Section 1

Learning Objectives

R Learning Objectives

1. Learn how to import data into R
2. Learn how to make a frequency table and a two-way frequency table in R
3. Learn how to make a bar chart in R
4. Learn how to find the five-number summary of a variable, and find a specific numeric summary (statistic) in R
5. Learn how to make a histogram in R
6. Learn how to make a box plot in R
7. Learn how to make side-by-side box plots in R

Statistical Learning Objectives

1. Understand when to make a bar chart
2. Be able to use these graphical summaries to discuss data
3. Understand when to make a bar chart versus a box plot or a histogram
4. Understand when to make a side-by-side box plot and how to use this type of comparison
5. Understand when to make a frequency/two-way frequency table versus a numerical summary
6. Be able to use these graphical and numerical summaries to discuss data

Functions covered in this lab

1. `read.csv()`
2. `head()`
3. `str()`
4. `table()`
5. `barplot()`
6. `summary()`
7. `hist()`
8. `min()`, `mean()`, `median()`, `max()`, `sd()`, `IQR()`
9. `boxplot()`

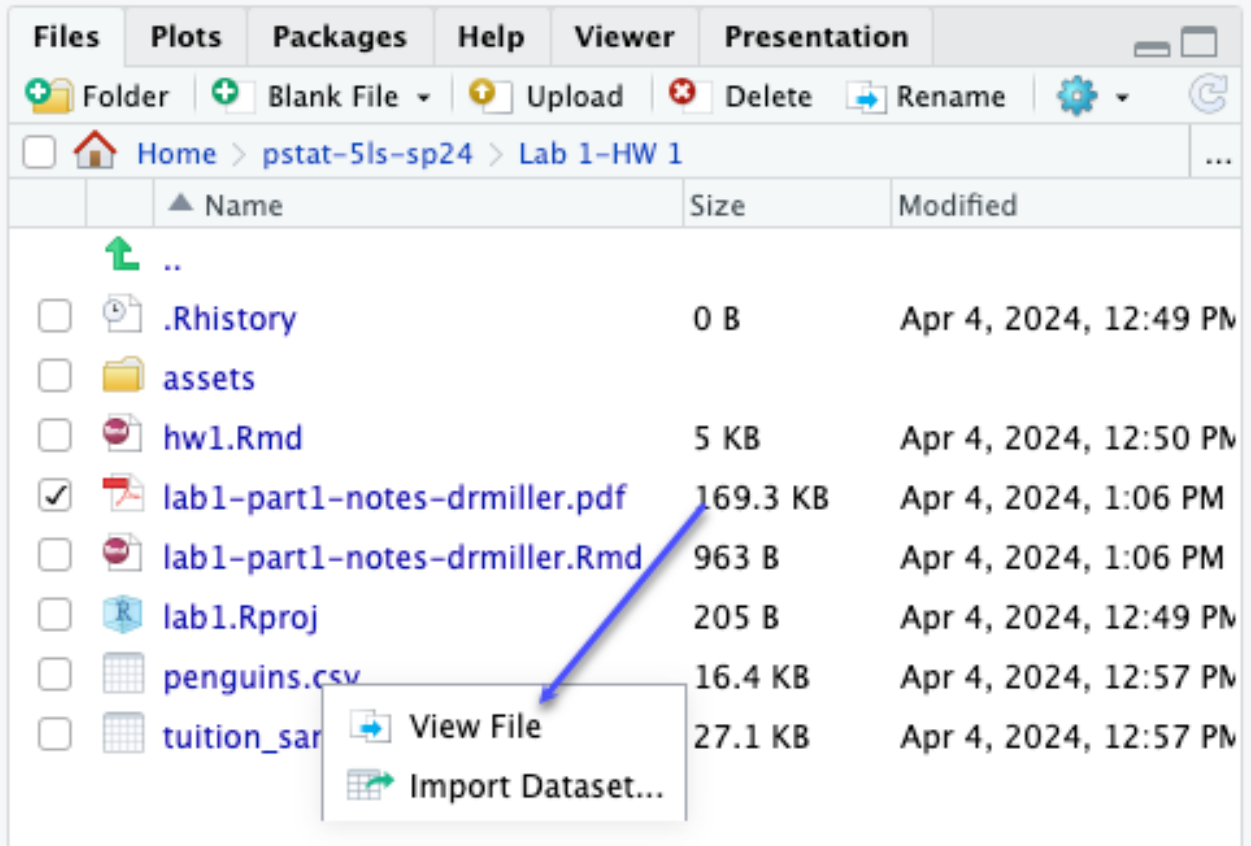
Section 2

Lab Tutorial

CSV files: a common way to store data

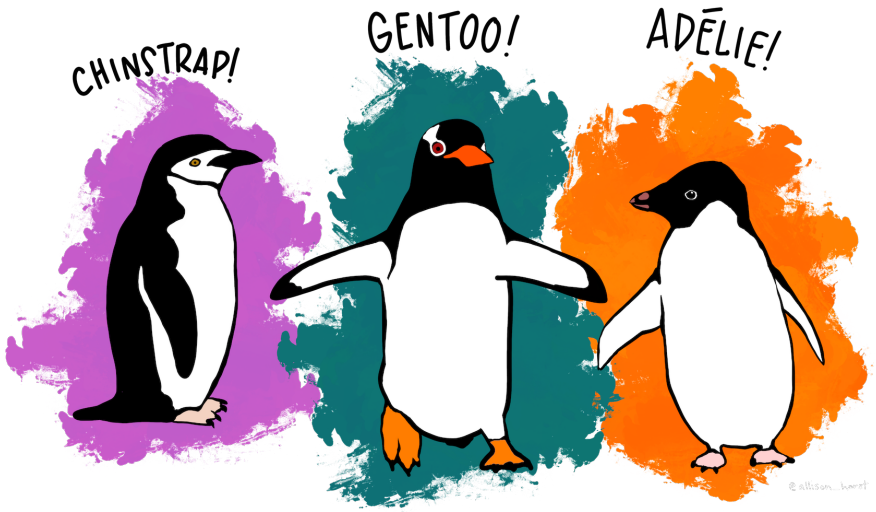
One common way to store data is to store it in a CSV file. CSV stands for "comma separated values".

Open the file "penguins.csv" from the files pane (lower right) to see what a .csv file looks like:



Palmer Penguins Data

We're going to start by working with a data set with data on 333 penguins collected from 3 islands in the Palmer Archipelago in Antarctica. Data were collected and made available by Dr. Kristen Gorman and the Palmer Station, Antarctica LTER, a member of the Long Term Ecological Research Network, and the data were prepared by Dr. Allison Horst.



Reading Data into R

We can **read** data into R using a function called `read.csv()`. The first argument to `read.csv()` is the name of a .csv file (here, `penguins.csv`), in quotes. We then store the results of `read.csv()` as an object called `penguins`.

```
penguins <- read.csv("penguins.csv", stringsAsFactors = TRUE)
```

The argument `stringsAsFactors = TRUE` has been added to the `read.csv()` function. This is important because we want to be able to distinguish between something called a *string*, which is a phrase or word, and a **factor**, which represents the levels of a categorical variable. Basically, by setting `stringsAsFactors = TRUE`, we are letting R know to expect that any words or phrases in the data actually relate to categorical variables.

Go ahead and run the `loadPenguins` chunk of your `lab1-notes.Rmd` markdown file, and verify that the `penguins` data is in your environment in the top right corner of your RStudio project.

