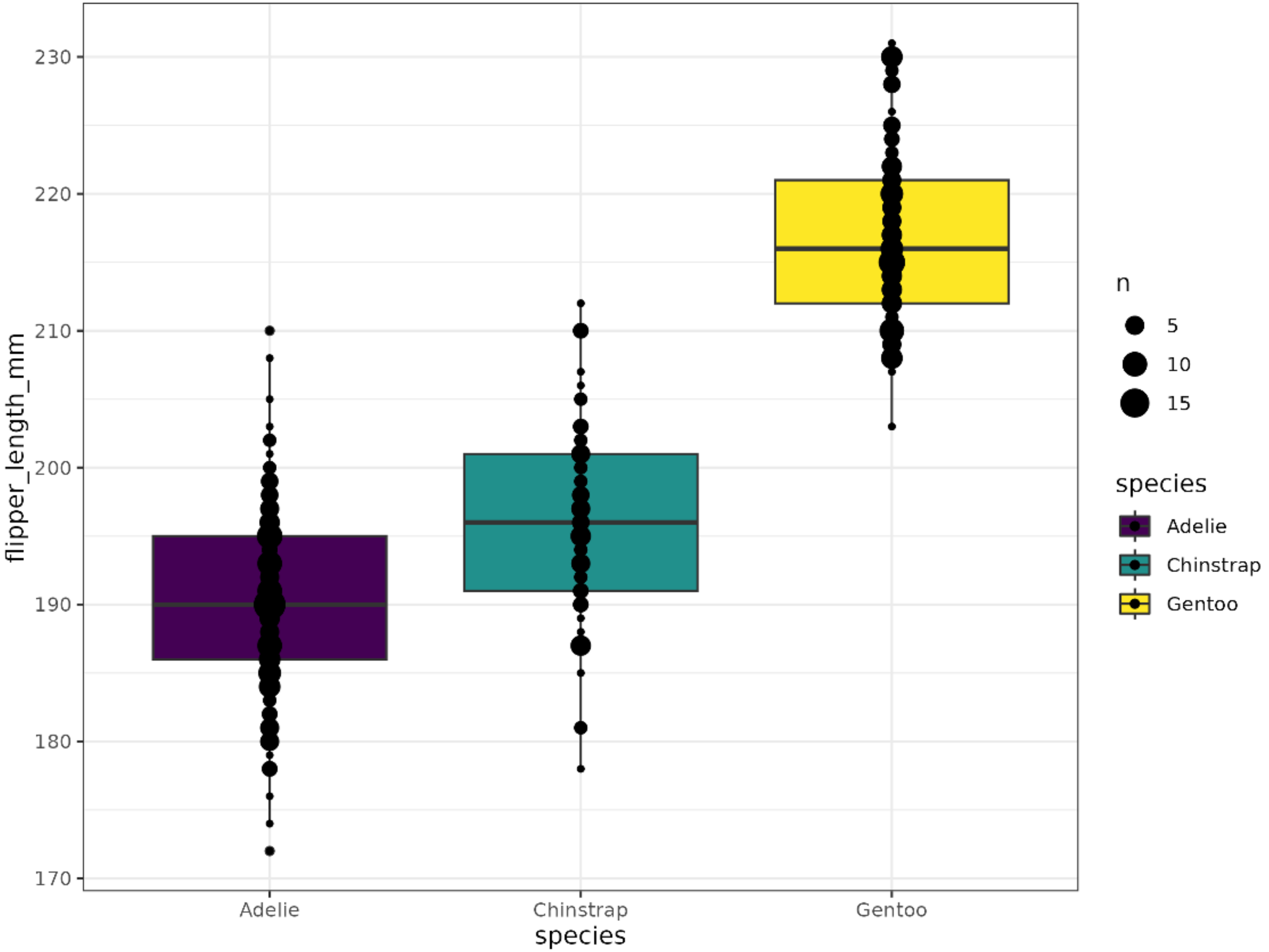
Visualizing Data in R

A primer on `ggplot2`



steffilazerte
@steffilazerte@fosstodon.org
@steffilazerte
steffilazerte.ca

Dr. Steffi LaZerte
Analysis and Data Tools for Science

Compiled: 2024-02-21

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)

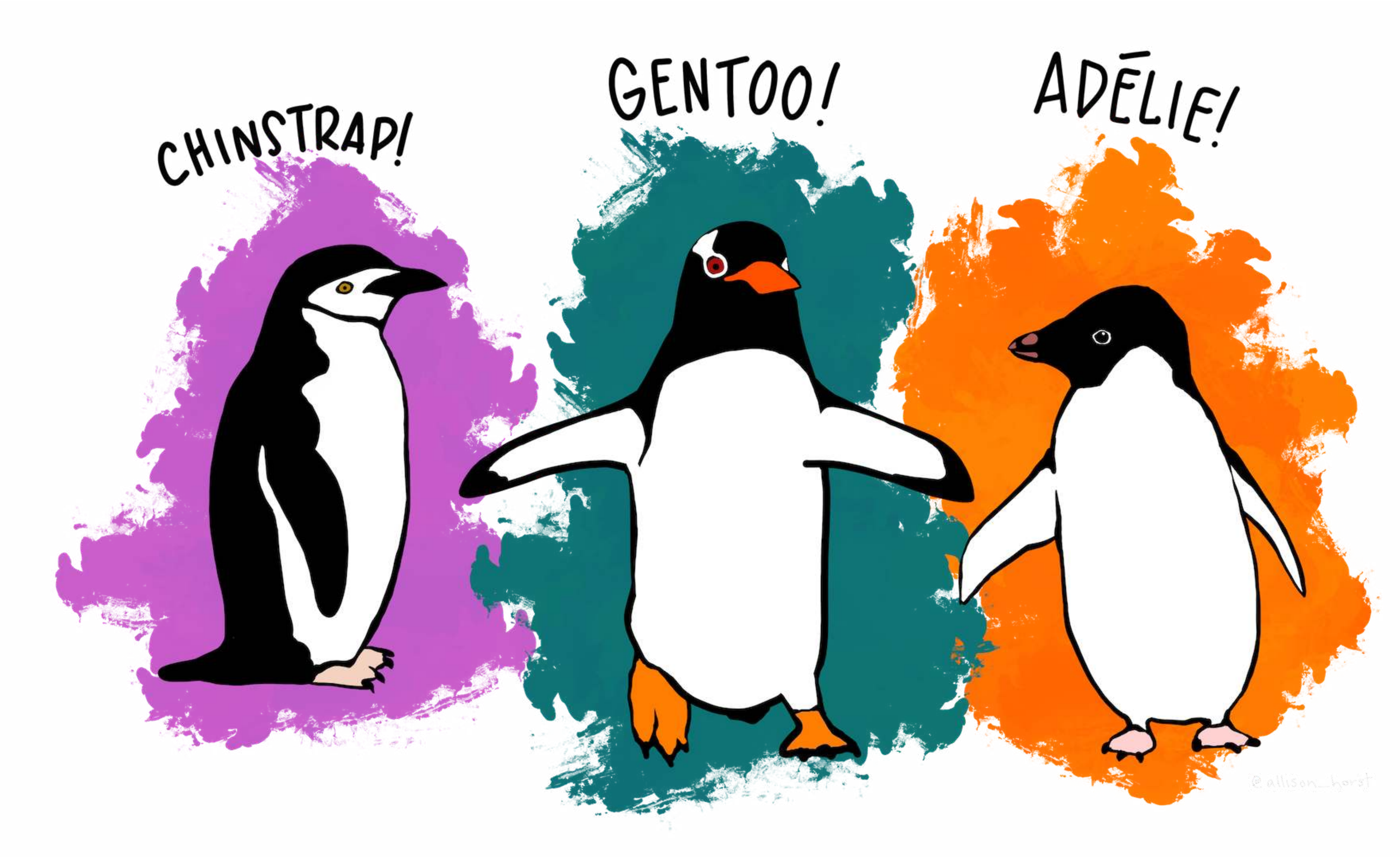
- Basic plot
- Common plot types
- Plotting by categories
- Adding statistics
- Customizing plots
- Annotating plots

2. Combining figures with `patchwork`

3. Saving figures


ggplot2:
Build a data
MASTERPIECE
me_Minimal
classic
me_bw
THEMES
gEOMS
aes
fill
color
gganimate
plotly
HORST '18

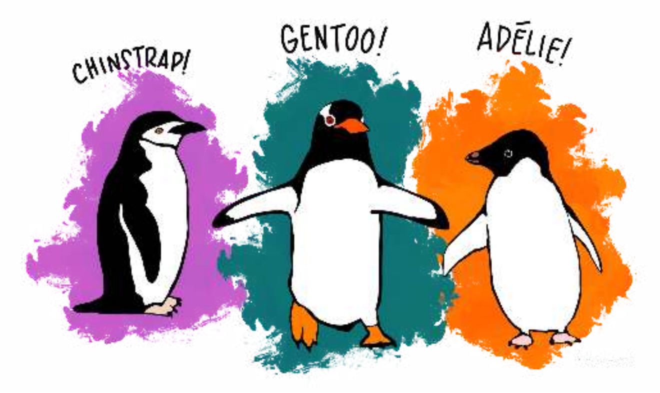
Our data set: Palmer Penguins!



Our data set: Palmer Penguins!

```r
library(palmerpenguins)
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

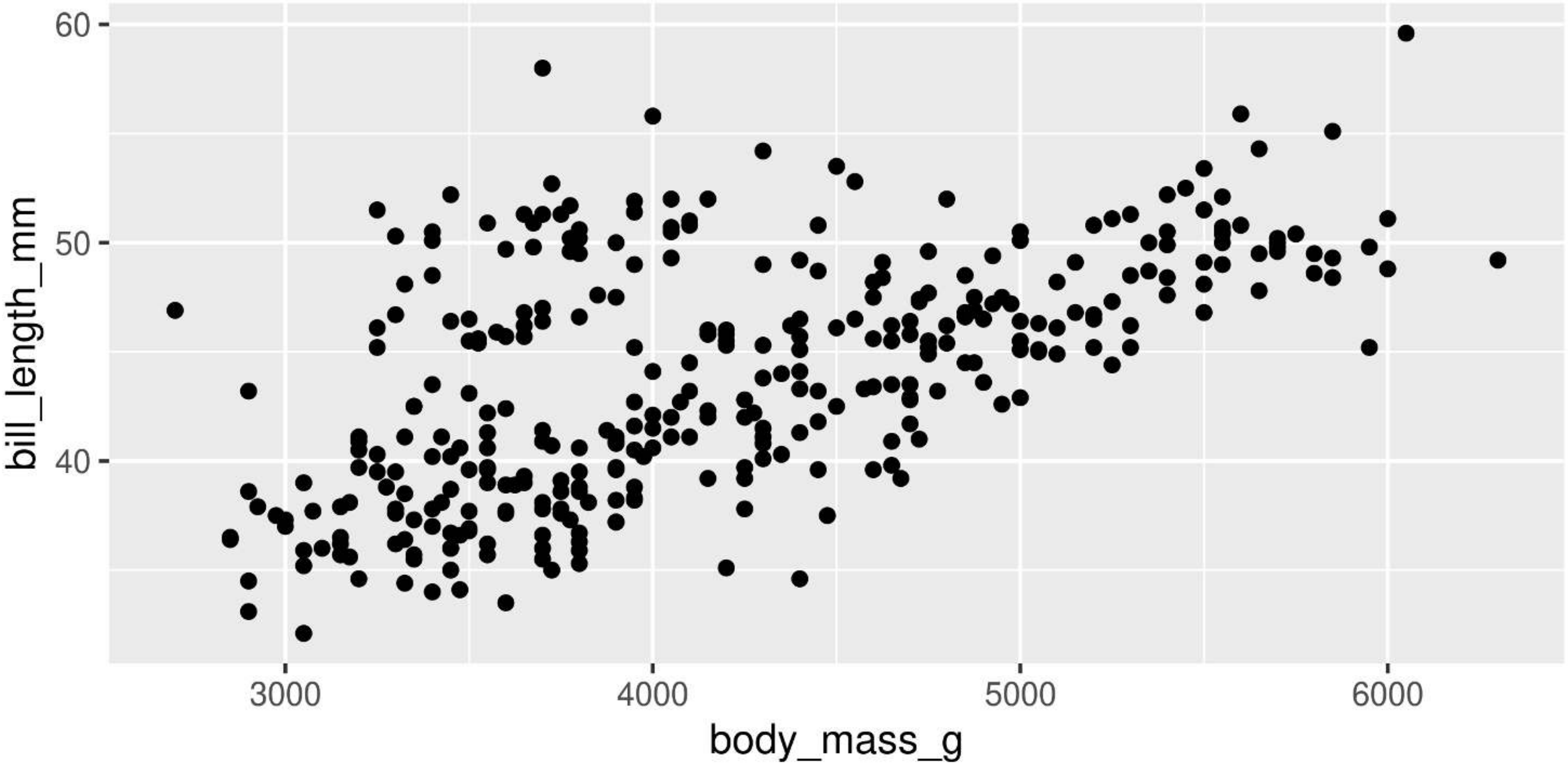
Side Note

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

```r
library(palmerpenguins)
library(tidyverse)

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

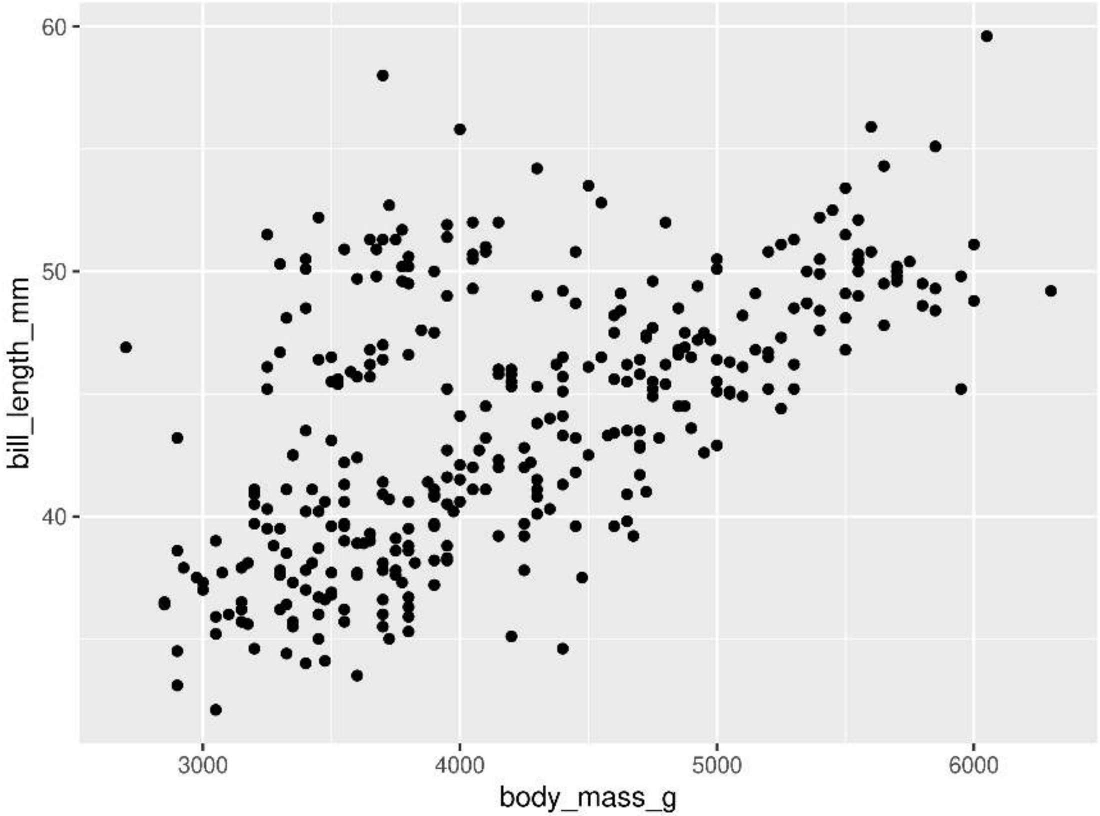


Break it down

```r
library(palmerpenguins)
library(tidyverse)

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

`library()`

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

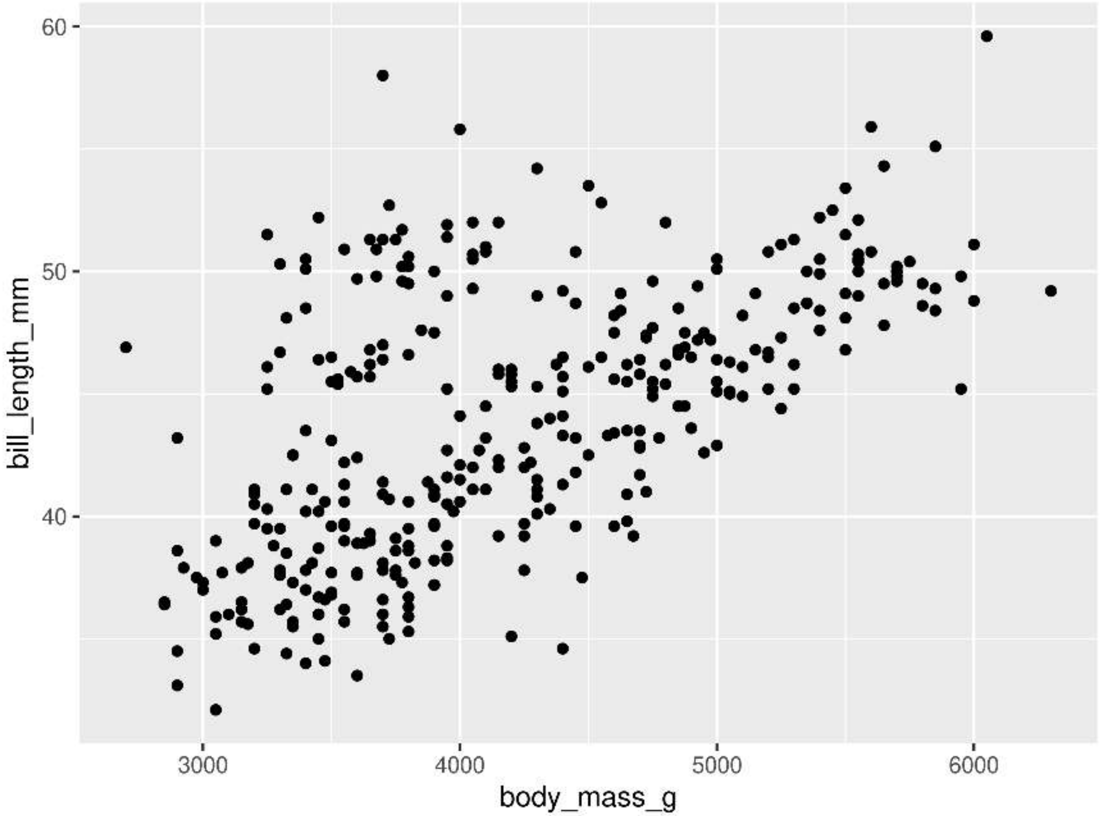


Break it down

```r
library(palmerpenguins)
library(tidyverse)

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

`library()`

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

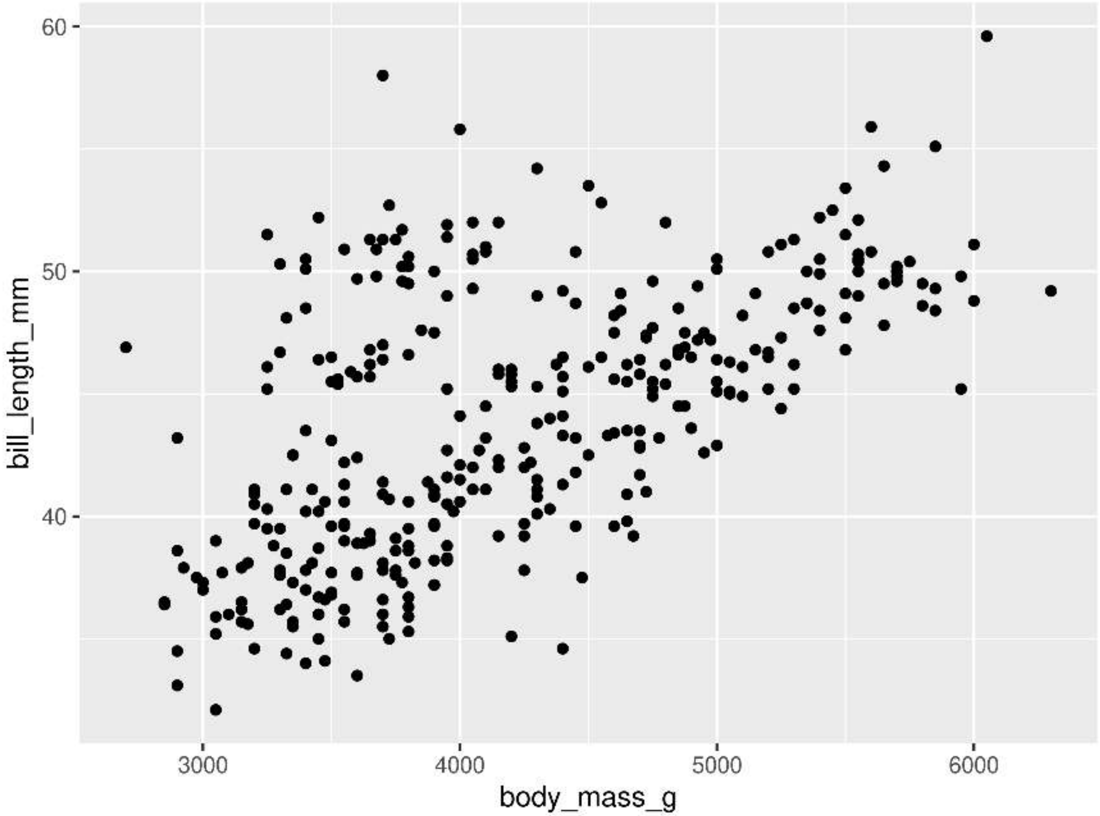


Break it down

```r
library(palmerpenguins)
library(tidyverse)

ggplot(data = penguins, aes(x = body_mass_g, y = bill_length_mm)) +
    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



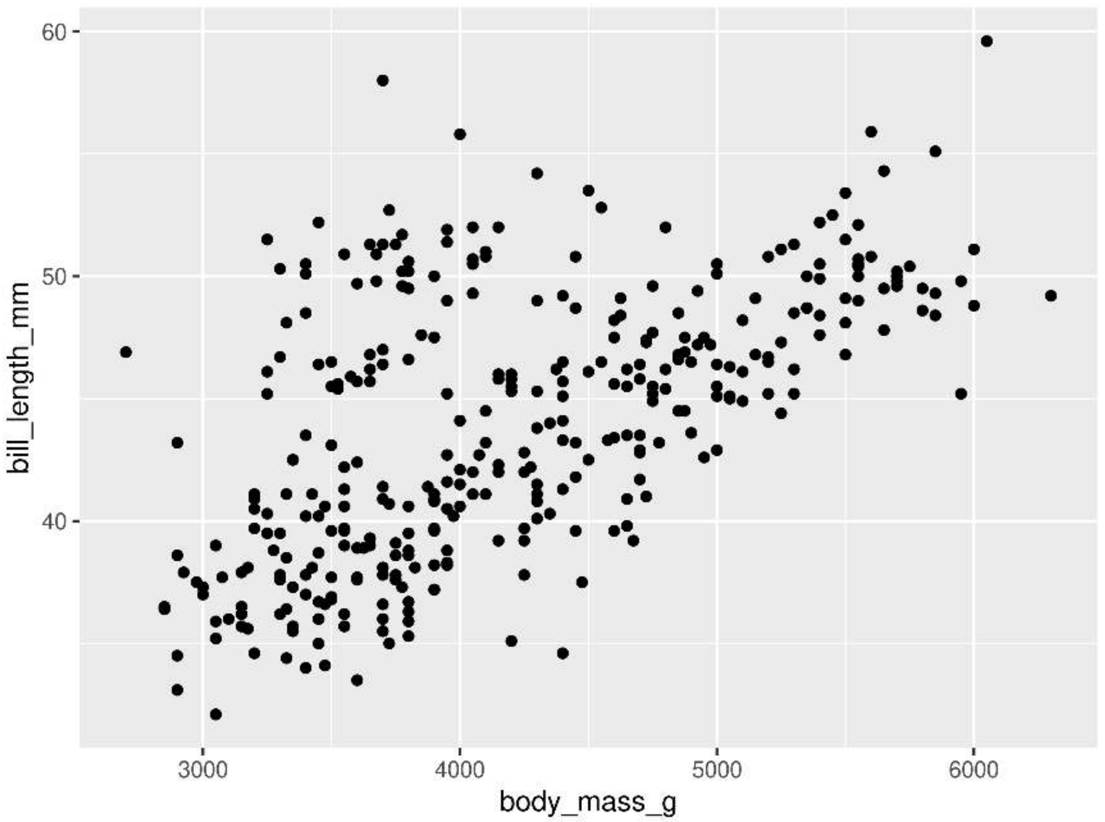
Break it down

```r
library(palmerpenguins)
library(tidyverse)

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

`geom_point()`

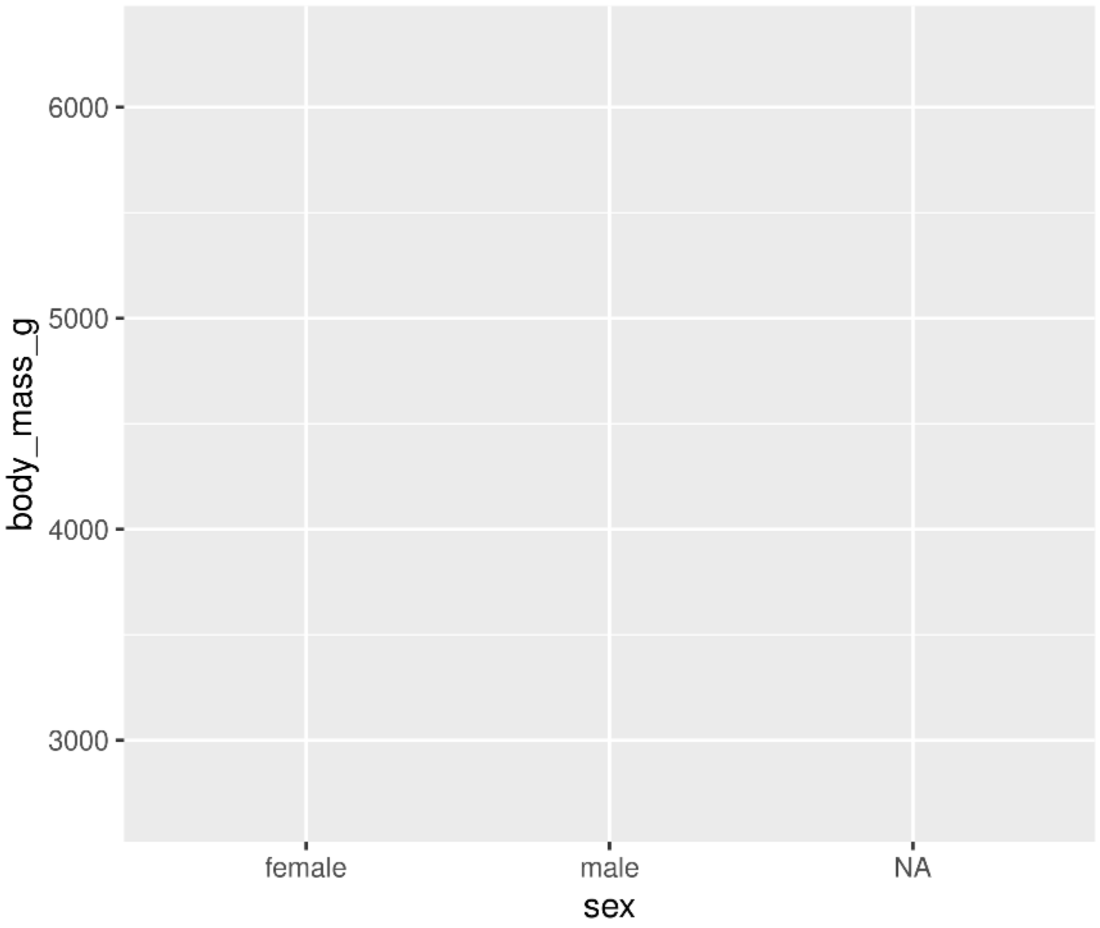
- Choose a `geom` function to display the data
- Always *added* to a `ggplot()` call with `+`

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

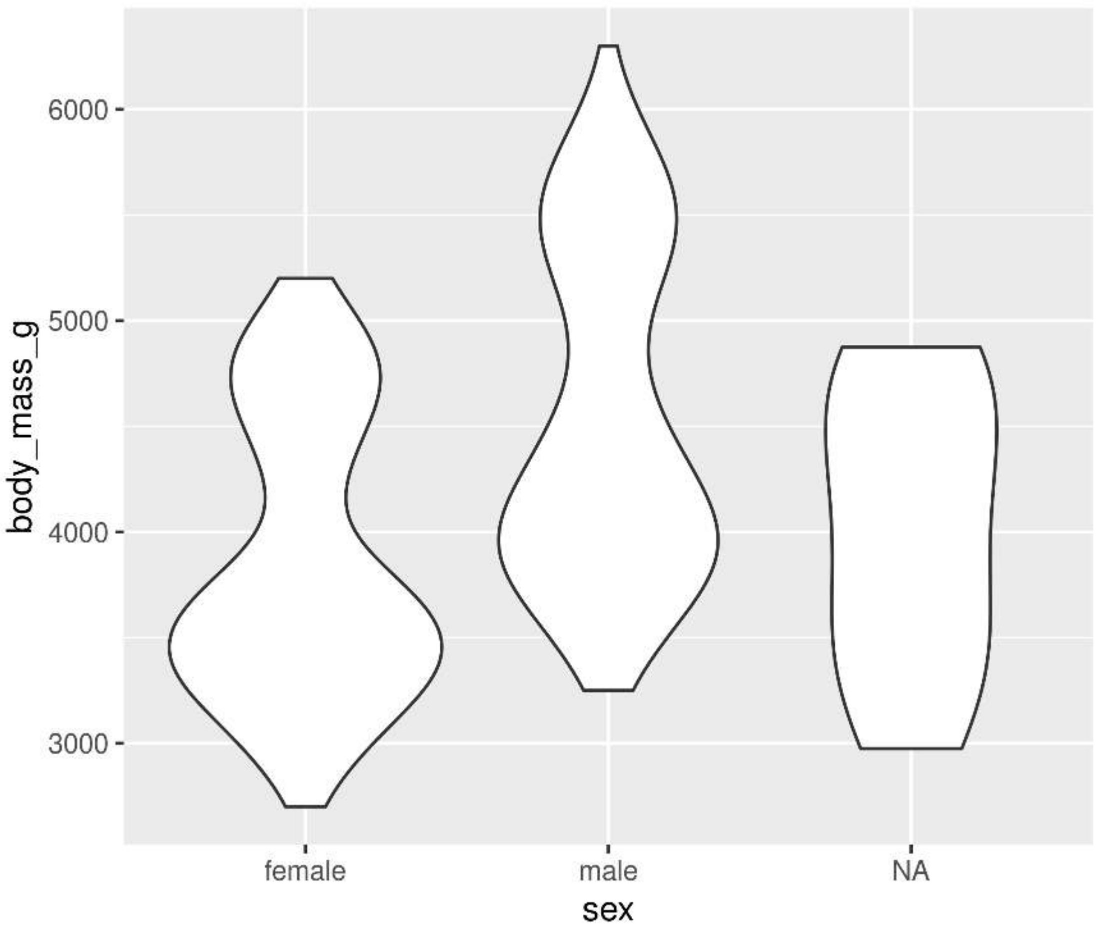


Plots are layered

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



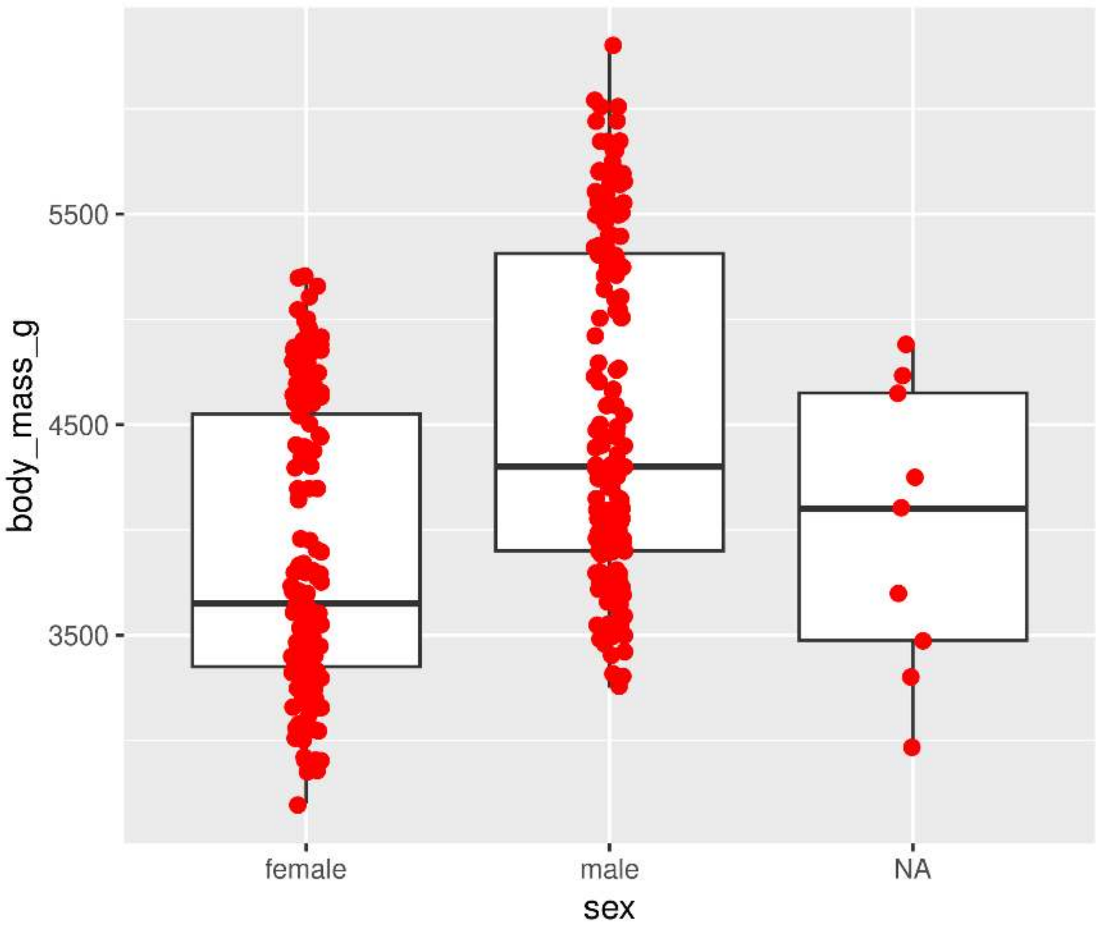
```r
ggplot(data = penguins, aes(x = sex, y = body_mass_g)) +
  geom_violin()
```



