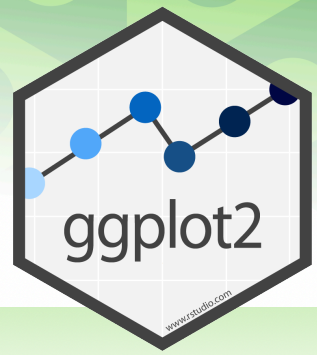


# Data visualization with ggplot2 : : CHEAT SHEET

Source: <https://github.com/rstudio/cheatsheets/blob/main/data-visualization-2.1.pdf>

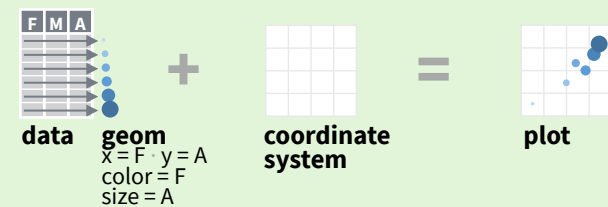


## Basics

**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.



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



Complete the template below to build a graph.

```
ggplot (data = <DATA>) +  
  <GEOM_FUNCTION> (mapping = aes (<MAPPINGS>),  
    stat = <STAT>, position = <POSITION>) +  
  <COORDINATE_FUNCTION> +  
  <FACET_FUNCTION> +  
  <SCALE_FUNCTION> +  
  <THEME_FUNCTION>
```

**required**

**Not required, sensible defaults supplied**

**ggplot**(data = mpg, aes(x = cty, y = hwy)) Begins a plot that you finish by adding layers to. Add one geom function per layer.

**last\_plot()** Returns the last plot.

**ggsave**("plot.png", width = 5, height = 5) Saves last plot as 5' x 5' file named "plot.png" in working directory. Matches file type to file extension.

## Aes Common aesthetic values.

**color** and **fill** - string ("red", "#RRGGBB")

**linetype** - integer or string (0 = "blank", 1 = "solid", 2 = "dashed", 3 = "dotted", 4 = "dotdash", 5 = "longdash", 6 = "twodash")

**lineend** - string ("round", "butt", or "square")

**linejoin** - string ("round", "mitre", or "bevel")

**size** - integer (line width in mm)

**shape** - integer/shape name or a single character ("a")

0 1 2 3 4 5 6 7 8 9 10 11 12  
□ ○ △ + × ◇ ▼ ☆ \* ◆ ⊕ ⊗ ⊞ ⊠  
13 14 15 16 17 18 19 20 21 22 23 24 25  
⊞ ⊠ ⊞ ⊠ ⊞ ⊠ ⊞ ⊠ ⊞ ⊠ ⊞ ⊠ ⊞ ⊠



## 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, curvature = 1) - x, xend, y, yend, alpha, angle, color, curvature, linetype, size)

**a + geom\_path**(lineend = "butt", linejoin = "round", linemitre = 1) - x, y, alpha, color, group, linetype, size

**a + geom\_polygon**(aes(alpha = 50)) - x, y, alpha, color, fill, group, subgroup, linetype, size

**b + geom\_rect**(aes(xmin = long, ymin = lat, xmax = long + 1, ymax = lat + 1)) - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

**a + geom\_ribbon**(aes(ymin = unemploy - 900, ymax = unemploy + 900)) - x, ymax, ymin, alpha, color, fill, group, linetype, size

### LINE SEGMENTS

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

**b + geom\_abline**(aes(intercept = 0, slope = 1))  
**b + geom\_hline**(aes(yintercept = lat))  
**b + geom\_vline**(aes(xintercept = long))

