

Package ‘clintools’

July 22, 2025

Type Package

Title Tools for Clinical Research

Version 0.9.10.1

Description Every research team have their own script for data management, statistics and most importantly hemodynamic indices. The purpose is to standardize scripts utilized in clinical research. The hemodynamic indices can be used in a long-format dataframe, and add both periods of interest (trigger-periods), and delete artifacts with deleter-files. Transfer function analysis (Claassen et al. (2016) <doi:10.1177/0271678X15626425>) and Mx (Czosnyka et al. (1996) <doi:10.1161/01.str.27.10.1829>) can be calculated using this package.

License MIT + file LICENSE

URL <https://github.com/lilleoel/clintools>

BugReports <https://github.com/lilleoel/clintools/issues>

Depends R (>= 3.5.0)

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.2.2

Suggests knitr, rmarkdown

Imports signal (>= 0.7-6), xml2 (>= 1.3.2), lme4 (>= 1.1-27.1), ggplot2 (>= 3.3.0), pROC (>= 1.18.0), irr (>= 0.84.1), nlme (>= 3.1-160), parameters (>= 0.19.0), stringi (>= 1.7.8), scales (>= 1.2.1), dplyr (>= 1.1.2), survival (>= 3.4-0), pander (>= 0.6.5)

NeedsCompilation no

Author Markus Harboe Olsen [cre, aut],
Christian Riberholt [ctb],
Ronan Berg [ctb],
Kirsten Moeller [ctb],
Janus Christian Jakobsen [ctb],
Aksel Karl Georg Jensen [ctb]

Maintainer Markus Harboe Olsen <oel@oelfam.com>
Repository CRAN
Date/Publication 2023-12-11 18:40:02 UTC

Contents

| | |
|-----------------------------|-----------|
| calcrel | 2 |
| cdm.fig | 3 |
| cdm.miss | 4 |
| clinmon | 5 |
| comparerel | 9 |
| danishcpr | 10 |
| df.data1000 | 10 |
| df.deleter | 11 |
| dilations | 11 |
| iscus | 13 |
| ortable | 14 |
| PLR3000 | 14 |
| questionnaire | 15 |
| quietly | 16 |
| recol | 17 |
| rowMedians | 17 |
| rrGcomp | 18 |
| sRCT | 19 |
| tbl | 20 |
| testdata10 | 21 |
| TFA | 22 |
| tfa_sample_data | 26 |
| tfa_sample_data_1 | 27 |
| tfa_sample_data_2 | 28 |
| Index | 29 |

| | |
|---------|---|
| calcrel | <i>Calculation of reliability (calcrel)</i> |
|---------|---|

Description

calcrel() is a function used to calculate different reliability measures, including Coefficient of variance, smallest real difference, intraclass correlation coefficient, and Bland-Altman plot derived bias with 95% limits of agreement.

Usage

calcrel(d1,d2)

Arguments

| | |
|----|--|
| d1 | list of numbers from measurement one |
| d2 | list of numbers from measurement two (same order as for measurement one) |

Value

Returns a nested list of reliability measures.

Examples

```
d1 <- rnorm(15,10,1)
d2 <- rnorm(15,10,1)
calcrel(d1,d2)
```

cdm.fig

*Central data monitoring to assess deviations (cdm.fig)***Description**

cdm.fig() is a function to assess any deviations

Usage

```
cdm.fig(df, col, site, meta_title, seedno, output, nmin)
```

Arguments

| | |
|------------|--|
| df | dataframe to be assessed for missing data |
| col | column to be assessed |
| site | column with sites |
| meta_title | Y-axis lab, if empty then it is the column name |
| seedno | the numeric site, if empty it is just Sys.Date() |
| output | if 'fig' then the figure is the output, any other will output the blinded site table |
| nmin | minimum number of variables in site to be presented |

Value

Returns a full markdown output.

Examples

```
## Not run:
cdm.fig(df,col="Mode of birth",
        site="maternal_trial_site")
cdm.fig(df,col="Gestational Age at birth",
        site="maternal_trial_site")
library(knitr)
kable(cdm.fig(df,col="Mode of birth",
              site="maternal_trial_site",output = ""),row.names=F)

## End(Not run)
```

| | |
|----------|------------------------------------|
| cdm.miss | <i>Quiet any output (cdm.miss)</i> |
|----------|------------------------------------|

Description

cdm.miss() is a small function which suppresses any output

Usage

```
cdm.miss(df, id, cols, date, lostFU, filter)
```

Arguments

| | |
|--------|---|
| df | dataframe to be assessed for missing data |
| id | column-name for unique id's |
| cols | columns to be assessed for missing data |
| date | column with the date of follow-up, i.e. when data is missing |
| lostFU | column for patients lost to follow up, TRUE/FALSE in the column |
| filter | how many should be shown in figures - 'all' for all, 'waiting' for those with missing or waiting for data, and 'missing' for only those with missing data |

Value

Returns a full markdown output.

Examples

```
## Not run:
cdm.miss(data,id=idcols[[1]],cols=missing.cols,lostFU="lostFU",
         date = "follow_up_date", filter="missing")

## End(Not run)
```

| | |
|---------|--|
| clinmon | <i>Hemodynamic Indices Calculated From Clinical Monitoring (clinmon)</i> |
|---------|--|

Description

clinmon() uses a *continuous* recording and returns a dataframe with hemodynamic indices for every period, epoch or block depending on the input. Calculates COest, CPPopt, CVRi, Dx, Mx, PI, PRx, PWA, RI, and Sx (see *Hemodynamic indices*).