Plots are layered

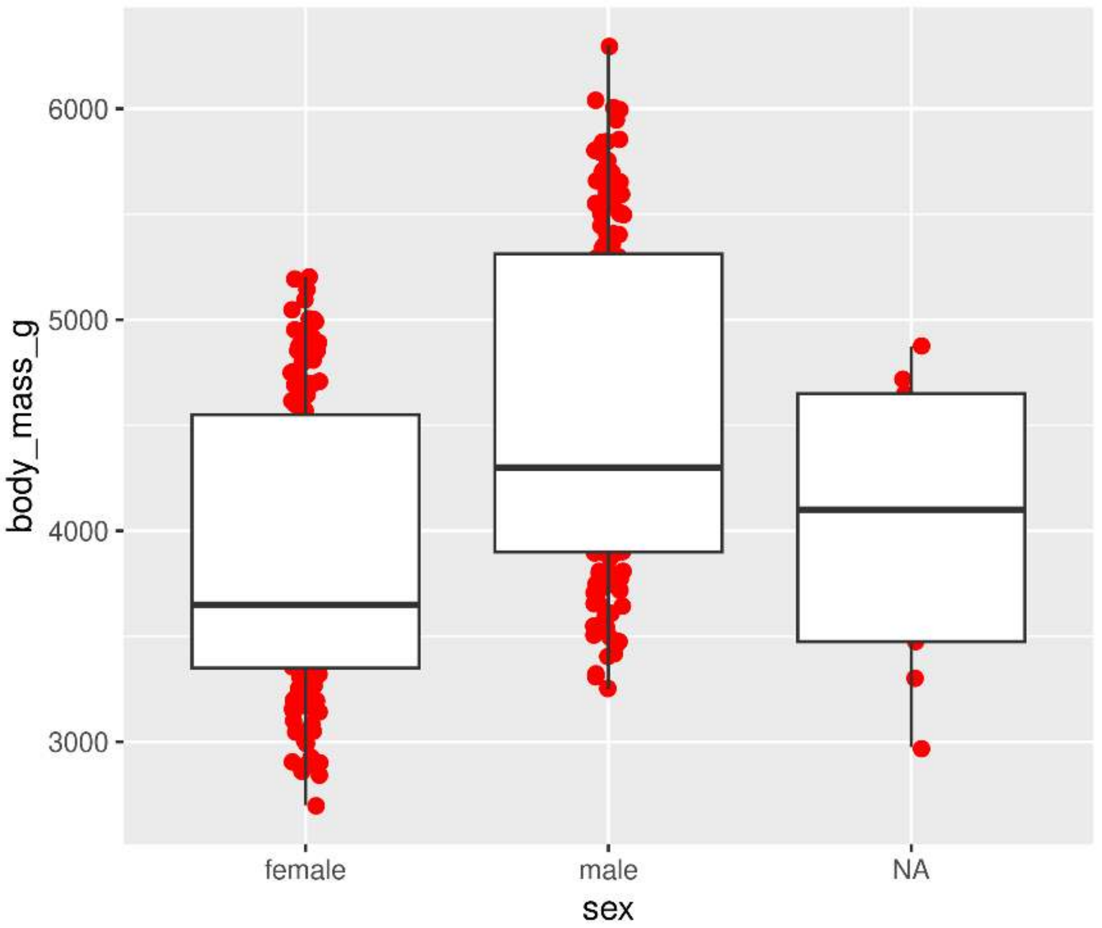
You can add multiple layers

```r
ggplot(data = penguins, aes(x = sex, y = body_mass_g)) +
  geom_boxplot() +
  geom_point(size = 2, colour = "red",
             position = position_jitter(width = 0.05))
```



Order matters

```r
ggplot(data = penguins, aes(x = sex, y = body_mass_g)) +
  geom_point(size = 2, colour = "red",
             position = position_jitter(width = 0.05)) +
  geom_boxplot()
```

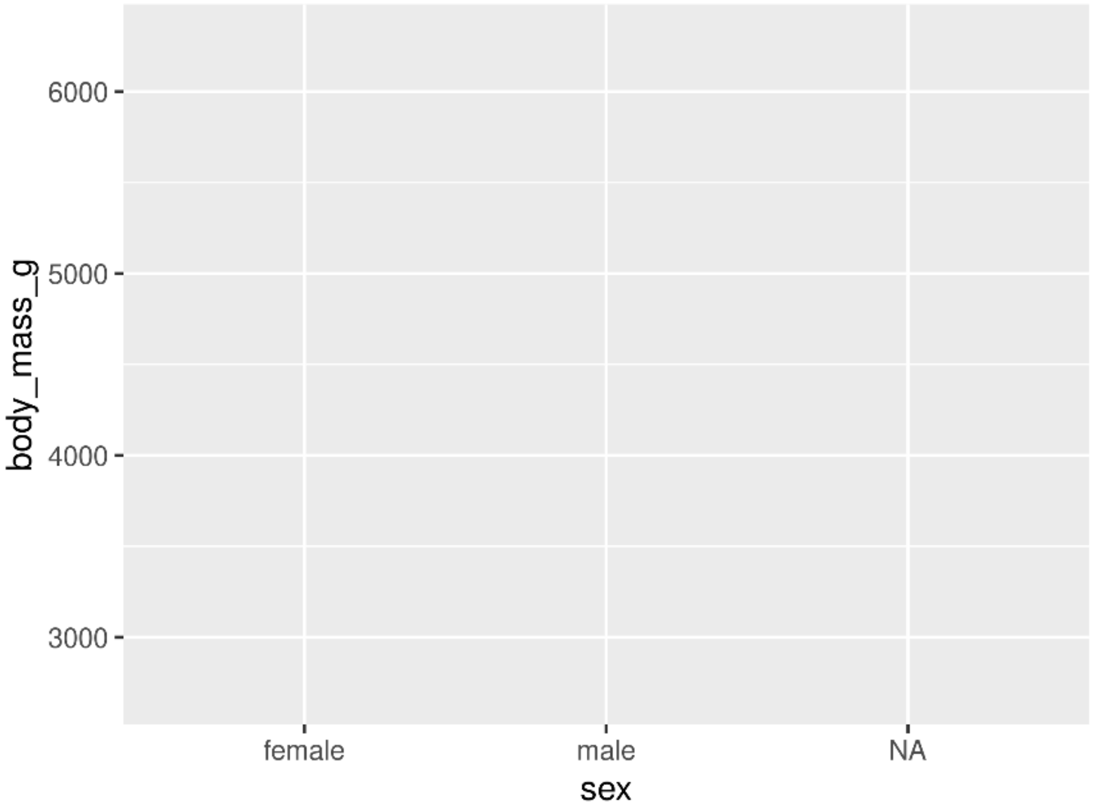


Plots are objects

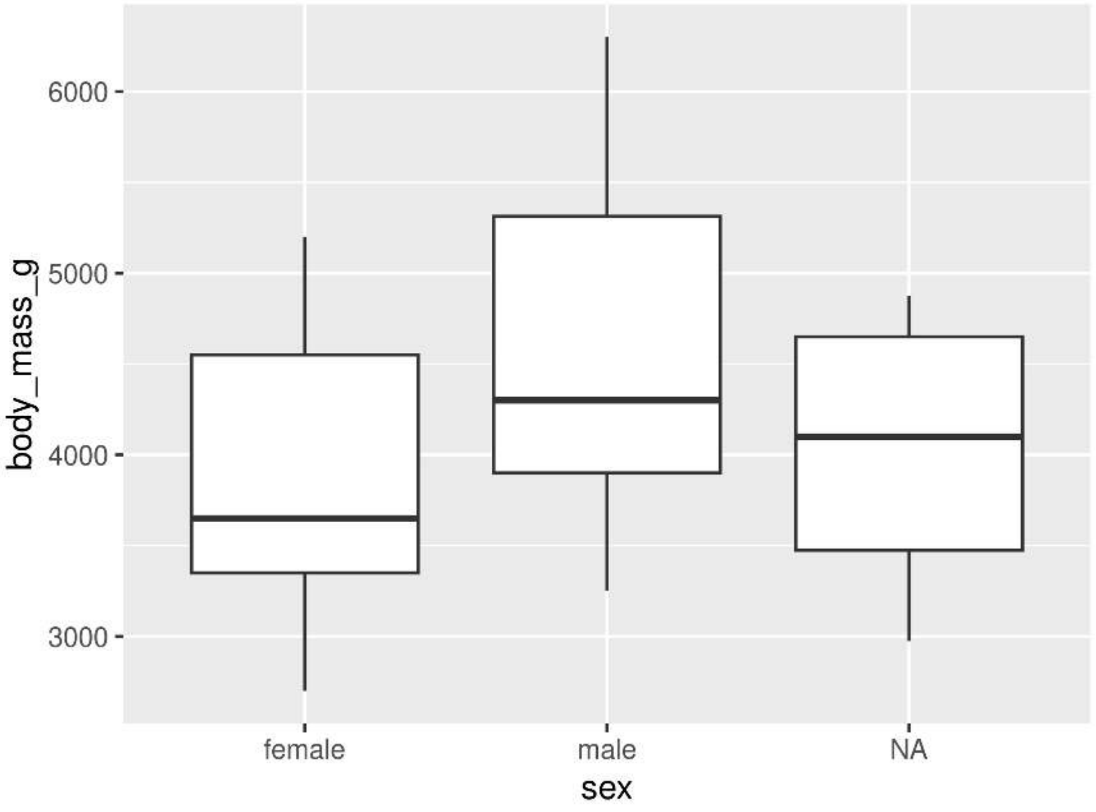
Any ggplot can be saved as an object

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

```r
g
```



```r
g + geom_boxplot()
```

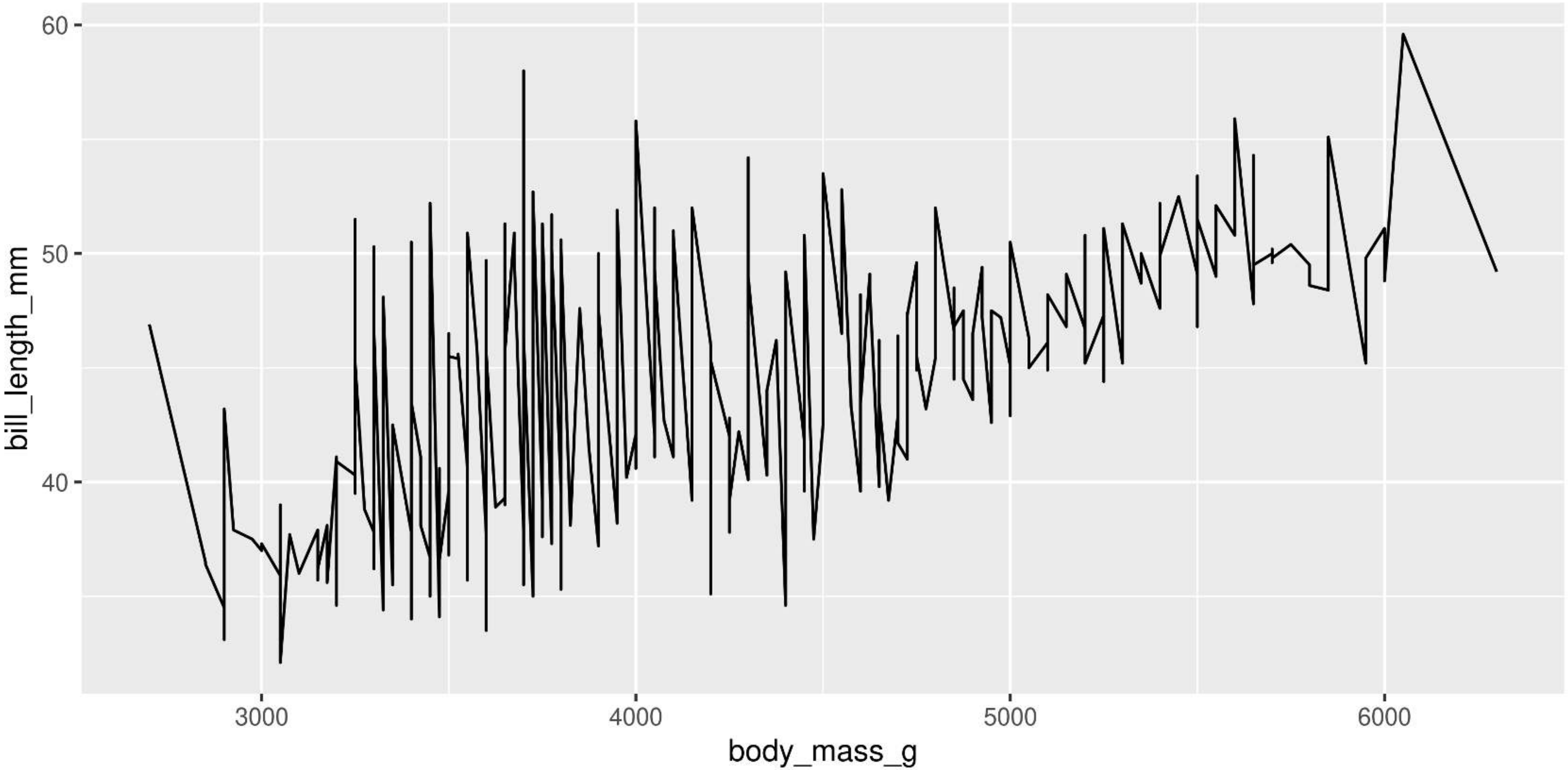


More Geoms

(Plot types)

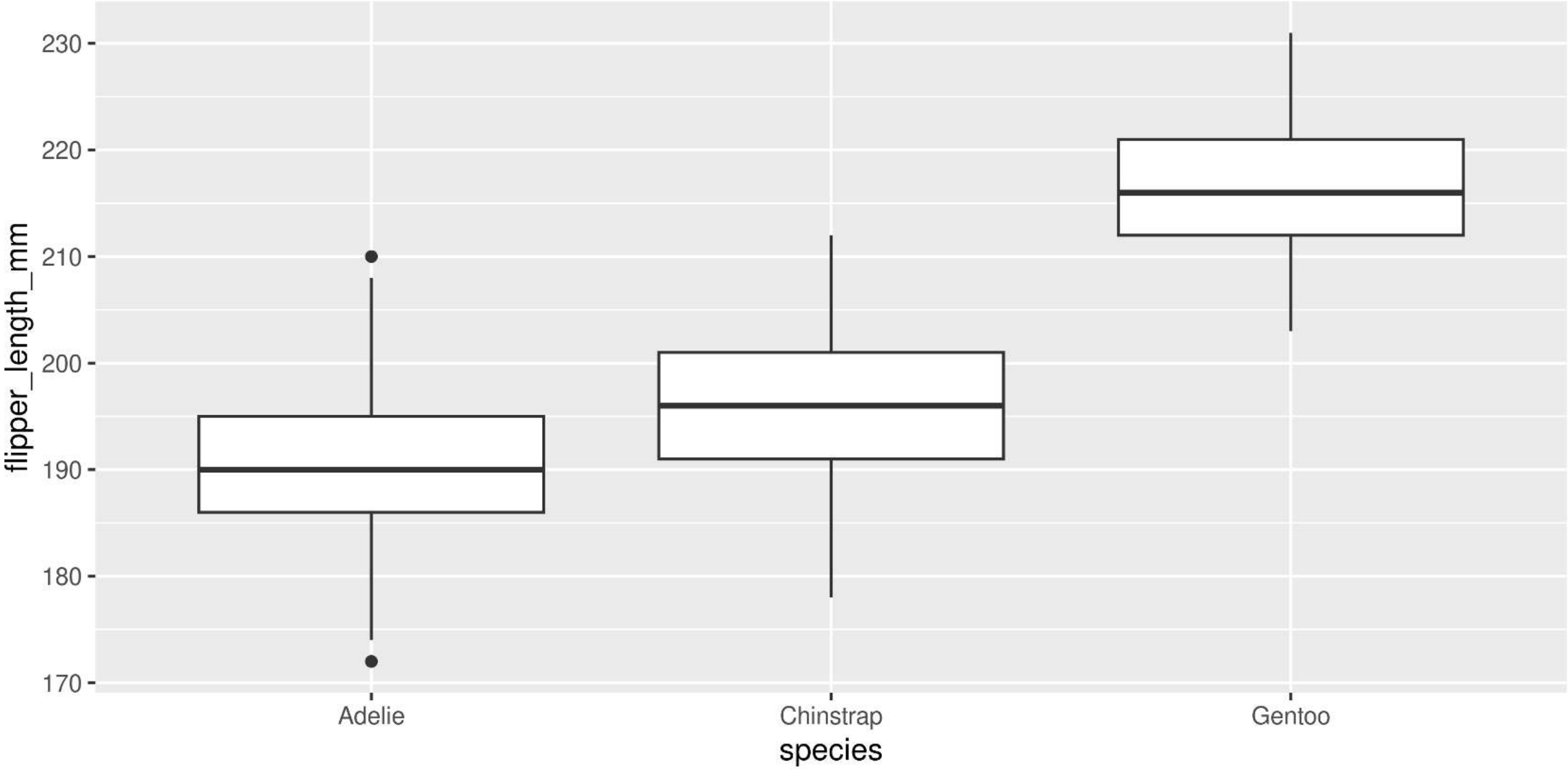
Geoms: Lines

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



Geoms: Boxplots

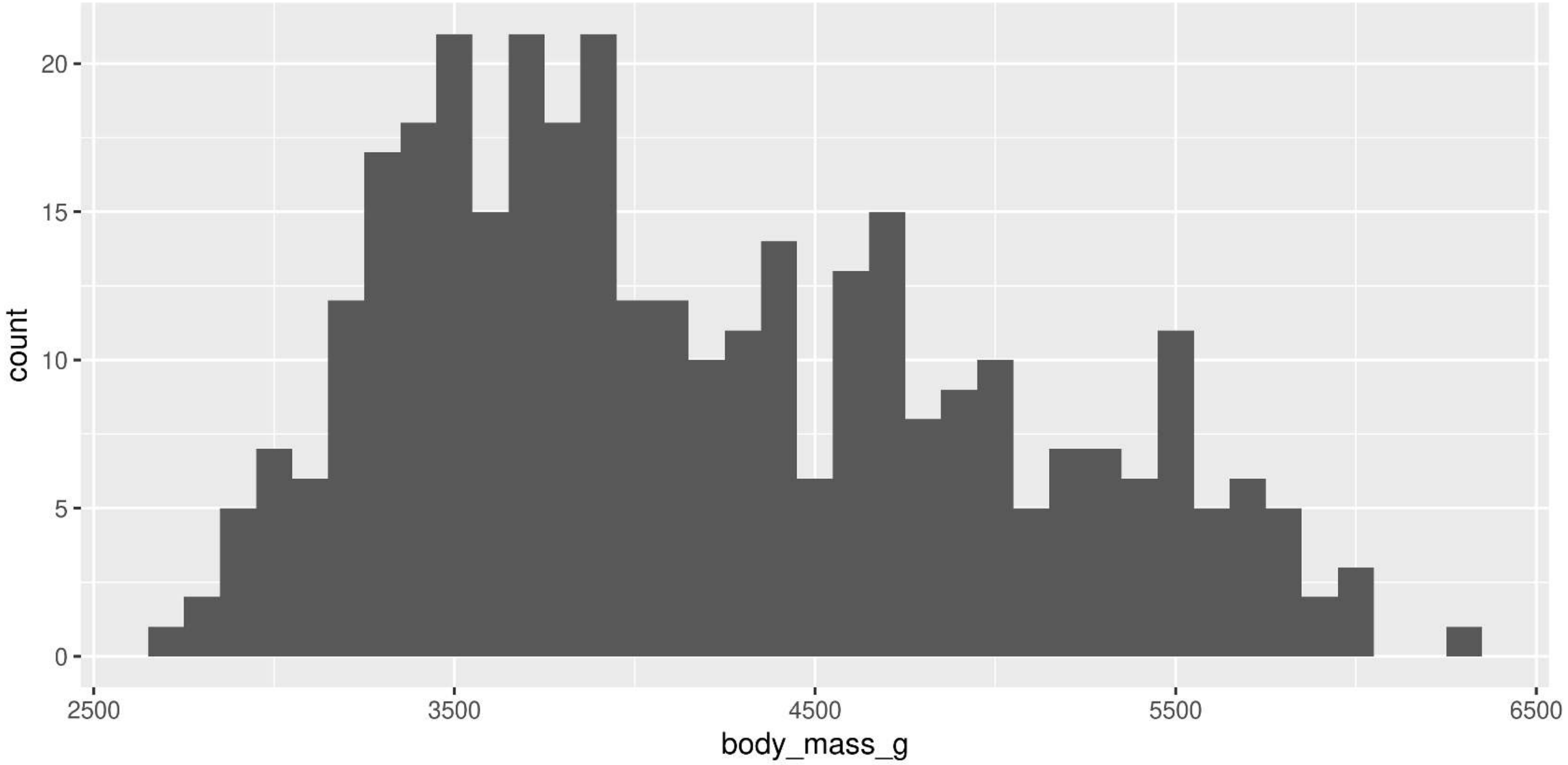
```r
ggplot(data = penguins, aes(x = species, y = flipper_length_mm)) +
  geom_boxplot()
```



Geoms: Histogram

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

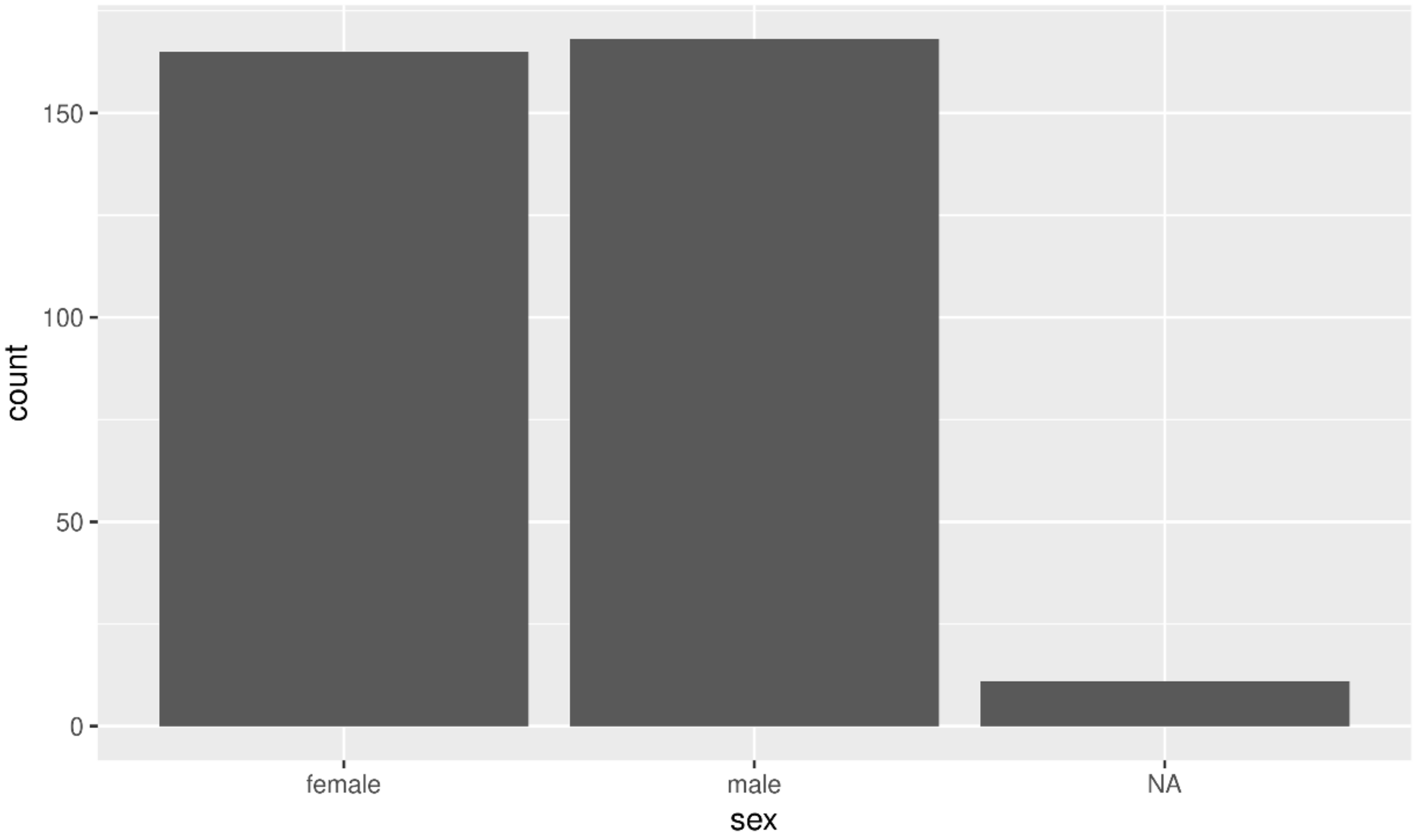
Note:
We only need 1 aesthetic here



Geoms: Barplots

Let `ggplot` count your data

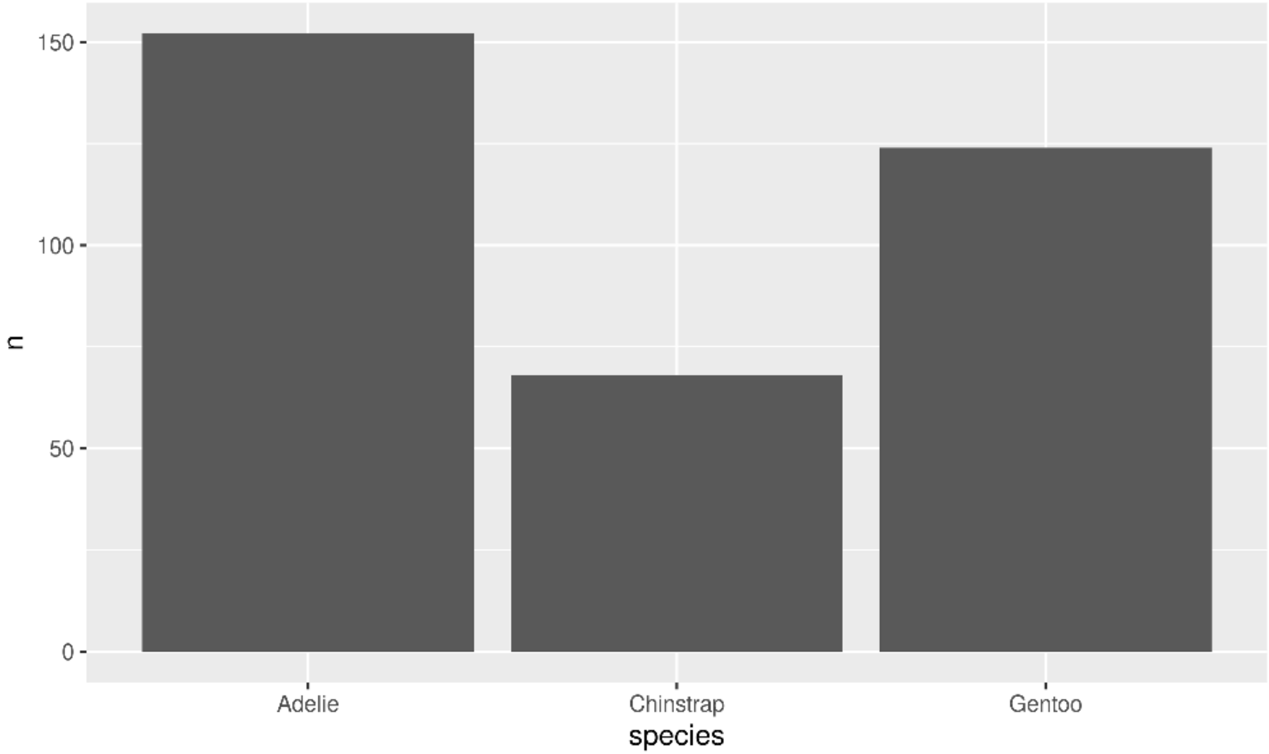
```r
ggplot(data = penguins, aes(x = sex)) +
  geom_bar()
```



Geoms: Barplots

You can also provide the counts

```r
# Create our own data frame
species_counts <- data.frame(species = c("Adelie", "Chinstrap", "Gentoo"),
                             n = c(152, 68, 124))

ggplot(data = species_counts, aes(x = species, y = n)) +
  geom_bar(stat = "identity")
```



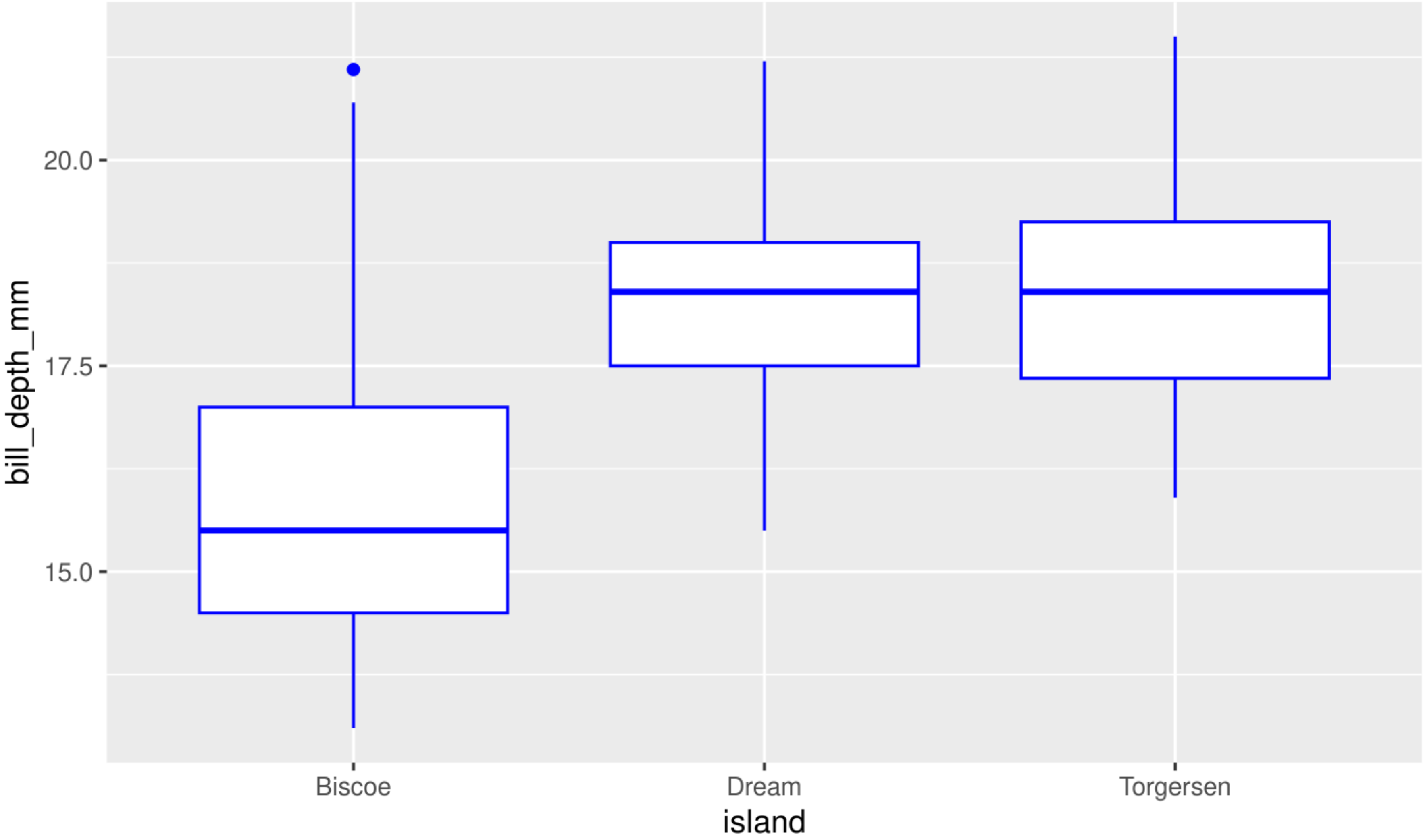
Your Turn: Create this plot

```
ggplot(data = ____, aes(x = ____, y = ____)) +
  geom_____(____)
```

Too Easy?

