Package 'scan'

May 9, 2026

Type Package

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

Version 0.68.1

Date 2026-04-27

Depends R (>= 4.1.0)

Imports stats, nlme, MCMCglmm, utils, methods, graphics, car, gt, knitr, kableExtra, readxl, magrittr, miniUI, rstudioapi, rlang, cli

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

Description A collection of procedures for analysing, visualising, and managing single-case data. Multi-phase and multi-baseline designs are supported. Analysing methods include regression models (multilevel, multivariate, bayesian), between case standardised mean difference, overlap indices ('PND', 'PEM', 'PAND', 'NAP', 'PET', 'tau-u', 'IRD', 'baseline corrected tau', 'CDC'), and randomization tests. Data preparation functions support outlier detection, handling missing values, scaling, and custom transformations. An export function helps to generate html, word, and latex tables in a publication friendly style. A shiny app allows to use scan in a graphical user interface.
More details can be found in the online book 'Analyzing single-case data with R and scan', Juergen Wilbert (2026) <https://jazznbass.github.io/scan-Book/>.

License GPL (>= 3)

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

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

LazyData true

Encoding UTF-8

RoxygenNote 7.3.3

Config/testthat/edition 3

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2026-04-28 05:10:02 UTC

Contents

add_dummy_variables . . . 3
add_l2 . . . 4
anova.sc_plm . . . 5
as.data.frame.scdf . . . 7
as_scdf . . . 8
batch_apply . . . 9
between_smd . . . 10
bplm . . . 12
coef.sc_plm . . . 15
combine . . . 16
convert . . . 17
describe . . . 18
design . . . 19
estimate_design . . . 22
example_data_sets . . . 24
export . . . 26
fetch . . . 29
fill_missing . . . 30
hplm . . . 32
import_scdf . . . 36
ird . . . 37
is.scdf . . . 38
moving_median . . . 39
mplm . . . 42
na.omit.scdf . . . 44
nap . . . 45
overlap . . . 47
pand . . . 48
pem . . . 51
pet . . . 53
plm . . . 54
plot.scdf . . . 59
plot_rand . . . 62
pnd . . . 62
power_test . . . 64

print.scdf . . . 66
print.sc_ac . . . 67
print.sc_bctau . . . 68
print.sc_cdc . . . 70
print.sc_outlier . . . 73
print.sc_rand . . . 75
random_scdf . . . 79
rci . . . 81
read_scdf . . . 82
rescale . . . 84
sample_names . . . 85
scdf . . . 85
select_cases . . . 88
select_phases . . . 89
set_vars . . . 90
shinyscan . . . 91
smd . . . 92
style_plot . . . 93
subset.scdf . . . 95
summary.scdf . . . 96
tau_u . . . 97
trend . . . 100
write_scdf . . . 102

Index **104**

`add_dummy_variables` *Add Dummy Variables for Piecewise Linear Models*

Description

Adds dummy variables to an scdf for calculating piecewise linear models.

Usage

```
add_dummy_variables(
  scdf,
  model = c("W", "H-M", "B&L-B"),
  contrast_level = c("first", "preceding"),
  contrast_slope = c("first", "preceding")
)
```

Arguments

`scdf`	A single-case data frame. See `scdf()` to learn about this format.
`model`	Model used for calculating the dummy parameters (see Huitema & McKean, 2000). Default is `model = "W"`. Possible values are: `"B&L-B"`, `"H-M"`, `"W"`, and deprecated `"JW"`.

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

Details

This function creates dummy variables for phase levels and phase slopes according to the specified piecewise regression model. It supports different contrast coding schemes for both level and slope contrasts.

Examples

```
add_dummy_variables(
 scdf = exampleABC,
 model = "W",
 contrast_level = "first",
 contrast_slope = "first"
)
```

---

add_l2 *Add level-2 data to an scdf*

Description

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

Usage

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

Arguments

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

Details

This function is mostly used in combination with the `hplm()` function. It adds level-2 variables to each single-case data frame in an scdf based on matching case names.

Value

An scdf with added level-2 variables.

Author(s)

Juergen Wilbert

See Also

hplm()

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

Examples

```
## Example with the default case variable name 'case'
Leidig2018 |> add_l2(Leidig2018_l2)
## Example with a different case variable name in the L2 data
Leidig2018_l2_renamed <- Leidig2018_l2
names(Leidig2018_l2_renamed)[2] <- "subject"
Leidig2018 |> add_l2(Leidig2018_l2_renamed, cvar = "subject")
```

---

anova.sc_plm *ANOVA Table for Piecewise Linear Models*

Description

Model comparison for piecewise regression models fitted with plm(), hplm(), or mplm() using likelihood ratio tests.

Usage

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

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

## S3 method for class 'sc_mplm'
anova(object, ...)
```

Arguments

object	An object containing the results returned by plm(), hplm(), or mplm().
...	additional objects for model comparison.

Details

The function performs likelihood ratio tests to compare nested piecewise regression models. It extracts the underlying model from the sc_plm, sc_hplm, or sc_mplm object and passes them to the generic anova() function for model comparison.

Value

An object of class anova containing the results of the model comparison.

Examples

```
## For glm models with family = "gaussian"
mod1 <- plm(exampleAB$Johanna, level = FALSE, slope = FALSE)
mod2 <- plm(exampleAB$Johanna)
anova(mod1, mod2)
## For glm models with family = "poisson"
mod0 <- plm(example_A24, formula = injuries ~ 1, family = "poisson")
mod1 <- plm(example_A24, trend = FALSE, family = "poisson")
anova(mod0, mod1, mod2)
## For glm with family = "binomial"
mod0 <- plm(
  exampleAB_score$Christiano,
  formula = values ~ 1,
  family = "binomial",
  var_trials = "trials"
)
mod1 <- plm(
  exampleAB_score$Christiano,
  trend = FALSE,
  family = "binomial",
  var_trials = "trials"
)
anova(mod0, mod1)
## For multilevel models:
mod0 <- hplm(Leidig2018, trend = FALSE, slope = FALSE, level = FALSE)
mod1 <- hplm(Leidig2018, trend = FALSE)
mod2 <- hplm(Leidig2018)
anova(mod0, mod1, mod2)
## For mplm
mod0 <- mplm(
  Leidig2018$`1a1`,
  update = . ~  1, dvar = c("academic_engagement", "disruptive_behavior")
)
mod1 <- mplm(
  Leidig2018$`1a1`,
  trend = FALSE,
  dvar = c("academic_engagement", "disruptive_behavior")
)
mod2 <- mplm(
  Leidig2018$`1a1`,
  dvar = c("academic_engagement", "disruptive_behavior")
)

anova(mod0, mod1, mod2)
```

`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. Defaults to "case".

Value

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

Author(s)

Juergen Wilbert

See Also

Other data manipulation functions: `add_l2()`, `as_scdf()`, `batch_apply()`, `fill_missing()`, `moving_median()`, `print.sc_outlier()`, `ranks()`, `rescale()`, `scdf()`, `select_cases()`, `set_vars()`, `shift()`, `smooth_cases()`, `standardize()`, `truncate_phase()`

Examples

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

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

---

as_scdf *as_scdf*

Description

Converts a data frame to an scdf object.

Usage

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

Arguments

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

Value

An scdf.

See Also

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

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

Description

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

Usage

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

Arguments

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

Details

If simplify is TRUE and the function returns a vector of values, the output is combined into a data frame with case names as additional columns.

This is particularly useful for applying statistical models or summary statistics to each case in an scdf.

Value

A list of the output of each function call.

Author(s)

Juergen Wilbert

See Also

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

Examples

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

between_smd *Between-Case Standardized Mean Difference*

Description

Calculates a standardized mean difference from a multilevel model as described in Pustejovsky et al. (2014)

Usage

```
between_smd(
  data,
  method = c("REML", "MCMCglmm"),
  ci = 0.95,
  include_residuals = TRUE,
  ...
)

## S3 method for class 'sc_bcsmd'
print(x, digits = 2, ...)

## S3 method for class 'sc_bcsmd'
export(
  object,
  caption = NA,
  footnote = NA,
  filename = NA,
  digits = 2,
  round = 2,
  ...
)
```

Arguments

data	Either an scdf or an object returned from the hplm() or bplm() function.
method	Either "REML" or "MCMglmm". This indicated which statistical method is applied to calculate the model.
ci	A numeric between 0 and 1 setting the width of the confidence interval (when method is REML) or the credible interval (when method is MCMCglmm). The default is 0.95 for a 95-percent interval.
include_residuals	Logical. See details.
...	Further arguments passed to the hplm() or bplm()function.
x	An object returned by baseline_smd().
digits	The minimum number of significant digits to be use. If set to "auto" (default), values are predefined.

object	An scdf or an object exported from a scan function.
caption	Character string with table caption. If left NA (default) a caption will be created based on the exported object.
footnote	Character string with table footnote. If left NA (default) a footnote will be created based on the exported object.
filename	String containing the file name. If a filename is given the output will be written to that file.
round	Integer passed to the digits argument used to round values.

Details

The BC-SMD is calculate as `BC-SMD = Phase difference / sqrt(residual + random_intercept)`. This is most closely related to Cohen's *d*. If you want to have the most exact estimation based on the between case variance, you have to exclude the residual variance by setting the argument `include_residuals = FALSE` you get `BC-SMD = Phase difference / sqrt(random_intercept)`. The 'base' model only includes the phase level as a predictor like originally proposed by Hedges et al. Whereas the 'Full plm' model includes the trend and the phase slope as additional predictors.

Value

An object of class sc_bcsmd. It is a list containing the following elements:

- `models`: A list of data frames containing the BC-SMD results for each model calculated.
- `ci`: The width of the confidence/credible interval.
- `method`: The method used for model estimation.

Functions

- `print(sc_bcsmd)`: Print results
- `export(sc_bcsmd)`: export results

Author(s)

Juergen Wilbert

References

Pustejovsky, J. E., Hedges, L. V., & Shadish, W. R. (2014). Design-Comparable Effect Sizes in Multiple Baseline Designs: A General Modeling Framework. Journal of Educational and Behavioral Statistics, 39(5), 368–393. https://doi.org/10.3102/1076998614547577

Examples

```
## Create a example scdf:
des <- design(
  n = 150,
  phase_design = list(A1 = 10, B1 = 10, A2 = 10, B2 = 10, C = 10),
  level = list(B1 = 1, A2 = 0, B2 = 1, C = 1),
```

```
  rtt = 0.7,
  random_start_value = TRUE
)
study <- random_scdf(des)

## Standard BC-SMD return:
between_smd(study)

## Specify the model and provide an hplm object:
model <- hplm(study, contrast_level = "preceding", slope = FALSE,  trend = FALSE)
between_smd(model)

## excluding the residuals gives a more accurate estimation:
between_smd(model, include_residuals = FALSE)
```

---

bplm *Bayesian Piecewise Linear Model (bplm)*

Description

Computes a bayesian (hierarchical) piecewise linear model based on a Markov chain Monte Carlo sampler. The function automatically creates the fixed and random part of the regression model.

Usage

```
bplm(
  data,
  dvar,
  pvar,
  mvar,
  model = c("W", "H-M", "B&L-B"),
  contrast_level = c("first", "preceding"),
  contrast_slope = c("first", "preceding"),
  trend = TRUE,
  level = TRUE,
  slope = TRUE,
  random_trend = FALSE,
  random_level = FALSE,
  random_slope = FALSE,
  fixed = NULL,
  random = NULL,
  update_fixed = NULL,
  ...
)

