Package ‘scan’

August 8, 2023

Type Package

Title Single-Case Data Analyses for Single and Multiple Baseline Designs

Version 0.60.0

Date 2023-08-08

Depends R (>= 3.5.0)

Imports stats, nlme, utils, methods, graphics, car, knitr, kableExtra, readxl, mblm, magrittr

Suggests testthat (>= 3.0.0), shiny, yaml, scplot


License GPL (>= 3)

URL https://github.com/jazznbass/scan/,
https://jazznbass.github.io/scan-Book/,
https://github.com/jazznbass/scan

BugReports https://github.com/jazznbass/scan/issues

LazyData true

Encoding UTF-8

RoxygenNote 7.2.3

Config/testthat/edition 3

NeedsCompilation no

Author Juergen Wilbert [cre, aut] (<https://orcid.org/0000-0002-8392-2873>),
Timo Lueke [aut] (<https://orcid.org/0000-0002-2603-7341>)}
R topics documented:

Maintainer  Juergen Wilbert <juergen.wilbert@uni-potsdam.de>
Repository  CRAN
Date/Publication  2023-08-08 13:00:02 UTC

R topics documented:

add_l2 ................................................................. 3
as.data.frame.scdf .................................................. 4
as_scdf ............................................................... 5
autocorr ............................................................... 6
batch_apply ........................................................... 7
cdc ..................................................................... 7
coeff_sc_plm ........................................................... 9
combine ................................................................. 10
convert ................................................................. 11
corrected_tau .......................................................... 12
describer ............................................................... 13
design .................................................................. 14
estimate_design ........................................................ 17
export ................................................................. 19
fill_missing ............................................................. 22
hplm ................................................................. 23
ird .............................................................. 27
is_scdf ............................................................... 28
moving_median .......................................................... 28
mplm ............................................................... 31
nap ................................................................. 33
outlier ................................................................. 34
overlap ............................................................... 36
pand ................................................................. 37
pem ................................................................. 40
pet ................................................................. 42
plm ................................................................. 43
plot.scdf ............................................................. 46
plot_rand .............................................................. 49
pnd ................................................................. 50
power_test ............................................................ 51
print.scdf ............................................................. 53
random_scdf .......................................................... 54
rand_test ............................................................ 55
rci ................................................................. 58
read_scdf ............................................................. 59
sample_names .......................................................... 60
scdf ................................................................. 61
select_cases ............................................................ 63
select_phases .......................................................... 64
set_vars ............................................................. 65
**add_l2**

Add level-2 data

**Description**

Merges variables with corresponding case names from a data.frame with an scdf file.

**Usage**

```r
add_l2(scdf, data_l2, cvar = "case")
```

**Arguments**

- `scdf`: A single-case data frame. See `scdf()` to learn about this format.
- `data_l2`: A level 2 dataset.
- `cvar`: Character string with the name of the "case" variable in the L2 dataset (default is 'case').

**Details**

This function is mostly used in combination with the `hplm()` function.

**Value**

An scdf

**See Also**

- `hplm()`
- Other data manipulation functions: `as.data.frame.scdf()`, `as_scdf()`, `fill_missing()`, `moving_median()`, `outlier()`, `ranks()`, `scdf()`, `select_cases()`, `set_vars()`, `shift()`, `smooth_cases()`, `standardize()`, `truncate_phase()`

**Examples**

```r
Leidig2018 %>% add_l2(Leidig2018_l2)
```
as.data.frame.scdf  

Creating a long format data frame from several single-case data frames (scdf).

Description

The as.data.frame function transposes an scdf into one long data frame. Additionally, a data frame can be merged that includes level 2 data of the subjects. This might be helpful to prepare data to be used with other packages than scan.

Usage

## S3 method for class 'scdf'
as.data.frame(x, ..., l2 = NULL, id = "case")

Arguments

x  An scdf object

... Not implemented

l2 A data frame providing additional variables at Level 2. The scdf has to have names for all cases and the Level 2 data frame has to have a column with corresponding case names.

id Variable name of the Level 2 data frame that contains the case names.

Value

Returns one data frame with data of all single-cases structured by the case variable.

Author(s)

Juergen Wilbert

See Also

Other data manipulation functions: add_l2(), as_scdf(), fill_missing(), moving_median(), outlier(), ranks(), scdf(), select_cases(), set_vars(), shift(), smooth_cases(), standardize(), truncate_phase()

Examples

## Convert the list of three single-case data frames from Grosche (2011)
### into one long data frame
Grosche2011
Grosche2011_long <- as.data.frame(Grosche2011)
Grosche2011_long
as_scdf

## Combine an scdf with data for l2
Leidig2018_long <- as.data.frame(Leidig2018, l2 = Leidig2018_l2)
names(Leidig2018_long)
summary(Leidig2018_long)

---

**as_scdf**

**as_scdf**

### Description

Converts a data frame to an scdf object.

### Usage

```r
as_scdf(
  object,
  cvar = "case",
  pvar = "phase",
  dvar = "values",
  mvar = "mt",
  phase_names = NULL,
  sort_cases = FALSE
)
```

### Arguments

- **object**: A data.frame
- **cvar**: Sets the "case" variable. Defaults to `case`.
- **pvar**: Sets the "phase" variable. Defaults to `phase`.
- **dvar**: Sets the "values" variable. Defaults to `values`.
- **mvar**: Sets the variable name of the "mt" variable. Defaults to `mt`.
- **phase_names**: A character vector with phase names. Defaults to the phase names provided in the phase variable.
- **sort_cases**: If set TRUE, the resulting list is sorted by label names (alphabetically increasing).

### Value

An scdf.

### See Also

Other data manipulation functions: `add_l2()`, `as.data.frame.scdf()`, `fill_missing()`, `moving_median()`, `outlier()`, `ranks()`, `scdf()`, `select_cases()`, `set_vars()`, `shift()`, `smooth_cases()`, `standardize()`, `truncate_phase()`
autocorr

Autocorrelation for single-case data

Description

The autocorr function calculates autocorrelations within each phase and across all phases.

Usage

autocorr(data, dvar, pvar, mvar, lag_max = 3, lag.max, ...)

Arguments

data A single-case data frame. See scdf() to learn about this format.
dvar Character string with the name of the dependent variable. Defaults to the attributes in the scdf file.
pvar Character string with the name of the phase variable. Defaults to the attributes in the scdf file.
mvar Character string with the name of the measurement time variable. Defaults to the attributes in the scdf file.
lag_max, lag.max The lag up to which autocorrelations will be computed.
... Further arguments passed to the acf() function

Value

A data frame containing separate autocorrelations for each phase and for all phases (for each single-case). If lag_max exceeds the length of a phase minus one, NA is returned for this cell.

Author(s)

Juergen Wilbert

See Also

acf()

Other regression functions: corrected_tau(), hplm(), mplm(), plm(), trend()

Examples

## Compute autocorrelations for a list of four single-cases up to lag 2.
autocorr(Huber2014, lag_max = 2)
**batch_apply**

*Apply a function to each element in an scdf.*

**Description**

This function applies a given function to each case of a multiple case scdf, returning a list of the output of each function call.

**Usage**

```
batch_apply(scdf, fn, simplify = FALSE)
```

**Arguments**

- `scdf` A list of inputs to apply the function to.
- `fn` The function to apply to each element. Use a . as a placeholder for the scdf (e.g. `describe(.)`).
- `simplify` If simplify is TRUE and fn returns a vector of values, batch_apply will return a data frame case names.

**Value**

A list of the output of each function call.

**Examples**

```
batch_apply(exampleAB, coef(plm(.)))
```

---

**cdc**

*Conservative Dual-Criterion Method*

**Description**

The cdc() function applies the Conservative Dual-Criterion Method (Fisher, Kelley, & Lomas, 2003) to scdf objects. It compares phase B data points to both phase A mean and trend (OLS, bi-split, tri-split) with an additional increase/decrease of .25 SD. A binomial test against a 50/50 distribution is computed and p-values below .05 are labeled "systematic change". 
Usage

```r
cdc(
  data,
  dvar,
  pvar,
  mvar,
  decreasing = FALSE,
  trend_method = c("OLS", "bisplit", "trisplit"),
  conservative = 0.25,
  phases = c(1, 2)
)
```

Arguments

- **data**: A single-case data frame. See `scdf()` to learn about this format.
- **dvar**: Character string with the name of the dependent variable. Defaults to the attributes in the scdf file.
- **pvar**: Character string with the name of the phase variable. Defaults to the attributes in the scdf file.
- **mvar**: Character string with the name of the measurement time variable. Defaults to the attributes in the scdf file.
- **decreasing**: If you expect data to be lower in the B phase, set `decreasing = TRUE`. Default is `decreasing = FALSE`.
- **trend_method**: Method used to calculate the trend line. Default is `trend_method = "OLS"`. Possible values are: "OLS", "bisplit", and "trisplit". "bisplit" and "trisplit" should only be used for cases with at least five data-points in both relevant phases.
- **conservative**: The CDC method adjusts the original mean and trend lines by adding (expected increase) or subtracting (expected decrease) an additional .25 SD before evaluating phase B data. Default is the CDC method with `conservative = .25`. To apply the Dual-Criterion (DC) method, set `conservative = 0`.
- **phases**: A vector of two characters or numbers indicating the two phases that should be compared. E.g., `phases = c("A", "C")` or `phases = c(2,4)` for comparing the second to the fourth phase. Phases could be combined by providing a list with two elements. E.g., `phases = list(A = c(1,3), B = c(2,4))` will compare phases 1 and 3 (as A) against 2 and 4 (as B). Default is `phases = c(1,2)`.

Value

- **cdc**: CDC Evaluation based on a p-value below .05.
- **cdc_exc**: Number of phase B datapoints indicating expected change.
- **cdc_nb**: Number of phase B datapoints.
- **cdc_p**: P value of Binomial Test.
- **cdc_all**: Overall CDC Evaluation based on all instances/cases of a Multiple Baseline Design.
coef.sc_plm

**N**  
Number of cases.

decreasing  
Logical argument from function call (see Arguments above).

conservative  
Numeric argument from function call (see Arguments above).

case_names  
Assigned name of single-case.

phases  
-

**Author(s)**

Timo Lueke

**References**


**See Also**


**Examples**

```r
## Apply the CDC method (standard OLS line)
design <- design(n = 1, slope = 0.2)
dat <- random_scdf(design, seed = 42)
cdc(dat)

## Apply the CDC with Koenig's bi-split and an expected decrease in phase B.
cdc(exampleAB_decreasing, decreasing = TRUE, trend_method = "bisplit")

## Apply the CDC with Tukey's tri-split, comparing the first and fourth phase.
cdc(exampleABAB, trend_method = "trisplit", phases = c(1,4))

## Apply the Dual-Criterion (DC) method (i.e., mean and trend without shifting).
cdc(exampleAB_decreasing, decreasing = TRUE, trend_method = "bisplit", conservative = 0)
```

---

**coef.sc_plm**  
*Extract coefficients from plm/hplm objects*

**Description**

Extract coefficients from plm/hplm objects
Usage

```r
## S3 method for class 'sc_plm'
coef(object, ...)
```

Arguments

- `object` plm or hplm object
- `...` not implemented

Value
data frame with coefficient table

Examples

```r
coefficients(plm(exampleAB$Johanna))
```

---

### combine

**Combine single-case data frames**

Usage

```r
combine(..., dvar = NULL, pvar = NULL, mvar = NULL, info = NULL, author = NULL)
```

Arguments

- `...` scdf objects
- `dvar` Character string. Name of the dependent variable. Defaults to the dependent variable of the first case provided.
- `pvar` Character string. Name of the phase variable. Defaults to the phase variable of the first case provided.
- `mvar` Character string. Name of the measurement-time variable. Defaults to the measurement-time variable of the first case provided.
- `info` additional information on the scdf file.
- `author` author of the data.

Value

A scdf. If not set differently, the attributes of this scdf are copied from the first scdf provided (i.e. the first argument of the function).
Description

Converts an scdf object into R code

Usage

```r
convert(
  scdf,
  file = "",
  study_name = "study",
  case_name = "case",
  inline = FALSE,
  indent = 2,
  silent = FALSE
)
```

Arguments

- `scdf`: A single-case data frame. See `scdf()` to learn about this format.
- `file`: A filename for exporting the syntax.
- `study_name`: Character string. Name of the study object.
- `case_name`: Character string. Name of the scdf objects.
- `inline`: If TRUE, phase definition is in an online version.
- `silent`: If TRUE, syntax is not printed to the console

Value

Returns a string (invisible).

See Also

Other io-functions: `read_scdf()`, `write_scdf()`

Examples

```r
filename <- tempfile()
convert(exampleABC, file = filename)
source(filename)
all.equal(study, exampleABC)
unlink(filename)
```
corrected_tau

Baseline corrected tau

Description

Kendall’s tau correlation for the dependent variable and the phase variable is calculated after correcting for a baseline trend.

Usage

```r
corrected_tau(
  data,
  dvar,
  pvar,
  mvar,
  phases = c(1, 2),
  alpha = 0.05,
  continuity = FALSE,
  repeated = FALSE,
  tau_method = "b"
)
```

Arguments

- **data**: A single-case data frame. See `scdf()` to learn about this format.
- **dvar**: Character string with the name of the dependent variable. Defaults to the attributes in the scdf file.
- **pvar**: Character string with the name of the phase variable. Defaults to the attributes in the scdf file.
- **mvar**: Character string with the name of the measurement time variable. Defaults to the attributes in the scdf file.
- **phases**: A vector of two characters or numbers indicating the two phases that should be compared. E.g., `phases = c("A", "C")` or `phases = c(2, 4)` for comparing the second to the fourth phase. Phases could be combined by providing a list with two elements. E.g., `phases = list(A = c(1, 3), B = c(2, 4))` will compare phases 1 and 3 (as A) against 2 and 4 (as B). Default is `phases = c(1, 2)`.
- **alpha**: Sets the p-value at and below which a baseline correction is applied.
- **continuity**: If TRUE applies a continuity correction for calculating p.
- **repeated**: If TRUE applies the repeated median method for calculating slope and intercept (`mblm()`)
- **tau_method**: Character with values "a" or "b" (default) indicating whether Kendall Tau A or Kendall Tau B is applied.
describe

Details

This method has been proposed by Tarlow (2016). The baseline data are checked for a significant autocorrelation (based on Kendall’s Tau). If so, a non-parametric Theil-Sen regression is applied for the baseline data where the dependent values are regressed on the measurement time. The resulting slope information is then used to predict data of the B-phase. The dependent variable is now corrected for this baseline trend and the residuals of the Theil-Sen regression are taken for further calculations. Finally, Kendall’s tau is calculated for the dependent variable and the dichotomous phase variable. The function here provides two extensions to this procedure: The more accurate Siegel repeated median regression is applied when repeated = TRUE and a continuity correction is applied when continuity = TRUE.

