NRI 7350

Plots and Loading Data

Artwork by @allison_horst

Check-in

- Everyone getting emails?
  - Email about having your own data?
  - Email about these slides?
- Everyone have access to these slides?
  https://steffilazerte.ca/NRI_Labs/slides.html

Getting started (again)

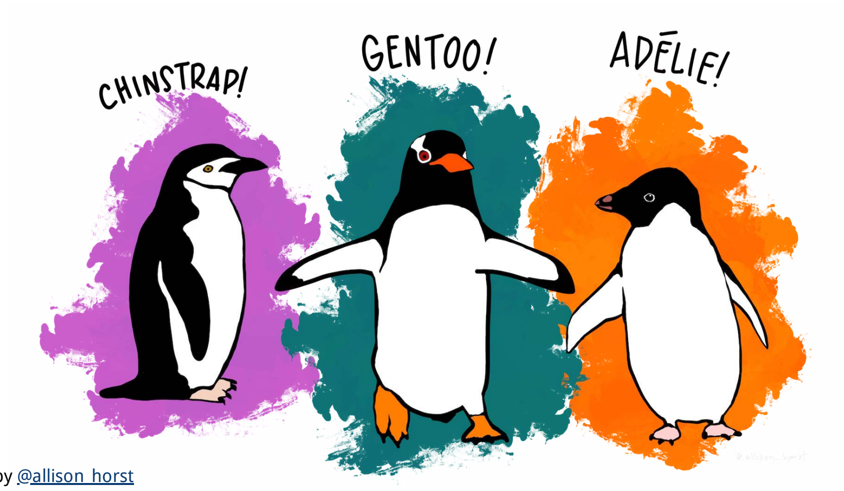
Open RStudio
Open your NRI project
Open a **new** script for today:
File > New File > R Script

Make sure to load `tidyverse` at the top:
`library(tidyverse)`

Creating Figures

Artwork by @allison_horst

Our data set: Palmer Penguins!

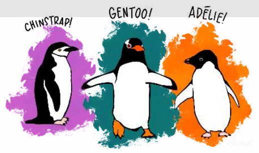


Artwork by @allison_horst

Our data set: Palmer Penguins!

```
library(palmerpenguins)
penguins
```

```
## # A tibble: 344 x 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
## # … with 334 more rows
```



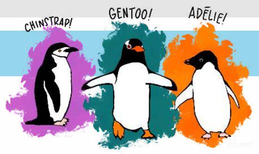
Artwork by @allison_horst

Our data set: Palmer Penguins!

```
library(palmerpenguins)
penguins
```

```
## # A tibble: 344 x 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
## # … with 334 more rows
```

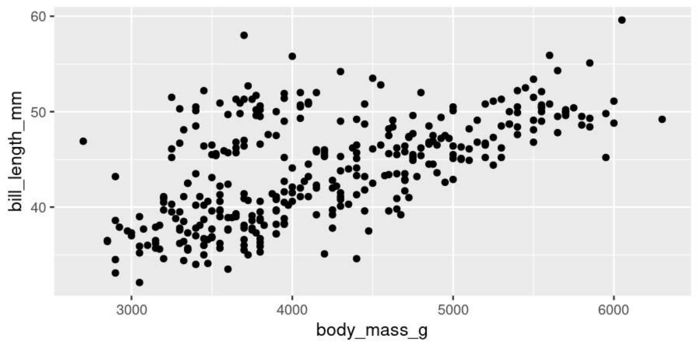
Your turn! Run this code and look at the output in the console



Artwork by @allison_horst

A basic plot

```
library(palmerpenguins)
library(tidyverse)

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



Break it down

```
library(palmerpenguins)
library(tidyverse)

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

`library(palmerpenguins)`

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

Break it down

```
library(palmerpenguins)
library(tidyverse)

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

`library(tidyverse)`

- Load the **`tidyverse`** package (which loads the **`ggplot2`** package)

Break it down

```
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

```
library(palmerpenguins)
library(tidyverse)

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

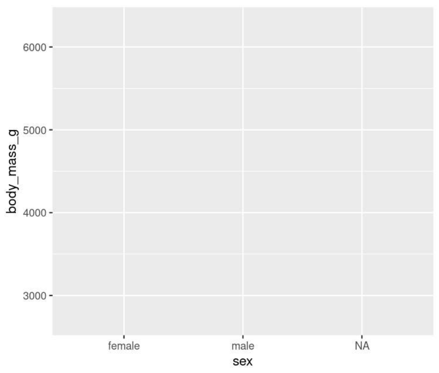
`geom_point()`

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

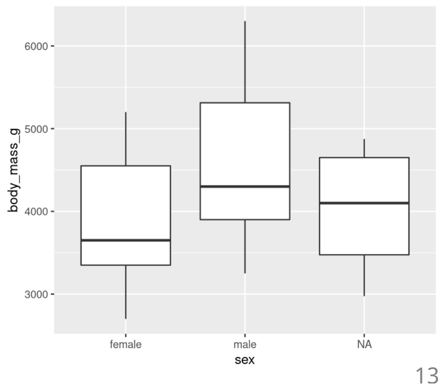
> ggplots are essentially layered objects, starting with a call to **`ggplot()`**

Plots are layered

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

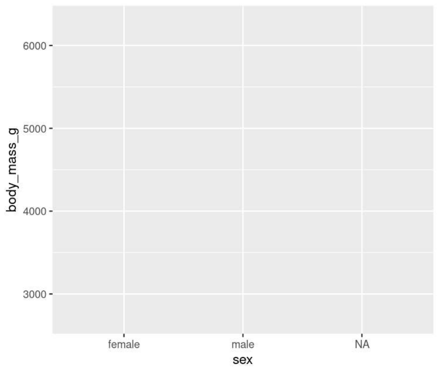


```
ggplot(data = penguins, aes(x = sex, y =
body_mass_g)) +
  geom_boxplot()
```

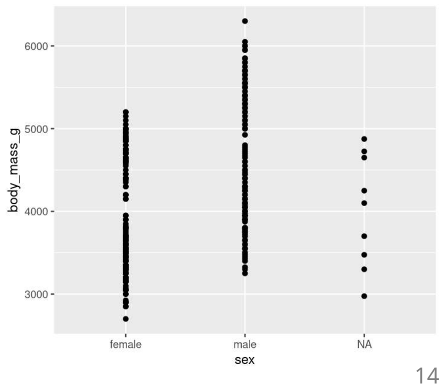


Plots are layered

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

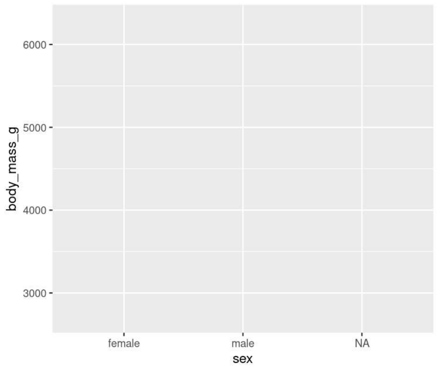


