Package ‘eatTools’

November 10, 2021

Type    Package
Title   Miscellaneous Functions for the Analysis of Educational Assessments
Version 0.7.1
Depends R (>= 3.5.0)
Imports stats, data.table
Description Miscellaneous functions for data cleaning and data analysis of educational assessments. Includes functions for descriptive analyses, character vector manipulations and weighted statistics. Mainly a lightweight dependency for the packages ‘eatRep’, ‘eatGADS’, ‘eatPrep’ and ‘eatModel’ (which will be subsequently submitted to ‘CRAN’).
License GPL (>= 2)
URL https://github.com/weirichs/eatTools
Suggests testthat, covr
NeedsCompilation no
Author Sebastian Weirich [aut, cre],
       Martin Hecht [aut],
       Karoline Sachse [aut],
       Benjamin Becker [aut],
       Nicole Mahler [aut]
Maintainer Sebastian Weirich <sebastian.weirich@iqb.hu-berlin.de>
Repository CRAN
Date/Publication 2021-11-10 11:20:02 UTC

R topics documented:

   addLeadingZerosToCharInt .............................................. 2
addLeadingZerosToCharInt

Add leading zeros to all columns that can be identified as integers in a character data.frame

Description

Adds leading zeros to all columns that can be identified as integers in a data.frame that consists of character columns only.

Usage

addLeadingZerosToCharInt(dat)

Arguments

dat a data.frame consisting of character columns only
asNumericIfPossible

Value

A data.frame of only character columns and the same dimension as the input data.frame where all columns with integers are all of the same arity now due to added leading zeros.

Author(s)

Karoline Sachse

Examples

dat <- data.frame(v1 = c("0", "300", "e", NA),
  v2 = c("0", "90", "10000", NA),
  v3 = c("k", "kk", "kkk", NA),
  v4 = NA,
  v5 = c("0", "90", "100", "1"))
dat <- set.col.type(dat)
addLeadingZerosToCharInt(dat)

asNumericIfPossible(x, maintain.factor.scores = TRUE, force.string = TRUE,
transform.factors = TRUE, varName = NULL)

Arguments

x

A vector or data frame which should be converted.

maintain.factor.scores

Logical: If TRUE, conversion of the factor levels is attempted (like in as.numeric(as.character(f))). If FALSE, the internal codes of the factor are returned (like in as.numeric(f)). See 'Details'. This argument is only evaluated if transform.factors = TRUE.

force.string

Logical indicating whether columns should be force to numeric, even if NAs are induced. If FALSE, affected columns are maintained. If TRUE, conversion is forced.

transform.factors

Logical indicating whether columns of class factor should be converted. If FALSE, columns of class factor are maintained. If TRUE, conversion of factors is attempted.

varName

Optional: Name of the corresponding variable. Doesn’t have to be changed by user.
Details

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

Author(s)

Sebastian Weirich, Karoline Sachse, Benjamin Becker

Examples

```r
dat <- data.frame(X1 = c(1,NA,0), X2 = c(a,NA,b),
                   X3 = c(TRUE,FALSE,FALSE),
                   X4 = as.factor(c(a,NA,b)),
                   X5 = as.factor(c(5,6,7)),
                   stringsAsFactors = FALSE)
str(dat)
asNumericIfPossible(dat)
asNumericIfPossible(dat, transform.factors=TRUE,
                   maintain.factor.scores=FALSE)
asNumericIfPossible(dat, transform.factors=TRUE,
                   maintain.factor.scores=TRUE)
```

---

catch_asNumericIfPossible

Use `asNumericIfPossible` with modified warning.

Description

This function uses `asNumericIfPossible` but lets the user change the warning issued by `asNumericIfPossible`. Suited for use in other R packages.

Usage

```r
catch_asNumericIfPossible(x, warn, maintain.factor.scores = TRUE,
                          force.string = TRUE, transform.factors = TRUE)
```

Arguments

- `x`: A vector or data frame which should be converted.
- `warn`: A character vector of length 1 with the desired warning.
- `maintain.factor.scores`: Logical: If `TRUE`, conversion of the factor levels is attempted (like in `as.numeric(as.character(f))`). If `FALSE`, the internal codes of the factor are returned (like in `as.numeric(f)`). See 'Details'. This argument is only evaluated if `transform.factors = TRUE`.
- `force.string`: Logical indicating whether columns should be force to numeric, even if NAs are induced. If `FALSE`, affected columns are maintained. If `TRUE`, conversion is forced.
contr.wec.weighted

transform.factors

Logical indicating whether columns of class factor should be converted. If FALSE, columns of class factor are maintained. If TRUE, conversion of factors is attempted.

