

Package ‘directadjusting’

July 14, 2026

Type Package

Title Directly Adjusted Estimates

Version 0.7.0

Description Compute estimates and confidence intervals of weighted averages quickly and easily. Weighted averages are computed using `data.table` for speed. Confidence intervals are approximated using the delta method with either using known formulae or via algorithmic or numerical derivation.

License MIT + file LICENSE

URL <https://github.com/FinnishCancerRegistry/directadjusting>

BugReports <https://github.com/FinnishCancerRegistry/directadjusting/issues>

Depends R (>= 2.10)

Imports `data.table`, `stats`

Config/roxygen2/version 8.0.0

Encoding UTF-8

Language en-GB

NeedsCompilation no

Author Joonas Miettinen [cre, aut] (ORCID:
<<https://orcid.org/0000-0001-8624-6754>>)

Maintainer Joonas Miettinen <joonas.miettinen@cancer.fi>

Repository CRAN

Date/Publication 2026-07-14 09:50:02 UTC

Contents

confidence_intervals	2
directly_adjusted_estimates	4
Index	11

confidence_intervals	<i>Confidence Intervals</i>
----------------------	-----------------------------

Description

Functions to compute confidence intervals.

Usage

```
delta_method_confidence_intervals(  
  statistics,  
  variances,  
  conf_lvl = 0.95,  
  conf_method = "identity"  
)
```

Arguments

- | | |
|-------------|--|
| statistics | [numeric] (no default)
Statistics for which to calculate confidence intervals. |
| variances | [numeric] (no default)
Variance estimates of statistics used to compute confidence intervals. |
| conf_lvl | [numeric] (default 0.95)
Confidence level of confidence intervals in]0, 1[. |
| conf_method | [character, call, list] (default "identity")
Delta method transformation to be applied. <ul style="list-style-type: none">• character: Use one of the pre-defined transformations. Table of options with the corresponding expressions: |

name	g	g_inv	g_gradient
identity	theta	g	1
log	log(theta)	exp(g)	1/theta
log-log	log(-log(theta))	exp(-exp(g))	1/(theta * log(theta))
logit	log(theta) - log(1 - theta)	1/(1 + exp(-g))	1/(theta - theta^2)

- call: A quoted R expression which produces the lower / upper limit when evaluated. E.g. `quote(theta * exp(z * theta_standard_error / theta))`.
- list: Contains both the transformation and its inverse. E.g. `list(g = quote(log(theta)), g_inv = quote(exp(g)))`.

Value

directadjusting::delta_method_confidence_intervals

Returns a `data.table` with columns `c("statistic", "variance", "ci_lo", "ci_hi")`.

Functions

directadjusting::delta_method_confidence_intervals

`directadjusting::delta_method_confidence_intervals` can be used to compute confidence intervals using the delta method. The following steps are performed:

- Compute confidence intervals based on `conf_method`, `statistics`, `variances`, and `conf_lvl`.
 - If `conf_method` is a string, a pre-defined set of mathematical expressions are used to compute the confidence intervals.
 - If `conf_method` is a call, it is evaluated with the variables `theta`, `theta_variance`, `theta_standard_error`, and `z`. This is done once for the lower and once for the upper bound of the confidence interval, so for the lower bound and `conf_level = 0.95` we use `z = stats::qnorm(p = (1 - conf_lvl) / 2)`.
 - If `conf_method` is a list, it must contain elements `g` and `g_inv`, e.g. `list(g = quote(log(theta)), g_inv = quote(exp(g)))`.
 - * `g` is passed to `[stats::deriv]`. If that fails, a numerical derivative is computed.
 - * With the derivative known the variance after the transformation is `variance * g_gradient ^ 2`.
 - * With the transformed variance known the transform confidence interval is calculated simply via `g(theta) + g_standard_error * z`.
 - * These transformation-scale confidence intervals are then converted back to the original scale using `g_inv`.
- Collect a `data.table` with the confidence intervals and with also the columns `statistics = statistics` and `variance = variances`.
- Add attribute named `ci_meta` to the `data.table`. This attribute is a list which contains elements `conf_lvl` and `conf_method`.
- Return `data.table` with columns `c("statistic", "variance", "ci_lo", "ci_hi")`.