```
ggplot(data = penguins, aes(x = sex, y =
body_mass_g)) +
  geom_point()
```

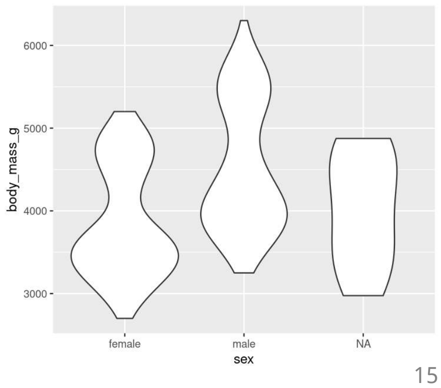


Plots are layered

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



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

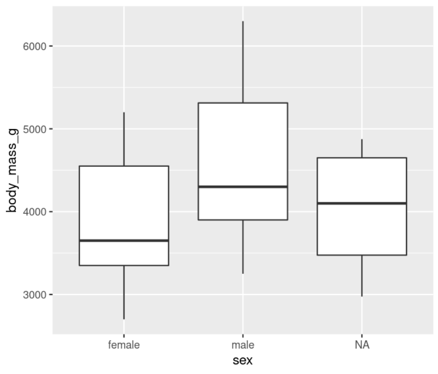


15 / 70

Plots are layered

You can add multiple layers

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

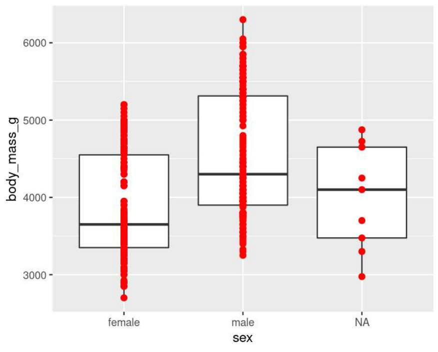


16 / 70

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")
```

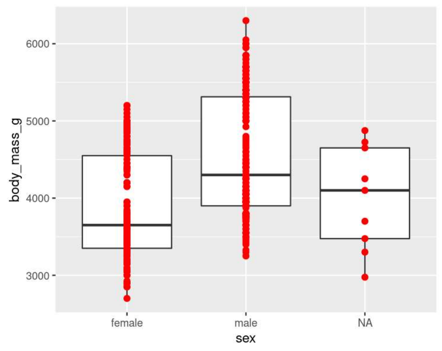


17 / 70

Plots are layered

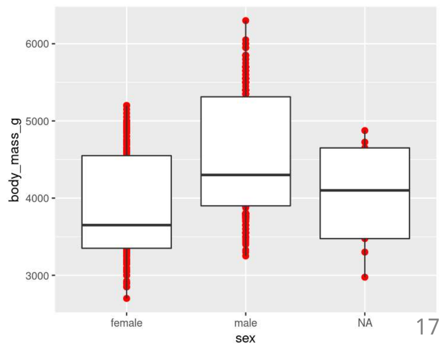
You can add multiple layers

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



Order matters

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



17 / 70

Plots are objects

Any ggplot can be saved as an object

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

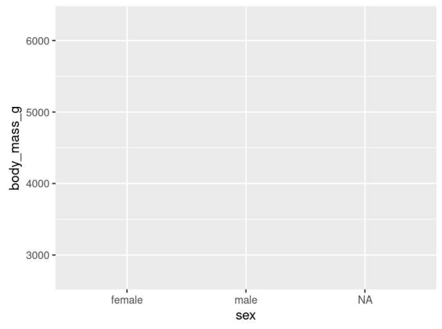
18 / 70

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
```



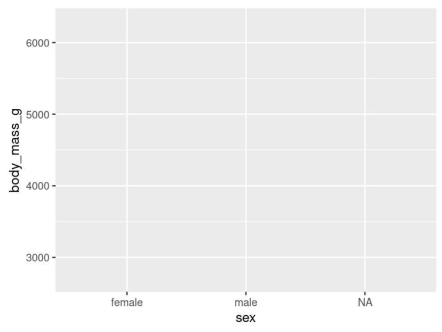
18 / 70

Plots are objects

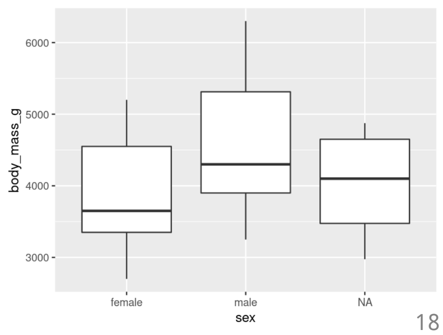
Any ggplot can be saved as an object

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

