Week 1: Is My Data Clean? Exploring, Diagnosing, and Visualizing Problems

Ronnie Steinitz

2025-09-18

Skills Learning – Lecture

We will be working using R Markdown files to save our code or script. An R Markdown file (saved as **'filename.Rmd'**) is a special type of document that lets you combine your R code, results (like tables and plots), and your written explanations all in one place.

It's especially useful because it creates a clear, reproducible record of your analysis that you (or others) can easily follow and re-run later. Unlike plain code files, R Markdown helps you communicate your work like a report, not just as raw code.

An Rmd consists of intermingled prose (narratives) and code. There are two types of code in an Rmd document: code chunks and inline R code:

Rmarkdown structure - code chunk

```
x <- 5  # radius of a circle
```

Now, we can also include the code (or data created), in the body of the document. This is called *inline code*. The radius of the circle is **5.**

> Useful commands:
> - To "comment out" (make text grey and not run the code): Ctrl + Shift + C
> - To create a new code chunk: Ctrl + Alt + i
> For more shortcuts, you can follow this link!

Alright, let's get started.

0. Load Required Packages

> *Packages* are like apps; functions are like tools inside the app.

Packages in R are like apps you install on your computer.
Just like you install Microsoft Word to write documents or Excel to do spreadsheets, in R, you install packages to help you do specific types of tasks — like making graphs (`ggplot2`), cleaning messy data (`janitor`), or analyzing statistics (`stats`).

Functions are the tools inside those apps.
For example, in Microsoft Word, once installed, you can use tools like bold, spellcheck, or insert image. In an R package like ggplot2, you get tools (functions) like geom_point() to make scatterplots or labs() to add axis labels.

Example: Think of `tidyverse` as the R version of Microsoft Office: it's a suite of tools (packages like `dplyr`, `ggplot2`, `readr`, etc.).

Each package comes with specialized functions — like `filter()` from `dplyr` to filter rows, or `read_csv()` from `readr` to import data.

```
# install.package("tidyverse")
# install.package("janitor")

# Load tidyverse for data wrangling and visualization
library(tidyverse)
```

In this course, we will use the `penguins` dataset from the `palmerpenguins` package, which is an innate dataset that comes with R. These data contain physical measurements, species info, and other metadata on three penguin species in Antarctica.

The `penguins` data included in R is fairly clean. We will use the `penguins_raw` data, which includes errors, miscategorizations, and missing data. The idea is that it mimics the type of data you might work with in your research.

Although you can download it directly from R, we will practice importing the *.csv* file from your working directory.

1. Load and Preview the Dataset

```
# check that you are in the correct working directory, the R-Ecology_Workshop folder
getwd()
```

```
## [1] "/Users/rsteinitz/Documents/github/R Data Analysis Course"
```

```
# If not, set your working directory to the correct folder (e.g., R-Ecology-Workshop), where all of you
# setwd("/Users/rsteinitz/Documents/github/R Data Analysis Course")

# Load dataset
data_raw <- read_csv("Week 1/Palmer Penguins Raw.csv")
```

```
## Rows: 344 Columns: 17
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (10): studyName, Species, Region, Island, Stage, Individual ID, Clutch C...
## dbl  (7): Sample Number, Bill Length (mm), Bill Depth (mm), Flipper Length (...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# Preview the structure
glimpse(data_raw, width = 80) # the 'width' command just limits how much is printed in each line of inf
```

```
## Rows: 344
## Columns: 17
## $ studyName             <chr> "PAL0708", "PAL0708", "PAL0708", "PAL0708", "PAL~
## $ `Sample Number`       <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 1~
## $ Species               <chr> "Adelie Penguin (Pygoscelis adeliae)", "Adelie P~
## $ Region                <chr> "Anvers", "Anvers", "Anvers", "Anvers", "Anvers"~
## $ Island                <chr> "Torgersen", "Torgersen", "Torgersen", "Torgerse~
## $ Stage                 <chr> "Adult, 1 Egg Stage", "Adult, 1 Egg Stage", "Adu~
## $ `Individual ID`       <chr> "N1A1", "N1A2", "N2A1", "N2A2", "N3A1", "N3A2", ~
## $ `Clutch Completion`   <chr> "Yes", "Yes", "Yes", "Yes", "Yes", "Yes", "No", ~
## $ `Date Egg`            <chr> "11/11/07", "11/11/07", "11/16/07", "11/16/07", ~
## $ `Bill Length (mm)`    <dbl> 39.1, 39.5, 40.3, NA, 36.7, 39.3, 38.9, 39.2, 34~
## $ `Bill Depth (mm)`     <dbl> 18.7, 17.4, 18.0, NA, 19.3, 20.6, 17.8, 19.6, 18~
## $ `Flipper Length (mm)` <dbl> 181, 186, 195, NA, 193, 190, 181, 195, 193, 190,~
## $ `Body Mass (g)`       <dbl> 3750, 3800, 3250, NA, 3450, 3650, 3625, 4675, 34~
## $ Sex                   <chr> "MALE", "FEMALE", "FEMALE", NA, "FEMALE", "MALE"~
## $ `Delta 15 N (o/oo)`   <dbl> NA, 8.94956, 8.36821, NA, 8.76651, 8.66496, 9.18~
## $ `Delta 13 C (o/oo)`   <dbl> NA, -24.69454, -25.33302, NA, -25.32426, -25.298~
## $ Comments              <chr> "Not enough blood for isotopes.", NA, NA, "Adult~
```

```r
# glimpse() is a cleaner display that str(), but does the same
str(data_raw)