Examples

```
# directadjusting::delta_method_confidence_intervals
dt_1 <- directadjusting::delta_method_confidence_intervals(
  statistics = 0.9,
  variances = 0.1,
  conf_lvl = 0.95,
  conf_method = "log"
)

# you can also supply your own math for computing the confidence intervals
dt_2 <- directadjusting::delta_method_confidence_intervals(
  statistics = 0.9,
  variances = 0.1,
  conf_lvl = 0.95,
  conf_method = quote(theta * exp(z * theta_standard_error / theta))
)

dt_3 <- directadjusting::delta_method_confidence_intervals(
  statistics = 0.9,
  variances = 0.1,
```

```

    conf_lvl = 0.95,
    conf_method = list(
      g = quote(log(theta)),
      g_inv = quote(exp(g))
    )
  )

dt_4 <- directadjusting::delta_method_confidence_intervals(
  statistics = 0.9,
  variances = 0.1,
  conf_lvl = 0.95,
  conf_method = list(
    g = quote(stats::qnorm(theta)),
    g_inv = quote(stats::pnorm(g))
  )
)
stopifnot(
  all.equal(dt_1, dt_2, check.attributes = FALSE),
  all.equal(dt_1, dt_3, check.attributes = FALSE)
)

```

directly_adjusted_estimates

Directly Adjusted Estimates

Description

Compute direct adjusted estimates from a table of statistics.

Usage

```

directly_adjusted_estimates(
  stats_dt,
  stat_col_nms,
  var_col_nms,
  stratum_col_nms = NULL,
  adjust_col_nms = NULL,
  conf_lvls = 0.95,
  conf_methods = "identity",
  weights = NULL
)

```

Arguments

stats_dt	[data.frame] (no default) a data.frame containing estimates and variance estimates of statistics
stat_col_nms	[character] (no default) names of columns in stats_dt containing estimates (statistics); NA statistics values cause also NA confidence intervals

var_col_nms	[character] (default NULL) • if NULL, no confidence intervals can (will) be computed • if character vector, names of columns in stats_dt containing variance estimates of the statistics specified in stat_col_nms with one-to-one correspondence; NA elements in var_col_nms cause no confidence intervals to be computed for those statistics; NA variance estimates in stats_dt cause NA confidence intervals; negative values cause an error; Inf values cause c(-Inf, Inf) intervals with confidence interval method "identity", etc.
stratum_col_nms	[NULL, character] (default NULL) names of columns in stats_dt by which statistics are stratified (and they should be stratified by these columns after direct adjusting)
adjust_col_nms	[NULL, character] (default NULL) Names of columns in stats_dt by which statistics are currently stratified and by which the statistics should be adjusted (e.g. "agegroup"). • NULL: If weights is a data.table, use its stratifying columns. Else no adjusting is performed. • character: Adjust by these columns.
conf_lvls	[numeric] (default 0.95) confidence levels for confidence intervals; you may specify each statistic (see stat_col_nms) its own level by supplying a vector of values; values other than between (0, 1) cause an error
conf_methods	[character, list] (default "identity") Method(s) to compute confidence intervals. Either one method for all stats (stat_col_nms) or otherwise this must be of length (length(stat_col_nms)). Each element is passed to [delta_method_confidence_intervals] separately. Can also be "none": This causes no confidence intervals to be calculated for the respective stat_col_nms element(s).
weights	[double, data.table, character] The weights need not sum to one as this is ensured internally. You may supply weights in one of the following ways: • double: A vector of weights, the length of which must match the number of strata defined by adjusting variables. • data.table: With one or more columns with names matching to those variables that are used to adjust estimates, and one column named weight. E.g. data.table(agegroup = 1:3, weight = c(100, 500, 400)).

Details

directadjusting::directly_adjusted_estimates computes weighted averages and their confidence intervals. Performs the following steps:

- Makes a new data.table with data from stats_dt without copying any column data to avoid modifying stats_dt itself.
- Handle argument weights in order to produce a data.table of weights if it wasn't one already.

