

# Visualizing Data in R

## A primer on `ggplot2`

 [steffilazerte](#)

 [@steffilazerte@fosstodon.org](mailto:@steffilazerte@fosstodon.org)

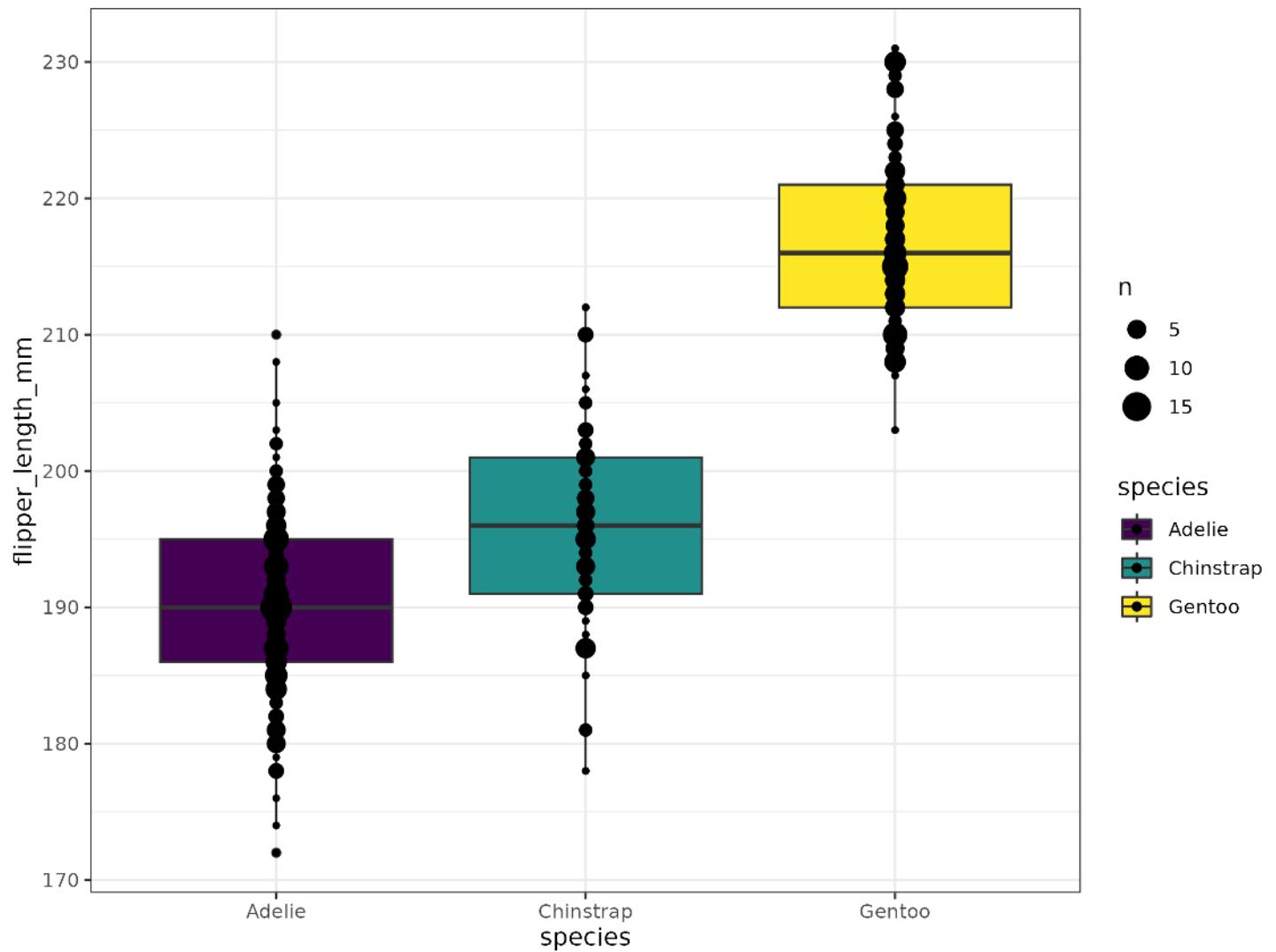
 [@steffilazerte](#)

 [steffilazerte.ca](#)

Dr. Steffi LaZerte



Analysis and Data Tools for Science



Compiled: 2023-04-19

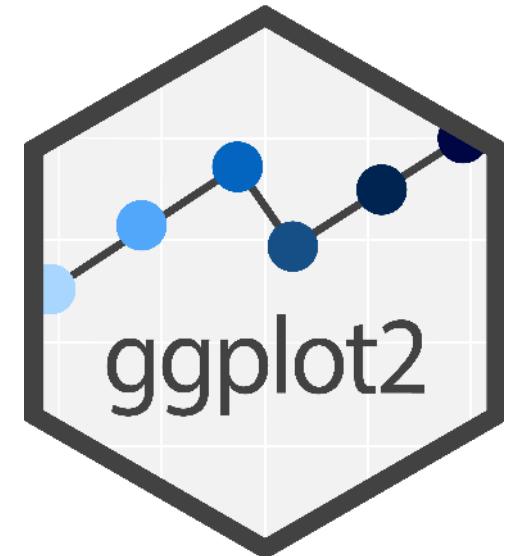
# First things first

- ⌚ Save previous script
- 📁 Open New File
  - (make sure you're in the RStudio Project)
- ✍ Write `library(tidyverse)` at the top
- ⌚ Save this new script
  - (consider names like `figs.R` or `2_figures.R`)

# Outline

## 1. Figures with `ggplot2` (A `tidyverse` package)

- Basic plot
- Common plot types
- Plotting by categories
- Adding statistics
- Customizing plots
- Annotating plots



## 2. Combining figures with `patchwork`

## 3. Saving figures

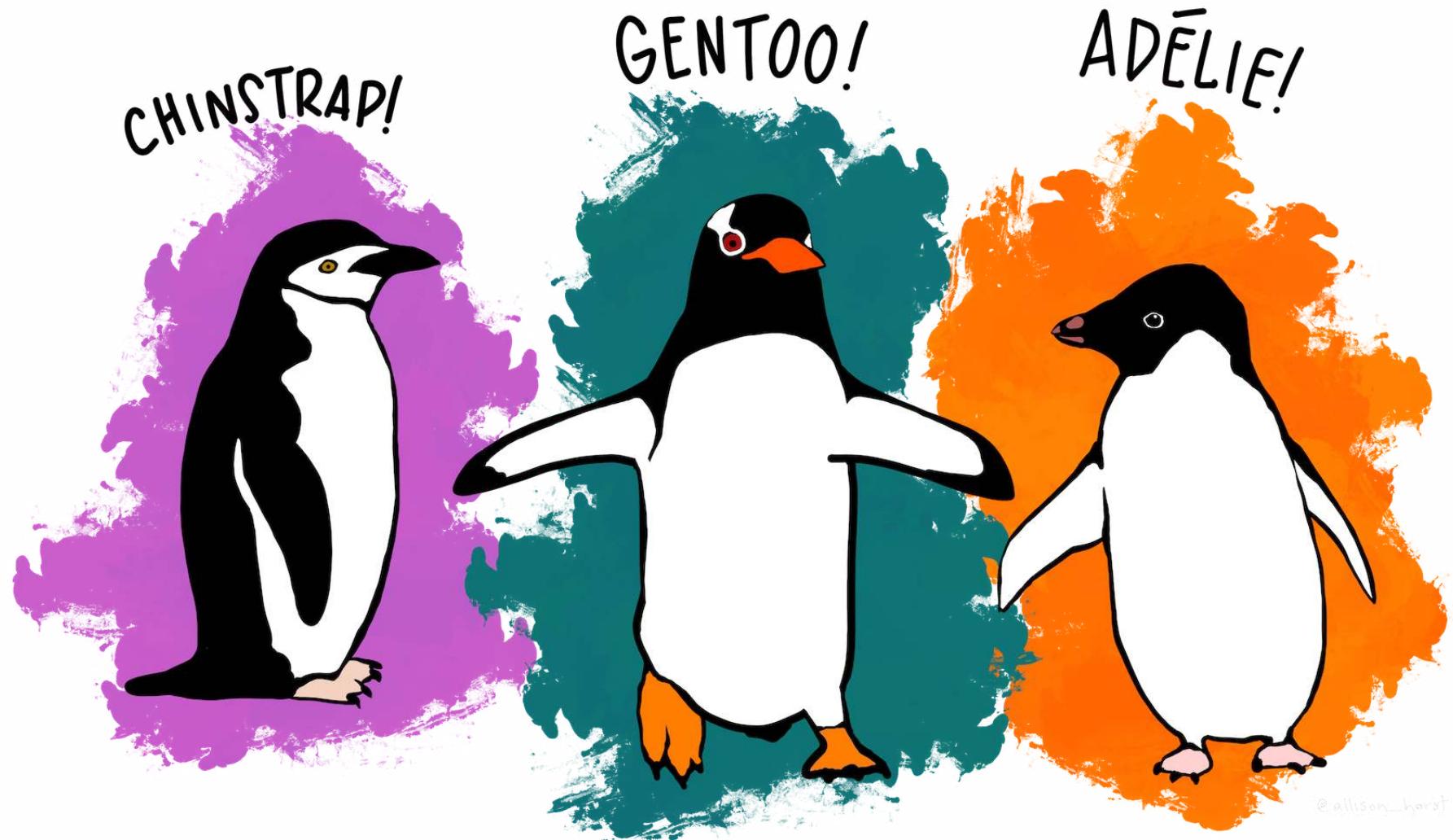
# ggplot2:

Build a data  
**MASTERPIECE**



HORST '18

# Our data set: Palmer Penguins!



# 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
  <fct>   <fct>        <dbl>        <dbl>          <int>        <int>
1 Adelie   Torgersen     39.1         18.7          181       3750
2 Adelie   Torgersen     39.5         17.4          186       3800
3 Adelie   Torgersen     40.3          18           195       3250
4 Adelie   Torgersen      NA           NA            NA        NA
5 Adelie   Torgersen     36.7         19.3          193       3450
6 Adelie   Torgersen     39.3         20.6          190       3650
7 Adelie   Torgersen     38.9         17.8          181       3625
8 Adelie   Torgersen     39.2         19.6          195       4675
9 Adelie   Torgersen     34.1         18.1          193       3475
10 Adelie  Torgersen      42           20.2          190       4250
# i 334 more rows
# i 2 more variables: sex <fct>, year <int>
```



Your turn!

Run this code and look at the output in the console

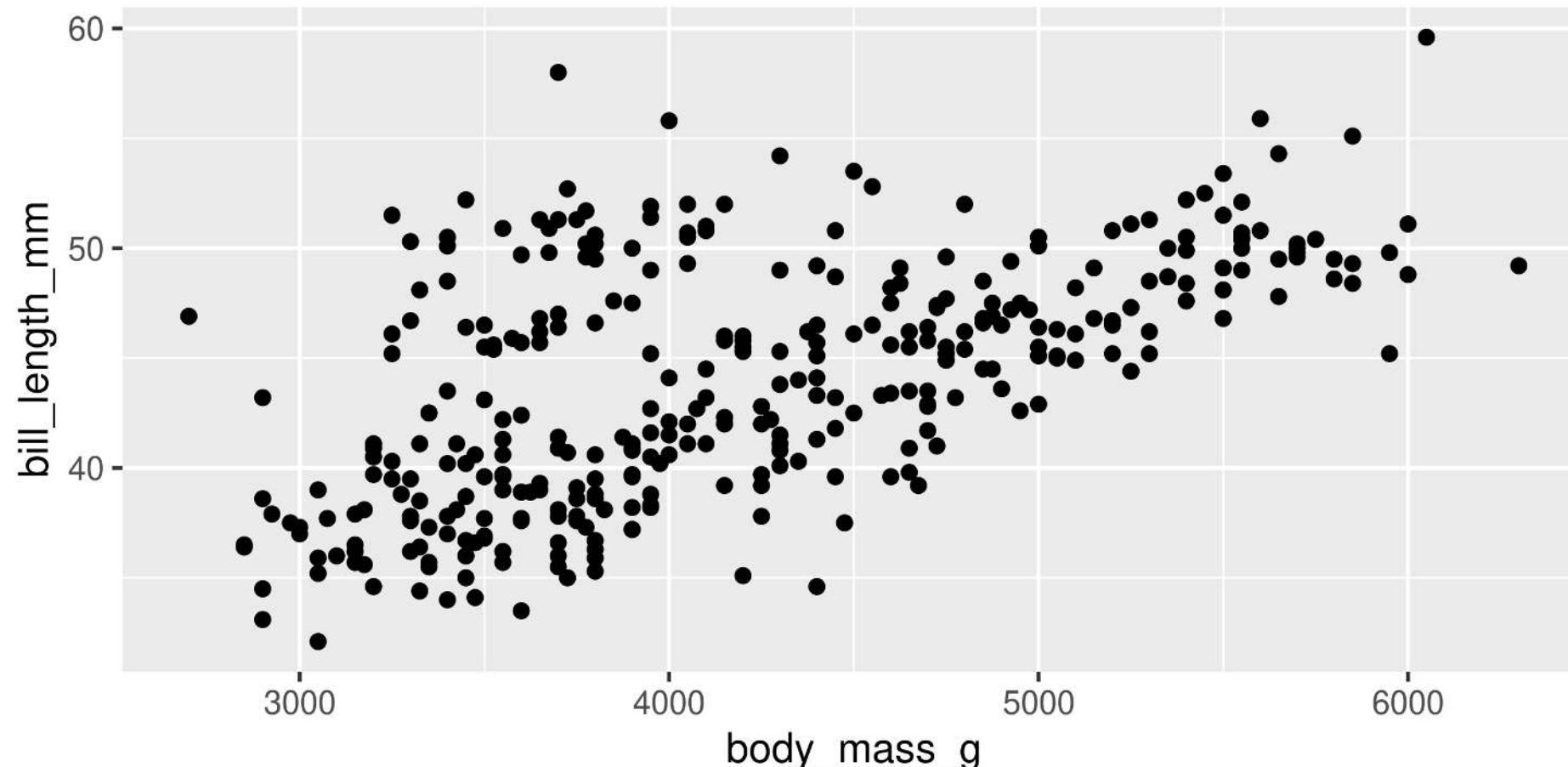
# Side Note

## Where did the **penguins** data set come from?

- Sometimes R packages contain data
- If you load a package (i.e. `library(palmerpenguins)`) you can use the data
- **Note** that here the data object is called **penguins** (not **palmerpenguins**)
- **Note** this is NOT how you'll load your own data

# A basic plot

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

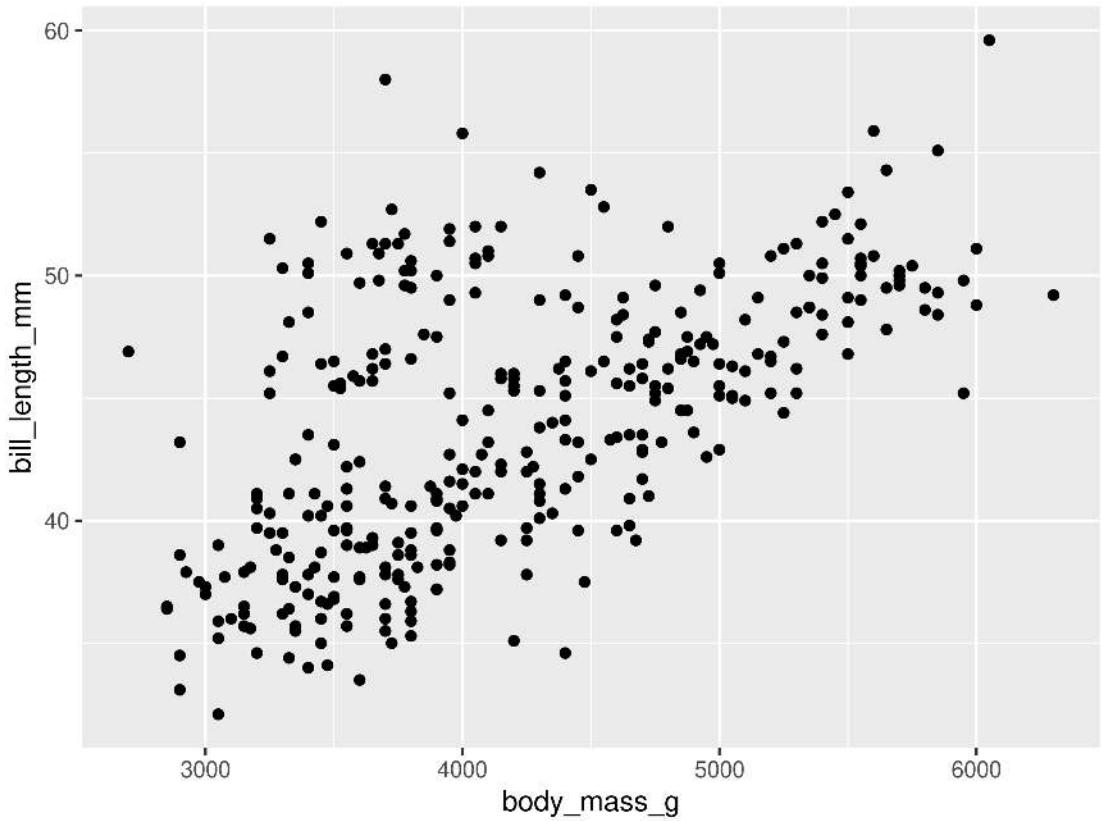


# Break it down

```
1 library(palmerpenguins)
2 library(tidyverse)
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