Peeking at the Data

Let's see what's in the data. We can peek at the first few (6, specifically) rows of the data using the `head()` function:

```
head(penguins)
```

```
##   species    island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
## 1  Adelie Torgersen           39.1          18.7                181        3750
## 2  Adelie Torgersen           39.5          17.4                186        3800
## 3  Adelie Torgersen           40.3          18.0                195        3250
## 4  Adelie Torgersen           36.7          19.3                193        3450
## 5  Adelie Torgersen           39.3          20.6                190        3650
## 6  Adelie Torgersen           38.9          17.8                181        3625
##      sex year
## 1   male 2007
## 2 female 2007
## 3 female 2007
## 4 female 2007
## 5   male 2007
## 6 female 2007
```

Peeking at the Data

We read that line of code as "*head* of *penguins*". Remember that `penguins` is what we named our data set. We can see that `penguins` contains a number of *variables*, like `species`, `island`, and more.

Run the `tryIt1` chunk in your notes to peek at the first 6 rows of the penguins data file.

Data

Variable name	Description
`species`	Penguin species (Adélie, Chinstrap, and Gentoo)
`island`	Island in Palmer Archipelago, Antarctica, on which the penguin was observed (Biscoe, Dream, or Torgersen)
`bill_length_mm`	A number denoting bill length (in millimeters)
`bill_depth_mm`	A number denoting bill depth (in millimeters)

Data Continued

Variable name	Description
`flipper_length_mm`	A whole number denoting flipper length (in millimeters)
`body_mass_g`	A whole number denoting penguin body mass (in grams)
`sex`	Penguin sex (female, male)
`year`	Study year (2007, 2008, 2009)

Another Way

We can also peek at the data using a function called `str()` (pronounced "stir", short for "structure"):

```
str(penguins)
```

```
## 'data.frame':    333 obs. of  8 variables:
##  $ species          : Factor w/ 3 levels "Adelie","Chinstrap",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ island           : Factor w/ 3 levels "Biscoe","Dream",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ bill_length_mm   : num  39.1 39.5 40.3 36.7 39.3 38.9 39.2 41.1 38.6 34.6 ...
##  $ bill_depth_mm    : num  18.7 17.4 18 19.3 20.6 17.8 19.6 17.6 21.2 21.1 ...
##  $ flipper_length_mm: int  181 186 195 193 190 181 195 182 191 198 ...
##  $ body_mass_g      : int  3750 3800 3250 3450 3650 3625 4675 3200 3800 4400 ...
##  $ sex              : Factor w/ 2 levels "female","male": 2 1 1 1 2 1 2 1 2 2 ...
##  $ year             : int  2007 2007 2007 2007 2007 2007 2007 2007 2007 2007 ...
```

Run the `tryIt2` chunk in your notes examine the structure of the penguins data file.

