

Creating Figures as an Intro to R

Using the `ggplot2` package

 [steffilazerte](#)
 @steffilazerte@fosstodon.org
 [@steffilazerte](#)
 [steffilazerte.ca](#)

Dr. Steffi LaZerte



Analysis and Data Tools for Science



Preamble

Me and my creatures



My garden



What about you?

- Name
- Background (Role, Area of study, etc.)
- Familiarity with R or Programming
- Creatures (furry, feathery, scaley, green or otherwise)?

Outline

1. A little about R
2. Creating figures with **ggplot2**
3. Combining figures with **patchwork**
4. Saving figures

Taken this or a similar workshop before?

During activities consider...

- Extra activities labeled “Too Easy?”
- Using your own data
- Exploring other aspects of ggplot2 that interest you

Feel free to ask questions even if it’s not the “official” activity!

What is R?

R is a Programming language

A programming **language** is a way to give instructions in order to get a computer to do something

- You need to know the language (i.e., the code)
- Computers don't know what you mean, only what you type (unfortunately)
- Spelling, punctuation, and capitalization all matter!

For example

R, what is 56 times 5.8?

```
56 * 5.8
```

```
[1] 324.8
```

Use code to tell R what to do

R, what is the average of numbers 1, 2, 3, 4?

```
mean(c(1, 2, 3, 4))
```

```
[1] 2.5
```

R, save this value for later

```
steffis_mean <- mean(c(1, 2, 3, 4))
```

R, multiply this value by 6

```
steffis_mean * 6
```

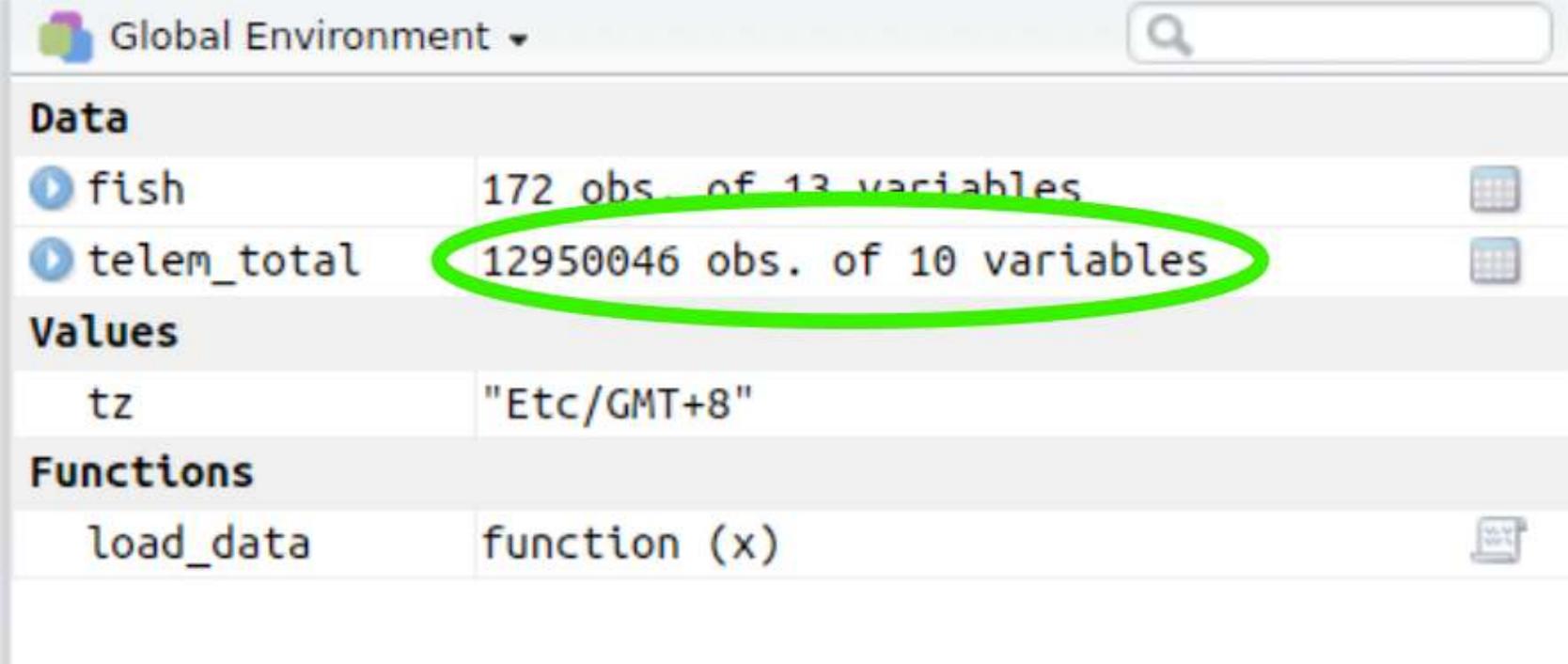
```
[1] 15
```

Why R?

R is hard

```
# Get in circle around city
circle <- data.frame()
cutoff <- 10
for(i in unique(gps$region)) {
  n <- nrow(gps[gps$region == i,]) ##number of IDs
  if(i == "wil") tmp <- geocode("Williams Lake, Canada")
  if(i == "kam") tmp <- geocode("Kamloops, Canada")
  if(i == "kel") tmp <- geocode("Kelowna, Canada")
  temp <- data.frame()
  for(a in 1:n){
    if(a <= cutoff) temp <- rbind(temp, gcDestination(lon = tmp$lon,
                                                       lat = tmp$lat,
                                                       bearing = (a*(360/(cutoff))-360/(cutoff)),
                                                       dist = 20,
                                                       dist.units = "km",
                                                       model = "WGS84"))
    if(a > cutoff) temp <- rbind(temp, gcDestination(lon = tmp$lon,
                                                       lat = tmp$lat,
                                                       bearing = ((a-cutoff)*(360/(max(table(gps$region
                                                       ))-10))-360/(max(table(gps$region))-cutoff)),
                                                       dist = 35,
                                                       dist.units = "km",
                                                       model = "WGS84"))
  }
  circle <- rbind(circle, cbind(temp,
                                region = i,
                                hab = gps$hab[gps$region == i],
                                spl = gps$spl.orig[gps$region == i],
```

But R is powerful (and reproducible)!



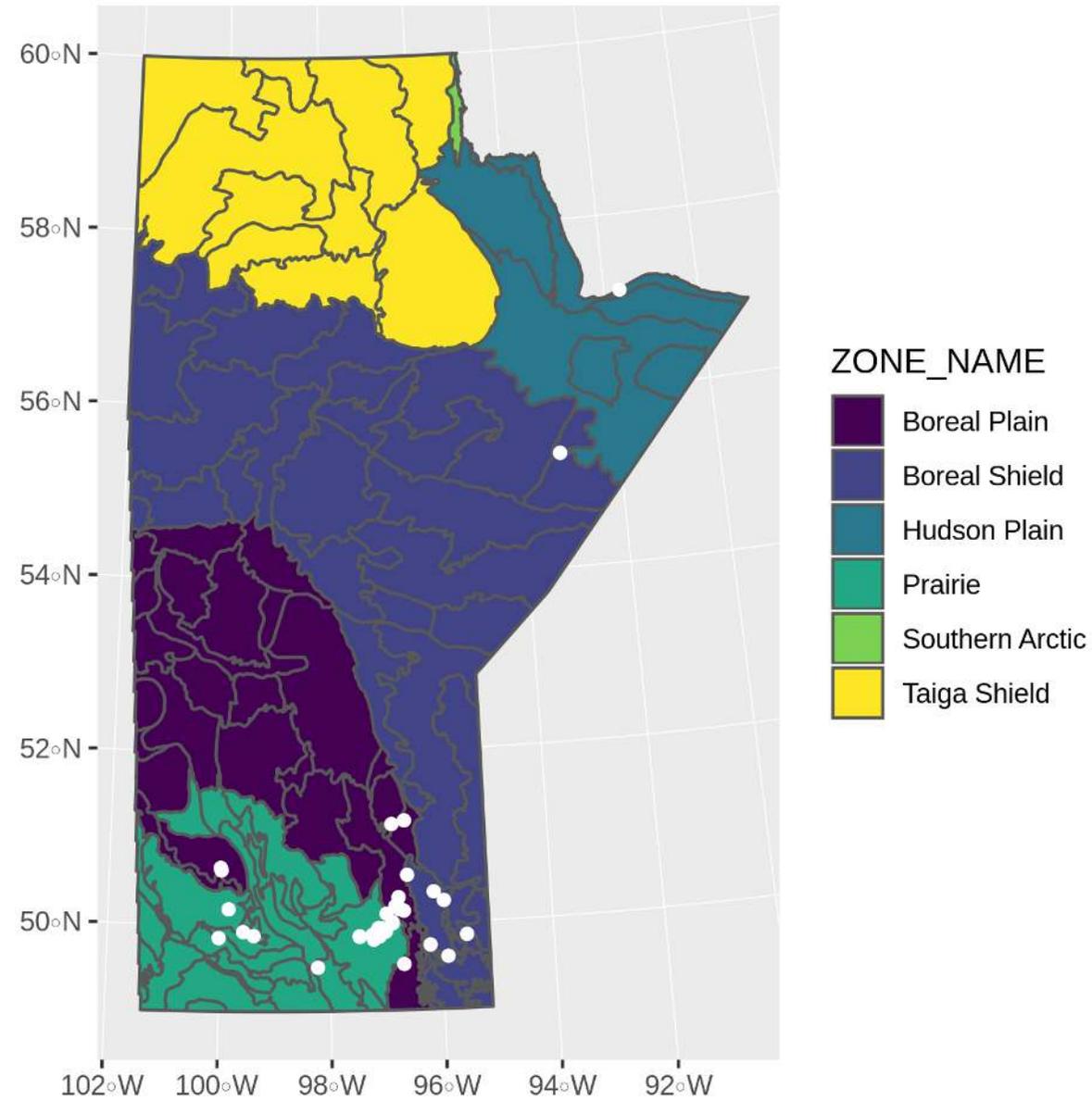
Global Environment

Data	
fish	172 obs. of 13 variables
telem_total	12950046 obs. of 10 variables

Values	
tz	"Etc/GMT+8"

Functions	
load_data	function (x)

R is also beautiful



R is affordable (i.e., free!)

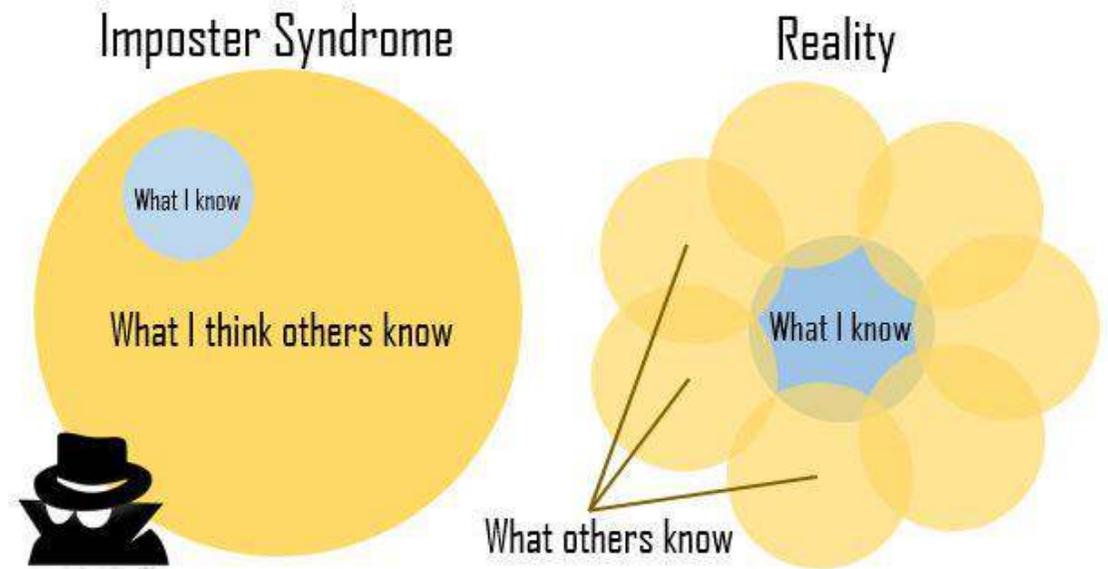
R is available as Free Software under the terms of the [Free Software Foundation's GNU General Public License](#) in source code form. It compiles and runs on a wide variety of UNIX platforms and similar systems (including FreeBSD and Linux), Windows and MacOS.

ImpostR Syndrome

Impost  R
Synddrome

ImpostR Syndrome

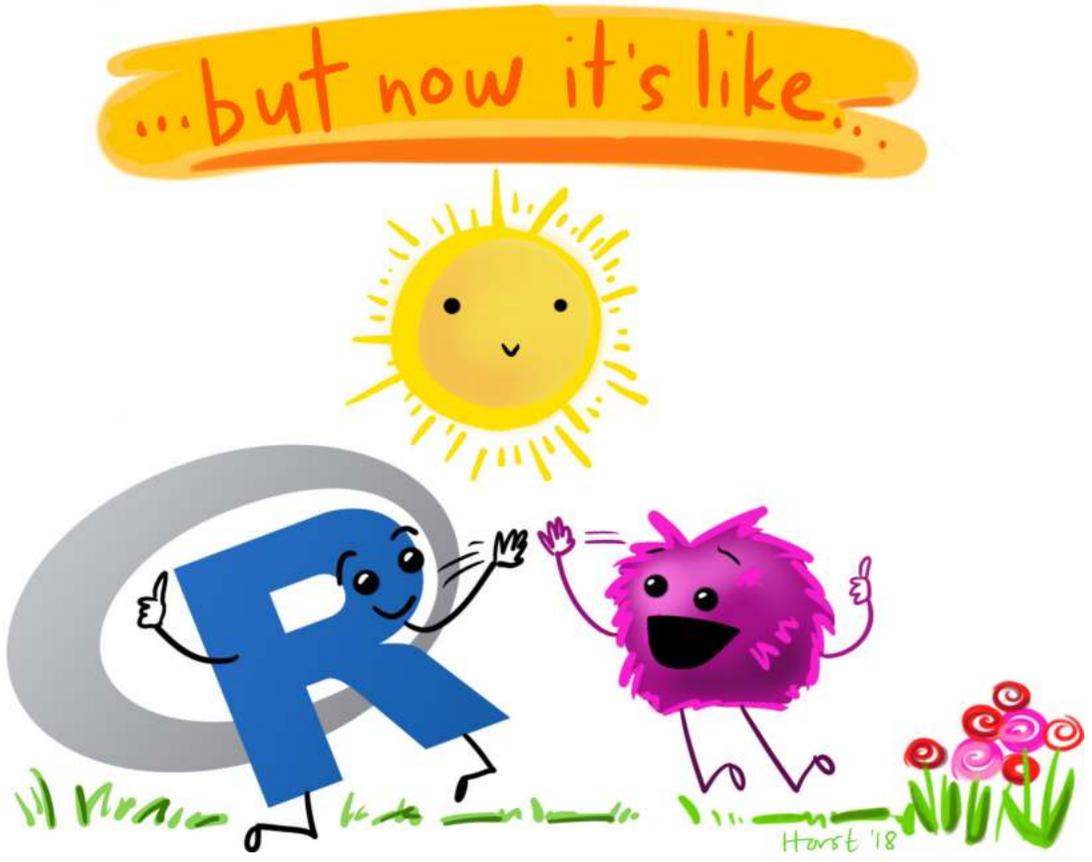
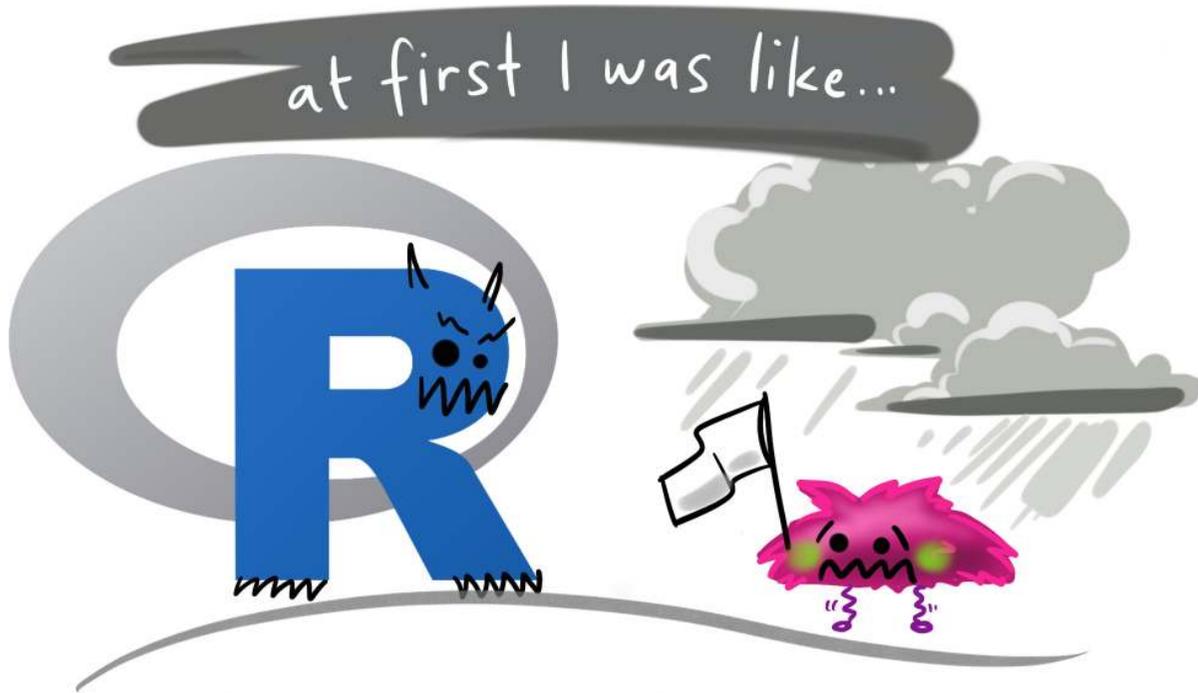
ImpostR Syndrome



Moral of the story?

Make friends, code in groups, learn together and don't beat yourself up

The Goal



About R

Code, Output, Scripts

Code

- The actual commands

Output

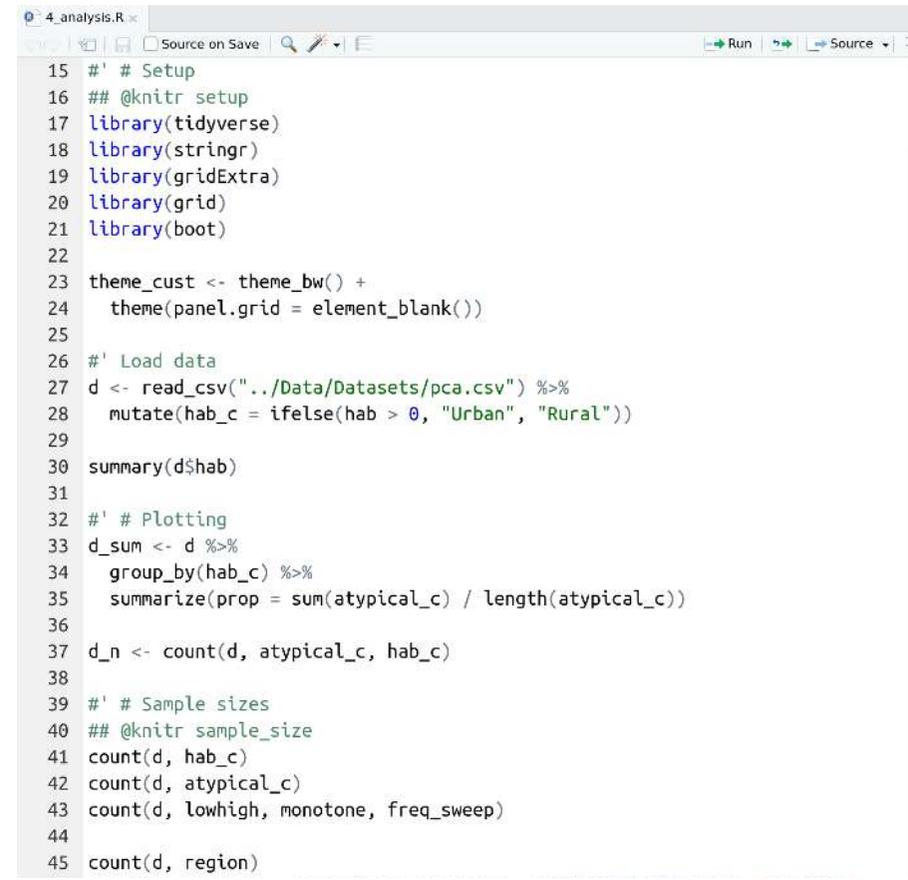
- The result of running code or a script

Script

- A text file full of code that you want to run
- You should always keep your code in a script

For example:

<pre>mean(c(1, 2, 3, 4))</pre>	Code
<pre>[1] 2.5</pre>	Output
	Script



```
15 #' # Setup
16 ## @knitr setup
17 library(tidyverse)
18 library(stringr)
19 library(gridExtra)
20 library(grid)
21 library(boot)
22
23 theme_cust <- theme_bw() +
24   theme(panel.grid = element_blank())
25
26 #' Load data
27 d <- read_csv("../Data/Datasets/pca.csv") %>%
28   mutate(hab_c = ifelse(hab > 0, "Urban", "Rural"))
29
30 summary(d$hab)
31
32 #' # Plotting
33 d_sum <- d %>%
34   group_by(hab_c) %>%
35   summarize(prop = sum(atypical_c) / length(atypical_c))
36
37 d_n <- count(d, atypical_c, hab_c)
38
39 #' # Sample sizes
40 ## @knitr sample_size
41 count(d, hab_c)
42 count(d, atypical_c)
43 count(d, lowhigh, monotone, freq_sweep)
44
45 count(d, region)
```

RStudio vs. R



RStudio



R

- **RStudio** is not R
- RStudio is a User Interface or IDE (integrated development environment)
 - (i.e., Makes coding simpler)

functions() - Do things, Return things

`mean()`, `read_csv()`, `ggplot()`, `c()`, etc.

- Always have `()`
- Can take **arguments** (think 'options')
 - `mean(x = c(2, 10, 45))`,
 - `mean(x = c(NA, 10, 2, 65), na.rm = TRUE)`
- Arguments defined by **name** or by **position**
 - With correct position, do not need to specify by name

By name:

```
mean(x = c(1, 5, 10))
```

```
[1] 5.333333
```

By position:

```
mean(c(1, 5, 10))
```

```
[1] 5.333333
```

R documentation

mean {base}

R Documentation

Arithmetic Mean

Description

Generic function for the (trimmed) arithmetic mean.

Usage

```
mean(x, ...)
```

```
## Default S3 method:
```

```
mean(x, trim = 0, na.rm = FALSE, ...)
```

Arguments

- `x` An R object. Currently there are methods for numeric/logical vectors and [date](#), [date-time](#) and [time interval](#) objects. Complex vectors are allowed for `trim = 0`, only.
- `trim` the fraction (0 to 0.5) of observations to be trimmed from each end of `x` before the mean is computed. Values of `trim` outside that range are taken as the nearest endpoint.
- `na.rm` a logical value indicating whether NA values should be stripped before the computation proceeds.
- `...` further arguments passed to or from other methods.

Data

Generally kept in `vectors` or `data.frames`

- These are objects with names (like functions)
- We can use `<-` to assign values to objects (assignment)

Vector (1 dimension)

```
my_data <- c("a", 100, "c")  
my_data
```

```
[1] "a"  "100" "c"
```

Data frame (2 dimensions)

```
my_data <- data.frame(site = c("s1", "s2", "s3"),  
                      count = c(101, 102, 103),  
                      treatment = c("a", "b", "c"))
```

```
my_data
```

	site	count	treatment
1	s1	101	a
2	s2	102	b
3	s3	103	c

rows x columns

Your first *real* code!

First Code

```
1 # First load the packages
2 library(palmerpenguins)
3 library(ggplot2)
4
5 # Now create the figure
6 ggplot(data = penguins, aes(x = body_mass_g, y = bill_length_mm, colour = species)) +
7   geom_point()
```

1. Copy/paste or type this into the **script** window in RStudio

- You may have to go to File > New File > R Script

2. Click on the **first line of code**

3. Run the code

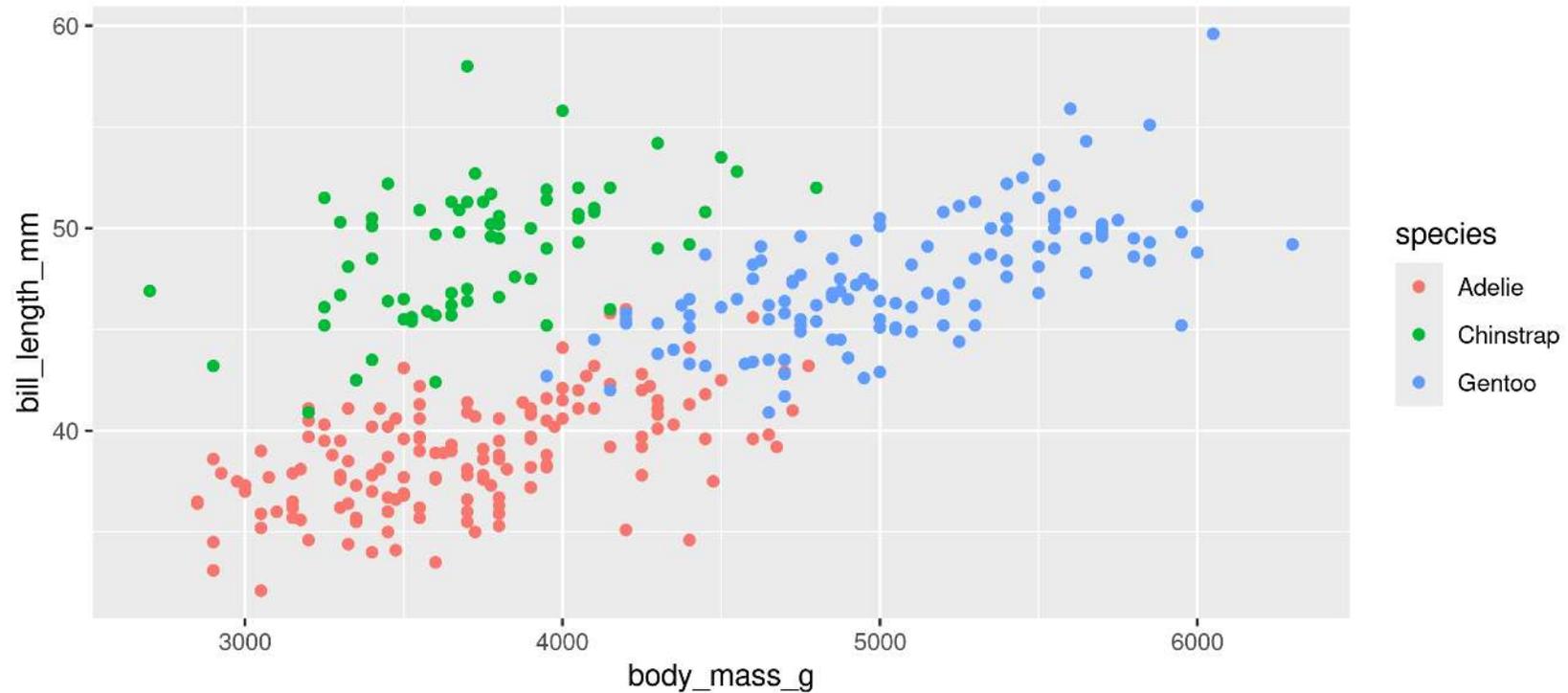
- Click 'Run' button (upper right) **or**
- Use the short-cut **Ctrl-Enter**

4. Repeat until all the code has run

First Code

```
1 # First load the packages
2 library(palmerpenguins)
3 library(ggplot2)
4
5 # Now create the figure
6 ggplot(data = penguins, aes(x = body_mass_g, y = bill_length_mm, colour = species)) +
7   geom_point()
```

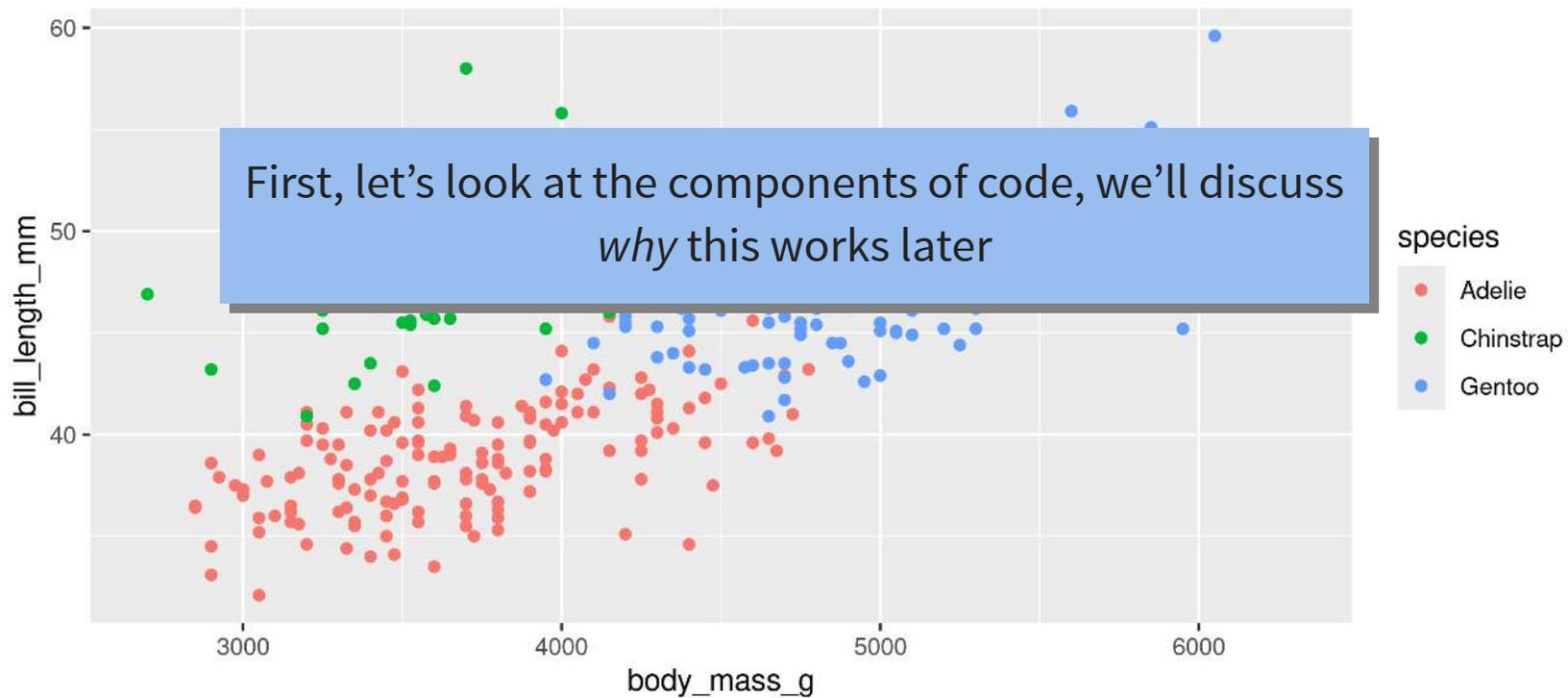
Warning: Removed 2 rows containing missing values or values outside the scale range
(`geom_point()`).