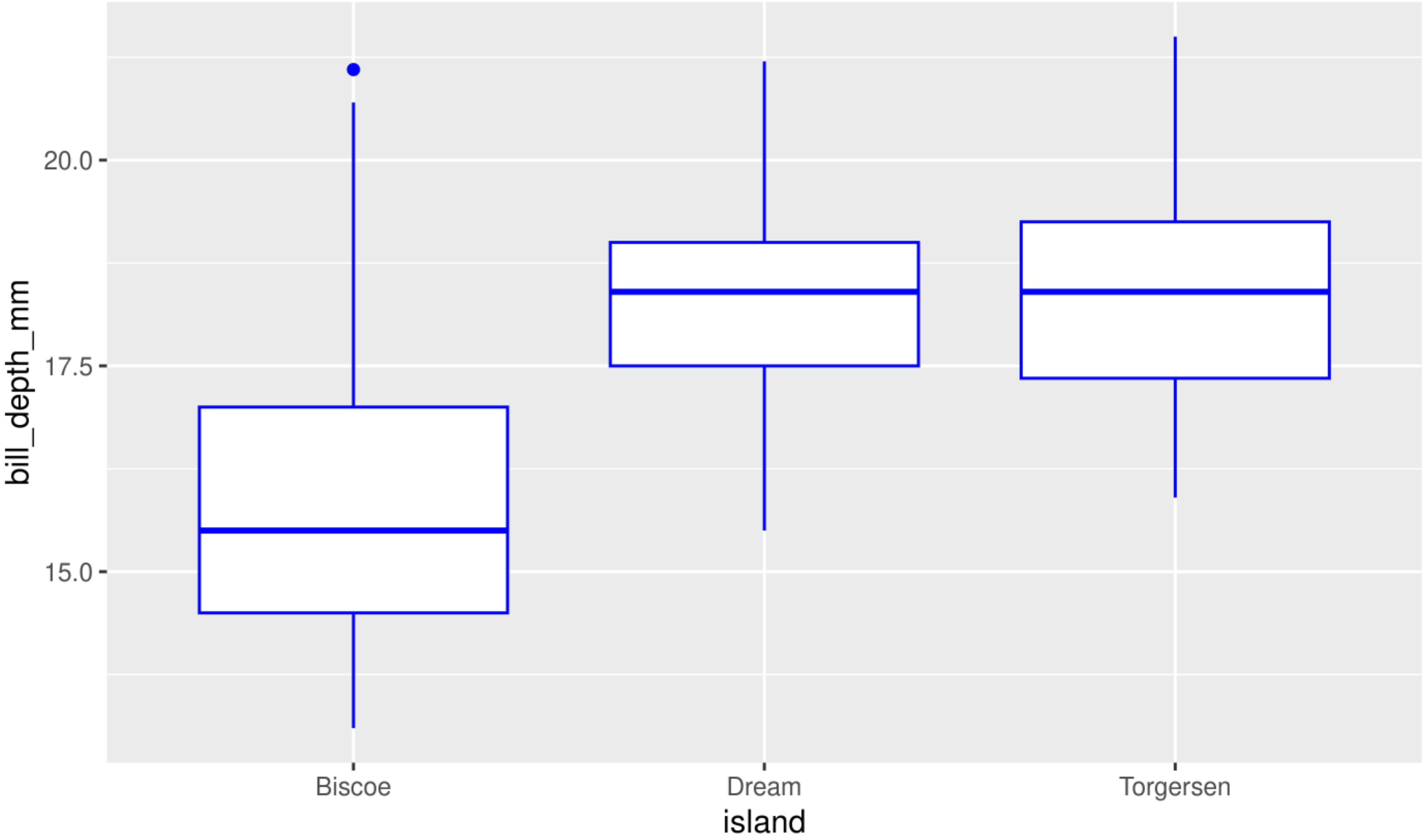
Plot points on top

Why not consider jittering them?



Your Turn: Create this plot

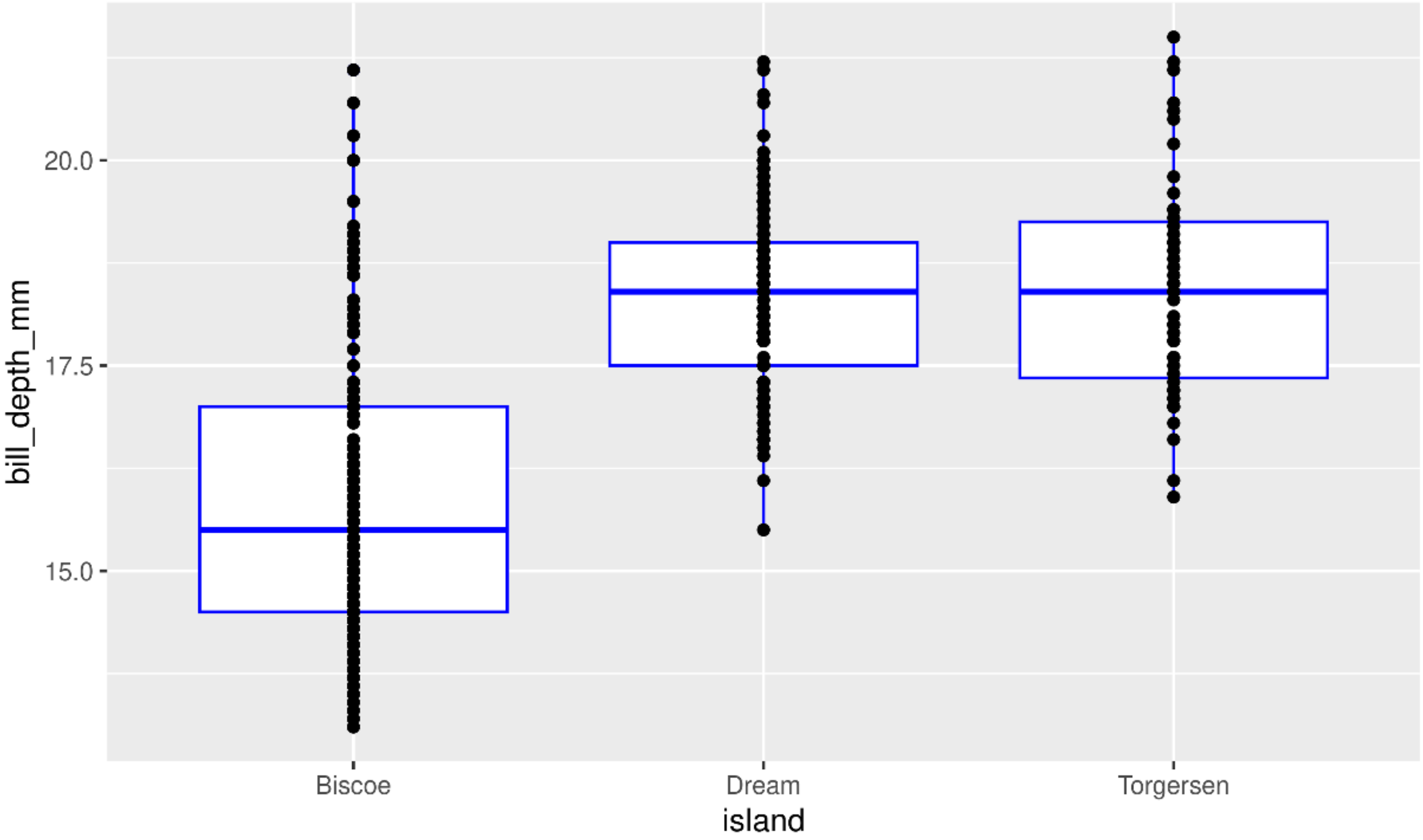
```r
ggplot(data = penguins, aes(x = island, y = bill_depth_mm)) +
  geom_boxplot(colour = "blue")
```



Your Turn: Create this plot

Too Easy?

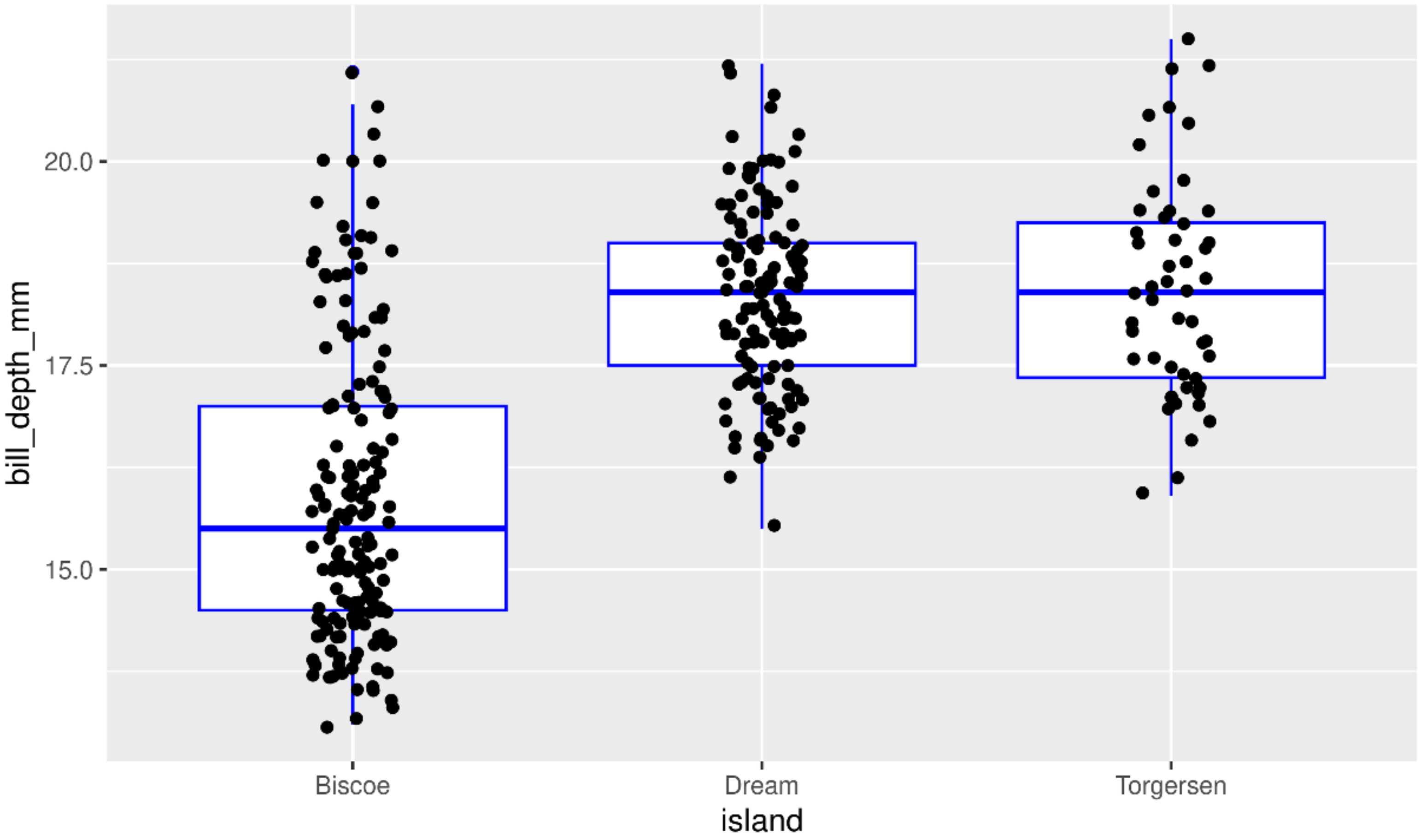
```r
ggplot(data = penguins, aes(x = island, y = bill_depth_mm)) +
  geom_boxplot(colour = "blue") +
  geom_point()
```



Your Turn: Create this plot

Too Easy?

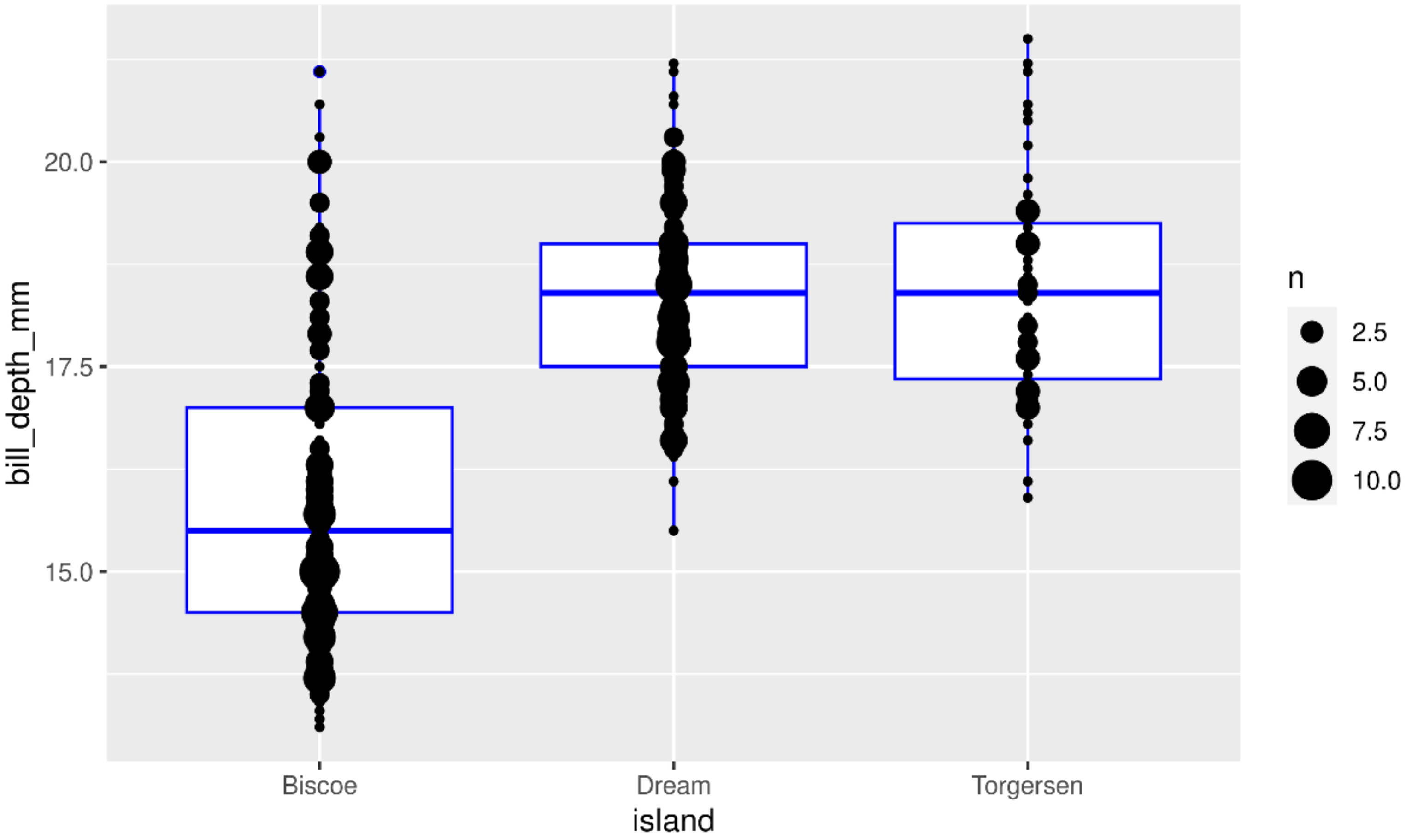
```r
ggplot(data = penguins, aes(x = island, y = bill_depth_mm)) +
  geom_boxplot(colour = "blue") +
  geom_point(position = position_jitter(width = 0.1))
```



Your Turn: Create this plot

Too Easy?

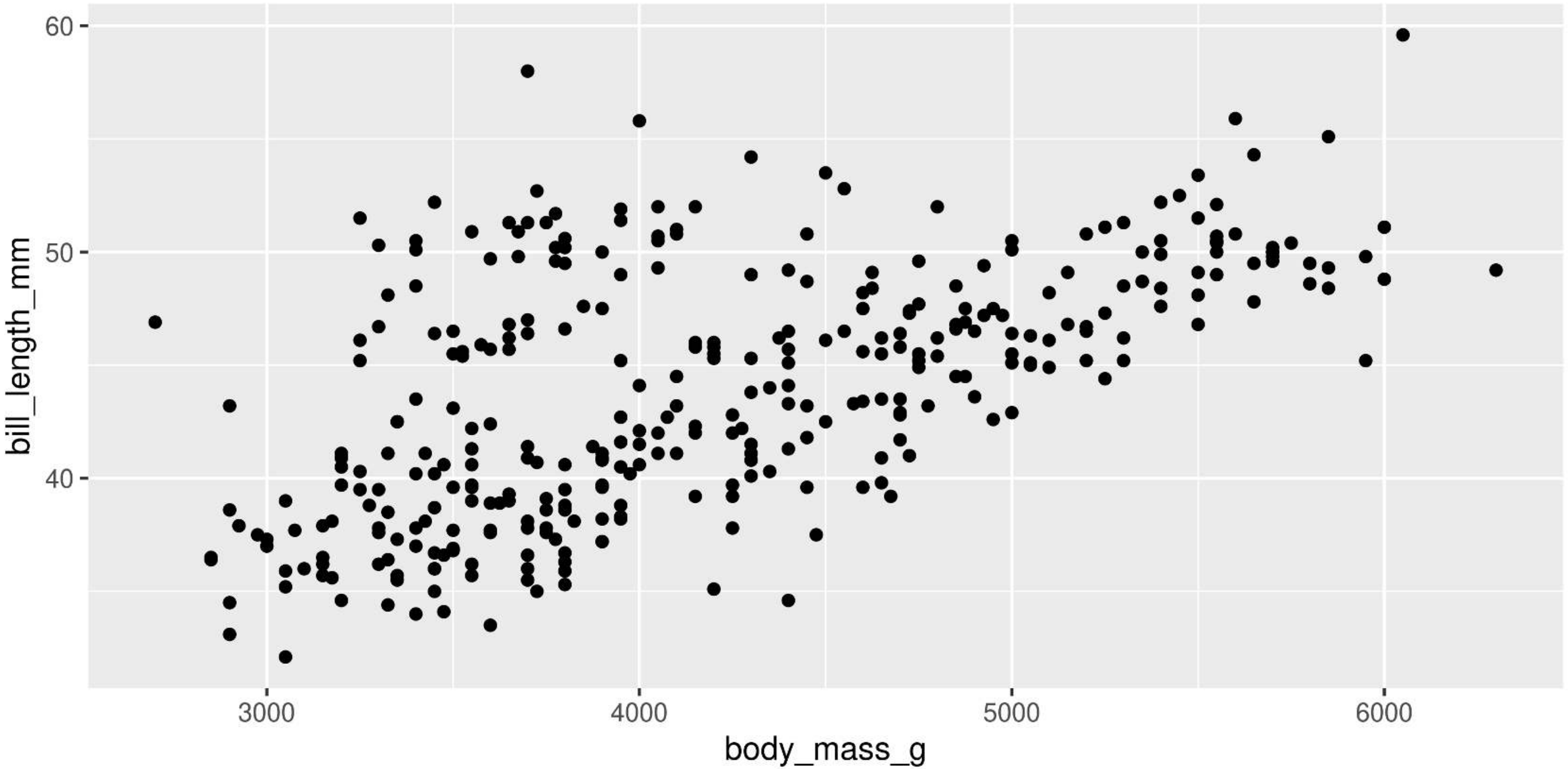
```r
ggplot(data = penguins, aes(x = island, y = bill_depth_mm)) +
  geom_boxplot(colour = "blue") +
  geom_count()
```



Showing data by group

Mapping aesthetics

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



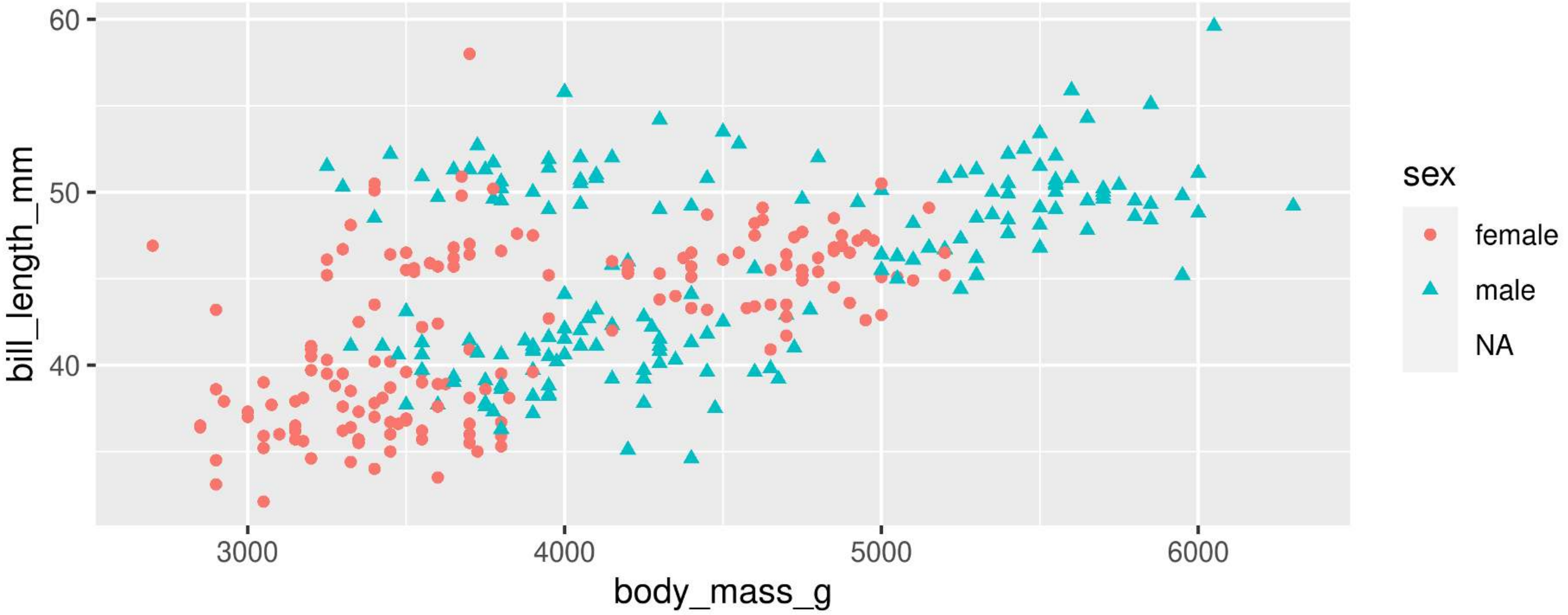
Mapping aesthetics

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

Mapping aesthetics

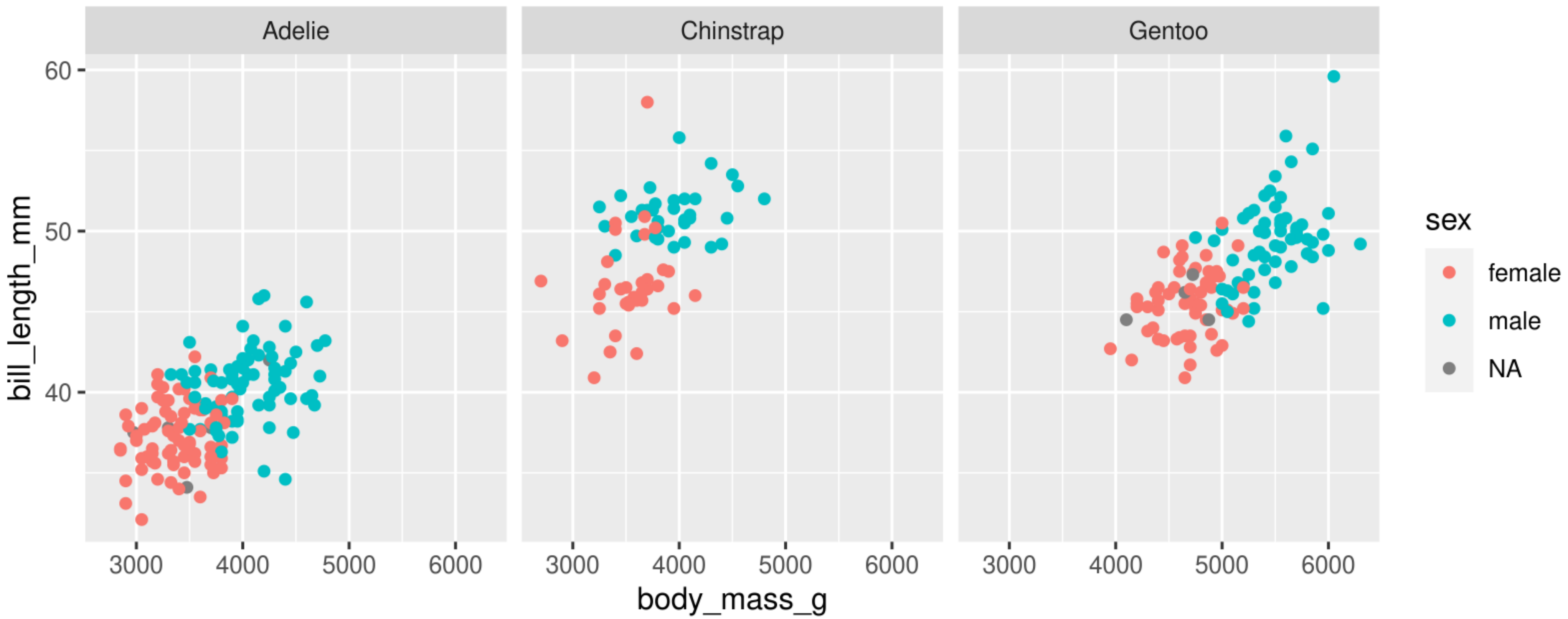
`ggplot` automatically populates the legends (combining where it can)

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



Faceting: `facet_wrap()`

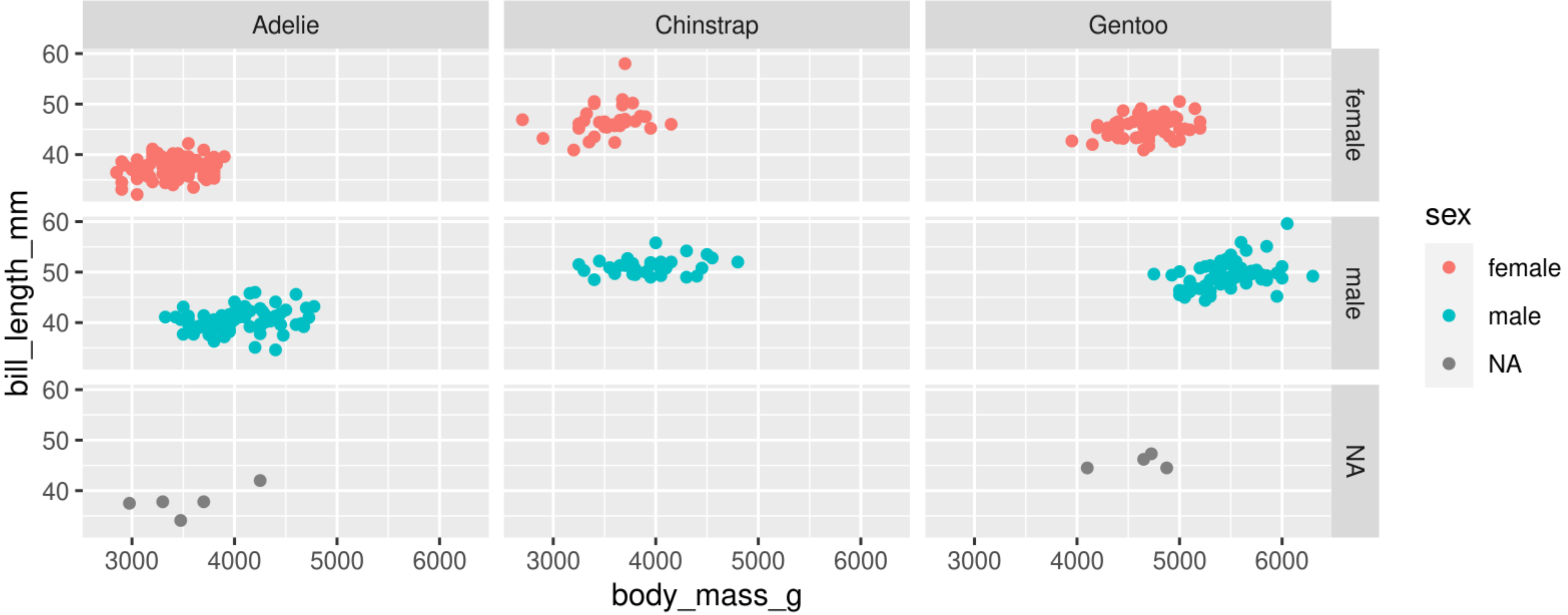
```r
ggplot(data = penguins, aes(x = body_mass_g, y = bill_length_mm, colour = sex)) +
  geom_point() +
  facet_wrap(~ species)
```



Split plots by **one** grouping variable

Faceting: `facet_grid()`

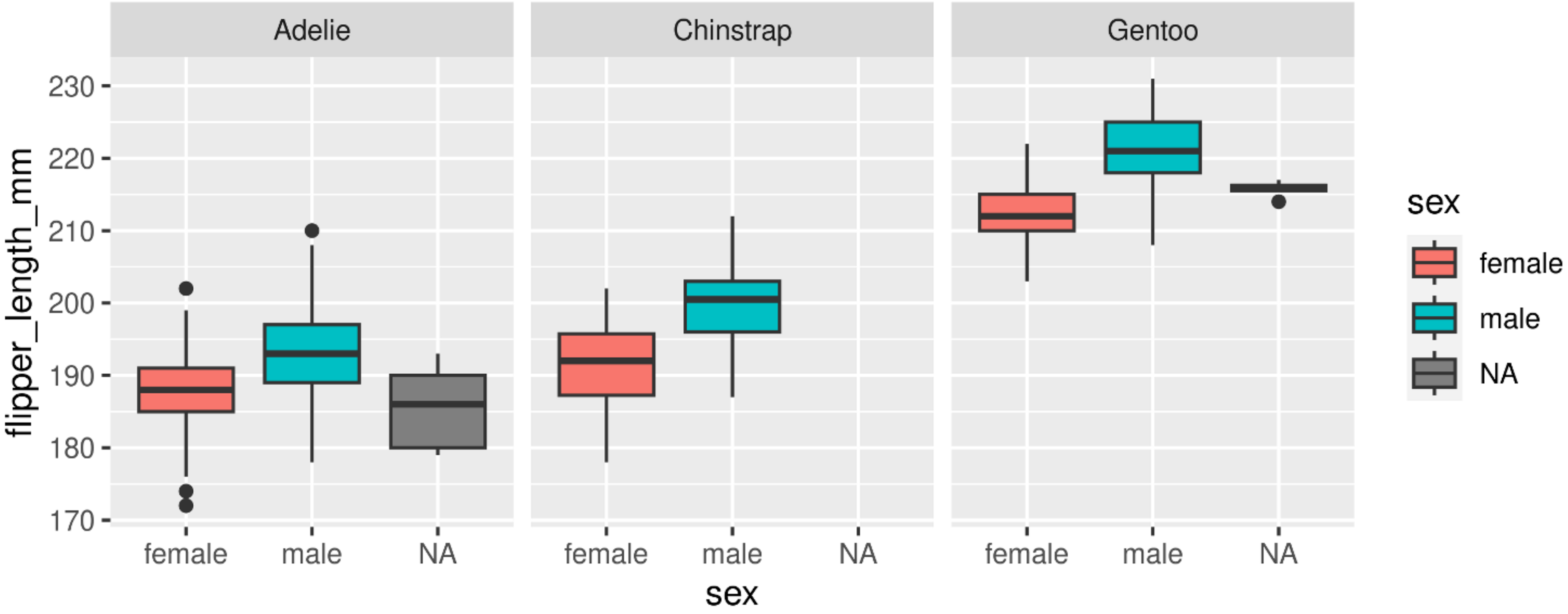
```r
ggplot(data = penguins, aes(x = body_mass_g, y = bill_length_mm, colour = sex)) +
  geom_point() +
  facet_grid(sex ~ species)
```



Split plots by **two** grouping variables

Your Turn: Create this plot

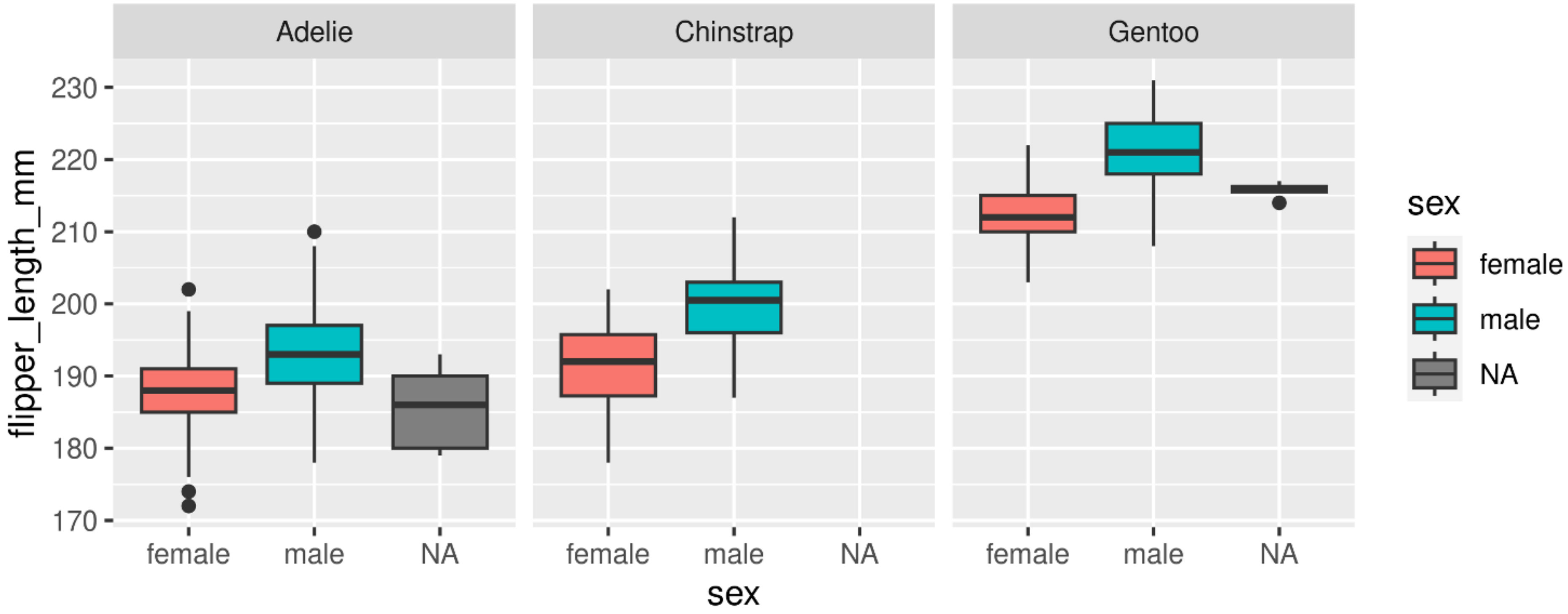
```
ggplot(data = ____, aes(______________________________________)) +
  ______________ +
  ______________
```