Frequency Tables

Let's start with the `species` variable. Is this a categorical or numeric variable? How do you know?

To make a frequency table of a categorical variable, we use the `table()` function. Try this code out on your `lab1-notes` file in the `tryit3` code chunk of your notes file. Note that code goes **in the middle** of the chunk and not on the first or last line of the chunk.

```
table(penguins$species)
```

```
##
##    Adelie Chinstrap    Gentoo
##       146        68       119
```

Frequency Tables continued

So, there are 119 Gentoo penguins in the data.

Notice that inside the table function, we have something that looks a little weird. We wrote `penguins$species`. This is how we tell R to use the `species` variable **inside the data frame** `penguins`. The dollar sign ($) tells R to look inside the data frame `penguins` for the column called `species`.

Frequency Table Common Error

It's very important that you tell R *which data frame* the variable you're interested in is from. Let's see what happens when we don't:

```
table(species)
```

```
## Error in eval(expr, envir, enclos): object 'species' not fo
```

Notice that we get an error message here, stating that the "object 'species' not found".

Two-Way Frequency Tables (Contingency Tables)

We can also make "two-way" frequency tables (sometimes called "contingency tables") to summarize counts for two categorical variables. Try this in the `tryit4` code chunk in your notes file.

```
table(penguins$species, penguins$island)
```

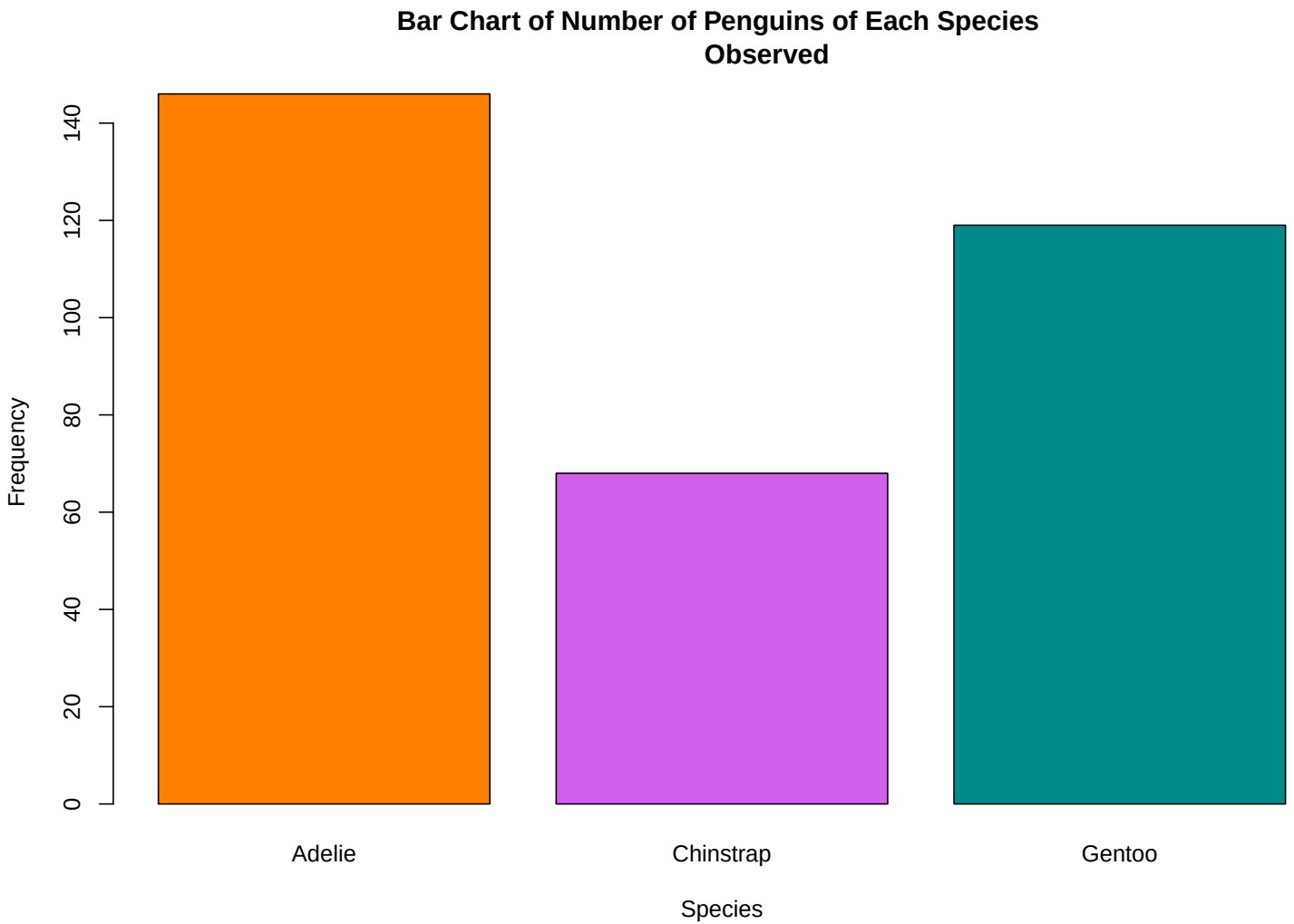
```
##
##               Biscoe Dream Torgersen
##    Adelie         44    55         47
##    Chinstrap       0    68          0
##    Gentoo        119     0          0
```

