

# Package ‘directadjusting’

May 8, 2026

**Type** Package

**Title** Directly Adjusted Estimates

**Version** 0.6.1

**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 integration.

**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`

**Encoding** UTF-8

**Language** en-GB

**RoxygenNote** 7.3.3

**NeedsCompilation** no

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

**Maintainer** Joonas Miettinen <[joonas.miettinen@cancer.fi](mailto:joonas.miettinen@cancer.fi)>

**Repository** CRAN

**Date/Publication** 2026-02-04 19:30:02 UTC

## Contents

|                                       |   |
|---------------------------------------|---|
| confidence_intervals . . . . .        | 2 |
| directly_adjusted_estimates . . . . . | 4 |

|              |           |
|--------------|-----------|
| <b>Index</b> | <b>10</b> |
|--------------|-----------|

---

confidence\_intervals *Confidence Intervals*

---

## 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"> <li>character: Use one of the pre-defined transformations. Table of options with the corresponding expressions:</li> </ul> |

| 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_lvl1 = 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_lvl1 = 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)<br>a data.frame containing estimates and variance estimates of statistics   |
| stat_col_nms | [character] (no default)<br>names of columns in stats_dt containing estimates (statistics); NA statistics values cause also NA confidence intervals |

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

## 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.
- Handles argument `weights` in order to produce a `data.table` of weights if it wasn't one already.
- Inserts the weights into `stats_dt`.

- Weights are merged into `stats_dt` in-place by making a left join on `weights_dt` using `stats_dt` and adding column `weight` resulting from this join into `stats_dt`.
- Re-scale weights to sum to one within each stratum defined by `stratum_col_nms`.
- Computes 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 `i` in `seq_along(stat_col_nm)`:
  - If `conf_methods[[i]]` is "none", doesn't compute confidence intervals.
  - Otherwise calls `[delta_method_confidence_intervals]`.
- Sets 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)`.
- Returns a `data.table`. Returned columns are those given via `stratum_col_nms`, `stat_col_nms`, and `var_col_nms`.

### Value

Returns a `data.table`. Returned columns are those given via `stratum_col_nms`, `stat_col_nms`, and `var_col_nms`.

### 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),
  var = 0.05 ^ 2
)

# you can use conf_method 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 = "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))
    )
  )
)
```

# Index

confidence\_intervals, 2

delta\_method\_confidence\_intervals  
(confidence\_intervals), 2

directly\_adjusted\_estimates, 4