Package ‘loose.rock’

April 29, 2021

Title Tools for Survival Analysis and Data Science
Version 1.2.0
Description Collection of functions to improve workflow 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
NeedsCompilation no
RoxygenNote 7.1.1
Imports digest, futile.options, ggplot2, graphics, MASS, methods, httr, dplyr, reshape2
Suggests biomaRt, knitr, rmarkdown, roxygen2, testthat, survival
VignetteBuilder knitr

BugReports https://github.com/averissimo/loose.rock/issues
URL https://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 2021-04-29 17:20:02 UTC

R topics documented:

balanced.cv.folds ................................................................. 2
balanced.cv.folds.from.vector .............................................. 3
balanced.train.and.test ....................................................... 4
balanced.cv.folds

Create balanced folds for cross validation

Description

Create balanced folds for cross validation

Usage

balanced.cv.folds(..., nfolds = 10)
balanced.cv.folds.from.vector

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

balanced.cv.folds.from.vector

Create balanced folds for cross validation

Description

Create balanced folds for cross validation

Usage

balanced.cv.folds.from.vector(dat, nfolds = 10, join.all = FALSE)

Arguments

dat vectors representing data
nfolds number of folds to be created
join.all join foldids in a single vector

Value

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

Examples

dat <- sample(c(TRUE, FALSE), 150, replace = TRUE)
based.cv.folds.from.vector(dat, nfolds = 2)
based.cv.folds.from.vector(dat, nfolds = 10)
based.cv.folds.from.vector(dat, nfolds = 10, join.all = TRUE)
based.cv.folds.from.vector(dat[1:5], nfolds = 10) # will give a warning
balanced.cv.folds.from.vector(dat[1:10], nfolds = 10) # will give a warning
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,TRUE,TRUE,
FALSE,TRUE,TRUE,TRUE,TRUE,TRUE,TRUE,TRUE,TRUE,TRUE,TRUE,TRUE,
FALSE,TRUE)
set2 <- !set1
balanced.train.and.test(set1, set2, train.perc = .9)

balanced.train.and.test.from.vector

Get a balanced test and train dataset

Description

Get a balanced test and train dataset

Usage

balanced.train.and.test.from.vector(dat, train.perc = 0.9, join.all = TRUE)
Arguments

dat vector of different types in data
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

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

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')
**build.function.digest**  
*Build digest of function from the actual code*

**Description**  
Build digest of function from the actual code

**Usage**  
`build.function.digest(fun)`

**Arguments**  
- **fun**  
  function call name

**Value**  
a digest

**Examples**  
`loose.rock:::build.function.digest(sum)`  
`loose.rock:::build.function.digest(c)`

---

**cache.compression**  
*Change cache.compression for run.cache*

**Description**  
Change cache.compression for run.cache

**Usage**  
`cache.compression(compression = NULL)`

**Arguments**  
- **compression**  
  see compression parameter in save function

**Value**  
the new compression

**Examples**  
`cache.compression("bzip2")`
calculate.result  
Calculate/load result and save if necessary

Description
This is where the actual work is done

Usage
```
calculate.result(path, compression, force.recalc, show.message, fun, ...)
```

Arguments
- **path**  
  path to save cache
- **compression**  
  compression used in save
- **force.recalc**  
  force to recalculate cache
- **show.message**  
  boolean to show messages
- **fun**  
  function to be called
- ...  
  arguments to said function

Value
```
result of fun(...)
```

Examples
```
loose.rock:::calculate.result(
  file.path(tempdir(), 'calculate.result.Rdata'),
  'gzip',
  FALSE,
  TRUE,
  sum,
  1, 2, 3
)
```

ccds.genes.internal  
Download coding genes from CCDS

Description
```
```

Usage
```
ccds.genes.internal()
```
Value

vector of coding genes

Examples

```r
loose.rock:::ccds.genes.internal()
```

---

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

Description

It retrieves from NCBI and ensembl

Usage

```r
coding.genes(verbos = TRUE, useCache = TRUE, extra.verbos = FALSE)
```

Arguments

- **verbose**
  - show messages with number of genes retrieved
- **useCache**
  - Boolean indicating whether the results cache should be used. Setting to FALSE will disable reading and writing of the cache. This argument is likely to disappear after the cache functionality has been tested more thoroughly.
- **extra.verbos**
  - This will make all function calls verbose, which could be a lot of information.