- Insert the weights into stats_dt.
 - Weights are merged into stats_dt in-place by making a left join of stats_dt and weights_dt.
 - Re-scale weights to sum to one within each stratum defined by stratum_col_nms.
- Compute weighted averages of stat_col_nms and var_col_nms (the latter with squared weights because they are variances) over adjust_col_nms. This results in a data.table without column(s) adjust_col_nms.
- For each statistic:
 - If the corresponding conf_methods element is "none", skip confidence intervals computation.
 - Otherwise call [delta_method_confidence_intervals].
- Set attribute directly_adjusted_estimates_meta. It is a list containing:
 - call: The call to directadjusting::directly_adjusted_estimates.
 - stat_col_nms: The argument as given by the user.
 - var_col_nms: The argument as given by the user.
 - stratum_col_nms: The argument as given by the user.
 - adjust_col_nms: The argument as given by the user.
 - conf_lvls: The argument, but always of length length(stat_col_nms).
 - conf_methods: The argument, but always of length length(stat_col_nms).
 - meta_dt: Contains columns
 - * stat_col_nm: Elements of argument stat_col_nms.
 - * stat_col_nm_w: Corresponding name of column containing weighted statistics. These columns names are in the output of directadjusting::directly_adjusted_estimates.
 - * var_col_nm: Elements of argument var_col_nms.
 - * var_col_nm_w: Like stat_col_nm_w but for variance column names.
 - * conf_method: Corresponding conf_methods elements.
 - * conf_lvl: Corresponding conf_lvls elements.
 - * ci_lo_col_nm_w: Name of confidence interval lower bound of the weighted statistic.
 - * ci_hi_col_nm_w: Name of confidence interval upper bound of the weighted statistic.
- Return a data.table. Returned columns are those given via stratum_col_nms, stat_col_nms, and var_col_nms and additional confidence interval columns. If weights was a data.table, the names of the stats, variances, and confidence bounds columns are identified by the respective suffix of the weight column, e.g. stat_col_nms = c("s", "r") and weights = data.table(age_group = 1:4, weight_1 = 1:4, weight2 = 4:1) will result in output columns s_1, s2, r_1, and r2 etc.

Value

Returns a data.table. Returned columns are those given via stratum_col_nms, stat_col_nms, and var_col_nms and additional confidence interval columns. If weights was a data.table, the names of the stats, variances, and confidence bounds columns are identified by the respective suffix of the weight column, e.g. stat_col_nms = c("s", "r") and weights = data.table(age_group = 1:4, weight_1 = 1:4, weight2 = 4:1) will result in output columns s_1, s2, r_1, and r2 etc.

News for version 0.7.0

`directadjusting::directly_adjusted_estimates` argument `adjust_col_nms` is now automatically determined based on `weights` if `weights` is a `data.frame` / `data.table`.

`directadjusting::directly_adjusted_estimates` can now handle `weights data.table` with more than one column containing weights, e.g. `weight_europe`, `weight_world`.

Examples

```
# directadjusting::directly_adjusted_estimates
library("data.table")
set.seed(1337)

offsets <- rnorm(8, mean = 1000, sd = 100)
baseline <- 100
hrs_by_sex <- rep(1:2, each = 4)
hrs_by_ag <- rep(c(0.75, 0.90, 1.10, 1.25), times = 2)
counts <- rpois(8, baseline * hrs_by_sex * hrs_by_ag)

# raw estimates
my_stats <- data.table::data.table(
  sex = rep(1:2, each = 4),
  ag = rep(1:4, times = 2),
  e = counts / offsets,
  v = counts / (offsets**2)
)

# adjusted by age group
my_adj_stats <- directly_adjusted_estimates(
  stats_dt = my_stats,
  stat_col_nms = "e",
  var_col_nms = "v",
  conf_lvls = 0.95,
  conf_methods = "log",
  stratum_col_nms = "sex",
  adjust_col_nms = "ag",
  weights = c(200, 300, 400, 100)
)

# adjusted by smaller age groups, stratified by larger age groups
my_stats[, "ag2" := c(1, 1, 2, 2, 1, 1, 2, 2)]
my_adj_stats <- directly_adjusted_estimates(
  stats_dt = my_stats,
  stat_col_nms = "e",
  var_col_nms = "v",
  conf_lvls = 0.95,
  conf_methods = "log",
  stratum_col_nms = c("sex", "ag2"),
  adjust_col_nms = "ag",
  weights = c(200, 300, 400, 100)
)
```

```

# with no adjusting columns defined you get the same table as input
# but with confidence intervals. this for the sake of
# convenience for programming cases where sometimes you want to adjust,
# sometimes not.
stats_dt_2 <- data.table::data.table(
  sex = 0:1,
  e = 0.0,
  v = 0.1
)
dt_2 <- directadjusting::directly_adjusted_estimates(
  stats_dt = stats_dt_2,
  stat_col_nms = "e",
  var_col_nms = "v",
  conf_lvls = 0.95,
  conf_methods = "identity",
  stratum_col_nms = "sex"
)
stopifnot(
  dt_2[["e"]] == stats_dt_2[["e"]],
  dt_2[["v"]] == stats_dt_2[["v"]],
  dt_2[["sex"]] == stats_dt_2[["sex"]]
)

# sometimes when adjusting rates or counts, there can be strata where the
# statistic is zero. these should be included in your statistics dataset
# if you still want the weighted average be influenced by the zero.
# otherwise you will get the wrong result. sometimes when naively tabulating
# a dataset with e.g. dt[, .N, keyby = "stratum"] one does not get a result
# row for a stratum that does not appear in the dataset even if we know that
# the stratum exists, for instance only the age groups 1-17 are present in
# the dataset.
stats_dt_3 <- data.table::data.table(
  age_group = 1:18,
  count = 17:0,
  var = 17:0
)

# this goes as intended
dt_3 <- directadjusting::directly_adjusted_estimates(
  stats_dt = stats_dt_3,
  stat_col_nms = "count",
  var_col_nms = "var",
  stratum_col_nms = NULL,
  adjust_col_nms = "age_group",
  weights = data.table::data.table(
    age_group = 1:18,
    weight = 18:1
  )
)

# this does not
dt_4 <- directadjusting::directly_adjusted_estimates(
  stats_dt = stats_dt_3[1:17, ],

