "Feature", "Value mean (sd) | N"

Author(s)

E.D. Gennatas

Examples

```
table1(iris)
```

`themes` *Print available rtemis themes*

Description

Print available rtemis themes

Usage

```
themes()
```

theme_black *Themes for* `mplot3` *and* `dplot3` *functions*

Description

Themes for `mplot3` and `dplot3` functions

Usage

```
theme_black(
  bg = "#000000",
  plot.bg = "transparent",
  fg = "#ffffff",
  pch = 16,
  cex = 1,
  lwd = 2,
  bty = "n",
  box.col = fg,
  box.alpha = 1,
  box.lty = 1,
  box.lwd = 0.5,
  grid = FALSE,
  grid.nx = NULL,
  grid.ny = NULL,
  grid.col = fg,
  grid.alpha = 0.2,
  grid.lty = 1,
  grid.lwd = 1,
  axes.visible = TRUE,
  axes.col = "transparent",
  tick.col = fg,
  tick.alpha = 0.5,
  tick.labels.col = fg,
  tck = -0.01,
  tcl = NA,
  x.axis.side = 1,
  y.axis.side = 2,
  labs.col = fg,
  x.axis.line = 0,
  x.axis.las = 0,
  x.axis.padj = -1.1,
  x.axis.hadj = 0.5,
  y.axis.line = 0,
  y.axis.las = 1,
  y.axis.padj = 0.5,
  y.axis.hadj = 0.5,
  xlab.line = 1.4,
```

```
  ylab.line = 2,
  zerolines = TRUE,
  zerolines.col = fg,
  zerolines.alpha = 0.5,
  zerolines.lty = 1,
  zerolines.lwd = 1,
  main.line = 0.25,
  main.adj = 0,
  main.font = 2,
  main.col = fg,
  font.family = getOption("rt.font", "Helvetica")
)

theme_blackgrid(
  bg = "#000000",
  plot.bg = "transparent",
  fg = "#ffffff",
  pch = 16,
  cex = 1,
  lwd = 2,
  bty = "n",
  box.col = fg,
  box.alpha = 1,
  box.lty = 1,
  box.lwd = 0.5,
  grid = TRUE,
  grid.nx = NULL,
  grid.ny = NULL,
  grid.col = fg,
  grid.alpha = 0.2,
  grid.lty = 1,
  grid.lwd = 1,
  axes.visible = TRUE,
  axes.col = "transparent",
  tick.col = fg,
  tick.alpha = 1,
  tick.labels.col = fg,
  tck = -0.01,
  tcl = NA,
  x.axis.side = 1,
  y.axis.side = 2,
  labs.col = fg,
  x.axis.line = 0,
  x.axis.las = 0,
  x.axis.padj = -1.1,
  x.axis.hadj = 0.5,
  y.axis.line = 0,
  y.axis.las = 1,
```

```
    y.axis.padj = 0.5,
    y.axis.hadj = 0.5,
    xlab.line = 1.4,
    ylab.line = 2,
    zerolines = TRUE,
    zerolines.col = fg,
    zerolines.alpha = 0.5,
    zerolines.lty = 1,
    zerolines.lwd = 1,
    main.line = 0.25,
    main.adj = 0,
    main.font = 2,
    main.col = fg,
    font.family = getOption("rt.font", "Helvetica")
  )

  theme_blackigrid(
    bg = "#000000",
    plot.bg = "#1A1A1A",
    fg = "#ffffff",
    pch = 16,
    cex = 1,
    lwd = 2,
    bty = "n",
    box.col = fg,
    box.alpha = 1,
    box.lty = 1,
    box.lwd = 0.5,
    grid = TRUE,
    grid.nx = NULL,
    grid.ny = NULL,
    grid.col = bg,
    grid.alpha = 1,
    grid.lty = 1,
    grid.lwd = 1,
    axes.visible = TRUE,
    axes.col = "transparent",
    tick.col = fg,
    tick.alpha = 1,
    tick.labels.col = fg,
    tck = -0.01,
    tcl = NA,
    x.axis.side = 1,
    y.axis.side = 2,
    labs.col = fg,
    x.axis.line = 0,
    x.axis.las = 0,
    x.axis.padj = -1.1,
```

```
  x.axis.hadj = 0.5,
  y.axis.line = 0,
  y.axis.las = 1,
  y.axis.padj = 0.5,
  y.axis.hadj = 0.5,
  xlab.line = 1.4,
  ylab.line = 2,
  zerolines = TRUE,
  zerolines.col = fg,
  zerolines.alpha = 0.5,
  zerolines.lty = 1,
  zerolines.lwd = 1,
  main.line = 0.25,
  main.adj = 0,
  main.font = 2,
  main.col = fg,
  font.family = getOption("rt.font", "Helvetica")
)

theme_darkgray(
  bg = "#121212",
  plot.bg = "transparent",
  fg = "#ffffff",
  pch = 16,
  cex = 1,
  lwd = 2,
  bty = "n",
  box.col = fg,
  box.alpha = 1,
  box.lty = 1,
  box.lwd = 0.5,
  grid = FALSE,
  grid.nx = NULL,
  grid.ny = NULL,
  grid.col = fg,
  grid.alpha = 0.2,
  grid.lty = 1,
  grid.lwd = 1,
  axes.visible = TRUE,
  axes.col = "transparent",
  tick.col = fg,
  tick.alpha = 0.5,
  tick.labels.col = fg,
  tck = -0.01,
  tcl = NA,
  x.axis.side = 1,
  y.axis.side = 2,
  labs.col = fg,
```

```
    x.axis.line = 0,
    x.axis.las = 0,
    x.axis.padj = -1.1,
    x.axis.hadj = 0.5,
    y.axis.line = 0,
    y.axis.las = 1,
    y.axis.padj = 0.5,
    y.axis.hadj = 0.5,
    xlab.line = 1.4,
    ylab.line = 2,
    zerolines = TRUE,
    zerolines.col = fg,
    zerolines.alpha = 0.5,
    zerolines.lty = 1,
    zerolines.lwd = 1,
    main.line = 0.25,
    main.adj = 0,
    main.font = 2,
    main.col = fg,
    font.family = getOption("rt.font", "Helvetica")
  )

  theme_darkgraygrid(
    bg = "#121212",
    plot.bg = "transparent",
    fg = "#ffffff",
    pch = 16,
    cex = 1,
    lwd = 2,
    bty = "n",
    box.col = fg,
    box.alpha = 1,
    box.lty = 1,
    box.lwd = 0.5,
    grid = TRUE,
    grid.nx = NULL,
    grid.ny = NULL,
    grid.col = "#404040",
    grid.alpha = 1,
    grid.lty = 1,
    grid.lwd = 1,
    axes.visible = TRUE,
    axes.col = "transparent",
    tick.col = "#00000000",
    tick.alpha = 1,
    tick.labels.col = fg,
    tck = -0.01,
    tcl = NA,
```

```
    x.axis.side = 1,
    y.axis.side = 2,
    labs.col = fg,
    x.axis.line = 0,
    x.axis.las = 0,
    x.axis.padj = -1.1,
    x.axis.hadj = 0.5,
    y.axis.line = 0,
    y.axis.las = 1,
    y.axis.padj = 0.5,
    y.axis.hadj = 0.5,
    xlab.line = 1.4,
    ylab.line = 2,
    zerolines = TRUE,
    zerolines.col = fg,
    zerolines.alpha = 0.5,
    zerolines.lty = 1,
    zerolines.lwd = 1,
    main.line = 0.25,
    main.adj = 0,
    main.font = 2,
    main.col = fg,
    font.family = getOption("rt.font", "Helvetica")
  )

  theme_darkgrayigrid(
    bg = "#121212",
    plot.bg = "#202020",
    fg = "#ffffff",
    pch = 16,
    cex = 1,
    lwd = 2,
    bty = "n",
    box.col = fg,
    box.alpha = 1,
    box.lty = 1,
    box.lwd = 0.5,
    grid = TRUE,
    grid.nx = NULL,
    grid.ny = NULL,
    grid.col = bg,
    grid.alpha = 1,
    grid.lty = 1,
    grid.lwd = 1,
    axes.visible = TRUE,
    axes.col = "transparent",
    tick.col = "transparent",
    tick.alpha = 1,
```

```
    tick.labels.col = fg,
    tck = -0.01,
    tcl = NA,
    x.axis.side = 1,
    y.axis.side = 2,
    labs.col = fg,
    x.axis.line = 0,
    x.axis.las = 0,
    x.axis.padj = -1.1,
    x.axis.hadj = 0.5,
    y.axis.line = 0,
    y.axis.las = 1,
    y.axis.padj = 0.5,
    y.axis.hadj = 0.5,
    xlab.line = 1.4,
    ylab.line = 2,
    zerolines = TRUE,
    zerolines.col = fg,
    zerolines.alpha = 0.5,
    zerolines.lty = 1,
    zerolines.lwd = 1,
    main.line = 0.25,
    main.adj = 0,
    main.font = 2,
    main.col = fg,
    font.family = getOption("rt.font", "Helvetica")
  )

  theme_white(
    bg = "#ffffff",
    plot.bg = "transparent",
    fg = "#000000",
    pch = 16,
    cex = 1,
    lwd = 2,
    bty = "n",
    box.col = fg,
    box.alpha = 1,
    box.lty = 1,
    box.lwd = 0.5,
    grid = FALSE,
    grid.nx = NULL,
    grid.ny = NULL,
    grid.col = fg,
    grid.alpha = 1,
    grid.lty = 1,
    grid.lwd = 1,
    axes.visible = TRUE,
```

```
  axes.col = "transparent",
  tick.col = fg,
  tick.alpha = 0.5,
  tick.labels.col = fg,
  tck = -0.01,
  tcl = NA,
  x.axis.side = 1,
  y.axis.side = 2,
  labs.col = fg,
  x.axis.line = 0,
  x.axis.las = 0,
  x.axis.padj = -1.1,
  x.axis.hadj = 0.5,
  y.axis.line = 0,
  y.axis.las = 1,
  y.axis.padj = 0.5,
  y.axis.hadj = 0.5,
  xlab.line = 1.4,
  ylab.line = 2,
  zerolines = TRUE,
  zerolines.col = fg,
  zerolines.alpha = 0.5,
  zerolines.lty = 1,
  zerolines.lwd = 1,
  main.line = 0.25,
  main.adj = 0,
  main.font = 2,
  main.col = fg,
  font.family = getOption("rt.font", "Helvetica")
)

theme_whitegrid(
  bg = "#ffffff",
  plot.bg = "transparent",
  fg = "#000000",
  pch = 16,
  cex = 1,
  lwd = 2,
  bty = "n",
  box.col = fg,
  box.alpha = 1,
  box.lty = 1,
  box.lwd = 0.5,
  grid = TRUE,
  grid.nx = NULL,
  grid.ny = NULL,
  grid.col = "#c0c0c0",
  grid.alpha = 1,
```

```
    grid.lty = 1,
    grid.lwd = 1,
    axes.visible = TRUE,
    axes.col = "transparent",
    tick.col = "#00000000",
    tick.alpha = 1,
    tick.labels.col = fg,
    tck = -0.01,
    tcl = NA,
    x.axis.side = 1,
    y.axis.side = 2,
    labs.col = fg,
    x.axis.line = 0,
    x.axis.las = 0,
    x.axis.padj = -1.1,
    x.axis.hadj = 0.5,
    y.axis.line = 0,
    y.axis.las = 1,
    y.axis.padj = 0.5,
    y.axis.hadj = 0.5,
    xlab.line = 1.4,
    ylab.line = 2,
    zerolines = TRUE,
    zerolines.col = fg,
    zerolines.alpha = 0.5,
    zerolines.lty = 1,
    zerolines.lwd = 1,
    main.line = 0.25,
    main.adj = 0,
    main.font = 2,
    main.col = fg,
    font.family = getOption("rt.font", "Helvetica")
  )

  theme_whiteigrid(
    bg = "#ffffff",
    plot.bg = "#E6E6E6",
    fg = "#000000",
    pch = 16,
    cex = 1,
    lwd = 2,
    bty = "n",
    box.col = fg,
    box.alpha = 1,
    box.lty = 1,
    box.lwd = 0.5,
    grid = TRUE,
    grid.nx = NULL,
```

```
  grid.ny = NULL,
  grid.col = bg,
  grid.alpha = 1,
  grid.lty = 1,
  grid.lwd = 1,
  axes.visible = TRUE,
  axes.col = "transparent",
  tick.col = "transparent",
  tick.alpha = 1,
  tick.labels.col = fg,
  tck = -0.01,
  tcl = NA,
  x.axis.side = 1,
  y.axis.side = 2,
  labs.col = fg,
  x.axis.line = 0,
  x.axis.las = 0,
  x.axis.padj = -1.1,
  x.axis.hadj = 0.5,
  y.axis.line = 0,
  y.axis.las = 1,
  y.axis.padj = 0.5,
  y.axis.hadj = 0.5,
  xlab.line = 1.4,
  ylab.line = 2,
  zerolines = TRUE,
  zerolines.col = fg,
  zerolines.alpha = 0.5,
  zerolines.lty = 1,
  zerolines.lwd = 1,
  main.line = 0.25,
  main.adj = 0,
  main.font = 2,
  main.col = fg,
  font.family = getOption("rt.font", "Helvetica")
)

theme_lightgraygrid(
  bg = "#dfdfdf",
  plot.bg = "transparent",
  fg = "#000000",
  pch = 16,
  cex = 1,
  lwd = 2,
  bty = "n",
  box.col = fg,
  box.alpha = 1,
  box.lty = 1,
```

```
  box.lwd = 0.5,
  grid = TRUE,
  grid.nx = NULL,
  grid.ny = NULL,
  grid.col = "#c0c0c0",
  grid.alpha = 1,
  grid.lty = 1,
  grid.lwd = 1,
  axes.visible = TRUE,
  axes.col = "transparent",
  tick.col = "#00000000",
  tick.alpha = 1,
  tick.labels.col = fg,
  tck = -0.01,
  tcl = NA,
  x.axis.side = 1,
  y.axis.side = 2,
  labs.col = fg,
  x.axis.line = 0,
  x.axis.las = 0,
  x.axis.padj = -1.1,
  x.axis.hadj = 0.5,
  y.axis.line = 0,
  y.axis.las = 1,
  y.axis.padj = 0.5,
  y.axis.hadj = 0.5,
  xlab.line = 1.4,
  ylab.line = 2,
  zerolines = TRUE,
  zerolines.col = fg,
  zerolines.alpha = 0.5,
  zerolines.lty = 1,
  zerolines.lwd = 1,
  main.line = 0.25,
  main.adj = 0,
  main.font = 2,
  main.col = fg,
  font.family = getOption("rt.font", "Helvetica")
)

theme_mediumgraygrid(
  bg = "#b3b3b3",
  plot.bg = "transparent",
  fg = "#000000",
  pch = 16,
  cex = 1,
  lwd = 2,
  bty = "n",
```

```
    box.col = fg,
    box.alpha = 1,
    box.lty = 1,
    box.lwd = 0.5,
    grid = TRUE,
    grid.nx = NULL,
    grid.ny = NULL,
    grid.col = "#d0d0d0",
    grid.alpha = 1,
    grid.lty = 1,
    grid.lwd = 1,
    axes.visible = TRUE,
    axes.col = "transparent",
    tick.col = "#00000000",
    tick.alpha = 1,
    tick.labels.col = fg,
    tck = -0.01,
    tcl = NA,
    x.axis.side = 1,
    y.axis.side = 2,
    labs.col = fg,
    x.axis.line = 0,
    x.axis.las = 0,
    x.axis.padj = -1.1,
    x.axis.hadj = 0.5,
    y.axis.line = 0,
    y.axis.las = 1,
    y.axis.padj = 0.5,
    y.axis.hadj = 0.5,
    xlab.line = 1.4,
    ylab.line = 2,
    zerolines = TRUE,
    zerolines.col = fg,
    zerolines.alpha = 0.5,
    zerolines.lty = 1,
    zerolines.lwd = 1,
    main.line = 0.25,
    main.adj = 0,
    main.font = 2,
    main.col = fg,
    font.family = getOption("rt.font", "Helvetica")
)
```