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

```
## S3 method for class 'sc_bplm'
export(
  object,
  caption = NA,
  footnote = NA,
  filename = NA,
  round = 2,
  nice = 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.
model	Model used for calculating the dummy parameters (see Huitema & McKean, 2000). Default is model = "W". Possible values are: "B&L-B", "H-M", "W", and deprecated "JW".
contrast_level	Either "first", "preceding" or a contrast matrix. If NA contrast_level is a copy of contrast.
contrast_slope	Either "first", "preceding" or a contrast matrix. If NA contrast_level is a copy of contrast.
trend	A logical indicating if a trend parameters is included in the model.
level	A logical indicating if a level parameters is included in the model.
slope	A logical indicating if a slope parameters is included in the model.
random_trend	If TRUE, includes a random trend effect.
random_level	If TRUE, includes a random level effect.
random_slope	If TRUE, includes a random slope effect.
fixed	A formula that overwrites the automatically created fixed part of the regression model that 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 L1 or L2 variables into the regression model.
random	A formula that overwrites the automatically created random part of the regression model.
update_fixed	An easier way to change the fixed model part (e.g., . ~ . + newvariable).

...	Further arguments passed to the MCMCglmm::MCMCglmm() function.
x	An object returned by bplm()
digits	The minimum number of significant digits to be use. If set to "auto" (default), values are predefined.
object	An scdf or an object exported from a scan function.
caption	Character string with table caption. If left NA (default) a caption will be created based on the exported object.
footnote	Character string with table footnote. If left NA (default) a footnote will be created based on the exported object.
filename	String containing the file name. If a filename is given the output will be written to that file.
round	Integer passed to the digits argument used to round values.
nice	If set TRUE (default) output values are rounded and optimized for publication tables.

Details

The function uses the MCMCglmm::MCMCglmm() function to fit the model. The default model includes fixed trend, level, and slope effects as well as a random intercept for each single-case. The fixed part of the model can be changed by providing a custom formula to the fixed argument or by using the update_fixed argument. The random part of the model can be changed by providing a custom formula to the random argument or by setting the random_trend, random_level, or random_slope arguments to TRUE.

Value

An object of class sc_bplm with element:

- model - List containing information about the applied model.
- N - Number of single-cases.
- formula - A list containing the fixed and the random formulas of the hplm model.
- mcmglmm - Object of class MCMglmm.
- contrast - List with contrast definitions.

Functions

- print(sc_bplm): Print results
- export(sc_bplm): Export results as html table (see export())

Author(s)

Juergen Wilbert

See Also

Other regression functions: fetch(), hplm(), mplm(), plm(), print.sc_ac(), print.sc_bctau(), trend()

Examples

```
# plm regression
bplm(example_A24)

# Multilevel plm regression with random intercept
bplm(exampleAB_50, nitt = 5000)

# Adding a random slope
bplm(exampleAB_50, random_level = TRUE, nitt = 5000)

# Custom fixed formula
bplm(exampleAB_50, update_fixed = values ~ -1 + mt + phaseB +
  interB, nitt = 5000)
```

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

Description

Extract coefficients from plm/hplm objects

Usage

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

Arguments

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

Value

data frame with coefficient table

Examples

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

combine *Combine single-case data frames into one scdf*

Description

Combines several single-case data frames (scdf) into one scdf object.

Usage

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

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

Arguments

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

Value

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

Author(s)

Juergen Wilbert

convert *Convert scdf to R code*

Description

Converts an scdf object into R code that can be used to recreate the object.

Usage

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

Arguments

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

Details

This function is useful for exporting scdf objects for sharing or documentation purposes. The generated R code can be sourced to recreate the original scdf object.

Value

Returns a string (invisible).

Author(s)

Juergen Wilbert

See Also

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

Examples

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

`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

It computes the number of measurements, number of missing values, mean, median, standard deviation, median average deviation, minimum, maximum, and trend (slope of dependent variable regressed on measurement-time) for each phase of each single-case included in an scdf.

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 for multiple random single-cases*

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 = c("normal", "gaussian", "poisson", "binomial"),
  random_start_value = FALSE,
  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_design = list(A1 = 10, B1 = 10, A2 = 10, B2 = 10). Use vectors if you want to define different values for each case phase_design = 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\ 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".

random_start_value
: If TRUE, the start_values are randomized based on the distribution.

n_trials
: If distribution (see below) is "binomial", n_trials is the number of trials/observations/items. E.g., if you simulate accuracy data with 10 items per measurement, set n_trials = 10. To assign different n_trials to several single-cases, use a vector of values (e.g. n_trials = c(10, 15, 20)). If the number of cases exceeds the length of the vector, values are recycled.

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)
```

`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. If s is provided, this value is used.
rtt	The reliability of the measurements. The reliability is estimated when rtt = NULL. If rtt is provided, this value is used for all single-cases.
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. Default is FALSE.
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. Default is TRUE.
model	Model used for calculating the dummy parameters (see Huitema & McKean, 2000). Default is model = "W". Possible values are: "B&L-B", "H-M", "W", and deprecated "JW".
...	Further arguments passed to the plm function used for parameter estimation. For example, the model argument can be used to select different piecewise regression models. See plm for details.

Details

The function uses the plm function to estimate parameters for each single-case. If more than two single-cases are included in the scdf, the between case variance depicting the overall performance (s) is estimated unless s is provided. The reliability of the measurements (rtt) is estimated for each case unless rtt is provided. If overall_rtt is set to TRUE, rtt estimations will be based on all cases and identical for each case. If overall_effects is set to TRUE, trend, level, and slope effect estimations will be identical for each case.

The resulting design object can be used as input for the random_scdf function to create new random scdf files based on the estimated parameters. This allows to create bootstrap samples or Monte-Carlo datasets based on the characteristics of an existing dataset.

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. This list can be used as input for the random_scdf function to create new random scdf files based on the estimated parameters.

Author(s)

Juergen Wilbert

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)
```

`example_data_sets` *Single-case example data sets*

Description

The scan package comes with a set of fictitious and authentic single-case study data. These data sets can be used to practice single-case data analysis and to reproduce results from the respective publications.

- **Beretvas2008** — Fictitious single-case intervention study from Beretvas & Chung, 2008.
- **Borckardt2014** — Fictitious daily pain ratings evaluating a psychological treatment from Borckardt & Nash, 2014.
- **byHeart2011** — Multiple-baseline (11 cases) flash card vocabulary learning (Wilbert, unpublished).
- **example_A24** — Number of injuries on a German autobahn before and after implementation of a speedlimit (130km/h) (Ministerium fuer Infrastruktur und Landesplanung. Land Brandenburg).
- **exampleA1B1A2B2** — Fictitious A1-B1-A2-B2 example dataset.
- **exampleA1B1A2B2_zvt** — Non-fictitious A1-B1-A2-B2 example with ZVT (intelligence measure) and D2 (concentration measure) scores.
- **exampleAB** — Fictitious AB example dataset with three cases.
- **exampleAB_50** — Fictitious AB example dataset (50 cases)s.
- **exampleAB_50.l2** — Level-2 data for **exampleAB_50**.
- **exampleAB_add** — Fictitious AB example with added covariates.

- **exampleAB_decreasing** — Fictitious AB example with an expected decreasing effect in the intervention phase.
- **exampleAB_mpd** — Fictitious example with different phase structures for each case.
- **exampleAB_score** — Fictitious AB example with binomial distributed score outcome.
- **exampleAB_simple** — Simple fictitious AB example with three cases.
- **exampleABAB** — Fictitious ABAB reversal design example.
- **exampleABC** — Fictitious ABC example dataset.
- **exampleABC_150** — Fictitious ABC example (150 cases).
- **exampleABC_50** — Fictitious ABC example (50 cases).
- **exampleABC_outlier** — Fictitious ABC example with outlier.
- **example_atd** — Fictitious AB alternating treatment design.
- **example_stranger** - Example for screen time of Stranger Things characters.
- **Grosche2011** — Multiple-baseline (three cases) from a direct-instructive reading intervention (Grosche, 2011).
- **Grosche2014** — Multiple-baseline (3×3 materials) reading intervention (Grosche, Lueke, & Wilbert, unpublished).
- **GruenkeWilbert2014** — Multiple-baseline (six cases) from a story mapping reading intervention (Gruenke, Wilbert, & Stegemann-Calder, 2013).
- **Huber2014** — Multiple-baseline (four cases) with DBR ratings from a behavioral compliance intervention (Huber, unpublished).
- **Huitema2000** — Fictitious single-case intervention study (Huitema & McKean, 2000).
- **Lenz2013** — Fictitious example (Lenz, 2013).
- **Leidig2018** — Multiple-baseline good behavior game intervention (Leidig et al., 2022).
- **Leidig2018_l2** — Level-2 data for **Leidig2018** (Leidig et al., 2022).
- **Parker2007** — Example dataset after Parker et al. (2007).
- **Parker2009** — Example dataset after Parker et al. (2009).
- **Parker2009b** — Example dataset after Parker & Vannest (2009).
- **Parker2011** — Example dataset after Parker et al. (2011).
- **Parker2011b** — Example from Parker, Vannest, & Davis (2011).
- **SSDforR2017** — Example from the R package SSDforR.
- **Tarlow2017** — Fictitious single-case intervention study (Tarlow, 2017).
- **Waddell2011** — Fictitious single-case intervention study (Waddell, Nassar, & Gustafson, 2011).

Usage

```
data(exampleAB, package = "scan")
```

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.

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.

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.

Grosche, M. (2011). Effekte einer direkt-instruktiven Foerderung der Lesegenauigkeit. *Empirische Sonderpaedagogik, 3*, 147–161.

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

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.

Leidig, T., Casale, G., Wilbert, J., Hennemann, T., Volpe, R. J., Briesch, A., & Grosche, M. (2022). Individual, generalized, and moderated effects of the good behavior game on at-risk primary school students: A multilevel multiple baseline study using behavioral progress monitoring. *Frontiers in Education, 7*. https://www.frontiersin.org/articles/10.3389/feduc.2022.917138

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(4), 194-204.

Parker, R. I., Vannest, K. J., & Brown, L. (2009). The improvement rate difference for single-case research. Exceptional Children, 75(2), 135-150.

Parker, R. I., & Vannest, K. (2009). An improved effect size for single-case research: Nonoverlap of all pairs. Behavior Therapy, 40(4), 357-367.

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

Parker, R. I., Vannest, K. J., & Davis, J. L. (2011). Effect Size in Single-Case Research: A Review of Nine Nonoverlap Techniques. Behavior Modification, 35(4), 303-322. https://doi.org/10.1177/0145445511399147

Tarlow, K. R. (2017). 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

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.

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

Description

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

Usage

```
export(object, ...)