Value

- a table with gene information

Examples

```r
# This can take a few minutes depending on the connection
res <- coding.genes()
nrow(res)
head(res)
```
coding.genes.ensembl

Ensembl coding genes, local function

Description
Ensembl coding genes, local function

Usage
coding.genes.ensembl( verbose = FALSE, useCache = TRUE )

Arguments
verbose show messages with number of genes retrieved
useCache Boolean indicating whether the results cache should be used. Setting to FALSE will disable reading and writing of the cache. This argument is likely to disappear after the cache functionality has been tested more thoroughly.

Value
a list with coding genes, mart and whether biomaRt had a problem, indicating that it shouldn’t be used.

Examples

res <- loose.rock:::coding.genes.ensembl(TRUE, TRUE)
nrow(res)
head(res)

create.directory.for.cache

Create directories for cache

Description
Create directories for cache

Usage
create.directory.for.cache(base.dir, parent.path)
Arguments

- base.dir: tentative base dir to create.
- parent.path: first 4 characters of digest that will become parent directory for the actual cache file (this reduces number of files per folder)

Value

- a list of updated base.dir and parent.dir

Examples

```r
loose.rock:::create.directory.for.cache(tempdir(), 'abcd')
## Not run:
loose.rock:::create.directory.for.cache(
  file.path(getwd(), 'run-cache'), 'abcd'
)
## End(Not run)
```

---

curl.workaround

Workaround for bug with curl when fetching specific ensembl mirror

Description

https://github.com/grimbough/biomaRt/issues/39

Usage

curl.workaround(expr, verbose = FALSE)

Arguments

- expr: expression
- verbose: if expression fails, then activates verbose on next call to curl.

Value

- result of expression

Examples

```r
## Not run:
loose.rock:::curl.workaround({
  biomaRt::useMart(
    biomart = "ensembl",
    dataset = 'hsapiens_gene_ensembl'
  )
})
## 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**

```r
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 \( \text{cov.matrix}[i,j] = \rho^{|i-j|} \)

Usage

```r
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

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

---

**getBM.internal**  
*Internal call to biomaRt::getBM*

Description

Depending on R version (<4.0.0) then it needs to have a special call

Usage

```r
getBM.internal(...)n
```

Arguments

- **...**: see biomaRt::getBM as all parameters are the same
getHsapiensMart.internal

Value
result of the biomaRt::getBM call

See Also
biomaRt::getBM

Examples

```r
mart <- loose.rock:::getHsapiensMart.internal()
res <- loose.rock:::getBM.internal(
  attributes = c("ensembl_gene_id","external_gene_name"),
  filters = 'biotype',
  values = c("protein_coding"),
  mart = mart,
  useCache = FALSE
)
nrow(res)
head(res)
```

getHsapiensMart.internal

*Get hsapiens mart from biomaRt*

Description
Get hsapiens mart from biomaRt

Usage

```r
gETCHSapiensMart.internal(
  verbose = FALSE,
  useCache = TRUE,
  domain = listEnsemblMirrors()
)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>verbose</td>
<td>add extra information in messages</td>
</tr>
<tr>
<td>useCache</td>
<td>use run.cache to speed up multiple calls</td>
</tr>
<tr>
<td>domain</td>
<td>vector of possible domains to call biomaRt</td>
</tr>
</tbody>
</table>

Value
biomaRt hsapiens mart
join.ensembl.and.ccds

Join genes from ensembl and ccds in a single table

Description

[INTERNAL] Finds the ensembl ids of genes from ccds

Usage

join.ensembl.and.cccds(
  ensembl.genes,
  ccds.genes,
  mart,
  useCache = TRUE,
  verbose = FALSE
)

Arguments

ensembl.genes  protein coding genes from ensembl
ccds.genes    protein coding genes from ccds
mart           biomaRt dataset to use
useCache       should biomart use cache
verbose        show extra messages

Value

table with ensembl_gene_id and external_gene_name columns
**listEnsemblMirrors**

*Internal function to list mirrors*

### Description

Internal function to list mirrors

### Usage

```r
listEnsemblMirrors()
```

### Value

list of ensembl mirrors

### Examples

```r
loose.rock:::listEnsemblMirrors()
```

---

**loose.rock**

*loose.rock: compilation of data science functions*

### 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.

### Usage

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

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'.