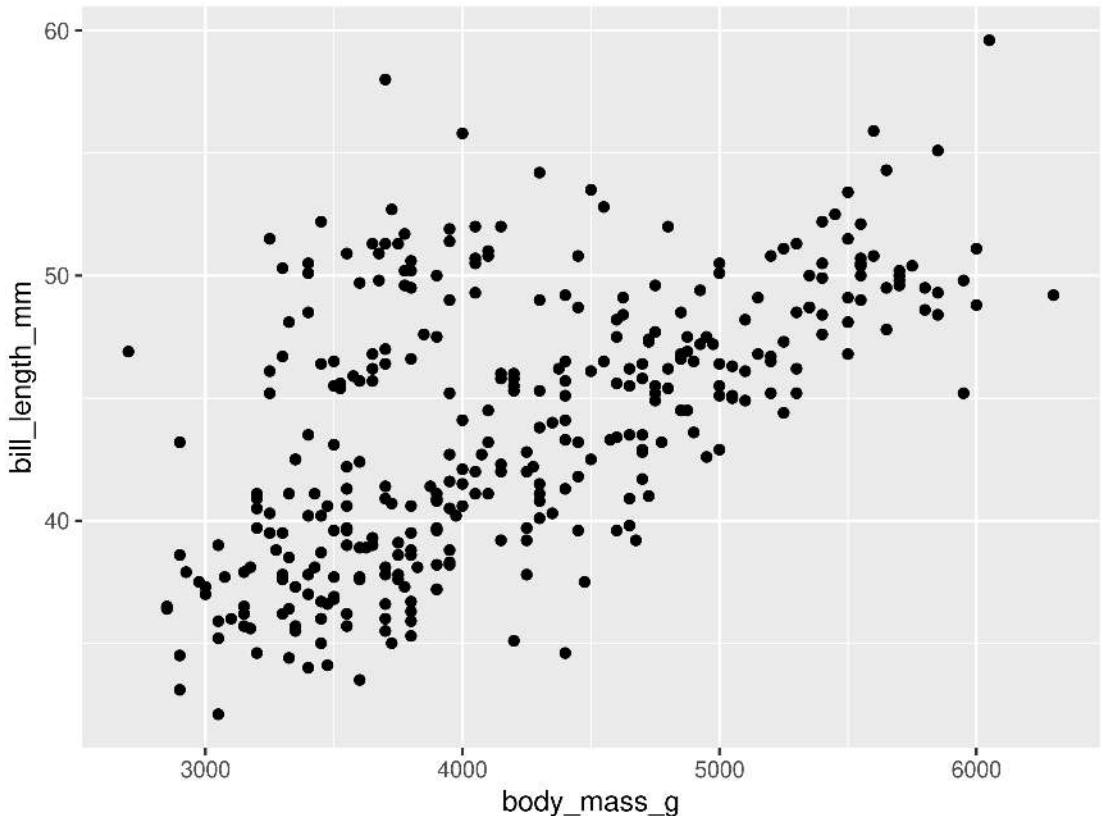


# Break it down

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

## library(ggplot2)

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

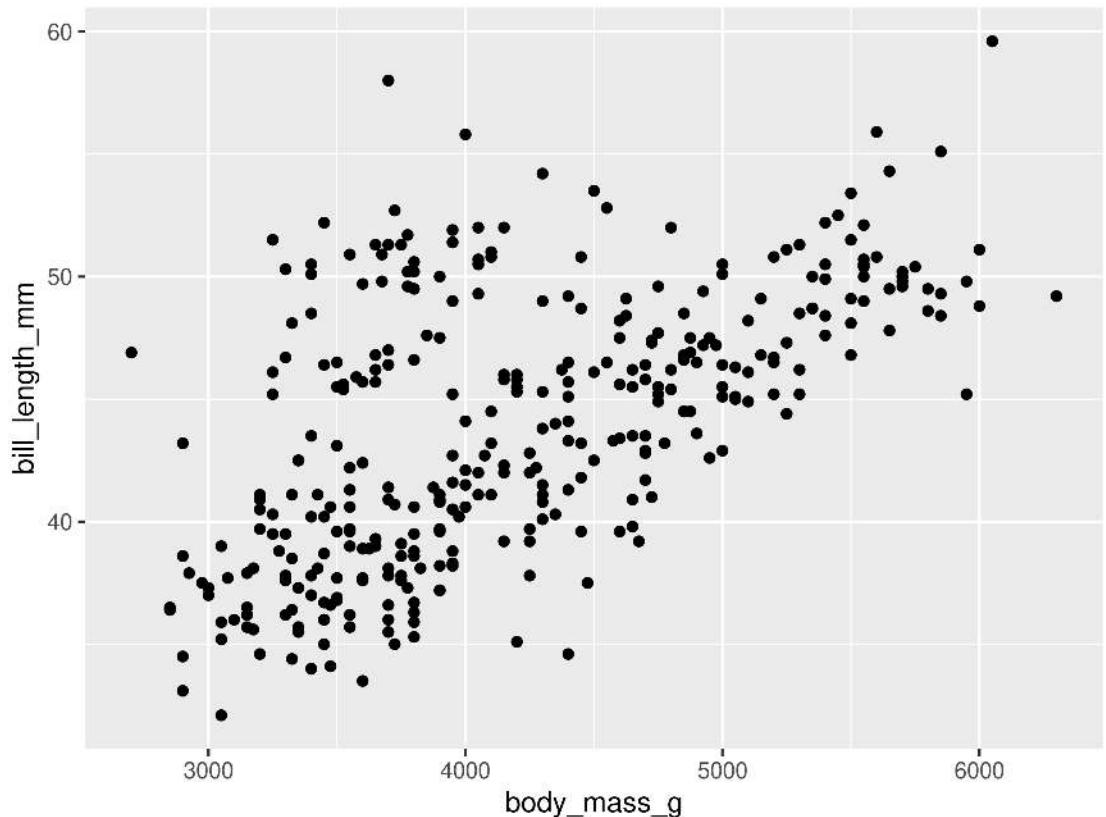


# Break it down

```
1 library(palmerpenguins)
2 library(tidyverse)
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



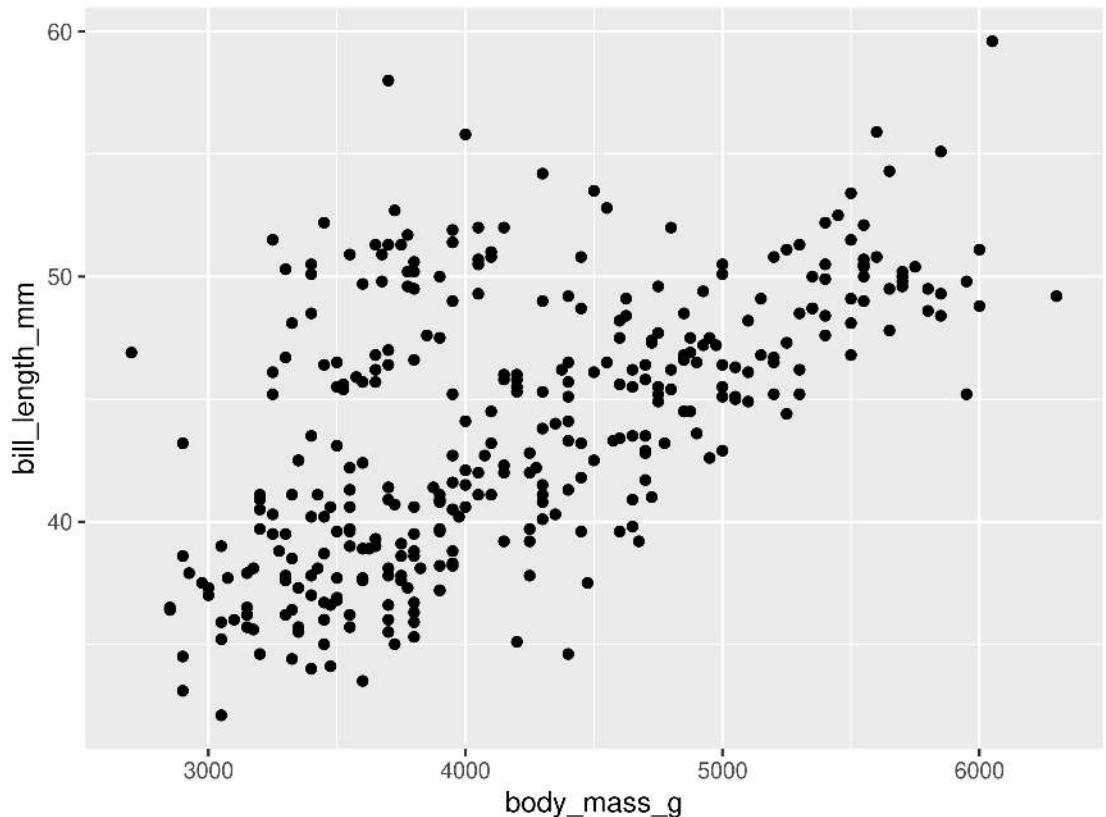
# Break it down

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

## geom\_point()

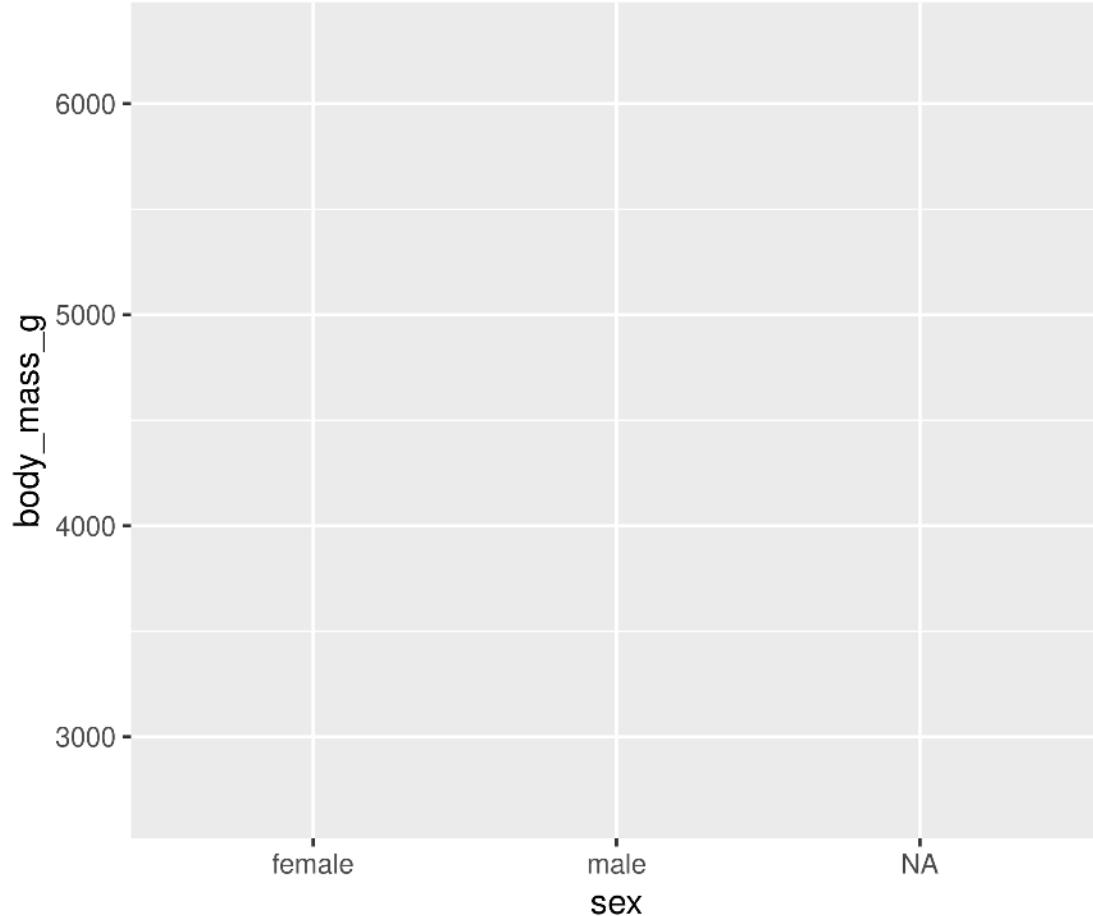
- Choose a **geom** function to display the data
- Always *added* to a **ggplot()** call with **+**

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

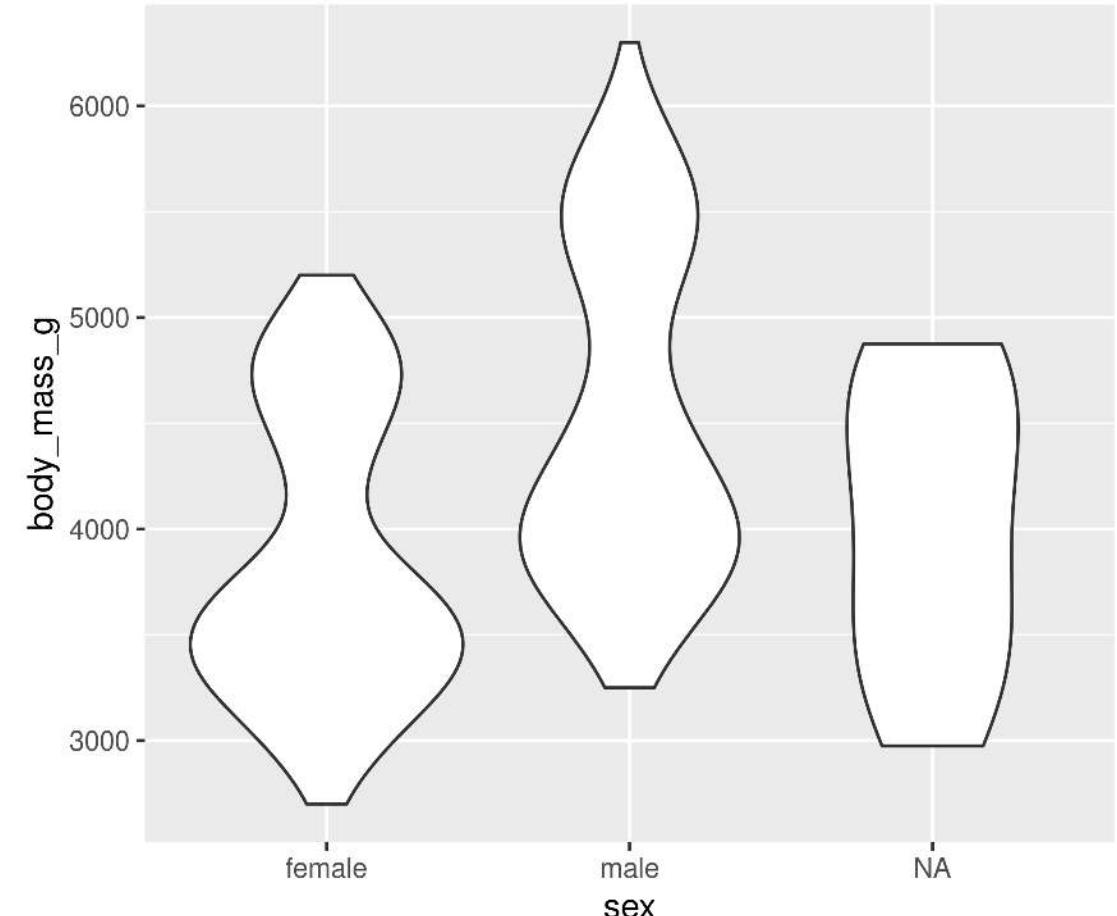


# Plots are layered

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



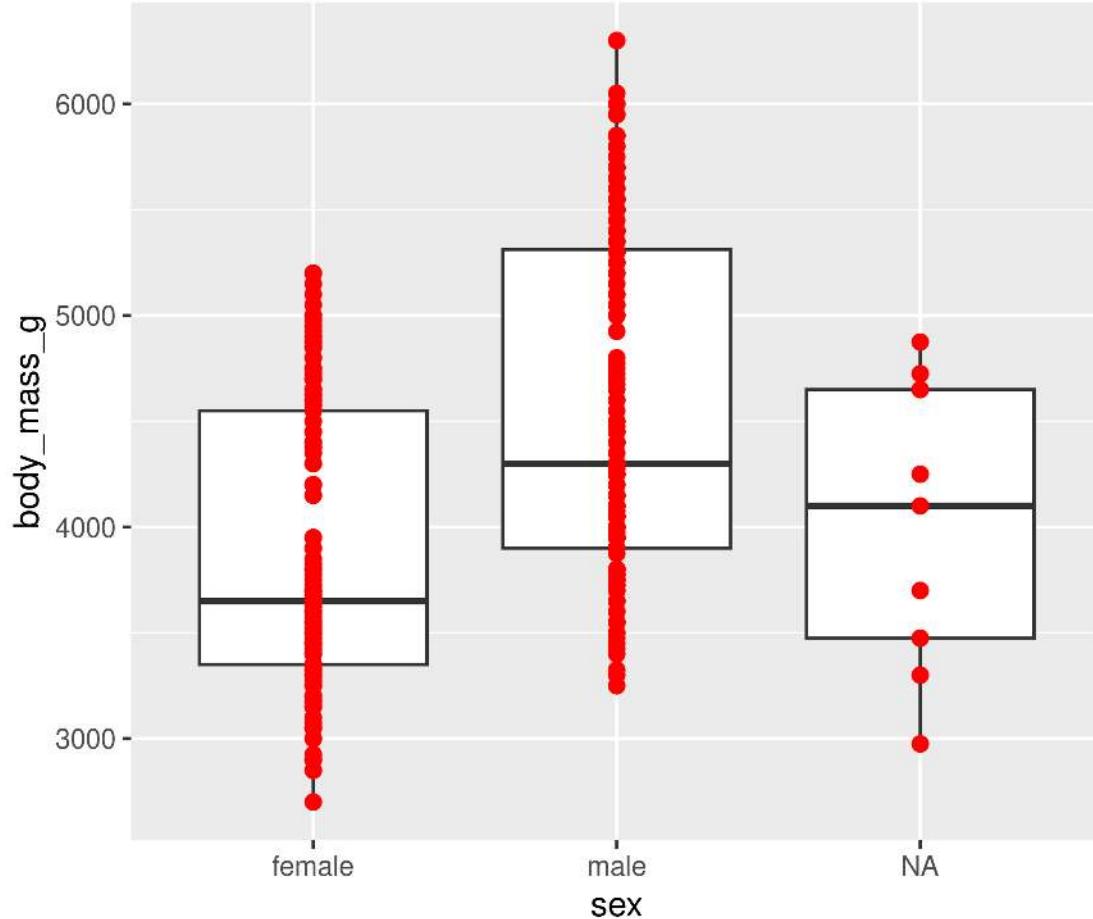
```
1 ggplot(data = penguins, aes(x = sex, y = body_mass_g)) +  
2   geom_violin()
```



# Plots are layered

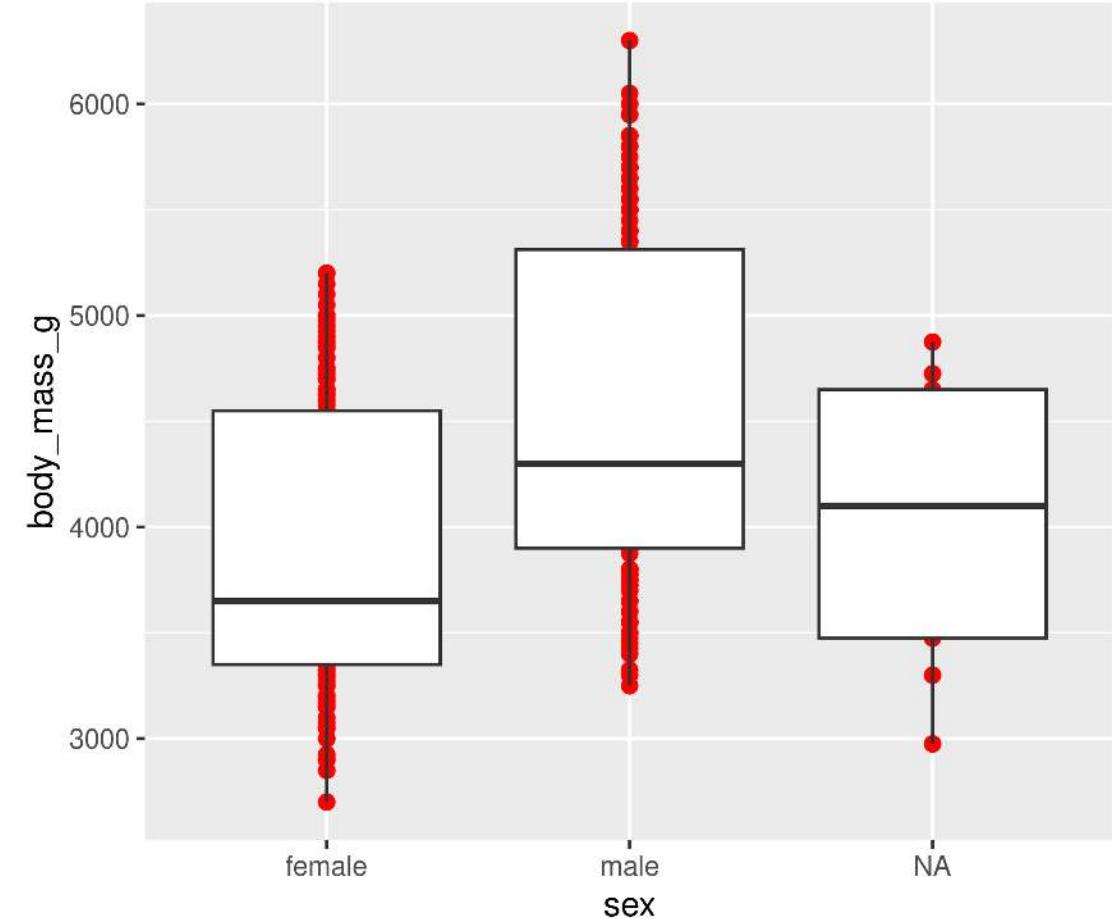
# You can add multiple layers

```
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()
```

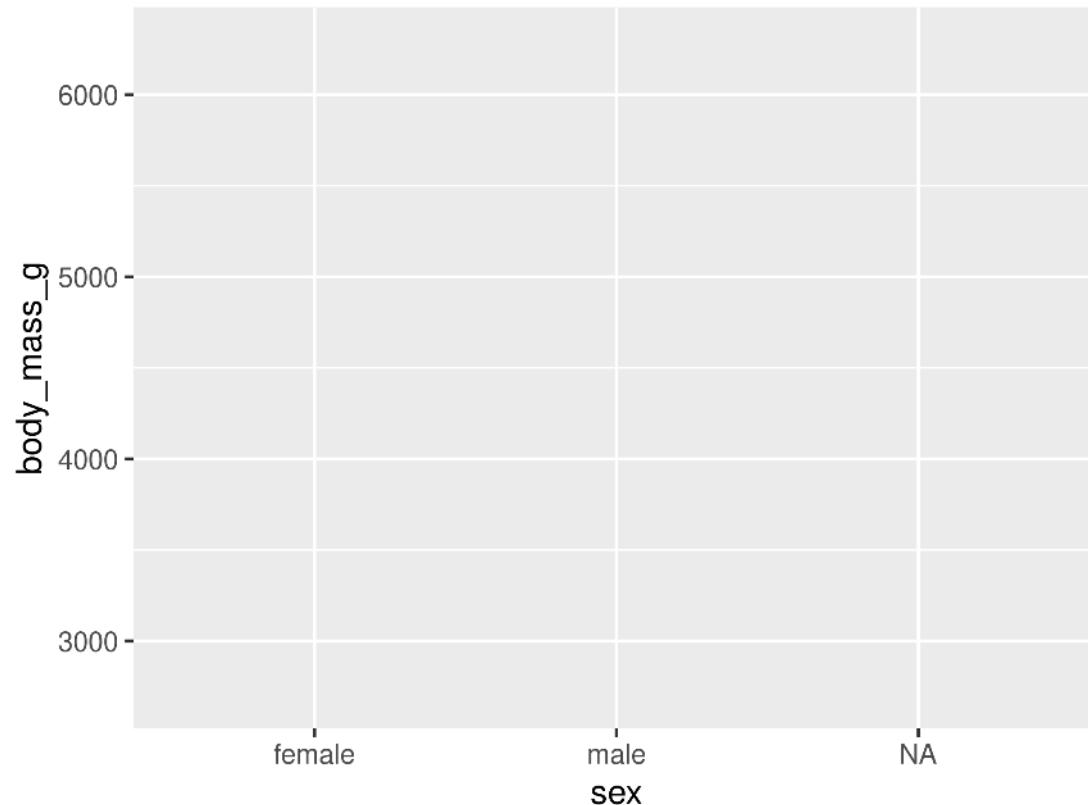


# Plots are objects

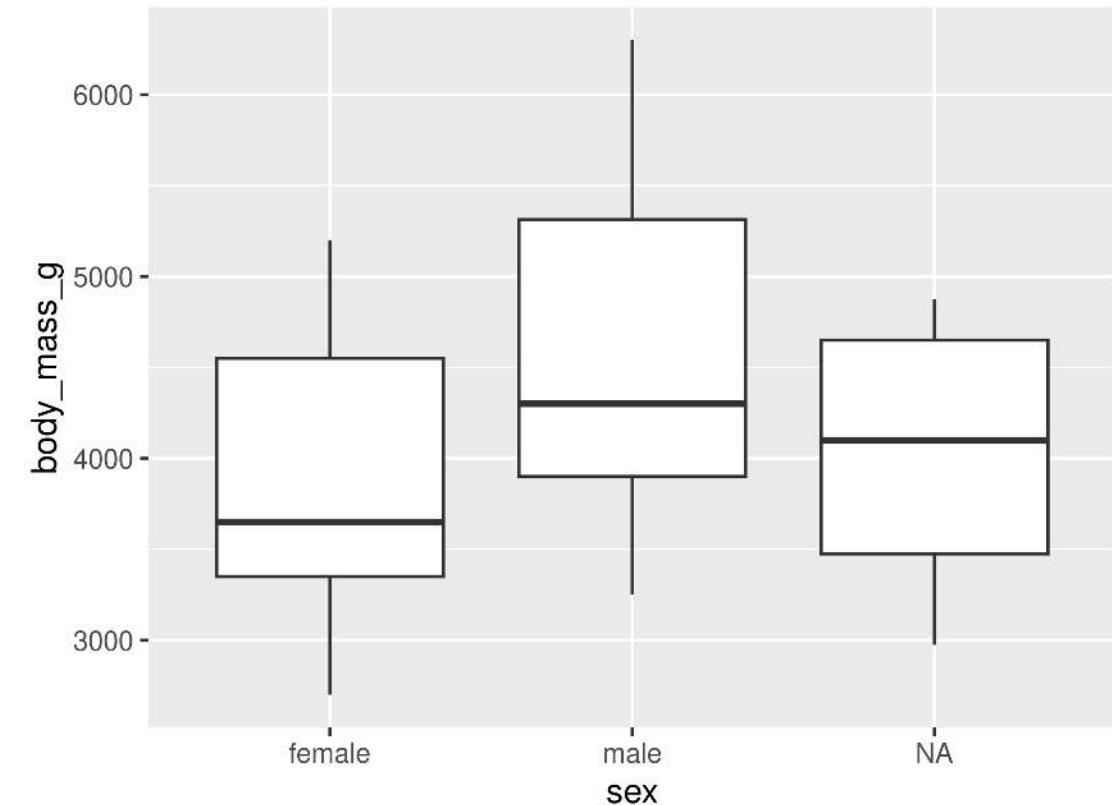
Any ggplot can be saved as an object

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

```
1 g
```



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

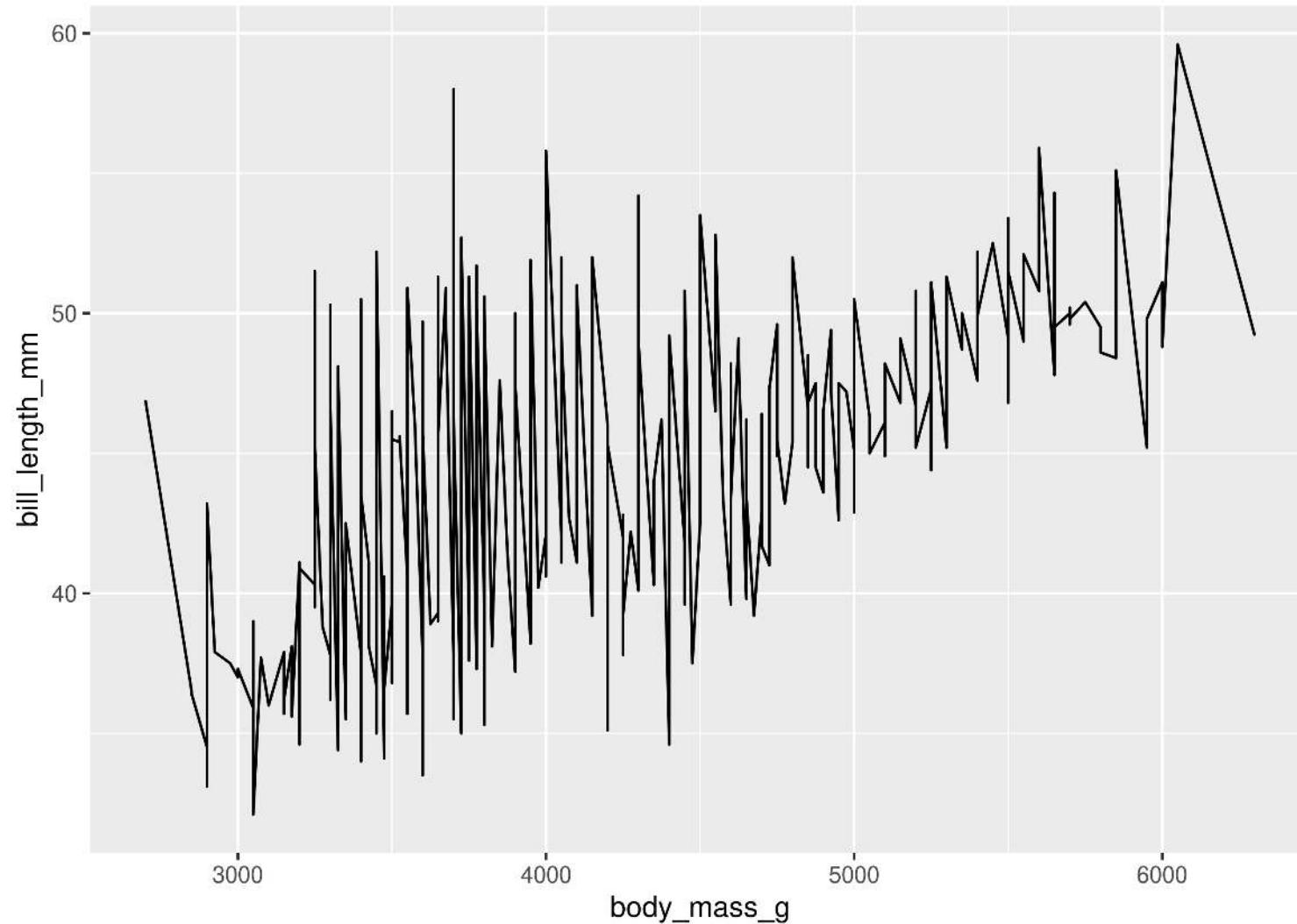


# **More Geoms**

(Plot types)

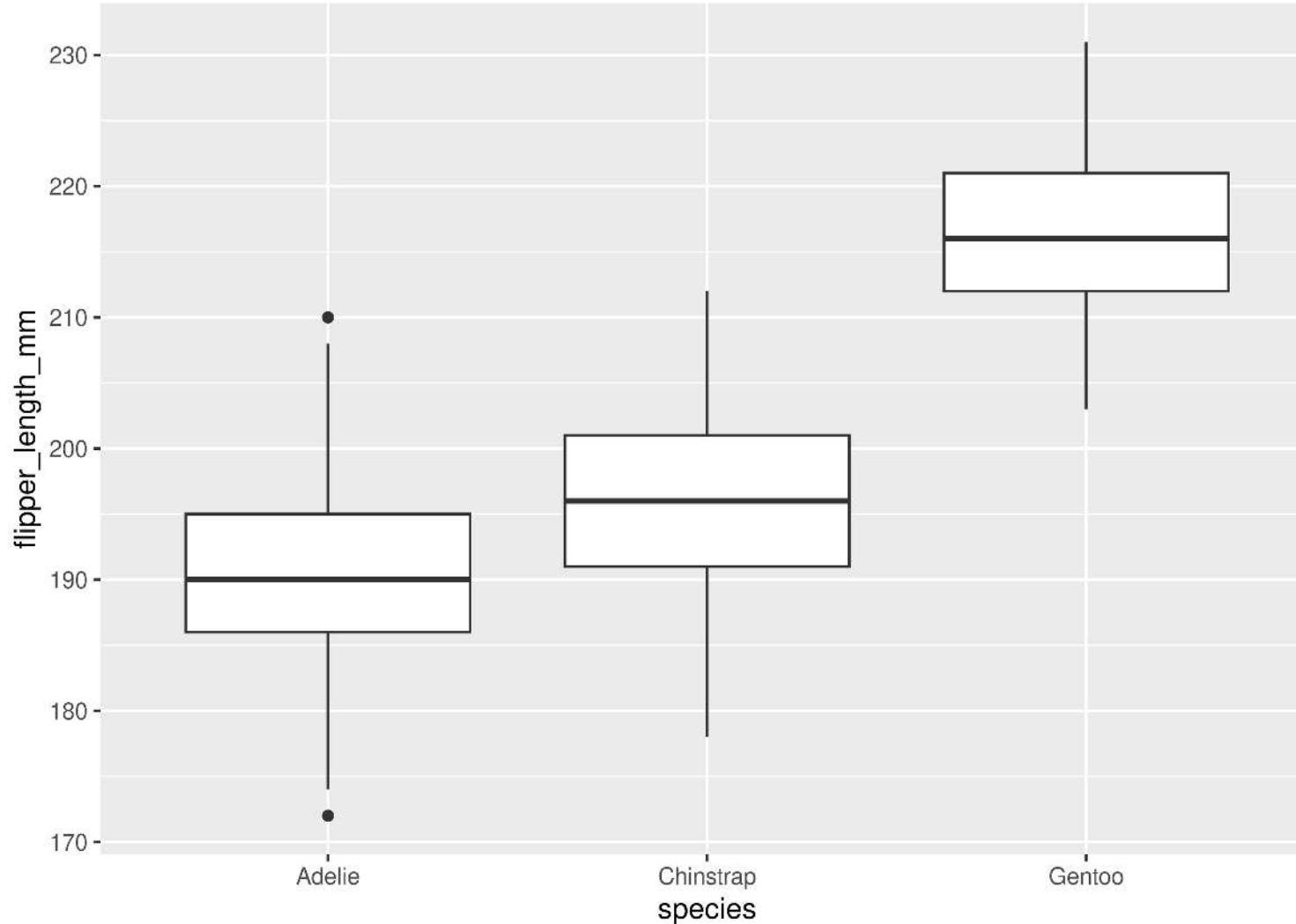
# Geoms: Lines

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



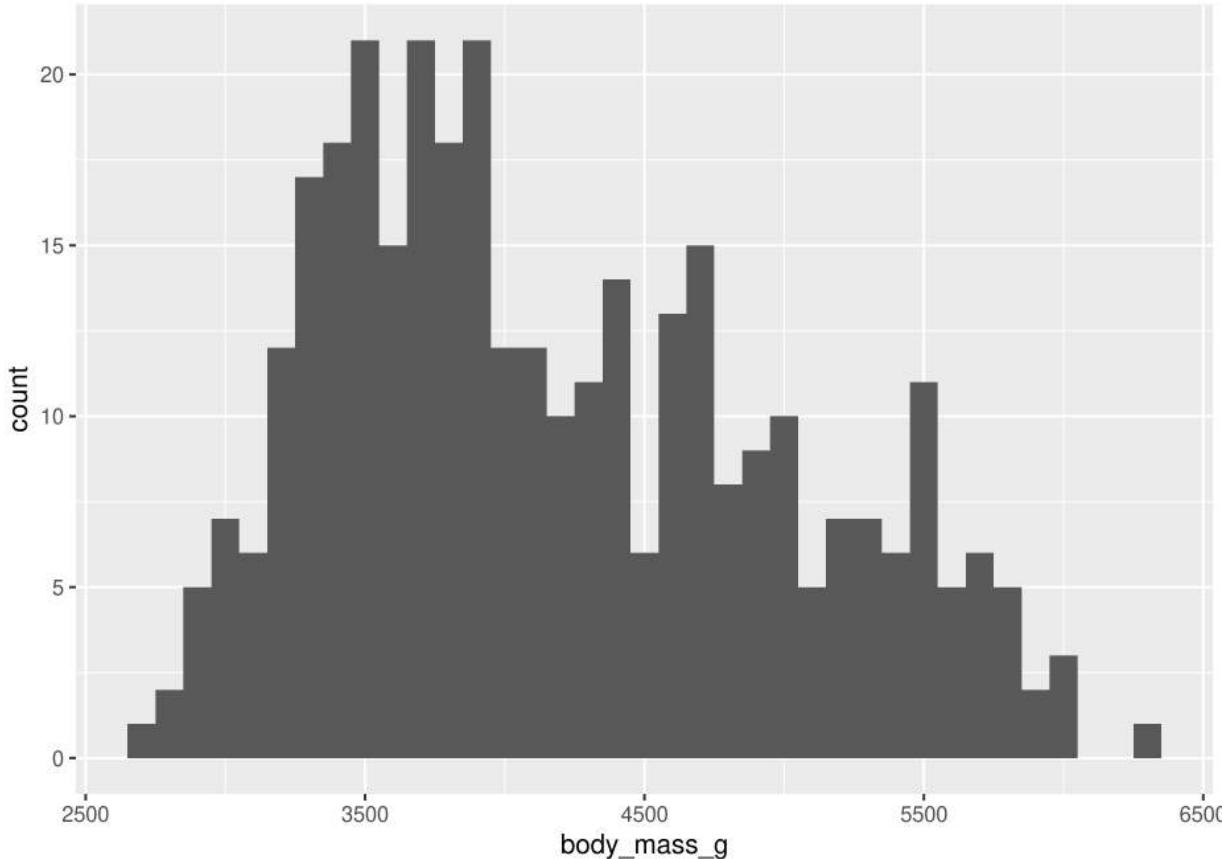
# Geoms: Boxplots

