Package ‘scan’

April 3, 2022

Type Package

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

Version 0.54.1

Date 2022-04-03

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

Depends R (>= 3.5.0)

Description A collection of procedures for analysing, visualising, and managing single-case data. These include piecewise linear regression models, multilevel models, overlap indices (PND, PEM, PAND, PET, tauU, baseline corrected tau, CDC), and randomization tests. Data preparation functions support outlier detection, handling missing values, scaling, truncating, rank transformation, and smoothing. An exporting function helps to generate html and latex tables in a publication friendly style. More details can be found at <https://jazznbass.github.io/scan-Book/>.

License GPL (>= 3)

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

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

LazyData true

Encoding UTF-8

RoxygenNote 7.1.2

NeedsCompilation no

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

Maintainer Juergen Wilbert <juergen.wilbert@uni-potsdam.de>

Repository CRAN

Date/Publication 2022-04-03 14:50:02 UTC

R topics documented:

`scan-package` *Single-Case Data Analyses*

Description

A collection of procedures for analysing, visualising, and managing single-case data.

Author(s)

Juergen Wilbert [aut, cre]

`.inheritParams` *Dummy function to inherit global descriptions of parameters*

Description

Dummy function to inherit global descriptions of parameters

Usage

```
.inheritParams(
  data,
  scdf,
  dvar,
  mvar,
  pvar,
  decreasing,
  phases,
  model,
  trend,
  level,
  slope,
  ...
)
```

Arguments

`data`	A single-case data frame. See `scdf` to learn about this format.
`scdf`	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.
`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("A","B")`.
`model`	Model used for calculating the slope parameter (see Huitema & McKean, 2000). Default is `model = "B&L-B"`. Possible values are: `"B&L-B"`, `"H-M"`, `"Mohr#1"`, `"Mohr#2"`, `"JW"`, `"JW2"`, and `"Manly"`.
`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.
`...`	Further arguments passed to the function.

`add_l2` *Add level 2*

Description

Add level 2 variables from a level 2 dataset to an scdf file

Usage

```
add_l2(data, data_l2, cvar = "case")
```

Arguments

`data`	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').

Value

An scdf

Examples

```
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: `fill_missing()`, `outlier()`, `ranks()`, `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

## 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 Converts a data frame to an scdf object*

Description

as_scdf Converts a data frame to an scdf object

Usage

```
as_scdf(object)
```

Arguments

`object`	A scdf object

---

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

`trend`, `plm`, `acf`

Examples

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

`autocorrSC` *List of old deprecated function names*

Description

This is a list with functions names that have been replaced by new names which are in line with R syntax guidelines. The old function names still work. They are wrappers that call the new function.

Usage

```
autocorrSC(...)

design_rSC(...)

longSCDF(...)
```

```
corrected_tauSC(...)

trendSC(...)

truncateSC(...)

style_plotSC(...)

style.plotSC(...)

smoothSC(...)

shiftSC(...)

tauUSC(...)

describeSC(...)

rankSC(...)

fillmissingSC(...)

outlierSC(...)

overlapSC(...)

power_testSC(...)

randSC(...)

rand.test(...)

rSC(...)

rciSC(...)

rCi(...)
```

Arguments

`...`	Arguments passed through to the new function.

Beretvas2008 *Single-case example data*

Description

The `scan` package comes with a set of fictitious and authentic single-case study data, by courtesy of the particular authors.

Usage

```
Beretvas2008
```

Format

An object of class `scdf` (inherits from `list`) of length 1.

Value

`Beretvas2008`	Fictitious single-case intervention study. **Reference:** Beretvas, S., & Chung, H. (2008). An evaluation of modified R2-change effect size indices for single-subject experimental designs. *Evidence-Based Communication Assessment and Intervention, 2*, 120-128.
`Borckardt2014`	Fictitious daily pain ratings evaluating a psychological treatment. **Reference:** Borckardt, J. J., & Nash, M. R. (2014). Simulation modelling analysis for small sets of single-subject data collected over time. *Neuropsychological Rehabilitation, 24*, 492-506.
`Huitema2000`	Fictitious single-case intervention study. **Reference:** Huitema, B. E., & McKean, J. W. (2000). Design specification issues in time-series intervention models. *Educational and Psychological Measurement, 60*, 38-58.
`Waddell2011`	Fictitious single-case intervention study. **Reference:** Waddell, D. E., Nassar, S. L., & Gustafson, S. A. (2011). Single-Case Design in Psychophysiological Research: Part II: Statistical Analytic Approaches. *Journal of Neurotherapy, 15*, 160-169.
`byHeart2011`	Multiple-baseline (11 cases) intervention study on flash card vocabulary learning by Juergen Wilbert.
`Grosche2011`	Multiple-baseline (3 cases) intervention study on a direct-instructive reading intervention. **Reference:** Grosche, M. (2011). Effekte einer direkt-instruktiven Foerderung der Lesegenauigkeit. *Empirische Sonderpaedagogik, 3*, 147-161.
`Grosche2014`	Multiple-baseline (3 cases x 3 materials) intervention study on a reading intervention. **Reference:** Grosche, M., Lueke, T., & Wilbert, J. (in prep.).
`GruenkeWilbert2014`	Multiple-baseline (6 cases) intervention study on story mapping. **Reference:** Gruenke, M., Wilbert, J., & Stegemann-Calder, K. (2013). Analyzing the effects of story mapping on the reading comprehension of children with low intellectual abilities. *Learning Disabilities: A Contemporary Journal, 11*, 51-64.
`Huber2014`	Multiple-baseline (4 cases) intervention study on behavioral compliance. Scores refer to compliant behavior in percent. **Reference:** Huber, C. (in prep.).
`Lenz2013`	Fictious example from the paper Lenz, A. S. (2013). Calculating Effect Size in Single-Case Research: A Comparison of Nonoverlap Methods. Measurement and Evaluation in Counseling and Development, 46(1), 64–73.

`Leidig2018`	
`Leidig2018_l2`	
`SSDforR2017`	Example from the R package SSDforR.
`Parker2011`	Example from Parker, R. I., Vannest, K. J., Davis, J. L., & Sauber, S. B. (2011). Combining Nonoverlap and Trend for Single-Case Research: Tau-U. Behavior Therapy, 42(2), 284–299. https://doi.org/10.1016/j.beth.2010.08.006

Author(s)

Juergen Wilbert

`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

