Package: plotbb (via r-universe)

October 18, 2026

Title Grammar of Graphics on 'base' Graphics

Version 1.0.0

Description Implements a grammar of graphics on top of the 'base' graphics system. A 'bbplot' object is built incrementally with a '+' operator from data, aesthetic mappings and layers, then rendered with base graphics, so plotbb layers can be mixed with ordinary base plotting calls. The package provides layers for points, lines, bars, histograms, densities, boxplots, violins, ribbons, smooths, error bars and tiles, together with colour and other aesthetic scales, automatic legends, faceting, themes, coordinate flipping and significance annotations.

Depends R (>= 4.1.0)

Imports graphics, stats, rlang, scales, methods, utils

Suggests ape, aplot (>= 0.0.8), dplyr, ggplotify (>= 0.1.0), quarto, knitr, testthat

VignetteBuilder quarto

ByteCompile true

License Artistic-2.0

Encoding UTF-8

Config/roxygen2/version 8.1.1

NeedsCompilation no

Author Guangchuang Yu [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-6485-8781>)

Maintainer Guangchuang Yu <guangchuangyu@gmail.com>

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

Date/Publication 2026-10-18 11:00:08 UTC

RemoteUrl https://github.com/cran/plotbb

RemoteRef HEAD

RemoteSha b95cffb6b1c78ae44246ddd0267c10fab97acb78

Contents

as.bbplot . . . 2
bb_aes . . . 3
bb_bar . . . 4
bb_build . . . 9
bb_coord_flip . . . 10
bb_facet_grid . . . 10
bb_facet_wrap . . . 11
bb_grid . . . 12
bb_group_values . . . 13
bb_labs . . . 14
bb_layer . . . 15
bb_legend . . . 16
bb_position . . . 18
bb_save . . . 18
bb_scale_col_palette . . . 20
bb_scale_fill_palette . . . 21
bb_scale_pch_manual . . . 23
bb_scale_x_continuous . . . 24
bb_theme . . . 25
bbplot . . . 26
bbplot_add . . . 27
set_bb_theme . . . 28
unset_bb_theme . . . 29
with_bb_theme . . . 29

Index **31**

`as.bbplot` *as.bbplot*

Description

convert a base plot function to a bbplot object

Usage

```
as.bbplot(fun)
```

Arguments

`fun`	a function that plot something in base graphics

Details

the base plot function will be plotted as a canvas and users can apply theme and add layers to it

Value

A bbplot object

Author(s)

Guangchuang Yu

---

bb_aes *bb_aes*

Description

aesthetic mapping

Usage

```
bb_aes(x, y, ...)
```

Arguments

x	x variable
y	y variable
...	other mappings

Details

aesthetic mapping for bbplot

Value

enquo expression describing variables mapping to aesthetic attributes

Author(s)

Guangchuang Yu

bb_bar *bb_boxplot*

Description

boxplot layer

density layer

histogram layer

layer

reference lines

ribbon layer

significance layer

smooth layer

violin layer

Usage

```
bb_bar(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  width = 0.9,
  inherit = TRUE,
  orientation = c("vertical", "horizontal"),
  ...
)

bb_boxplot(
  mapping = NULL,
  data = NULL,
  inherit = TRUE,
  orientation = c("vertical", "horizontal"),
  ...
)

bb_density(
  mapping = NULL,
  data = NULL,
  inherit = TRUE,
  orientation = c("vertical", "horizontal"),
  ...
)

bb_errorbar(
```

```
  mapping = NULL,
  data = NULL,
  orientation = "y",
  width = 0.5,
  inherit = TRUE,
  ...
)

bb_hist(
  mapping = NULL,
  data = NULL,
  inherit = TRUE,
  orientation = c("vertical", "horizontal"),
  ...
)

bb_line(mapping = NULL, data = NULL, inherit = TRUE, ...)

bb_lm(
  mapping = NULL,
  data = NULL,
  se = TRUE,
  level = 0.95,
  inherit = TRUE,
  ...
)

bb_point(
  mapping = NULL,
  data = NULL,
  position = "identity",
  inherit = TRUE,
  ...,
  seed = 1
)

bb_hline(
  yintercept = NULL,
  mapping = NULL,
  data = NULL,
  inherit = TRUE,
  expand_limits = FALSE,
  ...
)

bb_vline(
  xintercept = NULL,
  mapping = NULL,
```

```
  data = NULL,
  inherit = TRUE,
  expand_limits = FALSE,
  ...
)

bb_ribbon(
  mapping = NULL,
  data = NULL,
  inherit = TRUE,
  orientation = c("vertical", "horizontal"),
  ...
)

bb_segment(mapping = NULL, data = NULL, inherit = TRUE, ...)

bb_signif(mapping = NULL, data = NULL, inherit = TRUE, ...)

bb_smooth(
  mapping = NULL,
  data = NULL,
  inherit = TRUE,
  orientation = c("vertical", "horizontal"),
  ...
)

bb_text(mapping = NULL, data = NULL, inherit = TRUE, ...)

bb_tile(mapping = NULL, data = NULL, inherit = TRUE, ...)

bb_violin(
  mapping = NULL,
  data = NULL,
  inherit = TRUE,
  orientation = c("vertical", "horizontal"),
  ...
)
```

