Package: ggmeta (via r-universe)

October 22, 2026

Title Publication-Quality Forest and Funnel Plots with 'ggplot2'

Version 0.1.1.9000

Description A 'ggplot2' extension that creates publication-quality forest and funnel plots from 'meta' package objects or tidy data frames. Provides custom 'ggproto' geometries for study-level confidence intervals with weight-proportional squares, summary effect diamonds, prediction intervals, null-effect reference lines, and funnel-plot contours. Supports subgroup analysis, back-transformation of summary measures, on-the-fly effect pooling, effect and weight table columns, and journal-specific layout presets (JAMA, BMJ, RevMan5).

License GPL (>= 3)

URL https://github.com/drhrf/ggmeta, https://drhrf.github.io/ggmeta/

BugReports https://github.com/drhrf/ggmeta/issues

Depends R (>= 4.0)

Imports cli (>= 3.6.0), ggplot2 (>= 3.4.0), grid, rlang (>= 1.0.0), scales (>= 1.2.0)

Suggests covr (>= 3.6.0), knitr (>= 1.40), meta (>= 4.0), patchwork (>= 1.1.0), pkgdown (>= 2.0.0), rmarkdown (>= 2.20), testthat (>= 3.0.0), vdiffr (>= 1.0.0)

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

Config/testthat/edition 3

Config/Needs/website pkgdown

Config/roxygen2/version 8.1.0

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

Date/Publication 2026-09-22 15:31:38 UTC

RemoteUrl https://github.com/drhrf/ggmeta

RemoteRef HEAD

RemoteSha ba914f78b553dfdd8a3dbefb1c2c48536b56cb3b

Contents

format_effect . . . 2
fortify.meta . . . 3
geom_forest_ci . . . 4
geom_forest_diamond . . . 6
geom_forest_predict . . . 8
geom_forest_ref . . . 11
geom_forest_text . . . 12
geom_funnel_contour . . . 15
ggforest . . . 16
ggfunnel . . . 19
layout_bmj . . . 20
layout_jama . . . 21
layout_revman5 . . . 21
theme_forest . . . 22
theme_funnel . . . 23
tidy_meta . . . 23

Index **26**

`format_effect` *Format an effect estimate and confidence interval as text*

Description

A small helper for building the label of a `geom_forest_text()` column, producing strings such as `"1.40 (0.65 to 3.00)"`.

Usage

```
format_effect(estimate, ci_lower, ci_upper, digits = 2, sep = " to ")
```

Arguments

`estimate`, `ci_lower`, `ci_upper`	Numeric vectors of equal length.
`digits`	Number of decimal places. Default `2`.
`sep`	Separator between the confidence limits. Default `" to "`.

Value

A character vector; `NA` estimates become empty strings.

Examples

```
format_effect(c(1.4, 0.9), c(0.65, 0.6), c(3.0, 1.35))
```

`fortify.meta` *Fortify a 'meta' object for use with 'ggplot2'*

Description

This S3 method for `ggplot2::fortify()` converts **meta** objects to a tidy data frame. It is a wrapper around `tidy_meta()`.

Usage

```
## S3 method for class 'meta'
fortify(model, data, ...)
```

Arguments

`model`	An object of class `meta`.
`data`	Ignored. Included for S3 generic compatibility.
`...`	Additional arguments passed to `tidy_meta()`.

Value

A `data.frame` as returned by `tidy_meta()`.

Examples

```
if (requireNamespace("meta", quietly = TRUE)) {
library(meta)
library(ggplot2)
m <- metabin(event.e, n.e, event.c, n.c,
  data = data.frame(
    event.e = c(14, 30), n.e = c(100, 150),
    event.c = c(10, 25), n.c = c(100, 150)
  ),
  studlab = c("Study A", "Study B"),
  sm = "RR"
)
library(ggmeta)
ggplot(fortify(m), aes(x = estimate, y = studlab)) +
  geom_point()
}
```

geom_forest_ci *Study-level confidence intervals and point estimates for forest plots*

Description

geom_forest_ci() draws a confidence interval line and a weight-proportional square for each study in a forest plot. The square area scales with the study weight, making more precise studies visually prominent.