## S3 method for class 'sc_desc'
export(
  object,
  caption = NA,
  footnote = NA,
  filename = NA,
  flip = FALSE,
  round = 2,
  ...
)

## S3 method for class 'sc_nap'
export(
  object,
  caption = NA,
  footnote = NA,
  filename = NA,
  select = c("Case", "NAP", "NAP Rescaled", "w", "p", "d", "R²"),
  round = 2,
  ...
)

## S3 method for class 'sc_overlap'
export(
  object,
  caption = NA,
  footnote = NULL,
  filename = NA,
  round = 2,
  decimals = 2,
  flip = FALSE,
  ...
)

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

## S3 method for class 'sc_pet'
export(object, caption = NA, footnote = NA, filename = NA, round = 1, ...)

## S3 method for class 'sc_pnd'
export(
  object,
  caption = NA,
  footnote = NA,
```

```
  filename = NA,
  select = c("Case", "PND", "Total", "Exceeds"),
  round = 2,
  ...
)

## S3 method for class 'sc_power'
export(object, caption = NA, footnote = NA, filename = NA, round = 3, ...)

## S3 method for class 'sc_smd'
export(
  object,
  caption = NA,
  footnote = NA,
  filename = NA,
 select = c("Case", `Mean A` = "mA", `Mean B` = "mB", `SD A` = "sdA", `SD B` = "sdB",
   `SD Cohen` = "sd cohen", `SD Hedges` = "sd hedges", "Glass' delta", "Hedges' g",
    "Hedges' g correction", "Hedges' g durlak correction", "Cohen's d"),
  round = 2,
  decimals = 2,
  flip = FALSE,
  ...
)

## S3 method for class 'sc_trend'
export(
  object,
  caption = NA,
  footnote = NA,
  filename = NA,
  round = 3,
  decimals = NULL,
  ...
)

## S3 method for class 'scdf'
export(
  object,
  summary = FALSE,
  caption = NA,
  footnote = NA,
  filename = NA,
  cols,
  round = 2,
  ...
)

## S3 method for class 'scdf_summary'
```

```
export(object, caption = NA, footnote = NA, filename = NA, round = 2, ...)
```

Arguments

object	An scdf or an object exported from a scan function.
...	Further Arguments passed to internal functions.
caption	Character string with table caption. If left NA (default) a caption will be created based on the exported object.
footnote	Character string with table footnote. If left NA (default) a footnote will be created based on the exported object.
filename	String containing the file name. If a filename is given the output will be written to that file.
flip	If TRUE, some objects are exported with rows and columns flipped.
round	Integer passed to the digits argument used to round values.
select	A character vector containing the names of the variables to be included. If the vector is named, the variables will be renamed accordingly.
decimals	Decimal places that are reported.
summary	If TRUE, exports the summary of an scdf.
cols	Defines which columns are included when exporting an scdf. It is either a vector of variable names or the string "main" will select the central variables.

Details

The function uses either the kableExtra or the gt package to create the tables. The default engine can be set via the option `scan.export.engine`. Additional options for kable and kable_styling can be set via the options `scan.export.kable` and `scan.export.kable_styling`. The default options can be viewed and modified via `options("scan.export.kable")` and `options("scan.export.kable_styling")`.

Value

Returns or displays a specially formatted html (or latex) file.

---

fetch *Fetches elements from scan objects*

Description

The fetch function is a getter function for scan objects returned from regression functions such as `plm()`, `hplm()`, `bplm()`, and `mplm()`. It allows users to extract specific elements from these objects, such as the fitted model.

Usage

```
fetch(object, what, ...)
```

Arguments

object	Object returned from a scan function.
what	Element/part to be extracted. Currently, only "model" is supported to extract the fitted regression model.
...	Further parameters passed to the function.

Value

An object of the respective regression model class.

Author(s)

Juergen Wilbert

See Also

Other regression functions: bplm(), hplm(), mplm(), plm(), print.sc_ac(), print.sc_bctau(), trend()

Examples

```
# plm regression
model1 <- plm(example_A24)
fetch(model1, what = "model") |> summary()
# Multilevel plm regression
model2 <- hplm(exampleAB_50)
fetch(model2, what = "model") |> summary()
# Bayesian plm regression
model3 <- bplm(exampleAB_50, nitt = 5000)
fetch(model3, what = "model") |> summary()
```

---

fill_missing *Replacing missing measurement points in single-case data*

Description

The fillmissing() function replaces missing measurements in single-case data. It linearly interpolates missing data points between two existing measurements for all variables except the measurement time and phase. The measurement time variable is filled with the missing time points. The phase variable is copied from the previous measurement time point. If mt values are missing (NA), they are also interpolated if interpolate_na = TRUE.

Usage

```
fill_missing(data, dvar, mvar, pvar, interpolate_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.
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.
interpolate_na	If set TRUE, NA values in the measurement time variable are also interpolated. Default is TRUE.

Details

The fill_missing() function is designed to handle single-case data with missing measurement points. It performs linear interpolation to estimate the missing values based on the existing data points. The function iterates through each single-case in the provided single-case data frame (scdf) and identifies gaps in the measurement time variable. For each gap, it calculates the step size for linear interpolation and fills in the missing values for all target variables (i.e., all variables except the measurement time and phase). The interpolated data points are then added to the single-case data frame, and the final result is sorted by measurement time. This function is particularly useful for preparing single-case data for further analysis, such as calculating overlap indices or conducting randomization tests, where continuous measurement times are required. It ensures that the data is complete by filling in the missing measurement points in a systematic manner.

Value

A single-case data frame with interpolated missing data points.

Author(s)

Juergen Wilbert

See Also

Other data manipulation functions: add_l2(), as.data.frame.scdf(), as_scdf(), batch_apply(), moving_median(), print.sc_outlier(), ranks(), rescale(), scdf(), select_cases(), set_vars(), 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")
```

```
study

## An example with multiple interpolated variables

rolf_n <- exampleAB_add
rolf_n[[1]] <- rolf_n[[1]][-c(3,7,8),]
rolf_f <- fill_missing(rolf_n)
study1 <- c("original" = exampleAB_add, "interpolated" = rolf_f)
study1

## Example with missing NAs in measurement time
Maggie2 <- random_scdf(design(level = list(0,1)), seed = 123)
Maggie2_n <- Maggie2
Maggie2_n[[1]][c(5,12:14,20), "mt"] <- NA
Maggie2_f <- fill_missing(Maggie2_n)
study2 <- c("original" = Maggie2, "interpolated" = Maggie2_f)
study2
```

---

hplm *Hierarchical piecewise linear model / piecewise regression for multiple cases*

Description

The hplm() function computes a hierarchical piecewise regression model. It extends the standard piecewise regression model to multiple cases by estimating fixed and random effects. The function uses the lme function of the nlme package to fit linear mixed-effects models. The model can include random intercepts and random slopes for level, trend, and treatment effects. Additionally, it allows for the inclusion of autoregressive structures and unequal variances across phases. The function also provides options for likelihood ratio tests to compare models with and without random slope effects, as well as the calculation of intraclass correlations (ICC) to assess the proportion of variance attributable to between-case differences. This function is particularly useful for analyzing data from multiple single-case experimental designs (SCEDs) where observations are nested within cases.

Usage

```
hplm(
  data,
  dvar,
  pvar,
  mvar,
  model = c("W", "H-M", "B&L-B", "JW"),
  contrast = c("first", "preceding"),
  contrast_level = NA,
  contrast_slope = NA,
  method = c("ML", "REML"),
  control = list(opt = "optim"),
```

```
  random.slopes = FALSE,
  lr.test = FALSE,
  ICC = TRUE,
  trend = TRUE,
  level = TRUE,
  slope = TRUE,
  random_trend = FALSE,
  random_level = FALSE,
  random_slope = FALSE,
  fixed = NULL,
  random = NULL,
  ar = 0,
  unequal_variances = FALSE,
  update.fixed = NULL,
  data.l2 = NULL,
  ...
)

## S3 method for class 'sc_hplm'
print(x, digits = 3, bcsmd = FALSE, casewise = FALSE, ...)

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

## S3 method for class 'sc_hplm'
coef(object, casewise = 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.
model	Model used for calculating the dummy parameters (see Huitema & McKean, 2000). Default is model = "W". Possible values are: "B&L-B", "H-M", "W", and deprecated "JW".

contrast	Sets contrast_level and contrast_slope. Either "first", "preceding" or a contrast matrix. If NA contrast is ignored.
contrast_level	Either "first", "preceding" or a contrast matrix. If NA contrast_level is a copy of contrast.
contrast_slope	Either "first", "preceding" or a contrast matrix. If NA contrast_level is a copy of contrast.
method	Method used to fit your model. Pass "REML" to maximize the restricted log-likelihood or "ML" for maximized log-likelihood. Default is "ML".
control	A list of settings for the estimation algorithm, replacing the default values passed to the function lmeControl of the nlme package.
random.slopes	If random.slopes = TRUE random slope effects of the level, trend, and treatment parameter are estimated.
lr.test	If set TRUE likelihood ratio tests are calculated comparing model with vs. without random slope parameters.
ICC	If ICC = TRUE an intraclass-correlation is estimated.
trend	A logical indicating if a trend parameters is included in the model.
level	A logical indicating if a level parameters is included in the model.
slope	A logical indicating if a slope parameters is included in the model.
random_trend	If TRUE, includes a random trend trend effect.
random_level	If TRUE, includes a random level trend effect.
random_slope	If TRUE, includes a random slope trend effect.
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. Defaults to a random intercept model. The formula can be changed to include random slope effects for level, trend, and treatment effects.
ar	Maximal lag of autoregression. Modelled based on the Autoregressive-Moving Average (ARMA) function.
unequal_variances	Logical. If set TRUE, estimations are weighted by phase variances.
update.fixed	An easier way to change the fixed model part (e.g., . ~ . + newvariable).
data.l2	A data frame providing additional variables at Level 2. The scdf File has to have names for all cases and the Level 2 data frame has to have a column named 'cases' with the names of the cases the Level 2 variables belong to.
...	Further arguments passed to the lme function.
x	An object returned by hplm()
digits	The minimum number of significant digits to be use. If set to "auto" (default), values are predefined.

bcsmd	If TRUE, reports between-case standardized mean differences.
casewise	Returns the estimations for each case separately
object	An scdf or an object exported from a scan function.
caption	Character string with table caption. If left NA (default) a caption will be created based on the exported object.
footnote	Character string with table footnote. If left NA (default) a footnote will be created based on the exported object.
filename	String containing the file name. If a filename is given the output will be written to that file.
round	Integer passed to the digits argument used to round values.
nice	If set TRUE (default) output values are rounded and optimized for publication tables.

Value

An object of class sc_hplm.

- model | List containing infromation about
- N | Number of single-cases.
- formula | 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.
- contrast | List with contrast definitions.

Functions

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

Model specification

The fixed effects part of the model can be specified using the fixed argument, while the random effects part can be specified using the random argument. If not provided, default formulas based on the specified model type (e.g., "B&L-B") are created. The function also allows for the inclusion of autoregressive structures through the ar argument and unequal variances across phases through the unequal_variances argument.

Random slopes

By setting the random.slopes argument to TRUE, the model will include random slope effects for level, trend, and treatment effects. This allows for individual differences in how cases respond to these effects.

Likelihood ratio tests

If the `lr.test` argument is set to TRUE, the function will perform likelihood ratio tests to compare models with and without random slope effects. This helps to determine whether including random slopes significantly improves model fit.

Intraclass correlation

If the `ICC` argument is set to TRUE, the function will calculate the intraclass correlation coefficient (ICC) to assess the proportion of variance attributable to between-case differences. This provides insight into the degree of similarity among observations within the same case.