Arguments

bg	Color: Figure background
plot.bg	Color: Plot region background
fg	Color: Foreground color used as default for multiple elements like axes and

	labels, which can be defined separately
pch	Integer: Point character.
cex	Float: Character expansion factor.
lwd	Float: Line width.
bty	Character: Box type: "o", "l", "7", "c", "u", or "]", or "n". Default = "n" (no box)
box.col	Box color if `bty != "n"`
box.alpha	Float: Box alpha
box.lty	Integer: Box line type
box.lwd	Float: Box line width
grid	Logical: If TRUE, draw grid in plot regions
grid.nx	Integer: N of vertical grid lines
grid.ny	Integer: N of horizontal grid lines
grid.col	Grid color
grid.alpha	Float: Grid alpha
grid.lty	Integer: Grid line type
grid.lwd	Float: Grid line width
axes.visible	Logical: If TRUE, draw axes
axes.col	Axes colors
tick.col	Tick color
tick.alpha	Float: Tick alpha
tick.labels.col	Tick labels' color
tck	`graphics::parr`'s tck argument: Tick length, can be negative
tcl	`graphics::parr`'s tcl argument
x.axis.side	Integer: Side to place x-axis. Default = 1 (bottom)
y.axis.side	Integer: Side to place y-axis. Default = 2 (left)
labs.col	Labels' color
x.axis.line	Numeric: `graphics::axis`'s `line` argument for the x-axis
x.axis.las	Numeric: `graphics::axis`'s `las` argument for the x-axis
x.axis.padj	Numeric: x-axis' `padj`: Adjustment for the x-axis tick labels' position
x.axis.hadj	Numeric: x-axis' `hadj`
y.axis.line	Numeric: `graphics::axis`'s `line` argument for the y-axis
y.axis.las	Numeric: `graphics::axis`'s `las` argument for the y-axis
y.axis.padj	Numeric: y-axis' `padj`
y.axis.hadj	Numeric: y-axis' `hadj`
xlab.line	Numeric: Line to place `xlab`
ylab.line	Numeric: Line to place `ylab`

`zerolines`	Logical: If TRUE, draw lines on x = 0, y = 0, if within plot limits
`zerolines.col`	Zerolines color
`zerolines.alpha`	Float: Zerolines alpha
`zerolines.lty`	Integer: Zerolines line type
`zerolines.lwd`	Float: Zerolines line width
`main.line`	Float: How many lines away from the plot region to draw title.
`main.adj`	Float: How to align title. Default = 0 (left-align)
`main.font`	Integer: 1: Regular, 2: Bold
`main.col`	Title color
`font.family`	Character: Font to be used throughout plot.