```

```
## spc_tbl_ [344 x 17] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ studyName          : chr [1:344] "PAL0708" "PAL0708" "PAL0708" "PAL0708" ...
##  $ Sample Number      : num [1:344] 1 2 3 4 5 6 7 8 9 10 ...
##  $ Species            : chr [1:344] "Adelie Penguin (Pygoscelis adeliae)" "Adelie Penguin (Pygoscelis
##  $ Region             : chr [1:344] "Anvers" "Anvers" "Anvers" "Anvers" ...
##  $ Island             : chr [1:344] "Torgersen" "Torgersen" "Torgersen" "Torgersen" ...
##  $ Stage              : chr [1:344] "Adult, 1 Egg Stage" "Adult, 1 Egg Stage" "Adult, 1 Egg Stage" "
##  $ Individual ID      : chr [1:344] "N1A1" "N1A2" "N2A1" "N2A2" ...
##  $ Clutch Completion  : chr [1:344] "Yes" "Yes" "Yes" "Yes" ...
##  $ Date Egg           : chr [1:344] "11/11/07" "11/11/07" "11/16/07" "11/16/07" ...
##  $ Bill Length (mm)   : num [1:344] 39.1 39.5 40.3 NA 36.7 39.3 38.9 39.2 34.1 42 ...
##  $ Bill Depth (mm)    : num [1:344] 18.7 17.4 18 NA 19.3 20.6 17.8 19.6 18.1 20.2 ...
##  $ Flipper Length (mm): num [1:344] 181 186 195 NA 193 190 181 195 193 190 ...
##  $ Body Mass (g)      : num [1:344] 3750 3800 3250 NA 3450 ...
##  $ Sex                : chr [1:344] "MALE" "FEMALE" "FEMALE" NA ...
##  $ Delta 15 N (o/oo)  : num [1:344] NA 8.95 8.37 NA 8.77 ...
##  $ Delta 13 C (o/oo)  : num [1:344] NA -24.7 -25.3 NA -25.3 ...
##  $ Comments           : chr [1:344] "Not enough blood for isotopes." NA NA "Adult not sampled." ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   studyName = col_character(),
##   ..   `Sample Number` = col_double(),
##   ..   Species = col_character(),
##   ..   Region = col_character(),
##   ..   Island = col_character(),
##   ..   Stage = col_character(),
##   ..   `Individual ID` = col_character(),
```

```
##   ..   `Clutch Completion` = col_character(),
##   ..   `Date Egg` = col_character(),
##   ..   `Bill Length (mm)` = col_double(),
##   ..   `Bill Depth (mm)` = col_double(),
##   ..   `Flipper Length (mm)` = col_double(),
##   ..   `Body Mass (g)` = col_double(),
##   ..   Sex = col_character(),
##   ..   `Delta 15 N (o/oo)` = col_double(),
##   ..   `Delta 13 C (o/oo)` = col_double(),
##   ..   Comments = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
# notice the number of rows
# the number of columns
# and then the column names and type of data in each (which is determined when R reads in the dataset)

# Print the first few rows
head(data_raw)
```

```
## # A tibble: 6 x 17
##   studyName `Sample Number` Species          Region Island Stage `Individual ID`
##   <chr>               <dbl> <chr>            <chr>  <chr>  <chr> <chr>
## 1 PAL0708                 1 Adelie Penguin ~ Anvers Torge~ Adul~ N1A1
## 2 PAL0708                 2 Adelie Penguin ~ Anvers Torge~ Adul~ N1A2
## 3 PAL0708                 3 Adelie Penguin ~ Anvers Torge~ Adul~ N2A1
## 4 PAL0708                 4 Adelie Penguin ~ Anvers Torge~ Adul~ N2A2
## 5 PAL0708                 5 Adelie Penguin ~ Anvers Torge~ Adul~ N3A1
## 6 PAL0708                 6 Adelie Penguin ~ Anvers Torge~ Adul~ N3A2
## # i 10 more variables: `Clutch Completion` <chr>, `Date Egg` <chr>,
## #   `Bill Length (mm)` <dbl>, `Bill Depth (mm)` <dbl>,
## #   `Flipper Length (mm)` <dbl>, `Body Mass (g)` <dbl>, Sex <chr>,
## #   `Delta 15 N (o/oo)` <dbl>, `Delta 13 C (o/oo)` <dbl>, Comments <chr>
```

```
# Open the data in a new window
#View(data_raw)

# Let's clean up the column names, which will make it easier to index, or refer to them in functions
data <- janitor::clean_names(data_raw)

# what do they looks like now?
head(data)
```

```
## # A tibble: 6 x 17
##   study_name sample_number species             region island stage individual_id
##   <chr>              <dbl> <chr>               <chr>  <chr>  <chr> <chr>
## 1 PAL0708                1 Adelie Penguin (Py~ Anvers Torge~ Adul~ N1A1
## 2 PAL0708                2 Adelie Penguin (Py~ Anvers Torge~ Adul~ N1A2
## 3 PAL0708                3 Adelie Penguin (Py~ Anvers Torge~ Adul~ N2A1
## 4 PAL0708                4 Adelie Penguin (Py~ Anvers Torge~ Adul~ N2A2
## 5 PAL0708                5 Adelie Penguin (Py~ Anvers Torge~ Adul~ N3A1
## 6 PAL0708                6 Adelie Penguin (Py~ Anvers Torge~ Adul~ N3A2
## # i 10 more variables: clutch_completion <chr>, date_egg <chr>,
```

```
## #   bill_length_mm <dbl>, bill_depth_mm <dbl>, flipper_length_mm <dbl>,
## #   body_mass_g <dbl>, sex <chr>, delta_15_n_o_oo <dbl>, delta_13_c_o_oo <dbl>,
## #   comments <chr>
```

```
# you can also call up just a list of the column names
names(data)
```

```
##  [1] "study_name"        "sample_number"     "species"
##  [4] "region"            "island"            "stage"
##  [7] "individual_id"     "clutch_completion" "date_egg"
## [10] "bill_length_mm"    "bill_depth_mm"     "flipper_length_mm"
## [13] "body_mass_g"       "sex"               "delta_15_n_o_oo"
## [16] "delta_13_c_o_oo"   "comments"
```

```
# compare this to the raw data
names(data_raw)
```

```
##  [1] "studyName"           "Sample Number"       "Species"
##  [4] "Region"              "Island"              "Stage"
##  [7] "Individual ID"       "Clutch Completion"   "Date Egg"
## [10] "Bill Length (mm)"    "Bill Depth (mm)"     "Flipper Length (mm)"
## [13] "Body Mass (g)"       "Sex"                 "Delta 15 N (o/oo)"
## [16] "Delta 13 C (o/oo)"   "Comments"
```