Details

For details see asNumericIfPossible

Author(s)

Benjamin Becker

Examples

char <- c("a", "b", 1)
catch_asNumericIfPossible(x = char, warn = "Vector could not be converted")

contr.wec.weighted

Calculates contrasts for a weighted factor variable based on weighted effect coding

Description

Function works equivalent to contr.wec from the wec package, but allows for weighted contrasts.

Usage

contr.wec.weighted (x, omitted, weights)

Arguments

x grouping variable of class factor
omitted Label of the factor label that should be taken as the omitted category
weights Numeric vector of non-negative weights

Value

Returns a contrast matrix based on weighted effect coding.

Author(s)

Sebastian Weirich, based upon the contr.wec function of the wec package
Examples

```r
### exemplary data according to wec paper
dat <- data.frame ( group = as.factor(c(rep(1,3), rep(2,2))), wgt = c(2/3, 4/3, 2, 3/8, 5/8))
### default contrasts
contrasts(dat[,"group"])
### weighted effect coding for weighted data
contr.wec.weighted(x= dat[,"group"], omitted=1,weights=dat[,"wgt"])
### equal to weighted effect coding: wec::contr.wec(x= dat[,"group"], omitted=1)
contr.wec.weighted(x= dat[,"group"], omitted=1,weights=rep(1, nrow(dat)))
```

---

crop  

Remove Trailing and Leading Characters From Character Strings

Description

Similarly to the function `trim` from the gdata package, this function can be used to remove trailing and leading spaces from character strings. However, in contrast to `trim`, any character can be removed by crop.

Usage

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

Arguments

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

Author(s)

Martin Hecht, Sebastian Weirich

Examples

```r
str <- c(" 12 kk ", "op j q ", "110")
crop(str)
crop(str, "op")
```
**Describer**

*Descriptive statistics for one or several variables*

**Description**

Function computes descriptive statistics for one variable or several variables within a data frame.

**Usage**

`descr (variable, na = NA, p.weights = NULL, na.rm = FALSE, verbose=TRUE)`

**Arguments**

- **variable**
  - one variable or a data.frame with several variables
- **na**
  - optional values with should be considered a missing values
- **p.weights**
  - optional: vector with individual weights if weighted statistics should be computed
- **na.rm**
  - logical: should missings be removed prior to estimation?
- **verbose**
  - logical: Print messages to console?

**Value**

A data frame with the following columns:

- **N**
  - number of observations
- **N.valid**
  - number of non-missing observations
- **Missing**
  - number of missings
- **Minimum**
  - minimum of numeric variables
- **Maximum**
  - maximum of numeric variables
- **Sum**
  - sum of numeric variables
- **Mean**
  - arithmetic mean of numeric variables
- **std.err**
  - standard error of the arithmetic mean. Note: for weighted means, standard error is estimated according to Cochran (1977): `sigma_x^2 = n/((n - 1) * w_i^2) * Sigma(u_i^2 * (x_i - x))`. 
- **sig**
  - p value
- **Median**
  - median of numeric variables
- **SD**
  - standard deviation of numeric variables
- **Var**
  - variance of numeric variables

**Author(s)**

Sebastian Weirich
do_call_rbind_withName

References


Examples

data(mtcars)
descr(mtcars)

---

do_call_rbind_withName

Row bind a list while assigning names to rows

Description

Use do.call(rbind,...) on a list of data.frames while creating a new variable (colName) which contains, for example, the original list naming (name).

Usage

do_call_rbind_withName(df_list, name = names(df_list), colName)

Arguments

df_list A list of data.frames.
name Vector of names to fill colName. Default uses the names of df_list.
colName A single character; name for the new column.

Value

Returns a data.frame.

Author(s)

Benjamin Becker

Examples

### create example list
df_list <- lapply(mtcars, function(x) {
  data.frame(m = mean(x), sd = sd(x))
})

### transform to a single data.frame
do_call_rbind_withName(df_list, colName = "variable")
eatTools: Miscellaneous Functions for the Analysis of Educational Assessments

Description

The eatTools package provides various groups of functions. The main groups of functions include: transformation of vector types, modification of character variables, descriptive analyses and weighted statistics. The package’s purpose is mainly to function as a lightweight dependency for other packages.

Transformation of vector types

The functions `asNumericIfPossible` and `catch_asNumericIfPossible` transform character and factor variables to numeric. `facToChar` transforms factor variables to character. `set.col.type` allows manually setting the type of multiple variables within a `data.frame`.