```
g
```



```
g + geom_boxplot()
```

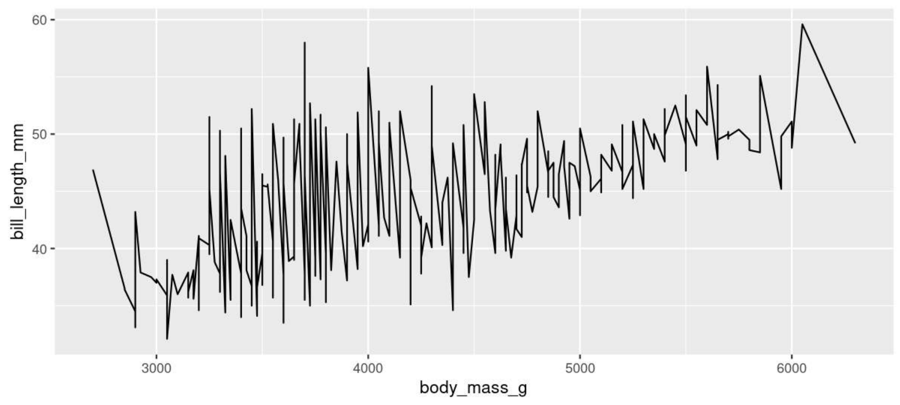


More Geoms

(Plot types)

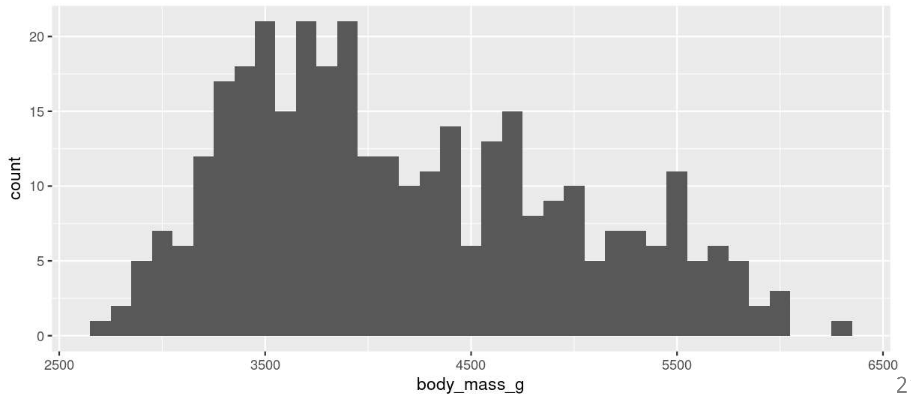
Geoms: Lines

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



Geoms: Histogram

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

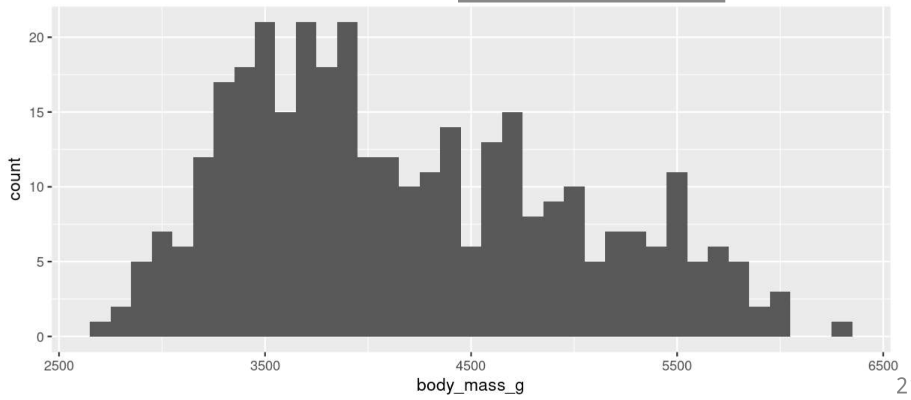


21 / 70

Geoms: Histogram

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

Note: We only need 1 aesthetic here (x)

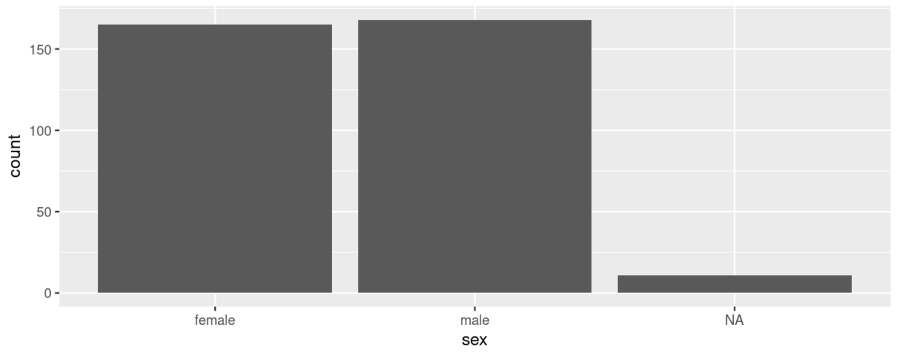


21 / 70

Geoms: Barplots

Let `ggplot` count your data

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

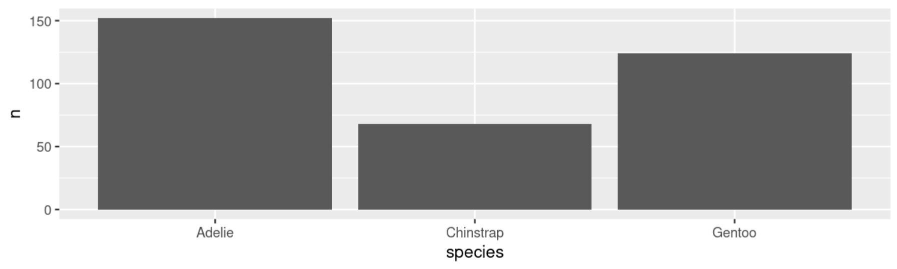


22 / 70

Geoms: Barplots

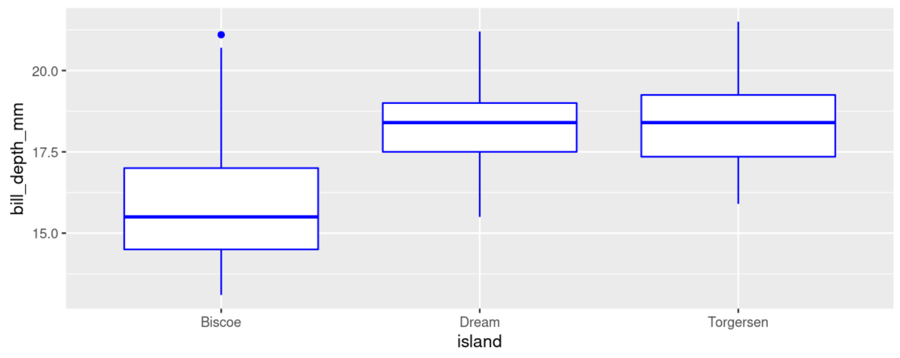
You can also provide the counts

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

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



Your Turn: Create this plot

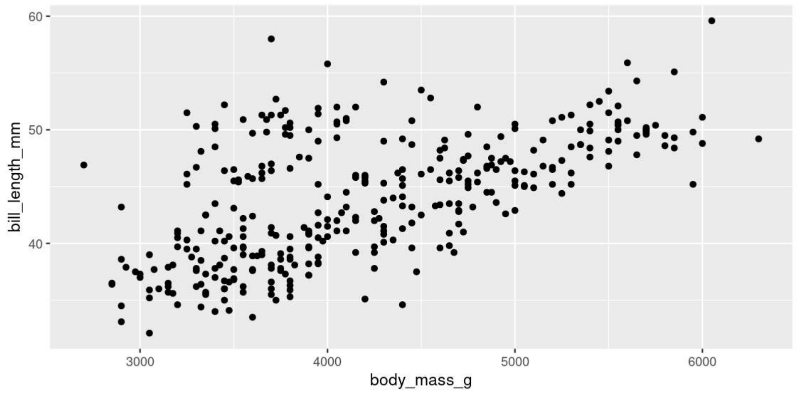
```r
library(ggplot2)

ggplot(data =        , aes(x =      , y =             )) +
  geom_       (               )
```



Showing data by group

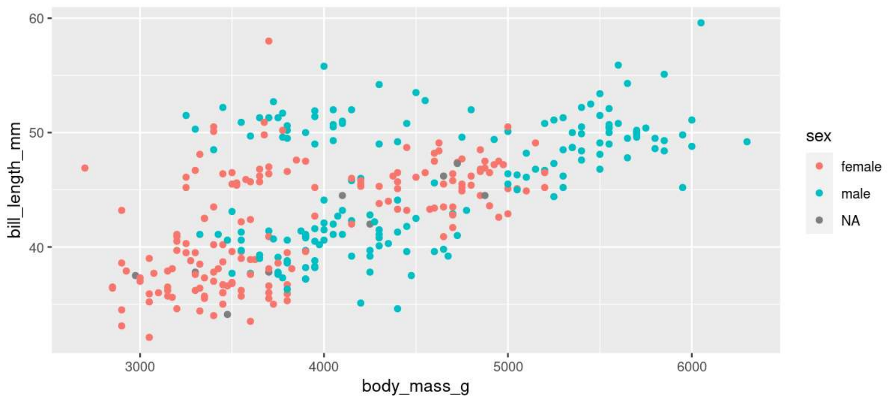
Mapping aesthetics

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

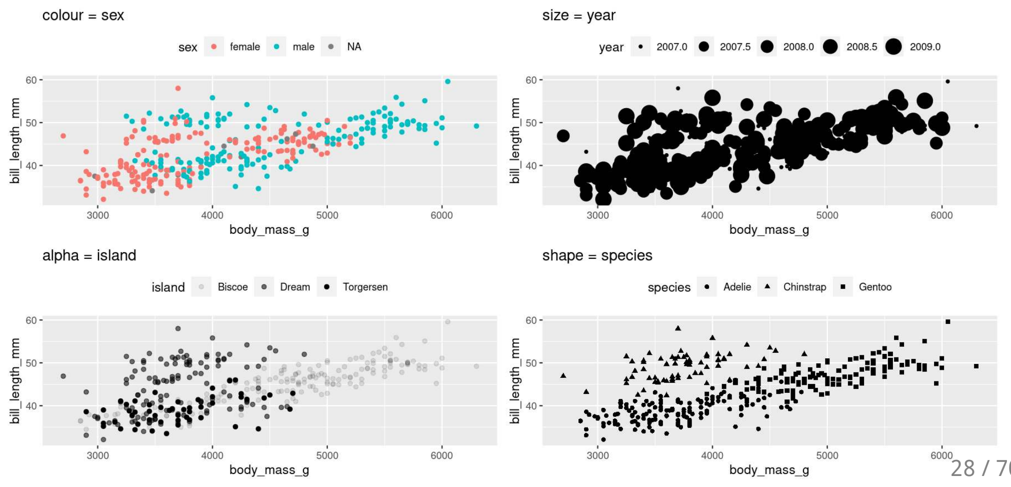


Mapping aesthetics

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



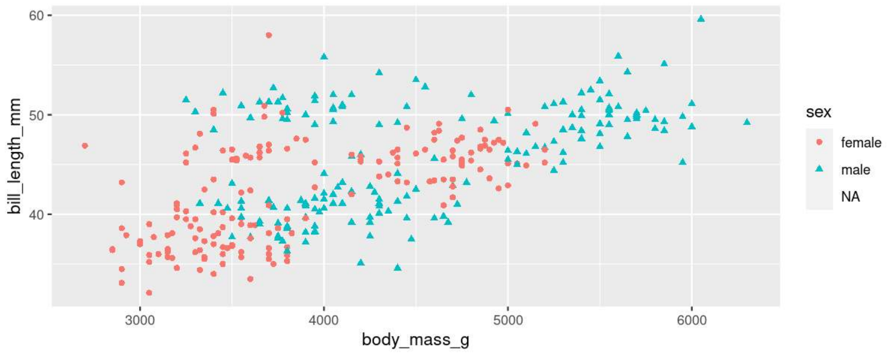
Mapping aesthetics



Mapping aesthetics

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

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

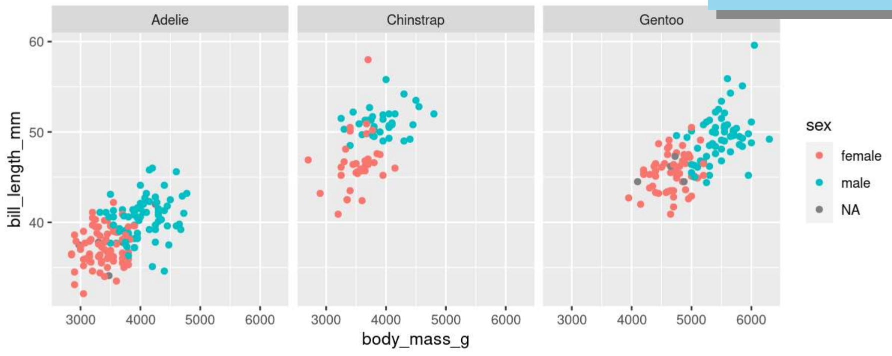


29 / 70

Faceting: `facet_wrap()`