First Code

```
1 # First load the packages
2 library(palmerpenguins)
3 library(ggplot2)
4
5 # Now create the figure
6 ggplot(data = penguins, aes(x = body_mass_g, y = bill_length_mm, colour = species)) +
7   geom_point()
```

Warning: Removed 2 rows containing missing values or values outside the scale range (`geom_point()`).

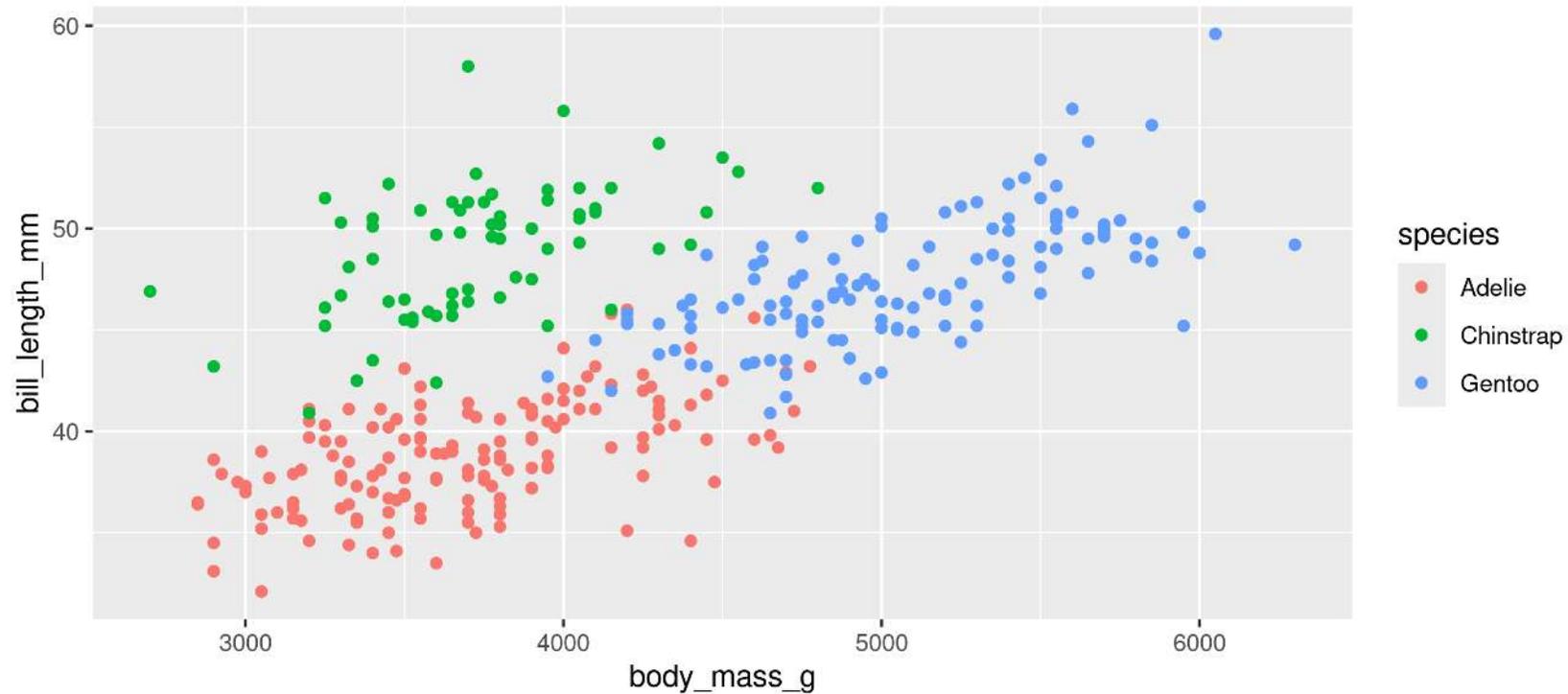


First Code

Packages ggplot2 and palmerpenguins

```
1 # First load the p
2 library(palmerpeng
3 library(ggplot2)
4
5 # Now create the figure
6 ggplot(data = penguins, aes(x = body_mass_g, y = bill_length_mm, colour = species)) +
7   geom_point()
```

Warning: Removed 2 rows containing missing values or values outside the scale range (`geom_point()`).



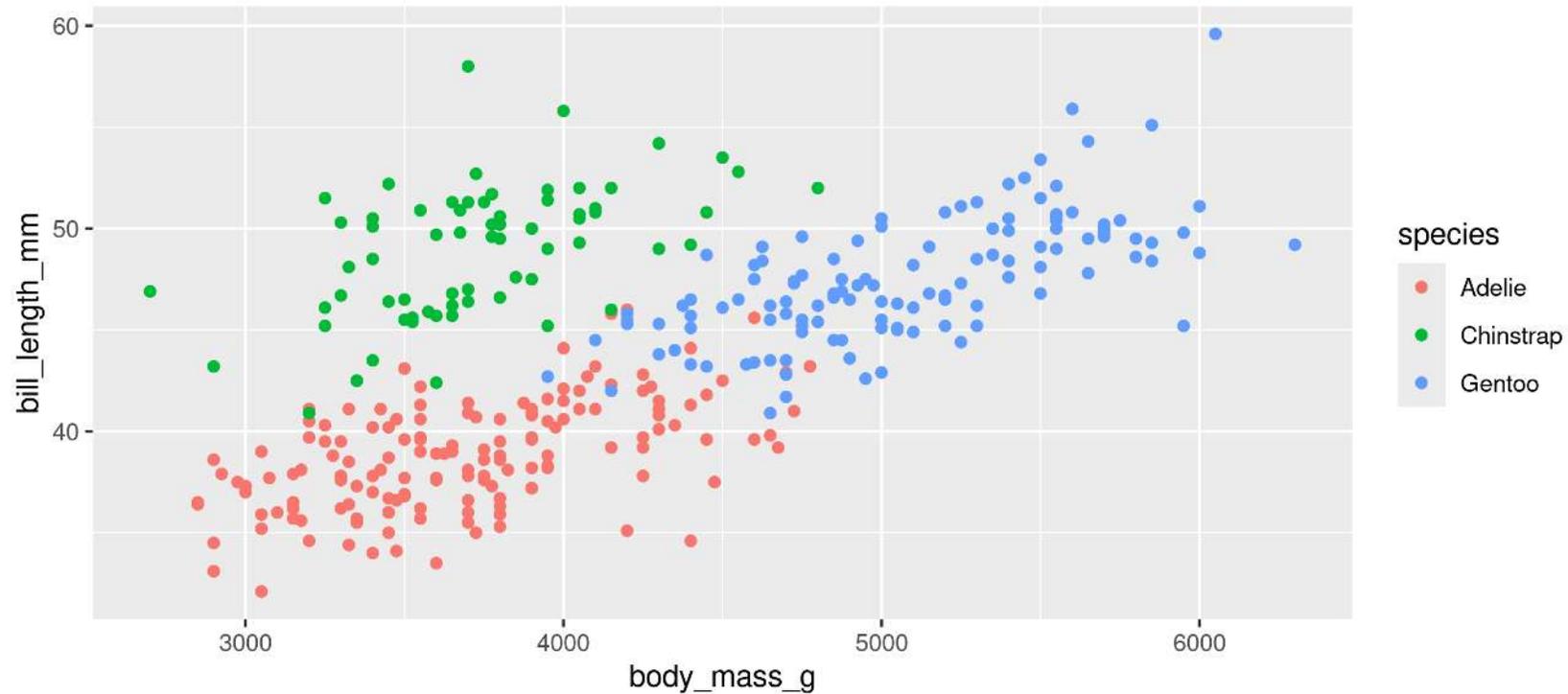
First Code

Functions

```
library(), ggplot(), aes(), geom_point()
```

```
1 # First load the p
2 library(palmerpeng
3 library(ggplot2)
4
5 # Now create the figure
6 ggplot(data = penguins, aes(x = body_mass_g, y = bill_length_mm, colour = species)) +
7   geom_point()
```

Warning: Removed 2 rows containing missing values or values outside the scale range (`geom_point()`).

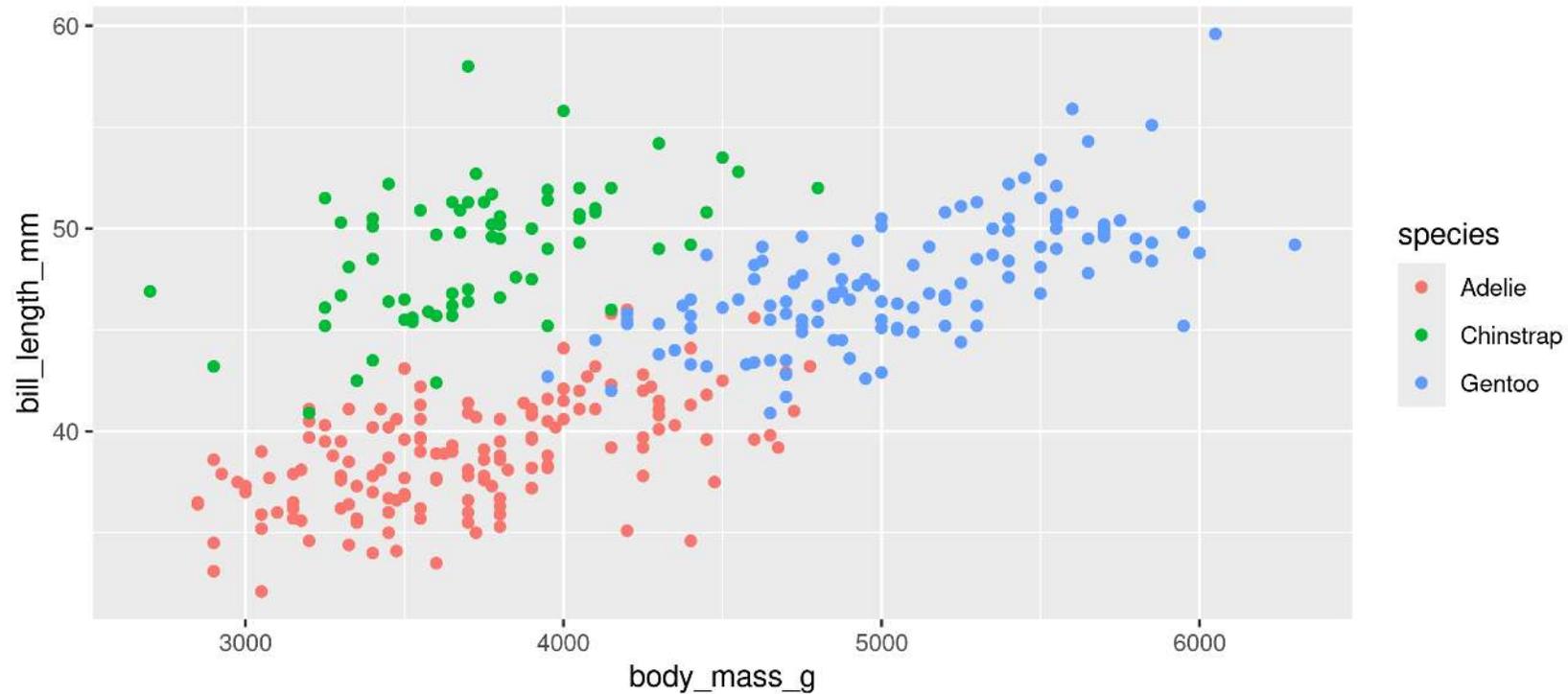


First Code

```
1 # First load the p
2 library(palmerpeng
3 library(ggplot2)
4
5 # Now create the figure
6 ggplot(data = penguins, aes(x = body_mass_g, y = bill_length_mm, colour = species)) +
7   geom_point()
```

+
(Specific to ggplot)

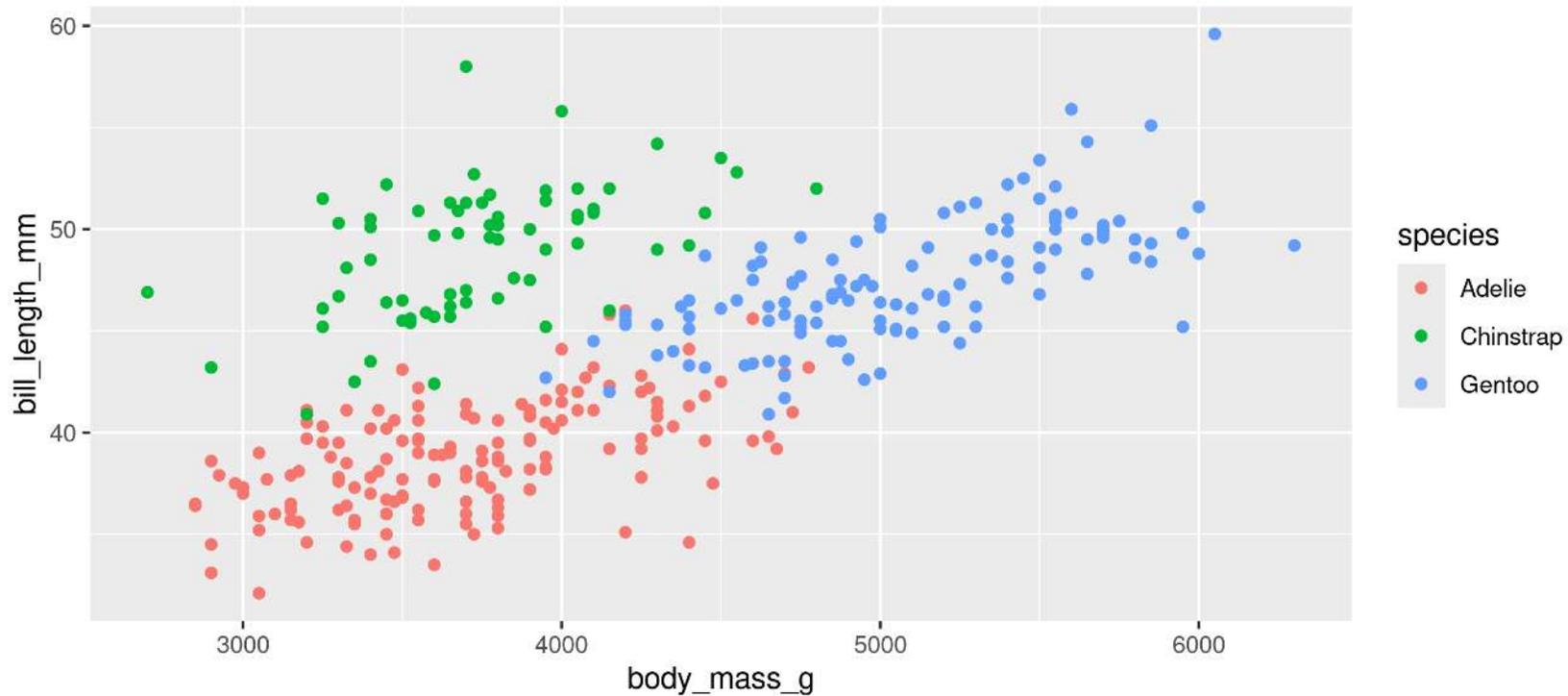
Warning: Removed 2 rows containing missing values or values outside the scale range
(`geom_point()`).



First Code

```
1 # First load the packages
2 library(palmerpenguins)
3 library(ggplot2)
4
5 # Now create the figure
6 ggplot(data = penguins, aes(x = body_mass_g, y = bill_length_mm, colour = species)) +
7   geom_point()
```

Warning: Removed 2 rows containing missing values or values outside the scale range (`geom_point()`).



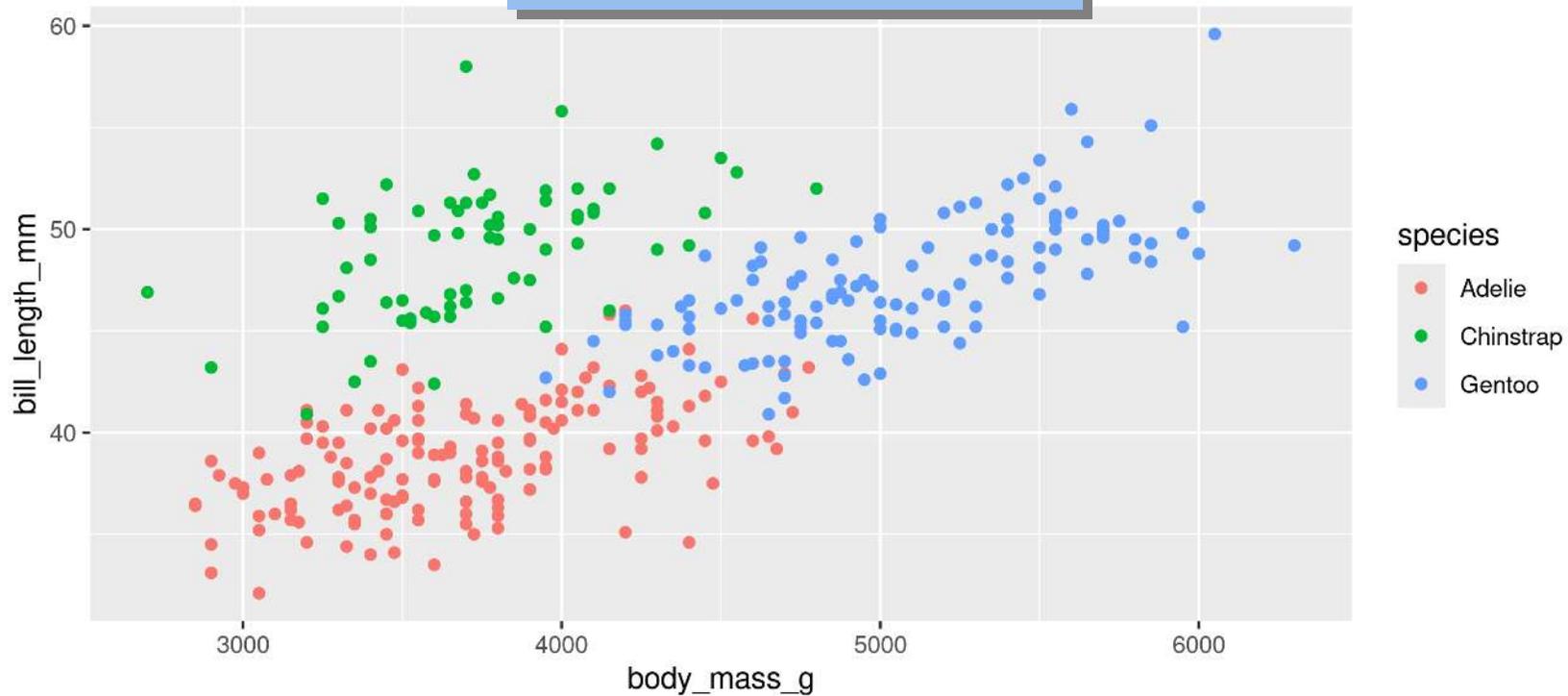
Figure!

First Code

```
1 # First load the packages
2 library(palmerpenguins)
3 library(ggplot2)
4
5 # Now create the figure
6 ggplot(data = penguins, aes(x = body_mass_g, y = bill_length_mm, colour = species)) +
7   geom_point()
```

Warning: Removed 2 rows containing missing values or values outside the scale range (`geom_point()`).

Warning

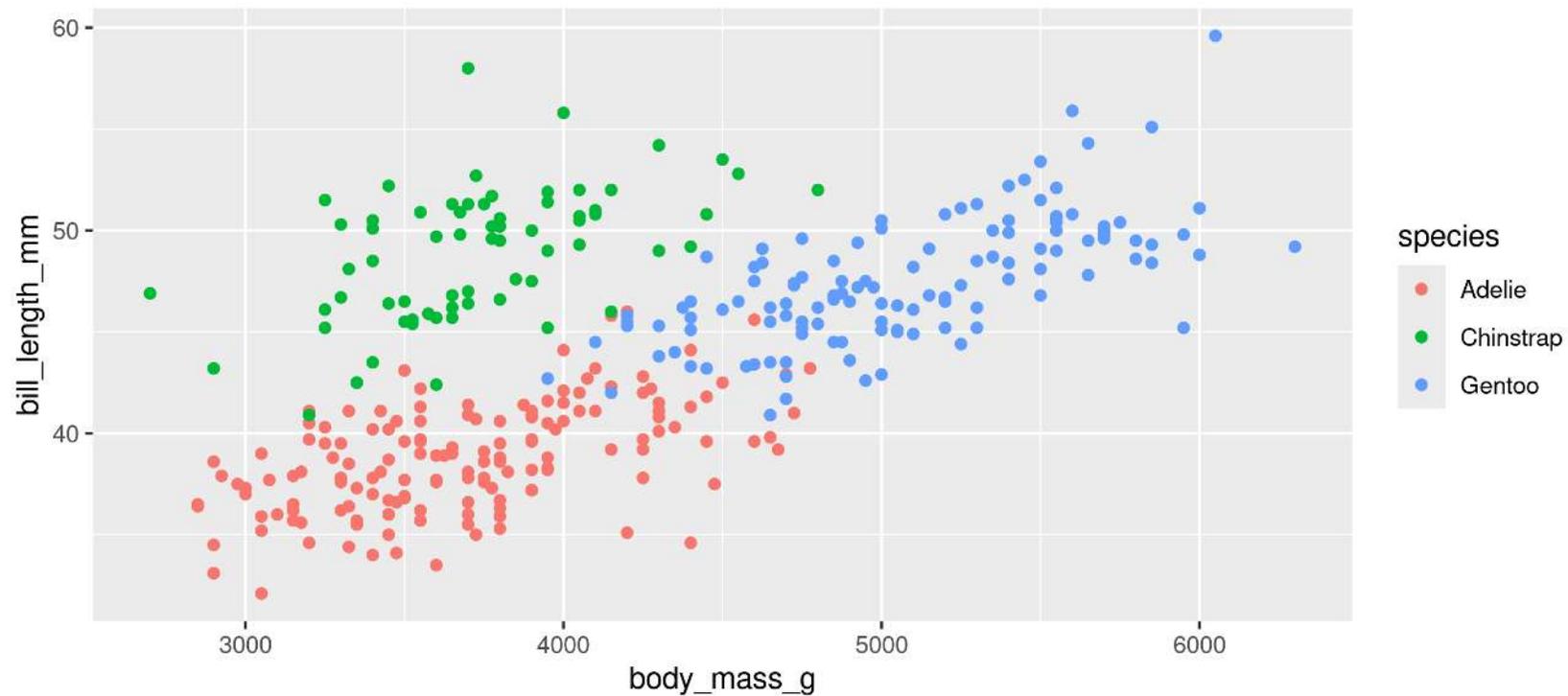


First Code

```
1 # First load the packages
2 library(palmerpenguins)
3 library(ggplot2)
4
5 # Now create the figure
6 ggplot(data = penguins, aes(x = body_mass_g, y = bill_length_mm, colour = species)) +
7   geom_point()
```

Comments

Warning: Removed 2 rows containing missing values or values outside the scale range
(`geom_point()`).



Now you have the vocabulary!

Let's get started

Our data set: Palmer Penguins!



CHINSTRAP!



GENTOO!



ADÉLIE!



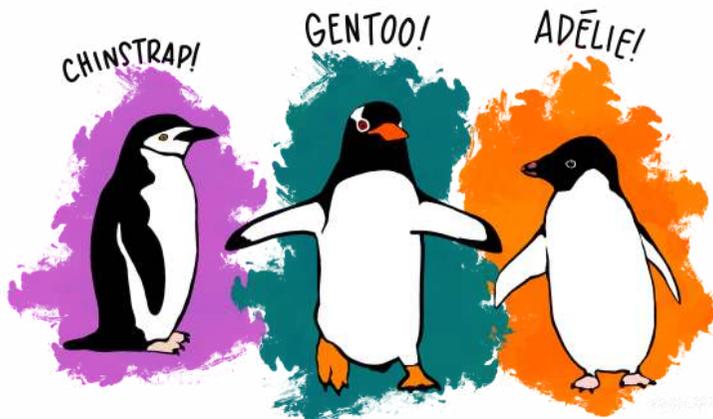
@allison_horst

Our data set: Palmer Penguins!



```
1 library(palmerpenguins)
2 penguins

# A tibble: 344 × 8
  species island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex year
  <fct>   <fct>         <dbl>         <dbl>         <int>         <int> <fct> <int>
1 Adelie Torgersen         39.1           18.7           181           3750 male  2007
2 Adelie Torgersen         39.5           17.4           186           3800 female 2007
3 Adelie Torgersen         40.3            18            195           3250 female 2007
4 Adelie Torgersen         NA              NA              NA              NA <NA>  2007
5 Adelie Torgersen         36.7           19.3           193           3450 female 2007
6 Adelie Torgersen         39.3           20.6           190           3650 male  2007
7 Adelie Torgersen         38.9           17.8           181           3625 female 2007
8 Adelie Torgersen         39.2           19.6           195           4675 male  2007
9 Adelie Torgersen         34.1           18.1           193           3475 <NA>  2007
10 Adelie Torgersen         42             20.2           190           4250 <NA>  2007
# i 334 more rows
```

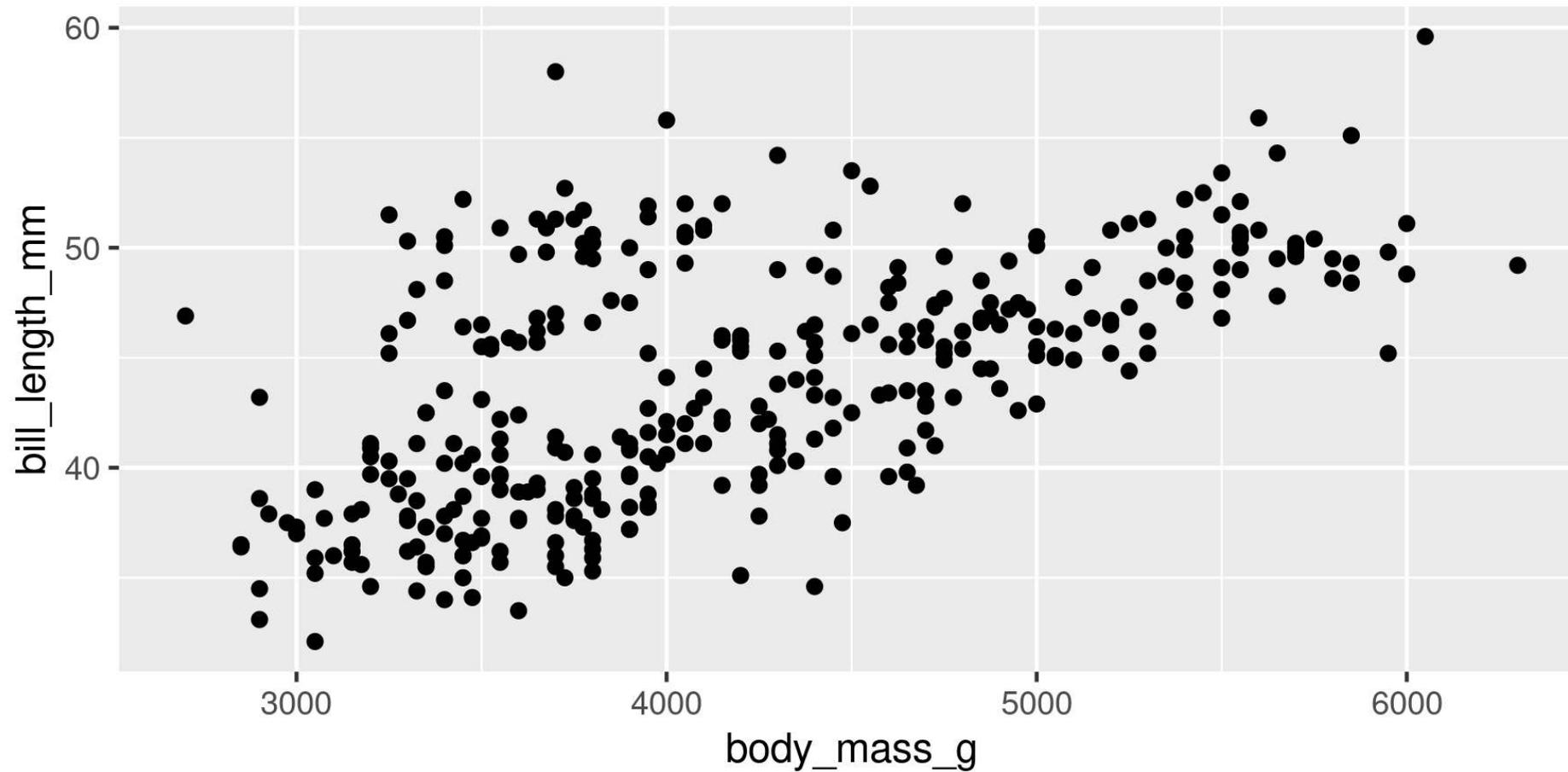


Your turn!