Modification of character variables

Multiple convenience functions exist for modification of character variables: removing certain pattern (`removePattern`), removing numerics (`removeNumeric`) and removing non numerics (`removeNonNumeric`), substituting multiple patterns within a string (`gsubAll`) and splitting strings into multiple or a fixed number of parts but at specific position (`halveString`).

Descriptive Statistics

The function `descr` provides simple descriptive statistics for a `data.frame`, but in a format especially useful for further automated processing (long format `data.frame`).

Weighted Statistics

`wtdVar` provides calculation of weighted variances (this can be done also by the package Hmisc, which has, however, a very high number of dependencies). `wtdTable` provides a weighted frequency table.

Description

Function is necessary for eatRep and eatModel as well and therefore exported to namespace.

Usage

`existsBackgroundVariables (dat, variable, warnIfMissing = FALSE)`
Arguments

dat | A data frame
variable | column number or variable name
warnIfMissing | Logical: gives a warning if the variable contains missing values

Value

a structured list of variable names

Author(s)

Sebastian Weirich

Examples

data(mtcars)
existsBackgroundVariables(mtcars, 2:4)

facToChar | Transform columns in a data frame

Description

Function transforms all data frame columns of a specific class into another class.

Usage

facToChar (dataFrame, from = "factor", to = "character")

Arguments

dataFrame | a data frame
from | which column class should be transformed?
to | target column class

Value

a data frame

Author(s)

Sebastian Weirich
### Examples

```r
data(mtcars)
### original classes
sapply(mtcars, class)
mtcars1 <- facToChar(mtcars, from = "numeric", to = "character")
sapply(mtcars1, class)
```

---

#### gsubAll

**Pattern matching and replacement**

---

**Description**

Function is a wrapper for `gsub()` which allows to replace more than one pattern.

**Usage**

```r
gsubAll ( string, old, new)
```

**Arguments**

- **string**: a character vector where matches are sought
- **old**: character vector containing strings to be matched in the given character vector named `string`
- **new**: a replacement for matched pattern

**Value**

character vector with replaced patterns

**Author(s)**

Benjamin Becker

**Examples**

```r
### replace all numbers by words
txt <- "1 example for 2 reasons in 4 seasons"
gsubAll ( txt, old = as.character(1:4), new = c("one", "two", "three", "four"))
```
**halveString**  
*Split string exactly in two parts*

**Description**

strsplit splits a string according to a specific sign. The number of occurrences of the splitting sign defines the number of splits. halveString allows to split the string in only two parts, no matter how often the splitting sign occurs.

**Usage**

halveString (string, pattern, first = TRUE )

**Arguments**

- **string**  
  A character vector.
- **pattern**  
  character vector (or object which can be coerced to such) to use for splitting.
- **first**  
  Logical: Relevant if the pattern occurs more than one time in the string. Defines whether the first (default) or last occurrence is used for splitting.

**Value**

A data.frame with two columns

**Author(s)**

Sebastian Weirich

**Examples**

```r
str <- c("John_Bolton", "Richard_Milhouse_Nixon", "Madonna")
strsplit(str, split = "_")
halveString(str, pattern = "_")
halveString(str, pattern = "_", first=FALSE)
```

**insert.col**  
*Insert Columns into a Data Frame in a Specific Position*

**Description**

Insert columns into a data frame in specific position

**Usage**

insert.col(dat, toinsert, after)
multiseq

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dat</td>
<td>A data frame</td>
</tr>
<tr>
<td>toinsert</td>
<td>Column name(s) or column number(s) of the columns to be reinserted</td>
</tr>
<tr>
<td>after</td>
<td>Column name or column number after which the columns specified in insert should be reinserted.</td>
</tr>
</tbody>
</table>

Value

A data frame with columns in specified positions.