References


See Also

Other regression functions: autocorr(), hplm(), mplm(), plm(), trend()

Examples

dat <- scdf(c(A = 33, 25, 17, 25, 14, 13, 15, B = 15, 16, 16, 5, 7, 9, 6, 5, 3, 3, 8, 11, 7))
corrected_tau(dat)

describe

Descriptive statistics for single-case data

Description

The describe() function provides common descriptive statistics for single-case data.

Usage

describe(data, dvar, pvar, mvar)

Arguments

data A single-case data frame. See scdf() to learn about this format.
dvar Character string with the name of the dependent variable. Defaults to the attributes in the scdf file.
pvar Character string with the name of the phase variable. Defaults to the attributes in the scdf file.
mvar Character string with the name of the measurement time variable. Defaults to the attributes in the scdf file.
Details

\( n \) = number of measurements; \( \text{mis} \) = number of missing values; \( m \) = mean; \( \text{md} \) = median; \( \text{sd} \) = standard deviation; \( \text{mad} \) = median average deviation; \( \text{min} \) = minimum; \( \text{max} \) = maximum; \( \text{trend} \) = weight of depended variable regressed on time (values ~ mt).

Value

A list containing a data frame of descriptive statistics (descriptives); the cse design (design); the number of cases (N)

Author(s)

Juergen Wilbert

See Also

\( \text{overlap()} \), \( \text{plot.scdf()} \)

Examples

```r
## Descriptive statistics for a study of three single-cases
describe(Grosche2011)

## Descriptives of a three phase design
describe(exampleABC)

## Write descriptive statistics to .csv-file
study <- describe(Wadde2011)
write.csv(study$descriptives, file = tempfile())
```

---

**design**

*Generate a single-case design matrix*

**Description**

Generates a parameter list used for generating multiple random single-cases. This is used within the \text{random_scdf} function and the \text{power_test} function and for other Monte-Carlo tasks.

**Usage**

```r
design(
  n = 1,
  phase_design = list(A = 5, B = 15),
  trend = 0,
  level = list(0),
  slope = list(0),
  start_value = 50,
)```

\begin{verbatim}

s = 10,
rtt = 0.8,
extreme_prop = list(0),
extreme_range = c(-4, -3),
missing_prop = 0,
distribution = c("normal", "gaussian", "poisson", "binomial"),
n_trials = NULL,
mt = NULL,
B_start = NULL,
m,
phase.design,
MT,
B.start,
extreme.p,
extreme.d,
missing.p

Arguments

n Number of cases to be designed (Default is n = 1).

phase_design, phase.design
A list defining the length and label of each phase. E.g., phase.length = list(A1 = 10, B1 = 10, A2 = 10, B2 = 10). Use vectors if you want to define different values for each case phase.length = list(A = c(10, 15), B = c(10, 15).

trend Defines the effect size of a trend added incrementally to each measurement across the whole data-set. To assign different trends to several single-cases, use a vector of values (e.g. trend = c(.1, .3, .5)). If the number of cases exceeds the length of the vector, values are recycled. When using a 'gaussian' distribution, the trend parameters indicate effect size changes. When using a binomial or poisson distribution, trend indicates an increase in points / counts per measurement.

level A list that defines the level increase (effect size d) at the beginning of each phase relative to the previous phase (e.g. list(A = 0, B = 1)). The first element must be zero as the first phase of a single-case has no level effect (if you have one less list element than the number of phases, scan will add a leading element with 0 values). Use vectors to define variable level effects for each case (e.g. list(A = c(0, 0), B = c(1, 2))). When using a 'gaussian' distribution, the level parameters indicate effect size d changes. When using a binomial or poisson distribution, level indicates an increase in points / counts with the onset of each phase.

slope A list that defines the increase per measurement for each phase compared to the previous phase. slope = list(A = 0, B = .1) generates an incremental increase of 0.1 per measurement starting at the B phase. The first list element must be zero as the first phase of a single-case has no slope effect (if you have one less list element than the number of phases, scan will add a leading element with 0 values). Use vectors to define variable slope effects for each case (e.g. list(A = c(0, 0), B = c(0.1, 0.2))). If the number of cases exceeds the length of

\end{verbatim}
the vector, values are recycled. When using a 'gaussian' distribution, the slope parameters indicate effect size \( d \) changes per measurement. When using a binomial or poisson distribution, slope indicates an increase in points / counts per measurement.

**start_value, \( m \)** Starting value at the first measurement. Default is 50. When distribution = "poisson" the start_value represents frequency. When distribution = "binomial" start_value must range between 0 and 1 and they represent the probability of an event. To assign different start values to several single-cases, use a vector of values (e.g. \( c(50, 42, 56) \)). If the number of cases exceeds the length of the vector, values are recycled. The \( m \) argument is deprecated.

**s** Standard deviation used to calculate absolute values from level, slope, trend effects and to calculate and error distribution from the rtt values. Set to 10 by default. To assign different variances to several single-cases, use a vector of values (e.g. \( s = c(5, 10, 15) \)). If the number of cases exceeds the length of the vector, values are recycled. if the distribution is 'poisson' or 'binomial' s is not applied.

**rtt** Reliability of the underlying simulated measurements. Set rtt = .8 by default. To assign different reliabilities to several single-cases, use a vector of values (e.g. \( rtt = c(.6, .7, .8) \)). If the number of cases exceeds the length of the vector, values are repeated. rtt has no effect when you’re using binomial or poisson distributions.

**extreme_prop, extreme.p** Probability of extreme values. extreme.p = .05 gives a five percent probability of an extreme value. A vector of values assigns different probabilities to multiple cases. If the number of cases exceeds the length of the vector, values are recycled.

**extreme_range, extreme.d** Range for extreme values. extreme_range = c(-7, -6) uses extreme values within a range of -7 and -6. In case of a binomial or poisson distribution, extreme_range indicates frequencies. In case of a gaussian (or normal) distribution it indicates effect size \( d \). Caution: the first value must be smaller than the second, otherwise the procedure will fail.

**missing_prop, missing.p** Portion of missing values. missing_prop = 0.1 creates 10\% different probabilities to multiple cases. If the number of cases exceeds the length of the vector, values are recycled.

**distribution** Distribution of the criteria variable. Default is "normal". Possible values are "normal", "binomial", and "poisson".

**n_trials** If distribution (see below) is "binomial", n_trials is the number of trials/observations/items.

**mt, MT** Number of measurements (in each study). Default is mt = 20.

**B_start, B.start** Phase B starting point. The default setting B_start = 6 would assign the first five scores (of each case) to phase A, and all following scores to phase B. To assign different starting points for a set of multiple single-cases, use a vector of starting values (e.g., B_start = c(6, 7, 8)). If the number of cases exceeds the length of the vector, values will be recycled.
estimate_design

Value

An object of class sc_design.

Author(s)

Juergen Wibert

Examples

```r
## Create random single-case data and inspect it
design <- design(
  n = 3, rtt = 0.75, slope = 0.1, extreme_prop = 0.1,
  missing_prop = 0.1
)
dat <- random_scdf(design, round = 1, random.names = TRUE, seed = 123)
describe(dat)

## And now have a look at poisson-distributed data
design <- design(
  n = 3, B_start = c(6, 10, 14), mt = c(12, 20, 22), start_value = 10,
  distribution = "poisson", level = -5, missing_prop = 0.1
)
dat <- random_scdf(design, seed = 1234)
pand(dat, decreasing = TRUE)
```

estimate_design

Estimate single-case design

Description

This function takes an scdf and extracts design parameters. The resulting object can be used to randomly create new scdf files with the same underlying parameters. This is useful for Monte-Carlo studies and bootstrapping procedures.

Usage

```r
estimate_design(
  data,
  dvar,
  pvar,
  mvar,
  s = NULL,
  rtt = NULL,
  overall_effects = FALSE,
  overall_rtt = TRUE,
  model = "JW",
  ...  
)
```
**estimate_design**

**Arguments**

- **data**
  A single-case data frame. See `scdf()` to learn about this format.

- **dvar**
  Character string with the name of the dependent variable. Defaults to the attributes in the scdf file.

- **pvar**
  Character string with the name of the phase variable. Defaults to the attributes in the scdf file.

- **mvar**
  Character string with the name of the measurement time variable. Defaults to the attributes in the scdf file.

- **s**
  The standard deviation depicting the between case variance of the overall performance. If more than two single-cases are included in the scdf, the variance is estimated if s is set to NULL.

- **rtt**
  The reliability of the measurements. The reliability is estimated when rtt = NULL.

- **overall_effects**
  If TRUE, trend, level, and slope effect estimations will be identical for each case. If FALSE, effects are estimated for each case separately.

- **overall_rtt**
  Ignored when rtt is set. If TRUE, rtt estimations will be based on all cases and identical for each case. If FALSE rtt is estimated for each case separately.

- **model**
  Model used for calculating the dummy parameters (see Huitema & McKean, 2000). Default is `model = "W"`. Possible values are: "B&L-B", "H-M", "W", and deprecated "JW".

- **...**
  Further arguments passed to the plm function used for parameter estimation.

**Value**

A list of parameters for each single-case. Parameters include name, length, and starting measurement time of each phase, trend, level, and slope effects for each phase, start value, standard deviation, and reliability for each case.

**Examples**

```r
# create a random scdf with predefined parameters
set.seed(1234)
design <- design(
  n = 10, trend = -0.02,
  level = list(0, 1), rtt = 0.8,
  s = 1
)
scdf <- random_scdf(design)

# Estimate the parameters based on the scdf and create a new random scdf
# based on these estimations
design_est <- estimate_design(scdf, rtt = 0.8)
scdf_est <- random_scdf(design_est)

# Analyze both datasets with an hplm model. See how similar the estimations
# are:
```
hplm(scdf, slope = FALSE)
hplm(scdf_est, slope = FALSE)

# Also similar results for pand and randomization tests:
pand(scdf)
pand(scdf_est)
rand_test(scdf)
rand_test(scdf_est)

---

`export`  
*Export scan objects to html or latex*

**Description**

Export creates html files of tables or displays them directly in the viewer pane of rstudio. When applied in rmarkdown/quarto, tables can also be created for pdf/latex output.

**Usage**

```r
export(object, ...)  
```

### S3 method for class 'sc_desc'

```r
e/export(
  object,
  caption = NA,
  footnote = NA,
  filename = NA,
  kable_styling_options = list(),
  kable_options = list(),
  flip = FALSE,
  ...
)  
```

### S3 method for class 'sc_nap'

```r
e/export(
  object,
  caption = NA,
  footnote = NA,
  filename = NA,
  kable_styling_options = list(),
  kable_options = list(),
  round = 2,
  ...
)  
```

### S3 method for class 'sc_overlap'

```r
e/export(
  object,
  ...
)  
```
caption = NA,
footnote = NULL,
filename = NA,
kable_styling_options = list(),
kable_options = list(),
round = 2,
flip = FALSE,
...
)

## S3 method for class 'sc_pem'
export(
  object,
  caption = NA,
  footnote = NA,
  filename = NA,
  kable_styling_options = list(),
kable_options = list(),
  round = 2,
  nice = TRUE,
...
)

## S3 method for class 'sc_plm'
export(
  object,
  caption = NA,
  footnote = NA,
  filename = NA,
  kable_styling_options = list(),
kable_options = list(),
  round = 2,
  nice = TRUE,
...
)

## S3 method for class 'sc_power'
export(
  object,
  caption = NA,
  footnote = NA,
  filename = NA,
  kable_styling_options = list(),
kable_options = list(),
...
)

## S3 method for class 'sc_smd'
export(}


object, caption = NA, footnote = NA, filename = NA, select = c("Case", 'Mean A' = "mA", 'Mean B' = "mB", 'SD A' = "sdA", 'SD B' = "sdB", 'SD Cohen' = "sd cohen", 'SD Hedges' = "sd hedges", "Glass' delta", "Hedges' g", "Hedges' g correction", "Hedges' g durlak correction", "Cohen's d"), kable_styling_options = list(), kable_options = list(), round = 2, flip = FALSE, ...

## S3 method for class 'sc_trend'
export(
  object, caption = NA, footnote = NA, filename = NA, kable_styling_options = list(), kable_options = list(), round = 2, ...
)

## S3 method for class 'scdf'
export(
  object, caption = NA, footnote = NA, filename = NA, kable_styling_options = list(), kable_options = list(), cols, ...
)

Arguments

object  An scdf or an object exported from a scan function.
...

Further Arguments passed to internal functions.
caption  Character string with table caption. If left NA (default) a caption will be created based on the exported object.
footnote  Character string with table footnote. If left NA (default) a footnote will be created based on the exported object.
filename  String containing the file name. If a filename is given the output will be written to that file.
The `fill_missing()` function replaces missing measurements in single-case data.

### Usage

```r
fill_missing(data, dvar, mvar, na.rm = TRUE)
```

### Arguments

- **data**: A single-case data frame. See `scdf()` to learn about this format.
- **dvar**: Character string with the name of the dependent variable. Defaults to the attributes in the scdf file.
- **mvar**: Character string with the name of the measurement time variable. Defaults to the attributes in the scdf file.
- **na.rm**: If set TRUE, NA values are also interpolated. Default is `na.rm = TRUE`.

### Details

This procedure is recommended if there are gaps between measurement times (e.g. MT: 1, 2, 3, 4, 5, ... 8, 9) or explicitly missing values in your single-case data and you want to calculate overlap indices (`overlap()`)

### Value

A single-case data frame with interpolated missing data points. See `scdf()` to learn about the SCDF Format.
**Author(s)**

Juergen Wilbert

**See Also**

Other data manipulation functions: `add_l2()`, `as.data.frame.scdf()`, `as_scdf()`, `moving_median()`, `outlier()`, `ranks()`, `scdf()`, `select_cases()`, `set_vars()`, `shift()`, `smooth_cases()`, `standardize()`, `truncate_phase()`

**Examples**

```r
## In his study, Grosche (2011) could not realize measurements each
## single week for all participants. During the course of 100 weeks,
## about 20 measurements per person at different times were administered.