`timeProc` *Time a process*

Description

`timeProc` measures how long it takes for a process to run

Usage

```
timeProc(..., verbose = TRUE)
```

Arguments

`...`	Command to be timed. (Will be converted using `as.expression`)
`verbose`	Logical: If TRUE, print messages to console

Author(s)

E.D. Gennatas

tohtml — *Generate* `CheckData` *object description in HTML*

Description

Generate `CheckData` object description in HTML

Usage

```
tohtml(
  x,
  name = NULL,
 css = list(font.family = "Helvetica", color = "#fff", background.color = "#242424")
)
```

Arguments

`x`	`CheckData` object
`name`	Character: Name of the data set
`css`	List: CSS styles

Author(s)

E.D. Gennatas

train_cv — *Tune, Train, and Test an* **rtemis** *Learner by Nested Resampling*

Description

`train` is a high-level function to tune, train, and test an **rtemis** model by nested resampling, with optional preprocessing and decomposition of input features

Usage

```
train_cv(
  x,
  y = NULL,
  alg = "ranger",
  train.params = list(),
  .preprocess = NULL,
  .decompose = NULL,
  weights = NULL,
  n.repeats = 1,
  outer.resampling = setup.resample(resampler = "strat.sub", n.resamples = 10),
```

```
  inner.resampling = setup.resample(resampler = "kfold", n.resamples = 5),
  bag.fn = median,
  x.name = NULL,
  y.name = NULL,
  save.mods = TRUE,
  save.tune = TRUE,
  bag.fitted = FALSE,
  outer.n.workers = 1,
  print.plot = FALSE,
  plot.fitted = FALSE,
  plot.predicted = TRUE,
  plot.theme = rtTheme,
  print.res.plot = FALSE,
  question = NULL,
  verbose = TRUE,
  res.verbose = FALSE,
  trace = 0,
  headless = FALSE,
  outdir = NULL,
  save.plots = FALSE,
  save.rt = ifelse(!is.null(outdir), TRUE, FALSE),
  save.mod = TRUE,
  save.res = FALSE,
  debug = FALSE,
  ...
)
```

Arguments

`x`	Numeric vector or matrix / data frame of features i.e. independent variables
`y`	Numeric vector of outcome, i.e. dependent variable
`alg`	Character: Learner to use. Options: select_learn
`train.params`	Optional named list of parameters to be passed to `alg`. All parameters can be passed as part of `...` as well
`.preprocess`	Optional named list of parameters to be passed to preprocess. Set using setup.preprocess, e.g. `decom = setup.preprocess(impute = TRUE)`
`.decompose`	Optional named list of parameters to be used for decomposition / dimensionality reduction. Set using setup.decompose, e.g. `decom = setup.decompose("ica", 12)`
`weights`	Numeric vector: Weights for cases. For classification, `weights` takes precedence over `ifw`, therefore set `weights = NULL` if using `ifw`. Note: If `weight` are provided, `ifw` is not used. Leave NULL if setting `ifw = TRUE`.
`n.repeats`	Integer: Number of times to repeat the outer resampling. This was added for completeness, but in practice we use either k-fold crossvalidation, e.g. 10-fold, especially in large samples, or a higher number of stratified subsamples, e.g. 25, for smaller samples

`outer.resampling`
: List: Output of setup.resample to define outer resampling scheme

`inner.resampling`
: List: Output of setup.resample to define inner resampling scheme

`bag.fn`
: Function to use to average prediction if `bag.fitted = TRUE`. Default = `median`

`x.name`
: Character: Name of predictor dataset

`y.name`
: Character: Name of outcome

`save.mods`
: Logical: If TRUE, retain trained models in object, otherwise discard (save space if running many resamples).

`save.tune`
: Logical: If TRUE, save the best.tune data frame for each resample (output of gridSearchLearn)

`bag.fitted`
: Logical: If TRUE, use all models to also get a bagged prediction on the full sample. To get a bagged prediction on new data using the same models, use predict.rtModCV

`outer.n.workers`
: Integer: Number of cores to use for the outer i.e. testing resamples. You are likely parallelizing either in the inner (tuning) or the learner itself is parallelized. Don't parallelize the parallelization

`print.plot`
: Logical: if TRUE, produce plot using `mplot3` Takes precedence over `plot.fitted` and `plot.predicted`.

`plot.fitted`
: Logical: if TRUE, plot True (y) vs Fitted

`plot.predicted`
: Logical: if TRUE, plot True (y.test) vs Predicted. Requires `x.test` and `y.test`

`plot.theme`
: Character: "zero", "dark", "box", "darkbox"

`print.res.plot`
: Logical: Print model performance plot for each resample. from all resamples. Defaults to TRUE

`question`
: Character: the question you are attempting to answer with this model, in plain language.

`verbose`
: Logical: If TRUE, print summary to screen.

`res.verbose`
: Logical: Passed to each individual learner's `verbose` argument

`trace`
: Integer: (Not really used) Print additional information if > 0.

`headless`
: Logical: If TRUE, turn off all plotting.

`outdir`
: Character: Path where output should be saved

`save.plots`
: Logical: If TRUE, save plots to outdir

`save.rt`
: Logical: If TRUE and `outdir` is set, save all models to `outdir`

`save.mod`
: Logical: If TRUE, save all output to an RDS file in `outdir` `save.mod` is TRUE by default if an `outdir` is defined. If set to TRUE, and no `outdir` is defined, outdir defaults to `paste0("./s.", mod.name)`

`save.res`
: Logical: If TRUE, save the full output of each model trained on differents resamples under subdirectories of `outdir`

`debug`
: Logical: If TRUE, sets `outer.n.workers` to 1, `options(error=recover)`, and options(warn = 2)

`...`
: Additional train.params to be passed to learner. Will be concatenated with `train.params`

Details

- Note on resampling: You should never use an outer resampling method with replacement if you will also be using an inner resampling (for tuning). The duplicated cases from the outer resampling may appear both in the training and testing sets of the inner resamples, leading to underestimated testing error.
- If there is an error while running either the outer or inner resamples in parallel, the error message returned by R will likely be unhelpful. Repeat the command after setting both inner and outer resample run to use a single core, which should provide an informative message.

The `train` command is replacing `elevate`. Note: specifying id.strat for the inner resampling is not yet supported.

Value

Object of class `rtModCV` (Regression) or `rtModCVClass` (Classification)

`error.test.repeats`
: the mean or aggregate error, as appropriate, for each repeat

`error.test.repeats.mean`
: the mean error of all repeats, i.e. the mean of `error.test.repeats`

`error.test.repeats.sd`
: if `n.repeats` > 1, the standard deviation of `error.test.repeats`

`error.test.res`
: the error for each resample, for each repeat

Author(s)

E.D. Gennatas

Examples

```
## Not run:
# Regression

x <- rnormmat(100, 50)
w <- rnorm(50)
y <- x %*% w + rnorm(50)
mod <- train(x, y)

# Classification

data(Sonar, package = "mlbench")
mod <- train(Sonar)

## End(Not run)
```

tunable *Print tunable hyperparameters for a supervised learning algorithm*

Description

Print tunable hyperparameters for a supervised learning algorithm

Usage

```
tunable(
  alg = c("glmnet", "svm", "cart", "ranger", "gbm", "xgboost", "lightgbm")
)
```

Arguments

alg	Character string: Algorithm name.

Value

Prints tunable hyperparameters for the specified algorithm.

Author(s)

EDG

typeset *Set type of columns*

Description

Given an index of columns, convert identified columns of data.frame to factor, ordered factor, or integer. A number of datasets are distributed with an accompanying index of this sort, especially to define which variables should be treated as categorical (here, factors) for predicting modeling. This functions aims to make data type conversions in those cases easier.

Usage

```
typeset(
  x,
  factor.index = NULL,
  orderedfactor.index = NULL,
  integer.index = NULL
)
```

Arguments

`x`	data frame: input whose columns' types you want to edit
`factor.index`	Integer, vector: Index of columns to be converted to factors using `factor(x)`
`orderedfactor.index`	Integer, vector: Index of columns to be converted to ordered factors using `factor(x, ordered = TRUE)`
`integer.index`	Integer, vector: Index of columns to be converted to integers using `as.integer`

Author(s)

E.D. Gennatas

`uci_heart_failure` *UCI Heart Failure Data*

Description

A subset of data from the World Health Organization Global Tuberculosis Report ...

Usage

```
uci_heart_failure
```

Format

`uci_heart_failure`**:**
A data frame with 299 rows and 13 columns:

DEATH_EVENT Boolean: If the patient died during the follow-up period

Source

https://archive.ics.uci.edu/dataset/519/heart+failure+clinical+records

---

uniprot_get *Get protein sequence from UniProt*

Description

Get protein sequence from UniProt

Usage

```
uniprot_get(
  accession = "Q9UMX9",
  baseURL = "https://rest.uniprot.org/uniprotkb",
  verbosity = 1
)
```

Arguments

accession	Character: UniProt Accession number - e.g. "Q9UMX9"
baseURL	Character: UniProt rest API base URL. Default = "https://rest.uniprot.org/uniprotkb"
verbosity	Integer: If > 0, print messages to console

Value

List with two elements: Annotation & Sequence

Author(s)

E.D. Gennatas

Examples

```
## Not run:
mapt <- uniprot_get("Q9UMX9")

## End(Not run)
```

---

uniquevalsperfeat *Unique values per feature*

Description

Get number of unique values per features

Usage

```
uniquevalsperfeat(x, excludeNA = FALSE)
```

Arguments

x	matrix or data frame input
excludeNA	Logical: If TRUE, exclude NA values from unique count.