Run this code and look at the output in the console

A basic plot

```
1 library(palmerpenguins)
2 library(ggplot2)
3
4 ggplot(data = penguins, aes(x = body_mass_g, y = bill_length_mm)) +
5   geom_point()
```



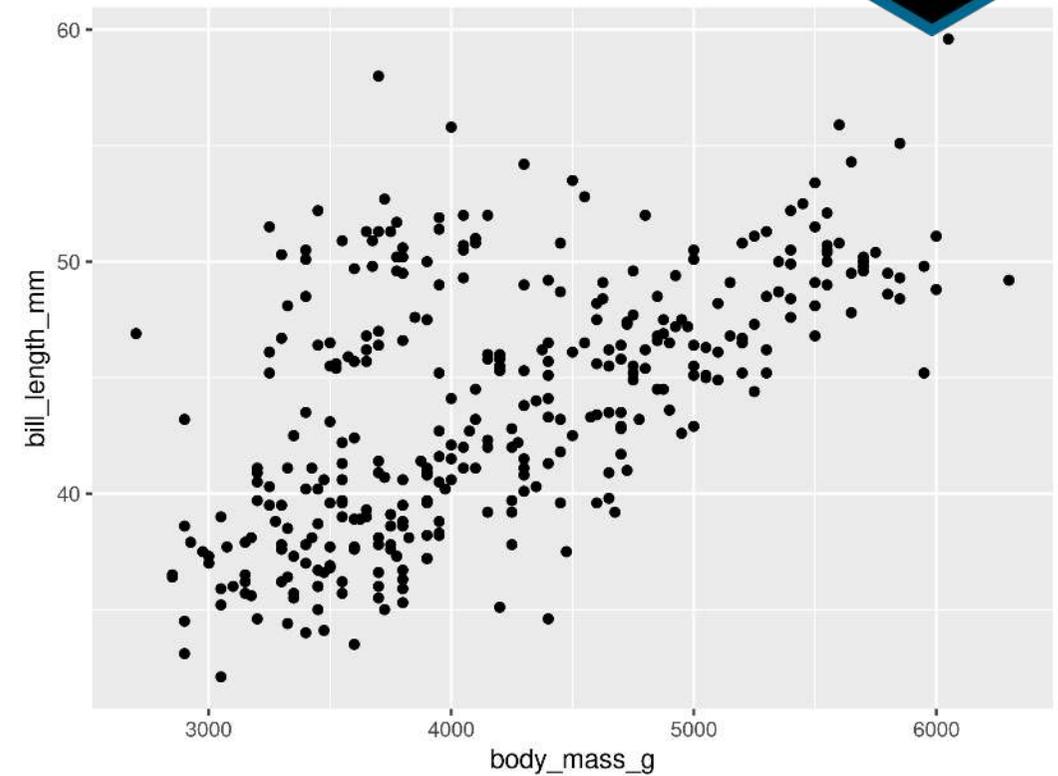
Break it down

```
1 library(palmerpenguins)
2 library(ggplot2)
3
4 ggplot(data = penguins, aes(x = body_mass_g, y = bill_length_mm)) +
5   geom_point()
```



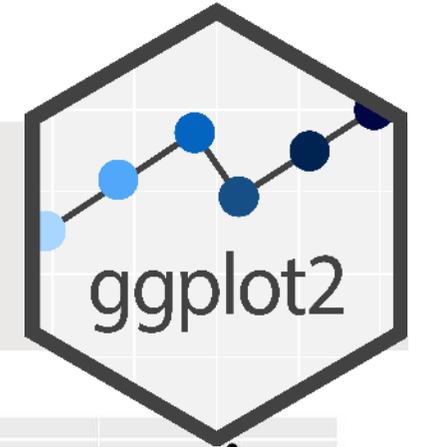
library(palmerpenguins)

- Load the `palmerpenguins` package
- Now we have access to `penguins` data



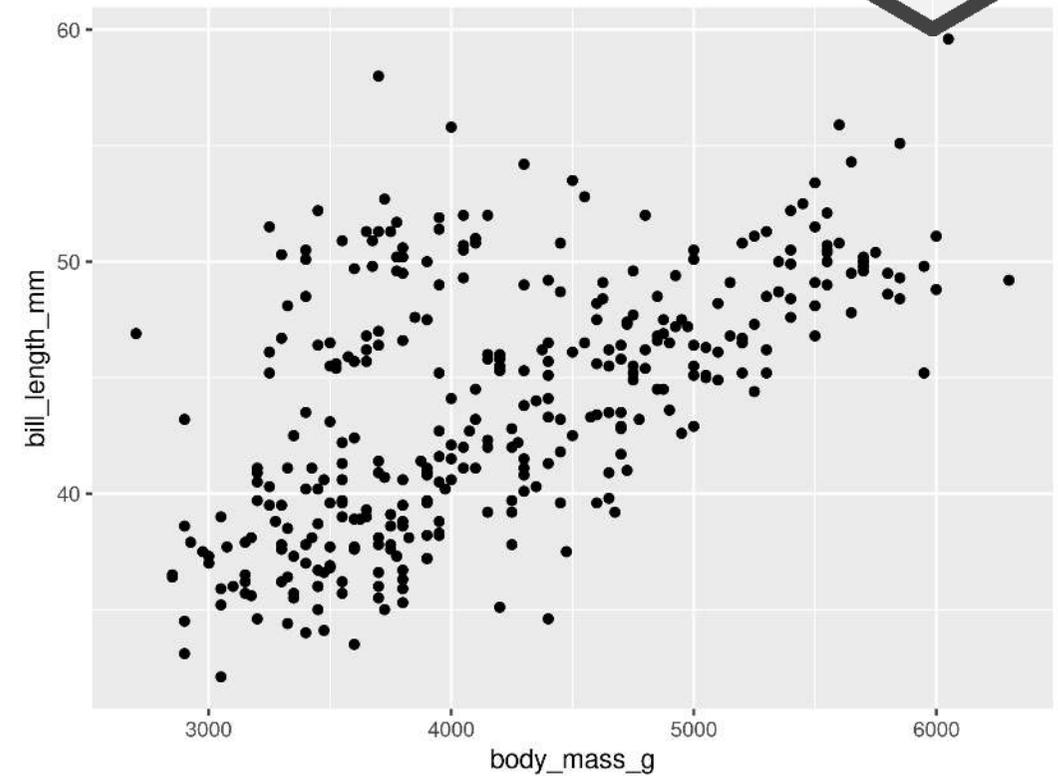
Break it down

```
1 library(palmerpenguins)
2 library(ggplot2)
3
4 ggplot(data = penguins, aes(x = body_mass_g, y = bill_length_mm)) +
5   geom_point()
```



library(ggplot2)

- Load the `ggplot2` package
- Now we have access to the `ggplot()` function
 - (and `aes()` and `geom_point()` etc.)

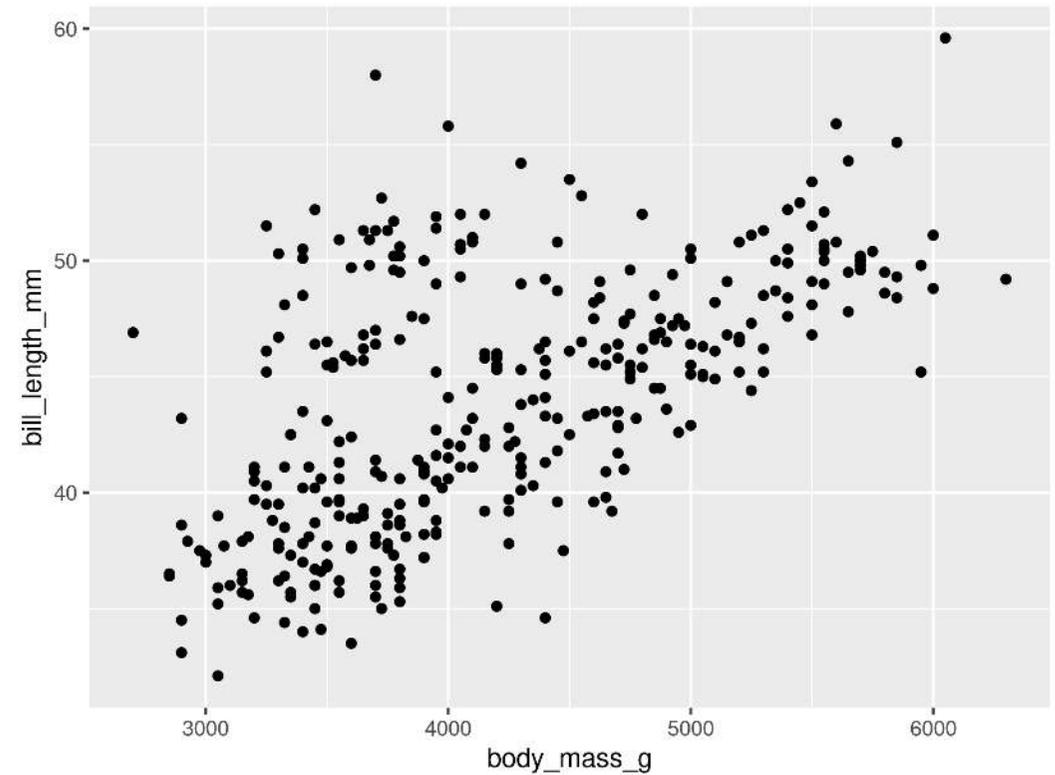


Break it down

```
1 library(palmerpenguins)
2 library(ggplot2)
3
4 ggplot(data = penguins, aes(x = body_mass_g, y = bill_length_mm)) +
5   geom_point()
```

ggplot()

- Set the attributes of your plot
- **data** = Dataset
- **aes** = Aesthetics (how the data are used)
- Think of this as your plot defaults



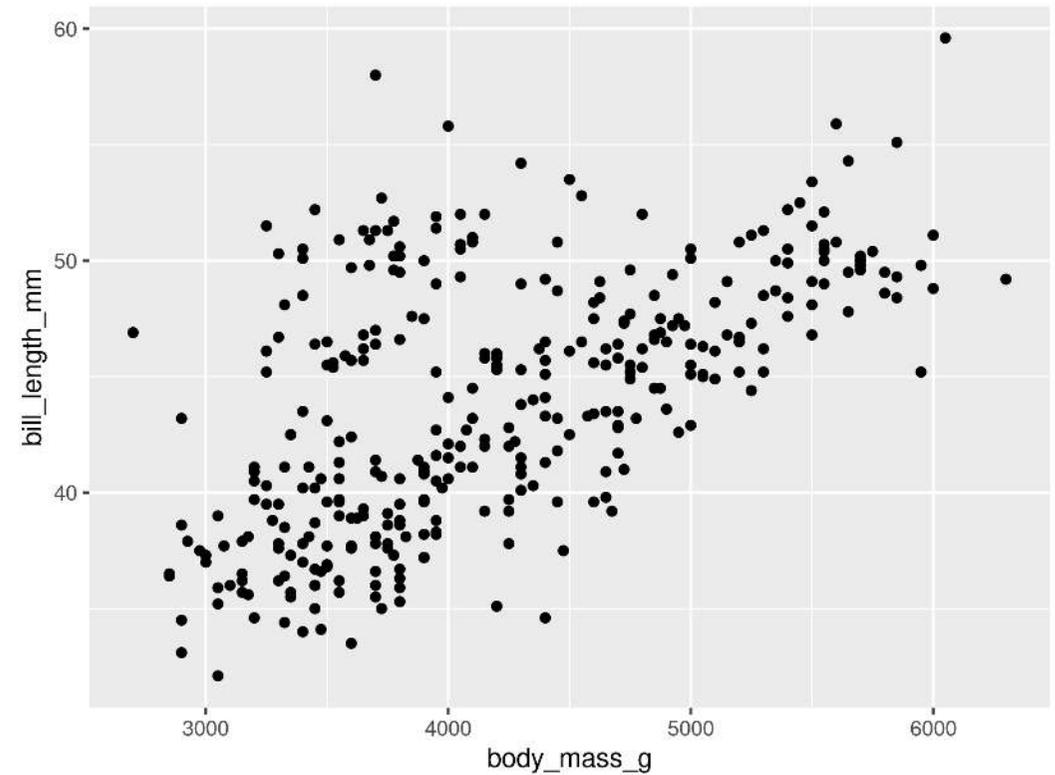
Break it down

```
1 library(palmerpenguins)
2 library(ggplot2)
3
4 ggplot(data = penguins, aes(x = body_mass_g, y = bill_length_mm)) +
5   geom_point()
```

geom_point()

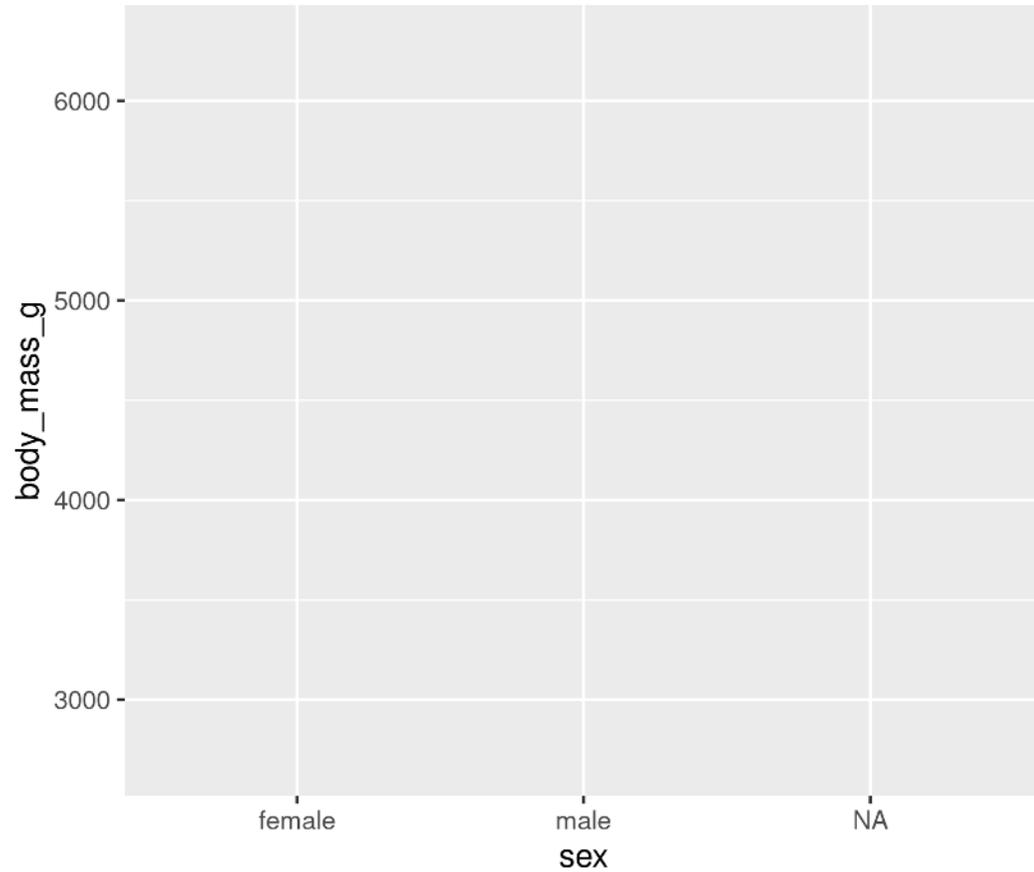
- Choose a `geom` function to display the data
- Always *added* to a `ggplot()` call with `+`

ggplots are essentially layered objects, starting with a call to `ggplot()`

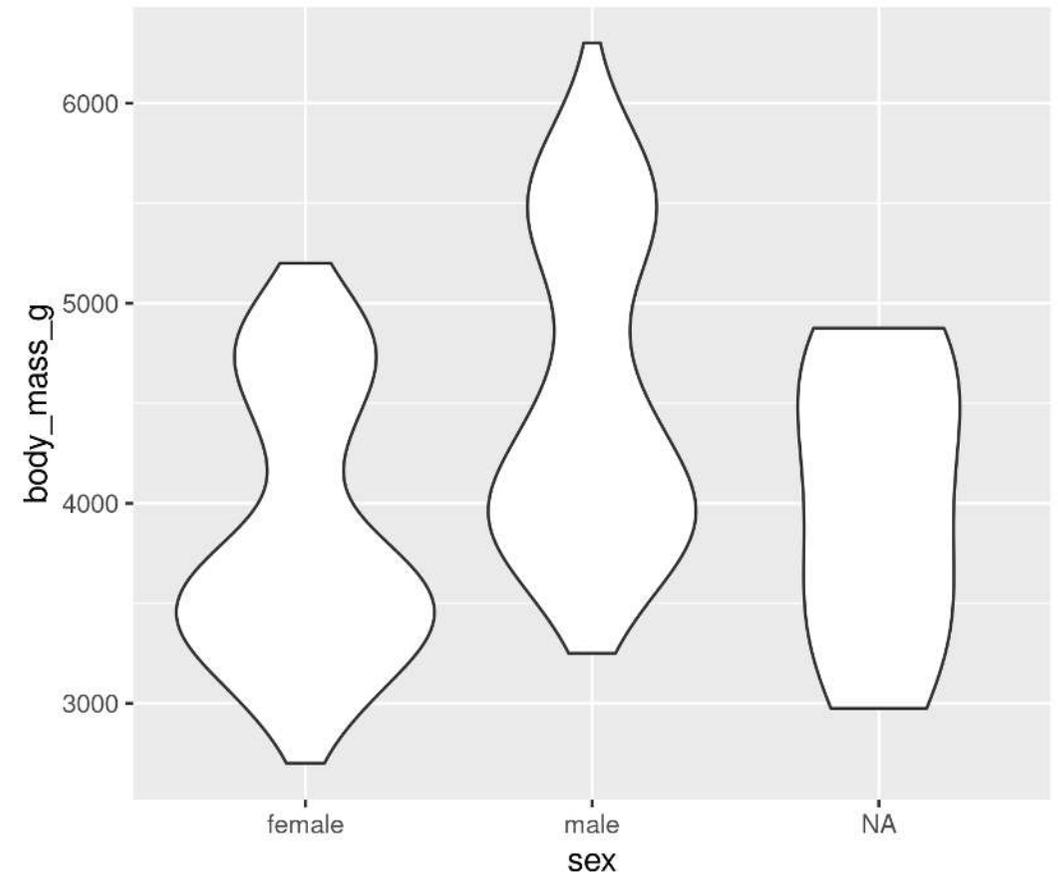


Plots are layered

```
1 ggplot(data = penguins, aes(x = sex, y = body_mass_g))
```



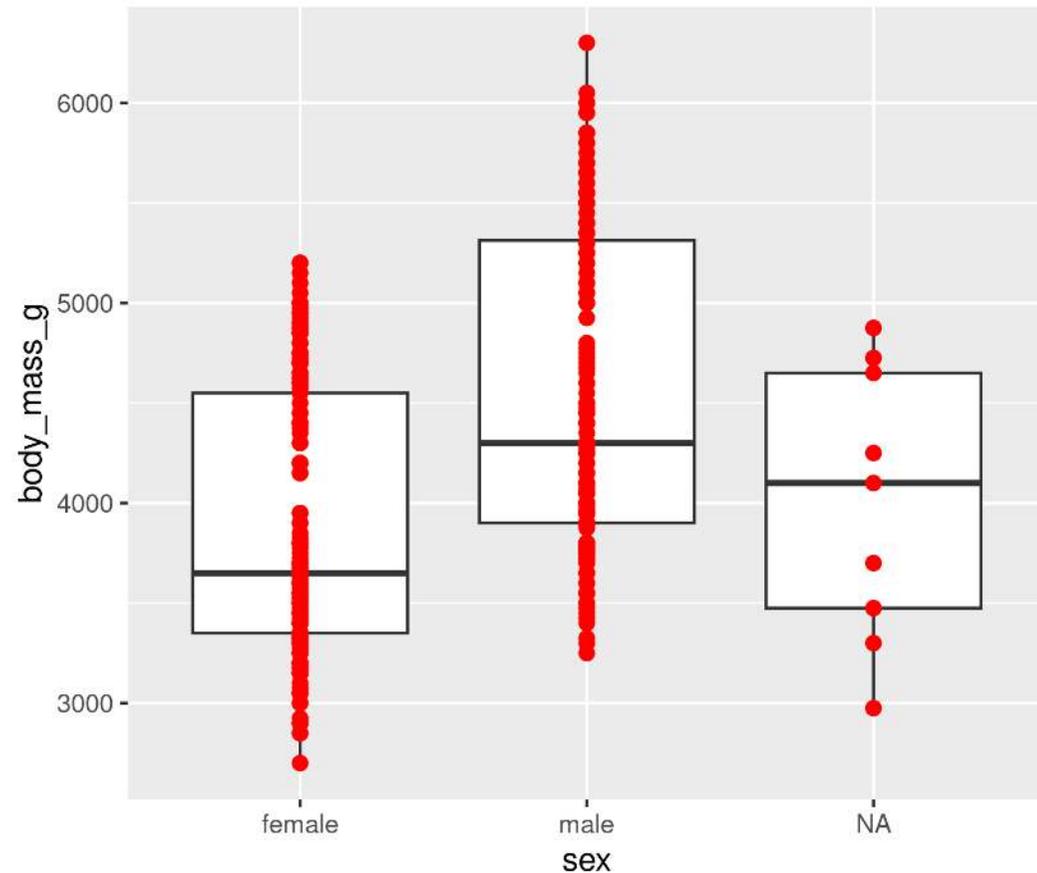
```
1 ggplot(data = penguins, aes(x = sex, y = body_mass_g)) +  
2   geom_violin()
```



Plots are layered

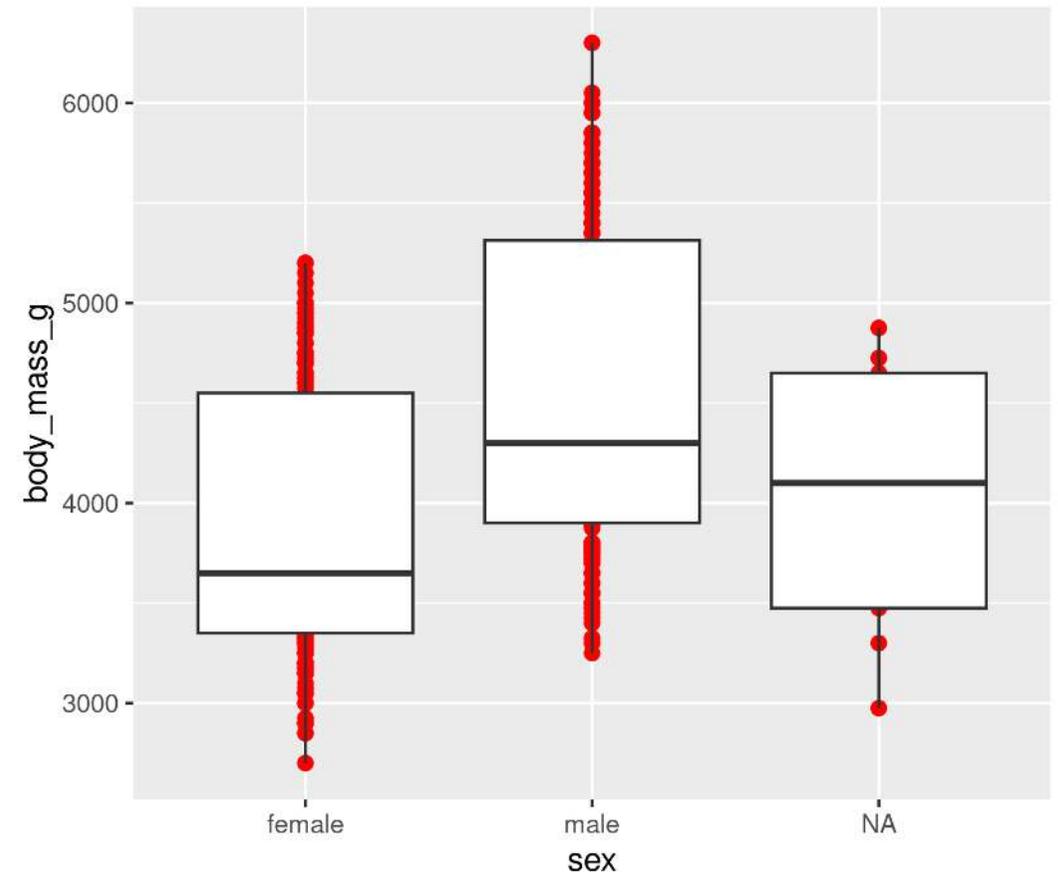
You can add multiple layers

```
1 ggplot(data = penguins, aes(x = sex, y = body_mass_g)) +  
2   geom_boxplot() +  
3   geom_point(size = 2, colour = "red")
```



Order matters

```
1 ggplot(data = penguins, aes(x = sex, y = body_mass_g)) +  
2   geom_point(size = 2, colour = "red") +  
3   geom_boxplot()
```

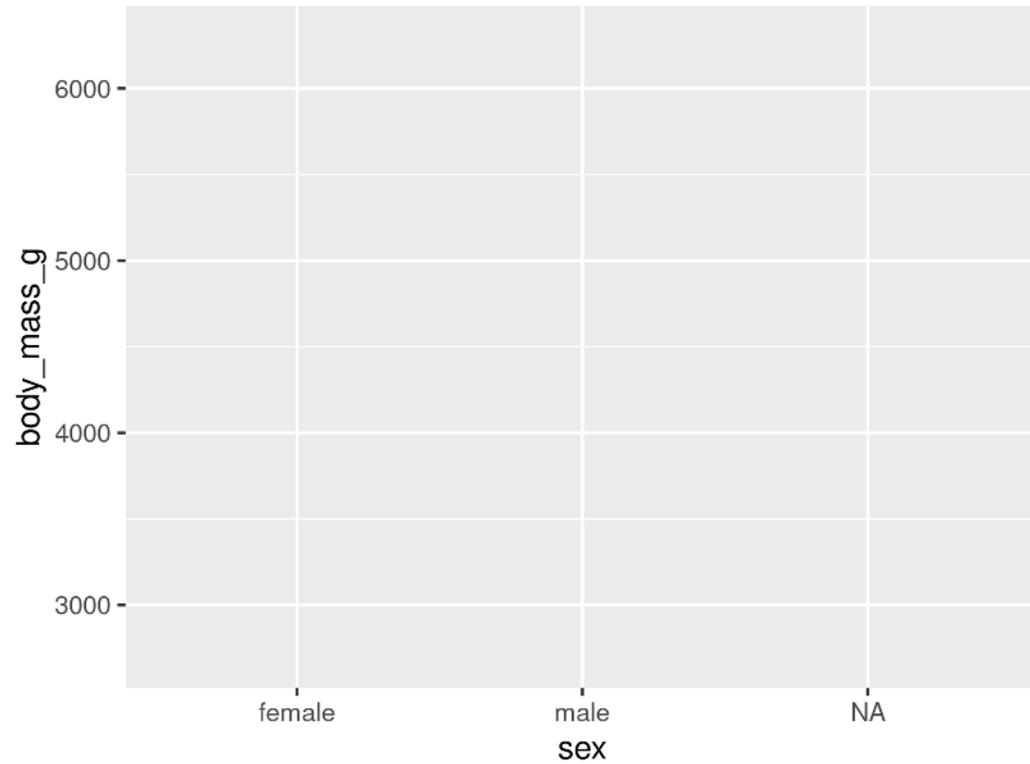


Plots are objects

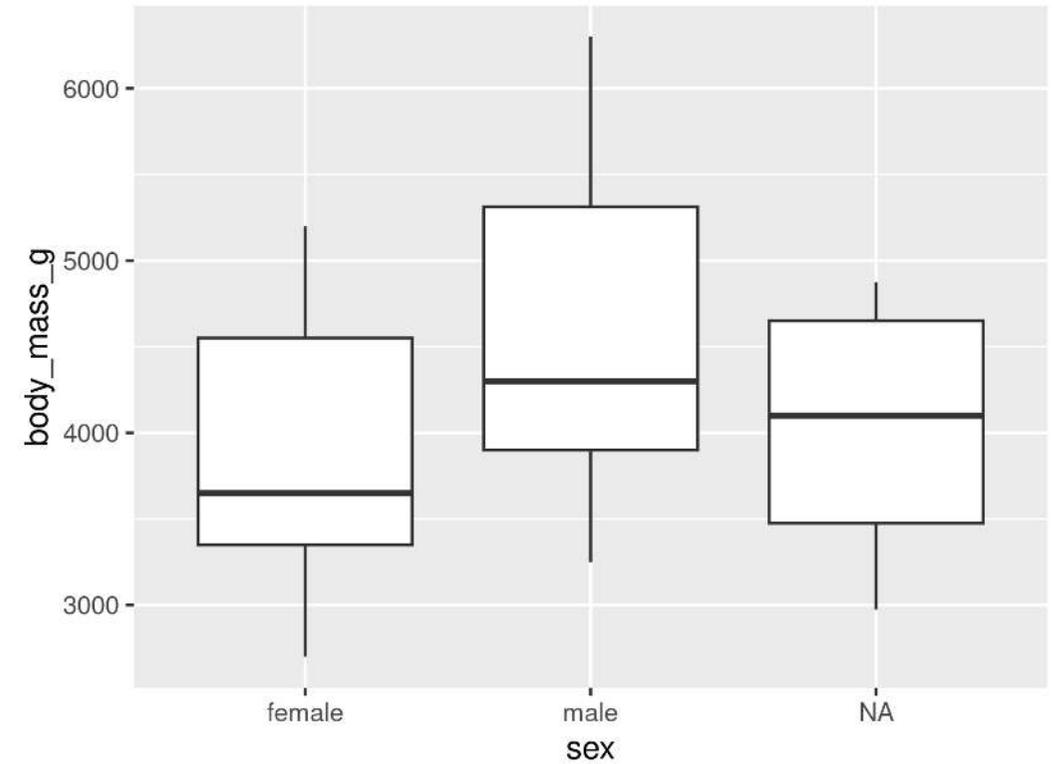
Any ggplot can be saved as an object

```
1 g <- ggplot(data = penguins, aes(x = sex, y = body_mass_g))
```

```
1 g
```



```
1 g + geom_boxplot()
```

