R you ready?
IntRo to RStudio and R Markdown
for open data and reproducibility

Unit 7:

Let's get plotting!

Mason A. Wirtz

@WirtzMason


GGPLOT ALL THE THINGS
memegenerator.ne

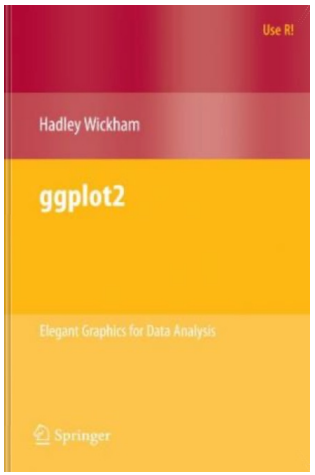
ggplot2: What is it?

ggplot2 is a package from the *tidyverse* and is one of the most-used plotting packages in R

ggplot2 plotting system →
https://bookdown.org/rdpeng/exdata/the-ggplot2-plotting-system-part-1.html#ggplot2-hello-world

ggplot2: Elegant Graphics for Data Analysis →
https://1lib.at/book/704124/daea5e

ggplot2 cheatsheet →
https://www.rstudio.com/resources/cheatsheets/



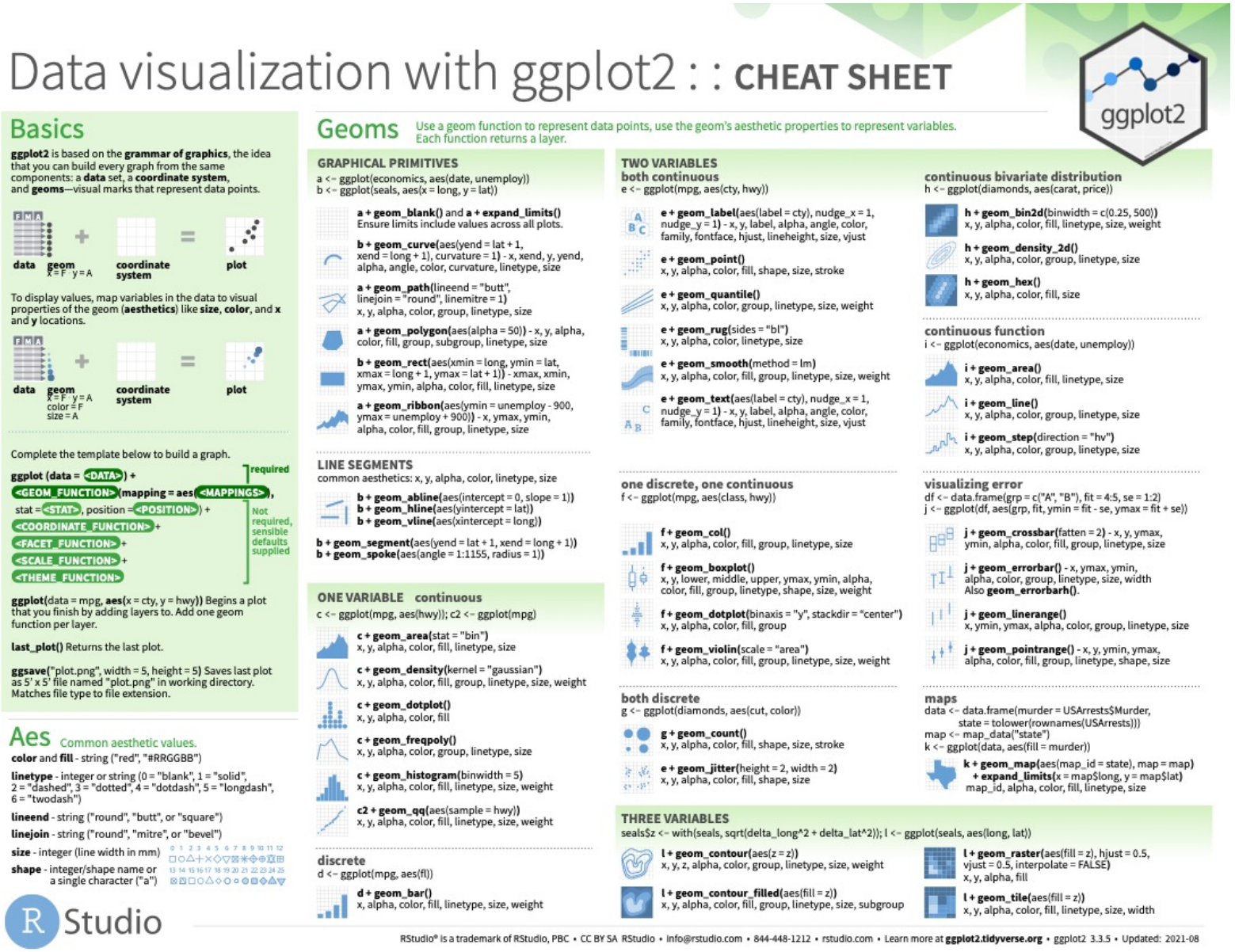
ggplot2: Why?

- Set of independent components that can be composed in different ways
- Not limited to pre-specified graphics
- Plots can be built iteratively
- i.e. plots are built using a layering principle
- beautiful faceting or multipanel plots
- can easily change or update plots

ggplot2: Terminology

- **Aesthetic mappings (aes())** → describe how variables in the data are mapped to aesthetic attributes
- **Geometric objects (geom)** → represents what we actually see on the plot
- **Statistical transformations (stats)** → summarise data in many useful ways
- **Scales** → map values in the data space to values in an aesthetic space
- **Faceting** → break up the data into subsets

ggplot2: cheat sheet



Data visualization with ggplot2 : : CHEAT SHEET

ggplot2

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 (x = F, y = A) | coordinate system | 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 (x = F, y = A, color = F, size = A) | coordinate system | plot

Complete the template below to build a graph.

```
ggplot (data = <DATA>) +                                  required
<GEOM_FUNCTION>(mapping = aes( <MAPPINGS>),
 stat = <STAT>, position = <POSITION>) +                  Not required, sensible defaults supplied
<COORDINATE_FUNCTION> +
<FACET_FUNCTION> +
<SCALE_FUNCTION> +
<THEME_FUNCTION>
```

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

RStudio

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

seals$z <- 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

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

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

ggplot2: Introduction

The workhorse function:

- **ggplot()**

Takes the arguments:

- **data = *data frame***
- **mapping = aes()**

ONE variable:

```
data frame %>%
    ggplot(aes(x = variable_x_axis))
```

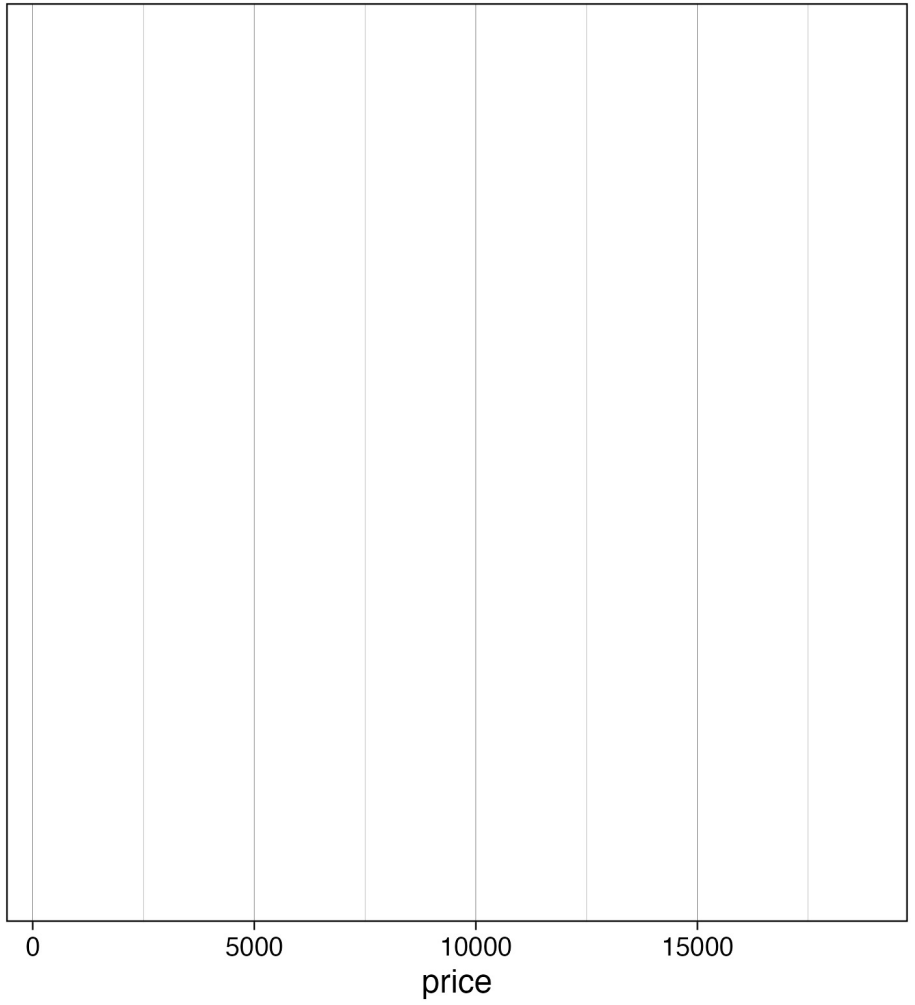
ggplot2: Introduction

Data frame

```
diamonds %>%
  ggplot(aes(x = price))
```

Add aesthetics

We defined our x-axis,
but we have not PLOTTED
anything!



So how do we plot?

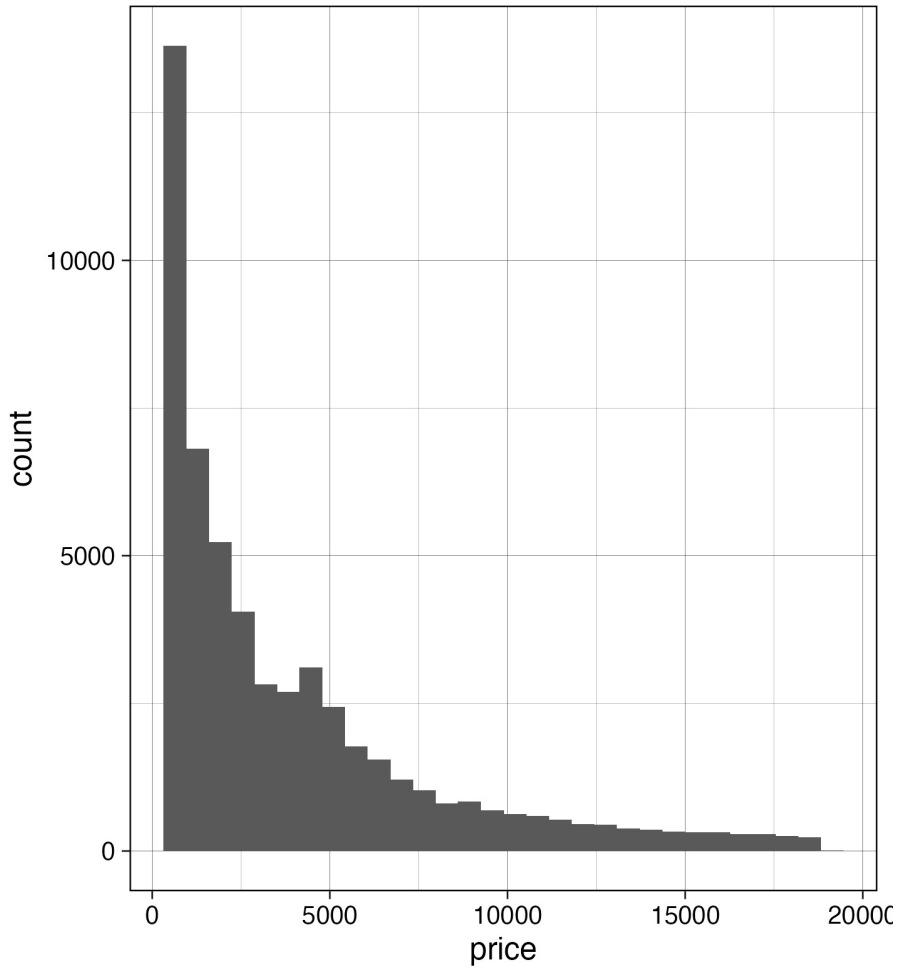
ggplot2: Plots of one variable

One CONTINUOUS variable:

- **Histogram:** *geom_histogram()*
- **Density plot:** *geom_density()*

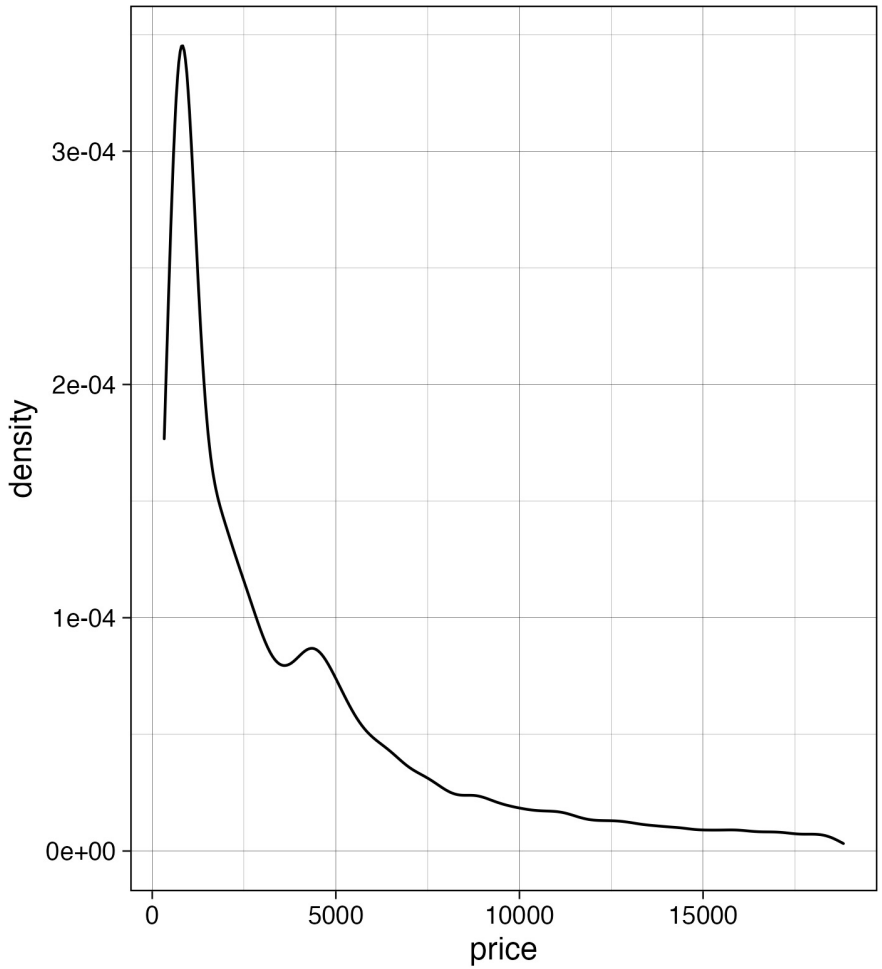
ggplot2: geom_histogram()

```
diamonds %>%
  ggplot(aes(x = price)) +
  geom_histogram()
```



ggplot2: geom_density()

```
diamonds %>%
  ggplot(aes(x = price)) +
  geom_density()
```