## Fill missing values in a single-case dataset with discontinuous
## measurement times
Grosche2011filled <- fill_missing(Grosche2011)
study <- c(Grosche2011[2], Grosche2011filled[2])
names(study) <- c("Original", "Filled")
plot(study)

## Fill missing values in a single-case dataset that are NA
Maggie <- random_scdf(design(level = list(0,1)), seed = 123)
Maggie_n <- Maggie
replace.positions <- c(10,16,18)
Maggie_n[[1]][replace.positions,"values"] <- NA
Maggie_f <- fill_missing(Maggie_n)
study <- c(Maggie, Maggie_n, Maggie_f)
names(study) <- c("original", "missing", "interpolated")
plot(study, marks = list(positions = replace.positions), style = "grid2")
```

---

**hplm**

**Hierarchical piecewise linear model / piecewise regression**

**Description**

The `hplm()` function computes a hierarchical piecewise regression model.

**Usage**

```r
hplm(
  data,
  dvar,
  pvar,
  mvar,
  model = c("W", "H-M", "B&L-B", "JW"),
)```
contrast = c("first", "preceding"),
contrast_level = NA,
contrast_slope = NA,
method = c("ML", "REML"),
control = list(opt = "optim"),
random.slopes = FALSE,
lr.test = FALSE,
ICC = TRUE,
trend = TRUE,
level = TRUE,
slope = TRUE,
fixed = NULL,
random = NULL,
update.fixed = NULL,
data.12 = NULL,
...
)

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

## S3 method for class 'sc_hplm'
export(
  object,
  caption = NA,
  footnote = NA,
  filename = NA,
  kable_styling_options = list(),
  kable_options = list(),
  round = 2,
  nice = TRUE,
  ...
)

## S3 method for class 'sc_hplm'
coef(object, ...)

Arguments

- **data**
  A single-case data frame. See `scdf()` to learn about this format.

- **dvar**
  Character string with the name of the dependent variable. Defaults to the attributes in the scdf file.

- **pvar**
  Character string with the name of the phase variable. Defaults to the attributes in the scdf file.

- **mvar**
  Character string with the name of the measurement time variable. Defaults to the attributes in the scdf file.

- **model**
  Model used for calculating the dummy parameters (see Huijema & McKeen, 2000). Default is `model = "W"`. Possible values are: "B&L-B", "H-M", "W", and
deprecated "JW".

**contrast**
Sets contrast_level and contrast_slope. Either "first", "preceding" or a contrast matrix.

**contrast_level**
Either "first", "preceding" or a contrast matrix. If NA contrast_level is a copy of contrast.

**contrast_slope**
Either "first", "preceding" or a contrast matrix. If NA contrast_level is a copy of contrast.

**method**
Method used to fit your model. Pass "REML" to maximize the restricted log-likelihood or "ML" for maximized log-likelihood. Default is "ML".

**control**
A list of settings for the estimation algorithm, replacing the default values passed to the function lmeControl of the nlme package.

**random.slopes**
If random.slopes = TRUE random slope effects of the level, trend, and treatment parameter are estimated.

**lr.test**
If set TRUE likelihood ratio tests are calculated comparing model with vs. without random slope parameters.

**ICC**
If ICC = TRUE an intraclass-correlation is estimated.

**trend**
A logical indicating if a trend parameters is included in the model.

**level**
A logical indicating if a level parameters is included in the model.

**slope**
A logical indicating if a slope parameters is included in the model.

**fixed**
Defaults to the fixed part of the standard piecewise regression model. The parameter phase followed by the phase name (e.g., phaseB) indicates the level effect of the corresponding phase. The parameter ’inter’ followed by the phase name (e.g., interB) addresses the slope effect based on the method provide in the model argument (e.g., "B&L-B"). The formula can be changed for example to include further L1 or L2 variables into the regression model.

**random**
The random part of the model.

**update.fixed**
An easier way to change the fixed model part (e.g., ~ . + newvariable).

**data.l2**
A dataframe providing additional variables at Level 2. The scdf File has to have names for all cases and the Level 2 dataframe has to have a column named 'cases' with the names of the cases the Level 2 variables belong to.

**...**
Further arguments passed to the lme function.

**x**
An object returned by hplm()

**object**
An scdf or an object exported from a scan function.

**caption**
Character string with table caption. If left NA (default) a caption will be created based on the exported object.

**footnote**
Character string with table footnote. If left NA (default) a footnote will be created based on the exported object.

**filename**
String containing the file name. If a filename is given the output will be written to that file.

**kable_styling_options**
list with arguments passed to the kable_styling function.

**kable_options**
list with arguments passed to the kable function.
round  Integer passed to the digits argument internally used to round values.
nice  If set TRUE (default) output values are rounded and optimized for publication tables.

Value

model  List containing information about the applied model
N  Number of single-cases.
formula  A list containing the fixed and the random formulas of the hplm model.
hplm  Object of class lme containing the multilevel model
model.0  Object of class lme containing the Zero Model.
ICC  List containing intraclass correlation and test parameters.
model.without  Object of class gls containing the fixed effect model.

Functions

- print(sc_hplm): Print results
- export(sc_hplm): Export results as html table (see export())
- coef(sc_hplm): Extract model coefficients

Author(s)

Juergen Wilbert

See Also

Other regression functions: autocorr(), corrected_tau(), mplm(), plm(), trend()

Examples

```r
## Compute hplm model on a MBD over fifty cases (restricted log-likelihood)
hplm(exampleAB_50, method = "REML", random.slopes = FALSE)

## Analyzing with additional L2 variables
Leidig2018 %>%
  add_l2(Leidig2018_l2) %>%
hplm(update.fixed = .~. + gender + migration + ITRF_TOTAL*phaseB,
slope = FALSE, random.slopes = FALSE, lr.test = FALSE
```

```
ird

IRD - Improvement rate difference

Description

ird() calculates the robust improvement rate difference as proposed by Parker and colleagues (2011).

Usage

ird(data, dvar, pvar, decreasing = FALSE, phases = c(1, 2))

## S3 method for class 'sc_ird'
print(x, digits = 3, ...)

Arguments

data A single-case data frame. See scdf() to learn about this format.
dvar Character string with the name of the dependent variable. Defaults to the attributes in the scdf file.
pvar Character string with the name of the phase variable. Defaults to the attributes in the scdf file.
decreasing If you expect data to be lower in the B phase, set decreasing = TRUE. Default is decreasing = FALSE.
phases A vector of two characters or numbers indicating the two phases that should be compared. E.g., phases = c("A", "C") or phases = c(2,4) for comparing the second to the fourth phase. Phases could be combined by providing a list with two elements. E.g., phases = list(A = c(1,3), B = c(2,4)) will compare phases 1 and 3 (as A) against 2 and 4 (as B). Default is phases = c(1,2).
x An object returned by ird()
digits The minimum number of significant digits to be use.
... Further arguments passed to the function.

Details

The adaptation of the improvement rate difference for single-case phase comparisons was developed by Parker and colleagues (2009). A variation called robust improvement rate difference was proposed by Parker and colleagues in 2011. This function calculates the robust improvement rate difference. It follows the formula suggested by Pustejovsky (2019). For a multiple case design, ird is based on the overall improvement rate of all cases which is the average of the irds for each case.

Functions

• print(sc_ird): Print results
References


---

is.scdf scdf objects Tests for objects of type "scdf"

Description

scdf objects Tests for objects of type "scdf"

Usage

is.scdf(x)

Arguments

x An object to be tested

Value

Returns TRUE or FALSE depending on whether its argument is of scdf type or not.

---

moving_median Transform every single case of a single case data frame

Description

Takes an scdf and applies transformations to each individual case. This is useful to calculate or modify new variables.

Usage

moving_median(x, lag = 1)

moving_mean(x, lag = 1)

local_regression(x, mt = 1:length(x), f = 0.2)

first_of(x, positions = 0)
### S3 method for class 'scdf'

```r
transform(`_data`, ...)  
```

#### Arguments

- **x**  
  A logical vector.

- **lag**  
  Number of values surrounding a value to calculate the average

- **mt**  
  A vector with measurement times.

- **f**  
  the proportion of surrounding data influencing each data point.

- **positions**  
  A numeric vector with relative positions to the first appearance of a TRUE value in x.

- **...**  
  Expressions.

- **_data**  
  An scdf.

#### Details

This function is a method of the generic transform function. Unlike the generic function, it calculates expressions serially. This means that the results of the calculation of one expression are the basis for the following computations. The `n` function returns the number of measurements in a case. The `all_cases` function is a helper function that extracts the values of a variable from all cases. It takes an expression as an argument. For example, `mean(all_cases(values))` calculates the mean of the values from all cases. `mean(all_cases(values[phase == "A"]))` will calculate the mean of all values where phase is A. The function `across_cases` allows to calculate new variables or replace existing variables across all cases. E.g., `across_cases(values_ranked = rank(values, na.last = "keep"))` will calculate a new variable with values ranked across all cases.

#### Value

An scdf.

#### See Also

Other data manipulation functions: `add_l2()`, `as.data.frame.scdf()`, `as_scdf()`, `fill_missing()`, `outlier()`, `ranks()`, `scdf()`, `select_cases()`, `set_vars()`, `shift()`, `smooth_cases()`, `standardize()`, `truncate_phase()`

#### Examples

```r  
## Creates a single-case with frequency distributions. The proportion and  
## percentage of the frequencies are calculated with transform:  
design <- design(  
n = 3,  
level = 5,  
distribution = "binomial",
```r
n_trials = 20,
start_value = 0.5
)
study <- random_scdf(design)
transform(study, proportion = values/trials, percentage = proportion * 100)

## Z standardize the dependent variable and add two new variables:
exampleAB %>%
  transform(
    values = scale(values),
    mean_values = mean(values),
    sd_values = sd(values)
  )

## Use `all` to calculate global variables.
exampleAB %>%
  transform(
    values_center_case = values - mean(values[phase == "A"]),
    values_center_global = values - mean(all(values[phase == "A"])),
    value_dif = values_center_case - values_center_global
  )

## Use `across_cases` to calculate or replace a variable with values from all cases. E.g., standardize the dependent variable:
exampleABC %>%
  transform(
    across_cases(values = scale(values))
  )

## Rank transform the values based on all cases vs. within each case:
exampleABC %>%
  transform(
    across_cases(values_across = rank(values, na.last="keep")),
    value_within = rank(values, na.last="keep")
  )

## Three helper functions to smooth the data
Huber2014$Berta %>%
  transform(
    "compliance (moving median)" = moving_median(compliance),
    "compliance (moving mean)" = moving_mean(compliance),
    "compliance (local regression)" = local_regression(compliance, mt)
  )

## Function first_of() helps to set NAs for specific phases.
## E.g., you want to replace the first two values of phase A and the first value of phase B and its preceding value.
byHeart2011 %>%
  transform(
    values = replace(values, first_of(phase == "A", 0:1), NA),
    values = replace(values, first_of(phase == "B", -1:0), NA)
  )
```
**mplm**

*Multivariate Piecewise linear model / piecewise regression*

**Description**

The `mplm()` function computes a multivariate piecewise regression model.

**Usage**

```r
mplm(
  data, dvar, mvar, pvar,
  model = c("W", "H-M", "B&L-B", "JW"),
  contrast = c("first", "preceding"),
  contrast_level = c(NA, "first", "preceding"),
  contrast_slope = c(NA, "first", "preceding"),
  trend = TRUE,
  level = TRUE,
  slope = TRUE,
  formula = NULL,
  update = NULL,
  na.action = na.omit,
  ...
)
```

## S3 method for class 'sc_mplm'

```r
print(x, digits = "auto", std = FALSE, ...)
```

**Arguments**

- `data` A single-case data frame. See `scdf()` to learn about this format.
- `dvar` Character string with the name of the dependent variable. Defaults to the attributes in the scdf file.
- `mvar` Character string with the name of the measurement time variable. Defaults to the attributes in the scdf file.
- `pvar` Character string with the name of the phase variable. Defaults to the attributes in the scdf file.
- `model` Model used for calculating the dummy parameters (see Huitema & McKean, 2000). Default is `model = "W"`. Possible values are: "B&L-B", "H-M", "W", and deprecated "JW".
- `contrast` Sets `contrast_level` and `contrast_slope`. Either "first", "preceding" or a contrast matrix.
- `contrast_level` Either "first", "preceding" or a contrast matrix. If NA `contrast_level` is a copy of contrast.
contrast_slope Either "first", "preceding" or a contrast matrix. If NA contrast_level is a copy of contrast.
trend  A logical indicating if a trend parameters is included in the model.
level  A logical indicating if a level parameters is included in the model.
slope  A logical indicating if a slope parameters is included in the model.
formula  Defaults to the standard piecewise regression model. The parameter phase followed by the phase name (e.g., \texttt{phaseB}) indicates the level effect of the corresponding phase. The parameter 'inter' followed by the phase name (e.g., \texttt{interB}) addresses the slope effect based on the method provide in the model argument (e.g., \texttt{"B&L-B"}). The formula can be changed for example to include further variables into the regression model.
update  An easier way to change the regression formula (e.g., \texttt{. ~ . + newvariable}).
na.action  Defines how to deal with missing values
...  Further arguments passed to the \texttt{lm} function.
x  Object returned from \texttt{mplm}().
digits  The minimum number of significant digits to be use. If set to "auto" (default), values are predefined.
std  If TRUE, a table with standardized estimates is included.