Usage

```
clinmon(df, variables,
        trigger = NULL, deleter = NULL,
        blocksize = 3, epochsize = 20,
        overlapping = FALSE, freq = 1000,
        blockmin = 0.5, epochmin = 0.5,
        output = "period", fast = FALSE)
```

Arguments

| | |
|-------------|---|
| df | Raw <i>continuous</i> recording with all numeric data and first column has to be time in seconds. (dataframe) |
| variables | Defining the type and order of the recorded variables as a list. Middle cerebral artery blood velocity ('mcav'), Arterial blood pressure ('abp'), cerebral perfusion pressure ('cpp'), intracranial pressure ('icp'), and heart rate ('hr') is currently supported. <i>It is necessary that time is the first row.</i> (list) |
| trigger | Trigger with two columns: first is start, and second is end of periods to be analyzed. Every row corresponds to a period. Default is NULL, which results in analysis of the full dataframe. (dataframe) |
| deleter | Deleter with two columns: first is start and second is end of period with artefacts, which need to be deleted. Every row is a period with artefacts. Default is NULL. (dataframe) |
| blocksize | Length of a block, in seconds. Default is 3. (numeric) |
| epochsize | Size of epochs in number of blocks. Default is 20. (numeric) |
| overlapping | The number of block which should overlap when calculating correlation based indices, and remain blank if overlapping calculations should not be utilized. Default is FALSE. (numeric) |
| freq | Frequency of recorded data, in Hz. Default is 1000. (numeric) |
| blockmin | Minimum measurements required to create a block in ratio. Default is 0.5 corresponding to 50%. If the block holds less than the defined ratio the block will be omitted. (numeric) |
| epochmin | Minimum number of blocks required to create an epoch in ratio. Default is 0.5 corresponding to 50%. If the epoch holds less than the defined ration the epoch will be omitted. (numeric) |

| | |
|--------|--|
| output | Select what each row should represent in the output. Correlation based indices are not presented when selecting blocks for every row. Currently 'block', 'epoch', 'period' or 'cppopt' is supported. Default is 'period'. (string) |
| fast | Select if you want the data to aggregated before analysis resulting in a faster, but perhaps more imprecise run, in Hz. Default is FALSE. (numeric) |

Details

Using a *continuous* raw recording, `clinmon()` calculates hemodynamic indices for every period, epoch or block depending on the chosen output.

`View(data)`

| time | abp | mcav |
|--------|-----|------|
| 7.00 | 78 | 45 |
| 7.01 | 78 | 46 |
| ... | ... | ... |
| 301.82 | 82 | 70 |
| 301.83 | 81 | 69 |

To calculate the indices insert the data and select the relevant variables.

```
clinmon(df=data, variables=c("abp", "mcav"))
```

See **Value** for output description.

Value

Returns a dataframe with the results, with either every blocks, epochs or periods as rows, depending on the chosen output.

The columns of the output are:

- period - The period number corresponding to the row-number in the trigger file.
- epoch - The epoch number, or if period is chosen as output it reflects the number of epochs in the period.
- block - The block number, or if period or epoch is chosen as output it reflects the number of blocks in the period or epoch.
- time_min - The minimum time value or the period, epoch or block.
- time_max - The maximum time value or the period, epoch or block.
- missing_percent - The percentage of missing data in the period, epoch or block.
- XX_mean - The mean value of each variable for the period, epoch or block.
- XX_min - The minimum value of each variable for the period, epoch or block.
- XX_max - The maximum value of each variable for the period, epoch or block.
- YY - The indices in each column.

Hemodynamic indices

COest | **Estimated cardiac output:**

Required variables: abp, hr; *Required output:* -.

Estimated cardiac output (COest) is calculated by utilizing the method described by Koenig et al. [1]:

$$COest = PP / (SBP + DBP) * HR$$

PP: Pulse pressure; SBP: systolic blood pressure; DBP: diastolic blood pressure; HR: heart rate.

CPPopt | **Optimal cerebral perfusion pressure:**

Required variables: abp, icp; *Required output:* period.

Optimal cerebral perfusion pressure (CPPopt) is calculated utilizing the method described by Steiner et al. [2]. The CPPopt return NA if CPPopt is the maximum or minimum CPP investigated. CPPopt is recommended to only be calculated after 'several hours' of recording:

$$CPPopt = The5mmHgCPPIntervalWithLowestMeanPRx$$

CPP: cerebral perfusion pressure; PRx: Pressure reactivity index.

CVRi | **Cardiovascular resistance index:**

Required variables: abp, mcav; *Required output:* -.

Cardiovascular resistance index (CVRi) is calculated utilizing the method described by Fan et al. [3]:

$$CVRi = meanABP / meanMCAv$$

ABP: arterial blood pressure; MCAv: middle cerebral artery blood velocity.

Dx | **Diastolic flow index:**

Required variables: cpp/abp, mcav; *Required output:* epoch, period.

Diastolic flow index (Dx) is calculated utilizing the method described by Reinhard et al. [4]:

$$Dxc = cor(meanCPP / minMCAv)$$

$$Dxa = cor(meanABP / minMCAv)$$

cor: correlation coefficient; CPP: cerebral perfusion pressure; ABP: arterial blood pressure; MCAv: middle cerebral artery blood velocity.

Mx | **Mean flow index:**

Required variables: cpp/abp, mcav; *Required output:* epoch, period.

Mean flow index (Mx) is calculated utilizing the method described by Czosnyka et al. [5]:

$$Mxc = cor(meanCPP / meanMCAv)$$

$$Mxa = cor(meanABP / meanMCAv)$$

cor: correlation coefficient; CPP: cerebral perfusion pressure; ABP: arterial blood pressure; MCAv: middle cerebral artery blood velocity.

