

# Package ‘stRoke’

April 13, 2023

**Title** Clinical Stroke Research

**Version** 23.4.1

**Description** This is an R-toolbox of custom functions for convenient data management and analysis in clinical health 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 TA-LOS trial (Kraglund et al (2018) <[doi:10.1161/STROKEAHA.117.020067](https://doi.org/10.1161/STROKEAHA.117.020067)>).

The win\_prob() function is based on work by Zou et al (2022) <[doi:10.1161/STROKEAHA.121.037744](https://doi.org/10.1161/STROKEAHA.121.037744)>.

The age\_calc() function is based on work by Becker (2020) <[doi:10.18637/jss.v093.i02](https://doi.org/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** 7.2.3

**LazyData** true

**Suggests** covr, knitr, rmarkdown, spelling, testthat (>= 3.0.0)

**Language** en-US

**Config/testthat/edition** 3

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

**Depends** R (>= 3.5.0)

**VignetteBuilder** knitr

**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** 2023-04-13 12:20:02 UTC

## R topics documented:

age_calc . . . . .	2
ci_plot . . . . .	3
contrast_text . . . . .	4
cprs . . . . .	5
cpr_check . . . . .	5
cpr_dob . . . . .	6
cpr_female . . . . .	7
ds2dd . . . . .	7
files_filter . . . . .	8
generic_stroke . . . . .	9
index_plot . . . . .	10
label_select . . . . .	10
metadata_names . . . . .	11
quantile_cut . . . . .	12
score . . . . .	13
source_lines . . . . .	14
talos . . . . .	14
win_prob . . . . .	15
write_ical . . . . .	16
<b>Index</b>	<b>18</b>

---

age_calc	<i>Calculate age in years, months, or days</i>
----------	--

---

### 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](https://doi.org/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")))
```

---

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")
```

---

contrast\_text

*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")

**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/596211-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	<i>Determine female sex from CPR</i>
------------	--------------------------------------

---

**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	<i>Data set to data dictionary function</i>
-------	---

---

**Description**

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  
)
```

**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.

**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	<i>Filter files in a folder</i>
--------------	---------------------------------

---

**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	<i>Generic stroke study outcome</i>
----------------	-------------------------------------

---

## 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	<i>Plot multidimensional cognitive test scores</i>
------------	--

---

### 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	<i>Helper function for labels in gtsummary</i>
--------------	--

---

### 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	<i>Vector of REDCap metadata headers</i>
----------------	--

---

**Description**

Vector of REDCap metadata headers

**Usage**

```
data(metadata_names)
```

**Format**

Vector of length 18 with REDCap metadata headers:

**metadata\_names** characterstrings

**See Also**

<https://www.project-redcap.org/>

---

`quantile_cut`*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**

<code>x</code>	Variable to cut.
<code>groups</code>	Number of groups.
<code>y</code>	alternative vector to draw quantile cuts from. Limits has to be within <code>x</code> . Default is <code>NULL</code> .
<code>na.rm</code>	Remove NA's. Default is <code>TRUE</code> .
<code>group.names</code>	Names of groups to split to. Default is <code>NULL</code> , giving intervals as names.
<code>ordered.f</code>	Set resulting vector as ordered. Default is <code>FALSE</code> .
<code>inc.outs</code>	Flag to include <code>min(x)</code> and <code>max(x)</code> as borders in case of <code>y!=NULL</code> .
<code>detail.list</code>	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	<i>Data frame with sample data of cognitive testing score</i>
-------	---

---

### 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	<i>Source Lines from a File</i>
--------------	---------------------------------

---

**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 <a href="#">source</a> .

**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)
```

---

talos	<i>Data frame with sample of TALOS data</i>
-------	---

---

**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](https://doi.org/10.1161/STROKEAHA.117.020067)

---

win_prob	<i>Calculates the probability of winning</i>
----------	--

---

**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](https://doi.org/10.1161/STROKEAHA.121.037744)

**Examples**

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

---

write\_ical

*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**

<code>df</code>	A data frame with the calendar data
<code>date</code>	The name of the event date column in the data frame
<code>date.end</code>	The name of the end date column in the data frame
<code>title</code>	The name of the title column in the data frame
<code>time.start</code>	The name of the start time column in the data frame
<code>time.end</code>	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

- \* **age**
  - age\_calc, 2
- \* **cpr**
  - cpr\_check, 5
  - cpr\_dob, 6
  - cpr\_female, 7
- \* **datasets**
  - cprs, 5
  - metadata\_names, 11
  - score, 13
  - talos, 14
- \* **date**
  - age\_calc, 2
- \* **quantile**
  - quantile\_cut, 12
- \* **time**
  - age\_calc, 2

age\_calc, 2

ci\_plot, 3

contrast\_text, 4

cpr\_check, 5

cpr\_dob, 6

cpr\_female, 7

cprs, 5

ds2dd, 7

files\_filter, 8

generic\_stroke, 9

index\_plot, 10

label\_select, 10

metadata\_names, 11

quantile\_cut, 12

score, 13

source, 14

source\_lines, 14

talos, 14

win\_prob, 15

write\_ical, 16