Value

\begin{itemize}
\item model Character string from function call (see arguments above).
\item full.model Full regression model list.
\end{itemize}

Functions

- \texttt{print(sc_mplm)}: Print results

Author(s)

Juergen Wilbert

See Also

Other regression functions: \texttt{autocorr()}, \texttt{corrected_tau()}, \texttt{hplm()}, \texttt{plm()}, \texttt{trend()}

Examples

```r
res <- mplm(Leidig2018$'1a1',
  dvar = c("academic_engagement", "disruptive_behavior")
)
print(res)
## also report standardized coefficients:
print(res, std = TRUE)
```
nap

Nonoverlap of all Pairs

Description

The nap() function calculates the nonoverlap of all pairs (NAP; Parker & Vannest, 2009). NAP summarizes the overlap between all pairs of phase A and phase B data points. If an increase of phase B scores is expected, a non-overlapping pair has a higher phase B data point. The NAP equals \( \frac{\text{number of pairs showing no overlap}}{\text{number of pairs}} \). Because NAP can only take values between 50 and 100 percent, a rescaled and therefore more intuitive NAP (0-100)

Usage

nap(data, dvar, pvar, decreasing = FALSE, phases = c(1, 2))

Arguments

data A single-case data frame. See scdf() to learn about this format.
dvar Character string with the name of the dependent variable. Defaults to the attributes in the scdf file.
pvar Character string with the name of the phase variable. Defaults to the attributes in the scdf file.
decreasing If you expect data to be lower in the B phase, set decreasing = TRUE. Default is decreasing = FALSE.
phases A vector of two characters or numbers indicating the two phases that should be compared. E.g., phases = c("A","C") or phases = c(2, 4) for comparing the second to the fourth phase. Phases could be combined by providing a list with two elements. E.g., phases = list(A = c(1, 3), B = c(2, 4)) will compare phases 1 and 3 (as A) against 2 and 4 (as B). Default is phases = c(1, 2).

Value

nap A data frame with NAP and additional values for each case.
N Number of cases.

Author(s)

Juergen Wilbert

References


See Also

Other overlap functions: cdc(), overlap(), pand(), pem(), pet(), pnd(), tau_u()
Examples

```r
## Calculate NAP for a study with lower expected phase B scores
## (e.g. aggressive behavior)
gretchen <- scdf(c(A = 12, 14, 9, 10, B = 10, 6, 4, 5, 3, 4))
nap(gretchen, decreasing = TRUE)

## Request NAP for all cases from the Grosche2011 scdf
nap(Grosche2011)
```

---

### outlier

#### Handling outliers in single-case data

**Description**

Identifies and drops outliers within a single-case data frame (scdf).

**Usage**

```r
outlier(
  data,
  dvar,
  pvar,
  mvar,
  method = c("MAD", "Cook", "SD", "CI"),
  criteria = 3.5
)
```

**Arguments**

- `data`: A single-case data frame. See `scdf()` to learn about this format.
- `dvar`: Character string with the name of the dependent variable. Defaults to the attributes in the scdf file.
- `pvar`: Character string with the name of the phase variable. Defaults to the attributes in the scdf file.
- `mvar`: Character string with the name of the measurement time variable. Defaults to the attributes in the scdf file.
- `method`: Specifies the method for outlier identification. Set `method = "MAD"` for mean average deviation, `method = "SD"` for standard deviations, `method = "CI"` for confidence intervals, `method = "Cook"` for Cook’s Distance based on the Piece-wise Linear Regression Model.
- `criteria`: Specifies the criteria for outlier identification. Based on the `method` setting.
Details

For method = "SD", criteria = 2 would refer to two standard deviations. For method = "MAD", criteria = 3.5 would refer to 3.5 times the mean average deviation. For method = "CI", criteria = 0.99 would refer to a 99 percent confidence interval. For method = "cook", criteria = "4/n" would refer to a Cook’s Distance greater than 4/n.

Value

data A single-case data frame with substituted outliers.
dropped.n A list with the number of dropped data points for each single-case.
dropped.mt A list with the measurement-times of dropped data points for each single-case (values are based on the mt variable of each single-case data frame).
sd.matrix A list with a matrix for each case with values for the upper and lower boundaries based on the standard deviation.
ci.matrix A list with a matrix for each single-case with values for the upper and lower boundaries based on the confidence interval.
cook A list of Cook’s Distances for each measurement of each single-case.
criteria Criteria used for outlier analysis.
N Number of single-cases.
case.names Case identifier.

Author(s)

Juergen Wilbert

See Also

Other data manipulation functions: add_l2(), as.data.frame.scdf(), as_scdf(), fill_missing(), moving_median(), ranks(), scdf(), select_cases(), set_vars(), shift(), smooth_cases(), standardize(), truncate_phase()

Examples

```r
## Identify outliers using 1.5 standard deviations as criterion
susanne <- random_scdf(level = 1.0)
res_outlier <- outlier(susanne, method = "SD", criteria = 1.5)
plot(susanne, marks = res_outlier)

## Identify outliers in the original data from Grosche (2011)
## using Cook’s Distance greater than 4/n as criterion
res_outlier <- outlier(Grosche2011, method = "Cook", criteria = "4/n")
plot(Grosche2011, marks = res_outlier)
```
Description

The `overlap` function provides the most common overlap indices for single-case data and some additional statistics.

Usage

```r
overlap(data, dvar, pvar, mvar, decreasing = FALSE, phases = c(1, 2))
```

Arguments

- `data`: A single-case data frame. See `scdf()` to learn about this format.
- `dvar`: Character string with the name of the dependent variable. Defaults to the attributes in the scdf file.
- `pvar`: Character string with the name of the phase variable. Defaults to the attributes in the scdf file.
- `mvar`: Character string with the name of the measurement time variable. Defaults to the attributes in the scdf file.
- `decreasing`: If you expect data to be lower in the B phase, set `decreasing = TRUE`. Default is `decreasing = FALSE`.
- `phases`: A vector of two characters or numbers indicating the two phases that should be compared. E.g., `phases = c("A", "B")` or `phases = c(2, 4)` for comparing the second to the fourth phase. Phases could be combined by providing a list with two elements. E.g., `phases = list(A = c(1, 3), B = c(2, 4))` will compare phases 1 and 3 (as A) against 2 and 4 (as B). Default is `phases = c(1, 2)`.

Details

See corresponding functions of PND, PEM, PET, NAP, PAND for calculation. Tau_U(A) reports "A vs. B - Trend A" whereas Tau_U(BA) reports "A vs. B + Trend B - Trend A". Base_Tau is baseline corrected tau (correction applied when autocorrelation in phase A is significant). Diff_mean is the mean difference. Diff_trend is the difference in the regression estimation of the dependent variable on measurement-time (`x ~ mt`) for each phase. SMD is the mean difference divided by the standard deviation of phase A. Hedges_g is the mean difference divided by the pooled standard deviation:

\[
\sqrt{\frac{(n_A-1)s_A^2 + (n_B-1)s_B^2}{n_A+n_B-2}}
\]

with a hedges correction applied: \(Hedges_g \times (1 - \frac{3}{4n-9})\).

Value

- `overlap`: A data frame consisting of the following indices for each single-case for all cases: PND, PEM, PET, NAP, PAND, IRD, Tau-U (A vs. B - Trend A), Diff_mean, Diff_trend, SMD, Hedges-g.
pand

phases.B  Selection for B phase.
design    Phase design.

Author(s)
Juergen Wilbert

See Also
Other overlap functions: cdc(), nap(), pand(), pem(), pet(), pnd(), tau_u()

Examples

## Display overlap indices for one single-case
overlap(Huitema2000, decreasing = TRUE)

## Display overlap indices for six single-cases
overlap(GruenkeWilbert2014)

## Combining phases for analysing designs with more than two phases
overlap(exampleA1B1A2B2, phases = list(c("A1", "A2"), c("B1", "B2")))

pand  Percentage of all non-overlapping data

Description
The pand() function calculates the percentage of all non-overlapping data (PAND; Parker, Hagan-Burke, & Vannest, 2007), an index to quantify a level increase (or decrease) in performance after the onset of an intervention.

Usage

pand(
  data,
  dvar,
  pvar,
  decreasing = FALSE,
  phases = c(1, 2),
  method = c("sort", "minimum")
)

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

## S3 method for class 'sc_pand'
```
export(
  object,
  caption = NA,
  footnote = NA,
  filename = NA,
  kable_styling_options = list(),
  kable_options = list(),
  round = 1,
  ...
)
```

**Arguments**

- **data**
  A single-case data frame. See `scdf()` to learn about this format.

- **dvar**
  Character string with the name of the dependent variable. Defaults to the attributes in the scdf file.

- **pvar**
  Character string with the name of the phase variable. Defaults to the attributes in the scdf file.

- **decreasing**
  If you expect data to be lower in the B phase, set `decreasing = TRUE`. Default is `decreasing = FALSE`.

- **phases**
  A vector of two characters or numbers indicating the two phases that should be compared. E.g., `phases = c("A","C")` or `phases = c(2,4)` for comparing the second to the fourth phase. Phases could be combined by providing a list with two elements. E.g., `phases = list(A = c(1,3), B = c(2,4))` will compare phases 1 and 3 (as A) against 2 and 4 (as B). Default is `phases = c(1,2)`.

- **method**
  Either "sort" or "minimum". See details.

- **x**
  An object returned by `pand()`

- **...**
  Further arguments passed to the function.

- **object**
  An scdf or an object exported from a scan function.

- **caption**
  Character string with table caption. If left NA (default) a caption will be created based on the exported object.

- **footnote**
  Character string with table footnote. If left NA (default) a footnote will be created based on the exported object.

- **filename**
  String containing the file name. If a filename is given the output will be written to that file.

- **kable_styling_options**
  list with arguments passed to the kable_styling function.

- **kable_options**
  list with arguments passed to the kable function.

- **round**
  Integer passed to the digits argument internally used to round values.

**Details**

PAND was proposed by Parker, Hagan-Burke, and Vannest in 2007. The authors emphasize that PAND is designed for application in a multiple case design with a substantial number of measurements, technically at least 20 to 25, but preferably 60 or more. PAND is defined as 100% minus
the percentage of data points that need to be removed from either phase in order to ensure nonoverlap between the phases. Several approaches have been suggested to calculate PAND, leading to potentially different outcomes. In their 2007 paper, Parker and colleagues present an algorithm for computing PAND. The algorithm involves sorting the scores of a time series, including the associated phases, and comparing the resulting phase order with the original phase order using a contingency table. To account for ties, the algorithm includes a randomization process where ties are randomly assigned to one of the two phases. Consequently, executing the algorithm multiple times could yield different results. It is important to note that this algorithm does not produce the same results as the PAND definition provided earlier in the same paper. However, it offers the advantage of allowing the calculation of an effect size measure \( \phi \), and the application of statistical tests for frequency distributions. Pustejovsky (2019) presented a mathematical formulation of Parker’s original definition for comparing two phases of a single case:

\[
PAND = \frac{1}{m+n} \max \{(i+j)I(y^A_i < y^B_{n+1-j})\}
\]

This formulation provides accurate results for PAND, but the original definition has the drawback of an unknown distribution under the null hypothesis, making a statistical test difficult. The \( \text{pand()} \) function enables the calculation of PAND using both methods. The first approach (\text{method = "sort"}) follows the algorithm described above, with the exclusion of randomization before sorting to avoid ambiguity. It calculates a \( \phi \) measure and provides the results of a chi-squared test and a Fisher exact test. The second approach (\text{method = "minimum"}) applies the aforementioned formula. The code of this function is based on the code of the \text{SingleCaseES} package (function \text{calc_PAND}). For a multiple case design, overlaps are calculated for each case, summed, and then divided by the total number of measurements. No statistical test is conducted for this method.

**Value**

- **pand**: Percentage of all non-overlapping data.
- **phi**: Effect size \( \phi \) based on expected and observed values.
- **perc_overlap**: Percentage of overlapping data points.
- **overlaps**: Number of overlapping data points.
- **n**: Number of data points.
- **N**: Number of cases.
- **n_a**: Number of data points in phase A.
- **n_b**: Number of data points in phase B.
- **matrix**: 2x2 frequency matrix of phase A and B comparisons.
- **matrix_counts**: 2x2 counts matrix of phase A and B comparisons.
- **chi_test**: A Chi-squared analysis of expected and observed data (\text{chisq.test()}).
- **fisher_test**: A Fisher exact test analysis of expected and observed data (\text{fisher.test()}).

**Functions**

- \text{print(sc_pand)}: Print results
- \text{export(sc_pand)}: Export results as html table (see \text{export()})
Author(s)
Juergen Wilbert

References

See Also
Other overlap functions: cdc(), nap(), overlap(), pem(), pet(), pnd(), tau_u()

Examples

```r
## REplication of the Parker et al. 2007 example
pand(Parker2007)

## Calculate the PAND with an expected decrease of phase B scores
Cubs <- scdf(c(20,22,24,17,21,13,10,9,20,9,18), B_start = 5)
pand(Cubs, decreasing = TRUE)
```

Description

The `pem` function returns the percentage of phase B data exceeding the phase A median. Additionally, a chi square test against a 50/50 distribution is computed. Different measures of central tendency can be addressed for alternative analyses.