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

Your Turn: Create this plot

```r
ggplot(data = penguins, aes(x = sex, y = flipper_length_mm, fill = sex)) +
  geom_boxplot() +
  facet_wrap(~ species)
```



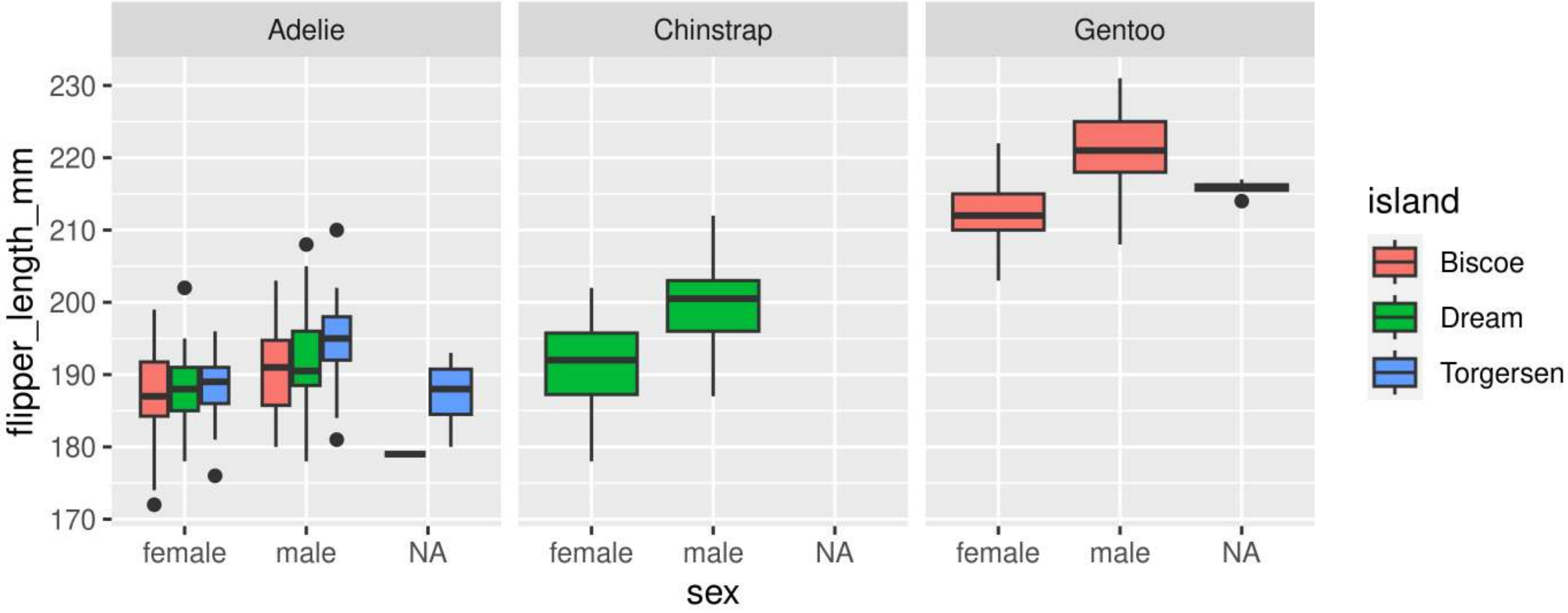
Hint: `colour` is for outlining with a colour, `fill` is for ‘filling’ with a colour

Too Easy? Split boxplots by sex **and** island

Your Turn: Create this plot

Too Easy?

```
ggplot(data = penguins, aes(x = sex, y = flipper_length_mm, fill = island)) +
  geom_boxplot() +
  facet_wrap(~ species)
```



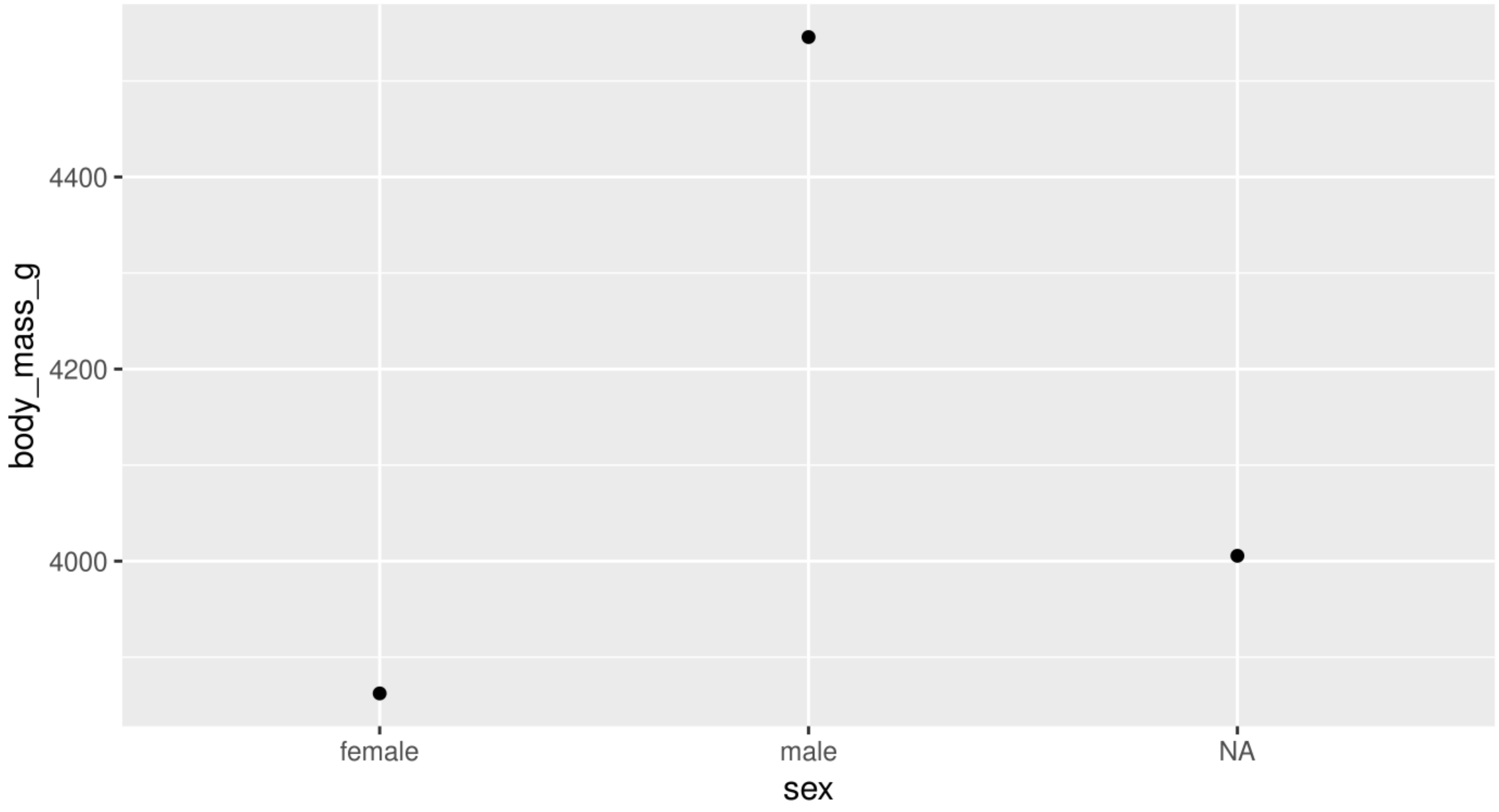
Small change (`fill = sex` to `fill = island`) results in completely different plot

Adding Statistics to Plots

Summarizing data

Add data means as points

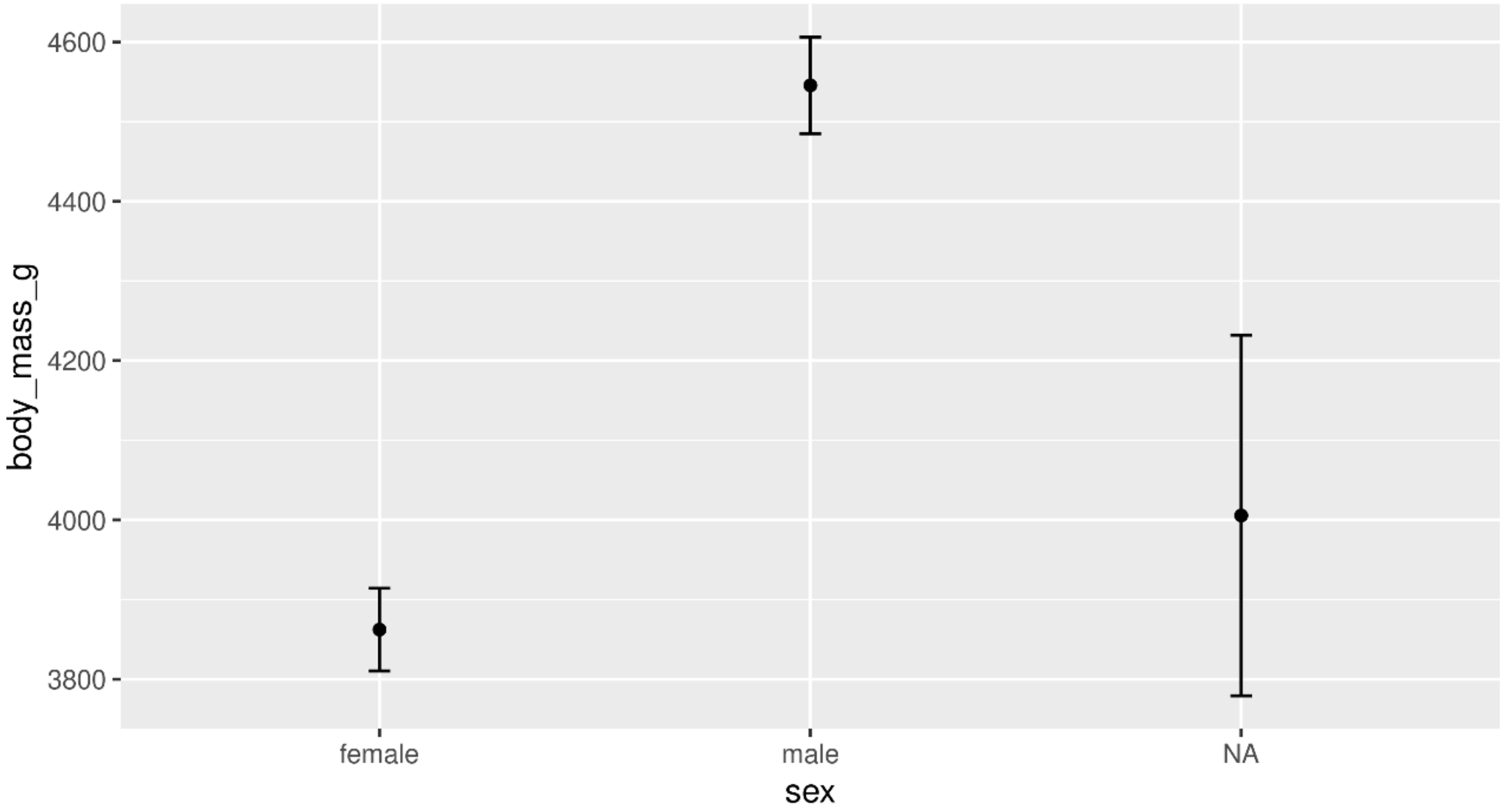
```r
ggplot(data = penguins, aes(x = sex, y = body_mass_g)) +
  stat_summary(geom = "point", fun = mean)
```



Summarizing data

Add error bars, calculated from the data

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

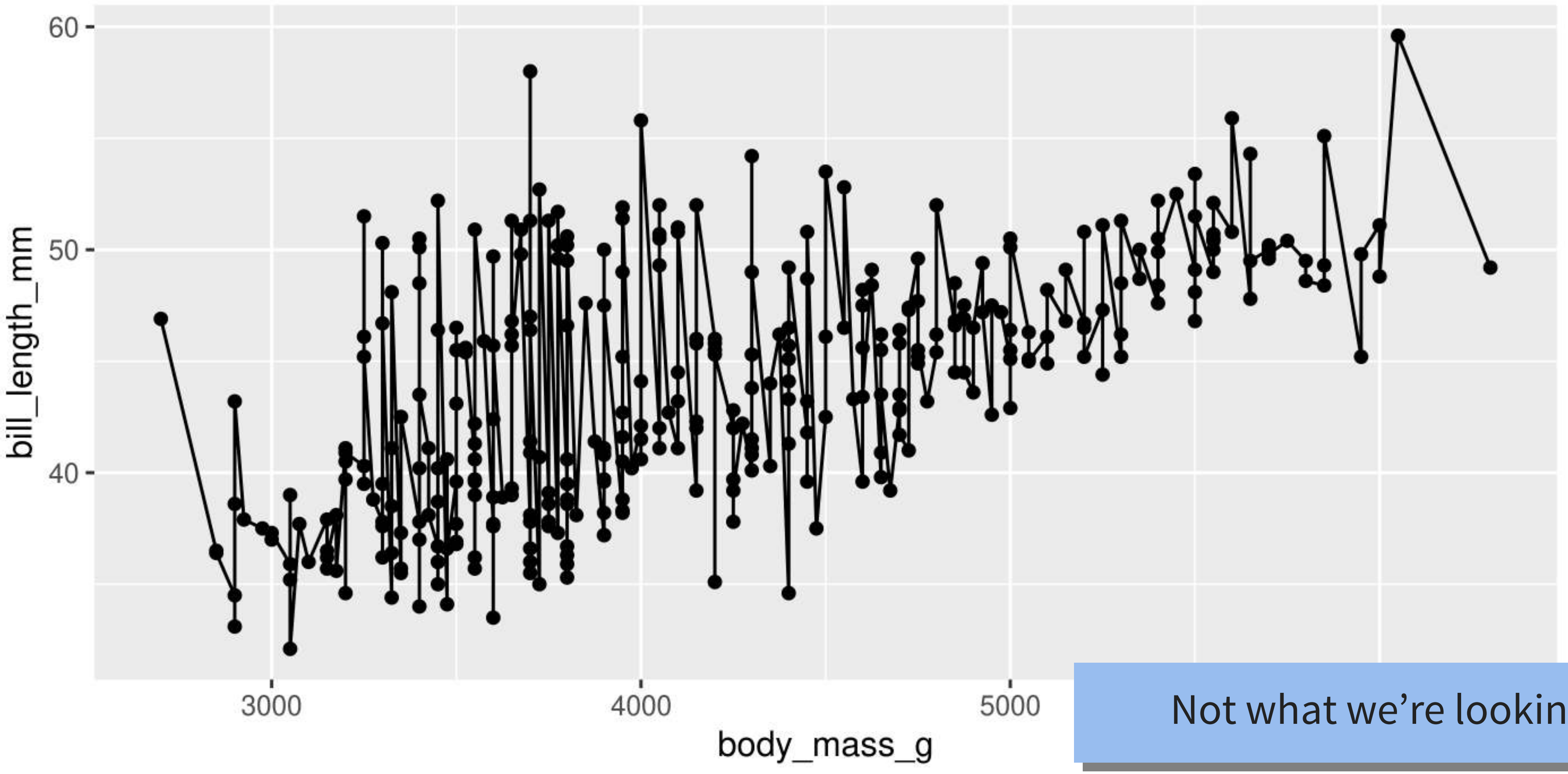


Trendlines / Regression Lines

Trendlines / Regression lines

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

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



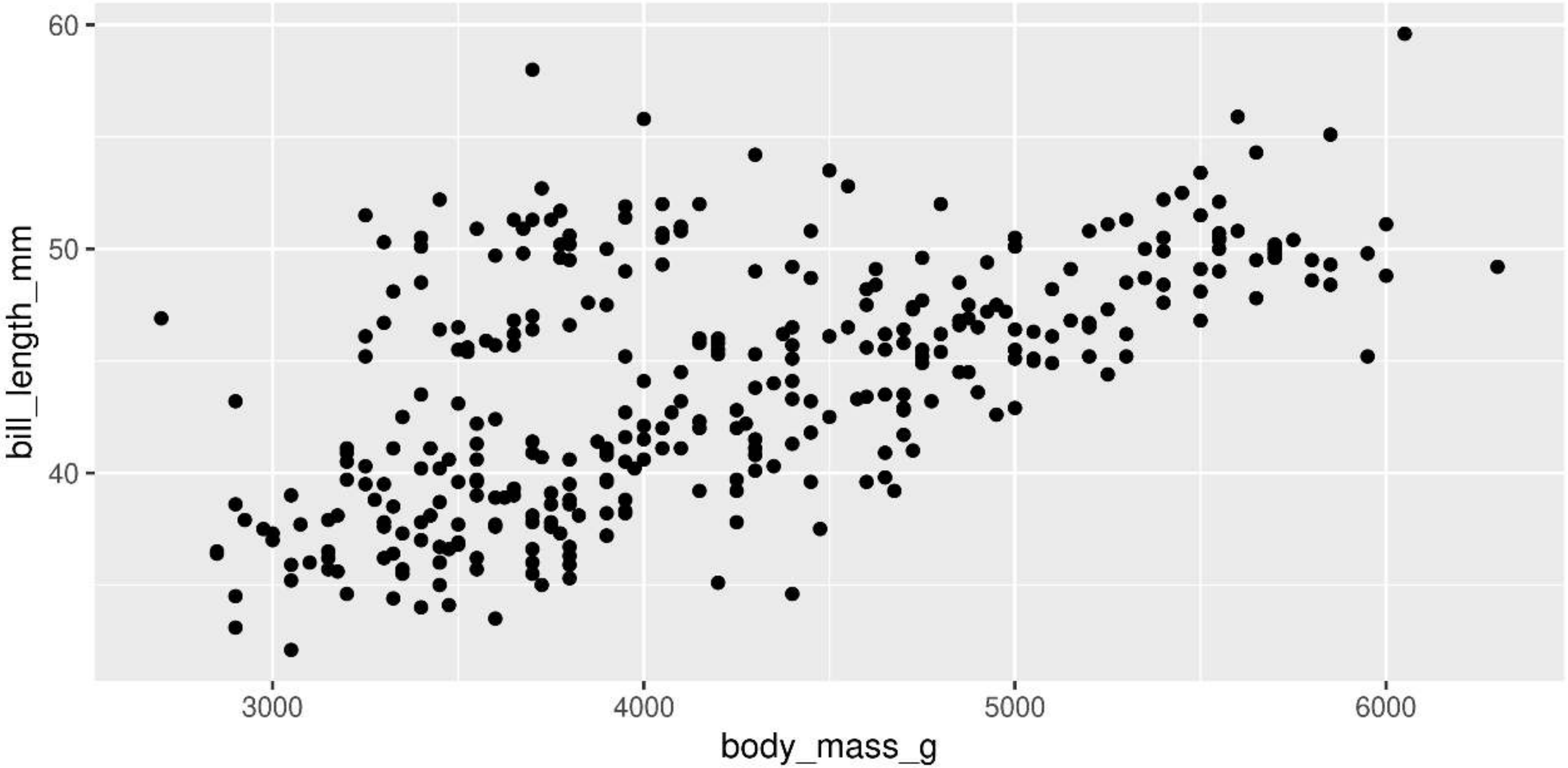
Not what we’re looking for

Trendlines / Regression lines

Let’s add a trend line properly

Start with basic plot:

