Package: rankdifferencetest (via r-universe)

November 12, 2026

Title Kornbrot's Rank Difference Test

Version 2025.12.4.999

Description Implements Kornbrot's rank difference test as described in <doi:10.1111/j.2044-8317.1990.tb00939.x>. This method is a modified Wilcoxon signed-rank test which produces consistent and meaningful results for ordinal or monotonically-transformed data.

URL https://brettklamer.com/work/rankdifferencetest, https://bitbucket.org/bklamer/rankdifferencetest

License GPL-3

LazyData TRUE

Depends R (>= 4.3.0)

Imports Rdpack, stats, exactRankTests, Hmisc, boot, generics

Suggests tinytest, rmarkdown, tibble, coin

RdMacros Rdpack

RoxygenNote 7.3.3

Roxygen list(markdown = TRUE)

Encoding UTF-8

Config/pak/sysreqs cmake make libicu-dev libuv1-dev

Repository https://bklamer.r-universe.dev

Date/Publication 2026-02-15 15:41:21 UTC

RemoteUrl https://bitbucket.org/bklamer/rankdifferencetest

RemoteRef HEAD

RemoteSha b1d01d0b66b06a1fe6907d9529c58e92917611f6

Contents

as.data.frame.rdt 2
kornbrot_table1 3

pmedian 4
print.rdt 7
rdpmedian 9
rdt 12
srt 18
tidy.rdt 26

Index **29**

`as.data.frame.rdt` *Coerce to a data frame*

Description

Coerce objects to a `data.frame` in the style of `broom::tidy()`.

Usage

```
## S3 method for class 'rdt'
as.data.frame(x, ...)

## S3 method for class 'srt'
as.data.frame(x, ...)

## S3 method for class 'rdpmedian'
as.data.frame(x, ...)

## S3 method for class 'pmedian'
as.data.frame(x, ...)
```

Arguments

x	Objects returned by: • `rdt()` or `rdt2()` • `srt()` or `srt2()` • `rdpmedian()` or `rdpmedian2()` • `pmedian()` or `pmedian2()`
...	Unused arguments.

Value

`data.frame`

Examples

```
#-------------------------------------------------------------------------
# as.data.frame() examples
#-------------------------------------------------------------------------
library(rankdifferencetest)

# Use example data from Kornbrot (1990)
data <- kornbrot_table1

rdt(
  data = data,
  formula = placebo ~ drug,
  conf_level = 0.95,
  alternative = "two.sided",
  distribution = "asymptotic",
  zero_method = "wilcoxon",
  correct = FALSE
) |>
  as.data.frame()

rdpmedian(
  data = data,
  formula = placebo ~ drug,
  conf_level = 0.95
) |>
  as.data.frame()
```

`kornbrot_table1` *Alertness example data*

Description

An example dataset as seen in table 1 from Kornbrot (1990). The time per problem was recorded for each subject under placebo and drug conditions for the purpose of measuring alertness.

Usage

```
kornbrot_table1
```

Format

A data frame with 13 rows and 3 variables:

subject Subject identifier

placebo The time required to complete a task under the placebo condition

drug The time required to complete a task under the drug condition

Source

Kornbrot DE (1990). "The rank difference test: A new and meaningful alternative to the Wilcoxon signed ranks test for ordinal data." *British Journal of Mathematical and Statistical Psychology*, **43**(2), 241–264. ISSN 00071102. doi:10.1111/j.20448317.1990.tb00939.x.

Examples

```
library(rankdifferencetest)
kornbrot_table1
```

`pmedian` *Pseudomedian*

Description

Computes the Hodges-Lehmann pseudomedian and bootstrap confidence interval.

Usage

```
pmedian(
  data,
  formula,
  conf_level = 0.95,
  conf_method = "percentile",
  n_resamples = 1000L,
  agg_fun = "error"
)

pmedian2(
  x,
  y = NULL,
  conf_level = 0.95,
  conf_method = "percentile",
  n_resamples = 1000L
)
```

Arguments

`data` (data.frame)
The data frame of interest.

`formula` (formula)
A formula of form:

y ~ group | block Use when `data` is in tall format. `y` is the numeric outcome, `group` is the binary grouping variable, and `block` is the subject/item-level variable indicating pairs of observations. `group` will be converted to a

factor and the first level will be the reference value. For example, when `levels(data$group) <- c("pre", "post")`, the focal level is 'post', so differences are `post - pre`. Pairs with non-finite values (infinite or missing) are silently dropped. See `agg_fun` for handling duplicate cases of grouping/blocking combinations.

y ~ x Use when `data` is in wide format. `y` and `x` must be numeric vectors. Differences are calculated as `data$y - data$x`. Pairs with non-finite values (infinite or missing) are silently dropped.

~ x Use when `data$x` represents pre-calculated differences or for the one-sample case. Non-finite values (infinite or missing) are silently dropped.

`conf_level` (Scalar numeric: `0.95`; `[0, 1)`)
The confidence level. If `conf_level = 0`, no confidence interval is calculated.

`conf_method` (Scalar character: `c("percentile", "bca")`)
The type of bootstrap confidence interval.

`n_resamples` (Scalar integer: `1000L`; `[10L, Inf)`)
The number of bootstrap resamples. If `conf_level = 0`, no resampling is performed.

`agg_fun` (Scalar character or function: `"error"`)
Used for aggregating duplicate cases of grouping/blocking combinations when data is in tall format and `formula` has structure `y ~ group | block`. `"error"` (default) will return an error if duplicate grouping/blocking combinations are encountered. Select one of `"first"`, `"last"`, `"sum"`, `"mean"`, `"median"`, `"min"`, or `"max"` for built in aggregation handling (each applies `na.rm = TRUE`). Or define your own function. For example, `myfun <- function(x) {as.numeric(quantile(x, 0.75, na.rm = TRUE))}`.

`x` (numeric)
Numeric vector of data. Values with non-finite values (infinite or missing) are silently dropped.

`y` (numeric: `NULL`)
Numeric vector of data or `NULL`. If `NULL` (default), a one-sample test is performed using `x`. If numeric, differences are calculated as `x - y`. Pairs with non-finite values (infinite or missing) are silently dropped.

Details

This function generates a confidence interval for the pseudomedian based on the observed data, not based on an inversion of the signed-rank test `srt()`.

The Hodges-Lehmann estimator is the median of all pairwise averages of the sample values.

$$\mathrm{HL} = \mathrm{median}\left\{\frac{x_i + x_j}{2}\right\}_{i \le j}$$

This pseudomedian is a robust, distribution-free estimate of central tendency for a single sample, or a location-shift estimator for paired data. It's resistant to outliers and compatible with rank-based inference.

