Making figures with ggplot2

Week 2 (10/8/25)

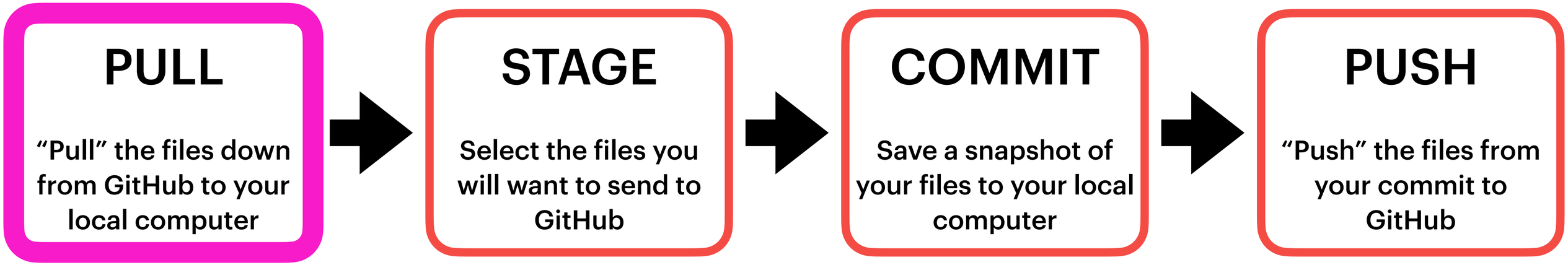
Artwork by Allison Horst

Stepfanie M. Aguillon

Outline of today's class

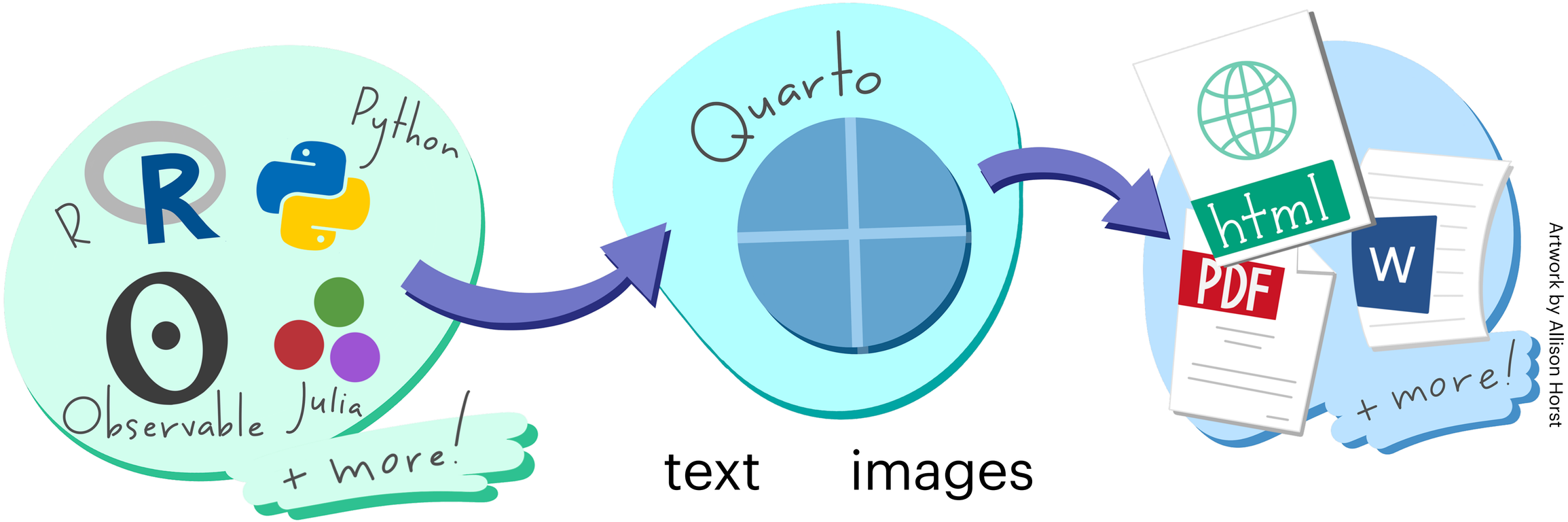
- Introduction to Quarto documents
- Practice creating Quarto documents
- Introduction to `ggplot2`
- Practice creating figures with `ggplot2`

Four steps in using GitHub



Introduction to Quarto

What is Quarto?



text images

runnable code

figures links

Artwork by Allison Horst

Artwork by Allison Horst

You're already familiar with Quarto!

Preview Code Blame Raw

Installation Guide

This course will require that your laptop has all necessary software downloaded and installed **before** our first lecture. This will likely only take ~30 minutes to complete, but please do this process in advance in case you run into issues!

Step 1. R and RStudio

You will need to install (or update to) the most recent versions of R and RStudio. You can find links to download both programs here https://posit.co/download/rstudio-desktop/. Make sure to install the **free** version of RStudio!

Once you have both installed on your local machine, open RStudio. It will automatically connect with the version of R you have installed and you should see the following at the top of the console:

```
R version 4.5.1 (2025-06-13) -- "Great Square Root"
Copyright (C) 2025 The R Foundation for Statistical Computing
```

text

images

links

code chunks

You're already familiar with Quarto!

Preview Code Blame Raw

Installation Guide

This course will require that your laptop has all necessary software downloaded and installed **before** our first lecture. This will likely only take ~30 minutes to complete, but please do this process in advance in case you run into issues!

Step 1. R and RStudio

You will need to install (or update to) the most recent versions of R and RStudio. You can find links to download both programs here https://posit.co/download/rstudio-desktop/. Make sure to install the **free** version of RStudio!

Once you have both installed on your local machine, open RStudio. It will automatically connect with the version of R you have installed and you should see the following at the top of the console:

```
R version 4.5.1 (2025-06-13) -- "Great Square Root"
Copyright (C) 2025 The R Foundation for Statistical Computing
```

install-guide.qmd README.md week1_PS.qmd

Render on Save Render Run Publish

Source Visual Outline

```
---
title: "Installation Guide"
format: gfm
---

This course will require that your laptop has all necessary software downloaded and
installed **before** our first lecture. This will likely only take \~30 minutes to
complete, but please do this process in advance in case you run into issues!

![Artwork by Allison Horst](./images/plan-on-it.jpg){width=75%}
```


```

## Step 1. R and RStudio

You will need to install (or update to) the most recent versions of R and RStudio. You can
find links to download both programs here <https://posit.co/download/rstudio-desktop/>.
Make sure to install the **free** version of RStudio!

Once you have both installed on your local machine, open RStudio. It will automatically
connect with the version of R you have installed and you should see the following at the
top of the console:

```{r, eval=FALSE}
R version 4.5.1 (2025-06-13) -- "Great Square Root"
Copyright (C) 2025 The R Foundation for Statistical Computing
```
```

Step 1. R and ...
Step 2. R pack...
Step 3. Slack
Step 4. GitHub
Step 5. Brief su...

8:64 (Top Level) Quarto

You're already familiar with Quarto!

Installation Guide

This course will require that your laptop has all necessary software downloaded and installed **before** our first lecture. This will likely only take ~30 minutes to complete, but please do this process in advance in case you run into issues!

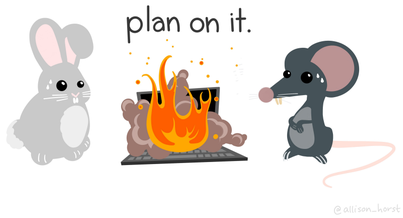


Figure 1: Artwork by Allison Horst

Step 1. R and RStudio

You will need to install (or update to) the most recent versions of R and RStudio. You can find links to download both programs here https://posit.co/download/rstudio-desktop/. Make sure to install the **free** version of RStudio!