Author(s)

Juergen Wilbert

See Also

Other regression functions: `bplm()`, `fetch()`, `mplm()`, `plm()`, `print.sc_ac()`, `print.sc_bctau()`, `trend()`

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
Leidig2018 |>
  add_l2(Leidig2018_l2) |>
  hplm(update.fixed = .~. + gender + migration + ITRF_TOTAL*phaseB,
       slope = FALSE, random.slopes = FALSE, lr.test = FALSE
  )
```

---

import_scdf *Import scdf – RStudio Addin*

Description

Import scdf – RStudio Addin

Usage

```
import_scdf()
```

ird *IRD - Improvement rate difference*

Description

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

Usage

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

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

## S3 method for class 'sc_ird'
export(object, caption = NA, footnote = NA, filename = NA, round = 3, ...)
```

Arguments

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

Details

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

Functions

- `print(sc_ird)`: Print results
- `export(sc_ird)`: Export results to html

References

Parker, R. I., Vannest, K. J., & Brown, L. (2009). The improvement rate difference for single-case research. Exceptional Children, 75(2), 135-150.

Parker, R. I., Vannest, K. J., & Davis, J. L. (2011). Effect Size in Single-Case Research: A Review of Nine Nonoverlap Techniques. Behavior Modification, 35(4), 303-322. https://doi.org/10.1177/0145445511399147

Pustejovsky, J. E. (2019). Procedural sensitivities of effect sizes for single-case designs with directly observed behavioral outcome measures. *Psychological Methods*, *24(2)*, 217-235. https://doi.org/10.1037/met0000179

See Also

Other overlap functions: `nap()`, `overlap()`, `pand()`, `pem()`, `pet()`, `pnd()`, `print.sc_cdc()`, `select_phases()`, `tau_u()`

---

is.scdf *Test for scdf objects*

Description

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.

Author(s)

Juergen Wilbert

`moving_median` *Transform every single case of a single case data frame*

Description

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

Usage

```
moving_median(x, lag = 1)

moving_mean(x, lag = 1)

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

set_na_at(x, first_of, positions = 0)

center_at(x, at = TRUE, shift = 0, part = 0)

first_of(x, positions = 0)

across_cases(...)

all_cases(...)

rowwise(...)

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

Arguments

`x`	A logical vector.
`lag`	Number of values surrounding a value to calculate the average
`mt`	A vector with measurement times.
`f`	the proportion of surrounding data influencing each data point.
`first_of`	A logical vector
`positions`	A numeric vector with relative positions to the first appearance of a TRUE value in x.
`at`	A logical vector. E.g. `phase == "A"`. The first TRUE value of that vector is the target position for centring. By default, this is the first position.
`shift`	A value indicating a shift in measurement times for centring. E.g. `shift = 4` will centre four measurement-times after the position defined by the `at` and `part` arguments.

part	A numeric value between 0 and 1. 0 refers to the first TRUE in the at vector, 1 to the last, and 0.5 to the midpoint of the sequence of TRUE values. E.g. if you want to centre at the middle of phase A, set at = phase == A, part = 0.5. Note: decimals are rounded to integers.
...	Expressions.
_data	An scdf.

Details

This function is a method of the generic transform() function. Unlike the generic version, expressions are evaluated **serially**: the result of one expression is used as the basis for subsequent computations.

Several helper functions can be used inside expressions:

- n(): returns the number of measurements in a case.
- all_cases(): extracts the values of a variable across all cases. Takes an expression as argument. For example:
  - mean(all_cases(values)) calculates the mean of values across all cases.
  - mean(all_cases(values[phase == "A"])) calculates the mean of all values where phase == "A".
- rowwise(): applies a calculation separately to each row. Example: rowwise(sum(values, mt, na.rm = TRUE)).
- across_cases(): creates new variables or replaces existing ones across all cases. Example: across_cases(values_ranked = rank(values, na.last = "keep")) ranks the values variable across all cases and creates a new variable values_ranked.

Value

An scdf.

Author(s)

Juergen Wilbert

See Also

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

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)
  )

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

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

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

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

## Function first_of() helps to set NAs for specific phases.
## E.g., you want to replace the first two values of phase A and the first
## value of phase B and its preceding value.

byHeart2011 |>
  transform(
    values = set_na_at(values, phase == "A", 0:1),
    values = set_na_at(values, phase == "B", -1:0)
  )
```

mplm *Multivariate Piecewise linear model / piecewise regression*

Description

The mplm() function computes a multivariate piecewise regression model. The function automatically creates the regression formula based on the provided data and the selected options. The default model includes trend, level, and slope effects for each dependent variable. The regression formula can be changed by providing a custom formula to the update argument.

Usage

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

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

## S3 method for class 'sc_mplm'
export(
  object,
  caption = NA,
  footnote = NA,
  filename = NA,
  nice = TRUE,
  std = FALSE,
  decimals = 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.
mvar	Character string with the name of the measurement time variable. Defaults to the attributes in the scdf file.
pvar	Character string with the name of the phase variable. Defaults to the attributes in the scdf file.
model	Model used for calculating the dummy parameters (see Huitema & McKean, 2000). Default is `model = "W"`. Possible values are: `"B&L-B"`, `"H-M"`, `"W"`, and deprecated `"JW"`.
contrast	Sets contrast_level and contrast_slope. Either "first", "preceding" or a contrast matrix. If NA contrast is ignored.
contrast_level	Either "first", "preceding" or a contrast matrix. If NA contrast_level is a copy of contrast.
contrast_slope	Either "first", "preceding" or a contrast matrix. If NA contrast_level is a copy of contrast.
trend	A logical indicating if a trend parameters is included in the model.
level	A logical indicating if a level parameters is included in the model.
slope	A logical indicating if a slope parameters is included in the model.
formula	Defaults to the standard piecewise regression model. The parameter phase followed by the phase name (e.g., `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.
x	Object returned from `mplm()`.
digits	The minimum number of significant digits to be use. If set to "auto" (default), values are predefined.
std	If TRUE, a table with standardized estimates is included.
object	An scdf or an object exported from a scan function.
caption	Character string with table caption. If left NA (default) a caption will be created based on the exported object.
footnote	Character string with table footnote. If left NA (default) a footnote will be created based on the exported object.
filename	String containing the file name. If a filename is given the output will be written to that file.
nice	If set TRUE (default) output values are rounded and optimized for publication tables.
decimals	Decimal places that are reported.

Details

The function currently only supports single-case data (i.e., one case per dataset). For multilevel piecewise regression models, please use the hplm function.

Value

`model`	Character string from function call (see arguments above).
`contrast`	List with contrast definitions.
`full.model`	Full regression model list.
`formula`	Formula of the mplm model.

Functions

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

Author(s)

Juergen Wilbert

See Also

Other regression functions: `bplm()`, `fetch()`, `hplm()`, `plm()`, `print.sc_ac()`, `print.sc_bctau()`, `trend()`

Examples

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

`na.omit.scdf` *Remove missing values from scdf*

Description

Removes any row with a missing value from each single-case data frame in an scdf.

Usage

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

Arguments

object	A scdf.
...	not implemented yet.

Value

A scdf object.

Author(s)

Juergen Wilbert

---

nap *Nonoverlap of all Pairs (NAP)*

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* where ties are counted as half non-overlaps. Because NAP can take values between 0 and 100 percent where values below 50 percent indicate an inverse effect, an nap rescaled from -100 to 100 percent where negative values indicate an inverse effect is also displayed ($nap_{rescaled} = 2 * nap - 100$).

Usage

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

Arguments

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

Details

If a decrease of phase B scores is expected, set the argument `decreasing = TRUE`.

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: `ird()`, `overlap()`, `pand()`, `pem()`, `pet()`, `pnd()`, `print.sc_cdc()`, `select_phases()`, `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)

## Calculate NAP for phase 1 and phase 3 of an ABAB design
nap(exampleABAB, phases = c(1, 3))
```

`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(1,2)`.

Details

It computes PND, PEM, PET, NAP, PAND, IRD, Tau-U, mean difference, difference in trend, SMD, and Hedges-g for each single-case included in an scdf.

See corresponding functions of PND, PEM, PET, NAP, PAND for calculation. Tau_U(A) reports "A vs. B - Trend A" whereas Tau_U(BA) reports "A vs. B + Trend B - Trend A". Base_Tau is baseline corrected tau (correction applied when autocorrelation in phase A is significant). Diff_mean is the mean difference. Diff_trend is the difference in the regression estimation of the dependent variable on measurement-time (`x ~ mt`) for each phase. SMD is the mean difference divided by the standard deviation of phase A. Hedges_g is the mean difference divided by the pooled standard deviation: $\sqrt{\frac{(n_A-1)sd_A^2+(n_B-1)sd_B^2}{n_A+n_B-2}}$ with a hedges correction applied: $Hedges_g * (1 - \frac{3}{4n-9}))$.

Value

overlap	A data frame consisting of the following indices for each single-case for all cases: PND, PEM, PET, NAP, PAND
phases.A	Selection for A phase.
phases.B	Selection for B phase.
design	Phase design.

Author(s)

Juergen Wilbert

See Also

pnd(), pem(), pet(), nap(), pand(), ird(), tau_u(), corrected_tau()

Other overlap functions: ird(), nap(), pand(), pem(), pet(), pnd(), print.sc_cdc(), select_phases(), 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,
  phases = c(1, 2),
  method = c("sort", "minimum")
)

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

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

Arguments

data	A single-case data frame. See scdf() to learn about this format.
dvar	Character string with the name of the dependent variable. Defaults to the attributes in the scdf file.
pvar	Character string with the name of the phase variable. Defaults to the attributes in the scdf file.
decreasing	If you expect data to be lower in the B phase, set decreasing = TRUE. Default is decreasing = FALSE.
phases	A vector of two characters or numbers indicating the two phases that should be compared. E.g., phases = c("A","C") or phases = c(2,4) for comparing the second to the fourth phase. Phases could be combined by providing a list with two elements. E.g., phases = list(A = c(1,3), B = c(2,4)) will compare phases 1 and 3 (as A) against 2 and 4 (as B). Default is phases = c(1,2).
method	Either "sort"" or "minimum". See details.
x	An object returned by pand()
...	Further arguments passed to the function.
object	An scdf or an object exported from a scan function.
caption	Character string with table caption. If left NA (default) a caption will be created based on the exported object.
footnote	Character string with table footnote. If left NA (default) a footnote will be created based on the exported object.
filename	String containing the file name. If a filename is given the output will be written to that file.
round	Integer passed to the digits argument used to round values.

Details

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

$$PAND = \frac{1}{m+n} max\{(i+j)I(y_i^A < y_{n+1-j}^B\}$$

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

Value

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

Functions

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

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.

Pustejovsky, J. E. (2019). Procedural sensitivities of effect sizes for single-case designs with directly observed behavioral outcome measures. *Psychological Methods*, *24(2)*, 217-235. https://doi.org/10.1037/met0000179

Pustejovsky JE, Chen M, Swan DM (2023). SingleCaseES: A Calculator for Single-Case Effect Sizes. R package version 0.7.1.9999, https://jepusto.github.io/SingleCaseES/.

See Also

Other overlap functions: `ird()`, `nap()`, `overlap()`, `pem()`, `pet()`, `pnd()`, `print.sc_cdc()`, `select_phases()`, `tau_u()`

Examples

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

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

---

pem *Percent exceeding the median (PEM)*

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(1,2).
...	Additional arguments for the FUN parameter (e.g. FUN = mean, trim = 0.1 will use the 10 percent trimmed arithmetic mean instead of the median for comparisons). The function must take a vector of numeric values and the na.rm argument and return a numeric value.

Details

The Percent Exceeding the Median is calculated as the percentage of data points in phase B that exceed the median of phase A. If the decreasing argument is set to TRUE, the percentage of data points in phase B that are below the median of phase A is calculated. The PEM is expressed as a percentage ranging from 0 to 100. Higher values indicate a greater degree of improvement from phase A to phase B.

Author(s)

Juergen Wilbert

See Also

Other overlap functions: `ird()`, `nap()`, `overlap()`, `pand()`, `pet()`, `pnd()`, `print.sc_cdc()`, `select_phases()`, `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 (PET)*

Description

The `pet` function returns the percentage of Phase B data points that exceed the prediction based on the Phase A trend. A binomial test against a 50/50 distribution is calculated. It also calculates the percentage of Phase B data points that exceed the upper (or lower) 95 percent confidence interval of the predicted progression.

Usage

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

Arguments

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

Details

The PET is calculated by first fitting a linear model to the Phase A data to estimate the trend. Then, for each data point in Phase B, it is determined whether it exceeds the predicted value from the Phase A trend. The PET is the percentage of Phase B data points that exceed this predicted value. Additionally, a binomial test is performed to assess whether the observed PET is significantly greater than what would be expected by chance (i.e., 50%).

Value

`PET`	Percent exceeding the trend.
`ci`	Width of confidence interval.
`decreasing`	Logical argument from function call (see Arguments above).

Author(s)

Juergen Wilbert

See Also

Other overlap functions: `ird()`, `nap()`, `overlap()`, `pand()`, `pem()`, `pnd()`, `print.sc_cdc()`, `select_phases()`, `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) for one case. The function automatically creates the default model which includes trend, level, and slope effects. The model can be changed with `trend`, `level`, and `slope` arguments or by providing a custom formula.

Usage

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

## S3 method for class 'sc_plm'
print(
  x,
  lag_max = 3,
  ci = 0.95,
  q = FALSE,
  r_squared = getOption("scan.rsquared"),
  ...
)

## S3 method for class 'sc_plm'
export(
  object,
  caption = NA,
  footnote = NA,
  filename = NA,
  nice = TRUE,
  ci = 0.95,
  q = FALSE,
  round = 2,
  r_squared = getOption("scan.rsquared"),
  ...
)
```