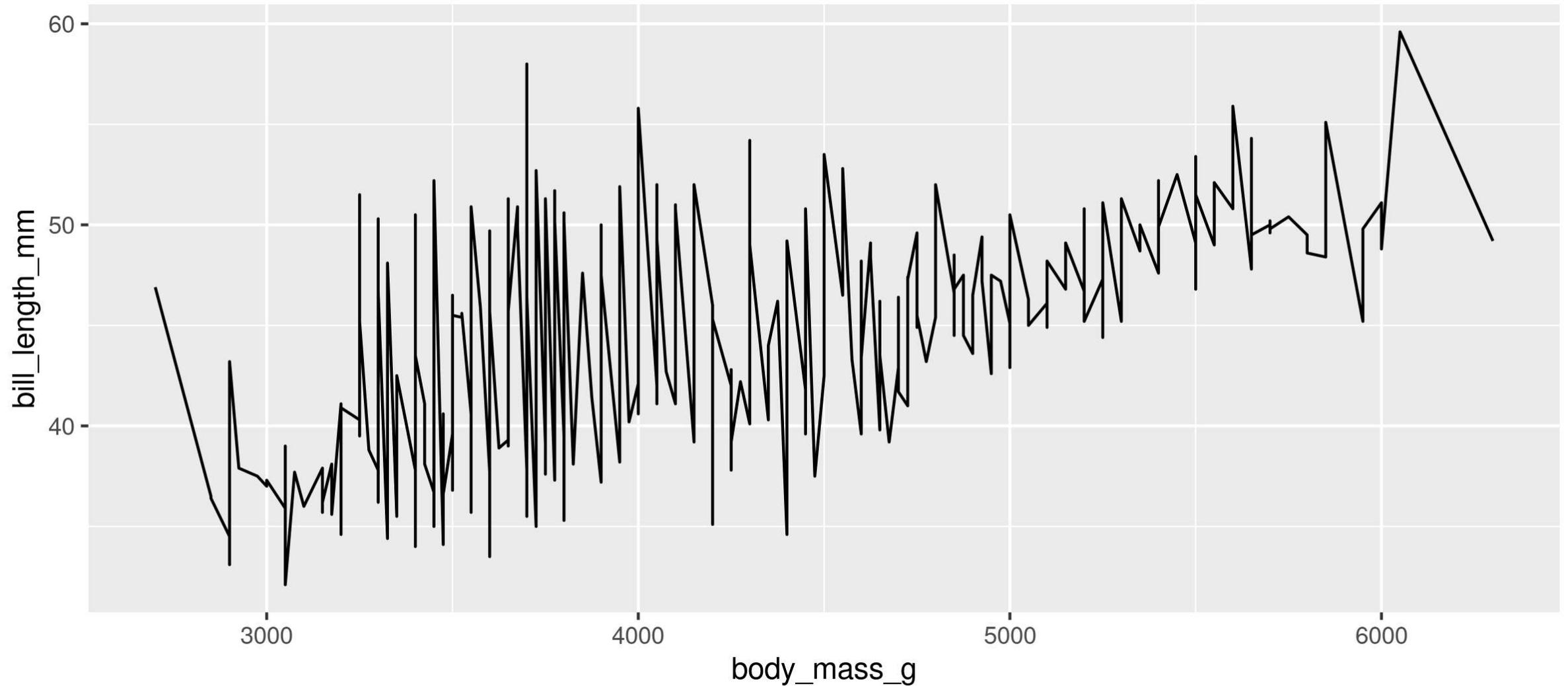


More Geoms

(Plot types)

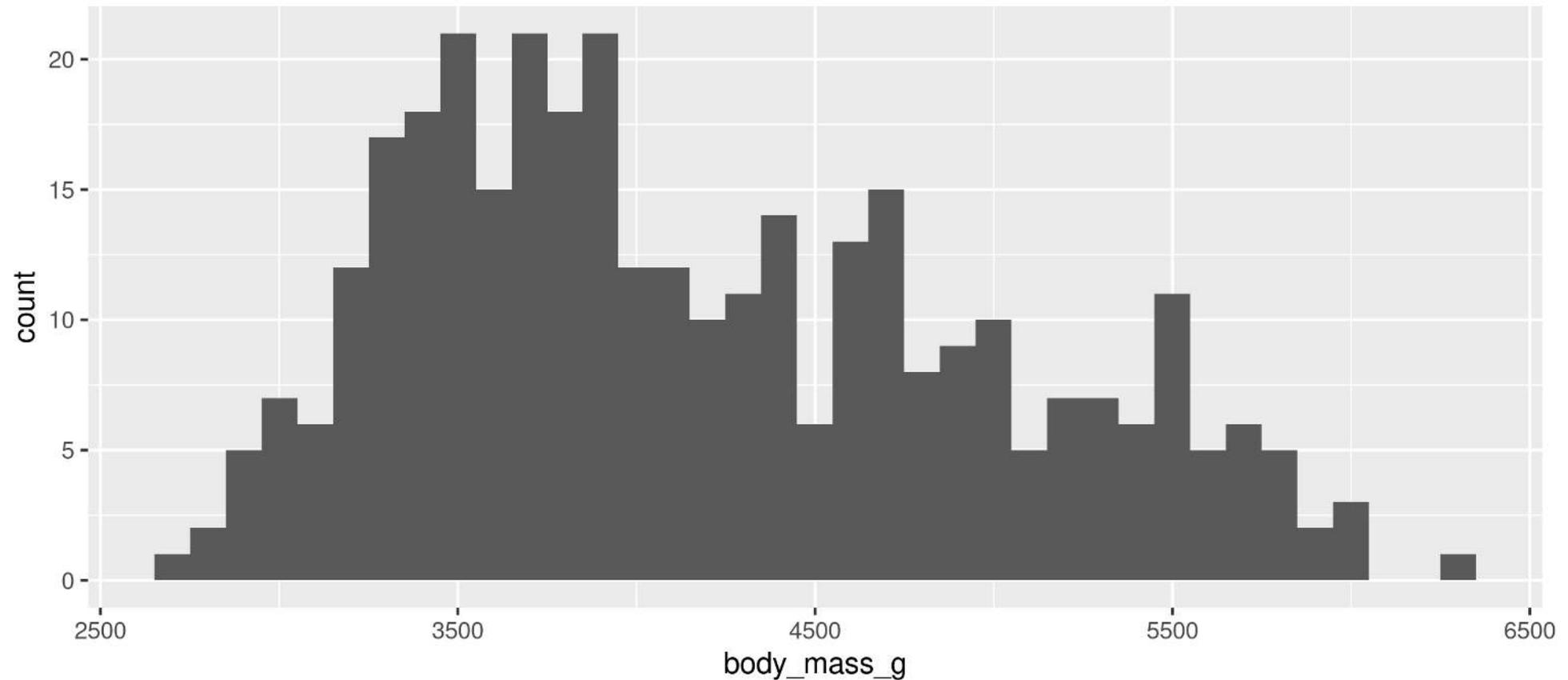
Geoms: Lines

```
1 ggplot(data = penguins, aes(x = body_mass_g, y = bill_length_mm)) +  
2   geom_line()
```



Geoms: Histogram

```
1 ggplot(data = penguins, aes(x = body_mass_g)) +  
2   geom_histogram(binwidth = 100)
```

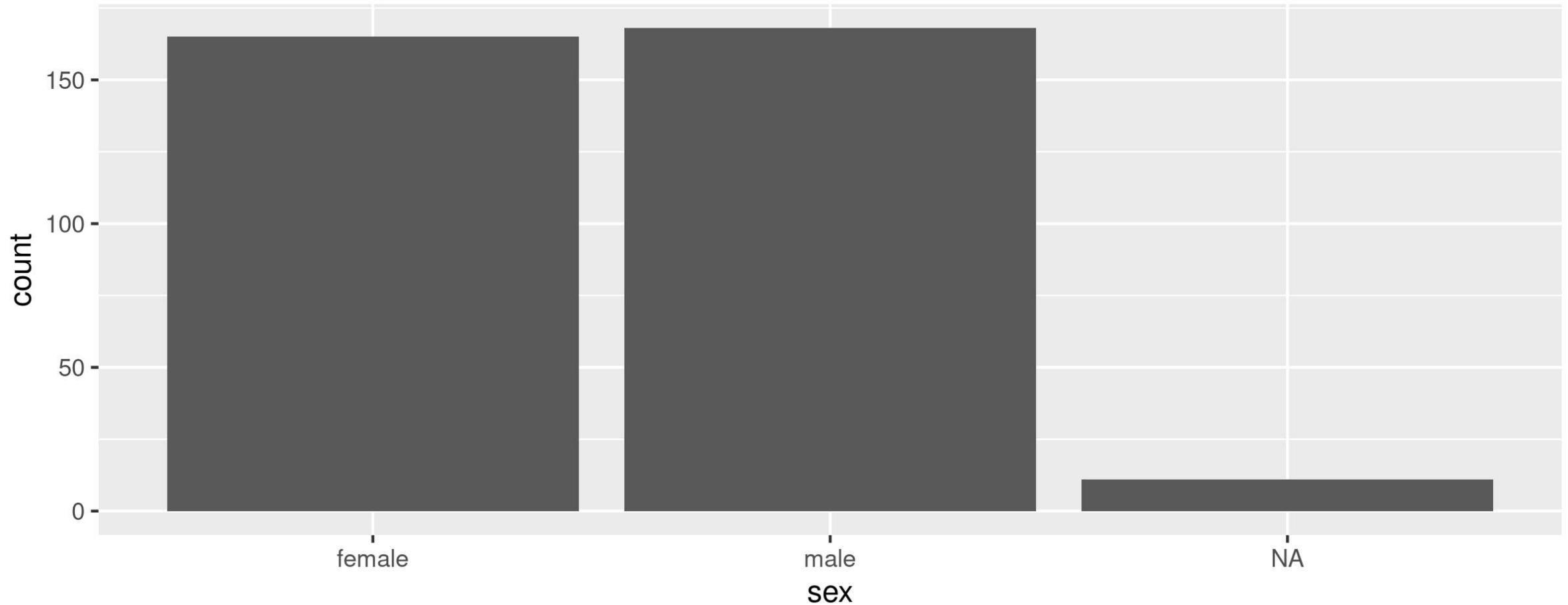


Note: We only need 1 aesthetic here

Geoms: Barplots

Let `ggplot` count your data

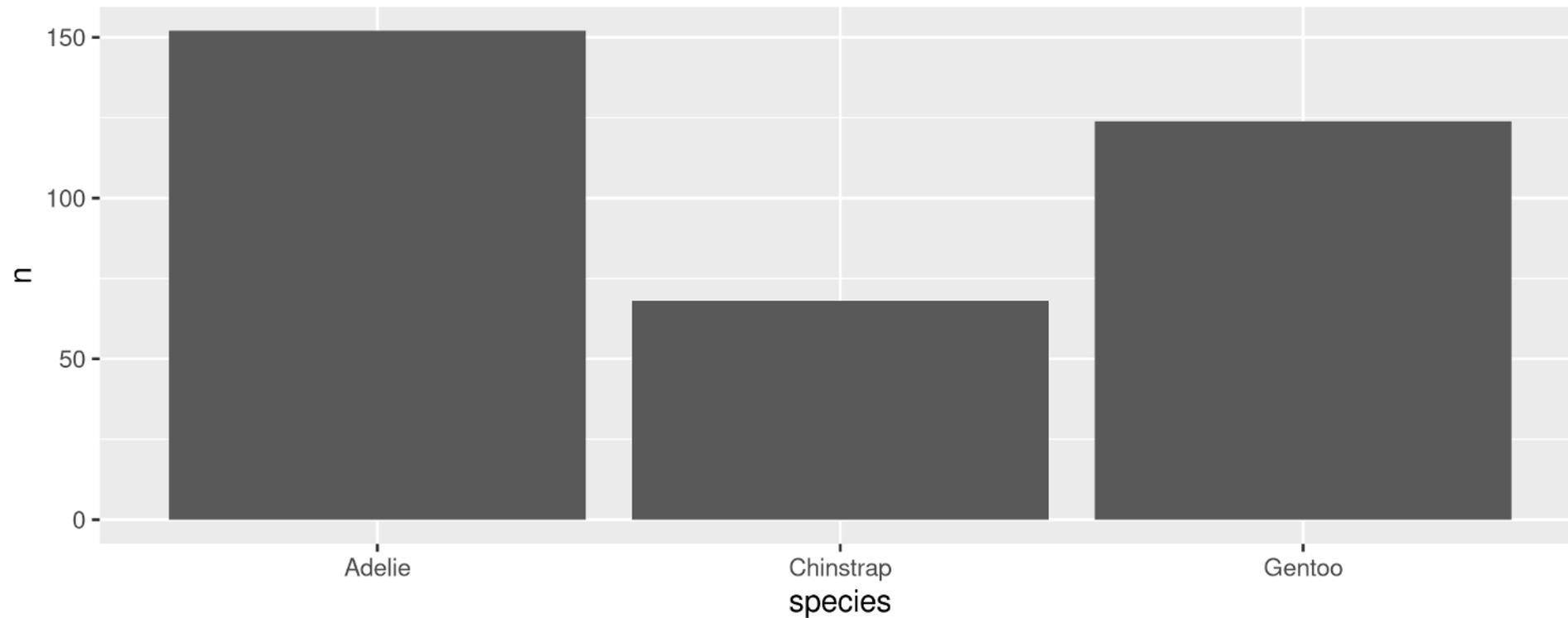
```
1 ggplot(data = penguins, aes(x = sex)) +  
2   geom_bar()
```



Geoms: Barplots

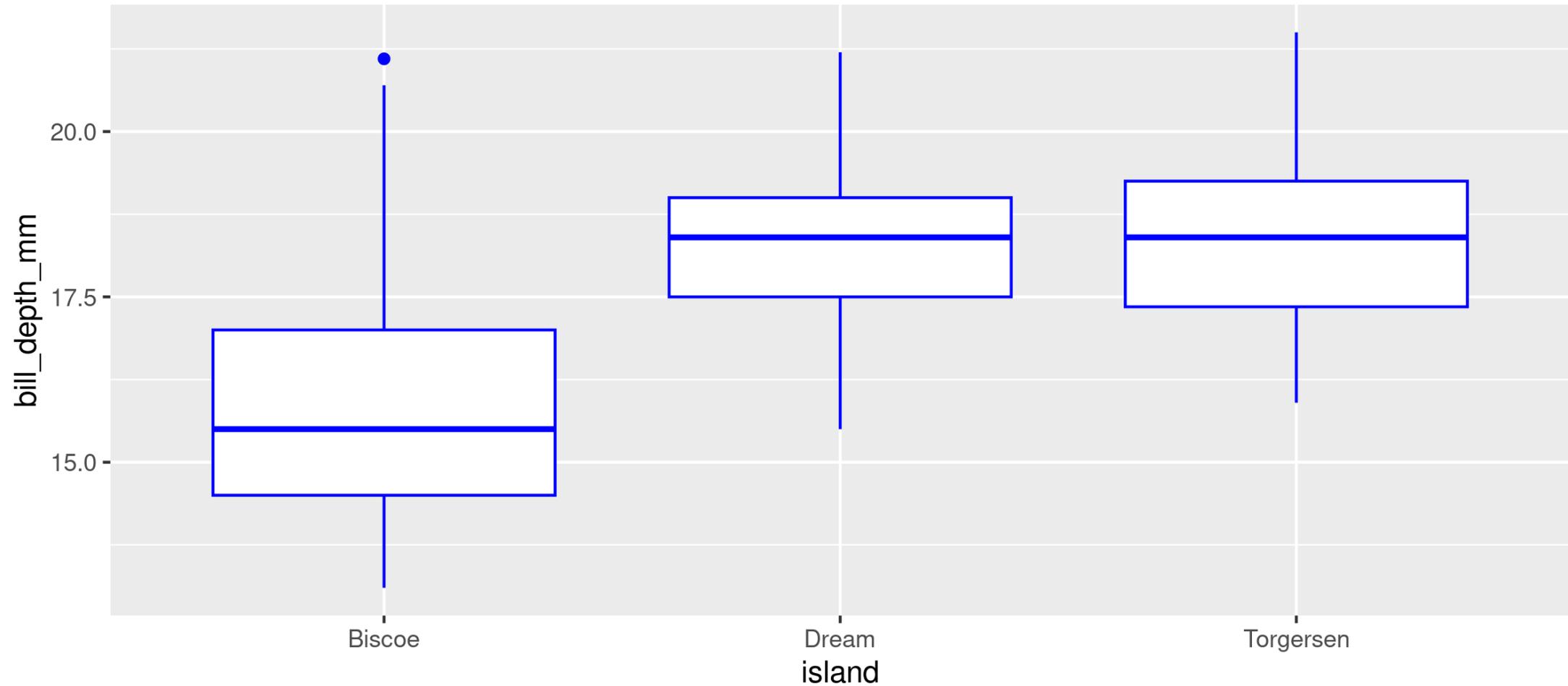
You can also provide the counts

```
1 # Create our own data frame
2 species_counts <- data.frame(species = c("Adelie", "Chinstrap", "Gentoo"),
3                               n = c(152, 68, 124))
4
5 ggplot(data = species_counts, aes(x = species, y = n)) +
6   geom_bar(stat = "identity")
```



Your Turn: Create this plot

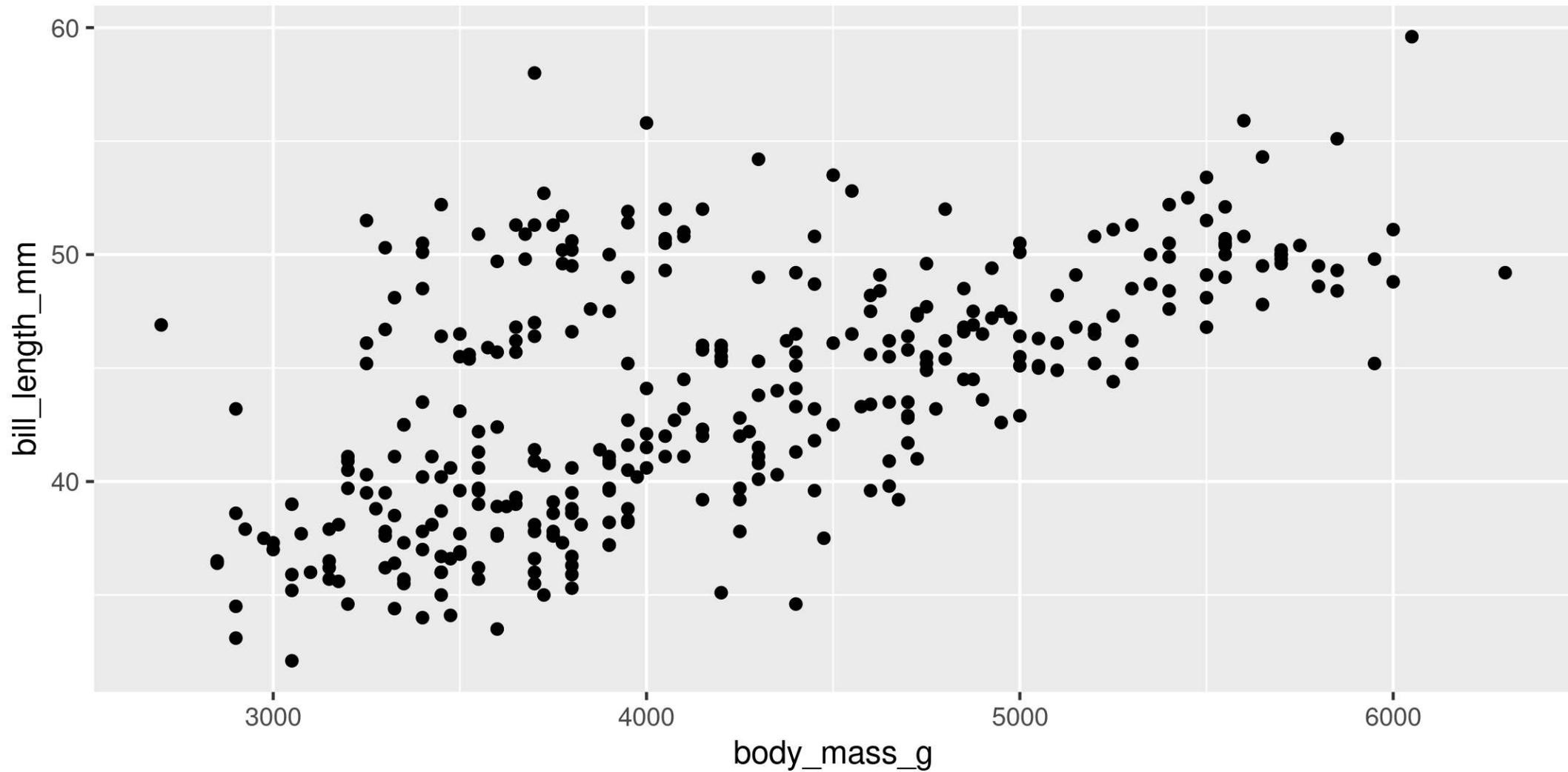
```
1 library(ggplot2)
2
3 ggplot(data = ____, aes(x = ____, y = ____)) +
4   geom____(____)
```



Showing data by group

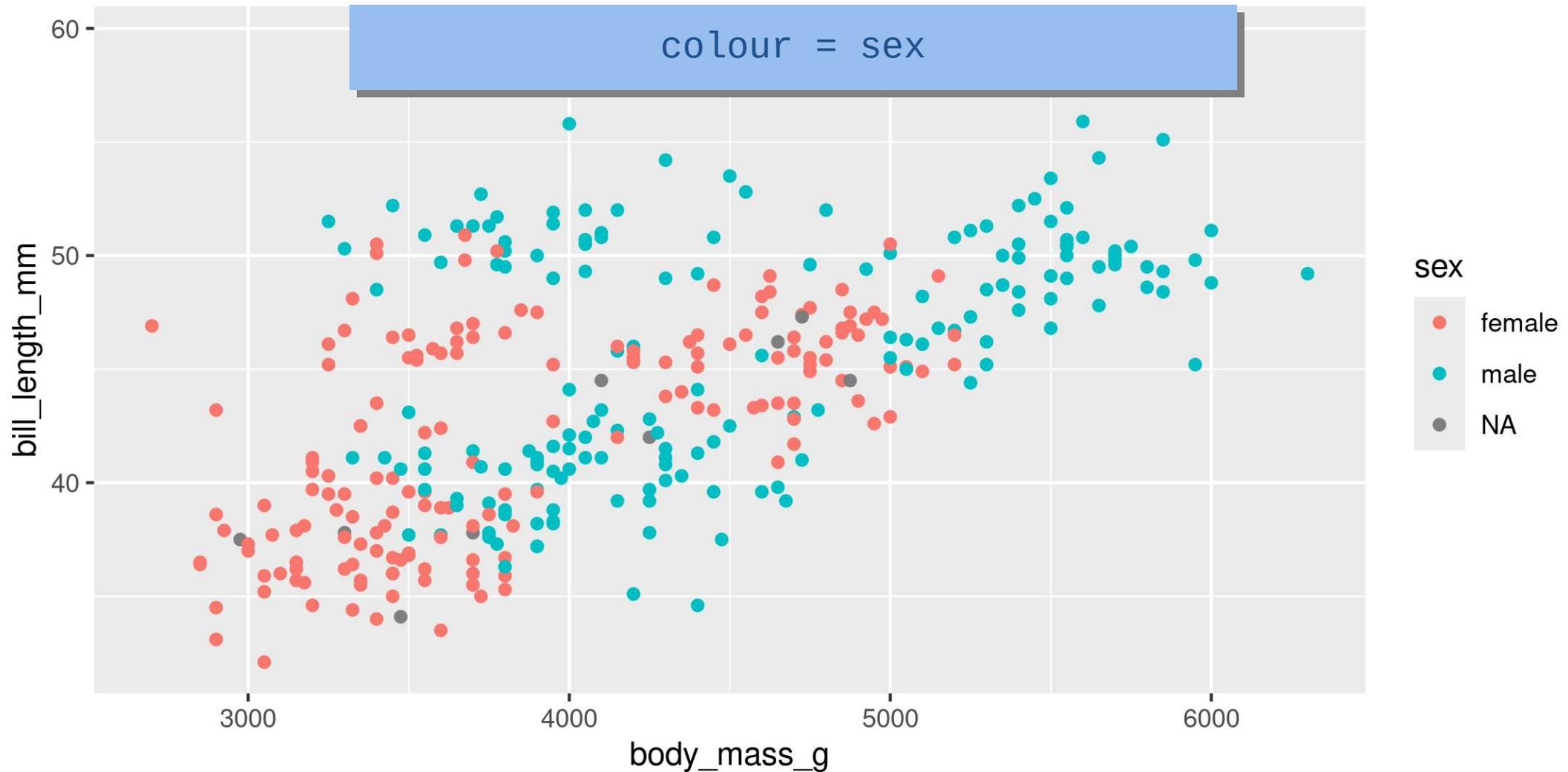
Mapping aesthetics

```
1 ggplot(data = penguins, aes(x = body_mass_g, y = bill_length_mm)) +  
2   geom_point()
```



Mapping aesthetics

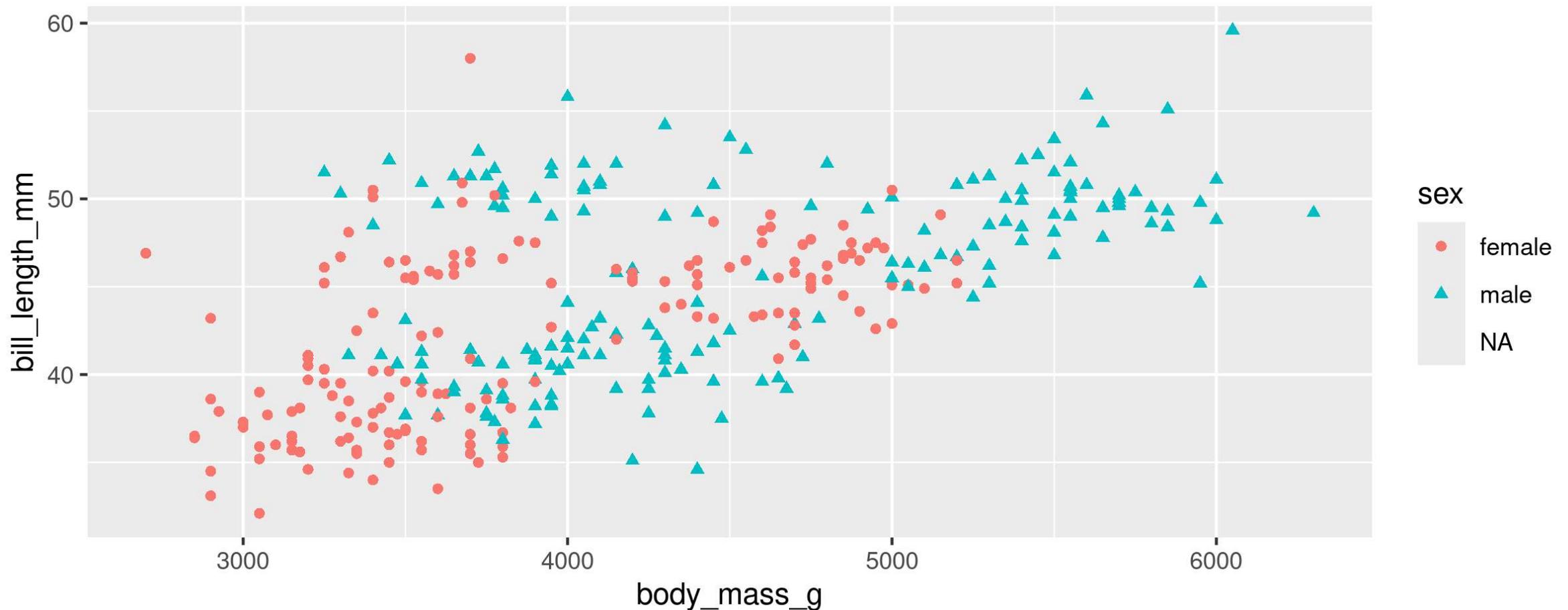
```
1 ggplot(data = penguins, aes(x = body_mass_g, y = bill_length_mm, colour = sex)) +  
2   geom_point()
```