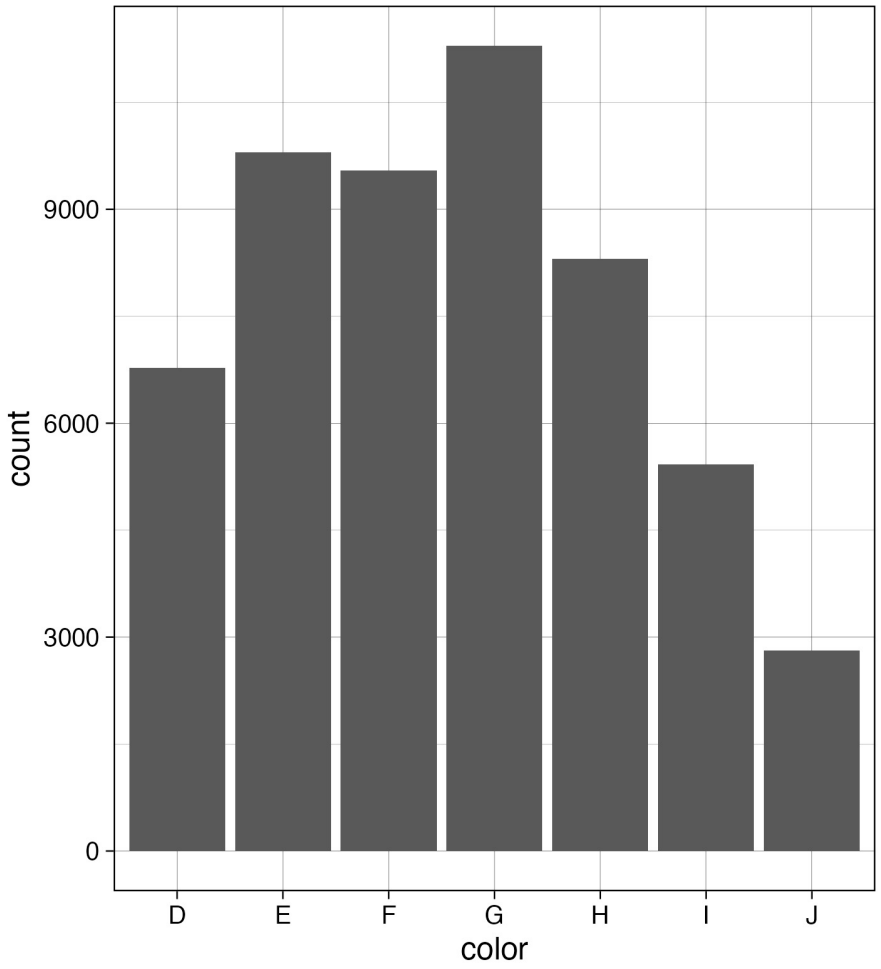
ggplot2: Plots of one variable

One CATEGORICAL variable:

- **Barplot:** *geom_bar()*

ggplot2: geom_bar()

```
diamonds %>%
  ggplot(aes(x = color)) +
  geom_bar()
```



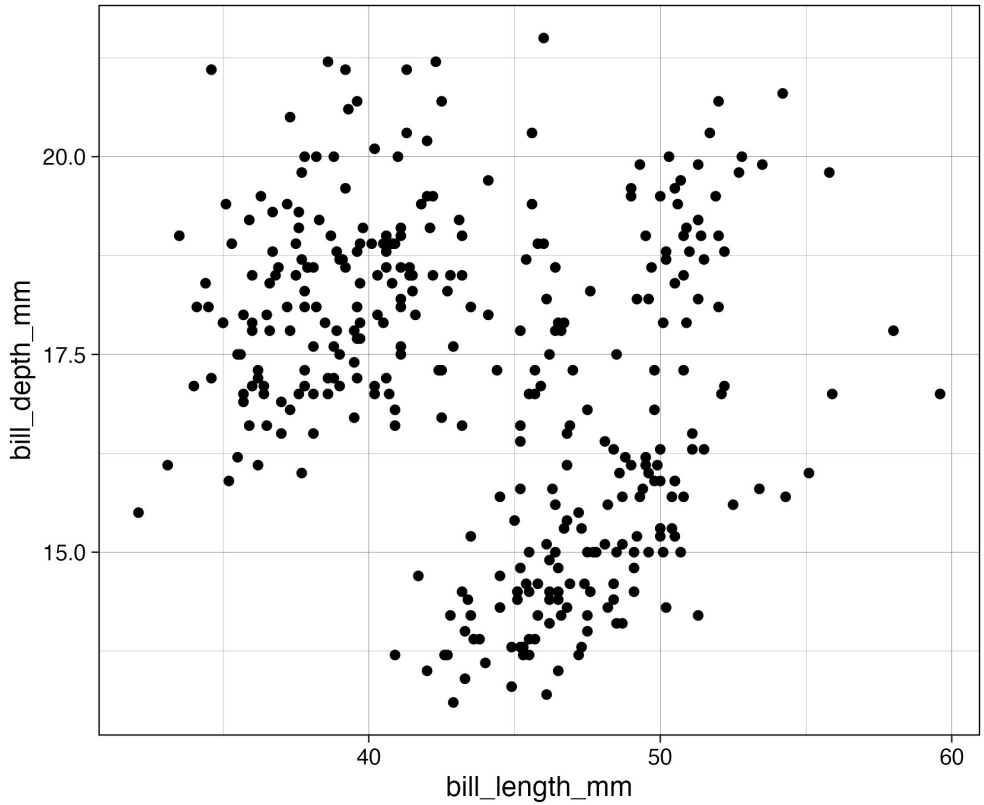
ggplot2: Plots of two variables

Two CONTINUOUS variables:

- **Scatter plot:** *geom_point()*
- **Scatter plot:** *geom_jitter()* → jittered (scattered) data points
- **Smoother:** *geom_smooth()* → linear and nonlinear lines