Arguments

`mapping`	aesthetic mapping
`data`	layer data
`stat`	one of 'identity' or 'count'
`width`	bar width or errorbar cap width
`inherit`	whether the layer inherits the plot mapping. With `inherit = FALSE` the layer uses only its own `mapping`, so a mapping that is not repeated locally is unavailable; the layer's `data` is still inherited from `bbplot()` unless given explicitly.

`orientation`	‘"vertical"‘ (default; ‘x‘ groups, ‘y‘ values) or ‘"horizontal"‘ (‘x‘ values, ‘y‘ groups).
`...`	additional parameters: ‘width‘ (default 0.9), ‘adjust‘ (1), ‘bw‘, ‘kernel‘, ‘trim‘, ‘n‘, ‘alpha‘, ‘border‘, ‘fill‘/‘col‘.
`se`	display confidence interval around smooth? (TRUE by default, see ggplot2)
`level`	level of confidence interval to use (0.95 by default)
`position`	one of ’identity’ or ’jitter’
`seed`	random seed used by ‘position = "jitter"‘. Defaults to 1, so that jittered positions are reproducible; set `seed = NULL` to jitter from the current random state. The caller’s random number stream is never modified.
`yintercept`	numeric y position(s) for horizontal lines
`expand_limits`	whether the intercept extends the axis range
`xintercept`	numeric x position(s) for vertical lines

Details

bbplot layers

‘bb_hline()‘ and ‘bb_vline()‘ draw horizontal and vertical reference lines. The position is either a constant vector (‘yintercept‘ / ‘xintercept‘) or a mapping of the same name, so thresholds can come from the data as well.

By default a reference line does **not** change the axis range: a line at the edge of the data should not silently extend the canvas. Set ‘expand_limits = TRUE‘ when the intercept is meant to be inside the visible range even though the data do not reach it.

A mapped intercept wins over the constant argument. With a mapped intercept, ‘col‘ and ‘lty‘ may also be mapped and are then applied per line; with a constant intercept those aesthetics are layer-wide constants.

When the layer brings its own data, inherited mappings are evaluated on that data too; use ‘inherit = FALSE‘ when the plot maps aesthetics the reference lines do not carry.

The layers follow the grammar contract: they inherit data and mapping, accept their own, draw in ‘+‘ order, honour constant ‘col‘, ‘lty‘ and ‘lwd‘, and are drawn once per facet panel.

‘bb_ribbon()‘ draws a band between ‘ymin‘ and ‘ymax‘ along ‘x‘, one polygon per group. It follows the grammar contract: it inherits the plot data and mapping, accepts its own, draws in ‘+‘ order, honours constants and contributes ‘x‘, ‘ymin‘ and ‘ymax‘ to the ranges.

Rows with a missing or non-finite ‘x‘, ‘ymin‘ or ‘ymax‘ are dropped, and so are rows with a missing group (with a warning). Within a group the points are sorted by ‘x‘, duplicates are kept, and a group needs at least two points to form a polygon (smaller groups are skipped with a warning). ‘ymin‘ and ‘ymax‘ may be given in either order: the band is drawn between them. ‘alpha‘ (between 0 and 1), ‘border‘, ‘lty‘ and ‘lwd‘ are layer-wide single values; ‘fill‘ (or ‘col‘) sets the interior and can be a constant or a mapping.

When the layer brings its own data, inherited mappings are still evaluated on that data: give the layer the variables it inherits, or set ‘inherit = FALSE‘ when the plot has aesthetics the band does not carry.

‘bb_smooth()‘ fits an explicit model per group and draws the fitted curve, optionally with a confidence band. The fit method is never implicit: choose ‘"lm"‘ or ‘"loess"‘ (the default), pass extra fitting arguments through ‘method.args‘, and control the band with ‘se‘ and ‘level‘.

‘x‘ and ‘y‘ are mapped from the layer’s data, ‘group‘ splits the fit, and a mapped ‘col‘ colours each curve. The ‘formula‘ response must be ‘y‘; transform the data or the mapping rather than the response, so the band stays on the drawn scale. The confidence band uses ‘fill‘ (or the curve colour at low alpha). Small samples that cannot support the requested fit are reported with a warning and fall back to a linear fit; a group with fewer than two points is skipped. Fits are computed when the layer is added, so repeated draws do not re-fit.