Value

Vector, integer of length `NCOL(x)` with number of unique values per column/feature

Author(s)

E.D. Gennatas

Examples

```
## Not run:
uniquevalsperfeat(iris)

## End(Not run)
```

`winsorize` *Winsorize vector*

Description

Replace extreme values by absolute or quantile threshold

Usage

```
winsorize(
  x,
  lo = NULL,
  hi = NULL,
  prob.lo = 0.025,
  prob.hi = 0.975,
  quantile.type = 7,
  verbose = TRUE
)
```

Arguments

x	Numeric vector: Input data
lo	Numeric: If not NULL, replace any values in `x` lower than this with this. Default = NULL
hi	Numeric: If not NULL, replace any values in `x` higher than this with this.
prob.lo	Numeric (0, 1): If not NULL and `lo = NULL`, find sample quantile that corresponds to this probability and set as `lo`.

prob.hi	Numeric (0, 1): If not NULL and `hi = NULL`, find sample quantile that corresponds to this probability and set as `hi`.
quantile.type	Integer: passed to `stats::quantile`
verbose	Logical: If TRUE, print messages to console.

Details

If both lo and prob.lo or both hi and prob.hi are NULL, cut-off is set to min(x) and max(x) respectively, i.e. no values are changed

Author(s)

E.D. Gennatas

Examples

```
# Winsorize a normally distributed variable
x <- rnorm(500)
xw <- winsorize(x)
# Winsorize an exponentially distributed variable only on
# the top 5% highest values
x <- rexp(500)
xw <- winsorize(x, prob.lo = NULL, prob.hi = .95)
```

---

xlsx2list *Read all sheets of an XLSX file into a list*

Description

Read all sheets of an XLSX file into a list

Usage

```
xlsx2list(
  x,
  sheet = NULL,
  startRow = 1,
  colNames = TRUE,
  na.strings = "NA",
  detectDates = TRUE,
  skipEmptyRows = TRUE,
  skipEmptyCols = TRUE
)
```

Arguments

x	Character: path or URL to XLSX file
sheet	Integer, vector: Sheet(s) to read. If NULL, will read all sheets in x
startRow	Integer, vector: First row to start reading. Will be recycled as needed for all sheets
colNames	Logical: If TRUE, use the first row of data
na.strings	Character vector: stringd to be interpreted as NA
detectDates	Logical: If TRUE, try to automatically detect dates
skipEmptyRows	Logical: If TRUE, skip empty rows
skipEmptyCols	Logical: If TRUE, skip empty columns

Author(s)

E.D. Gennatas

---

xselect_decom *Select* **rtemis** *cross-decomposer*

Description

Accepts decomposer name (supports abbreviations) and returns **rtemis** function name or the function itself. If run with no parameters, prints list of available algorithms.

Usage

```
xselect_decom(xdecom, fn = FALSE, desc = FALSE)
```

Arguments

xdecom	Character: Cross-decomposition name; case insensitive
fn	Logical: If TRUE, return function, otherwise return name of function. Default = FALSE
desc	Logical: If TRUE, return full name of algorithm. Default = FALSE

Value

Function or name of function (see param `fn`) or full name of algorithm (`desc`)

Author(s)

E.D. Gennatas

See Also

Other Cross-Decomposition: `x_CCA()`

xtdescribe — *Describe longitudinal dataset*

Description

This is a test to emulate the `xtdescribe` function in Stata.

Usage

```
xtdescribe(x, ID_col = 1, time_col = 2, n_patterns = 9)
```

Arguments

`x`	data.frame with longitudinal data
`ID_col`	Integer: The column position of the ID variable
`time_col`	Integer: The column position of the time variable
`n_patterns`	Integer: The number of patterns to display

Author(s)

EDG

x_CCA — *Sparse Canonical Correlation Analysis (CCA)*

Description

Run a sparse Canonical Correlation Analysis using the `PMA` package

Usage

```
x_CCA(
  x,
  z,
  x.test = NULL,
  z.test = NULL,
  y = NULL,
  outcome = NULL,
  k = 3,
  niter = 20,
  nperms = 50,
  permute.niter = 15,
  typex = "standard",
  typez = "standard",
  penaltyx = NULL,
```

```
    penaltyz = NULL,
    standardize = TRUE,
    upos = FALSE,
    vpos = FALSE,
    verbose = TRUE,
    n.cores = rtCores,
    outdir = NULL,
    save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
    ...
)
```

Arguments

x	Matrix: Training x dataset
z	Matrix: Training z dataset
x.test	Matrix (Optional): Testing x set
z.test	Matrix (Optional): Testing z set
y	Outcome vector (Optional): If supplied, linear combinations of x and z need to be additionaly correlated with this
outcome	Character: Type of outcome y: "survival", "multiclass", "quantitative"
k	Integer: Number of components
niter	Integer: Number of iterations
nperms	Integer: Number of permutations to run with CCA.permute The higher, the better.
permute.niter	Integer: Number of iterations to run for each permutation with CCA.permute
typex	Character: "standard", "ordered". Use "standard" if columns of x are unordered; lasso penalty is applied to enforce sparsity. Otherwise, use "ordered"; fused lasso penalty is applied, to enforce both sparsity and smoothness.
typez	Character: "standard", "ordered". Same as typex for z dataset
penaltyx	Float: The penalty to be applied to the matrix x, i.e. the penalty that results in the canonical vector u. If typex is "standard" then the L1 bound on u is penaltyx*sqrt(ncol(x)). In this case penaltyx must be between 0 and 1 (larger L1 bound corresponds to less penalization). If "ordered" then it's the fused lasso penalty lambda, which must be non-negative (larger lambda corresponds to more penalization).
penaltyz	Float: The penalty to be applied to the matrix z, i.e. the penalty that results in the canonical vector v. If typez is "standard" then the L1 bound on v is penaltyz*sqrt(ncol(z)). In this case penaltyz must be between 0 and 1 (larger L1 bound corresponds to less penalization). If "ordered" then it's the fused lasso penalty lambda, which must be non-negative (larger lambda corresponds to more penalization).
standardize	Logical: If TRUE, center and scale columns of x and z
upos	Logical: Require elements of u to be positive
vpos	Logical: Require elements of v to be positive

`verbose`	Logical: Print messages, including `trace` from `x_CCA.permute` and `PMA::CCA`
`n.cores`	Integer: Number of cores to use
`outdir`	Path to output directory. Default = NULL
`save.mod`	Logical: If TRUE, and `outdir` is defined, will save trained CCA model to `outdir`. Default = TRUE if `outdir` is set, otherwise FALSE
`...`	Additional arguments to be passed to `PMA::CCA`

Details

#' `x_CCA` runs `PMA::CCA`. If penaltyx is NULL, penaltyx *and* penaltyz will be estimated automatically using x_CCA.permute (adapted to run in parallel)

Author(s)

E.D. Gennatas

See Also

Other Cross-Decomposition: `xselect_decom()`

`zip2longlat` *Get Longitude and Lattitude for zip code(s)*

Description

Returns a data.table of longitude and lattitude for one or more zip codes, given an input dataset

Usage

```
zip2longlat(x, zipdt)
```

Arguments

`x`	Character vector: Zip code(s)
`zipdt`	data.table with "zip", "lng", and "lat" columns

`zipdist` *Get distance between pairs of zip codes*

Description

Get distance between pairs of zip codes

Usage

```
zipdist(x, y, zipdt)
```

Arguments

`x`	Character vector
`y`	Character vector, same length as `x`
`zipdt`	data.table with columns `zip`, `lng`, `lat`

Value

`data.table` with distances in meters

Author(s)

E.D. Gennatas

`%BC%` *Binary matrix times character vector*

Description

Binary matrix times character vector

Usage

```
x %BC% labels
```

Arguments

`x`	A binary matrix or data.frame
`labels`	Character vector length equal to `ncol(x)`

Value

a character vector

Author(s)

E.D. Gennatas

Index

∗ **Bayesian**
  s_BayesGLM, 400
∗ **Clustering**
  c_CMeans, 48
  c_DBSCAN, 49
  c_EMC, 50
  c_H2OKMeans, 51
  c_HARDCL, 52
  c_HOPACH, 53
  c_KMeans, 54
  c_MeanShift, 55
  c_NGAS, 56
  c_PAM, 57
  c_PAMK, 58
  c_SPEC, 59
∗ **Cross-Decomposition**
  x_CCA, 569
  xselect_decom, 568
∗ **Decomposition**
  d_H2OAE, 147
  d_H2OGLRM, 150
  d_ICA, 152
  d_Isomap, 153
  d_KPCA, 154
  d_LLE, 155
  d_MDS, 156
  d_NMF, 157
  d_PCA, 158
  d_SPCA, 159
  d_SVD, 161
  d_TSNE, 162
  d_UMAP, 163
∗ **Deep Learning**
  d_H2OAE, 147
  s_H2ODL, 432
  s_TFN, 529
∗ **Ensembles**
  s_AdaBoost, 392
  s_GBM, 417
  s_Ranger, 505
  s_RF, 509
∗ **Interpretable models**
  s_AddTree, 394
  s_C50, 405
  s_CART, 407
  s_GLM, 421
  s_GLMNET, 424
  s_GLMTree, 427
  s_LMTree, 475
∗ **Supervised Learning**
  s_AdaBoost, 392
  s_AddTree, 394
  s_BART, 397
  s_BayesGLM, 400
  s_BRUTO, 403
  s_C50, 405
  s_CART, 407
  s_CTree, 410
  s_EVTree, 413
  s_GAM, 415
  s_GBM, 417
  s_GLM, 421
  s_GLMNET, 424
  s_GLMTree, 427
  s_GLS, 430
  s_H2ODL, 432
  s_H2OGBM, 435
  s_H2ORF, 438
  s_HAL, 441
  s_KNN, 443
  s_LDA, 445
  s_LightCART, 447
  s_LightGBM, 450
  s_LM, 472
  s_LMTree, 475
  s_MARS, 479
  s_MLRF, 482
  s_NBayes, 486