**b + geom\_segment**(aes(yend = lat + 1, xend = long + 1))  
**b + geom\_spoke**(aes(angle = 1:1155, radius = 1))

### ONE VARIABLE continuous

```
c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)
```

**c + geom\_area**(stat = "bin")  
x, y, alpha, color, fill, linetype, size

**c + geom\_density**(kernel = "gaussian")  
x, y, alpha, color, fill, group, linetype, size, weight

**c + geom\_dotplot()**  
x, y, alpha, color, fill

**c + geom\_freqpoly()**  
x, y, alpha, color, group, linetype, size

**c + geom\_histogram**(binwidth = 5)  
x, y, alpha, color, fill, linetype, size, weight

**c2 + geom\_qq**(aes(sample = hwy))  
x, y, alpha, color, fill, linetype, size, weight

### discrete

```
d <- ggplot(mpg, aes(fl))
```

**d + geom\_bar()**  
x, alpha, color, fill, linetype, size, weight

### TWO VARIABLES

#### both continuous

```
e <- ggplot(mpg, aes(cty, hwy))
```

**e + geom\_label**(aes(label = cty), nudge\_x = 1, nudge\_y = 1) - x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

**e + geom\_point()**  
x, y, alpha, color, fill, shape, size, stroke

**e + geom\_quantile()**  
x, y, alpha, color, group, linetype, size, weight

**e + geom\_rug**(sides = "bl")  
x, y, alpha, color, linetype, size

**e + geom\_smooth**(method = lm)  
x, y, alpha, color, fill, group, linetype, size, weight

**e + geom\_text**(aes(label = cty), nudge\_x = 1, nudge\_y = 1) - x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

#### one discrete, one continuous

```
f <- ggplot(mpg, aes(class, hwy))
```

**f + geom\_col()**  
x, y, alpha, color, fill, group, linetype, size

**f + geom\_boxplot()**  
x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight

**f + geom\_dotplot**(binaxis = "y", stackdir = "center")  
x, y, alpha, color, fill, group

**f + geom\_violin**(scale = "area")  
x, y, alpha, color, fill, group, linetype, size, weight

#### both discrete

```
g <- ggplot(diamonds, aes(cut, color))
```

**g + geom\_count()**  
x, y, alpha, color, fill, shape, size, stroke

**e + geom\_jitter**(height = 2, width = 2)  
x, y, alpha, color, fill, shape, size

### THREE VARIABLES

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

**l + geom\_contour**(aes(z = z))  
x, y, z, alpha, color, group, linetype, size, weight

**l + geom\_contour\_filled**(aes(fill = z))  
x, y, alpha, color, fill, group, linetype, size, subgroup

#### continuous bivariate distribution

```
h <- ggplot(diamonds, aes(carat, price))
```

**h + geom\_bin2d**(binwidth = c(0.25, 500))  
x, y, alpha, color, fill, linetype, size, weight

**h + geom\_density\_2d()**  
x, y, alpha, color, group, linetype, size

**h + geom\_hex()**  
x, y, alpha, color, fill, size

#### continuous function

```
i <- ggplot(economics, aes(date, unemploy))
```

**i + geom\_area()**  
x, y, alpha, color, fill, linetype, size

**i + geom\_line()**  
x, y, alpha, color, group, linetype, size

**i + geom\_step**(direction = "hv")  
x, y, alpha, color, group, linetype, size

#### visualizing error

```
df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)  
j <- ggplot(df, aes(grp, fit, ymin = fit - se, ymax = fit + se))
```

**j + geom\_crossbar**(fatten = 2) - x, y, ymax, ymin, alpha, color, fill, group, linetype, size

**j + geom\_errorbar()** - x, ymax, ymin, alpha, color, group, linetype, size, width  
Also **geom\_errorbarh()**.

**j + geom\_linerange()**  
x, ymin, ymax, alpha, color, group, linetype, size

**j + geom\_pointrange()** - x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

#### maps

```
data <- data.frame(murder = USArrests$Murder,  
  state = tolower(rownames(USArrests)))
```

```
map <- map_data("state")  
k <- ggplot(data, aes(fill = murder))
```

**k + geom\_map**(aes(map\_id = state), map = map) + **expand\_limits**(x = map\$long, y = map\$lat)  
map\_id, alpha, color, fill, linetype, size

**l + geom\_raster**(aes(fill = z), hjust = 0.5, vjust = 0.5, interpolate = FALSE)  
x, y, alpha, fill

**l + geom\_tile**(aes(fill = z))  
x, y, alpha, color, fill, linetype, size, width