‘bb_violin()‘ draws one density polygon per group: ‘x‘ selects the group (a factor, character or numeric position), ‘y‘ is the numeric variable whose distribution is estimated with ‘stats::density()‘. The density is scaled to ‘width‘ in x units, so groups stay comparable; ‘adjust‘ and ‘bw‘ control the bandwidth, ‘kernel‘, ‘trim‘ and ‘n‘ are passed to ‘density()‘.

The layer follows the grammar contract: it inherits data and mapping, accepts its own, draws in ‘+‘ order, honours constant or mapped ‘fill‘/‘col‘, and contributes the group positions and the numeric values to the ranges. With ‘orientation = "horizontal"‘, map ‘x‘ to the numeric values and ‘y‘ to the groups; the default vertical orientation is the other way round. A group needs at least two finite values and a non-zero spread; other groups are skipped with a warning. Rows with a non-finite value are dropped.

Value

A modified bbplot object

A modified bbplot object

A modified bbplot object

A modified bbplot object

A modified bbplot object

A modified bbplot object

A modified bbplot object

A modified bbplot object

A modified bbplot object

Author(s)

Guangchuang Yu

Examples

```
library(plotbb)
p <- bbplot(mtcars, bb_aes(mpg, disp, col=factor(cyl))) +
  bb_point() + bb_lm(bb_aes(group=cyl), lwd=2)
```

bb_build *bb_build*

Description

Build a plot without drawing it

Usage

```
bb_build(plot)
```

Arguments

plot	a bbplot object

Details

‘bb_build()‘ performs the computation that ‘print()‘ would perform, without opening a device and without drawing: it trains the scales, resolves the mappings, collects each layer’s data and computes the limits (and the per-panel limits of a faceted plot). The result is an ordinary list, meant for inspection and for checking a plot before it is rendered.

The build is side-effect free with respect to graphics: it does not call ‘plot()‘, does not change ‘par()‘ and does not touch ‘.Random.seed‘ (jitter is applied when the plot is drawn, not when it is built). Training stores the trained scales in the plot’s internal environment; that is plot state, not graphics state. Repeated calls on the same object return the same structure and limits.

Value

A list with `data`, `mapping` (labels), `layers` (name, kind, data, mapping labels, inherit, parameter names), `scales` (trained), `limits`, `panels` (`NULL` or one entry per facet with its key, row count and limits), `labs`, `position` (the positional-scale summary), `facet` (a serializable summary: variables, scales, nrow, ncol and free_colour) and `warnings` collected while building.

See Also

bb_save

Examples

```
p <- bbplot(mtcars, bb_aes(mpg, disp, col = factor(cyl))) + bb_point()
str(bb_build(p), max.level = 2)
```

bb_coord_flip *bb_coord_flip*

Description

Flip the positional axes

Usage

```
bb_coord_flip()
```

Details

‘bb_coord_flip()‘ swaps the x and y axes. It is a coordinate operation, not a data transformation: axes labels, positional scales and the titles swap, and the supported layers are redrawn with x/y swapped.

Supported layers are ‘bb_point()‘, ‘bb_line()‘, ‘bb_text()‘, ‘bb_segment()‘, ‘bb_tile()‘, ‘bb_layer()‘, ‘bb_violin()‘, ‘bb_boxplot()‘, ‘bb_bar()‘, ‘bb_errorbar()‘, ‘bb_ribbon()‘, ‘bb_density()‘, ‘bb_hist()‘ and ‘bb_smooth()‘ (these switch to the opposite ‘orientation‘), plus constant or mapped ‘bb_hline()‘/‘bb_vline()‘ (which become the other reference line). Layers that have no orientation model (‘bb_lm()‘, ‘bb_signif()‘) raise an error rather than drawing in the wrong orientation.

Value

A modified bbplot object

Author(s)

Guangchuang Yu

bb_facet_grid *bb_facet_grid*

Description

Facet a bbplot object into a row/column grid.

Usage

```
bb_facet_grid(
  rows = NULL,
  cols = NULL,
  scales = "fixed",
  labeller = NULL,
  drop = TRUE,
  strip_width = NULL,
  free_colour = FALSE
)
```

Arguments

`rows`	a one-sided formula for the row variable, e.g. `~cyl`.
`cols`	a one-sided formula for the column variable, e.g. `~am`.
`scales`	scale behavior across panels, one of 'fixed', 'free', 'free_x', or 'free_y'. With free scales, x is shared within a column and y within a row, matching the grid layout.
`labeller`	optional function applied to each row/column key.
`drop`	if FALSE, unused factor levels still get panels.
`strip_width`	optional positive integer; wraps long strip labels.
`free_colour`	if TRUE, each panel trains its own col/fill domain from its rows (requires an explicit colour scale; an outside legend is then an error because it cannot describe several domains).

Details

At least one of `rows` and `cols` is required. Panels are the full combination of the row and column keys; a combination without rows is drawn empty rather than skipped. Row labels are placed at the right, column labels at the top. For `bb_build()`, each panel is reported with its row and column key, label, row count and limits.