s_NLA, 487
s_NLS, 489
s_NW, 492
s_PolyMARS, 494
s_PPR, 497
s_QDA, 501
s_QRNN, 503
s_Ranger, 505
s_RF, 509
s_RFSRC, 513
s_SDA, 518
s_SGD, 521
s_SPLS, 524
s_SVM, 527
s_TFN, 529
s_XGBoost, 534
s_XRF, 539
∗ **Survival Regression**
s_PSurv, 499
∗ **Tree-based methods**
s_AdaBoost, 392
s_AddTree, 394
s_BART, 397
s_C50, 405
s_CART, 407
s_CTree, 410
s_EVTree, 413
s_GBM, 417
s_GLMTree, 427
s_H2OGBM, 435
s_H2ORF, 438
s_LightCART, 447
s_LightGBM, 450
s_LMTree, 475
s_MLRF, 482
s_Ranger, 505
s_RF, 509
s_RFSRC, 513
s_XGBoost, 534
s_XRF, 539
∗ **datasets**
rtPalettes, 344
uci_heart_failure, 564
%BC%, 572

amazonCol *(*rtPalettes*)*, 344
any_constant, 13
appleCol *(*rtPalettes*)*, 344
array, *196*
as.data.tree.linadleaves, 13
as.data.tree.rpart, 14
as.data.tree.shyoptleaves, 14
as.list, *196*
auc, 15, *31*
auc_pairs, *15*, 16

bacc, 16
base::cut, *300*, *375*
berkeleyCol *(*rtPalettes*)*, 344
betas.lihad, 17
bias_variance, 18, *316*
binmat2vec, 19
bold, 19
boost, 20, *165*, *166*, *286*, *309*
bootstrap, 23, *324*
brier_score, 23
brownCol *(*rtPalettes*)*, 344

C, *392*, *394*, *405*, *486*, *501*
c_CMeans, 48, *50–54*, *56*, *57*, *59*, *60*
c_DBSCAN, *49*, 49, *51–54*, *56*, *57*, *59*, *60*
c_EMC, *49*, *50*, 50, *52–54*, *56*, *57*, *59*, *60*
c_H2OKMeans, *49–51*, 51, *53*, *54*, *56*, *57*, *59*, *60*
c_HARDCL, *49–52*, 52, *54*, *56*, *57*, *59*, *60*
c_HOPACH, *49–53*, 53, *54*, *56*, *57*, *59*, *60*
c_KMeans, *49–54*, 54, *56*, *57*, *59*, *60*
c_MeanShift, *49–54*, 55, *57*, *59*, *60*
c_NGAS, *49–54*, *56*, 56, *57*, *59*, *60*
c_PAM, *49–54*, *56*, *57*, 57, *59*, *60*
c_PAMK, *49–54*, *56*, *57*, 58, *60*
c_SPEC, *49–54*, *56*, *57*, *59*, 59
calibrate, 24
calibrate_cv, 25, *282*, *297*
californiaCol *(*rtPalettes*)*, 344
calpolyCol *(*rtPalettes*)*, 344
caltechCol *(*rtPalettes*)*, 344
cartLiteBoostTV, *309*
catrange, 26
catsize, 27
check_data, 29
check_files, 30
checkpoint_earlystop, 27
chicagoCol *(*rtPalettes*)*, 344
chill, 31
class_error, 31, *310*, *311*
class_imbalance, 32
clean_colnames, 33, *66*, *320*
clean_names, 33

clust, 34
cmuCol (rtPalettes), 344
coef.lihad, 34
coef.rtMod (rtMod-methods), 332
col2grayscale, 35
col2hex, 35
colMax, 36
color_fade, 42
color_invertRGB, 43
color_mean, 43
color_order, 44
color_separate, 45
color_sqdist, 45
colorAdjust, 36
colorGrad, 37, *91*, *107*, *231–233*, *364*
colorGrad.x, 39
colorgradient.x, 40
colorMix, 41
colorOp, 41
cols2list, 46
columbiaCol (rtPalettes), 344
cornellCol (rtPalettes), 344
create_config, 46, *321*
crules, 47
csuCol (rtPalettes), 344
cyan (bold), 19

d_H2OAE, 147, *151*, *153–161*, *163, 164*, *435*, *533*
d_H2OGLRM, *149*, 150, *153–161*, *163, 164*
d_ICA, *149*, *151*, 152, *154–161*, *163, 164*
d_Isomap, *149*, *151*, *153*, 153, *155–161*, *163, 164*
d_KPCA, *149*, *151*, *153, 154*, 154, *156–161*, *163, 164*
d_LLE, *149*, *151*, *153–155*, 155, *157–161*, *163, 164*
d_MDS, *149*, *151*, *153–156*, 156, *158–161*, *163, 164*
d_NMF, *149*, *151*, *153–157*, 157, *159–161*, *163, 164*
d_PCA, *149*, *151*, *153–158*, 158, *160, 161*, *163, 164*
d_SPCA, *149*, *151*, *153–159*, 159, *161*, *163, 164*
d_SVD, *149*, *151*, *153–160*, 161, *163, 164*
d_TSNE, *149*, *151*, *153–161*, 162, *164*
d_UMAP, *149*, *151*, *153–161*, *163*, 163
dartmouthCol (rtPalettes), 344
dat2bsplinemat, 60, *296*
dat2poly, 61
date2factor, 62
date2ym, 63
date2yq, 63
ddb_collect, 64
ddb_data, *64*, 65
ddSci, *26*, 66, *110*, *235*, *357*
decom, 67, *149*
dependency_check, 68
desaturate, 68
describe, 69
describe.rtModCV (rtModCV-methods), 338
df_movecolumn, 69
distillTreeRules, 70
dplot3_addtree, 71, *396*
dplot3_bar, 72, *105, 106*, *279, 280*
dplot3_box, 75, *304*
dplot3_calibration, 80
dplot3_cart, *14*, 82
dplot3_conf, 84
dplot3_fit, 86
dplot3_graphd3, 86
dplot3_graphjs, 87
dplot3_heatmap, 90
dplot3_leaflet, 93
dplot3_linad, 95
dplot3_pie, 97
dplot3_protein, 99
dplot3_pvals, 105
dplot3_spectrogram, 106
dplot3_table, 109
dplot3_ts, 110
dplot3_varimp, 113
dplot3_volcano, 114
dplot3_x, 118
dplot3_xt, 122
dplot3_xy, *81*, *86*, *112*, *117*, 126
dplot3_xyz, 132
drange, 136
dt_check_unique, 136
dt_describe, 137
dt_get_column_attr, 138
dt_get_duplicates, 138
dt_get_factor_levels, 139
dt_index_attr, 139
dt_inspect_type, 140
dt_keybin_reshape, 140
dt_merge, 141

dt_names_by_attr, 142
dt_names_by_class, 143
dt_Nuniqueperfeat, 143
dt_pctmatch, 144
dt_pctmissing, 144
dt_set_autotypes, 145
dt_set_clean_all, 146
dt_set_cleanfactorlevels, 145
dt_set_logical2factor, 146
dt_set_oneHot *(oneHot)*, 274

earlystop, 164, *366*
emoryCol *(rtPalettes)*, 344
ethCol *(rtPalettes)*, 344
expand.boost, 165
explain, 167
expression, *196*

f1, 167
factor_harmonize, *17*, 169, *284*, *362*, *383*
factor_NA2missing, 170
factoryze, 168
family, *522*
fct_describe, 170
firefoxCol *(rtPalettes)*, 344
fitted.rtMod *(rtMod-methods)*, 332
format.call, 171
formatLightRules, 172
formatRules, 172
fwhm2sigma, 173

gbm::gbm, *419*
get-names, 174
get_loaded_pkg_version, 175
get_mode, 176, *300*, *375*
get_rules, 177
get_vars_from_rules, 177
getcharacternames *(getnames)*, 174
getdatenames *(getnames)*, 174
getfactornames *(get-names)*, 174
getlogicalnames *(getnames)*, 174
getnames, 174
getnamesandtypes, 175
getnumericnames *(getnames)*, 174
ggtheme_dark, 178
ggtheme_light, 179
glm, *274*, *481*
glmLite, 180, *288*
gmean, 182
googleCol *(rtPalettes)*, 344
gp, 182
graph_node_metrics, 183
gray *(bold)*, 19
green *(bold)*, 19
gridCheck, 184
gtTable, 184

hal9001::fit_hal, *442*
hawaiiCol *(rtPalettes)*, 344
hilite *(bold)*, 19
hilitebig *(bold)*, 19
hmsCol *(rtPalettes)*, 344
htest, 185

illinoisCol *(rtPalettes)*, 344
imperialCol *(rtPalettes)*, 344
inspect_type, *140*, 187
invlogit, 187
iowaCol *(rtPalettes)*, 344
is_constant, 188
is_discrete, 189
italic *(bold)*, 19

jeffersonCol *(rtPalettes)*, 344
jhuCol *(rtPalettes)*, 344

kfold, 189, *325*

labelify, *78*, *113*, 190, *237*, *254*, *544*
lincoef, *181*, 191, *371*, *468*
list2csv, 193
logistic, 193
logit, 194
logloss, 194
loocv, 195
lotri2edgeList, 195
lsapply, 196

mae *(mse)*, 272
magenta *(bold)*, 19
make_key, 196
massGAM, 197, *314*
massGLAM, 198, *387*
massGLM, 200, *314*, *387*
massUni, 201
matchcases, 202
mcgillCol *(rtPalettes)*, 344
mergelongtreatment, 203
meta_mod, 204