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. Modelled 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 dummy parameters (see Huitema & McKean, 2000). Default is model = "W". Possible values are: "B&L-B", "H-M", "W", and deprecated "JW".
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.
contrast	Sets contrast_level and contrast_slope. Either "first", "preceding" or a contrast matrix. If NA contrast is ignored.
contrast_level	Either "first", "preceding" or a contrast matrix. If NA contrast_level is a copy of contrast.
contrast_slope	Either "first", "preceding" or a contrast matrix. If NA contrast_level is a copy of contrast.
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	Either "delta", "partial", or "none".
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 stats::glm() or nlme::gls() function.
x	Object

lag_max	Maximum lag to be reported for autocorrelation of residuals. Default is 3. Set FALSE for no report of autocorrelations.
ci	Print confidence intervals. Either FALSE, TRUE or a number between 0 and 1 (0.90 for a 90% intervals).
q	Logical. If set TRUE, Yule's Q is reported.
object	An scdf or an object exported from a scan function.
caption	Character string with table caption. If left NA (default) a caption will be created based on the exported object.
footnote	Character string with table footnote. If left NA (default) a footnote will be created based on the exported object.
filename	String containing the file name. If a filename is given the output will be written to that file.
nice	If set TRUE (default) output values are rounded and optimized for publication tables.
round	Integer passed to the digits argument used to round values.

Details

The function uses the glm function of the stats package or the gls function of the nlme package (for auto-regression models).

Value

An object of class sc_plm.

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.

Functions

- print(sc_plm): Print
- export(sc_plm): Export results as html table (see export())

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: `bplm()`, `fetch()`, `hplm()`, `mplm()`, `print.sc_ac()`, `print.sc_bctau()`, `trend()`

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, contrast = "preceding")

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

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

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

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

## A binomial regression (percentage as dependent variable)
exampleAB_score$Christiano |>
  transform(percentage = values/trials) |>
  set_dvar("percentage") |>
```

```
  plm(family = "binomial", var_trials = "trials", dvar_percentage = TRUE)
## Print
plm(exampleAB$Johanna) |>
  print(ci = 0.9, r_squared = c("delta", "partial"))
## Export
plm(exampleAB$Johanna) |> export()
```

`plot.scdf` *(Deprecated) 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\ 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\ `lines = list(type = "mean", trim = 0.2)` draws a 20\
- `"trend"` Separate lines for phase A and B trends.
- `"trendA"` OLS trend line for phase A, extrapolated throughout phase B.
- `"trendA_bisplit"` Split middle (bi-split) trend line for phase A, extrapolated throughout phase B.
- `"trendA_trisplit"` Tukey tri-split trend line for phase A, extrapolated throughout phase B.
- `"maxA/minA"` Line at the level of the highest or lowest phase A score.
- `"medianA"` Line at the phase A median score.
- `"meanA"` Line at the phase A 10\ using the additional argument (e.g., `lines = list(type = "meanA", trim = 0.2)`).
- `"plm"` Regression lines for piecewise linear regression model.
- `"plm.ar"` Regression lines for piecewise autoregression model. The lag is specified like this: `lines = list(type = "plm.ar", ar = 2)`. Default lag is set to 2.
- `"movingMean"` Draws a moving mean curve, with a specified lag: `lines = list(type = "movingMean", lag = 2)`. Default is a lag 1 curve.
- `"movingMedian"` Draws a moving median curve, with a specified lag: `lines = list(type = "movingMedian", lag = 3)`. Default is a lag 1 curve.
- `"loreg"` Draws a non-parametric local regression line. The proportion of data influencing each data point can be specified using `lines = list(type = "loreg"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

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,
  type = "hist",
  xlab = NULL,
  ylab = NULL,
  title = NULL,
  text_observed = "observed",
  color = "lightgrey",
  ...
)
```

Arguments

object	Object returned from the rand_test() function
type	'hist' or 'xy'.
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 (PND)*

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(1, 2))
```

Arguments

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

Details

PND is calculated by determining the number of data points in phase B that exceed the highest data point in phase A (or are lower than the lowest data point in phase A for decreasing data) divided by the total number of data points in phase B. This value is then multiplied by 100 to get a percentage value.

Value

PND	Percentage of non-overlapping data.

Author(s)

Juergen Wilbert

See Also

Other overlap functions: ird(), nap(), overlap(), pand(), pem(), pet(), print.sc_cdc(), select_phases(), 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 probability 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 = FALSE,
  binom_test_alpha = FALSE,
  binom_test_power = FALSE,
  binom_test_correct = FALSE,
  ci = FALSE,
  alpha_level = 0.05
)
```

Arguments

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

power_test — Logical. If TRUE, power is calculated.

binom_test — Shortcut. When set TRUE, binom_test_power is set to 0.80, binom_test_alpha is set to 0.05, and binom_test_correct is set to 0.875.

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

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

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

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

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

Details

Based on a `design()` object, a large number of single-cases are generated and re-analysed with a provided statistical function. The proportion of significant analyses 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 re-analysed. The proportion of significant analyses is the alpha error probability.

Value

A data frame with the power, alpha error, and correct proportions for each provided method. If binomial tests are requested, p values for these tests are also provided. If confidence intervals are requested, these are also provided.

Author(s)

Juergen Wilbert

See Also

`random_scdf()`, `design()`

Examples

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

```
power_test(design, n_sim = 10)

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

`print.scdf` *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)

print.sc_ac *Autocorrelation within and across phases*

Description

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

Usage

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

## S3 method for class 'sc_ac'
export(object, caption = NA, footnote = NA, filename = NA, round = 3, ...)

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

Arguments

x	An object returned by autocorr()
digits	The minimum number of significant digits to be use. If set to "auto" (default), values are predefined.
...	Further arguments passed to the acf() function.
object	An scdf or an object exported from a scan function.
caption	Character string with table caption. If left NA (default) a caption will be created based on the exported object.
footnote	Character string with table footnote. If left NA (default) a footnote will be created based on the exported object.
filename	String containing the file name. If a filename is given the output will be written to that file.
round	Integer passed to the digits argument used to round values.
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.

Details

Autocorrelations are computed using the `acf()` function from the stats package. For each single-case in the scdf object, a data frame is returned containing the autocorrelations for each phase and for all phases up to the specified lag.

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.

Functions

- `print(sc_ac)`: Print method
- `export(sc_ac)`: Export results to html

Author(s)

Juergen Wilbert

See Also

`acf()`

Other regression functions: `bplm()`, `fetch()`, `hplm()`, `mplm()`, `plm()`, `print.sc_bctau()`, `trend()`

Examples

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

---

print.sc_bctau *Baseline corrected tau*

Description

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

Usage

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

## S3 method for class 'sc_bctau'
export(
  object,
  caption = NA,
  footnote = NA,
```

```
  filename = NA,
  nice = TRUE,
  round = 2,
  ...
)

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

Arguments

x	An object returned by corrected_tau()
nice	If set TRUE (default) output values are rounded and optimized for publication tables.
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
object	An scdf or an object exported from a scan function.
caption	Character string with table caption. If left NA (default) a caption will be created based on the exported object.
footnote	Character string with table footnote. If left NA (default) a footnote will be created based on the exported object.
filename	String containing the file name. If a filename is given the output will be written to that file.
round	Integer passed to the digits argument used to round values.
data	A single-case data frame. See scdf() to learn about this format.
dvar	Character string with the name of the dependent variable. Defaults to the attributes in the scdf file.
pvar	Character string with the name of the phase variable. Defaults to the attributes in the scdf file.
mvar	Character string with the name of the measurement time variable. Defaults to the attributes in the scdf file.
phases	A vector of two characters or numbers indicating the two phases that should be compared. E.g., phases = c("A","C") or phases = c(2,4) for comparing the second to the fourth phase. Phases could be combined by providing a list with two elements. E.g., phases = list(A = c(1,3), B = c(2,4)) will compare phases 1 and 3 (as A) against 2 and 4 (as B). Default is phases = c(1,2).

alpha	Sets the p-value at and below which a baseline correction is applied.
continuity	If TRUE applies a continuity correction for calculating p
tau_method	Character with values "a" or "b" (default) indicating whether Kendall Tau A or Kendall Tau B is applied.

Details

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

Functions

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

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: `bplm()`, `fetch()`, `hplm()`, `mplm()`, `plm()`, `print.sc_ac()`, `trend()`

Examples

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

---

print.sc_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 labelled "systematic change".

Usage

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

## S3 method for class 'sc_cdc'
export(object, caption = NA, footnote = NA, filename = NA, nice = TRUE, ...)

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

Arguments

x	Object
nice	If set TRUE (default) output values are rounded and optimized for publication tables.
...	Further parameters passed to the print function
object	An scdf or an object exported from a scan function.
caption	Character string with table caption. If left NA (default) a caption will be created based on the exported object.
footnote	Character string with table footnote. If left NA (default) a footnote will be created based on the exported object.
filename	String containing the file name. If a filename is given the output will be written to that file.
data	A single-case data frame. See scdf() to learn about this format.
dvar	Character string with the name of the dependent variable. Defaults to the attributes in the scdf file.
pvar	Character string with the name of the phase variable. Defaults to the attributes in the scdf file.
mvar	Character string with the name of the measurement time variable. Defaults to the attributes in the scdf file.
decreasing	If you expect data to be lower in the B phase, set decreasing = TRUE. Default is decreasing = FALSE.
trend_method	Method used to calculate the trend line. Default is trend_method = "OLS". Possible values are: "OLS", "bisplit", and "trisplit". "bisplit", and "trisplit" should only be used for cases with at least five data-points in both relevant phases.

conservative	The CDC method adjusts the original mean and trend lines by adding (expected increase) or subtracting (expected decrease) an additional .25 SD before evaluating phase B data. Default is the CDC method with `conservative = .25`. To apply the Dual-Criterion (DC) method, set `conservative = 0`.
phases	A vector of two characters or numbers indicating the two phases that should be compared. E.g., `phases = c("A","C")` or `phases = c(2,4)` for comparing the second to the fourth phase. Phases could be combined by providing a list with two elements. E.g., `phases = list(A = c(1,3), B = c(2,4))` will compare phases 1 and 3 (as A) against 2 and 4 (as B). Default is `phases = c(1,2)`.

Value

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

Functions

- `print(sc_cdc)`: Print results
- `export(sc_cdc)`: Export html results

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

See Also

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

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
)
```

---

print.sc_outlier *Handling outliers in single-case data*

Description

Identifies and drops outliers within a single-case data frame (scdf). Outliers can be identified based on mean average deviation (MAD), standard deviation (SD), confidence intervals (CI), or Cook's Distance from a Piecewise Linear Regression Model.

