

# Package ‘Rrepest’

July 21, 2025

**Title** An Analyzer of International Large Scale Assessments in Education

**Version** 1.5.4

**Description** An easy way to analyze international large-scale assessments and surveys in education or any other dataset that includes replicated weights (Balanced Repeated Replication (BRR) weights, Jackknife replicate weights,...) while also allowing for analysis with multiply imputed variables (plausible values). It supports the estimation of univariate statistics (e.g. mean, variance, standard deviation, quantiles), frequencies, correlation, linear regression and any other model already implemented in R that takes a data frame and weights as parameters. It also includes options to prepare the results for publication, following the table formatting standards of the Organization for Economic Cooperation and Development (OECD).

**Depends** R (>= 4.2.0)

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.3.1

**Imports** data.table (>= 1.14.8), doParallel (>= 1.0.17), dplyr (>= 1.1.2), flextable (>= 0.7.2), foreach (>= 1.5.2), labelled (>= 2.9.1), magrittr (>= 2.0.3), officer (>= 0.6.2), parallel (>= 4.2.1), purrr (>= 0.3.4), stringr (>= 1.5.0), tibble (>= 3.1.8), tidyr (>= 1.2.0)

**LazyData** true

**NeedsCompilation** no

**Author** Rodolfo Ilizaliturri [aut, cre],  
Francesco Avvisati [aut],  
Francois Keslair [aut]

**Maintainer** Rodolfo Ilizaliturri <rodolfo.ilizaliturri@oecd.org>

**Repository** CRAN

**Date/Publication** 2025-02-19 13:30:02 UTC

## Contents

average\_groups . . . . . 2

coverage_daggers . . . . .	3
coverage_pct . . . . .	4
df_pisa18 . . . . .	5
df_talis18 . . . . .	5
est . . . . .	6
format_data_categ_vars . . . . .	7
format_data_cont_vars . . . . .	7
format_data_repest . . . . .	8
grouped_sum_freqs . . . . .	9
grp . . . . .	9
indep_diff . . . . .	10
inv_test . . . . .	11
is_prime . . . . .	11
n_obs_x . . . . .	12
paired_indep_diff . . . . .	13
Rrepest . . . . .	13
rrepest_pisa_age_gender . . . . .	16
rrepest_pisa_age_iscd . . . . .	16
talis18_tt3g23o_freq . . . . .	17
weighted.corr . . . . .	18
weighted.corr.cov.n . . . . .	19
weighted.cov . . . . .	19
weighted.iqr . . . . .	20
weighted.mode . . . . .	21
weighted.quant . . . . .	21
weighted.std . . . . .	22
weighted.var . . . . .	23

## Index 24

---

average_groups	<i>Group Averages</i>
----------------	-----------------------

---

## Description

Group Averages

## Usage

```
average_groups(
  res,
  data = NULL,
  group,
  by = NULL,
  over = NULL,
  est = NULL,
  svy = NULL,
  user_na = FALSE,
```

```
    ...  
  )
```

**Arguments**

- res (dataframe) df of results with b. and se. to average
- data (dataframe) df from which to get replicated weights
- group (grp function) that takes arguments group.name, column, cases to create averages at the end of dataframe
- by (string vector) column in which we'll break down results
- over (vector string) columns over which to do analysis
- est (est function) that takes arguments what = estimate, tgt = target, rgr = regressor
- svy (string) name of possible projects to analyse TALISSCH and TALISTCH
- user\_na (bool) TRUE: show nature of user defined missing values for by.var
- ... Additional arguments such as na\_to\_zero : (Bool) TRUE → will take NA as zero for the simple average calculation

**Value**

Dataframe with avergas or weighted averages (totals) in last rows for the selected cases

---

coverage_daggers	<i>Dagger or double dagger according to coverage level</i>
------------------	--

---

**Description**

Transform coverage columns in Rrepest to dagger and double dagger according to the coverage (cvge) column. Default levels are 75 (dagger) and 50 (double dagger). If coverage is above both levels, no symbol is produced (empty). Used with the coverage o option in Rrepest set to TRUE to produce columns with coverage percentages.