mgetnames, 206
mhist, 206, *257*, *259*
michiganCol *(*rtPalettes*)*, 344
microsoftCol *(*rtPalettes*)*, 344
mitCol *(*rtPalettes*)*, 344
mlegend, 208
mod_error, *85*, 209
mozillaCol *(*rtPalettes*)*, 344
mplot3_adsr, 210
mplot3_bar, 213
mplot3_box, 215
mplot3_conf, 218, *225*
mplot3_confbin, 221
mplot3_decision, 223
mplot3_fit, 224
mplot3_fret, 225
mplot3_graph, 226
mplot3_harmonograph, 229
mplot3_heatmap, 230, *259*
mplot3_img, *221*, 234, *245*
mplot3_laterality, 236
mplot3_lolli, 238
mplot3_missing, 240
mplot3_mosaic, 241
mplot3_pr, 243
mplot3_prp, 245
mplot3_res, 245, *281*
mplot3_roc, 246
mplot3_surv, 248
mplot3_survfit, 250
mplot3_varimp, 253
mplot3_x, 254
mplot3_xy, *151*, *209*, *210*, *212*, *223*, *225*, *248–250*, *259*, 259, *267*
mplot3_xym, *259*, 266
mplot_AGGTEobj, 268
mplot_hsv, 269
mplot_raster, 270
mse, 272
msew *(*mse*)*, 272
msuCol *(*rtPalettes*)*, 344
multigplot, 272

nCr, 273
nhsCol *(*rtPalettes*)*, 344
nihCol *(*rtPalettes*)*, 344
nlareg, *295*
nyuCol *(*rtPalettes*)*, 344

oddsratio, 273
oddsratiotable, 274
oneHot, 274
onehot2factor, 276
orange *(*bold*)*, 19
oxfordCol *(*rtPalettes*)*, 344

p.adjust, *106*
palettize, 277
pennCol *(*rtPalettes*)*, 344
pennLightPalette *(*rtPalettes*)*, 344
pennPalette *(*rtPalettes*)*, 344
pennstateCol *(*rtPalettes*)*, 344
permute, 277
pfread, 278
plot.massGAM, 279
plot.massGLM, 280
plot.resample, 281
plot.rtMod *(*rtMod-methods*)*, 332
plot.rtModCV *(*rtModCV-methods*)*, 338
plot.rtModCVCalibration, 281
plot.rtTest, 283
plotly.heat, 283
precision, 284
predict.addtree, 285
predict.boost, 285
predict.cartLite, 286
predict.cartLiteBoostTV, 287
predict.glmLite, 288
predict.glmLiteBoostTV, 288
predict.hytboost, 289
predict.hytboostnow, 290
predict.hytreenow, 290
predict.hytreew, 291
predict.LightRuleFit, 292
predict.lihad, 293, *462*
predict.linadleaves, 294
predict.nlareg, 295
predict.nullmod, 296
predict.rtBSplines, 296
predict.rtMeta *(*rtMeta-methods*)*, 332
predict.rtMod *(*rtMod-methods*)*, 332
predict.rtModBag *(*rtModBag-methods*)*, 334
predict.rtModCV, *561*
predict.rtModCV *(*rtModCV-methods*)*, 338
predict.rtModCVCalibration, 297
predict.rtModLite *(*rtMod-methods*)*, 332
predict.rtTLS, 297
predict.rulefit, 298

preprocess, *152*, *157*, *275*, 298, *303*, *373*, *397*, *560*
preprocess_, 302
present, 304
present_gridsearch, 306
previewcolor, 306
princetonCol *(*rtPalettes*)*, 344
print.addtree, 308
print.boost, 309
print.cartLiteBoostTV, 309
print.CheckData, 310
print.class_error, 310
print.glmLiteBoostTV, 311
print.gridSearch, 311
print.hytboost, 312
print.hytboostnow, 312
print.lihad, 313
print.linadleaves, 313
print.massGAM, 314
print.massGLM, 314
print.regError, 315
print.resample, 315
print.rtBiasVariance, 316
print.rtClust *(*rtClust-methods*)*, 330
print.rtDecom, 316
print.rtMod *(*rtMod-methods*)*, 332
print.rtModLite *(*rtModLite-methods*)*, 339
print.rtTLS, 317
print.surv_error, 317
prune.addtree, 318
psd, 318

qstat, 319

read, *47*, 319
read_config, 321
recycle, 322
red *(*bold*)*, 19
reg_error, 322
relu, 323
resample, *245*, *281*, *315*, 324, *363*, *365*, *368*, *373*, *377*
reset *(*bold*)*, 19
residuals.rtMod *(*rtMod-methods*)*, 332
reverseLevels, 325
revfactorlevels, 326
rfVarSelect, 326
rmse *(*mse*)*, 272
rnormmat, 327
rowMax, 328
rsd, 328
rsq, 329
rstudio_theme_rtemis, 329
rt_reactable, 354
rt_save, 355
rtClust-methods, 330
rtemis *(*rtemis-package*)*, 11
rtemis-package, 11
rtemis_palette, 330
rtInitProjectDir, 331
rtlayout, *214*, *217*, *220*, *237*, *239*, *258*, *265*, 331
rtMeta-methods, 332
rtMod-methods, 332
rtModBag-methods, 334
rtModClass *(*rtModClass-class*)*, 335
rtModClass-class, 335
rtModCV-methods, 338
rtModLite-methods, 339
rtModLog *(*rtModLog-class*)*, 340
rtModLog-class, 340
rtModLogger *(*rtModLogger-class*)*, 341
rtModLogger-class, 341
rtpalette, 343
rtPalettes, 344
rtROC, *247*, 351
rtset, 352
rtversion, 352
rtXDecom *(*rtXDecom-class*)*, 352
rtXDecom-class, 352
ruleDist, 356
rules2medmod, 357
runifmat, 357
rwthCol *(*rtPalettes*)*, 344

s_AdaBoost, 392, *397*, *399*, *402*, *404*, *407*, *410*, *412*, *414*, *415*, *417*, *421*, *424*, *427*, *430*, *432*, *435*, *438*, *441*, *443*, *445*, *447*, *450*, *454*, *474*, *477*, *482*, *485*, *487*, *489*, *491*, *493*, *496*, *499*, *503*, *505*, *509*, *512*, *515*, *520*, *523*, *526*, *529*, *533*, *538*, *539*, *543*
s_AddTree, *71*, *308*, *318*, *394*, 394, *399*, *402*, *404*, *407*, *410*, *412*, *414*, *415*, *417*, *421*, *424*, *427*, *430*, *432*, *435*, *438*, *441*, *443*, *445*, *447*, *450*, *454*, *474*, *477*, *482*, *485*, *487*, *489*, *491*, *493*, *496*, *499*, *503*, *505*, *509*, *512*, *515*,

520, 523, 526, 529, 533, 538, 539, 543
s_BART, 394, 397, 397, 402, 404, 407, 410, 412, 414, 415, 417, 421, 424, 427, 430, 432, 435, 438, 441, 443, 445, 447, 450, 454, 474, 477, 482, 485, 487, 489, 491, 493, 496, 499, 503, 505, 509, 512, 515, 520, 523, 526, 529, 533, 538, 539, 543
s_BayesGLM, 394, 397, 399, 400, 404, 407, 410, 412, 414, 417, 421, 424, 427, 430, 432, 435, 438, 441, 443, 445, 447, 450, 454, 474, 477, 482, 485, 487, 489, 491, 493, 496, 499, 503, 505, 509, 512, 515, 520, 523, 526, 529, 533, 538, 543
s_BRUTO, 394, 397, 399, 402, 403, 407, 410, 412, 414, 417, 421, 424, 427, 430, 432, 435, 438, 441, 443, 445, 447, 450, 454, 474, 477, 482, 485, 487, 489, 491, 493, 496, 499, 503, 505, 509, 512, 515, 520, 523, 526, 529, 533, 538, 543
s_C50, 394, 397, 399, 402, 404, 405, 410, 412, 414, 415, 417, 421, 424, 427, 430, 432, 435, 438, 441, 443, 445, 447, 450, 454, 474, 477, 482, 485, 487, 489, 491, 493, 496, 499, 503, 505, 509, 512, 515, 520, 523, 526, 529, 533, 538, 539, 543
s_CART, 83, 245, 394, 397, 399, 402, 404, 407, 407, 412, 414, 415, 417, 421, 424, 427, 430, 432, 435, 438, 441, 443, 445, 447, 450, 454, 474, 477, 482, 485, 487, 489, 491, 493, 496, 499, 503, 505, 509, 512, 515, 520, 523, 526, 529, 533, 538, 539, 543
s_CTree, 394, 397, 399, 402, 404, 407, 410, 410, 414, 415, 417, 421, 424, 427, 430, 432, 435, 438, 441, 443, 445, 447, 450, 454, 474, 477, 482, 485, 487, 489, 491, 493, 496, 499, 503, 505, 509, 512, 515, 520, 523, 526, 529, 533, 538, 539, 543
s_EVTree, 394, 397, 399, 402, 404, 407, 410, 412, 413, 417, 421, 424, 427, 430, 432, 435, 438, 441, 443, 445, 447, 450, 454, 474, 477, 482, 485, 487, 489, 491, 493, 496, 499, 503, 505, 509, 512, 515, 520, 523, 526, 529, 533, 538, 539, 543
s_GAM, 394, 397, 399, 402, 404, 407, 410, 412, 414, 415, 421, 424, 427, 430, 432, 435, 438, 441, 443, 445, 447, 450, 454, 474, 477, 482, 485, 487, 489, 491, 493, 496, 499, 503, 505, 509, 512, 515, 520, 523, 526, 529, 533, 538, 543
s_GBM, 70, 367, 394, 397, 399, 402, 404, 407, 410, 412, 414, 415, 417, 417, 424, 427, 430, 432, 435, 438, 441, 443, 445, 447, 450, 454, 474, 477, 482, 485, 487, 489, 491, 493, 496, 499, 503, 505, 509, 512, 515, 516, 520, 523, 526, 529, 533, 538, 539, 543
s_GLM, 394, 397, 399, 402, 404, 407, 410, 412, 414, 417, 421, 421, 427, 430, 432, 435, 438, 441, 443, 445, 447, 450, 454, 474, 477, 482, 485, 487, 489, 491, 493, 496, 499, 503, 505, 509, 512, 515, 520, 523, 526, 529, 533, 538, 543
s_GLMNET, 369, 394, 397, 399, 402, 404, 407, 410, 412, 414, 417, 421, 424, 424, 430, 432, 435, 438, 441, 443, 445, 447, 450, 454, 474, 477, 482, 485, 487, 489, 491, 493, 496, 499, 503, 505, 509, 512, 515, 517, 520, 523, 526, 529, 533, 538, 543
s_GLMTree, 394, 397, 399, 402, 404, 407, 410, 412, 414, 415, 417, 421, 424, 427, 427, 432, 435, 438, 441, 443, 445, 447, 450, 454, 474, 477, 482, 485, 487, 489, 491, 493, 496, 499, 503, 505, 509, 512, 515, 520, 523, 526, 529, 533, 538, 539, 543
s_GLS, 394, 397, 399, 402, 404, 407, 410, 412, 414, 417, 421, 424, 427, 430, 430, 435, 438, 441, 443, 445, 447, 450, 454, 474, 477, 482, 485, 487, 489, 491, 493, 496, 499, 503, 505, 509, 512, 515, 520, 523, 526, 529, 533, 538, 543
s_H2ODL, 149, 394, 397, 399, 402, 404, 407, 410, 412, 415, 417, 421, 424, 427, 430, 432, 432, 438, 441, 443, 445,