> To run a **single** line of code: Ctrl + Enter. Runs the line of code your courses is on. *Give it a try!*
>
> `janitor::clean_names()` is a great tool to standardize column names

. . . which makes column names easier to use in code. Instead of having to type something like:
`data$"Flipper Length (mm)"`
. . . a "cleaned" version of this name would be:
`data$flipper_length_mm`

- The function `clean_names()` automatically replaces spaces, parentheses, slashes, and symbols with clean underscores. For example, `"Body Mass (g)"` becomes `body_mass_g` — easier to read, write, and autocomplete.
- Helps avoid bugs! You don't have to remember to use quotation marks around column names or worry about weird symbols breaking your code.

2. Diagnosing Data Types and Structure

We want to understand what type of variables we're working with:
- Are they numeric, character, or factors? - Are some columns misclassified?

```
# Check data types:
glimpse(data, width = 80)
```

```
## Rows: 344
## Columns: 17
## $ study_name        <chr> "PAL0708", "PAL0708", "PAL0708", "PAL0708", "PAL0708~
## $ sample_number     <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 1~
## $ species           <chr> "Adelie Penguin (Pygoscelis adeliae)", "Adelie Pengu~
## $ region            <chr> "Anvers", "Anvers", "Anvers", "Anvers", "Anvers", "A~
## $ island            <chr> "Torgersen", "Torgersen", "Torgersen", "Torgersen", ~
## $ stage             <chr> "Adult, 1 Egg Stage", "Adult, 1 Egg Stage", "Adult, ~
## $ individual_id     <chr> "N1A1", "N1A2", "N2A1", "N2A2", "N3A1", "N3A2", "N4A~
## $ clutch_completion <chr> "Yes", "Yes", "Yes", "Yes", "Yes", "Yes", "No", "No"~
## $ date_egg          <chr> "11/11/07", "11/11/07", "11/16/07", "11/16/07", "11/~
## $ bill_length_mm    <dbl> 39.1, 39.5, 40.3, NA, 36.7, 39.3, 38.9, 39.2, 34.1, ~
## $ bill_depth_mm     <dbl> 18.7, 17.4, 18.0, NA, 19.3, 20.6, 17.8, 19.6, 18.1, ~
## $ flipper_length_mm <dbl> 181, 186, 195, NA, 193, 190, 181, 195, 193, 190, 186~
## $ body_mass_g       <dbl> 3750, 3800, 3250, NA, 3450, 3650, 3625, 4675, 3475, ~
## $ sex               <chr> "MALE", "FEMALE", "FEMALE", NA, "FEMALE", "MALE", "F~
## $ delta_15_n_o_oo   <dbl> NA, 8.94956, 8.36821, NA, 8.76651, 8.66496, 9.18718,~
## $ delta_13_c_o_oo   <dbl> NA, -24.69454, -25.33302, NA, -25.32426, -25.29805, ~
## $ comments          <chr> "Not enough blood for isotopes.", NA, NA, "Adult not~
```

```
# if you want to know what type of data exists in a particular column, you can target it directly:
class(data$island) # character
```

```
## [1] "character"
```

```
class(data$date_egg) # character, not date!
```

```
## [1] "character"
```

```
class(data$flipper_length_mm) # numeric (also labeled as "double")
```

```
## [1] "numeric"
```

```
# how many entries (rows) per each category in this variable
table(data$species)
```

```
##
##       Adelie Penguin (Pygoscelis adeliae)
##                                       152
## Chinstrap penguin (Pygoscelis antarctica)
##                                        68
##         Gentoo penguin (Pygoscelis papua)
##                                       124
```

```
# Information within a column
unique(data$island) # we see that there are 3 unique values that exist in this column
```

```
## [1] "Torgersen" "Biscoe"    "Dream"
```

```r
unique(data$flipper_length_mm) # many!!! how do we know how many?
```

```
##  [1] 181 186 195  NA 193 190 180 182 191 198 185 197 184 194 174 189 187 183 172
## [20] 178 188 196 179 200 192 202 205 208 203 199 176 210 201 211 230 218 215 219
## [39] 209 214 216 213 217 221 222 220 207 225 224 231 229 223 212 228 226 206
```

```r
length(unique(data$flipper_length_mm))
```

```
## [1] 56
```

```r
# ok, back to the Islands!
unique(data$island) # Torgersen, Biscoe, Dream
```

```
## [1] "Torgersen" "Biscoe"    "Dream"
```

```r
# I want to know how many rows of data exist for each
count(data, island) # count the number of rows for each unique entry in column "island"
```

```
## # A tibble: 3 x 2
##   island        n
##   <chr>     <int>
## 1 Biscoe      168
## 2 Dream       124
## 3 Torgersen    52
```

```r
# ================================================================================
# Q: so what does 'Biscoe - 168' tell us about the data??
# A: ---> that there are 168 rows of data (out of 344) that were collected on Biscoe Island.
# ================================================================================

count(data, flipper_length_mm) # you can also do this for a variable that has a many more unique entrie
```

```
## # A tibble: 56 x 2
##    flipper_length_mm     n
##                <dbl> <int>
##  1               172     1
##  2               174     1
##  3               176     1
##  4               178     4
##  5               179     1
##  6               180     5
##  7               181     7
##  8               182     3
##  9               183     2
## 10               184     7
## # i 46 more rows
```

3. Missing Data: Detection and Summary

Missing data (NA values) are common in real-world datasets. Let's identify where they exist and how much is missing.

```
# When R reads in a dataset, cells without values will typically be converted to NAs (Not Available or
# Now let's see if we can find missing data!
is.na(data$sex)
```

```
##   [1] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE
##  [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
##  [49] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [73] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [85] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [97] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [109] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [121] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [133] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [145] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [157] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [169] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
## [181] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [193] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [205] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [217] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [229] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [241] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [253] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [265] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
## [277] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [289] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [301] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [313] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [325] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [337] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
# ============================================================================
# Q: how many rows with missing sex data are there?
# A: ---> hard to tell, right?
# ============================================================================


# So this is a good start, but your data set might have THOUSANDS of rows. So this is not an effective

# How many NAs per column?
colSums(is.na(data)) # summarizes a statistic across all columns
```

```
##        study_name     sample_number           species            region
##                 0                 0                 0                 0
##            island             stage     individual_id clutch_completion
```

```
##                 0                 0                 0                 0
##          date_egg    bill_length_mm     bill_depth_mm flipper_length_mm
##                 0                 2                 2                 2
##       body_mass_g               sex   delta_15_n_o_oo   delta_13_c_o_oo
##                 2                11                14                13
##          comments
##               290
```

```
# this is also an option per a specific column
sum(is.na(data$flipper_length_mm))
```

```
## [1] 2
```

```
# ================================================================================
# ***Q: TRY FINDING HOW MANY NAs ARE IN A DIFFERENT COLUMN!***
# ================================================================================
# (hint: you will need to call the names of the column, or view the dataset!)
sum(is.na(data$date_egg))
```

```
## [1] 0
```

```
# Summary stats also help us spot missing values, as well as other statistics.
summary(data)
```

```
##   study_name        sample_number      species             region
##  Length:344         Min.   :  1.00   Length:344         Length:344
##  Class :character   1st Qu.: 29.00   Class :character   Class :character
##  Mode  :character   Median : 58.00   Mode  :character   Mode  :character
##                     Mean   : 63.15
##                     3rd Qu.: 95.25
##                     Max.   :152.00
##
##     island             stage           individual_id      clutch_completion
##  Length:344         Length:344         Length:344         Length:344
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##    date_egg         bill_length_mm  bill_depth_mm   flipper_length_mm
##  Length:344         Min.   :32.10   Min.   :13.10   Min.   :172.0
##  Class :character   1st Qu.:39.23   1st Qu.:15.60   1st Qu.:190.0
##  Mode  :character   Median :44.45   Median :17.30   Median :197.0
##                     Mean   :43.92   Mean   :17.15   Mean   :200.9
##                     3rd Qu.:48.50   3rd Qu.:18.70   3rd Qu.:213.0
##                     Max.   :59.60   Max.   :21.50   Max.   :231.0
##                     NA's   :2       NA's   :2       NA's   :2
##   body_mass_g       sex             delta_15_n_o_oo  delta_13_c_o_oo
##  Min.   :2700   Length:344         Min.   : 7.632   Min.   :-27.02
##  1st Qu.:3550   Class :character   1st Qu.: 8.300   1st Qu.:-26.32
##  Median :4050   Mode  :character   Median : 8.652   Median :-25.83
```

```
##  Mean   :4202                     Mean   : 8.733   Mean   :-25.69
##  3rd Qu.:4750                     3rd Qu.: 9.172   3rd Qu.:-25.06
##  Max.   :6300                     Max.   :10.025   Max.   :-23.79
##  NA's   :2                        NA's   :14       NA's   :13
##    comments
##  Length:344
##  Class :character
##  Mode  :character
##
##
##
##
```

```
# ================================================================================
# ***Q: So, what does this mean in terms of how we collect and store data????***
# ***A: ---> we want to make sure to keep NAs as empty cells; not put in "0" or "." or "none". Otherwis
# ================================================================================

# ================================================================================
# ***Q: What is the maximum flipper length?
# ***A: ---> 231
# ================================================================================

# ================================================================================
# ***Q: What is the average, or mean bill depth?
# ***A: ---> 17.15
# ================================================================================

# ================================================================================
# ***Q: What is the range of values in column 'delta_15_n'
# ***A: ---> 7.63220 - 10.02544
# ================================================================================
# there's an easier way to find out range:
range(data$delta_15_n_o_oo)
```

```
## [1] NA NA
```

```
# uh oh! NA NA???
range(data$delta_15_n_o_oo, na.rm = TRUE)  # Some functions can't calculate a result if there are missi
```

```
## [1]  7.63220 10.02544
```

> **Suggestion:** In your own dataset, always start with a quick visual and summary check like this. It helps catch early problems in data entry or formatting.

4. Convert Column Types (if needed)

Some variables might be stored as character but should actually be treated as **factors** — which are used to represent *categorical variables* (like sex: "male" and "female", or species: "Adelie", "Gentoo", "Chinstrap"). Converting to factors helps R treat them properly in visualizations, summaries, or statistical models (e.g., group comparisons).