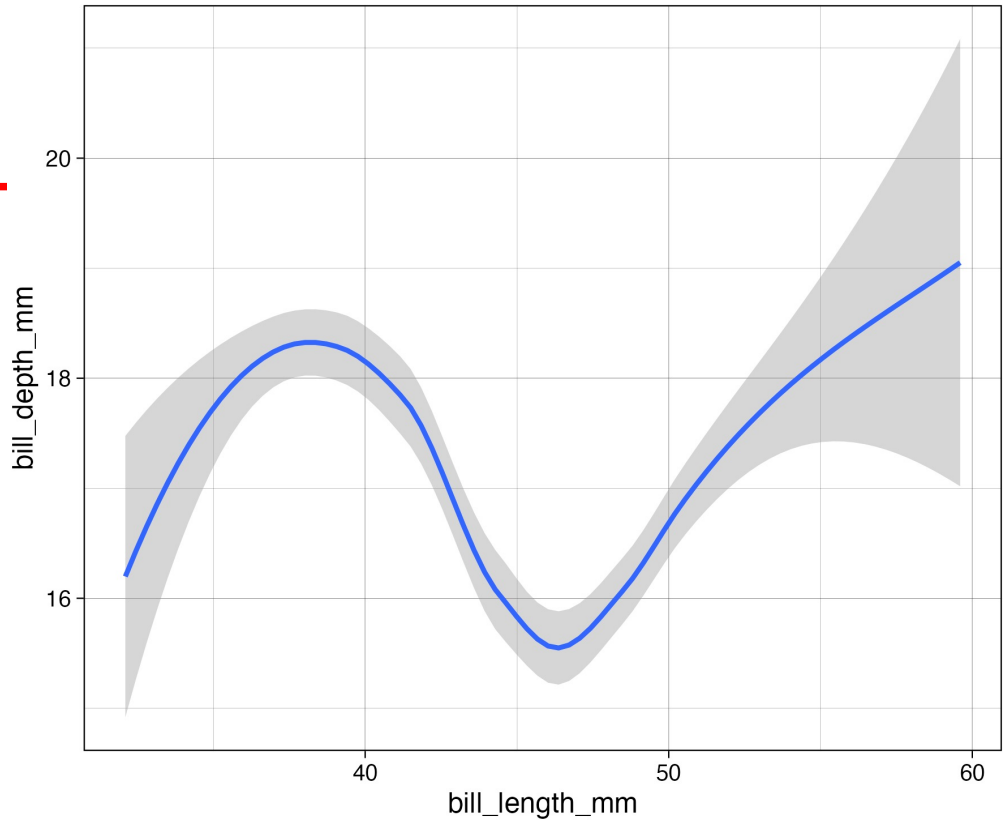
ggplot2: geom_point()

```
penguins %>%
  ggplot(aes(x = bill_length_mm,
             y = bill_depth_mm)) +
  geom_point()
```



ggplot2: geom_smooth()

```
penguins %>%
  ggplot(aes(x = bill_length_mm,
             y = bill_depth_mm)) +
  geom_smooth()
```



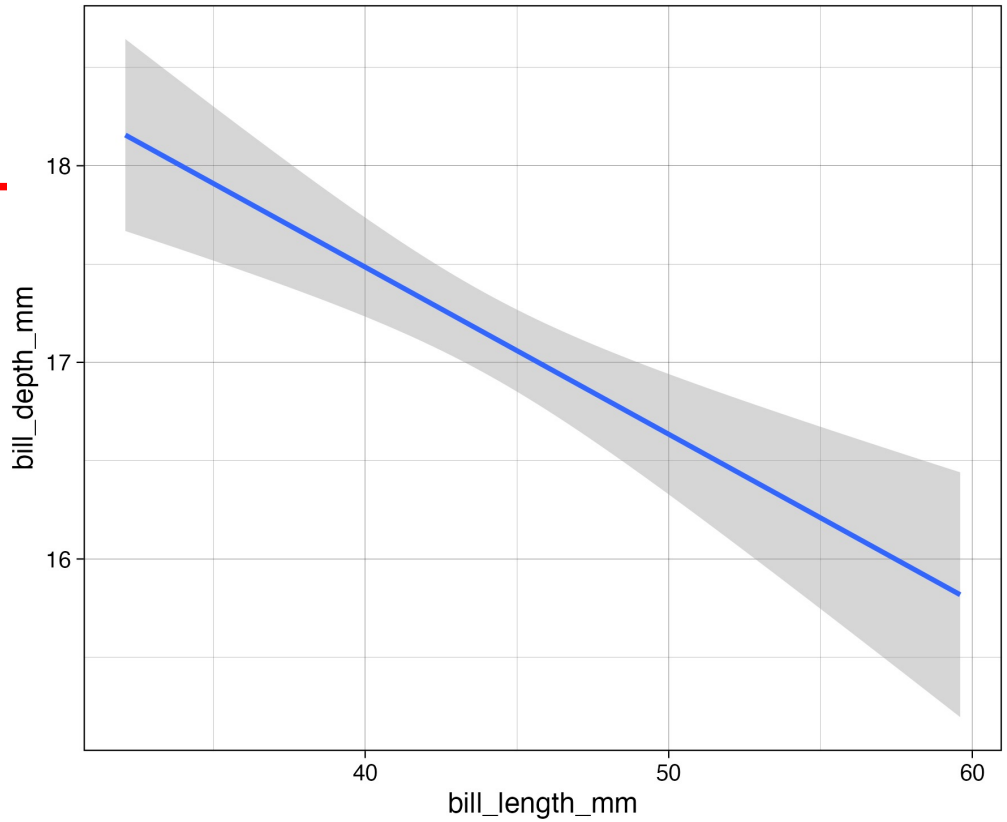
Default method:

Generalized additive model (GAM) with cubic spline

y ~ s(x, bs = "cs")

ggplot2: geom_smooth()

```
penguins %>%
  ggplot(aes(x = bill_length_mm,
             y = bill_depth_mm)) +
  geom_smooth(method = "lm")
```



Change method:

method = "lm"

→ Linear model

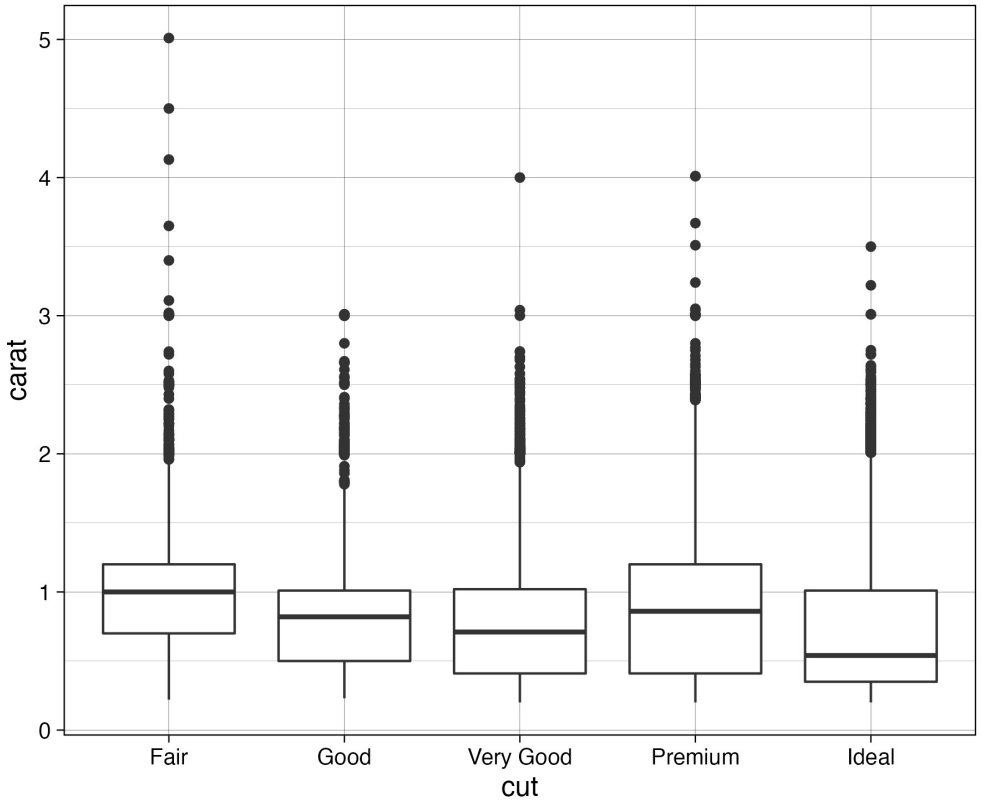
ggplot2: Plots of two variables

CATEGORICAL x and CONTINUOUS y:

- **Boxplot:** *geom_boxplot()*
- **Barplot:** *geom_bar(stat = "identity")*
- **Violin plot:** *geom_violin()*

