Package 'CornerstoneR'

February 12, 2019

Version 1.1.1

Title Collection of Scripts for Interface Between 'Cornerstone' and 'R'

Description

Collection of generic 'R' scripts which enable you to use existing 'R' routines in 'Cornerstone'.

The desktop application 'Cornerstone' (<https://www.camline.com/en/products/cornerstone/cornerstone-core.html>) is a data analysis software provided by 'camLine' that empowers engineering teams to find solutions even faster.
The engineers incorporate intensified hands-on statistics into their projects.
They benefit from an intuitive and uniquely designed graphical Workmap concept: you design experiments (DoE) and explore data, analyze dependencies, and find answers you can act upon, immediately, interactively, and without any programming.

While 'Cornerstone's' interface to the statistical programming language 'R' has been available since version 6.0, the latest interface with 'R' is even much more efficient.
'Cornerstone' release 7.1.1 allows you to integrate user defined 'R' packages directly into the standard 'Cornerstone' GUI.
Your engineering team stays in 'Cornerstone's' graphical working environment and can apply 'R' routines, immediately and without the need to deal with programming code.

Learn how to use 'R' packages in 'Cornerstone' 7.1.1 on 'camLineTV' YouTube channel (<https://www.youtube.com/watch?v=HEQHwq_laXU>) (available in German).

URL https://gitlab.com/camLine/CornerstoneR

BugReports https://gitlab.com/camLine/CornerstoneR/issues

License GPL-3

Encoding UTF-8

Depends R (>= 3.2.1)

Imports checkmate (>= 1.9.1) , data.table (>= 1.10) , ranger , vcd

Suggests knitr , rmarkdown , roxygen2 , testthat

ByteCompile yes

LazyData yes

RoxygenNote 6.1.1

NeedsCompilation no

Author Dirk Surmann [aut, cre] (<https://orcid.org/0000-0003-0873-137X>)

Maintainer Dirk Surmann <dirk.surmann@versuchsplanung.de>

Repository CRAN

Date/Publication 2019-02-12 17:10:03 UTC

R topics documented:

`CornerstoneR-package` *CornerstoneR: Collection of Scripts for Interface Between 'Cornerstone' and 'R'*

Description

Collection of generic 'R' scripts which enable you to use existing 'R' routines in 'Cornerstone'. — The desktop application 'Cornerstone' (<https://www.camline.com/en/products/cornerstone/cornerstone-core.html>) is a data analysis software provided by 'camLine' that empowers engineering teams to find solutions even faster. The engineers incorporate intensified hands-on statistics into their projects. They benefit from an intuitive and uniquely designed graphical Workmap concept: you design experiments (DoE) and explore data, analyze dependencies, and find answers you can act upon, immediately, interactively, and without any programming. — While 'Cornerstone's' interface to the statistical programming language 'R' has been available since version 6.0, the latest interface with 'R' is even much more efficient. 'Cornerstone' release 7.1.1 allows you to integrate user defined 'R' packages directly into the standard 'Cornerstone' GUI. Your engineering team stays in 'Cornerstone's' graphical working environment and can apply 'R' routines, immediately and without the need to deal with programming code. — Learn how to use 'R' packages in 'Cornerstone' 7.1.1 on 'camLineTV' YouTube channel (<https://www.youtube.com/watch?v=HEQHwq_laXU>) (available in German).

Author(s)

Maintainer: Dirk Surmann <dirk.surmann@versuchsplanung.de> (0000-0003-0873-137X)

See Also

Useful links:

- https://gitlab.com/camLine/CornerstoneR
- Report bugs at https://gitlab.com/camLine/CornerstoneR/issues

---

`carstats` *Data from carstats*

Description

Copy from the carstats sample dataset available in Cornerstone.

Usage

```
carstats
```

Format

A `data.table` object with 406 rows and 9 columns.

`fitFunction` *Fit Function to Data via nls*

Description

Fit predefined functions to data via nonlinear least squares using `nls`.

