Package: ggindex (via r-universe)

November 5, 2026

Title Numbered Focus Points for 'ggplot2' Scatter Plots

Version 0.1.0

Description A 'ggplot2' geom that highlights selected points of a scatter plot with numbered markers and lists their descriptions in a legend keyed by the same numbers, as on an annotated map.

License MIT + file LICENSE

URL https://github.com/discindo/ggindex, https://discindo.github.io/ggindex/

BugReports https://github.com/discindo/ggindex/issues

Depends R (>= 4.1)

Imports ggplot2 (>= 4.0.0), grid, scales

Suggests quarto, sf, testthat (>= 3.0.0), vdiffr

VignetteBuilder quarto

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.1.0

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

Date/Publication 2026-11-05 08:07:01 UTC

RemoteUrl https://github.com/discindo/ggindex

RemoteRef HEAD

RemoteSha 909ce58013b5dbea36c12668c859668b96094471

Contents

draw_key_focus 2
geom_focus 2
scale_info_discrete 6

Index **8**

draw_key_focus *Legend key for numbered markers*

Description

Draws the marker of `geom_focus()` with the label of its legend entry inside. The label is looked up from the `info` value of the key in `params$info_labels`; without a match only the marker is drawn.

Usage

```
draw_key_focus(data, params, size)
```

Arguments

`data`	A single row data frame with the scaled aesthetics.
`params`	A list of layer parameters.
`size`	Width and height of the key in mm (unused).

Value

A grid graphical object (grob) drawn as the key of a legend entry: a `gTree` with the marker and its label, or a `points` grob with the marker only when no label is found.

Examples

```
key <- data.frame(
  colour = "black", fill = "white", shape = 21, size = 6, stroke = 0.5,
  alpha = NA, family = "", fontface = 1, info = "Mazda RX4"
)
params <- list(info_labels = c("Mazda RX4" = "1"), text.size = 3.87)
grob <- draw_key_focus(key, params)
grid::grid.newpage()
grid::grid.draw(grob)
```

geom_focus *Numbered focus points*

Description

Draws a marker carrying a text `label` (typically a rank or index) at each observation. Without a `label` mapping, the `info` entries are numbered 1, 2, 3, ... in the order they appear in the data. Map `focus` to a logical variable to draw markers only for the observations where it is `TRUE`; add `ggplot2::geom_point()` to show the other observations as background. When `info` is mapped, a legend is added that pairs each numbered marker with its description.

Map `label`, `focus` and `info` in `geom_focus()` rather than in `ggplot2::ggplot()` when the plot has other layers. Layers inherit the plot mapping, so their observations would also appear in the `info` legend.

The markers use the standard `colour`, `fill`, `shape`, `size`, `stroke` and `alpha` aesthetics, so the usual ggplot2 scales apply to them. Their defaults come from the theme: a `paper` filled circle with an `ink` border and label. A marker grows beyond `size` when its label needs more room.

Markers are drawn at the data positions and don't avoid each other. To separate markers that overlap, map `nudge_x` or `nudge_y` and set `position = "nudge"`.

Usage

```
geom_focus(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  text.colour = NULL,
  text.color = NULL,
  text.size = 3.87,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

geom_sf_focus(
  mapping = ggplot2::aes(),
  data = NULL,
  stat = "sf_coordinates",
  position = "identity",
  ...,
  text.colour = NULL,
  text.color = NULL,
  text.size = 3.87,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  fun.geometry = NULL
)
```

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 `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through `....` Unknown arguments that are not part of the 4 categories below are ignored.

- Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, `colour = "red"` or `linewidth = 3`. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the `params`. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.
- When constructing a layer using a `stat_*()` function, the `...` argument can be used to pass on parameters to the `geom` part of the layer. An example of this is `stat_density(geom = "area", outline.type = "both")`. The geom's documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a `geom_*()` function, the `...` argument can be used to pass on parameters to the `stat` part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept.