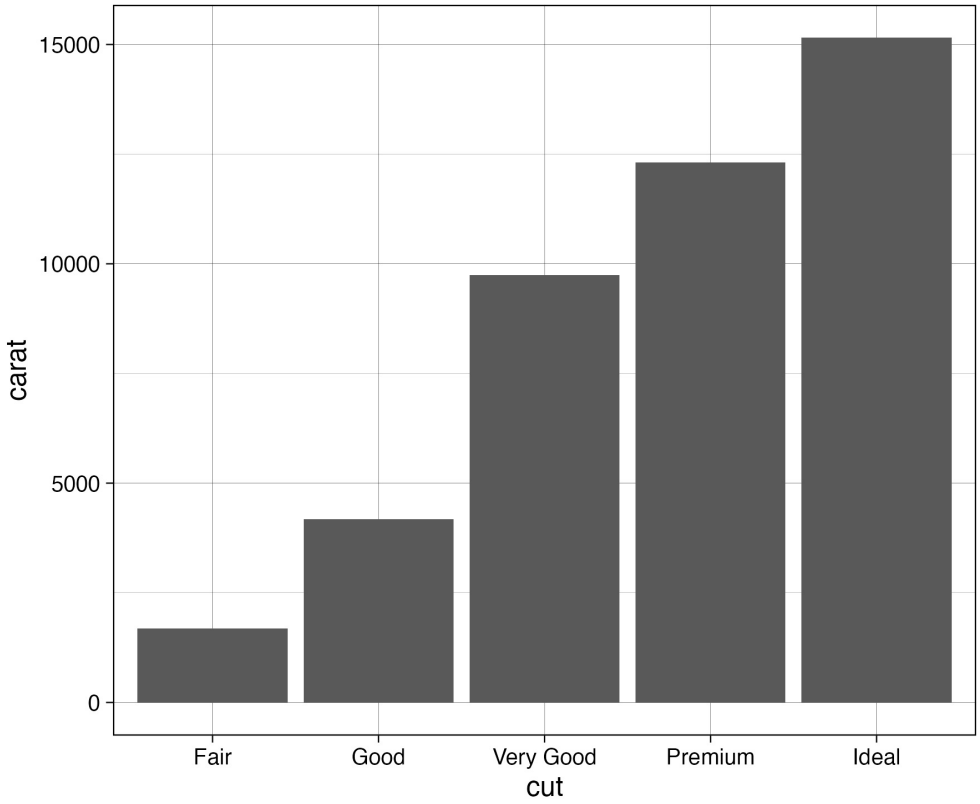
ggplot2: geom_boxplot()

```
diamonds %>%
  ggplot(aes(x = cut,
             y = carat)) +
  geom_boxplot()
```



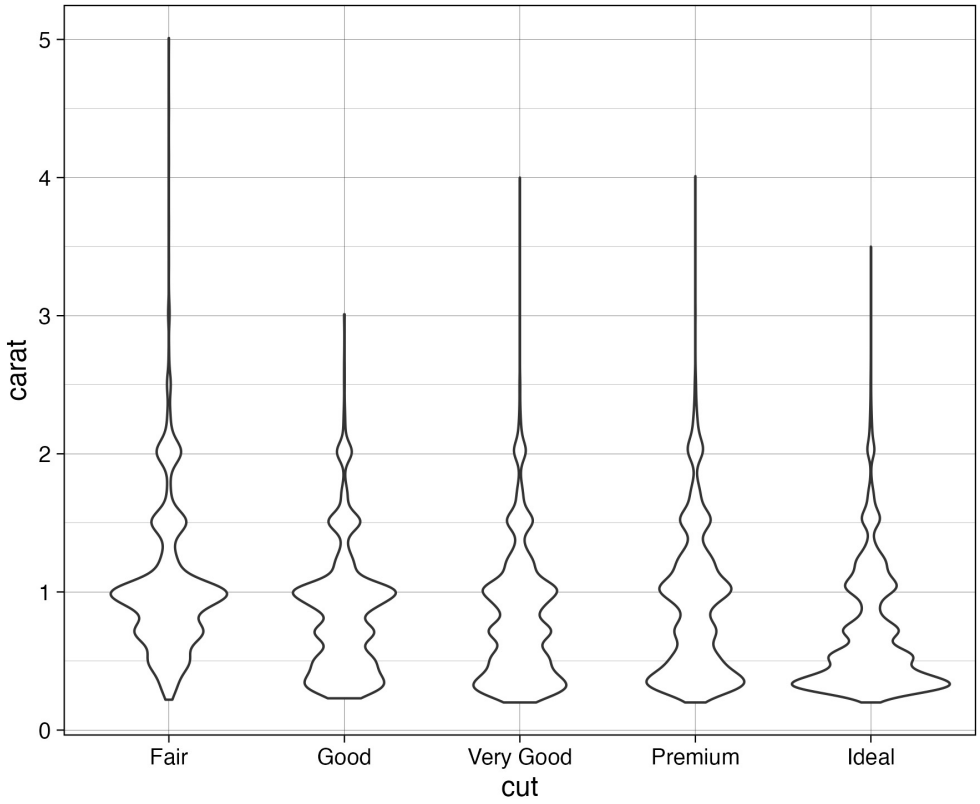
ggplot2: geom_bar()

```
diamonds %>%
  ggplot(aes(x = cut,
             y = carat)) +
  geom_bar(stat = "identity")
```



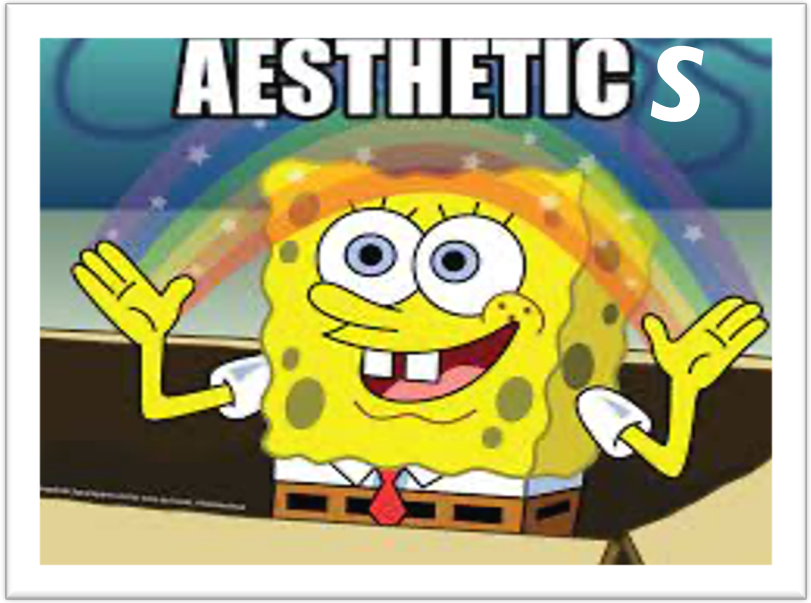
ggplot2: geom_violin()

```
diamonds %>%
  ggplot(aes(x = cut,
             y = carat)) +
  geom_violin()
```




AESTHETIC S

ggplot2: Aesthetics



Aesthetics:

- **Size**

  Change the size of e.g. data points

- **Shape**

  Change the shape of e.g. data points

- **Color**

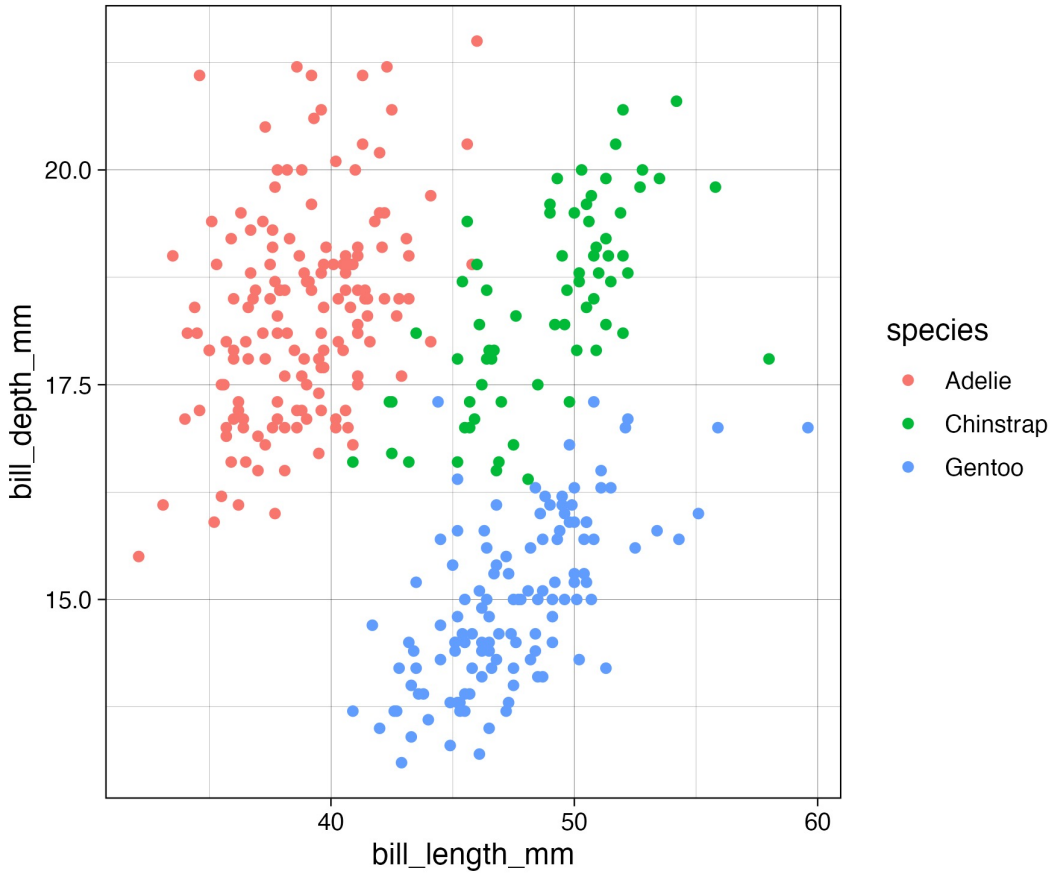
  Change the color of e.g. data points

- **Fill**

  Change the fill of geom, e.g. a boxplot

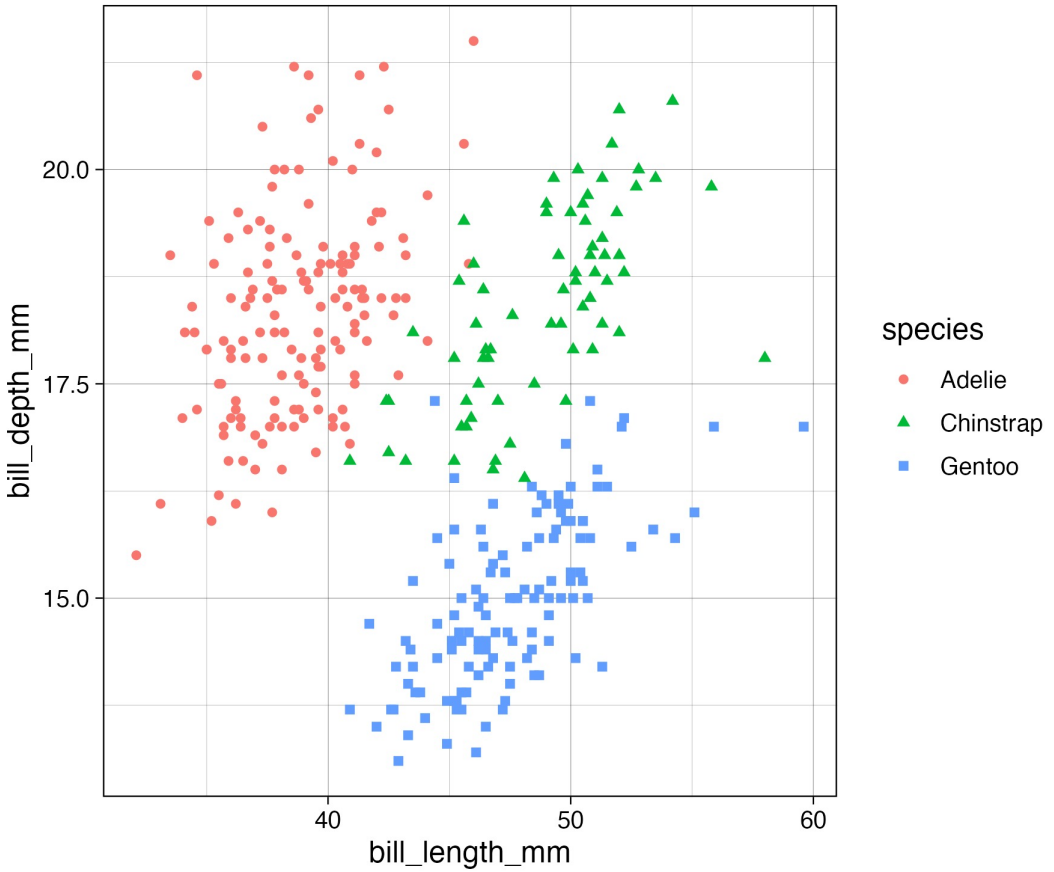
ggplot2: geom_point() + color

```
penguins %>%
  ggplot(aes(x = bill_length_mm,
             y = bill_depth_mm,
             color = species)) +
  geom_point()
```