Usage

```r
pem(
  data,
  dvar,
  pvar,
  decreasing = FALSE,
  binom.test = TRUE,
  chi.test = FALSE,
  FUN = median,
)```
phases = c(1, 2),
  ...
)

Arguments

data A single-case data frame. See scdf() to learn about this format.
dvar Character string with the name of the dependent variable. Defaults to the attributes in the scdf file.
pvar Character string with the name of the phase variable. Defaults to the attributes in the scdf file.
decreasing If you expect data to be lower in the B phase, set decreasing = TRUE. Default is decreasing = FALSE.
binom.test Computes a binomial test for a 50/50 distribution. Default is binom.test = TRUE.
chi.test Computes a Chi-square test. The default setting chi.test = FALSE skips the Chi-square test.
FUN Data points are compared with the phase A median. Use this argument to implement alternative measures of central tendency. Default is FUN = median
phases A vector of two characters or numbers indicating the two phases that should be compared. E.g., phases = c("A","C") or phases = c(2,4) for comparing the second to the fourth phase. Phases could be combined by providing a list with two elements. E.g., phases = list(A = c(1,3), B = c(2,4)) will compare phases 1 and 3 (as A) against 2 and 4 (as B). Default is phases = c(1,2).
...
Additional arguments for the FUN parameter (e.g. FUN = mean, trim = 0.1 will use the 10 percent trimmed arithmetic mean instead of the median for comparisons). The function must take a vector of numeric values and the na.rm argument and return a numeric value.

Author(s)

Juergen Wilbert

See Also

Other overlap functions: cdc(), nap(), overlap(), pand(), pet(), pnd(), tau_u()

Examples

## Calculate the PEM including the Binomial and Chi-square tests for a single-case
dat <- random_scdf(5, level = 0.5)
pem(dat, chi.test = TRUE)
The `pet` function returns the percentage of Phase B data points that exceed the prediction based on the Phase A trend. A binomial test against a 50/50 distribution is calculated. It also calculates the percentage of Phase B data points that exceed the upper (or lower) 95 percent confidence interval of the predicted progression.

Usage

```r
pet(data, dvar, pvar, mvar, ci = 0.95, decreasing = FALSE, phases = c(1, 2))
```

Arguments

data
- A single-case data frame. See `scdf()` to learn about this format.
dvar
- Character string with the name of the dependent variable. Defaults to the attributes in the scdf file.
pvar
- Character string with the name of the phase variable. Defaults to the attributes in the scdf file.
mvar
- Character string with the name of the measurement time variable. Defaults to the attributes in the scdf file.
ci
- Width of the confidence interval. Default is `ci = 0.95`.
decreasing
- If you expect data to be lower in the B phase, set `decreasing = TRUE`. Default is `decreasing = FALSE`.
phases
- A vector of two characters or numbers indicating the two phases that should be compared. E.g., `phases = c("A", "C")` or `phases = c(2, 4)` for comparing the second to the fourth phase. Phases could be combined by providing a list with two elements. E.g., `phases = list(A = c(1, 3), B = c(2, 4))` will compare phases 1 and 3 (as A) against 2 and 4 (as B). Default is `phases = c(1, 2)`.

Value

- **PET** Percent exceeding the trend.
- **PET.ci** Percent exceeding the upper / lower 95\% Test.
- **ci.percent** Width of confidence interval in percent.
- **se.factors** Standard error.
- **N** Number of cases.
- **decreasing** Logical argument from function call (see Arguments above).
- **case.names** Assigned name of single-case.
- **phases** -
Author(s)
Juergen Wilbert

See Also
Other overlap functions: cdc(), nap(), overlap(), pand(), pem(), pnd(), tau_u()

Examples

# Calculate the PET and use a 99%-CI for the additional calculation
# create random example data
design <- design(n = 5, slope = 0.2)
dat <- random_scdf(design, seed = 23)
pet(dat, ci = .99)

plm

---

Piecewise linear model / piecewise regression

Description
The plm function computes a piecewise regression model (see Huitema & McKean, 2000).

Usage

plm(
data,
dvar,
pvar,
mvar,
AR = 0,
model = c("W", "H-M", "B&L-B", "JW"),
family = "gaussian",
trend = TRUE,
level = TRUE,
slope = TRUE,
contrast = c("first", "preceding"),
contrast_level = c(NA, "first", "preceding"),
contrast_slope = c(NA, "first", "preceding"),
formula = NULL,
update = NULL,
na.action = na.omit,
r_squared = TRUE,
var_trials = NULL,
dvar_percentage = FALSE,
...
)

Arguments

data  A single-case data frame. See \code{scdf()} to learn about this format.
dvar Character string with the name of the dependent variable. Defaults to the attributes in the scdf file.
pvar Character string with the name of the phase variable. Defaults to the attributes in the scdf file.
mvar Character string with the name of the measurement time variable. Defaults to the attributes in the scdf file.
AR Maximal lag of autoregression. Modeled based on the Autoregressive-Moving Average (ARMA) function. When AR is set, the family argument must be set to \code{family = "gaussian"}.
model Model used for calculating the dummy parameters (see Huijtema & McKean, 2000). Default is \code{model = "W"}. Possible values are: \code{"B&L-B"}, \code{"H-M"}, \code{"W"}, and deprecated \code{"JW"}.
family Set the distribution family. Defaults to a gaussian distribution. See the \code{family} function for more details.
trend A logical indicating if a trend parameters is included in the model.
level A logical indicating if a level parameters is included in the model.
slope A logical indicating if a slope parameters is included in the model.
contrast Sets contrast_level and contrast_slope. Either \code{"first"}, \code{"preceding"} or a contrast matrix.
contrast_level Either \code{"first"}, \code{"preceding"} or a contrast matrix. If NA contrast_level is a copy of contrast.
contrast_slope Either \code{"first"}, \code{"preceding"} or a contrast matrix. If NA contrast_level is a copy of contrast.
formula Defaults to the standard piecewise regression model. The parameter phase followed by the phase name (e.g., phaseB) indicates the level effect of the corresponding phase. The parameter 'inter' followed by the phase name (e.g., interB) adresses the slope effect based on the method provide in the model argument (e.g., \code{"B&L-B"}). The formula can be changed for example to include further variables into the regression model.
update An easier way to change the regression formula (e.g., \code{~ . + newvariable}).
na.action Defines how to deal with missing values.
r_squared Logical. If TRUE, delta \code{r_squares} will be calculated for each predictor.
var_trials Name of the variable containing the number of trials (only for binomial regressions). If a single integer is provided this is considered to be a the constant number of trials across all measurements.
dvar_percentage Only for binomial distribution. If set TRUE, the dependent variable is assumed to represent proportions \([0, 1]\). Otherwise dvar is assumed to represent counts.

... Further arguments passed to the \code{glm} function.
Value

- `formula`: plm formula. Useful if you want to use the update or formula argument and you don’t know the names of the parameters.
- `model`: Character string from function call (see Arguments above).
- `F.test`: F-test values of modelfit.
- `r.squares`: Explained variance R squared for each model parameter.
- `ar`: Autoregression lag from function call (see Arguments above).
- `family`: Distribution family from function call (see Arguments above).
- `full.model`: Full regression model list from the gls or glm function.

Author(s)

Juergen Wilbert

References


See Also

Other regression functions: `autocorr()`, `corrected_tau()`, `hplm()`, `mplm()`, `trend()`

Examples

```r
## Compute a piecewise regression model for a random single-case
set.seed(123)
AB <- design(
    phase_design = list(A = 10, B = 20),
    level = list(A = 0, B = 1), slope = list(A = 0, B = 0.05),
    trend = 0.05
)
dat <- random_scdf(design = AB)
plm(dat, AR = 3)

## Another example with a more complex design
A1B1A2B2 <- design(
    phase_design = list(A1 = 15, B1 = 20, A2 = 15, B2 = 20),
    level = list(A1 = 0, B1 = 1, A2 = -1, B2 = 1),
    slope = list(A1 = 0, B1 = 0.0, A2 = 0, B2 = 0.0),
    trend = 0.0
)
dat <- random_scdf(design = A1B1A2B2, seed = 123)
plm(dat, contrast = "preceding")
```
## no slope effects were found. Therefore, you might want to the drop slope estimation:
plm(dat, slope = FALSE, contrast = "preceding")

## and now drop the trend estimation as well
plm(dat, slope = FALSE, trend = FALSE, contrast = "preceding")

## A poisson regression
example_A24 %>%
  plm(family = "poisson")

## A binomial regression (frequencies as dependent variable)
plm(exampleAB_score$Christiano, family = "binomial", var_trials = "trials")

## A binomial regression (percentage as dependent variable)
exampleAB_score$Christiano %>%
  transform(percentage = values/trials) %>%
  set_dvar("percentage") %>%
  plm(family = "binomial", var_trials = "trials", dvar_percentage = TRUE)

---

**plot.scdf**

*Plot single-case data*

### Description

This function provides a plot of a single-case or multiple single-cases.

### Usage

```
## S3 method for class 'scdf'
plot(...)

plotSC(
  data,
  dvar,
  pvar,
  mvar,
  ylim = NULL,
  xlim = NULL,
  xinc = 1,
  lines = NULL,
  marks = NULL,
  phase.names = NULL,
  xlab = NULL,
  ylab = NULL,
  main = "",
  case.names = NULL,
  style =getOption("scan.plot.style"),
  ...
)
```
Arguments

... Further arguments passed to the plot command.

data A single-case data frame. See scdf() to learn about this format.

dvar Character string with the name of the dependent variable. Defaults to the attributes in the scdf file.

pvar Character string with the name of the phase variable. Defaults to the attributes in the scdf file.

mvar Character string with the name of the measurement time variable. Defaults to the attributes in the scdf file.

ylim Lower and upper limits of the y-axis (e.g., ylim = c(0, 20) sets the y-axis to a scale from 0 to 20). With multiple single-cases you can use ylim = c(0, NA) to scale the y-axis from 0 to the maximum of each case. ylim is not set by default, which makes scan set a proper scale based on the given data.

xlim Lower and upper limits of the x-axis (e.g., xlim = c(0, 20) sets the x-axis to a scale from 0 to 20). With multiple single-cases you can use ylim = c(0, NA) to scale the x-axis from 0 to the maximum of each case. xlim is not set by default, which makes scan set a proper scale based on the given data.

xinc An integer. Increment of the x-axis. 1: each mt value will be printed, 2: every other value, 3: every third values etc.

tlines A list defining one or multiple lines or curves to be plotted. The argument is passed as a list (e.g., lines = list(type = "median", trim = 0.2)) adds a 20\% line. For multiple lines, provide a list element for each line (e.g., lines = list(list(type = "median", col = "red"), list(type = "mean", col = "blue"))). Possible lines are:

• "median" Separate lines for phase A and B medians.
• "mean" Separate lines for phase A and B means. By default it is 10\% lines = list(type = "mean", trim = 0.2) draws a 20\% line. For multiple lines, provide a list element for each line (e.g., lines = list(list(type = "median", col = "red"), list(type = "mean", col = "blue"))). Possible lines are:

• "trend" Separate lines for phase A and B trends.
• "trendA" OLS trend line for phase A, extrapolated throughout phase B.
• "trendA_bisplit" Split middle (bi-split) trend line for phase A, extrapolated throughout phase B.
• "trendA_trisplit" Tukey tri-split trend line for phase A, extrapolated throughout phase B.
• "maxA/minA" Line at the level of the highest or lowest phase A score.
• "medianA" Line at the phase A median score.
• "meanA" Line at the phase A 10\% using the additional argument (e.g., lines = list(type = "meanA", trim = 0.2)).
• "plm" Regression lines for piecewise linear regression model.
• "plm.ar" Regression lines for piecewise autoregression model. The lag is specified like this: lines = list(type = "plm.ar", ar = 2). Default lag is set to 2.
• "movingMean" Draws a moving mean curve, with a specified lag: lines = list(type = "movingMean", lag = 2). Default is a lag 1 curve.
- "movingMedian" Draws a moving median curve, with a specified lag: `lines = list(type = "movingMedian", lag = 3)`. Default is a lag 1 curve.
- "loreg" Draws a non-parametric local regression line. The proportion of data influencing each data point can be specified using `lines = list(type = "loreg", f = 0.66)`. The default is 0.5.
- "lty" Use this argument to define the line type. Examples are: "solid", "dashed", "dotted".
- "lwd" Use this argument to define the line’s thickness, e.g., `lwd = 4`.
- "col" Use this argument to define the line’s color, e.g., `col = "red"`.

marks
A list of parameters defining markings of certain data points.
- "positions" A vector or a list of vectors indicating measurement-times to be highlighted. In case of a vector, the marked measurement-times are the same for all plotted cases. In case of a list of vectors, marks are set differently for each case. The list must have the same length as there are cases in the data file.
- "col" Color of the marks.
- "cex" Size of the marks.

Use for example `marks = list(positions = c(1, 8, 15), col = "red", cex = 3)` to make the MTs one, eight and 18 appear big and red.

phase.names
By default phases are labeled based on the levels of the phase variable. Use this argument to specify different labels: `phase.names = c("Baseline","Intervention")`.

xlab
The label of the x-axis. Default is `xlab = "Measurement time"`.

ylab
The labels of the y-axis. Default is `ylab = "Score"`.

main
Main title of the plot.

case.names
Case names. If not provided, names are taken from the scdf. Set `case.names = ""` if you don’t like to include case names.

style
Either a character with the name of a pre-implemented style or a style object. See `style_plot` to learn about this format.

Value
Returns a plot of one or multiple single-cases.

Author(s)
Juergen Wilbert

See Also
`style_plot, describeSC, overlapSC`
Examples

```r
## Request the default plot of the data from Borckhardt (2014)
plot(Borckhardt2014)

## Plot the three cases from Grosche (2011) and visualize the phase A trend
plot(Grosche2011, style = "grid", lines = "trendA")

## Request the local regression line for Georg from that data set and customize the plot
plot(Grosche2011$Georg, style = "sienna", ylim = c(0,NA),
     xlab = "Training session", ylab = "Words per minute",
     phase.names = c("Baseline", "Intervention"), xinc = 5,
     lines = list(type = "loreg", f = 0.2, lty = "solid", col = "black", lwd = 3))

