Package 'HKRbook'

October 12, 2022

Type Package

Title Apps and Data for the Book ``Introduction to Statistics''

Version 0.1.3

Description Functions, Shiny apps and data for the book ``Introduction to Statistics'' by
Wolfgang Karl Härdle, Sigbert Klinke, and Bernd Rönz (2015) <doi:10.1007/978-3-319-17704-5>.

License GPL-3

Encoding UTF-8

Depends tools

Suggests cluster, knitr, magrittr, rmarkdown

Imports MASS, shiny, shinydashboard, shinyWidgets, shinydashboardPlus,
DT, highlight, formatR, scatterplot3d

RoxygenNote 7.2.1

VignetteBuilder knitr

NeedsCompilation no

Author Sigbert Klinke [aut, cre]

Maintainer Sigbert Klinke <sigbert@hu-berlin.de>

Repository CRAN

Date/Publication 2022-10-06 06:20:02 UTC

R topics documented:

---

checkPackages	*checkPackages*

Description

Checks if a package is installed without loading it. Returns a logical vector with TRUE or FALSE for each package checked.

Usage

```
checkPackages(
  ...,
  add = c("highlight", "formatR", "shiny", "shinydashboard", "shinydashboardPlus", "DT")
)
```

Arguments

...	character: name(s) of package
add	character: names of default packages to check (default: c("highlight", "formatR", "shiny", "shinydashboard", "shinydashboardPlus", "DT"))

Value

TRUE if successful otherweise an error will be thrown

Examples

```
checkPackages("graphics", add=NULL)          # checks if 'graphics' is installed
if (interactive()) checkPackages("graphics") # checks if 'graphics', 'shiny', ... are installed
```

---

distributionParams   *distributionParams*

Description

Computes approximate distribution parameters for the binomial, hypergeometric, Poisson, Exponential and normal distribution for a given mean (and standard deviation). With the sample and the population size the computation can be influenced.

Usage

```
distributionParams(mean, sd, n = 30, N = 60)
```

Arguments

mean	numeric: mean
sd	numeric: standard deviation (only used for the normal distribution)
n	integer: sample size (default: 30)
N	integer: population size (default: 60)

Value

a list of parameters for each distribution

Examples

```
# Compute approx. paramaters for a binomial distribution
distributionParams(mean=30*0.5, sd=sqrt(30*0.5*0.5))
```

---

gettext   *gettext*

Description

Returns a translation from loaded PO-file. If the message is not found in the PO-file then original text will be returned.

Usage

```
gettext(msg, utype = "vector")
```

Arguments

msg	character: message(s) to translate
utype	character: how to return the translated message as vector or named list

Value

translated messages

Examples

```
msgs <- c("two.sided", "less", "greater")
gettext(msgs)
# for use in Shiny "choices"
gettext(msgs, "name")
gettext(msgs, "numeric")
```

---

hm_cell	*hm_cell*

Description

- `hm_cell` or `hm_index` modify a data cell format (`fmt="%s"`), value (unnamed parameter) or style (`text_align="left"`)

- `hm_col` or `hm_row` modify a row or column format (`fmt="%s"`), value (unnamed parameter) or style (`text_align="left"`)

Usage

```
hm_cell(x, row = NULL, col = NULL, ..., byrow = FALSE)

hm_index(x, ind, ...)

hm_title(x, ...)

hm_colmargintitle(x, ...)

hm_rowmargintitle(x, ...)

hm_total(x, ...)

hm_table(x, ...)

hm_row(x, ind, ...)

hm_col(x, ind, ...)

hm_colmargin(x, ind, ...)

hm_rowmargin(x, ind, ...)

hm_tr(x, ind, ...)
```

Arguments

x	html_matrix object
row	integer: row(s) to access
col	integer: column(s) to access
...	elements to change
byrow	logical: order indices by row or column (default: FALSE)
ind	integer vector or matrix: access various (row and columns) elements (first column: row, second column: column)

Value

modified html_matrix object

Examples

```
l <- html_matrix(matrix(1:6, ncol=2))
# replace l[1,1] by NA
hm_cell(l, 1, 1, NA)
# replace l[1,1] by NA and set the text_align to center
hm_cell(l, 1, 1, NA, text_align="center")
# replace l[1,3] and l[2,1] by NA
rcind <- cbind(c(1,3), c(2, 1))
hm_index(l, rcind, NA)
# set a new title
hm_title(l, "new title")
# set a new row or column title
hm_row(l, 2, "row 2")
hm_col(l, 1, "col 1")
# set fmt by column or row
print(hm_cell(l, fmt=c("%.0f", "%.1f", "%.2f"), byrow=FALSE), which="fmt")
print(hm_cell(l, fmt=c("%.0f", "%.1f"), byrow=TRUE), which="fmt")
```

---

htmlTable	*htmlTable*

Description

Creates a HTML table from a two dimensional table object.