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

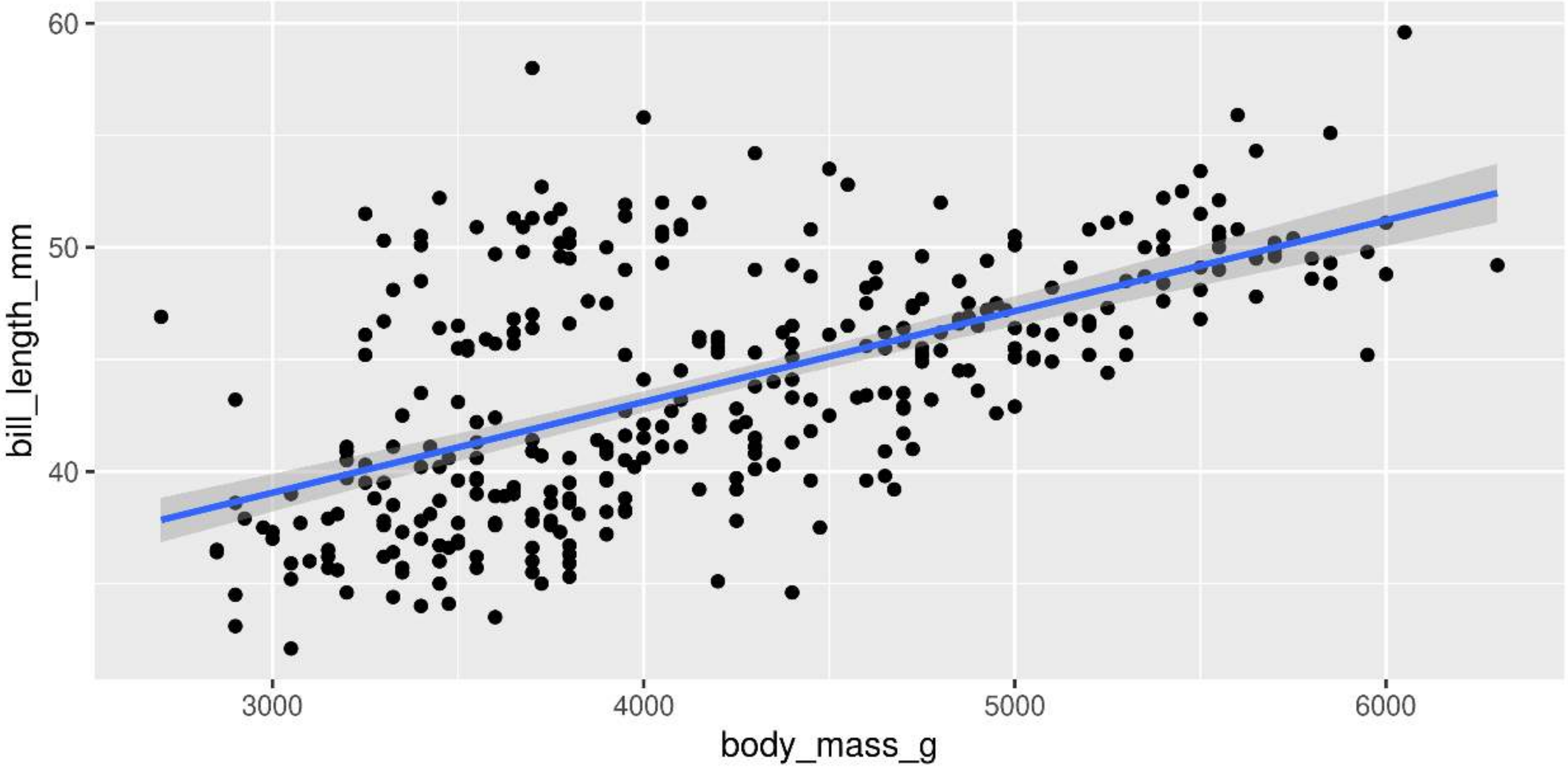


Trendlines / Regression lines

Add the `stat_smooth()`

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

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

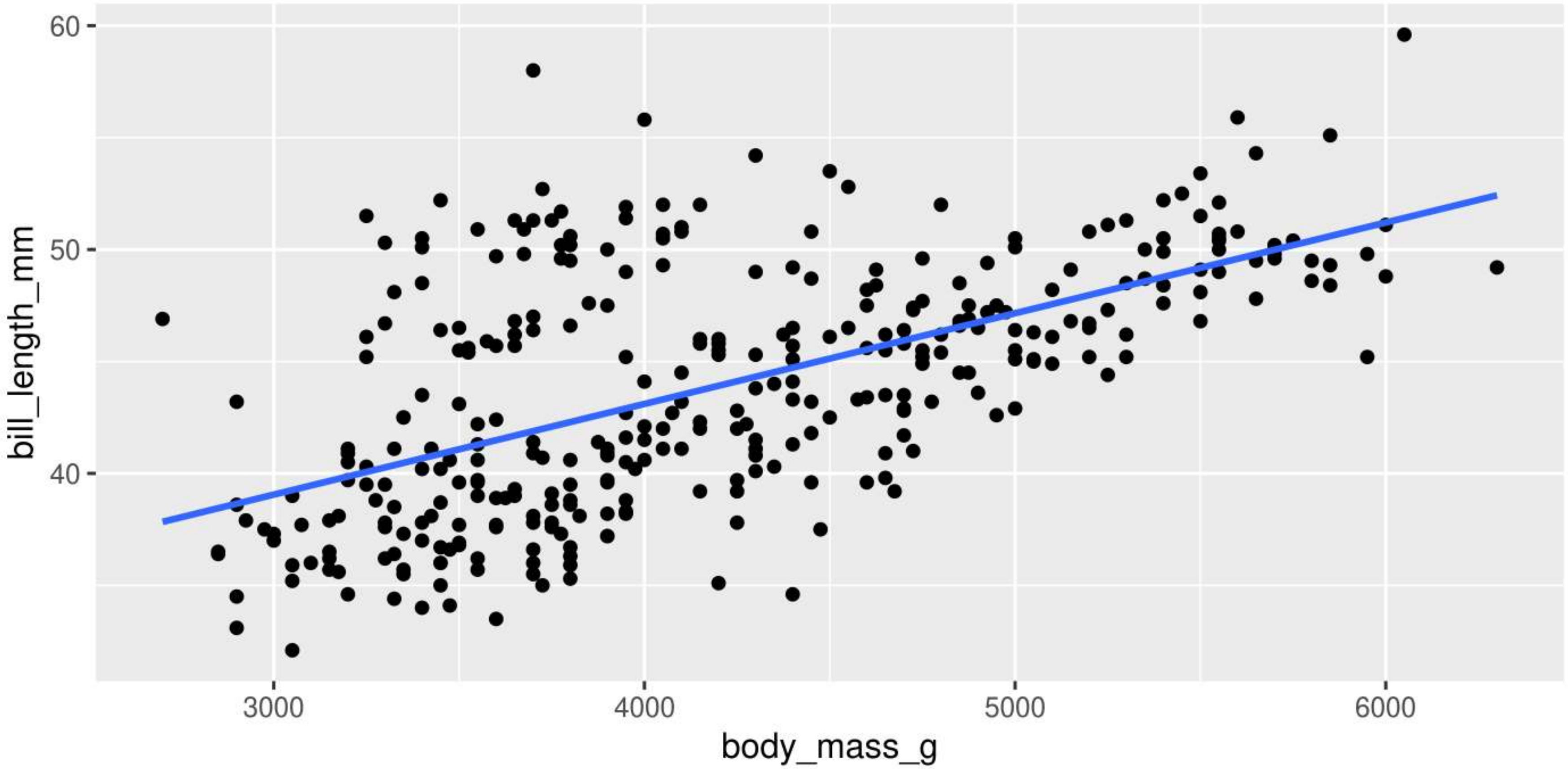


Trendlines / Regression lines

Add the `stat_smooth()`

- remove the grey ribbon `se = FALSE`

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

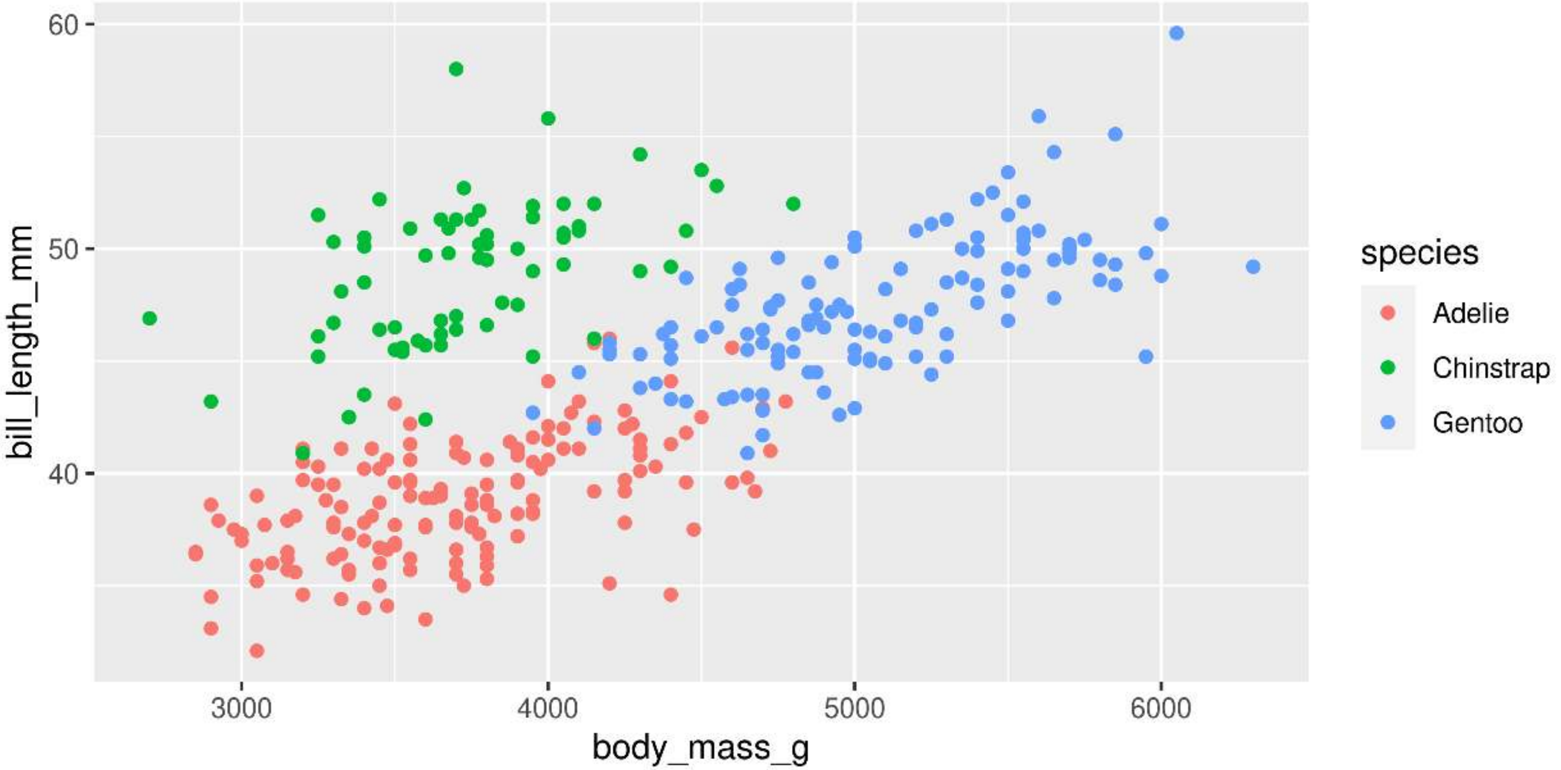


Trendlines / Regression lines

A line for each group

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

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

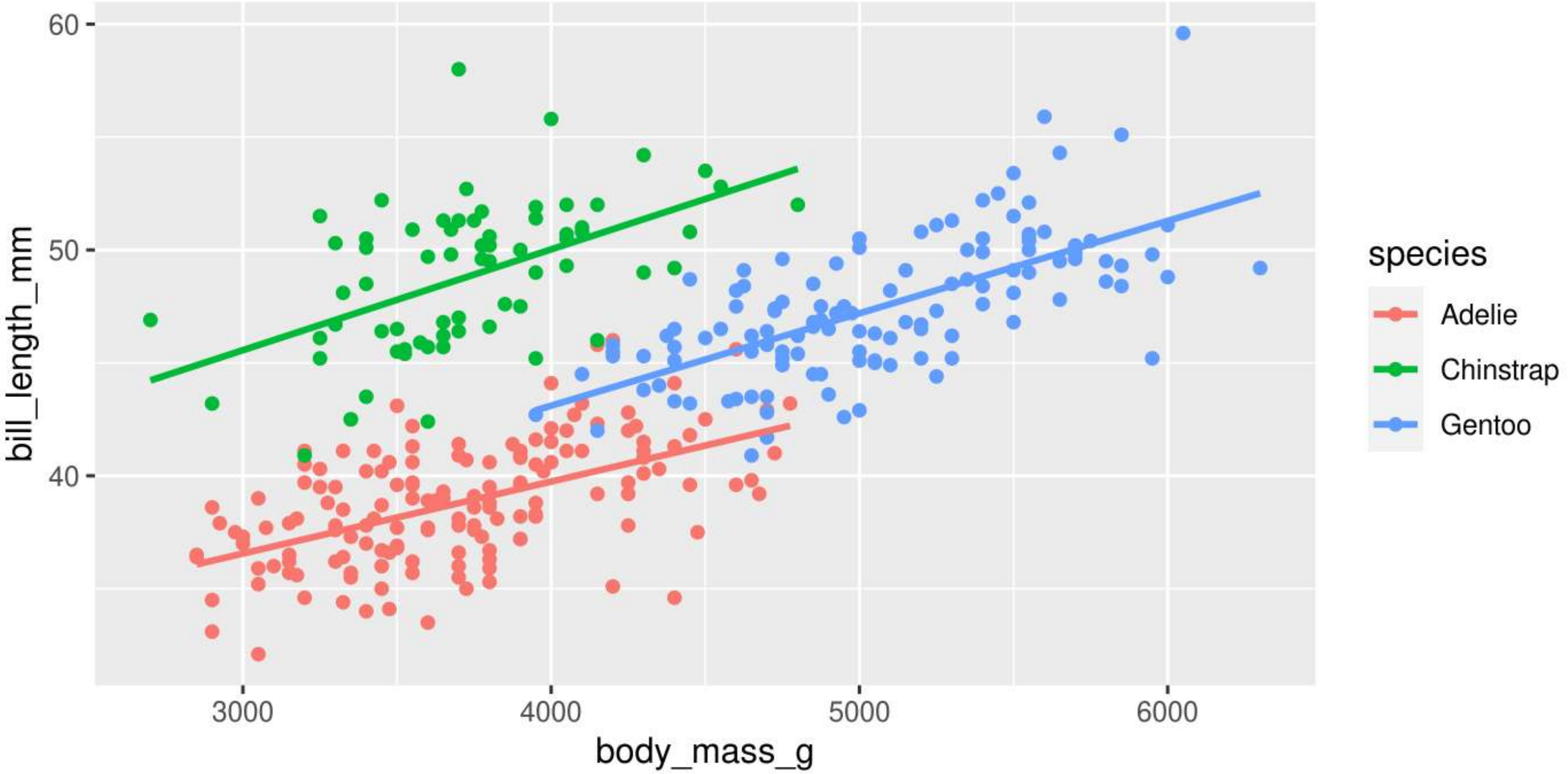


Trendlines / Regression lines

A line for each group

- `stat_smooth()` automatically uses the same grouping

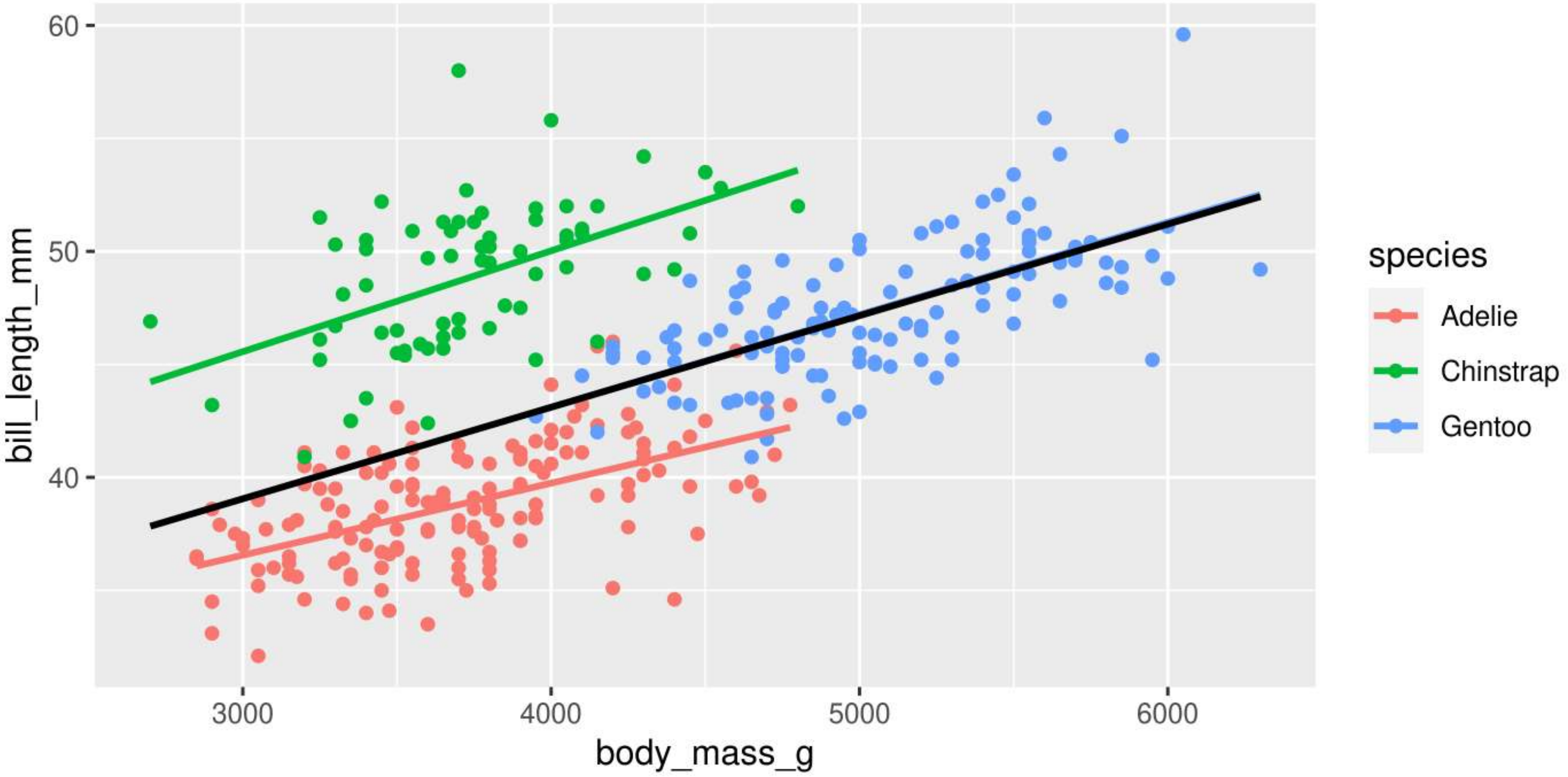
```r
g + stat_smooth(method = "lm", se = FALSE)
```



Trendlines / Regression lines

A line for each group AND overall

```
g +
  stat_smooth(method = "lm", se = FALSE) +
  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? Add regression lines for each species as well

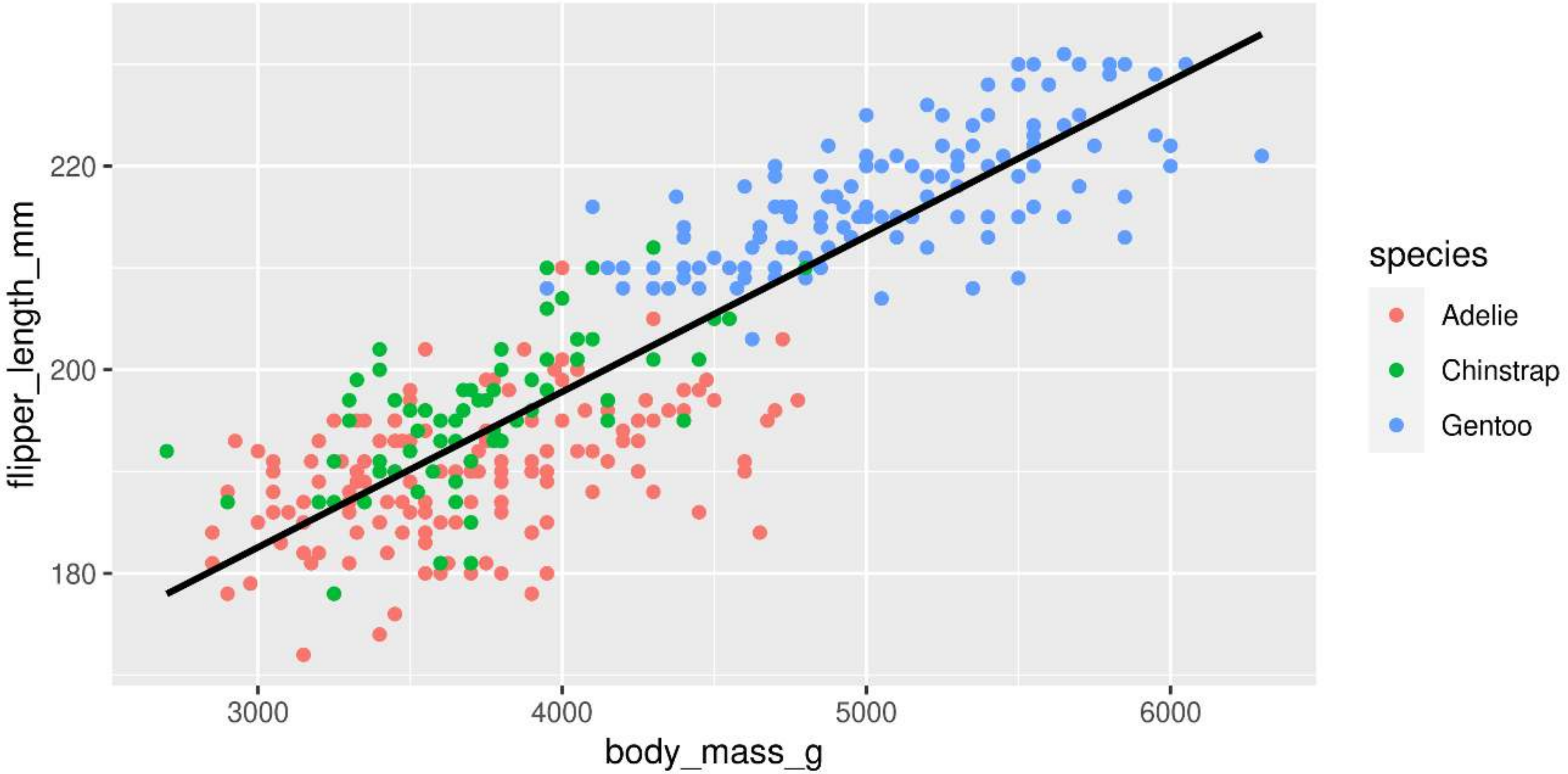
Can you make the species lines larger?

Can you indicate which points are female and which are male?

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*

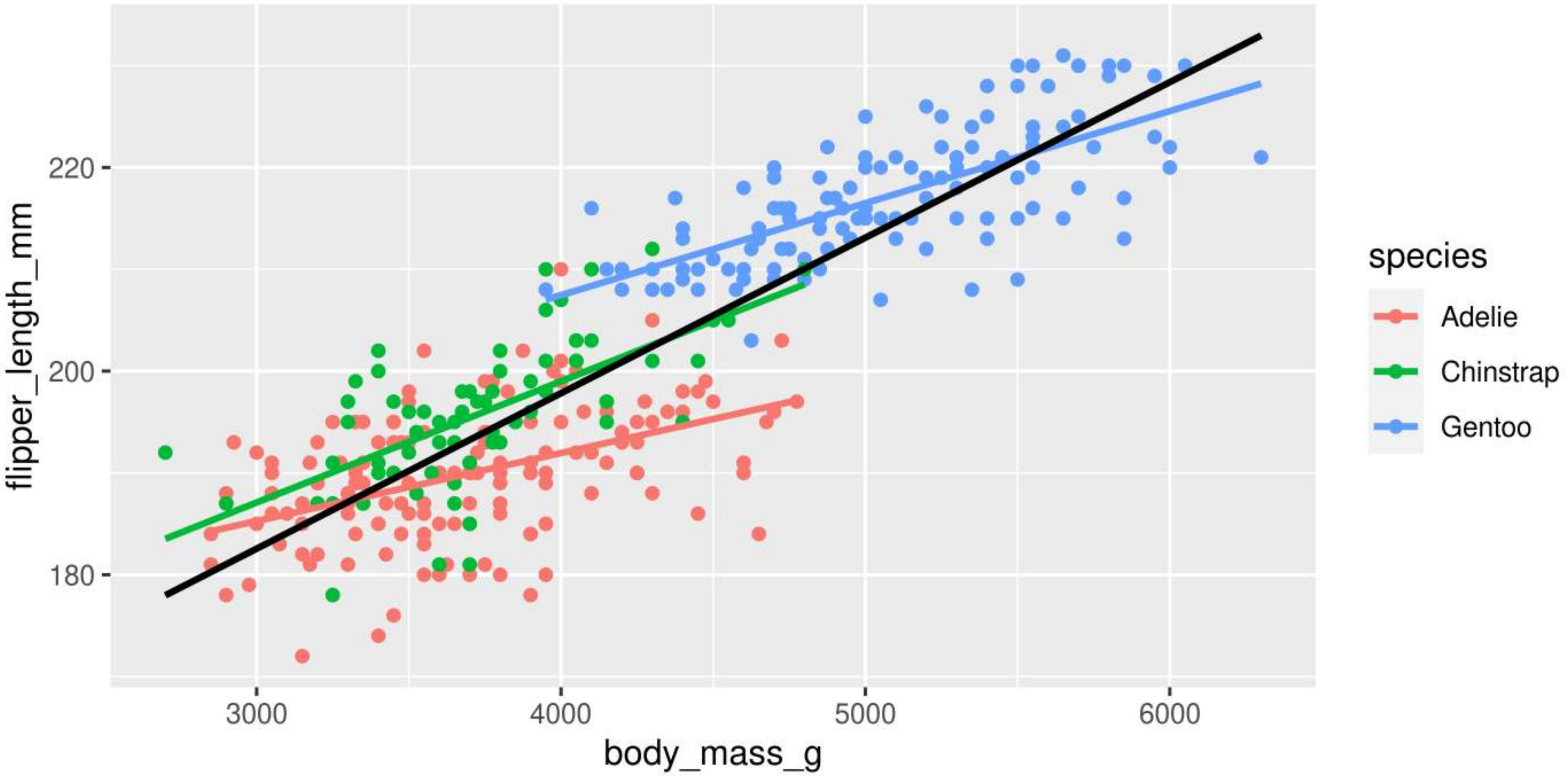
```r
ggplot(data = penguins, aes(x = body_mass_g, y = flipper_length_mm, colour = species)) +
  geom_point() +
  stat_smooth(method = "lm", se = FALSE, colour = "black")
```



Your Turn: Create this plot

Too Easy?

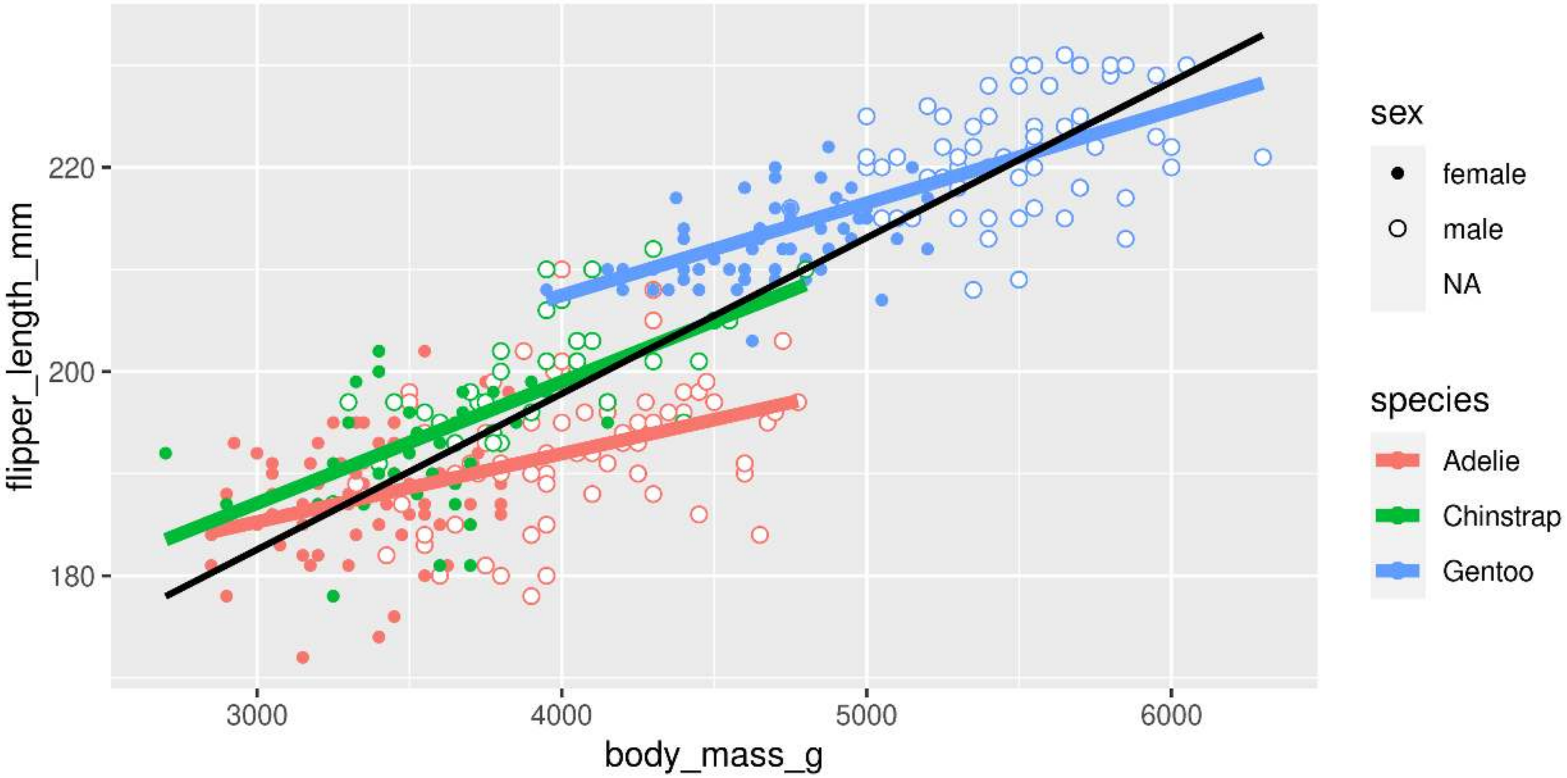
```r
ggplot(data = penguins, aes(x = body_mass_g, y = flipper_length_mm, colour = species)) +
  geom_point() +
  stat_smooth(method = "lm", se = FALSE) +
  stat_smooth(method = "lm", se = FALSE, colour = "black")
```



Your Turn: Create this plot

Too Easy?

```r
ggplot(data = penguins, aes(x = body_mass_g, y = flipper_length_mm,
                            colour = species)) +
  geom_point(aes(shape = sex), size = 2, fill = "white") +
  stat_smooth(method = "lm", se = FALSE, linewidth = 2) +
  stat_smooth(method = "lm", se = FALSE, colour = "black") +
  scale_shape_manual(values = c(20, 21))
```

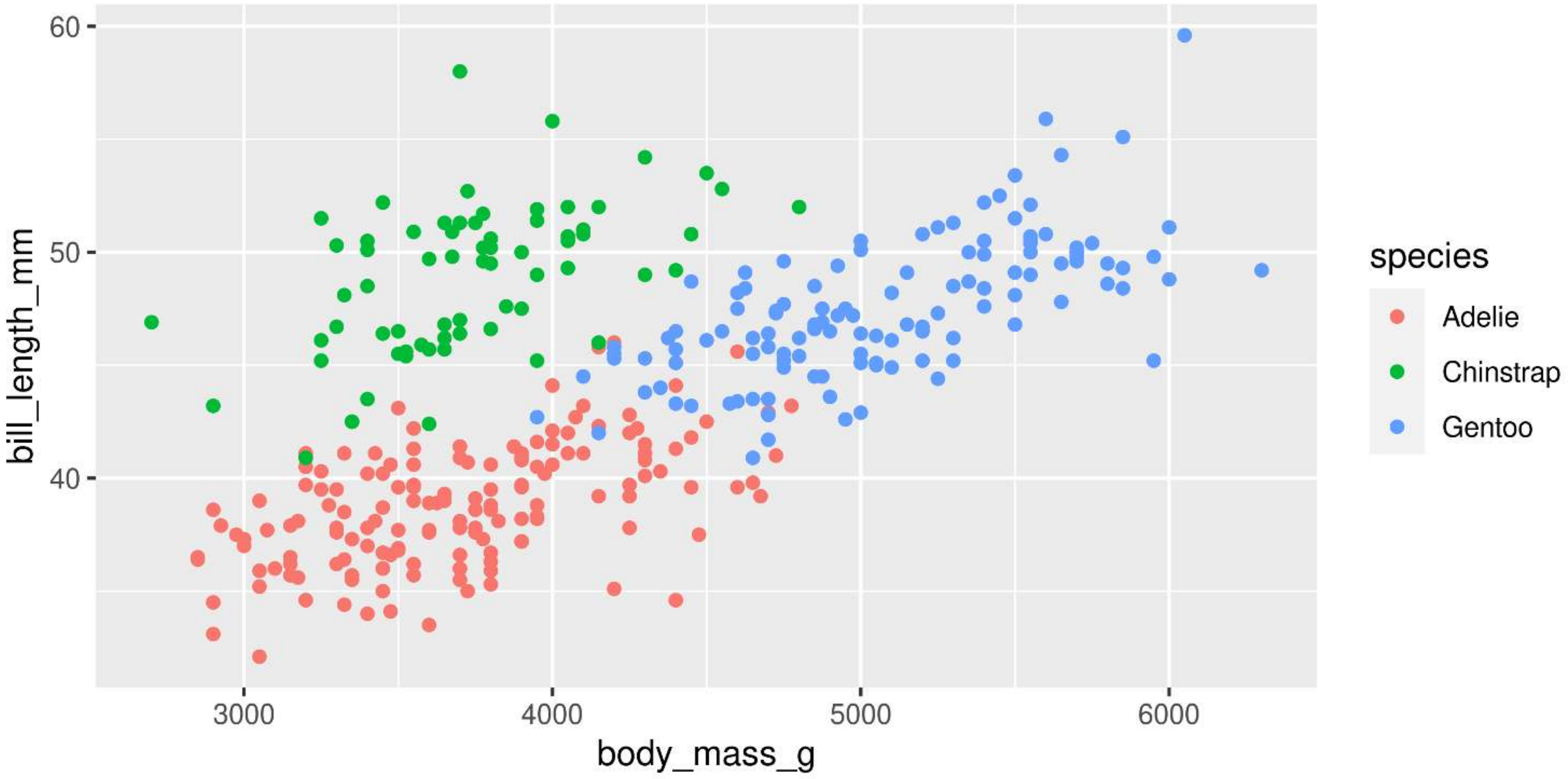


Customizing plots

Customizing: Starting plot

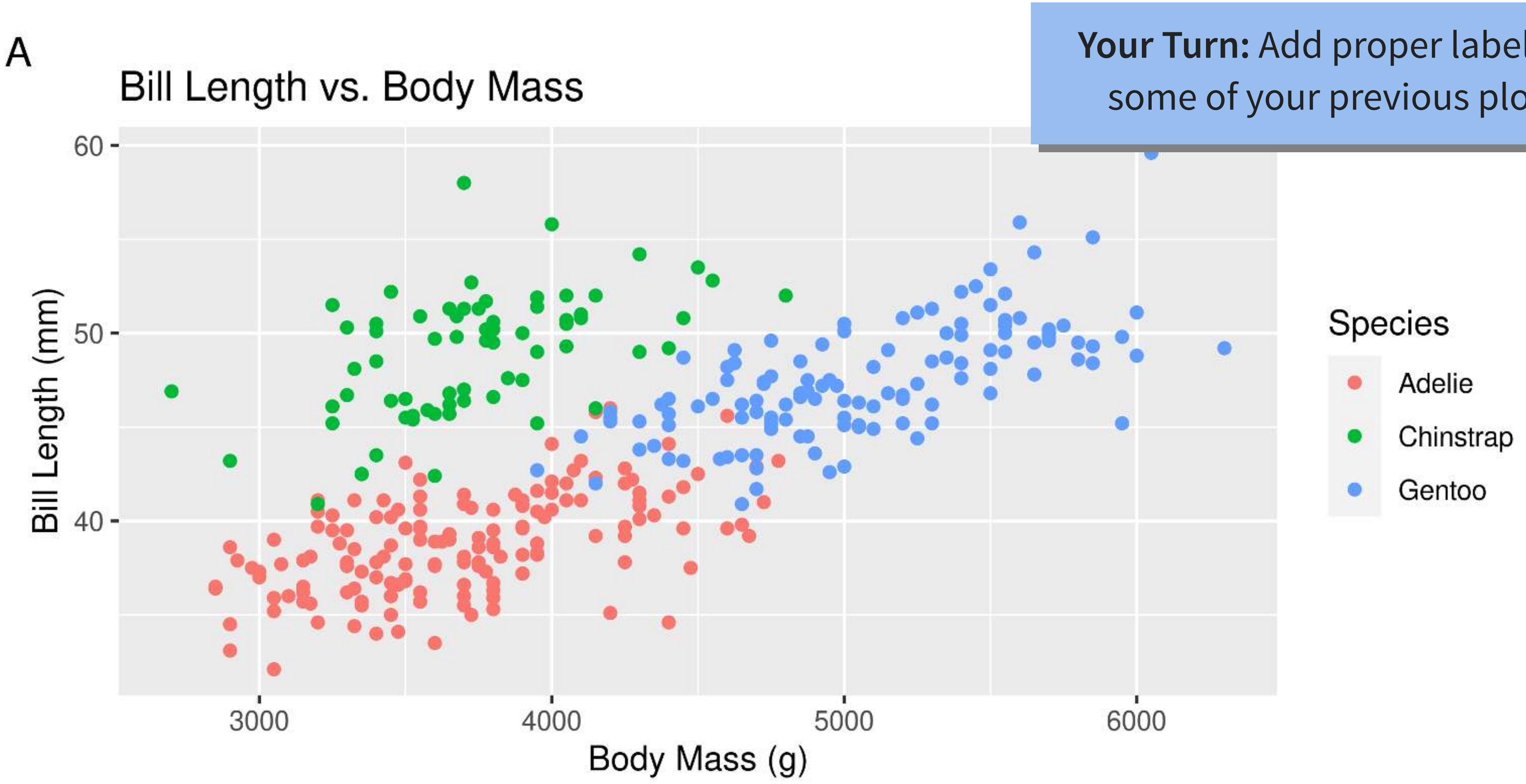
Let’s work with this plot

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



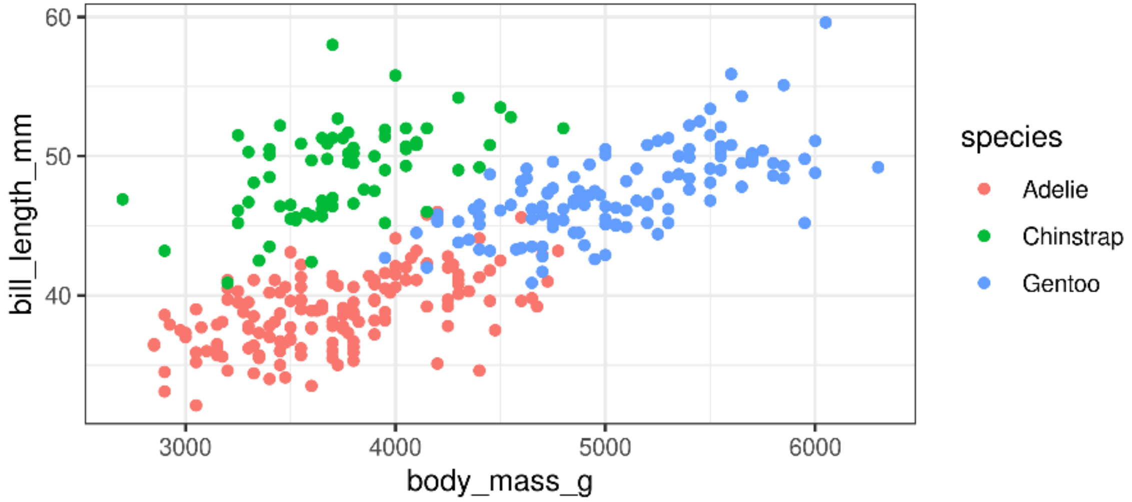
Customizing: Labels

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

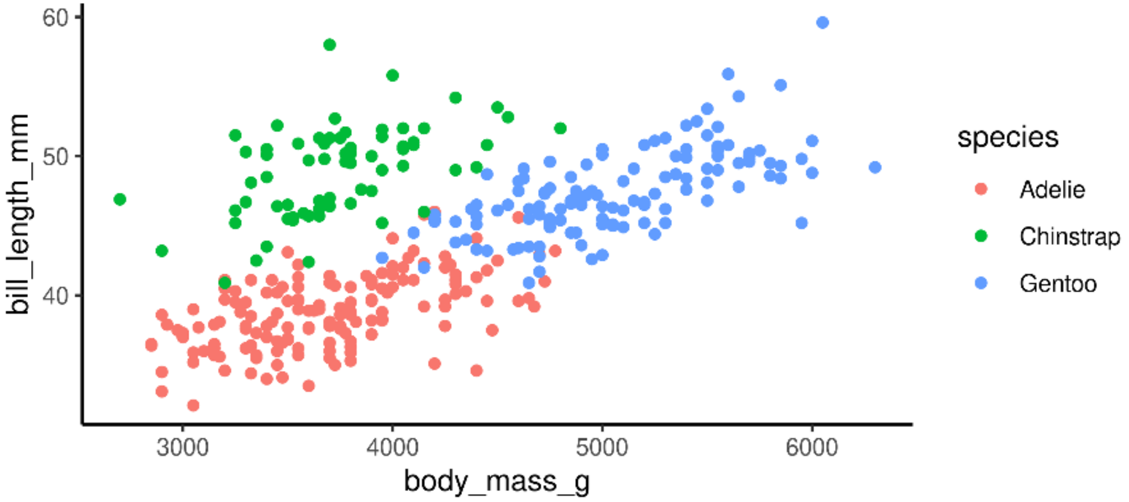


Your Turn: Add proper labels to some of your previous plots

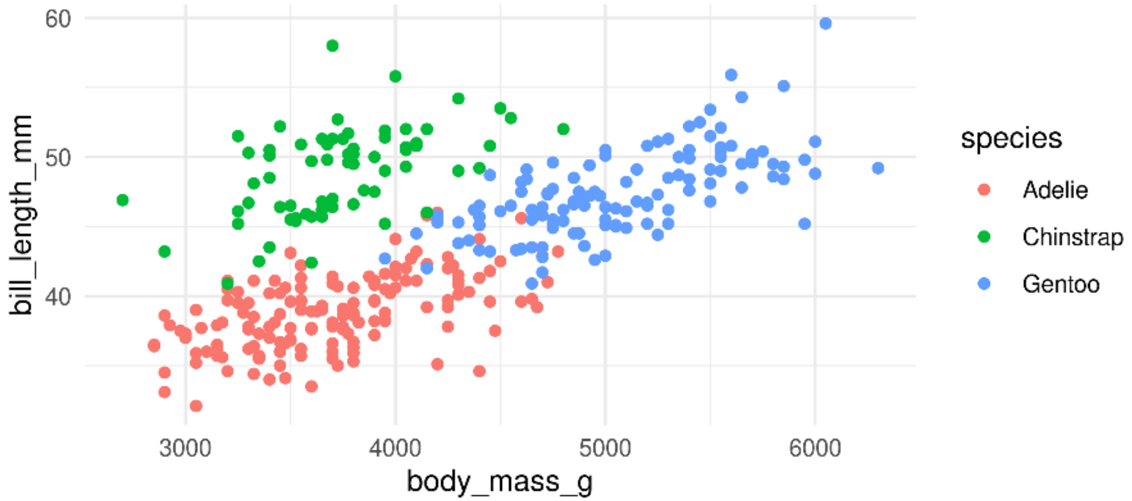
Customizing: Built-in themes



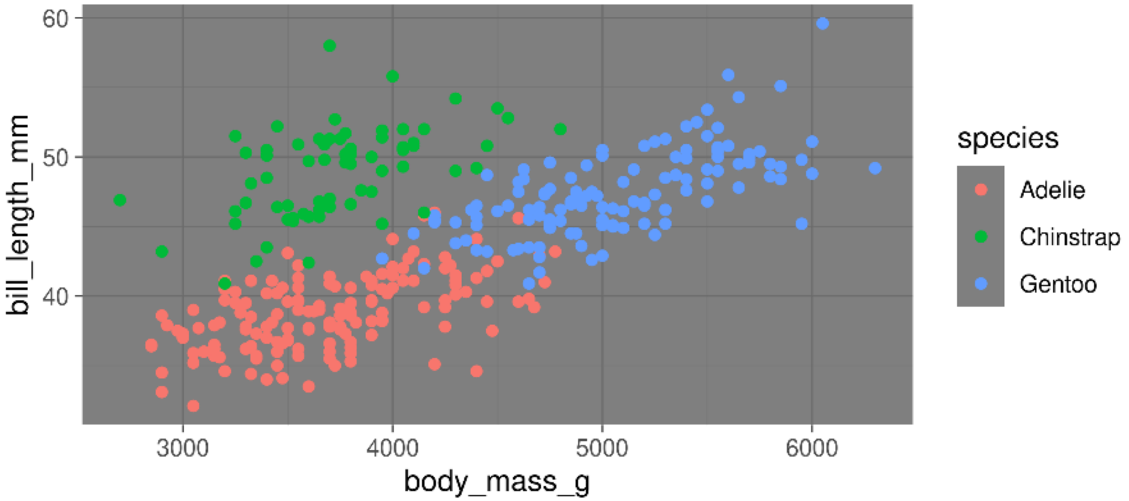
g + theme_bw()



g + theme_classic()



g + theme_minimal()



g + theme_dark()

Customizing: Axes

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

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

Common arguments

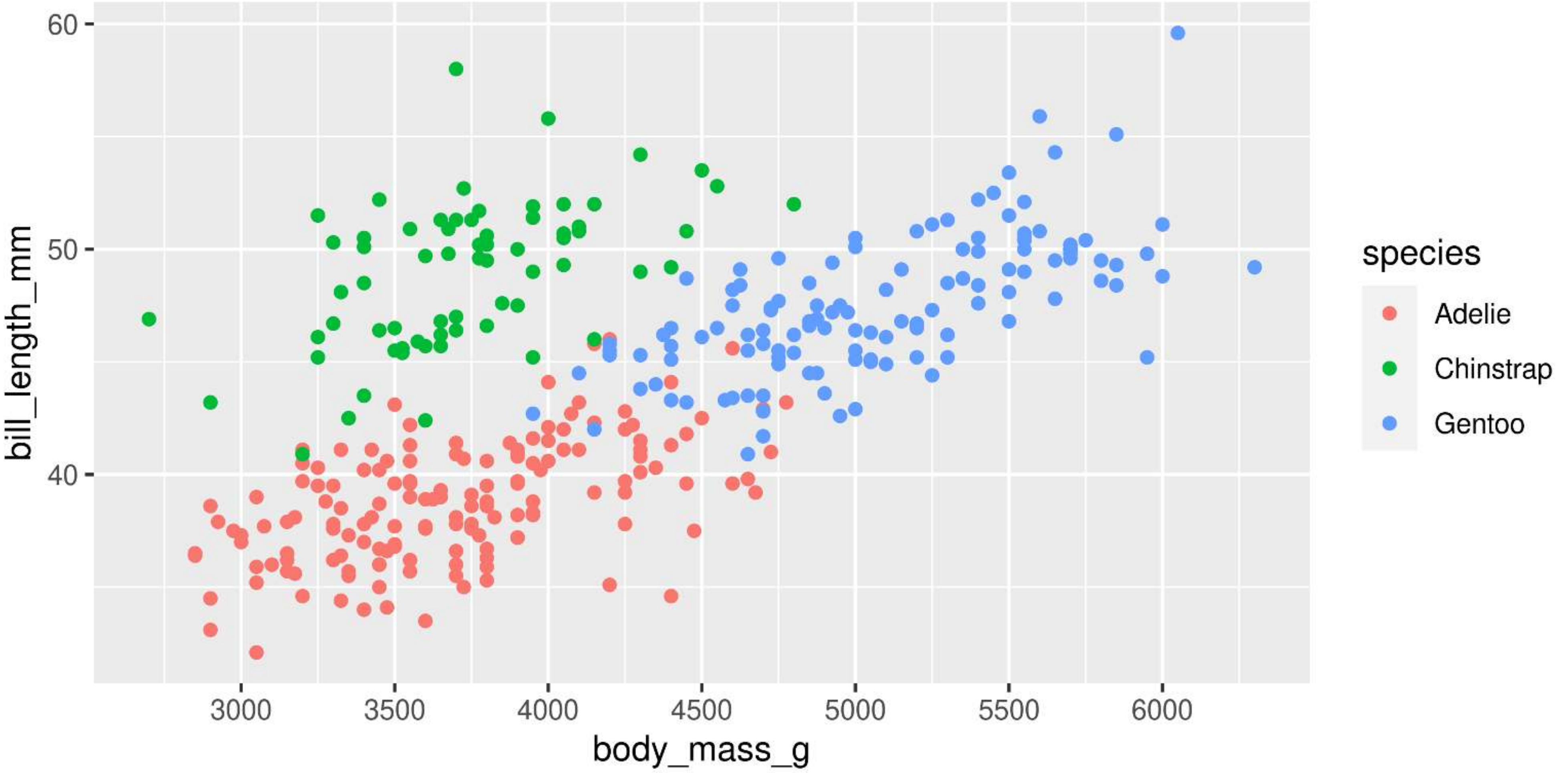
```r
g + scale_x_continuous(breaks = seq(0, 20, 10)) # Tick breaks
g + scale_x_continuous(limits = c(0, 15))       # xlim() is a shortcut for this
g + scale_x_continuous(expand = c(0, 0))        # Space between axis and data
```

Let’s take a look…

Customizing: Axes

Breaks

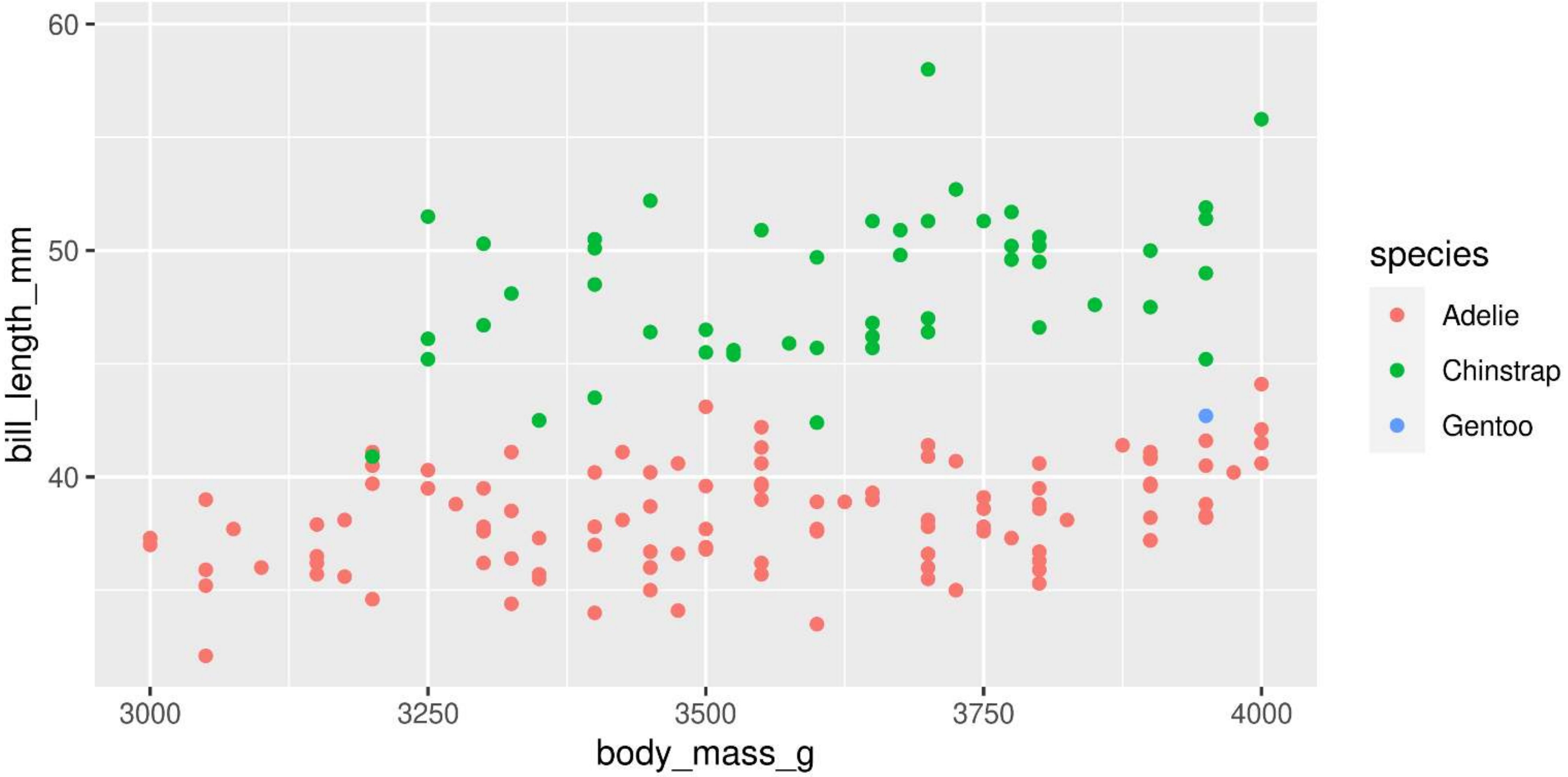
```r
g + scale_x_continuous(breaks = seq(2500, 6500, 500))
```



Customizing: Axes

Limits

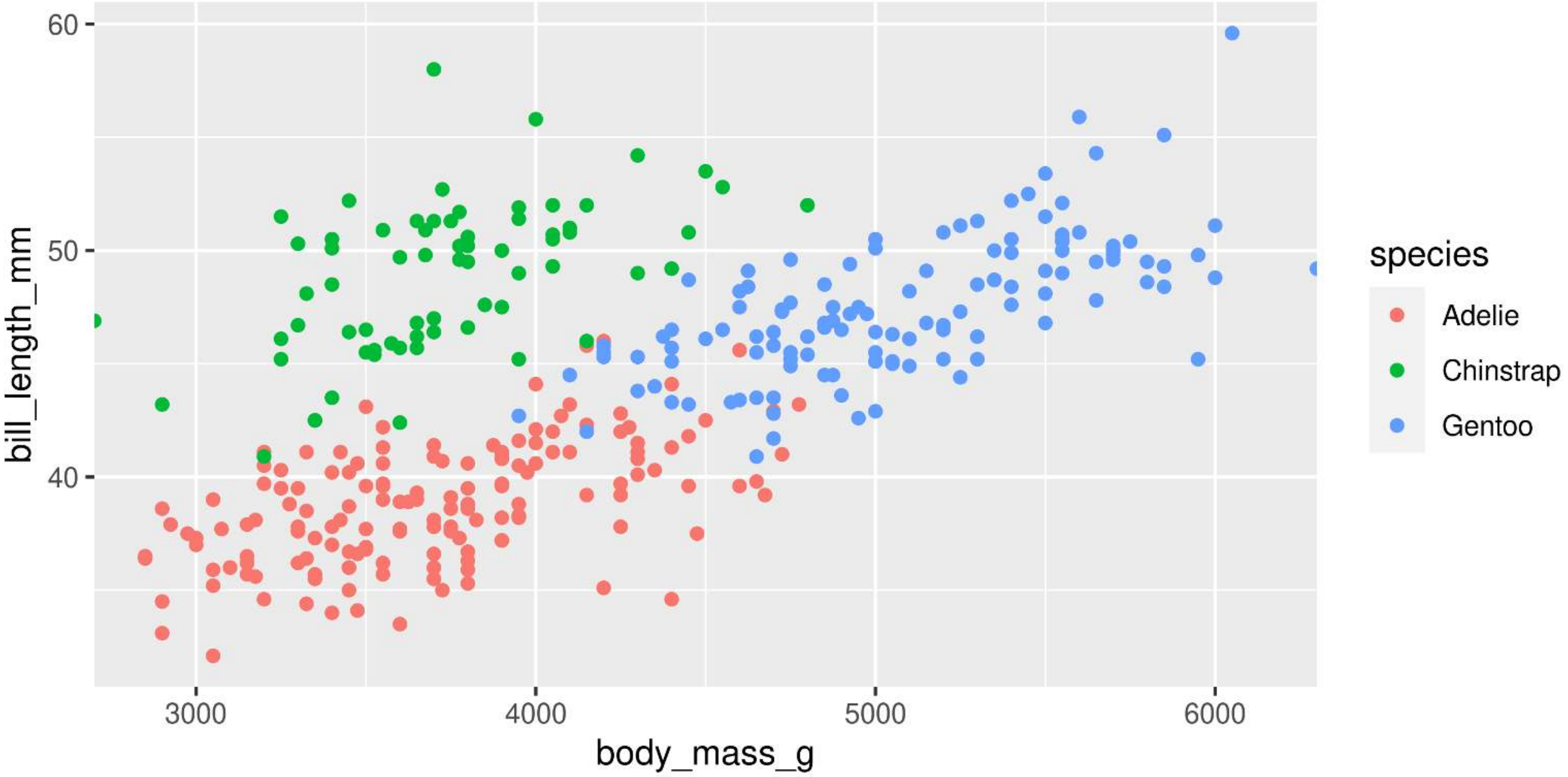
```r
g + scale_x_continuous(limits = c(3000, 4000))
```



Customizing: Axes

Space between origin and axis start

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

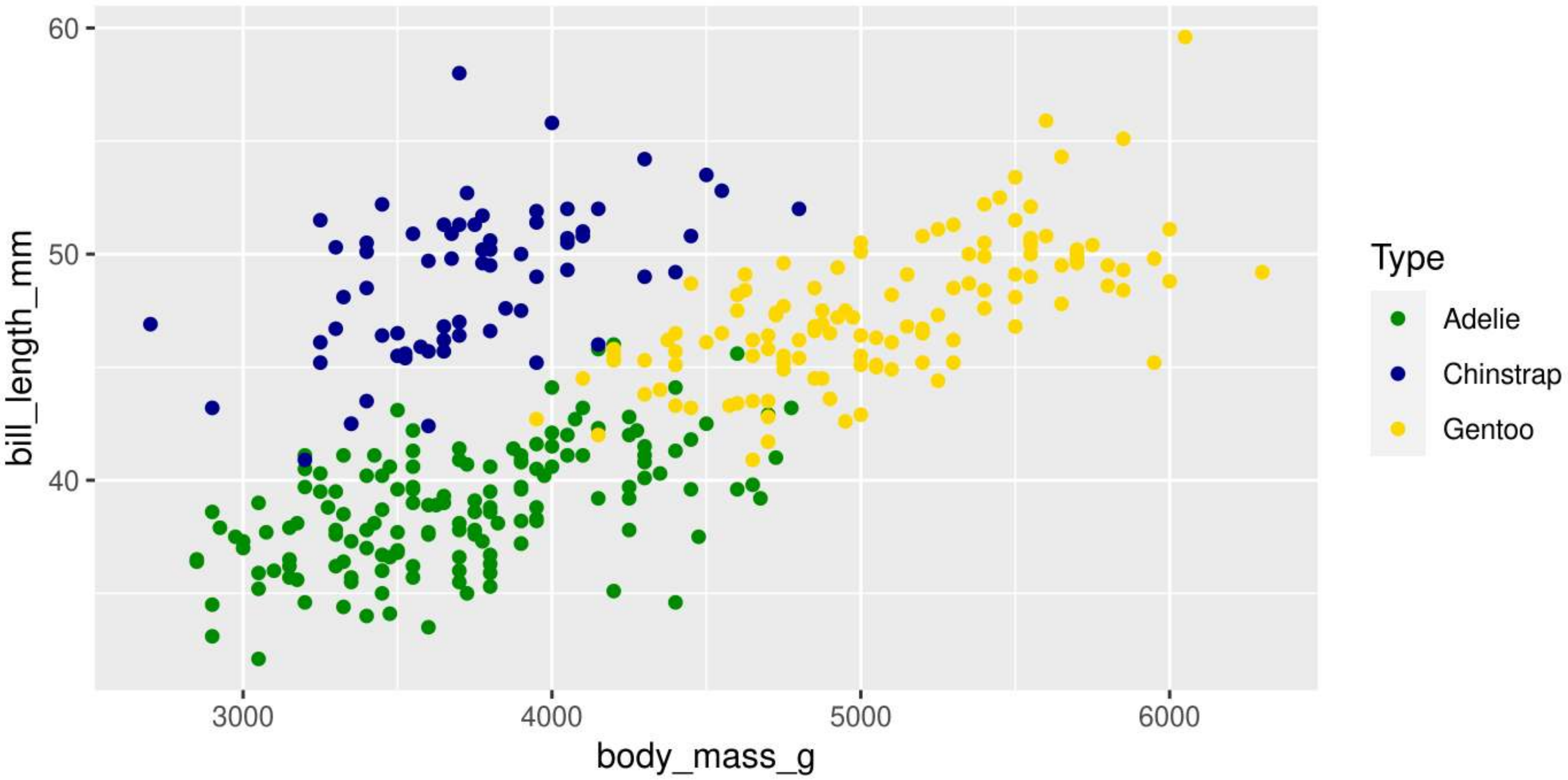


Customizing: Aesthetics

Using scales

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