PI | Gosling index of pulsatility:

Required variables: mcav; *Required output:* -.

Gosling index of pulsatility (PI) is calculated utilizing the method described by Michel et al. [6]:

$$PI = (systolicMCAv - diastolicMCAv) / meanMCAv$$

MCAv: middle cerebral artery blood velocity.

PRx | Pressure reactivity index:

Required variables: abp, icp; *Required output:* epoch, period.

Pressure reactivity index (PRx) is calculated utilizing the method described by Czosnyka et al. [7]:

$$PRx = cor(meanABP / meanICP)$$

cor: correlation coefficient; CPP: cerebral perfusion pressure; ICP: intracranial pressure.

PWA | Pulse wave amplitude:

Required variables: cpp/icp/abp/mcav; *Required output:* -.

Pulse wave amplitude (PWA) is calculated utilizing the method described by Norager et al. [8]:

$$PWA = systolic - diastolic$$

RI | Pourcelots resistive (resistance) index:

Required variables: mcav; *Required output:* -.

Pourcelots resistive (resistance) index (RI) is calculated utilizing the method described by Forster et al. [9]:

$$RI = (systolicMCAv - diastolicMCAv) / systolicMCAv$$

MCAv: middle cerebral artery blood velocity.

Sx | Systolic flow index:

Required variables: cpp/abp, mcav; *Required output:* epoch, period.

Systolic flow index (Sx) is calculated utilizing the method described by Czosnyka et al. [5]:

$$Sxc = cor(meanCPP / systolicMCAv)$$

$$Sxa = cor(meanABP / systolicMCAv)$$

cor: correlation coefficient; CPP: cerebral perfusion pressure; ABP: arterial blood pressure; MCAv: middle cerebral artery blood velocity.

References

1. Koenig et al. (2015) Biomed Sci Instrum. 2015;51:85-90. ([PubMed](#))
2. Steiner et al. (2002) Crit Care Med. 2002 Apr;30(4):733-8. ([PubMed](#))
3. Fan et al. (2018) Front Physiol. 2018 Jul 16;9:869. ([PubMed](#))
4. Reinhard et al. (2003) Stroke. 2003 Sep;34(9):2138-44. ([PubMed](#))
5. Czosnyka et al. (1996) Stroke. 1996 Oct;27(10):1829-34. ([PubMed](#))
6. Michel et al. (1998) Ultrasound Med Biol. 1998 May;24(4):597-9. ([PubMed](#))
7. Czosnyka et al. (1997) Neurosurgery. 1997 Jul;41(1):11-7; discussion 17-9. ([PubMed](#))
8. Norager et al. (2020) Acta Neurochir (Wien). 2020 Dec;162(12):2983-2989. ([PubMed](#))
9. Forster et al. (2017) J Paediatr Child Health. 2018 Jan;54(1):61-68. ([PubMed](#))

Examples

```
data(testdata)
clinmon(df.data10, variables=c('abp','mcav','hr'), freq=10)
```

comparerel

*Compare measures of reliability (comparerel)***Description**

comparerel() is a function which compares measures from calcrel using bootstrapping.

Usage

```
comparerel(d1,d2,d3,d4,n_boot,seedno)
```

Arguments

| | |
|--------|---|
| d1 | list of numbers from measurement one of sample 1 |
| d2 | list of numbers from measurement two of sample 1 (same order as for d1) |
| d3 | list of numbers from measurement one of sample 2 |
| d4 | list of numbers from measurement two of sample 2 (same order as for d2) |
| n_boot | numbers of iterations (default is 1000) |
| seedno | the seed number used for bootstrapping (default is) |

Value

Returns a nested list of difference between reliability measures using bootstrapping.

Examples

```
## Not run:
d1 <- rnorm(15,10,1)
d2 <- rnorm(15,10,1)
d3 <- rnorm(15,10,1)
d4 <- rnorm(15,10,1)
comparerel(d1,d2,d3,d4)

## End(Not run)
```

| | |
|-----------|--|
| danishcpr | <i>Danish CPR number to birthday and sex (danishcpr)</i> |
|-----------|--|

Description

danishcpr() converts a list of CPR-numbers to a corresponding list birthday and sex.

Usage

```
danishcpr(code)
```

Arguments

| | |
|------|----------------------------|
| code | list of CPR numbers (list) |
|------|----------------------------|

Value

Returns a list with \$birthday which is a list of dates, and \$sex which is a list of Male/Female.

Examples

```
## Not run:
cpr <- danishcpr(code)
birthdays <- cpr$birthday
sex <- cpr$sex

## End(Not run)
```

| | |
|-------------|----------------------------|
| df.data1000 | <i>Test-data - 1000 Hz</i> |
|-------------|----------------------------|

Description

Recording with four columns: time (t), non-invasive arterial blood pressure (abp), middle cerebral artery velocity measured using transcranial Doppler (mcav), and heart rate (hr).

Usage

```
data(testdata)
```

Format

An object of class "dataframe"; an example of the usage in [clinmon](#)-function.

References

Olsen MH et al. (Unpublished data, 2020) ([GitHub](#))

Examples

```
data(testdata)
variables <- c("abp", "mcav", "hr")
clinmon(df.data1000, variables, fast=50)
```

| | |
|------------|---------------------|
| df.deleter | <i>Test-deleter</i> |
|------------|---------------------|

Description

Deleter dataframe with two columns: start (start) and end (end) of the deleter-period.

Usage

```
data(testdata)
```

Format

An object of class "dataframe"; an example of the usage in [clinmon](#)-function.

