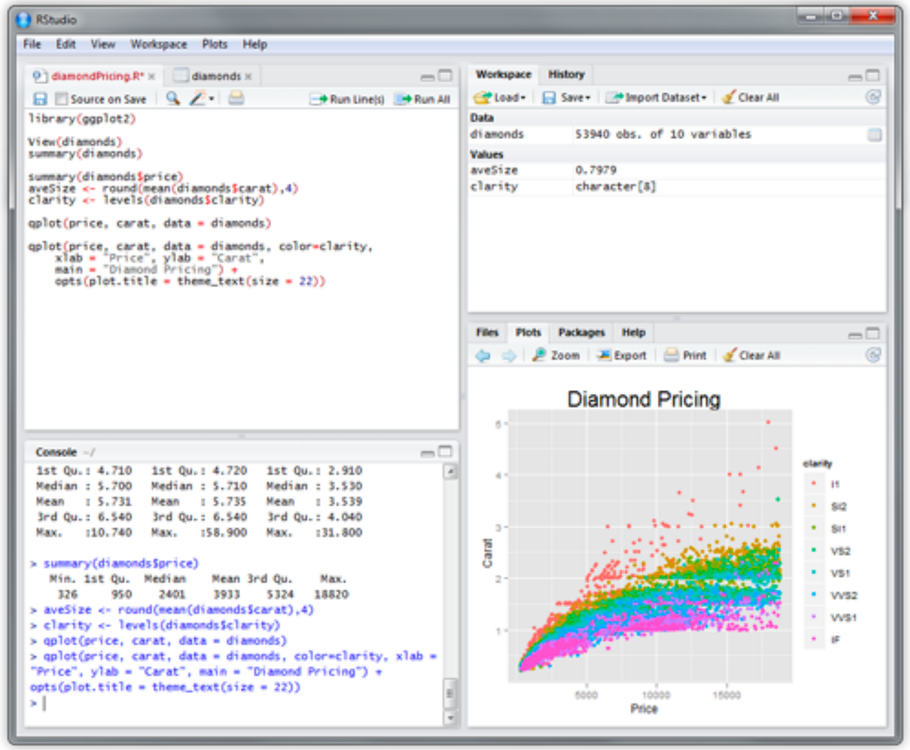
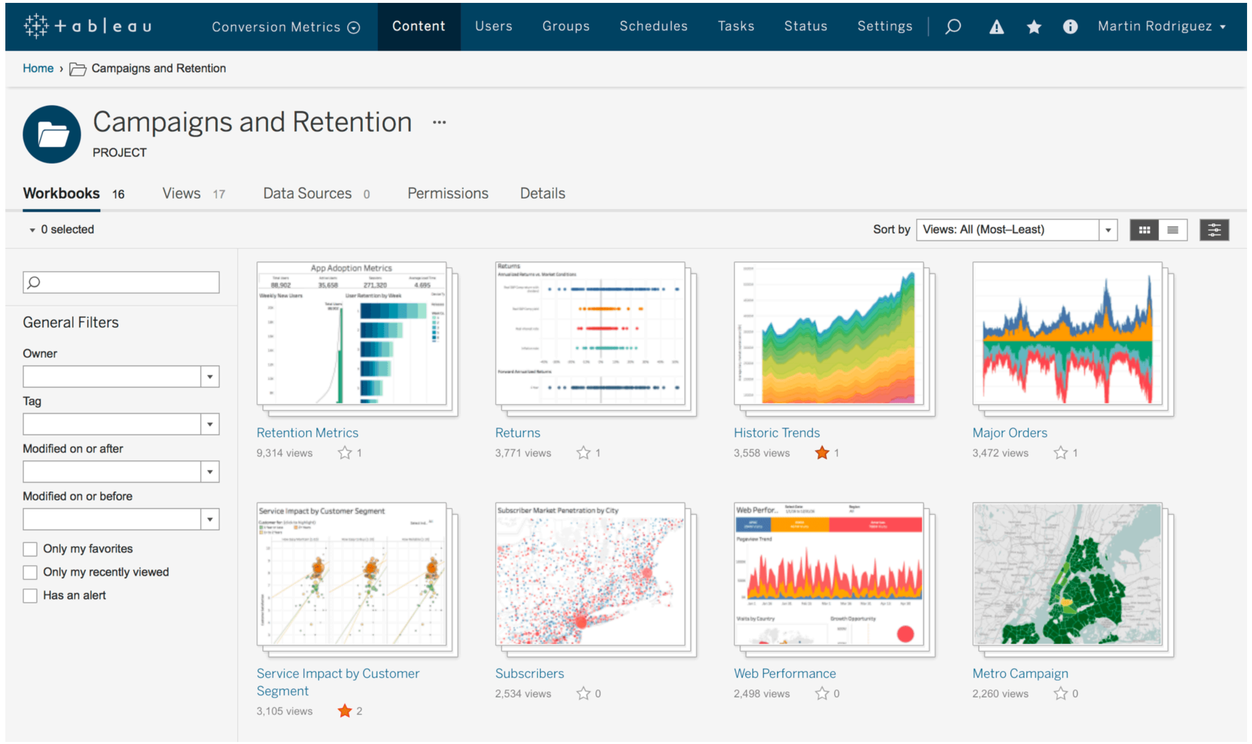
A tour of ggplot

STA313: Data Visualization

RStudio®

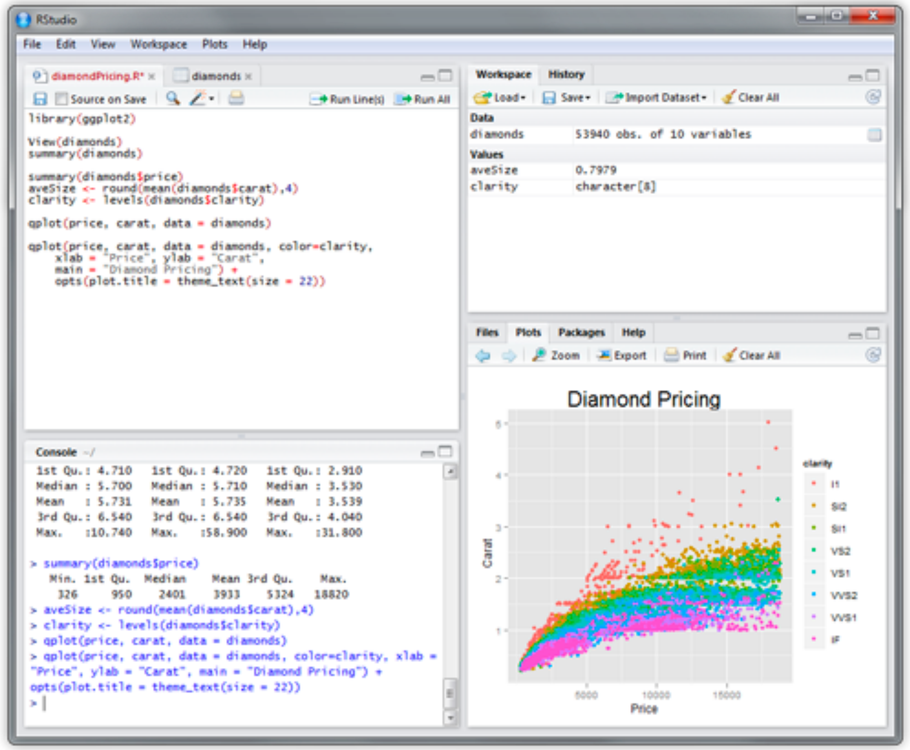
ggplot2 Shiny rmarkdown knitr





+ableau SOFTWARE

LAST WEEK

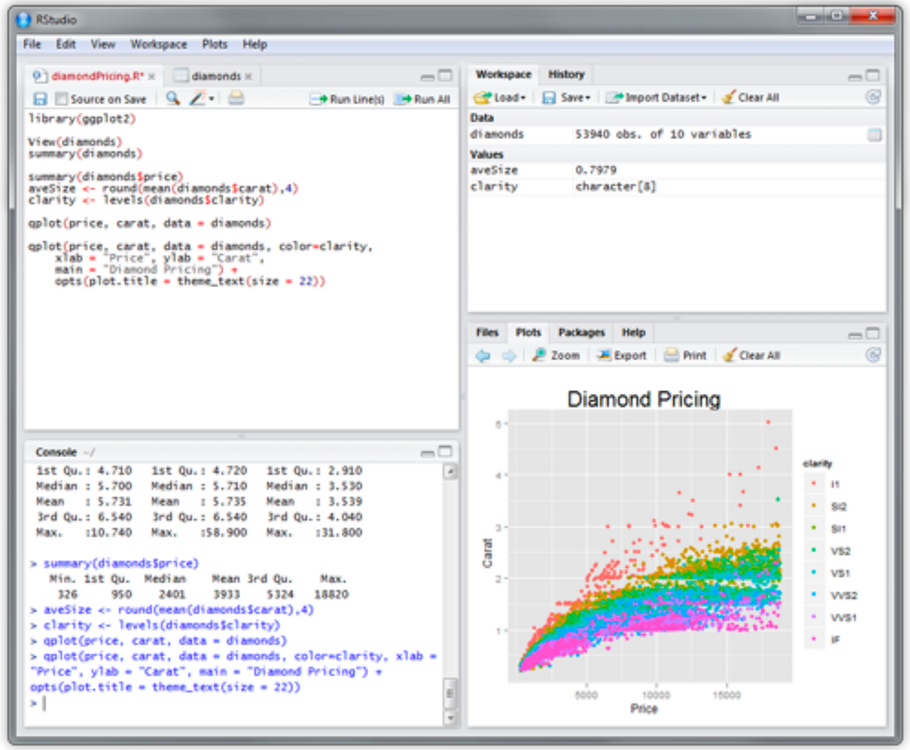

RStudio
File Edit View Workspace Plots Help
diamondPricing.R* diamonds
Source on Save Run Line(s) Run All
library(ggplot2)
View(diamonds)
summary(diamonds)
summary(diamonds$price)
aveSize <- round(mean(diamonds$carat),4)
clarity <- levels(diamonds$clarity)
qplot(price, carat, data = diamonds)
qplot(price, carat, data = diamonds, color=clarity,
xlab = "Price", ylab = "Carat",
main = "Diamond Pricing") +
opts(plot.title = theme_text(size = 22))
Workspace History
Load Save Import Dataset Clear All
Data
diamonds 53940 obs. of 10 variables
Values
aveSize 0.7979
clarity character[8]
Files Plots Packages Help
Zoom Export Print Clear All
Diamond Pricing
Carat
Price
clarity
I1
SI2
SI1
VS2
VS1
VVS2
VVS1
IF
Console ~/
1st Qu.: 4.710 1st Qu.: 4.720 1st Qu.: 2.910
Median : 5.700 Median : 5.710 Median : 3.530
Mean : 5.731 Mean : 5.735 Mean : 3.539
3rd Qu.: 6.540 3rd Qu.: 6.540 3rd Qu.: 4.040
Max. :10.740 Max. :58.900 Max. :31.800
> summary(diamonds$price)
Min. 1st Qu. Median Mean 3rd Qu. Max.
326 950 2401 3933 5324 18820
> aveSize <- round(mean(diamonds$carat),4)
> clarity <- levels(diamonds$clarity)
> qplot(price, carat, data = diamonds)
> qplot(price, carat, data = diamonds, color=clarity, xlab =
"Price", ylab = "Carat", main = "Diamond Pricing") +
opts(plot.title = theme_text(size = 22))

Conversion Metrics
Content
Users
Groups
Schedules
Tasks
Status
Settings
Martin Rodriguez
Campaigns and Retention
PROJECT
Workbooks 16
Views 17
Data Sources 0
Permissions
Details
General Filters
Owner
Tag
Modified on or after
Modified on or before
Only my favorites
Only my recently viewed
Has an alert
Retention Metrics
Returns
Historic Trends
Major Orders
Service Impact by Customer Segment
Subscribers
Web Performance
Metro Campaign

RStudio®

ggplot2 Shiny rmarkdown knitr

TODAY



Before we start…

Acknowledgment

This document is adapted from:

Chapter 3. Data visualisation, R for Data Science, https://r4ds.had.co.nz/data-visualisation.html;

and

An Introduction to ggplot2, UC Business Analytics R Programming Guide, https://uc-r.github.io/ggplot_intro

Requirements

You will need the following to follow along the examples.

1. R and RStudio
2. `ggplot2` library
3. `palmerpenguins` data set

- In RStudio Console, type

```
install.packages("ggplot2")
```

- If you are using JupyterHub, the library is pre-installed.

- We will use the data set which comes with the `palmerpenguins` library.
- In RStudio Console, type

```
install.packages("palmerpenguins")
```

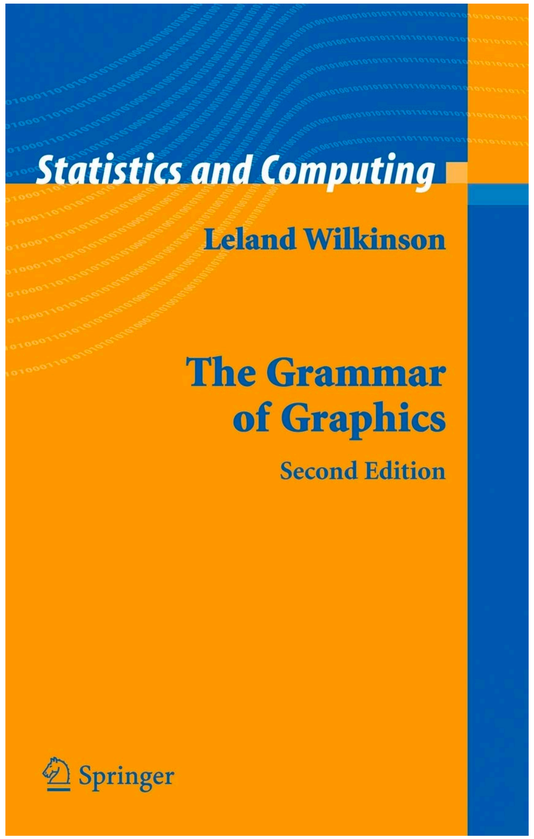
- For details, see the documentation by typing `?palmerpenguins` on the console.

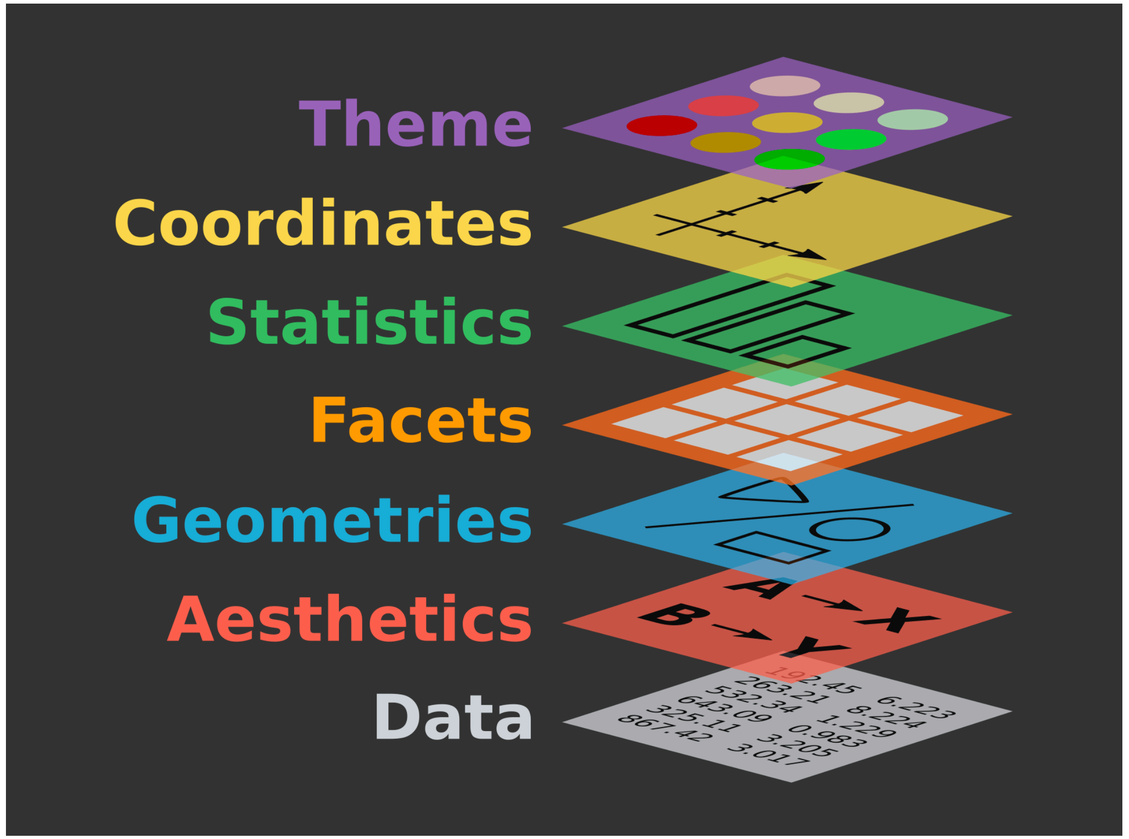
ggplot2

ggplot2

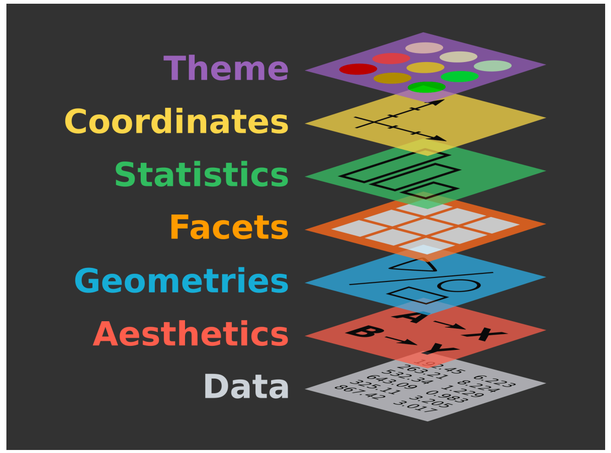
- **gg** in “ggplot2" stands for Grammar of Graphics
- Inspired by the book **Grammar of Graphics** by Leland Wilkinson

Grammar of Graphics





A **Grammar of Graphics** is a tool that enables us to concisely describe the components of a graph.



Grammar of Graphics

`ggplot2` implements the grammar of graphics to describe the components of a plot.

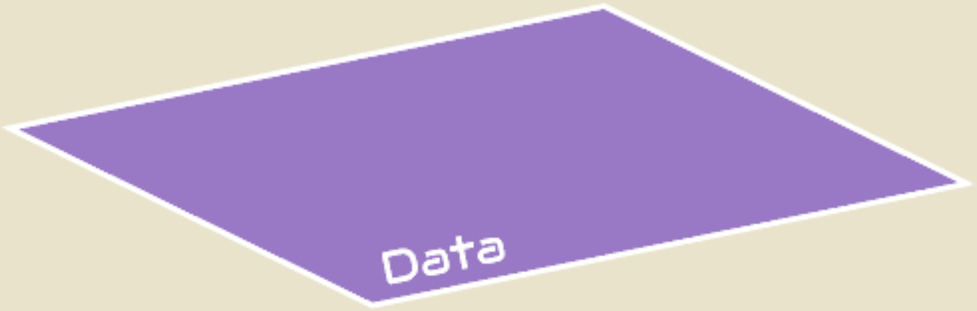
These include…

- the **DATA** being plotted
- the **AESTHETICS** of the geometric objects to which data variables are mapped
- the **GEOMETRIC OBJECTS** that appear on the plot
- the **FACETS** or groups of data displayed in different plots
- a **STATISTICAL TRANSFORMATION** used to calculate the data values used in the plot
- a **COORDINATE SYSTEM** used to organise the geometric objects
- a **POSITION ADJUSTMENT** for locating each geometric object on the plot
- a **SCALE** for each aesthetic mapping

Grammar of Graphics

```r
#-------------------------------- Visualising your boxplot ---------------

# Before plotting if not installed install.package("ggplot2")
# Then activate ggplot2 package
library(ggplot2)

# Create new variable for plot only x and y axis. ('data' and 'aestatics' layer)
plot <- ggplot(data=new.data, aes(x=Genre, y=Gross...US))

# Create new variable with geometries layer.
q <- plot + geom_jitter(aes(fill=Studio, size=Budget...mill.),
                        shape = 21, # this will shape a border around data points.
                        colour = "Black") + # with the border color of black.
  geom_boxplot(alpha=0.7, outlier.color = NA) # places the boxplot on the data points
                                              # and removes boxplot layer outliers.

# Change axis and title if needed.
q <- q +
  xlab("Genre") +
  ylab("Gross % US") +
  ggtitle("Domestic Gross % by Genre")

# Make your plot visually attractive and readable with the 'theme' function. (Theme layer)
q + theme(axis.title = element_text(colour = "Blue", size = 14),
          axis.text = element_text(size = 12),
          legend.title = element_text(size = 12),
          legend.text = element_text(size = 10),
          plot.title = element_text(size = 14, hjust = 0.5), # 'hjust' will center your text.
          panel.background = element_rect(fill = "#E6E3C5"))
```

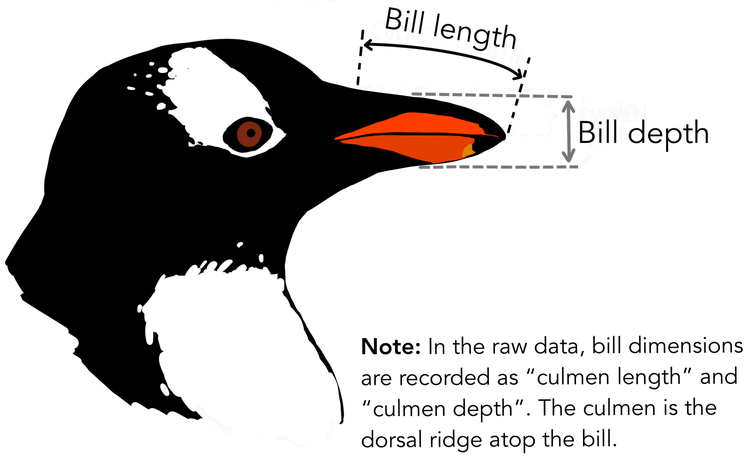
xy, 3902, 29, 9,
4756, x, 72, 633,
647, 617, 827, 3,
1, 21, 45, tyu, 6,
987, 457, 283, 8,
4, 5, 671, 34, 67,
x, 981, hu, 89, 5



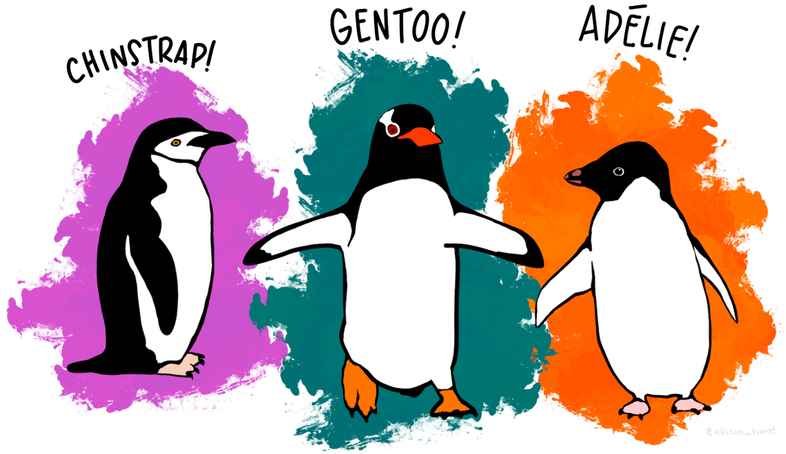
Source: https://medium.com/tdebeus/think-about-the-grammar-of-graphics-when-improving-your-graphs-18e3744d8d18

Live coding!