**Usage**

```
coverage_daggers(res_df, one_dagger = 75, two_dagger = 50)
```

**Arguments**

- res\_df (data frame) Rrepest output with columns for coverage (cvge)
- one\_dagger (numeric) Level at which the coverage is transformed into a dagger. 75 by default.
- two\_dagger (numeric) Level at which the coverage is transformed into a double dagger. 50 by default

**Value**

Dataframe with daggers or double daggers in coverage column

**Examples**

```
coverage_daggers(talis18_tt3g23o_freq, one_dagger = 95, two_dagger = 90)
```

---

coverage_pct	<i>Coverage percentage (<math>1 - \text{mean}(is.na)</math>) * 100</i>
--------------	--

---

**Description**

Computes the coverage percentage for the column/variable of interest.

**Usage**

```
coverage_pct(df, by, x, w = NULL, limit = NULL)
```

**Arguments**

df	(data frame) Data to analyse
by	(string vector) Variable(s) used for tabulating the variable of interest
x	(string) Variable of interest for which to compute the number of valid (i.e. non-missing) observations
w	(string) Vector of weights
limit	(numeric) Threshold at which, if lower, value will be TRUE

**Value**

Data frame containing the number of valid (i.e. non-missing) observations for the variable of interest

**Examples**

```
data(df_pisa18)
data(df_talis18)

coverage_pct(df = df_pisa18, by = "cnt", x = "wb173q03ha")
coverage_pct(df = df_talis18, by = "cntry", x = "tt3g01", w = "TCHWGT")
```

---

df_pisa18	<i>Program for International Student Assessment (PISA) 2018 noisy data subset</i>
-----------	---

---

### Description

A subset of noisy data from the Program for International Student Assessment (PISA) 2018 cycle for Mexico, Italy, and France.

This dataset is a subset of the PISA 2018 database produced by the OECD for the countries of France, Italy, and Mexico.

### Usage

```
df_pisa18
```

```
data(df_pisa18)
```

### Format

**df\_pisa18:**

A data frame in tibble format with 1269 rows and 1106 columns from which only some relevant variables are listed below. For a detailed description of all variables go to <https://www.oecd.org/en/data/datasets/pisa-2018-database.html#data>

**idlang** Country language

**CNT** Country ISO 3166-1 alpha-3 codes.

**st001d01t** Student International Grade

**st004d01t** Gender ...

A data frame with 1269 rows/observations and 1120 columns/variables.

### Source

<https://www.oecd.org/en/data/datasets/pisa-2018-database.html#data>

---

df_talis18	<i>Teaching and Learning International Survey (TALIS) 2018 noisy data subset</i>
------------	--

---

### Description

A subset of noisy data from the Teaching and Learning International Survey (TALIS) 2018 cycle for Mexico, Italy, and France.

This dataset is a subset of the TALIS 2018 database produced by the OECD for the countries of France, Italy, and Mexico.

Usage

```
df_talis18  
  
data(df_talis18)
```

Format

**df\_talis18:**  
A data frame in tibble format with 496 rows and 548 columns from which only some relevant variables are listed below. For a detailed description of all variables go to <https://www.oecd.org/en/data/datasets/talis-2018-database.html#data>  
**idlang** Country language  
**entry** Country ISO 3166-1 alpha-3 codes.  
**tt3g01** Gender  
**tt3g02** Age  
**tt3g03** Highest level of formal education completed ...  
A data frame with 548 rows/observations and 496 columns/variables.

Source

<https://www.oecd.org/en/data/datasets/talis-2018-database.html#data>

---

est	<i>Estimate list</i>
-----	----------------------

---

Description

Obtains a list with the arguments for the est() function used within the Rrepest() function.