Usage

```
geom_forest_ci(
  mapping = NULL,
  data = NULL,
  stat = "forest_ci",
  position = "identity",
  ...,
  ci_width = 0.3,
  point_size_range = c(1, 6),
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following: • A Stat ggproto subclass, for example StatCount. • A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count".

- For more information and other ways to specify the stat, see the layer stat documentation.

`position` A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following:

- The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`.
- For more information and other ways to specify the position, see the layer position documentation.

`...` Other arguments passed on to `ggplot2::layer()`. These are often aesthetics, used to set an aesthetic to a fixed value, like `colour = "red"` or `linewidth = 1`.

`ci_width` Width of the CI line end-marks as a proportion of the spacing between study rows. Default: `0.3`.

`point_size_range` Minimum and maximum point size in mm. Default: `c(1, 6)`. The side of each square is `min + (max - min) * sqrt(weight / max(weight))`, i.e. the part above the minimum grows with the square root of the weight. Missing or non-positive weights get the minimum size; when no study has a usable weight there is nothing to scale, so every square is drawn at the middle of the range (as `meta::forest()` does for a `method = "GLMM"` fit). A constant `size` (or a mapped `size` aesthetic) overrides the weight-based size.

`na.rm` If `FALSE` (default), missing values are removed with a warning. If `TRUE`, missing values are silently removed.

`show.legend` logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`inherit.aes` If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `annotation_borders()`.

Value

A ggplot2 layer.

Aesthetics

`geom_forest_ci()` understands the following aesthetics (required aesthetics are in **bold**):

- `x` — point estimate
- `xmin` — lower confidence limit
- `xmax` — upper confidence limit

- y — study position (usually a factor)
- weight — study weight for square sizing
- colour — line and square border colour (default: "black")
- fill — square fill colour (default: "black")
- alpha — transparency (default: 1)
- linewidth — CI line width (default: 0.5)
- size — override for square size; computed from weight by default
- linetype — CI line type (default: "solid")

Examples

```
library(ggplot2)
df <- data.frame(
  study = c("Study 1", "Study 2", "Study 3"),
  estimate = c(0.5, 0.8, 0.3),
  lower = c(0.2, 0.6, 0.1),
  upper = c(0.8, 1.0, 0.5),
  weight = c(1, 2, 0.5)
)
ggplot(df, aes(y = study, x = estimate, xmin = lower, xmax = upper,
               weight = weight)) +
  geom_forest_ref() +
  geom_forest_ci()
```

---

geom_forest_diamond *Summary effect diamond for forest plots*

Description

geom_forest_diamond() draws a diamond (lozenge) shape representing a summary effect estimate in a forest plot. The left and right tips mark the confidence limits and the widest point marks the estimate.

Usage

```
geom_forest_diamond(
  mapping = NULL,
  data = NULL,
  stat = "forest_diamond",
  position = "identity",
  ...,
  diamond_height = 0.4,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following: • A Stat ggproto subclass, for example StatCount. • A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count". • For more information and other ways to specify the stat, see the layer stat documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
...	Other arguments passed on to ggplot2::layer().
diamond_height	Height of the diamond as a proportion of the spacing between study rows. Default: 0.4.
na.rm	If FALSE (default), missing values are removed with a warning. If TRUE, missing values are silently removed.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. annotation_borders().

Value

A ggplot2 layer.

