Package 'eatTools'

December 18, 2012

Type Package

Title eatTools

Version 0.0.3

Depends R (>= 2.14.0), eatData

Imports psych, xlsx, car

Date 2012-12-18

Author Nicole Haag, Karoline Sachse, Sebastian Weirich, Martin Hecht and Thilo Siegle

Maintainer eat authors <eat-commits@lists.r-forge.r-project.org>

Description some more or less usefull/-less tools

License GPL (>= 2)

LazyLoad yes

LazyData yes

R topics documented:

asNumericIfPossible *Transform columns of a data.frame into numeric values if possible*

Description

In contrast to `as.numeric`, Function transforms only "transformable" columns of a data.frame into numeric values (i.e. without creating `NA` when transformation fails. Non-transformable columns are maintained. Optionally, only a logical vector is given, indicating which columns are transformable.

Usage

```
asNumericIfPossible ( dat, set.numeric = TRUE, transform.factors = FALSE, maintain.factor.scores
```

Arguments

`dat`	A data.frame which columns should be transformed.
`set.numeric`	Logical: If `TRUE`, data.frame with transformed columns is returned. If `FALSE`, a logical vector is returned, indicating which columns are transformable.
`transform.factors`	Logical: Should columns of class `factor` transformed? If `FALSE`, columns of class `factor` are maintained. If `TRUE`, columns of class `factor` are attempted to transform.
`maintain.factor.scores`	Logical. Only relevant if `transform.factors = TRUE`. If `TRUE`, the nominal values of the factor are transformed if possible. If `FALSE`, the integer numbers representing the factors' nominal values are returned. See details.
`verbose`	Logical: If `TRUE`, informations about the class of the columns in the data.frame are printed to the console.

Details

In R, factors may represent ordered categories or nominal variables. Depending on the meaning of the variable, a transformation of the nominal values (of a factor variable) to numeric values may be desirable or not. The arguments `transform.factors` and `maintain.factor.scores` serve to specify if and how factor variables should be transformed. See examples.

Value

Either a logic vector, indicating which columns in the data.frame are transformable according to the specified conditions, ora data.frame in which transformable columns are transformed.

Author(s)

Sebastian Weirich

Examples

```
( dat <- data.frame( X1 = c("1",NA,"0"), X2 = c("a",NA,"b"), X3 = c(TRUE,FALSE,FALSE),  X4 = as.factor(c
str(dat)
asNumericIfPossible(dat)
asNumericIfPossible(dat, transform.factors=TRUE, maintain.factor.scores=FALSE)
asNumericIfPossible(dat, transform.factors=TRUE, maintain.factor.scores=TRUE)
```

`collapseMissings` *Collapse Missings*

Description

converts character missings of different types to 0 or NA

Usage

```
collapseMissings(dat, missing.rule = NULL, items)
```

Arguments

`dat`	data frame containing character missings (e.g. type 'mbd' - missing by design)
`missing.rule`	list, definition how to recode distinct missings in dataset. See details for default.
`items`	character vector containing column names of the data frames whose character missings are to be collapsed

Details

Default `missing.rule` in `collapseMissings` is: text volume insufficient = 0 , missing not reached = 0 , missing coding impossible = NA , missing by design = NA , missing invalid response = 0 , missing by intention = 0

The results of `collapseMissings` will be written to a protocol file with `sunk`.

Value

A data frame with recoded missings.

Author(s)

Karoline Sachse, Martin Hecht

References

For missing types see http://code.google.com/p/zkdlib/wiki/MissingHandling

commonItems *identify common items of groups*

Description

This function identifies items that groups of persons have in common.

Usage

```
commonItems ( dat , group.var , na = NA , uncommon = FALSE , simplify = TRUE )
```

Arguments

dat	data.frame
group.var	group variable in data.frame , eihter numeric indicator of column or column name
na	missing specification
uncommon	if TRUE a vector of uncommon items is additionally returned
simplify	if TRUE a character vector is returned (only in case of 2 groups and uncommon=FALSE)

Details

dat must only contain the group variable and the items, if further variables are in dat they are treated as items

Value

returns a list of all group.var combinations with character vectors of common item names if uncommon=TRUE a vector of uncommon (unique) items of each group is additionally returned

names of list are both group names concatenated by "|"

Author(s)

Martin Hecht

Examples

```
data(science1)
d <- science1[,c("version",science1.items)]

# common items are listed for each combination of groups
str ( commonItems ( dat = d , group.var = "version" , na = "mbd" ) )

# uncommon items are returned as well
str ( commonItems ( dat = d , group.var = "version" , na = "mbd" , uncommon = TRUE ) )
```

commonItems.percent *calculate the percentage of common items of groups*

Description

This function calculates the percentage of items that groups of persons have in common.