Once you have both installed on your local machine, open RStudio. It will automatically connect with the version of R you have installed and you should see the following at the top of the console:

1

install-guide.qmd | README.md | week1_PS.qmd

Render on Save | Render | Run | Publish

Source | Visual | Outline

```
---
title: "Installation Guide"
format: gfm
---

This course will require that your laptop has all necessary software downloaded and
installed **before** our first lecture. This will likely only take \~30 minutes to
complete, but please do this process in advance in case you run into issues!

![Artwork by Allison Horst](./images/plan-on-it.jpg){width=75%}
```


```
## Step 1. R and RStudio

You will need to install (or update to) the most recent versions of R and RStudio. You can
find links to download both programs here <https://posit.co/download/rstudio-desktop/>.
Make sure to install the **free** version of RStudio!

Once you have both installed on your local machine, open RStudio. It will automatically
connect with the version of R you have installed and you should see the following at the
top of the console:

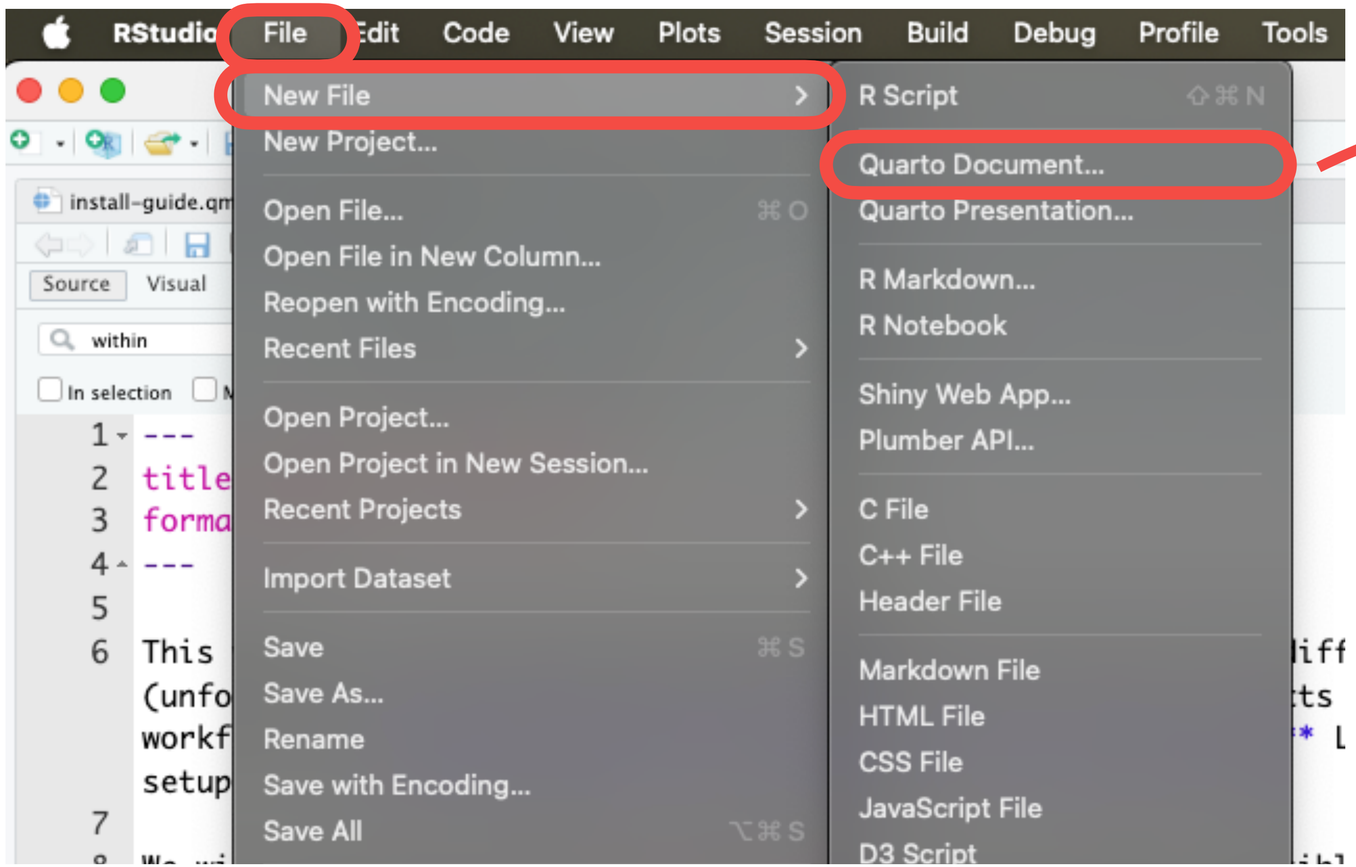
```{r, eval=FALSE}
R version 4.5.1 (2025-06-13) -- "Great Square Root"
Copyright (C) 2025 The R Foundation for Statistical Computing
```
```

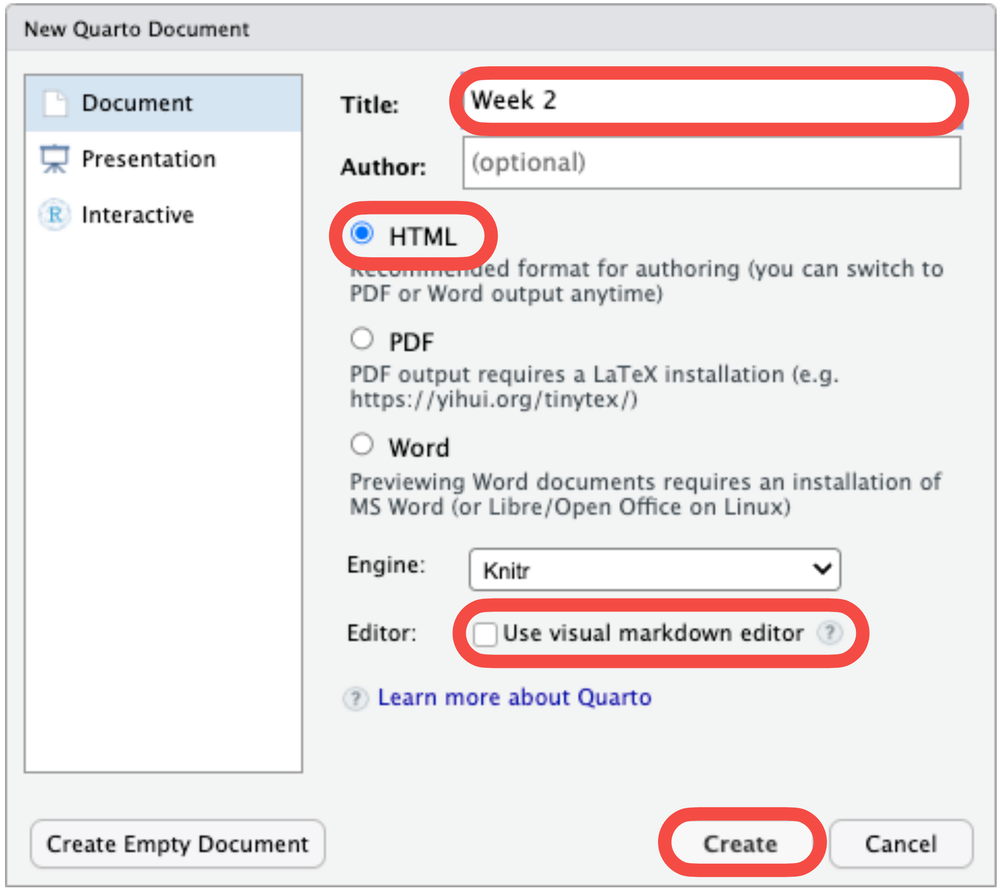
Step 1. R and ...
Step 2. R pack...
Step 3. Slack
Step 4. GitHub
Step 5. Brief su...

8:64 (Top Level) | Quarto

Practice, practice, practice...