```
g + scale_colour_manual(name = "Type", values = c("green4", "blue4", "gold"))
```

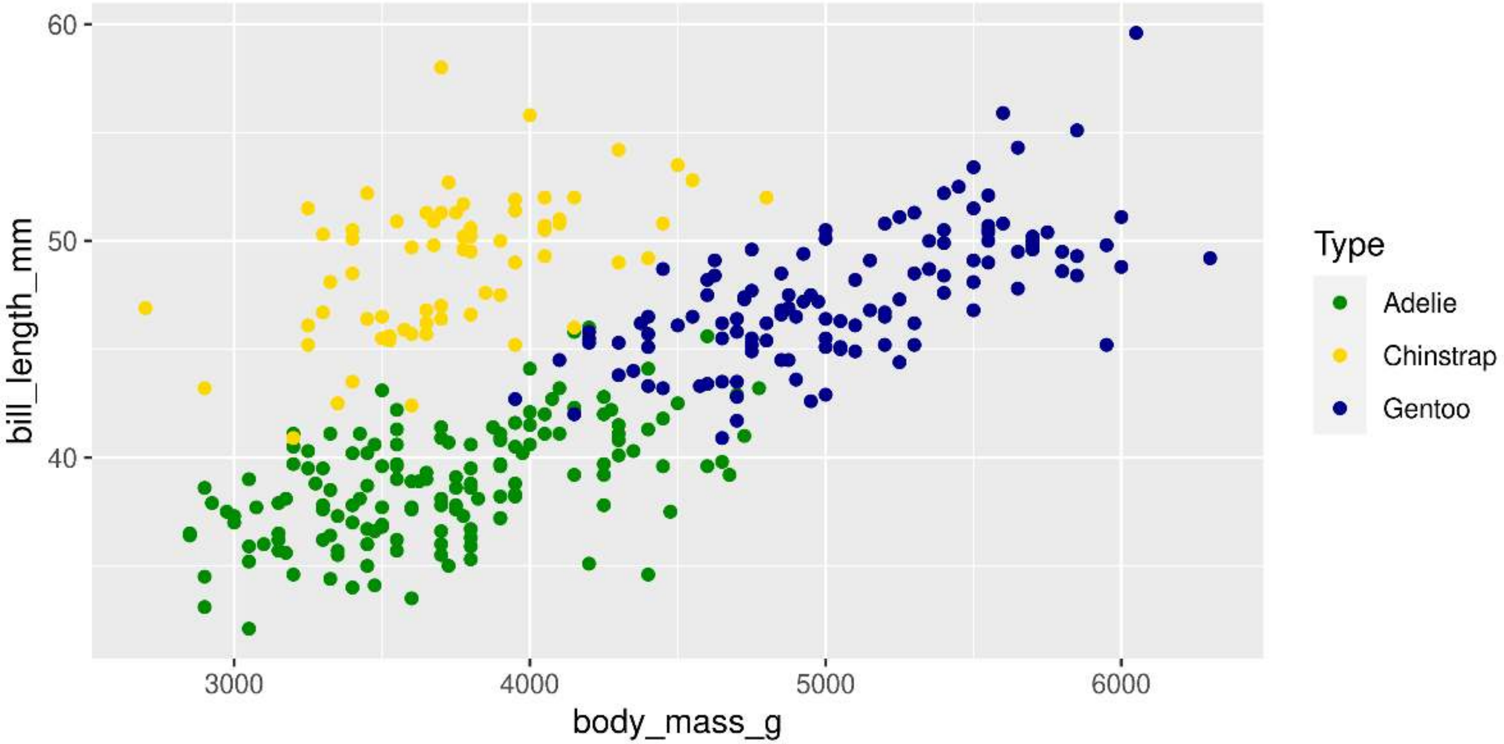


Customizing: Aesthetics

Using scales

Or be very explicit:

```
g + scale_colour_manual(
  name = "Type",
  values = c("Adelie" = "green4", "Gentoo" = "blue4", "Chinstrap" = "gold"),
  na.value = "black")
```

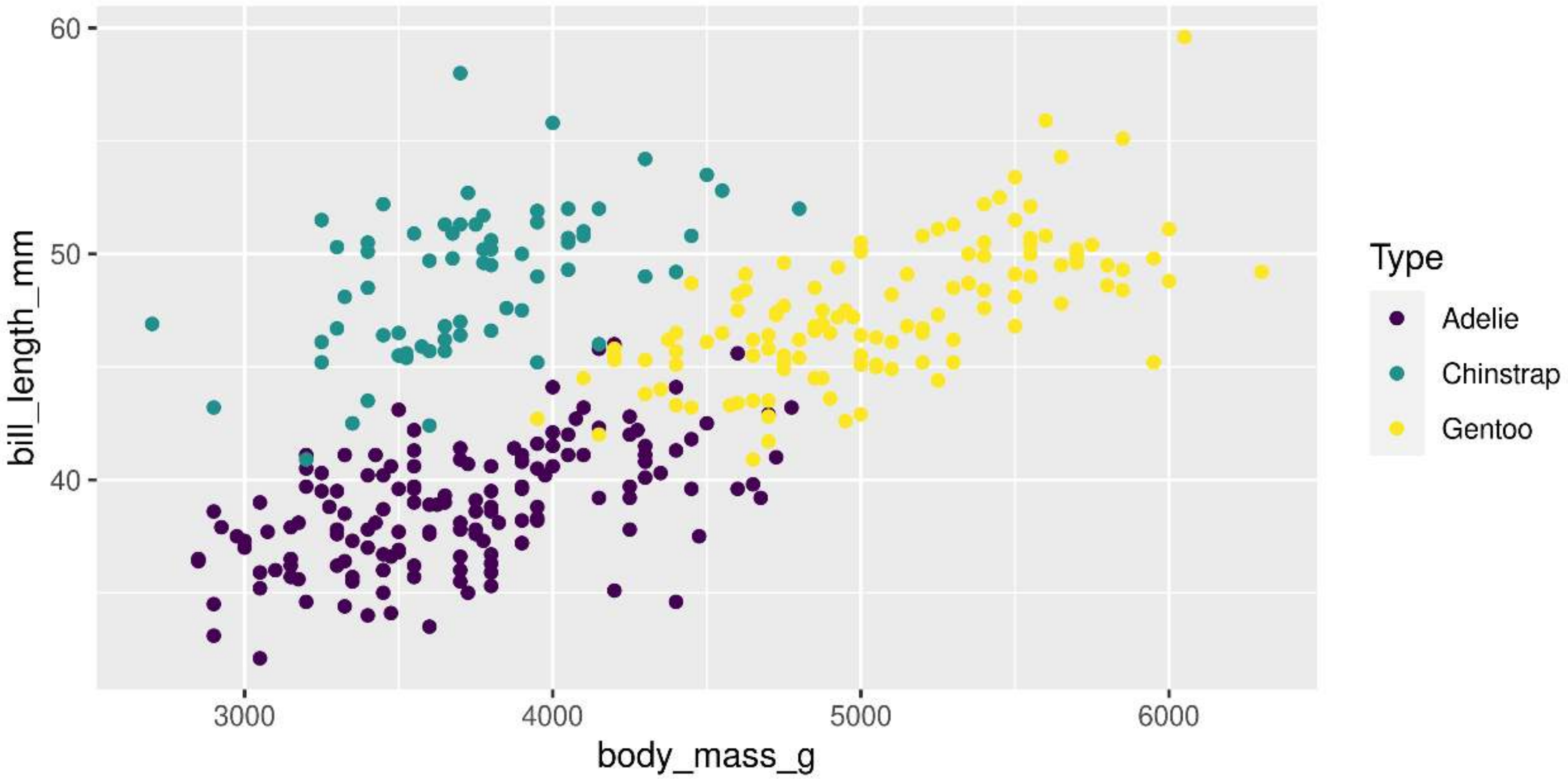


Customizing: Aesthetics

For colours, consider colour-blind-friendly scale

`viridis_d` for "discrete" data

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

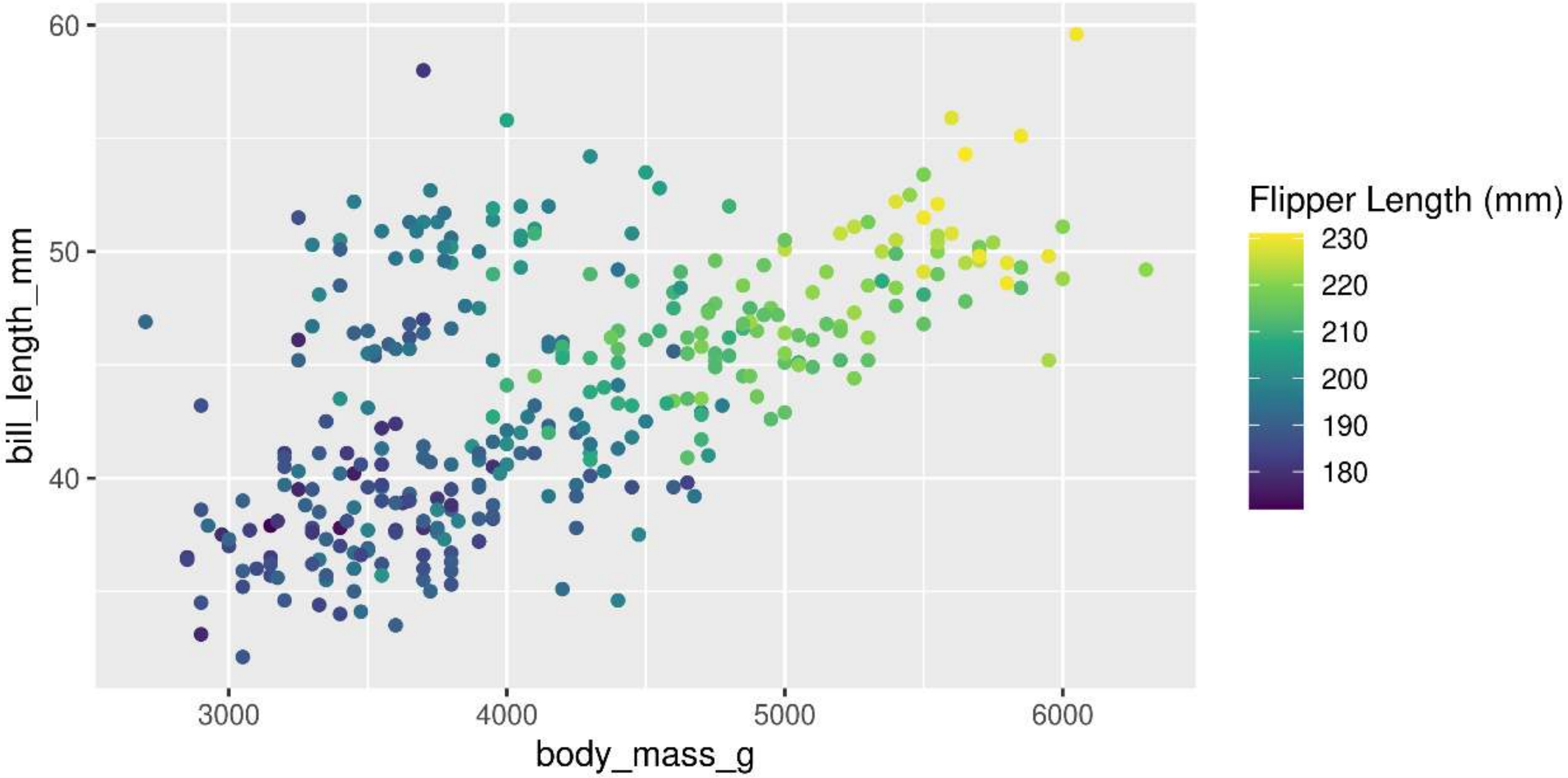


Customizing: Aesthetics

For colours, consider colour-blind-friendly scale

`viridis_c` for "continuous" data

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



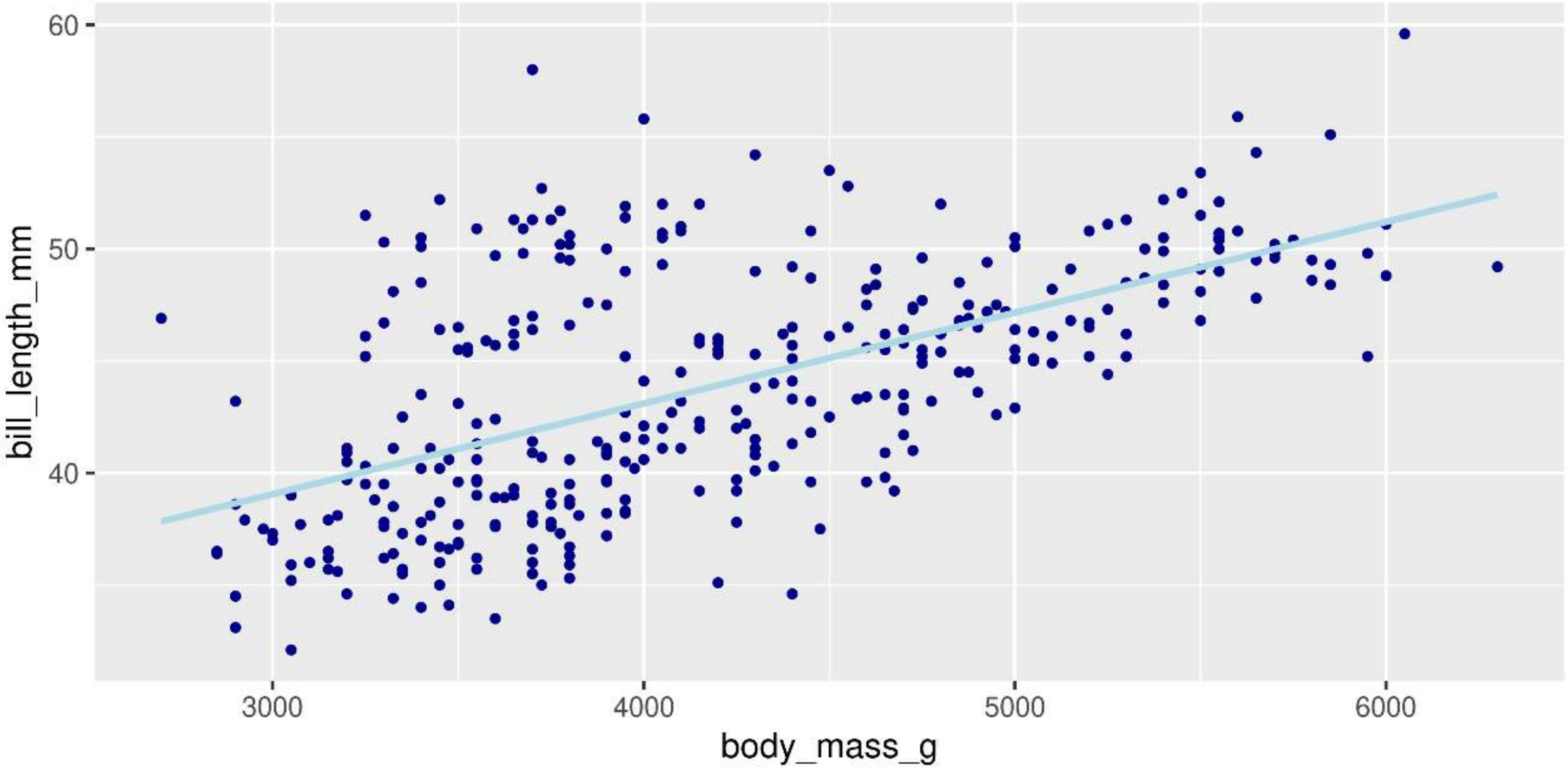
Customizing: Aesthetics

Forcing

Remove the association between a variable and an aesthetic

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

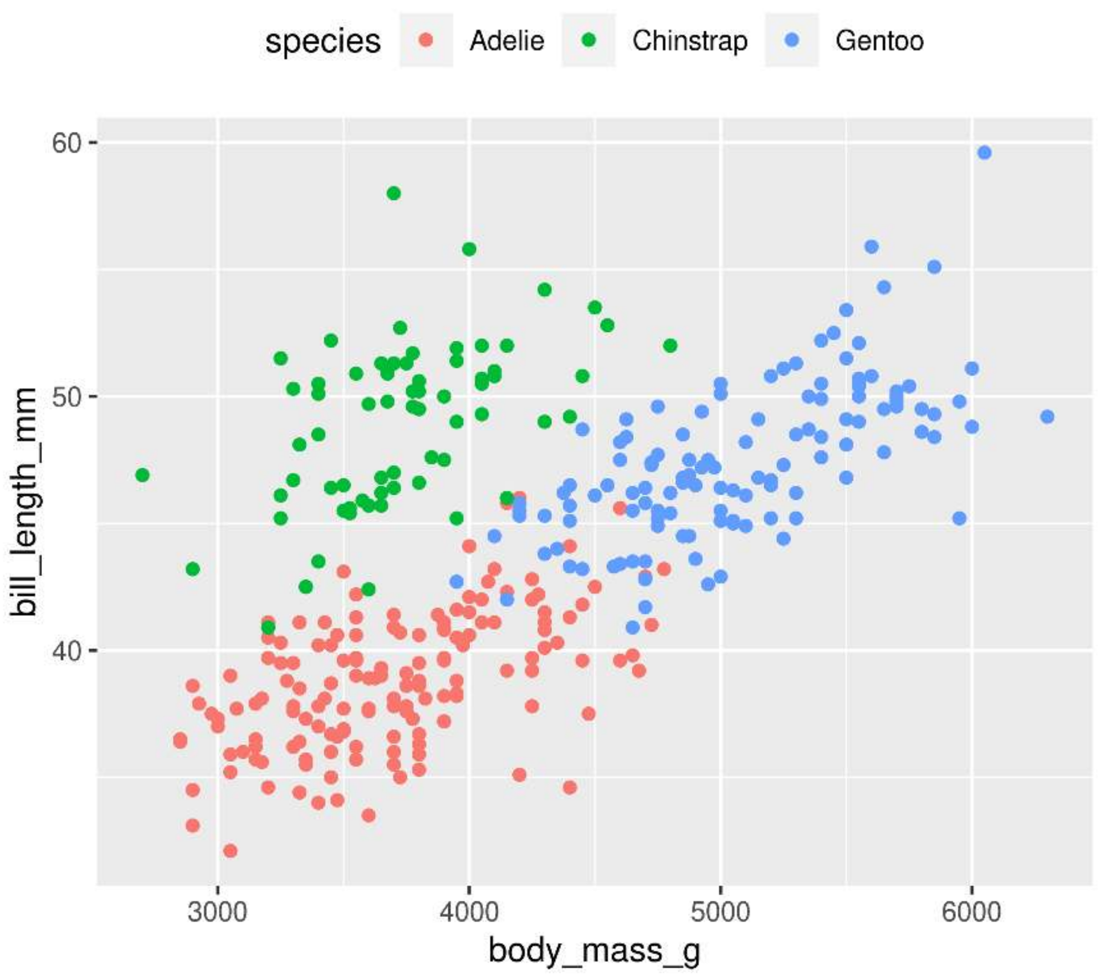
```
ggplot(data = penguins, aes(x = body_mass_g, y = bill_length_mm,
  geom_point(colour = "darkblue", size = 1) +
  stat_smooth(method = "lm", se = FALSE, colour = "lightblue")
```



Customizing: Legends placement

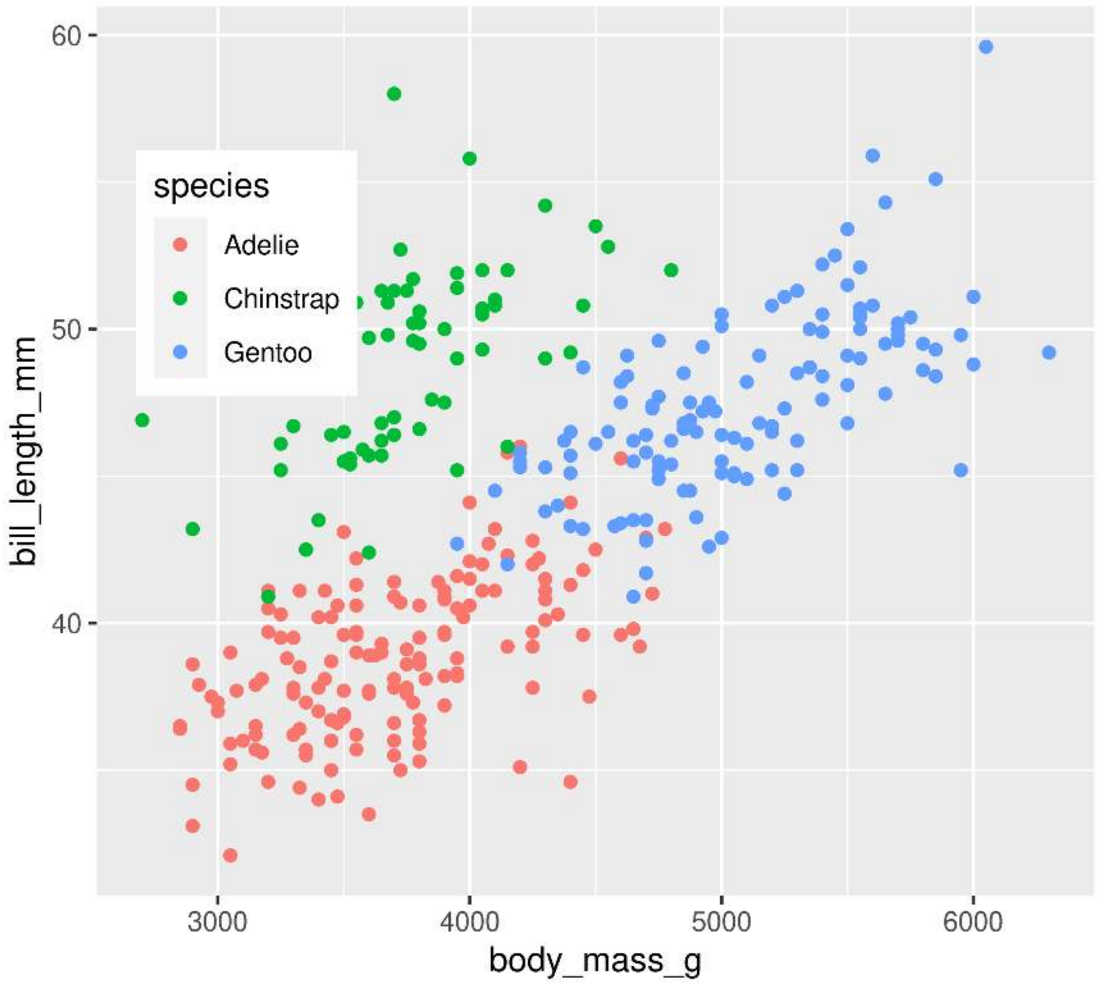
At the: top, bottom, left, right

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

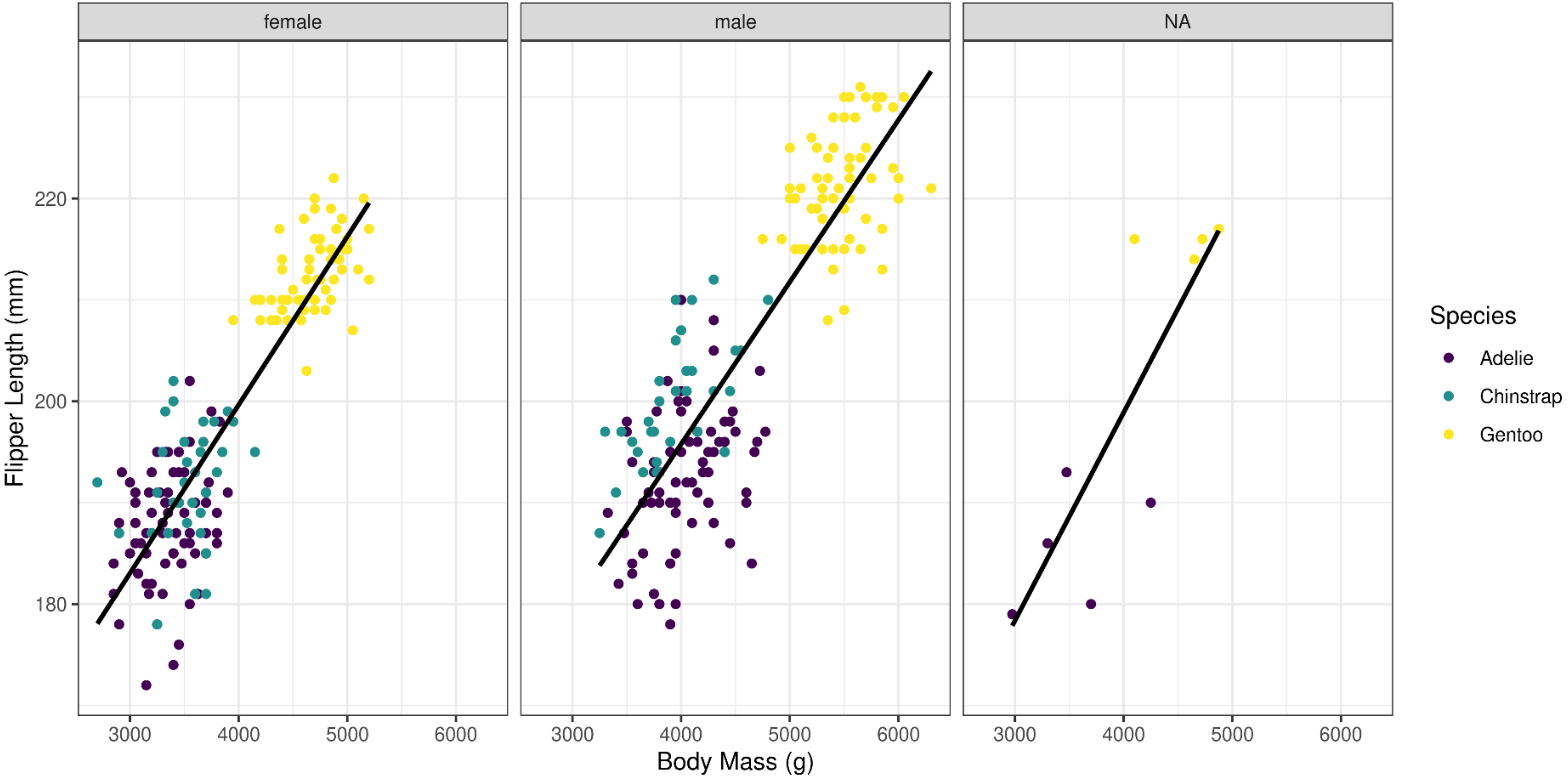


Exactly here

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



Your Turn: Create this plot

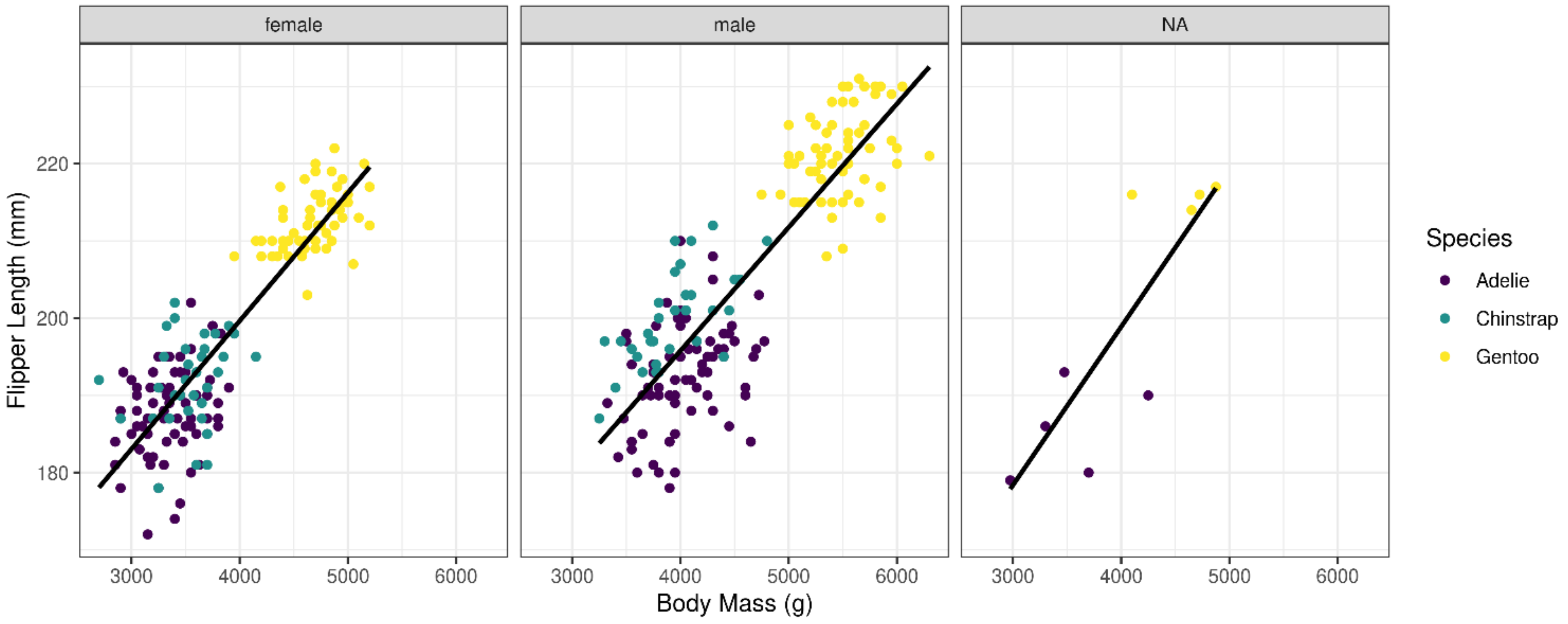


Too Easy?

Play with shape values >20 and fill and colour

Your Turn: Create this plot

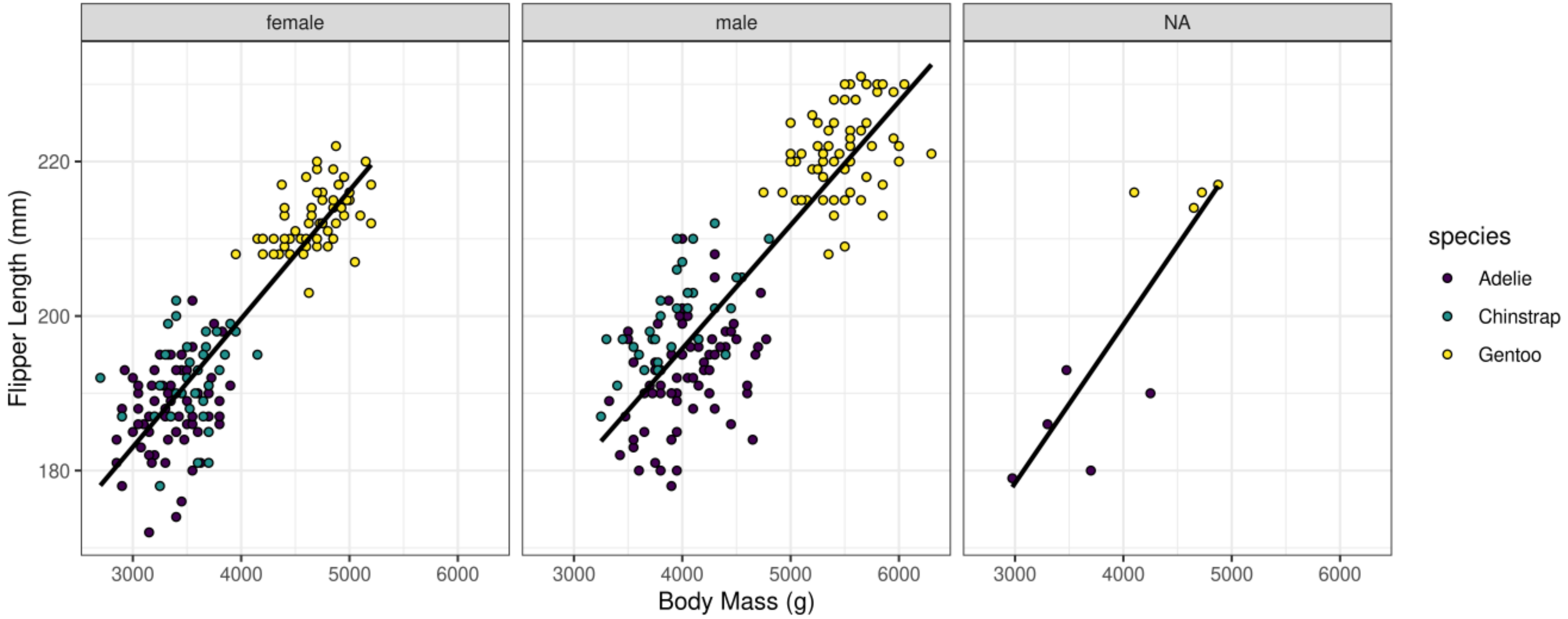
```r
ggplot(penguins, aes(x = body_mass_g, y = flipper_length_mm, colour = species)) +
  theme_bw() +
  geom_point() +
  stat_smooth(method = "lm", se = FALSE, colour = "black") +
  scale_colour_viridis_d() +
  facet_wrap(~ sex) +
  labs(x = "Body Mass (g)",
       y = "Flipper Length (mm)",
       colour = "Species")
```



Your Turn: Create this plot

Too easy?

