Package: stRoke (via r-universe)

October 26, 2024

Title Clinical Stroke Research

Version 24.10.1

Description A collection of tools for clinical trial data management and analysis in research and teaching. The package is mainly collected for personal use, but any use beyond that is encouraged. This package has migrated functions from 'agdamsbo/daDoctoR', and new functions has been added. Version follows months and year. See NEWS/Changelog for release notes. This package includes sampled data from the TALOS trial (Kraglund et al (2018) <doi:10.1161/STROKEAHA.117.020067>). The win_prob() function is based on work by Zou et al (2022) <doi:10.1161/STROKEAHA.121.037744>. The age_calc() function is based on work by Becker (2020) <doi:10.18637/jss.v093.i02>.

URL https://agdamsbo.github.io/stRoke/, https://github.com/agdamsbo/stRoke

BugReports https://github.com/agdamsbo/stRoke/issues

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

LazyData true

Suggests knitr, rmarkdown, testthat, here, spelling, usethis, pak, roxygen2, devtools

Config/testthat/edition 3

Imports calendar, dplyr, ggplot2, grDevices, gtsummary, lubridate, MASS, rankinPlot, stats, tidyr, utils, tibble, tidyselect

Depends R (>= 3.5.0)

VignetteBuilder knitr

Language en-US

NeedsCompilation no

Author Andreas Gammelgaard Damsbo [aut, cre] (<https://orcid.org/0000-0002-7559-1154>)

Maintainer Andreas Gammelgaard Damsbo <agdamsbo@clin.au.dk>

Repository CRAN

Date/Publication 2024-10-25 06:50:01 UTC

Contents

`add_padding` *MOVED Add padding to string*

Description

MOVED Add padding to string

Usage

```
add_padding(
  d,
  length = NULL,
  after = FALSE,
  pad = "0",
  lead = NULL,
  tail = NULL
)
```

Arguments

d	vector of strings or numbers
length	final string length
after	if padding should be added after as opposed to default before
pad	padding string of length 1
lead	leading string for all. Number or character vector. Cycled.
tail	tailing string for all. Number or character vector. Cycled.

Value

vector or character strings of same length.

Examples

```
add_padding(sample(1:200,5),tail="AA",lead=c(2,3,"e"))
```

---

age_calc *Calculate age in years, months, or days*

Description

Calculate age in years, months, or days

Usage

```
age_calc(dob, enddate = Sys.Date(), units = "years", precise = TRUE)
```

Arguments

dob	Date of birth
enddate	End date for age calculation (default is Sys.Date())
units	Units for age calculation (default is "years"). Can be c("days", "months", "years")
precise	Option to calculate age precisely (default is TRUE)

Value

numeric vector length 1

Source

doi:10.18637/jss.v093.i02

References

Becker, J.P. (2020). eeptools: An R Package for Teaching and Learning Ecology and Evolutionary Biology. Journal of Statistical Software, 93(2), 1-27.

Examples

```
trunc(age_calc(as.Date("1945-10-23"),as.Date("2018-09-30")))
```

---

chunks_of_n *MOVED Split to chunks of size n*

Description

MOVED Split to chunks of size n

Usage

```
chunks_of_n(d, n, label = NULL, even = FALSE, pattern = NULL)
```

Arguments

d	data. Can be vector or data frame.
n	number of chunks
label	naming prefix for chunk names
even	boolean to set if size of chunks should be evenly distributed.
pattern	regex pattern to extract names from provided vector. If data frame, will assume first column is name.

Value

List of length n

Examples

```
tail(chunks_of_n(seq_len(100),7),3)
tail(chunks_of_n(seq_len(100),7,even=TRUE),3)
ds <- data.frame(nm=paste0("Sub",
add_padding(rownames(stRoke::talos))),stRoke::talos)
head(chunks_of_n(ds,7,pattern="Sub[0-9]{3}",label="grp"),2)
## Please notice that no sorting is performed. This is on purpose to preserve
## original sorting. If sorting is intended, try something like this:
ds[order(ds$nm),] |> chunks_of_n(7,pattern="Sub[0-9]{3}",label="grp") |>
head(2)
```

---

ci_plot *Confidence interval plot with point estimate*

Description

Horizontal forest plot of point estimate with confidence intervals. Includes dichotomous or olr, depending on number of levels in "x". Title and axis labels can be added to the ggplot afterwards.