Usage

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

## S3 method for class 'sc_outlier'
export(object, caption = NA, footnote = NA, filename = NA, ...)

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

Arguments

x	An object returned by outlier()
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
object	An scdf or an object exported from a scan function.
caption	Character string with table caption. If left NA (default) a caption will be created based on the exported object.
footnote	Character string with table footnote. If left NA (default) a footnote will be created based on the exported object.
filename	String containing the file name. If a filename is given the output will be written to that file.
data	A single-case data frame. See scdf() to learn about this format.
dvar	Character string with the name of the dependent variable. Defaults to the attributes in the scdf file.
pvar	Character string with the name of the phase variable. Defaults to the attributes in the scdf file.
mvar	Character string with the name of the measurement time variable. Defaults to the attributes in the scdf file.
method	Specifies the method for outlier identification. Set method = "MAD" for mean average deiviation, method = "SD" for standard deviations, method = "CI" for confidence intervals, method = "Cook" for Cook's Distance based on the Piecewise Linear Regression Model.
criteria	Specifies the criteria for outlier identification. Based on the method setting.

Details

For method = "SD", criteria = 2 would refer t0 two standard deviations. For method = "MAD", criteria = 3.5 would refer to 3.5 times the mean average deviation. For method = "CI", criteria = 0.99 would refer to a 99 percent confidence interval. For method = "cook", criteria = "4/n" would refer to a Cook's Distance greater than 4/n.

Value

data	A single-case data frame with substituted outliers.
dropped.n	A list with the number of dropped data points for each single-case.
dropped.mt	A list with the measurement-times of dropped data points for each single-case (values are based on the mt varia
sd.matrix	A list with a matrix for each case with values for the upper and lower boundaries based on the standard deviatio
ci.matrix	A list with a matrix for each single-case with values for the upper and lower boundaries based on the confidenc
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.

Functions

- `print(sc_outlier)`: Print results
- `export(sc_outlier)`: Export html results

Author(s)

Juergen Wilbert

See Also

Other data manipulation functions: `add_l2()`, `as.data.frame.scdf()`, `as_scdf()`, `batch_apply()`, `fill_missing()`, `moving_median()`, `ranks()`, `rescale()`, `scdf()`, `select_cases()`, `set_vars()`, `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, method = "SD", criteria = 1.5)
res_outlier

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

---

print.sc_rand *Randomization Tests for single-case data*

Description

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

Usage

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

## S3 method for class 'sc_rand'
export(object, caption = NA, footnote = NA, filename = NA, ...)

rand_test(
  data,
  dvar,
  pvar,
  statistic = c("Mean B-A", "Mean A-B", "Median B-A", "Median A-B", "Mean |A-B|",
    "Median |A-B|", "SMD hedges", "SMD glass", "W-test", "T-test", "NAP",
    "NAP decreasing", "Slope B-A", "Slope A-B"),
  statistic_function = NULL,
  number = 500,
  complete = FALSE,
  limit = 5,
  startpoints = NA,
  exclude.equal = FALSE,
  phases = c(1, 2),
  graph = FALSE,
  output = NULL,
  seed = NULL
)
```

Arguments

x	An object returned by rand_test()
...	Further parameters passed to the print function
object	An scdf or an object exported from a scan function.
caption	Character string with table caption. If left NA (default) a caption will be created based on the exported object.
footnote	Character string with table footnote. If left NA (default) a footnote will be created based on the exported object.
filename	String containing the file name. If a filename is given the output will be written to that file.
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". See details.

`statistic_function`
: A list with a user defined function to calculate the statistic. When set, overwrites the `statistic` argument. See details.

`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. If a list of start points is provided, each single-case can have different start points (e.g., `startpoints = list(4:9, 5:10)` for a two-case design). Default is `startpoints = NA`, which uses the `limit`-parameter to define possible start points.

`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. If `exclude.equal = "auto"` is used, the function automatically sets `exclude.equal = TRUE` for single-case designs and to `FALSE` for multiple-case designs.

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

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

`output`
: (deprecated and not implemented)

`seed`
: A seed number for the random generator. Default is `NULL`.

Value

An object of class `sc_rand`. It is a list containing the following elements:

`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.
ecxlude.equal	see argument above
testdirection	Indicates the direction of the test ("greater" or "less").

Functions

- print(sc_rand): Print results
- export(sc_rand): Export html results

Details

Predefinded statisic:

Use the statistic argument to choose a predefnied statistic. 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.
- SMD hedges / SMD glass: Standardizes mean difference of B-A as Hedges's g or Glass' delta.
- NAP: Non-overlap of all pairs.

- `W-test`: Wilcoxon-test statistic W.
- `T-test`: T-test statistic t.

Create own statistic function:

Use the `statistic_function` argument to proved your own function in a list. This list must have an element named `statistic` with a function that takes two arguments `a` and `b` and returns a single numeric value. E.g. `list(statistic = function(a, b) mean(a) - mean(b)`. A second element of the list is named `aggregate` which takes a function with one numeric argument that returns a numeric argument. This function is used to aggregate the values of a multiple case design. If you do not provide this element, it uses the default `function(x) sum(x)/length(x)`. The third optional argument is `name` which provides a name for your user function.

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)
```

---

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. If design is NULL (default), the design parameters have to be specified via the ... argument. If a numeric value is provided instead of a design matrix, it is interpreted as the number of cases n.
round	Rounds the scores to the defined decimal. To round to the second decimal, set round = 2. Default is NA (no rounding).
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. If NULL (default), no seed
...	arguments that are directly passed to the design function for a more concise coding.

Details

The generated data can be normally distributed, Poisson-distributed, or binomially distributed. The default is normally distributed data.

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)
```

`rci` *Reliable change index*

Description

The `rci()` function computes 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 percent confidence interval.
`graph`	If set `TRUE`, a box plot of phase A and B scores is displayed. `graph = FALSE` by default.
`phases`	A vector of two characters or numbers indicating the two phases that should be compared. E.g., `phases = c("A","C")` or `phases = c(2,4)` for comparing the second to the fourth phase. Phases could be combined by providing a list with two elements. E.g., `phases = list(A = c(1,3), B = c(2,4))` will compare phases 1 and 3 (as A) against 2 and 4 (as B). Default is `phases = c(1,2)`.

Details

The reliable change index (RCI) indicates whether the change in scores from phase A to phase B is statistically reliable, that is, whether the change is larger than would be expected due to measurement error alone. The RCI can be calculated using different methods (Jacobsen & Truax, 1991; Christensen & Mendoza, 1986). The most common method is that of Jacobsen and Truax (1991) which uses the standard error of measurement in phase A to calculate the RCI. Christensen and Mendoza (1986) proposed an alteration of the RCI which uses the standard error of the difference between phase A and phase B means. Both methods are implemented in the `rci()` function.

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.

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

Arguments

`file`	Either a character string defining the file to be loaded (e.g. `"SC_Anita.csv"` (if left empty a dialog box for choosing will be opened) or a data.frame.
`cvar`	Sets the variable name of the "case" variable. Defaults to `"case"`.
`pvar`	Sets the variable name of the "phase" variable. Defaults to `"phase"`.
`dvar`	Sets the variable name of the "values" variable. Defaults to `"values"`.

`mvar` Sets the variable name of the "mt" variable. Defaults to `"mt"`.

`sort_cases, sort.labels`
If set TRUE, the resulting list is sorted by label names (alphabetically increasing).

`phase_names, phase.names`
A character vector with phase names. Defaults to the phase names provided in the phase variable.

`type` Format of the file to be loaded. Either "csv", "xlsx", "xls", "excel", "yml" is possible. By default (NA) the type is extracted from the file extension.

`na` Character vector of strings to interpret as missing values.

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

Details

The data file must be in a "long" format with one column for case identifiers, one column for phase identifiers, one column for measurement time, and one column for the dependent variable values. Each row represents a single measurement.

Value

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

Author(s)

Juergen Wilbert

See Also

`read.table()`, `readRDS()`

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

Examples

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

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

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

`rescale` *Rescales values of an scdf*

Description

This function scales the measured values of an scdf file. It allows for mean centering and standardization across all cases included in an scdf.

Usage

```
rescale(data, ..., m = 0, sd = 1)
```

Arguments

`data`	A single-case data frame. See `scdf()` to learn about this format.
`...`	Names of variables to be standardized. If none are given, all numeric variables are standardized.
`m`	The target mean. If set NULL, it is not changed.
`sd`	The target standard deviation. If set NULL, it is not changed.

Value

An scdf with the scaled values.

Author(s)

Juergen Wilbert

See Also

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

Examples

```
## Standardize a multiple case scdf and compute an hplm
exampleAB_50 |>
  rescale(values, mt) |>
  hplm()
```

sample_names *Samples random names*

Description

Generates random names for cases. Names are sampled from a predefined list of neutral, male, female, or mixed names.

Usage

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

Arguments

n	Number of names to sample.
type	"neutral", "male", "female", or "mixed" type of names to sample.
seed	A seed for the random number generator. If provided, the sampling will be reproducible.

Details

This function is useful for anonymizing case names in datasets or for generating random identifiers.

Value

A character vector with random names.

Examples