Aesthetics

`geom_forest_diamond()` understands the following aesthetics (required aesthetics are in **bold**):

- `x` — point estimate (diamond center)
- `xmin` — lower confidence limit (left tip)
- `xmax` — upper confidence limit (right tip)
- `y` — vertical position
- `colour` — diamond border colour (default: "black")
- `fill` — diamond fill colour (default: "black")
- `alpha` — fill transparency (default: 0.5)
- `linewidth` — border width (default: 0.5)
- `linetype` — border line type (default: "solid")

Examples

```
library(ggplot2)
df <- data.frame(
  model = c("Common effect", "Random effects"),
  estimate = c(0.22, 0.22),
  lower = c(-0.08, -0.08),
  upper = c(0.52, 0.52)
)
ggplot(df, aes(y = model, x = estimate, xmin = lower, xmax = upper)) +
  geom_forest_diamond(aes(fill = model))
```

---

geom_forest_predict *Prediction interval display for forest plots*

Description

`geom_forest_predict()` draws a prediction interval around a random-effects summary estimate. The prediction interval is wider than the confidence interval and represents the range where the true effect in a new study is expected to fall.

Usage

```
geom_forest_predict(
  mapping = NULL,
  data = NULL,
  stat = "forest_predict",
  position = "identity",
```

```
  ...,
  cap_width = 0.32,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following: • A Stat ggproto subclass, for example StatCount. • A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count". • For more information and other ways to specify the stat, see the layer stat documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
...	Other arguments passed on to ggplot2::layer().
cap_width	Width of the capped end-marks as a proportion of row spacing. Default: 0.32.
na.rm	If FALSE (default), missing values are removed with a warning. If TRUE, missing values are silently removed.

show.legend — logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.

inherit.aes — If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. annotation_borders().

Value

A ggplot2 layer.

Aesthetics

geom_forest_predict() understands the following aesthetics (required aesthetics are in **bold**):

- x — point estimate (typically the random effects estimate)
- xmin — lower prediction limit
- xmax — upper prediction limit
- y — vertical position
- colour — interval colour (default: "#6F9FBE")
- linewidth — interval line width (default: 0.45)
- linetype — interval line type (default: "dashed")
- alpha — transparency (default: 0.55)

Examples

```
library(ggplot2)
df <- data.frame(
  estimate = c(0.22),
  lower = c(-0.27),
  upper = c(0.70),
  model = c("Random effects")
)
ggplot(df, aes(y = model, x = estimate, xmin = lower, xmax = upper)) +
  geom_forest_diamond() +
  geom_forest_predict()
```

geom_forest_ref *Reference line at null effect for forest plots*

Description

`geom_forest_ref()` draws a vertical reference line at the null effect value — typically 0 for difference measures (MD, SMD) or 1 for ratio measures (RR, OR, HR). The line indicates where no treatment effect exists.

Usage

```
geom_forest_ref(
  mapping = NULL,
  data = NULL,
  ...,
  xintercept = 0,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

`mapping`	Set of aesthetic mappings created by `aes()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
`data`	The data to be displayed in this layer. There are three options: If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`. A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created. A `function` will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A `function` can be created from a `formula` (e.g. `~ head(.x, 10)`).
`...`	Other arguments passed on to `ggplot2::layer()`.
`xintercept`	X-axis intercept for the reference line. Default: `0` (null effect for difference measures).
`na.rm`	If `FALSE` (default), missing values are removed with a warning. If `TRUE`, missing values are silently removed.
`show.legend`	logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`inherit.aes` If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `annotation_borders()`.

Value

A ggplot2 layer.

Aesthetics

`geom_forest_ref()` understands the following aesthetics:

- `colour` — line colour (default: "gray50")
- `linewidth` — line width (default: 0.5)
- `linetype` — line type (default: "dashed")
- `alpha` — transparency (default: 0.8)

Examples

```
library(ggplot2)
df <- data.frame(
  study = c("Study 1", "Study 2"),
  estimate = c(0.5, 0.8),
  lower = c(0.2, 0.6),
  upper = c(0.8, 1.0),
  weight = c(1, 2)
)
ggplot(df, aes(y = study, x = estimate, xmin = lower,
               xmax = upper, weight = weight)) +
  geom_forest_ref(xintercept = 0) +
  geom_forest_ci()
```

---

geom_forest_text *Add an aligned text column to a forest plot*

Description

`geom_forest_text()` places a column of text labels aligned with the study rows of a forest plot — for example event counts, sample sizes, weights, or the effect estimate rendered as text. Rows align automatically through the shared `y` (study) aesthetic; the horizontal position of the column is set with the `x` argument. Place a column beside the data by widening the panel with `ggplot2::expand_limits()`.