```
cdc(
  data,
  dvar,
  pvar,
  mvar,
  decreasing = FALSE,
  trend_method = "OLS",
  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("A","B")`.

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.
`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

Fisher, W. W., Kelley, M. E., & Lomas, J. E. (2003). Visual Aids and Structured Criteria for Improving Visual Inspection and Interpretation of Single-Case Designs. *Journal of Applied Behavior Analysis, 36*, 387-406. https://doi.org/10.1901/jaba.2003.36-387

Examples

```
## 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)
```

---

check_scdf — *Validity check for an scdf object*

Description

Validity check for an scdf object

Usage

```
check_scdf(object)
```

Arguments

object	An scdf object

Value

TRUE or list with error and warning messages.

Examples

```
check_scdf(exampleAB)

check_scdf(c(exampleAB, exampleABC))
```

---

combine — *Combine single-case data frames*

Description

Combine single-case data frames

Usage

```
combine(..., dvar = NULL, pvar = NULL, mvar = NULL)

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

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.

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).

`convert` *Covert Converts an scdf object to a text syntax*

Description

Covert Converts an scdf object to a text syntax

Usage

```
convert(scdf, file = "", study_name = "study")
```

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 single case study name.

Examples

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

corrected_tau *Baseline corrected tau*

Description

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

Usage

```
corrected_tau(
  data,
  dvar,
  pvar,
  mvar,
  phases = c(1, 2),
  alpha = 0.05,
  continuity = TRUE,
  repeated = 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.
`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("A","B")`.
`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`)

Details

This method has been proposed by Tarlow (2016). The baseline data are checked for a significant autocorrelation (based on Kendalls Tau). If so, a non-parameteric 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, Kendalls 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` (both are the default settings).

References

Tarlow, K. R. (2016). An Improved Rank Correlation Effect Size Statistic for Single-Case Designs: Baseline Corrected Tau. Behavior Modification, 41(4), 427–467. https://doi.org/10.1177/0145445516676750

See Also

Other regression functions: `hplm()`, `mplm()`, `plm()`

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

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; mis = number of missing vaues; m = mean; md = median; sd = standard deviation; mad = median average deviation; min = minimum; max = maximum; 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

`overlap`, `plot.scdf`

Examples

```
## 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(Waddell2011)
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 `random_scdf` function and the `power_test` function and for other Monte-Carlo tasks.

Usage

```
design(
  n = 1,
  phase_design = list(A = 5, B = 15),
  trend = 0,
  level = list(0),
  slope = list(0),
  start_value = 50,
  s = 10,
  rtt = 0.8,
  extreme_prop = list(0),
  extreme_range = c(-4, -3),
  missing_prop = 0,
  distribution = "normal",
  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 *d* 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 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 = "poission"` the start_value represents frequency. When `distribution = "binomial"` start_value must range between 0 and 1 and they represent the probability of on 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% of all values as missing). A vector of values assigns different probabilities to multiple cases. If the number of cases exceeds the length of the vector, values are repeated.
`distribution`	Distribution of the criteria varible. 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.

Value

An object of class sc_design.

Author(s)

Juergen Wibert

Examples

```
## 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, correction = FALSE)
```

---

estimate_design *Estimate single-case design*

Description

This functions 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

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

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 slope parameter (see Huitema & McKean, 2000). Default is `model = "B&L-B"`. Possible values are: `"B&L-B"`, `"H-M"`, `"Mohr#1"`, `"Mohr#2"`, `"JW"`, `"JW2"`, and `"Manly"`.
`...`	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

```
# 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

This function is in an experimental status. Export creates html files of tables or displayes them directly in the viewer pane of rstudio. When applied in rmarkdown, tables can also be created for pdf/latex output.

Usage

```
export(object, ...)