Usage

```
ci_plot(
  ds,
  x = NULL,
  y = NULL,
  vars = NULL,
  dec = 3,
  lbls = NULL,
  title = NULL,
  method = "auto"
)
```

Arguments

ds	main input, either data set or logistic model
x	text string of main exposure variable
y	text string of outcome variable
vars	variables for multivariate analysis.
dec	Decimals in labels
lbls	Labels for variable names
title	Plot title. Can be specified later.
method	Character vector. The method for the regression. Can be c("auto", "model").

Value

ggplot element

Examples

```
# Auto plot
data(talos)
talos[,"mrs_1"]<-factor(talos[,"mrs_1"],ordered=TRUE)
ci_plot(ds = talos, x = "rtreat", y = "mrs_1",
vars = c("hypertension","diabetes"))
## Model plot
# iris$ord<-factor(sample(1:3,size=nrow(iris),replace=TRUE),ordered=TRUE)
# lm <- MASS::polr(ord~., data=iris, Hess=TRUE, method="logistic")
# ci_plot(ds = lm, method="model")
```

`color_plot` *Plot color examples with contrasting text*

Description

Plots color examples with contrasting text. Parameters are passed to contrast_text.

Usage

```
color_plot(
  colors,
  labels = TRUE,
  borders = NULL,
  cex_label = 1,
  ncol = NULL,
  ...
)
```

Arguments

`colors`	Vector of colors to plot
`labels`	Show color names. Default is TRUE
`borders`	Border parameter for 'rect()' function. Default is NULL
`cex_label`	Label size. Default is 1.
`ncol`	Desired number of columns. Default is ceiling of square root to the length of 'colors' vector provided.
`...`	Parameters for the

Value

base plot

Examples

```
par(bg=NULL)
colors <- sample(colors(),size = 20)
color_plot(colors, method="relative")
```

`contrast_text` *MOVED Contrast Text Color*

Description

Calculates the best contrast text color for a given background color.

Usage

```
contrast_text(
  background,
  light_text = "white",
  dark_text = "black",
  threshold = 0.5,
  method = "perceived_2",
  ...
)
```

Arguments

`background`	A hex/named color value that represents the background.
`light_text`	A hex/named color value that represents the light text color.
`dark_text`	A hex/named color value that represents the dark text color.
`threshold`	A numeric value between 0 and 1 that is used to determine the luminance threshold of the background color for text color.
`method`	A character string that specifies the method for calculating the luminance. Three different methods are available: c("relative","perceived","perceived_2")
`...`	parameter overflow. Ignored.

Details

This function aids in deciding the font color to print on a given background. The function is based on the example provided by teppo: https://stackoverflow.com/a/66669838/21019325. The different methods provided are based on the methods outlined in the StackOverflow thread: https://stackoverflow.com/questions/59621
to-determine-perceived-brightness-of-rgb-color

Value

A character string that contains the best contrast text color.

Examples

```
contrast_text(c("#F2F2F2", "blue"))

contrast_text(c("#F2F2F2", "blue"), method="relative")
```

`cprs` *Data frame of 200 cpr numbers*

Description

This is just a repeated sample of 8 synthesized cpr-numbers for testing purposes.

Usage

```
data(cprs)
```

Format

A data frame with 200 rows and 1 variable:

cpr Mixed format cpr-numbers, characters

See Also

https://da.wikipedia.org/wiki/231045-0637

`cpr_check` *CPR check*

Description

Checking validity of cpr number. Vectorised.

Usage

```
cpr_check(cpr)
```

Arguments

`cpr`	cpr-numbers as ddmmyy"-."xxxx or ddmmyyxxxx. Also mixed formatting. Vector or data frame column.