Usage

```
htmlTable(
  tab,
  vars = NULL,
  lines = NULL,
  cex = 1,
  title = "",
```

```
    rowsum = NULL,
    colsum = NULL,
    fmt = "%.0f",
    total = NULL,
    ...
)
```

Arguments

`tab`	two dimensional table object
`vars`	character: names of row and column variable
`lines`	character: final line (default: `NULL`)
`cex`	numeric: font size (default: `1`)
`title`	character: table title (default: `''`)
`rowsum`	character: add row sums at the right (default: `NULL`)
`colsum`	character: add column sums at the bottom (default: `NULL`)
`fmt`	character: format string for `sprintf` (default: `"%.0f"`)
`total`	character: add the grand total at the bottom left (default: `NULL`)
`...`	further parameters given to `html_matrix`

Value

html_matrix object

Examples

```
htab <- htmlTable(apply(Titanic,1:2,sum), c("Sex", "Class"), title="Titanic")
toHTML(htab, browser=interactive())
```

---

html_matrix	*html_matrix*

Description

Creates from a vector, matrix, array, or table a HTML representation of it. The HTML representation has one column and row more than the data. The additional row and column are used to have a title (top left), the column names (top), and the row names (left).

You can set the style attributes (`<td style="...">`) via `hm_cell`, `hm_title`, `hm_col`, and `hm_row`. For example: `hm_cell(hm, 1, 1, text_align="right")` will lead to (`<td style="text-align:right;">`) for the cell (1,1) and any unnamed element will change the cell value. Note: since `-` is an operator in R, we use `_` instead. Of course, you could use `"text-align"="right"`, but I'am lazy.

Usage

```
html_matrix(x, ...)

## Default S3 method:
html_matrix(
  x,
  ...,
  byrow = FALSE,
  numeric = list(text_align = "right"),
  integer = list(text_align = "right"),
  char = list(text_align = "left"),
  logical = list(text_align = "right"),
  border = "#999999"
)
```

Arguments

x	vector, matrix, array, table or html_matrix: input
...	further parameters
byrow	logical: create a row or column matrix if x is one-dimensional (default: FALSE)
numeric	list: list of HTML style properties for a cell if class(x[i,j])=="numeric" (default: list(text_align="right"))
integer	list: list of HTML style properties for a cell if class(x[i,j])=="integer" (default: list(text_align="right"))
char	list: list of HTML style properties for a cell if class(x[i,j])=="character" (default: list(text_align="left"))
logical	list: list of HTML style properties for a cell if class(x[i,j])=="logical" (default: list(text_align="right"))
border	character: vector of background color for a border cell (default: "#999999"))

Value

html_matrix returns a html_matrix, print returns invisible a character matrix

Examples

```
m <- matrix(1:6, ncol=2)
m
l <- html_matrix(m)
l
```

in_range   *in_range*

Description

Checks if x is between lower and upper,

Usage

```
in_range(x, lower, upper, rightmost.closed = TRUE, left.open = FALSE)
```

Arguments

x   numeric: vaklues to check

lower   numeric: lower bound

upper   numeric: upper bound

rightmost.closed

                 logical: if true then x<=upper is checked otherwise x<upper (default: TRUE)

left.open   logical: if true then upper<x is checked otherwise lower<=x (default: FALSE)

Value

a logical vector whether x is in range or not

Examples

```
in_range(-1:2, 0, 1)
```

is.ASCII   *is.ASCII*

Description

Checks if txt contains only ASCII characters.

Usage

```
is.ASCII(txt)
```

Arguments

txt   character: text to check

Value

logical

Examples

```
is.ASCII("Congratulations")
is.ASCII("Herzlichen Glückwunsch")
```

---

men_asso *Association*

Description

Shiny app for the association coefficients between two categorical variables. If no data are given then the default data from the book will be used. Otherwise the data will be stored as RDS file in a temporary directory.

Usage

```
men_asso(...)
```

Arguments

...	one or more data sets

Value

nothing

Examples

```
if (interactive()) men_asso()
if (interactive()) men_asso(HairEyeColor, Titanic)
```

---

men_bin *men_bin*

Description

Visualization of the probability mass and the cumulative distribution function of a binomial distribution.

Usage

```
men_bin(size = 10, prob = 0.5)
```

Arguments

size	integer: number of trials (zero or more)
prob	numeric: probability of success on each trial

Value

nothing

Examples

```
if (interactive()) men_bin()
if (interactive()) men_bin(20, 0.25)
```

---

men_ci1   *men_ci1*

Description

Shiny app for a confidence interval for the mean. The data used is considered as a population from which random samples can be drawn. If no data are given then the default data from the book will be used. Otherwise the data will be stored as RDS file in a temporary directory.