When do we keep them as characters? We keep variables as character when they are text labels or descriptions that are not meant to be grouped or modeled — for example, a column of penguin ID codes or free-text comments, like notes.

4.1 Check class

Let's check the "sex" column and convert it to a factor if needed:

```
# What type of data is stored in the "sex" column?
class(data$sex) # character
```

```
## [1] "character"
```

```
# Convert sex to factor (if not already)
data <- data %>%
  mutate(sex = as.factor(sex))

# so you can convert your data which was miscategorized to its correct class.

# We're also using something new here: the pipe operator (%>%). You can read it as "and then." It allow
```

Shortcut for pipes: Ctrl + Shift + M

R provides various functions for converting between these data types, typically prefixed with `as.`:
`as.numeric()` - Converts to numeric
`as.integer()` - Converts to integer
`as.logical()` - Converts to logical
`as.character()` - Converts to character
`as.factor()` - Converts to factor
`as.Date()` - Converts to Date

5. Basic Visualizations for Exploration

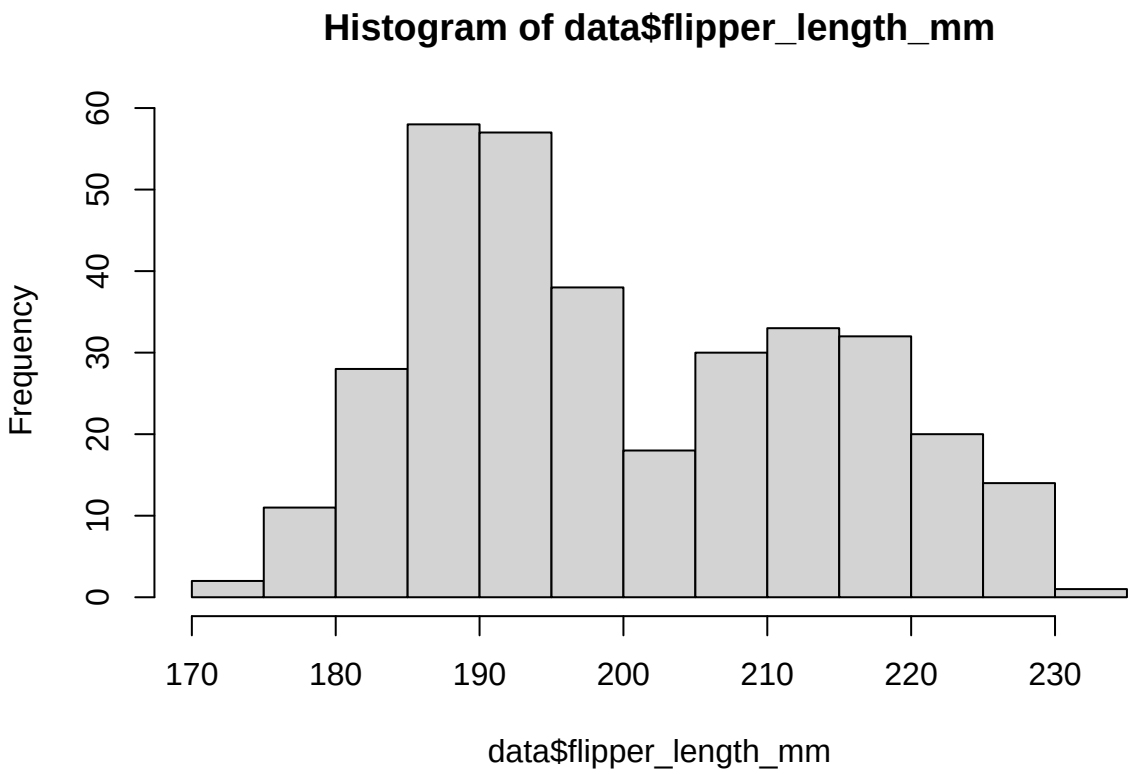
Visuals help us catch outliers, understand distributions, and explore relationships.

5.1 Histogram of a Numeric Variable

Histograms are plots that show how frequently values fall within certain ranges, helping us understand the distribution of a numeric variable. In ecology, we often use histograms to examine traits like body size, group size, or environmental variables to spot patterns such as skew, outliers, or multimodal distributions.

```
# The easiest way to quickly plot a histogram - a way to look at the distribution of values for a varia

hist(data$flipper_length_mm)
```




```
# Although you can use this format for reports, it is not very appealing, and not quite publication-qua
```

Instead, I prefer to use `ggplot` ('GG' stands for Grammar of Graphics).
And it is a mini-language within R that we use to create quick, polished, and publication-ready graphs.

There are a lot of different things you can do with ggplots, and we'll see many examples of these in the coming weeks.

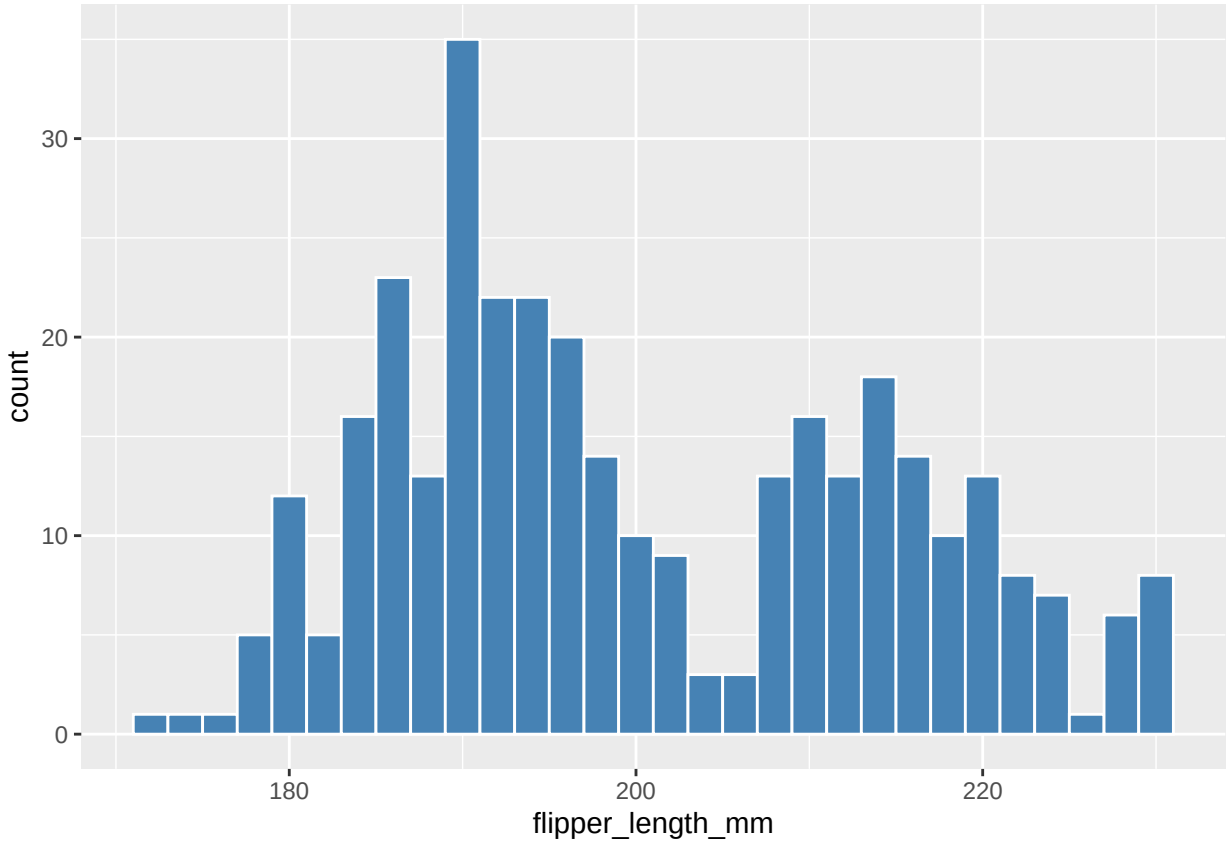
Every `ggplot` has three core components:
- **Data** – the dataset being visualized
- **Aesthetics** (`aes`) – mappings that connect variables to visual properties (like x-axis, y-axis, color)
- **Geometries** (`geom_`) – the type of plot you want to make (like `geom_point()` or `geom_bar()`).