Usage

```
fitFunction(dataset = cs.in.dataset(), preds = cs.in.predictors(),
  resps = cs.in.responses(), groups = cs.in.groupvars(),
  auxs = cs.in.auxiliaries(), scriptvars = cs.in.scriptvars(),
  return.results = FALSE, ...)
```

Arguments

`dataset`	[`data.frame`] Dataset with named columns. The names correspond to predictors and responses.
`preds`	[`character`] Character vector of predictor variables.
`resps`	[`character`] Character vector of response variables.

groups	[character] Character vector of group variables.
auxs	[character] Character vector of auxiliary variables.
scriptvars	[list] Named list of script variables set via the Cornerstone "Script Variables" menu. For details see below.
return.results	[logical(1)] If FALSE the function returns TRUE invisibly. If TRUE, it returns a list of results. Default is FALSE.
...	[ANY] Additional arguments to be passed to nls . Please consider possible script variables (scriptvars) to prevent duplicates.

Details

The following script variables are summarized in scriptvars list:

math.fun [character(1)]
Function selection for fitting data. It is possible to choose a predefined model, or compose a model manually by selecting User Defined.
Default is User Defined

preds.frml [character(1)]
Only required if math.fun is set to User Defined. Valid R formula for the right hand side (predictors) of the model equation.

resp.frml [character(1)]
Only required if math.fun is set to User Defined. Valid R formula for the left hand side (response) of the model equation.

start.vals [character(1)]
Only required if math.fun is set to User Defined. Specify starting values for all terms of the right hand side as a comma separated list with a period as decimal separator.

weights [character(1)]
Select a weighting variable from the auxiliary variables.

algo.nls [character(1)]
Specifies the algorithm to use. For details see nls.
Default is plinear.

Value

Logical [TRUE] invisibly or, if return.results = TRUE, list of resulting data.frame objects:

coeff	Estimated coefficients and standard errors for every variable.
predictions	Brushable dataset with predictions and residuals added to original values and groups, if available.

Examples

```
# Generate data from logistic function:
fun = function(x, a, b, c, d, sigma = 1) {
  a+(b-a) / (1+exp(-d*(x-c))) + rnorm(length(x), sd = sigma)
  }
library(data.table)
dt = data.table(  x1 = sample(seq(-10, 10, length.out = 100))
                , group1 = sample(x = c("A", "B"), replace = TRUE, size = 100)
                )
dt[group1 == "A", y1 := fun(x1, 1, 10, 1, 0.6, 0.1)]
dt[group1 == "B", y1 := fun(x1, 8, 2, -1, 0.3, 0.1)]
# Set script variables
scriptvars = list(math.fun = "Logistic", resp.frml = "", preds.frml = ""
                , start.vals = "", weights = "", algo.nls = "default"
                )
# Fit the logistic function:
res = fitFunction(dt, "x1", "y1", "group1", "", scriptvars, TRUE)
# Show estimated coefficients:
res$coeff
# Plot fitted vs. residuals:
plot(res$predictions$Fitted, res$predictions$Residuals)
```

LocalInterface *Local Interface Functions*

Description

CS-R interface functions are defined in package namespace via this file. Each function overwrites itself with the corresponding counterpart defined in the global environment from CS.

Usage

```
invokeFromR()

cs.in.auxiliaries(quote = FALSE)

cs.in.brushed()

cs.in.dataset()

cs.in.excluded()

cs.in.groupvars(quote = FALSE)

cs.in.predictors(quote = FALSE)

cs.in.responses(quote = FALSE)
```

```
cs.in.Robject(name = NA)

cs.in.scriptvars(name = NA)

cs.in.subsets()

cs.in.subsets.current()

cs.quote(x)

cs.out.dataset(data, name = NA, brush = FALSE)

cs.out.emf(name = NULL, width = 10, height = 10)

cs.out.png(name = NULL, width = 10, height = 10)

cs.out.Robject(R_object, name = NA)
```

Arguments

quote	[logical(1)] Quote all variables to cover invalid names. Use make.names as an alternative.
name	[character(1)] Name for output to Cornerstone.
x	[character(1)] String to check for invalid characters related to make.names. Add backticks, if necessary.
data	[data.frame] Dataset with named columns. The names correspond to predictors and responses.
brush	[logical(1)] Brushing of output dataset in Cornerstone across the R object.
width	[numeric(1)] Width of exported plotting object. See pdf.
height	[numeric(1)] Height of exported plotting object. See pdf.
R_object	[list] List of exported R objects to Cornerstone.