Usage

```
men_ci1(...)
```

Arguments

...	one or more data sets

Value

nothing

Examples

```
if (interactive()) men_ci1()
if (interactive()) men_ci1(stackloss)
```

---

men_ci2   *men_ci2*

Description

Shiny app for a confidence interval for the difference of two means. The data used is considered as a population from which random samples can be drawn. If no data are given then the default data from the book will be used. Otherwise the data will be stored as RDS file in a temporary directory.

Usage

```
men_ci2(...)
```

Arguments

...	one or more data sets

Value

nothing

Examples

```
if (interactive()) men_ci2()
if (interactive()) men_ci2(CO2)
```

---

men_cilen	*men_cilen*

Description

Shiny app for a length of a confidence interval for the mean.

Usage

```
men_cilen()
```

Value

nothing

Examples

```
if (interactive()) men_cilen()
```

---

men_cipi	*men_cipi*

Description

Shiny app for a confidence interval for the proportion. The data used is considered as a population from which random samples can be drawn. If no data are given then the default data from the book will be used. Otherwise the data will be stored as RDS file in a temporary directory.

Usage

```
men_cipi(...)
```

Arguments

...	one or more data sets

Value

nothing

Examples

```
if (interactive()) men_cipi()
if (interactive()) men_cipi(Titanic)
```

---

men_cisig	*men_cisig*

Description

Shiny app for a confidence interval for the variance. The data used is considered as a population from which random samples can be drawn. If no data are given then the default data from the book will be used. Otherwise the data will be stored as RDS file in a temporary directory.

Usage

```
men_cisig(...)
```

Arguments

...   one or more data sets

Value

nothing

Examples

```
if (interactive()) men_cisig()
if (interactive()) men_cisig(stackloss)
```

---

men_corr	*Correlation*

Description

Shiny app for the correlation coefficients between two numeric variables. If no data are given then the default data from the book will be used. Otherwise the data will be stored as RDS file in a temporary directory.

Usage

```
men_corr(...)
```

Arguments

... one or more data sets

Value

nothing

Examples

```
if (interactive()) men_corr()
if (interactive()) men_corr(iris)
```

---

men_die *men_die*

Description

Shiny app for detecting if a die is fair or unfair.

Usage

```
men_die()
```

Value

nothing

Examples

```
if (interactive()) men_die()
```

---

men_dot *men_dot*

Description

Shiny app for visualizing a univariate numeric variable as dotplot including univariate parameters. If no data are given then the default data from the book will be used. Otherwise the data will be stored as RDS file in a temporary directory.

Usage

```
men_dot(...)
```

Arguments

... one or more data sets

Value

nothing

Examples

```
if (interactive()) men_dot()
if (interactive()) men_dot(iris)
```

---

men_exp   *men_exp*

Description

Visualization of the density and the cumulative distribution function of a exponential distribution.

Usage

```
men_exp(rate = 1)
```

Arguments

rate   numeric: rate

Value

nothing

Examples

```
if (interactive()) men_exp()
if (interactive()) men_exp(3)
```

---

men_hall   *men_hall*

Description

Shiny app for the Monty Hall problem:

Usage

```
men_hall(pointdoor = 1, afteropen = 1)
```

Arguments

pointdoor   integer: to which door to point (default: 1)

afteropen   integer: play strategy 1=keep door, 2=change door (default: 1)

Details

Suppose you're on a game show, and you're given the choice of three doors: Behind one door is a car; behind the others, goats. You pick a door, say No. 1, and the host, who knows what's behind the doors, opens another door, say No. 3, which has a goat. He then says to you, "Do you want to pick door No. 2?" Is it to your advantage to switch your choice?

Value

nothing

Examples

```
if (interactive()) men_hall()
if (interactive()) men_hall(4, 2)
```

---

men_hist   *men_hist*

Description

Shiny app for visualizing a univariate numeric variable as histögram. If no data are given then the default data from the book will be used. Otherwise the data will be stored as RDS file in a temporary directory. #'

Usage

```
men_hist(...)
```

Arguments

...   one or more data sets

Value

nothing

Examples

```
if (interactive()) men_hist()
if (interactive()) men_hist(iris)
```

---

men_hyp   *men_hyp*

Description

Visualization of the probability mass and the cumulative distribution function of a hypergeometric distribution.

Usage

```
men_hyp(N = 60, M = 30, n = 20)
```

Arguments

N	integer: the number of black and white balls in the urn
M	integer: the number of white balls in the urn
n	integer: the number of balls drawn from the urn

Value

nothing

Examples

```
if (interactive()) men_hyp()
if (interactive()) men_hyp(50, 25, 10)
```

---

men_norm   *men_norm*

Description

Visualization of the density and the cumulative distribution function of a normal distribution.