```
sample_names(3)
```

scdf *Single case data frame constructor*

Description

`scdf()` is the constructor for objects of class `scdf`. It stores data from single-case studies in a structured format suitable for analysis with the `scan` package.

Usage

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

Arguments

`...`	One or more vectors representing measurement variables. See the *Details* section.
`B_start`	The first measurement point of phase B (simple coding; only applicable if the design follows a strict AB pattern).
`phase_starts`	A named vector defining the label and measurement time of each phase start. For example: `phase_starts = c(A1 = 1, B1 = 6, A2 = 14, B2 = 19)`.
`phase_design`	A named vector defining the length and label of each phase. For example: `phase_design = c(A1 = 10, B1 = 10, A2 = 10, B2 = 10)`.
`name`	Optional 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.

Details

If no variable matching the name of the dependent variable is provided (the default name is `values`, which can be changed via the `dvar` argument), and the first provided variable is unnamed, that variable will be interpreted as the dependent variable.

If no measurement-time variable is provided (default name `mt`, configurable via the `mvar` argument), measurement times are automatically defined as a sequence `(1, 2, 3, ..., n)`.

If the dependent variable is a **named vector**, the names will be used to define a phase design. For example, `values = c(A = 2, 3, 5, 4, 3, B = 6, 5, 4, 3)` will be interpreted as an AB phase design with five measurements in phase A and four in phase B.

If a vector matching the name of the phase variable is provided, it will be used to define the phase design directly. Otherwise, the phase design can be defined in three alternative ways: via the `B_start` argument, via the `phase_starts` argument, or via the `phase_design` argument.

If `B_start` is provided, a simple AB phase design is assumed, with phase A starting at measurement time 1 and phase B starting at the measurement time indicated by `B_start`. If `phase_starts` is

provided, the phase design is constructed based on the measurement times indicated in the vector. If `phase_design` is provided, it is used directly to define the phase design. If multiple of these options are provided, the priority order is: `phase_design`, `phase_starts`, `B_start`, phase variable in data frame, names of dependent variable.

If none of these options are provided, an error is raised.

The function can be used to create single-case data frames for multiple cases separately, which can then be combined into a list for multiple-case analyses.

See also the convenience function `transform.scdf` to add new variables to an existing `scdf` object.

See the package vignettes for further examples on how to create and work with `scdf` objects.

See the section on *Data structure* in the documentation of the `scan` package for further details on the `scdf` data structure.

Value

Returns a single-case data frame `scdf` suitable for all functions in the `scan` package.

Author(s)

Juergen Wilbert

See Also

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

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"
)
describe(klaas)

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

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

# Alternative: using phase_starts
```

```
klaas <- scdf(
  c(5, 7, 8, 5, 7, 12, 16, 18, 15, 14, 19),
  phase_starts = c(A = 1, B = 7), 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))
tonia <- scdf(c(A = 9, 8, 8, 7, 5, 7, B = 6, 14, 15, 12, 16))
mbd <- c(charlotte, theresa, tonia)
names(mbd) <- c("Charlotte", "Theresa", "Tonia")
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")))
```

`select_cases` *Select a subset of cases from an scdf*

Description

This function allows to select a subset of cases from an scdf by specifying either the case names or their numeric indices. Negative selection is also supported.

Usage

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

Arguments

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

Value

An scdf with a subset of cases.

Author(s)

Juergen Wilbert

See Also

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

Examples

```
select_cases(exampleAB, Johanna, Karolina)
select_cases(exampleAB, c(Johanna, Karolina))
select_cases(exampleAB, 1,2)
select_cases(exampleAB, 1:2)
select_cases(exampleAB, -Johanna)
select_cases(exampleAB, -c(Johanna, Karolina))
v <- c("Moritz", "Jannis")
select_cases(exampleA1B1A2B2, v)
```

---

select_phases *Select and combine phases for overlap analyses*

Description

Useful when working with pipe operators. This function allows to select and combine specific phases from an scdf for overlap analyses. For example, in an ABAB design one might want to combine both A phases and both B phases (e.g., A = c(1, 3), B = c(2, 4)). The resulting scdf can then be used in overlap functions such as overlap().

Usage

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

Arguments

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

Details

This function selects and combines the specified phases from the input scdf and returns the resulting scdf with the selected phases only. This is particularly useful when working with pipe operators, allowing to select and combine phases before applying overlap functions. The resulting scdf contains only the selected and combined phases, with phase labels "A" and "B". If phase_names = "auto", phase names are generated from the combination of the names of the recombined phases. For example, combining phases 1 and 3 will result in the phase name "1_3". This function simplifies the process of preparing data for overlap analyses by allowing users to easily select and combine phases in a single step.

Value

An scdf with selected phases

Author(s)

Juergen Wilbert

See Also

Other overlap functions: ird(), nap(), overlap(), pand(), pem(), pet(), pnd(), print.sc_cdc(), tau_u()

Examples

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

---

set_vars *Set analysis variables in an scdf object*

Description

This function allows to set or change the dependent variable, measurement-time variable, and phase variable in an scdf object.

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.

Value

An scdf object with updated variable settings.

Author(s)

Juergen Wilbert

See Also

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

Examples

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

---

shinyscan *A Shiny app for scan*

Description

Run a Shiny app with most of the scan functions.

Usage

```
shinyscan(
  scdf = NULL,
  quiet = TRUE,
  browser = c("external", "viewer"),
  theme = "cerulean",
  ...
)
```

Arguments

scdf	If you provide an *scdf* here, it will be loaded at startup.
quiet	If TRUE (default) does not report shiny messages in the console.
browser	c("external","viewer")
theme	Bootstrap 5 theme. Default is "cerulean".
...	Further arguments passed to the `shiny::runApp()` function.

Details

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

Value

This function launches a Shiny application.

Author(s)

Juergen Wilbert

---

smd *Standardized mean differences for single-case data*

Description

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

Usage

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

Details

It computes 'Cohen's d', 'Hedges' g', 'Hedges' g correction', 'Hedges' g durlak correction', and 'Glass' delta' for each single-case included in an scdf.

'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)
```

---

style_plot *(Deprecated) 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 of an scdf*

Description

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

Usage

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

Arguments

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

Details

For subsetting rows, a logical expression can be provided in the subset argument. Missing values are treated as FALSE. The expression is evaluated within each single-case data frame of the scdf.

Value

An scdf.

Examples

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

---

summary.scdf *Summary function for an scdf object*

Description

Provides a summary of an `scdf` object, including the number of cases, measurements per case, and design information.

Usage

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

Arguments

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

Details

The summary includes:

- Total number of cases in the `scdf`.
- A table listing each case with the number of measurements and design.
- Variable names with annotations for phase, measurement-time, and dependent variable.
- Additional information and author details if available.

Author(s)

Juergen Wilbert

tau_u *Tau-U for single-case data*

Description

This function calculates indices of the Tau-U family as proposed by Parker et al. (2011a). It allows to calculate Tau-U values for single cases as well as overall Tau-U values across several single cases by applying a meta analysis.

Usage

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

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

## S3 method for class 'sc_tauu'
export(
  object,
  caption = NA,
  footnote = NA,
  filename = NA,
  select = "auto",
  meta = FALSE,
  round = 3,
  decimals = 3,
  ...
)
```

Arguments

data	A single-case data frame. See scdf() to learn about this format.
dvar	Character string with the name of the dependent variable. Defaults to the attributes in the scdf file.
pvar	Character string with the name of the phase variable. Defaults to the attributes in the scdf file.
method	"complete" (default), "parker" or "tarlow". The "parker" calculates the number of possible pairs as described in Parker et al. (2011) which might lead to tau-U values greater than 1. "tarlow" follows an online calculator and R code developed by Tarlow (2017).
phases	A vector of two characters or numbers indicating the two phases that should be compared. E.g., phases = c("A","C") or phases = c(2,4) for comparing the second to the fourth phase. Phases could be combined by providing a list with two elements. E.g., phases = list(A = c(1,3), B = c(2,4)) will compare phases 1 and 3 (as A) against 2 and 4 (as B). Default is phases = c(1,2).
meta_analyses	If TRUE, a meta analysis is conducted.
ci	Confidence intervals level. If NULL or NA, no confidence intervals are calculated.
ci_method	String to specify the method for calculating the standard error of tau. Either "tau", "z", or "s" (not recommended).
meta_weight_method	String to specify the method for calculating the weights of the studies. Either "tau" or "z".
tau_method	Character with values "a" or "b" (default) indicating whether Kendall Tau A or Kendall Tau B is applied. Ignored for methods 'tarlow' and 'parker'.
continuity_correction	If TRUE, a continuity correction is applied for calculating p-values of correlations (here: S will be reduced by one before calculating Z). Ignored for methods 'tarlow' and 'parker'.
x	Object returned from tau_u().
complete	Print all parameters.
digits	The minimum number of significant digits to be use. If set to "auto" (default), values are predefined.
select	Character vector with name of variables to be included. When the vector is named, variables are renamed appropriately.
nice_p	If TRUE, p-values are printed in publication friendly form.
...	Further arguments passed to the function.
object	An scdf or an object exported from a scan function.
caption	Character string with table caption. If left NA (default) a caption will be created based on the exported object.
footnote	Character string with table footnote. If left NA (default) a footnote will be created based on the exported object.

filename	String containing the file name. If a filename is given the output will be written to that file.
meta	If TRUE, the results of the meta analysis will be exported. If FALSE, each single-case is exported.
round	Integer passed to the digits argument used to round values.
decimals	Decimal places that are reported.

Details

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

Value

table	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.

Functions

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

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.

Long, J. D., & Cliff, N. (1997). Confidence intervals for Kendall's tau. *British Journal of Mathematical and Statistical Psychology*, 50(1), 31–41. https://doi.org/10.1111/j.2044-8317.1997.tb01100.x

Parker, R. I., Vannest, K. J., & Davis, J. L. (2011a). Effect Size in Single-Case Research: A Review of Nine Nonoverlap Techniques. *Behavior Modification*, 35(4), 303–322. https://doi.org/10/dsdfs4

Parker, R. I., Vannest, K. J., Davis, J. L., & Sauber, S. B. (2011b). 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

Tarlow, K. R. (2017, March). Tau-U for single-case research (R code). Retrieved from http://ktarlow.com/stats/

See Also

Other overlap functions: ird(), nap(), overlap(), pand(), pem(), pet(), pnd(), print.sc_cdc(), select_phases()

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")
print(res, complete = TRUE)

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

---

trend *Trend analysis for single-cases data*

Description

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

Usage

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

Arguments

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

offset	(Deprecated. Please use first_mt). An offset for the first measurement-time of each phase. If offset = 0, the phase measurement is handled as MT 1. Default is offset = -1, setting the first value of MT to 0.
first_mt	A numeric setting the value for the first measurement-time. Default = 0.
model	A string or a list of (named) strings each depicting one regression model. This is a formula expression of the standard R class. The parameters of the model are values, mt and phase.

Details

The function computes separate regression models for each phase and for the whole data. By default two models are computed: a linear model and a quadratic model. Additionally, custom models can be specified using the model argument. The measurement time variable is adjusted such that the first measurement time point of each phase is set to the value specified in the first_mt argument (default = 0). This means that if first_mt = 0, the first measurement time point of each phase is set to 0, if first_mt = 1, the first measurement time point of each phase is set to 1, and so on. This adjustment allows for a more intuitive interpretation of the regression coefficients, especially the intercept, which then represents the estimated value at the beginning of each phase.