5.2 Histogram with ggplot

So let's build our first ggplot:

```
# Histogram
ggplot(data, aes(x = flipper_length_mm)) +
  geom_histogram(binwidth = 2, fill = "steelblue", color = "white")
```

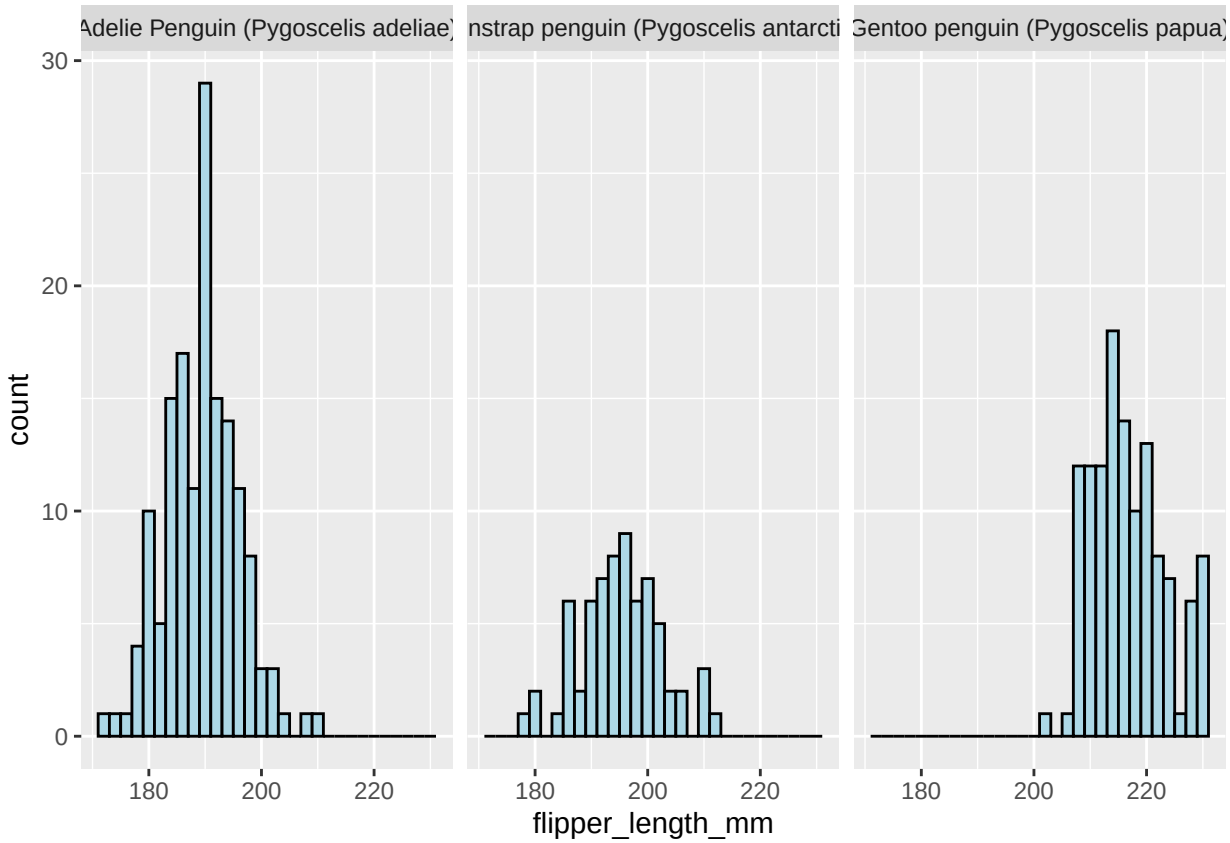
```
## Warning: Removed 2 rows containing non-finite outside the scale range
## (`stat_bin()`).
```




```
# + labs(title = "Distribution of Flipper Length", x = "Flipper Length (mm)", y = "Count")

# Use facet_wrap() to compare distributions across groups.
ggplot(data, aes(x = flipper_length_mm)) +
  geom_histogram(binwidth = 2, fill = "lightblue", color = "black") +
  facet_wrap(~ species) #+
```

```
## Warning: Removed 2 rows containing non-finite outside the scale range
## (`stat_bin()`).
```

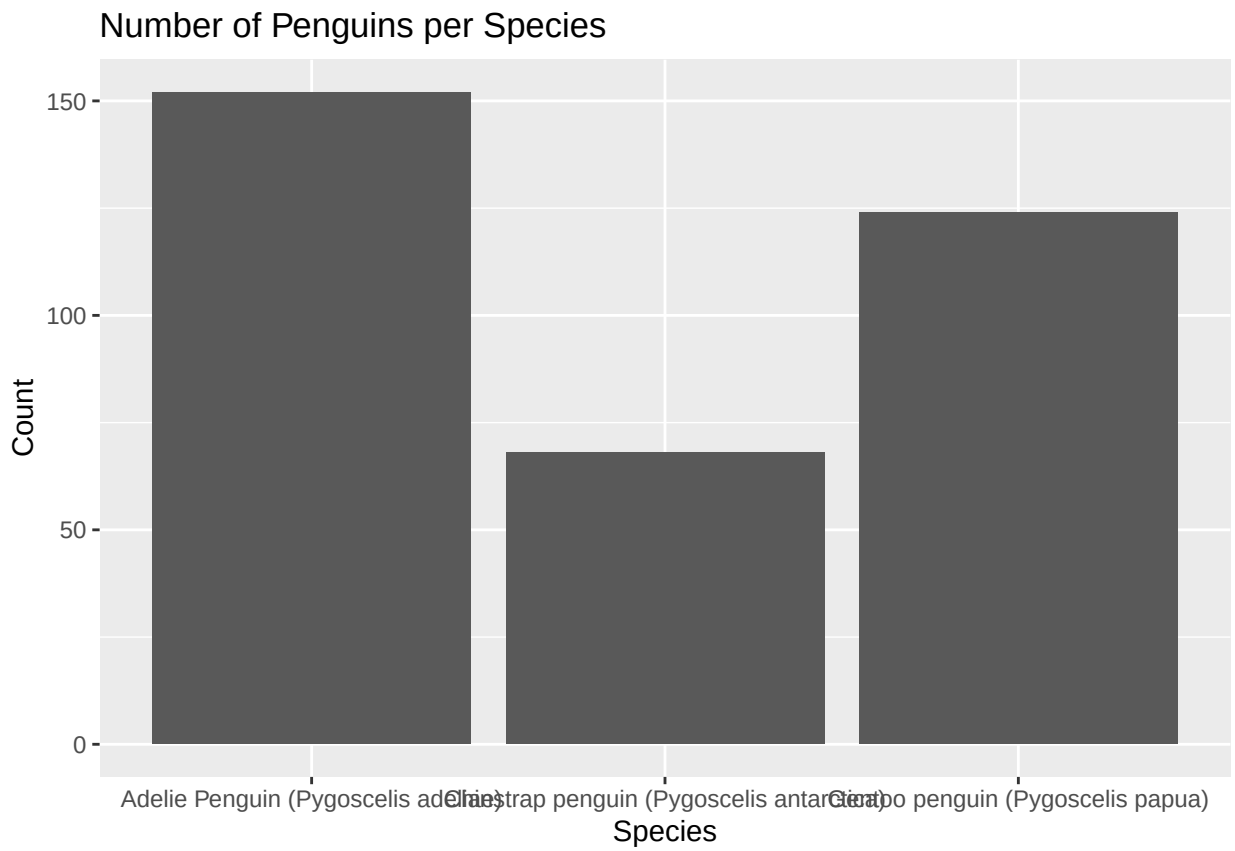



```
  #labs(title = "Flipper Length by Species", x = "Flipper Length (mm)", y = "Count") +
  #labs(title = "Flipper Length by Species", x = "Flipper Length (mm)", y = "Count")
```