Value

Logical vector of cpr validity

Examples

```
fsd<-c("2310450637", "010190-2000", "010115-4000",
"300450-1030","010150-4021")
cpr_check("2310450637")
cpr_check(fsd)
all(cpr_check(fsd))
```

---

cpr_dob *Extracting date of birth from CPR*

Description

For easy calculation. Does not handle cprs with letters (interim cpr)

Usage

```
cpr_dob(cpr, format = "%d-%m-%Y")
```

Arguments

cpr	cpr-numbers as ddmmyy"-."xxxx or ddmmyyxxxx. Also mixed formatting. Vector or data frame column.
format	character string of dob date format. Default is "%d-%m-%Y".

Value

character vector

Examples

```
cpr_dob("231045-0637")
fsd<-c("2310450637", "010190-2000", "010115-4000",
"300450-1030","010150-4021")
cpr_dob(fsd)
```

---

cpr_female *Determine female sex from CPR*

Description

Just checking if last number of a string is equal or not.

Usage

```
cpr_female(cpr)
```

Arguments

cpr	Vector. cpr-numbers as ddmmyy"-."xxxx or ddmmyyxxxx. Also mixed formatting. Vector or data frame column.

Value

Logical vector

Examples

```
cpr_female(stRoke::cprs[,1])
```

---

ds2dd DEPRECATED *Moved to REDCapCAST::ds2dd()* | *Data set to data dictionary function*

Description

DEPRECATED Moved to REDCapCAST::ds2dd() | Data set to data dictionary function

Usage

```
ds2dd(
  ds,
  record.id = "record_id",
  form.name = "basis",
  field.type = "text",
  field.label = NULL,
  include.column.names = FALSE,
  metadata = stRoke::metadata_names
)
```

Arguments

ds	data set
record.id	name or column number of id variable, moved to first row of data dictionary, character of integer. Default is "record_id".
form.name	vector of form names, character string, length 1 or length equal to number of variables. Default is "basis".
field.type	vector of field types, character string, length 1 or length equal to number of variables. Default is "text.
field.label	vector of form names, character string, length 1 or length equal to number of variables. Default is NULL and is then identical to field names.
include.column.names	Flag to give detailed output including new column names for original data set for upload.
metadata	Metadata dataframe. Default is the included stRoke::metadata_names.

Value

data.frame or list of data.frame and vector

Examples

```
talos$id <- seq_len(nrow(talos))
ds2dd(talos, record.id="id",include.column.names=FALSE)
```

`files_filter` *MOVED Filter files in a folder*

Description

This function filters files in a folder based on the provided filter.

Usage

```
files_filter(folder.path, filter.by, full.names = TRUE)
```

Arguments

`folder.path`	character. Path of the folder to be filtered
`filter.by`	character. Filter to be applied on the files
`full.names`	logical. Whether to return full file names or not

Value

character vector. Filtered files

Examples

```
# Gives path to files/folders with "tests" in the name in the
# working directory
files_filter(getwd(),"tests")
```

generic_stroke *Generic stroke study outcome*

Description

Includes table 1, grotta bars and ordinal logistic regression plot. Please just use this function for illustration purposes. To dos: modify grottaBar and include as own function.

Usage

```
generic_stroke(df, group, score, strata = NULL, variables = NULL)
```

Arguments

df	Data set as data frame
group	Variable to group by
score	Outcome measure variable
strata	Optional variable to stratify by
variables	String of variable names to include in adjusted OLR-analysis

Value

Returns list with three elements

Examples

```
# generic_stroke(df = stRoke::talos, group = "rtreat", score = "mrs_6",
# variables = c("hypertension","diabetes","civil"))
```

index_plot *Plot multidimensional cognitive test scores*

Description

Plot index scores from five dimensional cognitive testing. Includes option to facet.