```
1 ggplot(data = penguins, aes(x = species, y = flipper_length_mm)) +  
2   geom_boxplot()
```



# Geoms: Histogram

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

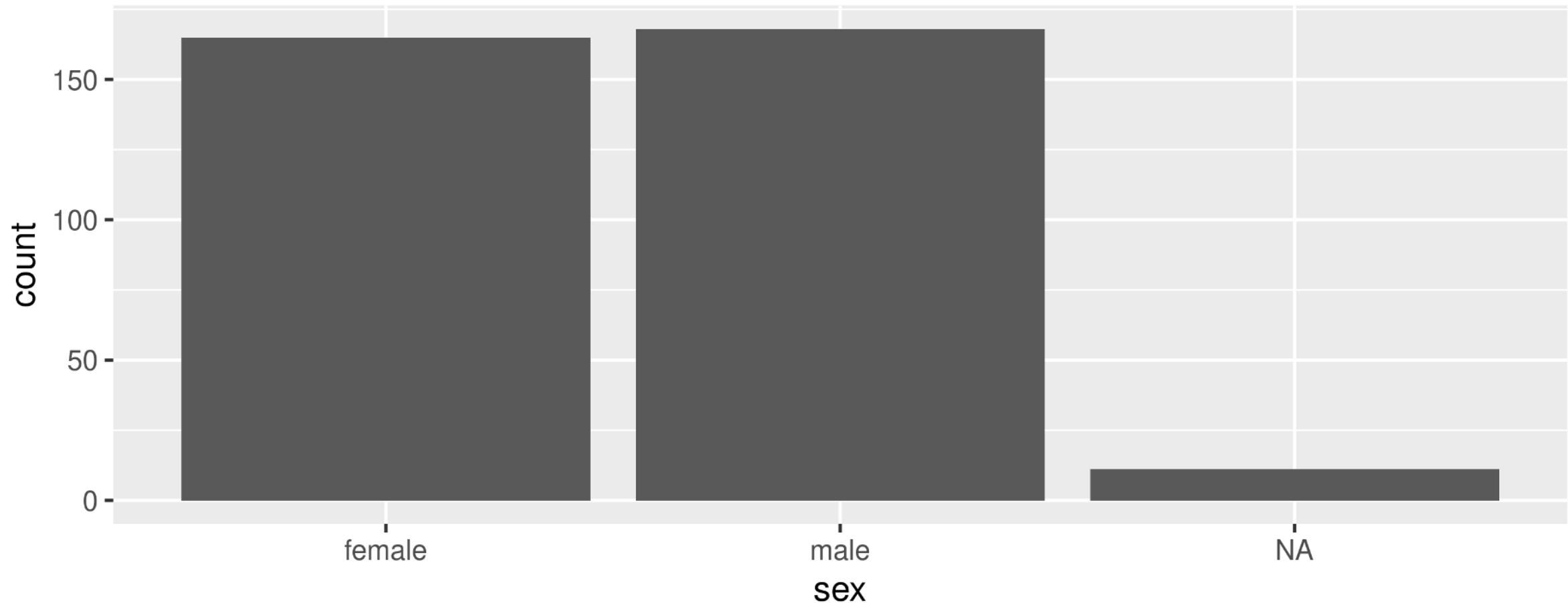


Note: We only need 1 aesthetic here

# Geoms: Barplots

Let `ggplot` count your data

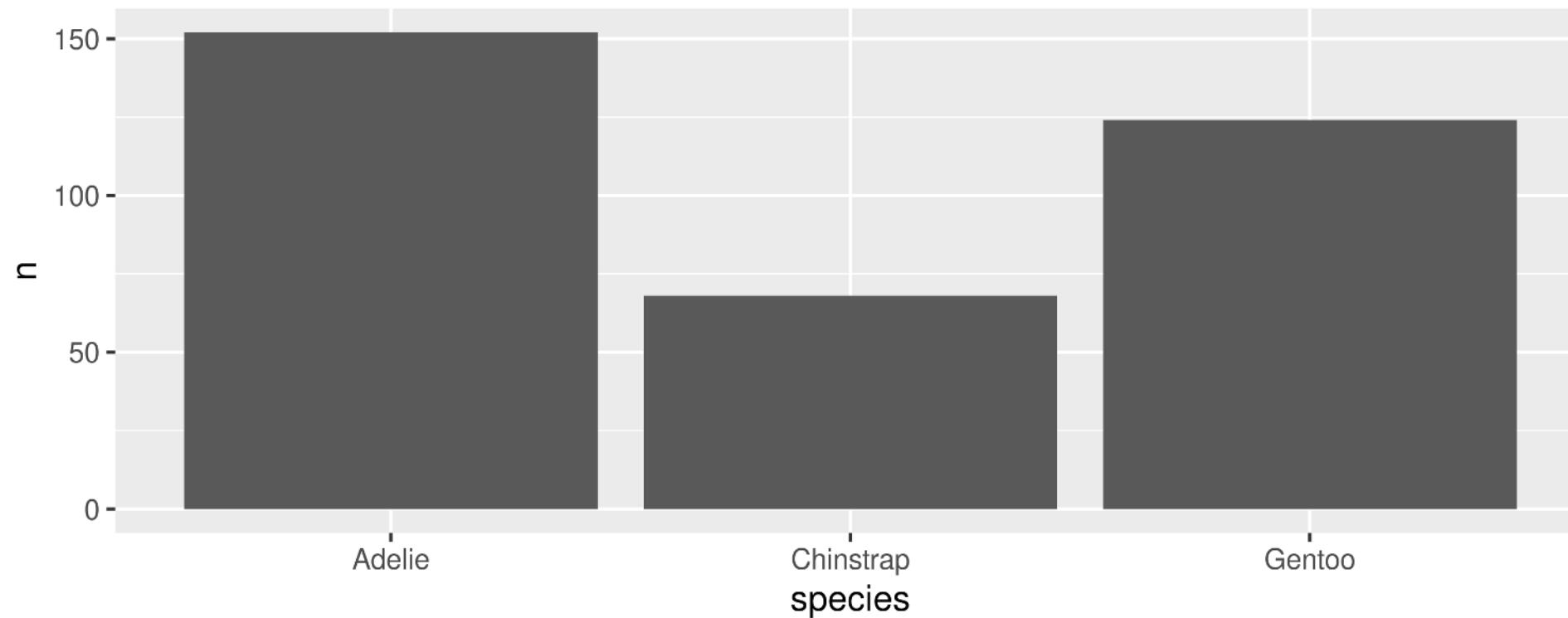
```
1 ggplot(data = penguins, aes(x = sex)) +  
2   geom_bar()
```



# Geoms: Barplots

You can also provide the counts

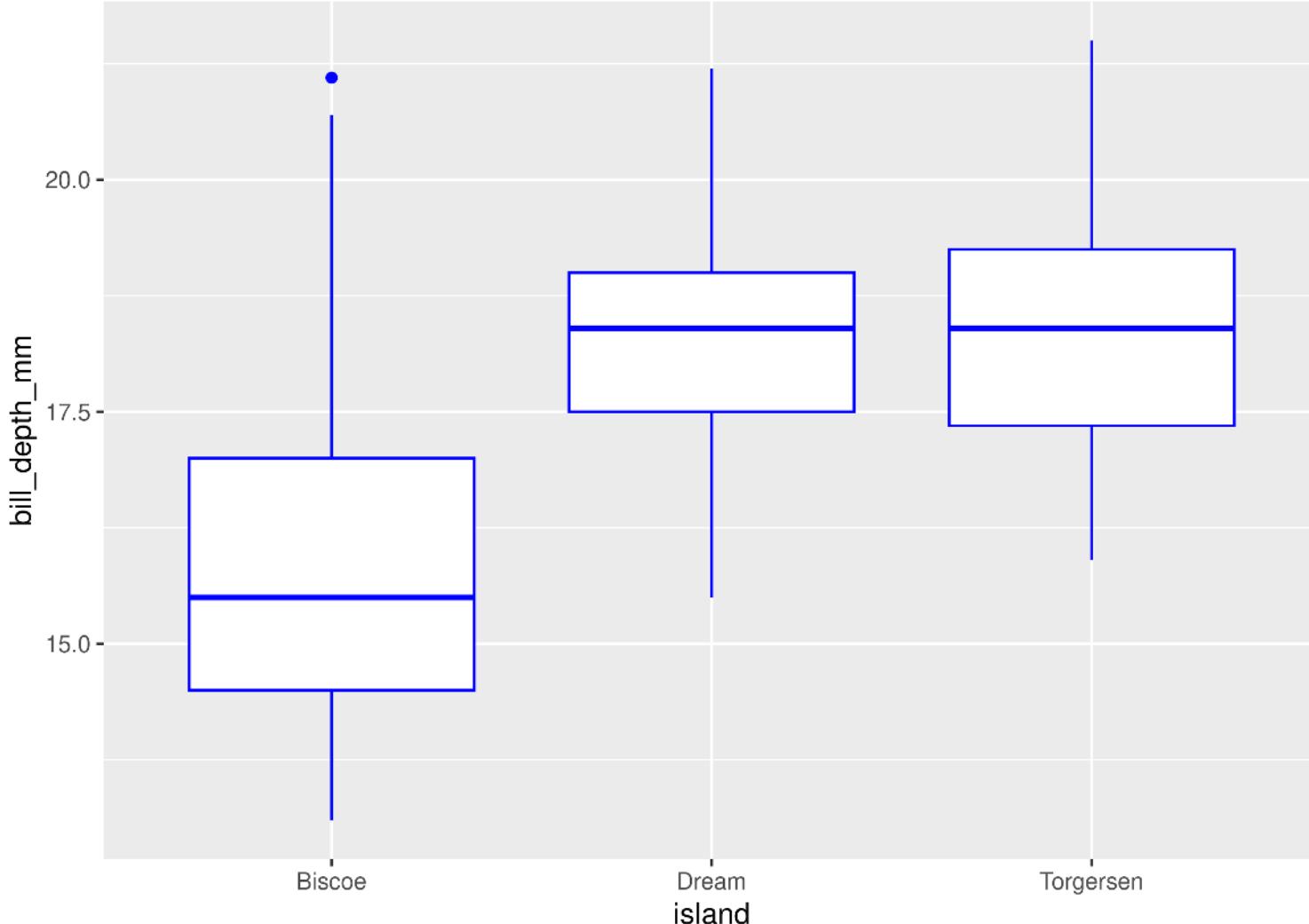
```
1 # Create our own data frame
2 species_counts <- data.frame(species = c("Adelie", "Chinstrap", "Gentoo"),
3                                n = c(152, 68, 124))
4
5 ggplot(data = species_counts, aes(x = species, y = n)) +
6   geom_bar(stat = "identity")
```



# Your Turn: Create this plot

```
1 ggplot(data = ___, aes(x = ___, y = ___)) +  
2   geom___(____)
```

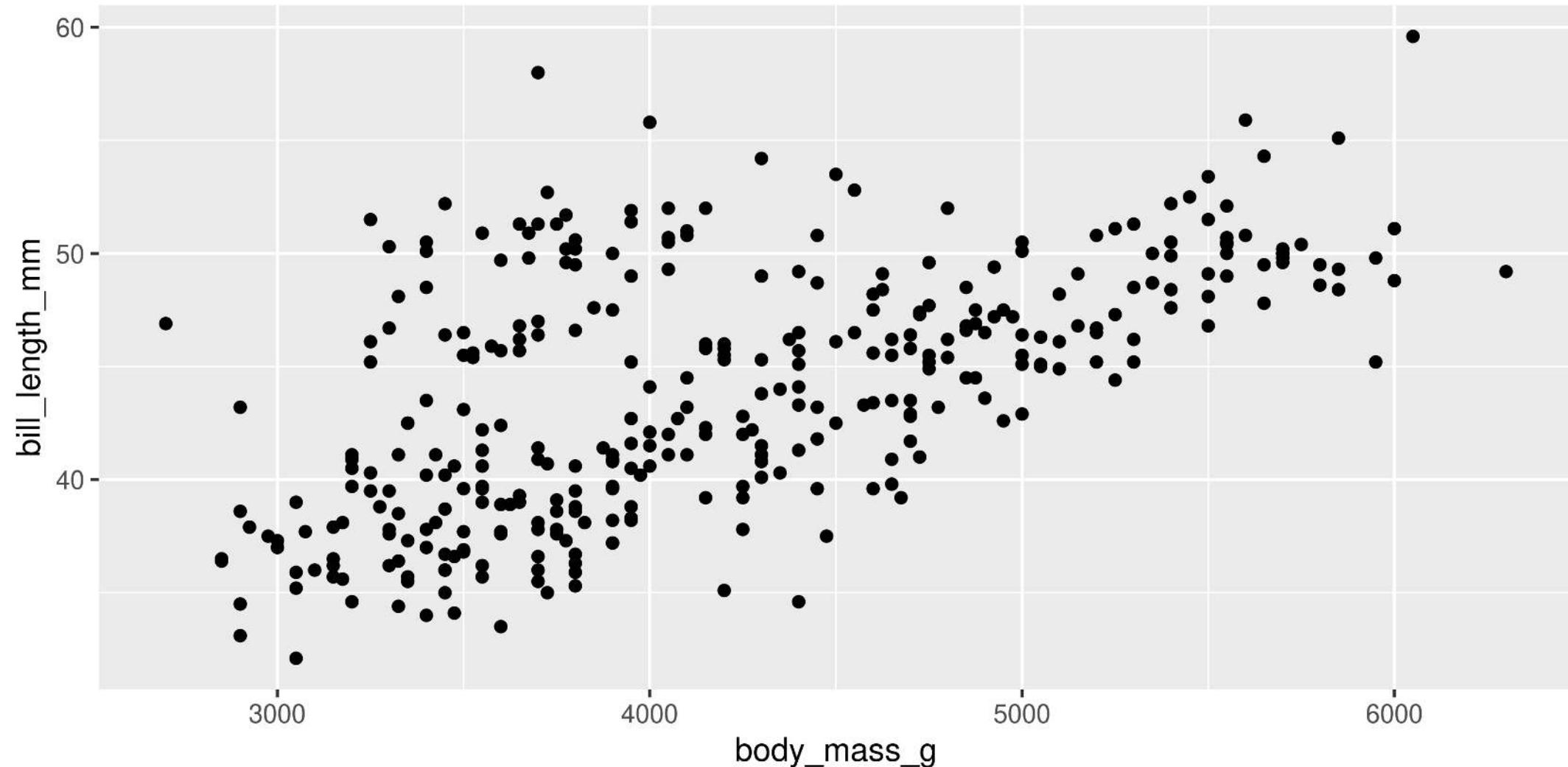
Too Easy?  
Plot points on top



# **Show**ing** data by group**

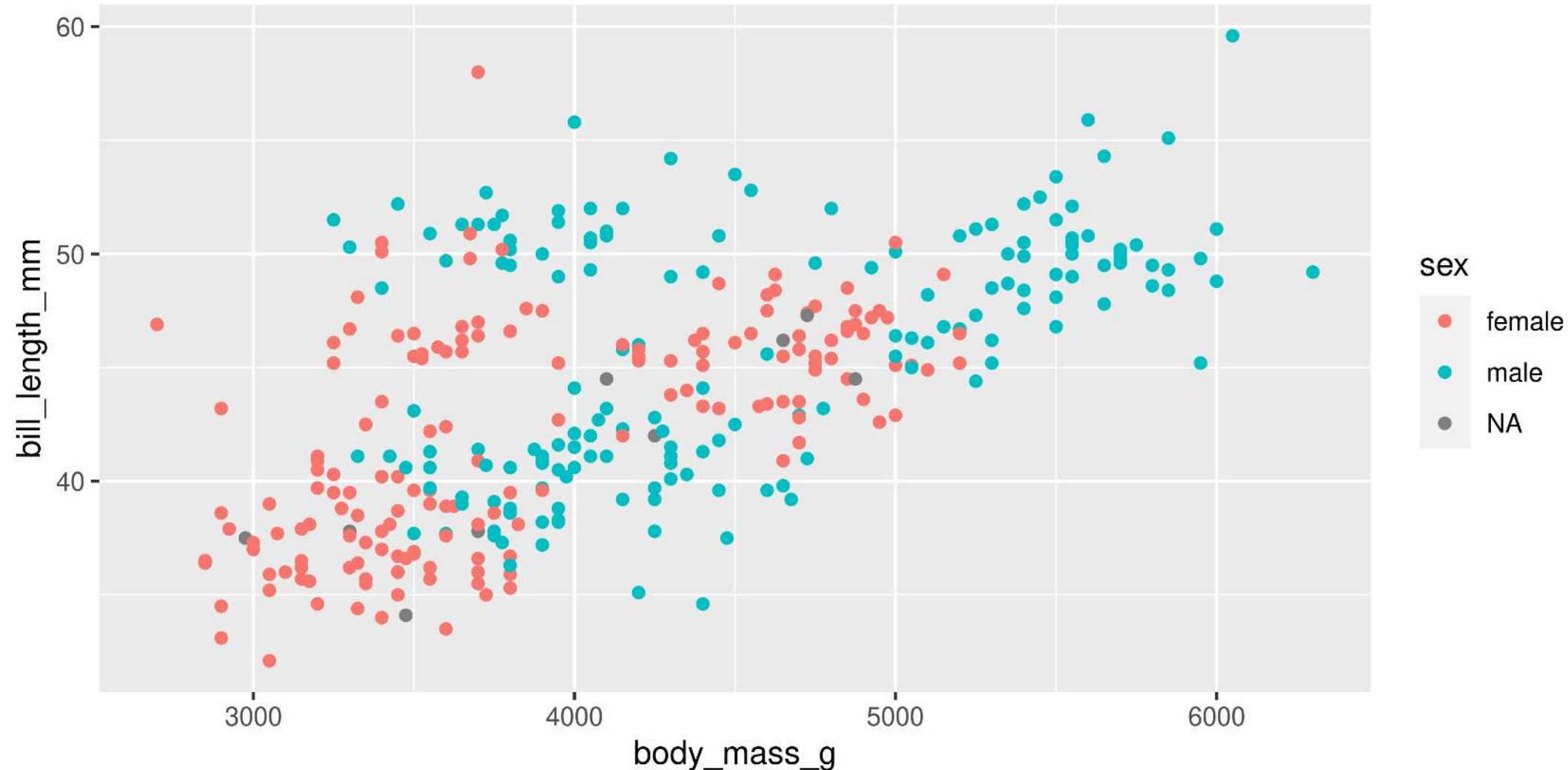
# Mapping aesthetics

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



# Mapping aesthetics

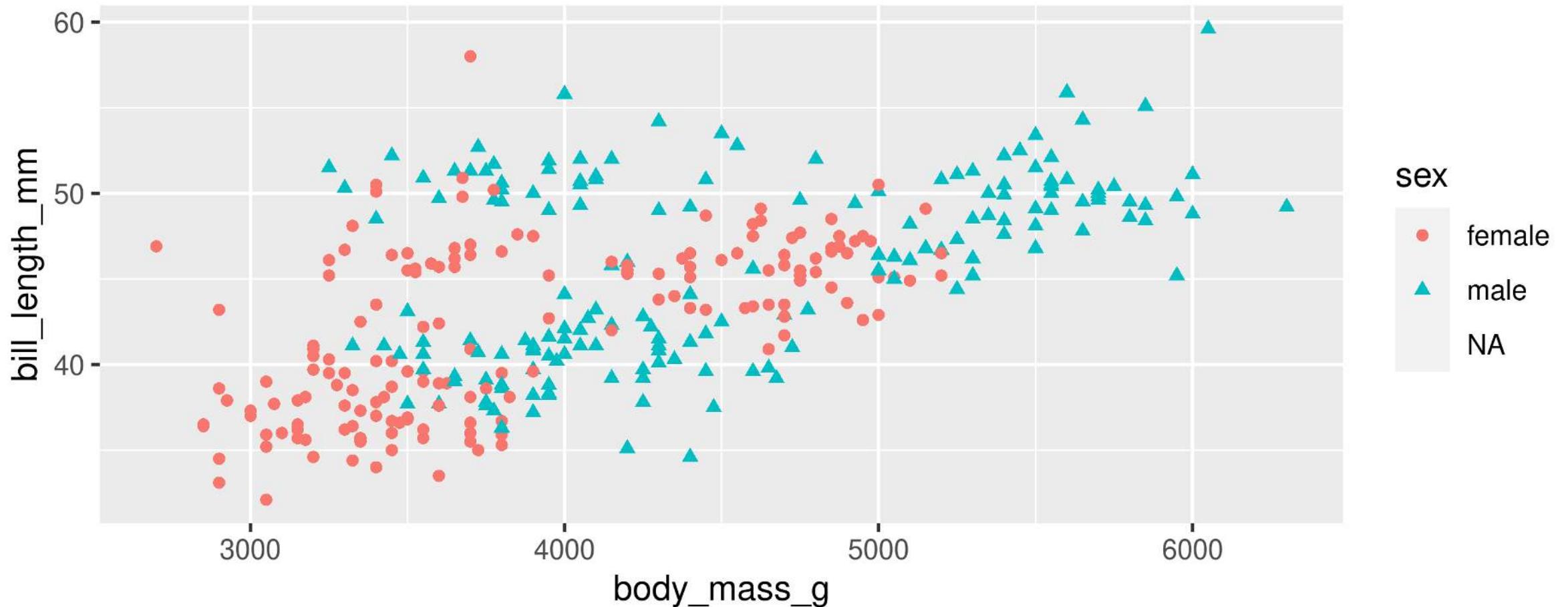
```
1 ggplot(data = penguins, aes(x = body_mass_g, y = bill_length_mm, colour = sex)) +  
2   geom_point()
```



# Mapping aesthetics

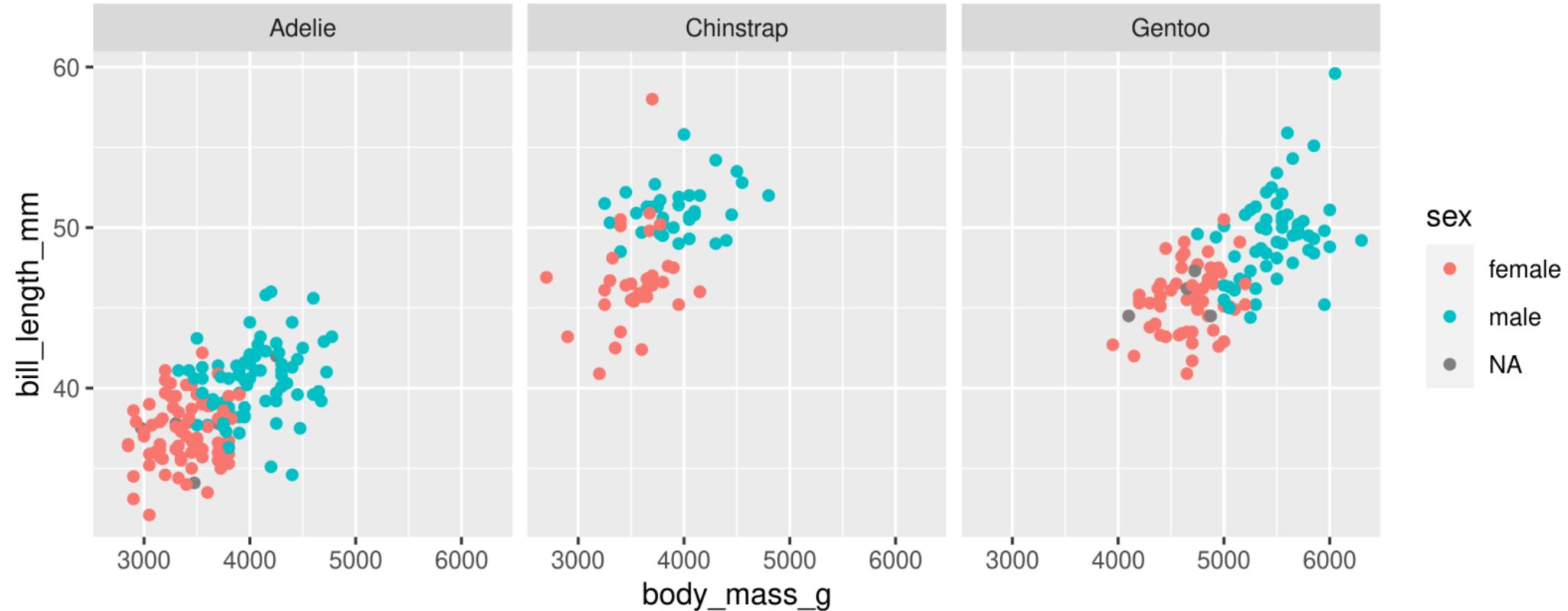
**ggplot** automatically populates the legends (combining where it can)

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



# Faceting: `facet_wrap()`

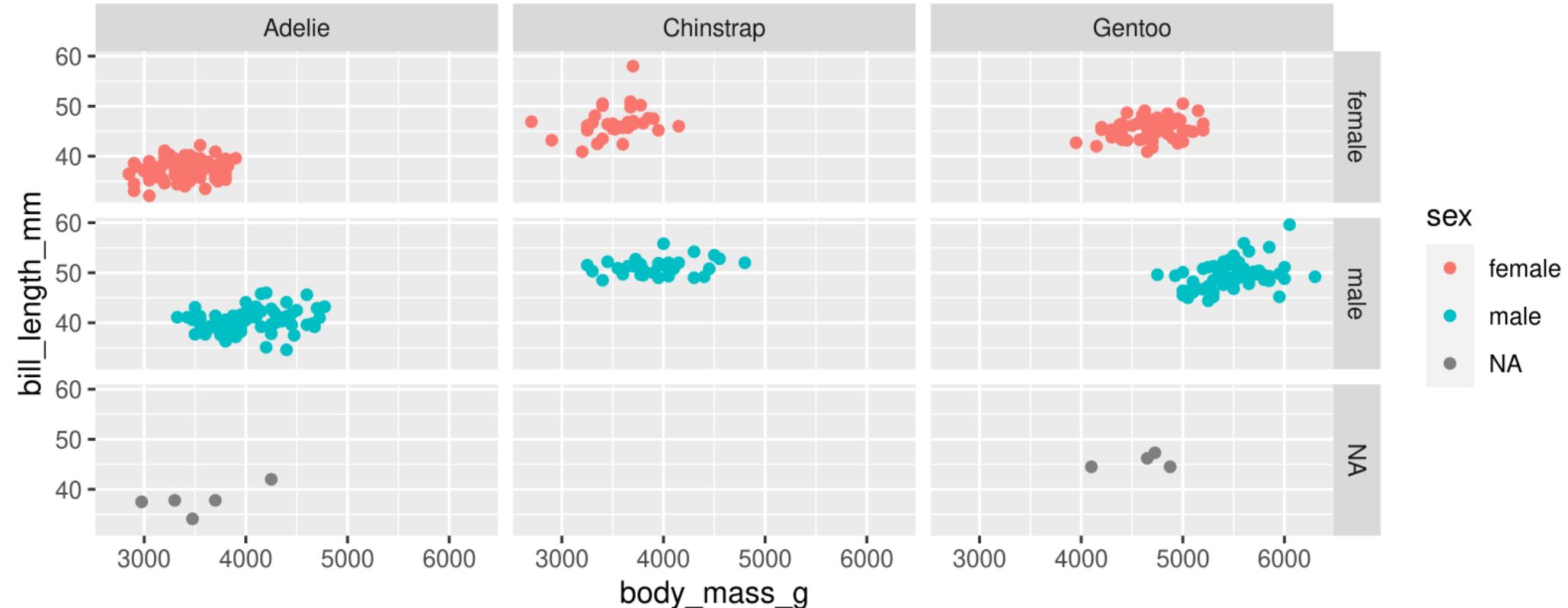
```
1 ggplot(data = penguins, aes(x = body_mass_g, y = bill_length_mm, colour = sex)) +  
2   geom_point() +  
3   facet_wrap(~ species)
```



Split plots by **one** grouping variable

# Faceting: `facet_grid()`

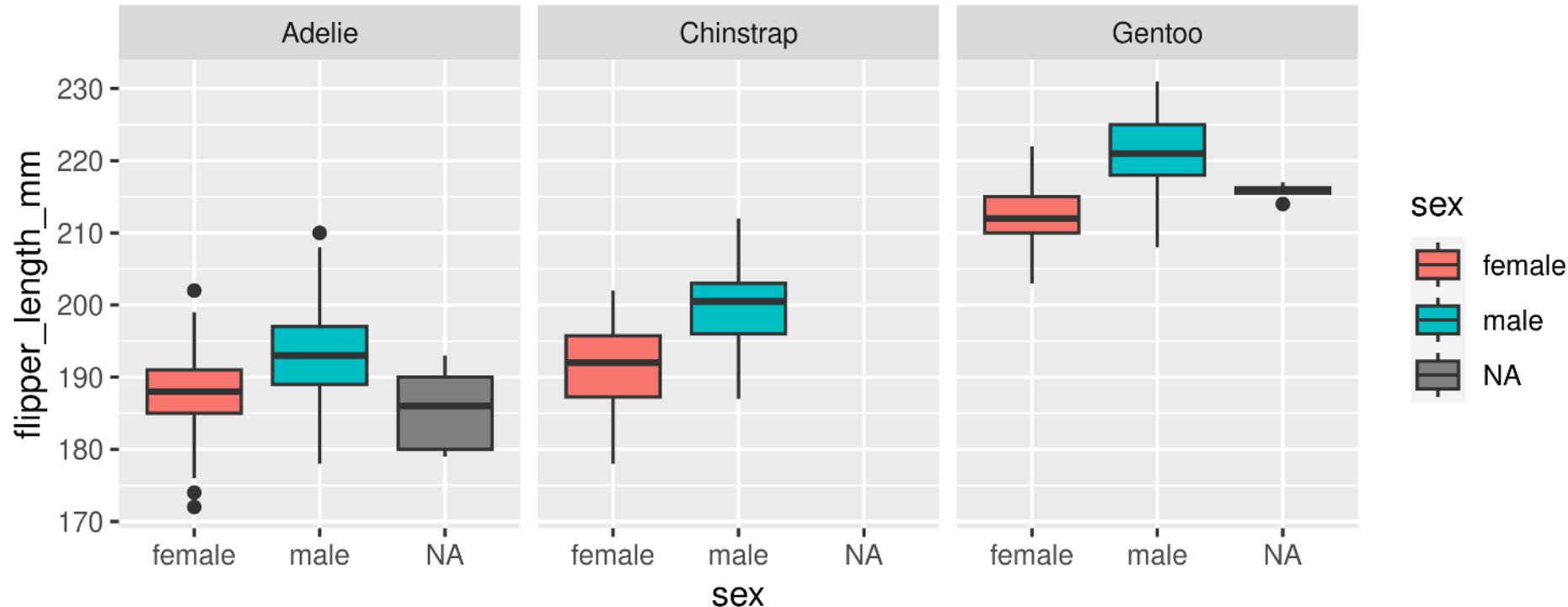
```
1 ggplot(data = penguins, aes(x = body_mass_g, y = bill_length_mm, colour = sex)) +  
2   geom_point() +  
3   facet_grid(species ~ sex)
```



Split plots by **two** grouping variables

# Your Turn: Create this plot

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



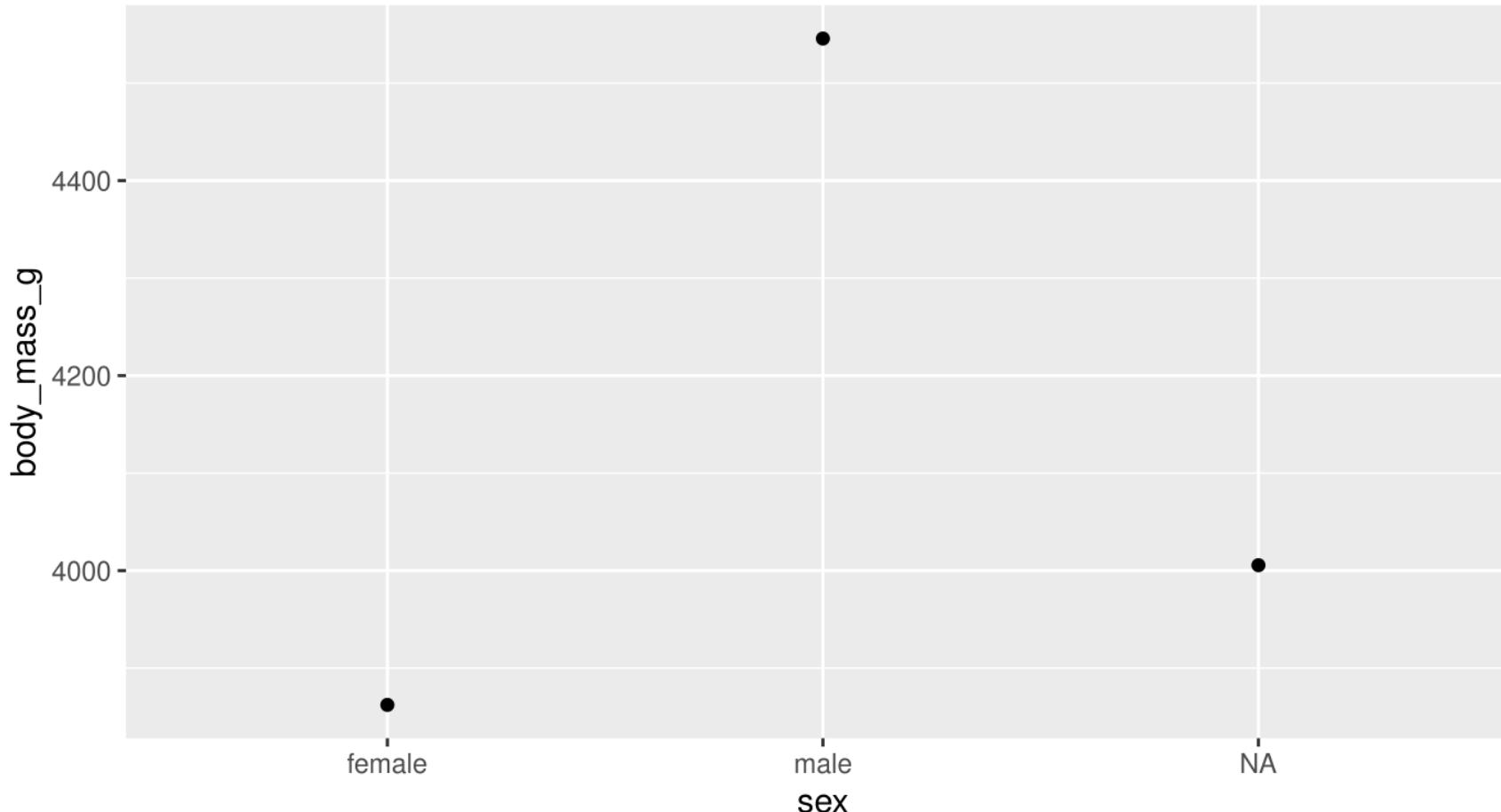
Hint: `colour` is for outlining with a colour, `fill` is for ‘filling’ with a colour  
Too Easy? Split boxplots by sex **and** island

# **Adding Statistics to Plots**

# Summarizing data

## Add data means as points

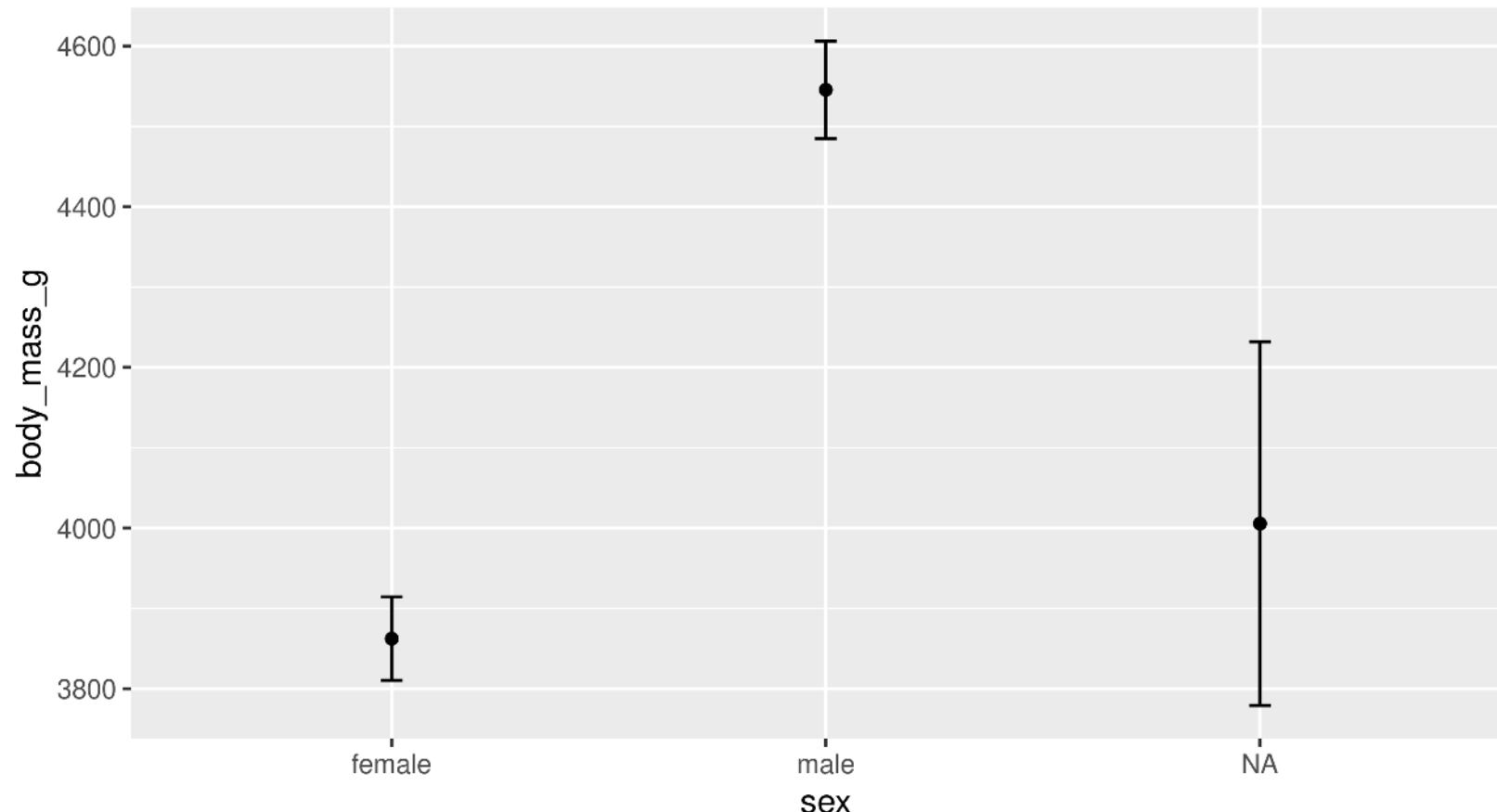
```
1 ggplot(data = penguins, aes(x = sex, y = body_mass_g)) +  
2   stat_summary(geom = "point", fun = mean)
```



# Summarizing data

Add error bars, calculated from the data