Value

futile.options::OptionsManager object

See Also

futile.options

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

```r
# 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 palette of colors*

### Description

Custom palette of colors

### Usage

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

### Arguments

- `ix` index for a color

### Value

a color

### Examples

```r
my.colors()
my.colors(5)
```
**my.symbols**

*Custom palette of symbols in plots*

**Description**

Custom palette of symbols in plots

**Usage**

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

**Arguments**

- **ix**
  - index for symbol

**Value**

- a symbol

**Examples**

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

---

**proper**

*Capitalizes all words in string*

**Description**

Capitalizes all words in string

**Usage**

```r
proper(x)
```

**Arguments**

- **x**
  - String

**Value**

- a capitalized string (all words)

**Examples**

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

Questions to ask when calling devtools::release()  

Description

This should be done when submitting to cran  

Usage

release_questions()  

Value

vector of questions

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

```r
# [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)
```

---

**Description**

Run function and save cache

**Usage**

```r
## 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
)
```
**save.run.cache**

**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**

```r
# [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)
```

**Description**

Saving the cache

**Usage**

```r
save.run.cache(result, path, compression, show.message)
```
**Arguments**

- **result**: main result to save
- **path**: path to the file to save
- **compression**: compression method to be used
- **show.message**: TRUE to show messages, FALSE otherwise

**Value**

result of save operation

**Examples**

```r
loose.rock:::save.run.cache(
  35, file.path(tempdir(), 'save.run.cache.Rdata'), FALSE, TRUE
)
```

---

**search.genes.internal**  
_Search genes in biomaRt_

**Description**

Search genes in biomaRt

**Usage**

```r
search.genes.internal(filters, values, mart, useCache = TRUE, verbose = FALSE)
```

**Arguments**

- **filters**: see biomaRt::getBM
- **values**: see biomaRt::getBM
- **mart**: see biomaRt::getBM
- **useCache**: see biomaRt::getBM
- **verbose**: see biomaRt::getBM

**Value**

data table with attributes as columns
Examples

```r
mart <- loose.rock:::getHsapiensMart.internal()
loose.rock:::search.genes.internal(
  'entrezgene_accession', 'HHLA3', mart, useCache = FALSE
)
loose.rock:::search.genes.internal(
  'external_gene_name', 'BRCA2', mart, useCache = FALSE
)
```

Description

Show messages option in run.cache

Usage

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

Arguments

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

Value

the show.message option

Examples

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

tempdir.cache

Temporary directory for runCache

Description

Temporary directory for runCache

Usage

```r
tempdir.cache()
```

Value

a path to a temporary directory used by runCache
**write.readme**

Write a file in run-cache directory to explain the origin

**Description**

Write a file in run-cache directory to explain the origin

**Usage**

```plaintext
write.readme(base.dir)
```

**Arguments**

- `base.dir`: directory where to build this file

**Examples**

```plaintext
loose.rock:::write.readme(tempdir())
```
Index

balanced.cv.folds, 2
balanced.cv.folds.from.vector, 3
balanced.train.and.test, 4
balanced.train.and.test.from.vector, 4
base.dir, 5
build.function.digest, 6
cache.compression, 6
calculate.result, 7
ccds.genes.internal, 7
coding.genes, 8
coding.genes.ensembl, 9
create.directory.for.cache, 9
curl.workaround, 10
digest.cache, 11
draw.cov.matrix, 11
gen.synth.xdata, 12
getBM.internal, 12
getHsapiensMart.internal, 13
join.ensembl.and.ccds, 14
listEnsemblMirrors, 15
loose.rock, 15
loose.rock.options, 15
multiplot, 16
my.colors, 17
my.symbols, 18
proper, 18
release_questions, 19
run.cache, 19
run.cache.function-method, 20
save.run.cache, 21
search.genes.internal, 22
show.message, 23
tempdir.cache, 23
write.readme, 24