Usage

```
geom_forest_text(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  x = NULL,
  hjust = 0,
  size = 3.2,
  na.rm = TRUE,
  show.legend = FALSE,
  inherit.aes = FALSE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following: • A Stat ggproto subclass, for example StatCount. • A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count". • For more information and other ways to specify the stat, see the layer stat documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter".

- For more information and other ways to specify the position, see the layer position documentation.

`...`	Other arguments passed on to `ggplot2::layer()`, often used to set an aesthetic to a fixed value, e.g. `colour = "grey30"` or `fontface = "bold"`.
`x`	Fixed horizontal position for the column, in x-axis data units. If `NULL`, `x` must be supplied through `mapping`.
`hjust`	Horizontal justification. Default `0` (left-aligned), so a column reads cleanly from its `x` position rightward.
`size`	Text size in millimetres. Default `3.2`.
`na.rm`	If `TRUE` (default), missing labels are dropped silently.
`show.legend`	logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.
`inherit.aes`	If `FALSE` (default) the layer does not inherit the forest plot's `x`/`xmin`/`xmax`/`weight` mapping, so only `y` and `label` need to be supplied.

Value

A ggplot2 layer.

Aesthetics

`geom_forest_text()` understands the following aesthetics (required aesthetics are in **bold**):

- `y` — row position (map to the same study variable as the forest layers)
- `label` — text to display
- `colour` — text colour (default: "black")
- `size` — text size; set via the `size` argument
- `fontface`, `family`, `angle`, `alpha`

Examples

```
library(ggplot2)
df <- data.frame(
  study = c("A", "B", "C"),
  estimate = c(0.5, 0.8, 0.3),
  lower = c(0.2, 0.6, 0.1),
  upper = c(0.8, 1.0, 0.5),
  n = c(120, 240, 90)
)
ggplot(df, aes(y = study, x = estimate, xmin = lower, xmax = upper)) +
  geom_forest_ci() +
  geom_forest_text(aes(y = study, label = n), x = 1.15) +
  expand_limits(x = 1.25)
```

`geom_funnel_contour` *Pseudo-confidence-interval contours for a funnel plot*

Description

`geom_funnel_contour()` draws the funnel-shaped pseudo confidence interval(s) around a reference effect. For a study with standard error `se`, the expected effect under no bias lies within `centre` +/- z * `se`, so the contours are straight lines fanning out from the apex at `se = 0`. On a funnel plot with a reversed standard-error axis they form the familiar funnel.

Usage

```
geom_funnel_contour(
  centre,
  se_max,
  level = 0.95,
  ...,
  colour = "grey55",
  linetype = "dashed",
  linewidth = 0.4,
  alpha = NA
)
```

Arguments

`centre`	Reference effect (usually the pooled estimate) the funnel is centred on, on the analysis scale.
`se_max`	Largest standard error the contours should reach (the bottom of the funnel).
`level`	Confidence level(s) for the contour(s), e.g. `0.95` or `c(0.95, 0.99)`. Default `0.95`.
`...`	Other arguments passed on to `ggplot2::layer()`.
`colour`, `linetype`, `linewidth`, `alpha`	Line appearance.

Details

The layer builds its own data from `centre`, `se_max`, and `level`, so it does not use the plot's data; add it to a plot that maps the effect to `x` and the standard error to `y`.

Value

A ggplot2 layer.

Examples

```
library(ggplot2)
studies <- data.frame(
  estimate = c(-0.5, -0.2, -0.4, 0.1, -0.3),
  se       = c(0.10, 0.30, 0.15, 0.35, 0.22)
)
ggplot(studies, aes(x = estimate, y = se)) +
  geom_funnel_contour(centre = -0.3, se_max = 0.4) +
  geom_point() +
  scale_y_reverse()
```

`ggforest` *Create a forest plot*

Description

The primary function for creating publication-quality forest plots. Accepts a **meta** object (created by `meta::metabin()`, `meta::metacont()`, etc.) or a tidy data frame (as returned by `tidy_meta()` or constructed manually).

Usage

```
ggforest(x, ...)