Usage

```
est(statistic, target, regressor = NULL)
```

Arguments

- statistic (string vector) Statistics of interest that can include mean ("mean"), variance ("var"), standard deviation ("std"), quantile ("quant"), inter-quantile range ("iqr"), frequency count ("freq"), correlation ("corr"), linear regression ("lm"), covariance ("cov") and any other statistics that are not pre-programmed into Rrepest but take a data frame and weights as parameters ("gen")
- target (string vector) Variable(s) of interest of the estimation
- regressor (string vector) Independent variable(s) to be included in a linear regression

Value

List containing the arguments for the est() function used within Rrepest() function

**Examples**

```
est(c("mean", "quant", .5, "corr"), c("pv1math", "pv1read", "pv1scie"))
```

---

```
format_data_categ_vars
```

*Format categorical variables as factor for Rrepest*

---

**Description**

Format categorical variables as factor for Rrepest

**Usage**

```
format_data_categ_vars(df, categ.vars, show_na = F)
```

**Arguments**

df	(data frame) Data to analyse.
categ.vars	(string vector) Categorical variables for analysis.
show_na	(bool) Keeps NAs as categories to get frequency from.

**Value**

Data frame with categorical variables as factors for frequencies or over variables.

**Examples**

```
format_data_categ_vars(df = mtcars, categ.vars = "cyl")
```

---

```
format_data_cont_vars
```

*Format continuous variables as numeric for Rrepest*

---

**Description**

Format continuous variables as numeric for Rrepest

**Usage**

```
format_data_cont_vars(df, cont.vars)
```

**Arguments**

df	(data frame) Data to be analysed.
cont.vars	(string vector) continuous variables for analysis

**Value**

Data frame with continuous variables converted to numeric for a continuous analysis (means, regression, etc.)

**Examples**

```
format_data_cont_vars(mtcars, "hp")
```

---

format_data_repest	<i>Formatting target, by, and over variables for Rrepest.</i>
--------------------	---

---

**Description**

Formatting target, by, and over variables for Rrepest.

**Usage**

```
format_data_repest(df, svy, x, by.over, user_na = F, ...)
```

**Arguments**

df	(data frame) Data to analyze.
svy	(string) Possible projects to analyse: PIAAC, PISA, TALISSCH, TALISTCH, etc.
x	(string vector) Target variables.
by.over	(string vector) Variables to break analysis by.
user_na	(bool) TRUE → show nature of user defined missing values
...	Optional arguments such as custom weights (cm.weights)

**Value**

Data frame with variables in numeric format for analysis.

**Examples**

```
df1 <- format_data_repest(df_pisa18, "PISA", "pv1math", "cnt")
df2 <- format_data_repest(df_pisa18, "PISA", "pv1math", c("cnt", "st004d01t"))
df3 <- format_data_repest(df_pisa18, "PISA", "pv1math", c("cnt", "st004d01t", "isced1"))
df4 <- format_data_repest(df_talis18, "TALISTCH", "tt3g02", "cntry", isced = 2)
```



---

grouped_sum_freqs	<i>Grouped frequency counts</i>
-------------------	---------------------------------

---

**Description**

Computes a data frame with frequency counts.

**Usage**

```
grouped_sum_freqs(data, small.level, big.level, w = NULL)
```

```
grouped_sum_freqs(data, small.level, big.level, w = NULL)
```

**Arguments**

data	(data frame) Data to analyze.
small.level	(string vector) all variables to get grouped sum.
big.level	(string vector) Must be fully contained in variables from small.level
w	(string) Numeric variable from which to get weights (if NULL then 1).

**Value**

Data frame containing the frequency counts

Data frame with frequencies from the grouped sum of small.level and big.level used for getting percentages.

**Examples**

```
grouped_sum_freqs(data = mtcars, small.level = c("cyl", "am"), big.level = c("cyl"))
```

```
grouped_sum_freqs(data = mtcars, small.level = c("cyl", "gear"), big.level = c("cyl"))
```

---

grp	<i>Group list</i>
-----	-------------------

---

**Description**

Obtains a list with the arguments for the grp() function used within Rrepest() function's average() and group() options.

**Usage**

```
grp(group.name, column, cases, full_weight = FALSE)
```

**Arguments**

group.name	(string) Name of the group to be displayed
column	(string) Column/variable of interest to be grouped
cases	(string vector) Rows/values to be included in the group
full_weight	(bool) TRUE: average of the group will be weighted average

**Value**

List containing the arguments for the grp() function used within Rrepest() function's average() and group() options.