Data is **R**eally **C**ool, so the first variable you give to `table()` is in the **r**ows of the table, and the second is in the **c**olumns.

Bar Charts in R

Let's explore our `penguins` data by making a plot that will help us visualize a categorical variable. We'll start by looking at the number of penguins observed of each species. Run the `tryit5` code chunk in your `lab1-notes` file to have R create your bar chart.

Bar Chart of species in penguins



Bar Chart Code for species in penguins

```
barplot(table(penguins$species),
     xlab = "Species",
     ylab = "Frequency",
     main = "Bar Chart of Number of Penguins of Each Species
            Observed",
     col = c("darkorange1", "mediumorchid2", "darkcyan"))
```

Bar Charts Code for `species` in `penguins` continued

Notice that we included the table from earlier in our code!

Also notice that we included some arguments, such as

- `xlab`, the label on the x (or horizontal) axis
- `ylab`, the label on the y (or vertical) axis
- `main`, the title of the graph, and
- a nice way to add some fun to an otherwise boring plot - `col` for colors. We have three species of penguins, so we picked out three colors that we thought best represented each species. There are lots and lots of colors that you can try. There is a color palette "cheat sheet" from UCSB's own National Center for Ecological Analysis and Synthesis posted on Canvas if you want to check it out.

How to Find Help in R

R has built-in "documentation" for every function. If you want to find that documentation, you can Google it, but that takes too long. So it's better to use R's built in help! In the R console, just type a question mark `?` followed by the name of the function you want help with, then hit enter. For example, `?barplot` will bring up the documentation for the `barplot()` function.

Let's try this in the `tryit6` code chunk in your notes file.

At the end of the help file you *may* find an example of how to use the function. These examples are generally super helpful! You can directly run them using the `example()` function – e.g., `example(barplot)`.

The most useful feature of help in R is a list of a function's arguments and a quick explanation of what each argument does. You may not be able to fully understand some of the terms in the documentation just yet, but try it out and your TA will be able to help!

Numerical Summaries

Let's start with the `flipper_length_mm` variable. Is this a categorical or quantitative variable? How do you know?

We can use R to summarize data numerically. We'll use the `summary()` function to do that for a given variable. Here, we'll summarize the `flipper_length_mm` variable, which is the length of the penguins' flippers (in millimeters). Try this code out on in the `tryit7` code chunk.

```
summary(penguins$flipper_length_mm)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     172     190     197     201     213     231
```

Numerical Summaries

You might have noticed that the `summary()` function doesn't give you the standard deviation of the variable. To get the standard deviation, use the `sd()` function in addition to the `summary()` function.

Summarize the `flipper_length_mm` variable again adding the code to get the standard deviation as well. Try this code out on in the `tryit8` code chunk.

```
summary(penguins$flipper_length_mm)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     172     190     197     201     213     231
```

```
sd(penguins$flipper_length_mm)
```

```
## [1] 14.01577
```

