Package 'StudyDataTools'

October 17, 2018

Type Package

Title X

Version 1.0

Date 2018-10-17

Author D. Bonnery

Maintainer D. Bonnery `<dbonnery@umd.edu>`

Description Data

Remotes tidyverse/magrittr

Depends RODBC,
ggplot2,
sqldf,
lattice,
printr,
knitr,
reshape2,
rlist,
devtools,
haven,
sas7bdat

License GPL (>= 2)

LazyLoad yes

LazyData true

RoxygenNote 6.0.1

R topics documented:

`automaticdatafConnect` *get data about a file on the server.*

Description

get data about a file on the server.

Usage

```
automaticdatafConnect(tablename, folder = getwd(), schema = NULL,
  dicoT = NULL, splitvar = NULL, Connect = NULL, Connectf = NULL,
  alwaysexclude = NULL)
```

Value

a list

`drop_last` *Drop last marginpos (cuts everything after last after "_")*

Description

Drop last marginpos (cuts everything after last after "_")

Usage

```
drop_last(x)
```

Arguments

`x`	a vector of character strings

Details

if x is "aa.x_a_1_f_1" returns "a_1_f_1"

Value

a vector of character strings

Examples

```
drop_last("AA.char1_La_Ld_Lrn1")
```

`exportRpackagedata_to_csvlibrary` *export all tables from an installed R data package to sas7bdat data files in a given library*

Description

export all tables from an installed R data package to sas7bdat data files in a given library

Usage

```
exportRpackagedata_to_csvlibrary(package, path_to_export_to = getwd(),
  tables = NULL, zip = TRUE)
```

Arguments

`package`	package to be transformed
`path_to_export_to`	path to export the data
`tables`	tables to extract, if NULL, all tables.
`zip`	(should the files be zipped (requires utils package))

Value

nothing

Examples

```
exportRpackagedata_to_csvlibrary("datasets")
```

`exportRpackagedata_to_sas7bdatlibrary` *export all tables from an installed R data package to sas7bdat data files in a given library*

Description

export all tables from an installed R data package to sas7bdat data files in a given library

Usage

```
exportRpackagedata_to_sas7bdatlibrary(package, path_to_export_to = getwd(),
  tables = NULL)
```

Arguments

`package`	package to be transformed
`SAS_library_path`	path to export the data

Value

nothing

Examples

```
exportRpackagedata_to_sas7bdatlibrary("datasets")
```

`exportRpackagedata_to_savlibrary` *export all tables from an installed R data package to sav data files in a given library*

Description

export all tables from an installed R data package to sav data files in a given library

Usage

```
exportRpackagedata_to_savlibrary(package, path_to_export_to = getwd(),
  tables = NULL, zip = TRUE)
```

Arguments

`package`	package to be transformed
`path_to_export_to`	path to export the data
`tables`	tables to extract, if NULL, all tables.
`zip`	(should the files be zipped (requires utils package))

Value

nothing

Examples

```
exportRpackagedata_to_csvlibrary("datasets")
```

`GeneralReversetransposefunction` *General Reverse Transpose function*

Description

General Reverse Transpose function

Usage

```
GeneralReversetransposefunction(TtableA, key)
```

Arguments

`key`	A list of variables (columns of the transposed table)
`table`	A dataframe

Value

A list: first element of the list is a dataframe, the transposed version of the orioginal table. Second element is a key to allow back transposition

Examples

```
data(tableA);data(TtableA);data(XKA);key<-XKA$key
RtableA=GeneralReversetransposefunction(TtableA,key)
ordertableA <-do.call(order,tableA[c(id1,id2)])
orderRtableA<-do.call(order,RtableA[c(id1,id2)])
identical(nrow(tableA),nrow(RtableA))
identical(lapply(tableA,class),lapply(RtableA,class))
identical(tableA[ordertableA,],RtableA[orderRtableA,])
identical(names(tableA),names(RtableA))
all (lapply(names(tableA),function(x){identical(tableA[ordertableA,x],RtableA[orderRtableA,x])}))
```

`Generaltransposefunction`

General Transpose function

Description

General Transpose function

Usage

```
Generaltransposefunction(tableA, id1, id2,
  origin = deparse(substitute(tableA)))
```

Arguments

`id1`	A list of variables (rows)
`id2`	A list of variables (columns of the transposed table), id2 can contain as a last element the strint "rn", if the variable rn is an index for the cells formed by the variables listed first in id2
`table`	A dataframe