Usage

```
men_norm(mean = 0, sd2 = 1)
```

Arguments

mean	numeric: mean
sd2	numeric: variance

Value

nothing

Examples

```
if (interactive()) men_norm()
if (interactive()) men_norm(1, 0.5)
```

---

men_parn	*men_parn*

Description

Shiny app for the distribution of sample parameters. The data used is considered as a population from which random samples can be drawn. If no data are given then the default data from the book will be used. Otherwise the data will be stored as RDS file in a temporary directory.

Usage

```
men_parn(...)
```

Arguments

... one or more data sets

Value

nothing

Examples

```
if (interactive()) men_parn()
if (interactive()) men_parn(iris)
```

---

men_poi	*men_poi Visualization of the probability mass and the cumulative distribution function of a Poisson distribution.*

Description

men_poi Visualization of the probability mass and the cumulative distribution function of a Poisson distribution.

Usage

```
men_poi(lambda = 5)
```

Arguments

lambda numeric: (non-negative) mean

Value

nothing

Examples

```
if (interactive()) men_poi()
if (interactive()) men_poi(3)
```

---

men_rank   *men_rank*

Description

Shiny app for the rank correlation coefficients between two ordered variables. If no data are given then the default data from the book will be used. Otherwise the data will be stored as RDS file in a temporary directory.

Usage

```
men_rank(...)
```

Arguments

...   one or more data sets

Value

nothing

Examples

```
if (interactive()) men_rank()
if (interactive()) {
  data("plantTraits", package="cluster")
  men_rank(plantTraits)
}
```

---

men_regr   *men_regr*

Description

Shiny app for a simple linear regression. If no data are given then the default data from the book will be used. Otherwise the data will be stored as RDS file in a temporary directory.

Usage

```
men_regr(...)
```

Arguments

```
...                   one or more data sets
```

Value

nothing

Examples

```
if (interactive()) men_regr()
if (interactive()) men_regr(stackloss)
```

---

men_tab   *Frequency tables*

Description

Shiny app for frequency tables for two categorical variables. If no data are given then the default data from the book will be used. Otherwise the data will be stored as RDS file in a temporary directory.

Usage

```
men_tab(...)
```

Arguments

```
...                   one or more data sets
```

Value

nothing

Examples

```
if (interactive()) men_tab()
if (interactive()) men_tab(HairEyeColor, Titanic)
```

---

men_terr   *men_terr*

Description

Shiny app for a test for the true mean. The data used is considered as a population from which random samples can be drawn. If no data are given then the default data from the book will be used. Otherwise the data will be stored as RDS file in a temporary directory.

Usage

```
men_terr(...)
```

Arguments

...	one or more data sets

Value

nothing

Examples

```
if (interactive()) men_terr()
if (interactive()) men_terr(iris)
```

---

men_time   *men_time*

Description

Shiny app for classical time series analysis If no data are given then the default data from the book will be used. Otherwise the data will be stored as RDS file in a temporary directory.

Usage

```
men_time(...)
```

Arguments

...	one or more time series

Value

    nothing

Examples

```
if (interactive()) men_time()
if (interactive()) men_time(co2)
```

---

  men_tmu1   *men_tmu1*

Description

    Shiny app for a test for the true mean. The data used is considered as a population from which random samples can be drawn. If no data are given then the default data from the book will be used. Otherwise the data will be stored as RDS file in a temporary directory.

Usage

```
men_tmu1(...)
```

Arguments

    ...   one or more data sets

Value

    nothing

Examples

```
if (interactive()) men_tmu1()
if (interactive()) men_tmu1(iris)
```

---

  men_tmu2   *men_tmu2*

Description

    Shiny app for a test on difference of two true means. The data used is considered as a population from which random samples can be drawn. If no data are given then the default data from the book will be used. Otherwise the data will be stored as RDS file in a temporary directory.

Usage

```
men_tmu2(...)
```

Arguments

...   one or more data sets

Value

nothing

Examples

```
if (interactive()) men_tmu2()
if (interactive()) men_tmu2(CO2)
```

---

men_tprop   *men_tprop*

Description

Shiny app for test on the proportion. The data used is considered as a population from which random samples can be drawn. If no data are given then the default data from the book will be used. Otherwise the data will be stored as RDS file in a temporary directory.

Usage

```
men_tprop(...)
```

Arguments

...   one or more data sets

Value

nothing

Examples

```
if (interactive()) men_tprop()
if (interactive()) men_tprop(Titanic)
```

---

men_vis   *men_vis*

Description

Shiny app for visualizing the univariate numeric variable, e.g. boxplot, stripchart, histogram, and
cumulative distribution function. If no data are given then the default data from the book will be
used. Otherwise the data will be stored as RDS file in a temporary directory.