```r
ggplot(penguins, aes(x = body_mass_g, y = flipper_length_mm, fill = species)) +
  theme_bw() +
  geom_point(shape = 21) +
  stat_smooth(method = "lm", se = FALSE, colour = "black", fill = NA) +
  scale_fill_viridis_d() +
  facet_wrap(~ sex) +
  labs(x = "Body Mass (g)",
       y = "Flipper Length (mm)",
       colour = "Species")
```



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)

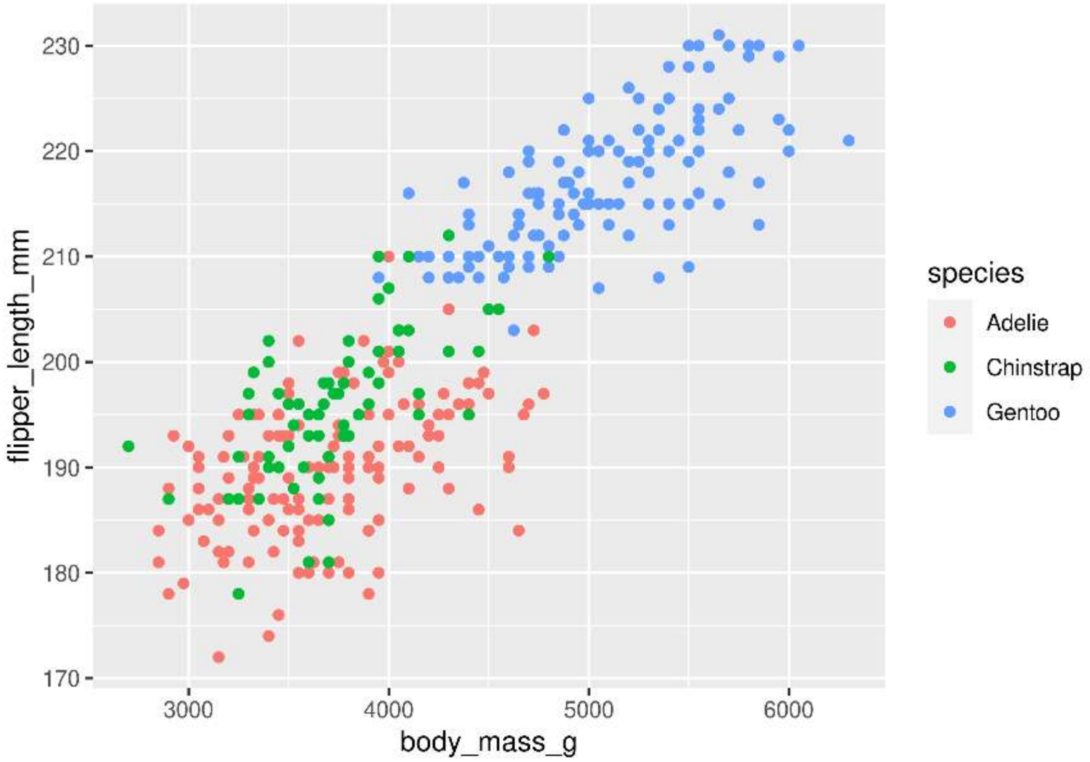
Order of operations

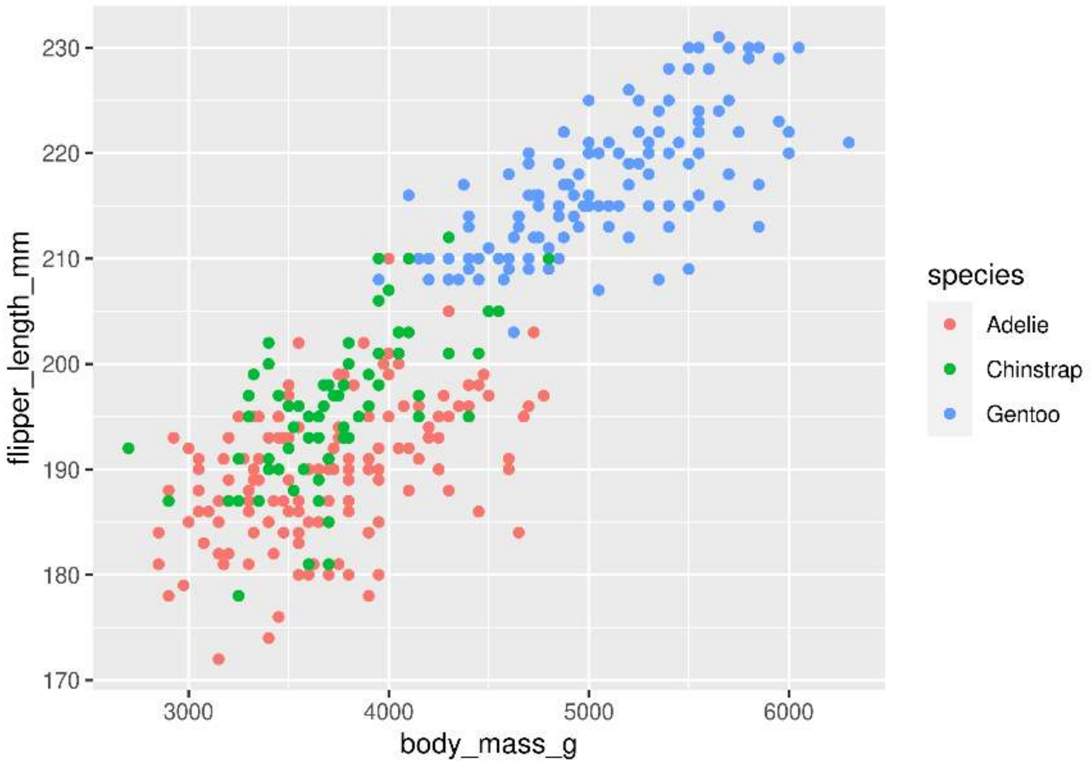
Where to put the `aes()`?

Sometimes it doesn’t matter…

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

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





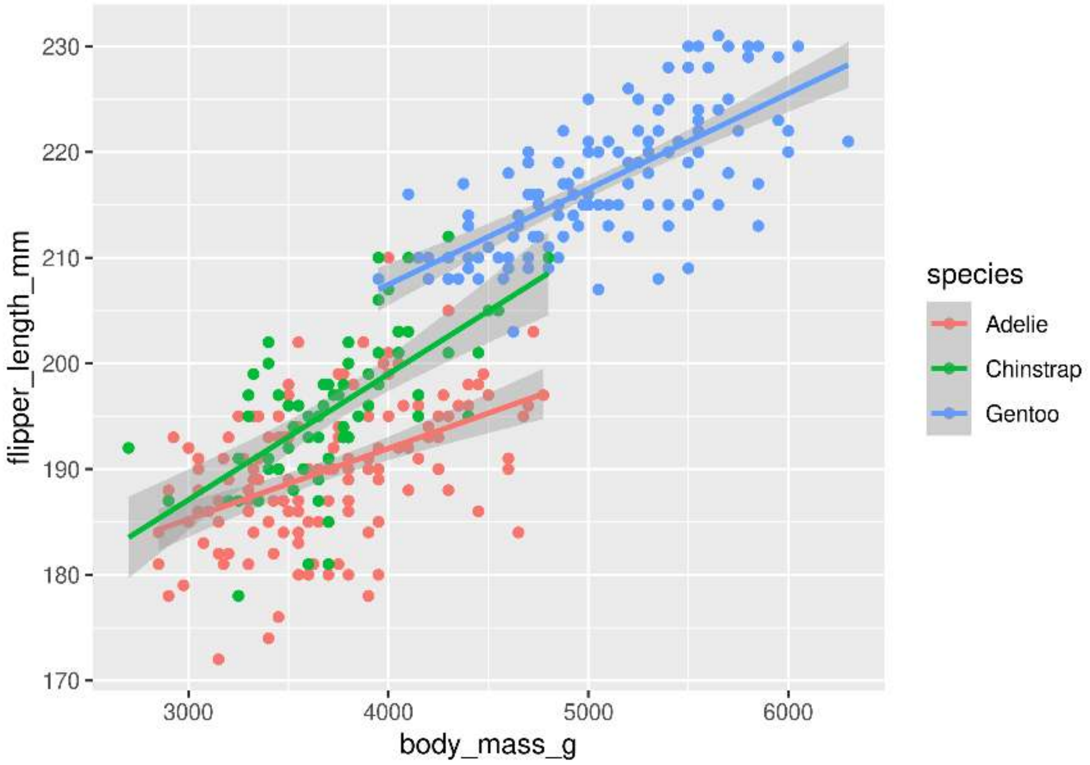
Order of operations

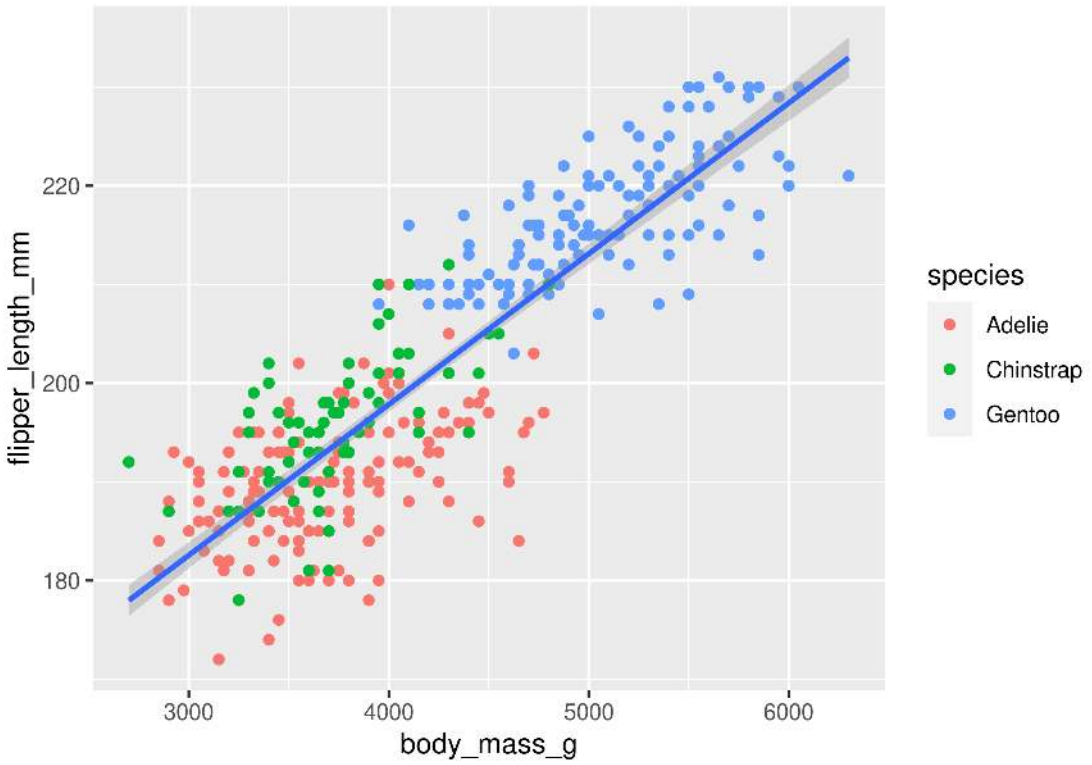
Where to put the `aes()`?

Sometimes it DOES matter…

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

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



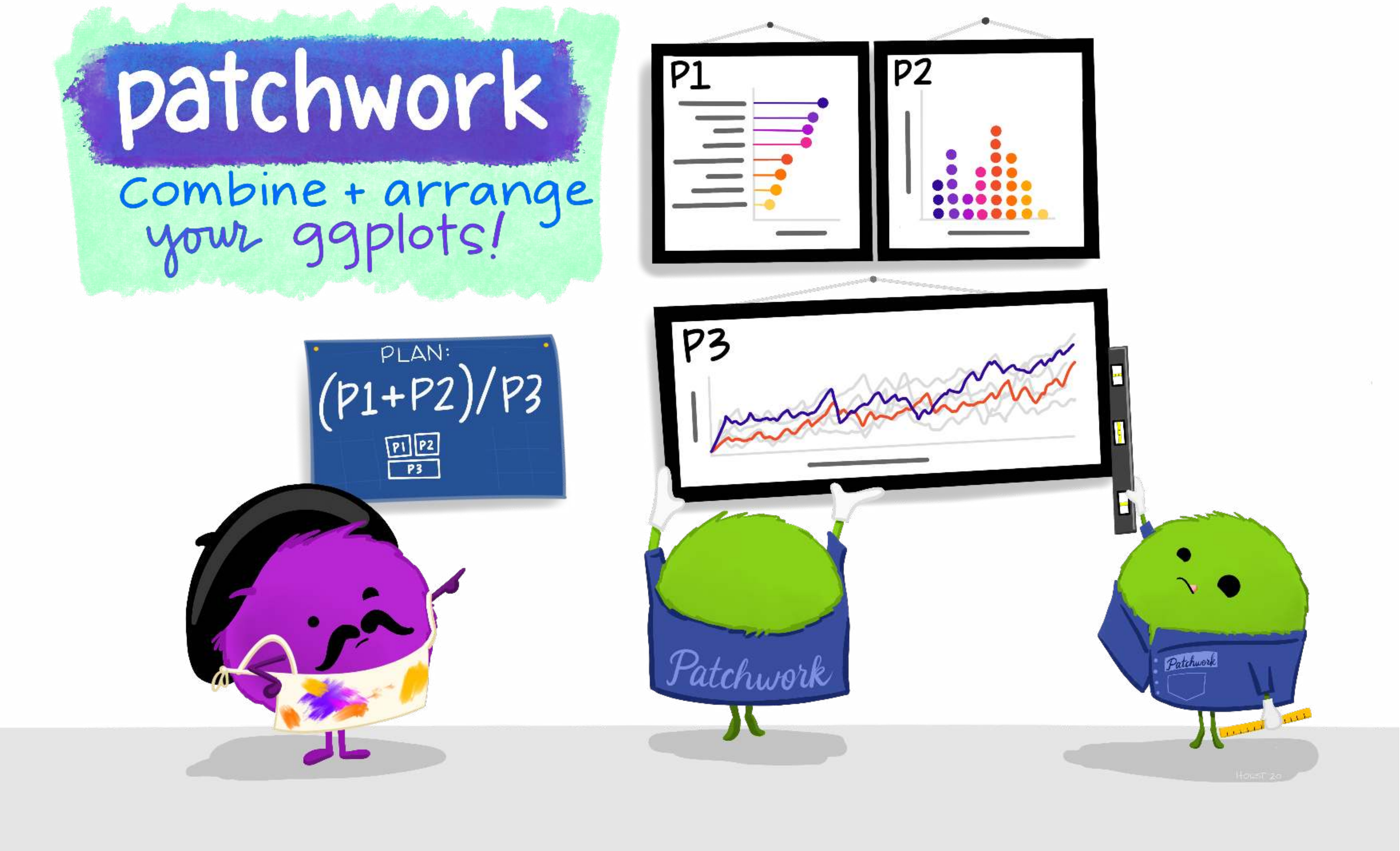


Applies to ALL lines in the ggplot including `stat_smooth()`

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

not `stat_smooth()`

Combining plots with `patchwork`


patchwork
combine + arrange your ggplots!
P1
P2
P3
PLAN:
(P1+P2)/P3
P1 P2
P3
Patchwork
Patchwork
Horst '20

Combining plots

Setup

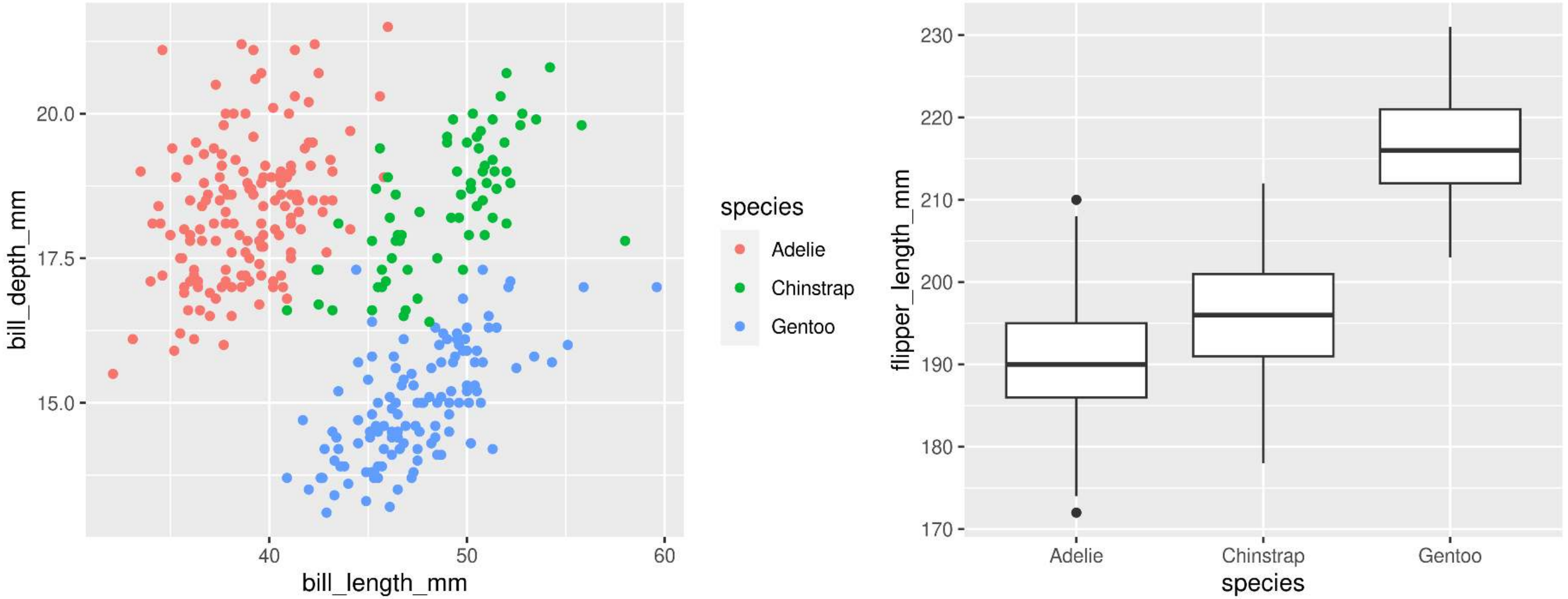
- Load `patchwork`
- Create a couple of different plots