Getting started in Quarto

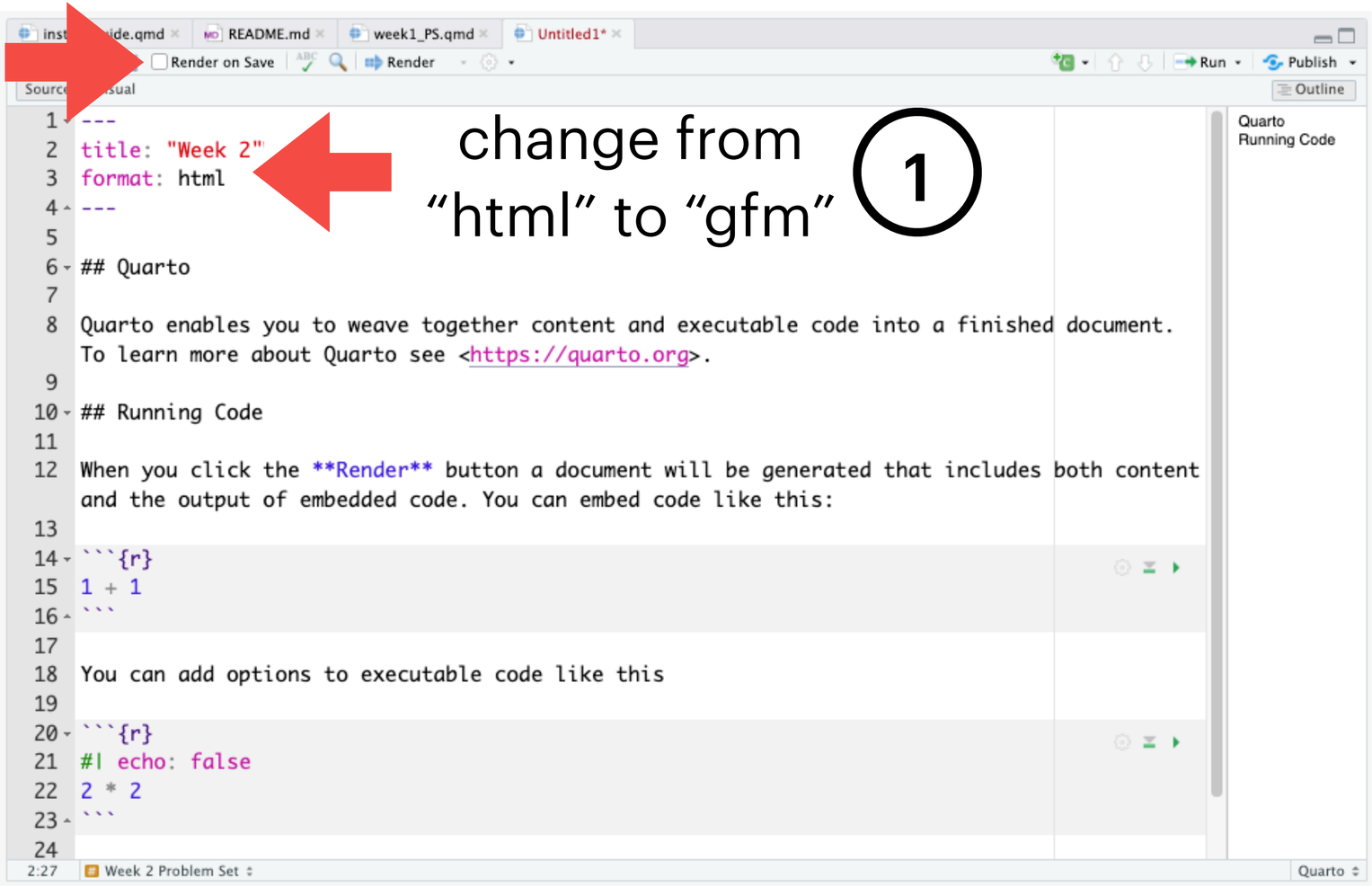




Getting started in Quarto

2

click “Render on Save”



change from “html” to “gfm”

1

3

save your file as

`eeb201_week2.qmd`

live coding

Getting started in Quarto

Week 2

Quarto

Quarto enables you to weave together content and executable code into a finished document. To learn more about Quarto see https://quarto.org.

Running Code

When you click the **Render** button a document will be generated that includes both content and the output of embedded code. You can embed code like this:

```
1 + 1
```

```
[1] 2
```

You can add options to executable code like this

```
[1] 4
```

The `echo: false` option disables the printing of code (only output is displayed).

Basic formatting options in Quarto

Headings

Markdown Syntax	Output
`# Heading 1`	# Heading 1
`## Heading 2`	## Heading 2
`### Heading 3`	### Heading 3
`#### Heading 4`	#### Heading 4
`##### Heading 5`	##### Heading 5
`###### Heading 6`	###### Heading 6

Text Formatting

Markdown Syntax	Output
`*italics*, **bold**, ***bold italics***`	*italics*, **bold**, ***bold italics***
`` `verbatim code` ``	`verbatim code`

Runnable Code

````
```{r}
1 + 1
```
````

https://quarto.org/docs/authoring/markdown-basics.html

Can you recreate this Quarto formatting?

Introduction

Hello!! I am a Quarto document. Have you ever wanted to combine `code`, **text**, and **figures** seamlessly into one document? *Well, you can easily do that with Quarto!*

What I can do

For instance, say you want to show your skills with `R` or the `tidyverse` to a friend. Well, you can run code directly within a Quarto document! Here, let's do some **simple math** to demonstrate:

```
x <- 1
y <- 2

sum(x+y)
```

```
[1] 3
```

You can actually do *a lot* with code in Quarto

Like what?

You ran code in a **code block** above, but maybe you want to run simple commands with in-line code instead (`like this`)? ***Well, Quarto can do that too!*** (*But you'll be learning that a bit later...*)

Introduction to `ggplot2`

What is `ggplot`?

"`ggplot2` has an underlying grammar... that allows you to compose graphs by combining independent components."

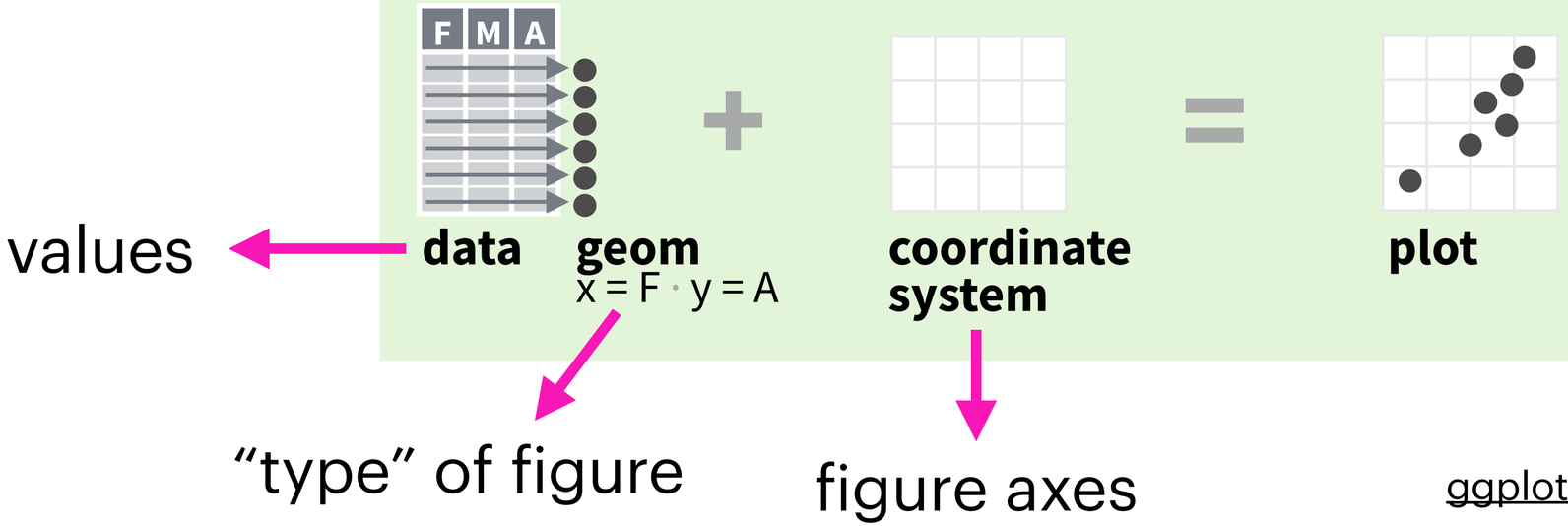
ggplot2: Elegant Graphics for Data Analysis

`tidyverse` package
for data visualization

What is ggplot?

"ggplot2 has an underlying grammar... that allows you to compose graphs by combining independent components."

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.



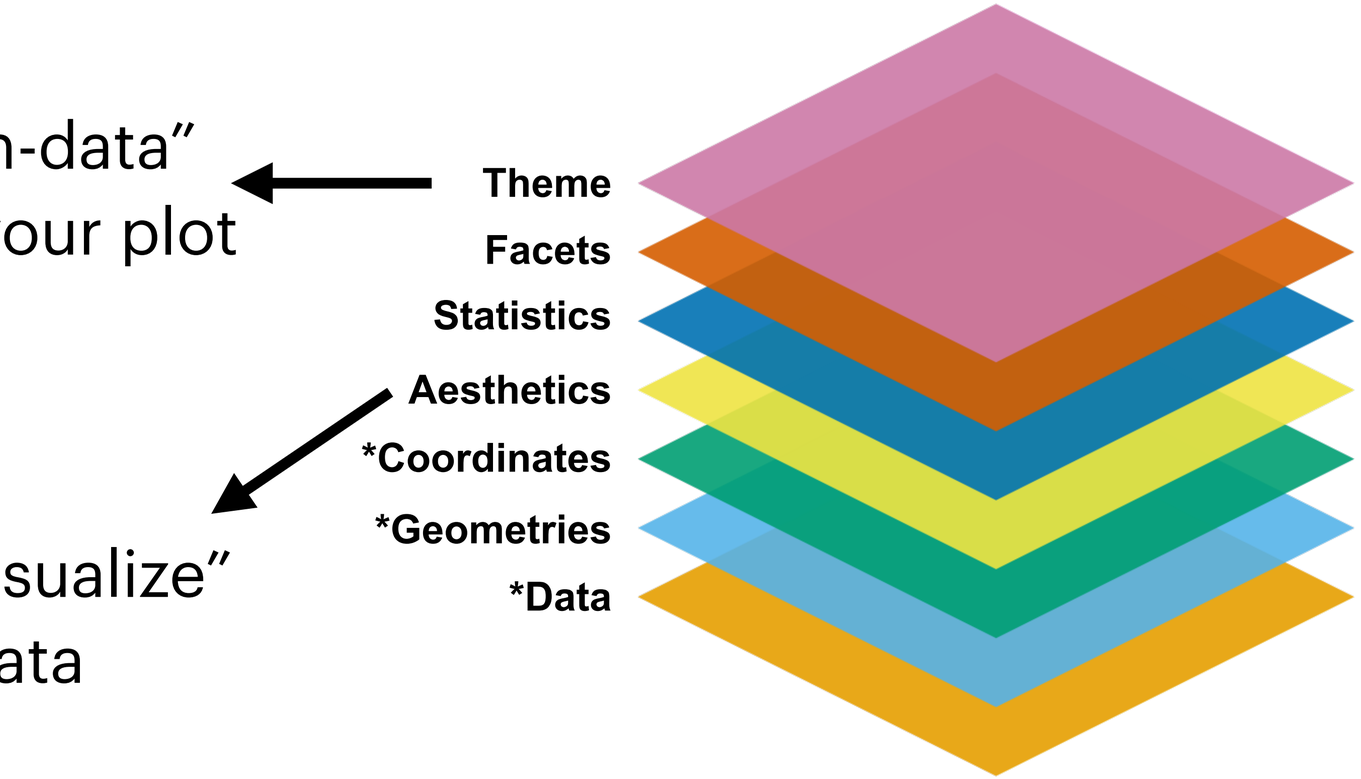
ggplot2: Elegant Graphics for Data Analysis

What is ggplot?