The percentile and BCa bootstrap confidence interval methods are described in chapter 5.3 of Davison and Hinkley (1997).

This function is mainly a wrapper for the function `Hmisc::pMedian()`.

Value

A list with the following elements:

Slot	Subslot	Name	Description
1		`pseudomedian`	Measure of centrality.
2		`lower`	Lower bound of confidence interval for the pseudomedian.
3		`upper`	Upper bound of confidence interval for the pseudomedian.
4		`method`	Estimate method.
5		`info`	Additional information.
5	1	`n_sample`	Number of observations in the original data.
5	2	`n_analytic`	Number of observations after removing non-finite values from the original data.
5	3	`data_type`	Data type.
5	4	`focal_name`	Name of the focal variable (differences are focal - reference).
5	5	`reference_name`	Name of the reference variable (differences are focal - reference).
6		`call`	A named list of the function's arguments (use `as.call()` to convert to a call).

References

Davison AC, Hinkley DV (1997). *Bootstrap Methods and their Application*, 1 edition. Cambridge University Press. ISBN 9780511802843. doi:10.1017/CBO9780511802843.

Harrell Jr FE (2025). *Hmisc: Harrell Miscellaneous*. R package version 5.2-4, https://hbiostat.org/R/Hmisc/.

See Also

`rdpmedian()`

Examples

```
#------------------------------------------------------------------------
# pmedian() example
#------------------------------------------------------------------------
library(rankdifferencetest)

# Use example data from Kornbrot (1990)
data <- kornbrot_table1

# Create tall-format data for demonstration purposes
data_tall <- reshape(
  data = kornbrot_table1,
  direction = "long",
  varying = c("placebo", "drug"),
  v.names = c("time"),
  idvar = "subject",
  times = c("placebo", "drug"),
  timevar = "treatment",
  new.row.names = seq_len(prod(length(c("placebo", "drug")), nrow(kornbrot_table1)))
)

# Subject and treatment should be factors. The ordering of the treatment factor
```

```
# will determine the difference (placebo - drug).
data_tall$subject <- factor(data_tall$subject)
data_tall$treatment <- factor(data_tall$treatment, levels = c("drug", "placebo"))

# Rate transformation inverts the rank ordering.
data$placebo_rate <- 60 / data$placebo
data$drug_rate <- 60 / data$drug
data_tall$rate <- 60 / data_tall$time

# Estimates
pmedian(
  data = data,
  formula = placebo ~ drug
)

pmedian(
  data = data_tall,
  formula = time ~ treatment | subject
)

pmedian2(
  x = data$placebo_rate,
  y = data$drug_rate
)

pmedian(
  data = data_tall,
  formula = rate ~ treatment | subject
)
```

`print.rdt` *Print results*

Description

Prints results in the style of `htest`.

Usage

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

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

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

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

Arguments

x	Objects returned by: • rdt() or rdt2() • srt() or srt2() • rdpmedian() or rdpmedian2() • pmedian() or pmedian2()
digits	(Scalar integer: 3; [1, Inf]) The number of significant digits to use.
...	Unused additional arguments.

Value

invisible(x)

Examples

```
#-------------------------------------------------------------------------
# print() examples
#-------------------------------------------------------------------------
library(rankdifferencetest)

# Use example data from Kornbrot (1990)
data <- kornbrot_table1

rdt(
  data = data,
  formula = placebo ~ drug,
  conf_level = 0.95,
  alternative = "two.sided",
  distribution = "asymptotic",
  zero_method = "wilcoxon",
  correct = FALSE
) |>
  print()

srt(
  data = data,
  formula = placebo ~ drug,
  conf_level = 0.95,
  alternative = "two.sided",
  distribution = "asymptotic",
  zero_method = "wilcoxon",
  correct = FALSE
) |>
  print()

rdpmedian(
  data = data,
  formula = placebo ~ drug,
  conf_level = 0.95
```

```
) |>
  print()

pmedian(
  data = data,
  formula = placebo ~ drug,
  conf_level = 0.95
) |>
  print()
```

`rdpmedian` *Rank difference pseudomedian*

Description

Computes the Hodges-Lehmann pseudomedian and bootstrap confidence interval for the differences of ranks.

Usage

```
rdpmedian(
  data,
  formula,
  conf_level = 0.95,
  conf_method = "percentile",
  n_resamples = 1000L,
  agg_fun = "error"
)

rdpmedian2(
  x,
  y,
  conf_level = 0.95,
  conf_method = "percentile",
  n_resamples = 1000L
)
```

Arguments

`data`	(data.frame) The data frame of interest.
`formula`	(formula) A formula of form: **y ~ group | block** Use when `data` is in tall format. `y` is the numeric outcome, `group` is the binary grouping variable, and `block` is the subject/item-level variable indicating pairs of observations. `group` will be converted to a

factor and the first level will be the reference value. For example, when `levels(data$group) <- c("pre", "post")`, the focal level is 'post', so differences are `post - pre`. Pairs with missing values are silently dropped. See `agg_fun` for handling duplicate cases of grouping/blocking combinations.

y ~ x Use when `data` is in wide format. `y` and `x` must be numeric vectors. Differences of ranks correspond with `data$y - data$x`. Pairs with missing values are silently dropped.

`conf_level` (Scalar numeric: `0.95`; `[0, 1)`)
The confidence level. If `conf_level = 0`, no confidence interval is calculated.

`conf_method` (Scalar character: `c("percentile", "bca")`)
The type of bootstrap confidence interval.

`n_resamples` (Scalar integer: `1000L`; `[10L, Inf)`)
The number of bootstrap resamples. If `conf_level = 0`, no resampling is performed.

`agg_fun` (Scalar character or function: `"error"`)
Used for aggregating duplicate cases of grouping/blocking combinations when data is in tall format and `formula` has structure `y ~ group | block`. `"error"` (default) will return an error if duplicate grouping/blocking combinations are encountered. Select one of `"first"`, `"last"`, `"sum"`, `"mean"`, `"median"`, `"min"`, or `"max"` for built in aggregation handling (each applies `na.rm = TRUE`). Or define your own function. For example, `myfun <- function(x) {as.numeric(quantile(x, 0.75, na.rm = TRUE))}`.

`x` (numeric)
Numeric vector of data. Differences of ranks correspond with `x - y`. Pairs with missing values are silently dropped.