## Default S3 method:
ggforest(x, ...)

## S3 method for class 'meta'
ggforest(
  x,
  ...,
  back_trans = c("auto", "exp", "none"),
  sort_studies = TRUE,
  show_summary = TRUE,
  show_predict = TRUE,
  show_hetstats = TRUE,
  null_effect = NULL,
  xlab = NULL
)

## S3 method for class 'data.frame'
ggforest(
  x,
  null_effect = 0,
  xlab = "Effect (95% CI)",
  ylab = NULL,
  title = NULL,
```

```
  caption = NULL,
  add_summary = FALSE,
  summary_method = c("common", "random"),
  level = 0.95,
  columns = NULL,
  effect_header = NULL,
  ci_args = list(),
  diamond_args = list(),
  diamond_colours = NULL,
  predict_args = list(),
  ref_args = list(),
  consensus = TRUE,
  consensus_args = list(),
  ...
)
```

Arguments

x	A **meta** object or a tidy data frame with columns `estimate`, `ci_lower`, `ci_upper`, `studlab`, and optionally `weight`, `is_summary`, `summary_type`, `subgroup`.
...	Additional arguments passed to methods.
back_trans	Back-transform ratio measures? `"auto"` (default), `"exp"`, or `"none"`.
sort_studies	Sort studies by effect estimate? Default: `TRUE`.
show_summary	Include summary effect rows? Default: `TRUE`.
show_predict	Include prediction interval? Default: `TRUE` (only when random effects model is present).
show_hetstats	Show heterogeneity statistics in plot caption? Default: `TRUE`.
null_effect	Null effect value for the reference line. Default: `0`.
xlab	X-axis label. Default: `"Effect (95% CI)"`.
ylab	Y-axis label. Default: `NULL` (no label — study labels serve as the y-axis text).
title	Plot title. Default: `NULL`.
caption	Plot caption. Default: `NULL`.
add_summary	For the data-frame method, if `TRUE` compute a pooled summary from the study rows (inverse-variance and/or DerSimonian-Laird) and draw it as a diamond — on-the-fly meta-analysis without the **meta** package. Needs a `se` column, or `ci_lower`/`ci_upper` (a 95% Wald interval) to recover it. When `null_effect = 1` (ratio measures), estimates and limits are taken to be on the ratio scale and are pooled on the log scale; a supplied `se` must then be the standard error of the log estimate. The `weight` column only sizes the squares and does not enter the pooled estimate. Studies with a missing estimate or a non-positive standard error are excluded with a message. Default: `FALSE`.
summary_method	Which pooled summaries to add when `add_summary = TRUE`: `"common"`, `"random"`, or both (default).
level	Confidence level for the pooled summary interval. Default `0.95`.

`columns` Add a `meta::forest()`-style table of text columns to the right of the plot. `TRUE` shows the effect estimate, 95% CI, and weight; or pass a subset/order such as `c("estimate", "ci")`. `NULL` (default) draws no columns.

`effect_header` Header for the estimate column (e.g. `"Hedges' g"`). Defaults to the summary measure (e.g. `"SMD"`, `"RR"`).

`ci_args, diamond_args, predict_args`
Lists of arguments used to restyle the study confidence intervals (`geom_forest_ci()`), the summary diamonds (`geom_forest_diamond()`), and the prediction interval (`geom_forest_predict()`). For example `predict_args = list(cap_width = 0.1, colour = "red")` or `ci_args = list(colour = "grey20", point_size_range = c(1, 5))`. This is the way to customise these elements: adding another `geom_forest_*()` layer to a `ggforest()` plot draws a *second* layer over every row rather than restyling the built-in one.

`diamond_colours`
Optional named colours for the summary diamonds, overriding the default palette. Names are `"common"`, `"random"`, `"subgroup_common"`, and `"subgroup_random"`, e.g. `c(common = "black", random = "steelblue")`.

`ref_args, consensus_args`
Lists of arguments for the null-effect reference line and the dotted "consensus" line (both `geom_forest_ref()`), e.g. `ref_args = list(linetype = "dashed", colour = "black")`.

`consensus` Draw the dotted consensus line at the pooled estimate? Default `TRUE` (only shown alongside a null line).

Value

A `ggplot` object. Add standard ggplot2 layers (themes, scales, labels) to further customize.

Examples