"ggplot2 has an underlying grammar... that allows you to compose graphs by combining independent components."

the "non-data" parts of your plot

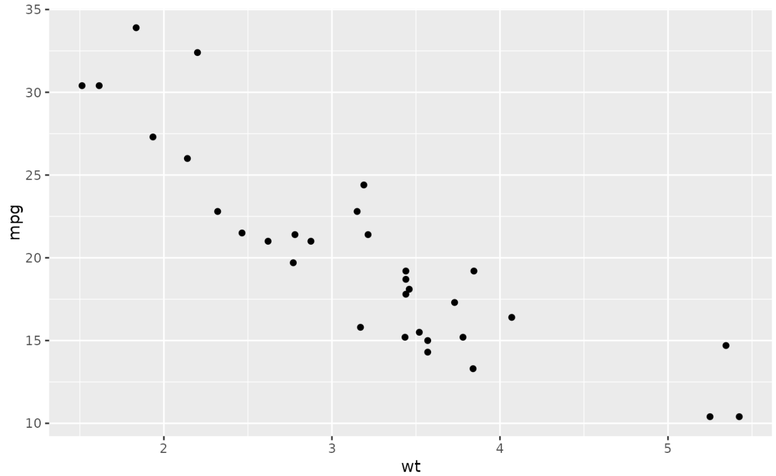
how you "visualize" your data



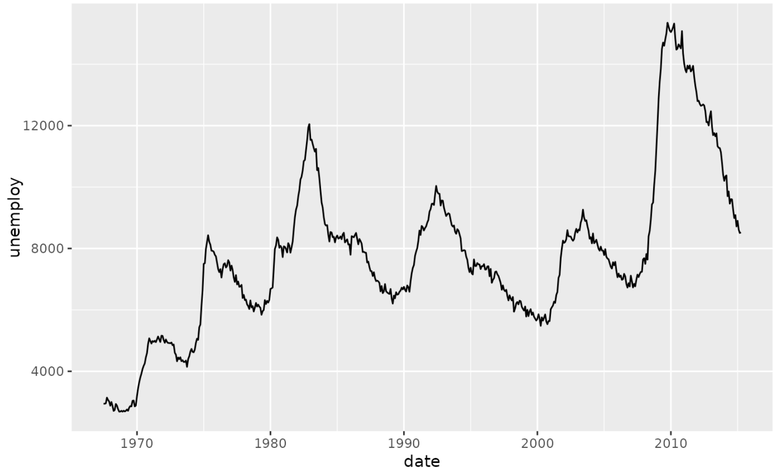
ggplot2: Elegant Graphics for Data Analysis

ggplot2 "geoms"

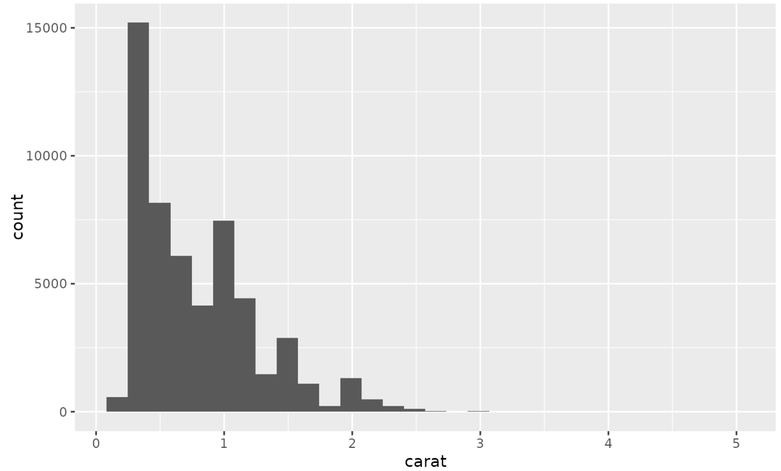
geom_point()



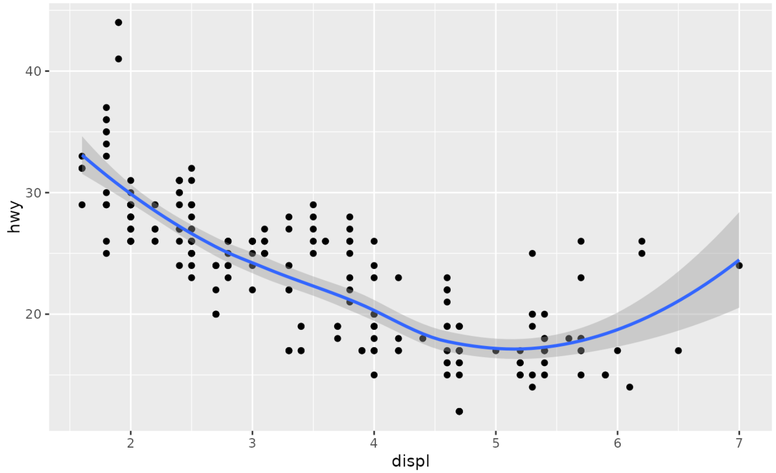
geom_line()



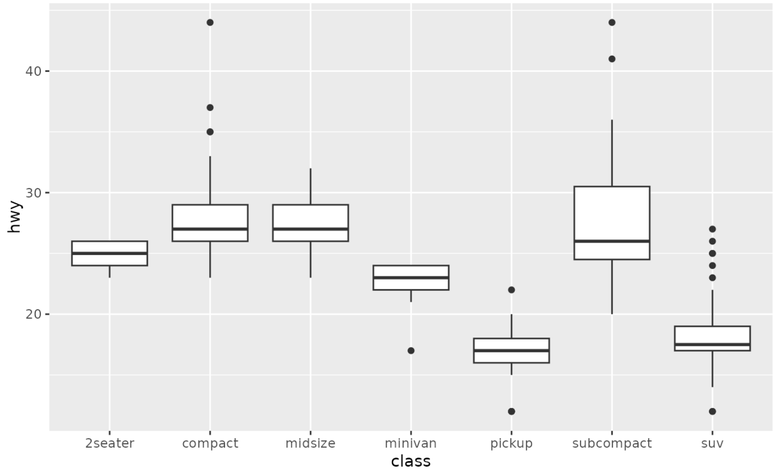
geom_histogram()



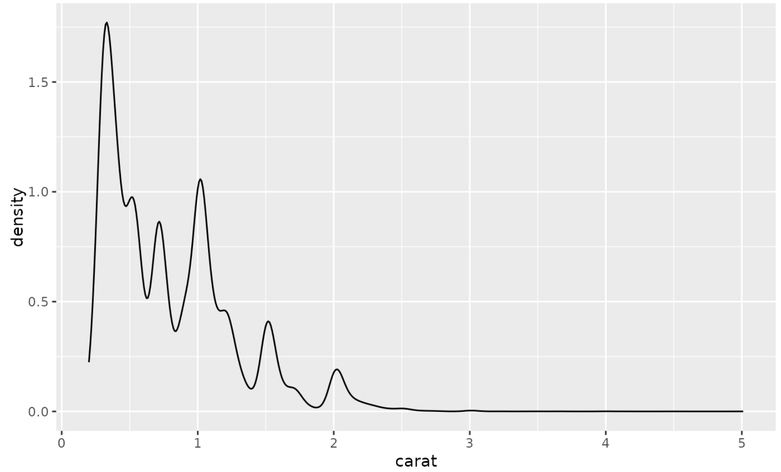
geom_smooth()



geom_boxplot()



geom_density()



ggplot2 "geoms"

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)**)** - 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)**)** - 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

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

Draw the appropriate geometric object depending on the simple features present in the data. aes() arguments: map_id, alpha, color, fill, linetype, linewidth.