```
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

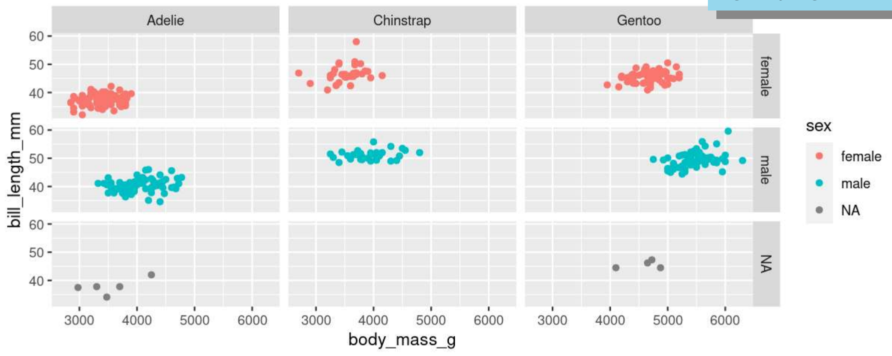


30 / 70

Faceting: `facet_grid()`

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

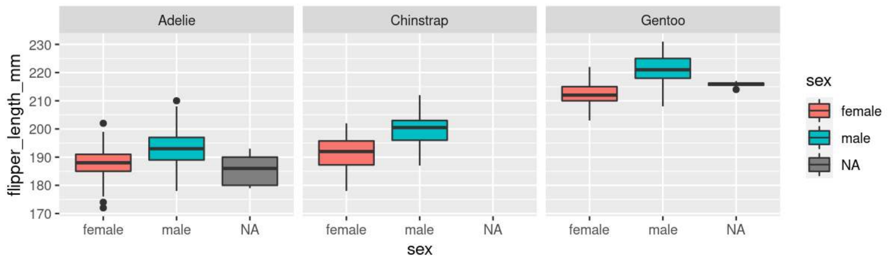
Split plots by **two** grouping variables



31 / 70

Your Turn: Create this plot

```
ggplot(data =          , aes(                                        )) +
                  +

```



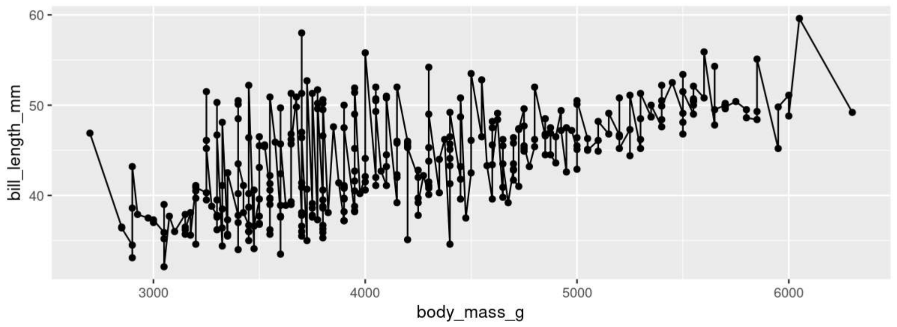
Hint: **`colour`** is for outlining with a colour, **`fill`** is for 'filling' with a colour

Trendlines / Regression Lines

Trendlines / Regression lines

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

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

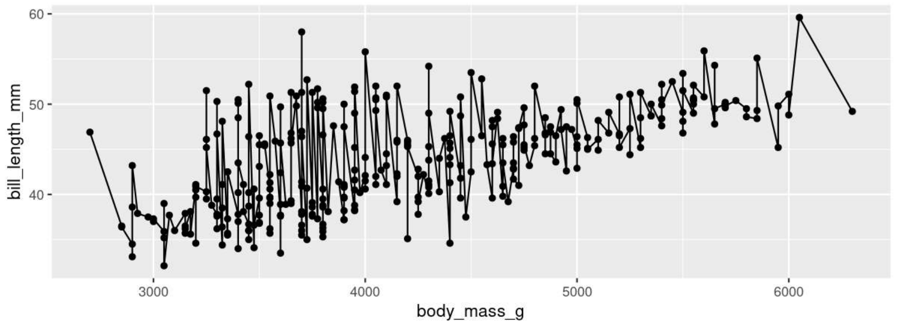


Trendlines / Regression lines

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

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

Not what we're looking for

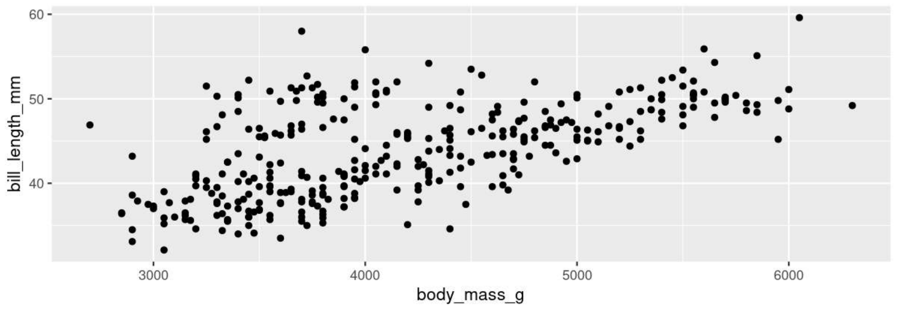


Trendlines / Regression lines

Let's add a trend line properly

Start with basic plot:

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

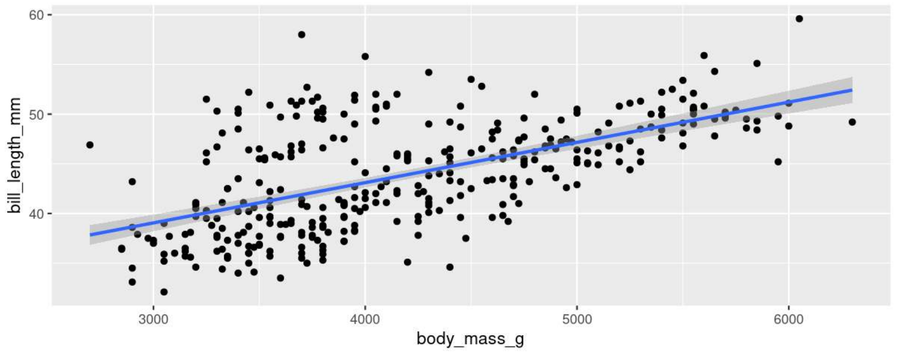


Trendlines / Regression lines

Add the `stat_smooth()`

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

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

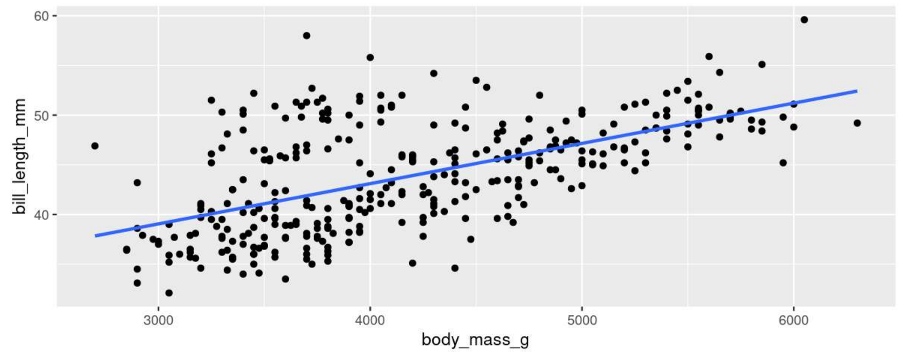


Trendlines / Regression lines

Add the stat_smooth()

- remove the grey ribbon **se = FALSE**

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

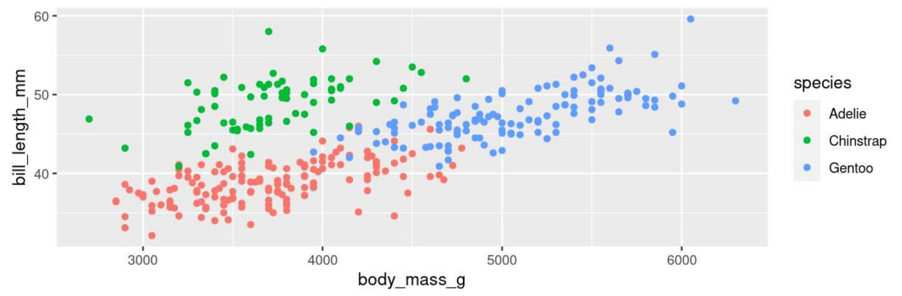


Trendlines / Regression lines

A line for each group

- Specify group (here we use **colour** to specify **sex**)

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

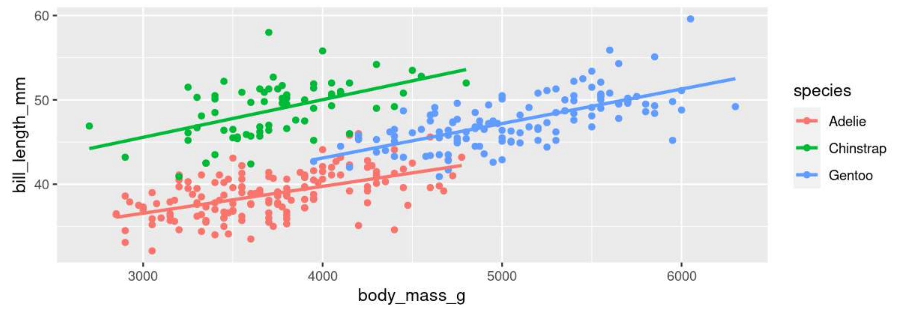


Using stats: Trendlines / Regression lines

A line for each group

- **stat_smooth()** automatically uses the same grouping

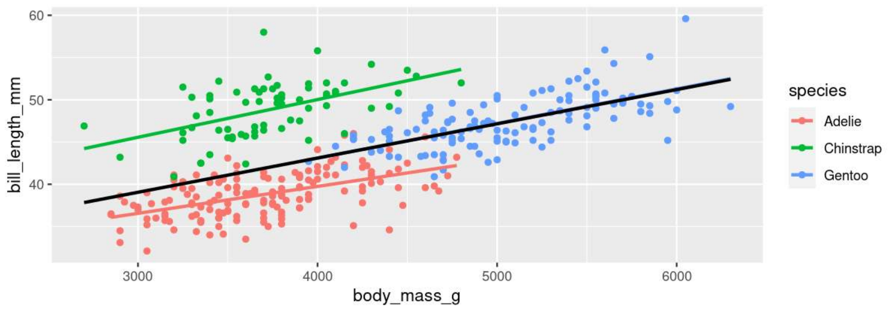
```
g + stat_smooth(method = "lm", se = FALSE)
```