Mapping aesthetics

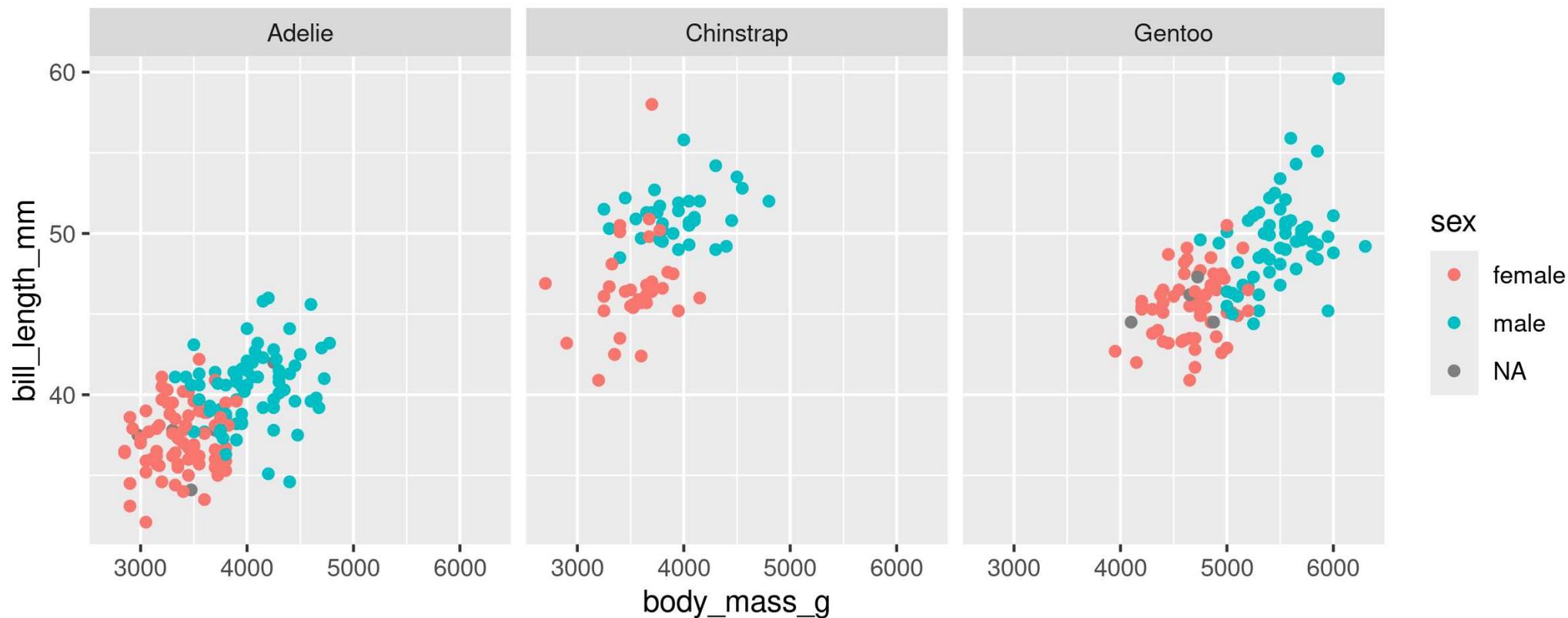
ggplot automatically populates the legends (combining where it can)

```
1 ggplot(data = penguins, aes(x = body_mass_g, y = bill_length_mm, colour = sex, shape = sex)) +  
2   geom_point()
```



Faceting: facet_wrap()

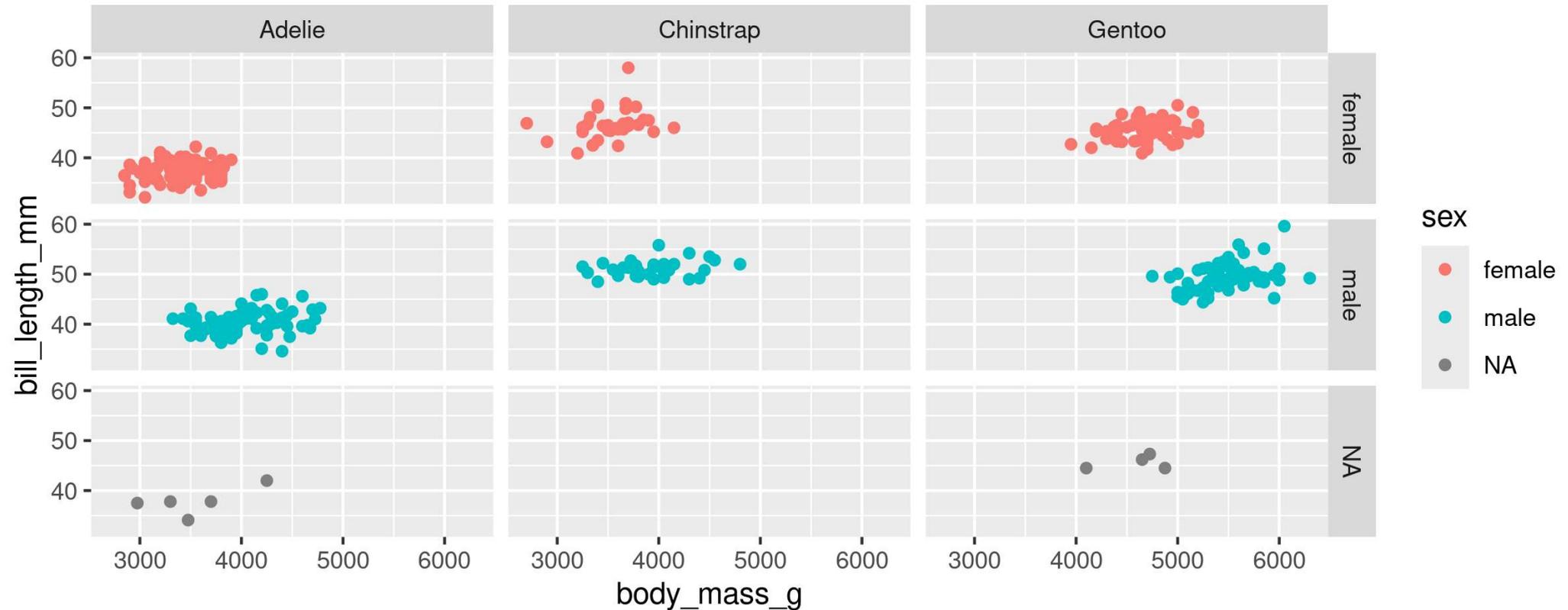
```
1 ggplot(data = penguins, aes(x = body_mass_g, y = bill_length_mm, colour = sex)) +  
2   geom_point() +  
3   facet_wrap(~ species)
```



Split plots by **one** grouping variable

Faceting: facet_grid()

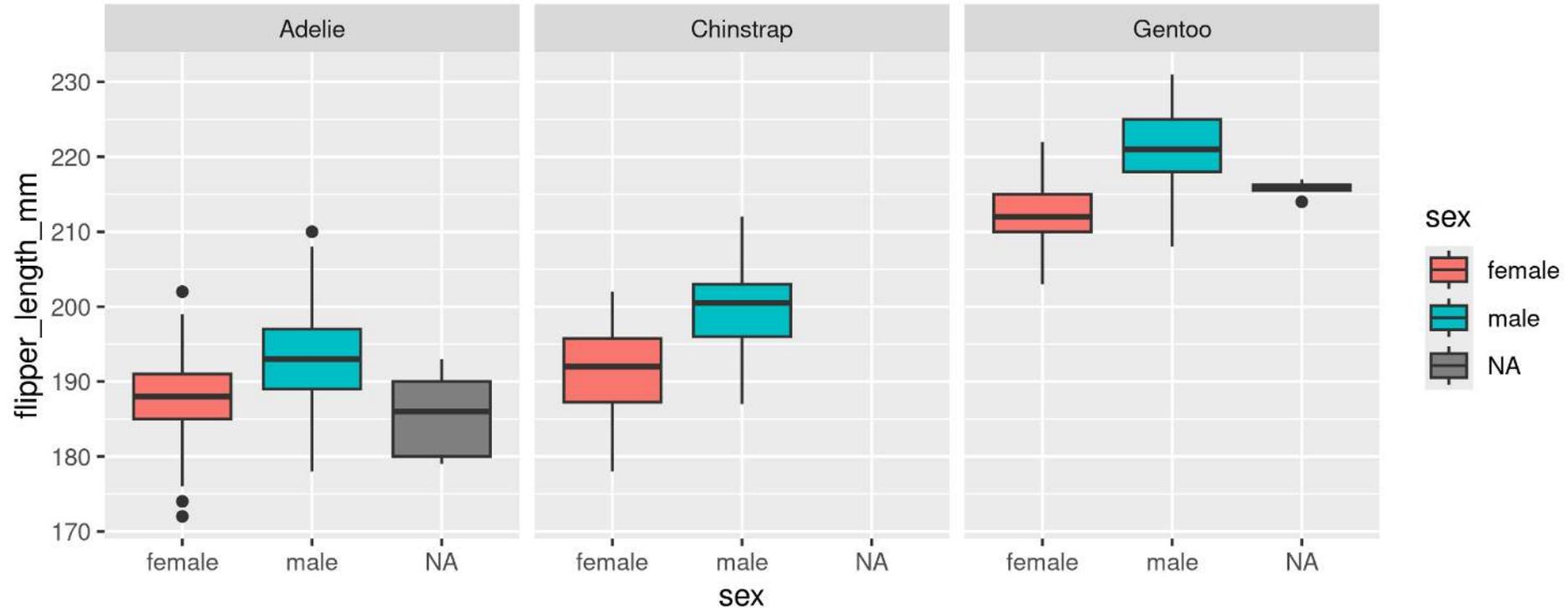
```
1 ggplot(data = penguins, aes(x = body_mass_g, y = bill_length_mm, colour = sex)) +  
2   geom_point() +  
3   facet_grid(sex ~ species)
```



Split plots by **two** grouping variables

Your Turn: Create this plot

```
1 ggplot(data = _____, aes(_____)) +  
2 _____ +  
3 _____
```



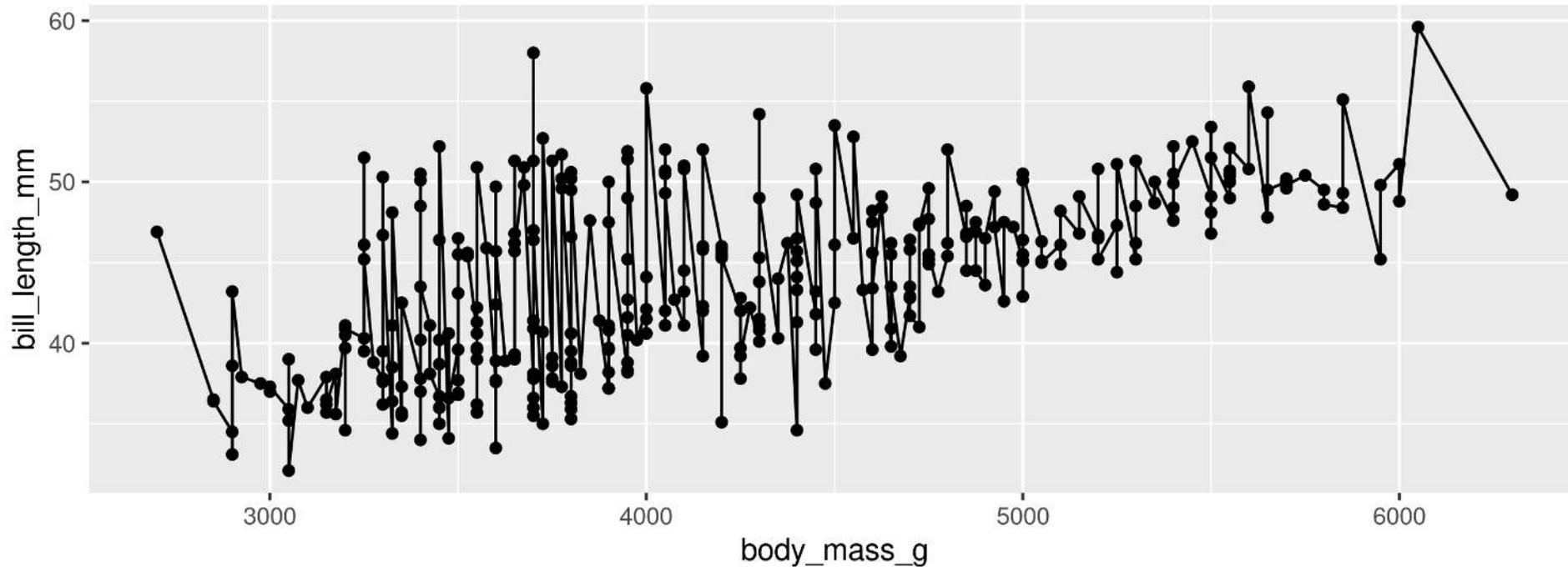
Hint: `colour` is for outlining with a colour, `fill` is for 'filling' with a colour
Too Easy? Split boxplots by sex **and** island

Trendlines / Regression Lines

Trendlines / Regression lines

`geom_line()` is connect-the-dots, not a trend or linear model

```
1 ggplot(data = penguins, aes(x = body_mass_g, y = bill_length_mm)) +  
2   geom_point() +  
3   geom_line()
```



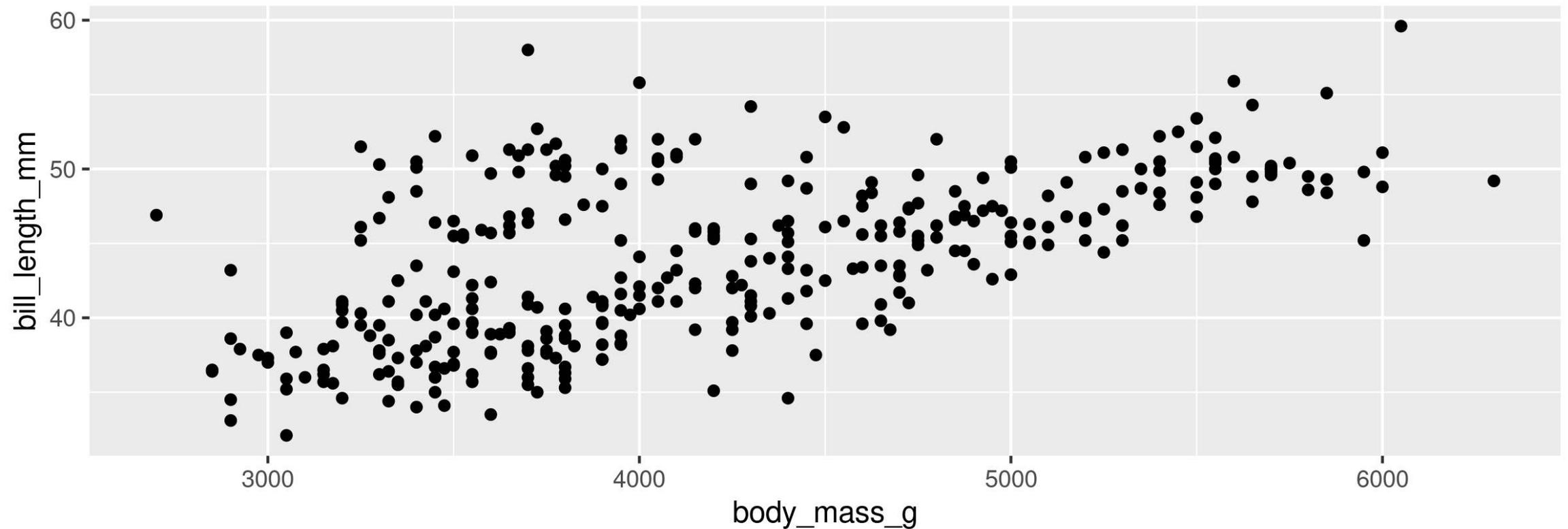
Not what we're looking for

Trendlines / Regression lines

Let's add a trend line properly

Start with basic plot:

```
1 g <- ggplot(data = penguins, aes(x = body_mass_g, y = bill_length_mm)) +  
2   geom_point()  
3 g
```

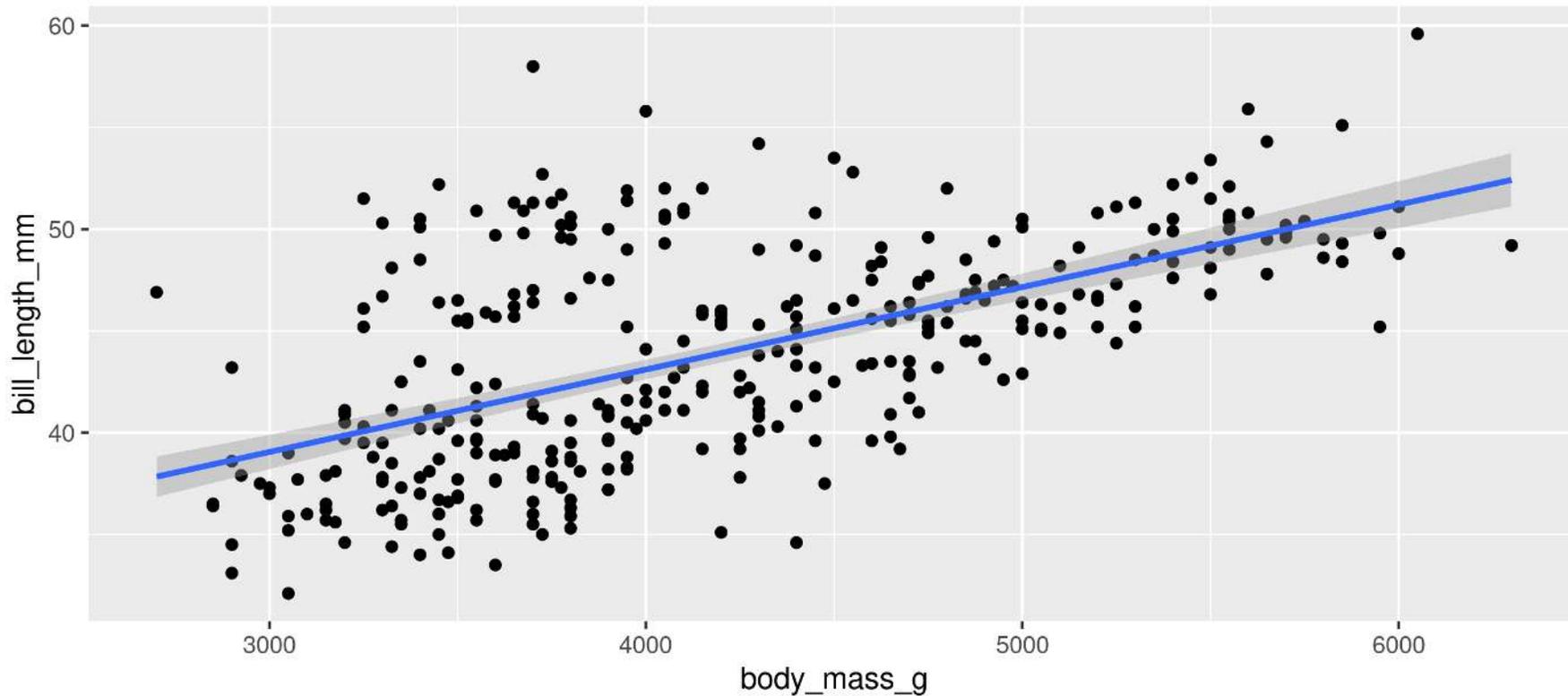


Trendlines / Regression lines

Add the `stat_smooth()`

- `lm` is for “linear model” (i.e. trendline)
- grey ribbon = standard error

```
1 g + stat_smooth(method = "lm")
```

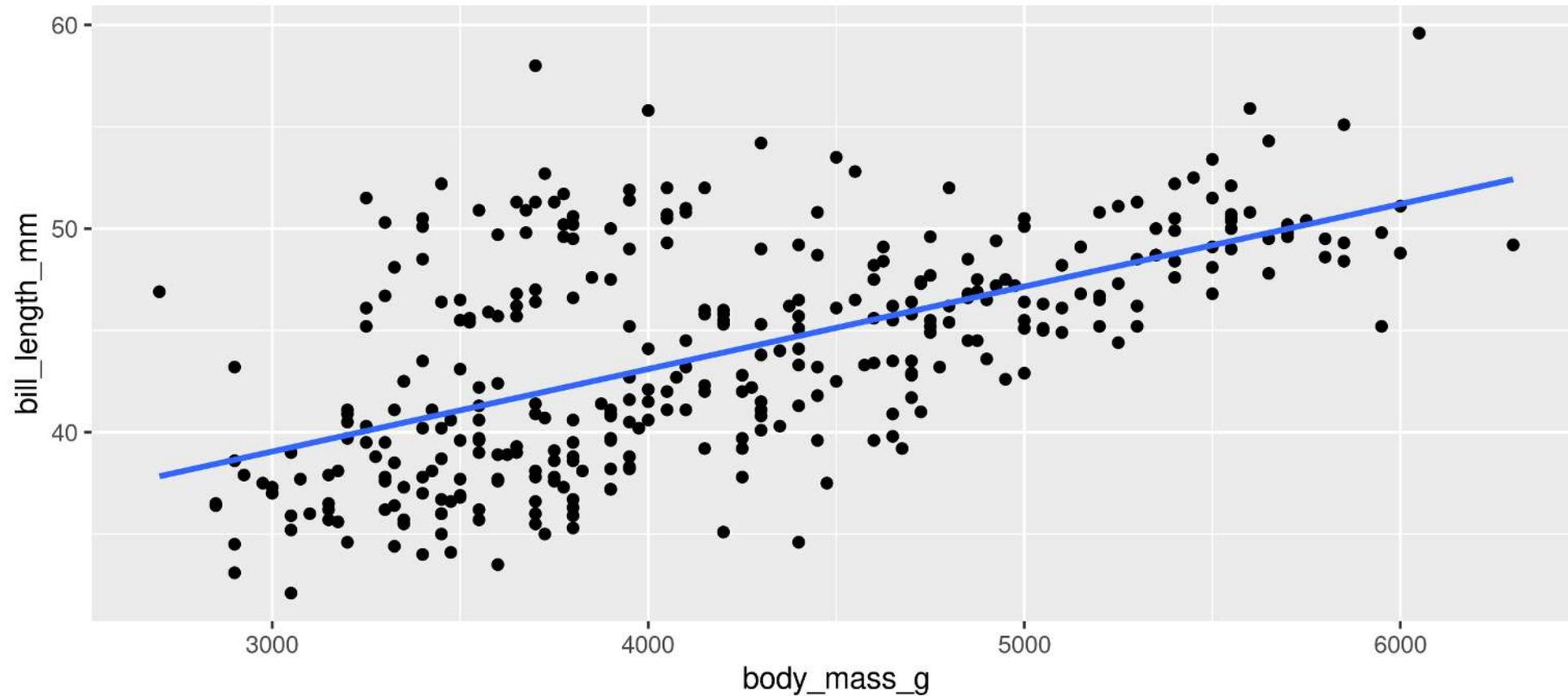


Trendlines / Regression lines

Add the `stat_smooth()`

- remove the grey ribbon `se = FALSE`

```
1 g + stat_smooth(method = "lm", se = FALSE)
```

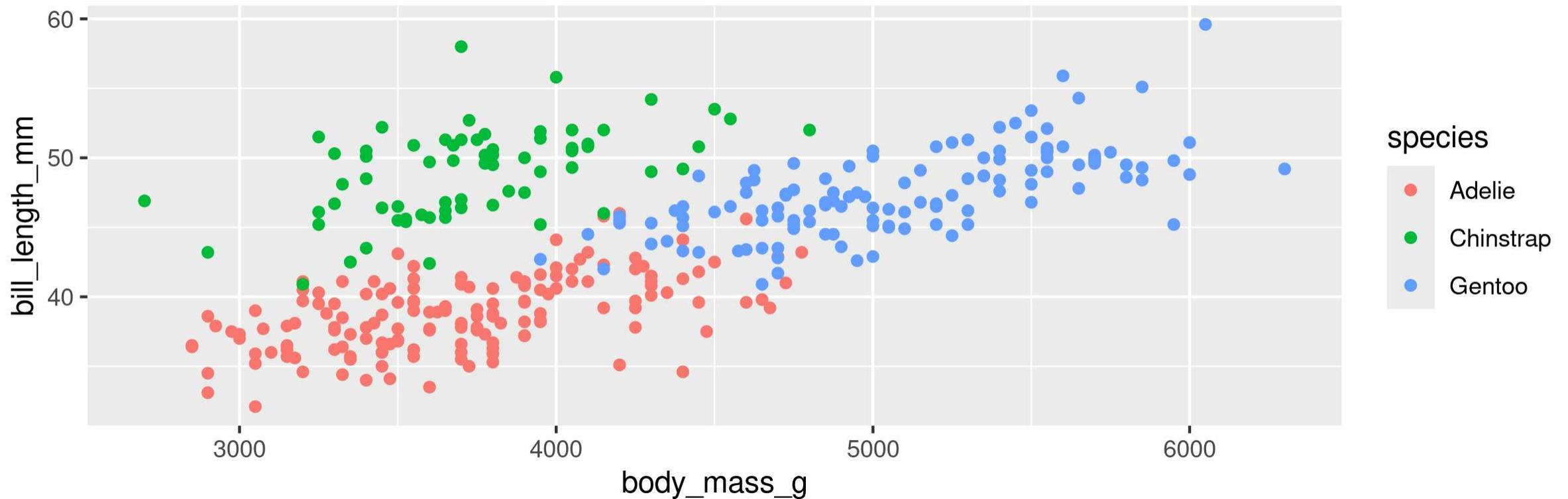


Trendlines / Regression lines

A line for each group

- Specify group (here we use `colour` to specify `species`)

```
1 g <- ggplot(data = penguins, aes(x = body_mass_g, y = bill_length_mm, colour = species)) +  
2   geom_point()  
3 g
```

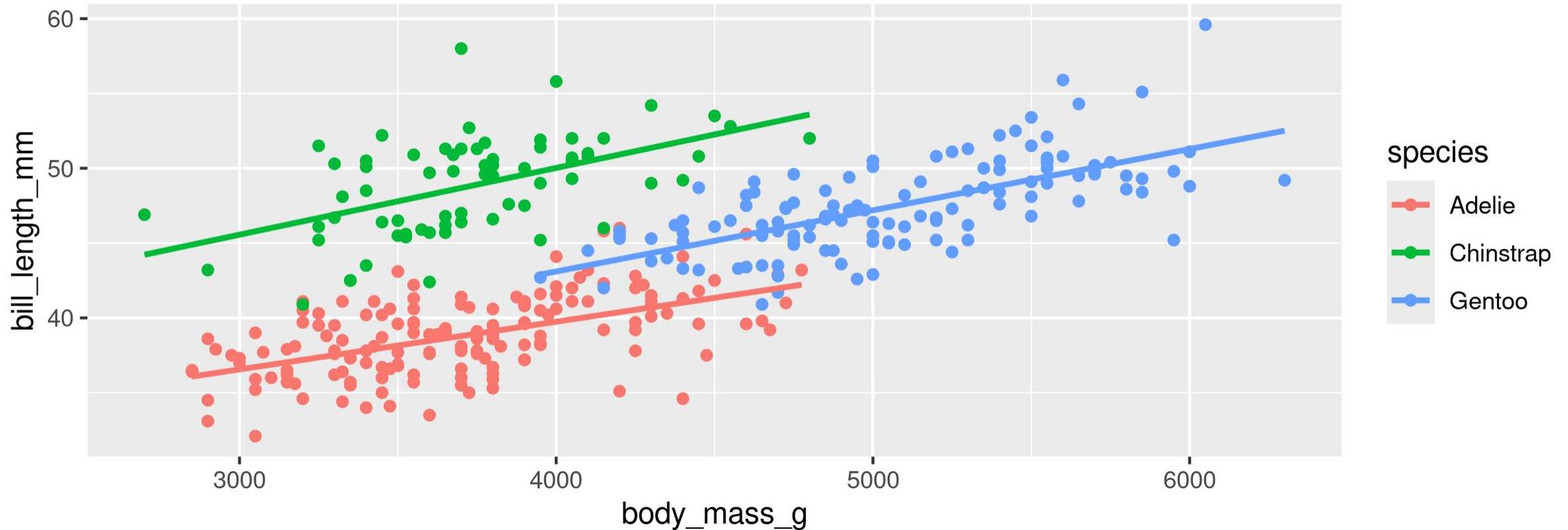


Trendlines / Regression lines

A line for each group

- `stat_smooth()` automatically uses the same grouping

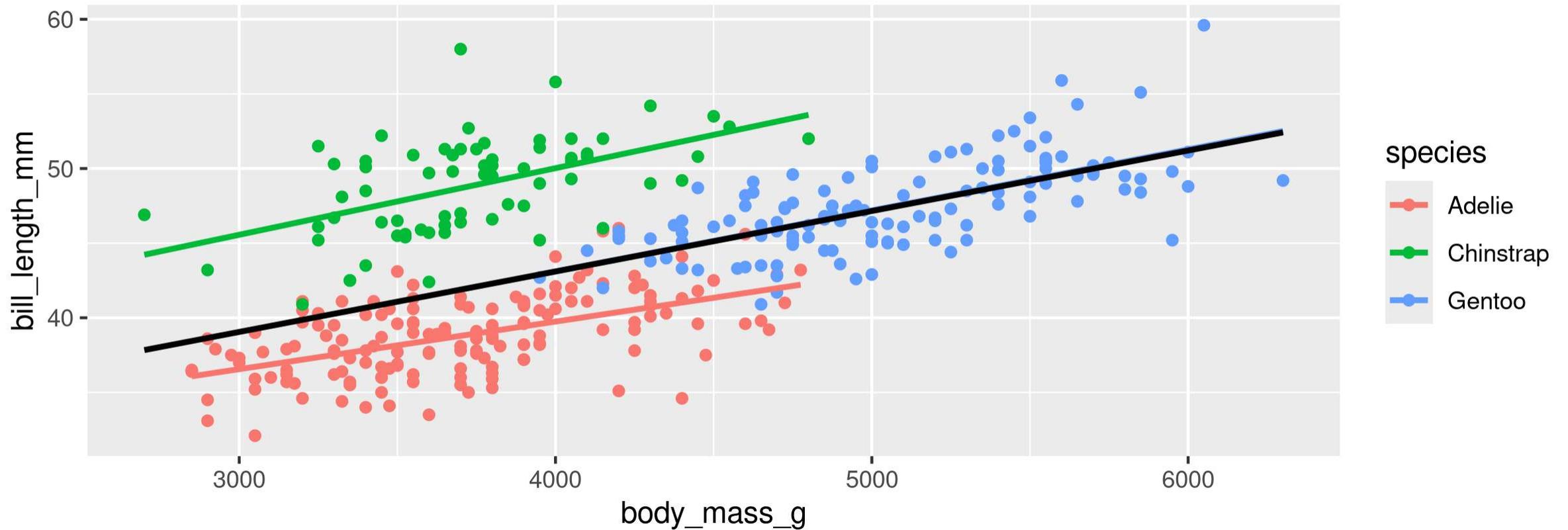
```
1 g + stat_smooth(method = "lm", se = FALSE)
```