nc <- **sf::st_read(**system.file("shape/nc.shp", package = "sf"**)**)

ggplot(nc) +
geom_sf(aes(fill = AREA)**)**

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

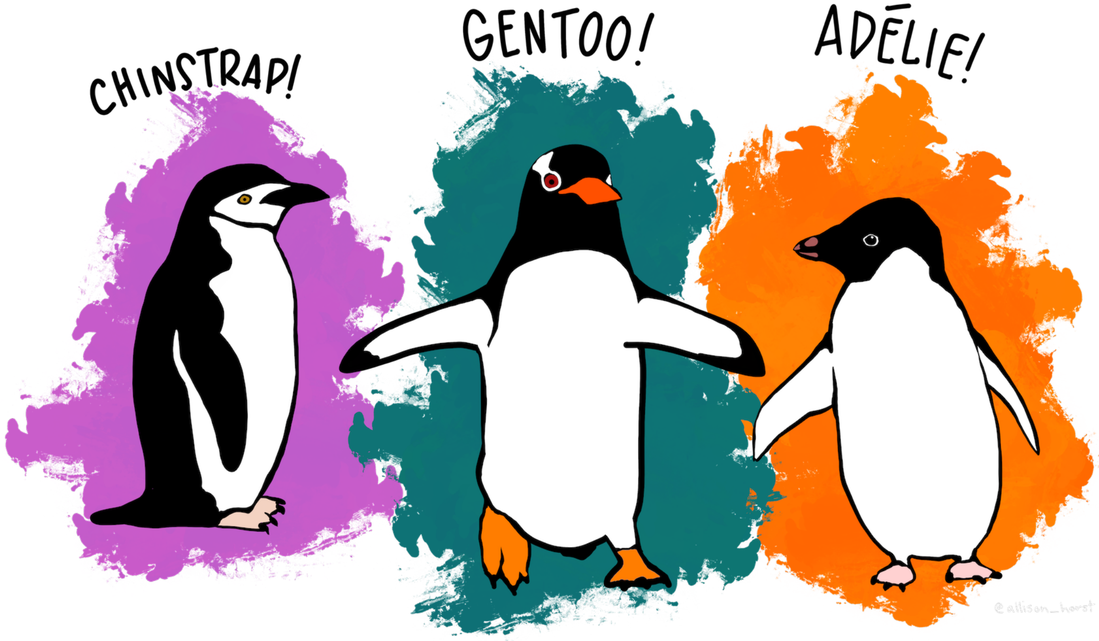
b + geom_abline(aes(intercept = 0, slope = 1)**)**
b + geom_hline(aes(yintercept = lat)**)**
b + geom_vline(aes(xintercept = long)**)**

b + geom_segment(aes(yend = lat + 1, xend = long + 1)**)**
b + geom_spoke(aes(angle = 1:1155, radius = 1)**)**

Practice, practice, practice...

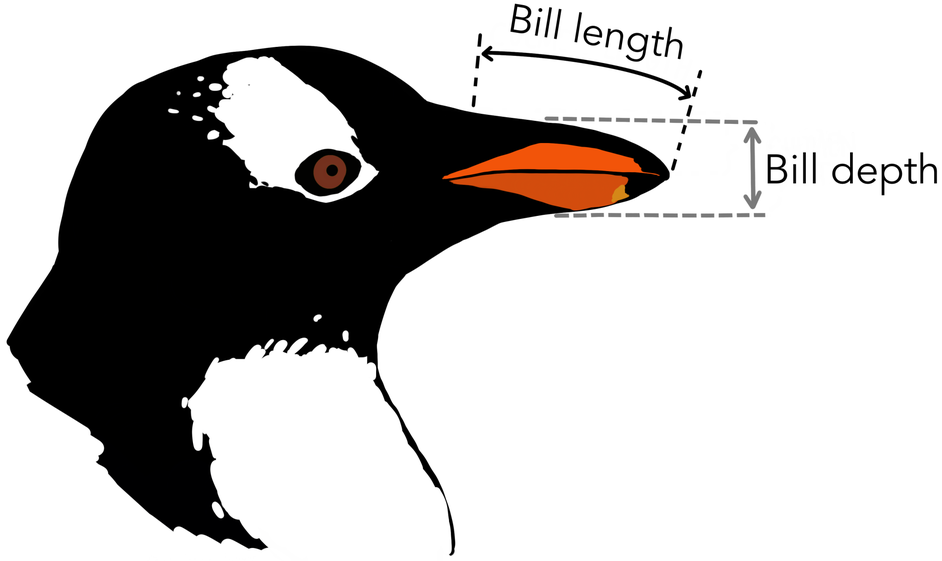
The `penguins` dataset

3 penguin species



Artwork by Allison Horst

Morphological measures



https://allisonhorst.github.io/palmerpenguins/

The `penguins` dataset

Snippet of the dataset:

species	island	bill_len	bill_dep	flipper_len	body_mass	sex	year
Adelie	Torgersen	39.1	18.7	181	3750	male	2007
Adelie	Torgersen	39.5	17.4	186	3800	female	2007
Adelie	Torgersen	40.3	18.0	195	3250	female	2007
Adelie	Torgersen	*NA*	*NA*	*NA*	*NA*	*NA*	2007
Adelie	Torgersen	36.7	19.3	193	3450	female	2007

https://allisonhorst.github.io/palmerpenguins/

Basic code for plotting in `ggplot`

need to load the package first!

```
library(tidyverse)

ggplot()
```

call the plotting function

Basic code for plotting in ggplot

```
library(tidyverse)
ggplot(data) +
  geom(mapping)
```

1 2 3

necessary in ggplot2 to tell R this is all one plot

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.

Basic code for plotting in ggplot

```
library(tidyverse)

ggplot(data = <DATA>) +
 geom_<TYPE>(mapping = aes(<MAPPINGS>))
```

which dataset to plot

"aesthetics mapping" = how variables are mapped to visual properties

what type of plot

what coordinates for plotting

Basic code for plotting in ggplot

```
library(tidyverse)

ggplot(data = <DATA>) +
 geom_<TYPE>(mapping = aes(<MAPPINGS>))
```