Value

A modified bbplot object

Author(s)

Guangchuang Yu

`bb_facet_wrap` *bb_facet_wrap*

Description

Facet a bbplot object into multiple panels.

Usage

```
bb_facet_wrap(
  facets,
  ncol = NULL,
  nrow = NULL,
  scales = "fixed",
  labeller = NULL,
  drop = TRUE,
  strip_width = NULL,
  free_colour = FALSE
)
```

Arguments

`facets`	a formula specifying faceting variable, e.g. `~cyl`.
`ncol`	number of columns.
`nrow`	number of rows.
`scales`	scale behavior across facets, one of 'fixed', 'free', 'free_x', or 'free_y'.
`labeller`	optional function applied to each panel key; the result is the strip label.
`drop`	if FALSE, unused factor levels still get a panel. Such a panel has no rows and falls back to the global limits, including on free scales.
`strip_width`	optional positive integer; wraps long strip labels with `strwrap()`.
`free_colour`	if TRUE, each panel trains its own col/fill domain from its rows (requires an explicit colour scale; an outside legend is then an error because it cannot describe several domains).

Details

`nrow` and `ncol` must be positive integers, and their product must be at least the number of panels; a layout that is too small is reported instead of silently overwriting a panel. By default facets are built from the keys that occur in the data, so unused factor levels do not create empty panels; set `drop = FALSE` to keep them.

Value

A modified bbplot object

Author(s)

Guangchuang Yu

`bb_grid` *bb_grid*

Description

add grid lines

Usage

```
bb_grid(col = "lightgray", lty = par("lty"), lwd = par("lwd"), bg = TRUE)
```

Arguments

`col`	line color
`lty`	line type
`lwd`	line width
`bg`	whether plot the grid lines as background

Value

A modified bbplot object

Author(s)

Guangchuang Yu

---

bb_group_values *bb_group_values*

Description

Resolve the grouping values of a statistical layer

Usage

```
bb_group_values(mapping, data, x = NULL)
```

Arguments

mapping	the effective aesthetic mapping
data	the layer data
x	optional x values; used when they are a factor or character

Details

The one grouping rule used by the statistical layers: an explicit ‘group‘ mapping wins, then a discrete ‘x‘, then a mapped ‘col‘, then a mapped ‘fill‘. Custom statistical layers can call this function so they group like the built-in ones.

Value

A vector of group values, or NULL when there is no grouping

Author(s)

Guangchuang Yu

bb_labs *bb_labs*

Description

change labels for bbplot

Usage

```
bb_labs(title = NULL, sub = NULL, xlab = NULL, ylab = NULL)

bb_title(title)

bb_sub(sub)

bb_xlab(xlab)

bb_ylab(ylab)
```

Arguments

title	title
sub	sub
xlab	xlab
ylab	ylab

Details

setting one or several of 'title', 'sub', 'xlab', and 'ylab'

Value

A modified bbplot object

Author(s)

Guangchuang Yu

Examples

```
library(plotbb)
p <- bbplot(mtcars, bb_aes(mpg, disp, col=factor(cyl))) + bb_point(pch=19)
p + bb_labs(title = "hello", sub = "just for demo",
            xlab="this is xlab", ylab = "this is ylab") +
    bb_title("hello world") # last one rules
```

bb_layer *bb_layer*

Description

build a custom layer

Usage

```
bb_layer(
  draw,
  mapping = NULL,
  data = NULL,
  inherit = TRUE,
  ...,
  name = "custom layer",
  limits = NULL,
  aesthetics = NULL,
  summary = FALSE,
  range = NULL
)
```

Arguments

draw	a function that draws the layer. Its first arguments are x and y (plus mapped aesthetics such as col); it receives the fixed aesthetics in ....
mapping	aesthetic mapping for this layer. Defaults to inheriting the plot mapping.
data	layer data. Defaults to the plot data.
inherit	whether to inherit the plot mapping, see bb_point(). A custom layer's data is inherited from bbplot() unless given.
...	fixed aesthetics and any other values passed to draw.
name	shown when the plot's layers are printed.
limits	optional range the drawing function needs, as list(x = c(low, high), y = c(low, high)). Both elements are optional. Use it for coordinates that are not mapped from the data, such as a reference line or an annotation drawn outside the data range.
aesthetics	optional mapped aesthetics the layer understands; include "fill" when a constant 'fill' is an interior colour.
summary	logical; TRUE when the layer computes its own coordinates and the plot-level x/y should not be added to the ranges.
range	optional function function(plot, data, mapping) returning list(x = ..., y = ...) with computed ranges for a statistical layer.

Details

‘bb_layer()‘ is the supported way to add your own drawing code to a ‘bbplot‘. It is an ordinary layer: it inherits the plot data and mapping, accepts its own ‘data‘ and ‘mapping‘, is drawn in ‘+‘ order, and takes part in the same input validation as the built-in layers.