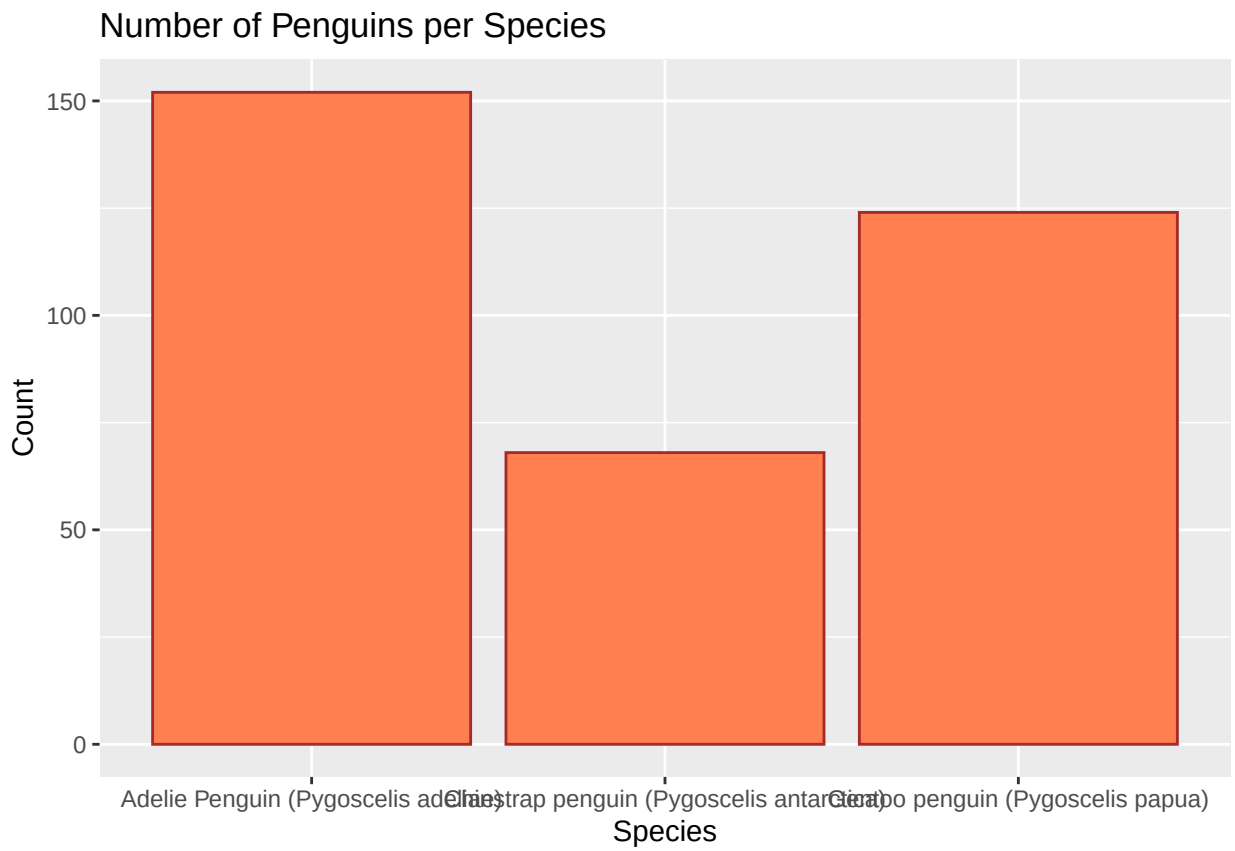
For the full set of R color names, check out this link!

5.3 Bar Plot of a Categorical Variable

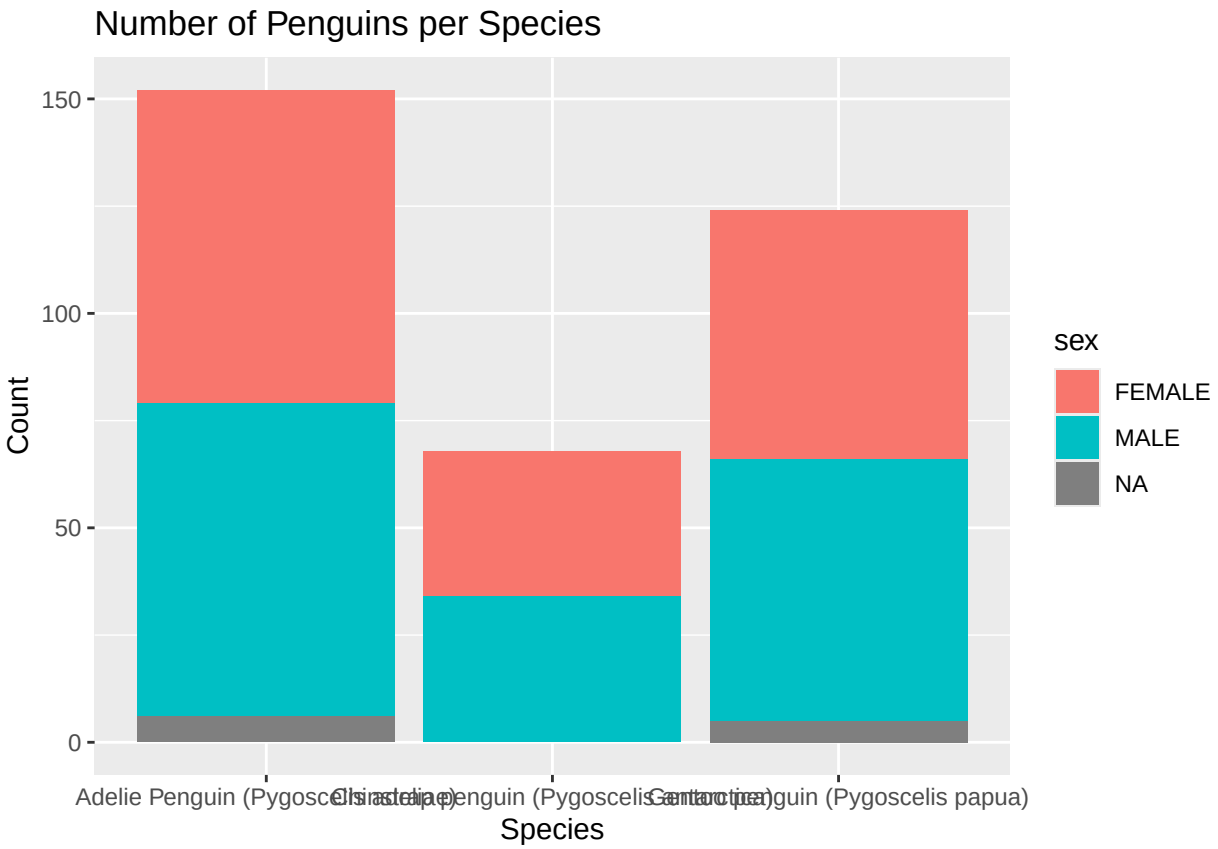
```
ggplot(data, aes(x = species)) +
  geom_bar() + # fill is the color inside the bar; color is the outlien
  labs(title = "Number of Penguins per Species", x = "Species", y = "Count")
```




```
ggplot(data, aes(x = species)) +
  geom_bar(fill = "coral", color = "brown") + # fill is the color inside the bar; color is the outlien
  labs(title = "Number of Penguins per Species", x = "Species", y = "Count")
```




```
ggplot(data, aes(x = species, fill = sex)) +
  geom_bar() + # fill is the color inside the bar; color is the outlien
  labs(title = "Number of Penguins per Species", x = "Species", y = "Count")
```