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

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

## 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_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_overlap'
export(
  object,
  caption = NA,
  footnote = NA,
  filename = NA,
  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,
  flip = TRUE,
  ...
```

```
)

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

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`	Character string with the filename. If a filename is provided the output will be written into this file.
`kable_styling_options`	list with arguments passed to the kable_styling function.
`kable_options`	list with arguments passed to the kable function.
`cols`	Defines which columns are included when a scdf is exported. It is either a vector with variable names or the string "main" will select the central variables.
`flip`	If TRUE, some objects are exported with rows and columns flipped.
`round`	value for the digits argument passed to the internally used round function.
`nice`	If set TRUE (default) output values are rounded and optimized for publication tables.

Value

Returns a specif formated html (or latex).

`fill_missing` *Replacing missing measurement times in single-case data*

Description

The `fillmissingSC` function replaces missing measurements in single-case data.

Usage

```
fill_missing(data, dvar, mvar, interpolation = "linear", 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.
`interpolation`	Alternative options not yet included. Default is `interpolation = "linear"`.
`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 (`overlapSC`) or a randomization test (`randSC`).

Value

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

Author(s)

Juergen Wilbert

See Also

Other data manipulation functions: `as.data.frame.scdf()`, `outlier()`, `ranks()`, `shift()`, `smooth_cases()`, `standardize()`, `truncate_phase()`

Examples

```
## 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

```
hplm(
  data,
  dvar,
  pvar,
  mvar,
  model = "B&L-B",
  method = "ML",
  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.l2 = 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.

`model`	Model used for calculating the slope parameter (see Huitema & McKean, 2000). Default is `model = "B&L-B"`. Possible values are: `"B&L-B"`, `"H-M"`, `"Mohr#1"`, `"Mohr#2"`, `"JW"`, `"JW2"`, and `"Manly"`.
`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) adresses 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.

Value

`model`	List containing infromation about the applied model
`N`	Number of single-cases.
`formla`	A list containing the fixed and the random formulas of the hplm model.
`hplm`	Object of class lme contaning 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.

Author(s)

Juergen Wilbert

See Also

Other regression functions: `corrected_tau()`, `mplm()`, `plm()`

Examples

```
## 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
hplm(Leidig2018, data.l2 = Leidig2018_l2,
     update.fixed = .~. + gender + migration + ITRF_TOTAL*phaseB,
     slope = FALSE, random.slopes = FALSE, lr.test = FALSE)
```

`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.

`mplm` *Multivariate Piecewise linear model / piecewise regression*

Description

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

Usage

```
mplm(
  data,
  dvar,
  mvar,
  pvar,
  model = "B&L-B",
  trend = TRUE,
  level = TRUE,
  slope = TRUE,
  formula = NULL,
  update = NULL,
  na.action = na.omit,
  ...
)
```

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 slope parameter (see Huitema & McKean, 2000). Default is `model = "B&L-B"`. Possible values are: `"B&L-B"`, `"H-M"`, `"Mohr#1"`, `"Mohr#2"`, `"JW"`, `"JW2"`, and `"Manly"`.
`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., 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., "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., . ~ . + newvariable).
`na.action`	Defines how to deal with missing values
`...`	Further arguments passed to the lm function.

Value

`model`	Character string from function call (see `Arguments` above).
`full.model`	Full regression model list

Author(s)

Juergen Wilbert

See Also

Other regression functions: `corrected_tau()`, `hplm()`, `plm()`

Examples

```
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 $number of pairs showing no overlap/number of pairs$. Because NAP can only take values between 50 and 100 percent, a rescaled and therefore more intuitive NAP (0-100%) is also displayed.

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("A","B")`.

Value

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

Author(s)

Juergen Wilbert

References

Parker, R. I., & Vannest, K. (2009). An improved effect size for single-case research: Nonoverlap of all pairs. *Behavior Therapy, 40*, 357-367.

See Also

Other overlap functions: `corrected_tau()`, `overlap()`, `pand()`, `pem()`, `pet()`, `pnd()`, `tau_u()`

Examples

```
## 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

```
outlier(data, dvar, pvar, mvar, criteria = c("MAD", "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.
`criteria`	Specifies the criteria for outlier identification. Set `criteria = c("SD",2)` to define two standard deviations as limit. This is also the default setting. To use the 99% Confidence Interval use `criteria = c("CI",0.99)`. Set `criteria = c("Cook","4/n")` to define any data point with a Cook's Distance greater than 4/n as an outlier, based on the Piecewise Linear Regression Model.