Usage

```
index_plot(
  ds,
  id = "id",
  sub_plot = "_is",
  scores = c("_is", "_lo", "_up", "_per"),
 dom_names = c("immediate", "visuospatial", "verbal", "attention", "delayed", "total"),
  facet.by = NULL
)
```

Arguments

ds	complete data frame
id	colname of id column. Base for colouring
sub_plot	main outcome scores variable to plot
scores	variables to subset for plotting. Has to follow standard naming (is to be changed)
dom_names	domain names for axis naming
facet.by	variable to base facet_grid on

Value

ggplot element

Examples

```
index_plot(stRoke::score[score$event=="A",])
```

---

label_select *Helper function for labels in gtsummary*

Description

Function to select labels from list of label pairs (format: age~"Age"). Alternative is to use attributes, eg from library(Hmisc).

Usage

```
label_select(lst, vec)
```

Arguments

lst	List of variables and labels (format: age~"Age")
vec	Vector of variables to be subset from the list

Value

List of labels ordered like vec, formatted like lst

Examples

```
vars<-c("hypertension", "diabetes", "mrs_1")
labels_all<-list(rtreat~"Trial treatment",
civil~"Cohabitation",
diabetes~"Known diabetes",
hypertension~"Known hypertension",
mrs_1~"One month mRS",
mrs_6~"Six months mRS",
```

```
'[Intercept]'~"Intercept")
label_select(labels_all,vars)

## With gtsummary::tbl_summary()
#stRoke::talos[vars] |>
#gtsummary::tbl_summary(label = label_select(labels_all,vars))
```

`metadata_names` *Vector of REDCap metadata headers*

Description

Vector of REDCap metadata headers

Usage

```
data(metadata_names)
```

Format

Vector of length 18 with REDCap metadata headers:

metadata_names characterstrings

See Also

project-redcap(dot)org (currently the certificate is broken)

`mfi_domains` *MFI domain score calculator*

Description

MFI domain score calculator

Usage

```
mfi_domains(
  ds,
  reverse = TRUE,
  reverse.vars = c(2, 5, 9, 10, 13, 14, 16, 17, 18, 19)
)
```

Arguments

`ds`	data set of MFI scores, 20 columns
`reverse`	reverse scoring
`reverse.vars`	variables/columns to reverse

Value

tibble of domain scores

Examples

```
mfi_mess <- data.frame(matrix(
sample(c(" 1. ", "2. -A", "3.", " 4  ", "5.", NA),200,replace=TRUE),ncol=20))
mfi_mess |> mfi_domains()
```

---

multi_rev *Reverses relevant MFI subscores*

Description

Reverses relevant MFI subscores

Usage

```
multi_rev(d, var)
```

Arguments

d	data frame or tibble
var	numeric vector of indices of columns to reverse

Value

data.frame or tibble depending on input

Examples

```
# rep_len(sample(1:5),length.out = 100) |> matrix(ncol=10) |> multi_rev(2:4)
```

---

n_chunks *Splits in n chunks*

Description

Splits in n chunks

Usage

```
n_chunks(d, n, ...)
```

Arguments

d	data
n	number of chunks
...	arguments passed to internal chunks_of_n()

Value

List of chunks

Examples

```
lengths(n_chunks(d=seq_len(100),n=7,even=TRUE))
lengths(n_chunks(d=seq_len(100),n=7,even=FALSE))
```

---

pase *Data frame with sample data of PASE score questionnaire*

Description

Contains non-identifiable organic trial data. Sample data labels are in Danish.

Usage

```
data(pase)
```

Format

A data frame with 200 rows and 21 variables:

sample_pase01 item 01, factor

sample_pase01b item 01b, factor

sample_pase02 item 02, factor

sample_pase02a item 02a, factor

sample_pase03 item 03, factor

sample_pase03b item 03b, factor

sample_pase04 item 04, factor

sample_pase04b item 04b, factor

sample_pase05 item 05, factor

sample_pase05b item 05b, factor

sample_pase06 item 06, factor

sample_pase06b item 06b, factor

sample_pase07 item 07, factor

sample_pase08 item 08, factor

sample_pase09a item 09a, factor

sample_pase09b item 09b, factor

sample_pase09c item 09c, factor

sample_pase09d item 09d, factor

sample_pase10 item 10, factor

sample_pase10a item 10a, numeric

sample_pase10b item 10b, factor

`pase_calc` *PASE score calculator*

Description

Calculates PASE score from raw questionnaire data.