```
1 ggplot(data = penguins, aes(x = sex, y = body_mass_g)) +  
2   stat_summary(geom = "point", fun = mean) +  
3   stat_summary(geom = "errorbar", width = 0.05, fun.data = mean_se)
```

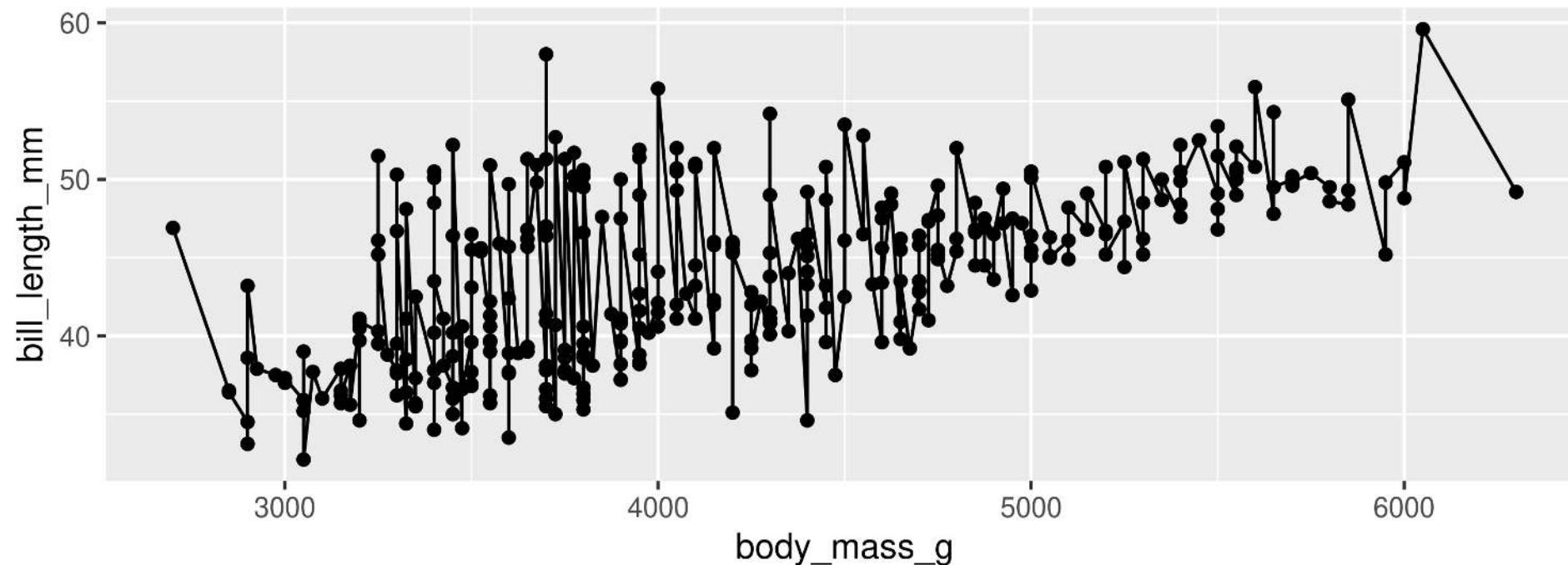


# Trendlines / Regression Lines

# 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()
```



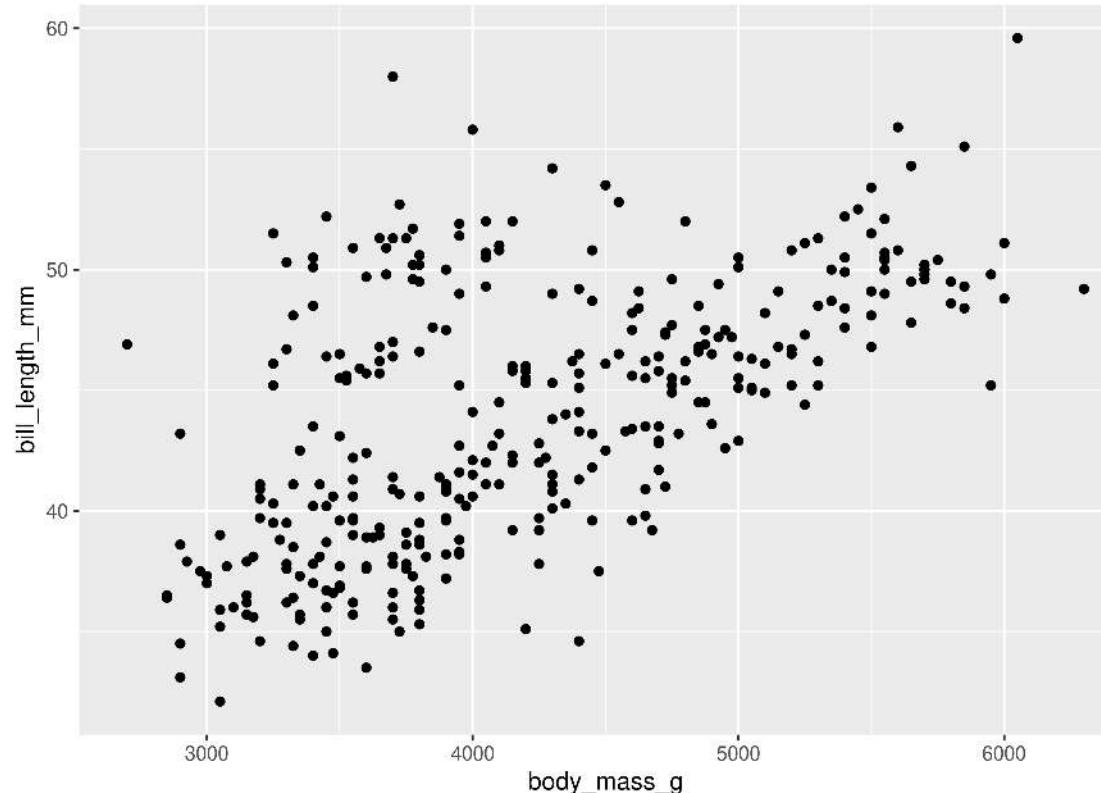
Not what we're looking for

# 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
```

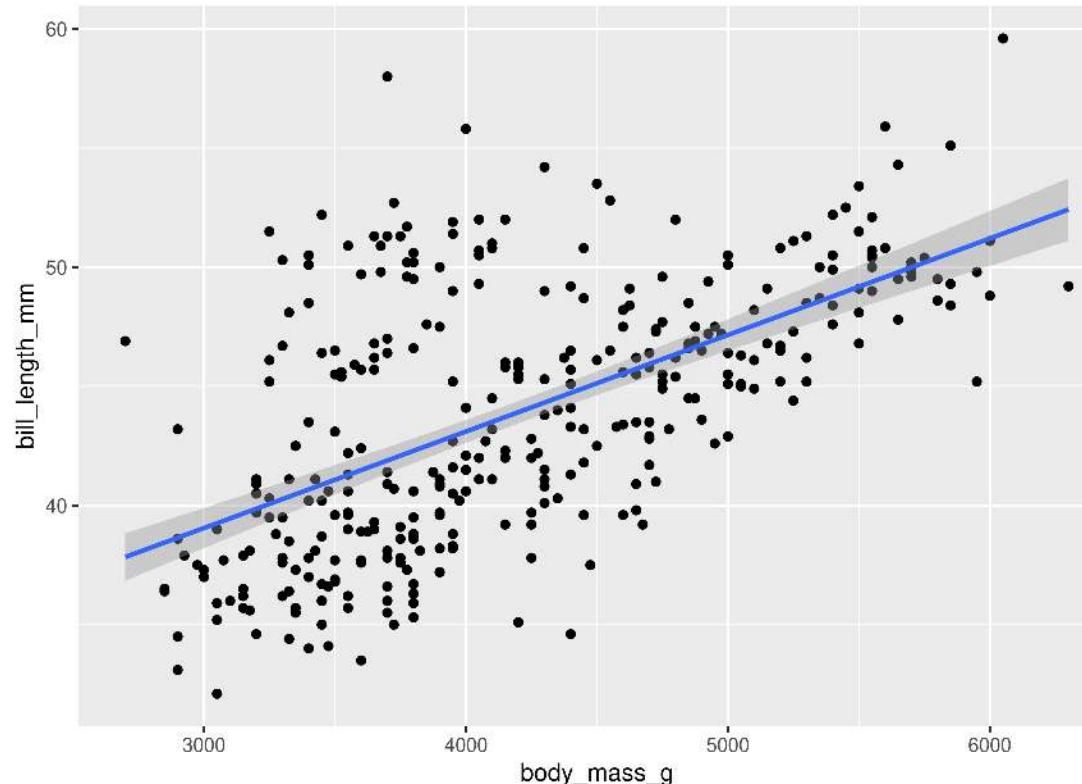


# Trendlines / Regression lines

## Add the `stat_smooth()`

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

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

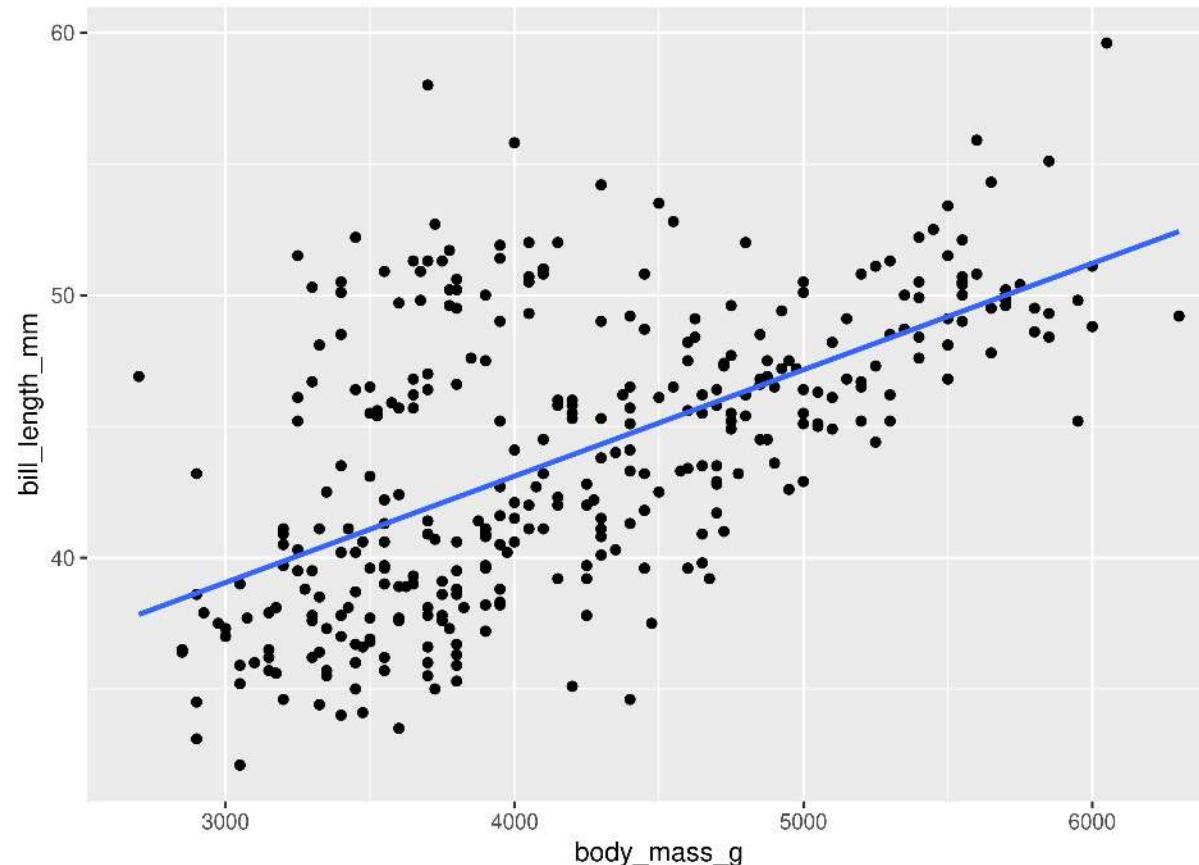


# Trendlines / Regression lines

## Add the `stat_smooth()`

- remove the grey ribbon `se = FALSE`

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

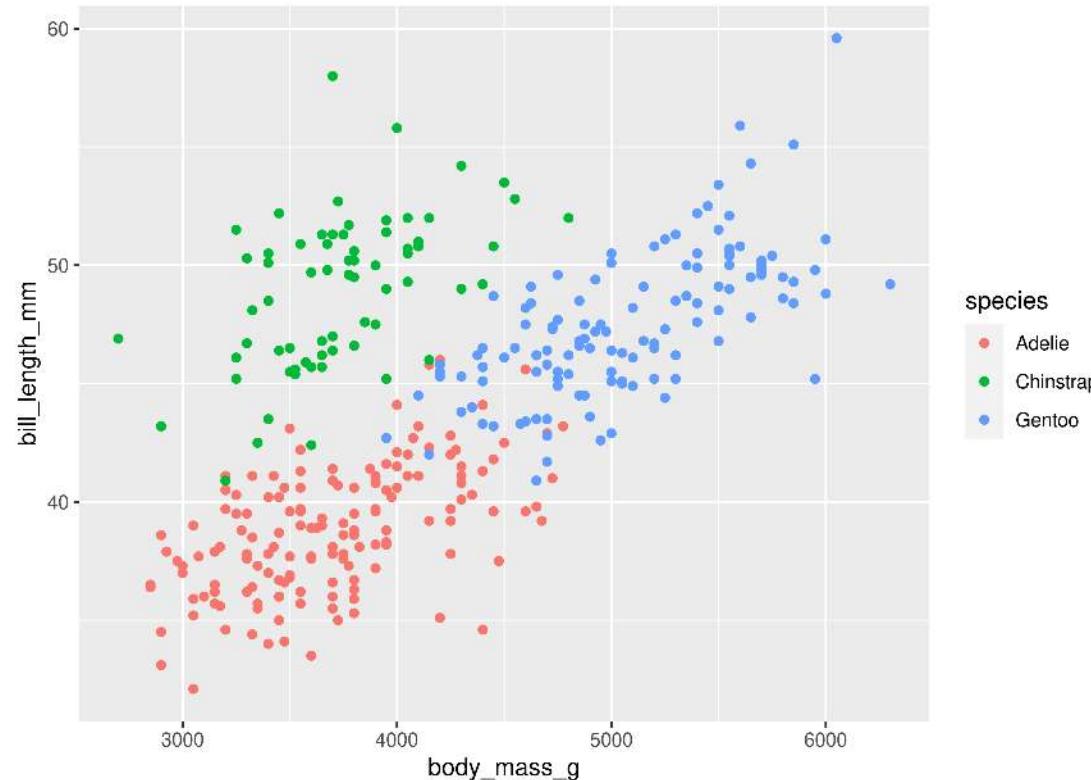


# Trendlines / Regression lines

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
```