Usage

```
men_vis(...)
```

Arguments

...	one or more data sets

Value

nothing

Examples

```
if (interactive()) men_vis()
if (interactive()) men_vis(iris)
```

---

mmstat.attrVar   *mmstat.attrVar*

Description

Returns the parameters for a variable. If `type="numeric"` then descriptive measures will be re-
turned. Otherwise absolute and relative frequencies will be returned. For using a subset of observa-
tion set `index`.

Usage

```
mmstat.attrVar(var, type, index = NULL)
```

Arguments

var	vector: values of a mmstat variable
type	character: type of values, allowed are `numvars`, `binvars`, `ordvars` or `facvars`
index	integer: observation numbers to use for computation, default is to use all obser-vations

Value

descriptive measures

Examples

```
# make sure that no other data sets are loaded
mmstat.set(datasets=NULL)
mmstat.getDataNames(mmstat.rds("CARS"))
# summary of first numeric variable in first data set in mmstat
var <- mmstat.getVar(1, 1, 'numeric')
mmstat.attrVar(var, "numeric")
# summary of first factor variable in first data set in mmstat
var <- mmstat.getVar(1, 1, 'factor')
mmstat.attrVar(var, 'factor')
```

---

mmstat.axis *mmstat.axis*

Description

Based on `range` the position of the labels are determined and the axis is plotted.

Usage

```
mmstat.axis(side, range, at, labels, ...)
```

Arguments

side	an integer specifying which side of the plot the axis is to be drawn on. The axis is placed as follows: 1=below, 2=left, 3=above and 4=right.
range	range: a data range
at	the points at which tick-marks are to be drawn. Non-finite (infinite, NaN or NA) values are omitted. By default (when NULL) tickmark locations are computed, see 'Details' below.
labels	this can either be a logical value specifying whether (numerical) annotations are to be made at the tickmarks, or a character or expression vector of labels to be placed at the tickpoints. (Other objects are coerced by [as.graphicsAnnot](#).) If this is not logical, at should also be supplied and of the same length. If labels is of length zero after coercion, it has the same effect as supplying TRUE.
...	further parameters to [graphics::axis](#)

Value

adds a axis to a plot

Examples

```
oldpar <- par(mfrow=c(1,2))
plot(iris[,1])
plot(iris[,1], axes=FALSE)
mmstat.axis(2, iris[,1])
par(oldpar)
```

---

mmstat.baraxis   *mmstat.baraxis*

Description

Based on range the position of the labels are determined and the axis is plotted.

Usage

```
mmstat.baraxis(side, range, at, labels, ...)
```

Arguments

side	an integer specifying which side of the plot the axis is to be drawn on. The axis is placed as follows: 1=below, 2=left, 3=above and 4=right.
range	range: a data range
at	the points at which tick-marks are to be drawn. Non-finite (infinite, NaN or NA) values are omitted. By default (when NULL) tickmark locations are computed, see 'Details' below.
labels	this can either be a logical value specifying whether (numerical) annotations are to be made at the tickmarks, or a character or expression vector of labels to be placed at the tickpoints. (Other objects are coerced by `as.graphicsAnnot`.) If this is not logical, at should also be supplied and of the same length. If labels is of length zero after coercion, it has the same effect as supplying TRUE.
...	further parameters to graphics::axis

Value

adds a axis to a plot

Examples

```
oldpar <- par(mfrow=c(1,2))
x <- 0:15
px <- dbinom(x, 10, 0.5)
plot(x, px, type="h")
plot(x, px, type="h", axes=FALSE)
mmstat.baraxis(1, range(x), at=x, labels=as.character(x))
par(oldpar)
```

mmstat.dec   *mmstat.dec*

Description

Computes the number of the significant digits based on the smallest non-zero difference of the sorted data.

Usage

```
mmstat.dec(x, ord = NULL)
```

Arguments

x	numeric: data vector
ord	index: subset of the ordered data (default: NULL)

Value

The number of significant digits and (the subset of) the order of the data.

Examples

```
x <- rnorm(20)
d <- mmstat.dec(x)
# create strings so that they are unique (if they were)
sprintf("%.*f", d$dec, x)
```

mmstat.getDataNames   *mmstat.getDataNames*

Description

Returns the names of data sets and stores them in the internal environment. The name of the data set is base name without extension.

Usage

```
mmstat.getDataNames(...)
```

Arguments

...	character: names of the data sets.

Value

the names of the data sets

Examples

```
files <- mmstat.rds("HAIR.EYE.COLOR", "TITANIC")
mmstat.getDataNames(files)
```

---

mmstat.getDatasets *mmstat.getDatasets*

Description

Reads data set(s) into the mmstat object.