Usage

```
commonItems.percent ( dat , group.var , na = NA , xlsx = NULL )
```

Arguments

dat	data.frame
group.var	group variable in data.frame , eihter numeric indicator of column or column name
na	missing specification
xlsx	full path (directory + file name) to Excel to be written (don't forget ".xlsx" suffix)

Details

dat must only contain the group variable and the items, if further variables are in dat they are treated as items

Value

returns a data.frame with common item percentage(s)

Author(s)

Martin Hecht

Examples

```
data(science1)
d <- science1[,c("version",science1.items)]
( commonItems.percent ( dat = d , group.var = "version" , na = "mbd" ) )
```

crop *crop*

Description

remove trailing and leading characters from character strings

Usage

```
crop ( x , char = " " )
```

Arguments

x	character string
char	character to be removed from beginning and end of x

Author(s)

Martin Hecht, Sebastian Weirich

---

fill.na *fill NAs in a vector*

Description

fill NAs with non-NA values depending on left (forward) or right (backward) non-NA value

Usage

```
fill.na ( vec , backwards = FALSE , na.rm = FALSE )
```

Arguments

vec	a vector
backwards	if FALSE NAs are filled forward, if TRUE NAs are filled backwards
na.rm	if TRUE NAs at start and end of vector are removed

Value

a vector with filled NAs

Author(s)

Martin Hecht

Examples

```
( vec <- c ( NA , 1 , NA , NA , 2 , NA , 3 , NA ) )
fill.na ( vec )
fill.na ( vec , backwards = TRUE )
```

make.dummies *dummy coded variables*

Description

creates dummy coded variables using dummy.code names of dummy coded variables can be customized and added to the input data.frame

Usage

```
make.dummies ( dat , cols , colname.as.prefix = TRUE , delimiter = "." , capitalize = FALSE , nc
```

Arguments

dat	data.frame
cols	colnames of variables to be dummy coded
colname.as.prefix	logical, if TRUE the original variable name is added as prefix
delimiter	logical, if TRUE (and colname.as.prefix = TRUE) variable name and "level" name are delimited by delimiter
capitalize	logical, if TRUE "levels" are capitalized
nchar	numeric, if TRUE "levels" are truncated to length nchar
add	logical, if TRUE dummy coded variables are added to dat
sort.into.dat	logical, if TRUE (and add = TRUE) dummy coded variables are added and sorted into dat
oneToColname	logical, if TRUE 1-values are set to colname of respective column, be aware that this changes the column class from numeric to character
zeroToNA	logical, if TRUE 0-values are set to NA, be aware that this changes the column class from numeric to character

Value

returns data.frame with dummy coded variables, depending on add either the original data.frame (dat) is appended with new dummy code variables or they are purely returned

Author(s)

Martin Hecht

Examples

```
## Not run:
data(science1)

science1.dum <- make.dummies ( science1 , c("sex","booklet") )
str ( science1.dum[,3:10] )

science1.dum <- make.dummies ( science1 , c("sex","booklet") , nchar = 1 )
str ( science1.dum[,3:10] )
```

```
science1.dum <- make.dummies ( science1 , c("sex","booklet") , delimiter = "_" )
str ( science1.dum[,3:10] )

science1.dum <- make.dummies ( science1 , c("sex","booklet") , delimiter = "" , capitalize = TRUE )
str ( science1.dum[,3:10] )

science1.dum <- make.dummies ( science1 , c("sex","booklet") , colname.as.prefix = FALSE )
str ( science1.dum[,3:10] )

science1.dum <- make.dummies ( science1 , c("sex","booklet") , sort.into.dat = FALSE )
str ( science1.dum[,(ncol(science1.dum)-5):ncol(science1.dum)] )

science1.dum <- make.dummies ( science1 , c("sex","booklet") , add = FALSE )
str ( science1.dum )

science1.dum <- make.dummies ( science1 , c("sex","booklet") , oneToColname = TRUE , zeroToNA = TRUE  )
str ( science1.dum )


## End(Not run)
```