Number Summaries continued

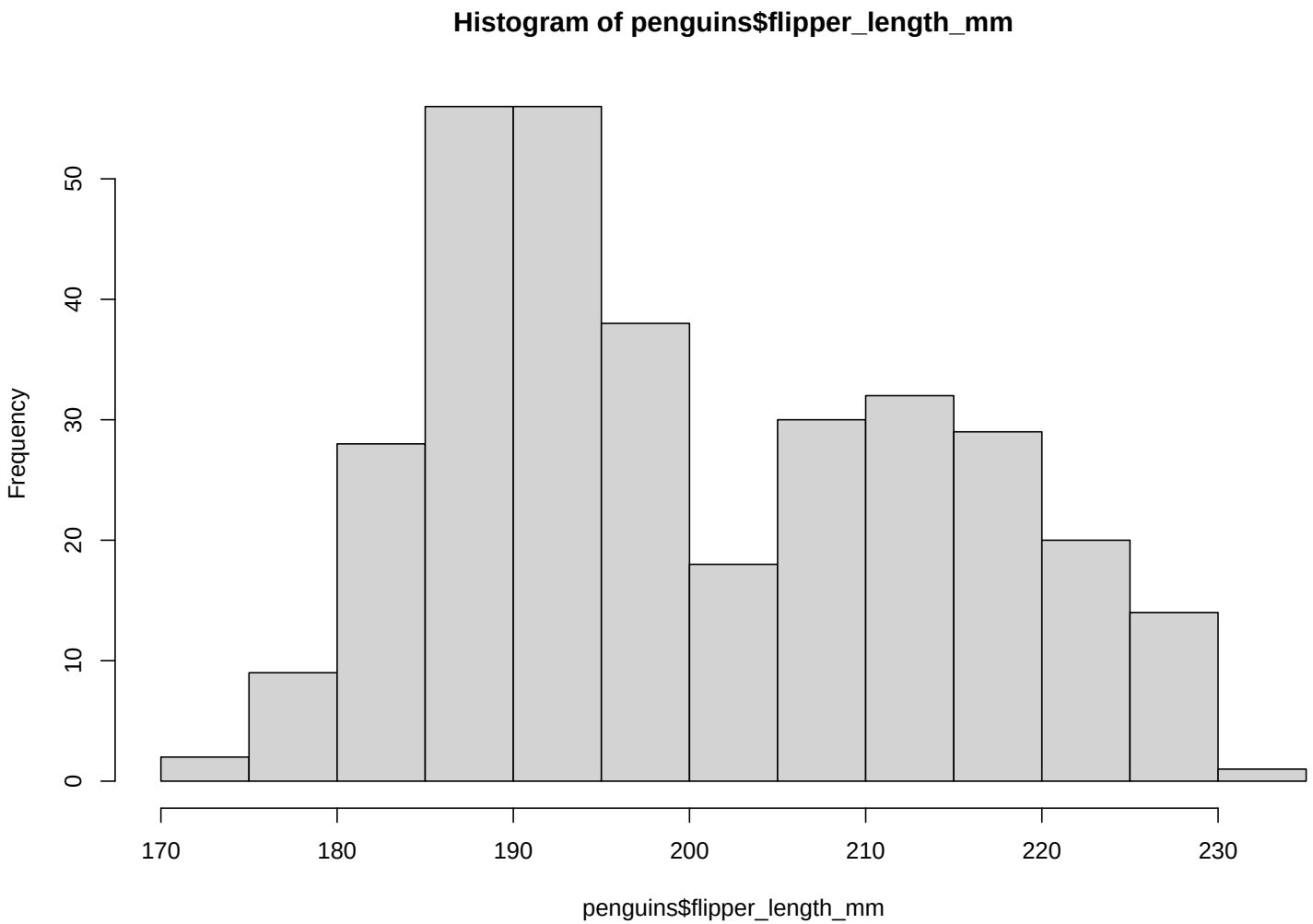
You can always get just the one numerical summary you're looking for using the function for that specific summary. Try these out in the `tryit9` code chunk.

```
min(penguins$flipper_length_mm)
mean(penguins$flipper_length_mm)
median(penguins$flipper_length_mm)
max(penguins$flipper_length_mm)
sd(penguins$flipper_length_mm)
IQR(penguins$flipper_length_mm)
```

Histograms in R

One type of graphical display for a quantitative variable is a histogram. Histograms in R are also pretty easy – you just use the `hist()` function.