Value

A list: first element of the list is a dataframe, the transposed version of the orioginal table. Second element is a key to allow back transposition

Examples

```
tableA<-sampledata(TRUE)
id1=c("id1a","id1b")
id2=c("id2a","id2b")
TtableA<-Generaltransposefunction(tableA,id1,id2)
```

`Generaltransposefunctionsimple` *Simple General Transpose function*

Description

Simple General Transpose function

Usage

```
Generaltransposefunctionsimple(tableA, id1, id2)
```

Arguments

`tableA`	A dataframe
`id1`	A list of variables (rows)
`id2`	A list of variables (columns of the transposed table)

Value

A data frame

Examples

```
tableA<-sampledata(TRUE)
id1=c("id1a","id1b")
id2=c("id2a","id2b")
TtableA<-Generaltransposefunctionsimple(tableA,id1,id2)
```

`get_cell` *get cell without the row number*

Description

get cell without the row number

Usage

```
get_cell(x, iscellrn = FALSE, iscell = FALSE)
```

Arguments

`x`	a vector of character strings

Details

if x is "aa.xoijj_a_1_f_1_" returns "a_1_f"

Value

a vector of character strings

Examples

```
get_cell("aa.x_1_2_3_4")#default
get_cell("1_2_3",TRUE)
get_cell("1_2_3",FALSE,TRUE)
unique(Tsampledata(TRUE)$variables))
unique(get_cell(Tsampledata(FALSE)$variables))
```

`get_cellrn` *Get cell and row number*

Description

Get cell and row number

Usage

```
get_cellrn(x)
```

Arguments

`x`	a vector of character strings

Details

if x is "aa.x_a_1_f_1" returns "a_1_f_1"

Value

a vector of character strings

Examples

```
get_cellrn("AA.char1_La_Ld_Lrn1")
data(TtableA);
unique(get_cellrn(names(TtableA)))
#Second example: no transposing variables
data(TtableB);data(XKB)
unique(get_cellrn(names(XKB)))
```

get_cellXXgroup *Get cell group*

Description

Get cell group

Usage

```
get_cellXXgroup(x, marginpos, iscellXX = TRUE)
```

Arguments

x	a vector of character strings
marginpos	a vector of integer

Details

if x is "a_1_f_2_aa.xoijj",marginpos=2 returns "1" if x is "a_1_f_2_aa.xoijj",marginpos=-2 returns "a_f_2" if x is "a_1_f_2_aa.xoijj",marginpos=c(1:2) returns "a_1"

Value

a vector of character strings

Examples

```
get_cellXXgroup(c("aa.x_1_2_3_4","bb.x_1_2_3_4"),2,iscellXX=FALSE)
get_cellXXgroup(c("1_2_3_4","1_2_3_4"),2:3,iscellXX=TRUE)
variables<-Tsampledata(TRUE)$variables
unique(get_cellXXgroup(variables,2,iscellXX=FALSE))
unique(get_cellXXgroup(variables,-2,iscellXX=FALSE))
get_cellXXgroup(variables[50],2,iscellXX=FALSE)
get_cellXXgroup(variables[50],-2,iscellXX=FALSE)

#Second example: no transposing variables
TK<-Tsampledata(FALSE)
unique(get_cellXXgroup(TK$variable,1,iscell=FALSE))
```

get_cellXXmarginscount *Get the number of margins for a cell*

Description

Get the number of margins for a cell

Usage

```
get_cellXXmarginscount(x, iscellXX = FALSE)
```

Arguments

x	a vector of character strings
iscell	a boolean indicating if x is a variable name or a cell name.

Details

if x is "aa.xoijj_a_1_f_1", cell=FALSE returns 4" if x is "a_1_f_1", cell=TRUE returns 4"

Value

a vector of integers.

Examples

```
get_cellXXmarginscount("1_2_3_4",iscellXX=TRUE)
get_cellXXmarginscount("aa.x_1_2_3_4",iscellXX=FALSE)
data(TtableA)
unique(get_cellXXmarginscount(names(TtableA),iscellXX=FALSE))
#Second example: no transposing variables
TK<-Tsampledata(FALSE)
unique(get_cellXXmarginscount(TK$variables))
```

get_cellXXsplit *split a cell*

Description

split a cell

Usage

```
get_cellXXsplit(x, marginpos = NULL, iscellXX = FALSE)
```

Arguments

`x`	a vector of character strings
`iscell`	x a boolean indicating if x is a cell

Details

if x is "aa.xoijj_a_1_f_1" returns c("a","1","f","1")