```
if (requireNamespace("meta", quietly = TRUE)) {
library(meta)
m <- metabin(event.e, n.e, event.c, n.c,
  data = data.frame(
    event.e = c(14, 30, 15, 22),
    n.e     = c(100, 150, 100, 120),
    event.c = c(10, 25, 12, 18),
    n.c     = c(100, 150, 100, 120)
  ),
  studlab = c("Study A", "Study B", "Study C", "Study D"),
  sm = "RR"
)
ggforest(m)
}
```

ggfunnel *Funnel plot from a meta-analysis*

Description

ggfunnel() draws a funnel plot: each study's effect estimate against its standard error (a measure of precision), with pseudo confidence-interval contours around the pooled effect. Asymmetry can signal small-study effects or publication bias. Like ggforest(), it accepts a **meta** object or a tidy data frame and returns an ordinary ggplot, so a forest and a funnel plot compose on one canvas with, for example, **patchwork**.

Usage

```
ggfunnel(x, ...)

## Default S3 method:
ggfunnel(x, ...)

## S3 method for class 'meta'
ggfunnel(x, ..., ref = c("common", "random"), level = 0.95)

## S3 method for class 'data.frame'
ggfunnel(
  x,
  centre = NULL,
  sm = NULL,
  level = 0.95,
  xlab = NULL,
  ylab = "Standard error",
  title = NULL,
  point_args = list(),
  contour_args = list(),
  ref_args = list(),
  ...
)
```

Arguments

x	A **meta** object, or a data frame with estimate and se columns (study effects and standard errors on the analysis scale).
...	Additional arguments passed to methods.
ref	Which pooled estimate to centre the funnel on: "common" (fixed effect, default) or "random".
level	Confidence level(s) for the funnel contours, e.g. 0.95 or c(0.95, 0.99). Default 0.95.

centre — Effect the funnel is centred on. Defaults to the inverse-variance (common-effect) estimate of the supplied studies.

sm — Summary measure (e.g. "RR", "PLOGIT"), used to label the x-axis and, for transformed measures (ratios, proportions, rates, correlations), to show back-transformed axis labels. Optional.

xlab, ylab — Axis labels. ylab defaults to "Standard error".

title — Plot title. Default NULL.

point_args, contour_args, ref_args — Lists of arguments used to restyle the study points (ggplot2::geom_point()), the funnel contours (geom_funnel_contour()), and the vertical reference line (ggplot2::geom_vline()). For example point_args = list(size = 3, fill = "black") or contour_args = list(colour = "grey70", linetype = "dotted").

Value

A ggplot object.

Examples

```
if (requireNamespace("meta", quietly = TRUE)) {
library(meta)
m <- metabin(event.e, n.e, event.c, n.c,
  data = data.frame(
    event.e = c(12, 8, 25, 18, 30, 15), n.e = c(120, 90, 200, 150, 250, 130),
    event.c = c(20, 14, 30, 28, 35, 25), n.c = c(118, 92, 205, 148, 245, 128)
  ),
  studlab = paste("Study", 1:6), sm = "RR"
)
ggfunnel(m)
}
```

---

layout_bmj *Apply BMJ-style formatting*

Description

Modifies a forest plot to match BMJ (British Medical Journal) style conventions.

Usage

```
layout_bmj(p)
```

Arguments

p — A ggplot object (typically from ggforest()).

Value

A ggplot object with BMJ-style formatting applied.

`layout_jama` *Apply JAMA-style formatting*

Description

Modifies a forest plot to match JAMA (Journal of the American Medical Association) style conventions.

Usage