References

Olsen MH et al. (Unpublished data, 2020) ([GitHub](#))

Examples

```
data(testdata)
variables <- c("abp", "mcav", "hr")
clinmon(df.data1000, variables, deleter=df.deleter, fast=50)
```

| | |
|-----------|--|
| dilations | <i>Assess dilations from PLR3000-output.</i> |
|-----------|--|

Description

dilations() is a function which converts longformat recording of pupillary size and uses markers to generate a dataframe, plot, and markdown output to inspect the results.

Usage

```
dilations(pupils, markers,
  remove_markers = NULL, add_markers = NULL,
  not_assess = NULL, artefacts_static = c(0.55, 9.95),
  artefacts_dynamic = c(`1` = 1.5, `0.66` = 1, `0.33` = 0.5),
  time_assess = c(`1` = 10, `3` = 5),
  sig.level = 0.05, min_change = NULL,
  resting_delay = c(`3` = 0))
```

Arguments

| | |
|-------------------|--|
| pupils | recording of pupillary function. long format dataframe with at least three columns: record_id, time, and size. |
| markers | time of markers. long format dataframe with at least two columns: record_id and time. |
| remove_markers | markers which should be removed. long format dataframe with at least two columns: record_id and time. The time column need one decimal. |
| add_markers | markers which should be added. long format dataframe with at least two columns: record_id and time. The time column need one decimal. |
| not_assess | a list of record ids which should no be assessed. |
| artefacts_static | a list of the limits of the artefacts. The first number in the list is the minimum size to be assessed and the second is the maximum size to be assessed |
| artefacts_dynamic | a named list where max change in millimeter within the duration (name in list) is removed. Will use the first 1 second of the recording to create a baseline, thus susceptible to artefacts in the start of the recording. |
| time_assess | This named list define the number of seconds which should be used in the assessment of dilations. The name is the number of periods-of-interest and the value is the seconds. |
| sig.level | This is the significance level to be used when comparing the size of the period-of-interest. The significance level corresponds to the Wilcoxon test used. |
| min_change | This is the minimum size of mm which needs to change before a dilation can be identified. Default is no minimum requirement. |
| resting_delay | This can be used if the subsequent resting period to compare should be delayed, i.e. if we should wait 5 second for those investigations with three periods of interest create a named list with periods of interest as names and seconds to delay as input. |

Value

Returns a nested list one dataframe of the results (`$dilations`), plot (`$plot$id[record id]`), and a markdown output (`$plot$markdown$id[record_id]`). The dilations dataframe include the following columns: Record ID (`record_id`); Patient ID (`pt_id`); Date (`date`); pupil side (`side`); start of the period (`min`); end of the period (`max`); length of period (`rec_length`); number of measurements (`n`); median size (`median`); P value when comparing with the previous period (`p_before`); P value when

comparing with the following period (p_after); and if dilation is identified (dilation, 1 is successful dilation and 0 is no dilation).

Examples

```
## Not run:
recordings <- PLR3000("C:/PLR3000/R_20200105_205901.xls")
dilations <- dilations(recordings$pupils,recordings$markers)

# The dataframe of the results
dilations$dilations

# The plot of one of the recordings
dilations$plot$id833

# The markdown output of one of the recordings
dilations$markdown$id833

## End(Not run)
```

iscus

ISCUSFlex-values to dataframe (iscus)

Description

iscus() is a function which converts XML files extracted from the Microdialysis-apparatur of ISCUSFlex apparatus to a dataframe.

Usage

```
iscus(filename)
```

Arguments

filename path to the XML-file with the measurements

Value

Returns a dataframe with the measurements.

Examples

```
## Not run:
iscus("C:/ISCUSfiles/7888e844-1c7a-40af-a3f2-3bb27a8dd9e5.xml")

## End(Not run)
```

| | |
|---------|--|
| ortable | <i>Logistic regression table with Odds ratio (ortable)</i> |
|---------|--|

Description

ortable() is a small function which utilises the output from the glm-function to print a dataframe with odds ratio, confidence limits, and p-values.

Usage

```
ortable(x, d, d_p, intercept, simple)
```

Arguments

| | |
|-----------|--|
| x | Utilises the output from a glm-function. (glm-output) |
| d | Refers to the number of digits for odds ratio and confidence intervals. Default is 2. (numeric) |
| d_p | Refers to the number of digits for odds ratio and confidence intervals. Default is 3. (numeric) |
| intercept | The intercept is presented in the table if TRUE. Default is FALSE. (boolean) |
| simple | Odds ratio and confidence intervals are merged into one column if TRUE. Default is TRUE. (boolean) |

Value

Returns a dataframe with with odds ratio, confidence limits, and p-values.

Examples

```
df <- data.frame(outcome=sample(0:1, 100,replace=TRUE),
  var=sample(0:100,100,replace=TRUE))
ortable(glm(outcome ~ ., data=df))
```

| | |
|---------|--|
| PLR3000 | <i>NeuroOpticsTM PLR-3000 pupillometer file to dataframe (PLR3000)</i> |
|---------|--|

Description

PLR3000() is a function which converts the XLS file imported from the eurOpticsTM PLR-3000 pupillometer to a nested list with two dataframes.

Usage

```
PLR3000(filename = NULL, df = NULL)
```

Arguments

filename path to the XLS-file with the measurements
 df the dataframe can also be used for the function if data is already imported.

Value

Returns a list with two dataframe, one with the measurements (pupils) and one with the markers (markers).