Trendlines / Regression lines

A line for each group AND overall

```
1 g +  
2   stat_smooth(method = "lm", se = FALSE) +  
3   stat_smooth(method = "lm", se = FALSE, colour = "black")
```



Your Turn: Create this plot

- A scatter plot: Flipper Length by Body Mass grouped by Species
- With *a single regression line for the overall trend*

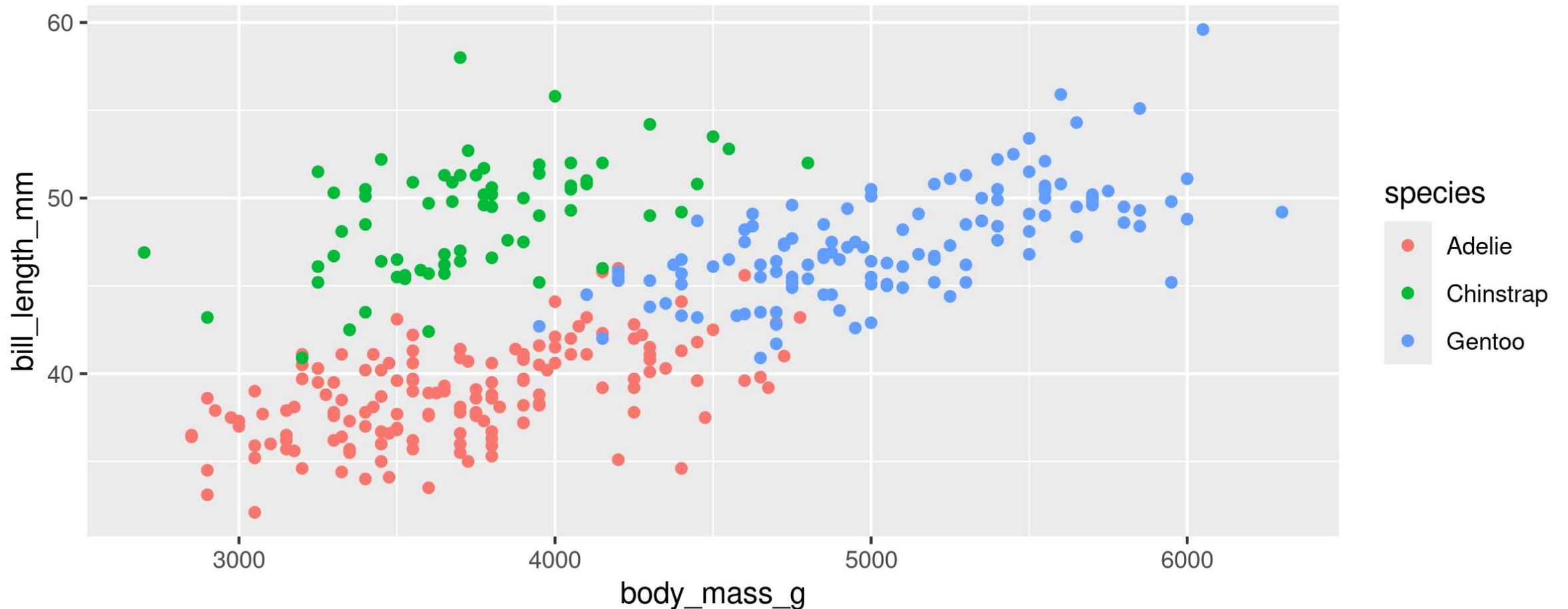
Too Easy? Create a separate plot for each sex as well

Customizing plots

Customizing: Starting plot

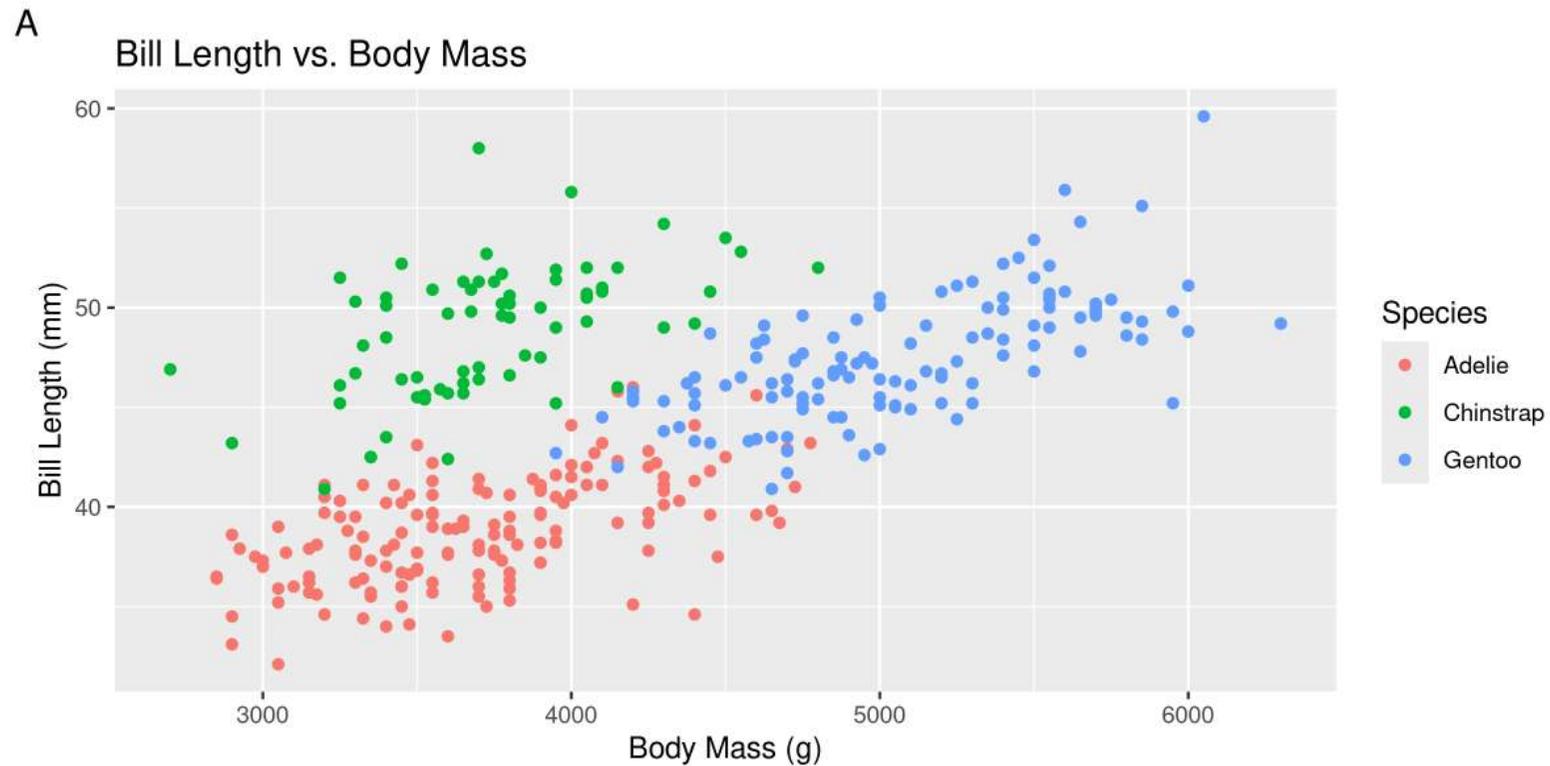
Let's work with this plot

```
1 g <- ggplot(data = penguins, aes(x = body_mass_g, y = bill_length_mm, colour = species)) +  
2   geom_point()
```



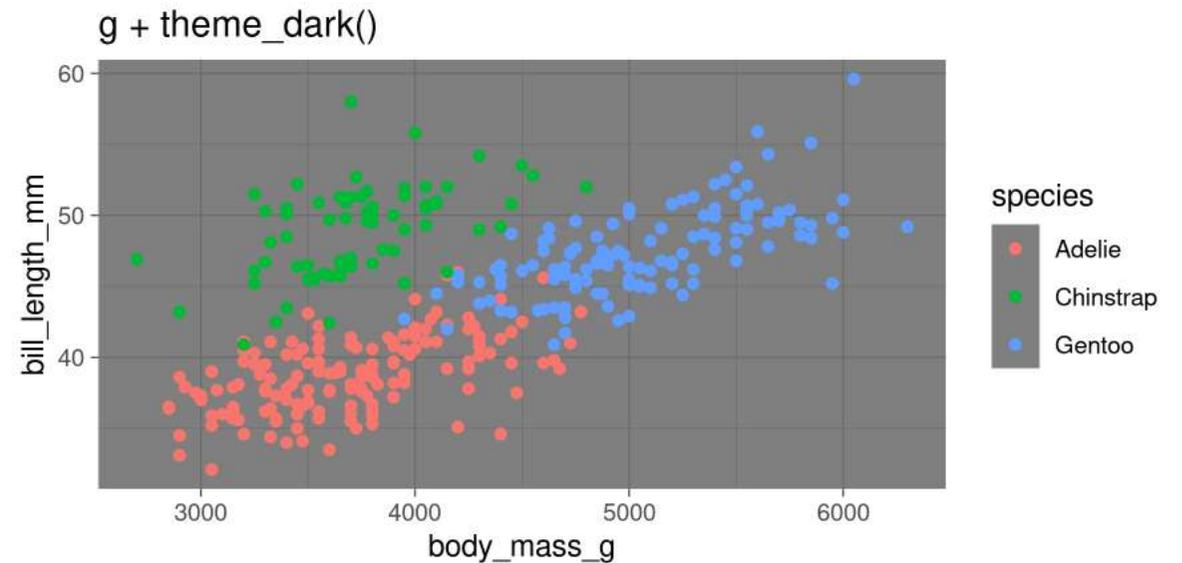
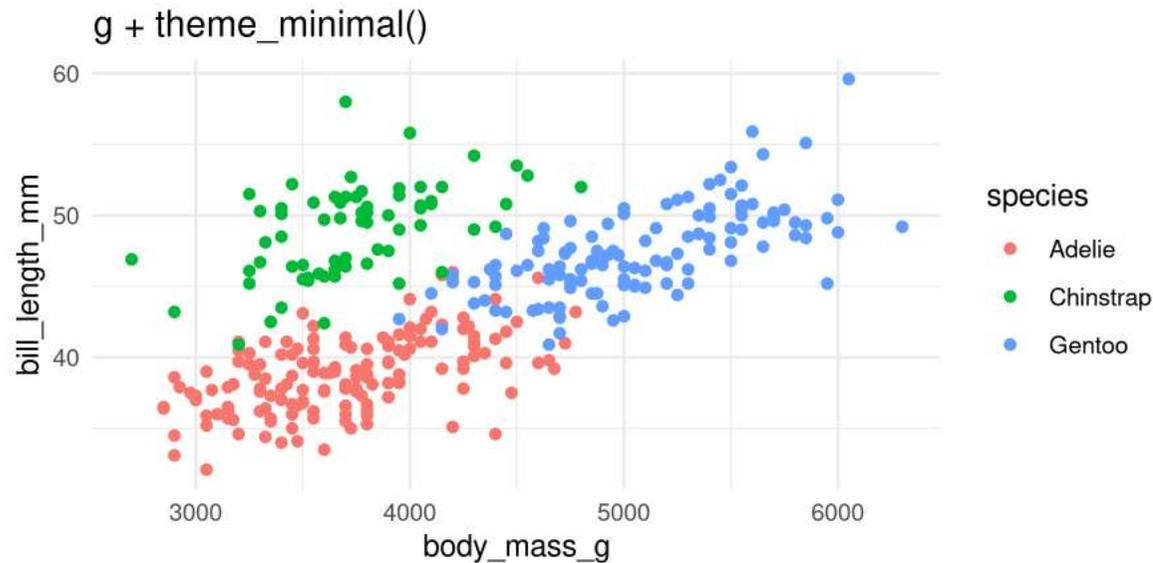
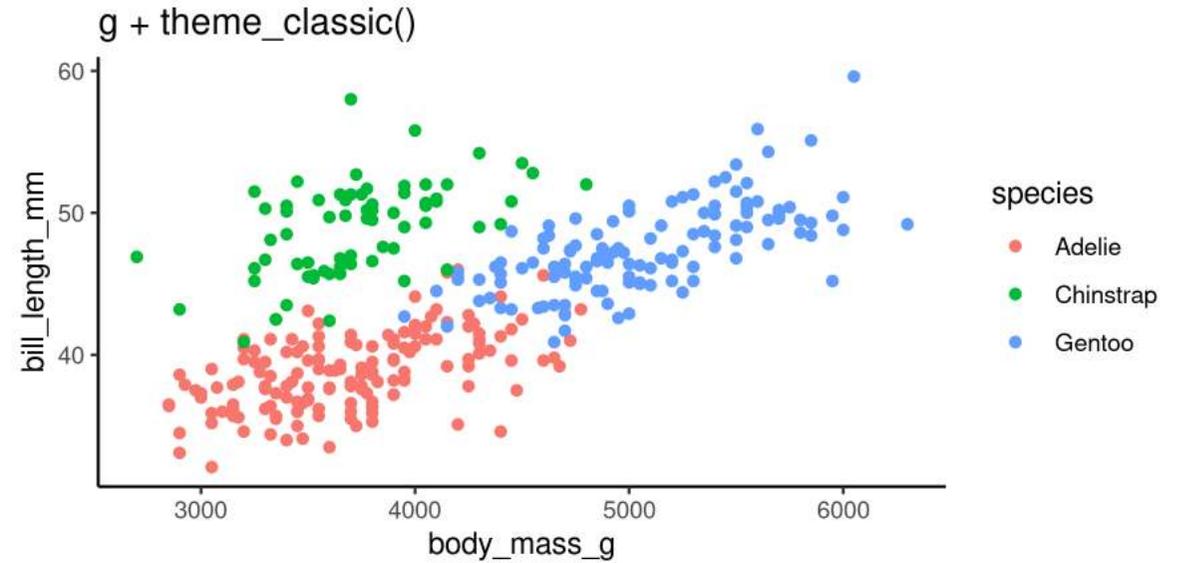
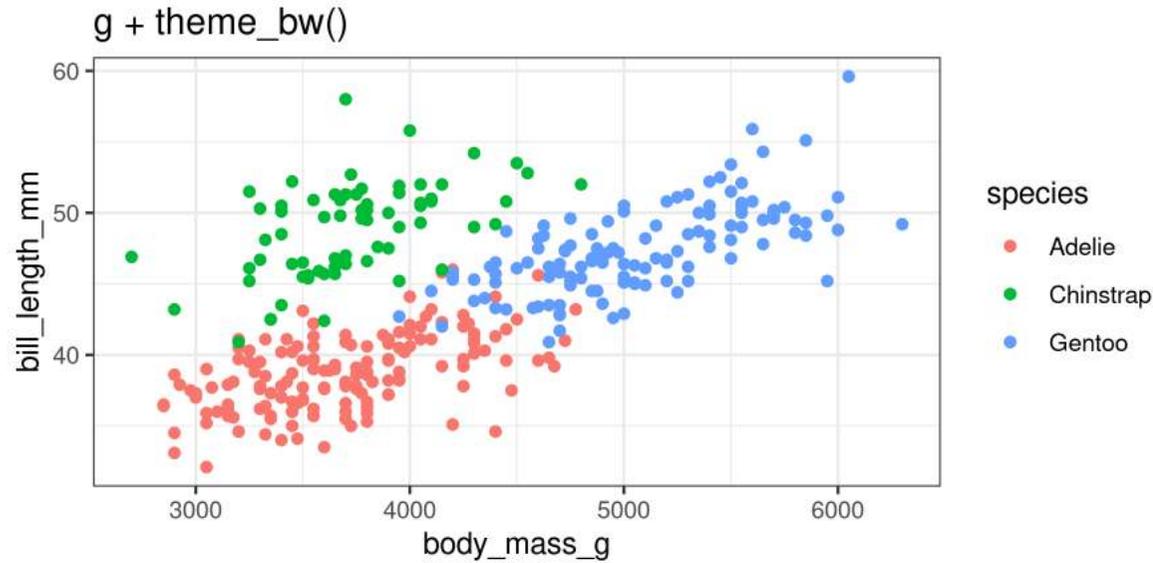
Customizing: Labels

```
1 g + labs(title = "Bill Length vs. Body Mass",  
2         x = "Body Mass (g)",  
3         y = "Bill Length (mm)",  
4         colour = "Species", tag = "A")
```



Practice for later: Add proper labels to some of your previous plots

Customizing: Built-in themes



Customizing: Axes

`scale_ + (x or y) + type (continuous, discrete, date, datetime)`

- `scale_x_continuous()`
- `scale_y_discrete()`
- etc.

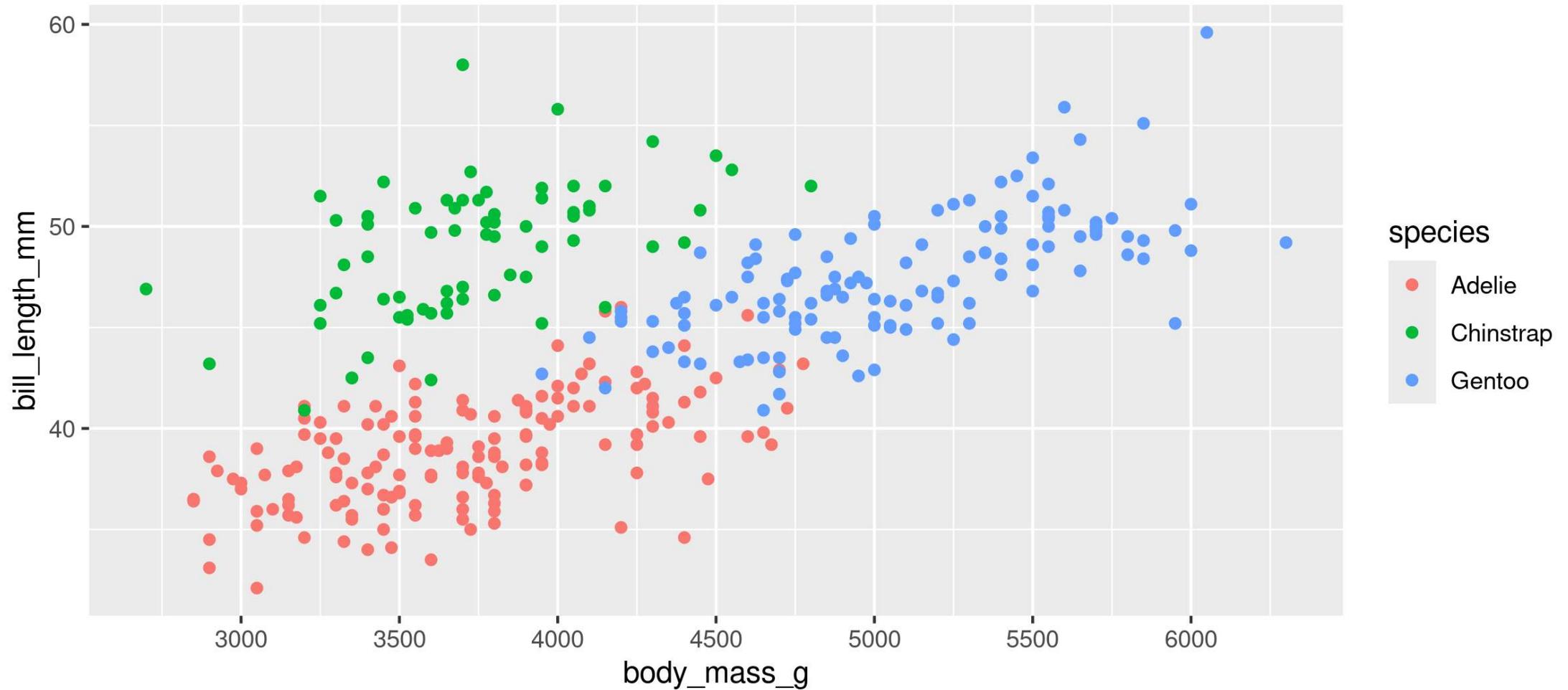
Common arguments

```
1 g + scale_x_continuous(breaks = seq(0, 20, 10)) # Tick breaks
2 g + scale_x_continuous(limits = c(0, 15))      # xlim() is a shortcut for this
3 g + scale_x_continuous(expand = c(0, 0))      # Space between axis and data
```

Customizing: Axes

Breaks

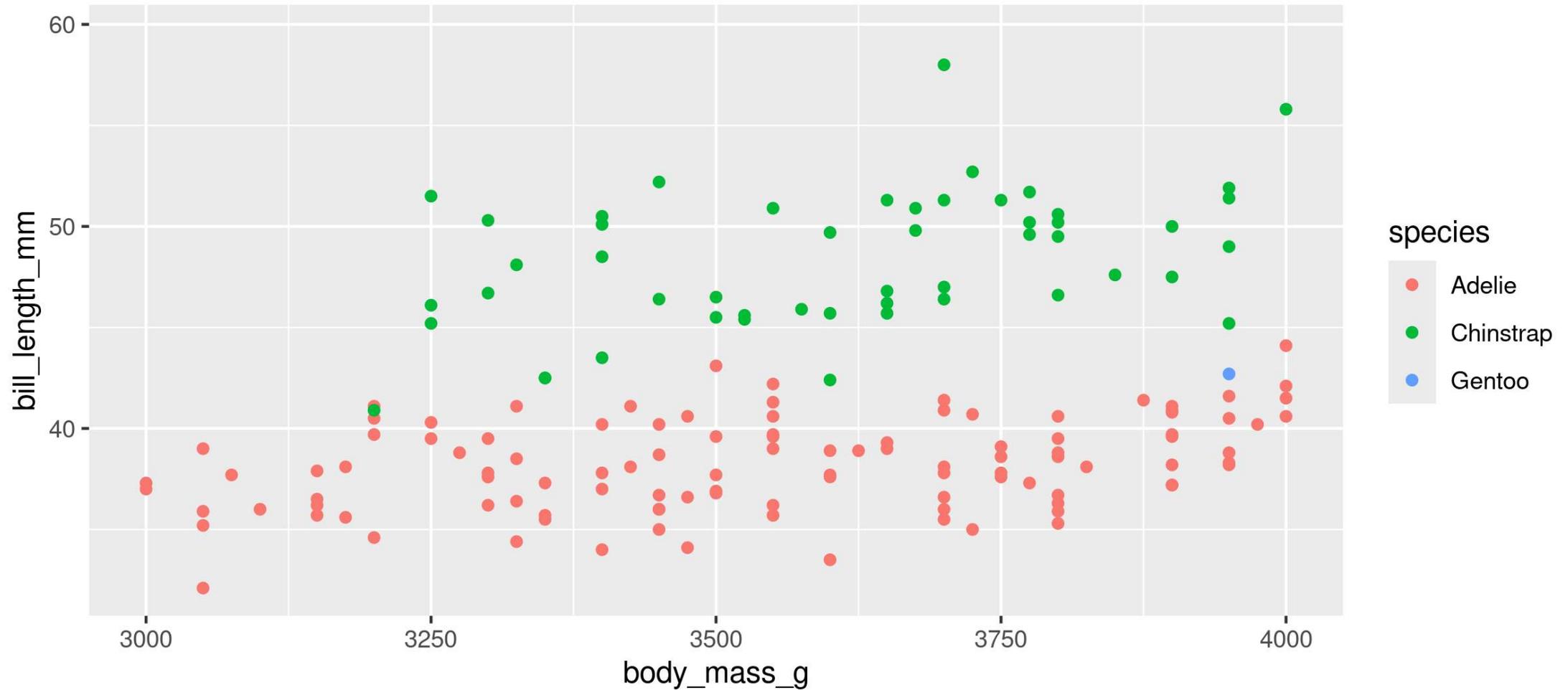
```
1 g + scale_x_continuous(breaks = seq(2500, 6500, 500))
```



Customizing: Axes

Limits

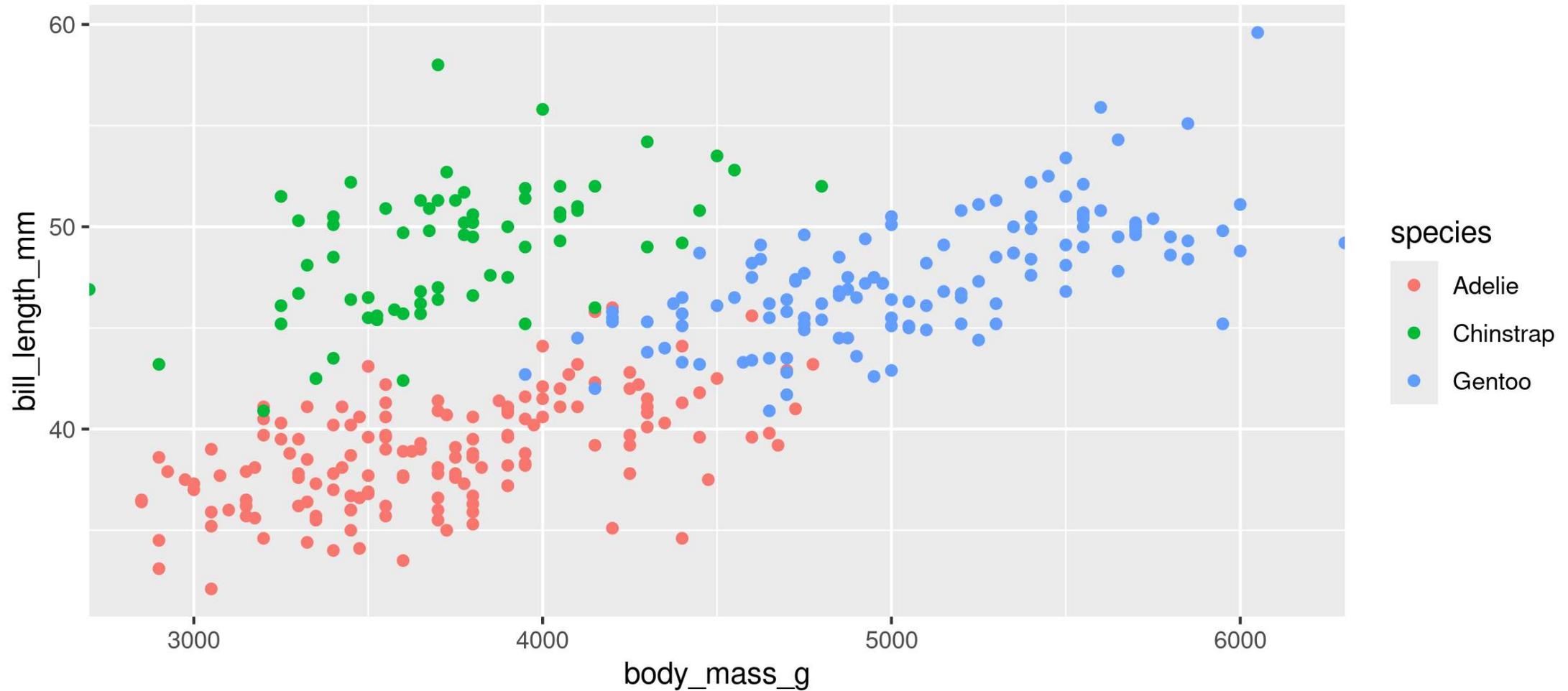
```
1 g + scale_x_continuous(limits = c(3000, 4000))
```



Customizing: Axes

Space between origin and axis start

```
1 g + scale_x_continuous(expand = c(0, 0))
```