The drawing function receives the resolved coordinates and, when the layer has mappings, the mapped aesthetics, plus whatever fixed aesthetics you pass through ‘...‘:

```
bb_layer(function(x, y, col, ...) { ... }, col = "tomato")
```

A custom layer takes part in the ranges of its mapped ‘x‘/‘y‘ values, like the built-in layers, and its ‘data‘ follows the ‘bb_facet_wrap()‘ panels. Coordinates that the drawing function computes itself (a smooth curve, a confidence band, a reference line) do not widen the axes on their own; pass them explicitly in ‘limits‘, or supply a ‘range‘ function for a statistical layer. Mapped aesthetics (‘col‘, ‘fill‘, ‘pch‘, ‘lty‘, ‘cex‘) use the shared scale and the automatic legend; a custom layer cannot define a new scale or legend type. If the drawing needs a different scale, use the public scale constructors or start from a canvas created by `as.bbplot()`.

Value

A layer object to be added to a `bbplot` with `+`.

See Also

`bb_point`, `bbplot_add`

Examples

```
library(plotbb)
p <- bbplot(mtcars, bb_aes(mpg, disp, col = factor(cyl)))
p +
  bb_layer(function(x, y, col, cex = 1, ...) {
    graphics::points(x = x, y = y, col = col, pch = 19, cex = cex)
  }, cex = 1.6)

## a reference line below the data needs the axis to make room for it
p +
  bb_point(pch = 19) +
  bb_layer(function(x, y, ...) {
    graphics::abline(h = min(y) - 50, col = "firebrick", lty = 2)
  }, limits = list(y = range(mtcars$disp) + c(-60, 0)))
```

`bb_legend` *bb_legend*

Description

Add legend for aesthetic mapping.

Usage

```
bb_legend(
  position = "topright",
  title = NULL,
  aesthetic = "col",
  pch = 19,
  bty = "n",
  outside = FALSE,
  orientation = c("vertical", "horizontal"),
  layer = NULL,
  style = c("auto", "bar"),
  title_width = NULL,
  width = 1.4,
  ...
)
```

Arguments

`position`	legend position passed to `graphics::legend`, or a numeric x position for an outside legend.
`title`	legend title.
`aesthetic`	currently supports `"col"`, `"fill"`, `"pch"`, `"lty"`, `"cex"`.
`pch`	point shape used in legend.
`bty`	box type passed to `graphics::legend`.
`outside`	logical, whether to place the legend outside the plotting area.
`orientation`	orientation of a continuous colour bar, `"vertical"` or `"horizontal"`.
`layer`	optional layer index or name: the legend then describes that layer's mapping. The colour scale is still the shared one, so this selects the mapping/title, not a separate domain.
`style`	`"auto"` or `"bar"`; a bar draws a discrete col/fill legend as adjacent colour blocks (a discrete gradient).
`title_width`	optional positive integer; wraps a long legend title.
`width`	relative width of an outside legend column (default 1.4).
`...`	additional parameters passed to `graphics::legend`.

Value

A modified bbplot object

Author(s)

Guangchuang Yu

bb_position *bb_position*

Description

jitter position for bb_point

Usage

```
bb_position(..., type = c("jitter", "identity"))
```

Arguments

`...`	parameters for the position, e.g. `width`, `height`, `seed`.
`type`	position type, one of 'identity' or 'jitter'.

Value

a position object that can be passed to `bb_point()`.

Author(s)

Guangchuang Yu

bb_save *bb_save*

Description

Save a plot to a file

Usage

```
bb_save(
  plot,
  filename,
  width = 7,
  height = 5,
  units = "in",
  res = 300,
  bg = "white",
  device = NULL,
  overwrite = FALSE,
  ...
)
```

Arguments

`plot`	a `bbplot` object
`filename`	output file
`width, height`	device size
`units`	one of `"in"`, `"cm"`, `"mm"`, `"px"`
`res`	resolution for raster devices
`bg`	background colour
`device`	optional device name (`"pdf"`, `"png"`, ...) or a device function such as `grDevices::pdf`
`overwrite`	whether an existing file may be replaced
`...`	additional arguments passed to the device

Details

‘bb_save()‘ draws into a temporary file next to the target and moves it into place only after the plot has been drawn. It never closes a device the caller had already open, and on failure it closes its own device, keeps the original error, removes the temporary file and leaves an existing target untouched (even when ‘overwrite = TRUE‘).

The device is chosen from the file extension (‘pdf‘, ‘png‘, ‘jpeg‘/‘jpg‘, ‘tiff‘/‘tif‘, ‘svg‘) unless ‘device‘ is given. Supplying a device that conflicts with a known extension is an error. Vector devices take their size in inches; raster devices take ‘width‘, ‘height‘, ‘units‘ and ‘res‘.

An existing file is never overwritten unless ‘overwrite = TRUE‘. A missing directory is an error.