```
layout_jama(p)
```

Arguments

`p`	A ggplot object (typically from `ggforest()`).

Value

A ggplot object with JAMA-style formatting applied.

`layout_revman5` *Apply RevMan5-style formatting*

Description

Modifies a forest plot to match Cochrane RevMan 5 style conventions.

Usage

```
layout_revman5(p)
```

Arguments

`p`	A ggplot object (typically from `ggforest()`).

Value

A ggplot object with RevMan5-style formatting applied.

`theme_forest` *Forest plot theme*

Description

A complete ggplot2 theme optimized for forest plots. Removes unnecessary grid lines, adjusts margins, and sets sensible defaults for forest plot aesthetics.

Usage

```
theme_forest(
  base_size = 11,
  base_family = "",
  base_line_size = base_size/22,
  base_rect_size = base_size/22
)
```

Arguments

`base_size`	Base font size in pts. Default: `11`.
`base_family`	Base font family. Default: `""`.
`base_line_size`	Base line size. Default: `base_size / 22`.
`base_rect_size`	Base rect size. Default: `base_size / 22`.

Value

A ggplot2 theme object (a list of class `"theme"`).

Examples

```
library(ggplot2)
df <- data.frame(
  study = c("Study 1", "Study 2"),
  estimate = c(0.5, 0.3),
  lower = c(0.2, 0.1),
  upper = c(0.8, 0.5),
  weight = c(1, 2)
)
ggplot(df, aes(y = study, x = estimate, xmin = lower,
               xmax = upper, weight = weight)) +
  geom_forest_ref() +
  geom_forest_ci() +
  theme_forest()
```

theme_funnel *Funnel plot theme*

Description

A light ggplot2 theme for `ggfunnel()` plots.

Usage

```
theme_funnel(base_size = 11, base_family = "")
```

Arguments

`base_size`	Base font size in pts. Default 11.
`base_family`	Base font family. Default `""`.

Value

A ggplot2 theme.

Examples

```
library(ggplot2)
df <- data.frame(estimate = c(-0.3, 0.1, -0.2), se = c(0.1, 0.3, 0.2))
ggplot(df, aes(estimate, se)) +
  geom_point() +
  scale_y_reverse() +
  theme_funnel()
```

tidy_meta *Tidy a 'meta' object into a plottable data frame*

Description

Converts objects of class `meta` (from the **meta** package) into a tidy data frame suitable for use with ggmeta geometries and `ggforest()`. The returned data frame has one row per study or summary.

Usage

```
tidy_meta(x, ...)

## Default S3 method:
tidy_meta(x, ...)

## S3 method for class 'data.frame'
tidy_meta(
```

```
  x,
  add_summary = FALSE,
  summary_method = c("common", "random"),
  level = 0.95,
  log_scale = FALSE,
  ...
)