Palmer Penguins



Note: In the raw data, bill dimensions are recorded as "culmen length" and "culmen depth". The culmen is the dorsal ridge atop the bill.

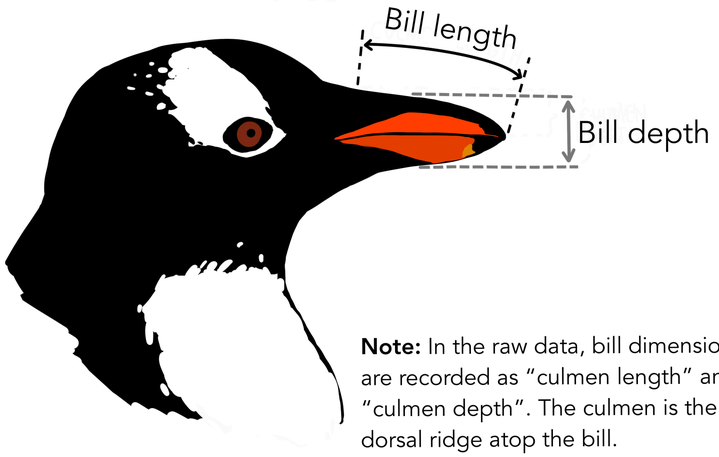


Measurements for penguins species, island in Palmer Archipelago, size (flipper length, body mass, bill dimensions), and sex

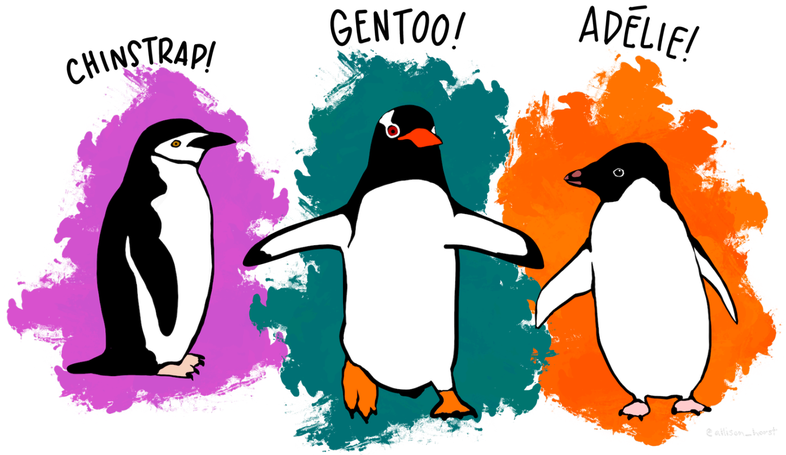
```
library(palmerpenguins)
?palmerpenguins
```

Source: https://allisonhorst.github.io/palmerpenguins/
Artwork by @allison_horst

Palmer Penguins



Note: In the raw data, bill dimensions are recorded as "culmen length" and "culmen depth". The culmen is the dorsal ridge atop the bill.

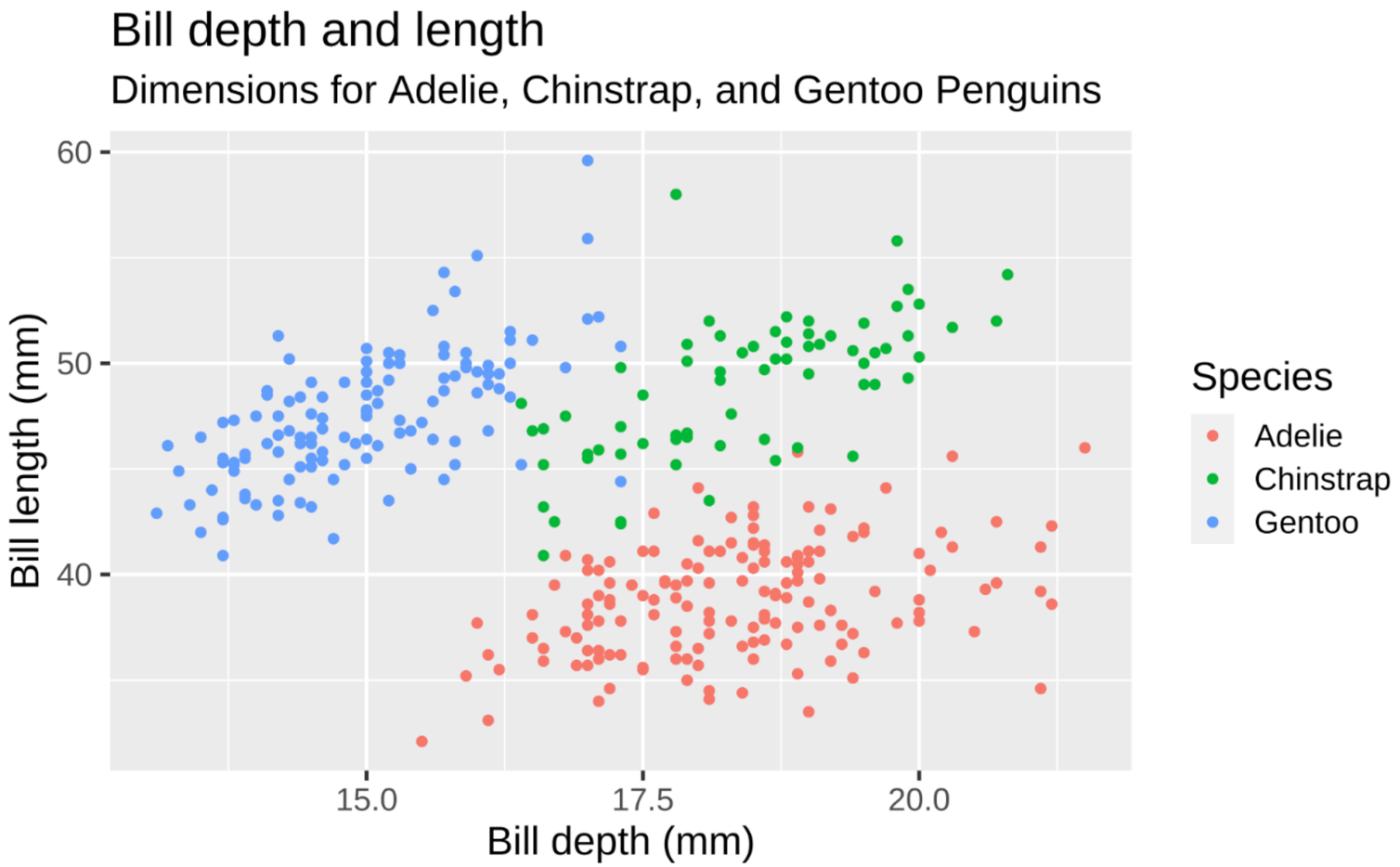


Measurements for penguins species, island in Palmer Archipelago, size (flipper length, body mass, bill dimensions), and sex

```
>  glimpse(penguins)
Rows: 344
Columns: 8
$ species           <fct> Adelie, Adelie, Adelie, Adelie, Ade…
$ island            <fct> Torgersen, Torgersen, Torgersen, To…
$ bill_length_mm    <dbl> 39.1, 39.5, 40.3, NA, 36.7, 39.3, 3…
$ bill_depth_mm     <dbl> 18.7, 17.4, 18.0, NA, 19.3, 20.6, 1…
$ flipper_length_mm <int> 181, 186, 195, NA, 193, 190, 181, 1…
$ body_mass_g       <int> 3750, 3800, 3250, NA, 3450, 3650, 3…
$ sex               <fct> male, female, female, NA, female, m…
$ year              <int> 2007, 2007, 2007, 2007, 2007, 2007,…
```

Source: https://allisonhorst.github.io/palmerpenguins/
Artwork by @allison_horst

Plot we will create



Code

```
ggplot(data = penguins,
       mapping = aes(x = bill_depth_mm, y = bill_length_mm,
                     colour = species)) +
  geom_point() +
  labs(title = "Bill depth and length",
       subtitle = "Dimensions for Adelie, Chinstrap, and Gentoo Penguins",
       x = "Bill depth (mm)", y = "Bill length (mm)",
       colour = "Species")
```

The Basic Template

Creating a plot with `ggplot2` takes the following basic template at minimum.

```
ggplot(data = [DATA],
       mapping = aes([MAPPINGS])) +
  [GEOMFUNCTION])
```

Provide the **DATA** to be plotted.

Specify the **GEOMETRIC OBJECT** with the corresponding function.

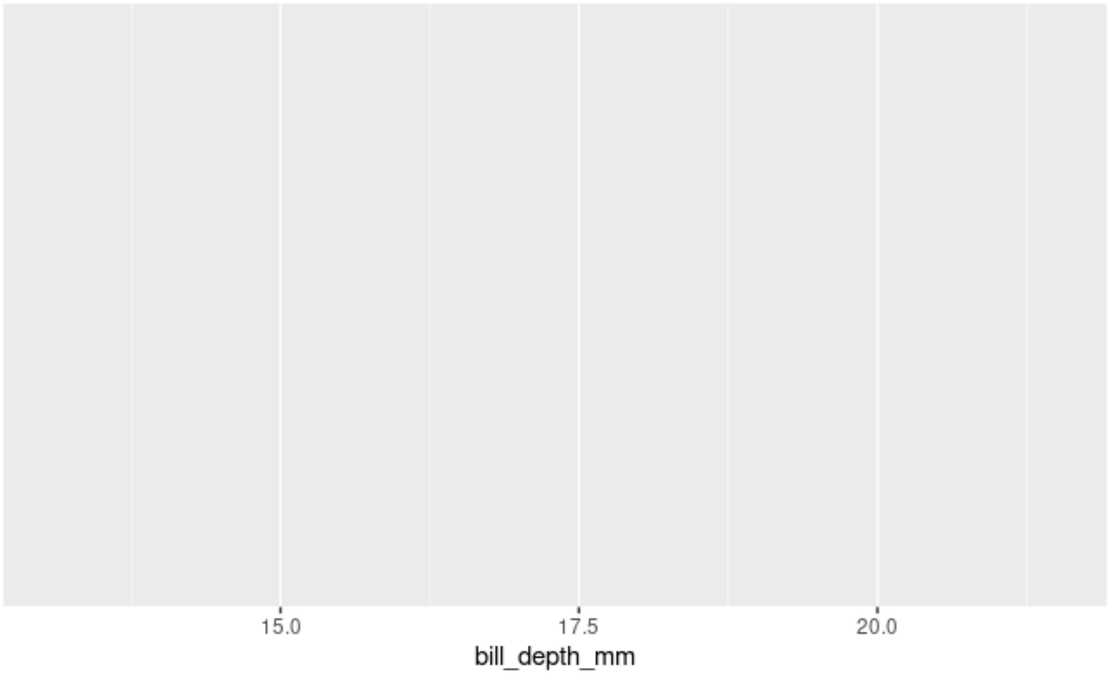
Map variables to **AESTHETICS**.

Start with the `penguins` data frame

```
ggplot(data = penguins)
```

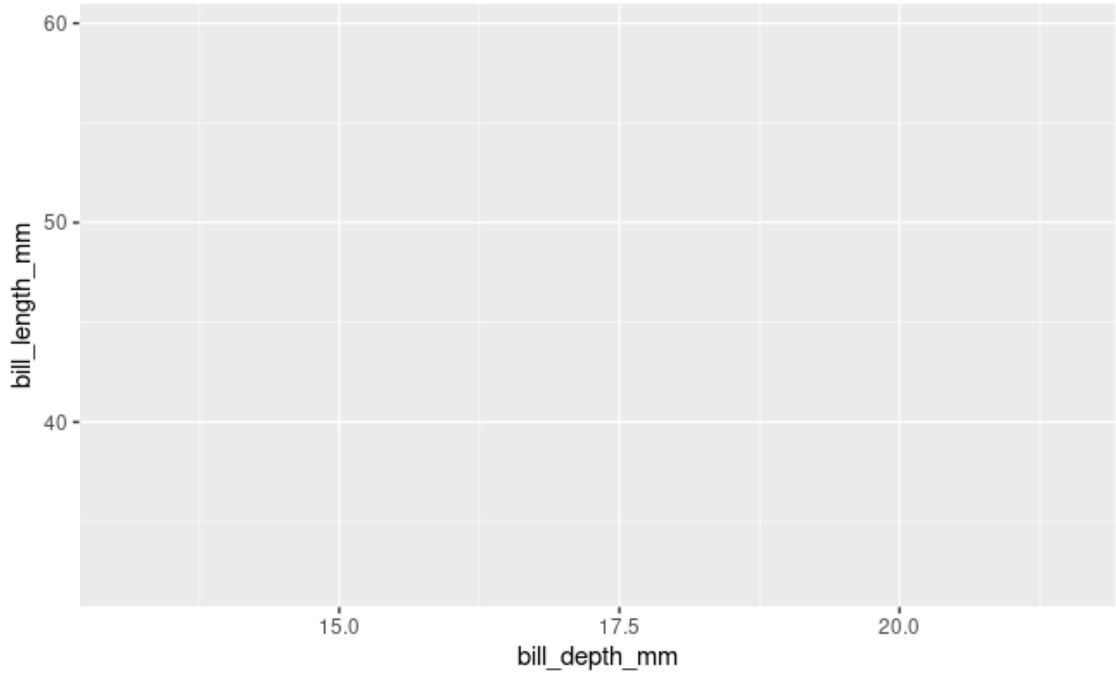
Start with the `penguins` data frame, map bill depth to the x-axis

```
ggplot(data = penguins,
       mapping = aes(x = bill_depth_mm))
```



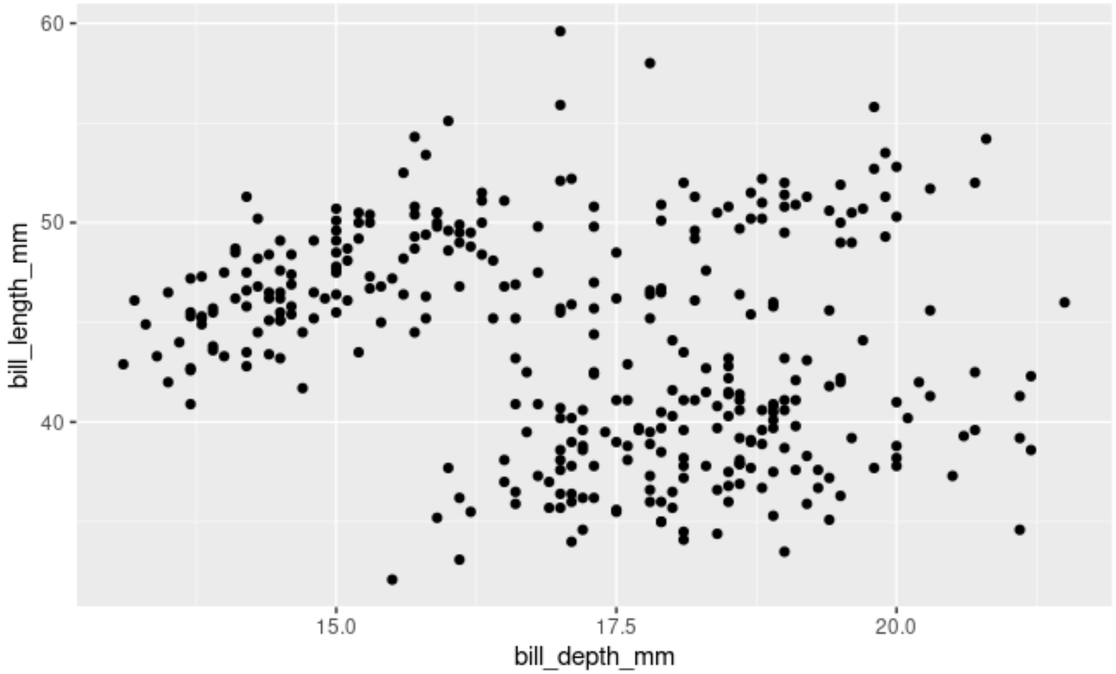
Start with the `penguins` data frame, map bill depth to the x-axis and map bill length to the y-axis

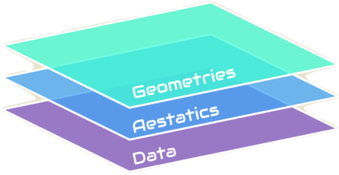
```
ggplot(data = penguins,
       mapping = aes(x = bill_depth_mm,
                     y = bill_length_mm))
```



Start with the `penguins` data frame, map bill depth to the x-axis and map bill length to the y-axis. Represent each observation with a point.

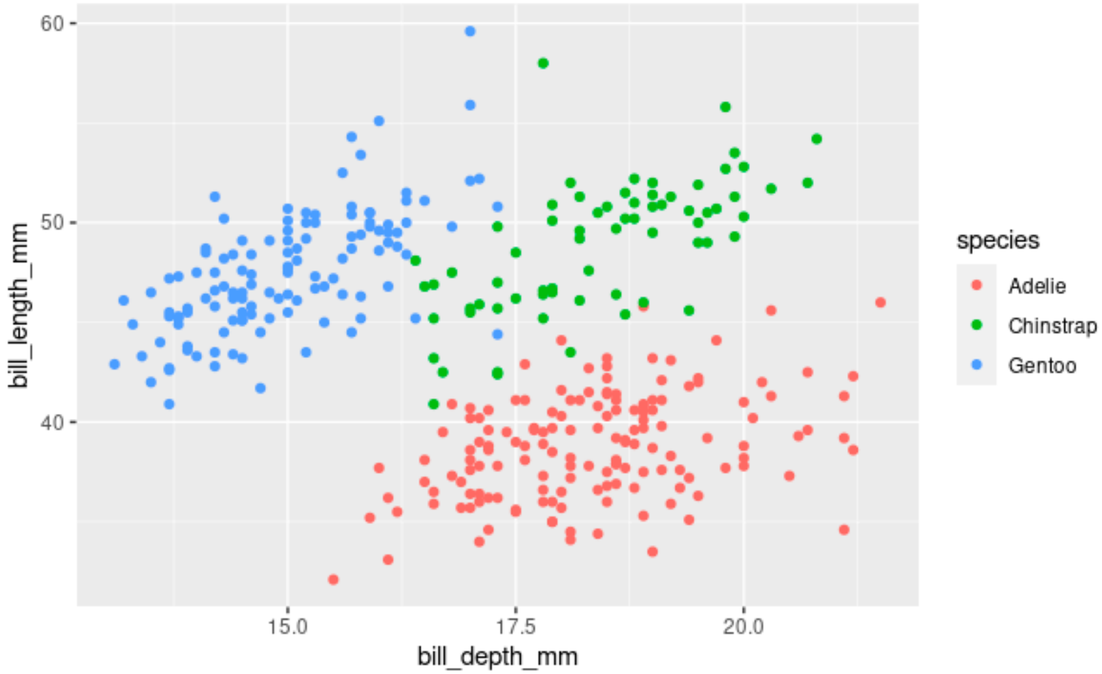
```
ggplot(data = penguins,
       mapping = aes(x = bill_depth_mm,
                     y = bill_length_mm)) +
         geom_point()
```