## Plot a random MBO over three cases and mark interesting MTs
dat <- random_scdf(design = design(3))
plot(dat, marks = list(positions = list(c(2,4,5),c(1,2,3),c(7,8,9)), col = "blue",
                      cex = 1.4), style = c("grid", "annotate", "tiny"))
```

---

**plot_rand**  
*Plot random distribution*

Description

This function takes the return of the rand_test function and creates a histogram with the distribution of the rand sample statistics.

Usage

```r
plot_rand(
  object,
  xlab = NA,
  ylab = "Frequency",
  title = "Random distribution",
  text_observed = "observed",
  color = "lightgrey",
  ...)
```

Arguments

- `object`: Object returned from the rand_test() function
- `xlab`: Label for the x-axis.
- `ylab`: Label for the y-axis.
- `title`: Plot title.
- `text_observed`: Text for marking the number of observed statistic.
color Bar color.

Further arguments passed to the plot function.

---

**pnd**

*Percentage of non-overlapping data*

**Description**

This function returns the percentage of non-overlapping data. Due to its error-proneness the PND should not be used, but nap or pand instead (see Parker & Vannest, 2009).

**Usage**

```r
pnd(data, dvar, pvar, decreasing = FALSE, phases = c(1, 2))
```

**Arguments**

- **data**: A single-case data frame. See `scdf()` to learn about this format.
- **dvar**: Character string with the name of the dependent variable. Defaults to the attributes in the scdf file.
- **pvar**: Character string with the name of the phase variable. Defaults to the attributes in the scdf file.
- **decreasing**: If you expect data to be lower in the B phase, set `decreasing = TRUE`. Default is `decreasing = FALSE`.
- **phases**: A vector of two characters or numbers indicating the two phases that should be compared. E.g., `phases = c("A","C")` or `phases = c(2, 4)` for comparing the second to the fourth phase. Phases could be combined by providing a list with two elements. E.g., `phases = list(A = c(1,3), B = c(2,4))` will compare phases 1 and 3 (as A) against 2 and 4 (as B). Default is `phases = c(1,2)`.

**Value**

`PND` Percentage of non-overlapping data.

**Author(s)**

Juergen Wilbert

**See Also**

Other overlap functions: `cdc()`, `nap()`, `overlap()`, `pand()`, `pem()`, `pet()`, `tau_u()`

**Examples**

```r
## Calculate the PND for multiple single-case data
pnd(GruenkeWilbert2014)
```
Empirical power analysis for single-case data

Description

Conducts a Monte-Carlo study on the test-power and alpha-error probability of a statistical function.

Usage

```r
power_test(
  design,
  method = c("plm_level", "rand", "tauU"),
  effect = "level",
  n_sim = 100,
  design_is_one_study = TRUE,
  alpha_test = TRUE,
  power_test = TRUE,
  binom_test = FALSE,
  binom_test_alpha = FALSE,
  binom_test_power = FALSE,
  binom_test_correct = FALSE,
  ci = FALSE,
  alpha_level = 0.05
)
```

Arguments

- **design**: An object returned from the `design` function.
- **method**: A (named) list that defines the methods the power analysis is based on. Each element can contain a function (that takes an scdf file and returns a p value) or a character string (the name of predefined functions). default: `method = list("plm_level", "rand", "tauU")` computes a power analysis based on `tau_u()`, `rand_test()` and `plm()` analyses. (Further predefined functions are: "plm_slope", "plm_poisson_level", "plm_poisson_slope", "hplm_level", "hplm_slope", "base_tau".
- **effect**: Either "level" or "slope". The respective effect of the provided design is set to 0 when computing the alpha-error proportion.
- **n_sim**: Number of sample studies created for the the Monte-Carlo study. Default is `n = 100`. Ignored if `design_is_one_study = FALSE`.
- **design_is_one_study**: If TRUE, the design is assumed to define all cases of one study that is repeatedly randomly created `n_sim` times. If false, the design is assumed to contain all cases from which a random sample is generated. This is useful for very specific complex simulation studies.
- **alpha_test**: Logical. If TRUE, alpha error is calculated.
- **power_test**: Logical. If TRUE, power is calculated.
binom_test  Shortcut. When set TRUE, binom_test_power is set to 0.80, binom_test_alpha is set to 0.05, and binom_test_correct is set to 0.875.

binom_test_alpha  Either FALSE or a value. If a value is provided, a binomial test is calculated testing if the alpha error proportion is less than the provided value.

binom_test_power  Either FALSE or a value. If a value is provided, a binomial test is calculated testing if the power is greater than the provided value.

binom_test_correct  Either FALSE or a value. If a value is provided, a binomial test is calculated testing if the correct proportion is greater than the provided value.

ci  Either FALSE or a value. If a value is provided, confidence intervals at the provided level are calculated for power, alpha error, and correct proportions.

alpha_level  Alpha level used to calculate the proportion of significant tests. Default is alpha_level = 0.05.

Details

Based on a design() object, a large number of single-cases are generated and re-analyzed with a provided statistical function. The proportion of significant analyzes is the test power. In a second step, a specified effect of the design object is set to 0 and again single-cases are generated and reanalyzed. The proportion of significant analyzes is the alpha error probability.

Author(s)

Juergen Wilbert

See Also

random_scdf(), design()

Examples

```r
## Assume you want to conduct a single-case study with 15 measurements
## (phases: A = 6 and B = 9) using a highly reliable test and
## an expected level effect of d = 1.4.
## A (strong) trend effect is trend = 0.05. What is the power?
## (Note: n_sims is set to 10. Set n_sims to 1000 for a serious calculation.)
design <- design(
  n = 1, phase_design = list(A = 6, B = 9),
  rtt = 0.8, level = 1.4, trend = 0.05
)
power_test(design, n_sim = 10)

## Would you achieve higher power by setting up a MBD with three cases?
design <- design(
  n = 3, phase_design = list(A = 6, B = 9),
  rtt = 0.8, level = 1.4, trend = 0.05
)
```
power_test(design, n_sim=10, method=list("hplm_level", "rand", "tauU_meta"))

print.scdf

## S3 method for class 'scdf'
print(
x,
cases =getOption("scan.print.cases"),
rows =getOption("scan.print.rows"),
cols =getOption("scan.print.cols"),
long =getOption("scan.print.long"),
digits =getOption("scan.print.digits"),
...
)

Arguments

x An scdf object

cases Number of cases to be printed. "fit" fits the number to the current screen width.

rows Number of rows to be printed.

cols Columns to be printed. "Main" only prints the dependent, measurement-time and phase variable.

long Logical. If TRUE cases are printed in one by a time.

digits Number of digits.

Details

Print options for scdf objects could be set globally: option(scan.print.cases = "all"), option(scan.print.rows = 10), option(scan.print.cols = "main"), option(scan.print.long = TRUE), option(scan.print.digits = 0), option(scan.print.scdf.name = FALSE)
random_scdf  

Single-case data generator

Description

The random_scdf function generates random single-case data frames for monte-carlo studies and demonstration purposes. design is used to set up a design matrix with all parameters needed for the random_scdf function.

Usage

random_scdf(design = NULL, round = NA, random_names = FALSE, seed = NULL, ...)

Arguments

design  
A design matrix which is created by design and specifies all parameters.

round  
Rounds the scores to the defined decimal. To round to the second decimal, set round = 2.

random_names  
Is FALSE by default. If set random_names = TRUE cases are assigned random first names. If set "neutral", "male" or "female" only gender neutral, male, or female names are chosen. The names are drawn from the 2,000 most popular names for newborns in 2012 in the U.S. (1,000 male and 1,000 female names).

seed  
A seed number for the random generator.

...  
arguments that are directly passed to the design function for a more concise coding.

Value

A single-case data frame. See scdf to learn about this format.

Author(s)

Juergen Wibert

Examples

```r
## Create random single-case data and inspect it
design <- design(
    n = 3, rtt = 0.75, slope = 0.1, extreme_prop = 0.1,
    missing_prop = 0.1
)
dat <- random_scdf(design, round = 1, random_names = TRUE, seed = 123)
describe(dat)

## And now have a look at poisson-distributed data
design <- design(
```
The `rand_test` function computes a randomization test for single or multiple baseline single-case data. The function is based on an algorithm from the SCRT package (Bulte & Onghena, 2009, 2012), but rewritten and extended for the use in AB designs.

### Usage

```r
rand_test(data, dvar, pvar, statistic = c("Mean B-A", "Mean A-B", "Median B-A", "Median A-B", "Mean |A-B|", "Median |A-B|"), number = 500, complete = FALSE, limit = 5, startpoints = NA, exclude.equal = FALSE, phases = c(1, 2), graph = FALSE, output = NULL, seed = NULL)
```

### Arguments

- **data**: A single-case data frame. See `scdf()` to learn about this format.
- **dvar**: Character string with the name of the dependent variable. Defaults to the attributes in the scdf file.
- **pvar**: Character string with the name of the phase variable. Defaults to the attributes in the scdf file.
- **statistic**: Defines the statistic on which the comparison of phases A and B is based on. Default setting is `statistic = "Mean B-A"`. The following comparisons are possible:
  - "Mean A-B": Uses the difference between the mean of phase A and the mean of phase B. This is appropriate if a decrease of scores was expected for phase B.
• "Mean B-A": Uses the difference between the mean of phase B and the mean of phase A. This is appropriate if an increase of scores was expected for phase B.
• "Mean |A-B|": Uses the absolute value of the difference between the means of phases A and B.
• "Median A-B": The same as "Mean A-B", but based on the median.
• "Median B-A": The same as "Mean B-A", but based on the median.

number
Sample size of the randomization distribution. The exactness of the p-value cannot exceed \(1/\text{number}\) (i.e., \(\text{number} = 100\) results in p-values with an exactness of one percent). Default is \(\text{number} = 500\). For faster processing use \(\text{number} = 100\). For more precise p-values set \(\text{number} = 1000\).

complete
If TRUE, the distribution is based on a complete permutation of all possible starting combinations. This setting overwrites the number Argument. The default setting is FALSE.

limit
Minimal number of data points per phase in the sample. The first number refers to the A-phase and the second to the B-phase (e.g., \(\text{limit} = c(5,3)\)). If only one number is given, this number is applied to both phases. Default is \(\text{limit} = 5\).

startpoints
Alternative to the \(\text{limit}\)-parameter \(\text{startpoints}\) exactly defines the possible start points of phase B (e.g., \(\text{startpoints} = 4:9\) restricts the phase B start points to measurements 4 to 9. \(\text{startpoints}\) overruns the \(\text{limit}\)-parameter.

exclude.equal
If set to \(\text{exclude.equal} = \text{FALSE}\), which is the default, random distribution values equal to the observed distribution are counted as null-hypothesis conform. That is, they decrease the probability of rejecting the null-hypothesis (increase the p-value). \(\text{exclude.equal}\) should be set to \(\text{TRUE}\) if you analyse one single-case design (not a multiple baseline data set) to reach a sufficient power. But be aware, that it increases the chance of an alpha-error.

phases
A vector of two characters or numbers indicating the two phases that should be compared. E.g., \(\text{phases} = c("A","C")\) or \(\text{phases} = c(2,4)\) for comparing the second to the fourth phase. Phases could be combined by providing a list with two elements. E.g., \(\text{phases} = \text{list}(A = c(1,3), B = c(2,4))\) will compare phases 1 and 3 (as A) against 2 and 4 (as B). Default is \(\text{phases} = c(1,2)\).

graph
If \(\text{graph} = \text{TRUE}\), a histogram of the resulting distribution is plotted. It is \(\text{FALSE}\) by default. \textit{Note: use the more versatile plot_rand() function instead.}

output
(deprecated and not implemented)

seed
A seed number for the random generator.

Value

\(\text{statistic}\)
Character string from function call (see Arguments above).

\(\text{N}\)
Number of single-cases.

\(\text{n1}\)
Number of data points in phase A.

\(\text{n2}\)
Number of data points in phase B.

\(\text{limit}\)
Numeric from function call (see Arguments above).
startpoints  A vector defining the start points passed from the function call (see Arguments above).

p.value   P-value of the randomization test for the given data.

number   Sample size of randomization distribution from function call (see Arguments above).

complete   Logical argument from function call (see Arguments above).

observed.statistic   Test statistic observed for the given single-case data. (see statistic in the Arguments above.)

Z   Z-value of observed test statistic.

p.z.single   Probability of z-value.

distribution   Test statistic distribution from randomized data sets.

possible.combinations   Number of possible combinations under the given restrictions.

auto.corrected.number   TRUE indicates that a corrected number of combinations was used. This happens, if the number of possible combinations (under the given restrictions) undergrads the requested number of combinations.

Author(s)

Juergen Wilbert

References


Examples

## Compute a randomization test on the first case of the byHeart2011 data and include a graph
rand_test(byHeart2011[1], statistic = "Median B-A", graph = TRUE, seed = 123)

## Compute a randomization test on the Grosche2011 data using complete permutation
rand_test(Grosche2011, statistic = "Median B-A", complete = TRUE, limit = 4, seed = 123)
**rci**

**Reliable change index**

**Description**

The `rci()` function computes three indices of reliable change (Wise, 2004) and corresponding descriptive statistics.

**Usage**

```r
rci(data, dvar, pvar, rel, ci = 0.95, graph = FALSE, phases = c(1, 2))
```

**Arguments**

- `data`: A single-case data frame. See `scdf()` to learn about this format.
- `dvar`: Character string with the name of the dependent variable. Defaults to the attributes in the scdf file.
- `pvar`: Character string with the name of the phase variable. Defaults to the attributes in the scdf file.
- `rel`: Reliability of the measure, used to compute the standard error.
- `ci`: Width of confidence interval as a decimal. Default is `ci = 0.95` applying a 95 percent confidence interval.
- `graph`: If set TRUE, a box plot of phase A and B scores is displayed. `graph = FALSE` by default.
- `phases`: A vector of two characters or numbers indicating the two phases that should be compared. E.g., `phases = c("A","C")` or `phases = c(2,4)` for comparing the second to the fourth phase. Phases could be combined by providing a list with two elements. E.g., `phases = list(A = c(1,3), B = c(2,4))` will compare phases 1 and 3 (as A) against 2 and 4 (as B). Default is `phases = c(1,2)`.

**Author(s)**

Juergen Wilbert

**References**


Examples