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: `as.data.frame.scdf()`, `fill_missing()`, `ranks()`, `shift()`, `smooth_cases()`, `standardize()`, `truncate_phase()`

Examples

```
## Identify outliers using 1.5 standard deviations as criterion
susanne <- random_scdf(level = 1.0)
res_outlier <- outlier(susanne, criteria = c("SD", 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, criteria = c("Cook", "4/n"))
plot(Grosche2011, marks = res_outlier)
```

overlap *Overlap indices for single-case data*

Description

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

Usage

```
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","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("A","B")`.

Details

See corresponding functions of PND, PEM, PET, NAP, PAND for calculation. Tau_U 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 standardd eviation of phase A. Hedges_g is the mean difference divided by the pooled standard deviation [sqrt(((nA - 1) * sdA^2 + (nB - 1) * sdB^2) / (nA + nB - 2))] with a hedges correction applied [(* (1 - (3 / (4 * n - 9)))].

Value

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

Author(s)

Juergen Wilbert

See Also

Other overlap functions: `corrected_tau()`, `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 analyszing 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, correction = TRUE, 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`.
`correction`	The default `correction = TRUE` makes `pand` use a frequency matrix, which is corrected for ties. A tie is counted as the half of a measurement in both phases. Set `correction = FALSE` to use the uncorrected matrix, which is not recommended.

`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("A","B")`.

Details

The PAND indicates nonoverlap between phase A and B data (like `PND`), but uses all data and is therefore not based on one single (probably unrepresentative) datapoint. Furthermore, PAND allows the comparison of real and expected associations (Chi-square test) and estimation of the effect size Phi, which equals Pearsons r for dichotomous data. Thus, phi-Square is the amount of explained variance. The original procedure for computing the PAND (Parker, Hagan-Burke, & Vannest, 2007) does not account for ambivalent datapoints (ties). The newer `NAP` overcomes this problem and has better precision-power (Parker, Vannest, & Davis, 2014).

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.
`nA`	Number of data points in phase A.
`nB`	Number of data points in phase B.
`pA`	Percentage of data points in phase A.
`pB`	Percentage 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.
`correlation`	A list of the `correlation` values: statistic, parameter, p.value, estimate, null.value, alternative, method, data.name, correction.
`correction`	Logical argument from function call (see `Arguments` above).

Author(s)

Juergen Wilbert

References

Parker, R. I., Hagan-Burke, S., & Vannest, K. (2007). Percentage of All Non-Overlapping Data (PAND): An Alternative to PND. *The Journal of Special Education, 40*, 194-204.

Parker, R. I., & Vannest, K. (2009). An Improved Effect Size for Single-Case Research: Nonoverlap of All Pairs. *Behavior Therapy, 40*, 357-367.

See Also

Other overlap functions: `corrected_tau()`, `nap()`, `overlap()`, `pem()`, `pet()`, `pnd()`, `tau_u()`

Examples

```
## Calculate the PAND for a MMBD over three cases
gunnar <- scdf(c(2,3,1,5,3,4,2,6,4,7), B_start = 5)
birgit <- scdf(c(3,3,2,4,7,4,2,1,4,7), B_start = 4)
bodo   <- scdf(c(2,3,4,5,3,4,7,6,8,7), B_start = 6)
mbd <- c(gunnar, birgit, bodo)
pand(mbd)
pand(bodo)

## 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)
```

pem *Percent exceeding the median*

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

```
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("A","B")`.
`...`	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: `corrected_tau()`, `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)
```

`pet` *Percent exceeding the trend*

Description

The `pet` function provides the percentage of phase B data points exceeding the prediction based on the phase A trend. A binomial test against a 50/50 distribution is computed. Furthermore, the percentage of phase B data points exceeding the upper (or lower) 95 percent confidence interval of the predicted progress is computed.

Usage

```
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("A","B")`.

Value

`PET`	Percent exceeding the trend.
`PET.ci`	Percent exceeding the upper / lower 95%-CI boundary.
`p`	P value of Binomial 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: `corrected_tau()`, `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 = "B&L-B",
  family = "gaussian",
  trend = TRUE,
  level = TRUE,
  slope = TRUE,
  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 `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 `family = "gaussian"`.
`model`	Model used for calculating the slope parameter (see Huitema & McKean, 2000). Default is `model = "B&L-B"`. Possible values are: `"B&L-B"`, `"H-M"`, `"Mohr#1"`, `"Mohr#2"`, `"JW"`, `"JW2"`, and `"Manly"`.
`family`	Set the distribution family. Defaults to a gaussian distribution. See the `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.
`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., "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., . ~ . + newvariable).
`na.action`	Defines how to deal with missing values.
`r_squared`	Logical. If TRUE, delta 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 glm function.

Value

`formula`	plm formula. Uselful 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

Beretvas, S., & Chung, H. (2008). An evaluation of modified R2-change effect size indices for single-subject experimental designs. *Evidence-Based Communication Assessment and Intervention, 2*, 120-128.

Huitema, B. E., & McKean, J. W. (2000). Design specification issues in time-series intervention models. *Educational and Psychological Measurement, 60*, 38-58.

See Also

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

Examples

```
## 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, model = "JW")

## no slope effects were found. Therefore, you might want to the drop slope
## estimation:
plm(dat, slope = FALSE, model = "JW")

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

## A poisson regression
example_A24 %>%
  transform(year = year - year[1])  %>%
  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.

`lines` A list defining one or multiple lines or curves to be plotted. The argument is passed as a list (e.g., `list(type = "median")`). Some of the procedures can be refined with an additional argument (e.g., `lines = list(type = "mean",trim = 0.2)` adds a 20% trimmed mean line. For multiple lines, provide a list element for each line (e.g., `list( list(type = "median",col = "red"),list(type = "trend",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%-trimmed. Other trims can be set, using a trim parameter (e.g., `lines = list(type = "mean",trim = 0.2)` draws a 20%-trimmed mean line).
- `"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%-trimmed mean score. Apply a different trim, by 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"m 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

```
## Request the default plot of the data from Borckhardt (2014)
plot(Borckardt2014)

## 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 MBD 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

```
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

```
pnd(data, dvar, pvar, decreasing = FALSE, phases = c("A", "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.
`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("A","B")`.