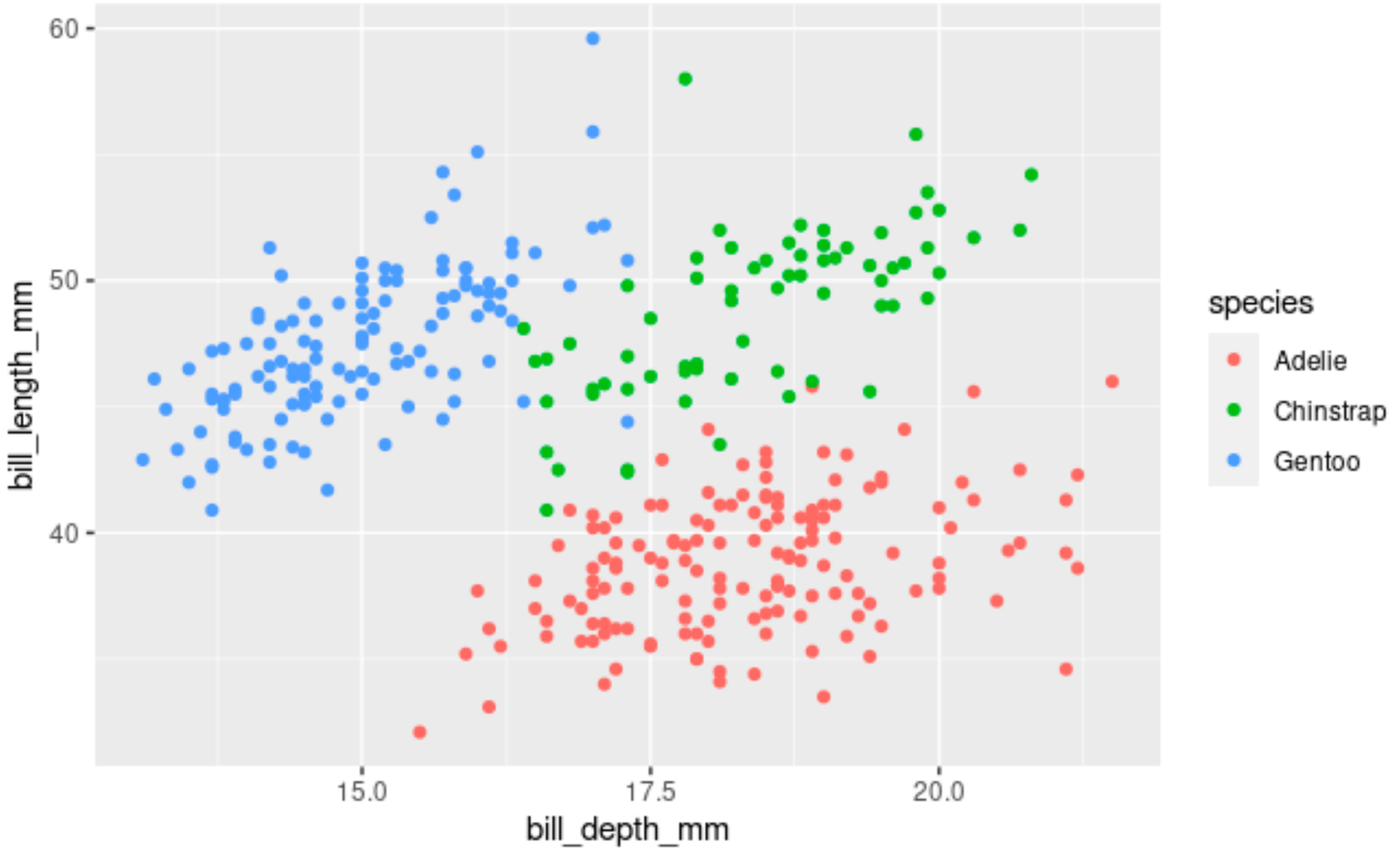


Start with the `penguins` data frame, map bill depth to the x-axis and map bill length to the y-axis. Represent each observation with a point and map species to the colour of each point.

```
ggplot(data = penguins,
       mapping = aes(x = bill_depth_mm,
                     y = bill_length_mm,
                     colour = species)) +
  geom_point()
```

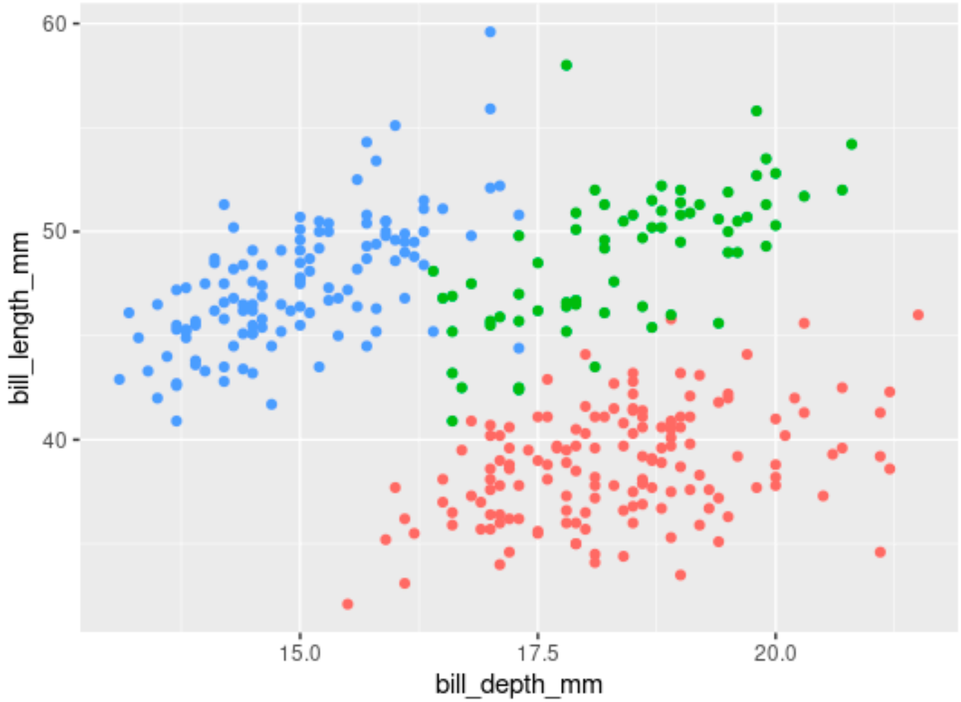


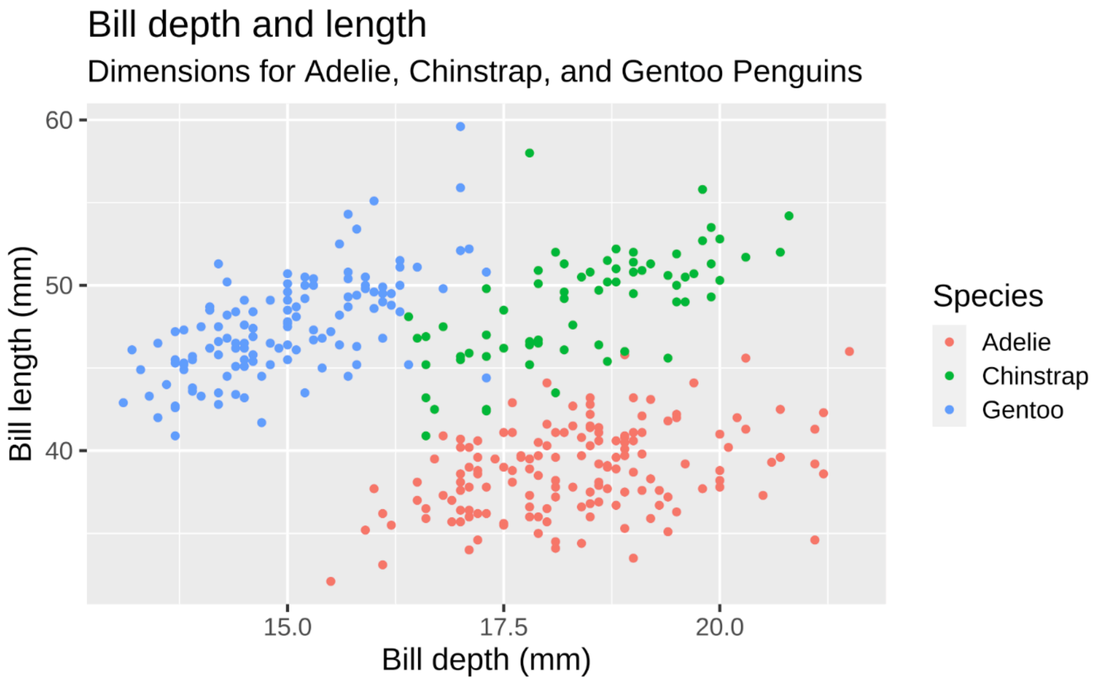
Our generated visualization has readability problems… Why?



Labels & Annotations

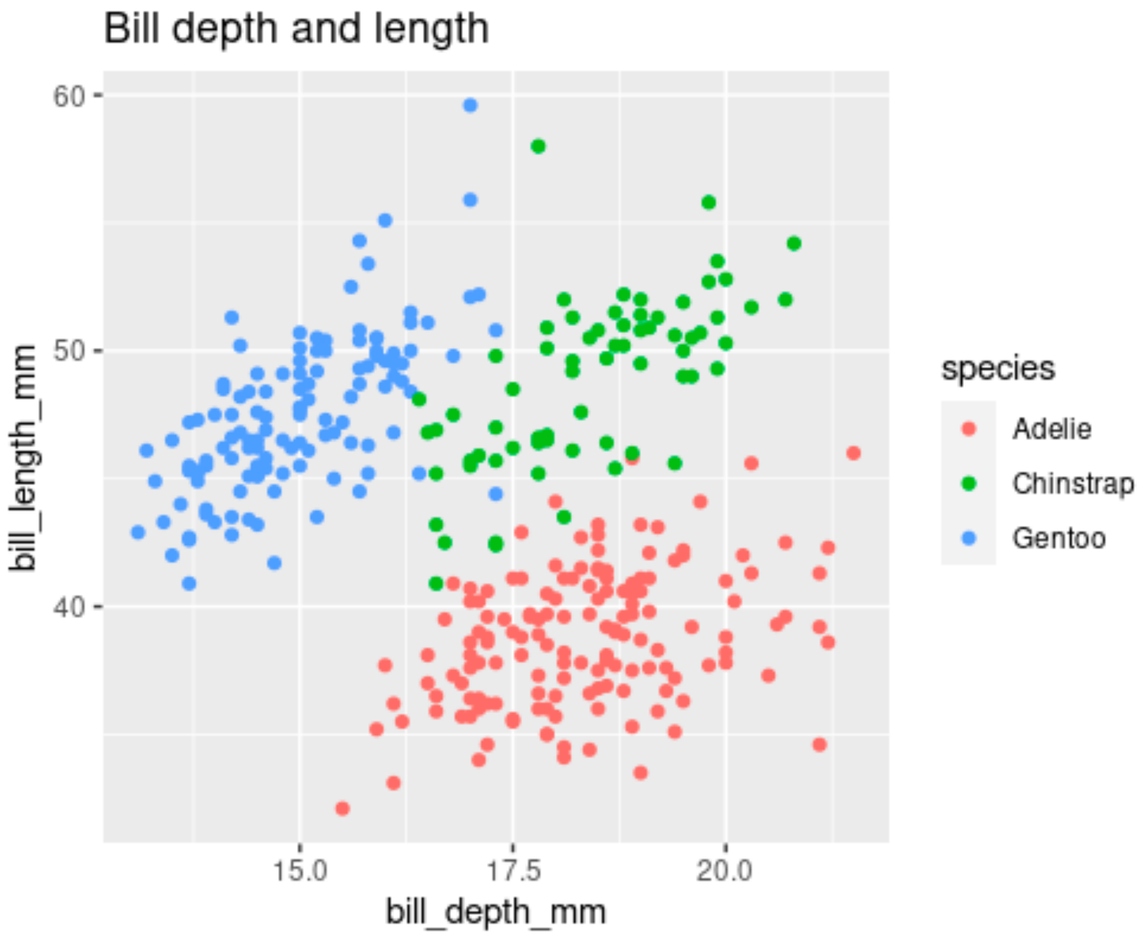
You can add axis/legend labels and titles using `labs()` function.





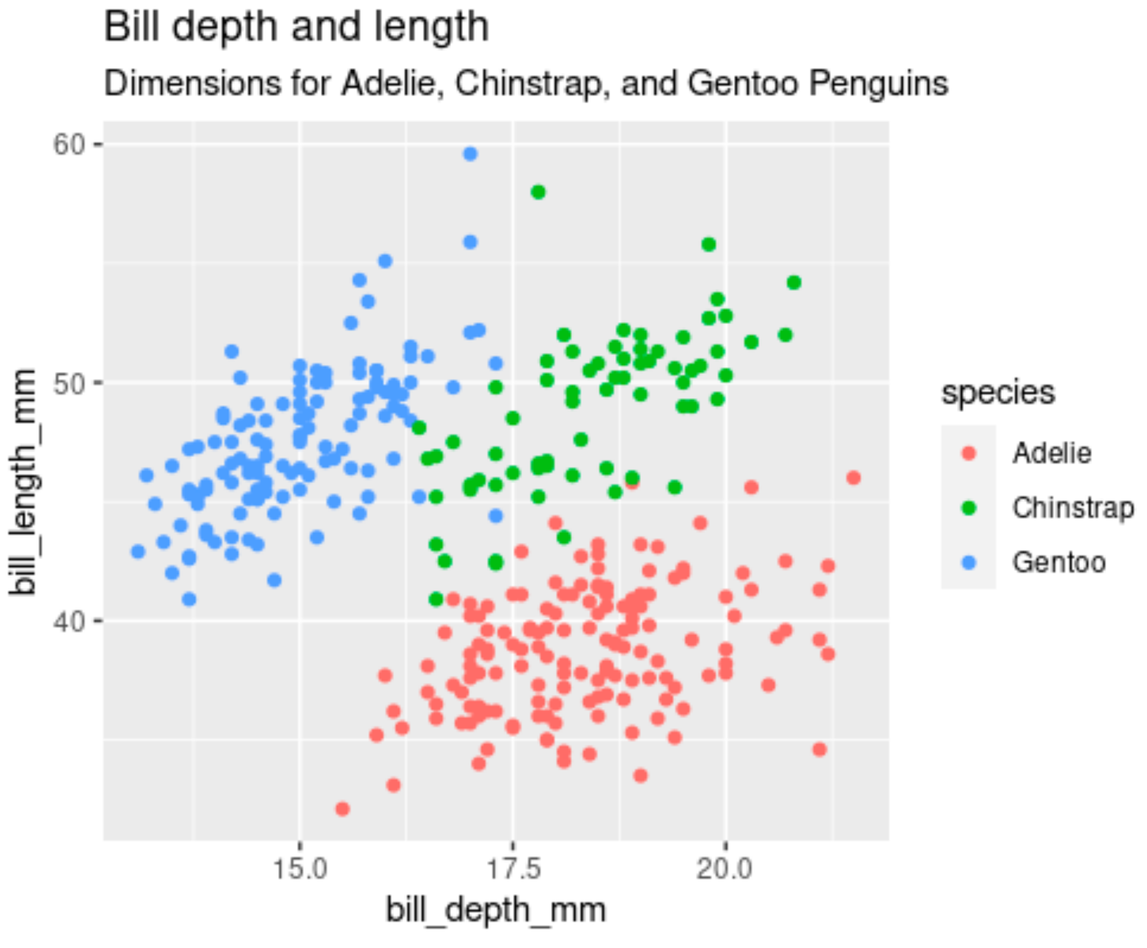
Start with the `penguins` data frame, map bill depth to the x-axis and map bill length to the y-axis. Represent each observation with a point and map species to the colour of each point. Title the plot "Bill depth and length"

```
ggplot(data = penguins,
       mapping = aes(x = bill_depth_mm,
                     y = bill_length_mm,
                     colour = species)) +
        geom_point() +
        labs(title = "Bill depth and length")
```



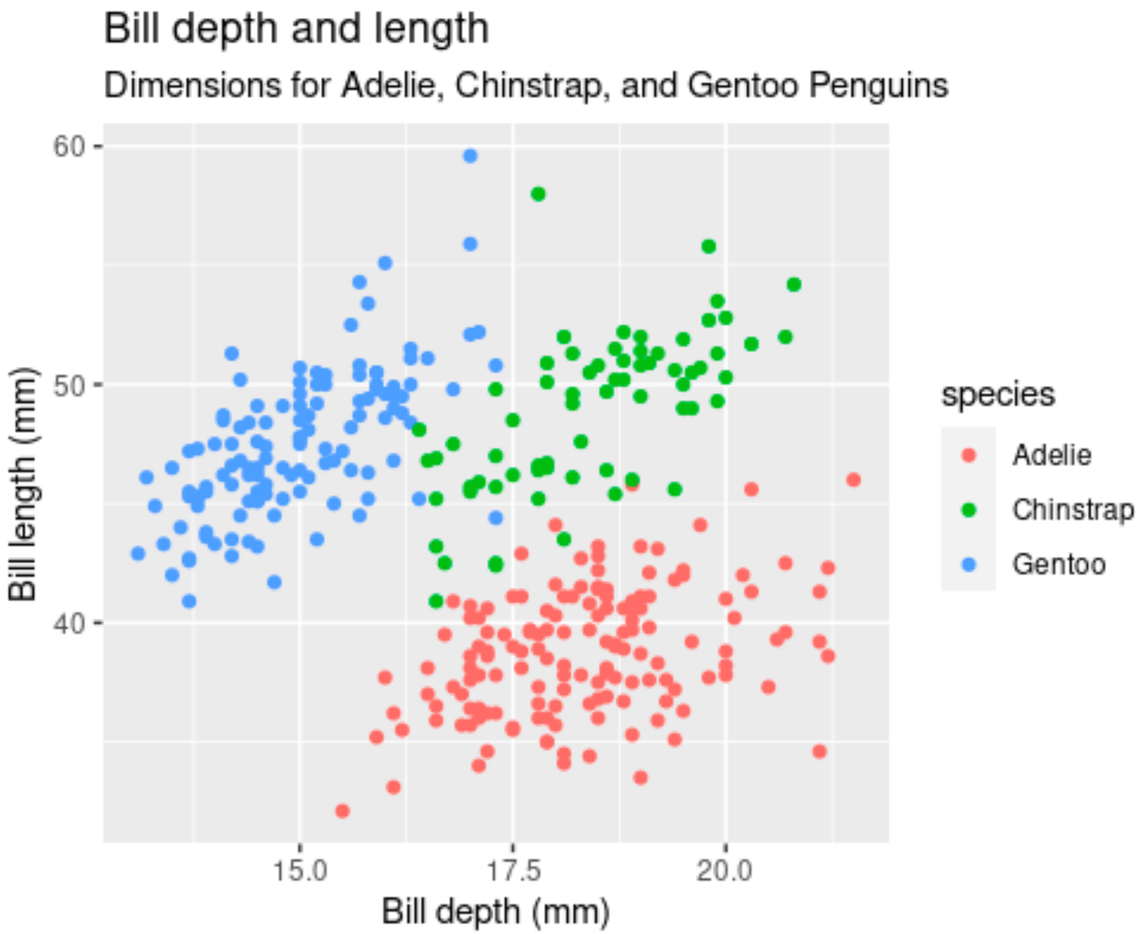
Start with the `penguins` data frame, map bill depth to the x-axis and map bill length to the y-axis. Represent each observation with a point and map species to the colour of each point. Title the plot "Bill depth and length", add the subtitle "Dimensions for Adelie, Chinstrap, and Gentoo Penguins".

```
ggplot(data = penguins,
       mapping = aes(x = bill_depth_mm,
                     y = bill_length_mm,
                     colour = species)) +
       geom_point() +
       labs(title = "Bill depth and length",
            subtitle = "Dimensions for Adelie, C
```



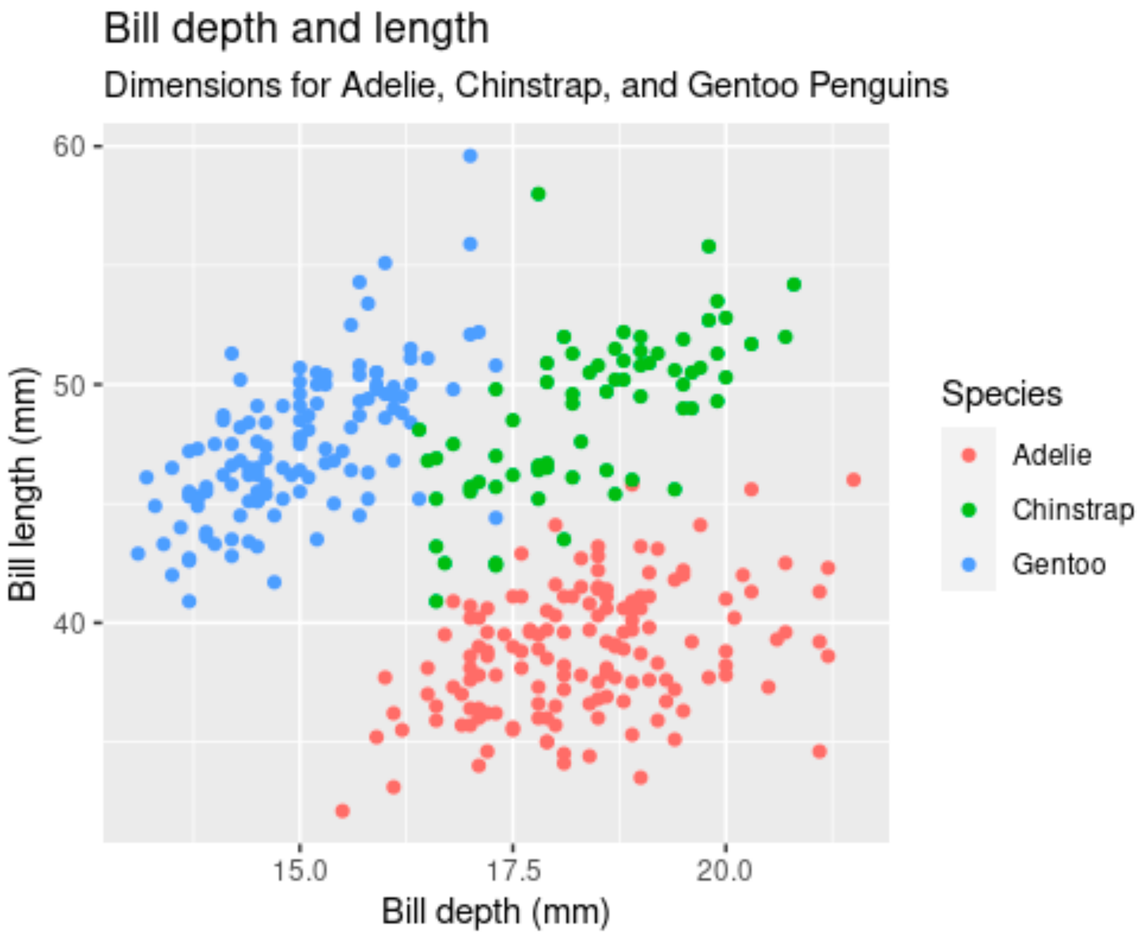
Start with the `penguins` data frame, map bill depth to the x-axis and map bill length to the y-axis. Represent each observation with a point and map species to the colour of each point. Title the plot "Bill depth and length", add the subtitle "Dimensions for Adelie, Chinstrap, and Gentoo Penguins". Label the x and y axes as "Bill depth (mm)" and "Bill length (mm)", respectively.

```
ggplot(data = penguins,
       mapping = aes(x = bill_depth_mm,
                     y = bill_length_mm,
                     colour = species)) +
        geom_point() +
        labs(title = "Bill depth and length",
             subtitle = "Dimensions for Adelie, C
             x = "Bill depth (mm)",
             y = "Bill length (mm)")
```