Description
creates a sequence for every unique value in a vector

Usage

multiseq(v)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>v</td>
<td>a vector</td>
</tr>
</tbody>
</table>

Value

a vector with multiple sequences

Author(s)

Martin Hecht

Examples

v <- c("a", "a", "a", "c", "b", "b", "a")
multiseq(v)
**na_omit_selection**  
*Drop rows containing missing values*

**Description**
Drop rows containing missing values in selected columns.

**Usage**
```
na_omit_selection(dat, varsToOmitIfNA)
```

**Arguments**
- `dat`: a data.frame
- `varsToOmitIfNA`: Name or column number of the variables which should be considered for row deletion due to NAs

**Value**
A data.frame with deleted rows

**Examples**
```
# Example data frame
v1 = c(1,NA,3), v2 = c(letters[1:2],NA),
v3 = c(NA, NA, TRUE), stringsAsFactors = FALSE)

na.omit(dat1)
na_omit_selection(dat1, "v2")
```

---

**num.to.cat**  
*Transform continuous variables into ordered factors*

**Description**
Function is useful if parameters on the ‘PISA’ metric should be transformed into competence levels.

**Usage**
```
num.to.cat(x, cut.points, cat.values = NULL)
```

**Arguments**
- `x`: Numeric vector.
- `cut.points`: Numeric vector with cut scores.
- `cat.values`: Optional: vector with labels for the cut scores. Note: if specified, length of cat.values should be length(cut.points)+1.
Value

Vector with factor values.

Author(s)

Sebastian Weirich

Examples

values <- rnorm(10,0,1.5) * 100 + 500
num.to.cat(x = values, cut.points = 390+0:3*75)
num.to.cat(x = values, cut.points = 390+0:3*75, cat.values = c("1a", "1b", 2:4))

---

readMultisep  Read in data.frames with separator characters >=1Byte

Description

Read in character separated data.frames with separator characters >=1Byte.

Usage

readMultisep(file, sep)

Arguments

file 
the name of the file which the data are to be read from.

sep 
the field separator character(s).

Value

A data frame containing a representation of the data in the file.

Examples

filePath <- tempfile(fileext = ".txt")
dat <- data.frame(v1 = c("0","300","e",NA),
v2=c("0","90","10000",NA),
v3=c("k","kk","kkk",NA),
v4=NA,
v5=c("0","90","100","1")
write.table(dat, file = filePath, row.names = FALSE, col.names = FALSE, sep = "]&;")
readMultisep(filePath, sep="]\&;")
**recodeLookup**  
Recode a variable according to a lookup table

**Description**
Recodes the values of a variable. Function resembles the recode function from the car package, but uses a lookup table to specify old and new values.

**Usage**
recodeLookup(var, lookup)

**Arguments**
- **var**: a vector (e.g. numeric, character, or factor)
- **lookup**: a data.frame with exact two columns. First column contains old values, second column new values. Values which do not occur in the old column remain unchanged.

**Value**
a vector of the same length as var with recoded values

**Examples**
num_var <- sample(1:10, size = 10, replace = TRUE)
lookup <- data.frame(old = c(2, 4, 6), new = c(200, 400, 600))
num_var2 <- recodeLookup(num_var, lookup)

---

**removeNonNumeric**  
Removes all non-numeric characters from a string.

**Description**
Function removes all non-numeric characters from a string.

**Usage**
removeNonNumeric ( string)

**Arguments**
- **string**: a character vector

**Value**
a character string
removeNumeric

Author(s)
Sebastian Weirich

Examples
str <- c(".d1.nh.120", "empty", "110", ".nh.dgd", "only.nh")
removeNonNumeric(str)

removeNumeric  Removes alphanumeric characters from a string.

Description
Function removes alphanumeric characters from a string.

Usage
removeNumeric ( string)

Arguments
string       a character vector

Value
a character string

Author(s)
Sebastian Weirich

Examples
str <- c(".d1.nh.120", "empty", "110", ".nh.dgd", "only.nh")
removeNumeric(str)
removePattern  
Removes a specified pattern from a string.

Description

Function remove a specified string from a character vector.

Usage

```
removePattern(string, pattern)
```

Arguments

- **string**: a character vector
- **pattern**: a character pattern

Value

a character string

Author(s)

Sebastian Weirich

Examples

```r
cr <- c(".d1.nh.120", "empty", "110", ".nh.dgd", "only.nh")
removePattern(cr, ".nh.")
```

roundDF  
Round a data.frame.

Description

Round all numeric variables in a data.frame, leave the other variables untouched. Column and row names are preserved.

Usage

```
roundDF(dat, digits = 3)
```

Arguments

- **dat**: A data.frame.
- **digits**: Integer indicating the number of decimal places.
set.col.type

Value

Returns the rounded data.frame.

Examples

roundDF(mtcars, digits = 0)

set.col.type

Set the Class of Columns in a Data Frame

Description

This function converts the classes of columns to character, numeric, logical, integer or factor.

Usage

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

Arguments

dat A data frame

col.type A named list of column names that are to be converted. The names of the list indicate the class to which the respective column should be converted (character, numeric, numeric.if.possible, logical, integer or factor)

verbose if TRUE details about converted columns are printed on the console

... Additional 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.