Value

`PND`	Percentage of non-overlapping data.

Author(s)

Juergen Wilbert

See Also

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

Examples

```
## Calculate the PND for multiple single-case data
pnd(GruenkeWilbert2014)
```

power_test — *Empirical power analysis for single-case data*

Description

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

Usage

```
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 = 0.5,
  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 return 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`	Either FALSE or a value. If a value is provided, a binomial test is calculated for the total correct identifications (alpha and power) against the provided value.
`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

```
## Assume you want to conduct a single-case study with 15 measurement
## (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.sc *Print methods for scan objects*

Description

Print methods for scan objects

Usage

```
## S3 method for class 'sc_ac'
print(x, digits = "auto", ...)

## S3 method for class 'sc_bctau'
```

```
print(x, nice = TRUE, digits = "auto", ...)

## S3 method for class 'sc_cdc'
print(x, nice = TRUE, ...)

## S3 method for class 'sc_desc'
print(x, digits = "auto", ...)

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

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

## S3 method for class 'sc_mplm'
print(x, digits = "auto", std = FALSE, ...)

## S3 method for class 'sc_nap'
print(x, digits = "auto", ...)

## S3 method for class 'sc_outlier'
print(x, digits = "auto", ...)

## S3 method for class 'sc_overlap'
print(x, digits = "auto", ...)

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

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

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

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

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

## S3 method for class 'sc_power'
print(x, duration = FALSE, ...)

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

## S3 method for class 'sc_rci'
```

```
print(x, ...)

## S3 method for class 'sc_smd'
print(x, digits = "auto", ...)

## S3 method for class 'sc_tauu'
print(x, complete = FALSE, digits = "auto", ...)

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

Arguments

`x`	Object
`digits`	The minimum number of significant digits to be use. If set to "auto" (default), values are predefined.
`...`	Further parameters passed to the print function
`nice`	If set TRUE (default) output values are rounded and optimized for publication tables.
`std`	If TRUE, a table with standardized estimates is included.
`duration`	If TRUE the duration for computation is printed.
`complete`	Print further parameters.

`print.scdf` *Print an scdf*

Description

Print an scdf

Usage

```
## 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.
`...`	Further arguments passed to the print function.

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

```
## 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, correction = FALSE)
```

`rand_test` *Randomization Tests for single-case data*

Description

The `randSC` 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

```
rand_test(
  data,
  dvar,
  pvar,
  statistic = "Mean B-A",
  number = 500,
  complete = FALSE,
  limit = 5,
  startpoints = NA,
  exclude.equal = FALSE,
  graph = FALSE,
  output = "c",
  phases = c("A", "B"),
  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 can not exceed $1/number$ (i.e., `number = 100` results in p-values with an exactness of one percent). Default is `number = 500`. For faster processing use `number = 100`. For more precise p-values set `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., `limit = c(5,3)`). If only one number is given, this number is applied to both phases. Default is `limit = 5`.
`startpoints`	Alternative to the `limit`-parameter `startpoints` exactly defines the possible start points of phase B (e.g., `startpoints = 4:9` restricts the phase B start points to measurements 4 to 9. `startpoints` overruns the `limit`-parameter.
`exclude.equal`	If set to `exclude.equal = 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). `exclude.equal` should be set to `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.
`graph`	If `graph = TRUE`, a histogram of the resulting distribution is plotted. It's `FALSE` by default.
`output`	If set to the default `output = "C"`, detailed information is provided. Set `output = "p"`, to only return the resulting p value.

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 and 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("A","B")`.
seed	A seed number for the random generator.

Value

statistic	Character string from function call (see `Arguments` above).
N	Number of single-cases.
n1	Number of data points in phase A.
n2	Number of data points in phase B.
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) undercuts the requested `number` of combinations.

Author(s)

Juergen Wilbert

References

Bulte, I., & Onghena, P. (2009). Randomization tests for multiple-baseline designs: An extension of the SCRT-R package. *Behavior Research Methods, 41*, 477-485.

Bulte, I., & Onghena, P. (2012). *SCRT: Single-Case Randomization Tests*. Available from: https://CRAN.R-project.org/package=SCRT

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)
```

---

ranks *Rank-transformation of single-case data files*

Description

Rank-transformation of single-case data files

Usage

```
ranks(data, var, grand = TRUE, ...)
```

Arguments

`data`	A single-case data frame. See `scdf` to learn about this format.
`var`	A string or string vector with the names of the variables to be ranked.
`grand`	If TRUE, ranks will be calculated across all cases. If FALSE ranks are calculated within each case.
`...`	Additional paramters passed to the `rank` function.

Value

An `scdf` object where the values of the variable(s) are replaced with ranks.

