Package 'CornerstoneR'

June 2, 2020

Version 2.0.1

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.
Additionally, your 'R' programming team develops corresponding 'R' packages detached from 'Cornerstone' in their favorite 'R' environment.
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

Language en-US

License GPL-3

Encoding UTF-8

Depends R (>= 3.2.1)

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

Suggests ggplot2 , knitr , prettydoc , rmarkdown , roxygen2 , testthat

ByteCompile yes

LazyData yes

RoxygenNote 7.1.0

VignetteBuilder knitr

NeedsCompilation no

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

Maintainer Dirk Surmann <dirk.surmann@camline.academy>

Repository CRAN

Date/Publication 2020-06-02 10:30:02 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. Additionally, your 'R' programming team develops corresponding 'R' packages detached from 'Cornerstone' in their favorite 'R' environment.

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

Author(s)

Maintainer: Dirk Surmann <dirk.surmann@camline.academy> (ORCID)

See Also

Useful links:

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

---

`carstats` *Data from carstats*

Description

Dataset of different cars and various values.

Format

A `data.table` object with 406 observations and 9 variables. The variables and their scale of measurement are as follows:

- Model: nominal
- Origin: nominal
- MPG: interval
- Cylinders: ordinal
- Displacement: interval
- Horsepower: interval
- Weight: interval
- Acceleration: interval
- Model.Year: interval

Source

Cornerstone sample dataset

fitFunction *Fit Function to Data via Nonlinear Regression*

Description

Fit predefined functions to data via nonlinear least squares using Levenberg-Marquardt algorithm via nlsLM.

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)]
: 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)]
: Required if `math.fun` is set to `User Defined`. Valid R `formula` for the left hand side (response) of the model equation.

limits [character(1)]
: Optional if `math.fun` is set to `User Defined`. Specifies minimum and maximum value for function `math.fun` as a comma separated list of `min` and `max`. It is possible to assign variables, e.g. `min=a`, which need start values in `start.vals`, as well as real numbers, e.g. `min=4.5`, with a period as decimal separator.

start.vals [character(1)]
: 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.

max.iter Maximum number of iterations. For details see `link[minpack.lm]{nls.lm.control}`

max.ftol Maximum relative error desired in the sum of squares. If `0`, the default is used. For details see `link[minpack.lm]{nls.lm.control}`

Value

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

`coeff`	Estimated coefficients and standard errors for each group. Convergence information is available for every group (for details see `link[minpack.lm]{nls.lm}`).
`vcov`	Variance-Covariance matrix of the main coefficients for the fitted model of each group (for details see `link[stats]{vcov}`).
`predictions`	Dataset to brush 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 = "", limits = ""
                , start.vals = "", weights = "", max.iter = 50, max.ftol = 0
                )
# Fit the logistic function:
res = fitFunction(dt, "x1", "y1", "group1", character(0), scriptvars, TRUE)
# Show estimated coefficients:
res$coeff
# Variance-Covariance matrix:
res$vcov
# 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 = 480, height = 480)

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.

---

matchNearestNeighbor *Match Nearest Neighbor Between Two Datasets*

Description

Match the nearest neighbor from a redirected Cornerstone Robject dataset (redirectDataset) to corresponding selected predictor variables. Predictor variables from both datasets are supposed to be numeric to apply the Euclidean distance calculated by dist2. The function returns a dataset with the nearest neighbor to every observation, matched by the predictor variables. Available response, group, and auxiliary variables from the redirected datasets are passed through, as well as, selected auxiliary variables. The calculated distance is attached.

Usage

```
matchNearestNeighbor(
  dataset = cs.in.dataset(),
  preds = cs.in.predictors(),
  auxs = cs.in.auxiliaries(),
  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.
auxs	[character] Character vector of auxiliary variables.
robject	[list] Named list of one redirectDataset object(s) 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.

Value

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

nearest.neighbor	Matched nearest neighbor which consists of predictor and available response, group, and auxiliary variables. The calculated distance is attached to this dataset.
runtimes	Run times for every input R object.

---

mosaicPlot *Mosaic Plot*

Description

Plots (extended) mosaic displays via mosaic. The last response variable is highlighted. A high-dimensional contingency table is calculated via structable from the given dataset. Flat contingency table splits predictors horizontally and optional responses vertically.

Usage

```
mosaicPlot(
  dataset = cs.in.dataset(),
  preds = cs.in.predictors(),
  resps = cs.in.responses(),
  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.
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 mosaic . Please consider possible script variables (scriptvars) to prevent duplicates.