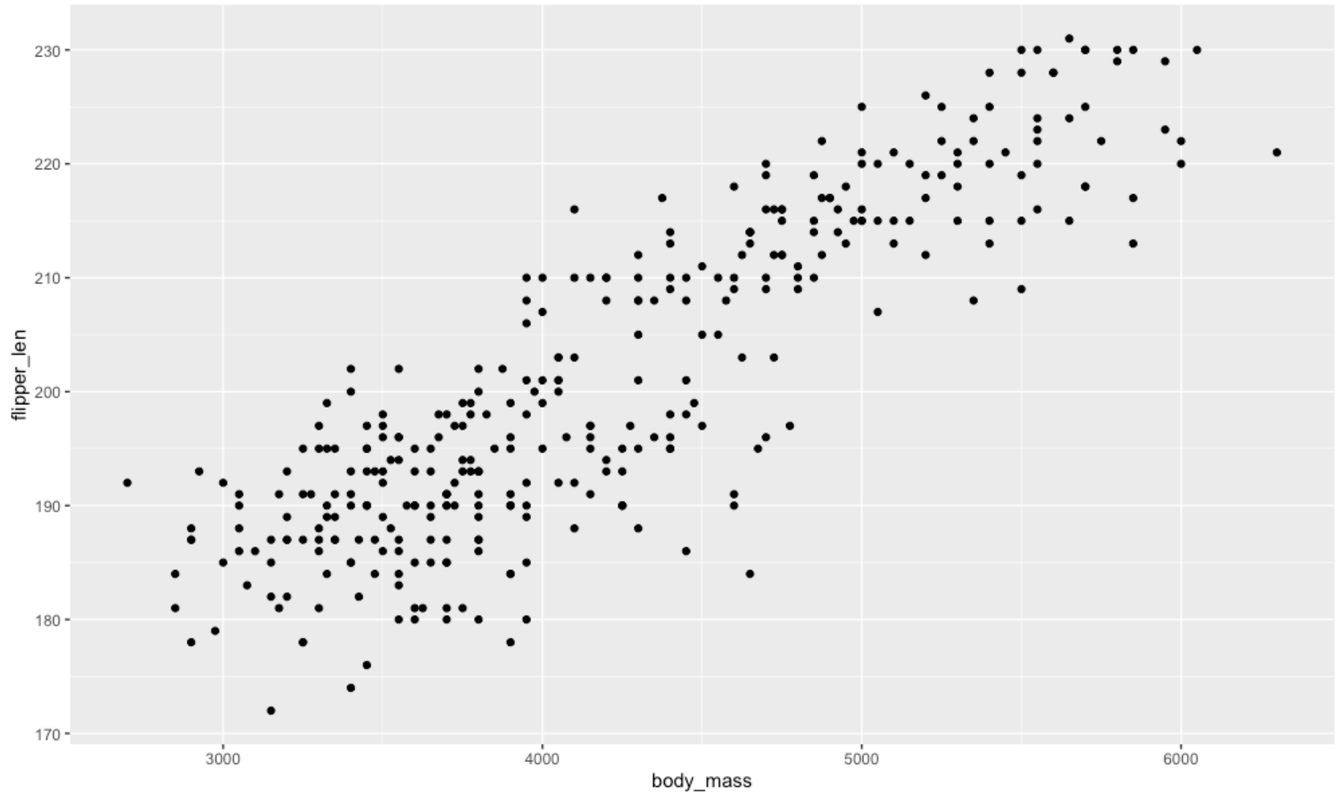
penguins

geom_point

x = body_mass, y = flipper_len

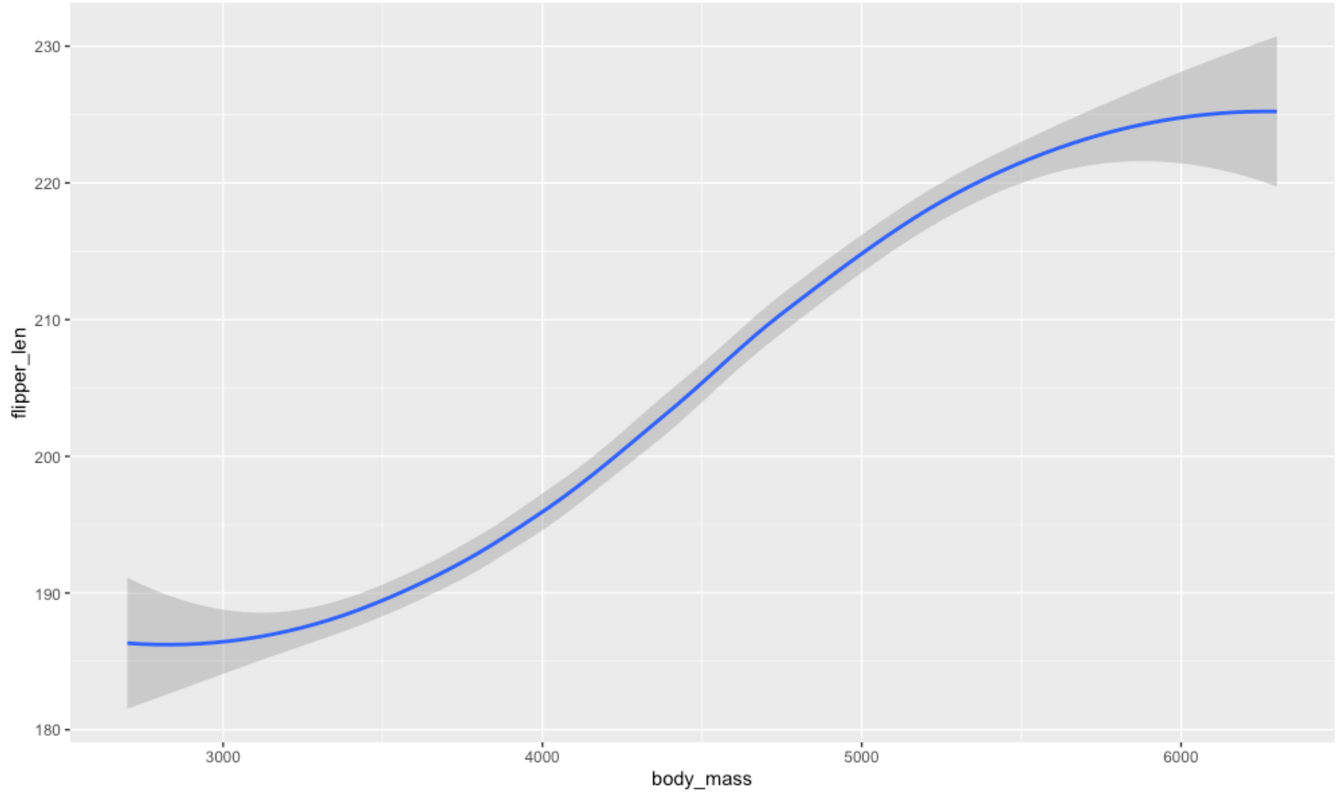
geom_point()

```
ggplot(data = penguins) +
  geom_point(mapping = aes(x = body_mass, y = flipper_len))
```



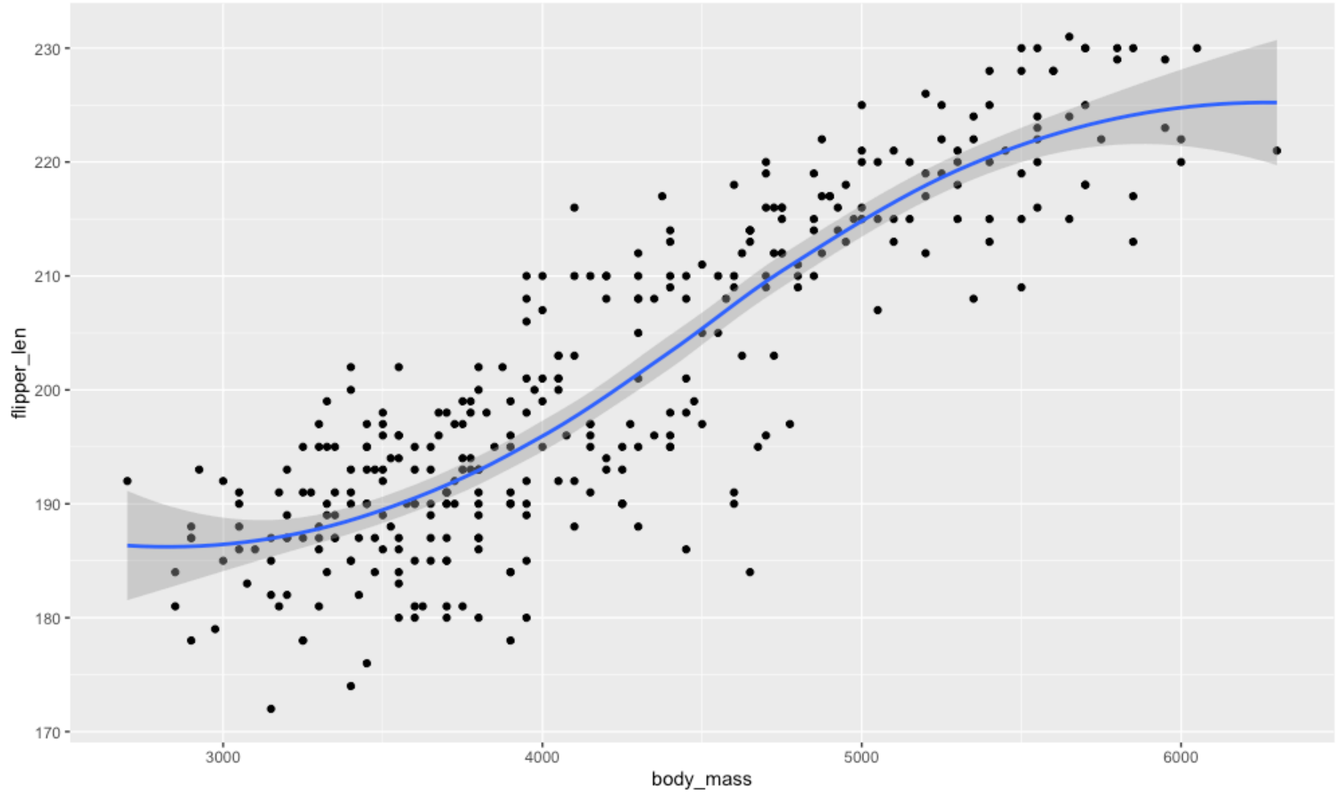
geom_smooth()

```
ggplot(data = penguins) +
  geom_smooth(mapping = aes(x = body_mass, y = flipper_len))
```