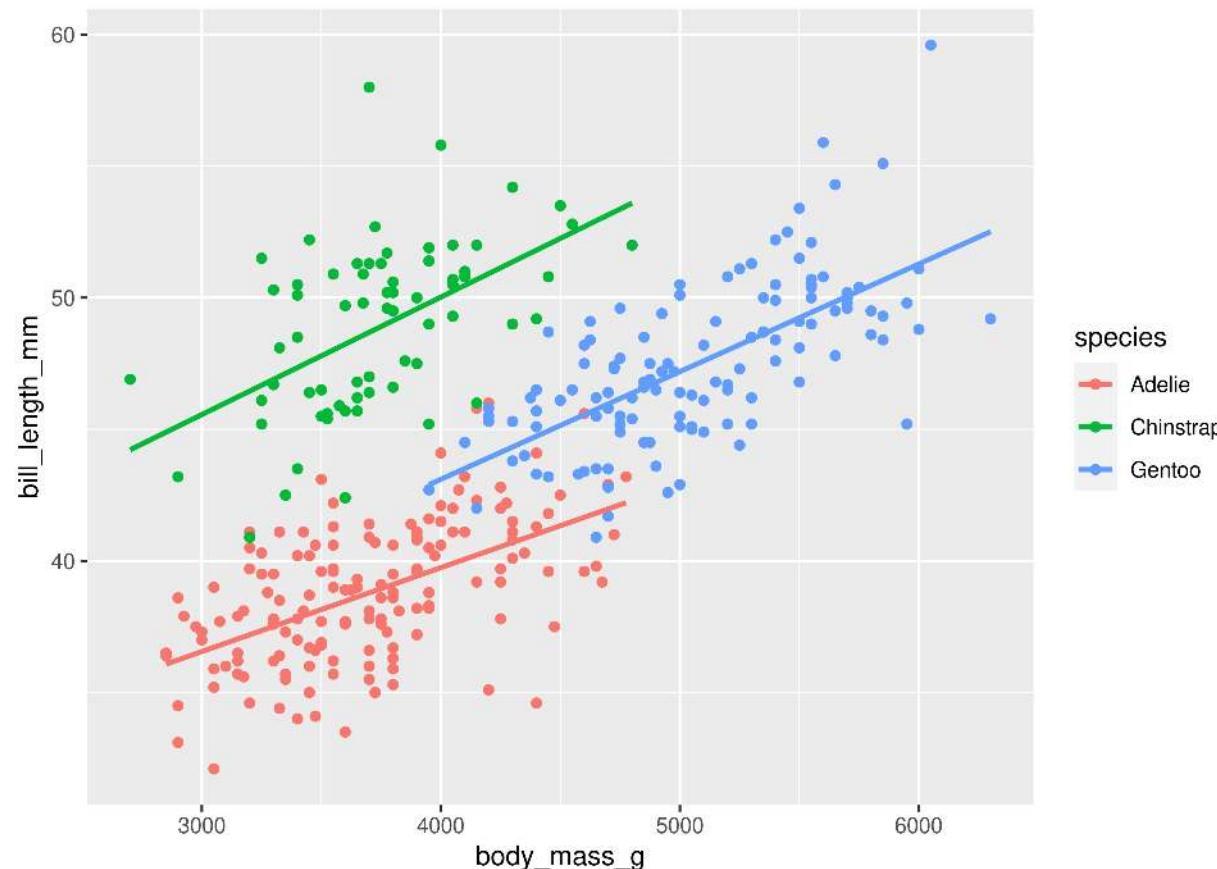


# Trendlines / Regression lines

A line for each group

- `stat_smooth()` automatically uses the same grouping

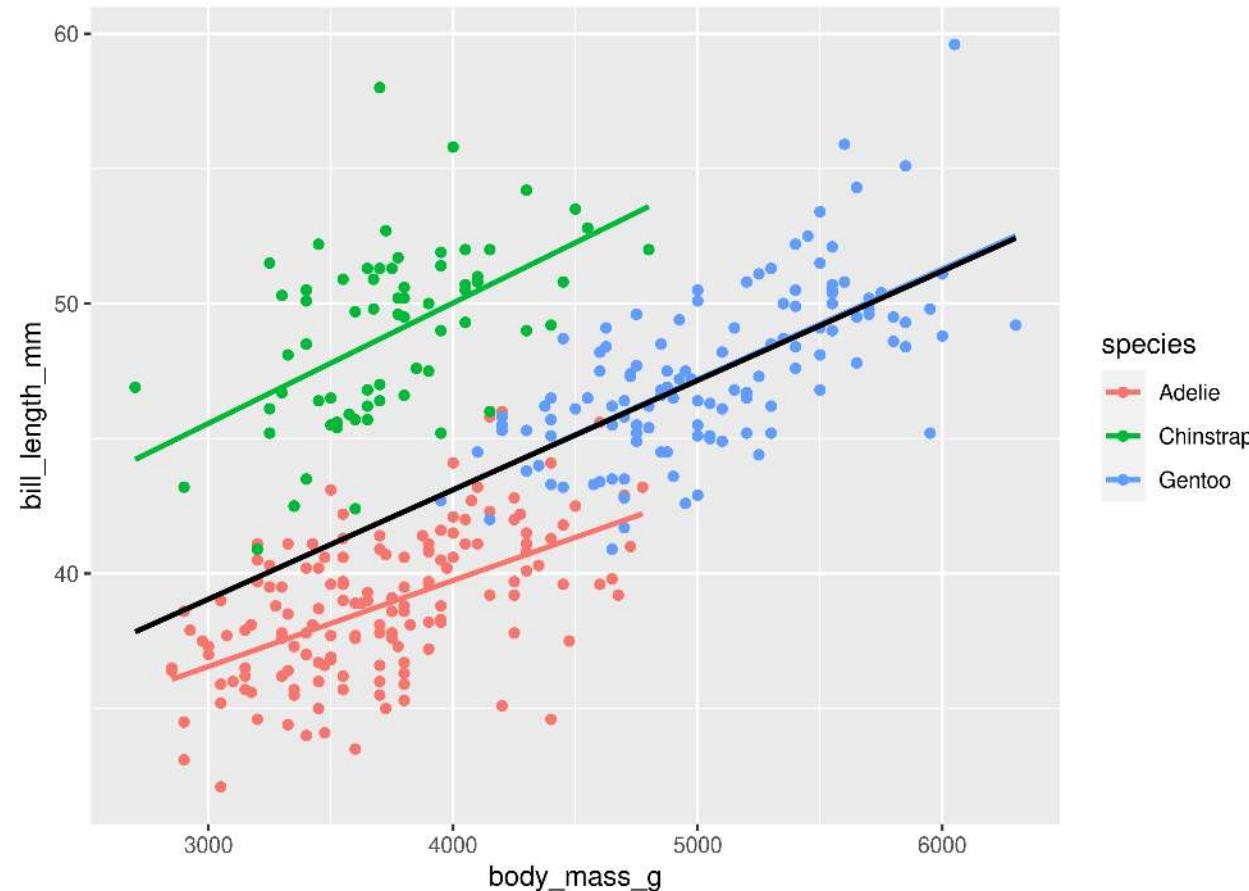
```
1 g + stat_smooth(method = "lm", se = FALSE)
```



# 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*

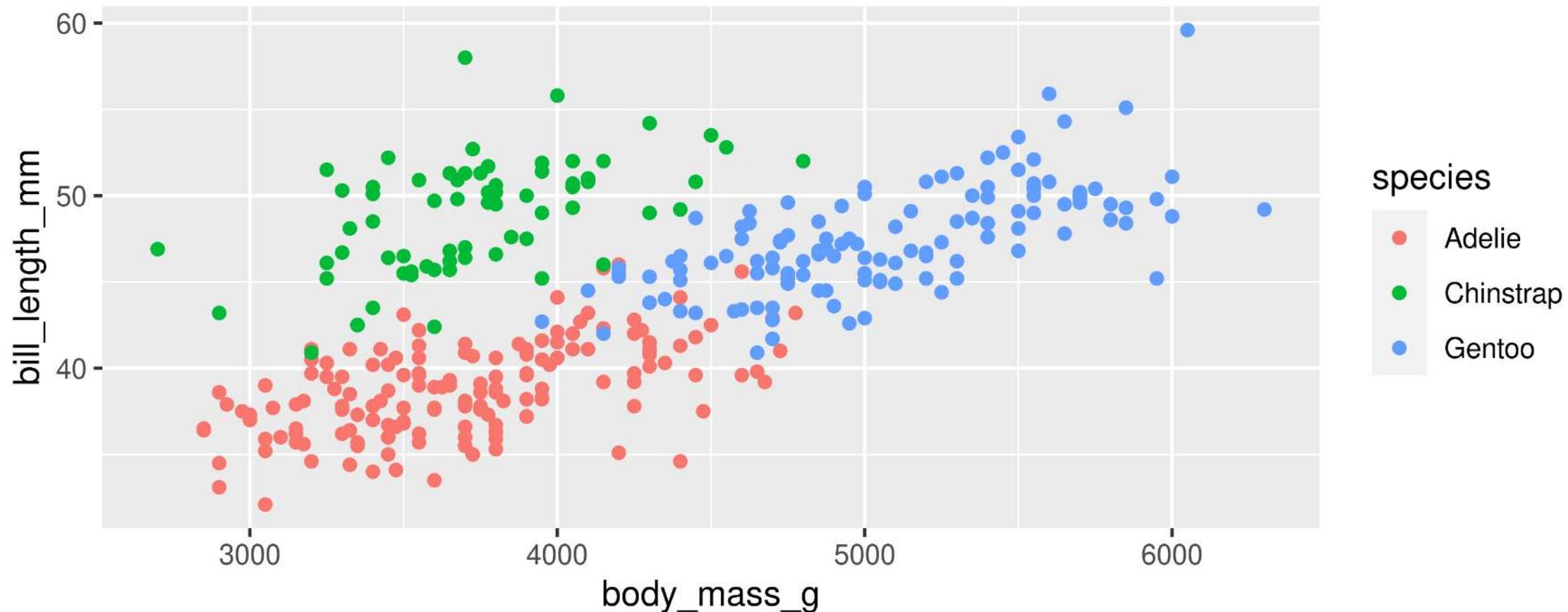
Too Easy? Create a separate plot for each sex as well

# **Customizing plots**

# Customizing: Starting plot

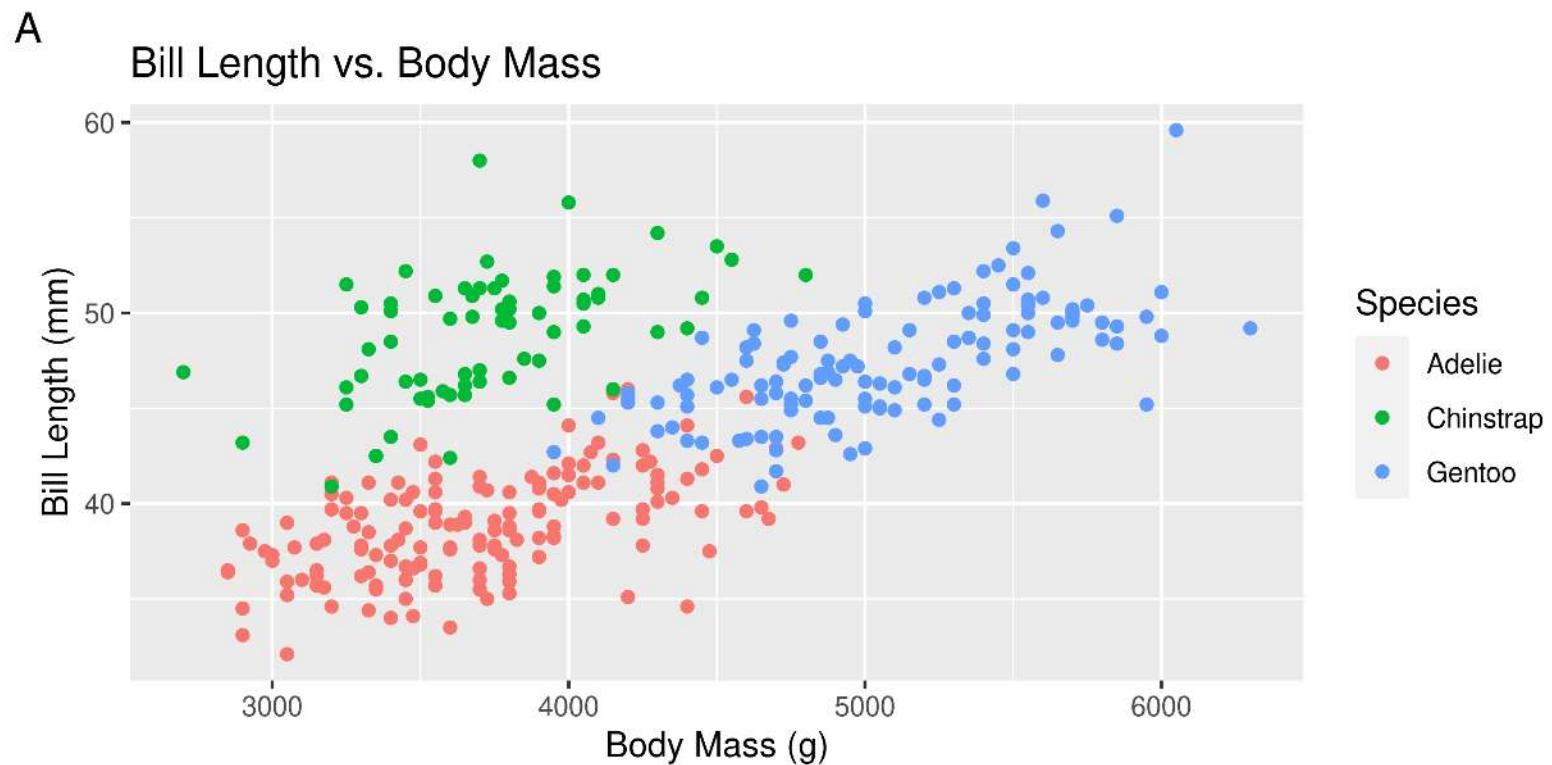
Let's work with this plot

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



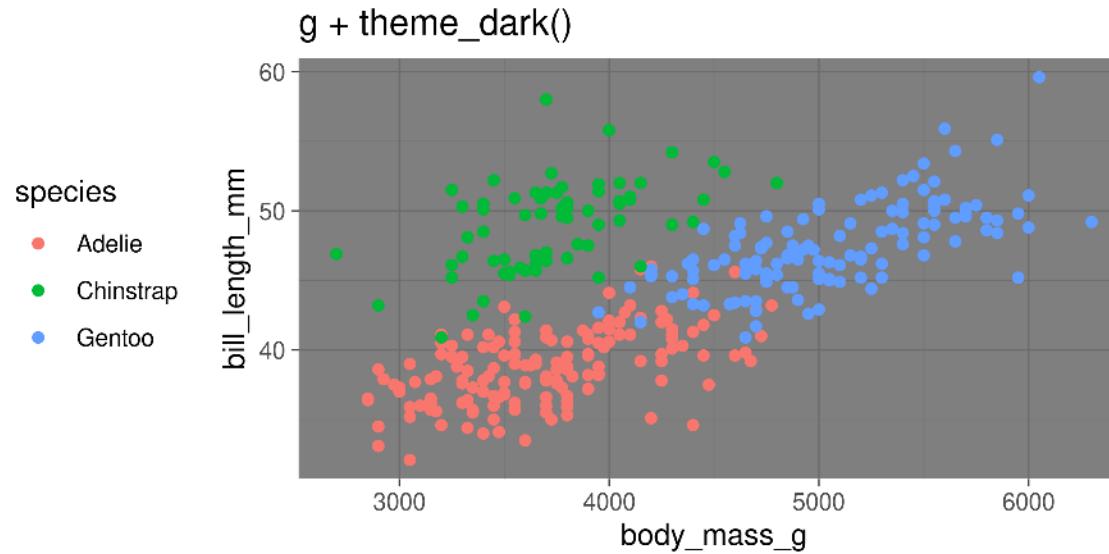
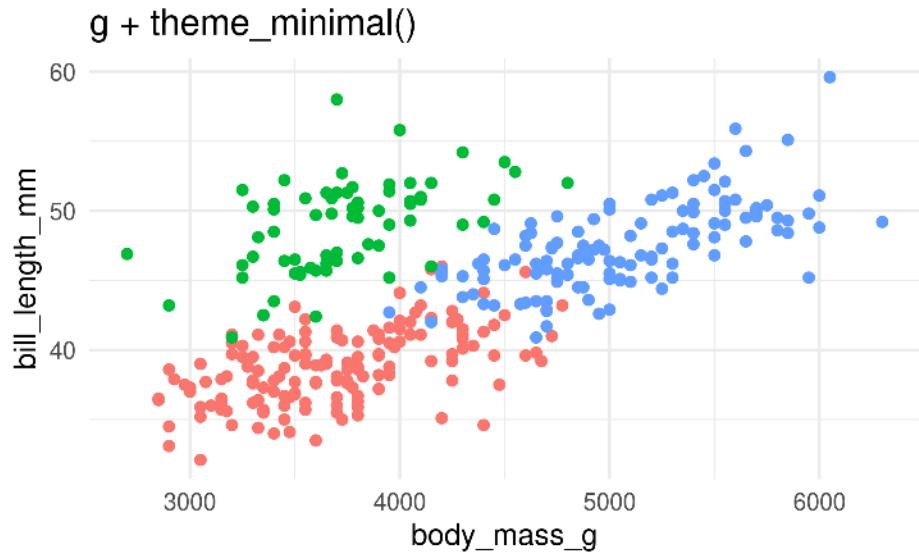
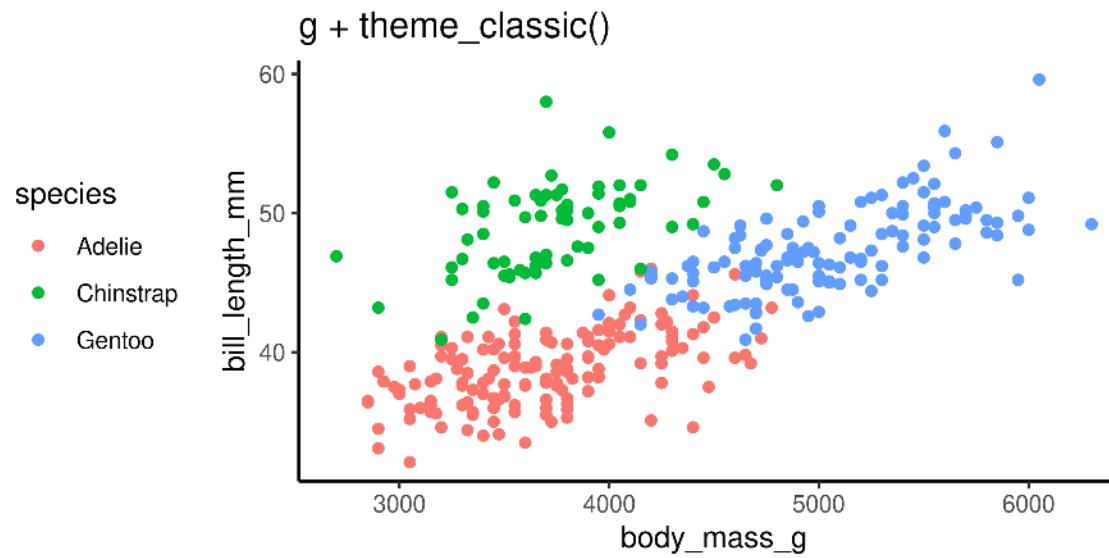
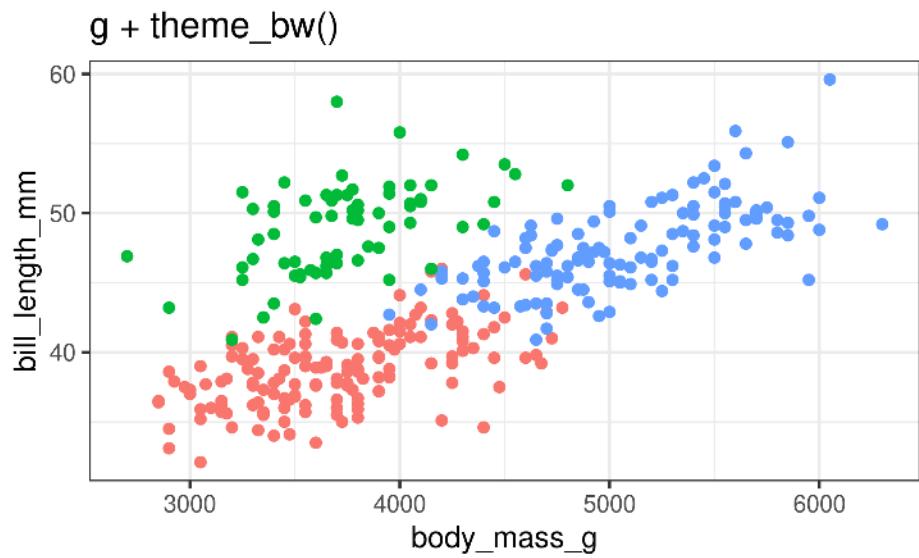
# Customizing: Labels

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



Your Turn: 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.