Value

a vector of character strings

Examples

```
get_cellXXsplit("aa.x_1_2_3_4",iscellXX=FALSE)
get_cellXXsplit("1_2_3_4",iscellXX=TRUE)
get_cellXXsplit("1_2_3_4",2:3,iscellXX=TRUE)
get_cellXXsplit("1_2_3_4",-(2:3),iscellXX=TRUE)
variables<-Tsampledata(TRUE)$variables
unique(get_cellXXsplit(variables,iscell=FALSE))
get_cellXXsplit(variables[50],iscell=FALSE)
get_cellXXsplit(variables[50],-(2:3),iscell=FALSE)
unique(get_cellXXsplit(variables,2,iscell=FALSE))
#Second example: no transposing variables
TK<-Tsampledata(FALSE)
unique(get_cellXXsplit(TK$variables,iscell=FALSE))
```

`get_missingind` *Get missing indicator for a cell or variable*

Description

Get missing indicator for a cell or variable

Usage

```
get_missingind(x, variables)
```

Arguments

`x`	a vector of character strings

Details

if x is "a_1_f_1_aa.xoijj" returns c("a","1","f","1")

Value

a vector of character strings

Examples

```
variables<-Tsampledata(TRUE)$variables
unlist(unique(get_missingind(variables,variables)))
variables<-Tsampledata(FALSE)$variables
unlist(unique(get_missingind(variables,variables)))
```

---

get_natural.predictors

get variable predecessors at margin

Description

get variable predecessors at margin

Usage

```
get_natural.predictors(x, variables = x, predictors = NULL)
```

Arguments

`x`	a vector of character strings
`variables`	a vector of character strings
`cells`	a vector of character strings containing the potential predecessors
`marginpos`	a vector of integers
`x`	a vector of character strings

Details

if x is "a_1_f_1_aa.xoijj" returns c("a","1","f","1")

if x is "a_1_f_1_aa.xoijj" returns c("a","1","f","1")

Value

a vector of character strings

a vector of character strings

Examples

```
get_XXpredecessoratmargin(cellXXs="aa.x_1_2_3_4", refcellXXs=c("bb.x_1_2_2_4","aa.x_1_2_2_4","aa.x_1_1_3_4"),2,
get_XXpredecessoratmargin(cellXXs=c("1_2_2_4","1_2_2_4","1_1_3_4","1_1_3_3"),iscellXX=FALSE)
data(XKA)
cells<-unique(get_cellrn(XKA$variables))
get_XXpredecessoratmargin(cells,marginpos=1,iscellXX=TRUE)
get_XXpredecessoratmargin(cells[10],cells,1,iscellXX=TRUE)
Get natural predictors

TK<-TtableA
get_natural.predictors(x=sample(names(TtableA),5),variables=names(TtableA))
```

`get_origin` *Get origin table*

Description

Get origin table

Usage

```
get_origin(x)
```

Arguments

`x`	a vector of character strings

Details

if x is "aa.xoijj_a_1_f_1_" returns c("aa")

Value

a vector of character strings

Examples

```
variables<-Tsampledata(TRUE)$variables
unlist(unique(get_origin(variables,variables)))
variables<-Tsampledata(FALSE)$variables
unlist(unique(get_origin(variables,variables)))
```

`get_presentind` *get the present indicator for a cell*

Description

get the present indicator for a cell

Usage

```
get_presentind(variables, refvariables = variables,
  rns = unlist(unique(get_cellrn(refvariables))))
```

Arguments

`x`	a vector of character strings

Details

if x is "a_1_f_1_aa.xoijj" returns c("a","1","f","1")

Value

a vector of character strings

Examples

```
get_presentind("AA.x_1_2_3_4","AA.present_1_2_3_4")
get_presentind("AA.present_1_2_3_4",c("AA.present_1_2_3_3","AA.present_1_2_3_4"))
variables<-Tsampledata(TRUE)$variables
variable<-"AA.present_La_La_Lrn1"
get_presentind(variable,variables)
unlist(unique(get_presentind(variables)))
variables<-Tsampledata(FALSE)$variables
unlist(unique(get_presentind(variables,variables)))
```

`get_var` *Get variable name*

Description

Get variable name

Usage

```
get_var(x)
```

Arguments

`x`	a vector of character strings

Details

if x is "aa.xoijj_a_1_f_1" returns "aa.xoijj"