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")
```



40 / 70

Your Turn: Create this plot

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

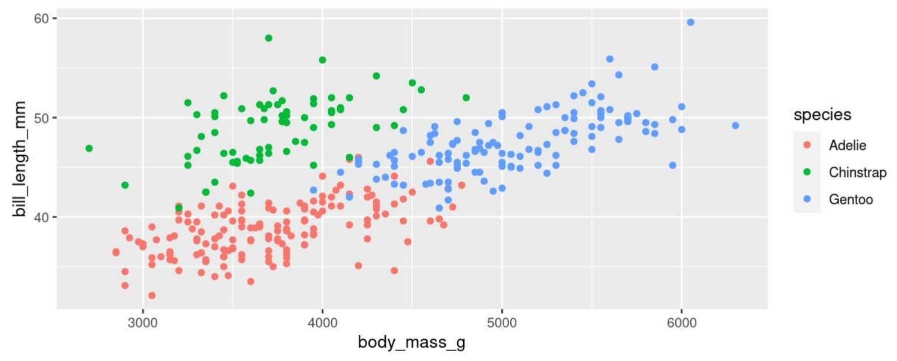
41 / 70

Customizing plots

Customizing: Starting plot

Let's work with this plot

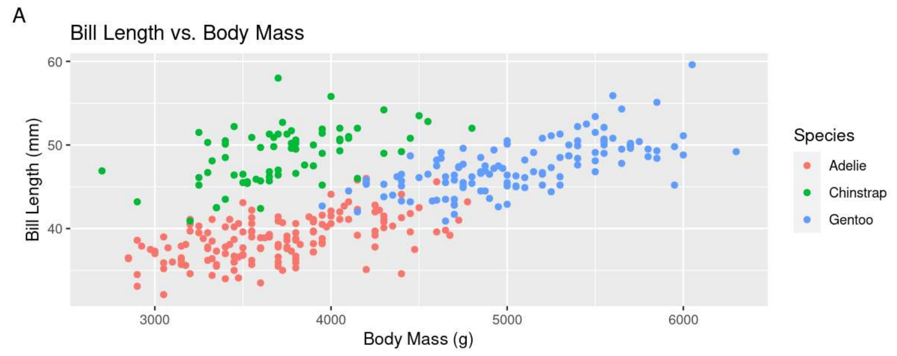
```r
g <- ggplot(data = penguins, aes(x = body_mass_g, y = bill_length_mm, colour = species)) +
  geom_point()
```



43 / 70

Customizing: Labels

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

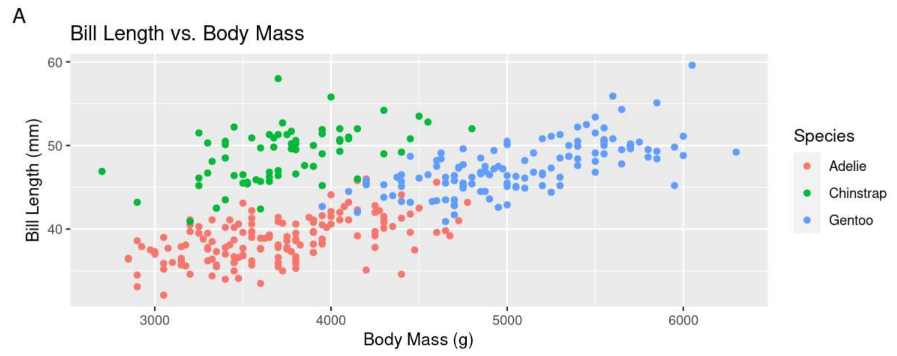


44 / 70

Customizing: Labels

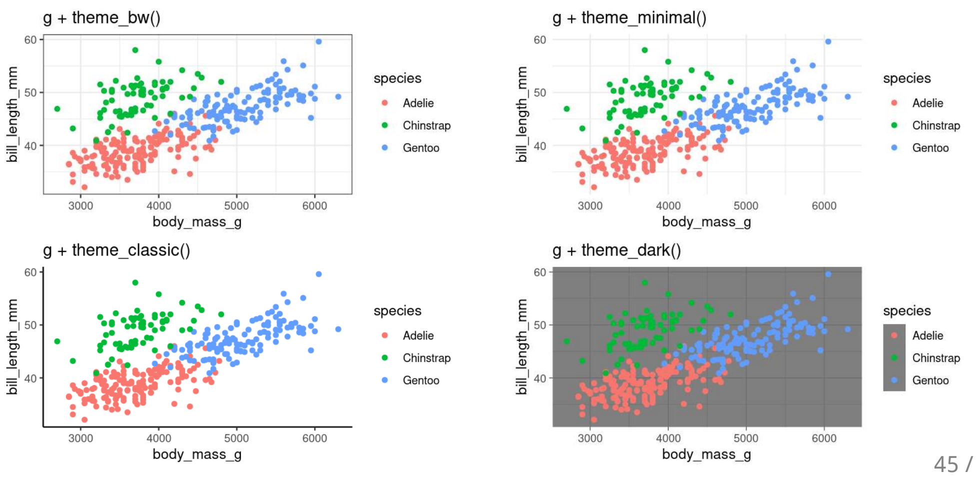
```r
g + labs(title = "Bill Length vs. Body Mass",
         x = "Body Mass (g)",
         y = "Bill Length (mm)",
         colour = "Species", tag = "A")
```

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



44 / 70

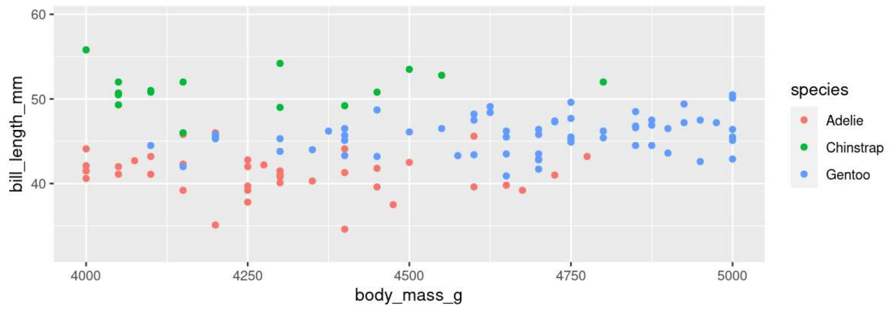
Customizing: Built-in themes



Customizing: Data range

Limit the data (exclude data)

```
g + xlim(c(4000, 5000))
```



```
## Warning: Removed 228 rows containing missing values (geom_point).
```

Customizing: Axes

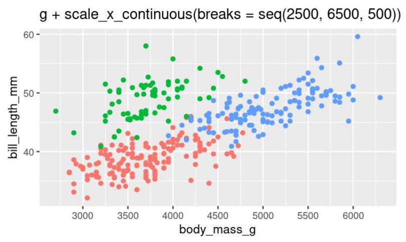
`scale_` + (**`x`** or **`y`**) + type (**`contiuous`**, **`discrete`**, **`date`**, **`datetime`**)

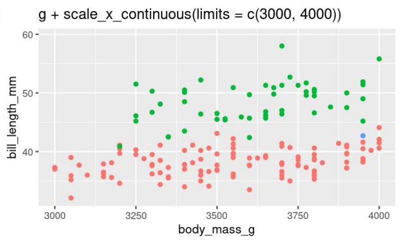
- **`scale_x_continuous()`**
- **`scale_y_discrete()`**
- etc.

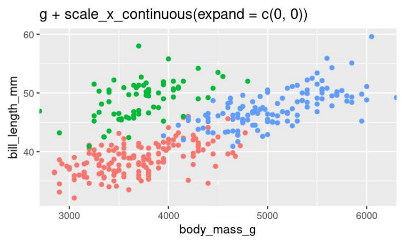
Common arguments

```
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
```

Customizing: Axes





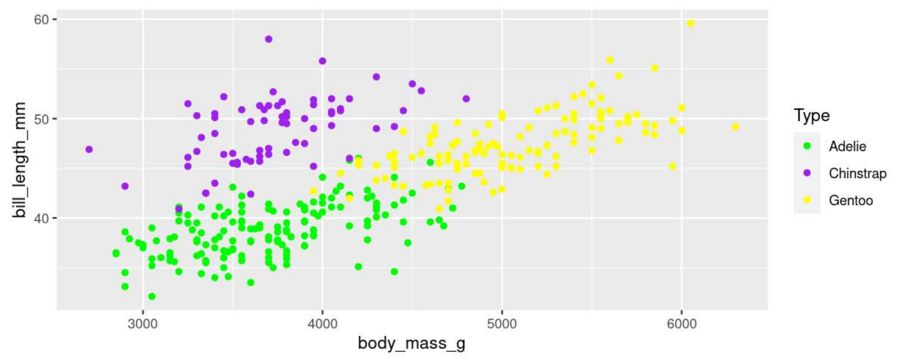


Customizing: Aesthetics

Using scales

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

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

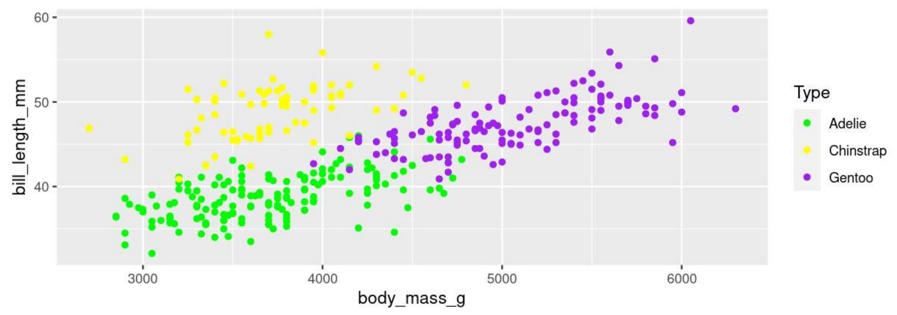


Customizing: Aesthetics

Using scales

Or be very explicit:

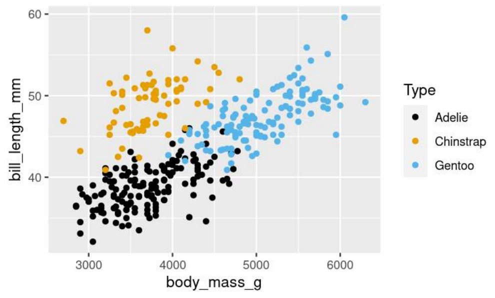
```
g + scale_colour_manual(name = "Type",
                        values = c("Adelie" = "green", "Gentoo" = "purple", "Chinstrap" = "yellow"),
                        na.value = "black")
```

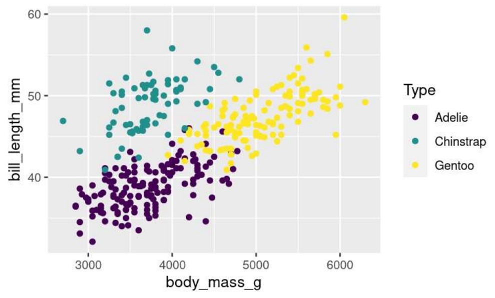


Customizing: Aesthetics

For colours, consider colour-blind-friendly scales