Usage

```
mmstat.getDatasets(...)
```

Arguments

...   character: file name(s) of RDS data file(s)

Value

the names of the data set(s)

Examples

```
# not used, deprecated??
```

---

mmstat.getLog *mmstat.getLog*

Description

Returns the internal log message as HTML. In a Shiny app the log message are updated every 100 milliseconds

Usage

```
mmstat.getLog(session)
```

Arguments

session   session object

Value

HTML code

Examples

```
# will work only in A Shiny app
if (interactive()) {
  require("shiny")
  ui <- fluidPage(
    titlePanel("getLog example"),
      sidebarLayout(sidebarPanel(
        actionButton("quit", "Quit")),
        mainPanel(textOutput("log"))
  )
 )
#
  server <- function(input, output, session) {
    observeEvent(input$quit, { stopApp() })
    output$log <- renderText({ mmstat.getLog(session) })
  }
#
  shinyApp(ui, server)
}
```

---

mmstat.getValues   *mmstat.getValues*

Description

`mmstat.getValues` returns a list with named elements. If the parameter is NULL then a default value stored `local` will be used. `mmstat.getValue` returns a value. If the parameter is NULL or NA then `def` will be returned

Usage

```
mmstat.getValues(local, ...)

mmstat.getValue(val, def)
```

Arguments

local	list: default values for the named parameter
...	list of named parameters
val	value for a parameter
def	default value for a parameter

Value

a list of requested parameters

Examples

```
def <- list(a=3)
mmstat.getValues(def, b=3, a=NULL)
mmstat.getValue(NA, 5)
mmstat.getValue(NULL, 5)
mmstat.getValue(3, 5)
```

---

mmstat.getVar   *mmstat.getVar*

Description

Returns the from a data set a variable from the given type.

Usage

```
mmstat.getVar(
  dataname = NULL,
  varname = NULL,
  vartype = NULL,
  na.action = stats::na.omit
)
```

Arguments

dataname	integer: number of data set
varname	integer: number of variable
vartype	character: variable type, one of numeric, binary, ordered, or factor
na.action	function: indicate what should happen when the data contain NAs (default: stats::na.omit)

Value

a variable of the given type

Examples

```
# make sure that no other data sets are loaded
mmstat.set(datasets=NULL)
mmstat.getDataNames(mmstat.rds("CARS"))
# summary of first numeric variable in first data set in mmstat
str(mmstat.getVar(1, 1, 'numeric'))
# summary of first factor variable in first data set in mmstat
str(mmstat.getVar(1, 1, 'factor'))
```

---

mmstat.getVariableNames

mmstat.getVariableNames

Description

Returns all variable names of data set stored in the internal environment.

Usage

```
mmstat.getVariableNames(name)
```

Arguments

name   character or numeric: name or index of data set

Value

vector of names

Examples

```
# Delete all stored data sets
mmstat.set(datasets=NULL)
# Load CAR data set into mmstat
mmstat.getDataNames(mmstat.rds("CARS"))
# Extract names of all variables
mmstat.getVariableNames(1)
```

---

mmstat.getVarNames   *mmstat.getVarNames*

Description

Returns the variable names of a specific type from a mmstat data set.

Usage

```
mmstat.getVarNames(dataname, vartype, which = NULL)
```

Arguments

dataname   character: name of data set

vartype   character: type of variable, either numeric, ordered, factor, or binary

which   integer: index number

Value

a vector or element of variable names which have the type `vartype`

Examples

```
# Load CAR data set into mmstat
mmstat.getDataNames(mmstat.rds("CARS"))
# Extract names of numeric variables
mmstat.getVarNames(1, "numeric")
```

---

mmstat.lang   *mmstat.lang*

Description

Loads a PO file for a translation into the internal environment.

Usage

```
mmstat.lang(pof = NULL)
```

Arguments

pof   character: file name

Value

nothing

Examples

```
mmstat.lang()
```

---

mmstat.log   *mmstat.log*

Description

Writes a message into the internal log.

Usage

```
mmstat.log(txt)
```

Arguments

txt   character: message to write

Value

nothing

Examples

```
mmstat.log("Test")
```

---

mmstat.math   *mmstat.math*

Description

Returns a math expression based on HTML special characters notation.

Usage

```
mmstat.math(txt)
```

Arguments

txt   character: input text

Value

expression

Examples

```
mmstat.math(" &bar(X);~&N(mu[0], sigma^2/n); ")
mmstat.math("&H[0];: &mu==mu[0]; vs. &H[1];: &mu!=mu[0]; ")
```

---

mmstat.merge   *mmstat.merge*

Description

Computes a new range from by a union of the two ranges.