`y` (numeric)
Numeric vector of data. Differences of ranks correspond with `x - y`. Pairs with missing values are silently dropped.

Details

This function generates a confidence interval for the pseudomedian based on the observed differences of ranks, not based on an inversion of the rank difference test `rdt()`.

The Hodges-Lehmann estimator is the median of all pairwise averages of the sample values.

$$\mathrm{HL} = \mathrm{median}\left\{\frac{x_i + x_j}{2}\right\}_{i \le j}$$

This pseudomedian is a robust, distribution-free estimate of central tendency for a single sample, or a location-shift estimator for paired data. It's resistant to outliers and compatible with rank-based inference.

The percentile and BCa bootstrap confidence interval methods are described in chapter 5.3 of Davison and Hinkley (1997).

This function is mainly a wrapper for the function `Hmisc::pMedian()`.

Value

A list with the following elements:

Slot	Subslot	Name	Description
1		`pseudomedian`	Measure of centrality.
2		`lower`	Lower bound of confidence interval for the pseudomedian.
3		`upper`	Upper bound of confidence interval for the pseudomedian.
4		`method`	Estimate method.
5		`info`	Additional information.
5	1	`n_sample`	Number of observations in the original data.
5	2	`n_analytic`	Number of observations after removing missing values from the original data.
5	3	`data_type`	Data type.
5	4	`focal_name`	Name of the focal variable (differences are focal - reference).
5	5	`reference_name`	Name of the reference variable (differences are focal - reference).
6		`call`	A named list of the function's arguments (use `as.call()` to convert to a call).

References

Davison AC, Hinkley DV (1997). *Bootstrap Methods and their Application*, 1 edition. Cambridge University Press. ISBN 9780511802843. doi:10.1017/CBO9780511802843.

Harrell Jr FE (2025). *Hmisc: Harrell Miscellaneous*. R package version 5.2-4, https://hbiostat.org/R/Hmisc/.

See Also

`pmedian()`

Examples

```
#-------------------------------------------------------------------------
# rdpmedian() example
#-------------------------------------------------------------------------
library(rankdifferencetest)

# Use example data from Kornbrot (1990)
data <- kornbrot_table1

# Create tall-format data for demonstration purposes
data_tall <- reshape(
  data = kornbrot_table1,
  direction = "long",
  varying = c("placebo", "drug"),
  v.names = c("time"),
  idvar = "subject",
  times = c("placebo", "drug"),
  timevar = "treatment",
  new.row.names = seq_len(prod(length(c("placebo", "drug")), nrow(kornbrot_table1)))
)

# Subject and treatment should be factors. The ordering of the treatment factor
```

```
# will determine the difference (placebo - drug).
data_tall$subject <- factor(data_tall$subject)
data_tall$treatment <- factor(data_tall$treatment, levels = c("drug", "placebo"))

# Rate transformation inverts the rank ordering.
data$placebo_rate <- 60 / data$placebo
data$drug_rate <- 60 / data$drug
data_tall$rate <- 60 / data_tall$time

# Estimates
rdpmedian(
  data = data,
  formula = placebo ~ drug
)

rdpmedian(
  data = data_tall,
  formula = time ~ treatment | subject
)

rdpmedian2(
  x = data$placebo_rate,
  y = data$drug_rate
)

rdpmedian(
  data = data_tall,
  formula = rate ~ treatment | subject
)
```

---

rdt *Rank difference test*

Description

Performs Kornbrot's rank difference test. The rank difference test is a modified Wilcoxon signed-rank test that produces consistent and meaningful results for ordinal or monotonically transformed data.

Usage

```
rdt(
  data,
  formula,
  conf_level = 0,
  conf_method = "inversion",
  n_resamples = 1000L,
  alternative = "two.sided",
```

```
  mu = 0,
  distribution = "auto",
  correct = TRUE,
  zero_method = "wilcoxon",
  agg_fun = "error",
  digits_rank = Inf,
  tol_root = 1e-04
)

rdt2(
  x,
  y,
  conf_level = 0,
  conf_method = "inversion",
  n_resamples = 1000L,
  alternative = "two.sided",
  mu = 0,
  distribution = "auto",
  correct = TRUE,
  zero_method = "wilcoxon",
  digits_rank = Inf,
  tol_root = 1e-04
)
```

Arguments

`data` (data.frame)
The data frame of interest.

`formula` (formula)
A formula of form:

- **y ~ group | block** Use when `data` is in tall format. `y` is the numeric outcome, `group` is the binary grouping variable, and `block` is the subject/item-level variable indicating pairs of observations. `group` will be converted to a factor and the first level will be the reference value. For example, when `levels(data$group) <- c("pre", "post")`, the focal level is 'post', so differences are `post - pre`. Pairs with missing values are silently dropped. See `agg_fun` for handling duplicate cases of grouping/blocking combinations.
- **y ~ x** Use when `data` is in wide format. `y` and `x` must be numeric vectors. Differences of ranks correspond with `data$y - data$x`. Pairs with missing values are silently dropped.

`conf_level` (Scalar numeric: `0`; `[0, 1)`)
The confidence level. Set to `0.95` for a 95% confidence interval. If `0` (default), no confidence interval is calculated.

`conf_method` (Scalar character: `c("inversion", "percentile", "bca")`)
The type of confidence interval. If `"inversion"` (default), the bounds are computed by inverting the hypothesis test. If `"percentile"`, the bounds are com-

puted using a percentile bootstrap. If "bca", the bounds are computed using a bias-corrected and accelerated (BCa) bootstrap.

n_resamples (Scalar integer: 1000L; [10, Inf))
The number of bootstrap resamples. Only used if "percentile" or "bca" confidence intervals are chosen.

alternative (Scalar character: c("two.sided", "greater", "less"))
The alternative hypothesis. Must be one of "two.sided" (default), "greater", or "less".

mu (Scalar numeric: 0; (-Inf, Inf))
Under the null hypothesis, differences of ranks are assumed to be symmetric around mu.

distribution (Scalar character: c("auto", "exact", "asymptotic"))
The method used to calculate the p-value. If "auto" (default), an appropriate method will automatically be chosen (distribution = "exact" when n < 50 or distribution = "asymptotic" otherwise). If "exact", the exact Wilcoxon signed-rank distribution is used. If "asymptotic", the asymptotic normal approximation is used.

correct (Scalar logical: c(TRUE, FALSE))
Whether or not to apply a continuity correction to the Z-statistic for the asymptotic approximation of the p-value.