Examples

```
## Not run:
PLR3000("C:/PLR3000/R_20200105_205901.xls")

## End(Not run)
```

| | |
|---------------|--|
| questionnaire | <i>Calculate scores from questionnaire (questionnaire)</i> |
|---------------|--|

Description

questionnaire() is a function which calculates scores from a questionnaire.

Usage

```
questionnaire(df,id,questions,scale,setting,prefix)
```

Arguments

df dataframe. (df)
 id Column name of participant id (string)
 questions Column names of ordered list of questions (list)
 scale name of the questionnaire (string)
 setting questionnaire specific settings (string)
 prefix prefix of column names of questionnaire scores (string)

Value

Returns summarised information in dataframe.

Examples

```
## Not run:

df <- df_b[,grepl("ssid|KIDSCREEN52_D_",colnames(df_b))][,c(1:53)]
k52 <- questionnaire(df,id = "ssid",
questions=colnames(df)[grepl("KIDSCREEN",colnames(df))],
scale="Kidscreen-52",
setting="proxy")#'

## End(Not run)
```

quietly

Quiet any output (quietly)

Description

quietly() is a small function which suppresses any output

Usage

```
quietly(x)
```

Arguments

x input to be suppressed

Value

Returns x, but without any output

Examples

```
## Not run:
tmp <- quietly(print("hello"))

## End(Not run)
```

| | |
|-------|-------------------------------|
| recol | <i>Reorder column (recol)</i> |
|-------|-------------------------------|

Description

recol() is a small function which can rename columns and change factors to relevant input

Usage

```
recol(df, old, new, factors=NULL, remove=TRUE, na=NA)
```

Arguments

| | |
|---------|---|
| df | dataframe. (df) |
| old | old column name. (string) |
| new | new column name. (string) |
| factors | Named list of factors. If no names then just in alphabetical order (named list) |
| remove | remove old column from dataframe (boolean) |
| na | the na.strings (list) |

Value

Returns the dataframe with new columns.

Examples

```
## Not run:
# NUMERIC VARIABLE
df <- recol(df, "X5d_55_Alcohol_127", "Alcohol")
# FACTOR VARIABLE
df <- recol(df, "X5d_55_Alcohol_127", "Alcohol", factors=c("No", "Yes"))

## End(Not run)
```

| | |
|------------|---|
| rowMedians | <i>find median of rows similar to rowMeans (rowMedians)</i> |
|------------|---|

Description

rowMedians() converts a dataframe to a list of row medians.

Usage

```
rowMedians(x, na.rm=FALSE)
```

Arguments

`x` a dataframe where median of row should be calculated. (data.frame)
`na.rm` Should missing values be omitted fro the calculations? (boolean)

Value

Returns a list of the median of each row in the inputted dataframe.

Examples

```
## Not run:
  rowMedians(df[,c("test1", "test2", "test3")])

## End(Not run)
```

rrGcomp

*Relative risk derived by G-computation (rrGcomp)***Description**

`rrGcomp()` is a small function which generates population-level (marginal) relative risks derived by G-computation. For models with random effects mixed-effects generalized linear model with a logit link with adjustment for stratification variables will be used, while those without random effects a logistic regression will be used. The code is based on the method used in the paper by Dankiewicz et al. (2021) N Engl J Med. Jun 17;384(24):2283-2294. ([PubMed](#))

Usage

```
rrGcomp(df, outcome_col, group_col,
  fixed_strata = NULL, random_strata = NULL,
  nbrIter = 5000, conf_level = 0.95)
```

Arguments

`df` the individual participant dataframe
`outcome_col` column name for the outcome column
`group_col` column name for the group column
`fixed_strata` list of column names for the fixed effect stratification columns
`random_strata` list of column names for the random effect stratification columns
`nbrIter` number of iterations to be used in the G-computation. The original paper used 5000, which is also the default.
`conf_level` the confidence level to be reported.

Value

Returns a list with relative risk (rr), simulated rr (simRR), lower- and upper confidence level (simLCL/simUCL), and the p-value (p_val)

Examples

```
df <- sRCT(n_sites=3,n_pop=50)
rrGcomp(df,outcome_col="outcome",group_col="Var1",random_strata="site",nbrIter=10)
```

| | |
|------|---|
| sRCT | <i>simulated Randomised Clinical Trial (sRCT)</i> |
|------|---|

Description

sRCT() is a function which simulates a randomised clinical trial with a binary outcome and returns a dataframe. This version is validated to be used for analysis of interaction in a factorial design.

Usage

```
sRCT(all_sizes = NULL, n_pop = 1000,
n_sites = 10, design = c(2,2,2),
rrr = c(0.05,0.05,0), outcome_risk = 0.492,
interaction = c(`1-2` = 0.05, `1-2-3` = -0.05),
site_re = 0.05)
```

Arguments

| | |
|--------------|---|
| all_sizes | Size of blocks in allocation table. If left empty the three lowest possible block sizes will be randomly assigned. |
| n_pop | Number of participants included in the trial. |
| n_sites | Number of sites |
| design | Number of sites as a list where each element corresponds to an intervention and the number in the element is the number of groups. So for a 2x2 factorial design c(2,2) should be used. |
| rrr | relative risk reduction for each intervention so for the abovementioned 2x2 factorial design with RRR of 0.05 and 0.10 we would use c(0.05,0.10). |
| outcome_risk | The baseline risk (probability in absolute percentage) of the dichotomous primary outcome. |
| interaction | Interaction between interventions with a named list. If interaction exists between intervention 1 and 2 we would use 1-2 = 0.05. |
| site_re | The size of the random effect of site, default is 0.05. |

Details

The sRCT function is continuously being developed to answer specific questions in simulation studies. sRCT will be updated and tested for each specific question. For each update the function will be validated for the current purpose and all previous purposes. sRCT is not validated for all simulation studies

Value

Returns a dataframe with an individual participant data frame.