```r
## Report the RCIs of the first case from the byHeart data and include a graph
ci(byHeart2011[1], graph = TRUE, rel = 0.8)
```

---

**read_scdf**

Load single-case data from files

**Description**

Use the `read_scdf` function to load single-case data csv, excel, or yaml files.

**Usage**

```r
read_scdf(
  file, 
  cvar = "case", 
  pvar = "phase", 
  dvar = "values", 
  mvar = "mt", 
  sort_cases = FALSE, 
  phase_names = NULL, 
  type = NA, 
  na = c("", "NA"), 
  sort.labels = NULL, 
  phase.names = NULL, 
  ...
)
```

**Arguments**

- **file**: Either a character string defining the file to be loaded (e.g. "SC_Ana.csv" (if left empty a dialog box for choosing will be opened) or a data.frame.
- **cvar**: Sets the variable name of the "case" variable. Defaults to "case".
- **pvar**: Sets the variable name of the "phase" variable. Defaults to "phase".
- **dvar**: Sets the variable name of the "values" variable. Defaults to "values".
- **mvar**: Sets the variable name of the "mt" variable. Defaults to "mt".
- **sort_cases**, **sort.labels**: If set TRUE, the resulting list is sorted by label names (alphabetically increasing).
- **phase_names**, **phase.names**: A character vector with phase names. Defaults to the phase names provided in the phase variable.
- **type**: Format of the file to be loaded. Either "csv", "xlsx", "xls", "excel", "yml" is possible. By default (NA) the type is extracted from the file extension.
sample_names

na Character vector of strings to interpret as missing values.

... Further arguments passed to the respective read function.

Value

Returns a single-case data frame. See scdf to learn about the format of these data frames.

Author(s)

Juergen Wilbert

See Also

read.table(), readRDS()

Other io-functions: convert(), write_scdf()

Examples

## Read SC-data from a file named "study1.csv" in your working directory
# study1 <- read_scdf("study1.csv")

## Read SC-data from a .csv-file with semicolon as field and comma as decimal separator
# study2 <- read_scdf("study2.csv", sep = ";", dec = ",")

## write_scdf and read_scdf
filename <- file.path(tempdir(), "test.csv")
write_scdf(exampleA1B1A2B2_zvt, filename)
dat <- read_scdf(filename, cvar = "case", pvar = "part", dvar = "zvt", mvar = "day")
res1 <- describe(exampleA1B1A2B2_zvt)$descriptives
res2 <- describe(dat)$descriptives
all.equal(res1,res2)

sample_names

<table>
<thead>
<tr>
<th>Samples random names</th>
</tr>
</thead>
</table>

Description

Samples random names

Usage

sample_names(n = 1, type = "neutral", seed = NULL)

Arguments

n Number of names

type "neutral", "male", "female", or "mixed"

seed A seed for the random number generator.
**Value**

A character vector with random names

**Examples**

```r
sample_names(3)
```

---

**scdf**

*Single case data frame*

**Description**

`scdf()` is the constructor for the scdf class. It stores single-case study data with one or more single-cases.

**Usage**

```r
cddf(
  values,
  B_start,
  mt,
  phase,
  phase_design = NULL,
  name = NULL,
  dvar = "values",
  pvar = "phase",
  mvar = "mt",
  ...
)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>values</code></td>
<td>A vector containing measurement values of the dependent variable.</td>
</tr>
<tr>
<td><code>B_start</code></td>
<td>The first measurement of phase B (simple coding if design is strictly AB).</td>
</tr>
<tr>
<td><code>mt</code></td>
<td>A vector defining measurement times. Default is <code>mt = (1,2,3,...,n)</code></td>
</tr>
<tr>
<td><code>phase</code></td>
<td>A vector defining phase assignments.</td>
</tr>
</tbody>
</table>
| `phase_design` | A vector defining the length and label of each phase. E.g., `phase_design = c(A1 = 10, B1 = 10, A2 = 10, B2 = 10)`.
| `name`      | A name for the case.                                                                                                                        |
| `dvar`      | Character string with the name of the dependent variable. Defaults to the attributes in the scdf file.                                    |
| `pvar`      | Character string with the name of the phase variable. Defaults to the attributes in the scdf file.                                         |
| `mvar`      | Character string with the name of the measurement time variable. Defaults to the attributes in the scdf file.                             |
| `...`       | Additional variables. E.g., `teacher = c(0,1,0,1,0,0,1), lesson = c(1,3,4,5,2,3)`                                                      |
Details

If the dependent variable is a named vector then the names are extracted to create a phase design (e.g., values = c(A = 2, 3, 5, 4, 3, B = 6, 5, 4, 3) will create an AB phase design with five and four measurements). An scdf contains several attributes: dvar The name of the dependent variable. phase The name of the phase variable. mt The name of the measurement time variable. author Information on the author of the data. info Further information on the data. E.g., a publication. dvar, phase, and mt are the defaults most of the scan function use. You can change the values of the attributes with the scdf_attr function (e.g., scdf_attr(exampleAB_add, "dvar") <- "depression" defines depression as the dependent variable. Please notice that all scan functions have arguments to define dvar, phase, and mt for a given analysis.

Value

Returns a single-case data frame scdf suitable for all functions of the scan package. Multiple data sets (e.g. from Multiple Baseline Designs) can be listed.

Author(s)

Juergen Wilbert

See Also

Other data manipulation functions: add_l2(), as.data.frame.scdf(), as_scdf(), fill_missing(), moving_median(), outlier(), ranks(), select_cases(), set_vars(), shift(), smooth_cases(), standardize(), truncate_phase()

Examples

```r
## Scores on a letter naming task were collected on eleven days in a row.
## The intervention started after the fifth measurement,
## so the first B phase measurement was 6 (B_start = 6).
##
klaas <- scdf(
  c(5, 7, 8, 5, 7, 12, 16, 18, 15, 14, 19),
  B_start = 6, name = "Klaas"
)
describe(klaas)

# Alternative coding 1:
klaas <- scdf(
  c(A = 5, 7, 8, 5, 7, B = 12, 16, 18, 15, 14, 19),
  name = "Klaas"
)

# Alternative coding 2:
klaas <- scdf(
  c(5, 7, 8, 5, 7, 12, 16, 18, 15, 14, 19),
  phase_design = c(A = 5, B = 6), name = "Klaas"
)

## Unfortunately in a similar study there were no data collected on
## select_cases

### Select a subset of cases

#### Description

Select a subset of cases

#### Usage

```r
select_cases(scdf, ...)
```

#### Arguments

- **scdf**
  
  A single-case data frame. See `scdf()` to learn about this format.

- **...**
  
  Selection criteria. Either numeric, objectnames, or as characters.
select_phases

Select and combine phases for overlap analyses

Description
Useful when working with `%>%` operators.

Usage

```r
select_phases(data, A, B, phase_names = "auto")
```

Arguments

data A single-case data frame. See scdf() to learn about this format.
A Selection of the A phase
B Selection of the B phase
phase_names A character vector with names for the resulting phases. The default "auto" generates phase names from the combination of the names of the recombined phases.

Value
An scdf with selected phases

Examples

```r
exampleA1B1A2B2_zvt %>%
  select_phases(A = c(1, 3), B = c(2, 4)) %>%
  overlap()
```
set_vars

Set analysis variables in an scdf

Description
Set analysis variables in an scdf

Usage
set_vars(data, dvar, mvar, pvar)
set_dvar(data, dvar)
set_mvar(data, mvar)
set_pvar(data, pvar)

Arguments
data A single-case data frame. See scdf() to learn about this format.
dvar Character string. Name of the dependent variable.
mvar Character string. Name of the measurement-time variable.
pvar Character string. Name of the phase variable.

See Also
Other data manipulation functions: add_l2(), as.data.frame.scdf(), as_scdf(), fill_missing(), moving_median(), outlier(), ranks(), scdf(), select_cases(), shift(), smooth_cases(), standardize(), truncate_phase()

Examples
exampleAB_add %>%
  set_dvar("depression") %>
  describe()

shinyscan

A Shiny app for scan

Description
Run a Shiny app with most of the scan functions.

Usage
shinyscan(quiet = TRUE, ...)

Arguments

quiet If TRUE (default) does not report shiny messages in the console.

... Further arguments passed to the shiny::runApp() function.

Details

This function launches a shiny application. You need to have scplot and shiny installed. These packages are suggested but not necessarily installed along with scan. shinyscan() will ask to install missing packages.

---

smd Standardized mean differences

---

Description

The smd() function provides various standardized mean effect sizes for single-case data.

Usage

smd(data, dvar, pvar, mvar, decreasing = FALSE, phases = c(1, 2))

Arguments

data A single-case data frame. See scdf() to learn about this format.
dvar Character string with the name of the dependent variable. Defaults to the attributes in the scdf file.
pvar Character string with the name of the phase variable. Defaults to the attributes in the scdf file.
mvar Character string with the name of the measurement time variable.Defaults to the attributes in the scdf file.
decreasing If you expect data to be lower in the B phase, set decreasing = TRUE. Default is decreasing = FALSE.
phases A vector of two characters or numbers indicating the two phases that should be compared. E.g., phases = c("A","C") or phases = c(2, 4) for comparing the second to the fourth phase. Phases could be combined by providing a list with two elements. E.g., phases = list(A = c(1, 3), B = c(2, 4)) will compare phases 1 and 3 (as A) against 2 and 4 (as B). Default is phases = c(1, 2).

Details

'sd cohen' is the (unweighted) average of the variance of phase A and B. 'sd Hedges' is the weighted average of the variance of phase A and B (with a degrees of freedom correction), 'Hedges' g' is the mean difference divided by 'sd Hedges'. 'Hedges' g correction' and 'Hedges' g durlak correction' are two approaches of correcting Hedges' g for small sample sizes. 'Glass' delta' is the mean difference divided by the standard deviation of the A-phase. 'Cohens d' is the mean difference divided by 'sd cohen'.

smd
**style_plot**

**Author(s)**
Juergen Wilbert

**See Also**
overlap(), describe()

**Examples**

smd(exampleAB)

---

**style_plot**  
*Create styles for single-case data plots*

**Description**

The `style_plot` function is used to create graphical styles for a single-case plot.

**Usage**

```r
style_plot(style = "default", ...)
```

**Arguments**

- **style**
  A character string or a vector of character strings with predefined styles.

- **...**
  Further arguments passed to the plot command.

**Details**

`style_plot("")` will return a list of predefined styles. Predefined styles can be combined with:

```r
style_plot(style = c("grid2", "tiny"))
```

where settings of a later style overwrite settings of the former. Additional style parameters are set following the style argument and can be combined with those:

```r
style_plot(style = "grid2", fill = "grey50", pch = 18).
```

**Value**

Returns a list to be provided for the style argument of the `plot.scdf()` function.

- **fill**
  If set, the area under the line is filled with the given color (e.g., fill = "tomato"). Use the standard R command `colors()` to get a list of all possible colours. fill is empty by default.

- **annotations**
  A list of parameters defining annotations to each data point. This adds the score of each MT to your plot.
  - **pos**
    Position of the annotations: 1 = below, 2 = left, 3 = above, 4 = right.
  - **col**
    Color of the annotations.
  - **cex**
    Size of the annotations.
  - **round**
    Rounds the values to the specified decimal.
• annotations = list(pos = 3, col = "brown", round = 1) adds scores rounded to one decimal above the data point in brown color to the plot.

• "names" A list of parameters defining the depiction of phase names (e.g. names = list(cex = 0.8, col = "red", side = 1): cex for size, col for color, and side for position). See mtext for more details.

• "lwd" Width of the plot line. Default is lwd = 2.

• "pch" Point type. Default is pch = 17 (triangles). Other options are for example: 16 (filled circles) or "A" (uses the letter A).

• "main" Main title of the plot.

• "mai" Sets the margins of the plot.

• "bty" Shape of the frame surrounding the inner plot

• "fill.bg" Background color of the plot. If a vector is provided, these colors will be assigned to phases (each phase name becomes a color).

• "grid" Color of a grid.

• "text.ABlag" Text displayed between phases.

• "cex.axis" Size of the axis annotations

• "las" Orientation of the axis annotations

• "col.lines" Color of the lines

• "col.dots" Color of the dots

• "col.seperator" Color of the phase separating lines

• "col.bg" Color of the outer plot

• "col" General color setting for the plot

• "col.text" Color of all labels of the plot.

Author(s)
Juergen Wilbert

See Also
plot.scdf()

Examples
newstyle <- style_plot(style = "default")
newstyle$text.ABlag <- c("START", "END")
newstyle$col.dots <- ""
newstyle$annotations <- list(cex = 0.6, col = "grey10", offset = 0.4)
newstyle$names <- list(cex = 0.8, col = "blue", side = 1, adj = 1, line = -1, at = 31)
newstyle$fill.bg <- c("grey99", "grey95", "grey90")
plot(exampleABC, style = newstyle, main = "Example Plot")
Description

This function is mainly used to filter rows by a logical expression. It has also arguments to filter variables and cases.

Usage

```r
## S3 method for class 'scdf'
subset(x, subset, select, cases, ...)
```

Arguments

- `x`: An scdf object.
- `subset`: Logical expression indicating rows to keep: missing values are taken as false.
- `select`: Expression, indicating columns to select from an scdf.
- `cases`: Expression, indicating cases to keep from an scdf.
- `...`: not implemented

Value

An scdf.

Examples

```r
exampleAB %>%
  subset((values < 60 & phase == "A") | (values >= 60 & phase == "B"))
subset(exampleAB_add, select = c(-cigarrets, -depression))
subset(exampleAB, cases = c(Karolina, Johanna))
subset(exampleA1B1A2B2, phase %in% c("A1", "B2"), cases = Pawel:Moritz)
```

summary.scdf

Summary function for an scdf

Description

Summary function for an scdf

Usage