Usage

```
mmstat.merge(range1, range2)
```

Arguments

range1   range: first range

range2   range: second range

Value

new range

Examples

```
mmstat.merge(c(0,1), c(0.5, 2)) # returns c(0, 2)
```

---

mmstat.plotTestRegions

mmstat.plotTestRegions

Description

Plots the test regions in a plot

Usage

```
mmstat.plotTestRegions(
  crit,
  xlim,
  ylim,
  cex,
  close = FALSE,
  col = "black",
  label = NULL,
  pos = 1
)
```

Arguments

crit	numeric(2): critical value(s)
xlim	numeric(2): the x limits of the plot
ylim	numeric(2): the y limits of the plot
cex	numeric: amount by which plotting text should be magnified relative to the default
close	logical: should the region box be closed by vertical lines (default: FALSE)
col	color: pecification for the default plotting color (default: "black")
label	unused
pos	unused

Value

adds test regions to a plot

Examples

```
x  <- (-30:30)/10
px <- dnorm(x)
plot(x, px, type="l", ylim=c(-0.25, max(px)), xlim=range(x))
mmstat.plotTestRegions(crit=c(-1.96, +1.96), xlim=range(x), ylim=c(-0.2, -0.1), cex=1)
```

---

mmstat.pos   *mmstat.pos*

Description

Returns a linear interpolation based on `minmax`.

Usage

```
mmstat.pos(minmax, pos)
```

Arguments

minmax	numeric(2): range to interpolate between
pos	numeric: proportion(s) to interpolate, usually between zero and one

Value

interpolated values

Examples

```
mmstat.pos(c(0,360), 0.5)
```

---

mmstat.range   *mmstat.range*

Description

Computes a range from several R objects by union.

Usage

```
mmstat.range(...)
```

Arguments

...	R objects

Value

range

Examples

```
mmstat.range(-5:5, 0:10) # returns c(-5, 10)
```

---

mmstat.rds   *mmstat.rds*

Description

Returns the full file names of all or specific data set that come with the package.

Usage

```
mmstat.rds(...)
```

Arguments

...   names of data sets

Value

full file names

Examples

```
mmstat.rds()                          # return all RDS file that come with the package
mmstat.rds("HAIR.EYE.COLOR", "TITANIC") # location of specific data sets
```

---

mmstat.round.down   *mmstat.round.down*

Description

Rounds down.

Usage

```
mmstat.round.down(x, digits = 0)
```

Arguments

x   numeric: values for rounding

digits   numeric: digits for rounding (default: 0)

Value

down rounded values

Examples

```
x <- runif(5)
cbind(x, mmstat.round.down(x, 1))
```

---

mmstat.round.up   *mmstat.round.up*

Description

Rounds up.

Usage

```
mmstat.round.up(x, digits = 0)
```

Arguments

x	numeric: values for rounding
digits	numeric: digits for rounding (default: 0)

Value

uprounded values

Examples

```
x <- runif(5)
cbind(x, mmstat.round.up(x, 1))
```

---

mmstat.set   *mmstat.set*

Description

mmstat.set sets one (or more) parameter to the internal environment. mmstat.get return one or more parameters from the internal environment.

Usage

```
mmstat.set(...)

mmstat.get(...)
```

Arguments

...   named parameters with values or names

Value

nothing

Examples

```
mmstat.set(debug=0)
mmstat.get("debug")
mmstat.get("debug", "shiny") # returns a list
```

---

mmstat.sliderInput   *mmstat.sliderInput*

Description

A modified `sliderInput` for `mmstat` which supports user defined tick marks.

Usage

```
mmstat.sliderInput(...)
```

Arguments

...   parameters for shiny::sliderInput

Value

the HTML output

Examples

```
ticks <- c(80, 85, 90, 95, 98, 99, 99.5, 99.9)
mmstat.sliderInput("id", "label", min=1, max=length(ticks), value=3, step=1, ticks=ticks)
```

---

mmstat.ticks *mmstat.ticks*

Description

Returns tick marks for a log based scale between `nmin` and `nin`.

Usage

```
mmstat.ticks(nin, nmin = 3, tin = 11)
```

Arguments

nin	integer: maximun of scale
nmin	integer: minimun of scale
tin	integer: number of desired tick marks

Value

vector of tick marks

Examples

```
mmstat.ticks(506)
```

---

mmstat.ui.call *mmstat.ui.call*

Description

Calls the underlying Shiny UI element (`selectInput, ...`).

Usage

```
mmstat.ui.call(inputId, ...)
```

Arguments

inputId	character: the input slot called
...	further parameters given to the call

Value

whatever the call to the underlying Shiny UI element returns

Examples

```
mmstat.ui.elem(inputId="alpha", type="significance")
mmstat.ui.call("alpha")
```

---

mmstat.ui.elem   *mmstat.ui.elem*

Description

Adds a new UI element to the app interface. The following types from Shiny are allowed:

- actionButton,
- checkboxInput,
- checkboxGroupInput,
- dateInput,
- dateRangeInput,
- fileInput,
- helpText,
- numericInput,
- radioButtons,
- selectInput,
- sliderInput,
- submitButton, and
- textInput.