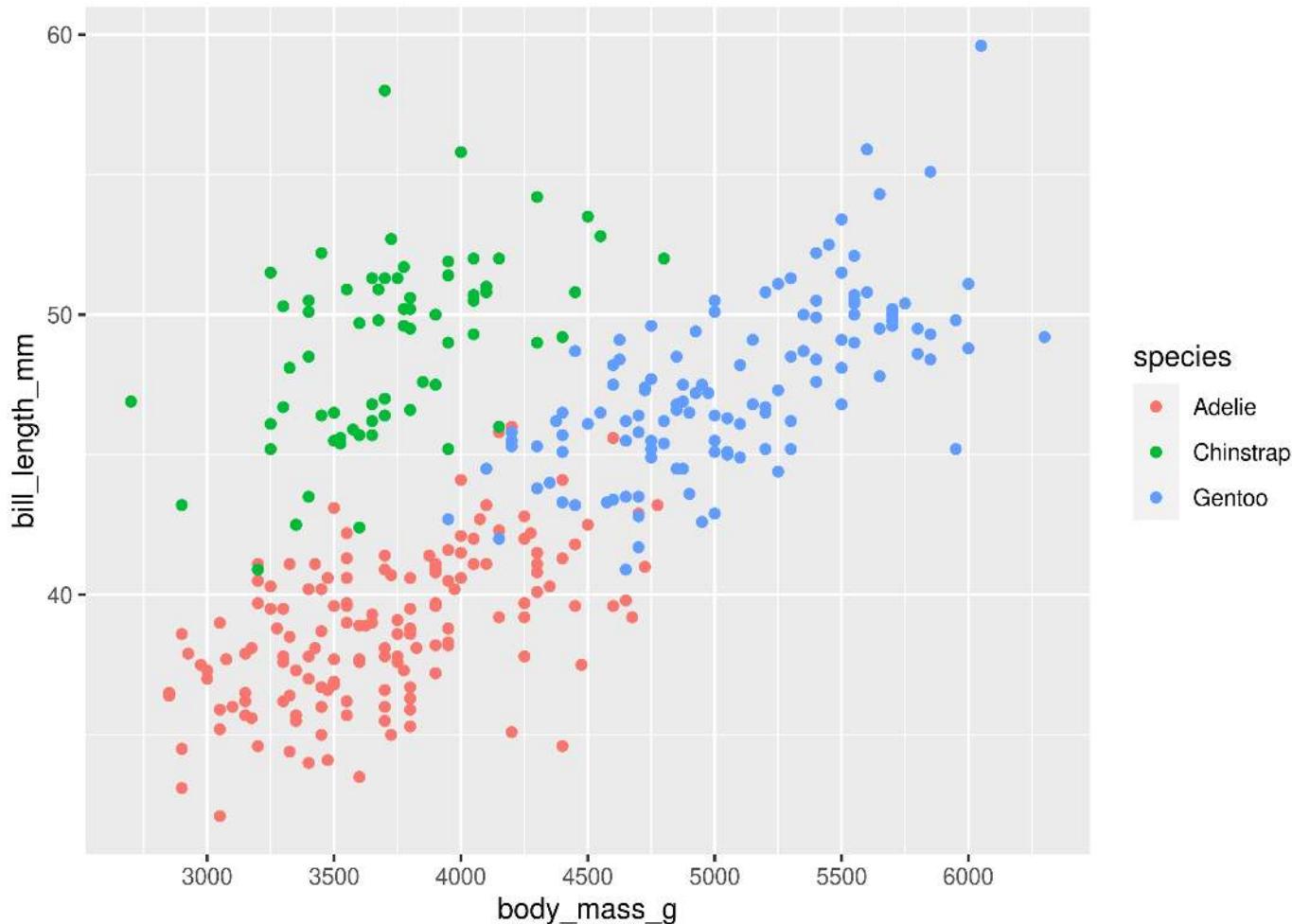
## Common arguments

```
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

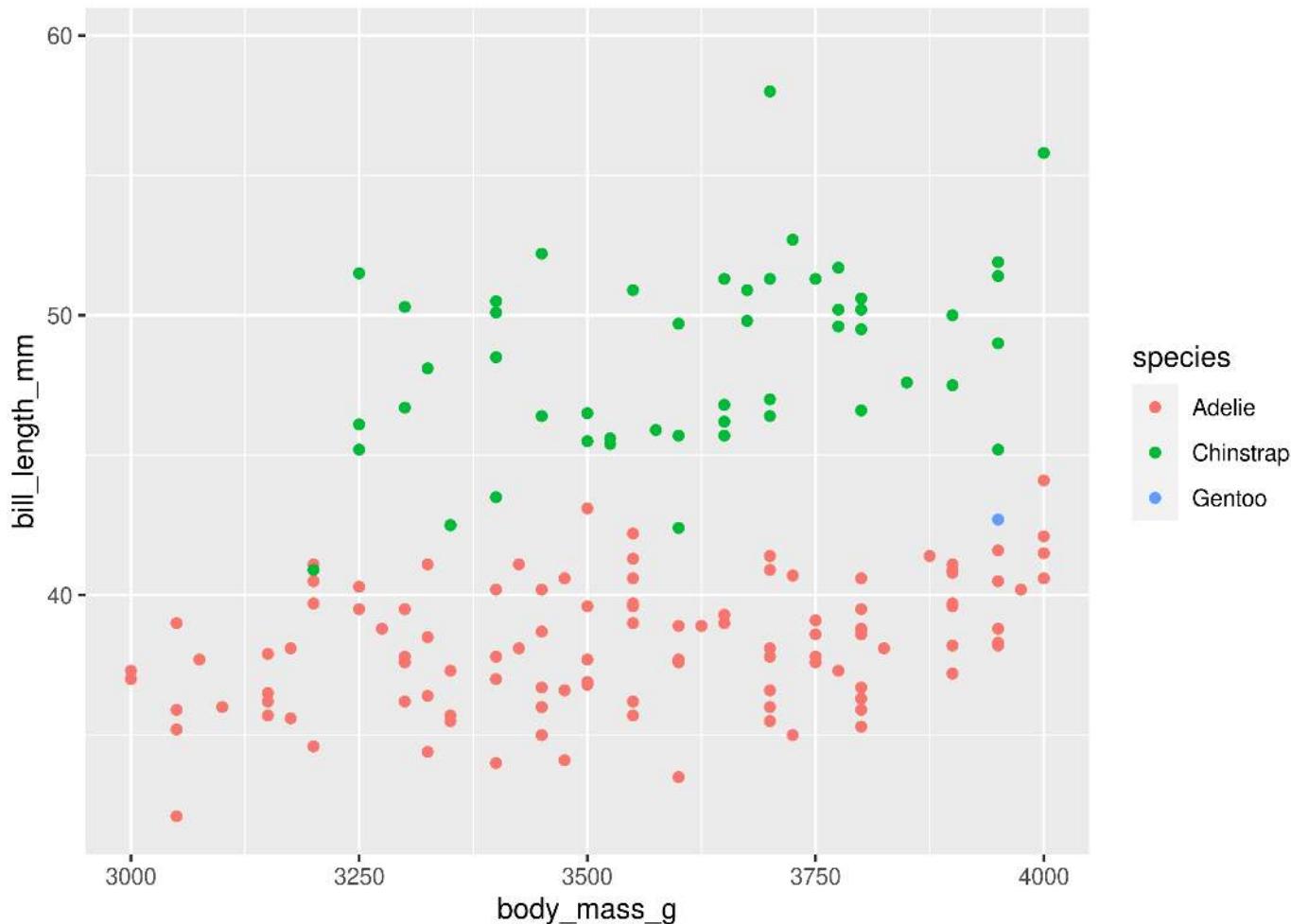
```
1 g + scale_x_continuous(breaks = seq(2500, 6500, 500))
```



# Customizing: Axes

## Limits

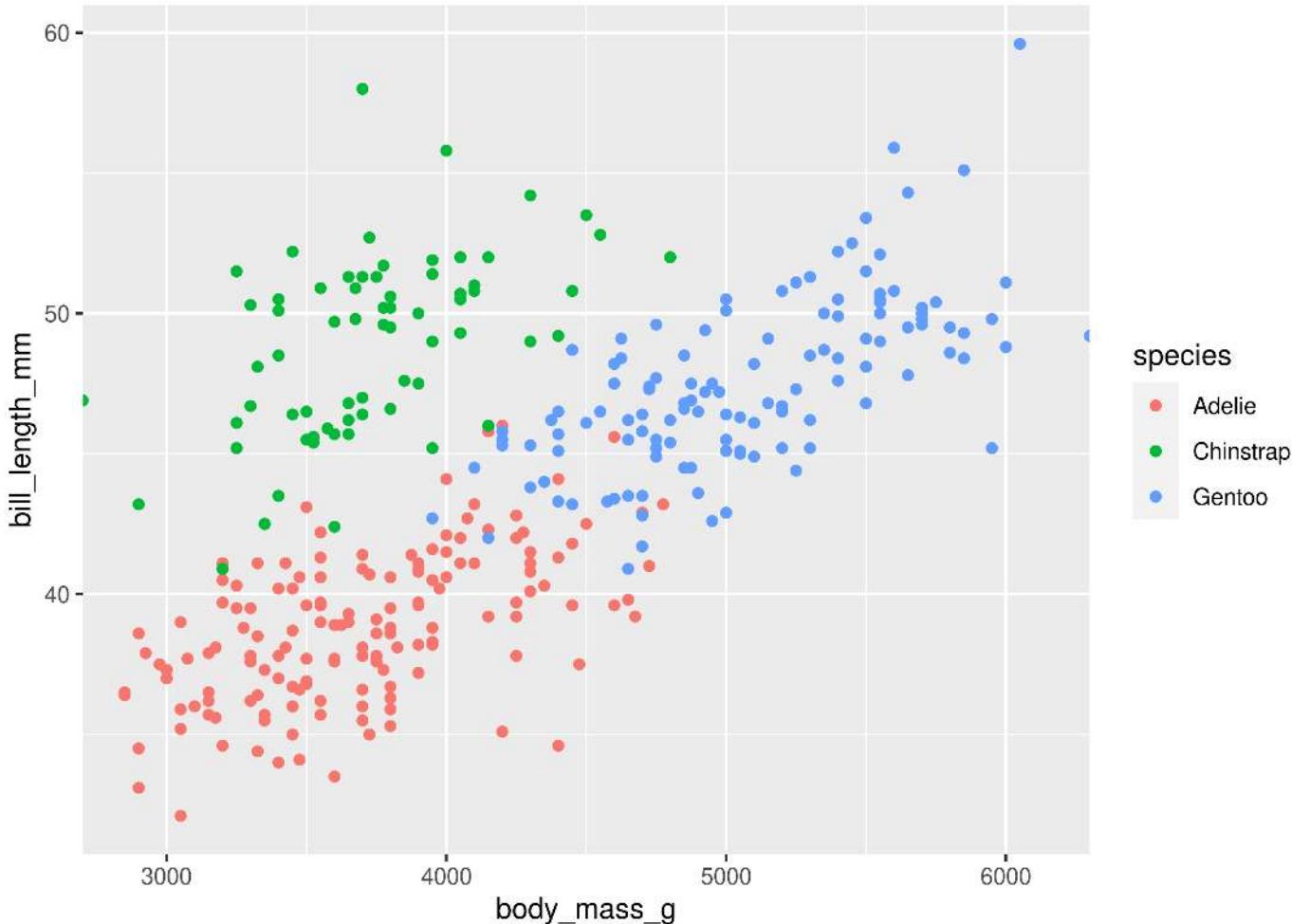
```
1 g + scale_x_continuous(limits = c(3000, 4000))
```



# Customizing: Axes

## Space between origin and axis start

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

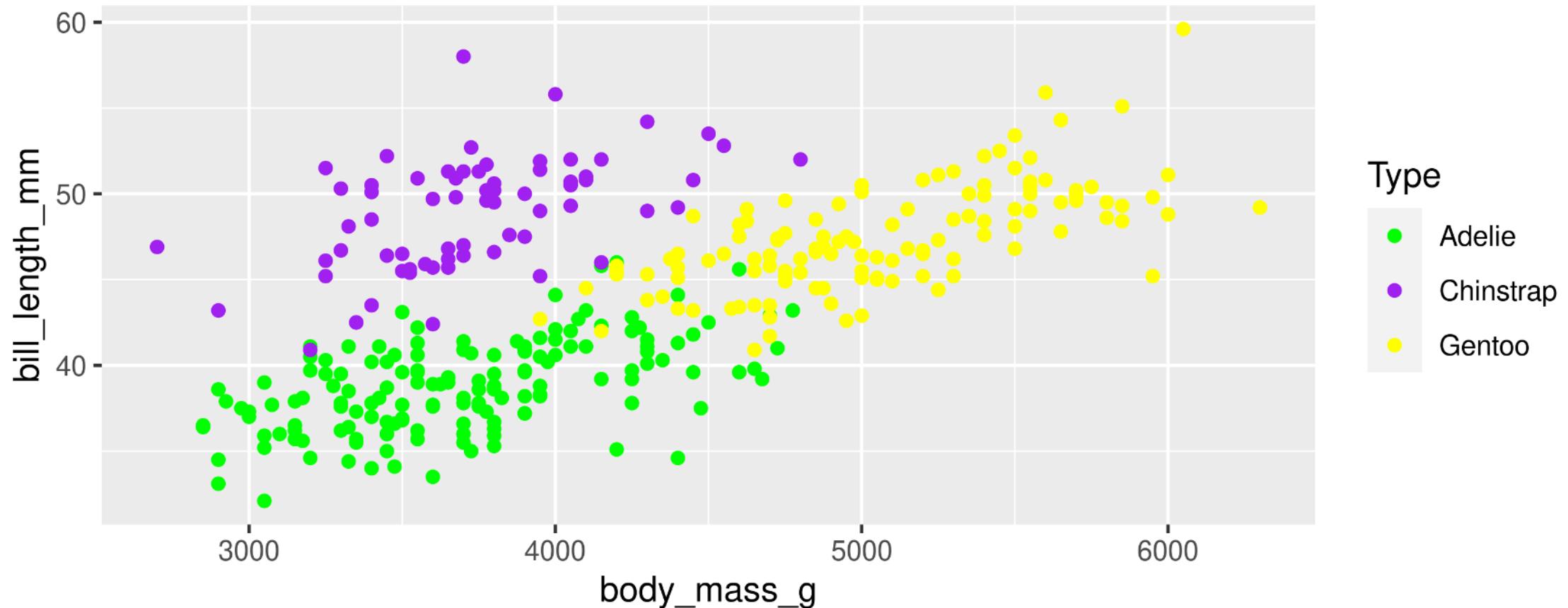


# Customizing: Aesthetics

## 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"))
```

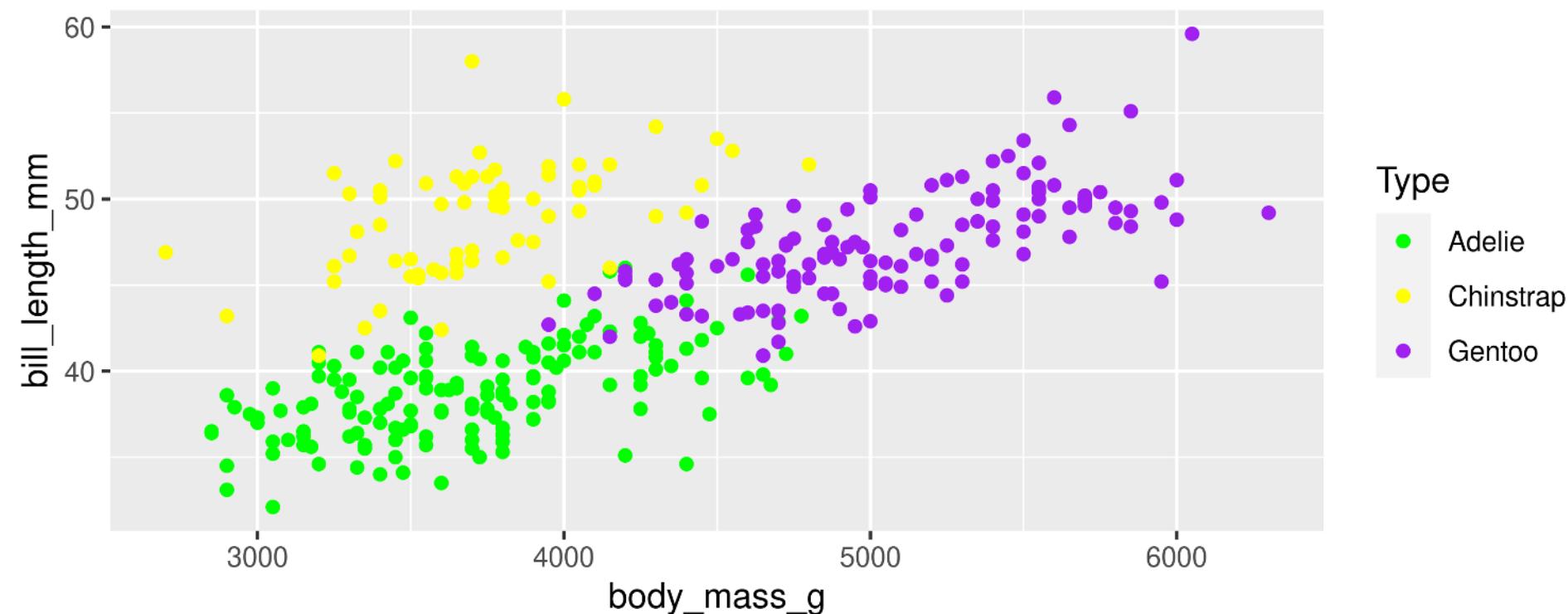


# Customizing: Aesthetics

## 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")
```

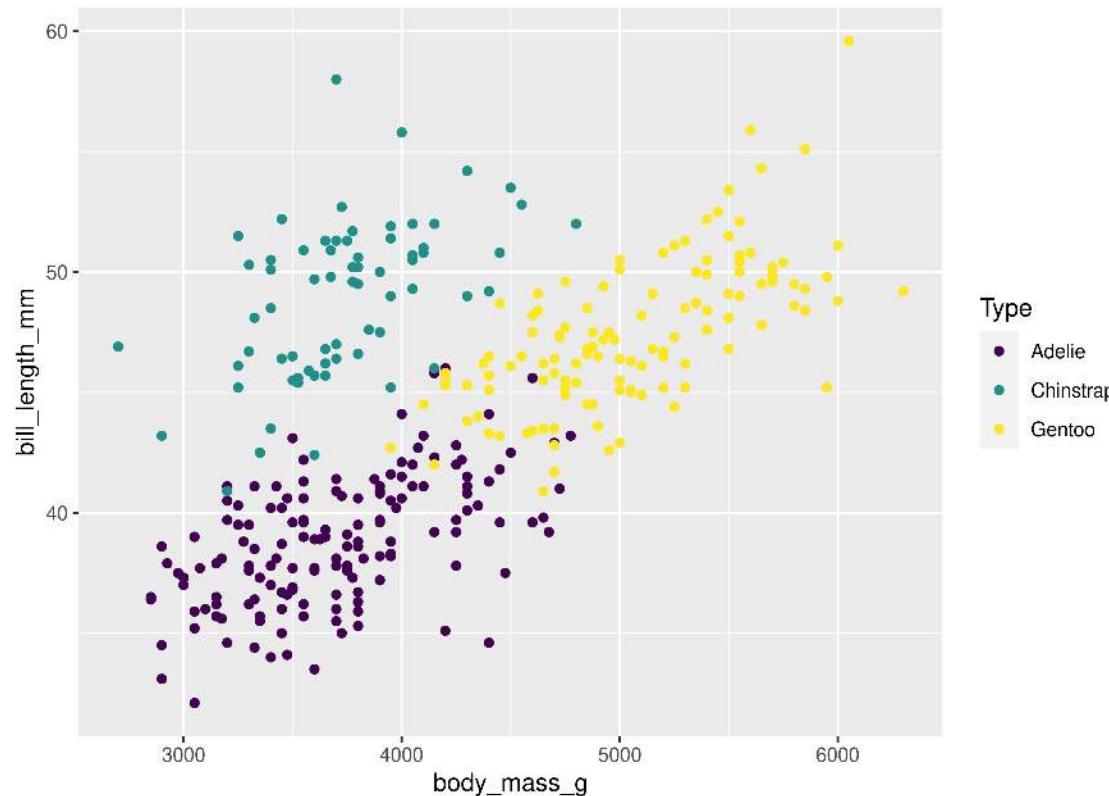


# Customizing: Aesthetics

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")
```

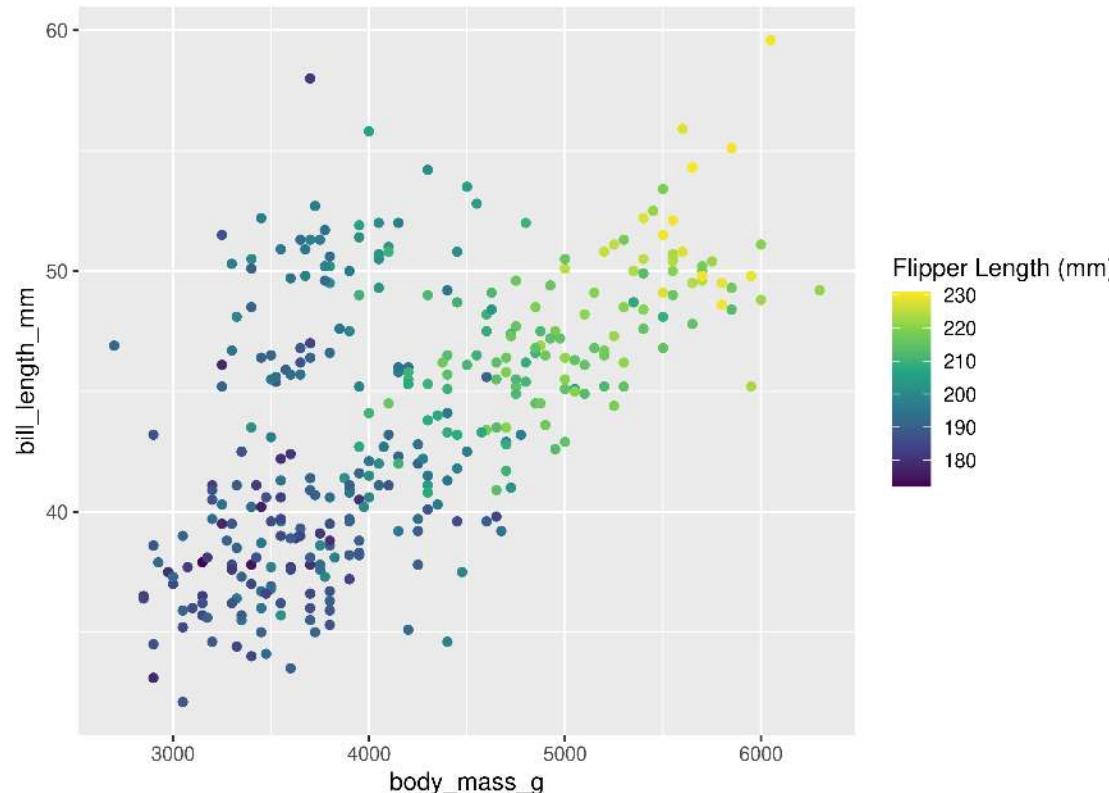


# Customizing: Aesthetics

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)")
```

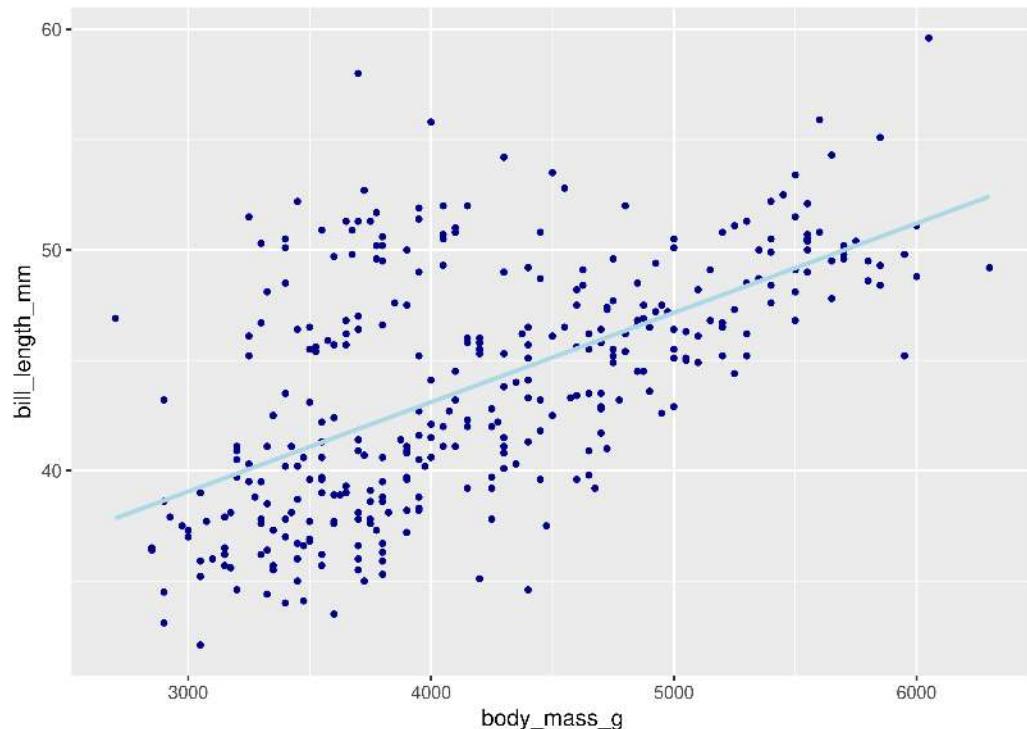


# Customizing: Aesthetics

## 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")
```