zero_method (Scalar character: c("wilcoxon", "pratt"))
The method used to handle differences of ranks equal to zero. If "wilcoxon" (default), zeros are removed prior to ranking (classic Wilcoxon convention). If "pratt", zeros are retained for ranking but excluded from the signed-rank sum.

agg_fun (Scalar character or function: "error")
Used for aggregating duplicate cases of grouping/blocking combinations when data is in tall format and formula has structure y ~ group | block. "error" (default) will return an error if duplicate grouping/blocking combinations are encountered. Select one of "first", "last", "sum", "mean", "median", "min", or "max" for built in aggregation handling (each applies na.rm = TRUE). Or define your own function. For example, myfun <- function(x) {as.numeric(quantile(x, 0.75, na.rm = TRUE))}.

digits_rank (Scalar integer: Inf; (0, Inf])
Controls ranking precision. If finite, ranks are computed from base::signif(abs(diffs), digits_rank). If Inf (default), ranks are computed from abs(diffs). Smaller values may introduce ties (because they no longer depend on extremely small numeric differences) and thus change averaged ranks and tie counts.

tol_root (Scalar numeric: 1e-4; (0, Inf))
For stats::uniroot(tol=tol_root) calls when conf_level > 0 and distribution = "asymptotic".

x (numeric)
Numeric vector of data. Differences of ranks correspond with x - y. Pairs with missing values are silently dropped.

y (numeric)
Numeric vector of data. Differences of ranks correspond with x - y. Pairs with missing values are silently dropped.

Details

For paired data, the Wilcoxon signed-rank test results in subtraction of the paired values. However, this subtraction is not meaningful for ordinal scale variables. In addition, any monotone transformation of the data will result in different signed ranks, thus different p-values. However, ranking the original data allows for meaningful addition and subtraction of ranks and preserves ranks over monotonic transformation. Kornbrot developed the rank difference test for these reasons.

Kornbrot recommends that the rank difference test be used in preference to the Wilcoxon signed-rank test in all paired comparison designs where the data are not both of interval scale and of known distribution. The rank difference test preserves good power compared to Wilcoxon's signed-rank test, is more powerful than the sign test, and has the benefit of being a true distribution-free test.

The procedure for Kornbrot's rank difference test is as follows:

1. Combine all $2n$ paired observations.
2. Order the values from smallest to largest.
3. Assign ranks $1, 2, \ldots, 2n$ with average rank for ties.
4. Perform the Wilcoxon signed-rank test using the paired ranks.

The test statistic for the rank difference test (D) is not exactly equal to the test statistic of the naive rank-transformed Wilcoxon signed-rank test (W^+). However, using W^+ should result in a conservative estimate for D, and they approach in distribution as the sample size increases. Kornbrot (1990) discusses methods for calculating D when $n \leq 7$ and $8 \leq n \leq 20$. `rdt()` uses W^+ instead of D.

See `srt()` for additional details about implementation of Wilcoxon's signed-rank test.

Value

A list with the following elements:

Slot	Subslot	Name	Description
1		`p_value`	p-value.
2		`statistic`	Test statistic. W^+ for the exact Wilcoxon signed-rank distribution. Z for the asym
3		`pseudomedian`	Measure of centrality.
4		`lower`	Lower bound of confidence interval for the pseudomedian. `NULL` if no CI requested
5		`upper`	Upper bound of confidence interval for the pseudomedian. `NULL` if no CI requested
6		`method`	Test method.
7		`info`	Additional test information.
7	1	`p_value_method`	Method used to calculate the p-value.
7	2	`pseudomedian_method`	Method used to calculate the pseudomedian.
7	3	`conf_method`	Method used to calculate the confidence interval.
7	4	`conf_level_achieved`	Achieved confidence level.
7	5	`n_sample`	Number of observations in the original data.
7	6	`n_analytic`	Number of observations after removing missing values from the original data.
7	7	`n_zeros`	Number of zeros among differences of ranks in the analytic data set.
7	8	`n_signed`	Number of nonzero differences of ranks in the analytic data set.
7	9	`n_ties`	Number of tied ranks after ranking the absolute differences of ranks.
7	10	`data_type`	Data type.
7	11	`focal_name`	Name of the focal variable (differences are focal - reference).

7	12	reference_name	Name of the reference variable (differences are focal - reference).
8		call	A named list of the function's arguments (use as.call() to convert to a call; call

References

Kornbrot DE (1990). "The rank difference test: A new and meaningful alternative to the Wilcoxon signed ranks test for ordinal data." *British Journal of Mathematical and Statistical Psychology*, **43**(2), 241–264. ISSN 00071102. doi:10.1111/j.20448317.1990.tb00939.x.

See Also

srt()

Examples

```
#-------------------------------------------------------------------------
# rdt() example
#-------------------------------------------------------------------------
library(rankdifferencetest)

# Use example data from Kornbrot (1990)
data <- kornbrot_table1

# Create tall-format data for demonstration purposes
data_tall <- reshape(
  data = kornbrot_table1,
  direction = "long",
  varying = c("placebo", "drug"),
  v.names = c("time"),
  idvar = "subject",
  times = c("placebo", "drug"),
  timevar = "treatment",
  new.row.names = seq_len(prod(length(c("placebo", "drug")), nrow(kornbrot_table1)))
)

# Subject and treatment should be factors. The ordering of the treatment factor
# will determine the difference (placebo - drug).
data_tall$subject <- factor(data_tall$subject)
data_tall$treatment <- factor(data_tall$treatment, levels = c("drug", "placebo"))

# Recreate analysis and results from table 3 (page 248) in Kornbrot (1990)
## Divide p-value by 2 for one-tailed probability.
rdt(
  data = data,
  formula = placebo ~ drug,
  alternative = "two.sided",
  distribution = "asymptotic",
  zero_method = "wilcoxon",
  correct = TRUE,
  conf_level = 0.95
)
```

```
rdt2(
  x = data$placebo,
  y = data$drug,
  alternative = "two.sided",
  distribution = "asymptotic",
  zero_method = "wilcoxon",
  correct = TRUE,
  conf_level = 0.95
)

# The same outcome is seen after transforming time to rate.
## Rate transformation inverts the rank ordering.
data$placebo_rate <- 60 / data$placebo
data$drug_rate <- 60 / data$drug
data_tall$rate <- 60 / data_tall$time

rdt(
  data = data_tall,
  formula = rate ~ treatment | subject,
  alternative = "two.sided",
  distribution = "asymptotic",
  zero_method = "wilcoxon",
  correct = TRUE,
  conf_level = 0.95
)

# In contrast to the rank difference test, the Wilcoxon signed-rank test
# produces differing results. See table 1 and table 2 (page 245) in
# Kornbrot (1990).
## Divide p-value by 2 for one-tailed probability.
srt(
  data = data,
  formula = placebo ~ drug,
  alternative = "two.sided",
  distribution = "asymptotic",
  zero_method = "wilcoxon",
  correct = TRUE,
  conf_level = 0.95
)

srt(
  data = data_tall,
  formula = rate ~ treatment | subject,
  alternative = "two.sided",
  distribution = "asymptotic",
  zero_method = "wilcoxon",
  correct = TRUE,
  conf_level = 0.95
)
```

srt *Signed-rank test*

Description

Performs Wilcoxon's signed-rank test.