```
library(ggthemes)
g + scale_colour_colorblind(name = "Type")
g + scale_colour_viridis_d(name = "Type")
```





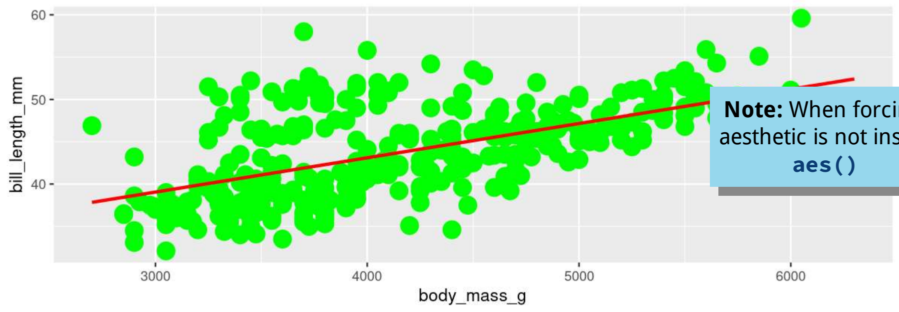
51 / 70

Customizing: Aesthetics

Forcing

Remove the association between a variable and an aesthetic

```
ggplot(data = penguins, aes(x = body_mass_g, y = bill_length_mm, colour = sex)) +
  geom_point(colour = "green", size = 5) +
  stat_smooth(method = "lm", se = FALSE, colour = "red")
```



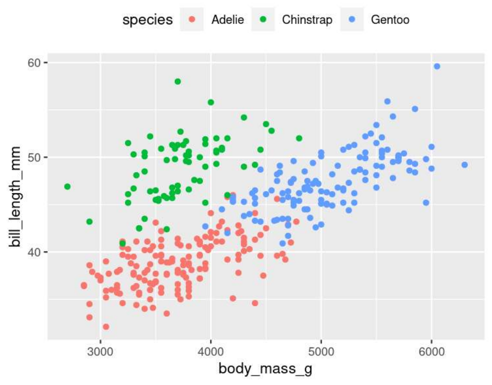
Note: When forcing, aesthetic is not inside `aes()`

52 / 70

Customizing: Legends placement

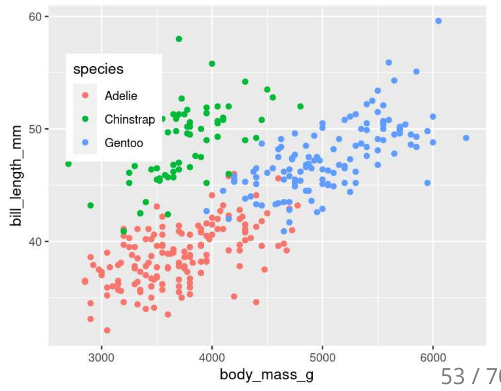
At the: top, bottom, left, right

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



Exactly here

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



53 / 70

Saving plots

Saving plots

RStudio Export

Demo

Saving plots

RStudio Export

Demo

`ggsave()`

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

ggsave(filename = "penguins_mass.png", plot = g)
```

```
## Saving 8 x 3.6 in image
```

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 = "penguin_mass.pdf", plot = g, dpi = 300,
       height = 80, width = 129, units = "mm")
```

Loading Data

Data types: What kind of data do you have?

Specific program files

Type	R Package	Function (example usage)
Excel (.xls, .xlsx)	`readxl`	`read_excel("file.xlsx", sheet = 1)`
Comma separated (.csv)	`readr`	`read_csv("file.csv")`
Tab separated (e.g, .txt, .dat)	`readr`	`read_tsv("file.txt")`
Space separated (e.g, .txt, .dat)	`readr`	`read_delim("file.dat", delim = " ")`
Fixed-width (e.g, .txt, .dat)	`readr`	`read_fwf("file.dat")`

Data types: What kind of data do you have?

Specific program files

Type	R Package	Function (example usage)
Excel (.xls, .xlsx)	`readxl`	`read_excel("file.xlsx", sheet = 1)`
Comma separated (.csv)	`readr`	`read_csv("file.csv")`
Tab separated (e.g, .txt, .dat)	`readr`	`read_tsv("file.txt")`
Space separated (e.g, .txt, .dat)	`readr`	`read_delim("file.dat", delim = " ")`
Fixed-width (e.g, .txt, .dat)	`readr`	`read_fwf("file.dat")`

Notes

1. You may be familiar with base functions (i.e. `read.csv()`, `read.table()`)
   These are perfectly acceptable, but `readr` is a bit more powerful and quick
2. It can be quicker and safer to save Excel files as a *.csv (Comma-separated-variables file) and then use `readr` package and `read_csv()` function
3. `readr` is a tidyverse package

Where is my data?

```
library(tidyverse) # Load tidyverse which includes readr package

my_data <- read_csv("weather.csv")
```

```
## Error: 'weather.csv' does not exist in current working directory
('/home/steffi/Projects/Teaching/UofM - NRI/2020/Site/_labs').
```

With no folder (just file name) R expects file to be in **Working directory**

Where is my data?

```
library(tidyverse) # Load tidyverse which includes readr package

my_data <- read_csv("weather.csv")
```

```
## Error: 'weather.csv' does not exist in current working directory
('/home/steffi/Projects/Teaching/UofM - NRI/2020/Site/_labs').
```

With no folder (just file name) R expects file to be in **Working directory**

Working directory is:

- Where your RStudio project is
- Your home directory (My Documents, etc.) [If not using RStudio Projects]
- Where you've set it (using `setwd()` or RStudio's Session > Set Working Directory)

Where is my data?

```r
library(tidyverse) # Load tidyverse which includes readr package

my_data <- read_csv("weather.csv")
```

```
## Error: 'weather.csv' does not exist in current working directory
('/home/steffi/Projects/Teaching/UofM - NRI/2020/Site/_labs').
```

With no folder (just file name) R expects file to be in **Working directory**

Working directory is:

- Where your RStudio project is
- Your home directory (My Documents, etc.) [If not using RStudio Projects]
- Where you've set it (using `setwd()` or RStudio's Session > Set Working Directory)

> Using Projects in RStudio is a great idea

Where is my data?

Absolute Paths

OS	Absolute Path
LINUX	/home/steffi/Documents/R Projects/mydata.csv
WINDOWS	C:/Users/steffi/My Documents/R Projects/mydata.csv
MAC	/users/steffi/Documents/R Projects/mydata.csv

Relative Paths

Path	Where to look
./mydata.csv	Here (current directory) (./)
../mydata.csv	Go up one directory (../)
./data/mydata.csv	Stay here (./), go into "data" folder (data/)
../data/mydata.csv	Go up one directory (../), then into "data" folder (data/)

Where is my data?

Absolute Paths

OS	Absolute Path
LINUX	/home/steffi/Documents/R Projects/mydata.csv
WINDOWS	C:/Users/steffi/My Documents/R Projects/mydata.csv
MAC	/users/steffi/Documents/R Projects/mydata.csv

With RStudio 'Projects' only need to use **relative** paths

Relative Paths

Path	Where to look
./mydata.csv	Here (current directory) (./)
../mydata.csv	Go up one directory (../)
./data/mydata.csv	Stay here (./), go into "data" folder (data/)
../data/mydata.csv	Go up one directory (../), then into "data" folder (data/)

Keep yourself organized

- Create an RStudio Project for each Project
- Create a specific "Data" folder within each project (one per project)

Folders look like:

```
- My Project
    - Data
        - mydata1.csv
        - mydata2.csv
    - myscript.R
    - My Project.Rproj
```

Now when you load data, you can use something like this: **"./Data/mydata1.csv"**

Checking / Cleaning your data

```
library(readxl)

my_data <- read_excel("my_data.xlsx")
```

```
head(my_data)
```

```
## # A tibble: 6 x 6
##   `Sample Number` Stage          `Date Egg`          `Body Mass (g)` ...5  ...6
##             <dbl> <chr>          <dttm>                        <dbl> <lgl> <lgl>
## 1               1 Adult, 1 Egg … 2007-11-11 00:00:00            3750 NA    NA
## 2               2 Adult, 1 Egg … 2007-11-11 00:00:00            3800 NA    NA
## 3               3 Adult, 1 Egg … 2007-11-16 00:00:00            3250 NA    NA
## 4               4 Adult, 1 Egg … 2007-11-16 00:00:00              NA NA    NA
## 5               5 Adult, 1 Egg … 2007-11-16 00:00:00            3450 NA    NA
## 6               6 Adult, 1 Egg … 2007-11-16 00:00:00            3650 NA    NA
```

Checking / Cleaning your data

```
tail(my_data)
```

```
## # A tibble: 6 x 6
##   `Sample Number` Stage          `Date Egg`          `Body Mass (g)` ...5  ...6
##             <dbl> <chr>          <dttm>                        <dbl> <lgl> <lgl>
## 1              63 Adult, 1 Egg … 2009-11-19 00:00:00            3650 NA    NA
## 2              64 Adult, 1 Egg … 2009-11-19 00:00:00            4000 NA    NA
## 3              65 Adult, 1 Egg … 2009-11-21 00:00:00            3400 NA    NA
## 4              66 Adult, 1 Egg … 2009-11-21 00:00:00            3775 NA    NA
## 5              67 Adult, 1 Egg … 2009-11-21 00:00:00            4100 NA    NA
## 6              68 Adult, 1 Egg … 2009-11-21 00:00:00            3775 NA    NA
```

Checking / Cleaning your data

```
tail(my_data)
```

```
## # A tibble: 6 x 6
##   `Sample Number` Stage          `Date Egg`          `Body Mass (g)` ...5  ...6
##             <dbl> <chr>          <dttm>                        <dbl> <lgl> <lgl>
## 1              63 Adult, 1 Egg … 2009-11-19 00:00:00            3650 NA    NA
## 2              64 Adult, 1 Egg … 2009-11-19 00:00:00            4000 NA    NA
## 3              65 Adult, 1 Egg … 2009-11-21 00:00:00            3400 NA    NA
## 4              66 Adult, 1 Egg … 2009-11-21 00:00:00            3775 NA    NA
## 5              67 Adult, 1 Egg … 2009-11-21 00:00:00            4100 NA    NA
## 6              68 Adult, 1 Egg … 2009-11-21 00:00:00            3775 NA    NA
```

- Looks like we have some extra, empty, columns... (`..5`, `..6`)
- Also looks like some column names might not work well in R
  - (Anything with a space or special character, i.e. `Date Egg` and `Body Mass (g)`

Checking / Cleaning your data

- When loading data that was in Excel (etc.) it can often have some funky things going on
- Use the `janitor` package to quickly fix some of those problems (`library(janitor)`)

Column names

```
my_data <- clean_names(my_data)
head(my_data)
```

```
## # A tibble: 6 x 6
##   sample_number stage              date_egg            body_mass_g x5    x6
##           <dbl> <chr>              <dttm>                    <dbl> <lgl> <lgl>
## 1             1 Adult, 1 Egg Stage 2007-11-11 00:00:00        3750 NA    NA
## 2             2 Adult, 1 Egg Stage 2007-11-11 00:00:00        3800 NA    NA
## 3             3 Adult, 1 Egg Stage 2007-11-16 00:00:00        3250 NA    NA
## 4             4 Adult, 1 Egg Stage 2007-11-16 00:00:00          NA NA    NA
## 5             5 Adult, 1 Egg Stage 2007-11-16 00:00:00        3450 NA    NA
## 6             6 Adult, 1 Egg Stage 2007-11-16 00:00:00        3650 NA    NA
```

Checking / Cleaning your data

- When loading data that was in Excel (etc.) it can often have some funky things going on
- Use the `janitor` package to quickly fix some of those problems (`library(janitor)`)

Empty rows/columns

```
my_data <- remove_empty(my_data, which = c("rows", "cols"))
```

```
head(my_data)
```

```
## # A tibble: 6 x 4
##   sample_number stage              date_egg            body_mass_g
##           <dbl> <chr>              <dttm>                    <dbl>
## 1             1 Adult, 1 Egg Stage 2007-11-11 00:00:00        3750
## 2             2 Adult, 1 Egg Stage 2007-11-11 00:00:00        3800
## 3             3 Adult, 1 Egg Stage 2007-11-16 00:00:00        3250
## 4             4 Adult, 1 Egg Stage 2007-11-16 00:00:00          NA
## 5             5 Adult, 1 Egg Stage 2007-11-16 00:00:00        3450
## 6             6 Adult, 1 Egg Stage 2007-11-16 00:00:00        3650
```

Loading your data

This blazing fast intro to loading/cleaning will not cover the many, *many*, **many**, **MANY** ways that data can be weird.

Let me know if (when) you run into problems and we can trouble shoot together!

Your Turn! / Homework

- Create a **new RStudio Project** for your class project
- Create a **"Data" folder** inside this project folder
  - Files pane > New Folder
- **Add data** to it (if you have data)
  - Use your computers folder navigator for this
- **Create a new script** in the main folder
  - [Menu] File > New > R Script
- Add code to this script to **load your data into R**
  - Load the appropriate packages (`tidyverse`, `readxl`)
  - Use the appropriate function given your data type (e.g., `read_csv()` for .csv, `read_excel` for .xlsx)
  - Use the appropriate file location (e.g, `"./Data/my_data.csv"`)
    Remember quotes (" ") around the *entire* file location
- **Explore your data** - Click on your data in the Environment pane and take a look!

Your Turn! / Homework

- Create a **new RStudio Project** for your class project
- Create a **"Data" folder** inside this project folder
  - Files pane > New Folder
- **Add data** to it (if you have data)
  - Use your computers folder navigator for this
- **Create a new script** in the main folder
  - [Menu] File > New > R Script
- Add code to this script to **load your data into R**
  - Load the appropriate packages (`tidyverse`, `readxl`)
  - Use the appropriate function given your data type (e.g., `read_csv()` for .csv, `read_excel` for .xlsx)
  - Use the appropriate file location (e.g, `"./Data/my_data.csv"`)
    Remember quotes (" ") around the *entire* file location
- **Explore your data** - Click on your data in the Environment pane and take a look!

If we run out of time, we'll check in next week to see how this went

Wrapping Up!

Wrapping up: Common mistakes

Figures

- 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()`, i.e. `ggplot(data = penguins, aes(x = factor(year), y = body_mass_g))`

Loading data

- Not using RProjects which makes it hard to find your data
- Expecting your data to be something it's not
  - (open your data in a text editor or spreadsheet program to take a look)
- Using the wrong function (i.e you used `read_csv()` when you should have used `read_tsv()`)

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
- Cookbook for R - by Winston Chang
  - See also R Graphics Cookbook by Winston Chang
- R for Data Science
  - Data Visualization
  - Workflow and Projects
  - Data Import