Value

The file name, invisibly

See Also

`bb_build`

Examples

```
p <- bbplot(mtcars, bb_aes(mpg, disp)) + bb_point()
f <- tempfile(fileext = ".pdf")
bb_save(p, f)
unlink(f)
```

bb_scale_col_palette *bb_scale_col_palette*

Description

change col palette

Usage

```
bb_scale_col_palette(
  palette = NULL,
  limits = NULL,
  na.value = "grey50",
  drop = TRUE,
  breaks = NULL,
  labels = NULL,
  minor_breaks = NULL
)

bb_scale_col_manual(
  values,
  limits = NULL,
  na.value = "grey50",
  drop = TRUE,
  breaks = NULL,
  labels = NULL,
  minor_breaks = NULL
)

bb_scale_col_gradient(
  low = "#132B43",
  high = "#56B1F7",
  limits = NULL,
  na.value = "grey50",
  oob = c("squish", "censor"),
  breaks = NULL,
  labels = NULL,
  minor_breaks = NULL
)
```

Arguments

palette	color palette
limits	optional limits of the trained domain. Values outside the limits follow ‘oob‘; levels outside the limits are not shown.
na.value	colour used for missing values and for values not covered by a discrete scale.

drop	whether unobserved factor levels are dropped from the trained scale and the legend.
breaks	optional breaks for a continuous colour bar. Defaults to ‘pretty()‘ on the trained domain.
labels	optional labels for the breaks (continuous) or a named vector renaming discrete levels.
minor_breaks	optional positions for minor ticks on a continuous colour bar; ‘NULL‘ uses the midpoints between the major breaks.
values	colors for discrete scale.
low	low color for continuous scale.
high	high color for continuous scale.
oob	out-of-bounds behaviour for a continuous scale: ‘"squish"‘ (default) clamps to the limits, ‘"censor"‘ maps them to ‘na.value‘.

Value

A modified bbplot object

Author(s)

Guangchuang Yu

`bb_scale_fill_palette` *bb_scale_fill_palette*

Description

Fill scales

Usage

```
bb_scale_fill_palette(
  palette = NULL,
  limits = NULL,
  na.value = "grey50",
  drop = TRUE,
  breaks = NULL,
  labels = NULL,
  minor_breaks = NULL
)

bb_scale_fill_manual(
  values,
  limits = NULL,
  na.value = "grey50",
```

```
  drop = TRUE,
  breaks = NULL,
  labels = NULL,
  minor_breaks = NULL
)

bb_scale_fill_gradient(
  low = "#132B43",
  high = "#56B1F7",
  limits = NULL,
  na.value = "grey50",
  oob = c("squish", "censor"),
  breaks = NULL,
  labels = NULL,
  minor_breaks = NULL
)
```

Arguments

palette	color palette
limits	optional limits of the trained domain. Values outside the limits follow ‘oob‘; levels outside the limits are not shown.
na.value	colour used for missing values and for values not covered by a discrete scale.
drop	whether unobserved factor levels are dropped from the trained scale and the legend.
breaks	optional breaks for a continuous colour bar. Defaults to ‘pretty()‘ on the trained domain.
labels	optional labels for the breaks (continuous) or a named vector renaming discrete levels.
minor_breaks	optional positions for minor ticks on a continuous colour bar; ‘NULL‘ uses the midpoints between the major breaks.
values	colors for discrete scale.
low	low color for continuous scale.
high	high color for continuous scale.
oob	out-of-bounds behaviour for a continuous scale: ‘"squish"‘ (default) clamps to the limits, ‘"censor"‘ maps them to ‘na.value‘.

Details

‘fill‘ is a separate aesthetic from ‘col‘: use these constructors when a layer maps ‘fill‘ (‘bb_tile()‘, ‘bb_density()‘, ‘bb_bar()‘, ‘bb_boxplot()‘). They take the same arguments as the ‘col‘ scales.

bb_scale_pch_manual *aesthetic scales*

Description

aesthetic scales

Usage

```
bb_scale_pch_manual(values, limits = NULL, na.value = 1, drop = TRUE)

bb_scale_lty_manual(values, limits = NULL, na.value = 1, drop = TRUE)

bb_scale_cex_manual(values, limits = NULL, na.value = 1, drop = TRUE)

bb_scale_cex_continuous(
  range = c(0.8, 1.6),
  limits = NULL,
  na.value = 1,
  oob = c("squish", "censor")
)
```

Arguments

values	values used for mapping.
limits	optional limits of the trained domain.
na.value	value used for missing values.
drop	whether unobserved levels are dropped.
range	numeric length-2 range for continuous mapping.
oob	out-of-bounds behaviour, ‘"squish"‘ or ‘"censor"‘.