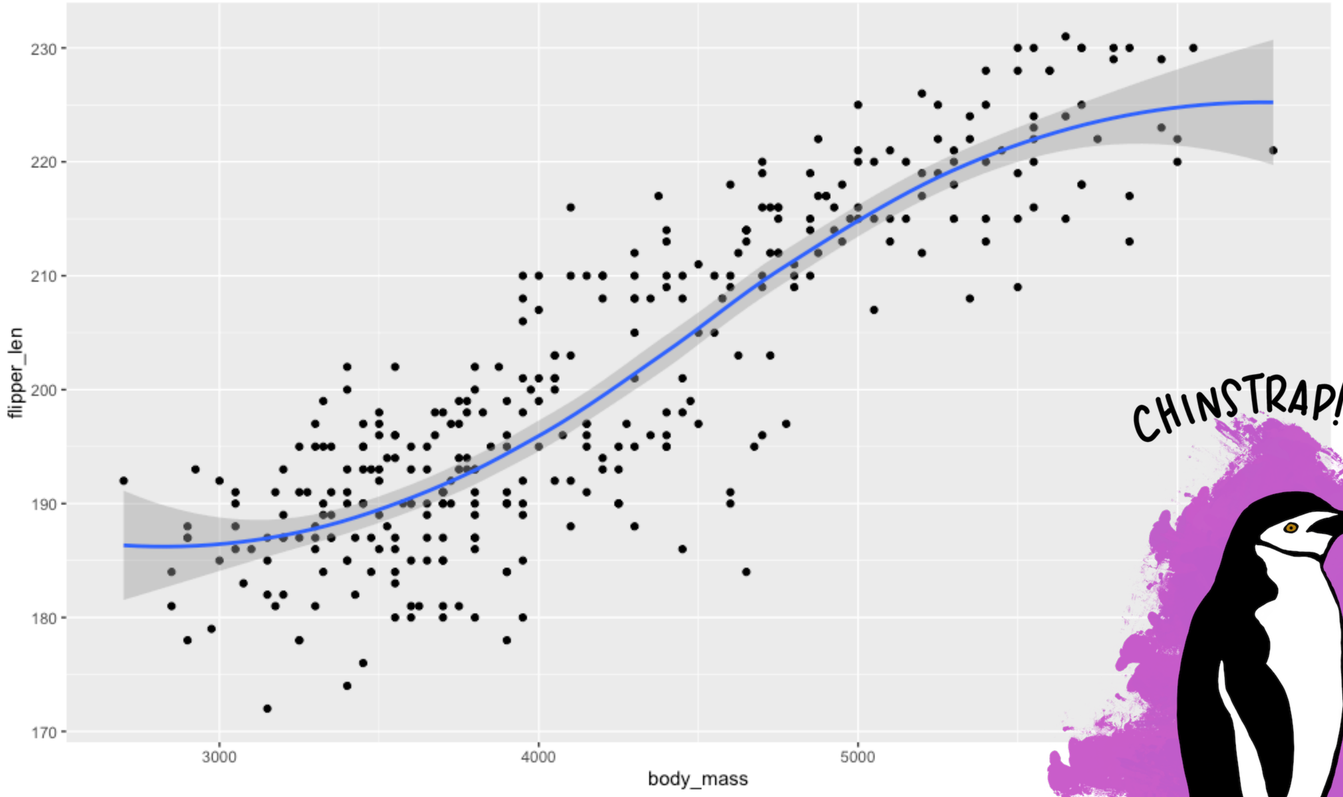


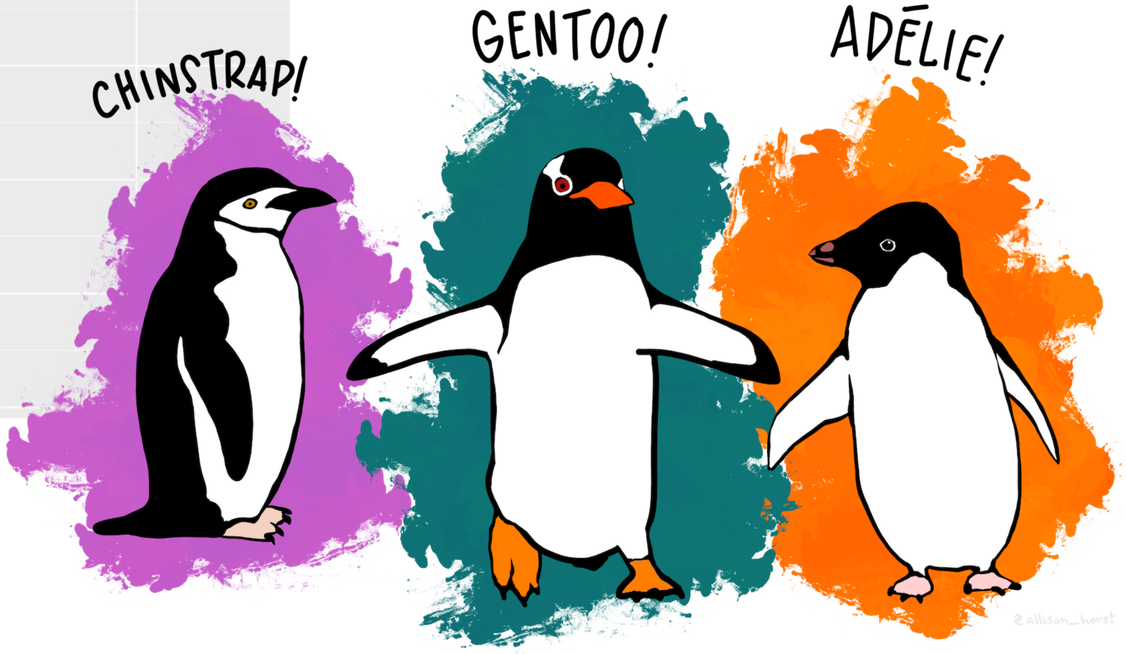
geom_point() + geom_smooth()

```
ggplot(data = penguins) +
  geom_point(mapping = aes(x = body_mass, y = flipper_len)) +
  geom_smooth(mapping = aes(x = body_mass, y = flipper_len))
```



Getting more advanced...



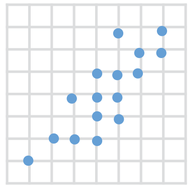


Artwork by Allison Horst

Where would you add in species?

```
ggplot(data = penguins)+
  geom_point(mapping = aes(x = body_mass, y = flipper_len))
```

"aesthetics mapping" = how variables are mapped to visual properties

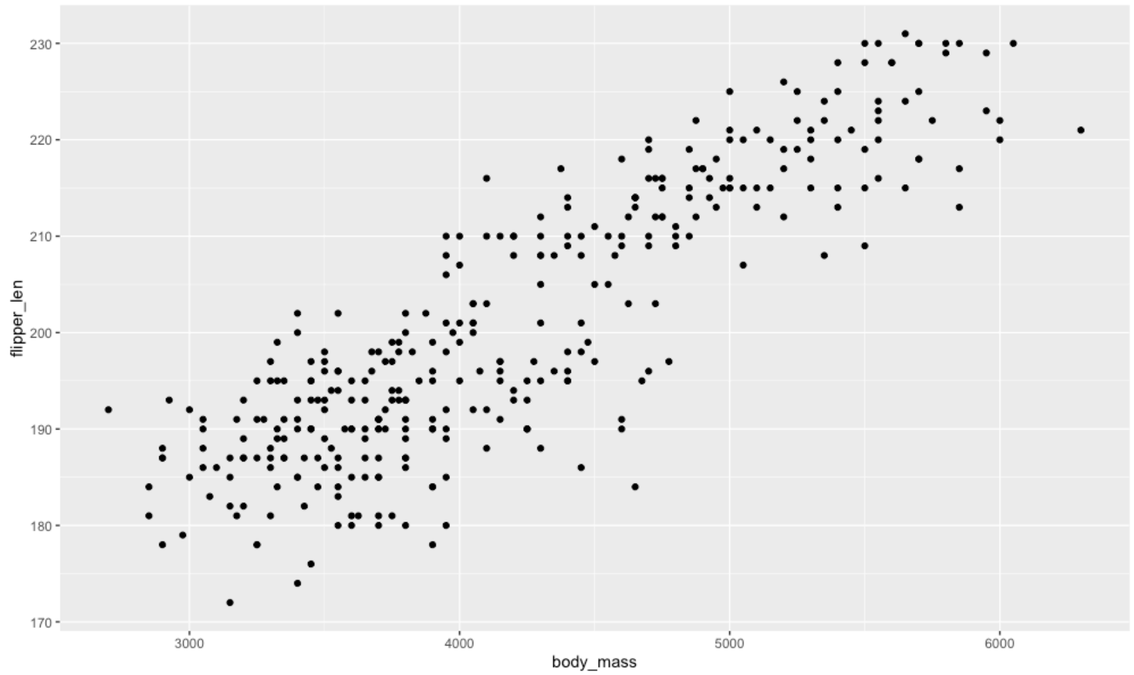


e + geom_point()
x, y, alpha, color, fill, shape, size, stroke

ggplot2 Cheat Sheet

Where would you add in species?

```
ggplot(data = penguins) +
  geom_point(mapping = aes(x = body_mass, y = flipper_len, color = species))
```



Aesthetics or not aesthetics?

```
ggplot(data = penguins) +
  geom_point(mapping = aes(x = body_mass, y = flipper_len, color = species))
```

```
ggplot(data = penguins) +
  geom_point(mapping = aes(x = body_mass, y = flipper_len), color = species)
```


```
Error: object 'species' not found
```

Aesthetics or not aesthetics?

```
ggplot(data = penguins) +
  geom_point(mapping = aes(x = body_mass, y = flipper_len, color = species))
```

```
ggplot(data = penguins) +
  geom_point(mapping = aes(x = body_mass, y = flipper_len), color = "red")
```

Aesthetics or not aesthetics?