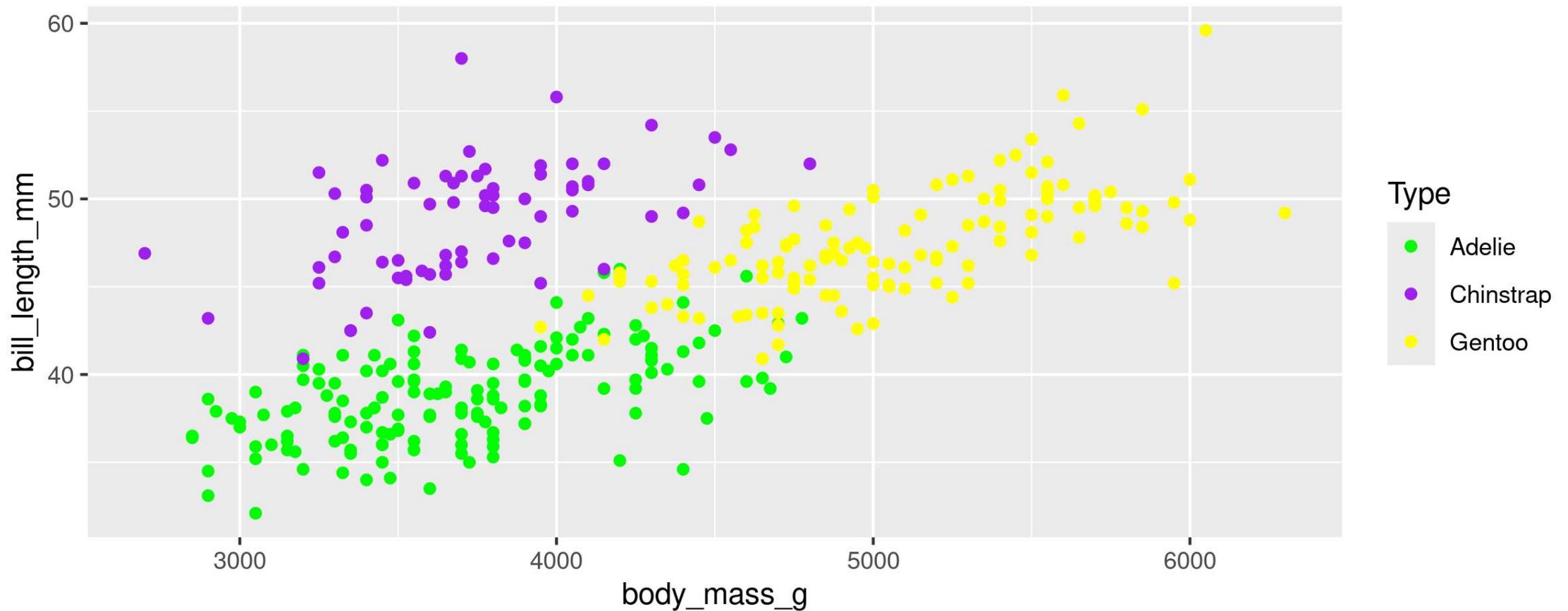


Customizing: Aesthetics

Using scales

`scale_` + aesthetic (colour, fill, size, etc.) + type (manual, continuous, datetime, etc.)

```
1 g + scale_colour_manual(name = "Type", values = c("green", "purple", "yellow"))
```

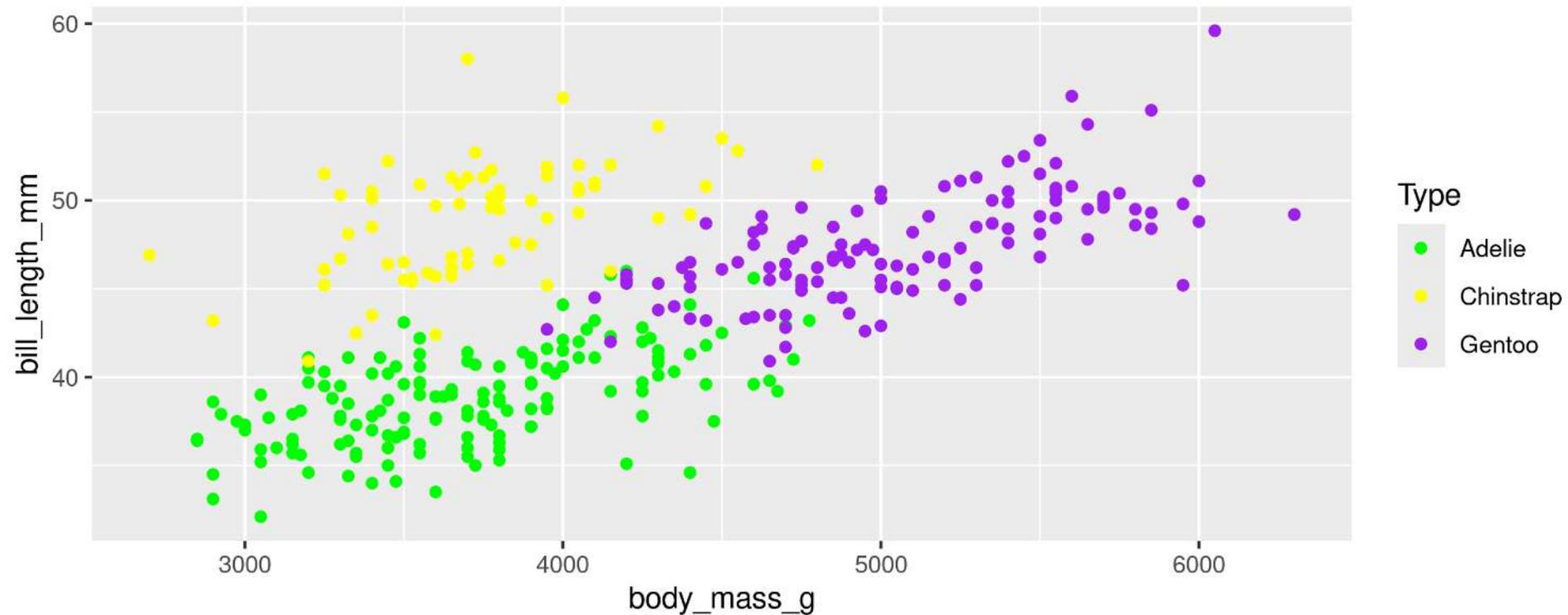


Customizing: Aesthetics

Using scales

Or be very explicit:

```
1 g + scale_colour_manual(  
2   name = "Type",  
3   values = c("Adelie" = "green", "Gentoo" = "purple", "Chinstrap" = "yellow"),  
4   na.value = "black")
```

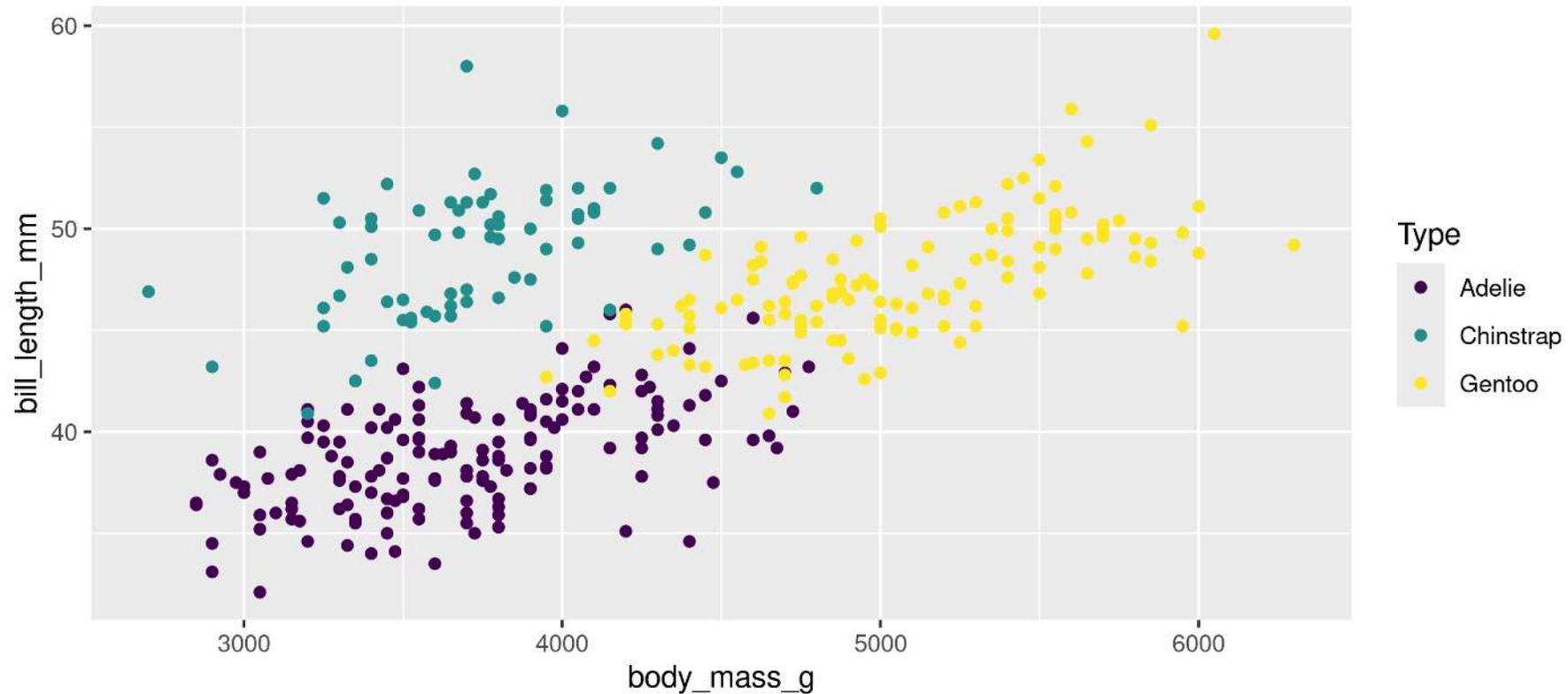


Customizing: Aesthetics

For colours, consider colour-blind-friendly scale

`viridis_d` for “discrete” data

```
1 ggplot(data = penguins, aes(x = body_mass_g, y = bill_length_mm, colour = species)) +  
2   geom_point() +  
3   scale_colour_viridis_d(name = "Type")
```

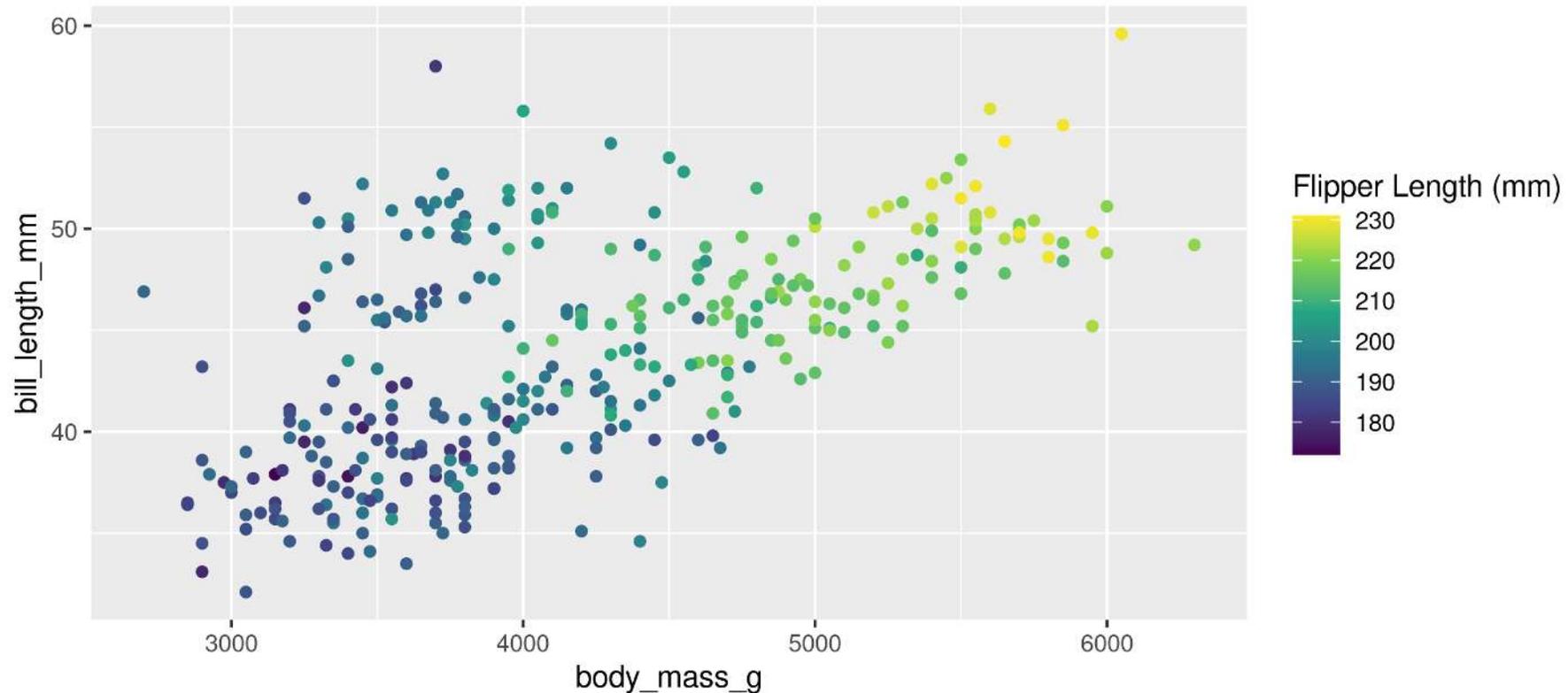


Customizing: Aesthetics

For colours, consider colour-blind-friendly scale

`viridis_c` for “continuous” data

```
1 ggplot(data = penguins, aes(x = body_mass_g, y = bill_length_mm, colour = flipper_length_mm)) +  
2   geom_point() +  
3   scale_colour_viridis_c(name = "Flipper Length (mm)")
```

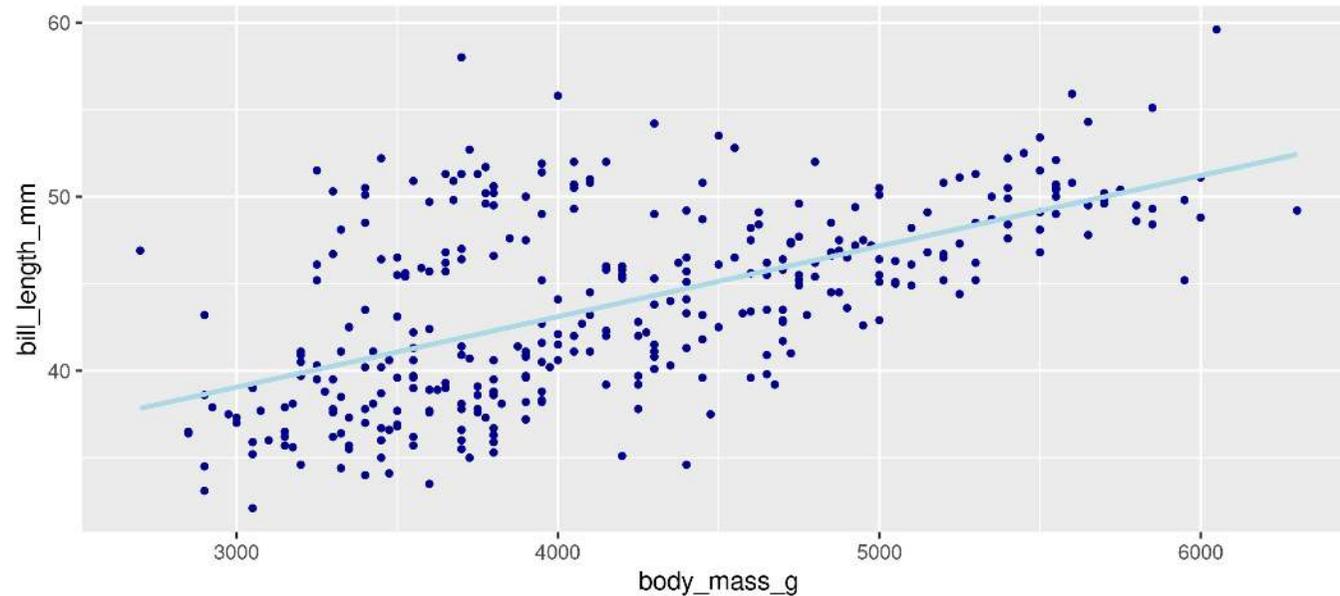


Customizing: Aesthetics

Forcing

Remove the association between a variable and an aesthetic

```
1 ggplot(data = penguins, aes(x = body_mass_g, y = bill_length_mm, colour = sex)) +  
2   geom_point(colour = "darkblue", size = 1) +  
3   stat_smooth(method = "lm", se = FALSE, colour = "lightblue")
```

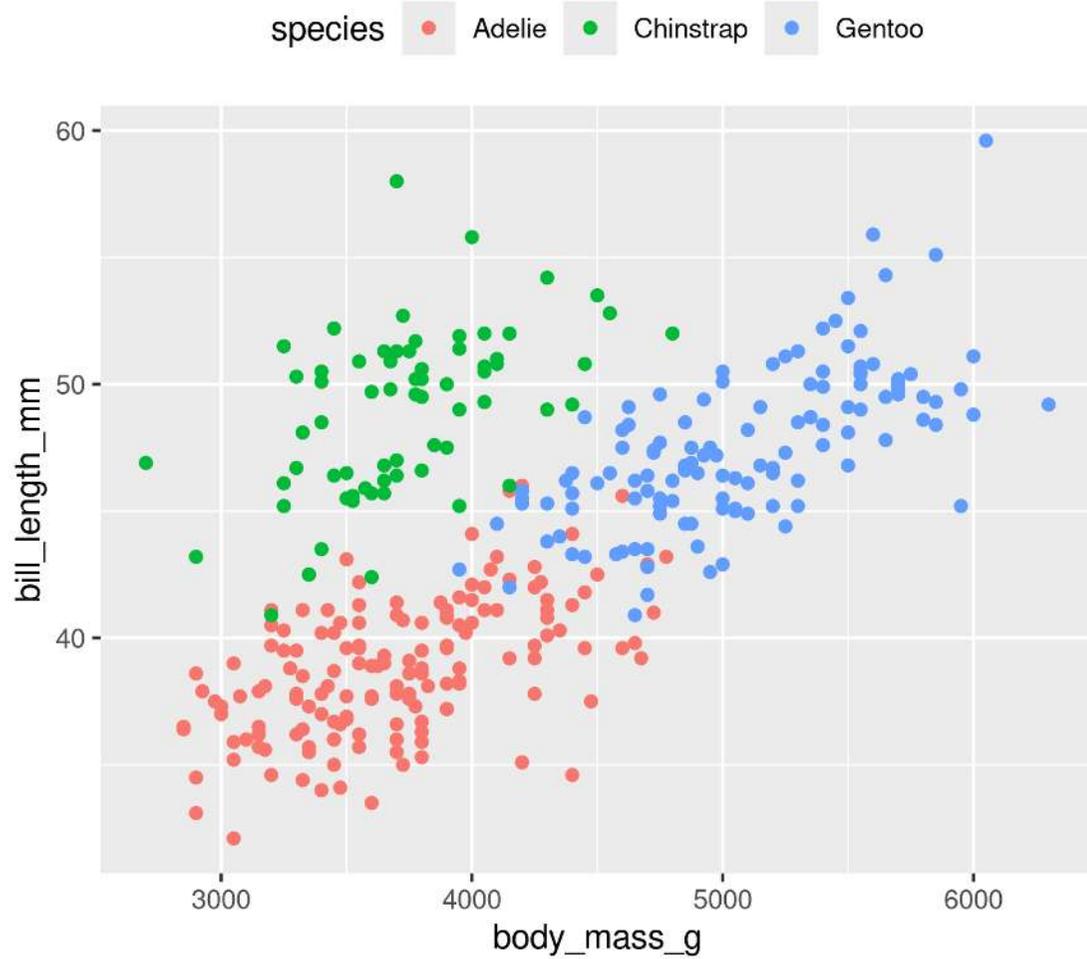


Note: When forcing, aesthetic is not inside `aes()`

Customizing: Legends placement

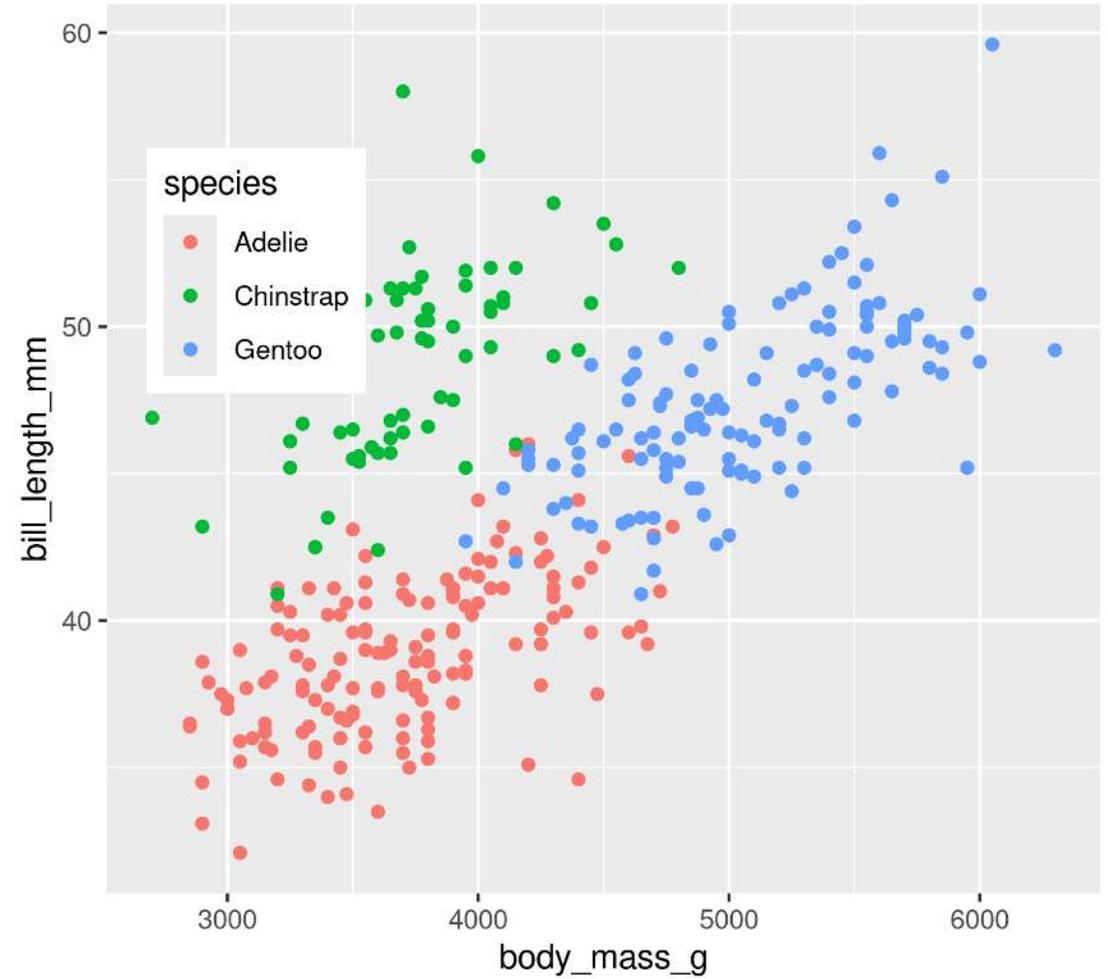
At the: top, bottom, left, right

```
1 g + theme(legend.position = "top")
```



Exactly here

```
1 g + theme(legend.position = c(0.15, 0.7))
```



Combining plots

Combining plots with patchwork

Setup

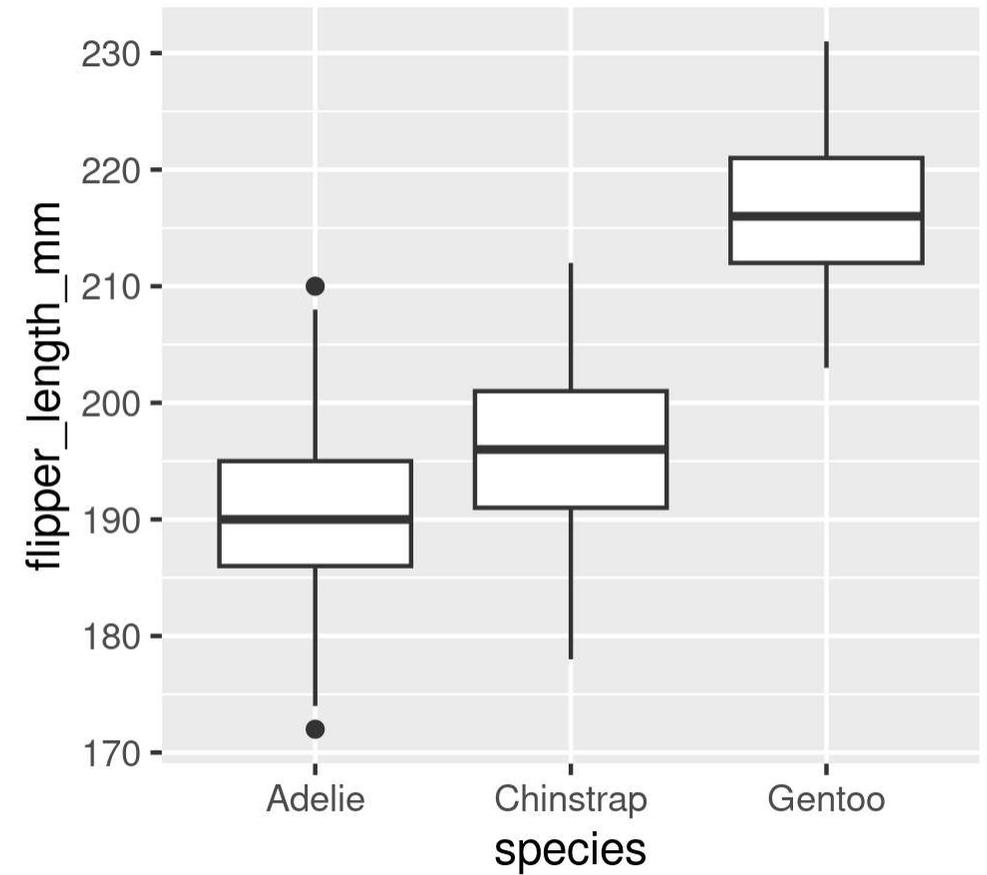
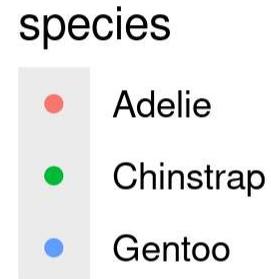
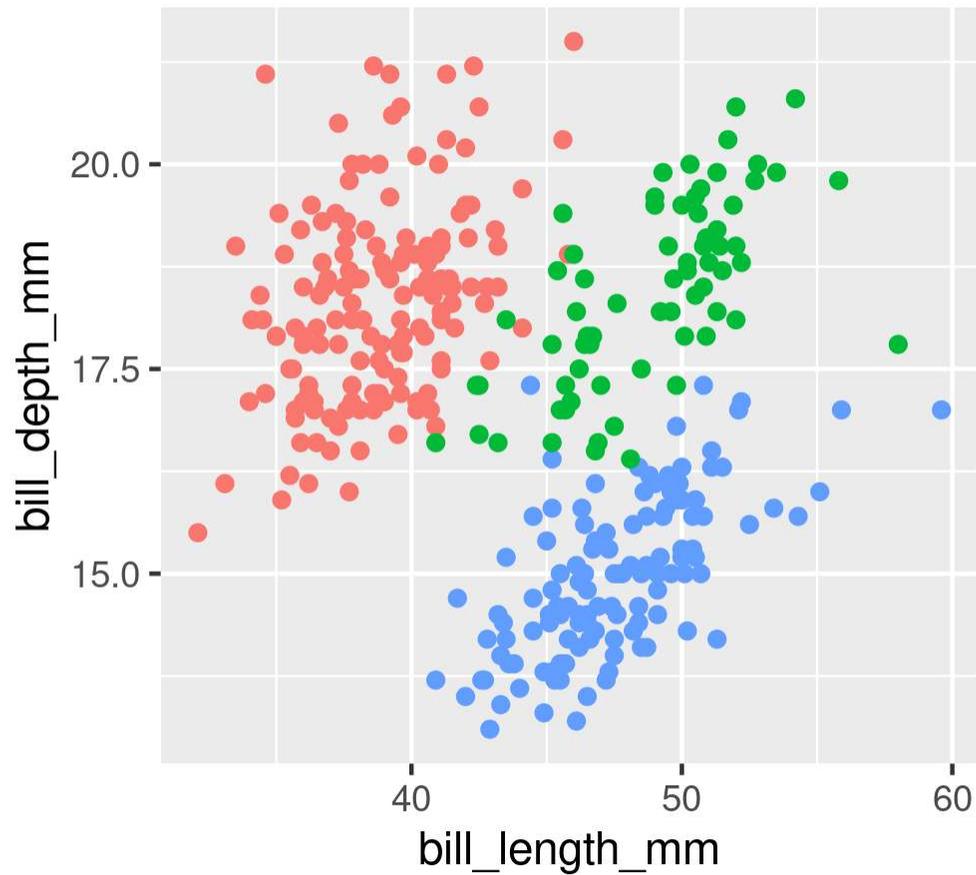
- Load `patchwork`
- Create a couple of different plots

```
1 library(patchwork)
2
3 g1 <- ggplot(data = penguins, aes(x = bill_length_mm, y = bill_depth_mm, colour = species)) +
4   geom_point()
5
6 g2 <- ggplot(data = penguins, aes(x = species, y = flipper_length_mm)) +
7   geom_boxplot()
8
9 g3 <- ggplot(data = penguins, aes(x = flipper_length_mm, y = body_mass_g, colour = species)) +
10  geom_point()
```