---

mosaicPlot *Mosaic Plot*

Description

Plot extended mosaic via mosaic.

Usage

```
mosaicPlot(dataset = cs.in.dataset(), preds = cs.in.predictors(),
  resps = cs.in.responses(), ...)
```

Arguments

dataset	[data.frame] Dataset with named columns. The names correspond to predictors and responses.
preds	[character] Character vector of predictor variables.
resps	[character] Character vector of response variables.
...	[ANY] Additional arguments to be passed to mosaic . Please consider possible script variables (scriptvars) to prevent duplicates.

Examples

```
# Draw mosaic plot from 'Titanic' data:
mosaicPlot(as.data.frame(Titanic), c("Class", "Sex", "Age", "Survived"), "Freq")
```

---

randomForest *Random Forest*

Description

Random Forest via ranger. Predicts response variables or brushed set of rows from predictor variables, using Random Forest classification or regression.

Usage

```
randomForest(dataset = cs.in.dataset(), preds = cs.in.predictors(),
  resps = cs.in.responses(), brush = cs.in.brushed(),
  scriptvars = cs.in.scriptvars(), return.results = FALSE, ...)
```

Arguments

dataset	[data.frame] Dataset with named columns. The names correspond to predictors and responses.
preds	[character] Character vector of predictor variables.
resps	[character] Character vector of response variables.

brush	[logical] Logical vector of length nrow(dataset). Flags brushed rows in Cornerstone.
scriptvars	[list] Named list of script variables set via the Cornerstone "Script Variables" menu. For details see below.
return.results	[logical(1)] If FALSE the function returns TRUE invisibly. If TRUE, it returns a list of results. Default is FALSE.
...	[ANY] Additional arguments to be passed to ranger . Please consider possible script variables (scriptvars) to prevent duplicates.

Details

The following script variables are summarized in scriptvars list:

brush.pred [logical(1)]
Use brush vector as additional predictor.
Default is FALSE.

use.rows [character(1)]
Rows to use in model fit. Possible values are all, unbrushed, or brushed.
Default is all.

num.trees [integer(1)]
Number of trees to fit in ranger.
Default is 500.

importance.mode [character(1)]
Variable importance mode. For details see ranger.
Default is permutation.

respect.unordered.factors [character(1)]
Handling of unordered factor covariates. For details see ranger.
Default is NULL.

Value

Logical [TRUE] invisibly or, if return.results = TRUE, list of resulting data.frame objects:

statistics	General statistics about the random forest.
importances	Variable importances of prediction variables in descending order of importance (most important first)
predictions	Brushable dataset with predicted values for dataset. The original input and other columns can be added to this dataset through the menu Columns -> Add from Parent ....
confusion	For categorical response variables or brush state only. A table with counts of each distinct combination of predicted and actual values.
rgobjects	List of ranger.forest objects with fitted random forests.

See Also

randomForestPredict

Examples

```
# Fit random forest to iris data:
res = randomForest(iris, c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width"), "Species"
                , scriptvars = list(brush.pred = FALSE, use.rows = "all", num.trees = 500
                                    , importance.mode = "permutation"
                                    , respect.unordered.factors = "NULL"
                                    )
                   , brush = rep(FALSE, nrow(iris)), return.results = TRUE
                  )
# Show general statistics:
res$statistics
# Prediction
randomForestPredict(iris[, 1:4], c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width")
                    , robject = res$rgobjects
                    , return.results = TRUE
                    )
```

---

randomForestPredict *Random Forest Prediction*

Description

Random Forest prediction via predict.ranger. Predicts response variables from predictor variables, using ranger objects. All ranger objects have to work on the same set of prediction variables. These variables are exactly available in the prediction dataset. A response is not necessary, it will be predicted via this function.

Usage

```
randomForestPredict(dataset = cs.in.dataset(),
  preds = cs.in.predictors(), robject = cs.in.Robject(),
  return.results = FALSE, ...)
```

Arguments

dataset	[data.frame] Dataset with named columns. The names correspond to predictors and responses.
preds	[character] Character vector of predictor variables.
robject	[list] Named list of ranger objects set via Cornerstone menu "Input R Objects".