Value

A modified bbplot object

Author(s)

Guangchuang Yu

bb_scale_x_continuous *bb_scale_x_continuous*

Description

Positional scales

Usage

```
bb_scale_x_continuous(
  limits = NULL,
  breaks = NULL,
  labels = NULL,
  trans = c("identity", "log10")
)

bb_scale_y_continuous(
  limits = NULL,
  breaks = NULL,
  labels = NULL,
  trans = c("identity", "log10")
)

bb_scale_x_log10(limits = NULL, breaks = NULL, labels = NULL)

bb_scale_y_log10(limits = NULL, breaks = NULL, labels = NULL)

bb_scale_x_date(
  limits = NULL,
  breaks = NULL,
  labels = NULL,
  format = "%Y-%m-%d"
)

bb_scale_y_date(
  limits = NULL,
  breaks = NULL,
  labels = NULL,
  format = "%Y-%m-%d"
)
```

Arguments

limits	optional two limits for the axis.
breaks	optional break positions; ‘NULL‘ uses the base axis.
labels	optional labels for the breaks; requires ‘breaks‘.

`trans`	axis transform: ‘"identity"‘ or ‘"log10"‘.
`format`	date format for a date axis.

Details

These scales control the x and y axes: user limits, breaks, labels, a log10 transform (‘bb_scale_x_log10()‘, ‘bb_scale_y_log10()‘) and date axes (‘bb_scale_x_date()‘, ‘bb_scale_y_date()‘). They are separate from the aesthetic scales (‘col‘, ‘fill‘, ‘pch‘, ‘lty‘, ‘cex‘).

‘limits‘ sets the final axis range; values outside it are clipped by the device. A log scale requires positive limits. Date scales use ‘graphics::axis.Date()‘ and take Date limits/breaks.

Value

A modified bbplot object

Author(s)

Guangchuang Yu

`bb_theme` *bb_theme*

Description

bbplot theme

Usage

```
bb_theme(...)

bb_theme_expand(...)

bb_theme_grey(...)

bb_theme_deepblue(...)

bb_theme_bw(...)

bb_theme_minimal(...)
```

Arguments

`...`	parameters for graphics::par

Details

setting visual details of bbplot

Value

A modified bbplot object

Author(s)

Guangchuang Yu

---

bbplot *bbplot*

Description

bbplot

Usage

```
bbplot(data, mapping = bb_aes(), auto_legend = TRUE)
```

Arguments

`data`	data
`mapping`	variable mapping
`auto_legend`	if TRUE (the default), mapped aesthetics get outside legends automatically; explicit ‘bb_legend()‘ components take over the aesthetics they name.

Details

a grammar of graphics built on base graphics. The bbplot class contains data (input data), mapping (aesthetic mapping), layer (a list of plot layers), theme (theme setting) and labs (label setting, including title, subtitle, x and y labels).

‘+‘ adds components (layers, themes, labels, scales, legends, facets) to a single plot and draws them in the order they were added. It does not combine two complete plots: arrange several plots with `aplot::plot_list()` instead.

Every layer inherits the plot `data` and `mapping` and may override them: a local mapping overrides the global mapping aesthetic by aesthetic, and `inherit = FALSE` opts out of the global mapping entirely. Aesthetic constants such as `bb_point(col = "tomato")` apply to the whole layer and win over a mapped aesthetic.

Value

bbplot object

Author(s)

Guangchuang Yu

Examples

```
library(plotbb)
p <- bbplot(mtcars, bb_aes(mpg, disp, col=factor(cyl)))
p + bb_grid(col='grey50', lty='dashed') + bb_point(pch=19)
```

`bbplot_add` *bbplot_add*

Description

Add custom objects to bbplot

Usage

```
bbplot_add(object, plot)
```

Arguments

`object`	An object to add to the plot
`plot`	The bbplot object to add ‘object‘ to

Details

This generic is the extension point for new kinds of components. A method receives the object and the plot and must return the modified plot; it should describe the resulting plot instead of drawing, and can rely on the public `bb_layer()` (or the other public constructors) to add the drawing.

A method must not draw directly, change the device or rely on internal helpers such as `build_layer()`; those are implementation details and are not part of the stable contract. Packages register the method with `S3method(bbplot_add, <class>)` in their `NAMESPACE`. A component added this way takes part in the `+` order and in the graphics-state restoration like every other component, and it is re-applied to every facet panel. The layers a method adds are not added a second time when a panel is rebuilt: the owning component is re-run instead.

Value

A modified bbplot object

Author(s)

Guangchuang Yu

See Also

`bb_layer`

Examples

```
bb_note <- function(label) {
    structure(list(label = label), class = "bb_note")
}

bbplot_add.bb_note <- function(object, plot) {
    plot + bb_layer(function(x, y, ...) {
        graphics::text(mean(range(x)), mean(range(y)), object$label)
    })
}

bbplot(mtcars, bb_aes(mpg, disp)) +
    bb_point(pch = 19) +
    bb_note("a third-party component")
```

`set_bb_theme` *set_bb_theme*

Description

set bb_theme

Usage

```
set_bb_theme(theme)
```

Arguments

`theme`	bb_theme

Details

setting bb_theme for ordinary base plot command. It internally use par to set global graphic parameters. Users need to explictely call unset_bb_theme() to restore original setting.