**Examples**

```
append(grp("OECD Average", "CNTRY", c("HUN", "MEX")), grp("Europe", "CNTRY", c("ITA", "FRA")))
```

---

indep_diff	<i>Independent Differences of columns</i>
------------	---

---

**Description**

Get in a dataframe all bivariate combinations of differences and standard errors from columns starting with b. and se. assuming independence. For a time series the vector must be order from oldest to newest.

**Usage**

```
indep_diff(df, vec_series)
```

**Arguments**

df	(dataframe) Dataframe that contains the columns with b. and se. on their name to get the difference from
vec_series	(numeric vector) Column names to get difference from, not including "b." at the start. Must have a column in dataframe also including "se.".

**Value**

Dataframe with extra columns containing the difference of the columns along with their standard error.

**Examples**

```
indep_diff(rrepest_pisa_age_isced, paste0("mean.age..ISCED level ", c("1", "3A", "2")))
```

---

 inv\_test

*inv\_test*


---

### Description

Invert test column from Rrepest test = TRUE by name on "b." and "se." in the column name and by sign (\*-1) on "b."

### Usage

```
inv_test(data, name_index)
```

### Arguments

data (dataframe) df to analyze

name\_index (string/numeric) name or index for the estimate (b.) columns containing the data for the test in Rrepest)

### Value

Dataframe cointaining inverted test column names for "b." and "se." according to Rrepest structure and column multiplied by (-1) for "b."

### Examples

```
inv_test(rrepest_pisa_age_gender, name_index = 6)
```

---

 is\_prime

*Check if a number is prime*


---

### Description

To check if a number n is prime, you only need to check for factors up to the square root of n. This is because if n has a factor greater than its square root, it must also have a smaller factor (since a factor is a number that divides n without leaving a remainder). This method significantly reduces the number of checks needed to determine if a number is prime.

### Usage

```
is_prime(n)
```

### Arguments

n (numeric) Number to verify if it prime

**Value**

(bool) TRUE if the number is prime, FALSE if the number is not prime

**Examples**

```
is_prime(0)
is_prime(1)
is_prime(7)
is_prime(10)
is_prime(100011869)
```

---

n_obs_x	<i>Number of valid (i.e. non-missing) observations for column/variable x</i>
---------	--

---

**Description**

Computes the number of valid (i.e. non-missing) observations for the column/variable of interest.

**Usage**

```
n_obs_x(df, by, x, svy = NULL)
```

**Arguments**

- df (data frame) Data to analyse with lowercase column names.
- by (string vector) Variable(s) used for tabulating the variable of interest
- x (string) Variable of interest for which to compute the number of valid (i.e. non-missing) observations
- svy (string) Survey settings that must be equal to one of the following: ALL, IALS, ICCS, ICILS, IELS, PBTS, PIAAC, PIRLS, PISA, PISA00S, PISA2015, SSES, SSES2023, SVY, TALISSCH, TALISTCH, TALISEC\_LEADER, TALISEC\_STAFF, TIMSS

**Value**

Data frame containing the number of valid (i.e. non-missing) observations for the variable of interest

**Examples**

```
n_obs_x(df = df_pisa18, by = "cnt", x = "wb173q03ha", svy = "PISA2015")
n_obs_x(df = df_talis18, by = "cntry", x = "tt3g01", svy = "TALISTCH")
```

---

paired_indep_diff	<i>Paired independent differences</i>
-------------------	---------------------------------------

---

**Description**

Get differences and standard errors from two columns starting with b. and se. assuming independence. The second column will be subtracted from the first.

**Usage**

```
paired_indep_diff(df, col1, col2)
```

**Arguments**

df	(dataframe) Dataframe that contains the columns with b. and se. on their name to get the difference from
col1	(numeric vector) Column name, not including "b.". Must have a column in dataframe also including "se.".
col2	(numeric vector) Column name, not including "b.". Must have a column in dataframe also including "se.".