`return.results` [logical(1)]
If FALSE the function returns TRUE invisibly. If TRUE, it returns a list of results. Default is FALSE.

`...` [ANY]
Additional arguments to be passed to ranger . Please consider possible script variables (scriptvars) to prevent duplicates.

Value

Logical [TRUE] invisibly or, if `return.results = TRUE`, list of resulting data.frame objects:

`predictions` Brushable dataset with predicted values for dataset. The original input and other columns can be added to this dataset through the menu `Columns -> Add from Parent ...`.

See Also

randomForest

---

reshapeLong *Reshape Grouped Data to Long*

Description

Reshaping grouped data via melt to 'long' format. The responses are merged in one column, with its column name in an additional column. This column is split into multiple columns, if a split character is given. All predictors are merged multiple times corresponding to the number or responses.

Usage

```
reshapeLong(dataset = cs.in.dataset(), preds = cs.in.predictors(),
  resps = cs.in.responses(), scriptvars = cs.in.scriptvars(),
  return.results = FALSE, ...)
```

Arguments

`dataset` [data.frame]
Dataset with named columns. The names correspond to predictors and responses.

`preds` [character]
Character vector of predictor variables.

`resps` [character]
Character vector of response variables.

`scriptvars` [list]
Named list of script variables set via the Cornerstone "Script Variables" menu. For details see below.

`return.results` [logical(1)]
If FALSE the function returns TRUE invisibly. If TRUE, it returns a list of results. Default is FALSE.

`...` [ANY]
Additional arguments to be passed to melt . Please consider possible script variables (scriptvars) to prevent duplicates.

Details

One script variables is summarized in scriptvars list:

split [character(1)]
Split character to split response names into multiple columns. Default is "_".

Value

Logical [TRUE] invisibly or, if return.results = TRUE, list of resulting data.frame object:

`reshapeLong` Dataset with reshaped data.

Examples

```
# Data to transform:
library(data.table)
dtTest = data.table(i_1 = c(1:4, NA, 5), i_2 = c(51, 61, NA , 71, 81, 91)
                    , f1 = factor(sample(c(letters[1:3], NA), 6, TRUE))
                    , f2 = factor(c("z", "a", "x", "c", "x", "x"), ordered = TRUE)
                    )
# Reshape to long format:
reshapeLong(dtTest, c("i_1", "i_2"), c("f1", "f2"), list(split = "_"), return.results = TRUE)
```

---

reshapeWide *Reshape Grouped Data to Wide*

Description

Reshaping grouped data via dcast to 'wide' format with rows for each unique combination of group variables. The response are arranged in separate columns for each datum in predictors. If a combination of groups identifies multiple rows, the number of rows in a group is returned to CS for the whole dataset instead of the response variable value.

Usage

```
reshapeWide(dataset = cs.in.dataset(), preds = cs.in.predictors(),
  resps = cs.in.responses(), groups = cs.in.groupvars(),
  scriptvars = cs.in.scriptvars(), return.results = FALSE, ...)
```

Arguments

dataset	[data.frame] Dataset with named columns. The names correspond to predictors and responses.
preds	[character] Character vector of predictor variables.
resps	[character] Character vector of response variables.
groups	[character] Character vector of group variables.
scriptvars	[list] Named list of script variables set via the Cornerstone "Script Variables" menu. For details see below.
return.results	[logical(1)] If FALSE the function returns TRUE invisibly. If TRUE, it returns a list of results. Default is FALSE.
...	[ANY] Additional arguments to be passed to dcast . Please consider possible script variables (scriptvars) to prevent duplicates.

Details

One script variables is summarized in scriptvars list:

nodrop [logical(1)]
Drop missing combinations (FALSE) or include all (TRUE). Default is FALSE.

Value

Logical [TRUE] invisibly or, if return.results = TRUE, list of resulting data.frame object:

reshapeWide	Dataset with reshaped data.

Examples

```
# Reshape dataset to wide format:
reshapeWide(Indometh, "Subject", "time", "conc", list(nodrop = FALSE), return.results = TRUE)
```

Index