```

```

    stat_col_nms = "count",
    var_col_nms = "var",
    stratum_col_nms = NULL,
    adjust_col_nms = "age_group",
    weights = data.table::data.table(
      age_group = 1:18,
      weight = 18:1
    )
  )

# the weighted average that included the zero is smaller
stopifnot(
  dt_3[["count"]] < dt_4[["count"]]
)

# NAs are allowed and produce in turn NAs silently.
stats_dt_5 <- data.table::data.table(
  age_group = 1:18,
  count = c(NA, 16:0),
  var = c(NA, 16:0)
)
dt_5 <- directadjusting::directly_adjusted_estimates(
  stats_dt = stats_dt_5,
  stat_col_nms = "count",
  var_col_nms = "var",
  adjust_col_nms = "age_group",
  weights = data.table::data.table(
    age_group = 1:18,
    weight = 18:1
  )
)
stopifnot(
  is.na(dt_5)
)

stats_dt_6 <- data.table::data.table(
  age_group = 1:4,
  survival = c(0.20, 0.40, 0.60, 0.80),
  survival_var = 0.05^2
)
# you can use conf_methods to pass whatever to
# `delta_method_confidence_intervals`.
dt_6 <- directadjusting::directly_adjusted_estimates(
  stats_dt = stats_dt_6,
  stat_col_nms = "survival",
  var_col_nms = "survival_var",
  adjust_col_nms = "age_group",
  weights = data.table::data.table(
    age_group = 1:4,
    weight = 1:4
  ),
  conf_methods = list(
    list(

```

```

      g = quote(stats::qnorm(theta)),
      g_inv = quote(stats::pnorm(g))
    )
  )
)

# the wame weights data.table can contain more than one weight column
dt_7 <- directadjusting::directly_adjusted_estimates(
  stats_dt = stats_dt_6,
  stat_col_nms = "survival",
  var_col_nms = "survival_var",
  adjust_col_nms = "age_group",
  weights = data.table::data.table(
    age_group = 1:4,
    weight_1 = 1:4,
    weight2 = 4:1
  )
)
stopifnot(
  c("survival_1", "survival2") %in% names(dt_7)
)

# ... and multiple statistics columns as well
data.table::set(
  x = stats_dt_6,
  j = c("risk", "risk_var"),
  value = list(
    1 - stats_dt_6[["survival"]],
    stats_dt_6[["survival_var"]]
  )
)
dt_8 <- directadjusting::directly_adjusted_estimates(
  stats_dt = stats_dt_6,
  stat_col_nms = c("survival", "risk"),
  var_col_nms = c("survival_var", "risk_var"),
  adjust_col_nms = "age_group",
  weights = data.table::data.table(
    age_group = 1:4,
    weight_1 = 1:4,
    weight2 = 4:1
  )
)
stopifnot(
  c("survival_1", "survival2", "risk_1", "risk2") %in% names(dt_8)
)

```

Index

confidence_intervals, [2](#)

delta_method_confidence_intervals
(confidence_intervals), [2](#)

directly_adjusted_estimates, [4](#)