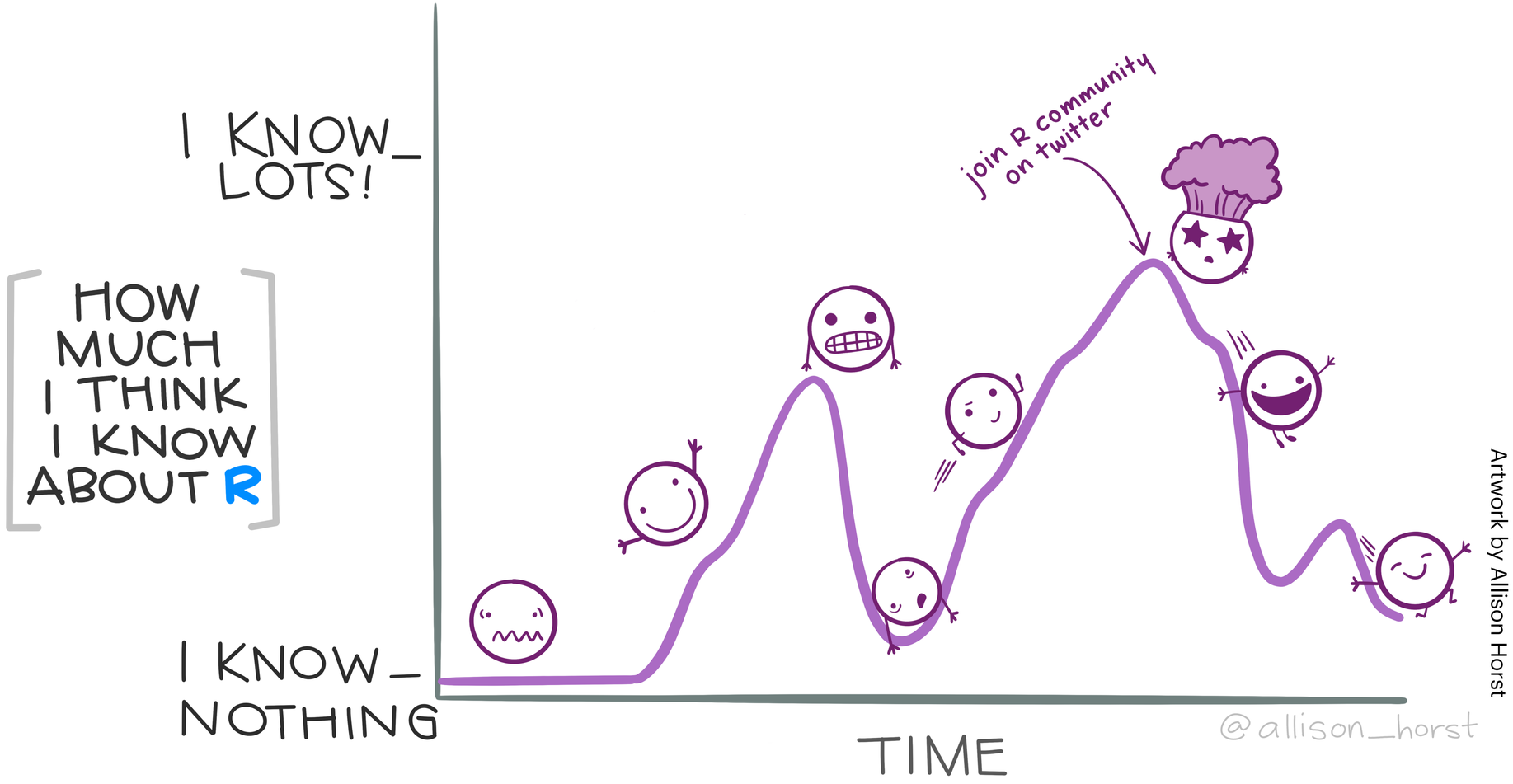
visual properties *inside* the aesthetics will use your *variables* for mapping

```
ggplot(data = penguins) +
  geom_point(mapping = aes(x = body_mass, y = flipper_len, color = species))
```

```
ggplot(data = penguins) +
  geom_point(mapping = aes(x = body_mass, y = flipper_len), color = "red")
```

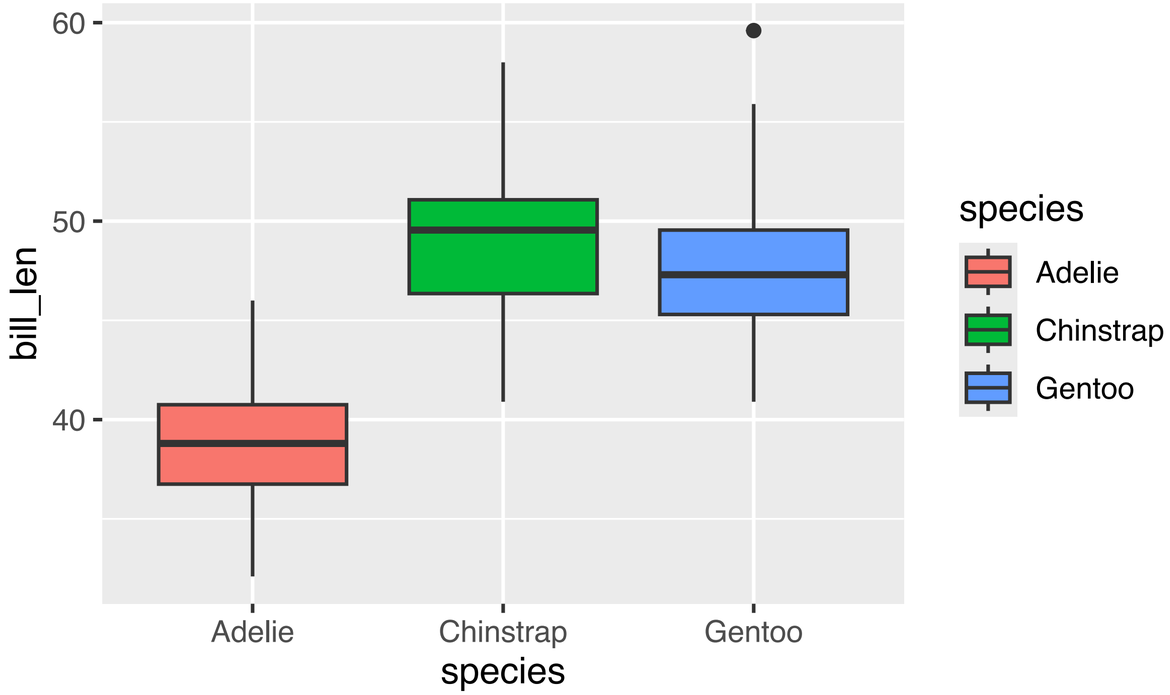
visual properties *outside* the aesthetics will influence the *entire plot*

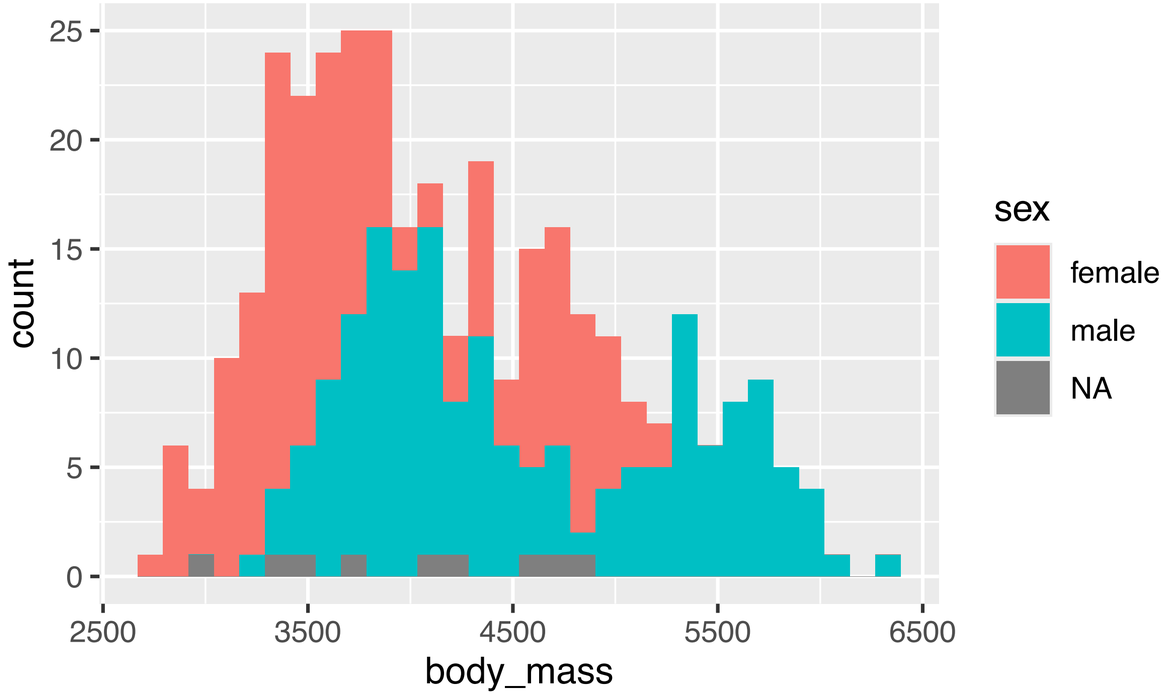
Learning new things is really hard!



live coding

Can you recreate these plots?





Further reading

- Markdown Basics
- Wickham, Çetinkaya-Rundel, & Grolemund 2017, *R 4 Data Science*, Chapter 1
- Data Visualization with `ggplot2` Cheatsheet
- Wickham, *ggplot2: Elegant Graphics for Data Analysis (3e)*