Usage

```
srt(
  data,
  formula,
  conf_level = 0,
  conf_method = "inversion",
  n_resamples = 1000L,
  alternative = "two.sided",
  mu = 0,
  distribution = "auto",
  correct = TRUE,
  zero_method = "wilcoxon",
  agg_fun = "error",
  digits_rank = Inf,
  tol_root = 1e-04
)

srt2(
  x,
  y = NULL,
  conf_level = 0,
  conf_method = "inversion",
  n_resamples = 1000L,
  alternative = "two.sided",
  mu = 0,
  distribution = "auto",
  correct = TRUE,
  zero_method = "wilcoxon",
  digits_rank = Inf,
  tol_root = 1e-04
)
```

Arguments

data	(data.frame) The data frame of interest.
formula	(formula) A formula of form:

y ~ group | block Use when `data` is in tall format. `y` is the numeric outcome, `group` is the binary grouping variable, and `block` is the subject/item-level variable indicating pairs of observations. `group` will be converted to a factor and the first level will be the reference value. For example, when `levels(data$group) <- c("pre", "post")`, the focal level is 'post', so differences are `post - pre`. Pairs with non-finite values (infinite or missing) are silently dropped. See `agg_fun` for handling duplicate cases of grouping/blocking combinations.

y ~ x Use when `data` is in wide format. `y` and `x` must be numeric vectors. Differences are calculated as `data$y - data$x`. Pairs with non-finite values (infinite or missing) are silently dropped.

~ x Use when `data$x` represents pre-calculated differences or for the one-sample case. Non-finite values (infinite or missing) are silently dropped.

`conf_level` (Scalar numeric: `0`; `[0, 1)`)
The confidence level. Set to `0.95` for a 95% confidence interval. If `0` (default), no confidence interval is calculated.

`conf_method` (Scalar character: `c("inversion", "percentile", "bca")`)
The type of confidence interval. If `"inversion"` (default), the bounds are computed by inverting the hypothesis test. If `"percentile"`, the bounds are computed using a percentile bootstrap. If `"bca"`, the bounds are computed using a bias-corrected and accelerated (BCa) bootstrap.

`n_resamples` (Scalar integer: `1000L`; `[10, Inf)`)
The number of bootstrap resamples. Only used if `"percentile"` or `"bca"` confidence intervals are chosen.

`alternative` (Scalar character: `c("two.sided", "greater", "less")`)
The alternative hypothesis. Must be one of `"two.sided"` (default), `"greater"`, or `"less"`.

`mu` (Scalar numeric: `0`; `(-Inf, Inf)`)
Under the null hypothesis, `x` or `x - y` is assumed to be symmetric around `mu`.

`distribution` (Scalar character: `c("auto", "exact", "asymptotic")`)
The method used to calculate the p-value. If `"auto"` (default), an appropriate method will automatically be chosen (`distribution = "exact"` when n < 50 or `distribution = "asymptotic"` otherwise). If `"exact"`, the exact Wilcoxon signed-rank distribution is used. If `"asymptotic"`, the asymptotic normal approximation is used.

`correct` (Scalar logical: `c(TRUE, FALSE)`)
Whether or not to apply a continuity correction to the Z-statistic for the asymptotic approximation of the p-value.

`zero_method` (Scalar character: `c("wilcoxon", "pratt")`)
The method used to handle differences equal to zero. If `"wilcoxon"` (default), zeros are removed prior to ranking (classic Wilcoxon convention). If `"pratt"`, zeros are retained for ranking but excluded from the signed-rank sum.

`agg_fun` (Scalar character or function: `"error"`)
Used for aggregating duplicate cases of grouping/blocking combinations when data is in tall format and `formula` has structure `y ~ group | block`. `"error"` (default) will return an error if duplicate grouping/blocking combinations are

encountered. Select one of "first", "last", "sum", "mean", "median", "min", or "max" for built in aggregation handling (each applies na.rm = TRUE). Or define your own function. For example, myfun <- function(x) {as.numeric(quantile(x, 0.75, na.rm = TRUE))}.

digits_rank
(Scalar integer: Inf; (0, Inf])
Controls ranking precision. If finite, ranks are computed from base::signif(abs(diffs), digits_rank). If Inf (default), ranks are computed from abs(diffs). Smaller values may introduce ties (because they no longer depend on extremely small numeric differences) and thus change averaged ranks and tie counts.

tol_root
(Scalar numeric: 1e-4; (0, Inf))
For stats::uniroot(tol=tol_root) calls when conf_level > 0 and distribution = "asymptotic".

x
(numeric)
Numeric vector of data. Values with non-finite values (infinite or missing) are silently dropped.

y
(numeric: NULL)
Numeric vector of data or NULL. If NULL (default), a one-sample test is performed using x. If numeric, differences are calculated as x - y. Pairs with non-finite values (infinite or missing) are silently dropped.

Details

The procedure for Wilcoxon's signed-rank test is as follows:

1. For one-sample data x or paired samples x and y, where mu is the measure of center under the null hypothesis, define the values used for analysis as (x - mu) or (x - y - mu).
2. Define 'zero' values as (x - mu == 0) or (x - y - mu == 0).
   - zero_method = "wilcoxon": Remove values equal to zero.
   - zero_method = "pratt": Keep values equal to zero.
3. Order the absolute values from smallest to largest.
4. Assign ranks $1, 2, \ldots, n$ to the ordered absolute values, using mean rank for ties.
   - zero_method = "pratt": remove values equal to zero and their corresponding ranks.
5. Calculate the test statistic.
   Calculate W^+ as the sum of the ranks for positive values. Let r_i denote the absolute-value rank of the i-th observation after applying the chosen zero handling and ranking precision and v_i denote the values used for analysis. Then

$$W^+ = \sum_{i:v_i>0} r_i.$$

   $W^+ + W^- = n(n+1)/2$, thus either can be calculated from the other. If the null hypothesis is true, W^+ and W^- are expected to be equal.
   - distribution = "exact": Use W^+ which takes values between 0 and $n(n+1)/2$.
   - distribution = "asymptotic": Use the standardized test statistic $Z = \frac{W^+ - E_0(W^+) - cc}{Var_0(W^+)^{1/2}}$ where $Z \sim N(0,1)$ asymptotically.