Themes are single-level: calling ‘set_bb_theme()‘ while a theme is already active replaces it without losing the original graphics parameters, so ‘unset_bb_theme()‘ always restores the state from before the *first* call. Prefer ‘with_bb_theme()‘ when the theme should be scoped to a block of code.

Value

setting selected theme as default (has side effect and will affect other base plot)

Author(s)

Guangchuang Yu

Examples

```
library(plotbb)
set_bb_theme(bb_theme_deepblue)
bbplot(mtcars, bb_aes(mpg, disp, col=factor(cyl))) + bb_point(pch=19)
unset_bb_theme()
```

---

unset_bb_theme *unset_bb_theme*

Description

unset bb_theme

Usage

```
unset_bb_theme()
```

Details

remove all the themes by set_bb_theme. The saved pre-theme graphics parameters are discarded afterwards, so the function is safe to call repeatedly.

Value

unset theme (i.e., restore par setting)

Author(s)

Guangchuang Yu

---

with_bb_theme *with_bb_theme*

Description

Run code with a theme applied to the current device

Usage

```
with_bb_theme(theme, code)
```

Arguments

theme	a ‘bb_theme‘ object or a function returning one.
code	an expression to evaluate.

Details

‘with_bb_theme()‘ applies ‘theme‘ with ‘par()‘ for the duration of ‘code‘ and restores the previous graphics parameters afterwards, including when ‘code‘ signals an error. Nothing is stored in options, so it can be nested and does not interact with ‘set_bb_theme()‘.

Value

the value of ‘code‘, invisibly.

Author(s)

Guangchuang Yu

Examples

```
library(plotbb)
with_bb_theme(bb_theme_deepblue(), {
  bbplot(mtcars, bb_aes(mpg, disp, col=factor(cyl))) + bb_point(pch=19)
})
```

Index

as.bbplot, 2

bb_aes, 3
bb_bar, 4
bb_boxplot *(bb_bar)*, 4
bb_build, 9, *19*
bb_coord_flip, 10
bb_density *(bb_bar)*, 4
bb_errorbar *(bb_bar)*, 4
bb_facet_grid, 10
bb_facet_wrap, 11
bb_grid, 12
bb_group_values, 13
bb_hist *(bb_bar)*, 4
bb_hline *(bb_bar)*, 4
bb_labs, 14
bb_layer, 15, *27*
bb_legend, 16
bb_line *(bb_bar)*, 4
bb_lm *(bb_bar)*, 4
bb_point, *16*
bb_point *(bb_bar)*, 4
bb_position, 18
bb_ribbon *(bb_bar)*, 4
bb_save, *9*, 18
bb_scale_cex_continuous *(bb_scale_pch_manual)*, 23
bb_scale_cex_manual *(bb_scale_pch_manual)*, 23
bb_scale_col_gradient *(bb_scale_col_palette)*, 20
bb_scale_col_manual *(bb_scale_col_palette)*, 20
bb_scale_col_palette, 20
bb_scale_fill_gradient *(bb_scale_fill_palette)*, 21
bb_scale_fill_manual *(bb_scale_fill_palette)*, 21
bb_scale_fill_palette, 21
bb_scale_lty_manual *(bb_scale_pch_manual)*, 23
bb_scale_pch_manual, 23
bb_scale_x_continuous, 24
bb_scale_x_date *(bb_scale_x_continuous)*, 24
bb_scale_x_log10 *(bb_scale_x_continuous)*, 24
bb_scale_y_continuous *(bb_scale_x_continuous)*, 24
bb_scale_y_date *(bb_scale_x_continuous)*, 24
bb_scale_y_log10 *(bb_scale_x_continuous)*, 24
bb_segment *(bb_bar)*, 4
bb_signif *(bb_bar)*, 4
bb_smooth *(bb_bar)*, 4
bb_sub *(bb_labs)*, 14
bb_text *(bb_bar)*, 4
bb_theme, 25
bb_theme_bw *(bb_theme)*, 25
bb_theme_deepblue *(bb_theme)*, 25
bb_theme_expand *(bb_theme)*, 25
bb_theme_grey *(bb_theme)*, 25
bb_theme_minimal *(bb_theme)*, 25
bb_tile *(bb_bar)*, 4
bb_title *(bb_labs)*, 14
bb_violin *(bb_bar)*, 4
bb_vline *(bb_bar)*, 4
bb_xlab *(bb_labs)*, 14
bb_ylab *(bb_labs)*, 14
bbplot, 26
bbplot_add, *16*, 27

set_bb_theme, 28

unset_bb_theme, 29

with_bb_theme, 29