447, *450*, *454*, *474*, *477*, *482*, *485*, *487*, *489*, *491*, *493*, *496*, *499*, *503*, *505*, *509*, *512*, *515*, *520*, *523*, *526*, *529*, *533*, *538*, *543*
s_H2OGBM, *394*, *397*, *399*, *402*, *404*, *407*, *410*, *412*, *415*, *417*, *421*, *424*, *427*, *430*, *432*, *435*, 435, *441*, *443*, *445*, *447*, *450*, *454*, *474*, *477*, *482*, *485*, *487*, *489*, *491*, *493*, *496*, *499*, *503*, *505*, *509*, *512*, *515*, *520*, *523*, *526*, *529*, *533*, *538*, *539*, *543*
s_H2ORF, *394*, *397*, *399*, *402*, *404*, *407*, *410*, *412*, *415*, *417*, *421*, *424*, *427*, *430*, *432*, *435*, *438*, 438, *443*, *445*, *447*, *450*, *454*, *474*, *477*, *482*, *485*, *487*, *489*, *491*, *493*, *496*, *499*, *503*, *505*, *509*, *512*, *515*, *520*, *523*, *526*, *529*, *533*, *538*, *539*, *543*
s_HAL, *394*, *397*, *399*, *402*, *404*, *407*, *410*, *412*, *415*, *417*, *421*, *424*, *427*, *430*, *432*, *435*, *438*, *441*, 441, *445*, *447*, *450*, *454*, *474*, *477*, *482*, *485*, *487*, *489*, *491*, *493*, *496*, *499*, *503*, *505*, *509*, *512*, *515*, *520*, *523*, *526*, *529*, *533*, *538*, *543*
s_KNN, *394*, *397*, *399*, *402*, *404*, *407*, *410*, *412*, *415*, *417*, *421*, *424*, *427*, *430*, *432*, *435*, *438*, *441*, *443*, 443, *447*, *450*, *454*, *474*, *477*, *482*, *485*, *487*, *489*, *491*, *493*, *496*, *499*, *503*, *505*, *509*, *512*, *515*, *520*, *523*, *526*, *529*, *533*, *538*, *543*
s_LDA, *394*, *397*, *399*, *402*, *404*, *407*, *410*, *412*, *415*, *417*, *421*, *424*, *427*, *430*, *432*, *435*, *438*, *441*, *443*, *445*, 445, *450*, *454*, *474*, *477*, *482*, *485*, *487*, *489*, *491*, *493*, *496*, *499*, *503*, *505*, *509*, *512*, *515*, *520*, *523*, *526*, *529*, *533*, *538*, *543*
s_LightCART, *394*, *397*, *399*, *402*, *404*, *407*, *410*, *412*, *415*, *417*, *421*, *424*, *427*, *430*, *432*, *435*, *438*, *441*, *443*, *445*, *447*, 447, *454*, *474*, *477*, *482*, *485*, *487*, *489*, *491*, *493*, *496*, *499*, *503*, *505*, *509*, *512*, *515*, *520*, *523*, *526*, *529*, *533*, *538*, *539*, *543*
s_LightGBM, *369*, *394*, *397*, *399*, *402*, *404*, *407*, *410*, *412*, *415*, *417*, *421*, *424*, *427*, *430*, *432*, *435*, *438*, *441*, *443*, *445*, *447*, *450*, 450, *459*, *474*, *477*, *482*, *485*, *487*, *489*, *491*, *493*, *496*, *499*, *503*, *505*, *509*, *512*, *515*, *520*, *523*, *526*, *529*, *533*, *538*, *539*, *543*
s_LightRF, 455
s_LightRuleFit, *177*, *369*, 458
s_LIHAD, *294*, *370*, 461
s_LIHADBoost, 463
s_LINAD, *95*, *96*, 466
s_LINOA, 469
s_LM, *394*, *397*, *399*, *402*, *404*, *407*, *410*, *412*, *415*, *417*, *421*, *424*, *427*, *430*, *432*, *435*, *438*, *441*, *443*, *445*, *447*, *450*, *454*, 472, *477*, *482*, *485*, *487*, *489*, *491*, *493*, *496*, *499*, *503*, *505*, *509*, *512*, *515*, *520*, *523*, *526*, *529*, *533*, *538*, *543*
s_LMTree, *394*, *397*, *399*, *402*, *404*, *407*, *410*, *412*, *415*, *417*, *421*, *424*, *427*, *430*, *432*, *435*, *438*, *441*, *443*, *445*, *447*, *450*, *454*, *474*, 475, *482*, *485*, *487*, *489*, *491*, *493*, *496*, *499*, *503*, *505*, *509*, *512*, *515*, *520*, *523*, *526*, *529*, *533*, *538*, *539*, *543*
s_LOESS, 477
s_LOGISTIC, 479
s_MARS, *372*, *394*, *397*, *399*, *402*, *404*, *407*, *410*, *412*, *415*, *417*, *421*, *424*, *427*, *430*, *432*, *435*, *438*, *441*, *443*, *445*, *447*, *450*, *454*, *474*, *477*, 479, *485*, *487*, *489*, *491*, *493*, *496*, *499*, *503*, *505*, *509*, *512*, *515*, *520*, *523*, *526*, *529*, *533*, *538*, *543*
s_MLRF, *394*, *397*, *399*, *402*, *404*, *407*, *410*, *412*, *415*, *417*, *421*, *424*, *427*, *430*, *432*, *435*, *438*, *441*, *443*, *445*, *447*, *450*, *454*, *474*, *477*, *482*, 482, *487*, *489*, *491*, *493*, *496*, *499*, *503*, *505*, *509*, *512*, *515*, *520*, *523*, *526*, *529*, *533*, *538*, *539*, *543*
s_MULTINOM, 485
s_NBayes, *394*, *397*, *399*, *402*, *404*, *407*, *410*, *412*, *415*, *417*, *421*, *424*, *427*, *430*, *432*, *435*, *438*, *441*, *443*, *445*, *447*, *450*, *454*, *474*, *477*, *482*, *485*, 486, *489*, *491*, *493*, *496*, *499*, *503*, *505*, *509*, *512*, *515*, *520*, *523*, *526*, *529*,