Histogram of Flipper Length of Penguins



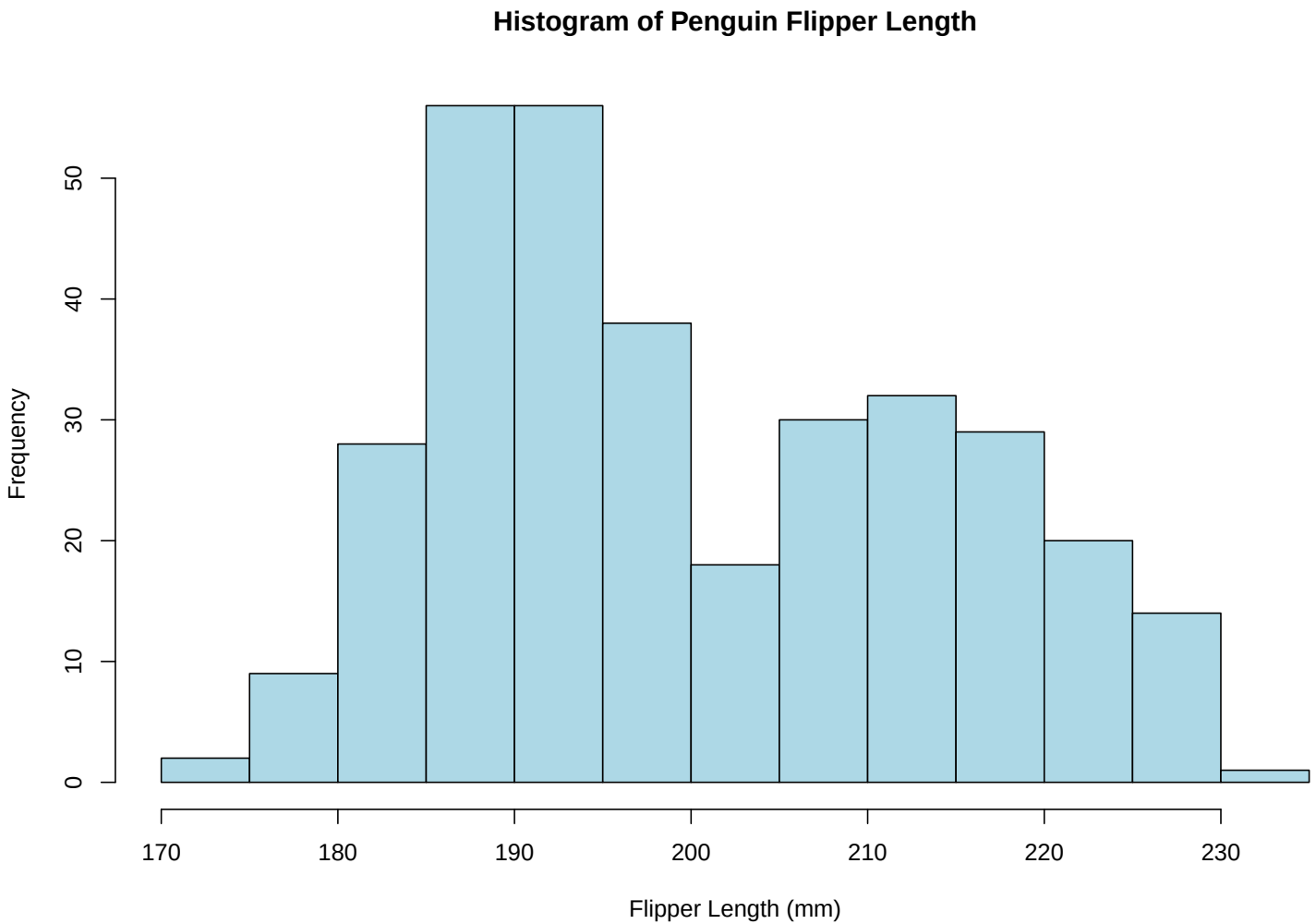
Don't Forget Labels and Titles!

```
hist(penguins$flipper_length_mm)
```

So here we've got a histogram. Notice that we didn't provide the `main`, `xlab`, and `ylab` arguments that we'd normally use for a plot title and an axis label, but R still gave us a title and labels. This is nice, but the labels are *horrible*: nobody (other than you) knows what `penguin$flipper_length_mm` means, so we don't want to use that as a title or axis label.

The moral of the story is to **always provide `main`, `xlab`, and `ylab` arguments when making a plot!**

That Same Histogram with Labels and a Title



Histogram Code

Try this code out in the `tryit10` code chunk. Watch out for the dreaded typos! If you get an error message, try to debug it yourself before asking for help!

```
hist(penguins$flipper_length_mm,
     main = "Histogram of Penguin Flipper Length",
     xlab = "Flipper Length (mm)",
     col = "lightblue")
```

Describing Histograms

Recall from lecture that we describe distributions by addressing four aspects:

1. Shape (number of modes + symmetry or lack thereof)
2. Center
3. Spread/Variability
4. Outliers

A handy mnemonic to remember what to comment about is **SOCS**:

Shape **O**utliers **C**enter **S**pread

Note: Be sure to mention whether there are or are not outliers. Not saying anything doesn't let us know that you know to check for outliers.

Describing Histograms

Describe the distribution of flipper lengths.

Describing Histograms

Do you think that the mean is the best measure of center for the flipper lengths? Why or why not?

Describing Histograms

Describe the variability (spread) of the flipper lengths.