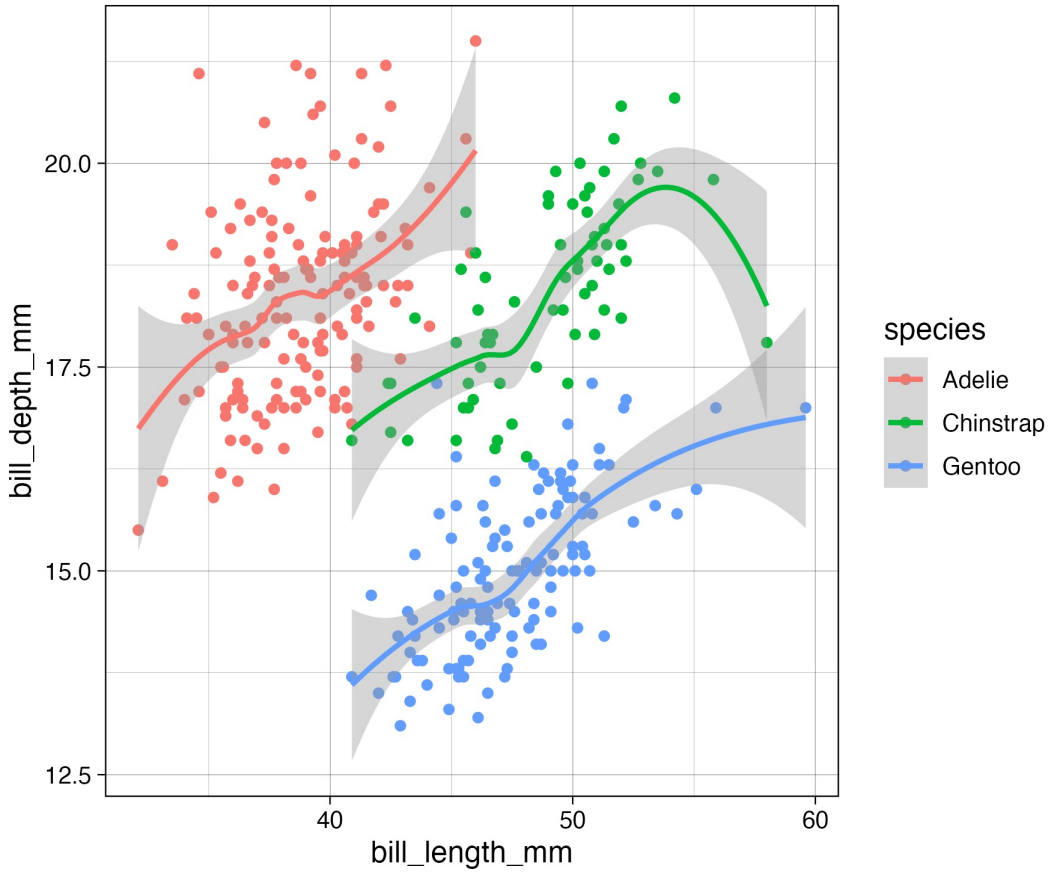
ggplot2: geom_point() + color + shape

```
penguins %>%
  ggplot(aes(x = bill_length_mm,
             y = bill_depth_mm,
             color = species,
             shape = species)) +
  geom_point()
```



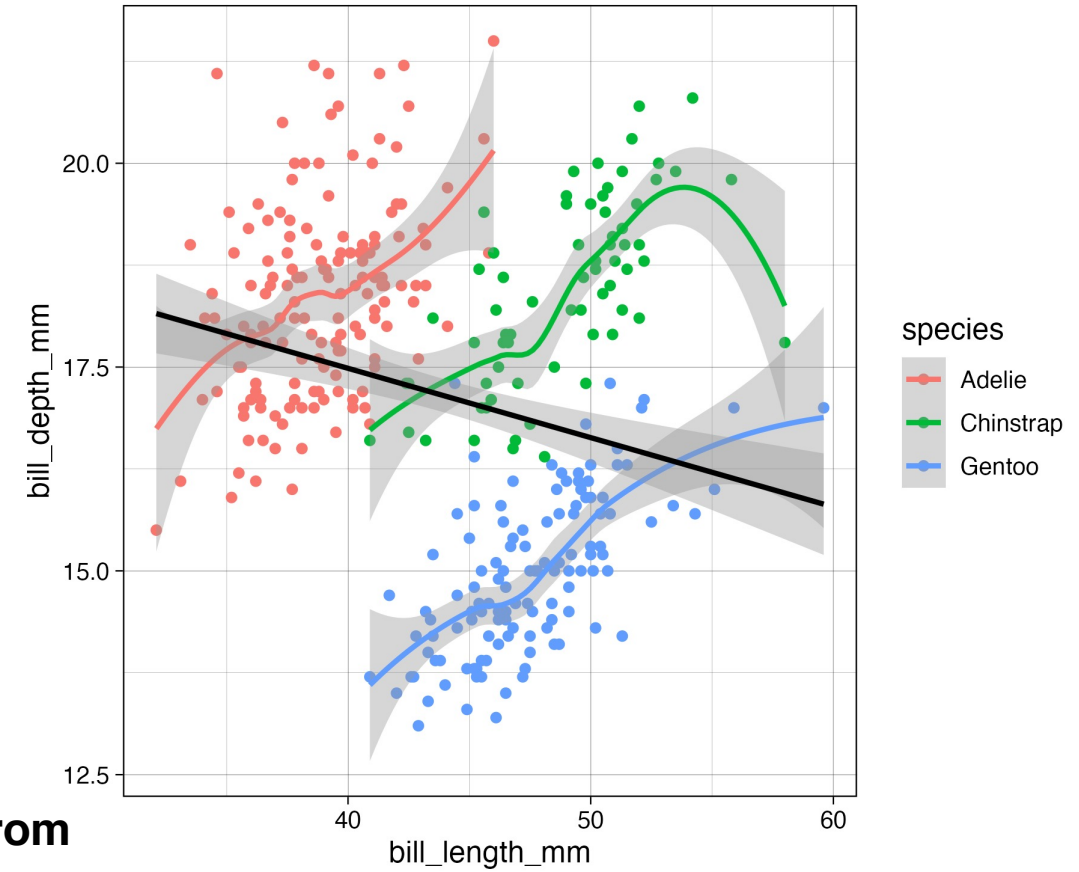
ggplot2: geom_point() + color + geom_smooth

```
penguins %>%
  ggplot(aes(x = bill_length_mm,
             y = bill_depth_mm,
             color = species)) +
  geom_point() +
  geom_smooth()
```



ggplot2: geom_point() + color + geom_smooth + geom_smooth()

```
penguins %>%
  ggplot(aes(x = bill_length_mm,
             y = bill_depth_mm,
             color = species)) +
  geom_point() +
  geom_smooth() +
  geom_smooth(method = "lm",
              color = "black")
```



Overall between-species trend different from individual species group trends;
i.e. *Simpson's paradox*

ggplot2: Where to put the arguments?

OVERALL aesthetics:

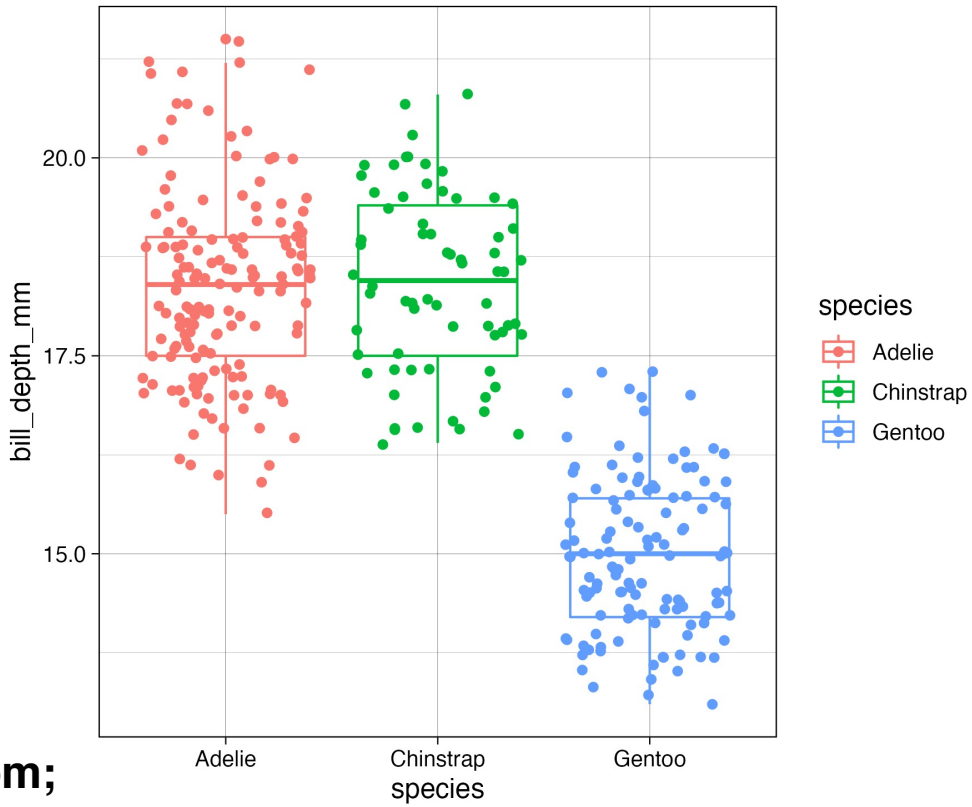
- **Place in the ggplot(aes(…))**

SPECIFIC aesthetics (i.e. only apply to ONE geom)