Value

A data frame with column classes changed according to the specifications in col.type

Author(s)

Martin Hecht, Karoline Sachse

See Also

asNumericIfPossible
tablePattern

Creates skeleton for frequency tables with desired values

Description

Function takes values and creates a frequency table including these values. Models behavior of factor variables.

Usage

```r
tablePattern (x, pattern = NULL, weights, na.rm = TRUE, useNA = c("no", "ifany", "always"))
```

Arguments

- `x`  
  a vector
- `pattern`  
  desired values for table output
- `weights`  
  optional: weights
- `na.rm`  
  should missing values be removed
- `useNA`  
  whether to include [NA] values in the table

Value

a frequency table

Author(s)

Sebastian Weirich

Examples

```r
grades <- c(1,1,3,4,2,3,4,5,5,3,2,1)
table(grades)
tablePattern(grades, pattern = 1:6)
```
tableUnlist

Frequency table for data frames, e.g. across multiple columns

Description

Replaces the somehow buggy function combination table(unlist(data)).

Usage

tableUnlist(dataFrame, useNA = c("no", "ifany", "always"))

Arguments

dataFrame Data frame with more than one column.
useNA whether to include NA values in the table. See help file of table for more details.

Value

A frequency table

Author(s)

Sebastian Weirich

Examples

dat <- data.frame ( matrix ( data = sample(0:1,200,replace=TRUE), nrow=20, ncol=10))
tableUnlist(dat)

whereAre

Matches a scalar with elements of a vector.

Description

The function closely resembles the match function, but allows for multiple matches.

Usage

whereAre(a,b,verbose=TRUE)

Arguments

a a scalar
b a numeric or character vector
verbose logical: print messages on console?
wideToLong

Transform wide format data sets into the long format necessary for eatRep analyses

Description

Data from large-scale assessments often are provided in the wide format. This function easily transform data into the long format required by eatRep.

Usage

wideToLong (datWide, noImp, imp, multipleColumns = TRUE, variable.name = "variable", value.name = "value")

Arguments

datWide
    Data set in the wide format, i.e. one row per person

noImp
    character vector of non-imputed variables which are desired for following analyses

imp
    Named list of character vectors which include the imputed variables which are desired for following analyses

multipleColumns
    Logical: use one column for each imputed variable (if more than one imputed variable is used)? Alternatively, only one column for all imputed variables is used (this is the default behavior of the melt function from the reshape2 package).

variable.name
    Applies only if multipleColumns = "FALSE": name of variable used to store measured variable names

value.name
    Applies only if multipleColumns = "FALSE": name of variable used to store values
wtdTable

Computed weighted frequency tables

Description

This function works quite equally as the `wtd.table` function from the Hmisc package.

Usage

```
wtdTable(x, weights, na.rm = FALSE)
```

Arguments

- `x`: a character or category or factor vector
- `weights`: a numeric vector of non-negative weights
- `na.rm`: set to `FALSE` to suppress checking for NAs. If `TRUE`, NAs are removed from `x` as well as from `weights` prior to variance estimation.

Value

A frequency table
wtdVar

Computed weighted variance

Description

This function works quite equally as the wtd.var function from the Hmisc package.

Usage

wtdVar(x, weights, na.rm = FALSE)

Arguments

x numeric vector
weights a numeric vector of non-negative weights
na.rm set to FALSE to suppress checking for NAs. If TRUE, NAs are removed from x as well as from weights prior to variance estimation.

Value

a scalar

Author(s)

Benjamin Becker

Examples

x <- c(50, 1, 25)
w <- c(1, 4, 1)
wtdVar(x, w)
Index

addLeadingZerosToCharInt, 2
asNumericIfPossible, 3, 5, 9

catch_asNumericIfPossible, 4, 9
contr.wec.weighted, 5
crop, 6
descr, 7, 9
do_call_rbind_withName, 8
eatTools, 9
eexistsBackgroundVariables, 9

facToChar, 9, 10
gsubAll, 9, 11
halveString, 9, 12
insert.col, 12
multiseq, 13

na_omit_selection, 14
num.to.cat, 14

readMultisep, 15
recodeLookup, 16
removeNonNumeric, 9, 16
removeNumeric, 9, 17
removePattern, 9, 18
roundDF, 18

set.col.type, 9, 19

tablePattern, 20
tableUnlist, 21

whereAre, 21
wideToLong, 22
wtdTable, 9, 23
wtdVar, 9, 24