```r
## S3 method for class 'scdf'
summary(object, all_cases = FALSE, ...)
```
tau_u

**tau_u**  
*Tau-U for single-case data*

**Arguments**

- `object`  
  `scdf`  
- `all_cases`  
  IF TRUE, more that 10 cases are summarized  
- ...  
  not in use

**Description**

This function calculates indices of the Tau-U family as proposed by Parker et al. (2011a).

**Usage**

```r
tau_u(
  data,
  dvar,
  pvar,
  tau_method = c("b", "a"),
  method = c("complete", "parker"),
  phases = c(1, 2),
  meta_analyses = TRUE,
  ci = 0.95,
  ci_method = c("z", "tau", "s"),
  meta_weight_method = c("z", "tau"),
  continuity_correction = FALSE,
  meta_method = NULL
)
```

```r
## S3 method for class 'sc_tauu'
print(
  x,
  complete = FALSE,
  digits = "auto",
  select = c("Tau", "CI lower", "CI upper", "SD_S", "Z", "p"),
  nice_p = TRUE,
  ...
)
```

```r
## S3 method for class 'sc_tauu'
export(
  object,
  caption = NA,
  footnote = NA,
  filename = NA,
  select = "auto",
)
tau_u

kable_styling_options = list(),
kable_options = list(),
meta = TRUE,
...
)

Arguments

data A single-case data frame. See scdf() to learn about this format.
dvar Character string with the name of the dependent variable. Defaults to the attributes in the scdf file.
pvar Character string with the name of the phase variable. Defaults to the attributes in the scdf file.
tau_method Character with values "a" or "b" (default) indicating whether Kendall Tau A or Kendall Tau B is applied.
method "complete" (default) or "parker". The latter calculates the number of possible pairs as described in Parker et al. (2011) which might lead to tau-U values greater than 1.
phases A vector of two characters or numbers indicating the two phases that should be compared. E.g., phases = c("A", "C") or phases = c(2,4) for comparing the second to the fourth phase. Phases could be combined by providing a list with two elements. E.g., phases = list(A = c(1,3), B = c(2,4)) will compare phases 1 and 3 (as A) against 2 and 4 (as B). Default is phases = c(1,2).
meta_analyses If TRUE, a meta analysis is conducted.
ci Confidence intervals
.ci_method String to specify the method for calculating the standard error of tau. Either "tau", "z", or "s" (not recommended).
.meta_weight_method String to specify the method for calculating the weights of the studies. Either "tau" or "z".
continuity_correction If TRUE, a continuity correction is applied for calculating p-values of correlations (here: S will be reduced by one before calculating Z)
.meta_method (not implemented) All meta analyses are based on a fixed model.
x Object returned from tau_u().
complete Print all parameters.
digits The minimum number of significant digits to be use. If set to "auto" (default), values are predefined.
select Character vector with name of variables to be included. When the vector is named, variables are renamed appropriately.
nice_p If TRUE, p-values are printed in publication friendly form.
... Further arguments passed to the function.
object An scdf or an object exported from a scan function.
caption Character string with table caption. If left NA (default) a caption will be created based on the exported object.

footnote Character string with table footnote. If left NA (default) a footnote will be created based on the exported object.

filename String containing the file name. If a filename is given the output will be written to that file.

kable_styling_options list with arguments passed to the kable_styling function.

kable_options list with arguments passed to the kable function.

meta If TRUE, the results of the meta analysis will be exported. If FALSE, each single-case is exported.

Details

Tau-U is an inconsistently operationalized construct. Parker et al. (2011b) describe a method which may result in Tau-U outside the [-1;1] interval. A different implementation of the method (provided at http://www.singlecaseresearch.org/calculators/tau-u) uses tau-b (instead of tau-a as in the original formulation by Parker). Bossart et. al (2018) describe inconsistencies in the results from this implementation as well. Another problems lies in the calculation in overall Tau-U values from several single cases. The function presented here applies a meta-analysis to gain the overall values. Each tau value is weighted by the inverse of the variance (ie. the tau standard error). The confidence intervals for single cases are calculated by Fisher-Z transforming tau, calculating the confidence intervals, and inverse transform them back to tau (see Long & Cliff, 1997).

Value

<table>
<thead>
<tr>
<th>Description</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>A data frame containing statistics from the Tau-U family, including: Pairs, positive and negative comparisons, S, and Tau</td>
</tr>
<tr>
<td>matrix</td>
<td>The matrix of comparisons used for calculating the statistics.</td>
</tr>
<tr>
<td>tau_u</td>
<td>Tau-U value.</td>
</tr>
</tbody>
</table>

Functions

- `print(sc_tauu)`: Print results
- `export(sc_tauu)`: Export results as html table

Author(s)

Juergen Wilbert

References


**See Also**

Other overlap functions: `cdc()`, `nap()`, `overlap()`, `pand()`, `pem()`, `pet()`, `pnd()`

**Examples**

tau_u(Grosche2011$Eva)

```r
## Replicate tau-U calculation from Parker et al. (2011)
bob <- scdf(c(A = 2, 3, 5, 3, B = 4, 5, 5, 7, 6), name = "Bob")
res <- tau_u(bob, method = "parker", tau_method = "a")
print(res, complete = TRUE)
```

```
## Request tau-U for all single-cases from the Grosche2011 data set
tau_u(Grosche2011)
```

---

**trend**

*Trend analysis for single-cases data*

**Description**

The `trend()` function provides an overview of linear trends in single-case data. By default, it provides the intercept and slope of a linear and quadratic regression of measurement time on scores. Models are calculated separately for each phase and across all phases. For more advanced use, you can add regression models using the R-specific formula class.

**Usage**

```r
trend(
data, 
dvar, 
pvar, 
mvar, 
offset = "deprecated", 
first_mt = 0, 
model = NULL)
```

Arguments

data A single-case data frame. See scdf() to learn about this format.
dvar Character string with the name of the dependent variable. Defaults to the attributes in the scdf file.
pvar Character string with the name of the phase variable. Defaults to the attributes in the scdf file.
mvar Character string with the name of the measurement time variable. Defaults to the attributes in the scdf file.
offset (Deprecated. Please use first_mt). An offset for the first measurement-time of each phase. If offset = 0, the phase measurement is handled as MT 1. Default is offset = -1, setting the first value of MT to 0.
first_mt A numeric setting the value for the first measurement-time. Default = 0.
model A string or a list of (named) strings each depicting one regression model. This is a formula expression of the standard R class. The parameters of the model are values, mt and phase.

Value

trend A matrix containing the results (Intercept, B and beta) of separate regression models for phase A, phase B, and the whole data.
offset Numeric argument from function call (see arguments section).

Author(s)

Juergen Wilbert

See Also
describe()

Other regression functions: autocorr(), corrected_tau(), hplm(), mplm(), plm()

Examples

```r
## Compute the linear and squared regression for a random single-case
design <- design(slope = 0.5)
matthea <- random_scdf(design)
trend(matthea)

## Besides the linear and squared regression models compute two custom models:
## a) a cubic model, and b) the values predicted by the natural logarithm of the
## measurement time.
design <- design(slope = 0.3)
ben <- random_scdf(design)
trend(ben, offset = 0, model = c("Cubic" = values ~ I(mt^3), "Log Time" = values ~ log(mt)))
```
**Description**

This function restructures and writes single-case data into a .csv-file.

**Usage**

```r
write_scdf(data, filename = NULL, sep = ",", dec = ".", ...)```

**Arguments**

- `data`: A single-case data frame. See `scdf()` to learn about this format.
- `filename`: A character string defining the output file name (e.g. "SC_data.csv").
- `sep`: The field separator string. Values within each row of `x` are separated by this string.
- `dec`: The string to use for decimal points in numeric or complex columns: must be a single character.
- `...`: Further arguments passed to `write.table`.

**Details**

This is a wrapper for the `write.table` function with predefined parameters.

**Author(s)**

Juergen Wilbert

**See Also**

`write.table()`, `saveRDS()`

Other io-functions: `convert()`, `read_scdf()`

**Examples**

```r
## write single-case data to a .csv-file
filename <- tempfile(fileext = ".csv")
jessica <- random_scdf(design(level = .5))
write_scdf(jessica, tempfile())

## write multiple cases to a .csv-file with semicolon as field and comma as decimal separator
write_scdf(Grosche2011, filename, sep = ";", dec = ",")

## read_scdf and write_scdf
write_scdf(exampleA1B1A2B2_zvt, filename)
```
dat <- read_scdf(filename, cvar = "case", pvar = "part",
                 dvar = "zvt", mvar = "day")
res1 <- describe(exampleA1B1A2B2_zvt)$descriptives
res2 <- describe(dat)$descriptives
all.equal(res1, res2)
# Index

- **Autocorrelation**
  - autocorr, 6
- **Serial correlation**
  - autocorr, 6
- **data manipulation functions**
  - add_l2, 3
  - as.data.frame.scdf, 4
  - as.scdf, 5
  - fill_missing, 22
  - moving_median, 28
  - outlier, 34
  - scdf, 61
  - select_cases, 63
  - set_vars, 65
- **datagen**
  - design, 14
  - random_scdf, 54
- **io-functions**
  - convert, 11
  - read_scdf, 59
  - write_scdf, 75
- **io**
  - convert, 11
  - read_scdf, 59
  - write_scdf, 75
- **manip**
  - as.data.frame.scdf, 4
  - fill_missing, 22
  - outlier, 34
- **mc functions**
  - random_scdf, 54
- **mc functions**
  - design, 14
- **overlap functions**
  - cdc, 7
  - nap, 33
  - overlap, 36
  - pand, 37
  - pem, 40
- **pet, 42**
  - pnd, 50
  - tau_u, 70
- **overlap**
  - cdc, 7
- **regression functions**
  - autocorr, 6
  - corrected_tau, 12
  - hplm, 23
  - mplm, 31
  - plm, 43
  - trend, 73
- **regression**
  - autocorr, 6
- **transform**
  - add_l2, 3
  - across_cases(moving_median), 28
  - add_l2, 3, 4, 5, 23, 29, 35, 62, 64, 65
  - all_cases(moving_median), 28
  - as.data.frame.scdf, 3, 4, 5, 23, 29, 35, 62, 64, 65
  - as.scdf(scdf), 61
  - as_scdf, 3, 4, 5, 23, 29, 35, 62, 64, 65
  - autocorr, 6, 13, 26, 32, 45, 74
  - batch_apply, 7
  - c.scdf(combine), 10
  - cdc, 7, 33, 37, 40, 41, 43, 50, 73
  - chisq.test(), 39
  - coef.sc_hplm(hplm), 23
  - coef.sc_plm, 9
  - combine, 10
  - convert, 11, 60, 75
  - correctedTau, 6, 12, 26, 32, 45, 74
  - describe, 13
  - describe(), 67, 74

77
describeSC, 48

design, 14
design(), 52

estimate_design, 17
export, 19
export(), 26, 39
export.sc_hplm (hplm), 23
export.sc_pand (pand), 37
export.sc_tauu (tau_u), 70

fill_missing, 3–5, 22, 29, 35, 62, 64, 65

first_of (moving_median), 28
f Fisher.test(), 39

hplm, 6, 13, 23, 32, 45, 74
hplm(), 3, 23, 25

ird, 27
ird(), 27
is.scdf, 28

local_regression (moving_median), 28

mblm(), 12

moving_mean (moving_median), 28

moving_median, 3–5, 23, 28, 35, 62, 64, 65
mplm, 6, 13, 26, 31, 45, 74
mplm(), 31, 32
mtext, 68

nap, 9, 33, 37, 40, 41, 43, 50, 73
nap(), 33

outlier, 3–5, 23, 29, 34, 62, 64, 65
overlap, 9, 33, 36, 40, 41, 43, 50, 73
overlap(), 14, 22, 67
overlapSC, 48

pand, 9, 33, 37, 37, 41, 43, 50, 73
pand(), 38

pem, 9, 33, 37, 40, 40, 43, 50, 73
pet, 9, 33, 37, 40, 41, 42, 50, 73
plm, 6, 13, 26, 32, 43, 74
plm(), 51
plot.scdf, 46
plot.scdf(), 14, 67, 68
plot.rand, 49
plot.rand(), 56
plotSC (plot.scdf), 46

pnd, 9, 33, 37, 40, 41, 43, 50, 73
power_test, 51
print.sc_hplm (hplm), 23
print.sc_ird (ird), 27
print.sc_mplm (mplm), 31
print.sc_pand (pand), 37
print.sc_tauu (tau_u), 70
print.scdf, 53

rand_test, 55
rand_test(), 22, 51
random_scdf, 54
random_scdf(), 52
ranks, 3–5, 23, 29, 35, 62, 64, 65
rci, 58
read.table(), 60
read_scdf, 11, 59, 75
readRDS(), 60

sample_names, 60
saveRDS(), 75
scdf, 3–5, 23, 29, 35, 54, 60, 61, 64, 65
scdf(), 3, 6, 8, 11–13, 18, 22, 24, 27, 31, 33,
34, 36, 38, 41, 42, 44, 47, 50, 55, 58,
63–66, 71, 74, 75

scdf-class (scdf), 61

select_cases, 3–5, 23, 29, 35, 62, 63, 65
select_phases, 64

set_dvar (set_vars), 65
set_mvar (set_vars), 65
set_pvar (set_vars), 65
set_vars, 3–5, 23, 29, 35, 62, 64, 65
shift, 3–5, 23, 29, 35, 62, 64, 65
shinySc, 65
smd, 66
smooth_cases, 3–5, 23, 29, 35, 62, 64, 65
standardize, 3–5, 23, 29, 35, 62, 64, 65
style_plot, 48, 67

subset.scdf, 69
summary.scdf, 69

tau_u, 9, 33, 37, 40, 41, 43, 50, 70
tau_u(), 51, 71

transform.scdf (moving_median), 28
trend, 6, 13, 26, 32, 45, 73
truncate_phase, 3–5, 23, 29, 35, 62, 64, 65

write.table(), 75
write_scdf, 11, 60, 75