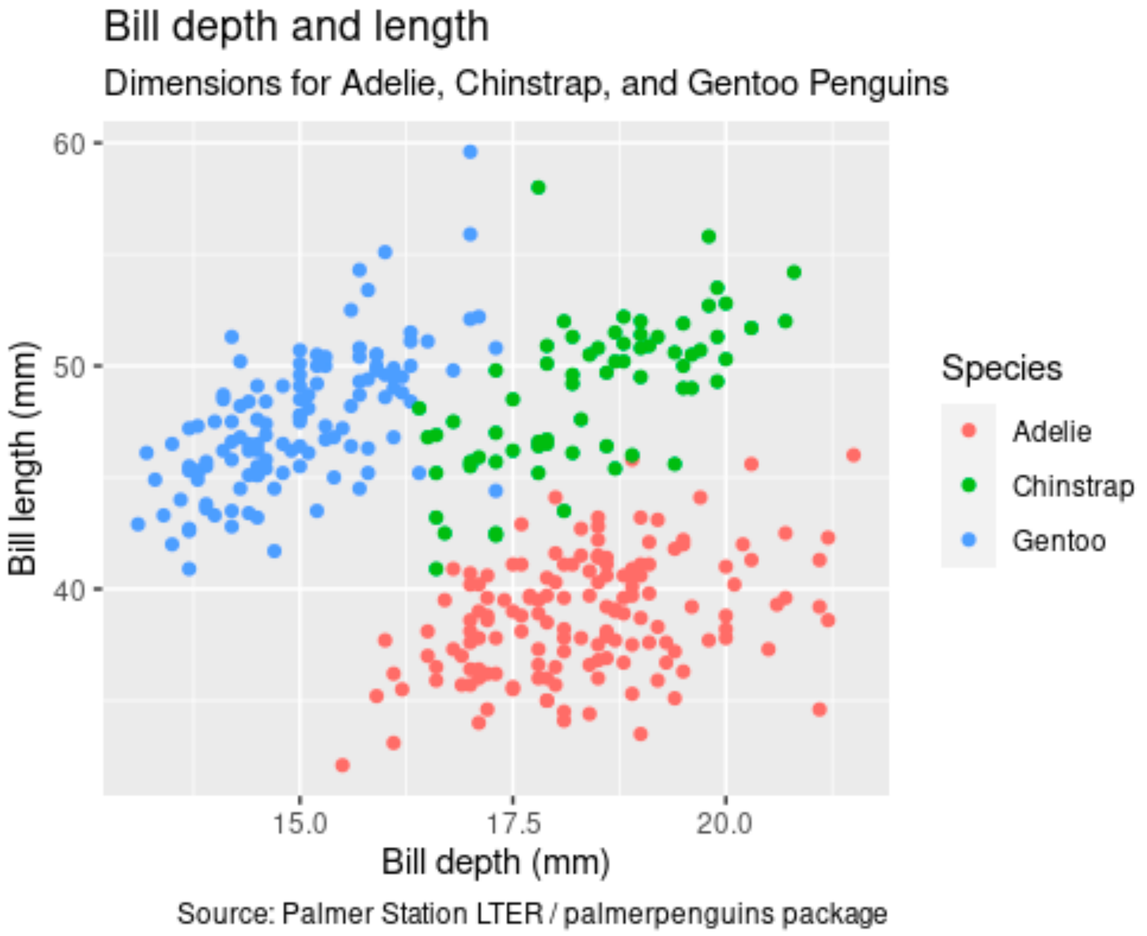
Start with the `penguins` data frame, map bill depth to the x-axis and map bill length to the y-axis. Represent each observation with a point and map species to the colour of each point. Title the plot "Bill depth and length", add the subtitle "Dimensions for Adelie, Chinstrap, and Gentoo Penguins". Label the x and y axes as "Bill depth (mm)" and "Bill length (mm)", respectively. Label the legend "Species".

```
ggplot(data = penguins,
       mapping = aes(x = bill_depth_mm,
                     y = bill_length_mm,
                     colour = species)) +
        geom_point() +
        labs(title = "Bill depth and length",
             subtitle = "Dimensions for Adelie, Chi
             x = "Bill depth (mm)",
             y = "Bill length (mm)",
             colour = "Species")
```



Start with the `penguins` data frame, map bill depth to the x-axis and map bill length to the y-axis. Represent each observation with a point and map species to the colour of each point. Title the plot "Bill depth and length", add the subtitle "Dimensions for Adelie, Chinstrap, and Gentoo Penguins". Label the x and y axes as "Bill depth (mm)" and "Bill length (mm)", respectively. Label the legend "Species". Add a caption for the data source.

```
ggplot(data = penguins,
       mapping = aes(x = bill_depth_mm,
                     y = bill_length_mm,
                     colour = species)) +
       geom_point() +
       labs(title = "Bill depth and length",
            subtitle = "Dimensions for Adelie, Ch
            x = "Bill depth (mm)",
            y = "Bill length (mm)",
            colour = "Species",
            caption = "Source: Palmer Station LTE
```



Application Exercise

Using the `mpg` data set, map the engine displacement values to the x-axis and the highway miles per gallon to the y-axis.
Represent each observation with a point.
Apply the “darkgreen" colour to the entire geometry.
Add a meaningful title (and subtitle), and labels.

Argument names

You can omit the names of first two arguments when building plots with `ggplot( ).`

```
ggplot(data = penguins,
       mapping = aes(x = bill_depth_mm,
                     y = bill_length_mm,
                     colour = species)) +
  geom_point()
```

```
ggplot(penguins,
       aes(x = bill_depth_mm,
           y = bill_length_mm,
           colour = species)) +
  geom_point()
```

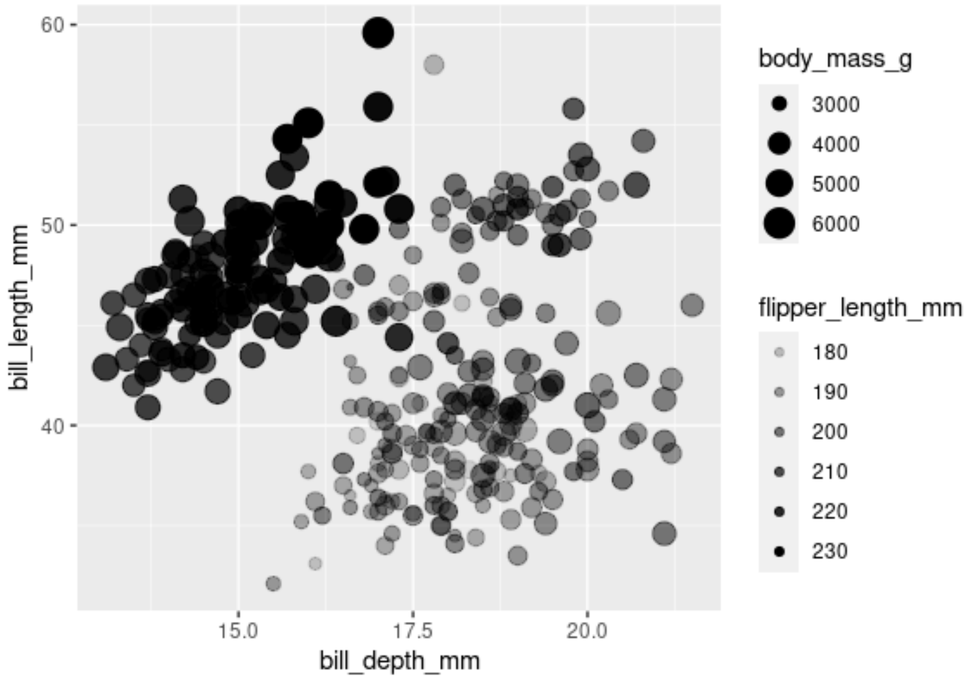
Aesthetics options

Commonly used characteristics of plotting characters that can be mapped to a specific variable in the data are

- `colour` (colour and fill)
- `shape`
- `size`
- `alpha` (transparency)

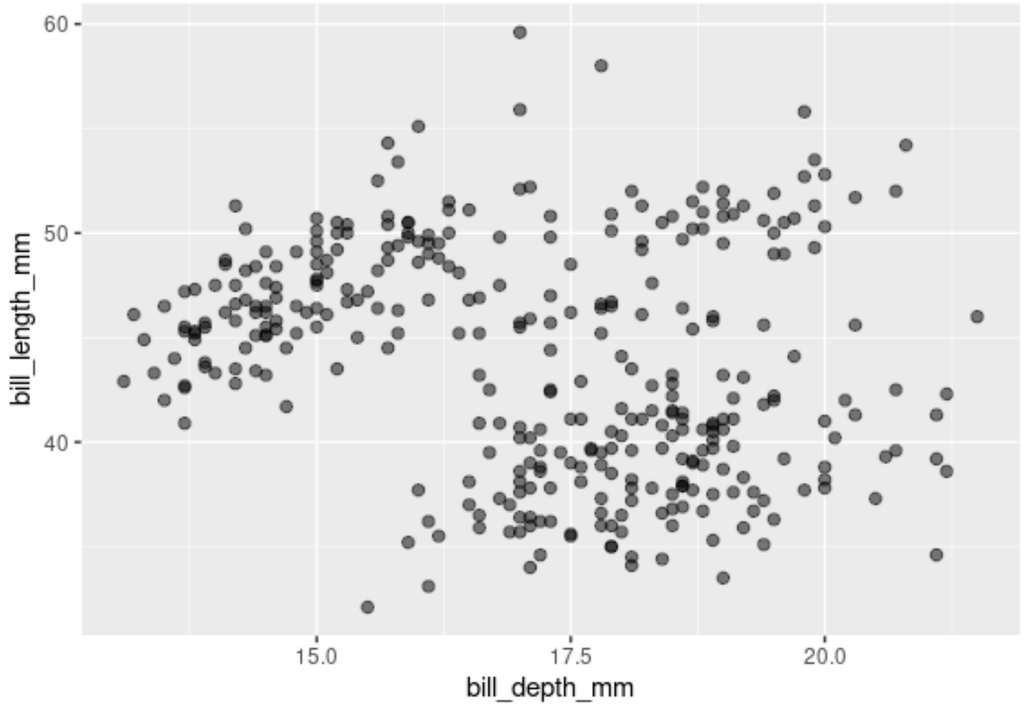
Mapping

```
ggplot(penguins,
       aes(x = bill_depth_mm,
           y = bill_length_mm,
           size = body_mass_g,
           alpha = flipper_length_mm)) +
  geom_point()
```



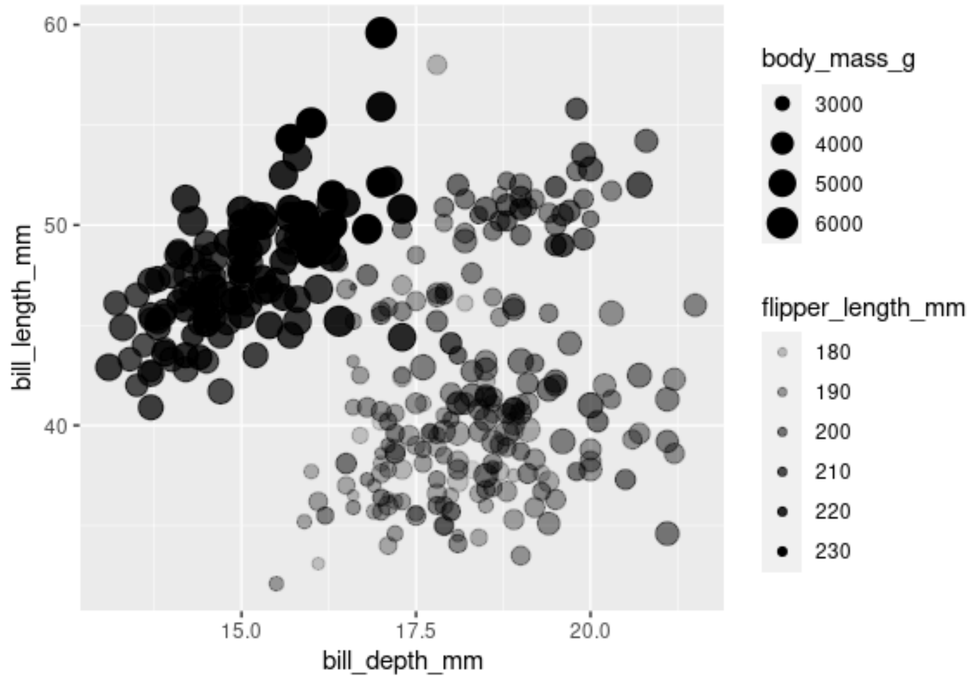
Setting

```
ggplot(penguins,
       aes(x = bill_depth_mm,
           y = bill_length_mm)) +
  geom_point(size = 2, alpha = 0.5)
```



Mapping

```
ggplot(penguins,
       aes(x = bill_depth_mm,
           y = bill_length_mm,
           size = body_mass_g,
           alpha = flipper_length_mm)) +
  geom_point()
```



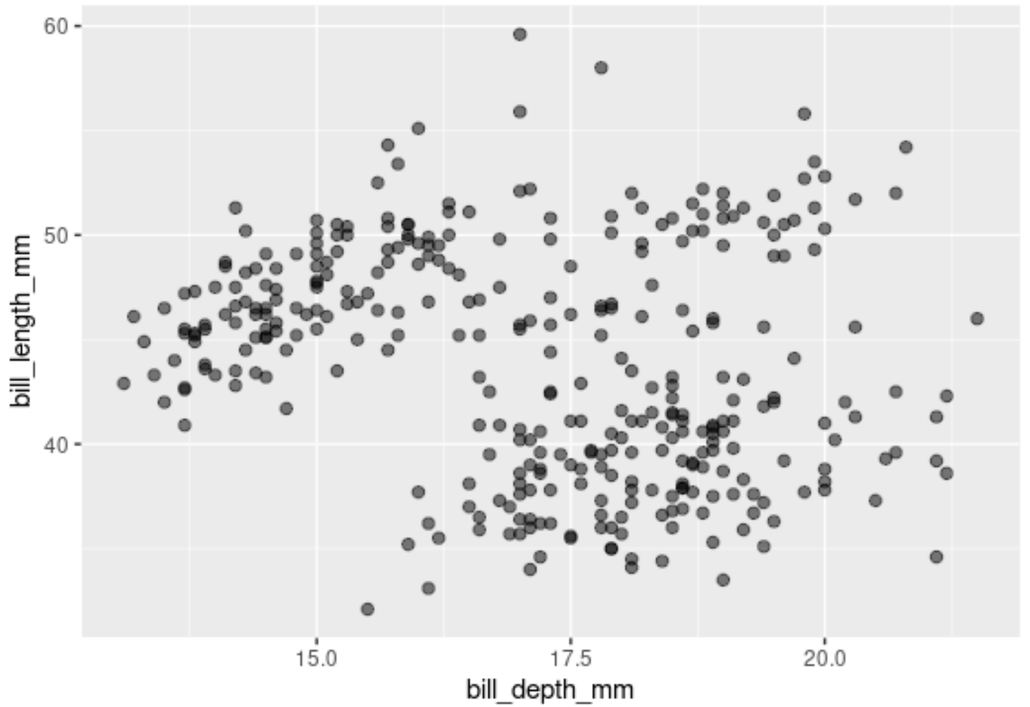
- `Mapping:` Determine the size, colour, alpha, etc.. of points based on the values of a variable in the data
- Goes into `aes()`

Setting

- `Setting:` Determine the size, colour, alpha, etc.. of points globally, and **not** based on the values of a variable in the data

- Goes into `geom_*()`
  (geom_point in this example,
  but we'll learn about other geoms next!)

```
ggplot(penguins,
       aes(x = bill_depth_mm,
           y = bill_length_mm)) +
  geom_point(size = 2, alpha = 0.5)
```



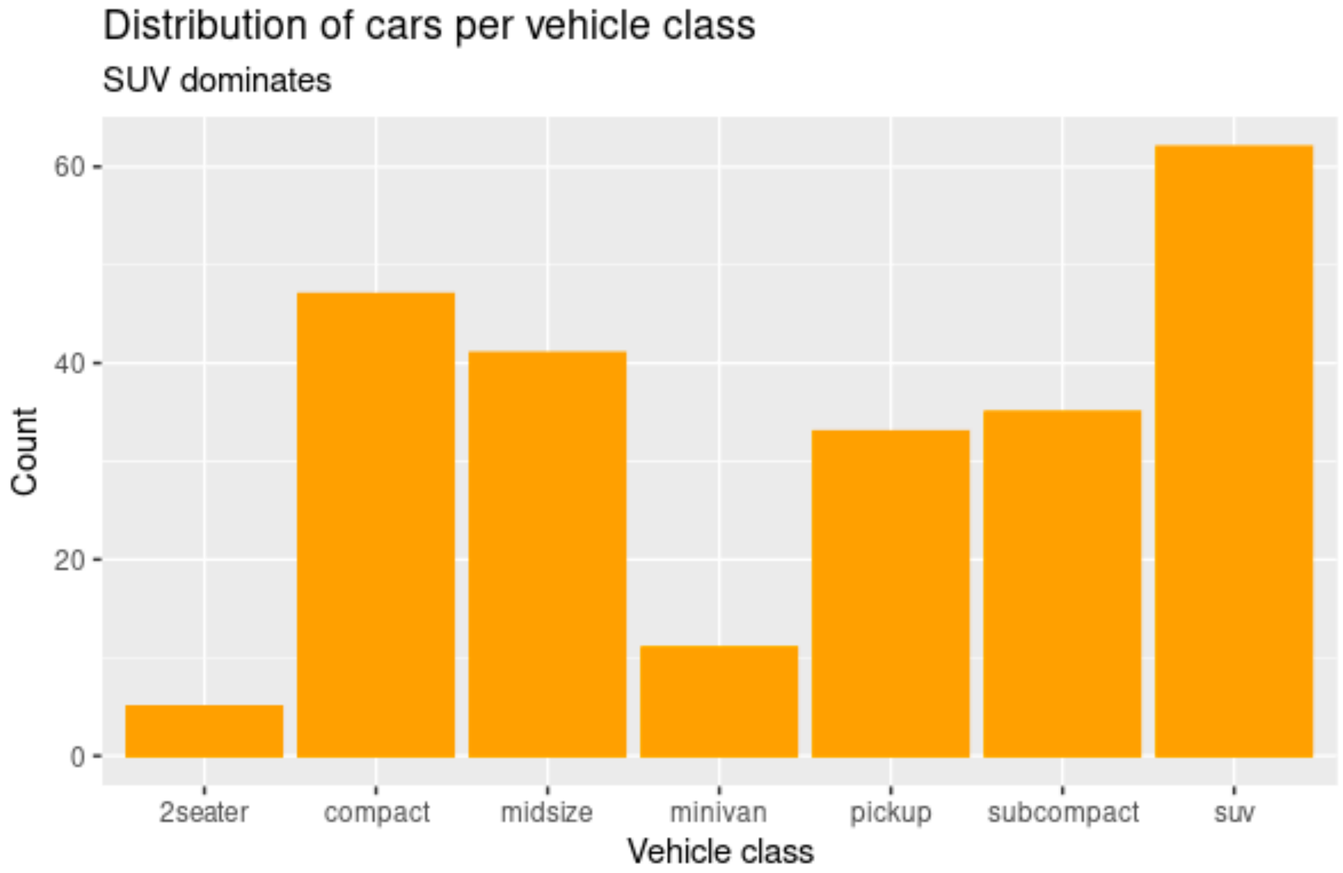
Geometric Objects

`ggplot2` makes it easy to plot different geometric objects using `geom_*()` functions.

- `geom_point()` for scatterplots
- `geom_line()` for line charts
- `geom_smooth()` for smoothed lines
- `geom_bar()` for bar charts
- `geom_histogram()` for histograms
- `geom_polygon()` for arbitrary shapes
- `geom_map()` for maps
- etc.

Application Exercise

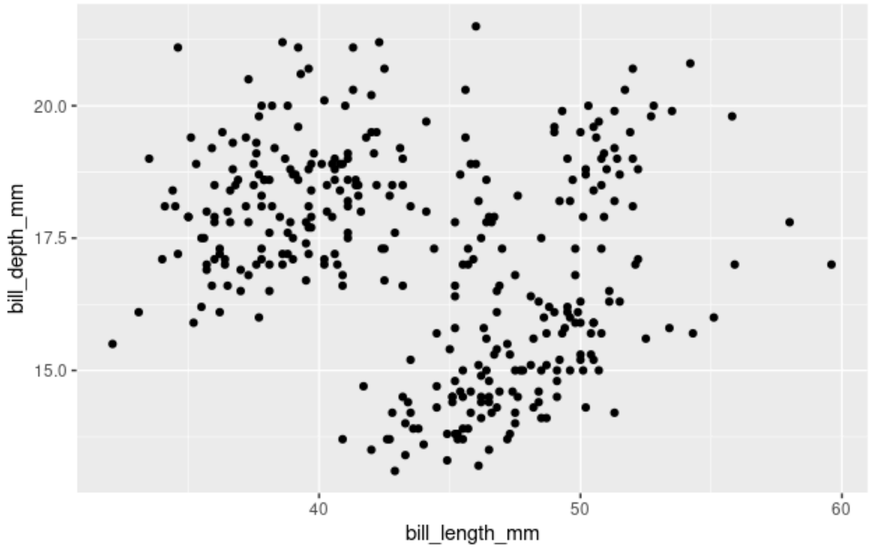
Using the `mpg` dataset, recreate the chart below.



