Data visualization with ggplot2 :: CHEAT SHEET

Basics

ggplot2 is based on the grammar of graphics, the idea that you can build every graph from the same components: a data set, a coordinate system, and geoms—visual marks that represent data points.

data + geom + coordinate = plot

To display values, map variables in the data to visual properties of the geom (aesthetics) like size, color, and x and y locations.

data + geom + coordinate = plot

Geoms

Use a geom function to represent data points, use the geom’s aesthetic properties to represent variables. Each function returns a layer.

GRAPHICAL PRIMITIVES

a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))

a + geom_blank() and a + expand_limits() Ensure limits include values across all plots.
b + geom_curve(aes(yend = lat = 1), xend = long + 1, curve = 1 - x, yend, alpha, angle, color, curve, lineend, size)
a + geom_path(lineend = “butt”, linejoin = “round”, linemitre = 1) x, y, alpha, color, group, linetype, size

ONE VARIABLE

x, y, alpha, color, linetype, size

GENERAL PURPOSE

i + geom_point() x, y, alpha, color, fill, shape, size, stroke

TWO VARIABLES

both continuous

f + geom_col() x, y, alpha, color, fill, group, linetype, size

e + geom_smooth(method = lm) x, y, alpha, color, fill, group, linetype, size, weight

continuous bivariate distribution

h + geom_bin2d(binwidth = c(0.25, 500)) x, y, alpha, color, fill, linetype, size, weight

GEOMETRIC PRIMITIVES

STATS

a + geom_histogram(binwidth = 5) x, y, alpha, color, fill, lineend, size

TWO VARIABLES, one continuous

f + geom_point(aes(size = fit)) x, y, alpha, color, fill, shape, size, stroke

continuous function

i + geom_area(x, y, alpha, color, fill, linetype, size)

LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size

LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size

ONE VARIABLE

continuous

x, y, alpha, color, linetype, size

TWO VARIABLES

both continuous

x, y, alpha, color, fill, linetype, size

discrete

d <- ggplot(mpg, aes(f))
x, alpha, color, fill, linetype, size

THREE VARIABLES

seals$s2 <- with(seals, sqrt(delta_long^2 + delta_lat^2)); l + ggplot(seals, aes(long, lat))

OBSERVATIONAL

alt = "" color = "black"

RStudio® is a trademark of RStudio, PBC  •  CC BY SA  RStudio  •  info@rstudio.com  •  844-448-1212  •  rstudio.com  •  Learn more at ggplot2.tidyverse.org  •  ggplot2 3.3.5  •  Updated: 2021-08
Stats

A stat builds new variables to plot (e.g., count, prop).

data ~ stat + geom + coord + theme + theme

Visualize a stat by changing the default stat of a geom function, `stat_bar(stat="count")` or by using a stat function, `stat_count geom="bar"`, which calls a default geom to make a layer (equivalent to a geom function). Use `..x..` syntax to map stat variables to aesthetics.

Example:
```r
p + geom_bar(aes(x = ..x..))
```

Scales

Scales map data values to the visual values of an aesthetic. To change a mapping, add a new scale.

Example:
```r
p <- ggplot(mapping = aes(x = ..x..))
```

Coordinate Systems

Coordinate systems are prepackaged, but can also be created with `scale_*_identity()` or `scale_*_manual()`. They control the position of things on the plot, and can be used to modify the default position of the axes or to change the scale of the data used. `ggplot()` positions the coordinate system automatically.

Example:
```r
p <- ggplot(mapping = aes(x = ..x..)) + coord_cartesian(xlim = c(0, 100))
```

Position Adjustments

Position adjustments determine how to arrange geoms that would otherwise occupy the same space.

Example:
```r
p + geom_bar(position = "fill")
```

Faceting

Facets divide a plot into subplots based on the values of one or more discrete variables.

Example:
```r
p + facet_wrap(~ ..x..)
```

Themes

Themes customize aspects of the theme such as axis, legend, panel, and facet properties.

Example:
```r
p + theme(legend.position = "bottom")
```

Zooming

Without clipping (preferred):
```r
p + coord_cartesian(xlim = c(0, 100), ylim = c(0, 100))
```

With clipping (removes unseen data points):
```r
p + coord_cartesian(xlim = c(0, 100), ylim = c(0, 100))
```