**Value**

Dataframe with extra columns containing the difference of the columns along with their standard error.

**Examples**

```
paired_indep_diff(rrepest_pisa_age_gender, "mean.age..Male", "mean.age..Female")
```

---

Rrepest	<i>Estimation using replicate weights</i>
---------	---

---

**Description**

Estimates statistics using replicate weights (Balanced Repeated Replication (BRR) weights, Jackknife replicate weights,...), thus accounting for complex survey designs in the estimation of sampling variances. It is designed specifically to be used with the data sets produced by the Organization for Economic Cooperation and Development (OECD), some of which include the Programme for the International Assessment of Adult Competencies (PIAAC), Programme for International Student Assessment (PISA) and Teaching and Learning International Survey (TALIS) data sets, but works for any educational large-scale assessment and survey that uses replicated weights. It also allows for analyses with multiply imputed variables (plausible values); where plausible values are used, average estimator across plausible values is reported and the imputation error is added to the variance estimator.

**Usage**

```
Rrepest(
  data,
  svy,
  est,
  by = NULL,
  over = NULL,
  test = FALSE,
  user_na = FALSE,
  show_na = FALSE,
  flag = FALSE,
  fast = FALSE,
  tabl = FALSE,
  average = NULL,
  total = NULL,
  coverage = FALSE,
  invert_tests = FALSE,
  save_arg = FALSE,
  cores = NULL,
  ...
)
```

**Arguments**

<code>data</code>	(data frame) Data to analyse
<code>svy</code>	(string) Declares the survey settings. It must be equal to one of the following: ALL, IALS, ICCS, ICILS, IELS, PBTS, PIAAC, PIRLS, PISA, PISA00S, PISA2015, SSES, SSES2023, SVY, TALISSCH, TALISTCH, TALISEC_LEADER, TALISEC_STAFF, TIMSS.
<code>est</code>	(est function) Specifies the estimates of interest. It has three arguments: statistics type, target variable and an (optional) regressor list in case of a linear regression.
<code>by</code>	(string vector) Produces separate estimates by levels of the variable(s) specified by the string vector.
<code>over</code>	(string vector) Requests estimates to be obtained separately for each level of categorical variable(s) identified by the string vector.
<code>test</code>	(bool) if TRUE: Computes the difference between estimates obtained for the lowest and highest values of the 'over' variable(s). (See 'over' option above.) It is useful to test for differences between dependent samples (e.g. female-male).
<code>user_na</code>	(bool) if TRUE: Shows the nature of user defined missing values.
<code>show_na</code>	(bool) if TRUE: Includes missing values (i.e. NAs) when estimating frequencies for the variable of interest.
<code>flag</code>	(bool) if TRUE: Replaces estimation results that are based on fewer observations than required for reporting with NaN. When used with the PIAAC survey settings, it checks if each estimation result is based on at least 30 observations. When used with the PISA, PISA00S, PISA2015 survey settings, it checks if each estimation result is based on at least 30 observations and 5 schools. When

	used with the TALISSCH survey settings, it checks if each estimation result is based on at least 10 schools. When used with the TALISTCH survey settings, it checks if each estimation result is based on at least 30 observations and 10 schools.
<code>fast</code>	(bool) if TRUE: Computes estimates by using only 6 replicated weights.
<code>tabl</code>	(bool) if TRUE: Creates customisable and transferable tables using the flextable R package.
<code>average</code>	(grp function) Computes an arithmetic average (or weighted average). It has three arguments: name of the average, column/variable used for computing the average, rows/observations included in the average. It has three arguments: name of the group, column/variable used for computing the group, rows/observations included in the group.
<code>total</code>	(grp function) Computes an average weighted by the estimated size of the target population covered.
<code>coverage</code>	(bool/numeric) TRUE: shows column next to se. Numeric: Shows NaN if bellow the set coverage.
<code>invert_tests</code>	(bool) Invert test columns from Rrepest test = TRUE by name on "b." and "se." in the column name and by sign (*-1) on "b."
<code>save_arg</code>	(bool) TRUE: returns a named list with the estimation data frame and all arguments used in Rrepest.
<code>cores</code>	(numeric) NULL: Will recruit max-1 cores when doing PVs. Else, will recruit the specified number of cores for PVs
<code>...</code>	Other optional parameters include: <code>iscd</code> = Filters the data used for analysis by ISCED level (e.g. <code>iscd = 2</code> ), <code>n.pvs</code> = Customizes the number of plausible values used in the estimation (e.g. <code>n.pvs = 5</code> ), <code>cm.weights</code> = Customizes the weights used in the estimation (e.g. <code>cm.weights = c("finw", paste0("repw", 1:22))</code> ), <code>var.factor</code> = Customizes the variance factor used in the estimation (e.g. <code>var.factor = 1/(0.5^2)</code> ), <code>z.score</code> = <code>qnorm(1-0.05/2)</code>