Examples

```
sRCT()
```

| | |
|-----|-----------------------------|
| tbl | <i>Create a table (tbl)</i> |
|-----|-----------------------------|

Description

tbl() is a function which create a dataframe, which can be copied directly into word or presented in as a summary table.

Usage

```
tbl(df, strata, vars, render.numeric,
    render.factor, tests, paired,
    digs_n, digs_f, digs_p, digs_s,
    only_stats, strata.fixed, strata.random,
    time.to, present.missing, markdown, caption)
```

Arguments

| | |
|----------------|--|
| df | dataframe. (df) |
| strata | Column name of stratification (string) |
| vars | Column names of variables of interest (list) |
| render.numeric | list of presentation of numeric variables (list) |
| render.factor | presentation of factors, with simple removing one factor when only two exists |
| tests | list of tests carried out, currently the following works: t.test, wilcox.test, fisher.test, auc, lm, and glm. (list) |
| paired | if tests should be paired (boolean) |
| digs_n | digits for numeric (numeric) |
| digs_f | digits for factors (numeric) |
| digs_p | digits for p-values (numeric) |

| | |
|------------------------------|---|
| <code>digs_s</code> | digits for statistics (numeric) |
| <code>only_stats</code> | if only stats should be presented (boolean) |
| <code>strata.fixed</code> | list of columns which should be used as fixed stratification (list) |
| <code>strata.random</code> | list of columns which should be used as random stratification (list) |
| <code>time.to</code> | Column name of the time column for cox regression (list) |
| <code>present.missing</code> | default is dynamic where non-missing variables are not presented. |
| <code>markdown</code> | default is true and output is pander, while false output is a dataframe (boolean) |
| <code>caption</code> | Table caption only in use when markdown is true (string) |

Value

Returns summarised information in dataframe.

Examples

```
## Not run:
hmm <- tbl(df, strata="group",
  vars = c("Gestational Age at birth", "Maternal preeclampsia"),
  tests=c("wilcox.test", "glm"), only_stats=F, strata.random = "site",
  markdown=F)
pander::pander(hmm, keep.line.breaks = TRUE, split.tables=Inf, row.names = F)

## End(Not run)
```

testdata10

Test-data - 10 Hz

Description

Recording with four columns: time (t), non-invasive arterial blood pressure (abp), middle cerebral artery velocity measured using transcranial Doppler (mcav), and heart rate (hr).

Usage

```
data(testdata)
```

Format

An object of class "dataframe"; an example of the usage in [clinmon](#)-function.

References

Olsen MH et al. (Unpublished data, 2020) ([GitHub](#))

Examples

```
data(testdata)
variables <- c("abp", "mcav", "hr")
clinmon(df.data10, variables, freq=10)
```

TFA

Transfer function analysis of dynamic cerebral autoregulation (TFA)

Description

TFA() calculates dynamic cerebral autoregulation through a transfer function analysis from a *continuous* recording. This function follows the recommendations from Claassen et al. [1] and mimicks the matlab script created by David Simpsons in 2015 ([Matlab TFA function](#)). TFA() also includes the possibility to analyse raw recordings with application of cyclic (beat-to-beat) average with the possibility of utilizing interpolation. (see **details**).

Usage

```
TFA(df, variables,
    trigger = NULL, deleter = NULL,
    freq = 1000, fast = 50, raw_data = FALSE,
    interpolation = 3, output = "table",
    vlf = c(0.02, 0.07), lf = c(0.07, 0.2),
    hf = c(0.2, 0.5), detrend = FALSE,
    spectral_smoothing = 3,
    coherence2_thresholds = cbind(c(3:15),
    c(0.51, 0.40, 0.34, 0.29, 0.25, 0.22, 0.20, 0.18,
    0.17, 0.15, 0.14, 0.13, 0.12)),
    apply_coherence2_threshold = TRUE,
    remove_negative_phase = TRUE,
    remove_negative_phase_f_cutoff = 0.1,
    normalize_ABP = FALSE,
    normalize_CBFV = FALSE,
    window_type = 'hanning',
    window_length = 102.4,
    overlap = 59.99,
    overlap_adjust = TRUE,
    na_as_mean = TRUE)
```

Arguments

| | |
|-----------|---|
| df | Raw <i>continuous</i> recording with numeric data and first column has to be time in seconds. (dataframe) |
| variables | Definition of the type and order of recorded variables as a list. Middle cerebral artery blood velocity ('mcav') and arterial blood pressure ('abp') is currently supported. (list) |

| | |
|---|--|
| trigger | Trigger with two columns: first is start, and second is end of period to be analyzed. Every row is a period for analysis. Default is NULL, which results in analysis of the full dataframe. (dataframe) |
| deleter | Deleter with two columns: first is start and second is end of period with artefacts, which need to be deleted. Every row is a period with artefacts. Default is NULL. (dataframe) |
| freq | Frequency of recorded data, in Hz. Default is 1000. (numeric) |
| fast | Select if you want the data to aggregated resulting in a faster, but perhaps more imprecise run, in Hz. Default is 50 (numeric) |
| raw_data | Select TRUE if the data is raw and cyclic mean should be calculated. NB: this function have not been validated, why validated methods for calculating cyclic mean are preferred. Only 1 period can be analysed using raw_data. Default is FALSE (boolean) |
| interpolation | Select the number of beats which should be interpolated. Default is up to 3 beats and 0 results in no interpolation. (numeric) |
| output | Select what the output should be. 'table' results in a dataframe with values for the three frequencies defined by Claassen et al. [1]; 'long' results in a dataframe with the results in a long format; 'plot' results in a dataframe which can help plot gain, phase and coherence; 'plot-peak' results in a dataframe, which can be used to validate the cyclic average, and 'raw' results in a nested list with results primarily for debugging. Default is 'table'. (string) |
| vlf, lf, hf, detrend, spectral_smoothing, coherence2_thresholds | See TFA-parameters |
| apply_coherence2_threshold, remove_negative_phase | See TFA-parameters |
| remove_negative_phase_f_cutoff, normalize_ABP | See TFA-parameters |
| normalize_CBFV, window_type, window_length, overlap | See TFA-parameters |
| overlap_adjust, na_as_mean | See TFA-parameters |