Note: These are *exploratory* plots. We're not yet testing hypotheses—just getting familiar with our data's structure and quirks.

Skills Application – Lab

6. Practice Prompts

Each student should now begin a **new R Markdown file** in the same project and try the following with either their own dataset or a provided one:

If you are working on **your own dataset**, then create a Markdown file to which you will be using and continuously adding. - Name it: *Lastname_Firstname_Data*

If you are working on a **example dataset**, then create a Markdown file just for this week. - Name it: *Lastname_Firstname_Week1*

Then, begin the lab activity: 0. Import the dataset into the environment and save as an object.

1. Identify which variables have missing values.
2. Check for misclassified columns (e.g., text stored as factors, numbers stores as text).
3. Convert at least one column to the correct type.
4. Get summary statistics for two variables of different classes.
5. Create one bar plot (categorical variable) and one histogram (numeric variable).

6. Add a title and axis labels to each plot.
7. Knit the script into an HTML file, which will be saved to your working directory

---

Summary

This week we:
- Loaded and explored a new dataset
- Investigated data types and missing values
- Visualized numeric and categorical variables
- knitted code into coding documentation

Next week, we'll start wrangling our data using `filter()`, `select()`, and `mutate()` from `dplyr`.

Optional - additional functions

To explore on your own time, ahead of next week

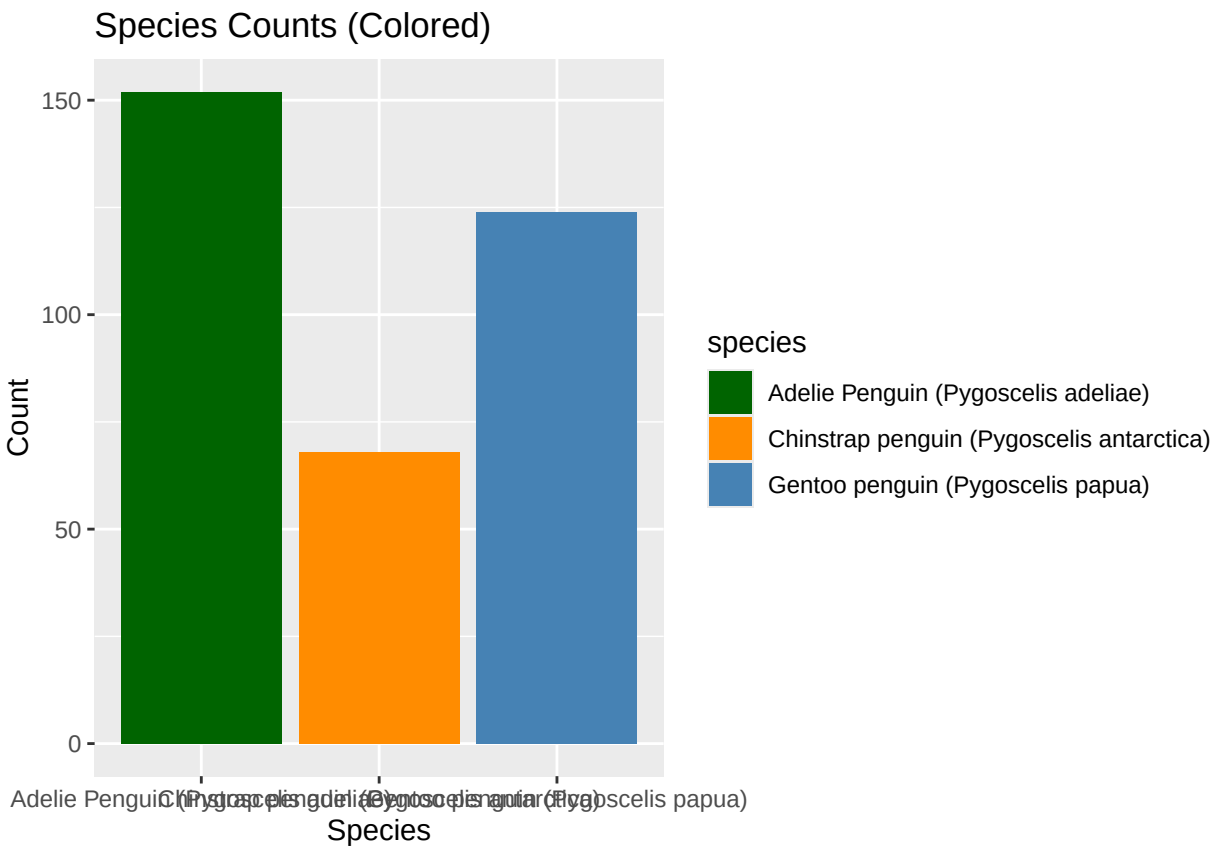
```
# See which individual records have the most missing data.
# Number of missing values per row
data %>%
  mutate(row_missing = rowSums(is.na(.))) %>%
  arrange(desc(row_missing)) %>%
  head()
```

```
## # A tibble: 6 x 18
##   study_name sample_number species             region island stage individual_id
##   <chr>              <dbl> <chr>               <chr>  <chr>  <chr> <chr>
## 1 PAL0708                4 Adelie Penguin (Py~ Anvers Torge~ Adul~ N2A2
## 2 PAL0910              120 Gentoo penguin (Py~ Anvers Biscoe Adul~ N38A2
## 3 PAL0708                9 Adelie Penguin (Py~ Anvers Torge~ Adul~ N5A1
## 4 PAL0708               12 Adelie Penguin (Py~ Anvers Torge~ Adul~ N6A2
## 5 PAL0708               48 Adelie Penguin (Py~ Anvers Dream  Adul~ N29A2
## 6 PAL0708                1 Adelie Penguin (Py~ Anvers Torge~ Adul~ N1A1
## # i 11 more variables: clutch_completion <chr>, date_egg <chr>,
## #   bill_length_mm <dbl>, bill_depth_mm <dbl>, flipper_length_mm <dbl>,
## #   body_mass_g <dbl>, sex <fct>, delta_15_n_o_oo <dbl>, delta_13_c_o_oo <dbl>,
## #   comments <chr>, row_missing <dbl>
```

```
# Find out whether certain groups (e.g., sexes or species) are more likely to have missing values.
# Compare missing values in body mass across species
data %>%
  group_by(species) %>%
  summarise(missing_body_mass = sum(is.na(body_mass_g)))
```