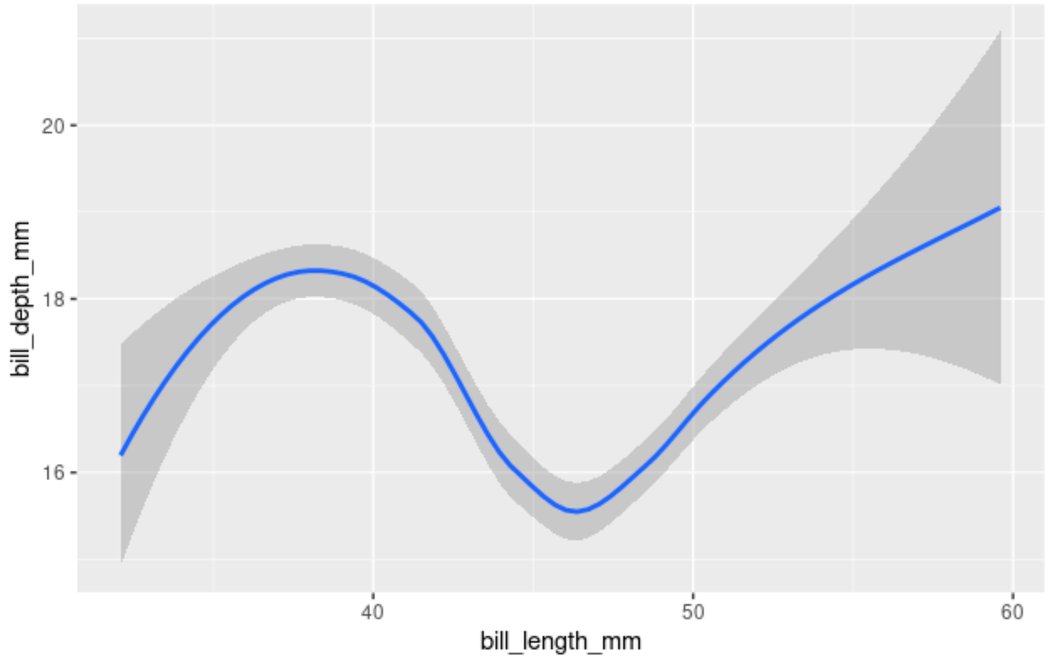
Geometric Objects

Most `geom_*()` require x and y mappings at the minimum.

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




```
ggplot(data = penguins) +
  geom_smooth(mapping = aes(
    x = bill_length_mm, y = bill_depth_mm))
```

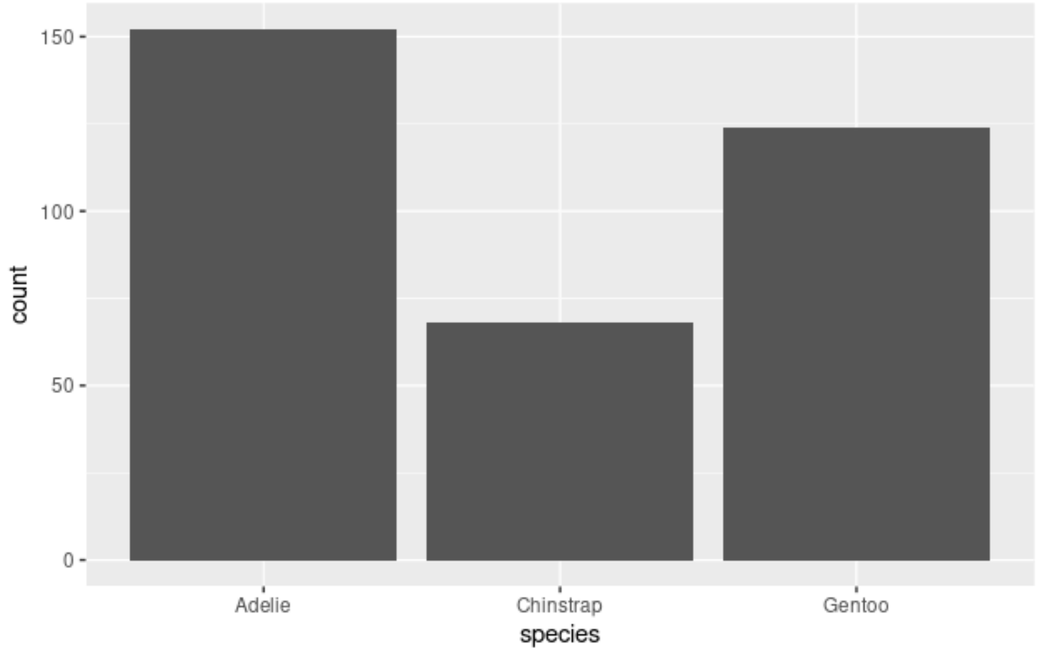


`geom_smooth()` uses polynomial regression to smooth the curve for data with less than 1,000 observations by default. For details, see *methods* in the documentation.

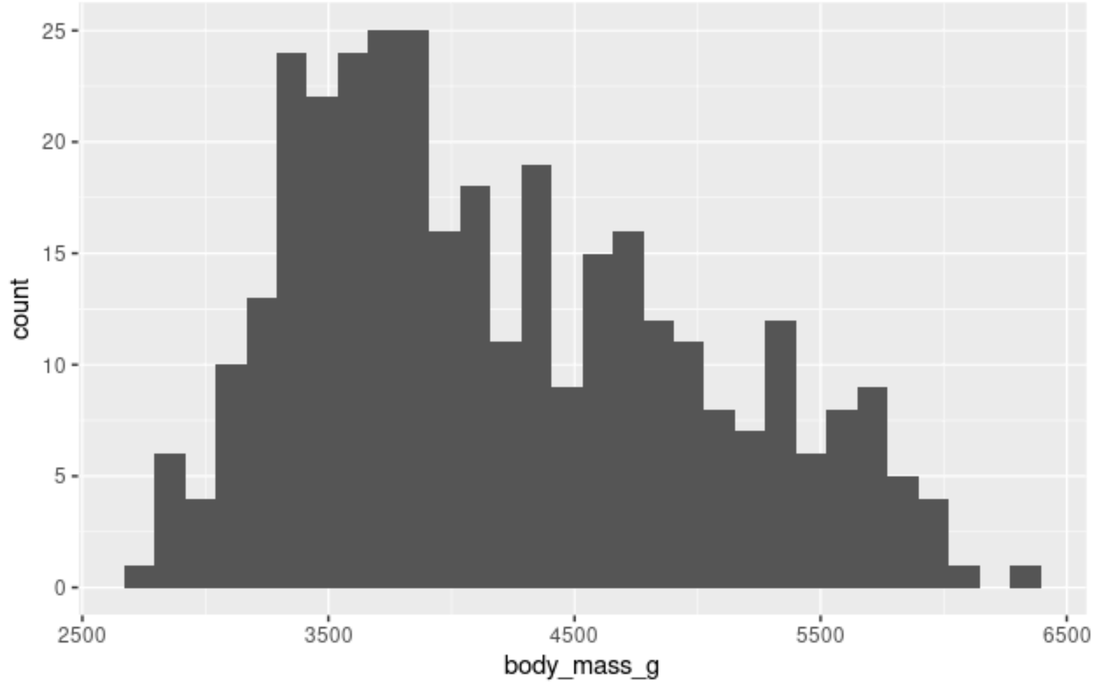
Geometric Objects

Some `geom_*()` only require a single mapping.

```
ggplot(data = penguins) +
  geom_bar(
    mapping = aes(x = species))
```



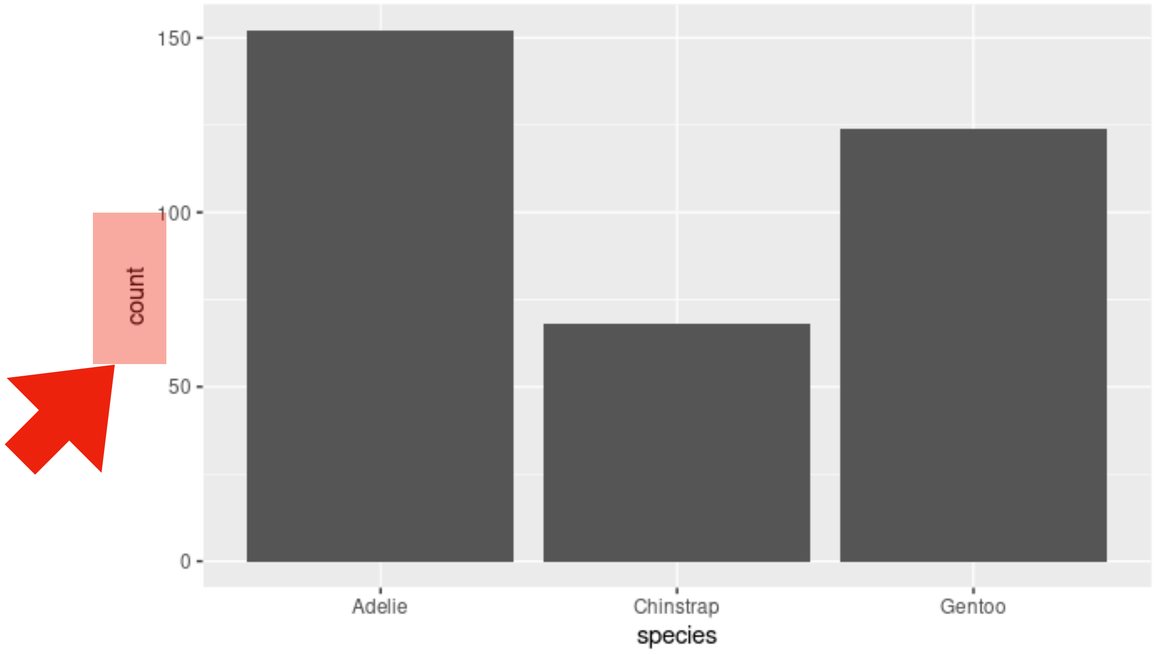

```
ggplot(data = penguins) +
  geom_histogram(
    mapping = aes(x = body_mass_g))
```



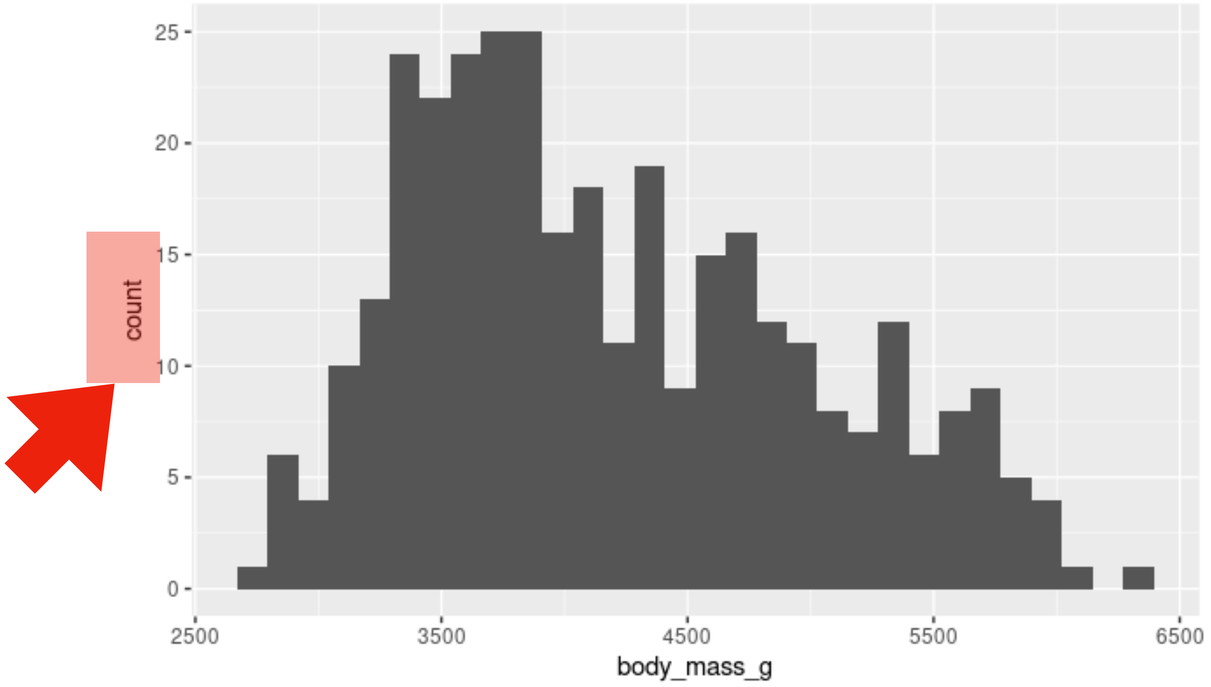
Statistical Transformations

The “count" values mapped to the y-axes below are not part of the source data set `penguins`. They are the results of statistical transformations.

```
ggplot(data = penguins) +
  geom_bar(
    mapping = aes(x = species))
```



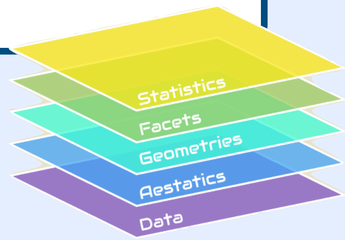

```
ggplot(data = penguins) +
  geom_histogram(
    mapping = aes(x = body_mass_g))
```



Statistical Transformations

```
ggplot([DATA],
       aes([MAPPINGS],
       stat = [STATFUNCTION])) +
  [GEOMFUNCTION]
```

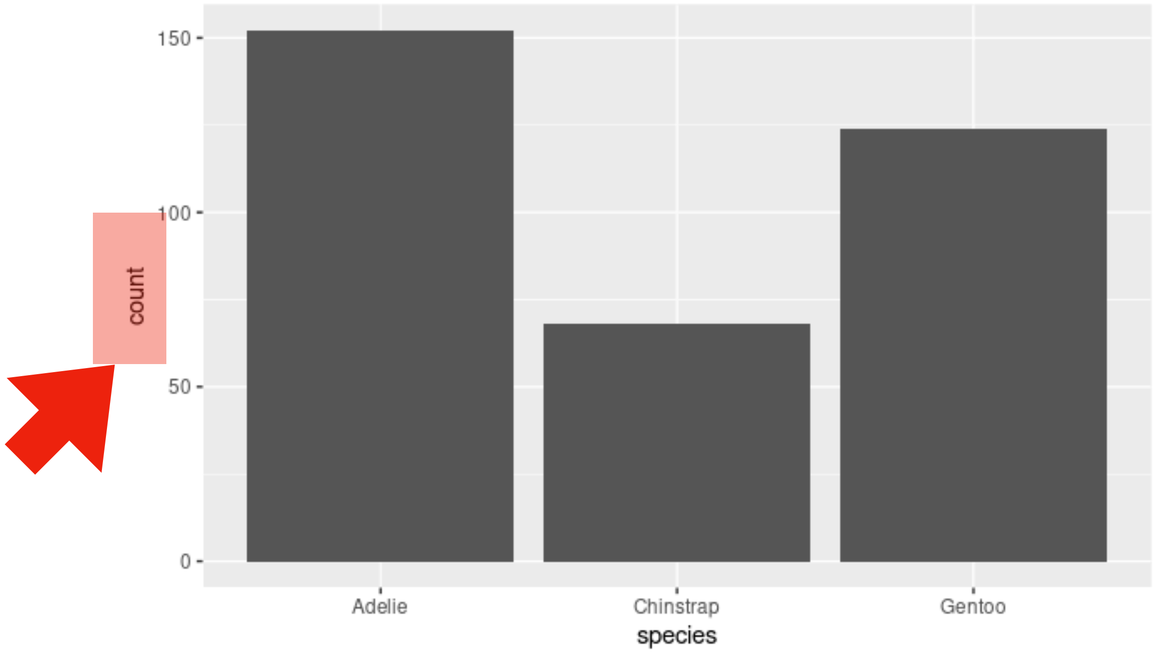
Specify a
STATISTICAL TRANSFORMATION
with the corresponding function.



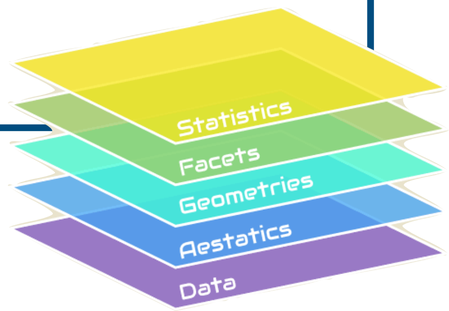
Statistical Transformations

`geom_bar()` automatically applied `stat_count()` transformation to count the number of records per unique value of `x`.

```
ggplot(data = penguins) +
  geom_bar(
    mapping = aes(x = species),
    stat = "count")
```



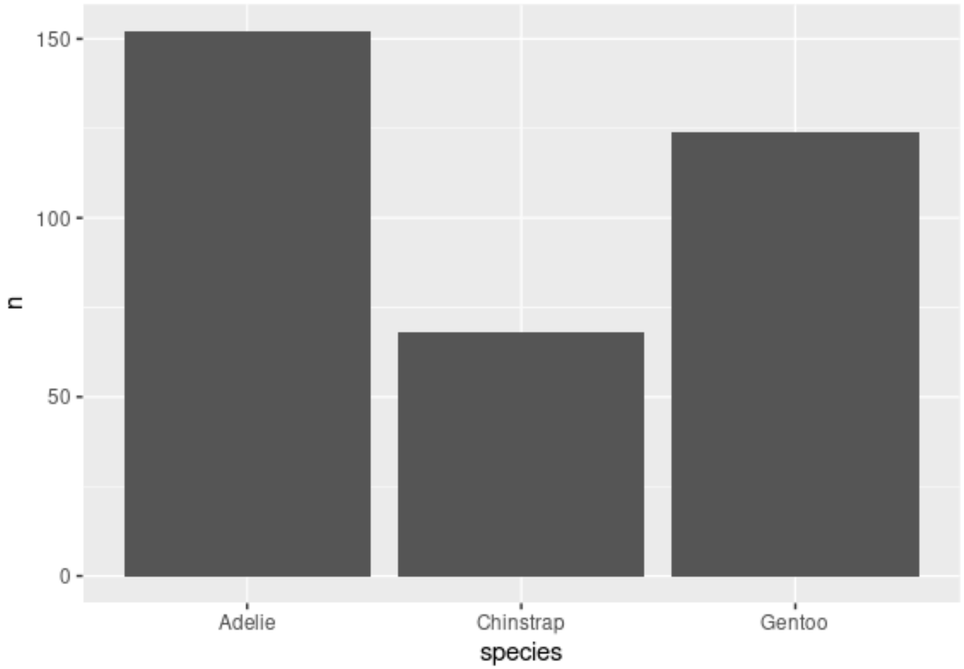
- `geom_bar()` applies the `stat_count()` transformation by default.
- You can change this behaviour by changing the value of **`stat`** argument.



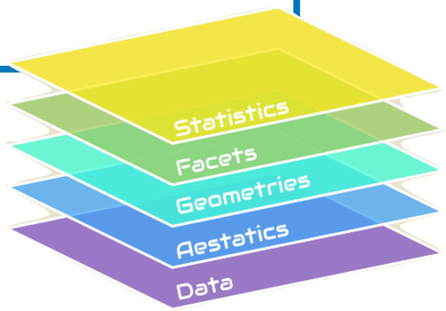
Statistical Transformations

You can change the type of statistical transformation applied by changing the value of `stat` argument.

```
ggplot(data = dplyr::count(penguins, species)) +
  geom_bar(
    mapping = aes(x = species, y = n),
    stat = "identity")
```



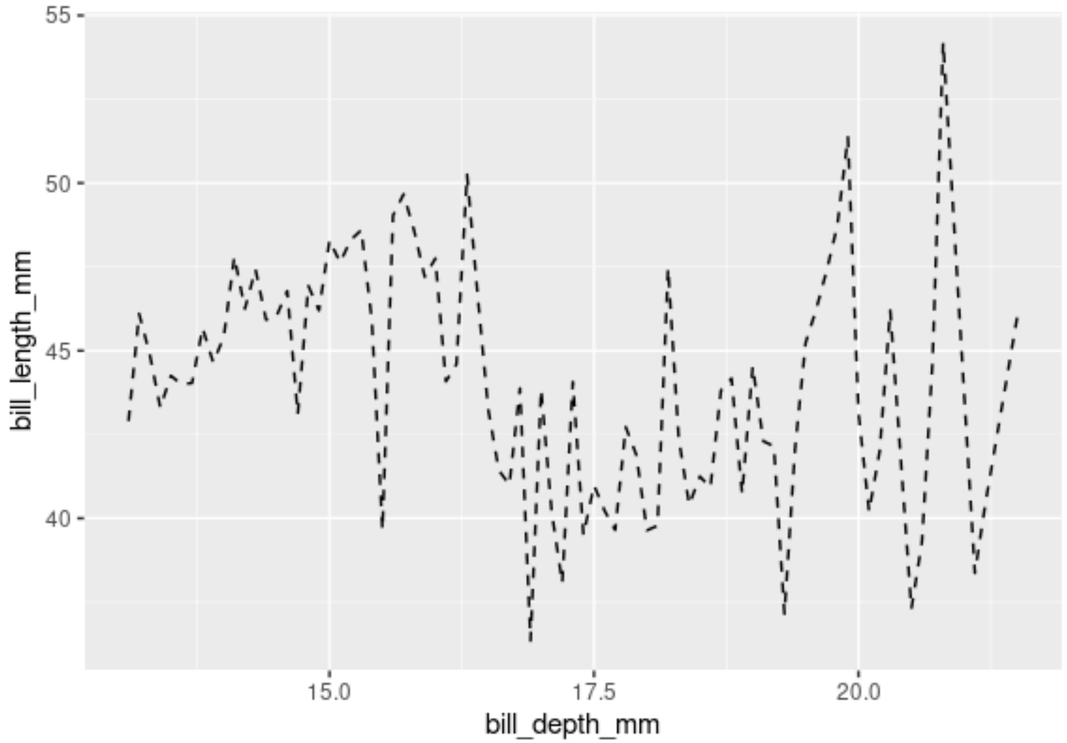
- For example, using `stat="identity"` takes the values as it is.
- Note that we need to supply the y-axis mapping in this case.
- `dplyr::count()` function takes care of the counting before passing data to `ggplot()` in this example.