Value

a vector of character strings

Examples

```
get_var("aa.x_1_2_3_4")
data(TtableA)
unique(get_var(names(TtableA)))
#Second example: no transposing variables
TK<-Tsampledata(FALSE)
unique(get_var(TK$variables))
```

`get_XXpredecessoratmargin` *get cell predecessors at margin*

Description

get cell predecessors at margin

Usage

```
get_XXpredecessoratmargin(XXs, refXXs = XXs, marginpos = NULL,
  iscellXX = FALSE)
```

Arguments

`XXs`	a vector of character strings
`refXXs`	a vector of character strings containing the potential predecessors
`marginpos`	a vector of integers

Details

if XXs is "aa.xoijj_a_1_f_1" and refXXs contains "aa.xoijj_a_1_e_1" and marginpos=3 returns "aa.xoijj_a_1_e_1" if XXs is "aa.xoijj_a_1_f_2" and refXXs contains "aa.xoijj_a_1_f_1" and marginpos=NULL returns "aa.xoijj_a_1_f_1" if XXs is "id1" and iscellXX=FALSE whatever refXXs returns character(0) if XXs is "" and iscellXX=FALSE whatever refXXs returns character(0) if XXs is "b_1_f_1" and iscellXX=TRUE and refXXs contains "a_1_f_1" returns "a_1_f_1"

Value

a vector of character strings

Examples

```
get_XXpredecessoratmargin(XXs="aa.x_1_2_3_4", refXXs=c("bb.x_1_2_2_4","aa.x_1_2_2_4","aa.x_1_1_3_4"),2,iscellXX
get_XXpredecessoratmargin(XXs=c("1_2_2_4","1_2_2_4","1_1_3_4","1_1_3_3"),iscellXX=TRUE)
get_XXpredecessoratmargin(XXs="1_1_3_4",refXXs=c("1_2_2_4","1_2_2_4","1_1_3_4","1_1_3_3"),iscellXX=TRUE)
data(XKA)
cells<-unique(get_cellrn(XKA$variables))
get_XXpredecessoratmargin(cells,marginpos=1,iscellXX=TRUE)
get_XXpredecessoratmargin(cells[10],cells,1,iscellXX=TRUE)
```

ggplot_missing *Create missing chart*

Description

Create missing chart

Usage

```
ggplot_missing(x, reordonne = FALSE)
```

Arguments

x	a dataframe
reordonne	a boolean

Value

a ggplot graph

Examples

```
library(reshape2)
library(ggplot2)
library(plyr)
library(magrittr)
X=cars
for(i in 1:40){
  X[sample(1:50,1,replace=TRUE),sample(1:2,1,replace=TRUE)]<-NA}
ggplot_missing(X,reordonne=TRUE)
ggplot_missing(X,reordonne=FALSE)createallautomaticRMD(schema="SDP")
```

ggplot_missing2 *Create missing chart*

Description

Create missing chart

Usage

```
ggplot_missing2(X, reordonne = TRUE, keep = NULL)
```

Arguments

`X`	a dataframe
`reordonne`	a boolean
`keep`	a boolean

Value

a ggplot graph

Examples

```
library(reshape2)
library(ggplot2)
library(plyr)
X=cars
X$year=sample(2012:2017,nrow(cars),replace=TRUE)
for(i in 1:40){
 X[sample(1:50,1,replace=TRUE),sample(1:2,1,replace=TRUE)]<-NA}
ggplot_missing2(X,keep="year")
```

`missing.summary` *Percentage of missing for each variable*

Description

Percentage of missing for each variable

Usage

```
missing.summary(X, info2 = NULL)
```

Arguments

`X`	a data frame
`info2`	a data frame with two variables named c("COLUMN_NAME","CONSTRAINT_TYPE")

Details

Percentage of missing for each variable of a data frame.

Value

a data frame

predictor.matrix.default *Define a default predictor matrix*

Description

Define a default predictor matrix

Usage

```
predictor.matrix.default(variables)
```

Arguments

`variables`	a vector of character strings

Details

Returns the lower diagonal matrix with ones.