Author(s)

Juergen Wilbert

See Also

Other data manipulation functions: `as.data.frame.scdf()`, `fill_missing()`, `outlier()`, `shift()`, `smooth_cases()`, `standardize()`, `truncate_phase()`

Examples

```
Huber2014_rank <- ranks(Huber2014, var = "compliance")
plot(Huber2014_rank, style = "grid2")
```

rci *Reliable change index*

Description

CAUTION! This function is still under development and not ready for use! The `rciSC` function computes three indices of reliable change (Wise, 2004) and corresponding descriptive statistics.

Usage

```
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%-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("A","B")`.

Author(s)

Juergen Wilbert

References

Christensen, L., & Mendoza, J. L. (1986). A method of assessing change in a single subject: An alteration of the RC index. *Behavior Therapy, 17*, 305-308.

Hageman, W. J. J., & Arrindell, W. A. (1993). A further refinement of the reliable change (RC) index by improving the pre-post difference score: Introducing RCID. *Behaviour Research and Therapy, 31*, 693-700.

Jacobson, N. S., & Truax, P. (1991). Clinical Significance: A statistical approach to defining meaningful change in psychotherapy research. *Journal of Consulting and Clinical Psychology, 59*, 12-19.

Wise, E. A. (2004). Methods for analyzing psychotherapy outcomes: A review of clinical significance, reliable change, and recommendations for future directions. *Journal of Personality Assessment, 82*, 50 - 59.

Examples

```
## Report the RCIs of the first case from the byHeart data and include a graph
rci(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

```
read_scdf(
  filename,
  data = NULL,
  sort.labels = FALSE,
  cvar = "case",
  pvar = "phase",
  dvar = "values",
  mvar = "mt",
  phase.names = NULL,
  sep = ",",
  dec = ".",
  type = NA,
  ...
)

readSC.excel(...)

readSC(...)
```

Arguments

`filename`	A character string defining the file to be loaded (e.g. `"SC_Anita.csv"`. If left empty a dialog box for choosing will be opened.
`data`	A data frame. As an alternative to `filname`.
`sort.labels`	If set TRUE, the resulting list is sorted by label names (alphabetically increasing).
`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"`.

`phase.names`	A character vector with phase names. Defaults to the phase names provided in the phase variable.
`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.
`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.
`...`	Further arguments passed to the repective read function (`read.table`, `read_excel`, `read_yaml` command.

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`, `writeSC`, `scdf`, `readRDS`

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` *Samples random names*

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

```
sample_names(3)
```

`scdf` *Single case data frame*

Description

The `scdf` class stores single-case study data with one or more single-cases.

Usage

```
scdf(
  values,
  B_start,
  mt,
  phase,
  phase_design = NULL,
  name = NULL,
  dvar = "values",
  pvar = "phase",
  mvar = "mt",
  phase.design,
  B.start,
  ...
)
```

Arguments

`values`	A vector containing measurement values of the dependent variable.
`B_start, B.start`	The first measurement of phase B (simple coding if design is strictly AB).
`mt`	A vector defining measurement times. Default is `mt = (1,2,3,...,n)`.
`phase`	A vector defining phase assignments.
`phase_design, 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 measuresments). 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

Examples

```
## 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"
)
plot(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
## days 3 and 9. Use NA to pass them to the function:
emmi <- scdf(c(5, 7, NA, 5, 7, 12, 16, 18, NA, 14, 19),
  phase_design = c(A = 5, B = 6), name = "Emmi"
)
describe(emmi)

## In a MBD over three cases, data were collected eleven days in a row.
## Intervention starting points differ between subjects as they were
## randomly assigned. The three SCDFs are then combined in a list for
## further conjoined analyses.
charlotte <- scdf(c(A = 5, 7, 10, 5, 12, B = 7, 10, 18, 15, 14, 19))
theresa <- scdf(c(A = 3, 4, 3, 5, B = 7, 4, 7, 9, 8, 10, 12))
antonia <- scdf(c(A = 9, 8, 8, 7, 5, 7, B = 6, 14, 15, 12, 16))
mbd <- c(charlotte, theresa, antonia)
names(mbd) <- c("Charlotte", "Theresa", "Antonia")
overlap(mbd)

## In a classroom-based intervention it was not possible to measure outcomes
## every day, but only on schooldays. The sequence of measurements is passed
## to the package by using a vector of measurement times.
frida <- scdf(
  c(A = 3, 2, 4, 2, 2, 3, 5, 6, B = 8, 10, 8, 12, 14, 13, 12),
  mt = c(1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 14, 15, 16, 17, 18)
)
summary(frida)
describe(frida)

## example with two independent variables and four phases
jim <- scdf(
  zvt = c(47, 58, 76, 63, 71, 59, 64, 69, 72, 77, 76, 73),
  d2 = c(131, 134, 141, 141, 140, 140, 138, 140, 141, 140, 138, 140),
  phase_design = c(A1 = 3, B1 = 3, A2 = 3, B2 = 3), dvar = "zvt"
)
overlap(jim, phases = list(c("A1", "A2"), c("B1", "B2")))
```

`scdf_attr` *Set and get scdf attributes*

Description

Set and get scdf attributes

Usage

```
scdf_attr(x, var)

scdf_attr(x, var) <- value
```

Arguments

`x`	Variable
`var`	Attribute
`value`	set value

Value

Attribute value

`select_cases` *Select a subset of cases*

Description

Select a subset of cases

Usage

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

Arguments

`scdf`	A single-case data frame. See `scdf` to learn about this format.
`...`	Selection criteria. Either numeric or as characters.