533, 538, 543
s_NLA, 394, 397, 399, 402, 404, 407, 410, 412, 415, 417, 421, 424, 427, 430, 432, 435, 438, 441, 443, 445, 447, 450, 454, 474, 477, 482, 485, 487, 487, 491, 493, 496, 499, 503, 505, 509, 512, 515, 520, 523, 526, 529, 533, 538, 543
s_NLS, 128, 133, 262, 394, 397, 399, 402, 404, 407, 410, 412, 415, 417, 421, 424, 427, 430, 432, 435, 438, 441, 443, 445, 447, 450, 454, 474, 477, 482, 485, 487, 489, 489, 493, 496, 499, 503, 505, 509, 512, 515, 520, 523, 526, 529, 533, 538, 543
s_NW, 394, 397, 399, 402, 404, 407, 410, 412, 415, 417, 421, 424, 427, 430, 432, 435, 438, 441, 443, 445, 447, 450, 454, 474, 477, 482, 485, 487, 489, 491, 492, 496, 499, 503, 505, 509, 512, 515, 520, 523, 526, 529, 533, 538, 543
s_POLY, 494
s_PolyMARS, 394, 397, 399, 402, 404, 407, 410, 412, 415, 417, 421, 424, 427, 430, 432, 435, 438, 441, 443, 445, 447, 450, 454, 474, 477, 482, 485, 487, 489, 491, 493, 494, 499, 503, 505, 509, 512, 515, 520, 523, 526, 529, 533, 538, 543
s_PPR, 394, 397, 399, 402, 404, 407, 410, 412, 415, 417, 421, 424, 427, 430, 432, 435, 438, 441, 443, 445, 447, 450, 454, 474, 477, 482, 485, 487, 489, 491, 493, 496, 497, 503, 505, 509, 512, 515, 520, 523, 526, 529, 533, 538, 543
s_PSurv, 499
s_QDA, 394, 397, 399, 402, 404, 407, 410, 412, 415, 417, 421, 424, 427, 430, 432, 435, 438, 441, 443, 445, 447, 450, 454, 474, 477, 482, 485, 487, 489, 491, 493, 496, 499, 501, 505, 509, 512, 515, 520, 523, 526, 529, 533, 538, 543
s_QRNN, 394, 397, 399, 402, 404, 407, 410, 412, 415, 417, 421, 424, 427, 430, 432, 435, 438, 441, 443, 445, 447, 450, 454, 474, 477, 482, 485, 487, 489, 491, 493, 496, 499, 503, 503, 509, 512, 515, 520, 523, 526, 529, 533, 538, 543
s_Ranger, 376, 394, 397, 399, 402, 404, 407, 410, 412, 415, 417, 421, 424, 427, 430, 432, 435, 438, 441, 443, 445, 447, 450, 454, 474, 477, 482, 485, 487, 489, 491, 493, 496, 499, 503, 505, 505, 512, 515, 520, 523, 526, 529, 533, 538, 539, 543
s_RF, 70, 394, 397, 399, 402, 404, 407, 410, 412, 415, 417, 421, 424, 427, 430, 432, 435, 438, 441, 443, 445, 447, 450, 454, 474, 477, 482, 485, 487, 489, 491, 493, 496, 499, 503, 505, 509, 509, 515, 520, 523, 526, 529, 533, 538, 539, 543
s_RFSRC, 394, 397, 399, 402, 404, 407, 410, 412, 415, 417, 421, 424, 427, 430, 432, 435, 438, 441, 443, 445, 447, 450, 454, 474, 477, 482, 485, 487, 489, 491, 493, 496, 499, 503, 505, 509, 512, 513, 520, 523, 526, 529, 533, 538, 539, 543
s_RLM, 515
s_RuleFit, 177, 516
s_SDA, 394, 397, 399, 402, 404, 407, 410, 412, 415, 417, 421, 424, 427, 430, 432, 435, 438, 441, 443, 445, 447, 450, 454, 474, 477, 482, 485, 487, 489, 491, 493, 496, 499, 503, 505, 509, 512, 515, 518, 523, 526, 529, 533, 538, 543
s_SGD, 394, 397, 399, 402, 404, 407, 410, 412, 415, 417, 421, 424, 427, 430, 432, 435, 438, 441, 443, 445, 447, 450, 454, 474, 477, 482, 485, 487, 489, 491, 493, 496, 499, 503, 505, 509, 512, 515, 520, 521, 526, 529, 533, 538, 543
s_SPLS, 394, 397, 399, 402, 404, 407, 410, 412, 415, 417, 421, 424, 427, 430, 432, 435, 438, 441, 443, 445, 447, 450, 454, 474, 477, 482, 485, 487, 489, 491, 493, 496, 499, 503, 505, 509, 512, 515, 520, 523, 524, 529, 533, 538, 543

s_SVM, *394*, *397*, *399*, *402*, *404*, *407*, *410*, *412*, *415*, *417*, *421*, *424*, *427*, *430*, *432*, *435*, *438*, *441*, *443*, *445*, *447*, *450*, *454*, *474*, *477*, *482*, *485*, *487*, *489*, *491*, *493*, *496*, *499*, *503*, *505*, *509*, *512*, *515*, *520*, *523*, *526*, 527, *533*, *538*, *543*
s_TFN, *149*, *394*, *397*, *399*, *402*, *404*, *407*, *410*, *412*, *415*, *417*, *421*, *424*, *427*, *430*, *432*, *435*, *438*, *441*, *443*, *445*, *447*, *450*, *454*, *474*, *477*, *482*, *485*, *487*, *489*, *491*, *493*, *496*, *499*, *503*, *505*, *509*, *512*, *515*, *520*, *523*, *526*, *529*, 529, *538*, *543*
s_TLS, *297*, *317*, 533
s_XGBoost, *394*, *397*, *399*, *402*, *404*, *407*, *410*, *412*, *415*, *417*, *421*, *424*, *427*, *430*, *432*, *435*, *438*, *441*, *443*, *445*, *447*, *450*, *454*, *474*, *477*, *482*, *485*, *487*, *489*, *491*, *493*, *496*, *499*, *503*, *505*, *509*, *512*, *515*, *520*, *523*, *526*, *529*, *533*, 534, *543*
s_XRF, *394*, *397*, *399*, *402*, *404*, *407*, *410*, *412*, *415*, *417*, *421*, *424*, *427*, *430*, *432*, *435*, *438*, *441*, *443*, *445*, *447*, *450*, *454*, *474*, *477*, *482*, *485*, *487*, *489*, *491*, *493*, *496*, *499*, *503*, *505*, *509*, *512*, *515*, *520*, *523*, *526*, *529*, *533*, *538*, *539*, 539
savePMML, 358
scrippsCol *(*rtPalettes*)*, 344
se, 359
select_clust, *12*, *128*, *133*, *262*, 360
select_decom, *12*, 360
select_learn, *12*, *21*, *166*, *205*, 361, *560*
selectiter, 359
sensitivity, *17*, 362
seql, 362
setdiffsym, 363
setup.bag.resample, 363
setup.color, 364
setup.cv.resample, 365
setup.decompose, 366, *560*
setup.earlystop, *21*, 366
setup.GBM, 367
setup.grid.resample, 368
setup.LightRuleFit, 369
setup.LIHAD, 370
setup.lincoef, *370*, 371, *462*
setup.MARS, 372
setup.meta.resample, 373
setup.preprocess, 373, *560*
setup.Ranger, 376
setup.resample, *18*, *25*, *47*, *325*, *367*, *376*, 377, *391*, *396*, *409*, *419*, *425*, *429*, *442*, *449*, *453*, *457*, *460*, *481*, *495*, *498*, *507*, *511*, *519*, *525*, *528*, *537*, *542*, *561*
sfsuCol *(*rtPalettes*)*, 344
sge_submit, 377
sigmoid, 379
size, 379
softmax, 380
softplus, 380, *488*
sortedlines, 380
sparsernorm, 381
sparseVectorSummary, 382
sparsify, 382
specificity, *17*, 383
stanfordCol *(*rtPalettes*)*, 344
stats::ppr, *498*
stderror, 383
strat.boot, *324*, 384
strat.sub, *324*, 385
strata2factor, 385
summarize, 386
summary.massGAM, 387
summary.massGLM, 387
summary.rtMod *(*rtMod-methods*)*, 332
summary.rtModCV *(*rtModCV-methods*)*, 338
surv_error, *317*, 388
svd1, 388
synth_multimodal, 389
synth_reg_data, 391

table1, 543
techCol *(*rtPalettes*)*, 344
texasCol *(*rtPalettes*)*, 344
theme_black, 545
theme_blackgrid *(*theme_black*)*, 545
theme_blackigrid *(*theme_black*)*, 545
theme_darkgray *(*theme_black*)*, 545
theme_darkgraygrid *(*theme_black*)*, 545
theme_darkgrayigrid *(*theme_black*)*, 545
theme_lightgraygrid *(*theme_black*)*, 545
theme_mediumgraygrid *(*theme_black*)*, 545
theme_white *(*theme_black*)*, 545

theme_whitegrid *(theme_black)*, 545
theme_whiteigrid *(theme_black)*, 545
themes, 544
timeProc, 558
tohtml, 559
torontoCol *(rtPalettes)*, 344
train_cv, *25, 26*, *297*, *304, 305*, *325*, *366*, *373*, *394*, *399*, *404*, *407*, *410*, *412*, *414*, *417*, *421*, *424*, *427*, *430*, *435*, *438*, *440, 441*, *443*, *445*, *447*, *450*, *454*, *474*, *477, 478*, *482*, *485*, *489*, *491*, *493*, *496*, *499*, *501*, *503*, *505*, *509*, *512*, *515*, *520*, *523*, *526*, *529*, *533*, *538*, *543*, 559
tunable, 563
typeset, 563

ubcCol *(rtPalettes)*, 344
ucdCol *(rtPalettes)*, 344
uci_heart_failure, 564
uciCol *(rtPalettes)*, 344
uclaCol *(rtPalettes)*, 344
uclCol *(rtPalettes)*, 344
ucmercedCol *(rtPalettes)*, 344
ucrColor *(rtPalettes)*, 344
ucsbCol *(rtPalettes)*, 344
ucscCol *(rtPalettes)*, 344
ucsdCol *(rtPalettes)*, 344
ucsfLegacyCol *(rtPalettes)*, 344
ucsfPalette *(rtPalettes)*, 344
umdCol *(rtPalettes)*, 344
underline *(bold)*, 19
uniprot_get, 565
uniquevalsperfeat, 565
usfCol *(rtPalettes)*, 344
uwCol *(rtPalettes)*, 344

vanderbiltCol *(rtPalettes)*, 344

washuCol *(rtPalettes)*, 344
winsorize, 566

x_CCA, *568*, 569
xlsx2list, 567
xselect_decom, *12*, 568, *571*
xtdescribe, 569

yaleCol *(rtPalettes)*, 344

zip2longlat, 571
zipdist, 572