```r
library(patchwork)

g1 <- ggplot(data = penguins, aes(x = bill_length_mm, y = bill_depth_mm, colour = species)) +
  geom_point()

g2 <- ggplot(data = penguins, aes(x = species, y = flipper_length_mm)) +
  geom_boxplot()

g3 <- ggplot(data = penguins, aes(x = flipper_length_mm, y = body_mass_g, colour = species)) +
  geom_point()
```

Combining plots with `patchwork`

Side-by-Side 2 plots

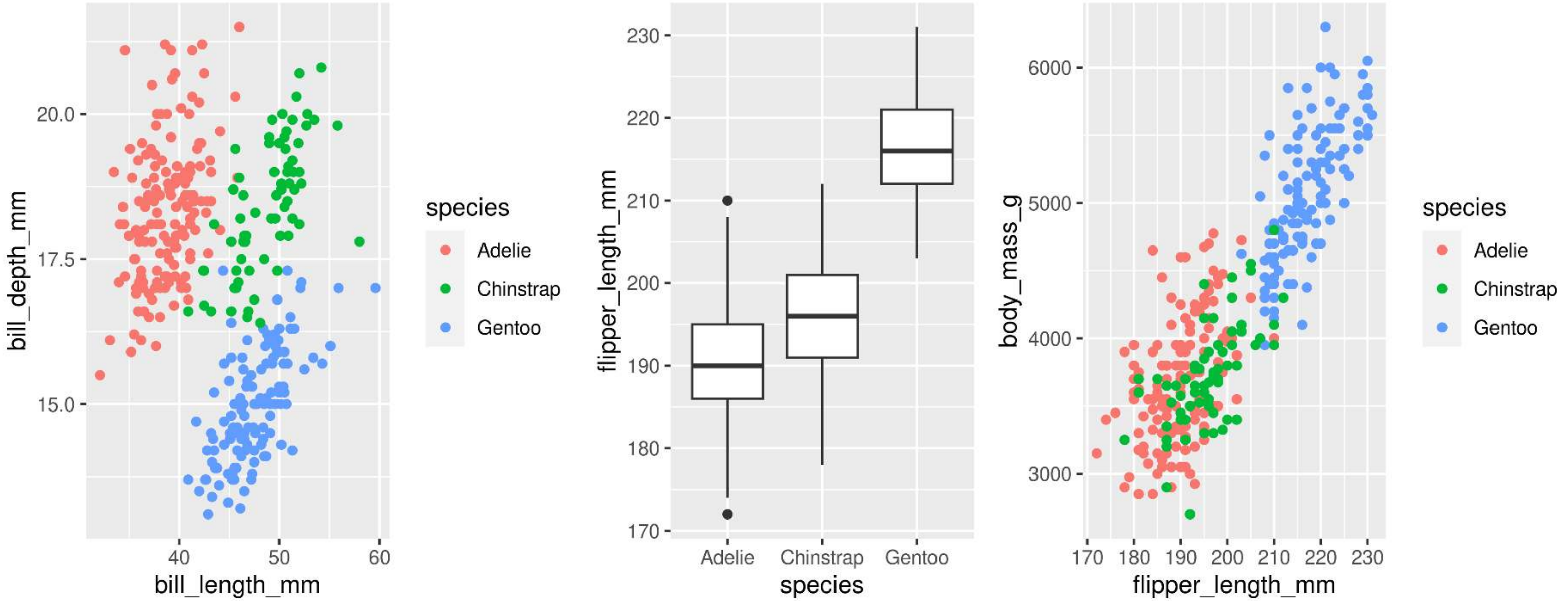
```r
g1 + g2
```



Combining plots with `patchwork`

Side-by-Side 3 plots

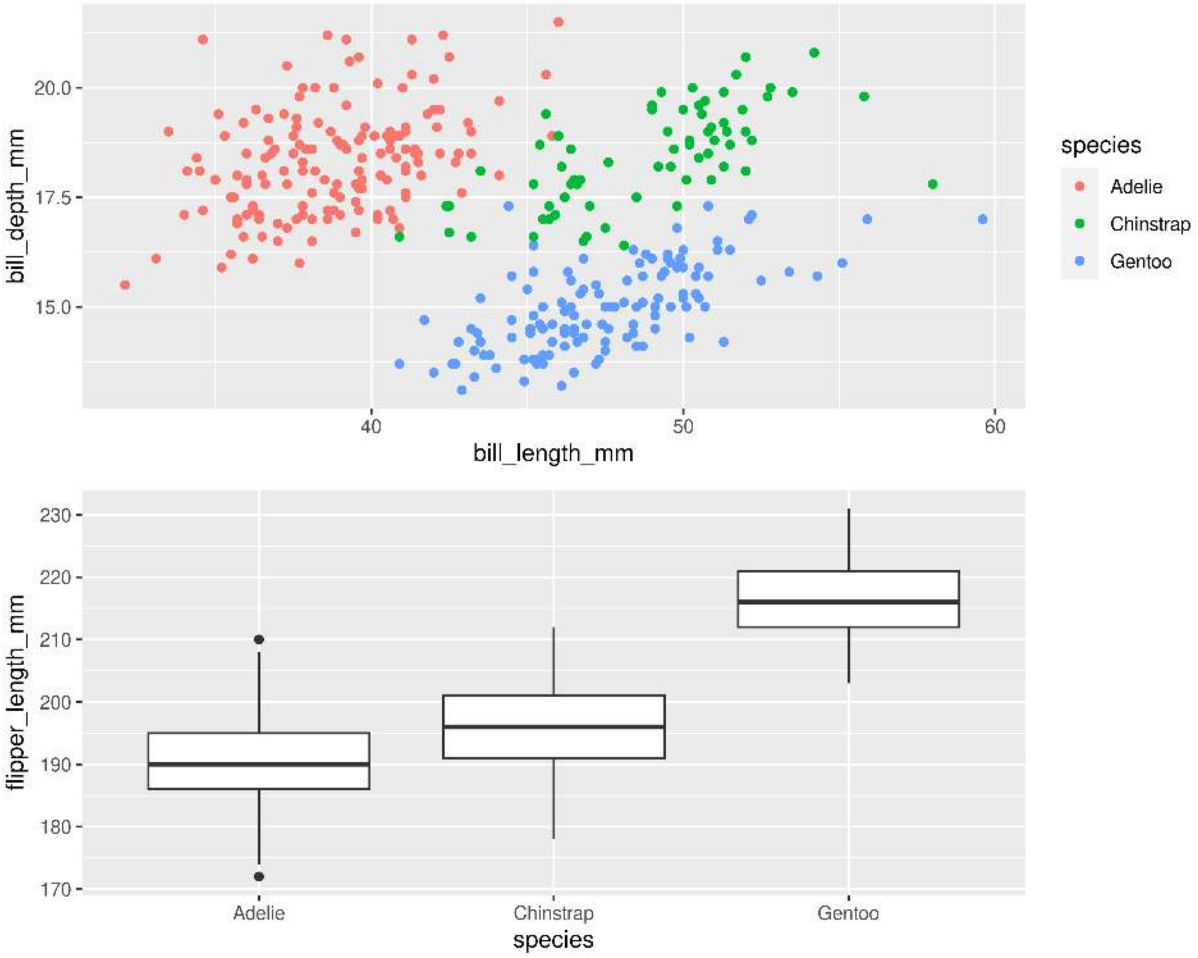
```
g1 + g2 + g3
```



Combining plots with `patchwork`

Stacked 2 plots

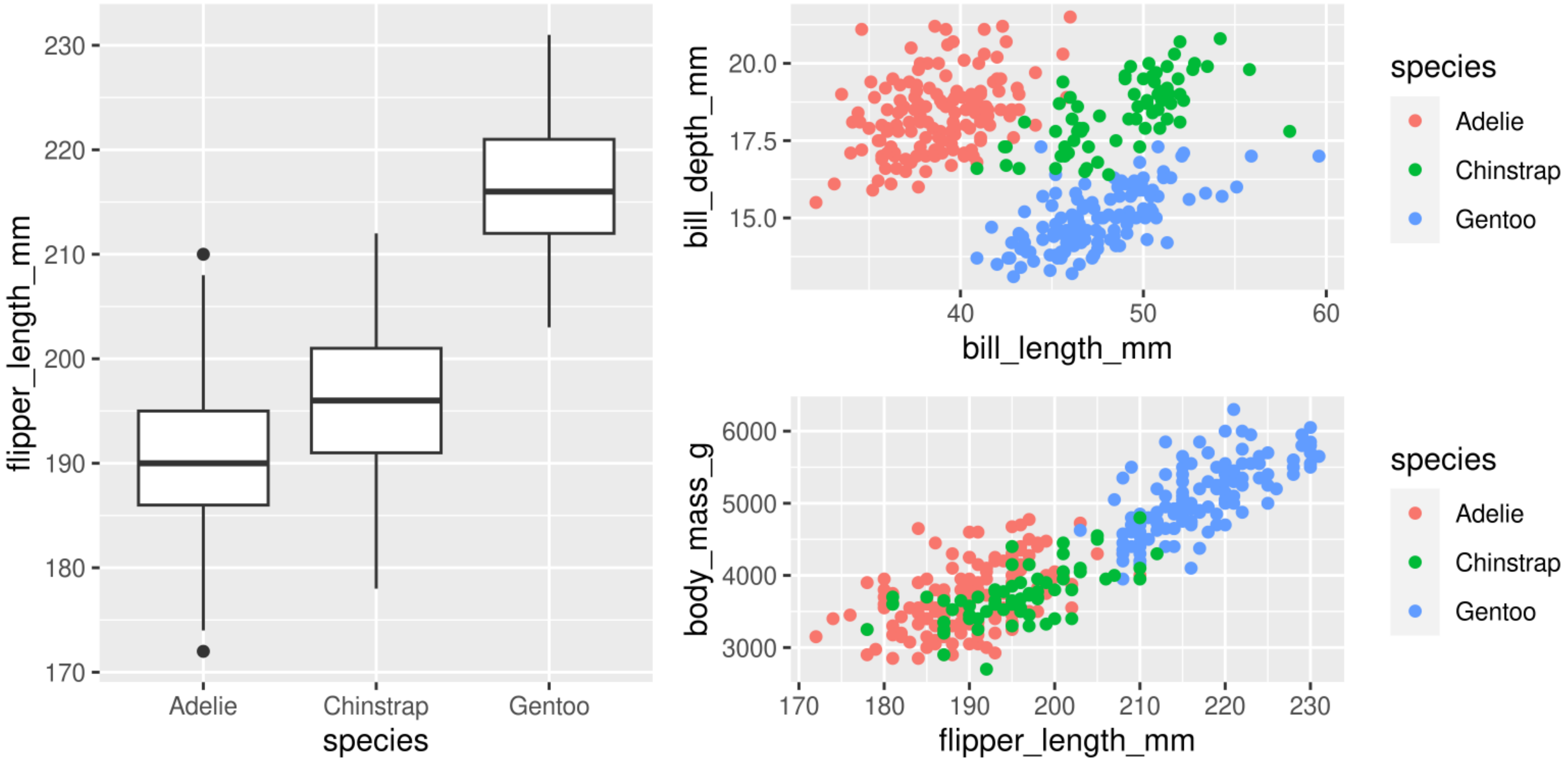
```
g1 / g2
```



Combining plots with `patchwork`

More complex arrangements

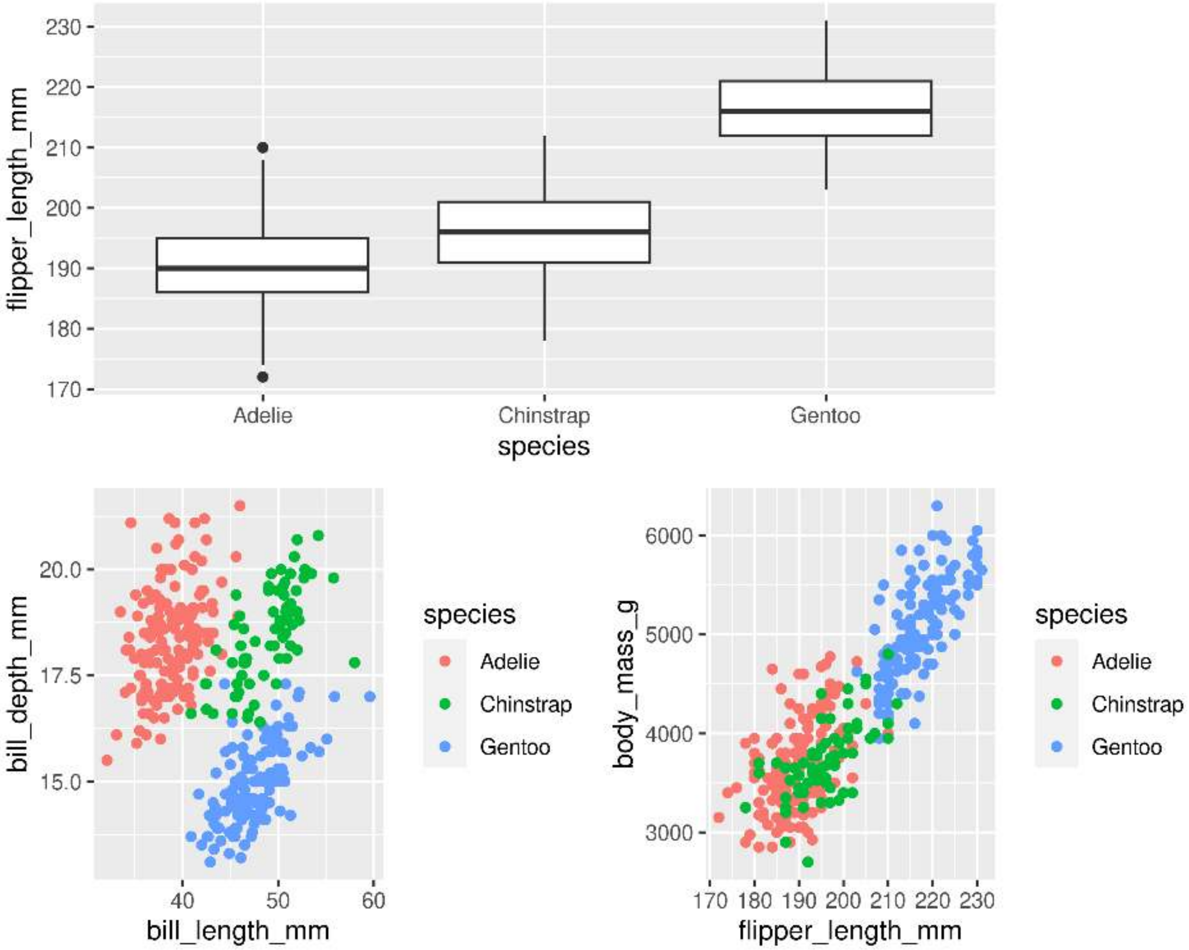
```
g2 + (g1 / g3)
```



Combining plots with `patchwork`

More complex arrangements

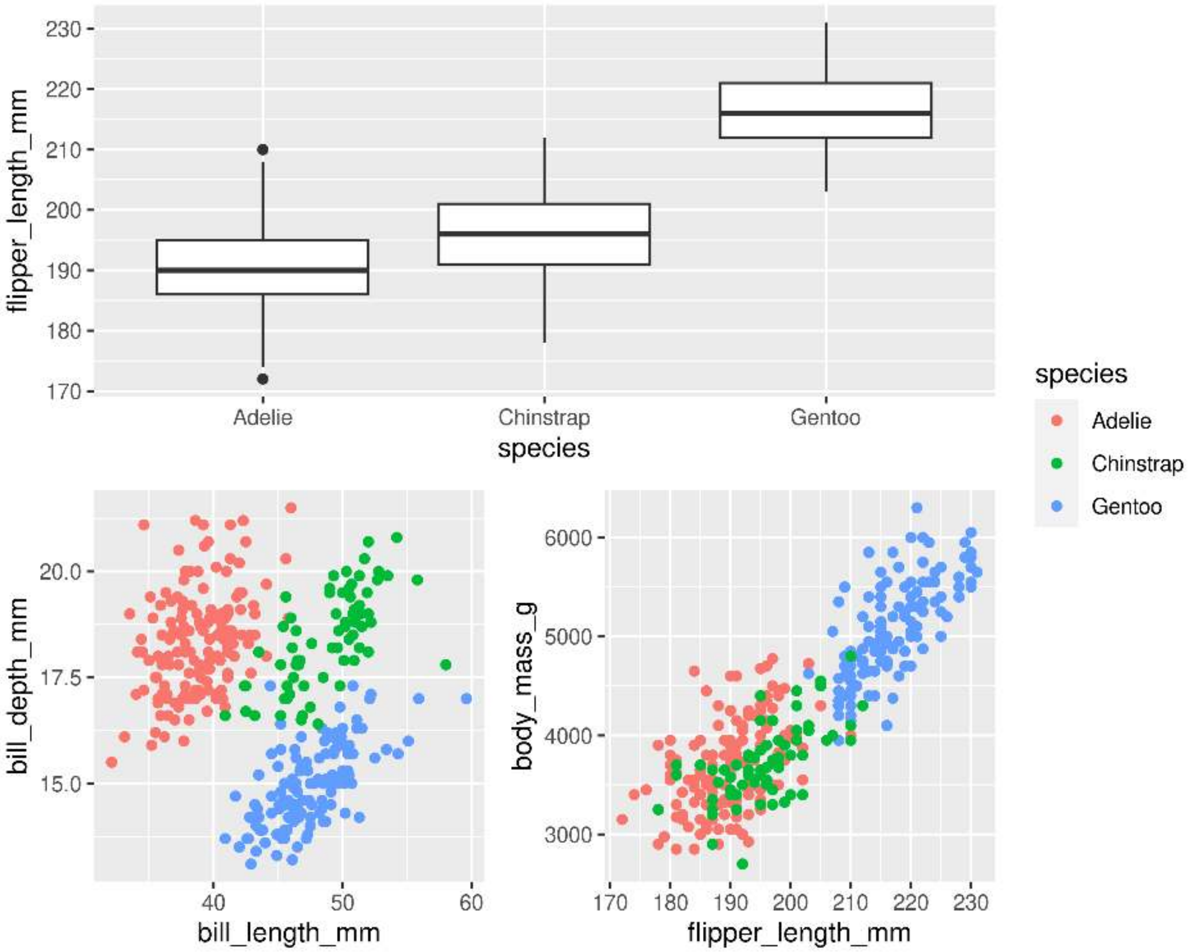
```
g2 / (g1 + g3)
```



Combining plots with `patchwork`

“collect” common legends

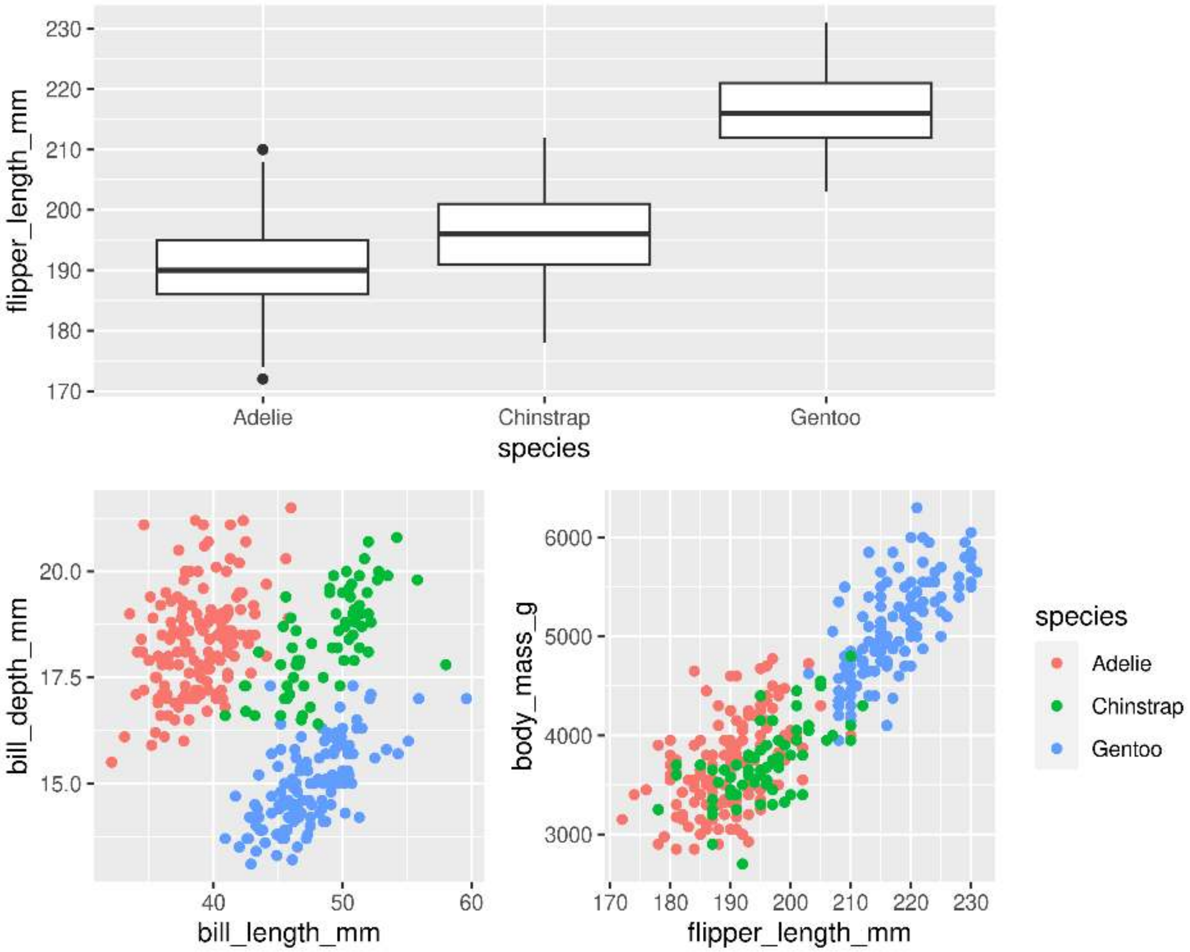
```
g2 / (g1 + g3) + plot_layout(guides = "collect")
```



Combining plots with `patchwork`

“collect” common legends

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



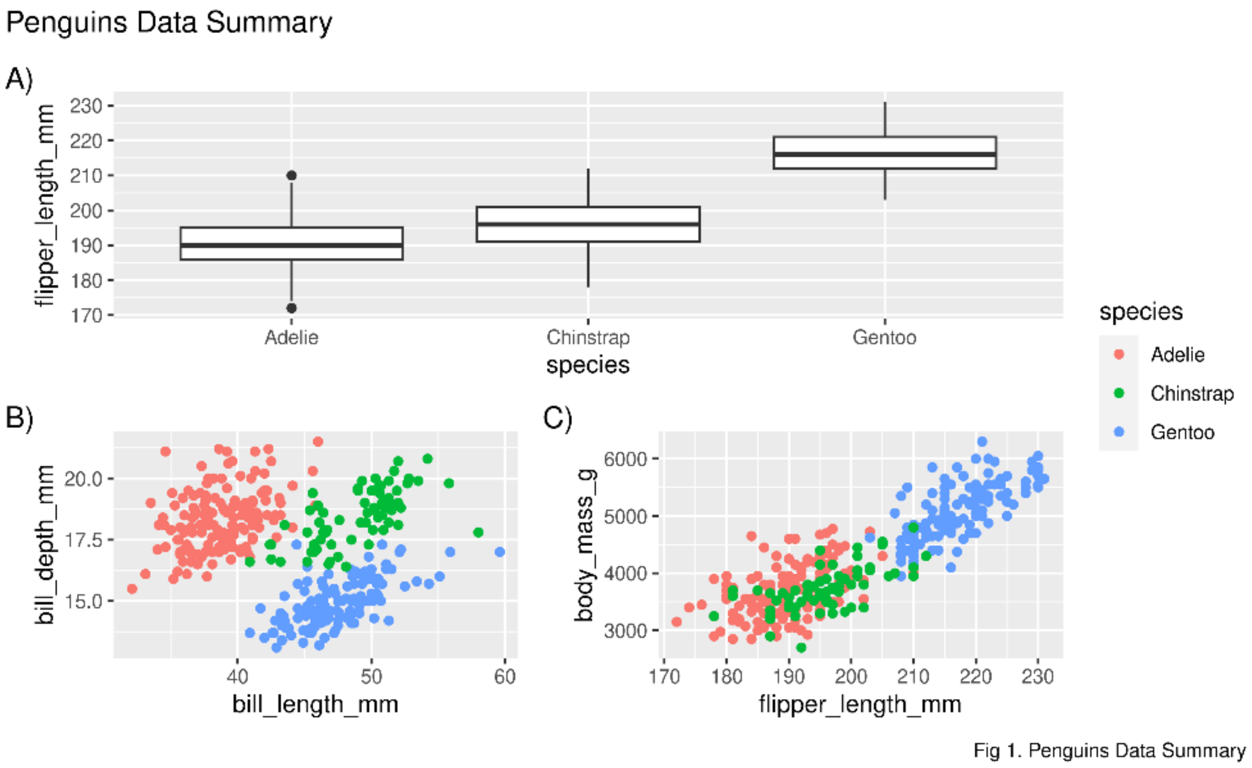
Combining plots with `patchwork`

Annotate

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

Too Easy?

Can you figure out how to collect common axes as well?



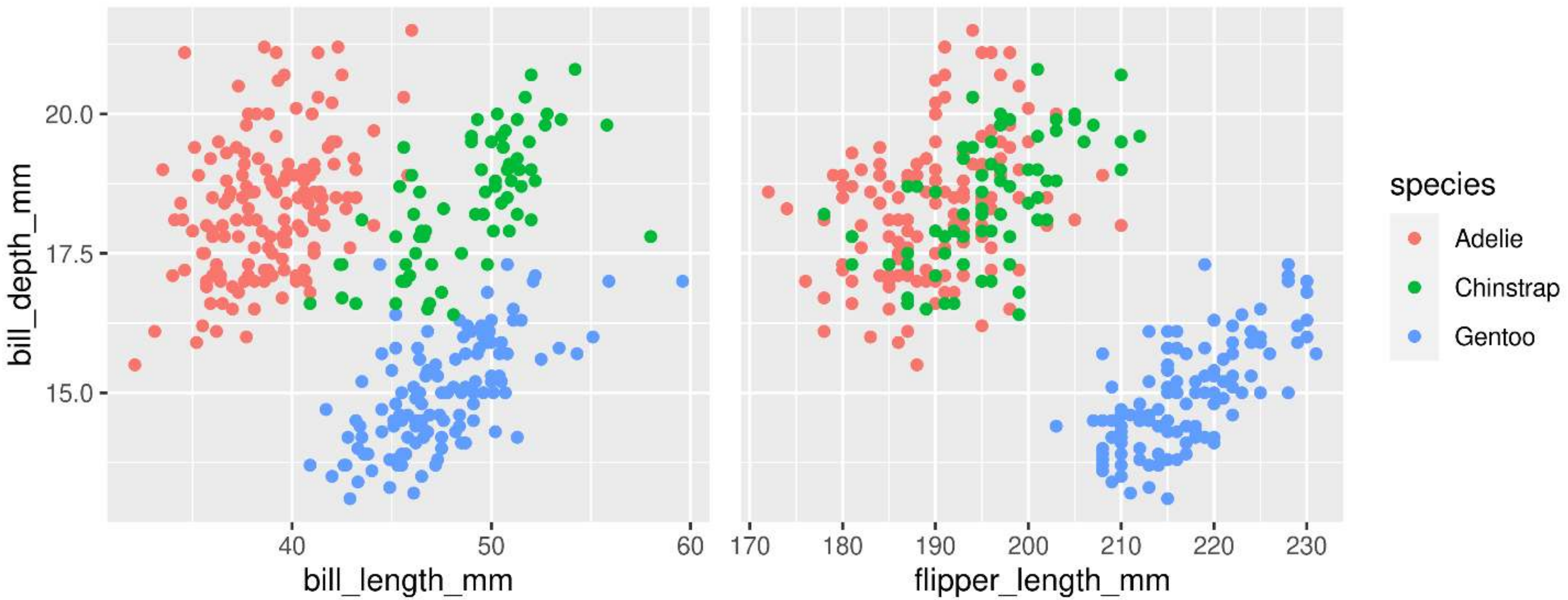
Your Turn

Combine any 3 figures

Your Turn: Combine plots

Too easy?

```r
g1 <- ggplot(data = penguins, aes(x = bill_length_mm, y = bill_depth_mm, colour = species)) +
  geom_point()

g2 <- ggplot(data = penguins, aes(x = flipper_length_mm, y = bill_depth_mm, colour = species)) +
  geom_point()

g1 + g2 + plot_layout(guides = "collect", axes = "collect")
```



Saving plots

Saving plots

RStudio Export

Demo

`ggsave()`

```r
g <- ggplot(penguins, aes(x = sex, y = bill_length_mm)) +
  geom_boxplot()

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

```r
g <- ggplot(penguins, aes(x = sex, y = body_mass_g)) +
  geom_boxplot() +
  labs(x = "Sex", y = "Body Mass (g)") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

ggsave(filename = "penguins_mass.pdf", plot = g, dpi = 300,
       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:

```r
n <- count(penguins, island)

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

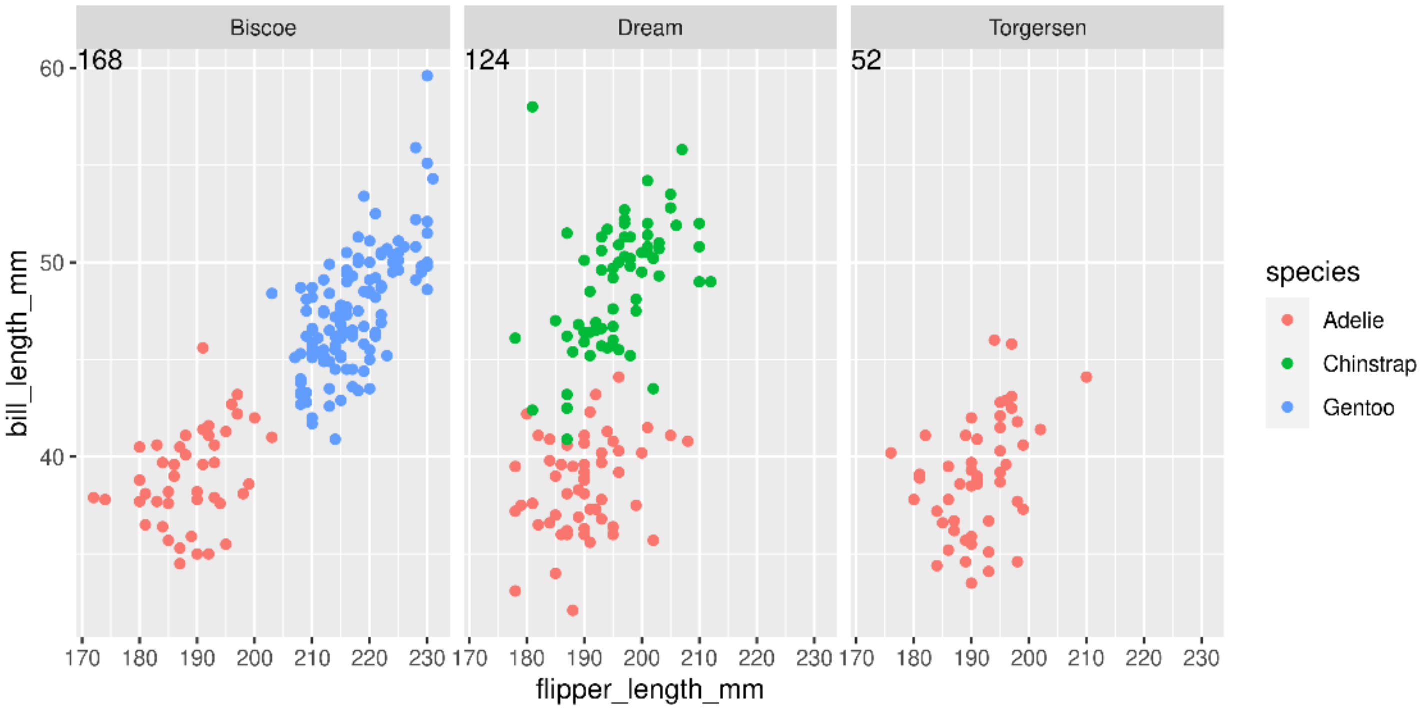
```
Error in `geom_text()`:
! Problem while computing aesthetics.
ℹ Error occurred in the 2nd layer.
Caused by error:
! object 'species' not found
```

Wrapping up: Common mistakes

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

Either move the aesthetic…

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



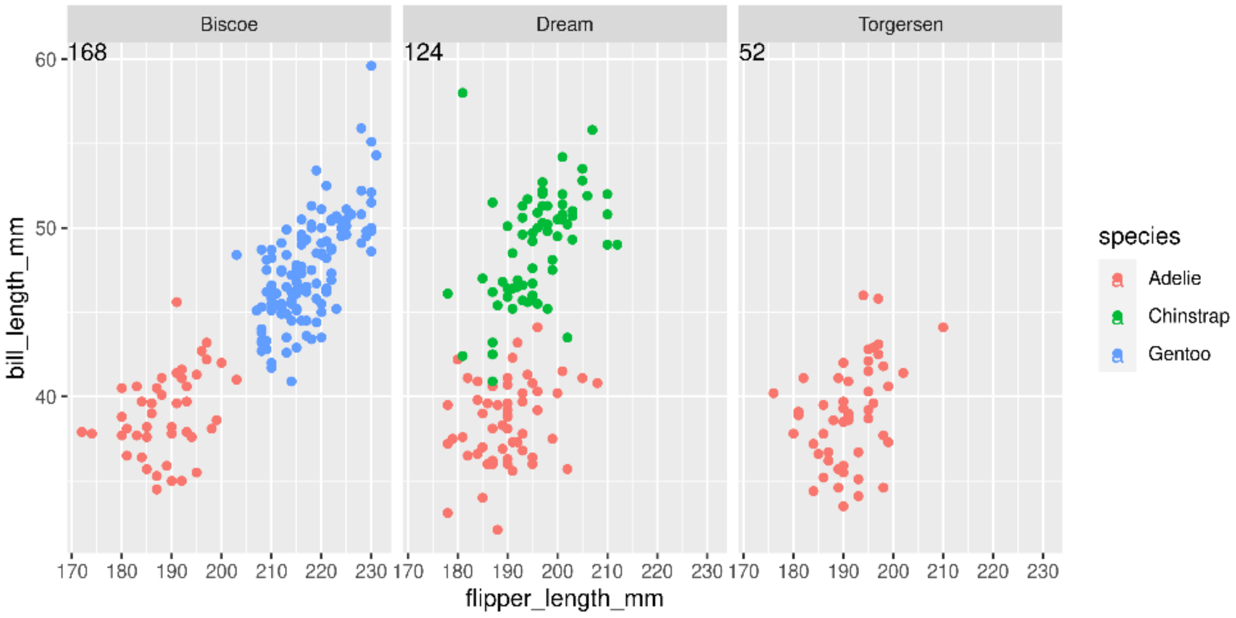
Wrapping up: Common mistakes

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…

```r
ggplot(penguins, aes(x = flipper_length_mm, y = bill_length_mm, colour = species)) +
  geom_point() +
  facet_wrap(~ island) +
  geom_text(data = n, aes(label = n, colour = NULL),
            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