Value

An scdf with a subset of cases

Examples

```
select_cases(exampleAB, "Johanna", "Karolina")
# or: select_cases(exampleAB, c("Johanna", "Karolina"))
select_cases(exampleAB, 1,2)
# or: select_cases(exampleAB, 1:2)
select_cases(exampleAB, "-Johanna")
# or: select_cases(exampleAB, c("-Johanna", "-Karolina"))
```

`select_phases` *Select and combine phases for overlap analyses*

Description

Useful when working with

Usage

```
select_phases(data, A, B)
```

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

Value

An scdf with selected phases

Examples

```
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.

Examples

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

`shift` *Shift values in a single-case data file*

Description

This function has been replaced by the much more versatile ‘transform‘ function. Shifting the values might be helpful in cases where the measurement time is given as a time variable (see example below).

Usage

```
shift(data, value, var)
```

Arguments

`data`	A single-case data frame. See `scdf` to learn about this format.
`value`	Number by which to shift the values
`var`	Character string with the name of the target variable. Defaults to the measurement time variable.

Value

A scdf with shifted data

See Also

Other data manipulation functions: `as.data.frame.scdf()`, `fill_missing()`, `outlier()`, `ranks()`, `smooth_cases()`, `standardize()`, `truncate_phase()`

Examples

```
### Shift the measurement time for a better estimation of the intercept
ex <- shift(example_A24, value = -1996)
plm(ex)

# Please use transform instead:
ex <- transform(example_A24, year = year - 1996)
plm(ex)
```

---

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("A","B")`.

Details

'sd cohen' is the (unweigted) 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'.

Author(s)

Juergen Wilbert

See Also

`overlap`, `describe`

Examples

```
smd(exampleAB)
```

`smooth_cases` *Smoothing single-case data*

Description

The `smooth_cases` function provides procedures to smooth single-case data (i.e., to eliminate noise). A moving average function (mean- or median-based) replaces each data point by the average of the surrounding data points step-by-step. With a local regression function, each data point is regressed by its surrounding data points.

Usage

```
smooth_cases(data, dvar, mvar, FUN = "movingMedian", intensity = 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.
`mvar`	Character string with the name of the measurement time variable. Defaults to the attributes in the scdf file.
`FUN`	Function determining the smoothed scores. Default `FUN = "movingMedian"` is a moving Median function. Further possible values are: `"movingMean"` and a non-parametric `"localRegression"`.
`intensity`	For `FUN = "movingMedian"` and `"movingMean"` it is the lag used for computing the average. Default is `intensity = 1`. In case of `FUN = "localRegression"` it is the proportion of surrounding data influencing each data point, which is `intensity = 0.2` by default.

Value

Returns a data frame (for each single-case) with smoothed data points. See `scdf` to learn about the format of these data frames.

Author(s)

Juergen Wilbert

See Also

Other data manipulation functions: `as.data.frame.scdf()`, `fill_missing()`, `outlier()`, `ranks()`, `shift()`, `standardize()`, `truncate_phase()`

Examples

```
## Use the three different smoothing functions and compare the results
study <- c(
  "Original" = Huber2014$Berta,
  "Moving Median" = smooth_cases(Huber2014$Berta, FUN = "movingMedian"),
  "Moving Mean" = smooth_cases(Huber2014$Berta, FUN = "movingMean"),
  "Local Regression" = smooth_cases(Huber2014$Berta, FUN = "localRegression")
)
plot(study)
```

standardize *Standardize values of an scdf file*

Description

This function scales the measured values of an scdf file. It allows for mean centering and standardization based on each single-case data set or a scaling across all cases included in an scdf.

Usage

```
standardize(
  data,
  var,
  center = TRUE,
  scale = FALSE,
  m = 0,
  sd = 1,
  grand = TRUE
)
```

Arguments

`data`	A single-case data frame. See `scdf` to learn about this format.
`var`	A character string or a vector of character strings with variable names that should be scaled.
`center`	If set TRUE, data are mean centered.
`scale`	If set TRUE, the standard deviation is set.
`m`	The target mean for centering.
`sd`	The target standard deviation for scaling
`grand`	If set TRUE, scaling is based on the mean and standard deviation of all values across all single-cases within the scdf.

Value

An scdf with the scaled values.