## Value

Data frame containing estimation "b." and standard error "se."

## Examples

```
data(df_pisa18)

Rrepest(data = df_pisa18,
  svy = "PISA2015",
  est = est("mean", "AGE"),
  by = c("CNT"))
```

---

rrepest\_pisa\_age\_gender

*Rrepest table of results for PISA 2018 showing age and gender*


---

### Description

A table of results from Rrepest of the mean age of students broken down by gender for PISA 2018.

### Usage

```
rrepest_pisa_age_gender
```

### Format

```
rrepest_pisa_age_gender:
```

A data frame in tibble format with 3 rows and 7 columns:

**cnt** Country ISO 3166-1 alpha-3 codes.

**b.mean.age..Female** Mean age for female students.

**se.mean.age..Female** Standar error of the mean age for female students.

**b.mean.age..Male** Mean age for male students.

**se.mean.age..Male** Standar error of the mean age for male students.

**b.mean.age..(Female-Male)** Difference of the mean age from female to male students.

**se.mean.age..(Female-Male)** Standard error of the difference of the mean age from female to male students.

### Source

<https://www.oecd.org/en/data/datasets/pisa-2018-database.html#data>

---

rrepest\_pisa\_age\_isced

*Rrepest table of results for PISA 2018 showing the age and completed schooling level of students' mothers*


---

### Description

A table of results from Rrepest of the mean age of students broken down by completed schooling level of students' mothers for PISA 2018.

### Usage

```
rrepest_pisa_age_isced
```



**Format**

rrepest\_pisa\_age\_isced:

A data frame in tibble format with 3 rows and 7 columns:

**cnt** Country ISO 3166-1 alpha-3 codes.

**b.mean.age..ISCED level 1** Mean age of students whose mothers completed ISCED 1.

**se.mean.age..ISCED level 1** Standard error of the mean age of students whose mothers completed ISCED 1.

**b.mean.age..ISCED level 2** Mean age of students whose mothers completed ISCED 2.

**se.mean.age..ISCED level 2** Standard error of the mean age of students whose mothers completed ISCED 2.

**b.mean.age..ISCED level 3A** Mean age of students whose mothers completed ISCED 3A.

**se.mean.age..ISCED level 3A** Standard error of the mean age of students whose mothers completed ISCED 3A.

**b.mean.age..ISCED level 3B, 3C** Mean age of students whose mothers completed ISCED 3B/3C.

**se.mean.age..ISCED level 3B, 3C** Standard error of the mean age of students whose mothers completed ISCED 3B/3C.

**b.mean.age..She did not complete ISCED level 1** Mean age of students whose mothers did not complete ISCED 1.

**se.mean.age..She did not complete ISCED level 1** Standard error of the mean age of students whose mothers did not complete ISCED 1.

**b.mean.age..(ISCED level 1-She did not complete ISCED level 1)** Mean age difference between ISCED 1 completed vs. non-ISCED 1 completed mothers.

**se.mean.age..(ISCED level 1-She did not complete ISCED level 1)** Standard error of the mean age difference between ISCED 1 completed vs. non-ISCED 1 completed mothers. ...