Additionally some standard statistical UI elements are supported (links go to the Shiny element used):

- sampleSize,
- drawSample,
- speedSlider,
- confidenceLevel,
- significance,
- testHypotheses,
- dataSet,
- variable1,
- variableN, and
- fontSize.

Partially these elements have default settings which can be overwritten.

Usage

```
mmstat.ui.elem(inputId, type, ...)
```

Arguments

inputId	character: input slot that will be used to access the value
type	character: element type
...	further named parameter to Shiny UI elements

Value

nothing

Examples

```
mmstat.ui.elem(inputId="alpha", type="significance")
```

---

mmstat.ui.update   *mmstat.ui.update*

Description

Call for a update of an underlying Shiny UI element (selectInput, ...).

Usage

```
mmstat.ui.update(inputId, ...)
```

Arguments

inputId	character: the input slot called
...	further parameters given to the call

Value

whatever the update to the underlying Shiny UI element returns

Examples

```
mmstat.ui.elem(inputId="alpha", type="significance")
mmstat.ui.call("alpha")
```

---

mmstat.warn   *mmstat.warn*

Description

Writes a warning text into the log object in the internal mmstat object.

Usage

```
mmstat.warn(cond, txt)
```

Arguments

cond	logical: condition to test
txt	character: text to write if cond is true

Value

nothing

Examples

```
mmstat.warn(TRUE, "just a true seen")
```

---

resetpar   *resetpar*

Description

Resets the par if necessary.

Usage

```
resetpar(oldpar)
```

Arguments

oldpar	graphical parameters

Value

nothing

Examples

```
par("mar")
oldpar <- par(no.readonly = TRUE)
par(mar = c(0,0,0,0))
par("mar")
resetpar(oldpar)
par("mar")
```

---

stopif *stopif*

Description

A equivalent to stopifnot: if cond is TRUE then a error is thrown.

Usage

```
stopif(cond, txt)
```

Arguments

cond	logical: condition to test
txt	character: error message

Value

nothing

Examples

```
if (interactive()) stopif(1+1==2, "1+1 can not be 2, this is fake science!")
```

---

table2dataframe *table2dataframe*

Description

Converts a table to a full data frame.

Usage

```
table2dataframe(tab, ...)
```

Arguments

tab	table: contingency table
...	further parameters given to base::as.data.frame.table

Value

a data frame with `sum(tab)` rows and `length(dim(tab))` cols

Examples

```
table2dataframe(Titanic)
```

`toHTML.html_matrix`   *toHTML*

Description

Returns a HTMl representation of a matrix and optionally shows the result in the browser. If you decide to view the result in a browser then the HTML will be written to a temporary file and `utils::browseURL()` called

Usage

```
## S3 method for class 'html_matrix'
toHTML(x, browser = FALSE, ...)

## S3 method for class 'table'
toHTML(x, browser = FALSE, ...)

## S3 method for class 'matrix'
toHTML(x, browser = FALSE, ...)
```

Arguments

x	html_matrix object
browser	logical: show HTML in a browser (default: FALSE)
...	further parameters to `utils::browseURL()`

Value

html_matrix object

Examples

```
library("tools")
m    <- matrix(1:12, ncol=4)
hm   <- html_matrix(m)
html <- toHTML(hm, browser=interactive())
```

---

toRDS   *toRDS*

Description

Saves one or more data sets in RDS format to a temporary directory (`tmpdir()`). Data sets must have the class `ts` or something that can be converted to a data frame, e.g. `matrix`, `table`, etc.

Usage

```
toRDS(...)
```

Arguments

```
...              data sets to save
```

Value

returns the name of the created files

Examples

```
toRDS(Titanic) # saves to tempdir/Titanic.rds
```

---

ucfirst   *ucfirst*

Description

Uppercases the first character in `txt`.

Usage

```
ucfirst(txt)
```

Arguments

```
txt              character:
```

Value

character

Examples

```
ucfirst("hello world")
```

---

zebra *zebra*

Description

zebra

Usage

```
zebra(x, col = c("#FFFFFF", "#CCCCCC"), byrow = TRUE)
```

Arguments

x	html_matrix object
col	a vector of colors to zebra with (default:c("#FFFFFF", "#CCCCCC"))
byrow	logical: zebra by row or by column (default: TRUE)

Value

html_matrix object

Examples

```
library("magrittr")
library("tools")
m    <- matrix(1:12, ncol=4)
hm   <- html_matrix(m) %>% zebra()
html <- toHTML(hm, browser=interactive())
```

---

zzz *zzz*

Description

Checks if all necessary packages are installed.

Usage

```
zzz()
```

Value

a logical vector which of the required packages are available

Examples

```
zzz()
```

Index