Author(s)

Juergen Wilbert

See Also

Other data manipulation functions: `as.data.frame.scdf()`, `fill_missing()`, `outlier()`, `ranks()`, `shift()`, `smooth_cases()`, `truncate_phase()`

Examples

```
## Standardize a multiple case scdf and compute an hplm
ex_sc <- standardize(exampleAB_50, var = "values", center = TRUE, scale = TRUE)
hplm(ex_sc)
```

---

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

```
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 `style_plot(style = c("grid2","tiny"))` where settings of a latter style overwrite settings of the former. Additional style paramters are set following the style argument and can be combined with those: `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 seperating 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")
```

subset.scdf *Subset cases, rows, and variables*

Description

Subset cases, rows, and variables

Usage

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

Arguments

`x`	An scdf object.
`filter`	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

```
subset(exampleAB, (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 functio for a scdf*

Description

Summary functio for a scdf

Usage

```
## S3 method for class 'scdf'
summary(object, ...)
```

Arguments

`object`	scdf
`...`	not in use

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

Description

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

Usage

```
tau_u(
  data,
  dvar,
  pvar,
  tau_method = "b",
  method = "complete",
  phases = c(1, 2),
  meta_method = "random",
  ci = 0.95,
  continuity_correction = 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.
`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("A","B")`.
`meta_method`	Character string. If set "random", a random-effect meta-analysis is calculated. If set "fixed", a fixed-effect meta-analysis is calculated. If set "none", no meta-analysis is conducted (may be helpful to speed up analyses).
`ci`	Confidence interval for meta analyzes.
`continuity_correction`	If TRUE, a continuity correction is applied for calculating p-values of correlations. This parameter is not yet implemented.

Details

Tau-U is an inconsistently operationalized construct. Parker et al. (2011) describe a method which may result in Tau-U lager than 1. 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 metaanalyzes to gain the overall values. Each tau value is weighted by the inverse of the variance (ie. the tau standard error). Tau values are not converted to Pearson r values. The argument `"meta_method"` calculates a random-effect model ("random") or a fixed effect model ("fixed").

Value

`table`	A data frame containing statistics from the Tau-U family, including: Pairs, positive and negative comparisons, S, and Tau
`matrix`	The matrix of comparisons used for calculating the statistics.
`tau_u`	Tau-U value.

Author(s)

Juergen Wilbert

References

Brossart, D. F., Laird, V. C., & Armstrong, T. W. (2018). Interpreting Kendall's Tau and Tau-U for single-case experimental designs. *Cogent Psychology, 5(1)*, 1–26. https://doi.org/10.1080/23311908.2018.1518687.

Parker, R. I., Vannest, K. J., Davis, J. L., & Sauber, S. B. (2011). Combining Nonoverlap and Trend for Single-Case Research: Tau-U. *Behavior Therapy, 42*, 284-299.

See Also

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

Examples

```
tau_u(Grosche2011$Eva)

## 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)
```

transform.scdf *Transform variables in every single case of a single case data frame*

Description

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

Usage

```
## S3 method for class 'scdf'
transform(`_data`, ...)
```

Arguments

_data	A scdf.
...	Expressions.

Details

This function is a method of the generic transformation function. Unlike this one, expressions are calculated serially. This means that the results of the calculation of an expression are the basis for the following calculations.

Value

A scdf.

Examples

```
## 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",
 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)
  )
```

`trend` *Trend analysis for single-cases data*

Description

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

Usage

```
trend(data, dvar, pvar, mvar, offset = -1, 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`	An offset for the first measurement-time of each phase (MT). If `offset = 0`, the phase measurement is handled as MT 1. Default is `offset = -1`, setting the first value of MT to 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`, `autocorr`, `plm`

Examples

```
## 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)))
```

---

truncate_phase *Truncate single-case data*

Description

This function truncates data points at the beginning and / or end of each phase in each case.

Usage

```
truncate_phase(
  data,
  dvar,
  pvar,
  truncate = list(A = c(0, 0), B = c(0, 0)),
  na = 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.
`truncate`	A list with a vector of two (beginning and end) values for each phase defining the number of data points to be deleted. For lists of single-case data frames, the truncation is adapted to the length of each phase for each single case.
`na`	If FALSE, the truncated measurement times are deleted. If TRUE, NAs are set for the dependent variable.

Value

A truncated data frame (for each single-case).

Author(s)

Juergen Wilbert

See Also

Other data manipulation functions: `as.data.frame.scdf()`, `fill_missing()`, `outlier()`, `ranks()`, `shift()`, `smooth_cases()`, `standardize()`

Examples

```
# Truncate the first two data points of both phases and compare the two data sets
study <- c(
  "Original" = byHeart2011[1],
  "Selected" = truncate_phase(byHeart2011[1], truncate = list(A = c(2, 0), B = c(2, 0)))
)
plot(study)
```

---

write_scdf — *Write data into a .csv-file*

Description

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

Usage

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

writeSC(...)
```

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 just a simple wrapper for ‘write_csv(as.data.frame(data), filename, row.names = FALSE)‘.

Author(s)

Juergen Wilbert

See Also

write.table, read_scdf, saveRDS

Examples

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

## 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)
```

$.scdf *Select a scdf*

Description

Select a scdf

Usage

```
## S3 method for class 'scdf'
x$i

## S3 method for class 'scdf'
x[i]
```

Arguments

x	A scdf object
i	A case name from x

Value

A scdf

`%>%` *Pipe*

Description

Several functions in scan are designed to work with pipes `%>%`. This pipe function is directly imported from magrittr.

Details

Since R 4.1 a pipe operator |> is included in base R. This pipe operator can be used along with scan perfectly fine.

Index