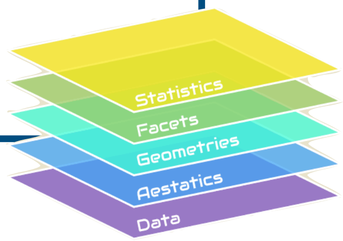
Statistical Transformations

We can also call `stat_*()` functions to directly specify the type of statistical transformations.

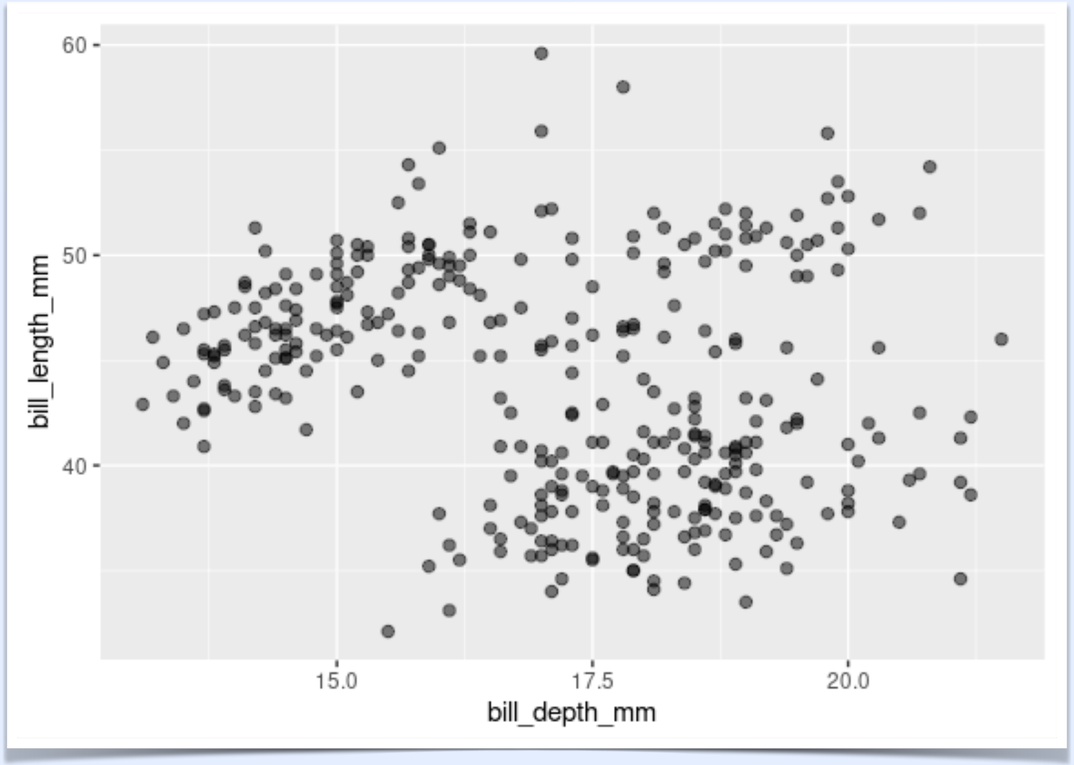
```
ggplot(penguins, aes(bill_depth_mm, bill_length_mm)) +
  stat_summary(fun = "mean", geom = "line",
               linetype = "dashed")
```

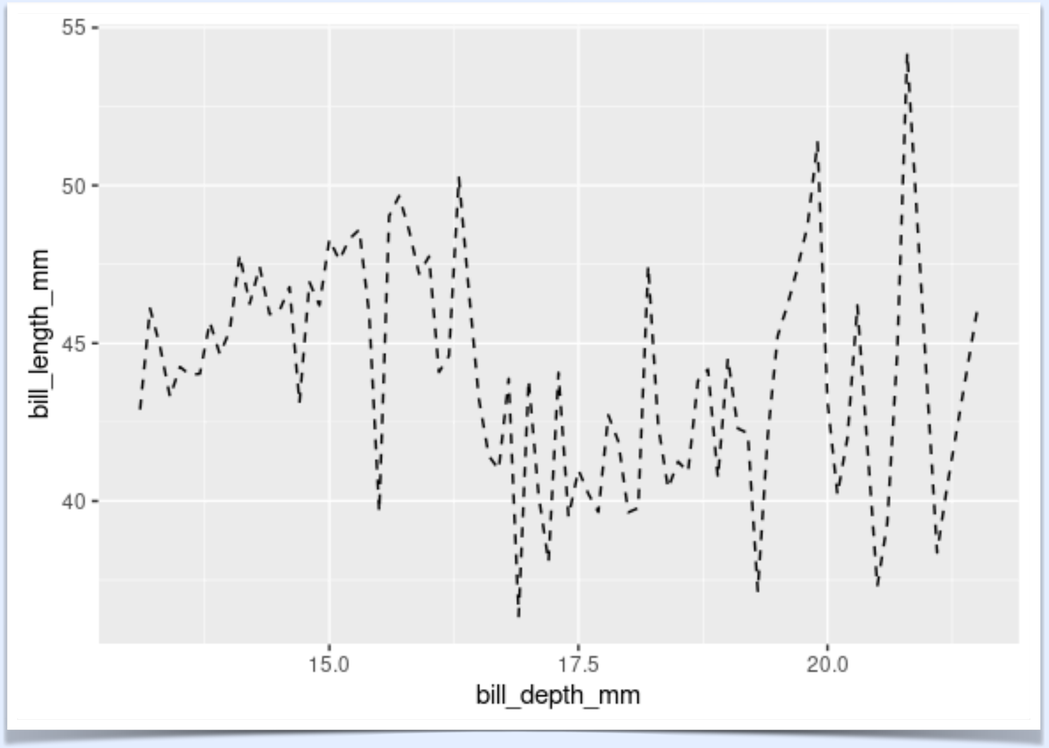


- `stat_summary()` transforms values of y using `fun` and places `geom` objects for each unique value of x.
- To apply the function after grouping values of x, you can use `stat_summary_bin()`.



Layering Geometric Objects

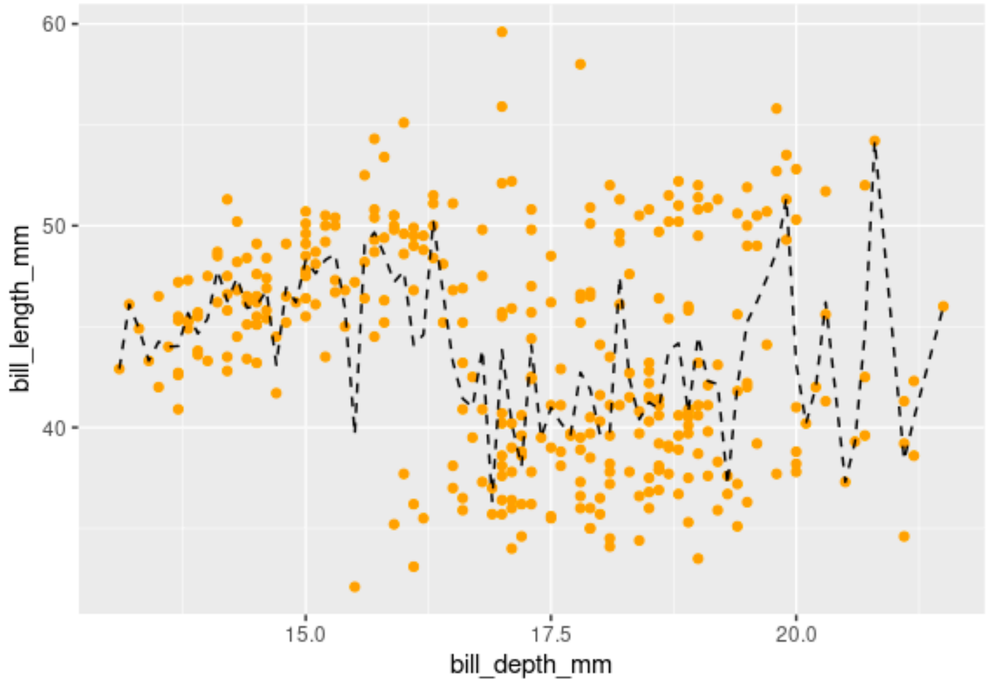




Layering Geometric Objects

Using multiple `geom_*()` functions, you can layer different geometric objects on a single plot.

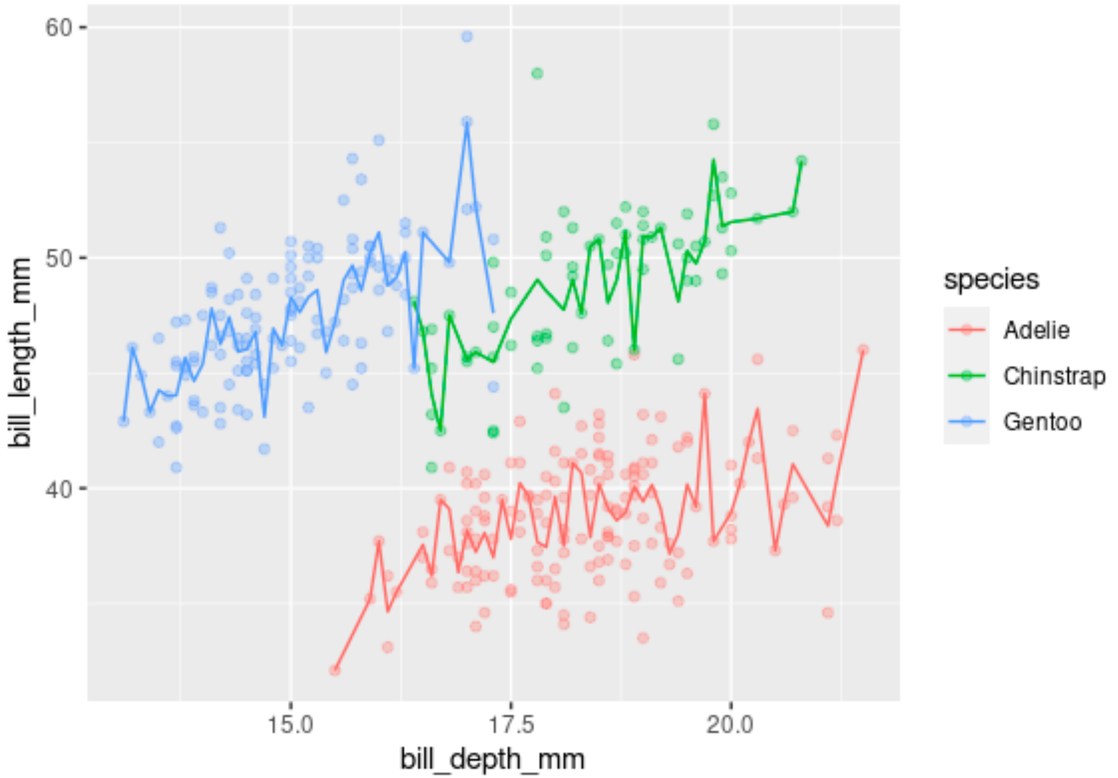
```
ggplot(penguins) +
  geom_point(aes(bill_depth_mm,
                 bill_length_mm),
             colour = "orange") +
  stat_summary(aes(bill_depth_mm,
                   bill_length_mm),
               fun = "mean",
               geom = "line",
               linetype = "dashed")
```



Layering Geometric Objects

We can specify aesthetic mapping in `ggplot()` to apply them commonly across the layers.

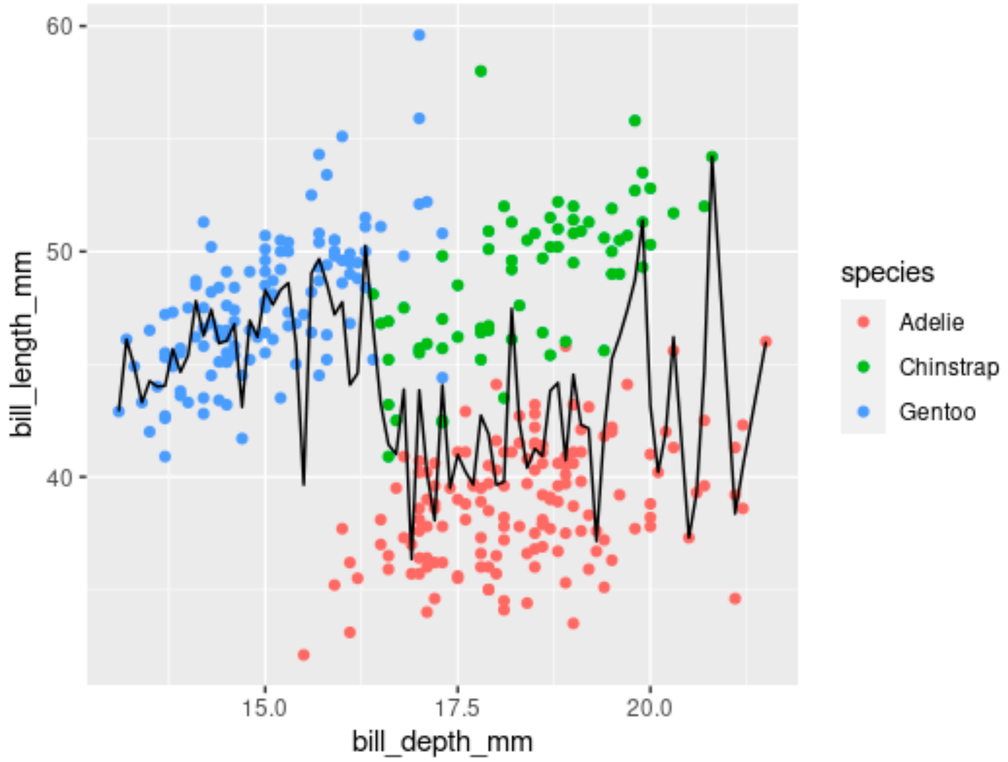
```
ggplot(penguins,
       aes(bill_depth_mm,
           bill_length_mm,
           colour = species)) +
  geom_point(alpha = 0.3) +
  stat_summary(fun = "mean",
               geom = "line")
```



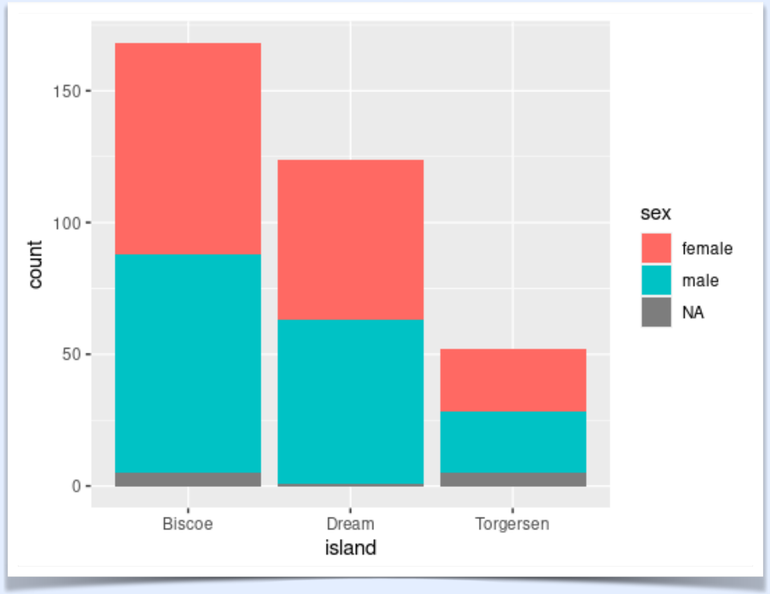
Layering Geometric Objects

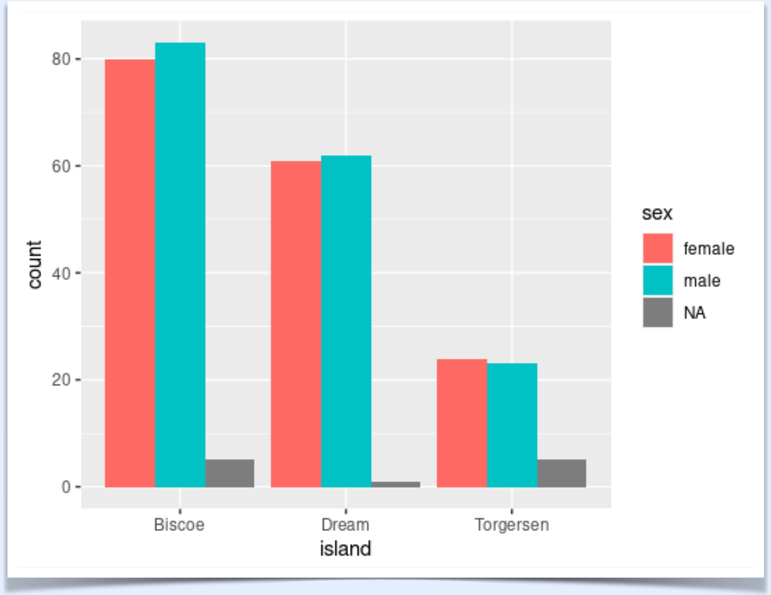
Or, we can specify aesthetic mapping within each `geom_*()` function to control the visual characteristics per layer.

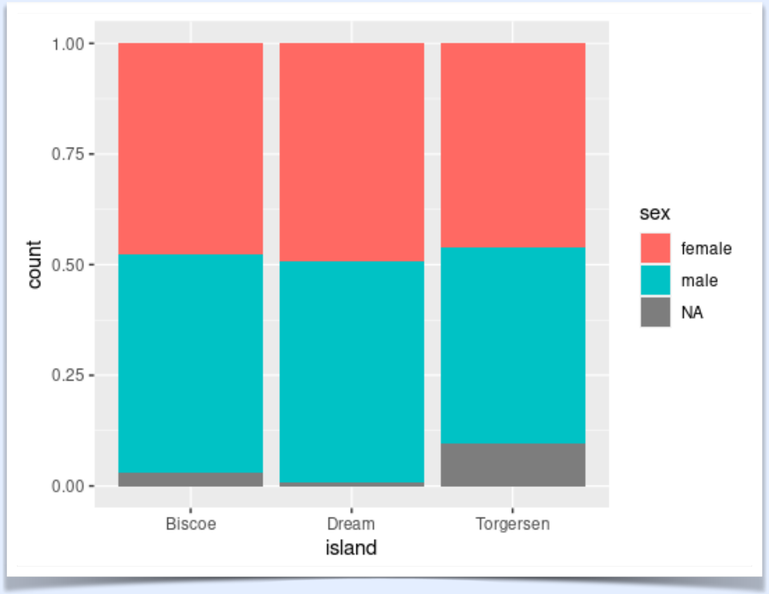
```
ggplot(penguins,
       aes(bill_depth_mm,
           bill_length_mm)) +
  geom_point(aes(colour = species)) +
  stat_summary(fun = "mean",
               geom = "line",
               colour = "black")
```



Position Adjustments





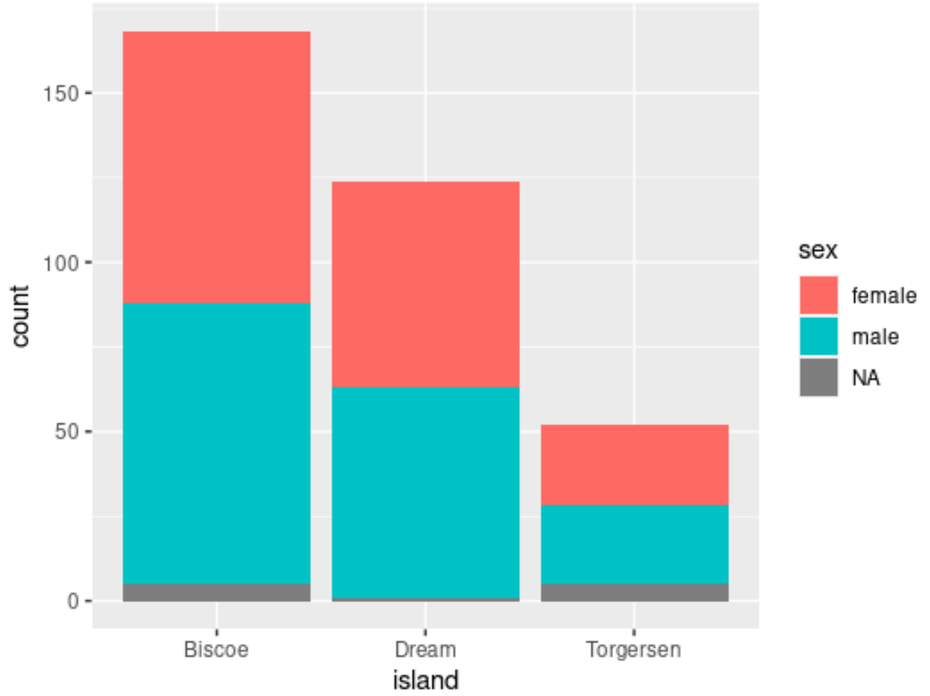


Position Adjustments

Each geom also has a default position adjustment. The effect of different position adjustment is most noticeable in `geom_bar()`.

```
ggplot(penguins) +
  geom_bar(aes(x = island, fill = sex))
```

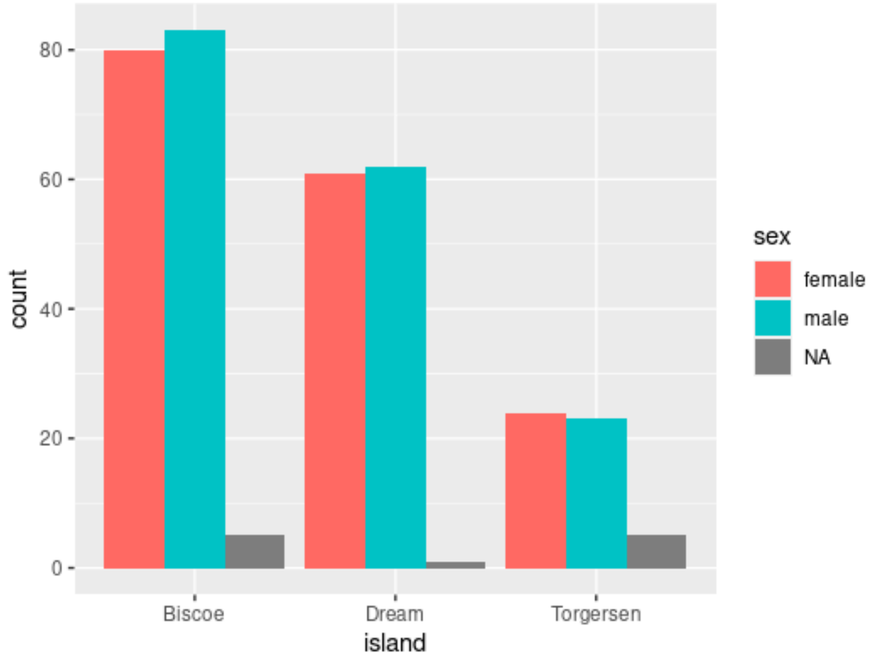
The default behaviour for positioning subcategories is to "`stack`".