`modus` *modus*

Description

calculates modus (most frequent value)

Usage

```
modus ( x , randTies = FALSE )
```

Arguments

`x`	a vector
`randTies`	if ties occur draw a randomized value out of tied values

Value

returns the modus (most frequent element)

Author(s)

Martin Hecht

Examples

```
## Not run:
x <- c ( 1 , 1 , 2 , 2 )
( modus ( x ) )
( modus ( x , randTies = TRUE ) )

x <- c ( 1 , NA , NA )
( modus ( x ) )
```

```
x <- c ( "x" , "x" , "y" )
( modus ( x ) )

## End(Not run)
```

---

multiseq *multiple sequences*

Description

creates a sequence for every unique value in a vector

Usage

```
multiseq ( v )
```

Arguments

v	a vector

Value

a vector with multiple sequences

Author(s)

Martin Hecht

Examples

```
v <- c("a", "a", "a", "c", "b", "b" , "a" )
( multiseq ( v ) )
```

---

rmNA *remove NA columns and rows from data*

Description

remove columns and rows that are completely NA from data.frame or matrix

Usage

```
rmNA ( dat , remove = TRUE , verbose = FALSE )
```

Arguments

dat	data.frame or matrix
remove	if TRUE columns and rows are removed, if FALSE a list of identified columns and rows is returned
verbose	if TRUE removed columns and rows are printed on output window

Value

depends on option remove

Author(s)

Martin Hecht

See Also

rmNAcols, rmNArows

Examples

```
( mat <- matrix( c( 1,1,1,1,1,NA, 1,1,1,1,NA,NA, 1,1,1,NA,NA,NA, 1,1,NA,NA,NA,NA, 1,NA,NA,NA,NA,NA, NA,NA,N
( rmNA ( mat , verbose = TRUE ) )
( rmNA ( mat , remove = FALSE ) )
```

---

rmNAcols *remove NA columns from data*

Description

remove columns that are completely or partially NA from data.frame or matrix

Usage

```
rmNAcols ( dat , rows = NULL , tolerance = 0 , cumulate = TRUE , remove = TRUE , verbose = FALSE
```

Arguments

dat	data.frame or matrix
rows	rows to include, can be a list of vectors to specify row subsets
tolerance	number of non-NA cells that are "tolerated", can be a list corresponding to rows
cumulate	if TRUE, tolerance is cumulated; if FALSE, exact tolerance is used
remove	if TRUE, columns and rows are removed; if FALSE, identified columns are returned
verbose	if TRUE removed columns and rows are printed on output window

Value

depends on option remove

Author(s)

Martin Hecht

See Also

calls rmNA and rmNArows

Examples

```
# example matrix
( mat <- matrix( c( 1,1,1,1,1,1, 1,1,1,1,1,NA, 1,1,1,1,NA,NA, 1,1,1,NA,NA,NA, 1,1,NA,NA,NA,NA, 1,NA,NA,NA,N

# remove column with entirely NA (column 7)
rmNAcols( mat , verbose = TRUE )

# remove column with NA on rows 3, 4, 5 (columns 5, 6, 7)
rmNAcols( mat , c(3,4,5) , verbose = TRUE )
rmNAcols( mat , c(-1,-2,-6) , verbose = TRUE )

# tolerance=1 , 1 non-NA is permitted (columns 6 and 7)
rmNAcols( mat , tolerance=1 , verbose = TRUE )

# tolerance=6 , 6 non-NA are permitted (all columns are removed)
rmNAcols( mat , tolerance=6 , verbose = TRUE )

# do not cumulate / exact tolerance (column 1)
rmNAcols( mat , tolerance=6 , cumulate=FALSE , verbose = TRUE )

# two subsets of rows
rmNAcols( mat , rows = list( c(1, 2), c(4, 5) ) , verbose = TRUE )

# two subsets of rows with different tolerance
rmNAcols( mat , rows = list( c(1), c(2, 3, 4, 5) ) , tolerance = list( 0 , 1 ) , verbose = TRUE )

# identify cols, no deletion
rmNAcols( mat , rows = list( c(1, 2), c(3, 4, 5) ) , tolerance = list( 0 , 1 ) , remove = FALSE )
```

