Package ‘loose.rock’

August 31, 2018

Title Set of Functions to Use in Survival Analysis and in Data Science

Version 1.0.9

Description Collection of functions to improve work-flow in survival analysis and data science. The package features include: the generation of balanced datasets, live retrieval of protein coding genes from two public databases, generation of random matrix based on covariance matrix, cache function to store function results. This work was supported by two grants from the Portuguese Foundation for Science and technology, and the EU Commission under SOUND project.

Depends R (>= 3.4)

License GPL (>= 3)

Encoding UTF-8

LazyData true

NeedsCompilation no

RoxygenNote 6.1.0

Imports biomaRt, digest, futile.options, ggfortify, ggplot2, graphics, grDevices, MASS, methods, dplyr, reshape2, rlang, stats

Suggests knitr, rmarkdown, devtools, roxygen2, testthat, survival

VignetteBuilder knitr

BugReports https://www.github.com/averissimo/loose.rock/issues

URL https://www.github.com/averissimo/loose.rock

Author André Veríssimo [aut, cre],
Susana Vinga [aut]

Maintainer André Veríssimo <andre.verissimo@tecnico.ulisboa.pt>

Repository CRAN

Date/Publication 2018-08-31 19:10:03 UTC

R topics documented:

`balanced.cv.folds` *Create balanced folds for cross validation*

Description

Create balanced folds for cross validation

Usage

```
balanced.cv.folds(..., nfolds = 10)
```

Arguments

`...`	vectors representing data
`nfolds`	number of folds to be created

Value

list with given input, nfolds and result. The result is a list matching the input with foldid attributed to each position.

Examples

```
balanced.cv.folds(1:10, 11:15, nfolds = 2)
balanced.cv.folds(1:10, 11:13, nfolds = 10) # will give a warning
balanced.cv.folds(1:100, 101:133, nfolds = 10)
```

balanced.train.and.test *Get a balanced test and train dataset*

Description

Get a balanced test and train dataset

Usage

```
balanced.train.and.test(..., train.perc = 0.9, join.all = TRUE)
```

Arguments

`...`	vectors of index (could be numeric or logical)
`train.perc`	percentage of dataset to be training set
`join.all`	join all index in the end in two vectors (train and test vectors)

Value

train and test index vectors (two lists if ‘join.all = FALSE‘, two vectors otherwise)

Examples

```
set1 <- seq(20)
balanced.train.and.test(set1, train.perc = .9)
####
set.seed(1985)
set1 <- rbinom(20, prob = 3/20, size = 1) == 1
balanced.train.and.test(set1, train.perc = .9)
####
set1 <- c(TRUE,TRUE,TRUE,TRUE,TRUE,TRUE,TRUE,TRUE,FALSE,
TRUE,TRUE,TRUE,TRUE,TRUE,TRUE,TRUE,TRUE,TRUE,FALSE,TRUE)
set2 <- !set1
balanced.train.and.test(set1, set2, train.perc = .9)
```

base.dir *change base.dir for run.cache*

Description

change base.dir for run.cache

Usage

```
base.dir(path = NULL)
```

Arguments

`path`	to base directory where cache is saved

Value

the new path

Examples

```
base.dir('/tmp/cache')
```

`coding.genes` *Retrive coding genes from known databases*

Description

It retrieves from NCBI and

Usage

```
coding.genes(verbose = TRUE)
```

Arguments

`verbose`	show messages with number of genes retrieved

Value

a table with gene information

Examples

```
# This can take a few minutes depending on the connection
## Not run:
    coding.genes()

## End(Not run)
```

digest.cache *Default digest method*

Description

Sets a default caching algorithm to use with run.cache

Usage

```
digest.cache(val)
```

Arguments

`val`	object to calculate hash over

Value

a hash of the sha256

Examples

```
digest.cache(c(1,2,3,4,5))
digest.cache('some example')
```

draw.cov.matrix *Plot covariance heatmap from matrix*

Description

Plot covariance heatmap from matrix

Usage

```
draw.cov.matrix(my.matrix, fun = stats::cov, ...)
```

Arguments

`my.matrix`	matrix to calculate the covariance
`fun`	function to use
`...`	arguments to pass to fun function

Value

a ggplot2

Examples

```
draw.cov.matrix(matrix(rnorm(100), ncol = 10))
draw.cov.matrix(gen.synth.xdata(10, 10, .2))
```

`gen.synth.xdata` *Generate xdata matrix with pre-determined covariance*

Description

Covariance matrix is created using for each position cov.matrix[i,j] = rho^|i-j|

Usage

```
gen.synth.xdata(n.obs, n.vars, rho, my.mean = rep(0, n.vars))
```

Arguments

`n.obs`	number of observations
`n.vars`	number of variables
`rho`	value used to calculate rho^|i-j| . values between 1 and 0
`my.mean`	vector of mean variables

Value

a matrix of xdata

Examples

```
gen.synth.xdata(100, 8, .75)
gen.synth.xdata(1000, 5, .2)
cov(gen.synth.xdata(n.obs = 10, n.vars = 10, rho = .2))
```

`loose.rock` *loose.rock: compiles my generic function*

Description

loose.rock: compiles my generic function

`loose.rock.options` *Constants for ’loose.rock’ package*

Description

Log level constants and the logger options.