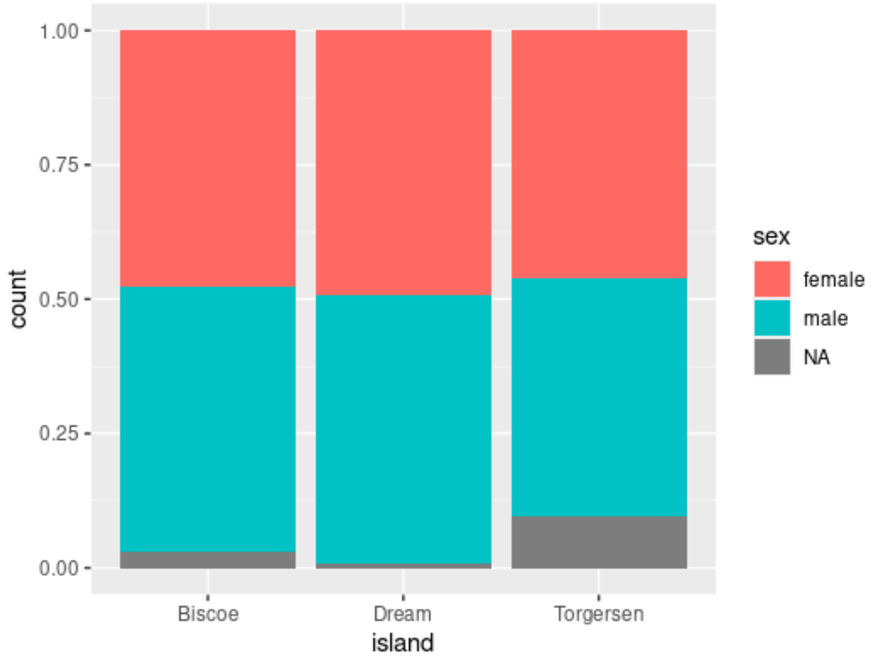
Position Adjustments

Using the `position` argument, you can specify other position adjustments for bar charts.

```
ggplot(penguins) +
  geom_bar(aes(x = island, fill = sex),
           position = "dodge")
```

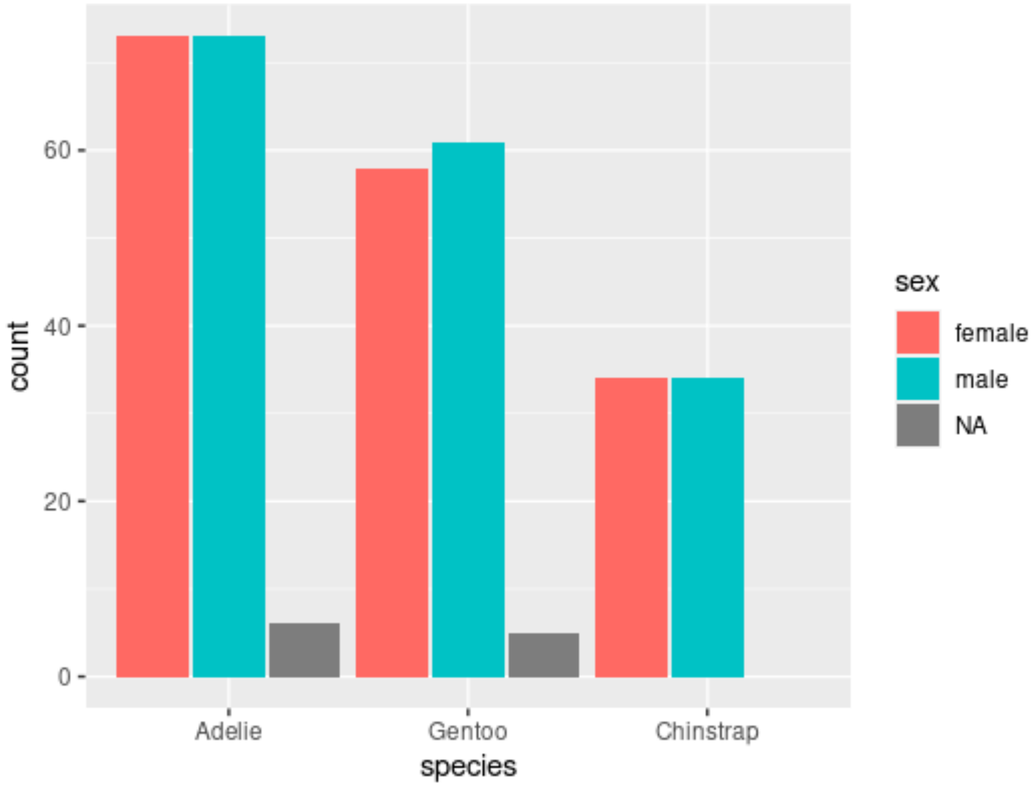



```
ggplot(penguins) +
  geom_bar(aes(x = island, fill = sex),
           position = "fill")
```



Position Adjustments

To sort categorical variables by occurrences, you can use `fct_infreq` function from `forcats` library. `position_dodge` function allows adjusting the dodge positioning behaviour such as the widths of individual bars.

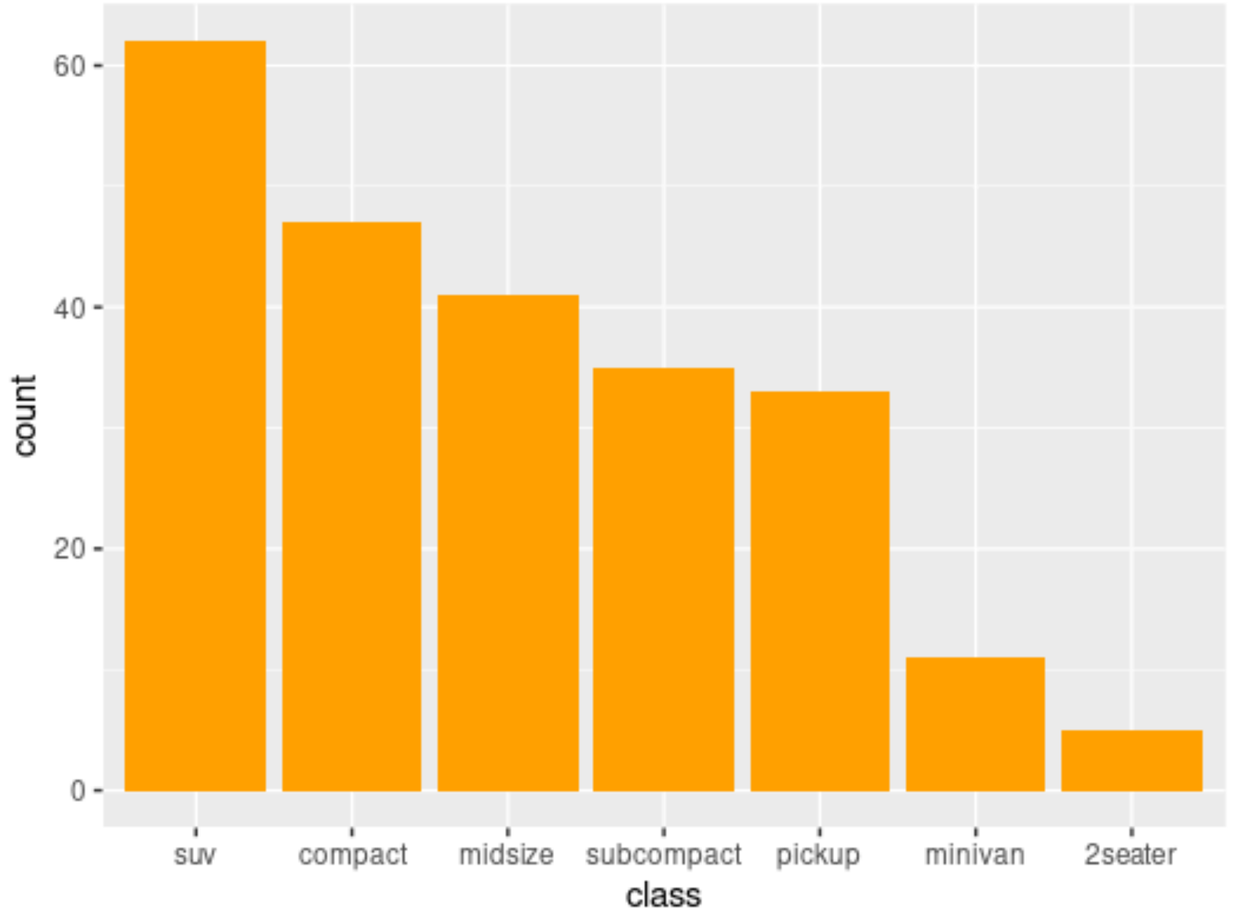



```r
library(dplyr)
mutate(
  penguins, species = forcats::fct_infreq(species)) %>%
  ggplot() +
  geom_bar(aes(x = species, fill = sex),
           position = position_dodge(0.95, "single"))
```

`mutate()` and `%>%` operator are from `dplyr` library.

Application Exercise

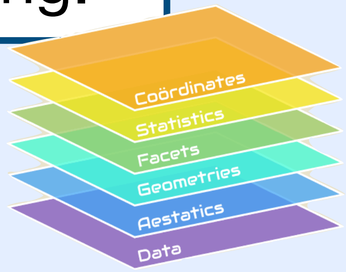
Using the `mpg` dataset, recreate the chart below.



Scales & Coordinate Systems

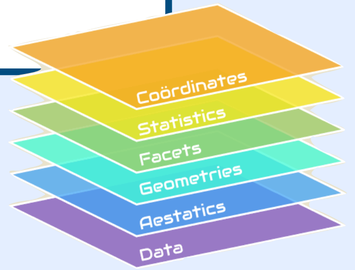
```
ggplot([DATA]) +
  [GEOMFUNCTION] +
  [SCALEFUNCTION]
```

Specify a **SCALE**
for aesthetic mapping.




```
ggplot([DATA]) +
  [GEOMFUNCTION] +
  [COORDFUNCTION]
```

Specify a
COORDINATE SYSTEM
for the plot.

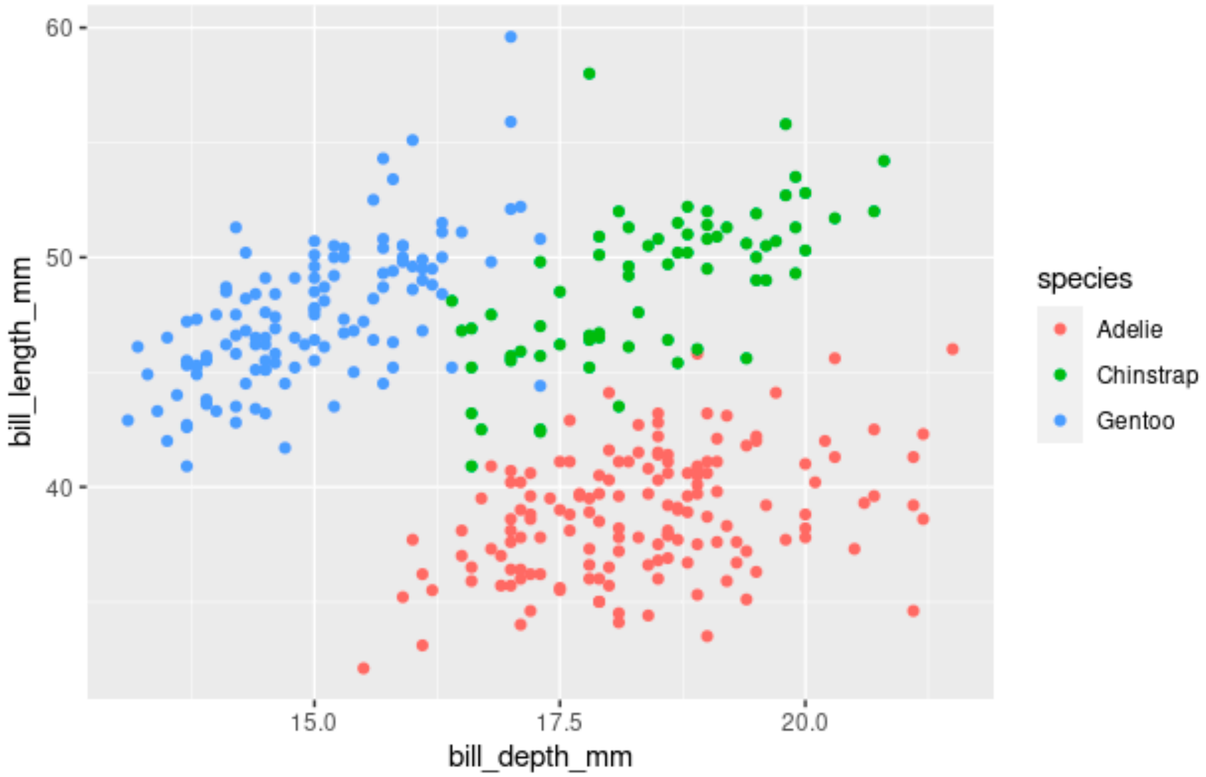


Scales

`ggplot()` uses a particular scale for each aesthetic mapping specified. You can manage them using `scale_<aesthetic>_<scale type>()`.

For example, the plot on the right uses `scale_x_continuous()`, `scale_y_continuous()`, and `scale_colour_discrete()` by default.

```
ggplot(penguins) +
  geom_point(aes(bill_depth_mm,
                 bill_length_mm,
                 colour = species))
```



Scales

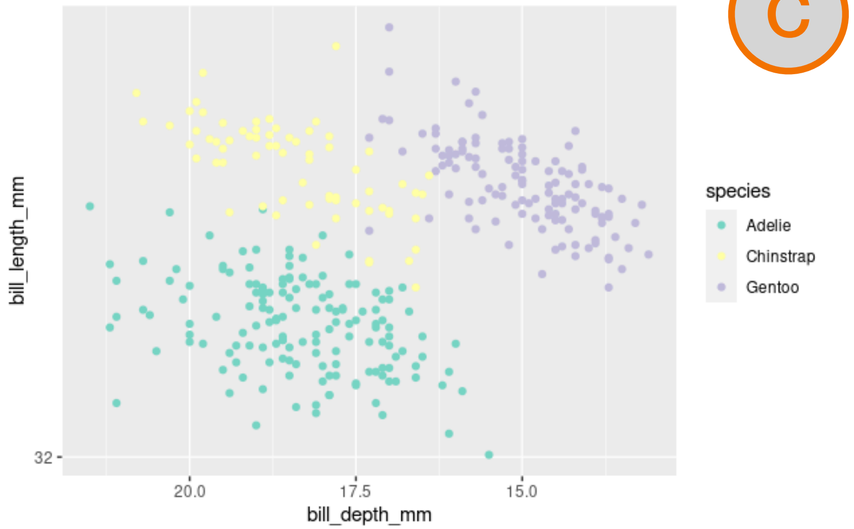
We can apply transformations to the scales using the functions such as reversing (A), log transform (B), and applying a specific colour palette (C).

```
ggplot(penguins) +
  geom_point(aes(bill_depth_mm,
                 bill_length_mm,
                 colour = species)) +
  scale_x_reverse() +
  scale_y_continuous(trans = "log2") +
  scale_colour_brewer(palette = "Set3")
```

A

B

C

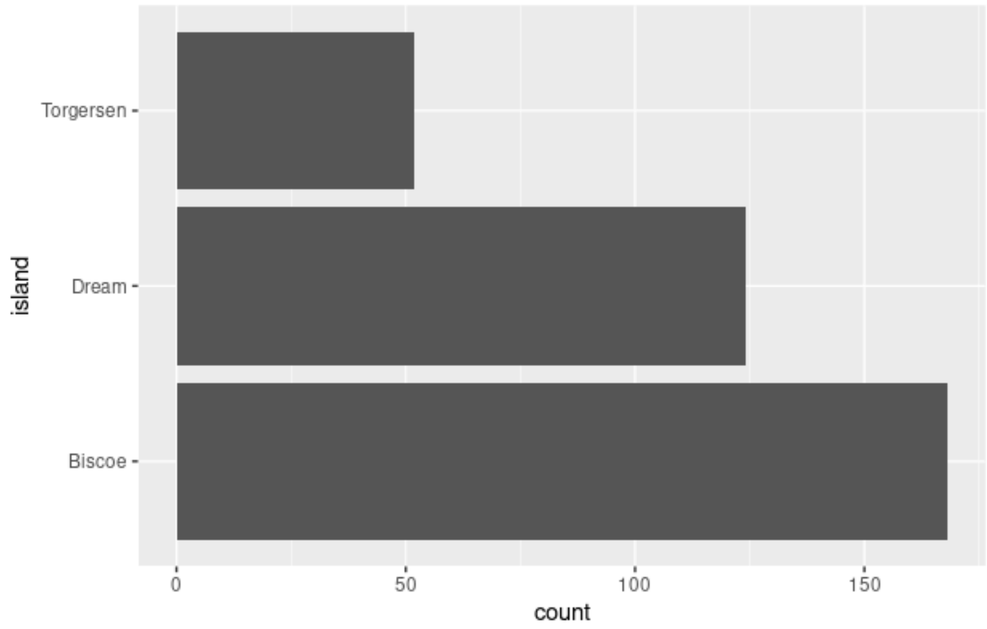


`scale_colour_brewer()` provides access to colour palettes from RColorBrewer (https://rdrr.io/cran/RColorBrewer/man/ColorBrewer.html).

Coordinate Systems

`coord_*()` functions allow customizing the coordinate system. For example, `coord_flip()` flips the x- and y-axes. This can help avoid overlapping axis labels without rotating them.

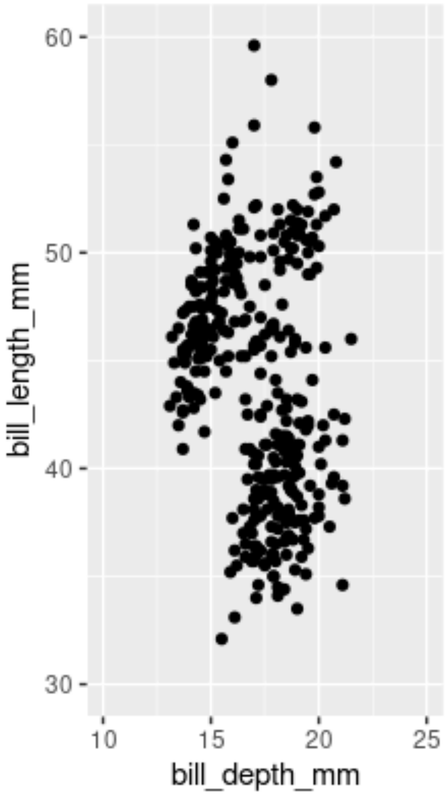
```
mutate(
  penguins,
  island = forcats::fct_infreq(island)) %>%
  ggplot() +
  geom_bar(aes(x = island)) +
  coord_flip()
```



Coordinate Systems

`coord_equal()` can specify the ratio between the physical lengths of a unit on the x-axis and a unit on the y-axis. `coord_*()` functions also allows setting axis limits.