Combining plots with patchwork

Side-by-Side 2 plots

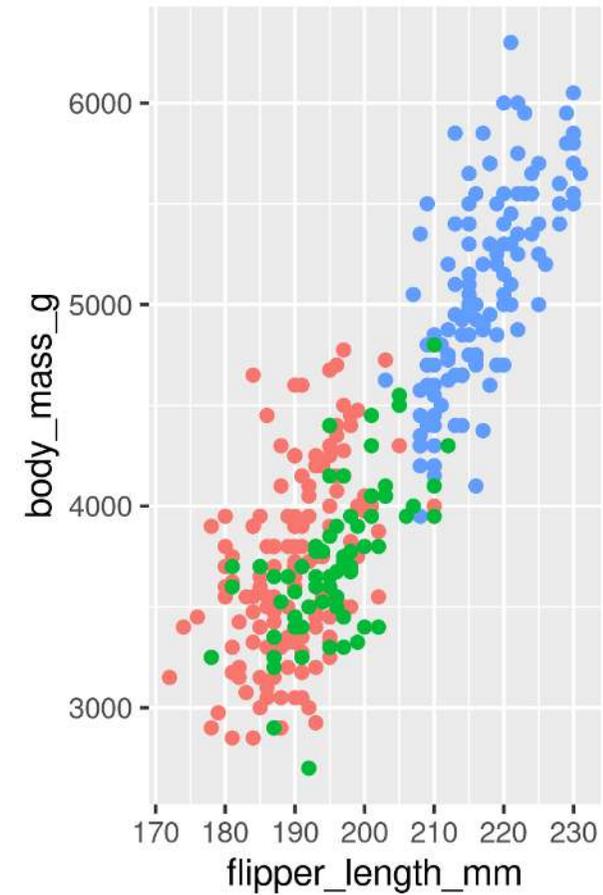
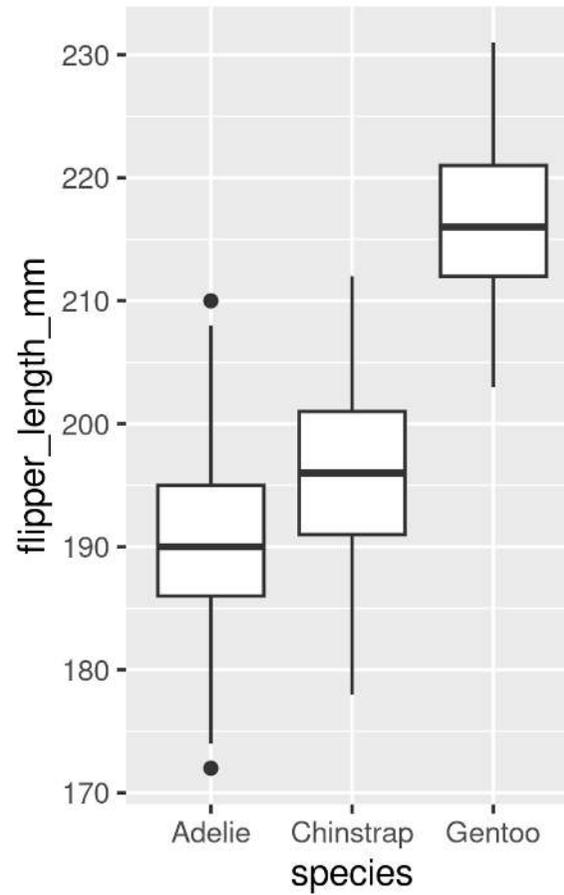
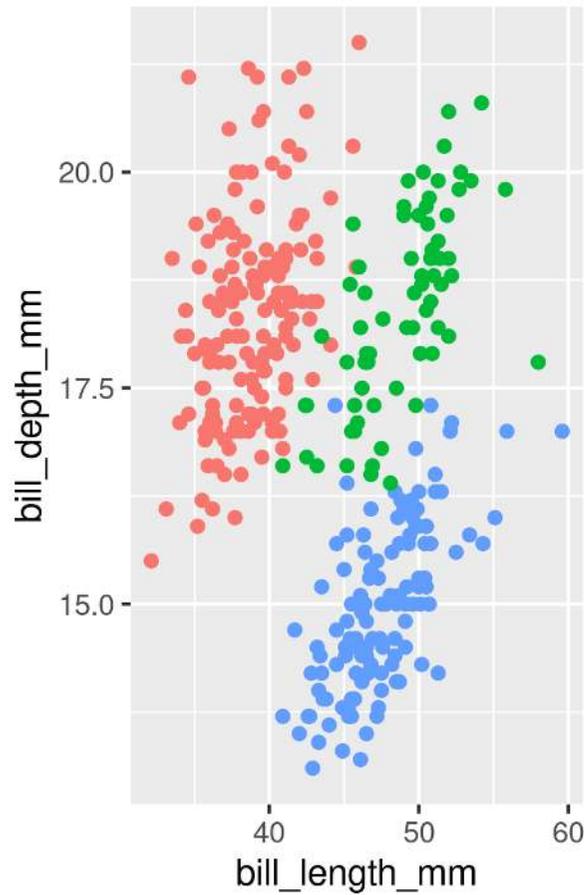
```
1 g1 + g2
```



Combining plots with patchwork

Side-by-Side 3 plots

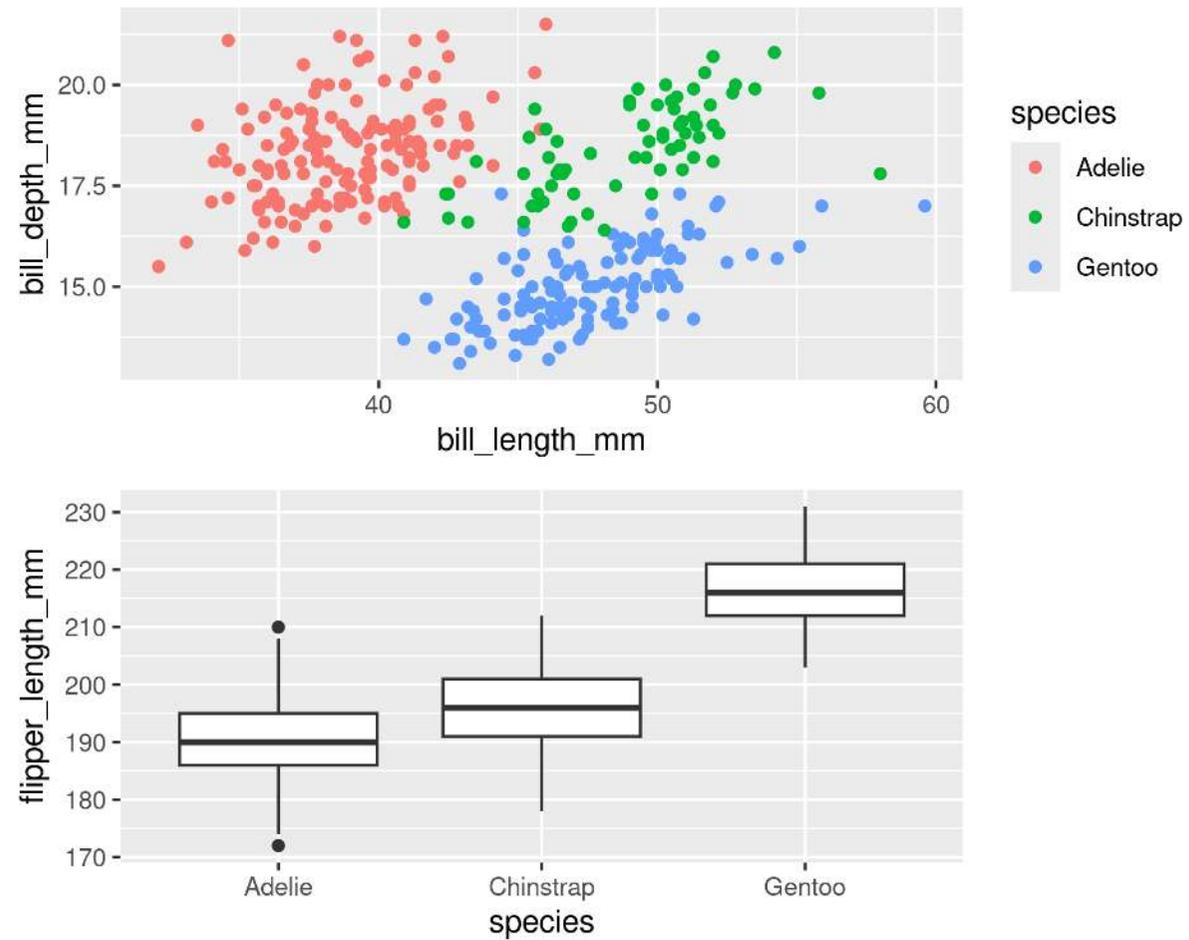
```
1 g1 + g2 + g3
```



Combining plots with patchwork

Stacked 2 plots

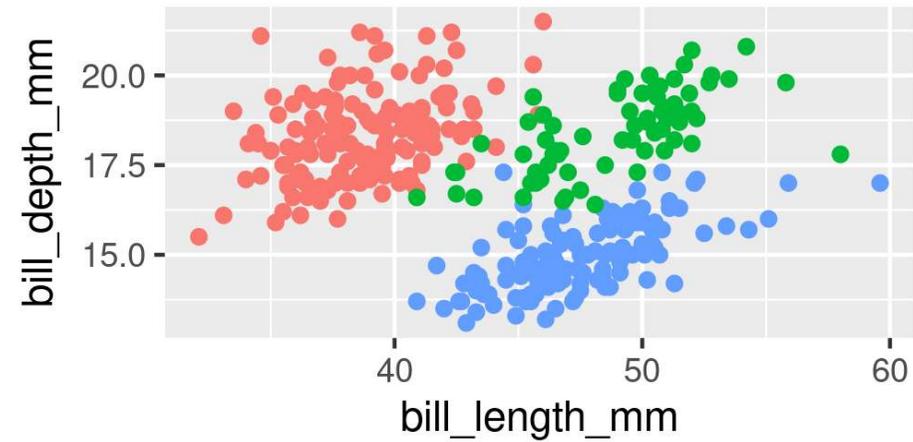
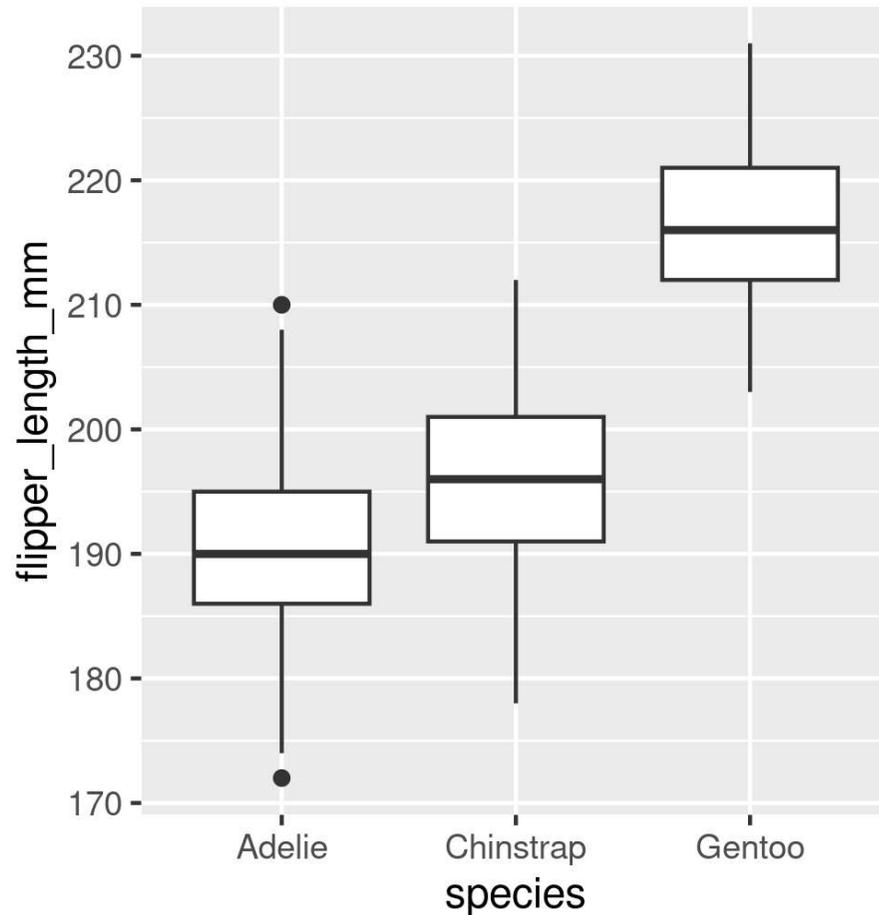
1 g1 / g2



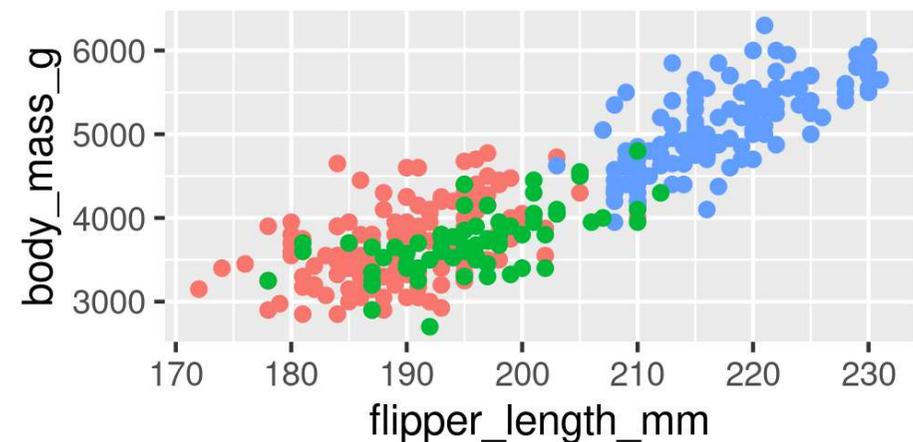
Combining plots with patchwork

More complex arrangements

```
1 g2 + (g1 / g3)
```



species



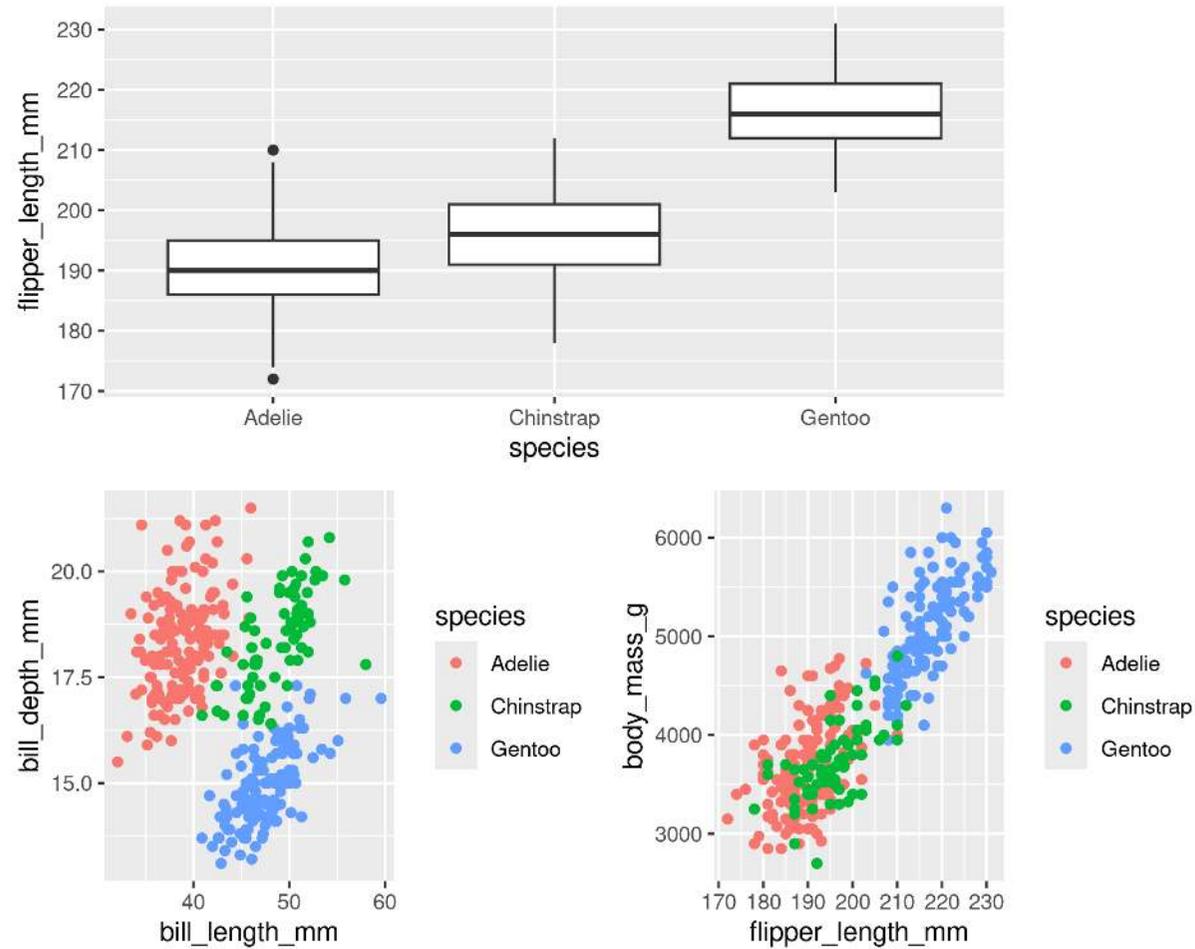
species



Combining plots with patchwork

More complex arrangements

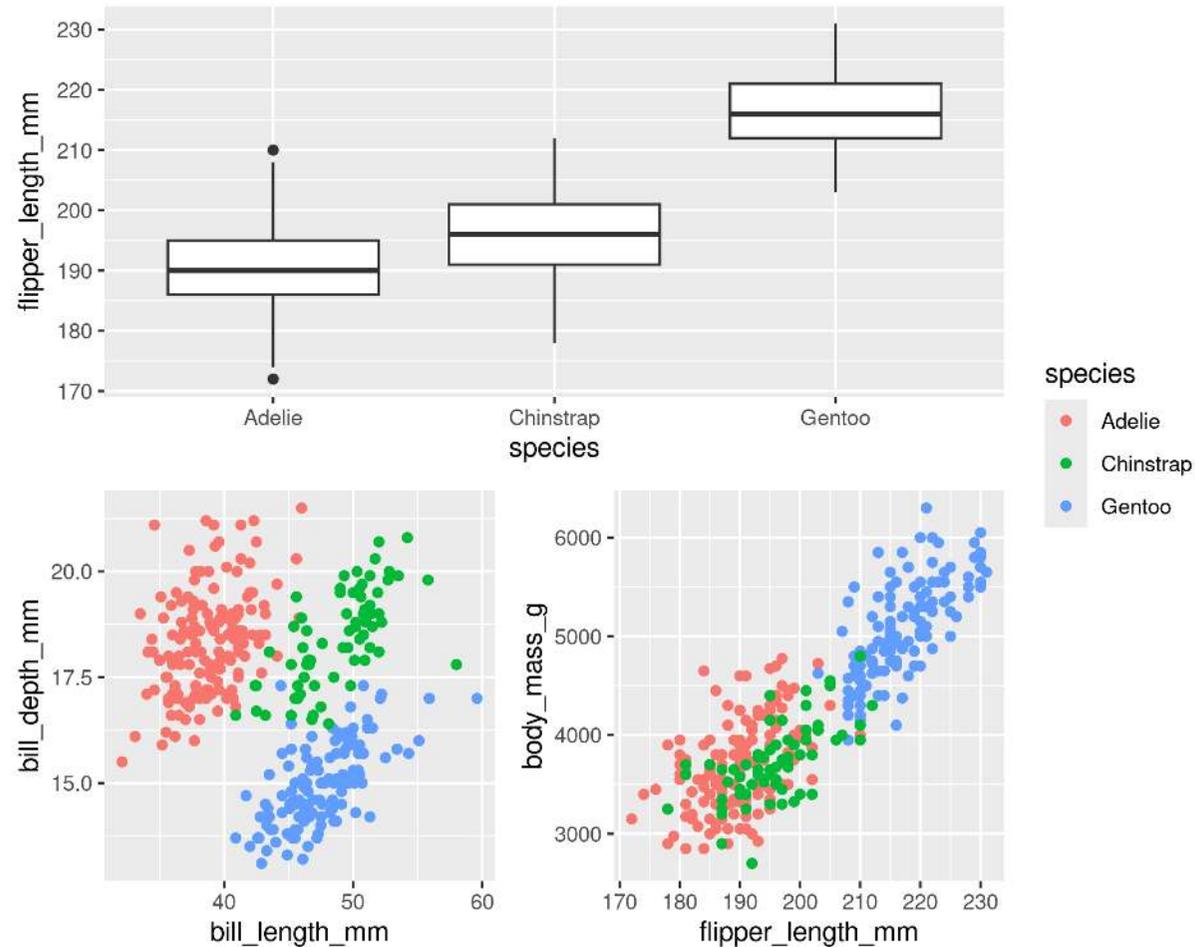
```
1 g2 / (g1 + g3)
```



Combining plots with patchwork

“collect” common legends

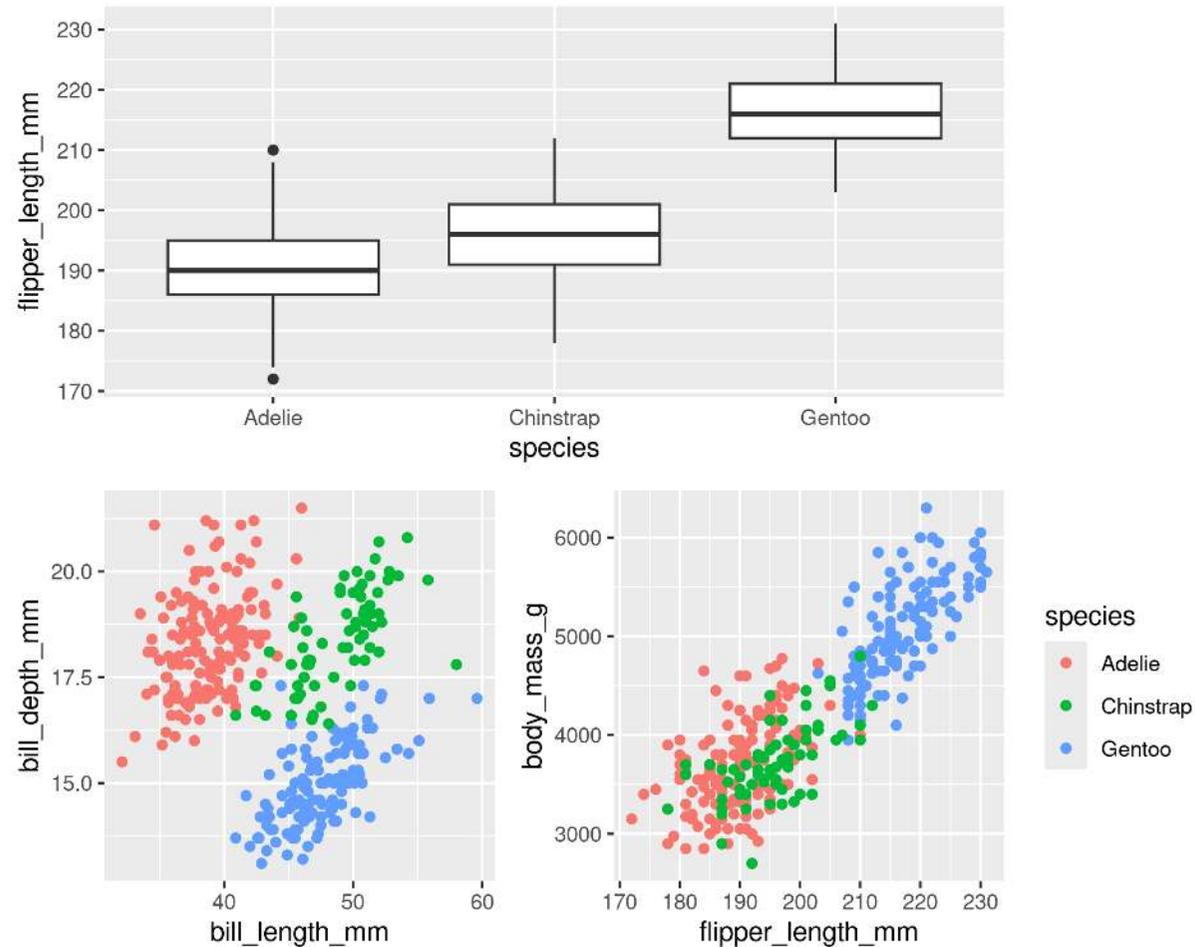
```
1 g2 / (g1 + g3) + plot_layout(guides = "collect")
```



Combining plots with patchwork

“collect” common legends

```
1 g2 / (g1 + g3 + plot_layout(guides = "collect"))
```



Combining plots with patchwork

Annotate

```
1 g2 / (g1 + g3) +  
2   plot_layout(guides = "collect") +  
3   plot_annotation(title = "Penguins Data Summary",  
4                   caption = "Fig 1. Penguins Data Summary",  
5                   tag_levels = "A",  
6                   tag_suffix = ")")
```

Penguins Data Summary

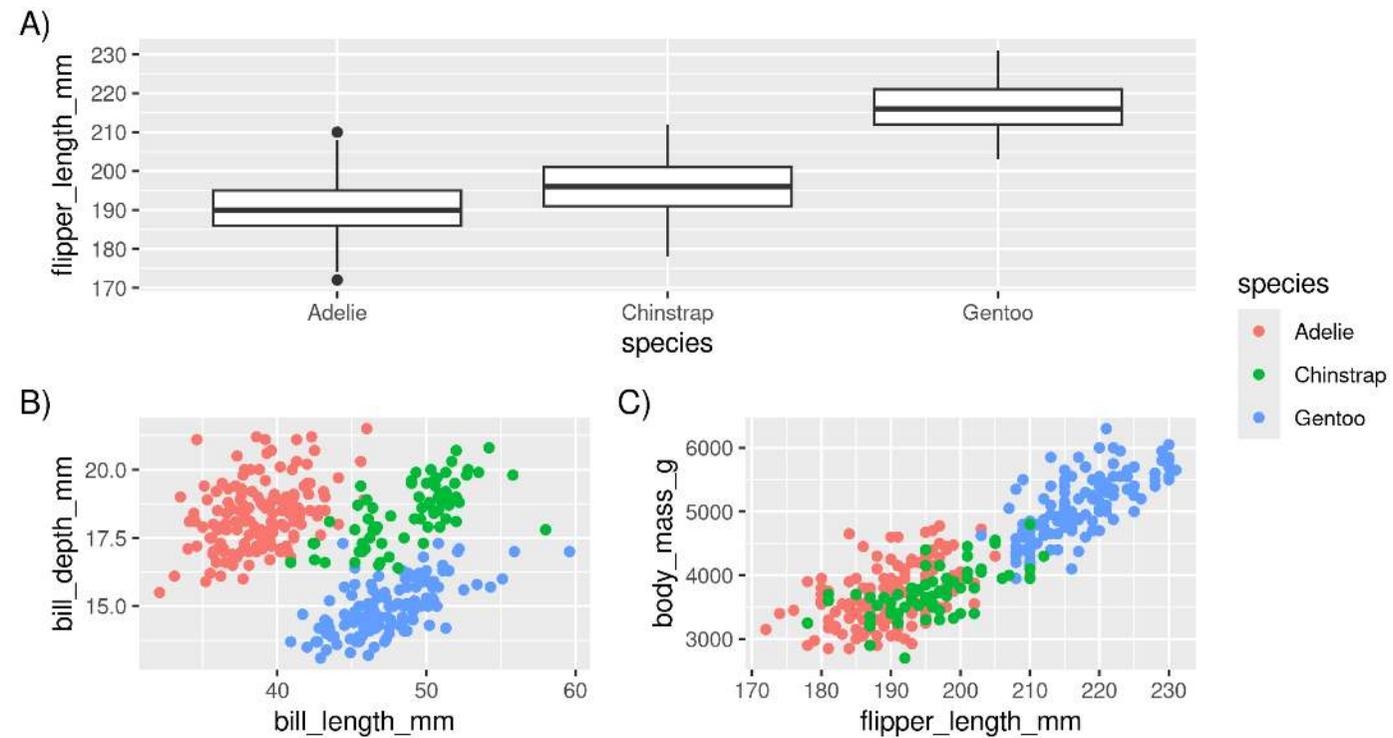


Fig 1. Penguins Data Summary

Saving plots

Saving plots

RStudio Export

Demo

ggsave()

```
1 g <- ggplot(penguins, aes(x = sex, y = bill_length_mm, fill = year)) +  
2   geom_boxplot()  
3  
4 ggsave(filename = "penguins_mass.png", plot = g)
```

Saving plots

Publication quality plots

- Many publications require ‘lossless’ (pdf, svg, eps, ps) or high quality formats (tiff, png)
- Specific sizes corresponding to columns widths
- Minimum resolutions

```
1 g <- ggplot(penguins, aes(x = sex, y = body_mass_g)) +  
2   geom_boxplot() +  
3   labs(x = "Sex", y = "Body Mass (g)") +  
4   theme(axis.text.x = element_text(angle = 45, hjust = 1))  
5  
6 ggsave(filename = "penguins_mass.pdf", plot = g, dpi = 300,  
7         height = 80, width = 129, units = "mm")
```

Wrapping up

Wrapping up: Common mistakes

- The **package** is `ggplot2`, the function is just `ggplot()`
- Did you remember to put the **+** at the **end** of the line?
- **Order matters!**
 - If you're using custom `theme()`'s, make sure you put these lines **after** bundled themes like `theme_bw()`, or they will be overwritten
- Variables like 'year' are treated as continuous, but are really categories
 - Wrap them in `factor()`
 - e.g. `ggplot(data = penguins, aes(x = factor(year), y = body_mass_g))`

Wrapping up: Further reading (all Free!)

- RStudio > Help > Cheatsheets > Data Visualization with ggplot2
- [ggplot2 book v3](#) (Hadley Wickham, Danielle Navarro, and Thomas Lin Pedersen)
- [patchwork website](#)
- [Cookbook for R](#) (Winston Chang)
- [R for Data Science](#) (Hadley Wickham & Garrett Grolemund)
 - [Chp 3 Data Visualization](#)
- [Data Visualization: A practical introduction](#) (Kieran Healy)
- [Using Special Symbols](#) (Steffi LaZerte)

Thank you!

Dr. Steffi LaZerte 
Analysis and Data Tools for Science

 steffilazerte.ca

 sel@steffilazerte.ca

Slides created with Quarto on 2024-09-25

Extra

Your Turn!

Create a figure with...

- Custom colour mapping (i.e. `scales_...`)
- Clear, human-readable labels
- More than one graph, each one tagged (e.g., A) or B))
- With more than one geom type
- At least one scatterplot with regression line



OR... Load your own data and create a figure of your own!