Usage

```
loose.rock.options(..., simplify = FALSE, update = list())
```

Arguments

`...`	TODO
`simplify`	TODO
`update`	pair list of update to options

Details

The logging configuration is managed by ’loose.rock.options’, a function generated by OptionsManager within ’futile.options’.

See Also

`futile.options`

`multiplot` *Multiple plot*

Description

Not mine, taken from http://www.cookbook-r.com/Graphs/Multiple_graphs_on_one_page_(ggplot2)/

Usage

```
multiplot(..., plotlist = NULL, ncol = 1, layout = NULL)
```

Arguments

`...`	ggplot objects
`plotlist`	ggplot objects (alternative)
`ncol`	Number of columns in layout
`layout`	A matrix specifying the layout. If present, ’ncol’ is ignored If the layout is something like matrix(c(1,2,3,3), nrow=2, byrow=TRUE),

Value

nothing

Examples

```
# First plot
library(ggplot2)
p1 <- ggplot(ChickWeight, aes(x=Time, y=weight, colour=Diet, group=Chick)) +
  geom_line() +
  ggtitle("Growth curve for individual chicks")
# Second plot
p2 <- ggplot(ChickWeight, aes(x=Time, y=weight, colour=Diet)) +
  geom_point(alpha=.3) +
  geom_smooth(alpha=.2, size=1) +
  ggtitle("Fitted growth curve per diet")
multiplot(p1, p2, ncol = 2)
```

my.colors *Custom pallete of colors*

Description

Custom pallete of colors

Usage

```
my.colors(ix = NULL)
```

Arguments

ix	index for a color

Value

a color

Examples

```
my.colors()
my.colors(5)
```

my.symbols *Custom pallete of symbols in plots*

Description

Custom pallete of symbols in plots

Usage

```
my.symbols(ix = NULL)
```

Arguments

ix	index for symbol

Value

a symbol

Examples

```
my.symbols()
my.symbols(2)
```

proper *Capitalizes all words in string*

Description

Capitalizes all words in string

Usage

```
proper(x)
```

Arguments

x	String

Value

a capitalized string (all words)

Examples

```
proper('i saw a dEaD parrot')
```

run.cache *Run function and save cache*

Description

This method saves the function that's being called

Usage

```
run.cache(fun, ..., seed = NULL, base.dir = NULL,
  cache.prefix = "generic_cache", cache.digest = list(),
  show.message = NULL, force.recalc = FALSE, add.to.hash = NULL)
```

Arguments

fun	function call name
...	parameters for function call
seed	when function call is random, this allows to set seed beforehand
base.dir	directory where data is stored
cache.prefix	prefix for file name to be generated from parameters (...)
cache.digest	cache of the digest for one or more of the parameters
show.message	show message that data is being retrieved from cache
force.recalc	force the recalculation of the values
add.to.hash	something to add to the filename generation

Value

the result of fun(...)

Examples

```
# [optional] save cache in a temporary directory
# otherwise it writes to the current directory
# to folder named run-cache
base.dir(tempdir())
#
run.cache(c, 1, 2, 3, 4)
#
# next three should use the same cache
#  note, the middle call should be a little faster as digest is not calculated
#   for the first argument
run.cache(c, 1, 2, 3, 4)
run.cache(c, 1, 2, 3, 4, cache.digest = list(digest.cache(1)))
run.cache(c, a=1, 2, c= 3, 4)
```

run.cache,function-method

Run function and save cache

Description

Run function and save cache

Usage

```
## S4 method for signature '`function`'
run.cache(fun, ..., seed = NULL,
  base.dir = NULL, cache.prefix = "generic_cache",
  cache.digest = list(), show.message = NULL, force.recalc = FALSE,
  add.to.hash = NULL)
```

Arguments

`fun`	function call name
`...`	parameters for function call
`seed`	when function call is random, this allows to set seed beforehand
`base.dir`	directory where data is stored
`cache.prefix`	prefix for file name to be generated from parameters (...)
`cache.digest`	cache of the digest for one or more of the parameters
`show.message`	show message that data is being retrieved from cache
`force.recalc`	force the recalculation of the values
`add.to.hash`	something to add to the filename generation

Value

the result of fun(...)

Examples

```
# [optional] save cache in a temporary directory
# otherwise it writes to the current directory
# to folder named run-cache
base.dir(tempdir())
#
run.cache(c, 1, 2, 3, 4)
#
# next three should use the same cache
#  note, the middle call should be a little faster as digest is not calculated
#   for the first argument
run.cache(c, 1, 2, 3, 4)
run.cache(c, 1, 2, 3, 4, cache.digest = list(digest.cache(1)))
run.cache(c, a=1, 2, c= 3, 4)
```

`show.message` *Show messages option in run.cache*

Description

Show messages option in run.cache

Usage

```
show.message(show.message = NULL)
```

Arguments

`show.message`	boolean indicating to show messages or not

Value

the show.message option

Examples

```
show.message(FALSE)
```

`tempdir.cache` *Temporary directory for runCache*

Description

Temporary directory for runCache

Usage

```
tempdir.cache()
```

Value

a path to a temporary directory used by runCache

Index