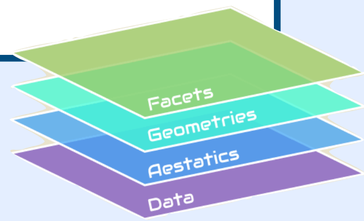
```
ggplot(penguins) +
  geom_point(aes(bill_depth_mm,
                 bill_length_mm)) +
  coord_equal(xlim = c(10,25),
              ylim = c(30,60))
```



Facets

```
ggplot([DATA]) +
  [GEOMFUNCTION] +
  [FACETFUNCTION]
```

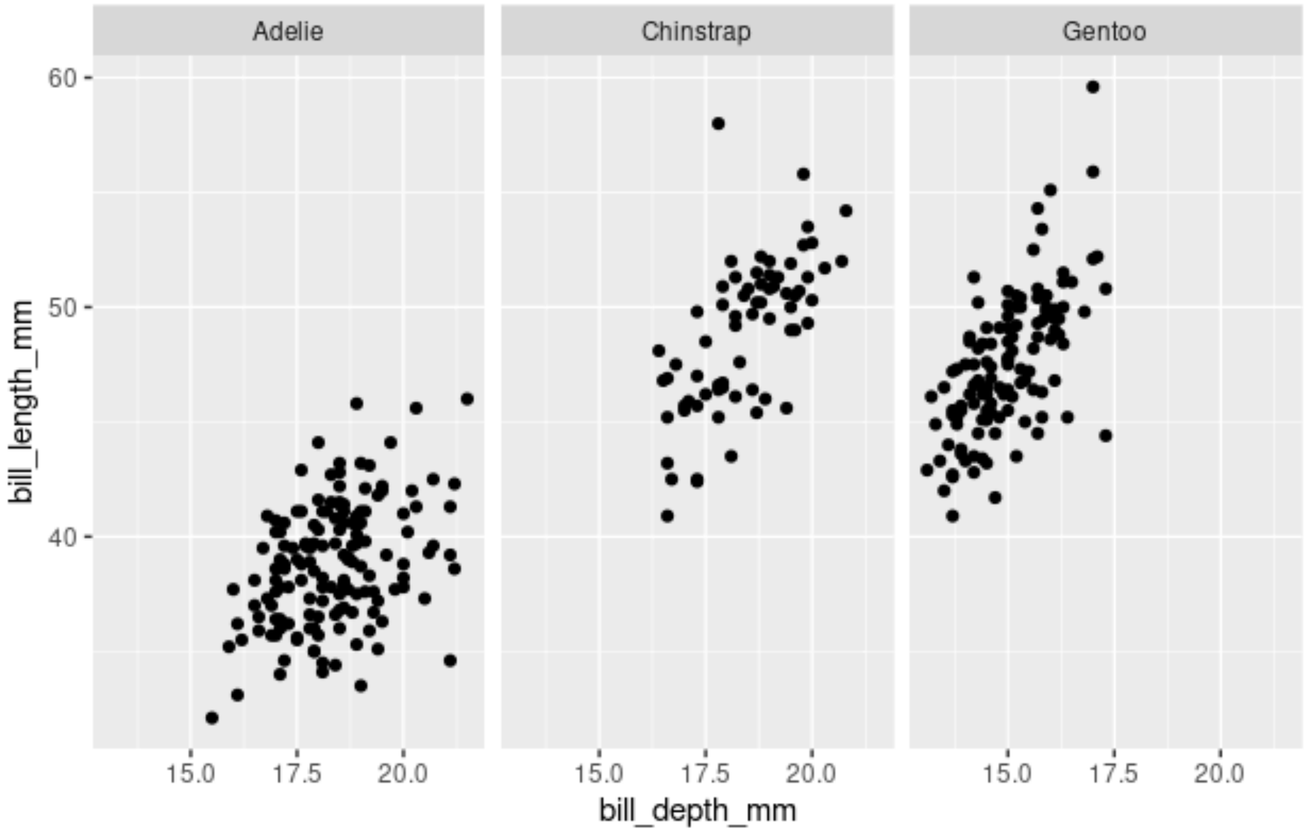
Specify **FACETS** or groups of data displayed in different plots.



Facets

`ggplot()` uses facets to group data into subplot. Using `facet_wrap()` with `vars()`, you can create subplots into specified number of rows and/or columns.

```
ggplot(penguins) +
  geom_point(aes(bill_depth_mm,
                 bill_length_mm)) +
  facet_wrap(vars(species), nrow = 1)
```



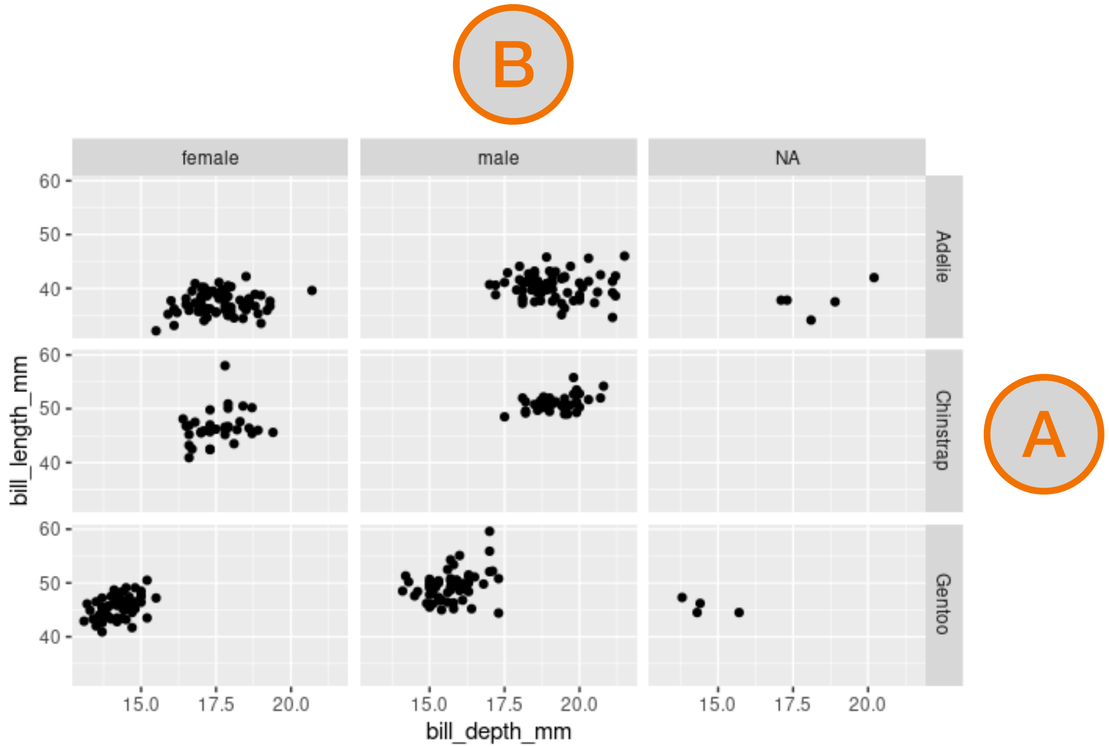
Facets

`facet_grid()` divides groups based on the variable on the *left* of ~ into different *rows* (A) and the variables on the *right* into different *columns* (B).

```
ggplot(penguins) +
  geom_point(aes(bill_depth_mm,
                 bill_length_mm)) +
  facet_grid(species ~ sex)
```

A

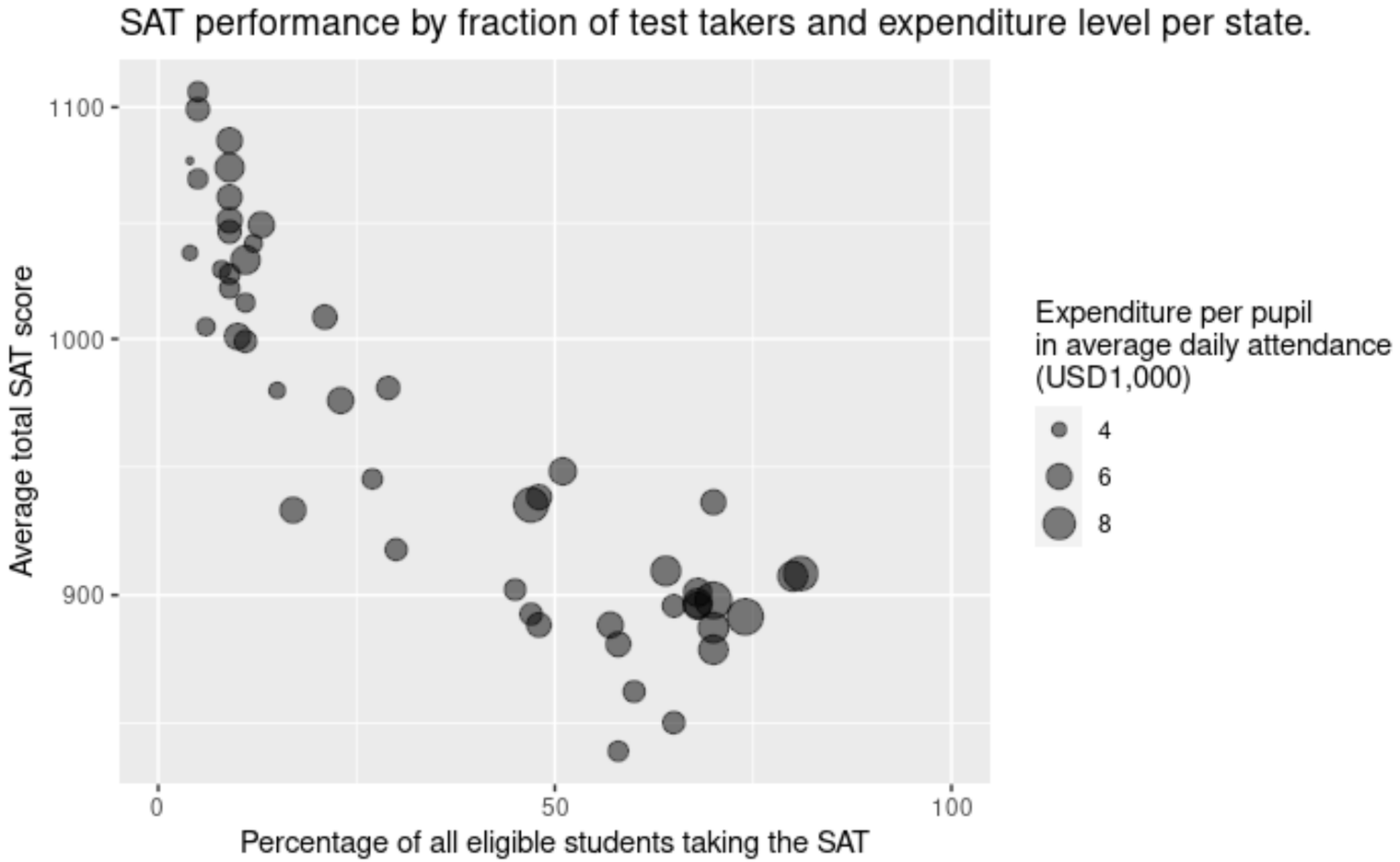
B



Exercises

Application Exercise

Reproduce this plot following the steps on the next slide.



Application Exercise

1. The data comes from `mosaicData` package. If you are using JupyterHub, it comes pre-installed. Otherwise, install the package first.

2. Load the library with `library("mosaicData")` then the data with `data(SAT)`. This loads the data into `SAT` object.

3. Create a scatter plot with by mapping `frac` to x-axis, `sat` to y-axis, and `expend` to size. Use `alpha` argument to make the points semi-transparent.

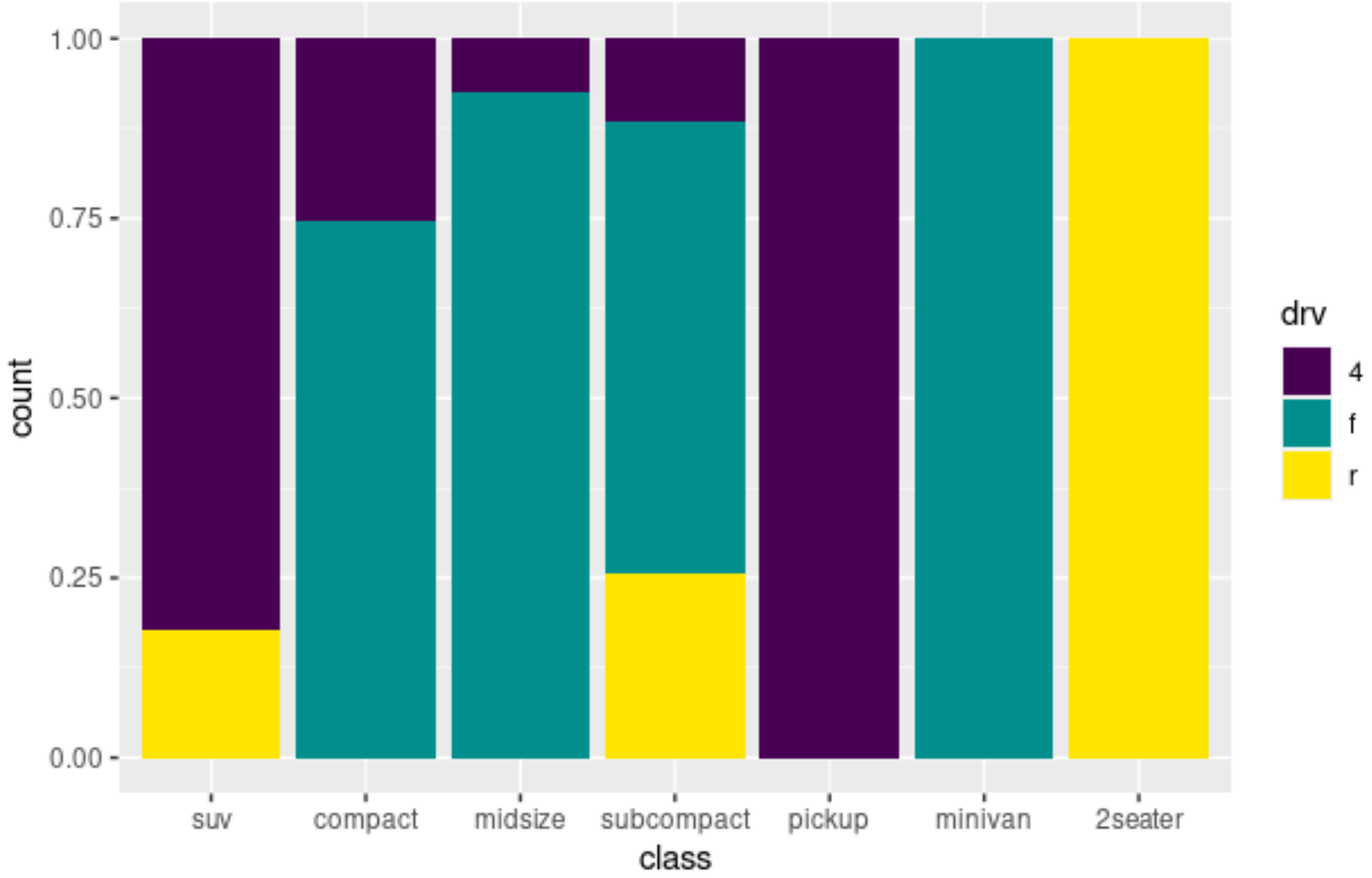
4. Use a `coord_*()` function to set the x-axis limit to 0 to 100.

5. Use `scale_*()` functions to transform the y-axis scale to the log scale and to place breaks only at 0, 50, and 100 along the x-axis.

6. Use `labs()` function to add appropriate labels.

Application Exercise

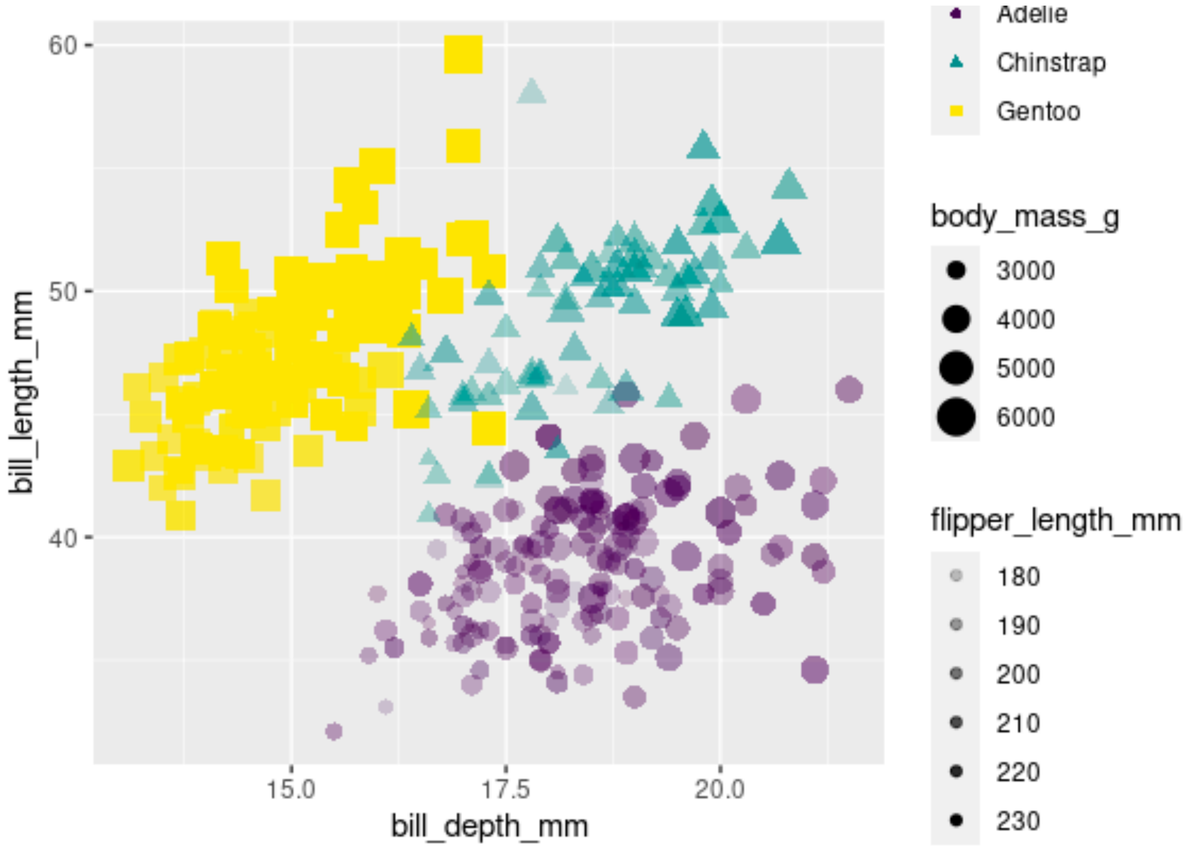
Using the `mpg` dataset, recreate the chart below.



More examples & Resources

Mapping Aesthetics

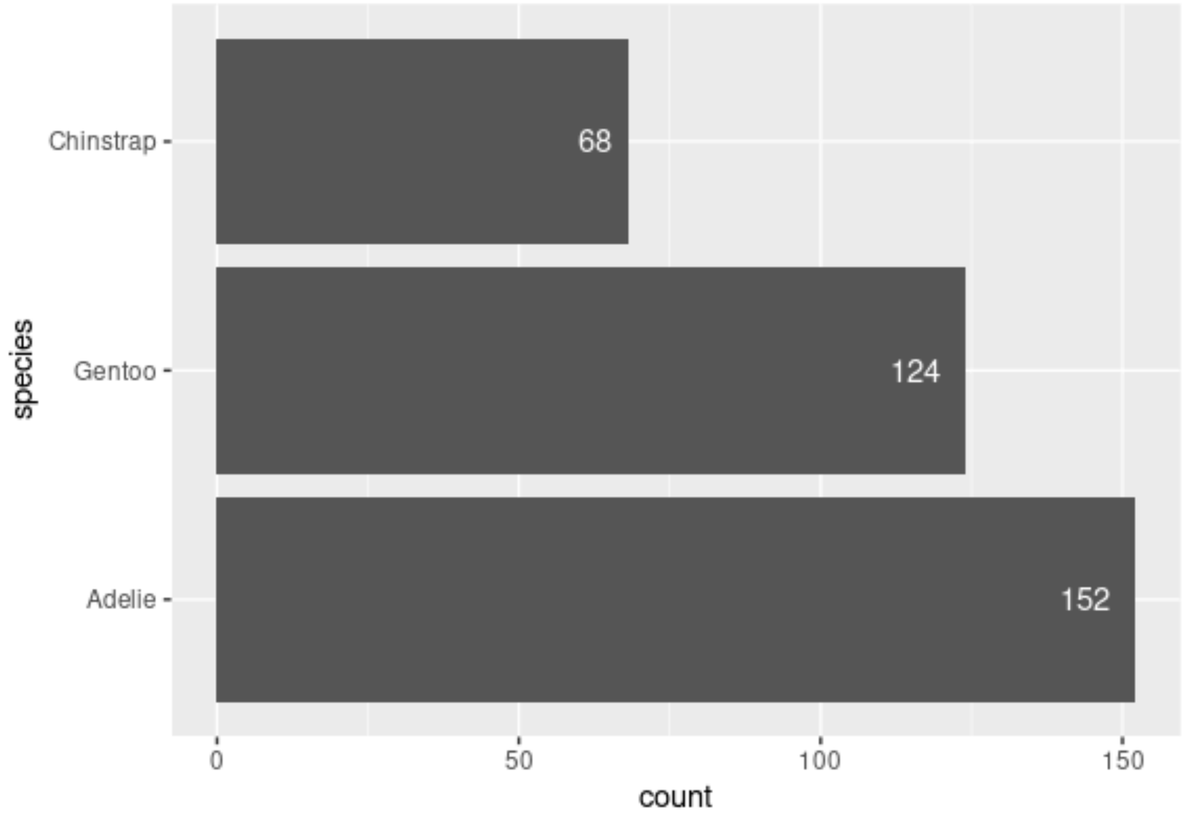
```
ggplot(penguins,
       aes(x = bill_depth_mm,
           y = bill_length_mm,
           colour = species,
           shape = species,
           size = body_mass_g,
           alpha = flipper_length_mm)) +
  geom_point() +
  scale_colour_viridis_d()
```



Labels & Annotations

You can also add text annotations using `geom_text()` or `geom_label()` functions.
`hjust` (`vjust`) argument can shift the label horizontally (vertically).

```
mutate(
  penguins,
  species = forcats::fct_infreq(species)) %>%
  ggplot(aes(x = species)) +
  geom_bar() +
  geom_text(
    aes(label = ..count..),
    stat = "count",
    hjust = 1.5, colour = "white") +
  coord_flip()
```



Resources

- **Chapter 3. Data visualisation, R for Data Science**
  https://r4ds.had.co.nz/data-visualisation.html
- **An Introduction to ggplot2, UC Business Analytics R Programming Guide**
  https://uc-r.github.io/ggplot_intro
- **Data Visualization with ggplot2 Cheat Sheet, RStudio**
  https://github.com/rstudio/cheatsheets/blob/master/data-visualization-2.1.pdf