- `zero_method = "wilcoxon"`: Under the null hypothesis, the expected mean and variance are

$$E_0(W^+) = \frac{n(n+1)}{4}$$
$$Var_0(W^+) = \frac{n(n+1)(2n+1)}{24} - \frac{\sum(t^3 - t)}{48},$$

  where t is the number of ties for each unique ranked absolute value.
- `zero_method = "pratt"`: Under the null hypothesis, the expected mean and variance are

$$E_0(W^+) = \frac{n(n+1)}{4} - \frac{n_{zeros}(n_{zeros}+1)}{4}$$
$$Var_0(W^+) = \frac{n(n+1)(2n+1) - n_{zeros}(n_{zeros}+1)(2n_{zeros}+1)}{24} - \frac{\sum(t^3 - t)}{48},$$

  where t is the number of ties for each unique ranked absolute value.
- `correct = TRUE`: The continuity correction is defined by:

$$cc = \begin{cases} 0.5 \times \text{sign}(W^+ - E_0(W^+)) & \text{two-sided alternative} \\ 0.5 & \text{greater than alternative} \\ -0.5 & \text{less than alternative.} \end{cases}$$

- `correct = FALSE`: Set $cc = 0$.

Alternatively, $E_0(W^+)$ and $Var_0(W^+)$ can be calculated without need for closed form expressions that include zero correction and/or tie correction. Consider each rank r_i (with averaged rank for ties) as a random variable X_i that contributes to W^+. Under the null hypothesis, each rank has 50% chance of being positive or negative. So X_i can take values

$$X_i = \begin{cases} r_i & \text{with probability } p = 0.5 \\ 0 & \text{with probability } 1 - p = 0.5. \end{cases}$$

Using $Var(X_i) = E(X_i^2) - E(X_i)^2$ where

$$\begin{aligned} E(X_i) &= p \cdot r_i + (1-p) \cdot 0 \\ &= 0.5 r_i \\ E(X_i^2) &= p \cdot r_i^2 + (1-p) \cdot 0^2 \\ &= 0.5 r_i^2 \\ E(X_i)^2 &= (0.5 r_i)^2, \end{aligned}$$

it follows that $Var(X_i) = 0.5 r_i^2 - (0.5 r_i)^2 = 0.25 r_i^2$. Hence, $E_0(W^+) = \frac{\sum r_i}{2}$ and $Var_0(W^+) = \frac{\sum r_i^2}{4}$.

6. Calculate the p-value.
   - `distribution = "exact"`
     - No zeros among the differences and ranks are tie free
       * Use the Wilcoxon signed-rank distribution as implemented in `stats::psignrank()` to calculate the probability of being as or more extreme than W^+.

  - Zeros present and/or ties present
    * Use the Shift-Algorithm from Streitberg and Röhmel (1984) as implemented in `exactRankTests::pperm()`.
- `distribution = "asymptotic"`
  - Use the standard normal distribution as implemented in `stats::pnorm()` to calculate the probability of being as or more extreme than Z.

Hypotheses and test assumptions:

The signed-rank test hypotheses are stated as:

- Null: `(x)` or `(x - y)` is centered at `mu`.
- Two-sided alternative: `(x)` or `(x - y)` is not centered at `mu`.
- Greater than alternative: `(x)` or `(x - y)` is centered at a value greater than `mu`.
- Less than alternative: `(x)` or `(x - y)` is centered at a value less than `mu`.

The signed-rank test traditionally assumes the differences are independent with identical, continuous, and symmetric distribution. However, not all of these assumptions may be required (Pratt and Gibbons 1981). The 'identically distributed' assumption is not required, keeping the level of test as expected for the hypotheses as stated above. The symmetry assumption is not required when using one-sided alternative hypotheses. For example:

- Null: `(x)` or `(x - y)` is symmetric and centered at `mu`.
- Greater than alternative: `(x)` or `(x - y)` is stochastically larger than `mu`.
- Less than alternative: `(x)` or `(x - y)` is stochastically smaller than `mu`.

Zero handling:

`zero_method = "pratt"` uses the method by Pratt (1959), which first rank-transforms the absolute values, including zeros, and then removes the ranks corresponding to the zeros. `zero_method = "wilcoxon"` uses the method by Wilcoxon (1950), which first removes the zeros and then rank-transforms the remaining absolute values. Conover (1973) found that when comparing a discrete uniform distribution to a distribution where probabilities linearly increase from left to right, Pratt's method outperforms Wilcoxon's. When testing a binomial distribution centered at zero to see whether the parameter of each Bernoulli trial is $\frac{1}{2}$, Wilcoxon's method outperforms Pratt's.

Pseudomedians:

Hodges-Lehmann estimator:
The Hodges-Lehmann estimator is the median of all pairwise averages of the sample values.

$$\mathrm{HL} = \mathrm{median}\left\{\frac{x_i + x_j}{2}\right\}_{i \le j}$$

This pseudomedian is a robust, distribution-free estimate of central tendency for a single sample, or a location-shift estimator for paired data. It's resistant to outliers and compatible with rank-based inference. This statistic is returned when `conf_level = 0` (for all test methods) or when confidence intervals are requested for an exact Wilcoxon test with zero-free data and tie-free ranks.

Midpoint around expected signed-rank statistic (exact CI-centered estimator):
The exact test for data which contains zeros or whose ranks contain ties uses the Streitberg-Rohmel Shift-algorithm. When a confidence interval is requested under this scenario, the estimated pseudomedian is a midpoint around the expected rank sum. This midpoint is the average

of the largest shift value d whose signed-rank statistic does not exceed the null expectation and the smallest d whose statistic exceeds it.
In detail, let $W^+(d)$ be the Wilcoxon signed-rank sum at shift d, and let E_0 denote the null expectation (e.g., $\sum r_i/2$ when ranks are r_i). Then

$$\hat{d}_{\text{mid}} = \frac{1}{2}\left(\min\{d : W^+(d) \le \lceil E_0 \rceil\} + \max\{d : W^+(d) > E_0\}\right)$$

This pseudomedian is a discrete-compatible point estimate that centers the exact confidence interval obtained by inverting the exact signed-rank distribution. It may differ from the Hodges-Lehmann estimator when data are tied or rounded.