**Source**

<https://www.oecd.org/en/data/datasets/pisa-2018-database.html#data>

---

tal18_tt3g23o_freq	<i>Rrepest table of results for TALIS 2018 showing a frequency for other areas of professional development</i>
--------------------	--

---

**Description**

A table of results from Rrepest for professional development broken down by completed schooling level of students' mothers for the Teaching and Learning International Survey (TALIS) 2018 cycle

**Usage**

tal18\_tt3g23o\_freq

**Format**

`talis18_tt3g23o_freq`:

A data frame in tibble format with 3 rows and 7 columns:

**cntry** Country ISO 3166-1 alpha-3 codes.

**b.tt3g23o.No** Percentage of teachers with No other areas of professional development

**se.tt3g23o.No** Standard error of the percentage of teachers with No other areas of professional development

**cvge.tt3g23o.No** Coverage of the percentage of teachers with No other areas of professional development

**b.tt3g23o.Yes** Percentage of teachers with other areas of professional development

**se.tt3g23o.Yes** Standard error of the percentage of teachers with other areas of professional development

**cvge.tt3g23o.Yes** Coverage of the percentage of teachers with other areas of professional development

**Source**

<https://www.oecd.org/en/data/datasets/talis-2018-database.html#data>

---

weighted.corr

*Weighted bivariate correlation*

---

**Description**

Computes the weighted Pearson correlation coefficient of two numeric vectors.

**Usage**

```
weighted.corr(x, y, w, na.rm = TRUE)
```

**Arguments**

<code>x</code>	(numeric vector) Variable of interest x for computing the correlation
<code>y</code>	(numeric vector) Variable of interest y for computing the correlation
<code>w</code>	(numeric vector) Vector with the weights
<code>na.rm</code>	(bool) if TRUE: Excludes missing values before computing the correlation

**Value**

Scalar containing the Pearson correlation coefficient

**Examples**

```
data(df_talis18)
```

```
weighted.corr(x = df_talis18$t3stake, y = df_talis18$t3team, w = df_talis18$tchwgt)
```

---

weighted.corr.cov.n      *Multivariate correlation and covariance*

---

### Description

Computes multivariate correlation and covariance for the variables of interest.

### Usage

```
weighted.corr.cov.n(
  data,
  x,
  w = rep(1, length(data[x[1]])),
  corr = TRUE,
  na.rm = TRUE
)
```

### Arguments

data	(data frame) Data to analyse
x	(string vector) Variables of interest for which to compute the correlation/covariance
w	(string) Name of the numeric variable representing the weights
corr	(bool) if TRUE: Computes correlation; if FALSE: Computes covariance
na.rm	(bool) if TRUE: Excludes missing values before computing the correlation/covariance.

### Value

Data frame containing each pairwise bivariate correlation/covariance

### Examples

```
data(df_talis18)

weighted.corr.cov.n(df_talis18, c("t3stake", "t3team", "t3stud"), "tchwgt")
```

---

weighted.cov      *Weighted bivariate covariance*

---

### Description

Computes the weighted covariance coefficient of two numeric vectors.

### Usage

```
weighted.cov(x, y, w, na.rm = TRUE)
```

**Arguments**

x	(numeric vector) Variable of interest x for computing the covariance
y	(numeric vector) Variable of interest y for computing the covariance
w	(numeric vector) Vector with the weights
na.rm	(bool) if TRUE: Excludes missing values before computing the covariance

**Value**

Scalar containing the covariance

**Examples**

```
data(df_talis18)

weighted.cov(x = df_talis18$t3stake, y = df_talis18$t3team, w = df_talis18$tchwgt)
```

---

weighted.iqr	<i>Weighted inter-quantile range</i>
--------------	--------------------------------------

---

**Description**

Computes the weighted inter-quantile range of a numeric vector.