Value

A list of class sc_trend containing:

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

Author(s)

Juergen Wilbert

See Also

describe()

Other regression functions: bplm(), fetch(), hplm(), mplm(), plm(), print.sc_ac(), print.sc_bctau()

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,
  model = list("Cubic" = values ~ mt^3, "Log Time" = values ~ log(mt)),
  first_mt = 1 # must be set to 1 because log(0) would be -Inf
)
```

`write_scdf` *Data output: Write single-case data to a .csv-file*

Description

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

Usage

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

Arguments

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

Details

This is a wrapper for the write.table function with predefined parameters. It converts the single-case data file into a data frame and writes it to a .csv-file. Each single-case data set is stacked below each other and an additional column "case" is added to identify the different single-cases.

Value

Invisibly returns `NULL`.

Author(s)

Juergen Wilbert

See Also

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

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

Examples

```
## write single-case data to a .csv-file
filename <- tempfile(fileext = ".csv")
write_scdf(exampleAB, 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)
```

Index

∗ **Autocorrelation**
print.sc_ac, 67
∗ **Serial correlation**
print.sc_ac, 67
∗ **data manipulation functions**
add_l2, 4
as.data.frame.scdf, 7
as_scdf, 8
batch_apply, 9
fill_missing, 30
moving_median, 39
print.sc_outlier, 73
rescale, 84
scdf, 85
select_cases, 88
set_vars, 90
∗ **datagen**
design, 19
random_scdf, 79
∗ **datasets**
example_data_sets, 24
∗ **data**
combine, 16
∗ **io-functions**
convert, 17
read_scdf, 82
write_scdf, 102
∗ **io**
convert, 17
read_scdf, 82
write_scdf, 102
∗ **manipulation**
combine, 16
∗ **manip**
as.data.frame.scdf, 7
batch_apply, 9
fill_missing, 30
print.sc_outlier, 73
rescale, 84
∗ **mc fucntions**
random_scdf, 79
∗ **mc functions**
design, 19
∗ **overlap functions**
ird, 37
nap, 45
overlap, 47
pand, 48
pem, 51
pet, 53
pnd, 62
print.sc_cdc, 70
select_phases, 89
tau_u, 97
∗ **overlap**
overlap, 47
print.sc_cdc, 70
select_phases, 89
∗ **regression functions**
bplm, 12
fetch, 29
hplm, 32
mplm, 42
plm, 54
print.sc_ac, 67
print.sc_bctau, 68
trend, 100
∗ **regression**
fetch, 29
print.sc_ac, 67
∗ **transform**
add_l2, 4

acf(), *67, 68*
across_cases *(*moving_median*)*, 39
add_dummy_variables, 3
add_l2, 4, *7–9*, *31*, *40*, *75*, *84*, *87*, *89*, *91*
all_cases *(*moving_median*)*, 39
anova.sc_hplm *(*anova.sc_plm*)*, 5

anova.sc_mplm *(anova.sc_plm)*, 5
anova.sc_plm, 5
as.data.frame.scdf, *5*, 7, *8, 9*, *31*, *40*, *75*, *84*, *87*, *89*, *91*
as.scdf *(scdf)*, 85
as_scdf, *5*, *7*, 8, *9*, *31*, *40*, *75*, *84*, *87*, *89*, *91*
autocorr *(print.sc_ac)*, 67
autocorr(), *67*

batch_apply, *5*, *7, 8*, 9, *31*, *40*, *75*, *84*, *87*, *89*, *91*
Beretvas2008 *(example_data_sets)*, 24
between_smd, 10
Borckardt2014 *(example_data_sets)*, 24
bplm, 12, *30*, *36*, *44*, *58*, *68*, *70*, *101*
bplm(), *10*, *14*
byHeart2011 *(example_data_sets)*, 24

c.scdf *(combine)*, 16
cdc *(print.sc_cdc)*, 70
center_at *(moving_median)*, 39
coef.sc_hplm *(hplm)*, 32
coef.sc_plm, 15
combine, 16
convert, 17, *83*, *102*
corrected_tau *(print.sc_bctau)*, 68
corrected_tau(), *48*, *69*

describe, 18
describe(), *93*, *101*
design, 19
design(), *65*

estimate_design, 22
example_A24 *(example_data_sets)*, 24
example_atd *(example_data_sets)*, 24
example_data_sets, 24
example_stranger *(example_data_sets)*, 24
exampleA1B1A2B2 *(example_data_sets)*, 24
exampleA1B1A2B2_zvt *(example_data_sets)*, 24
exampleAB *(example_data_sets)*, 24
exampleAB_50 *(example_data_sets)*, 24
exampleAB_add *(example_data_sets)*, 24
exampleAB_decreasing *(example_data_sets)*, 24
exampleAB_mpd *(example_data_sets)*, 24
exampleAB_score *(example_data_sets)*, 24
exampleAB_simple *(example_data_sets)*, 24
exampleABAB *(example_data_sets)*, 24
exampleABC *(example_data_sets)*, 24
exampleABC_150 *(example_data_sets)*, 24
exampleABC_50 *(example_data_sets)*, 24
exampleABC_outlier *(example_data_sets)*, 24
export, 26
export(), *14*, *35*, *51*, *57*
export.sc_ac *(print.sc_ac)*, 67
export.sc_bcsmd *(between_smd)*, 10
export.sc_bctau *(print.sc_bctau)*, 68
export.sc_bplm *(bplm)*, 12
export.sc_cdc *(print.sc_cdc)*, 70
export.sc_hplm *(hplm)*, 32
export.sc_ird *(ird)*, 37
export.sc_mplm *(mplm)*, 42
export.sc_outlier *(print.sc_outlier)*, 73
export.sc_pand *(pand)*, 48
export.sc_plm *(plm)*, 54
export.sc_rand *(print.sc_rand)*, 75
export.sc_tauu *(tau_u)*, 97

fetch, *14*, 29, *36*, *44*, *58*, *68*, *70*, *101*
fill_missing, *5*, *7–9*, 30, *40*, *75*, *84*, *87*, *89*, *91*
first_of *(moving_median)*, 39

Grosche2011 *(example_data_sets)*, 24
Grosche2014 *(example_data_sets)*, 24
GruenkeWilbert2014 *(example_data_sets)*, 24

hplm, *14*, *30*, 32, *44*, *58*, *68*, *70*, *101*
hplm(), *4*, *5*, *10*, *32*, *34*
Huber2014 *(example_data_sets)*, 24
Huitema2000 *(example_data_sets)*, 24

import_scdf, 36
ird, 37, *46*, *48*, *51*, *53*, *54*, *63*, *72*, *90*, *100*
ird(), *37*, *48*
is.scdf, 38

Leidig2018 *(example_data_sets)*, 24
Leidig2018_l2 *(example_data_sets)*, 24
Lenz2013 *(example_data_sets)*, 24
lm(), *43*
local_regression *(moving_median)*, 39

MCMCglmm::MCMCglmm(), *14*
moving_mean *(moving_median)*, 39

moving_median, *5*, *7–9*, *31*, 39, *75*, *84*, *87*, *89*, *91*
mplm, *14*, *30*, *36*, 42, *58*, *68*, *70*, *101*
mplm(), *42*, *43*
mtext, *94*

na.omit.scdf, 44
nap, *38*, 45, *48*, *51*, *53*, *54*, *62*, *63*, *72*, *90*, *100*
nap(), *45*, *48*
nlme::gls(), *56*

outlier *(*print.sc_outlier*)*, 73
outlier(), *74*
overlap, *38*, *46*, 47, *51*, *53*, *54*, *63*, *72*, *90*, *100*
overlap(), *19*, *89*, *93*

pand, *38*, *46*, *48*, 48, *53*, *54*, *62*, *63*, *72*, *90*, *100*
pand(), *48*, *49*
Parker2007 *(*example_data_sets*)*, 24
Parker2009 *(*example_data_sets*)*, 24
Parker2009b *(*example_data_sets*)*, 24
Parker2011 *(*example_data_sets*)*, 24
Parker2011b *(*example_data_sets*)*, 24
pem, *38*, *46*, *48*, *51*, 51, *54*, *63*, *72*, *90*, *100*
pem(), *48*
pet, *38*, *46*, *48*, *51*, *53*, 53, *63*, *72*, *90*, *100*
pet(), *48*
plm, *14*, *30*, *36*, *44*, 54, *68*, *70*, *101*
plm(), *64*
plot.scdf, 59
plot.scdf(), *19*, *94*, *95*
plot_rand, 62
plot_rand(), *77*
plotSC *(*plot.scdf*)*, 59
pnd, *38*, *46*, *48*, *51*, *53*, *54*, 62, *72*, *90*, *100*
pnd(), *48*
power_test, 64
print.sc_ac, *14*, *30*, *36*, *44*, *58*, 67, *70*, *101*
print.sc_bcsmd *(*between_smd*)*, 10
print.sc_bctau, *14*, *30*, *36*, *44*, *58*, *68*, 68, *101*
print.sc_bplm *(*bplm*)*, 12
print.sc_cdc, *38*, *46*, *48*, *51*, *53*, *54*, *63*, 70, *90*, *100*
print.sc_hplm *(*hplm*)*, 32
print.sc_ird *(*ird*)*, 37
print.sc_mplm *(*mplm*)*, 42
print.sc_outlier, *5*, *7–9*, *31*, *40*, 73, *84*, *87*, *89*, *91*
print.sc_pand *(*pand*)*, 48
print.sc_plm *(*plm*)*, 54
print.sc_rand, 75
print.sc_tauu *(*tau_u*)*, 97
print.scdf, 66

rand_test *(*print.sc_rand*)*, 75
rand_test(), *64*, *76*
random_scdf, 79
random_scdf(), *65*
ranks, *5*, *7–9*, *31*, *40*, *75*, *84*, *87*, *89*, *91*
rci, 81
read.table(), *83*
read_scdf, *17*, 82, *102*
readRDS(), *83*
rescale, *5*, *7–9*, *31*, *40*, *75*, 84, *87*, *89*, *91*
rowwise *(*moving_median*)*, 39

sample_names, 85
saveRDS(), *102*
scan, *24*, *87*
scdf, *5*, *7–9*, *31*, *40*, *75*, *80*, *83*, *84*, 85, *89*, *91*
scdf(), *3*, *4*, *13*, *17*, *18*, *22*, *31*, *33*, *37*, *42*, *45*, *47*, *49*, *52*, *53*, *56*, *59*, *63*, *67*, *69*, *71*, *74*, *76*, *81*, *84*, *89–91*, *93*, *98*, *100*, *102*
scdf-class *(*scdf*)*, 85
select_cases, *5*, *7–9*, *31*, *40*, *75*, *84*, *87*, 88, *91*
select_phases, *38*, *46*, *48*, *51*, *53*, *54*, *63*, *72*, 89, *100*
set_dvar *(*set_vars*)*, 90
set_mvar *(*set_vars*)*, 90
set_na_at *(*moving_median*)*, 39
set_pvar *(*set_vars*)*, 90
set_vars, *5*, *7–9*, *31*, *40*, *75*, *84*, *87*, *89*, 90
shift, *5*, *7–9*, *31*, *40*, *75*, *84*, *87*, *89*, *91*
shinyscan, 91
smd, 92
smooth_cases, *5*, *7–9*, *31*, *40*, *75*, *84*, *87*, *89*, *91*
SSDforR2017 *(*example_data_sets*)*, 24
standardize, *5*, *7–9*, *31*, *40*, *75*, *84*, *87*, *89*, *91*
stats::glm(), *56*
style_plot, *61*, 93
subset.scdf, 95
summary.scdf, 96

Tarlow2017 *(*example_data_sets*)*, 24

tau_u, *38*, *46*, *48*, *51*, *53, 54*, *63*, *72*, *90*, 97
tau_u(), *48*, *64*, *98*
transform.scdf, *87*
transform.scdf *(*moving_median*)*, 39
trend, *14*, *30*, *36*, *44*, *58*, *68*, *70*, 100
truncate_phase, *5*, *7–9*, *31*, *40*, *75*, *84*, *87*, *89*, *91*

Waddell2011 *(*example_data_sets*)*, 24
write.table(), *102*
write_scdf, *17*, *83*, 102