`rmNArows` *remove NA rows from data*

Description

remove rows that are completely or partially NA from data.frame or matrix

Usage

```
rmNArows ( dat , cols = NULL , tolerance = 0 , cumulate = TRUE , remove = TRUE , verbose = FALSE
```

Arguments

`dat`	data.frame or matrix
`cols`	columns to include, can be a list of vectors to specify column subsets
`tolerance`	number of non-NA cells that are "tolerated", can be a list corresponding to `cols`
`cumulate`	if `TRUE`, `tolerance` is cumulated; if `FALSE`, exact `tolerance` is used
`remove`	if `TRUE`, columns and rows are removed; if `FALSE`, identified rows are returned
`verbose`	if `TRUE` removed columns and rows are printed on output window

Value

depends on option `remove`

Author(s)

Martin Hecht

See Also

calls rmNA and rmNAcols

Examples

```
# example matrix
( mat <- matrix( c( 1,1,1,1,1, 1,1,1,1,NA, 1,1,1,NA,NA, 1,1,NA,NA,NA, 1,NA,NA,NA,NA, NA,NA,NA,NA,NA ) , nco

# remove row with entirely NA (row 6)
rmNArows( mat , verbose = TRUE )

# remove row with NA on column 3, 4, 5 (rows 4, 5, 6)
rmNArows( mat , c(3,4,5) , verbose = TRUE )
rmNArows( mat , c(-1,-2) , verbose = TRUE )

# tolerance=1 , 1 non-NA is permitted (rows 5 and 6)
rmNArows( mat , tolerance=1 , verbose = TRUE )

# tolerance=5 , 5 non-NA are permitted (all rows are removed)
rmNArows( mat , tolerance=5 , verbose = TRUE )

# do not cumulate / exact tolerance (row 1 is removed)
rmNArows( mat , tolerance=5 , cumulate=FALSE , verbose = TRUE )
rmNArows( mat , tolerance=5 , cumulate=FALSE , remove = FALSE )

# two subsets of columns
rmNArows( mat , cols = list( c(1, 2), c(4, 5) ) , verbose = TRUE )

# two subsets of columns with different tolerance
rmNArows( mat , cols = list( c(1), c(2, 3, 4, 5) ) , tolerance = list( 0 , 1 ) , verbose = TRUE )

# identify rows, no deletion
rmNArows( mat , cols = list( c(1), c(2, 3, 4, 5) ) , tolerance = list( 0 , 1 ) , remove = FALSE )
```

---

set.col.type *set type of variable in data.frame*

Description

converts type of column(s) to "character" , "numeric" , "logical" , "integer" or "factor"

Usage

```
set.col.type ( dat , col.type = list ( "character" = NULL ) , verbose = FALSE , ... )
```

Arguments

dat	data.frame
col.type	named list of variable names that are to be converted. names of list is conversion type ("character" , "numeric" , "numeric.if.possible" , "logical" , "integer" or "factor")
verbose	if TRUE variables that have been converted are printed
...	arguments to be passed to asNumericIfPossible

Details

use col.type="numeric.if.possible" if conversion to numeric should be tested upfront, see asNumericIfPossible for details

Author(s)

Martin Hecht

Examples

```
str ( d <- data.frame ( "var1" = 1 , "var2" = TRUE , "var3" = FALSE , "var4" = as.factor ( 1 ) , "var5" = a
str ( set.col.type ( d ) )
str ( set.col.type ( d , list ( "numeric" = NULL ) ) )
str ( set.col.type ( d , list ( "character" = c ( "var1" , "var2" ) , "numeric" = "var3" , "logical" = "var
str ( set.col.type ( d , list ( "numeric.if.possible" = NULL ) ) )
str ( set.col.type ( d , list ( "numeric.if.possible" = NULL ) , transform.factors = TRUE ) )
str ( set.col.type ( d , list ( "numeric.if.possible" = NULL ) , transform.factors = TRUE , maintain.factor
```

Index