## S3 method for class 'meta'
tidy_meta(
  x,
  back_trans = c("auto", "exp", "none"),
  sort_studies = TRUE,
  add_summary = TRUE,
  add_predict = TRUE,
  add_subgroups = TRUE,
  ...
)
```

Arguments

x	An object of class meta, e.g. created by meta::metabin(), meta::metacont(), or meta::metagen().
...	Additional arguments passed to methods.
add_summary	For the data-frame method, if TRUE compute an inverse-variance / DerSimonian-Laird pooled summary from the study rows and append it (on-the-fly meta-analysis, no **meta** package required). Needs a se column, or ci_lower/ci_upper to recover it. Default FALSE.
summary_method	Which pooled summaries to add when add_summary = TRUE: "common", "random", or both (default).
level	Confidence level for the pooled summary interval. Default 0.95.
log_scale	For the data-frame method with add_summary = TRUE: are the effects ratios (RR, OR, HR, ...) given on the natural scale? If TRUE, they are pooled on the log scale and the summary is exponentiated back; a supplied se must be the standard error of the log estimate. Default FALSE.
back_trans	Should the estimate and confidence limits be back-transformed to the natural scale? If "auto" (default), each summary measure is back-transformed with its correct inverse via meta::backtransf() (exponentiation for ratios, inverse-logit for PLOGIT, Fisher's z to correlation for ZCOR, etc.); linear measures are left unchanged. Study rows of meta::metaprop() and meta::metarate() objects get the observed event / n or event / time instead, as in meta::forest(), because the stored TE of a study with zero or all events is continuity-corrected and so disagrees with its exact confidence limits. Use "exp" to force exponentiation or "none" to keep the analysis scale.
sort_studies	If TRUE (default), sort studies by effect estimate (most favorable at top).
add_predict	If TRUE (default), include prediction interval row when available.

add_subgroups If TRUE (default), include subgroup headers and within-group summary rows when subgroups are present.

Value

A data.frame with one row per study or summary, with columns:

studlab Study label (character)

estimate Point estimate (numeric)

ci_lower Lower confidence limit (numeric)

ci_upper Upper confidence limit (numeric)

se Standard error (numeric)

weight Study weight (numeric, NA for summaries, and for every study of a method = "GLMM" fit, which uses none)

p_value P-value (numeric)

n Sample size or person-time (numeric, optional)

event Number of events (numeric, optional)

is_summary Logical, TRUE for summary rows

summary_type Character: "none", "common", "random", "subgroup", or "predict"

subgroup Subgroup label (character or NA)

The returned data frame has the following attributes:

sm summary measure type (e.g. "RR", "OR", "MD")

null_effect null effect value for reference line

method meta-analysis method

common logical, TRUE if common effect model was used

random logical, TRUE if random effects model was used

tau heterogeneity estimate tau

k number of studies

Examples

```
if (requireNamespace("meta", quietly = TRUE)) {
library(meta)
m <- metabin(event.e, n.e, event.c, n.c,
  data = data.frame(
    event.e = c(14, 30), n.e = c(100, 150),
    event.c = c(10, 25), n.c = c(100, 150)
  ),
  studlab = c("Study A", "Study B"),
  sm = "RR"
)
tidy_meta(m)
}
```

Index

aes(), *4*, *7*, *9*, *11*, *13*
annotation_borders(), *5*, *7*, *10*, *12*

format_effect, 2
fortify(), *4*, *7*, *9*, *11*, *13*
fortify.meta, 3

geom_forest_ci, 4
geom_forest_ci(), *18*
geom_forest_diamond, 6
geom_forest_diamond(), *18*
geom_forest_predict, 8
geom_forest_predict(), *18*
geom_forest_ref, 11
geom_forest_ref(), *18*
geom_forest_text, 12
geom_funnel_contour, 15
geom_funnel_contour(), *20*
GeomForestCI *(*geom_forest_ci*)*, 4
GeomForestDiamond
        *(*geom_forest_diamond*)*, 6
GeomForestPredict
        *(*geom_forest_predict*)*, 8
GeomForestRef *(*geom_forest_ref*)*, 11
ggforest, 16
ggforest(), *19–21*, *23*
ggfunnel, 19
ggfunnel(), *23*
ggplot(), *4*, *7*, *9*, *11*, *13*
ggplot2::expand_limits(), *12*
ggplot2::fortify(), *3*
ggplot2::geom_point(), *20*
ggplot2::geom_vline(), *20*
ggplot2::layer(), *5*, *7*, *9*, *11*, *14*, *15*

layer position, *5*, *7*, *9*, *14*
layer stat, *5*, *7*, *9*, *13*
layout_bmj, 20
layout_jama, 21
layout_revman5, 21

meta::backtransf(), *24*
meta::forest(), *24*
meta::metabin(), *16*, *24*
meta::metacont(), *16*, *24*
meta::metagen(), *24*
meta::metaprop(), *24*
meta::metarate(), *24*

StatForestCI *(*geom_forest_ci*)*, 4
StatForestDiamond
        *(*geom_forest_diamond*)*, 6
StatForestPredict
        *(*geom_forest_predict*)*, 8
StatForestRef *(*geom_forest_ref*)*, 11

theme_forest, 22
theme_funnel, 23
tidy_meta, 23
tidy_meta(), *3*, *16*