Root of standardized signed-rank statistic (asymptotic CI-centered estimator):
A similar algorithm is used to estimate the pseudomedian when a confidence interval is requested under the asymptotic test scenario. This pseudomedian is the value of the shift d for which the standardized signed-rank statistic equals zero under the asymptotic normal approximation.
In detail, let $W^+(d)$ be the signed-rank sum, with null mean $E_0(d)$ and null variance $\mathrm{Var}_0(d)$ (with possible tie and continuity corrections). Define

$$Z(d) = \frac{W^+(d) - E_0(d)}{\sqrt{\mathrm{Var}_0(d)}}.$$

The pseudomedian is the root

$$\hat{d}_{\text{root}} = \{d : Z(d) = 0\}.$$

This pseudomedian is a continuous-compatible point estimate that centers the asymptotic confidence interval. It's the solution to the test-inversion equation under a normal approximation. It's sensitive to tie/zero patterns through $\mathrm{Var}_0(d)$, may include a continuity correction, and is not guaranteed to equal the Hodges-Lehmann estimator or the exact midpoint.

Confidence intervals:

Exact Wilcoxon confidence interval:
The exact Wilcoxon confidence interval is obtained by inverting the exact distribution of the signed-rank statistic. It uses the permutation distribution of the Wilcoxon statistic and finds bounds where cumulative probabilities cross $\alpha/2$ and $1 - \alpha/2$. Endpoints correspond to quantiles from `stats::qsignrank()`. This interval guarantees nominal coverage under the null hypothesis without relying on asymptotic approximations. It respects discreteness of the data and may produce conservative intervals when the requested confidence level is not achievable (with warnings).

Exact Wilcoxon confidence interval using the Shift-algorithm:
The exact Wilcoxon confidence interval using the Shift-algorithm is obtained by enumerating all candidate shifts and inverting the exact signed-rank distribution. In detail, it generates all pairwise averages $\frac{x_i + x_j}{2}$, evaluates the signed-rank statistic for each candidate shift, and determines bounds using `exactRankTests::pperm()` and `exactRankTests::qperm()`.

Asymptotic Wilcoxon confidence interval:
The asymptotic Wilcoxon confidence interval is obtained by inverting the asymptotic normal approximation of the signed-rank statistic. In detail, Define a standardized statistic:

$$Z(d) = \frac{W^+(d) - E_0(d) - cc}{\sqrt{\mathrm{Var}_0(d)}}$$

where $W^+(d)$ is the signed-rank sum at shift d. Then solve for d such that $Z(d)$ equals the normal quantiles using `stats::uniroot()`. This interval may not be reliable for small samples or heavily tied data.

Bootstrap confidence intervals:
The percentile and BCa bootstrap confidence interval methods are described in chapter 5.3 of Davison and Hinkley (1997).

History:

`stats::wilcox.test()` is the canonical function for the Wilcoxon signed-rank test. `exactRankTests::wilcox.exact()` implemented the Streitberg-Rohmel Shift-algorithm for exact inference when zeros and/or ties are present. coin::coin superseded **exactRankTests** and includes additional methods. `srt()` reimplements these functions so the best features of each is available in a fast and easy to use format.

Value

A list with the following elements:

Slot	Subslot	Name	Description
1		`p_value`	p-value.
2		`statistic`	Test statistic. W^+ for the exact Wilcoxon signed-rank distribution. Z for the asym
3		`pseudomedian`	Measure of centrality.
4		`lower`	Lower bound of confidence interval for the pseudomedian. `NULL` if no CI requested
5		`upper`	Upper bound of confidence interval for the pseudomedian. `NULL` if no CI requested
6		`method`	Test method.
7		`info`	Additional test information.
7	1	`p_value_method`	Method used to calculate the p-value.
7	2	`pseudomedian_method`	Method used to calculate the pseudomedian.
7	3	`conf_method`	Method used to calculate the confidence interval.
7	4	`conf_level_achieved`	Achieved confidence level.
7	5	`n_sample`	Number of observations in the original data.
7	6	`n_analytic`	Number of observations after removing non-finite values from the original data.
7	7	`n_zeros`	Number of zeros among differences in the analytic data set.
7	8	`n_signed`	Number of nonzero differences in the analytic data set.
7	9	`n_ties`	Number of tied ranks after ranking the absolute differences.
7	10	`data_type`	Data type.
7	11	`focal_name`	Name of the focal variable (differences are focal - reference).
7	12	`reference_name`	Name of the reference variable (differences are focal - reference).
8		`call`	A named list of the function's arguments (use `as.call()` to convert to a call; `call`

References

Wilcoxon F (1945). "Individual Comparisons by Ranking Methods." *Biometrics Bulletin*, **1**(6), 80. ISSN 00994987. doi:10.2307/3001968.

Wilcoxon F (1950). "SOME RAPID APPROXIMATE STATISTICAL PROCEDURES." *Annals of the New York Academy of Sciences*, **52**(6), 808–814. ISSN 00778923, 17496632. doi:10.1111/j.17496632.1950.tb53974.x.

Pratt JW, Gibbons JD (1981). *Concepts of Nonparametric Theory*, Springer Series in Statistics. Springer New York, New York, NY. ISBN 9781461259336 9781461259312. doi:10.1007/9781-461259312.

Pratt JW (1959). "Remarks on Zeros and Ties in the Wilcoxon Signed Rank Procedures." *Journal of the American Statistical Association*, **54**(287), 655–667. ISSN 0162-1459, 1537-274X. doi:10.1080/01621459.1959.10501526.

Cureton EE (1967). "The Normal Approximation to the Signed-Rank Sampling Distribution When Zero Differences are Present." *Journal of the American Statistical Association*, **62**(319), 1068–1069. ISSN 0162-1459, 1537-274X. doi:10.1080/01621459.1967.10500917.

Conover WJ (1973). "On Methods of Handling Ties in the Wilcoxon Signed-Rank Test." *Journal of the American Statistical Association*, **68**(344), 985–988. ISSN 0162-1459, 1537-274X. doi:10.1080/01621459.1973.10481460.

Hollander M, Wolfe DA, Chicken E (2014). *Nonparametric statistical methods*, Third edition edition. John Wiley & Sons, Inc, Hoboken, New Jersey. ISBN 9780470387375.