Details

Using a *continuous* raw recording, TFA() calculates dynamic cerebral autoregulation trough a transfer function analysis. This function utilizes the recommendations from Claassen et al [1] and mimicks the matlab script created by David Simpsons in 2015.

View(data)

| time | abp | mcav |
|--------|-----|------|
| 7.00 | 78 | 45 |
| 7.01 | 78 | 46 |
| ... | ... | ... |
| 301.82 | 82 | 70 |

301.83 81 69

To calculate the variables insert the data and select the relevant variables.

```
TFA(df=data, variables=c("abp","mcav"))
```

See **Value** for output description.

Value

TFA() returns a dataframe depending on the output selected. 'table' results in a dataframe with values for the three frequencies defined by Claassen et al. [1]; 'long' results in a dataframe with the results in a long format; 'plot' results in a dataframe which can help plot gain, phase and coherence; 'plot-peak' results in a dataframe, which can be used to validate the cyclic average, and 'raw' results in a nested list with results primarily for debugging.

Some generic variables are listed below:

- abp_power - The blood pressure power measured in mmHg².
- cbfv_power - The cerebral blood flow velocity power measured in cm²*s⁻²
- coherence - Coherence.
- gain_not_normal - Not normalized gain measured in cm*s⁻¹*mmHg⁻¹.
- gain_normal - Normalized gain measured in %*mmHg⁻¹.
- phase - Phase measured in radians.

output = 'table':

Wide format output table with period, VLF, LF, and HF as columns, and the TFA-variables as rows.

| period | variable | vlf | lf | hf |
|--------|-------------|------|------|------|
| 1 | abp_power | 6.25 | 1.56 | 0.21 |
| 1 | cbfv_power | 3.22 | 2.25 | 0.30 |
| ... | ... | ... | ... | ... |
| 3 | gain_normal | 1.04 | 1.48 | 1.85 |
| 3 | phase | 53.0 | 25.4 | 9.38 |

output = 'long':

Long format output table which can be manipulated depending on the intended use, with period, interval, variables and values as columns.

| period | interval | variable | values |
|--------|----------|------------|--------|
| 1 | hf | abp_power | 6.25 |
| 1 | hf | cbfv_power | 3.22 |
| ... | ... | ... | ... |
| 2 | vlf | gain_norm | 1.85 |

2 vlf phase 9.38

output = 'plot':

Plot format output table which can be used to draw figures with gain, phase and coherence depending on frequency.

| period | freq | gain | phase | coherence |
|--------|------|------|-------|-----------|
| 1 | 0.00 | 0.16 | 0.00 | 0.04 |
| 1 | 0.01 | 0.29 | 4.22 | 0.29 |
| ... | ... | ... | ... | ... |
| 2 | 1.55 | 1.15 | -43.2 | 0.64 |
| 2 | 1.56 | 1.16 | -41.1 | 0.42 |

TFA-parameters

A series of parameters that control TFA analysis (window-length, frequency bands ...). If this is not provided, default values, corresponding to those recommended in the white paper, will be used. These default values are given below for each parameter.

- **vlf** Limits of *very low frequency* band (in Hz). This corresponds to the mathematical inclusion of [X:Y[. Default is c(0.02-0.07).
- **lf** Limits of *low frequency* band (in Hz). This corresponds to the mathematical inclusion of [X:Y[. Default is c(0.07-0.2).
- **hf** Limits of *high frequency* band (in Hz). This corresponds to the mathematical inclusion of [X:Y[. Default is c(0.2-0.5).
- **detrend** Linear detrending of data prior to TFA-analysis (detrending is carried out as one continuous trend over the whole length of the recording, not segment-by-segment). Default is FALSE.
- **spectral_smoothing** The length, in samples, of the triangular spectral smoothing function. Note that this must be an odd number, to ensure that smoothing is symmetrical around the centre frequency. Default is 3.
- **coherence2_thresholds** The critical values (alpha=5%, second column) for coherence for a number of windows (first column, here from 3 to 15). These values were obtained by Monte Carlo simulation, using the default parameter settings for the TFA-analysis (Hanning window, overlap of 50% and 3-point spectral smoothing was assumed). These values should be recalculated for different settings. Note that if `overlap_adjust=TRUE`, the overlap will vary depending on the length of data. With an overlap of 60% (see below), the critical values increase by between 0.04 (for 3 windows) and 0.02 (for 15 windows). Default is `cbind(c(3:15), c(0.51, 0.40, 0.34, 0.29, 0.25, 0.22, 0.20, 0.18, 0.17, 0.15, 0.14, 0.13, 0.12))`.
- **apply_coherence2_threshold** Apply the thresholds given above to the TFA-estimates. All frequencies with magnitude-squared coherence below the threshold value are excluded from averaging when calculating the mean values of gain and phase across the bands. Note that low values of coherence are not excluded in the average of coherence across the bands. Default is TRUE.