Value

a matrix

Examples

```
variables<-Tsampledata(TRUE)$variables
predictor.matrix.default(TK$variables)
```

predictor.matrix.rate *predictor.matrix.rate*

Description

predictor.matrix.rate

Usage

```
predictor.matrix.rate(variables, nopredictor = character(0),
  allpredictor = character(0), marginposs = integer(0))
```

Arguments

`x`	a vector of character strings

Details

if x is "aa.xoijj_a_1_f_1_" returns c("a","1","f","1")

Value

a vector of character strings

`runCompare` *Shiny App to visualize Data*

Description

Shiny App to visualize Data

Usage

```
runCompare(package1 = NULL, package2 = NULL)
```

Examples

```
package1<-NULL
package2<-NULL
runCompare()
```

`sampledata` *Sample data for transposition*

Description

Sample data for transposition

Usage

```
sampledata(transposingvariables = TRUE)
```

Arguments

`transposingvariables`	a boolean. If TRUE, transposing id variables are created.

Value

a data frame with id variables, numeric, factor and character variables.

SDPSYN *General SDP function.*

Description

General SDP function.

Usage

```
SDPSYN(TtableA, asis = NULL, notpredictor = asis, replicate = 1,
  Sparameters = Sparameters.default.f(TtableA, asis, notpredictor),
  STtableA = plyr::rdply(replicate, TtableA[asis]))
```

Examples

```
TK<-Tsampledata(TRUE)
TtableA<-TK$TtableA;key=TK$key;
STtableA=as.data.frame(matrix(NA,nrow(TtableA),0))
Sparameters=Sparameters.default.f(ref.table=TtableA)
Sparameters_i<-Sparameters[[54]]
Split=Sparameters_i$split[[1]]
Split=Sparameters_i$split[[2]]
library(synthpop)
TtableA[sapply(TtableA,is.character)]<-lapply(TtableA[sapply(TtableA,is.character)],as.factor)
STtableA<-SDPSYN(TtableA)
StableA<-StudyDataTools::GeneralReversetransposefunction(STtableA,key)
```

Sparameters.default.f *Default synthetisation parameters based on variable names*

Description

Default synthetisation parameters based on variable names

Usage

```
Sparameters.default.f(ref.table, asis = NULL, notpredictor = NULL,
  variables = Sparameters.variables.reorder.default(names(ref.table)),
 predictors.matrix = predictor.matrix.default(variables)[!is.element(variables,
  asis), !is.element(variables, notpredictor)], moresplits = NULL,
  preferredmethod = "rf")
```

Arguments

a	vector of character strings

Examples

```
data(TtableA)
ref.table<-TtableA
Sparameters.default.f(ref.table=TtableA)
```

`Sparameters.variables.reorder.default`
General Default ordering of variables for synthetisation based on name of the variable.

Description

General Default ordering of variables for synthetisation based on name of the variable.

Usage

```
Sparameters.variables.reorder.default(variables, orderwithinorigin = NULL,
  id = NULL)
```

Arguments

`variables`	vector of character strings, indicating names of variables
`orderwithinorigin`	a list, see example

Value

a list.

Examples

```
TK<-Tsampledata(TRUE)
Sparameters.variables.reorder.default(names(TK$TtableA))
#Second example: no transposing variables
TK<-Tsampledata(FALSE)
orderwithinorigin=c("AA.factor1","AA.factor2")
Sparameters.variables.reorder.default(names(TK$TtableA),orderwithinorigin)
```

`Tsampledata` *Transposed sample data.*

Description

Transposed sample data.

Usage

```
Tsampledata(transposingvariables = TRUE)
```

Arguments

`transposingvariables`	a boolean. If TRUE, stransposing id variables are created.

Details

Tsampledata(x) is Generaltransposefunction(Tsampledata(x))

Value

a data frame with id variables, numeric, factor and character variables.

`TTsampledata` *Transposed sample data.*

Description

Transposed sample data.

Usage

```
TTsampledata(transposingvariables = TRUE)
```

Arguments

`transposingvariables`	a boolean. If TRUE, stransposing id variables are created.

Details

Tsampledata(x) is Generaltransposefunction(Tsampledata(x))

Value

a data frame with id variables, numeric, factor and character variables.

`var.summary` *Summary for each variable in table.*

Description

Summary for each variable in table.

Usage

```
var.summary(X, datadic = NULL)
```

Arguments

`X`	a data frame
`datadic`	a data dictionnary

Value

a list

Examples

```
data(cars)
var.summary(cars)
```

`var.summaryConnect` *Summary for each variable in table.*

Description

Summary for each variable in table.

Usage

```
var.summaryConnect(X, datadic = NULL)
```

Arguments

`X`	a data frame
`datadic`	a data dictionnary

Value

a list

Examples

```
data(cars)
var.summaryConnect(cars)
```

Index