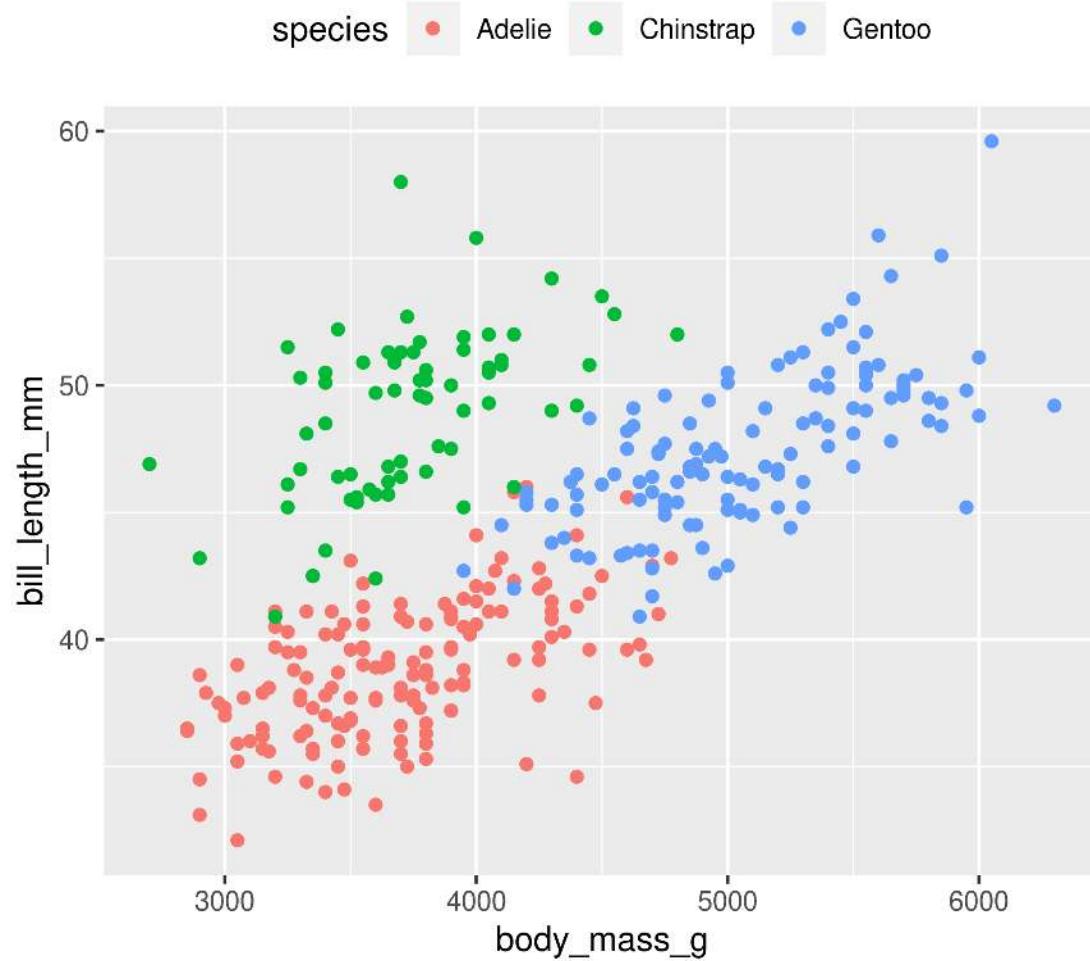


Note: When forcing, aesthetic is  
not inside `aes()`

# Customizing: Legends placement

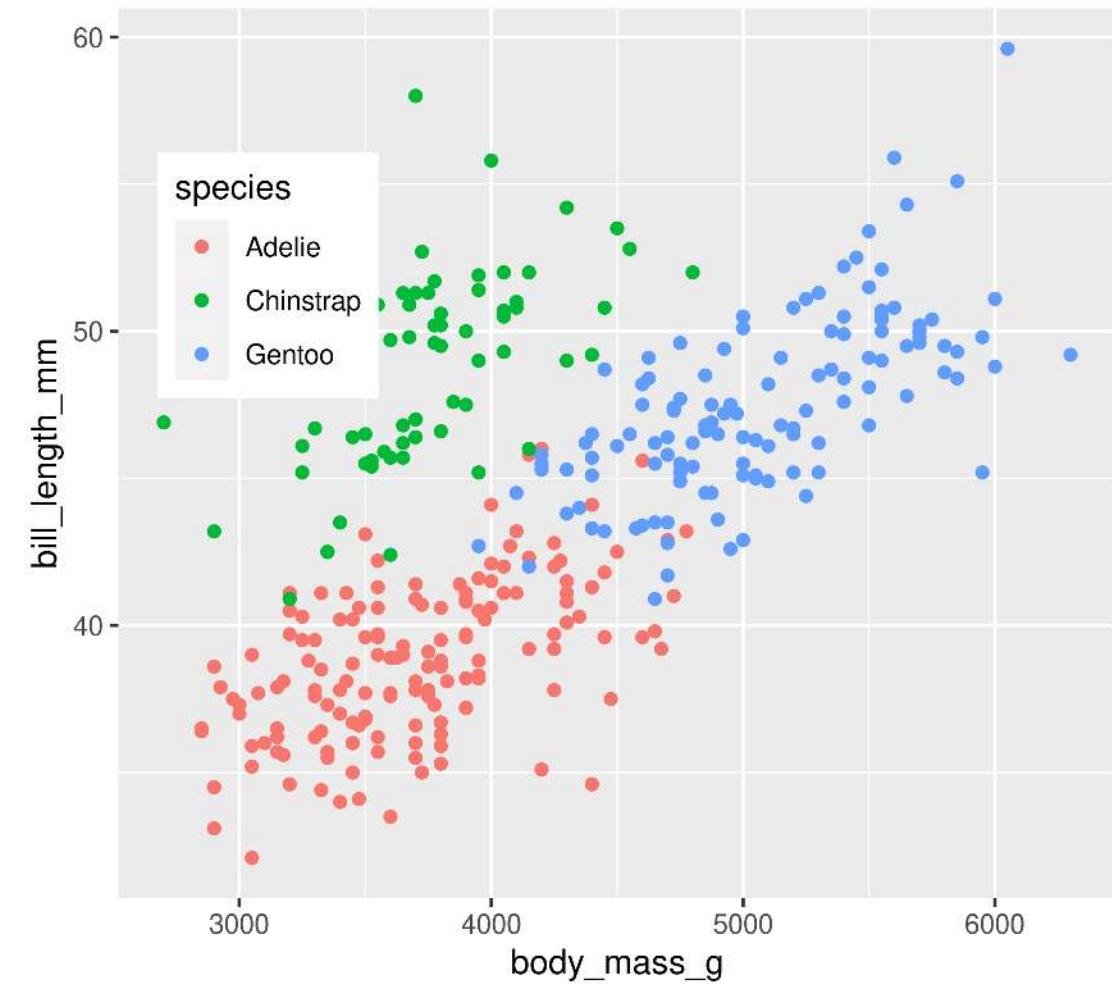
# At the: top, bottom, left, right

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



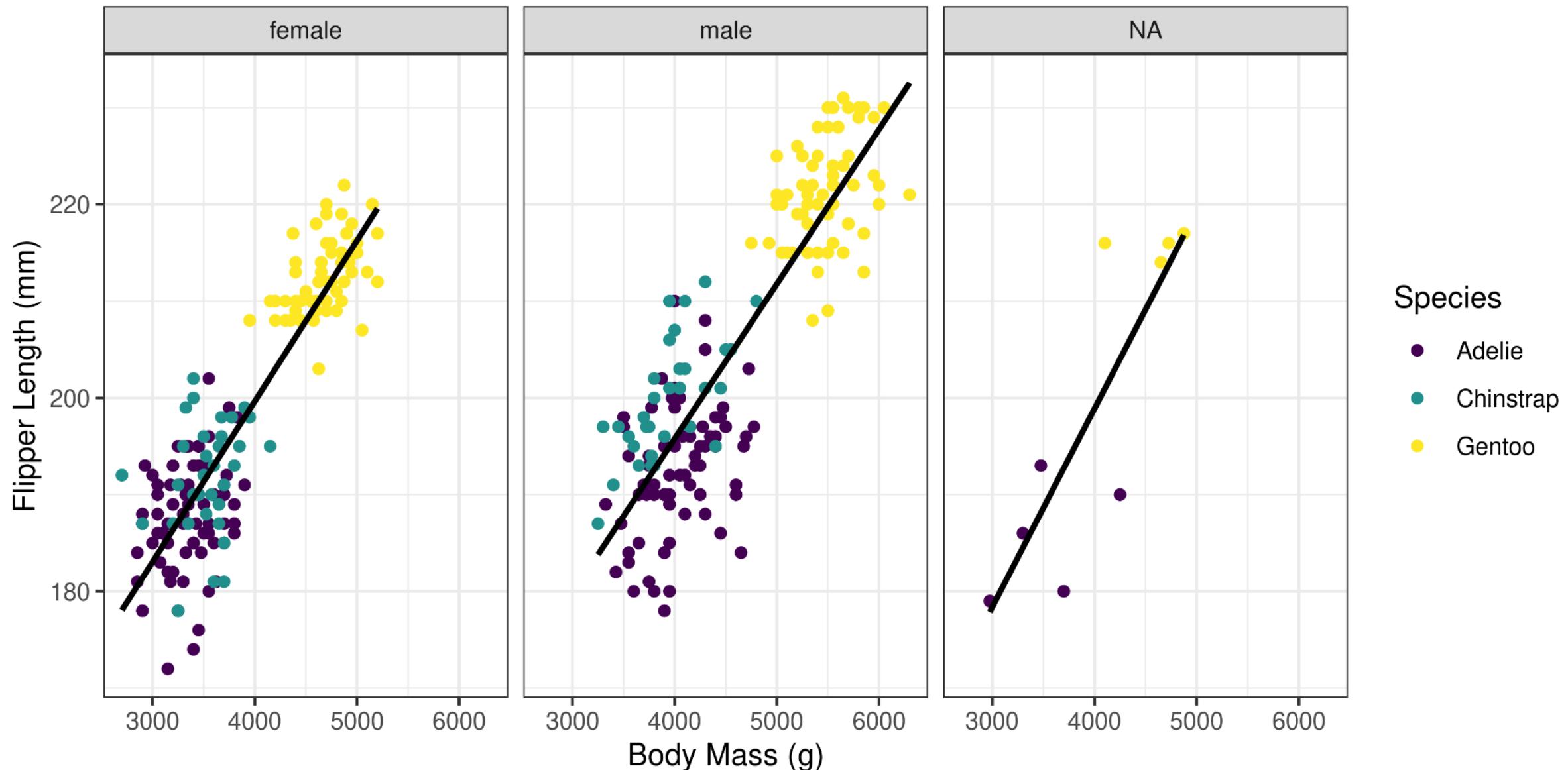
# Exactly here

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



# Your Turn: Create this plot

Too Easy?  
Create a plot of your own data



## **Side note: Order of operations**

# Order of operations

Remember...

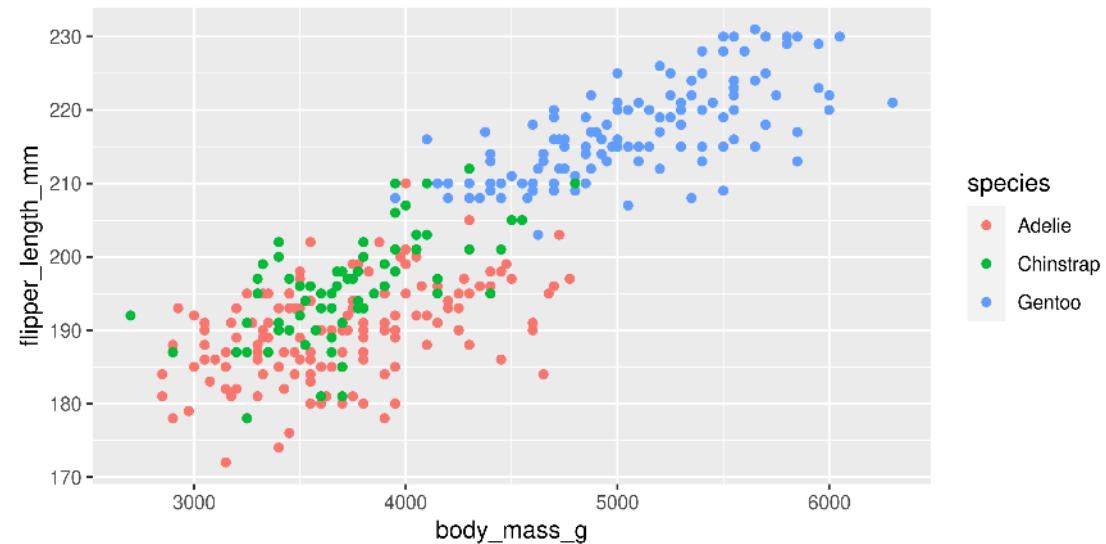
- `ggplot()` is the default line (all options passed down)
- The other lines are *added* with the `+` (options only apply to this line)

# Order of operations

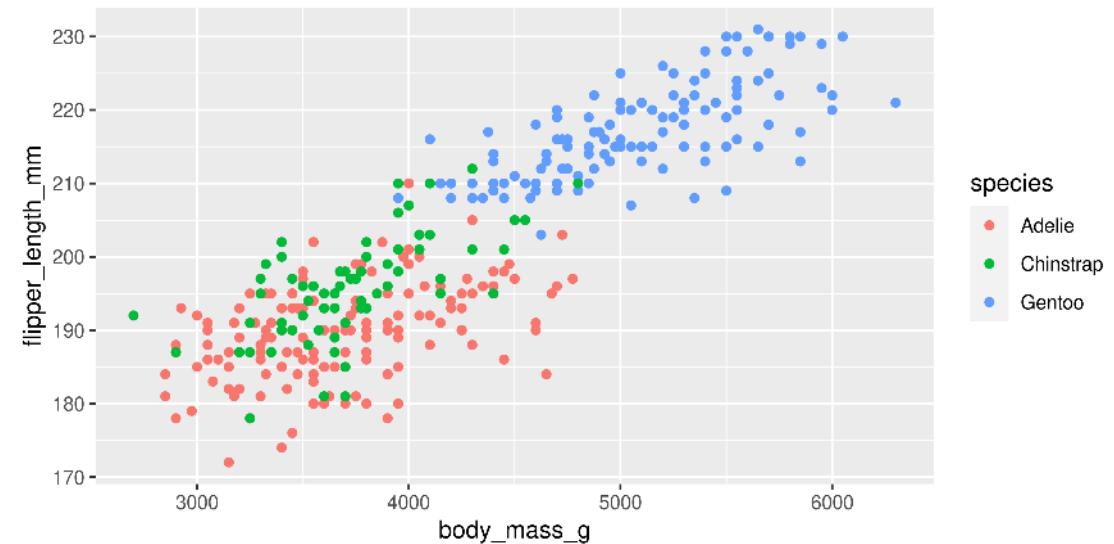
Where to put the `aes()`?

Sometimes it doesn't matter...

```
1 ggplot(penguins, aes(x = body_mass_g,
2                         y = flipper_length_mm,
3                         colour = species)) +
4   geom_point()
```



```
1 ggplot(penguins, aes(x = body_mass_g,
2                         y = flipper_length_mm)) +
3   geom_point(aes(colour = species))
```

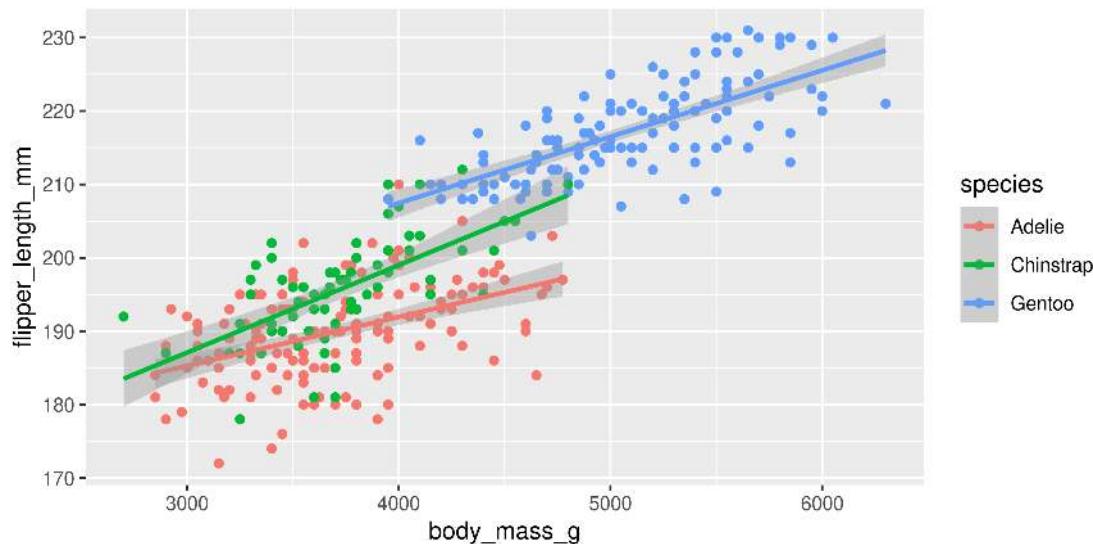


# Order of operations

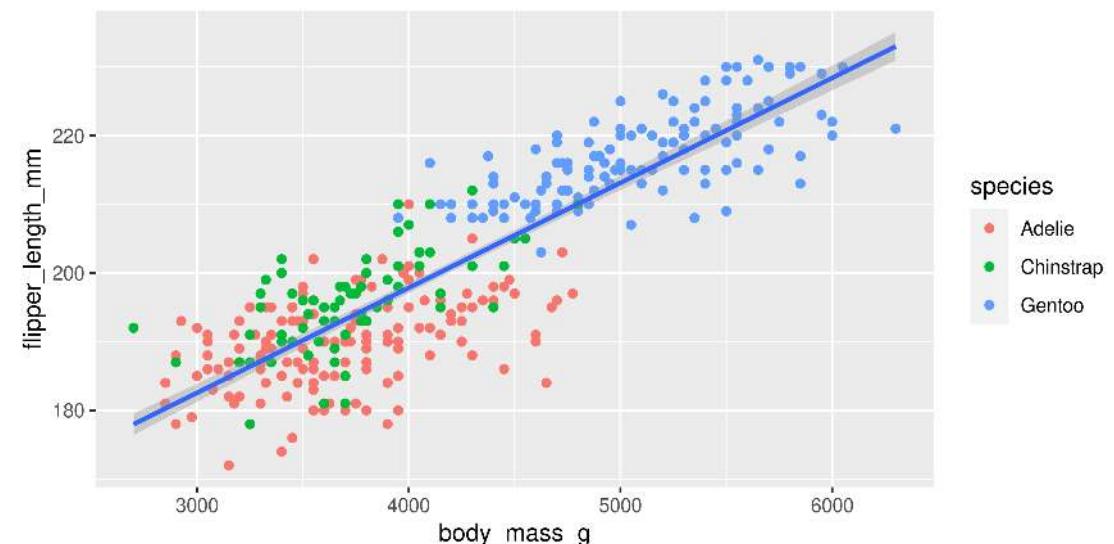
Where to put the `aes()`?

Sometimes it DOES matter...

```
1 ggplot(penguins, aes(x = body_mass_g,
2                         y = flipper_length_mm,
3                         colour = species)) +
4   geom_point() +
5   stat_smooth(method = "lm")
```



```
1 ggplot(penguins, aes(x = body_mass_g,
2                         y = flipper_length_mm)) +
3   geom_point(aes(colour = species)) +
4   stat_smooth(method = "lm")
```