- `remove_negative_phase` Remove (ignore) negative values of phase in averaging across bands. Negative phase values are removed only for frequencies below the frequency given below, when calculating the average phase in bands. Default is TRUE.
- `remove_negative_phase_f_cutoff` The cut-off frequency below-which negative phase values are neglected (only if `remove_negative_phase` is TRUE). Default is 0.1.
- `normalize_ABP` Normalize ABP by dividing by the mean and multiplying by 100, to express ABP change in %. Note that mean-values are always removed from ABP prior to analysis. Default is FALSE.
- `normalize_CBFV` Normalize CBFV by dividing by the mean and multiplying by 100, to express CBFV change in %. Note that the band-average values of gain are always calculated both with and without normalization of CBFV, in accordance with the recommendations. Note also that mean-values are always removed from CBFV prior to analysis. Default is FALSE.
- `window_type` Chose window 'hanning' or 'boxcar'. Default is 'hanning'.
- `window_length` Length of the data-window, in seconds. Default is 102.4.
- `overlap` Overlap of the windows, in %. If `overlap_adjust` is TRUE (see below), then this value may be automatically reduced, to ensure that windows cover the full length of data. Default is 59.99% rather than 60%, so that with data corresponding to 5 windows of 100 s at an overlap of 50%, 5 windows are indeed chosen.
- `overlap_adjust` Ensure that the full length of data is used (i.e. the last window finishes as near as possible to the end of the recording), by adjusting the overlap up to a maximum value given by `params.overlap`. Default is TRUE.
- `na_as_mean` Changes all missing non-interpolated values to the mean value of the corresponding variable. This have not been adressed in the paper by Claassen, and to ensure the dataframes are not 'gathered' this should generate the most stable results. Default is TRUE.

References

1. Claassen et al. (2016) J Cereb Blood Flow Metab. 2016 Apr;36(4):665-80. ([PubMed](#))

Examples

```
data(tfa_sample_data)
TFA(tfa_sample_data[,c(1:3)], variables=c("abp","mcav"), freq=10)
```

| | |
|-----------------|------------------------|
| tfa_sample_data | <i>TFA sample data</i> |
|-----------------|------------------------|

Description

Dataframe with data provided by Prof. Simpsons, with time (t), arterial blood pressure (abp), left MCAv (mcav_l), right MCAv (mcav_r), and end-tidal CO2 (etco2).

Usage

```
data(tfa_sample_data)
```

Format

An object of class "dataframe"; an example of the usage in [TFA](#)-function.

Source

[GitHub](#)

References

- Simpsons D (2015) ([Cerebral Autoregulation Research Network](#))
- Claassen et al. (2016) J Cereb Blood Flow Metab. 2016 Apr;36(4):665-80. ([PubMed](#))

Examples

```
data(tfa_sample_data)
TFA(tfa_sample_data[,c(1:3)], variables=c("abp", "mcav"), freq=10)
```

| | |
|-------------------|----------------------------|
| tfa_sample_data_1 | <i>TFA sample data - 1</i> |
|-------------------|----------------------------|

Description

Dataframe with data provided by Prof. Simpsons, with time (t), arterial blood pressure (abp), left MCAv (mcav_l), right MCAv (mcav_r), and end-tidal CO2 (etco2).

Usage

```
data(tfa_sample_data)
```

Format

An object of class "dataframe"; an example of the usage in [TFA](#)-function.

Source

[GitHub](#)

References

- Simpsons D (2015) ([Cerebral Autoregulation Research Network](#))
- Claassen et al. (2016) J Cereb Blood Flow Metab. 2016 Apr;36(4):665-80. ([PubMed](#))

Examples

```
data(tfa_sample_data)
TFA(tfa_sample_data_1[,c(1:3)], variables=c("abp", "mcav"), freq=10)
```

| | |
|-------------------|----------------------------|
| tfa_sample_data_2 | <i>TFA sample data - 2</i> |
|-------------------|----------------------------|

Description

Dataframe with data provided by Prof. Simpsons, with time (t), arterial blood pressure (abp), left MCAv (mcav_l), right MCAv (mcav_r), and end-tidal CO2 (etco2).

Usage

```
data(tfa_sample_data)
```

Format

An object of class "dataframe"; an example of the usage in [TFA](#)-function.

Source

[GitHub](#)

References

- Simpsons D (2015) ([Cerebral Autoregulation Research Network](#))
- Claassen et al. (2016) J Cereb Blood Flow Metab. 2016 Apr;36(4):665-80. ([PubMed](#))

Examples

```
data(tfa_sample_data)
TFA(tfa_sample_data_2[,c(1:3)], variables=c("abp", "mcav"), freq=10)
```

Index

* datasets

- df.data1000, 10
- df.deleter, 11
- testdata10, 21
- tfa_sample_data, 26
- tfa_sample_data_1, 27
- tfa_sample_data_2, 28

- calcrel, 2
- cdm.fig, 3
- cdm.miss, 4
- clinmon, 5, 10, 11, 21
- comparerel, 9

- danishcpr, 10
- df.data10 (testdata10), 21
- df.data1000, 10
- df.deleter, 11
- dilations, 11

- iscus, 13

- ortable, 14

- PLR3000, 14
- print.rrGcomp (rrGcomp), 18

- questionnaire, 15
- quietly, 16

- recol, 17
- rowMedians, 17
- rrGcomp, 18

- sRCT, 19

- tbl, 20
- testdata10, 21
- TFA, 22, 27, 28
- tfa_sample_data, 26
- tfa_sample_data_1, 27
- tfa_sample_data_2, 28