- **Place in respective geom**

ggplot2: geom_boxplot() + geom_jitter()

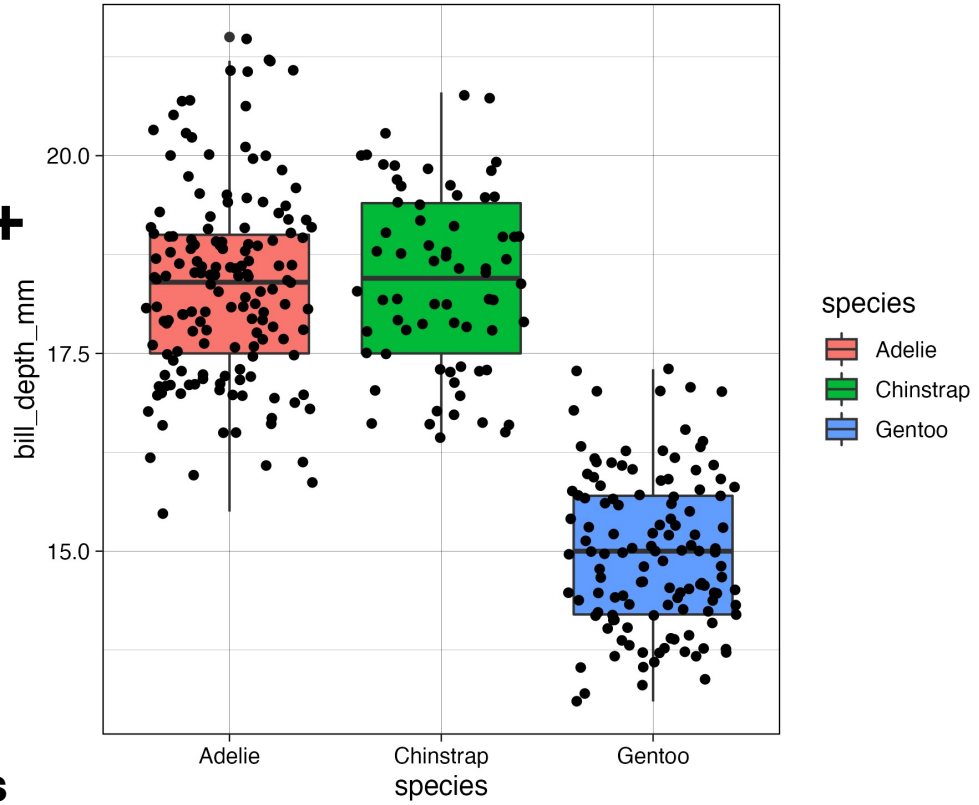
```
penguins %>%
  ggplot(aes(x = species,
             y = bill_depth_mm,
             color = species)) +
  geom_boxplot() +
  geom_jitter()
```



COLOR changes the OUTER color of a geom; FILL changes how e.g. boxplot and violinplots are filled

ggplot2: geom_boxplot() + color + geom_jitter()

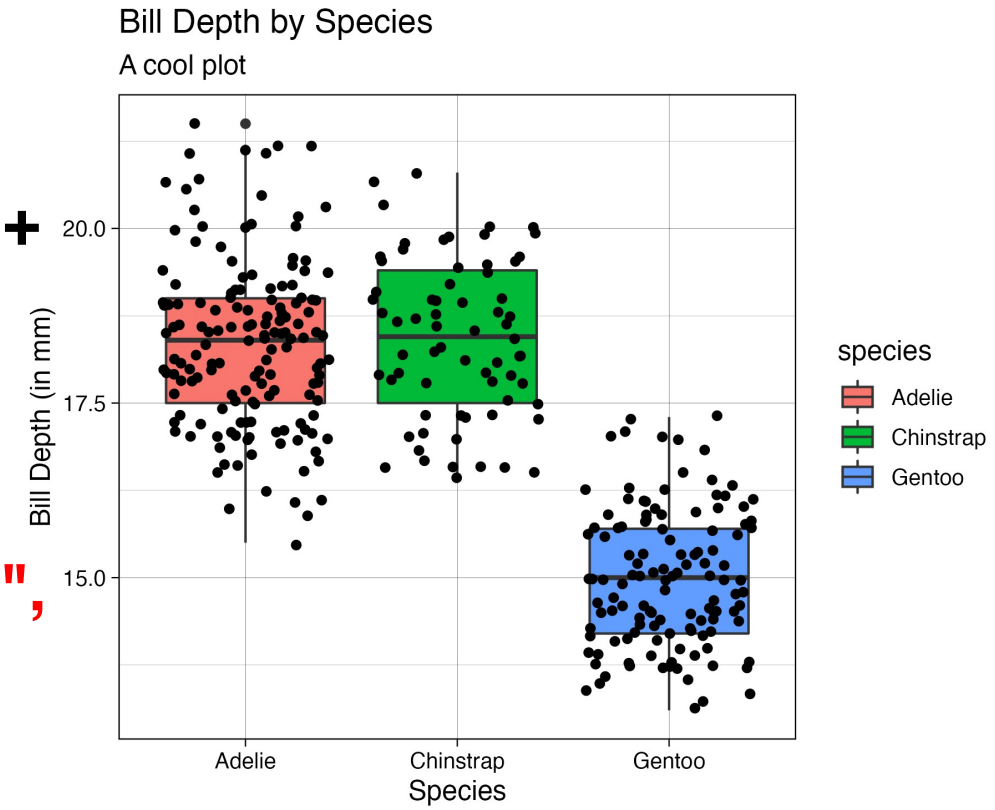
```
penguins %>%
  ggplot(aes(x = species,
             y = bill_depth_mm)) +
  geom_boxplot(aes(fill = species)) +
  geom_jitter()
```



COLOR changes the OUTER color of a geom; FILL changes how e.g. boxplot and violinplots are filled

ggplot2: Add plot labels via labs()

```
penguins %>%
  ggplot(aes(x = species,
             y = bill_depth_mm)) +
  geom_boxplot(aes(fill = species)) +
  geom_jitter() +
  labs(x = "Species",
       y = "Bill Depth (in mm)",
       title = "Bill Depth by Species",
       subtitle = "A cool plot")
```




WITH GREAT POWER COMES
A LOT OF CUSTOMIZATION
imgflip.com

ggplot2: Saving ggplots

SAVE your plot as an object (just like usual, using the = or -> operator)

e.g.

```
billDepth = penguins %>%
  ggplot(aes(x = species,
             y = bill_depth_mm)) +
  geom_boxplot(aes(fill = species)) +
  geom_jitter() +
  labs(x = "Species",
       y = "Bill Depth (in mm)",
       title = "Bill Depth by Species",
       subtitle = "A cool plot")
```

```
ggsave(plot = …,
       filename = "RELATIVE PATH")
```

e.g.

```
ggsave(plot = billDepth,
       filename = "./Figures/billDepth.png)
```

DON'T FORGET TO SPECIFY WHAT TYPE OF FILE YOU ARE SAVING THE PLOT AS (.png, .pdf, .eps, etc.)


LET'S GET OUR HANDS DIRTY
makeameme.org