Bauer DF (1972). "Constructing Confidence Sets Using Rank Statistics." *Journal of the American Statistical Association*, **67**(339), 687–690. ISSN 0162-1459, 1537-274X. doi:10.1080/01621459.1972.10481279.

Streitberg B, Röhmel J (1984). "Exact nonparametrics in APL." *SIGAPL APL Quote Quad*, **14**(4), 313–325. ISSN 0163-6006. doi:10.1145/384283.801115.

Hothorn T (2001). "On exact rank tests in R." *R News*, **1**(1), 11-12. ISSN 1609-3631. https://journal.r-project.org/articles/RN-2001-002/.

Hothorn T, Hornik K (2002). "Exact Nonparametric Inference in R." In Härdle W, Rönz B (eds.), *Compstat*, 355–360. Physica-Verlag HD, Heidelberg. ISBN 9783790815177 9783642574894. doi:10.1007/9783642574894_52.

Hothorn T, Hornik K, Wiel MAVD, Zeileis A (2008). "Implementing a Class of Permutation Tests: The coin Package." *Journal of Statistical Software*, **28**(8). ISSN 1548-7660. doi:10.18637/jss.v028.i08.

Davison AC, Hinkley DV (1997). *Bootstrap Methods and their Application*, 1 edition. Cambridge University Press. ISBN 9780511802843. doi:10.1017/CBO9780511802843.

Hothorn T, Hornik K (2022). *exactRankTests: Exact Distributions for Rank and Permutation Tests*. R package version 0.8-35.

R Core Team (2025). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/.

See Also

rdt(), stats::wilcox.test(), coin::wilcoxsign_test()

Examples

```
#------------------------------------------------------------------------
# srt() example
#------------------------------------------------------------------------
library(rankdifferencetest)

# Use example data from Kornbrot (1990)
data <- kornbrot_table1

# Create tall-format data for demonstration purposes
data_tall <- reshape(
```

```
  data = kornbrot_table1,
  direction = "long",
  varying = c("placebo", "drug"),
  v.names = c("time"),
  idvar = "subject",
  times = c("placebo", "drug"),
  timevar = "treatment",
  new.row.names = seq_len(prod(length(c("placebo", "drug")), nrow(kornbrot_table1)))
)

# Subject and treatment should be factors. The ordering of the treatment factor
# will determine the difference (placebo - drug).
data_tall$subject <- factor(data_tall$subject)
data_tall$treatment <- factor(data_tall$treatment, levels = c("drug", "placebo"))

# Rate transformation inverts the rank ordering.
data$placebo_rate <- 60 / data$placebo
data$drug_rate <- 60 / data$drug
data_tall$rate <- 60 / data_tall$time

# In contrast to the rank difference test, the Wilcoxon signed-rank test
# produces differing results. See table 1 and table 2 (page 245) in
# Kornbrot (1990).
## Divide p-value by 2 for one-tailed probability.
srt(
  data = data,
  formula = placebo ~ drug,
  alternative = "two.sided",
  distribution = "asymptotic",
  zero_method = "wilcoxon",
  correct = TRUE,
  conf_level = 0.95
)

srt(
  data = data_tall,
  formula = rate ~ treatment | subject,
  alternative = "two.sided",
  distribution = "asymptotic",
  zero_method = "wilcoxon",
  correct = TRUE,
  conf_level = 0.95
)
```

`tidy.rdt` *Coerce to a data frame*

Description

Coerce objects to a 'tibble' data frame in the style of `broom::tidy()`.

Usage

```
## S3 method for class 'rdt'
tidy(x, ...)

## S3 method for class 'srt'
tidy(x, ...)

## S3 method for class 'rdpmedian'
tidy(x, ...)

## S3 method for class 'pmedian'
tidy(x, ...)
```

Arguments

x	Objects returned by: • rdt() or rdt2() • srt() or srt2() • rdpmedian() or rdpmedian2() • pmedian() or pmedian2()
...	Unused arguments.

Value

tbl_df

Examples

```
#-------------------------------------------------------------------------
# tidy() examples
#-------------------------------------------------------------------------
library(rankdifferencetest)

# Use example data from Kornbrot (1990)
data <- kornbrot_table1

rdt(
  data = data,
  formula = placebo ~ drug,
  conf_level = 0.95,
  alternative = "two.sided",
  distribution = "asymptotic",
  zero_method = "wilcoxon",
  correct = FALSE
) |>
  tidy()

rdpmedian(
  data = data,
```

```
  formula = placebo ~ drug,
  conf_level = 0.95
) |>
  tidy()
```

Index

∗ **datasets**
    kornbrot_table1, 3

as.data.frame.pmedian
        *(*as.data.frame.rdt*)*, 2
as.data.frame.rdpmedian
        *(*as.data.frame.rdt*)*, 2
as.data.frame.rdt, 2
as.data.frame.srt *(*as.data.frame.rdt*)*, 2

base::signif, *14*, *20*

coin::coin, *24*
coin::wilcoxsign_test(), *25*

exactRankTests::pperm(), *22, 23*
exactRankTests::qperm(), *23*
exactRankTests::wilcox.exact(), *24*

Hmisc::pMedian(), *5*, *10*

kornbrot_table1, 3

pmedian, 4
pmedian(), *2*, *8*, *11*, *27*
pmedian2 *(*pmedian*)*, 4
pmedian2(), *2*, *8*, *27*
print.pmedian *(*print.rdt*)*, 7
print.rdpmedian *(*print.rdt*)*, 7
print.rdt, 7
print.srt *(*print.rdt*)*, 7

rdpmedian, 9
rdpmedian(), *2*, *6*, *8*, *27*
rdpmedian2 *(*rdpmedian*)*, 9
rdpmedian2(), *2*, *8*, *27*
rdt, 12
rdt(), *2*, *8*, *10*, *15*, *25*, *27*
rdt2 *(*rdt*)*, 12
rdt2(), *2*, *8*, *27*

srt, 18
srt(), *2*, *5*, *8*, *15, 16*, *24*, *27*
srt2 *(*srt*)*, 18
srt2(), *2*, *8*, *27*
stats::pnorm(), *22*
stats::psignrank(), *21*
stats::qsignrank(), *23*
stats::uniroot, *14*, *20*
stats::uniroot(), *24*
stats::wilcox.test(), *24, 25*

tidy.pmedian *(*tidy.rdt*)*, 26
tidy.rdpmedian *(*tidy.rdt*)*, 26
tidy.rdt, 26
tidy.srt *(*tidy.rdt*)*, 26