including `stat_smooth()` in the `ggplot`

`geom_point()` in the `ggplot`  
Applies to only the

# Combining plots

# Combining plots with patchwork

## Setup

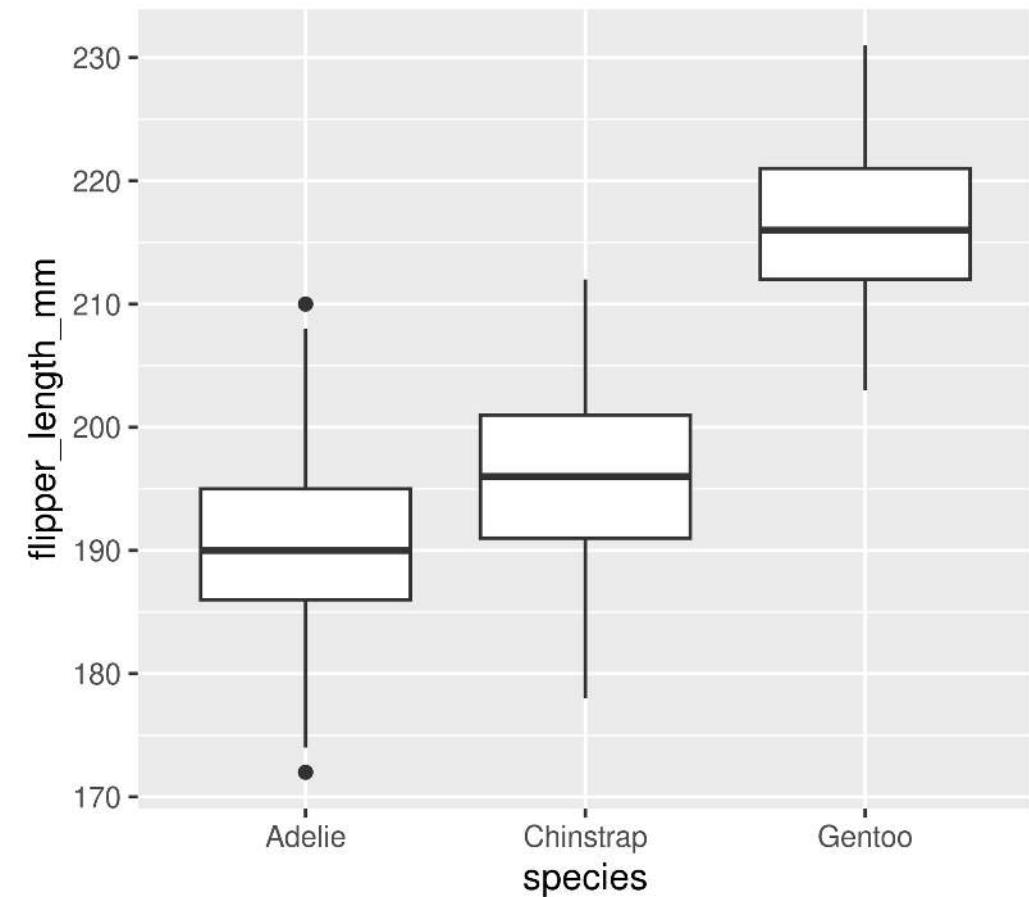
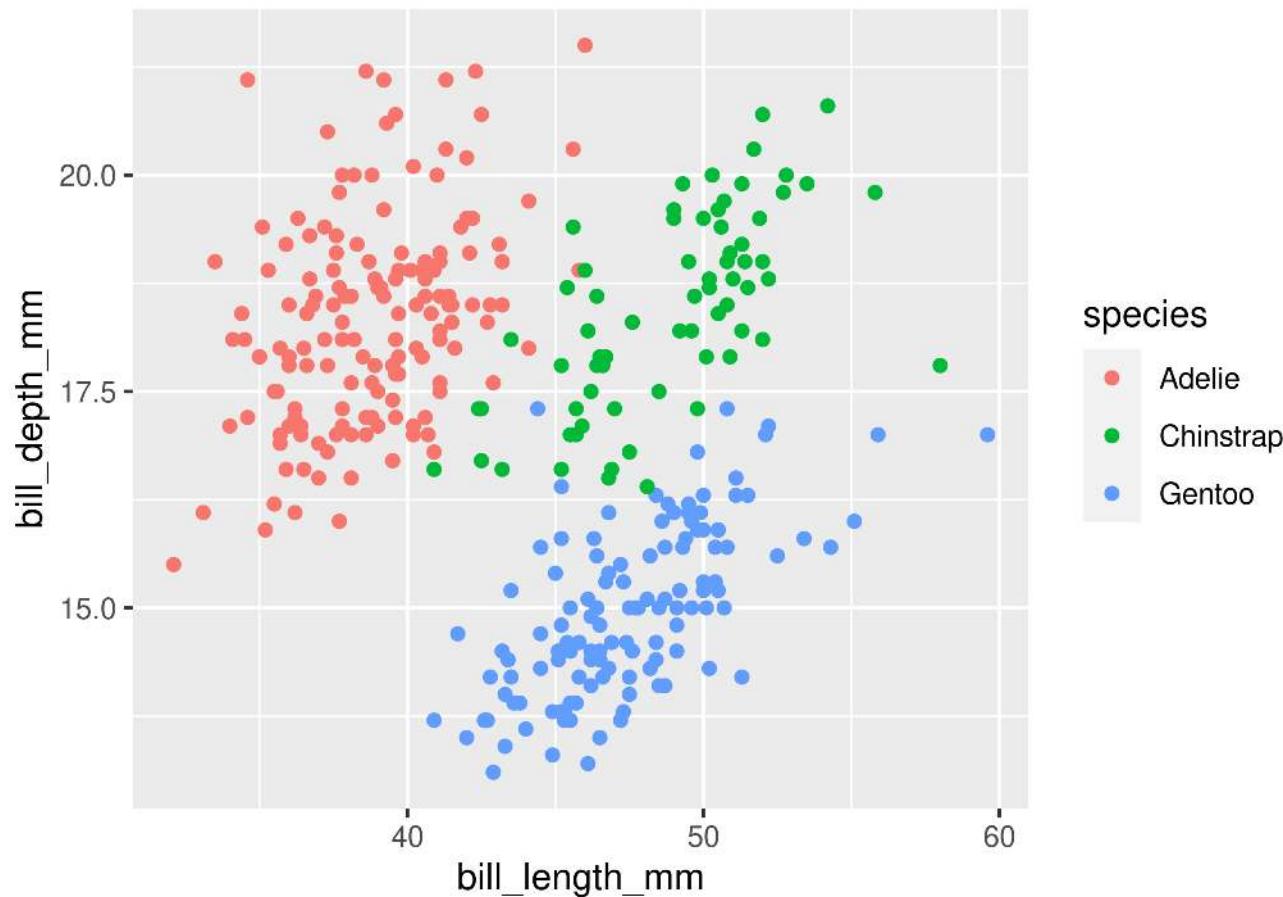
- Load **patchwork**
- Create a couple of different plots

```
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

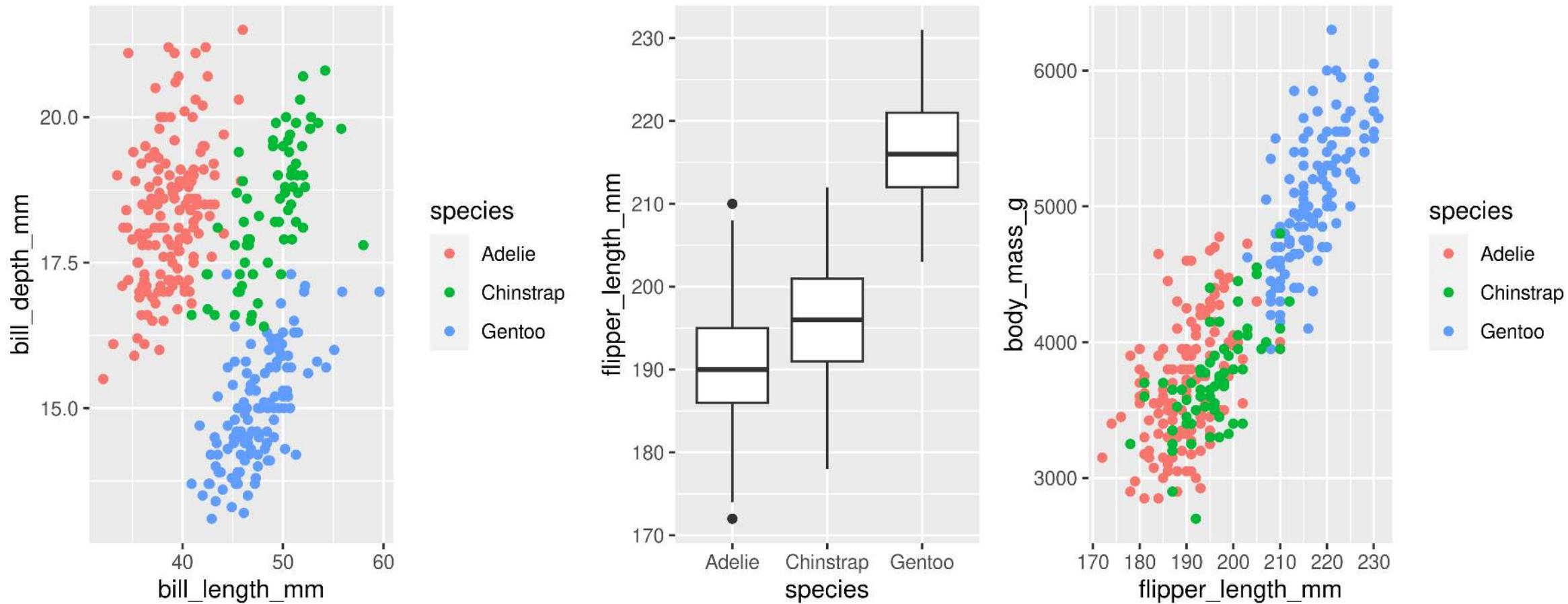
```
1 g1 + g2
```



# Combining plots with patchwork

## Side-by-Side 3 plots

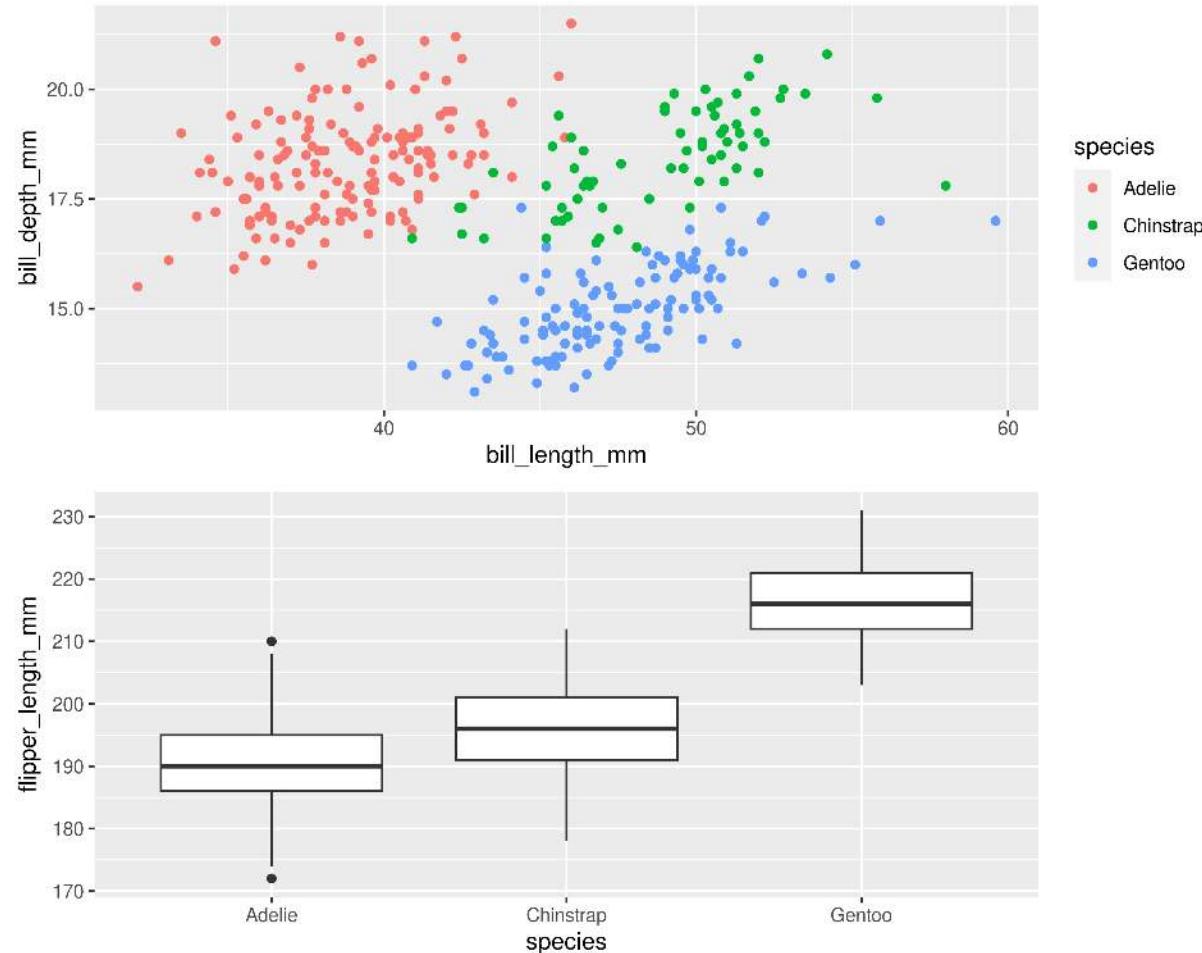
```
1 g1 + g2 + g3
```



# Combining plots with patchwork

## Stacked 2 plots

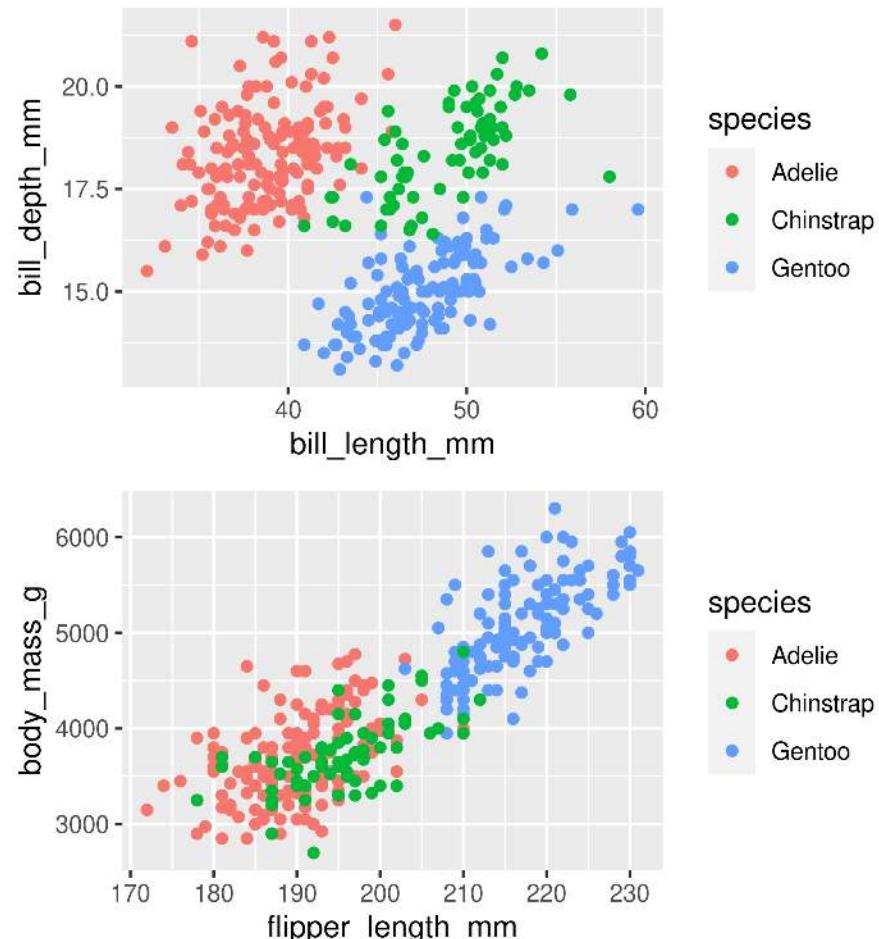
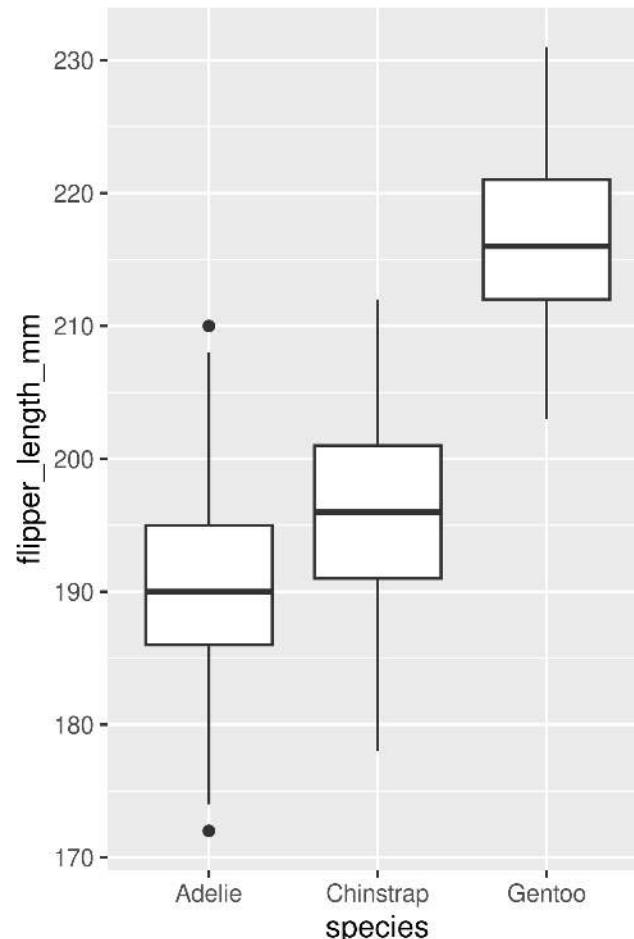
```
1 g1 / g2
```



# Combining plots with patchwork

## More complex arrangements

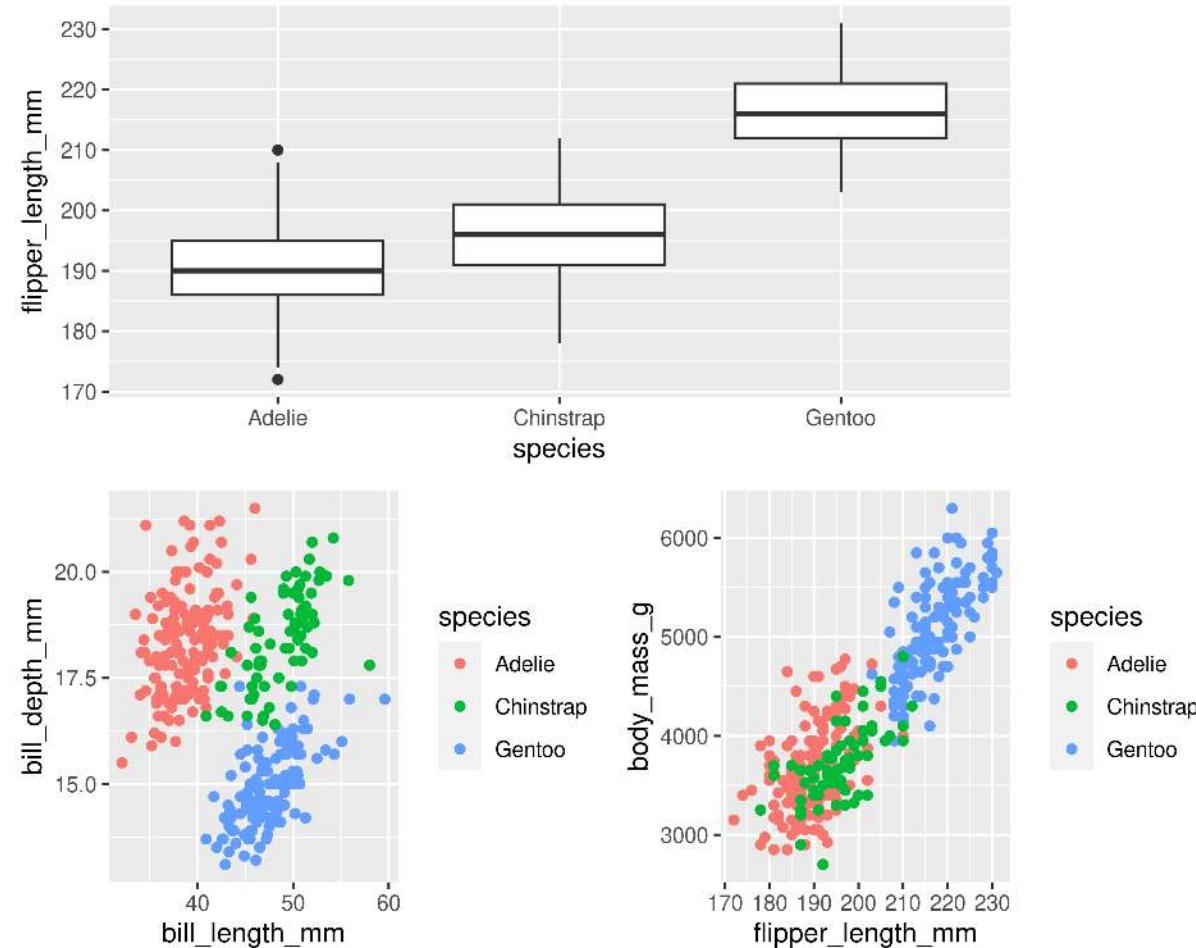
```
1 g2 + (g1 / g3)
```



# Combining plots with patchwork

## More complex arrangements

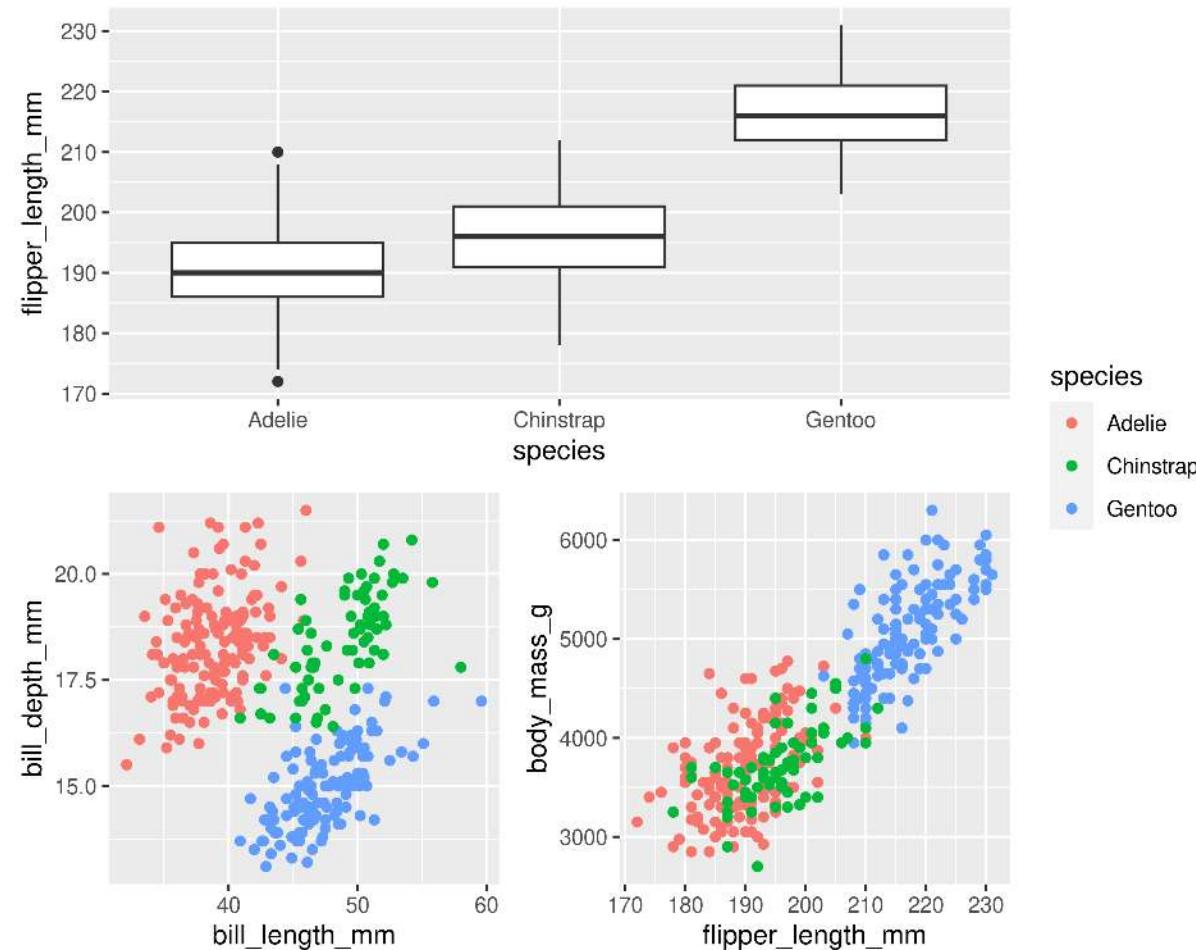
```
1 g2 / (g1 + g3)
```



# Combining plots with patchwork

“collect” common legends

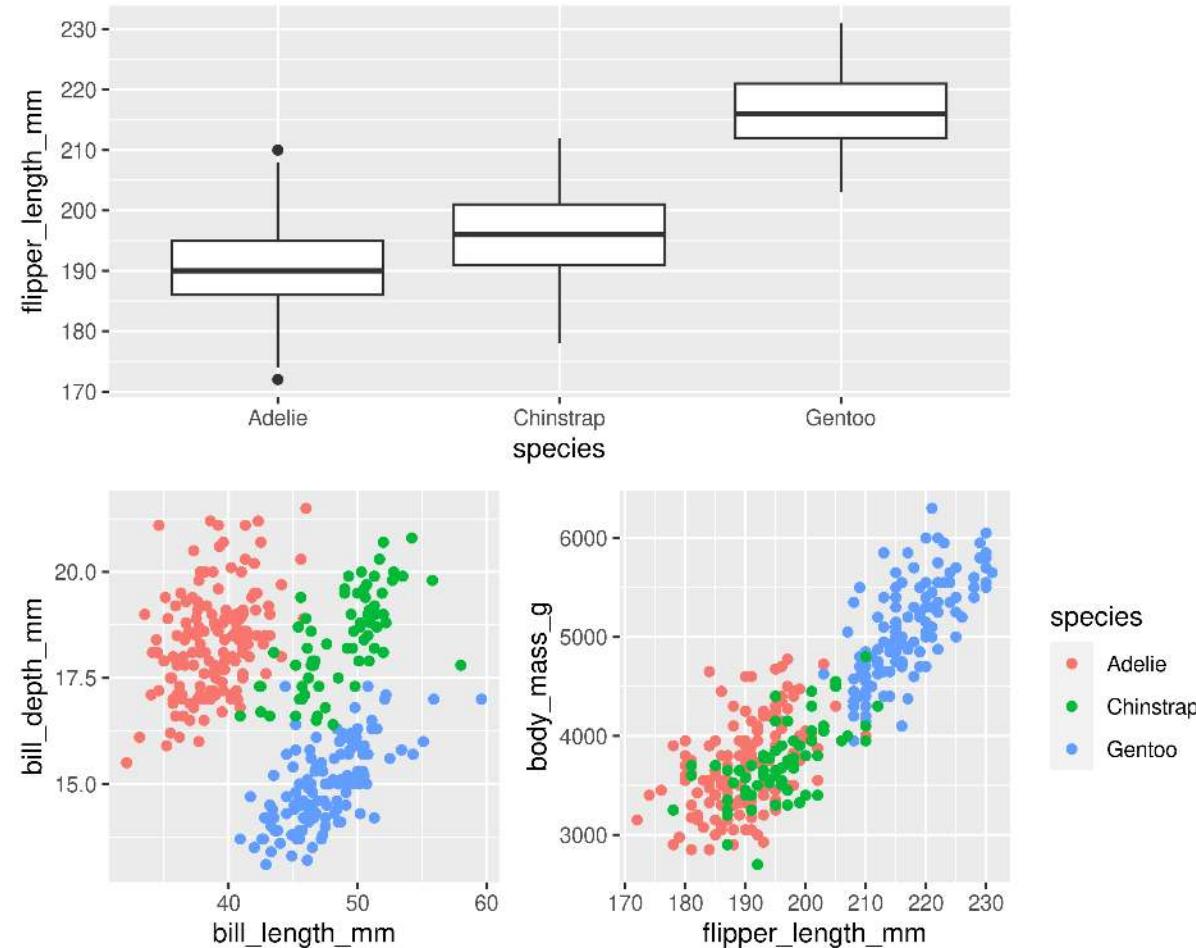
```
1 g2 / (g1 + g3) + plot_layout(guides = "collect")
```



# 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 = ")")
```

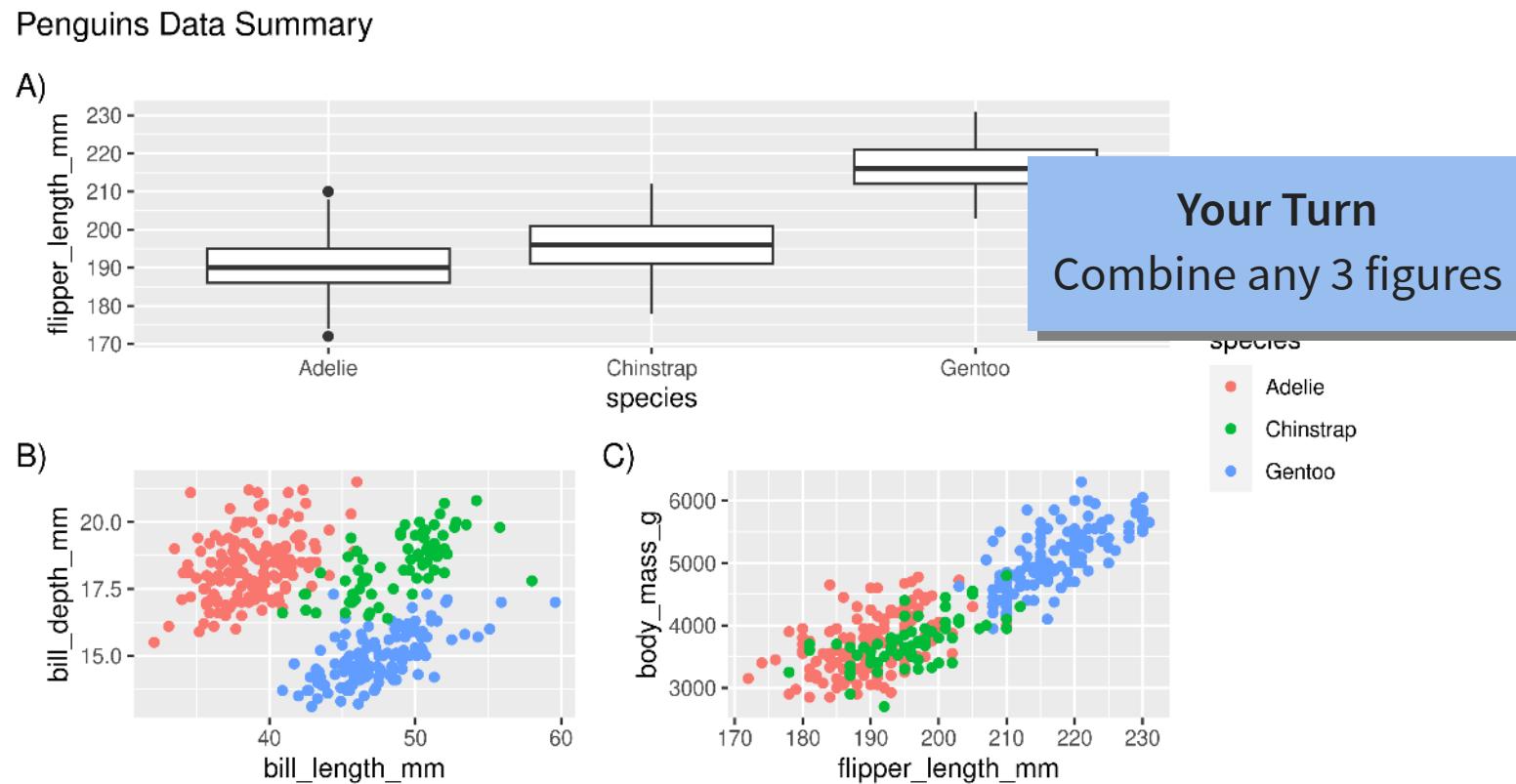


Fig 1. Penguins Data Summary



# Saving plots

# Saving plots

## RStudio Export

*Demo*

### ggsave()

```
1 g <- ggplot(penguins, aes(x = sex, y = bill_length_mm)) +  
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: 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: Common mistakes

## I get an error regarding an object that can't be found or aesthetic length?

You are probably trying to plot two different datasets, and you make references to variables in the `ggplot()` call that don't exist in one of the datasets:

```
1 n <- count(penguins, island)
2
3 ggplot(data = penguins, aes(x = flipper_length_mm, y = bill_length_mm, colour = species)) +
4   geom_point() +
5   facet_wrap(~ island) +
6   geom_text(data = n, aes(label = n),
7             x = -Inf, y = +Inf, hjust = 0, vjust = 1)
```

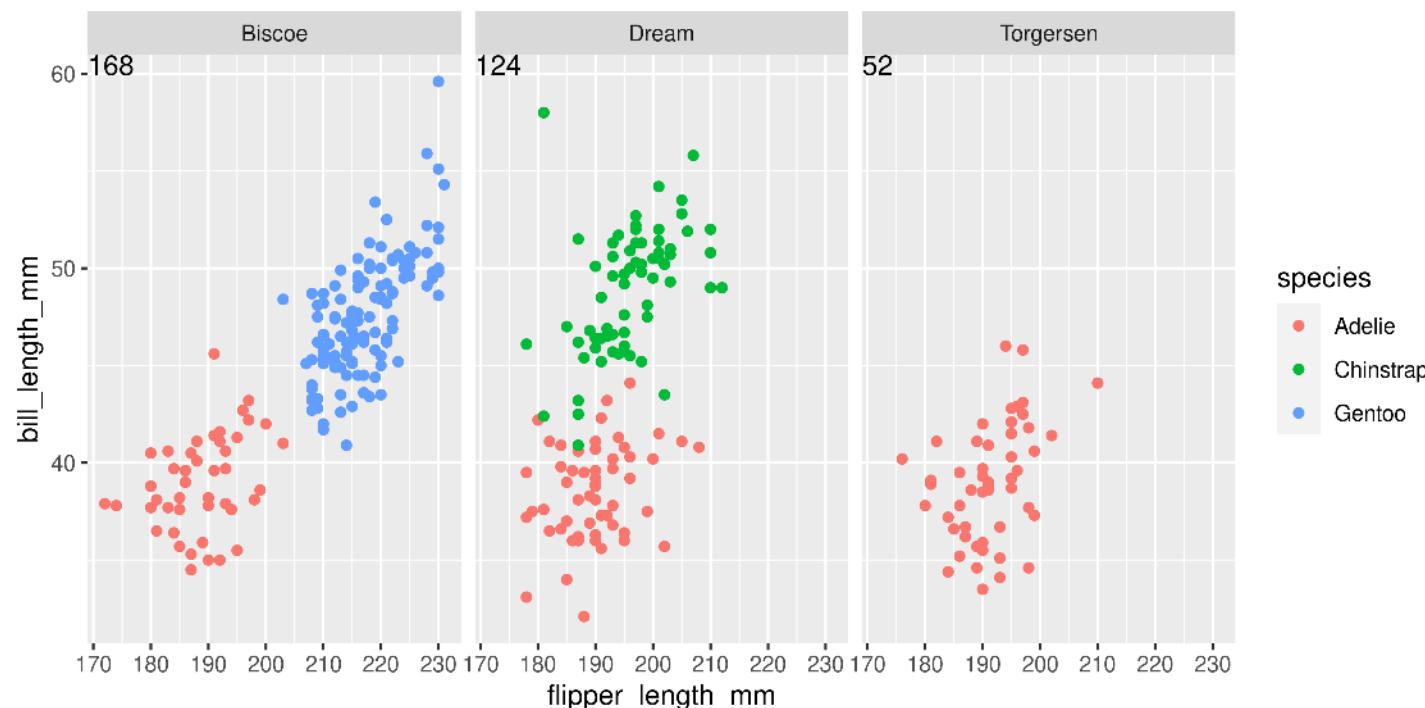
```
Error in `geom_text()`:
! Problem while computing aesthetics.
  i Error occurred in the 2nd layer.
Caused by error in `FUN()`:
! object 'species' not found
```

# Wrapping up: Common mistakes

I get an error regarding an object that can't be found or aesthetic length?

Either move the aesthetic...

```
1 ggplot(penguins, aes(x = flipper_length_mm, y = bill_length_mm)) +  
2   geom_point(aes(colour = species)) +  
3   facet_wrap(~ island) +  
4   geom_text(data = n, aes(label = n),  
5             x = -Inf, y = +Inf, hjust = 0, vjust = 1)
```



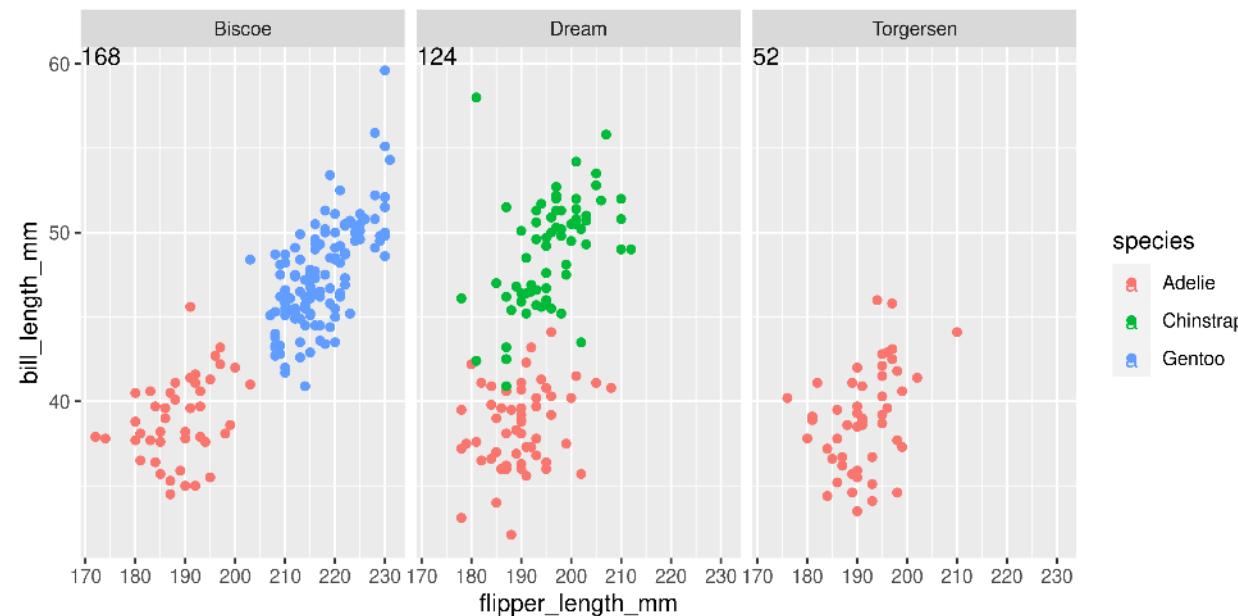
# Wrapping up: Common mistakes

I get an error regarding an object that can't be found or aesthetic length?

Either move the aesthetic...

Or assign it to **NULL** where it is missing...

```
1 ggplot(penguins, aes(x = flipper_length_mm, y = bill_length_mm, colour = species)) +  
2   geom_point() +  
3   facet_wrap(~ island) +  
4   geom_text(data = n, aes(label = n, colour = NULL),  
5             x = -Inf, y = +Inf, hjust = 0, vjust = 1)
```



# Wrapping up: Further reading (all Free!)

- RStudio > Help > Cheatsheets > Data Visualization with ggplot2
- [ggplot2 book v3](#) by Hadley Wickham, Danielle Navarro, and Thomas Lin Pedersen
- [patchwork website](#)
- [Cookbook for R](#) by Winston Chang
- [R for Data Science](#) by Hadley Wickham and Garret Grolemund
  - [Chapter on Data Visualization](#)
- [Data Visualization: A practical introduction](#) by Kieran Healy