**Usage**

```
weighted.iqr(x, w = rep(1, length(x)), rang = c(0.25, 0.75))
```

**Arguments**

x	(numeric vector) Variable of interest for which to compute the inter-quantile range
w	(numeric vector) Vector with the weights
rang	(numeric vector) Two numbers between 0 and 1 indicating the desired inter-quantile range

**Value**

Scalar containing the inter-quantile range

**Examples**

```
weighted.iqr(x = mtcars$mpg, w = mtcars$wt, rang = c(.5,.9))
```

---

weighted.mode	<i>Mode</i>
---------------	-------------

---

**Description**

Calculate the arithmetic mode of a vector. If multiple elements have the same frequency, all of them will be displayed.

**Usage**

```
weighted.mode(x, w = rep(1, length(x)))
```

**Arguments**

x	(numeric vector) vector from which we'll obtain the mode
w	(numeric vector) vector of weights. If not provided, it defaults to non-weighted mode.

**Value**

(numeric vector) one or multiple elements that will be the arithmetic mode

**Examples**

```
weighted.mode(c(1,2,3,4,5,4,3,4,5,3))
weighted.mode(c(NA,1,3,NA))
```

---

weighted.quant	<i>Weighted quantile</i>
----------------	--------------------------

---

**Description**

Computes weighted quantiles of a numeric vector.

**Usage**

```
weighted.quant(x, w = rep(1, length(x)), q = 0.5)
```

**Arguments**

x	(numeric vector) Variable of interest for which to compute the quantile
w	(numeric vector) Vector with the weights
q	(numeric vector) A number between 0 and less than 1 indicating the desired quantile

**Value**

Scalar containing the quantile

**Examples**

```
weighted.quant(x = mtcars$mpg, w = mtcars$wt, q = seq(.1,.9,.1))
```

---

weighted.std	<i>Weighted standard deviation</i>
--------------	------------------------------------

---

**Description**

Computes the weighted standard deviation of a numeric vector.

**Usage**

```
weighted.std(x, w, na.rm = TRUE)
```

**Arguments**

x	(numeric vector) Variable of interest for which to compute the standard deviation.
w	(numeric vector) Vector with the weights.
na.rm	(bool) if TRUE: Excludes missing values before computing the standard deviation

**Value**

Scalar containing the standard deviation

**Examples**

```
data(df_talis18)
weighted.std(df_talis18$TT3G02, df_talis18$TRWGT1)
```

---

weighted.var	<i>Weighted variance</i>
--------------	--------------------------

---

**Description**

Computes the weighted variance of a numeric vector.

**Usage**

```
weighted.var(x, w, na.rm = TRUE)
```

**Arguments**

x	(numeric vector) Variable of interest for which to compute the variance
w	(numeric vector) Vector with weights
na.rm	(bool) if TRUE: Excludes missing values before computing the variance

**Value**

Scalar containing the variance

**Examples**

```
data(df_talis18)

weighted.var(df_talis18$TT3G02, df_talis18$TRWGT1)
```

# Index

- \* **datasets**
  - df\_pisa18, [5](#)
  - df\_talis18, [5](#)
  - rrepest\_pisa\_age\_gender, [16](#)
  - rrepest\_pisa\_age\_isced, [16](#)
  - talis18\_tt3g23o\_freq, [17](#)
- average\_groups, [2](#)
- coverage\_daggers, [3](#)
- coverage\_pct, [4](#)
- df\_pisa18, [5](#)
- df\_talis18, [5](#)
- est, [6](#)
- format\_data\_categ\_vars, [7](#)
- format\_data\_cont\_vars, [7](#)
- format\_data\_repest, [8](#)
- grouped\_sum\_freqs, [9](#)
- grp, [9](#)
- indep\_diff, [10](#)
- inv\_test, [11](#)
- is\_prime, [11](#)
- n\_obs\_x, [12](#)
- paired\_indep\_diff, [13](#)
- Rrepest, [13](#)
- rrepest\_pisa\_age\_gender, [16](#)
- rrepest\_pisa\_age\_isced, [16](#)
- talis18\_tt3g23o\_freq, [17](#)
- weighted.corr, [18](#)
- weighted.corr.cov.n, [19](#)
- weighted.cov, [19](#)
- weighted.iqr, [20](#)
- weighted.mode, [21](#)
- weighted.quant, [21](#)
- weighted.std, [22](#)
- weighted.var, [23](#)