- The key_glyph argument of layer() may also be passed on through .... This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

text.colour, text.color
: Colour of the labels. NULL uses the colour aesthetic, the marker border colour.

text.size
: Size of the labels in mm.

na.rm
: If FALSE, the 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().

fun.geometry
: A function that takes a sfc object and returns a sfc_POINT with the same length as the input. If NULL, function(x) sf::st_point_on_surface(sf::st_zm(x)) will be used. Note that the function may warn about the incorrectness of the result if the data is not projected, but you can ignore this except when you really care about the exact locations.

Value

A ggplot2 layer (a ggproto object of class LayerInstance, or LayerSf for geom_sf_focus()). Add it to a plot with + to draw the numbered markers.

Aesthetics

Required: x, y, and label or info. Optional: label (numbers the info entries when not mapped), focus (logical; when not mapped, every observation gets a marker), info, colour, fill, shape, size, stroke, alpha, family, fontface.

geom_sf_focus() places the markers on simple features from the 'sf' package, at the coordinates computed by ggplot2::stat_sf_coordinates(): the point itself, or a point on the surface of a line or polygon.

Examples

```
library(ggplot2)

data <- data.frame(
  x = 1:10,
  y = c(3, 5, 2, 8, 6, 4, 7, 1, 9, 5),
  rank = 1:10,
  focus = rep(c(TRUE, FALSE), each = 5),
  info = sprintf("Info for %s", 1:10)
)
```

```
ggplot(data, aes(x, y, label = rank, info = info)) +
  geom_focus() +
  theme_minimal()

# Number a subset and keep the rest as background points. Map label,
# focus and info in geom_focus(): other layers would inherit them from
# ggplot(), and their data would train the info legend.
ggplot(data, aes(x, y)) +
  geom_point(colour = "grey60") +
  geom_focus(aes(label = rank, focus = focus, info = info)) +
  theme_minimal()


# Number counties on a map
nc <- sf::st_read(system.file("shape/nc.shp", package = "sf"), quiet = TRUE)
nc <- sf::st_transform(nc, 32119)
nc_large <- nc[nc$AREA > 0.2, ]

ggplot(nc) +
  geom_sf(fill = "grey90", colour = "white") +
  geom_sf_focus(aes(info = NAME), data = nc_large) +
  theme_void()
```

`scale_info_discrete` *Scale for the* `info` *aesthetic of* `geom_focus()`

Description

Builds the legend that lists the description of each numbered marker. It is added automatically when `info` is mapped; call it directly to change the legend title or other legend settings.

Usage

```
scale_info_discrete(name = ggplot2::waiver(), ..., guide = "legend")
```

Arguments

`name`	The name of the scale. Used as the axis or legend title. If `waiver()`, the default, the name of the scale is taken from the first mapping used for that aesthetic. If `NULL`, the legend title will be omitted.
`...`	Other arguments passed on to `ggplot2::discrete_scale()`.
`guide`	A function used to create a guide or its name. See `guides()` for more information.

Value

A ggplot2 scale object (class `ScaleDiscreteIdentity`, a `ggproto` object). Add it to a plot with `+` to control the legend that maps each `info` value to its numbered marker.

Examples

```
cars <- mtcars
cars$model <- rownames(cars)
library(ggplot2)
ggplot(cars, aes(wt, mpg)) +
  geom_point(colour = "grey60") +
  geom_focus(aes(label = rank(-mpg), focus = mpg > 30, info = model)) +
  scale_info_discrete(name = "Most efficient")
```

Index

aes(), *3*
annotation_borders(), *5*

draw_key_focus, 2

fortify(), *4*

geom_focus, 2
geom_focus(), *2*, *6*
geom_sf_focus *(*geom_focus*)*, 2
GeomFocus *(*geom_focus*)*, 2
ggplot(), *3*
ggplot2::discrete_scale(), *6*
ggplot2::geom_point(), *2*
ggplot2::ggplot(), *3*
ggplot2::stat_sf_coordinates(), *5*
guides(), *6*

key glyphs, *5*

layer position, *4*
layer stat, *4*
layer(), *4*, *5*

scale_info_discrete, 6