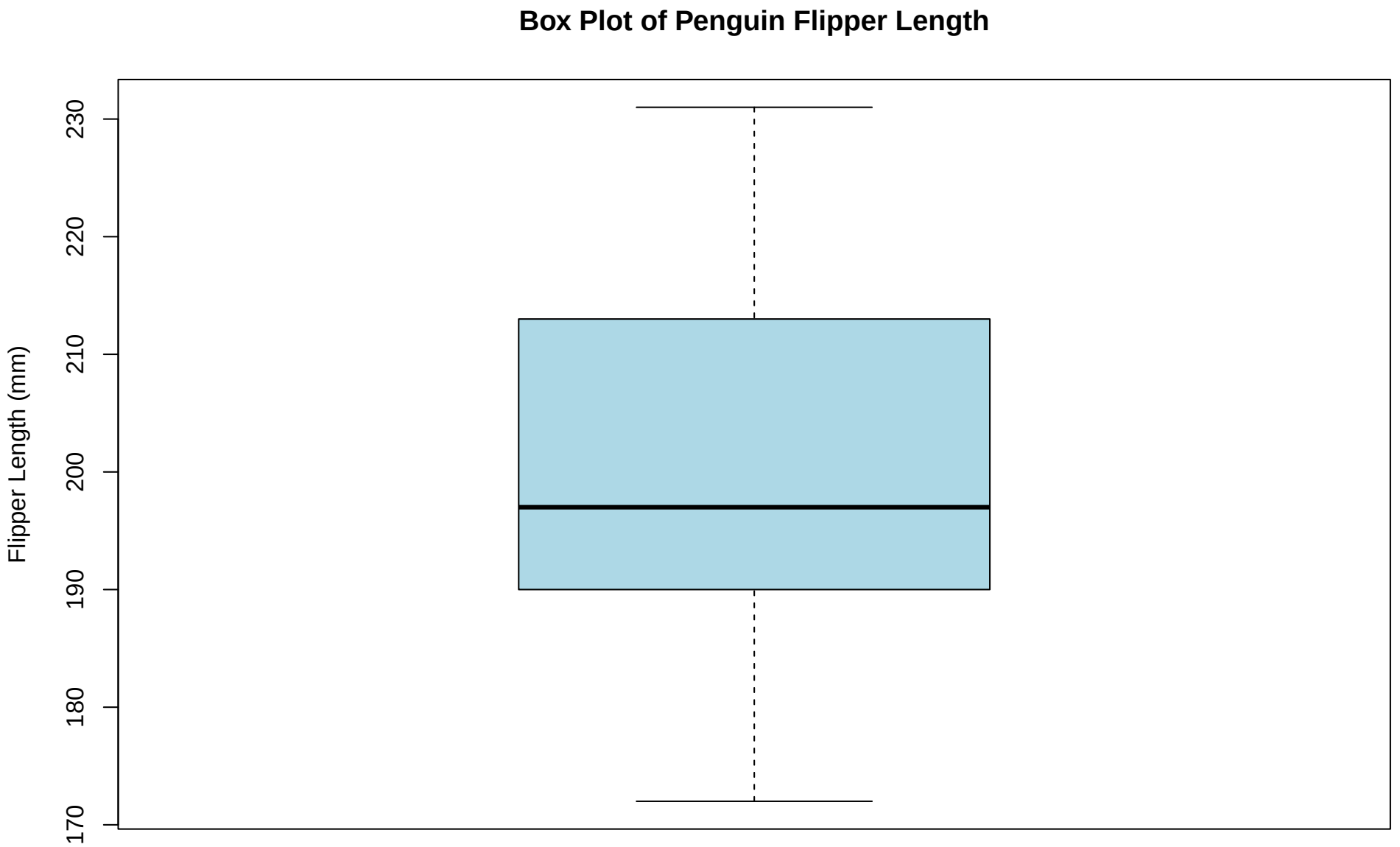
Describing Histograms

Are there any outliers or other unusual features that you'd like to mention about the distribution of flipper lengths?

Box Plots in R

Another type of graphical display for a quantitative variable is a box plot. The command for making a box plot in R is pretty simple: it's just `boxplot()`. To make a box plot of a single variable, just give R the name of the data set, a dollar sign ($), then the name of the variable. Also provide the arguments `main` and `ylab` for a plot title and y-axis label.

Box Plot of the Flipper Length of Penguins



Code for the Box Plot of the Flipper Length of Penguins

Try this code out yourself in the `tryit11` code chunk. Watch out for the dreaded typos! If you get an error message, try to debug it yourself before asking for help!

```
boxplot(penguins$flipper_length_mm,
        main = "Box Plot of Penguin Flipper Length",
        ylab = "Flipper Length (mm)",
        col = "lightblue")
```

Describing Box Plots

True or False:

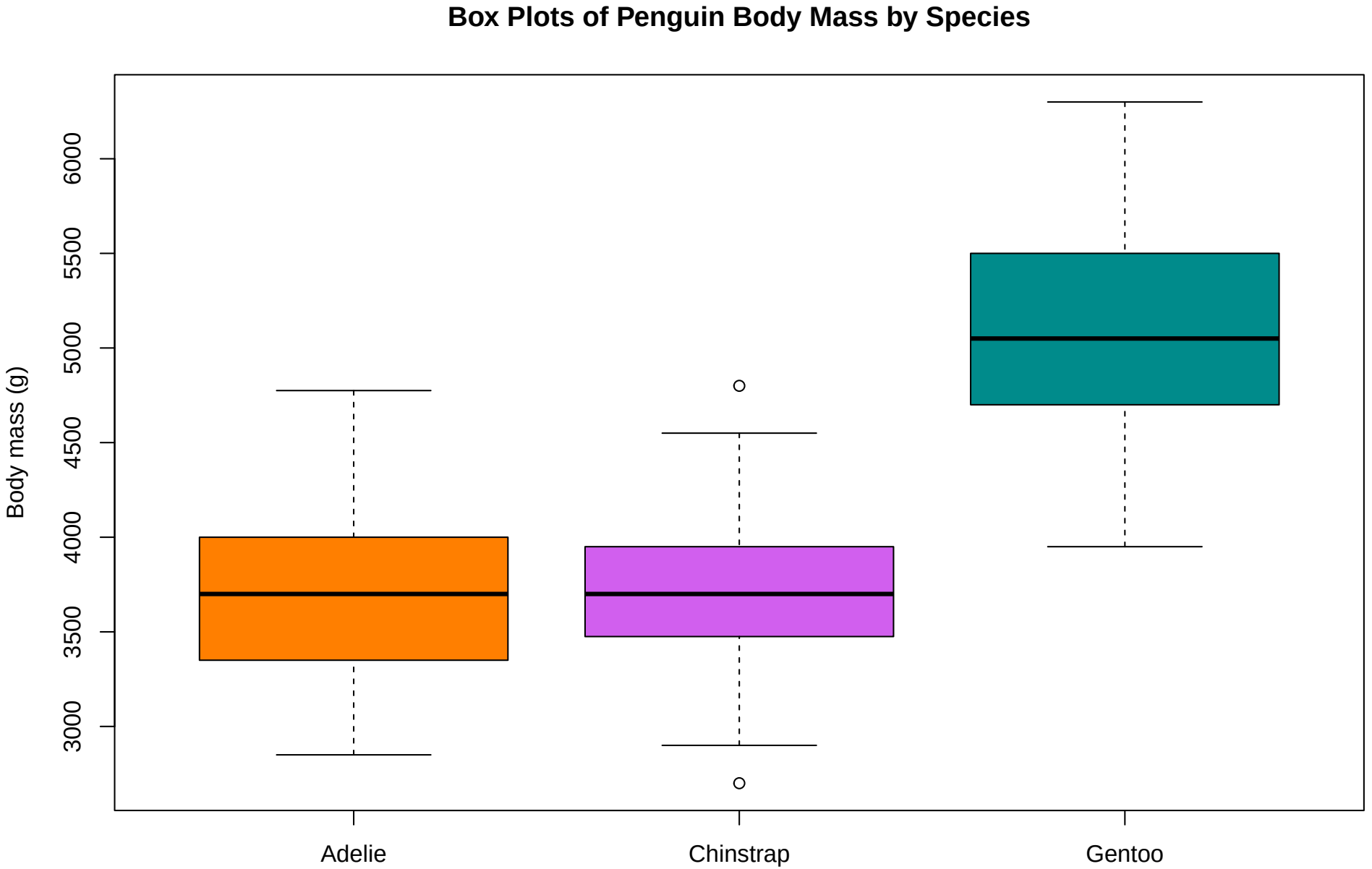
The box plot of flipper lengths appears to be unimodal and symmetric.

Side-by-side Box Plots

Sometimes we're interested in comparing two or more groups using "side-by-side" box plots. We can compare the different species of penguins' body masses in this way, still using the `box plot` function. We have provided the code in the `tryit12` code chunk.

```
boxplot(penguins$body_mass_g ~ penguins$species,
        main = "Box Plots of Penguin Body Mass by Species",
        ylab = "Body mass (g)",
        xlab = "Species",
        col = c("darkorange1", "mediumorchid2", "darkcyan"))
```

Side-by-side Box Plots Continued



Penguin Body Mass By Species

Does it appear that a penguin's body mass is related to its species, for the penguins in Palmer Archipelago? Why or why not?

Another Way to Code Side-by-side Box Plots

Another way to get the same side-by-side box plots the to specify the variables themselves and adding in the code `data = penguins`.

It's up to you which of the two ways to specify the variables you use. We all have different things we prefer in coding. Try out some different options to find your style!

```
boxplot(body_mass_g ~ species, data = penguins,
        main = "Box Plots of Penguin Body Mass by Species",
        ylab = "Body mass (g)",
        xlab = "Species",
        col = c("darkorange1", "mediumorchid2", "darkcyan"))
```

Another Side-by-side Box Plot

Now, in the `tryit13` code chunk, you will make a side-by-side box plot of the numeric variable `flipper_length_mm` by the categorical variable `island` in the `penguins` data.

Another Side-by-side Box Plot