Usage

```
pase_calc(ds, adjust_work = FALSE, consider.missing = c("Not available"))
```

Arguments

`ds`	data set
`adjust_work`	flag to set whether to include 10b type 1.
`consider.missing`	character vector of values considered missing. Default is TRUE.

Details

Labelling should be as defined by the questionnaire. 02-06 should start with 0:3, 02a-06b should start with 1:4.

Regarding work scoring:

The score calculation manual available for the PASE questionnaire, all types of work should be included. According to the article by Washburn RA. et al (1999) sitting work is not included in the item 10 score. This differentiation is added with the option to set `adjust_work` to exclude item 10b category 1 work (set `TRUE`).

Regarding output:

Output includes sub scores as well as sums, but also to columns assessing data quality and completeness. If any field has not been filled, `score_incompletes` will return `TRUE`. If all measures are missing `score_missings` is `TRUE`. If `adjust_work==TRUE`, 10b has to be filled, or `score_incompletes` will be set `TRUE`.

Value

data.frame

Examples

```
summary(pase_calc(stRoke::pase)[,13])
str(pase_calc(stRoke::pase))
```

`print.win_Prob` *Prints win_prob results*

Description

Prints win_prob results

Usage

```
## S3 method for class 'win_Prob'
print(x, ...)
```

Arguments

`x`	win_prob results.
`...`	ignored for now

Value

Prints win_prob statistics.

`quantile_cut` *MOVED Easy function for splitting numeric variable in quantiles*

Description

Using base/stats functions cut() and quantile().

Usage

```
quantile_cut(
  x,
  groups,
  y = NULL,
  na.rm = TRUE,
  group.names = NULL,
  ordered.f = FALSE,
  inc.outs = FALSE,
  detail.list = FALSE
)
```

Arguments

x	Variable to cut.
groups	Number of groups.
y	alternative vector to draw quantile cuts from. Limits has to be within x. Default is NULL.
na.rm	Remove NA's. Default is TRUE.
group.names	Names of groups to split to. Default is NULL, giving intervals as names.
ordered.f	Set resulting vector as ordered. Default is FALSE.
inc.outs	Flag to include min(x) and max(x) as borders in case of y!=NULL.
detail.list	flag to include details or not

Value

vector or list with vector and details (length 2)

Examples

```
aa <- as.numeric(sample(1:1000,2000,replace = TRUE))
x <- 1:450
y <- 6:750
summary(quantile_cut(aa,groups=4,detail.list=FALSE)) ## Cuts quartiles
```

---

score *Data frame with sample data of cognitive testing score*

Description

Contains non-identifiable organic trial data from a five-dimensional cognitive test.

Usage

```
data(score)
```

Format

A data frame with 20 rows and 26 variables:

id id

event event

a_is domain a index score

b_is domain b index score

c_is domain c index score

d_is domain d index score

e_is domain e index score

i_is total index score

a_lo domain a lower ci

b_lo domain b lower ci

c_lo domain c lower ci

d_lo domain d lower ci

e_lo domain e lower ci

i_lo total lower ci

a_up domain a upper ci

b_up domain b upper ci

c_up domain c upper ci

d_up domain d upper ci

e_up domain e upper ci

i_up total upper ci

a_per domain a percentile

b_per domain b percentile

c_per domain c percentile

d_per domain d percentile

e_per domain e percentile

i_per total percentile

`source_lines` *Source Lines from a File*

Description

Sources specific lines from a file

Usage

```
source_lines(file, lines, ...)
```

Arguments

`file`	A character string giving the path to the file to be sourced.
`lines`	A numeric vector of line numbers to be sourced.
`...`	Additional arguments to be passed to `source`.

Value

The result of `source`.

See Also

This function is borrowed from a gist by christophergandrud.