Value

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

long.contingency
: Contingency table in long format.

Examples

```
# Draw mosaic plot from 'titanic' data:
mosaicPlot(titanic, c("Class", "Age", "Sex", "Survived"))
res = mosaicPlot(titanic, c("Class", "Age"), c("Sex", "Survived"), return.results = TRUE)
print(res)
```

---

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, non-brushed, 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 and outputs to Cornerstone or, if return.results = TRUE, list of resulting data.frame objects:

statistics	General statistics about the random forest.
importances	Variable importance of prediction variables in descending order of importance (most important first)
predictions	Dataset to brush 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 = "ignore"
                                    )
                  , 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 object(s) 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 and outputs to Cornerstone or, if return.results = TRUE, list of resulting data.frame objects:

predictions	Dataset to brush 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

redirectDataset *Redirect Dataset*

Description

Redirect input dataset to an output R object.

Usage

```
redirectDataset(
  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.

Details

The following script variables are summarized in scriptvars list:

remove.pattern [character(1)]
: The given pattern is removed in all variable names via gsub. Leading and / or trailing whitespaces are removed using trimws. Default is "".

Value

Logical [TRUE] invisibly and outputs to Cornerstone cs.out.Robject or, if return.results = TRUE, list of resulting data.frame objects and character(n) vectors:

dataset	Input dataset.
predictors	Vector of predictors.
responses	Vector of responses.
groups	Vector of groups.
auxiliaries	Vector of auxiliaries.

The list is wrapped in an additional list to get the same return value corresponding to cs.in.Robject.

---

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 and outputs to Cornerstone 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)
```

---

reshapeTranspose *Transpose Data*

Description

Transpose data via transpose. All predictors, responses, groups, and auxiliaries are transpose.

Usage

```
reshapeTranspose(
  dataset = cs.in.dataset(),
  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.
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 transpose . 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 and outputs to Cornerstone or, if return.results = TRUE, list of resulting data.frame object:

reshapeTranspose
Dataset with transposed 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))
# Reshape to long format:
reshapeTranspose(dtTest, groups = character(0), list(convert.numeric = TRUE), 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(),
  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 dcast . Please consider possible script variables (scriptvars) to prevent duplicates.

Details

One script variables is summarized in `scriptvars` list:

drop [logical(1)]
Drop missing combinations (TRUE) or include all (FALSE). Default is TRUE.
For details see dcast.

Value

Logical [TRUE] invisibly and outputs to Cornerstone or, if `return.results` = TRUE, list of resulting data.frame object:

`reshapeWide` Dataset with reshaped data.

Examples

```
# Reshape dataset to wide format:
reshapeWide(Indometh, "time", "conc", "Subject", character(0)
            , list(drop = TRUE, aggr.fun = "mean"), return.results = TRUE
            )
```

---

rundata *Voltage of 26 Different Runs*

Description

The datasets contains 26 different runs in each case measuring the voltage over time. The S-shaped curves are different for each run.

Format

A data.table object with 7826 observations and 3 variables. The discrete variable 'FileName' has 26 levels corresponding to the file name of the source data from Cornerstone. The remaining other two variables are continuous.

Source

Cornerstone sample dataset

showVersions *Show Versions of R and CornerstoneR*

Description

Write the versions of R and CornerstoneR in a Cornerstone dataset.

Usage

```
showVersions(return.results = FALSE)
```

Arguments

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

Value

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

versions	Dataset with versions of R and CornerstoneR.

Examples

```
res = showVersions(return.results = TRUE)
res$versions
```

titanic *Survival of Passengers on the Titanic*

Description

This data set provides information on the fate of passengers on the fatal maiden voyage of the ocean liner Titanic, summarized according to economic status (Class), Age, Sex, and Survival.

Format

A data.table object with 2201 observations and 4 variables. The discrete variables and their levels are as follows:

- Class: 1st, 2nd, 3rd, Crew
- Age: Child, Adult
- Sex: Male, Female
- Survived: Yes, No

Source

Cornerstone sample dataset

References

Dawson, Robert J. MacG. (1995), The Unusual Episode Data Revisited. *Journal of Statistics Education*, **3**. https://www.amstat.org/publications/jse/v3n3/datasets.dawson.html

Index