Examples

```
test_file <- tempfile(fileext = ".R")
writeLines(c("# Line 1", "2+2", "# Line 3"), test_file)
source_lines(test_file, 1:2, echo=TRUE)
```

`str_extract` *Extract string based on regex pattern*

Description

DEPRECATION: moved to `agdamsbo/project.aid`

Usage

```
str_extract(d, pattern)
```

Arguments

`d`	vector of character strings
`pattern`	regex pattern to match

Details

Use base::strsplit to

Value

vector of character strings

Examples

```
ls <- do.call(c,lapply(sample(4:8,20,TRUE),function(i){
paste(sample(letters,i,TRUE),collapse = "")}))
ds <- do.call(c,lapply(1:20,function(i){
paste(sample(ls,1),i,sample(ls,1),"23",sep = "_")}))
str_extract(ds,"([0-9]+)")
```

talos *Data frame with sample of TALOS data*

Description

Contains of non-identifiable subset of data from the TALOS trial.

Usage

```
data(talos)
```

Format

A data frame with 200 rows and 6 variables:

rtreat Randomisation

mrs_1 Modified Rankin scale score at follow-up

mrs_6 Modified Rankin scale score at end of study

hypertension Known hypertension

diabetes Known diabetes

civil Cohabitation status

Source

doi:10.1161/STROKEAHA.117.020067

win_prob *Calculates the probability of winning*

Description

Calculates the probability of winning (winP). In the referenced article Zou et al (2022) proposes a method for calculating probability of winning with a confidence interval an p-value testing.

Usage

```
win_prob(
  data,
  response = NULL,
  group = NULL,
  alpha = 0.05,
  beta = 0.2,
  group.ratio = 1,
  sample.size = FALSE,
  print.tables = FALSE,
  dec = 3
)
```

Arguments

data	A data frame containing the response and group variable.
response	The name of the response variable. Takes first column if empty.
group	The name of the group variable. Takes second column if empty.
alpha	The alpha level for the hypothesis test. Default is 0.05.
beta	The beta level for the sample size calculation. Default is 0.2.
group.ratio	The ratio of group sizes. Default is 1.
sample.size	Flag to include sample size calculation. Default is FALSE.
print.tables	Flag to print cumulative tables. Default is FALSE.
dec	Numeric for decimals to print. Default is 3.

Value

A list containing the win_prob statistics.

Source

doi:10.1161/STROKEAHA.121.037744

Examples

```
win_prob(data=stRoke::talos,response="mrs_6",group="rtreat")
```

---

write_ical *MOVED Write ical object*

Description

This function creates an ical file based on a data frame with mixed events. Export as .ics file using `calendar::ic_write()`.

Usage

```
write_ical(
  df,
  date = "date",
  date.end = NA,
  title = "title",
  time.start = "start",
  time.end = "end",
  place = NA,
  place.def = NA,
  time.def = "10:00:00",
  time.dur = 60,
  descr = NA,
```

```
  link = NA,
  t.zone = "CET"
)
```

Arguments

df	A data frame with the calendar data
date	The name of the event date column in the data frame
date.end	The name of the end date column in the data frame
title	The name of the title column in the data frame
time.start	The name of the start time column in the data frame
time.end	The name of the end time column in the data frame
place	The name of the place column in the data frame
place.def	Default location to use when place is NA
time.def	Default start time to use when time.start is NA
time.dur	Default duration of the event in minutes, if time.end is NA
descr	Name of description/notes column if any.
link	Name of link column, if any.
t.zone	A character string of time zone for events. The string must be a time zone that is recognized by the user's OS.

Value

ical object

See Also

calendar package icalendar standard webpage

Examples

```
df <- data.frame(
  date = c("2020-02-10", "2020-02-11"),
  date.end = c("2020-02-13",NA),
  title = c("Conference", "Lunch"),
  start = c("12:00:00", NA),
  time.end = c("13:00:00", NA),
  note = c("Hi there","Remember to come"),
  link = c("https://icalendar.org","https://agdamsbo.github.io/stRoke/")
)

write_ical(
  df,
  date = "date",
  date.end = "date.end",
  title = "title",
  time.start = "start",
```

```
  time.end = "time.end",
  place.def = "Conference Room",
  descr = "note",
  link = "link"
)
```

Index