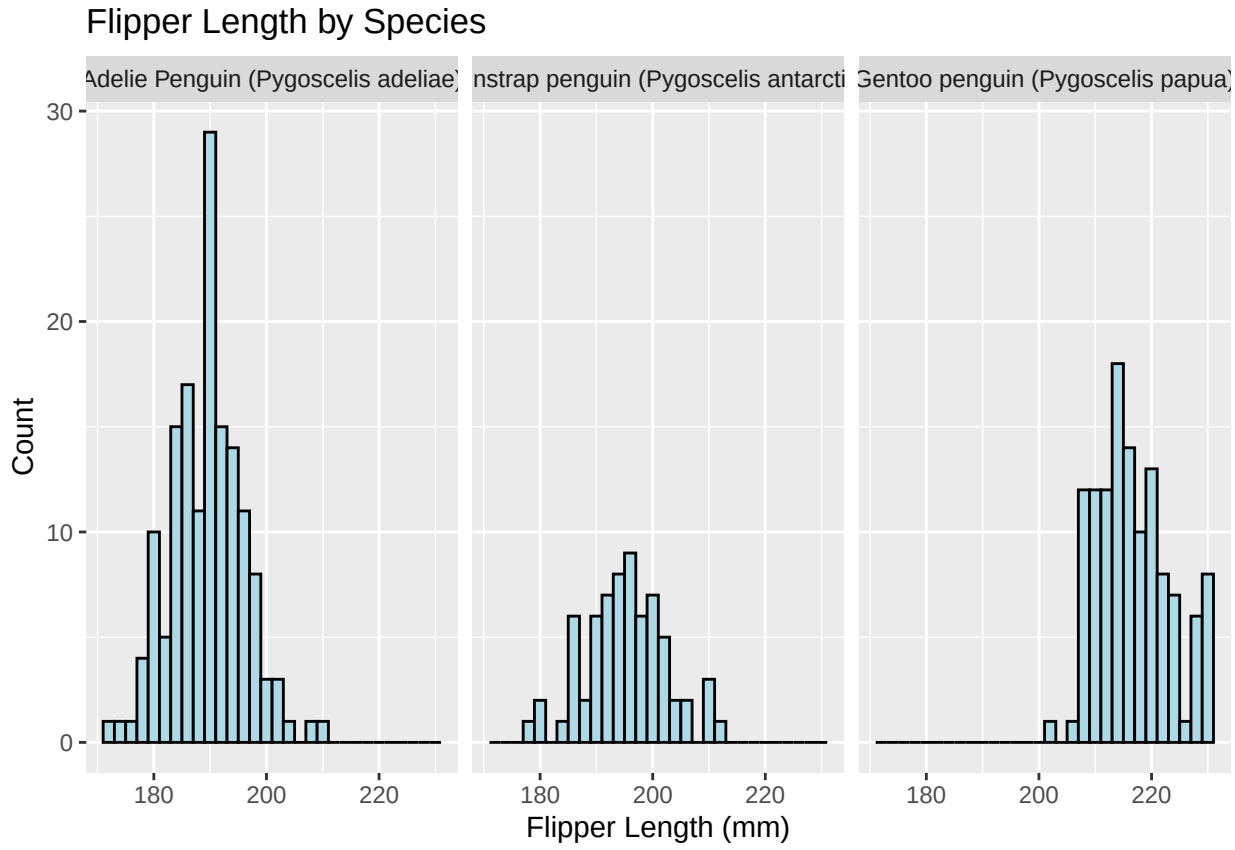
```
## # A tibble: 3 x 2
##   species                                    missing_body_mass
##   <chr>                                                  <int>
## 1 Adelie Penguin (Pygoscelis adeliae)                        1
## 2 Chinstrap penguin (Pygoscelis antarctica)                  0
## 3 Gentoo penguin (Pygoscelis papua)                          1
```

```r
# Customize colors in a bar plot using scale_fill_manual().
ggplot(data, aes(x = species, fill = species)) +
  geom_bar() +
  scale_fill_manual(values = c("darkgreen", "darkorange", "steelblue")) +
  labs(title = "Species Counts (Colored)", x = "Species", y = "Count")
```



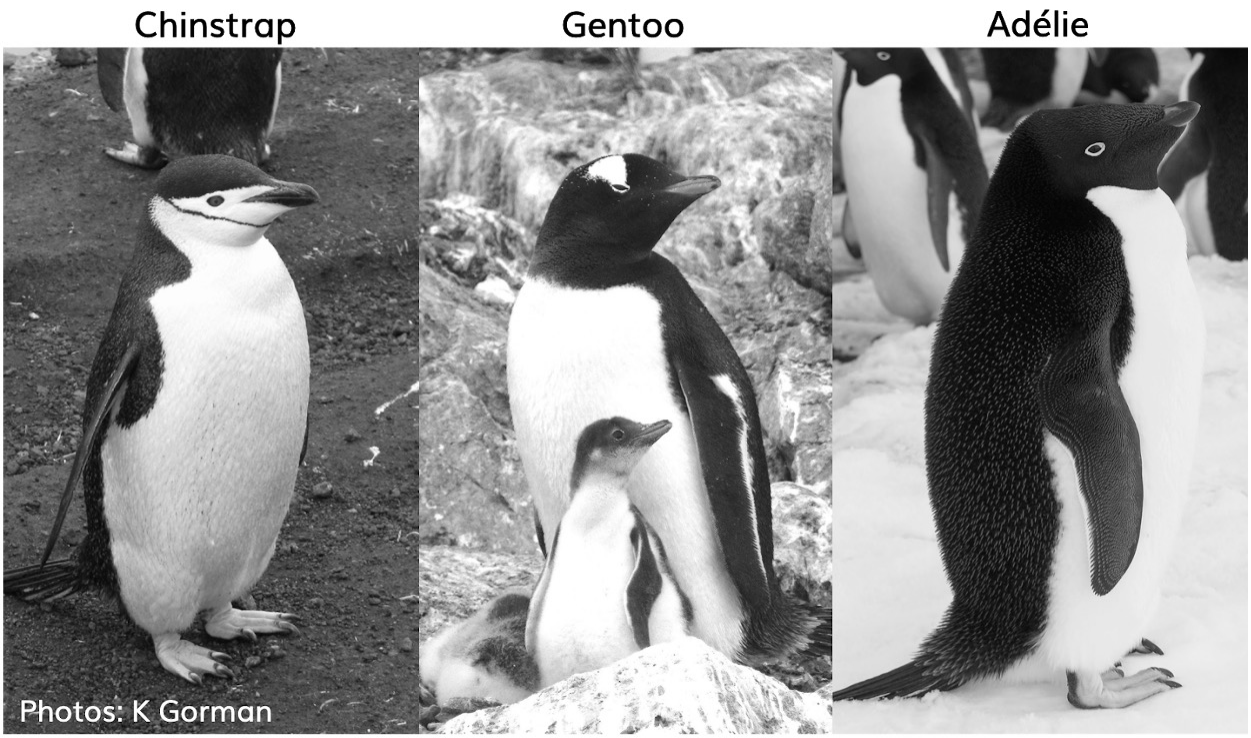

```r
# Use facet_wrap() to compare distributions across groups.
ggplot(data, aes(x = flipper_length_mm)) +
  geom_histogram(binwidth = 2, fill = "lightblue", color = "black") +
  facet_wrap(~ species) +
  labs(title = "Flipper Length by Species", x = "Flipper Length (mm)", y = "Count")
```

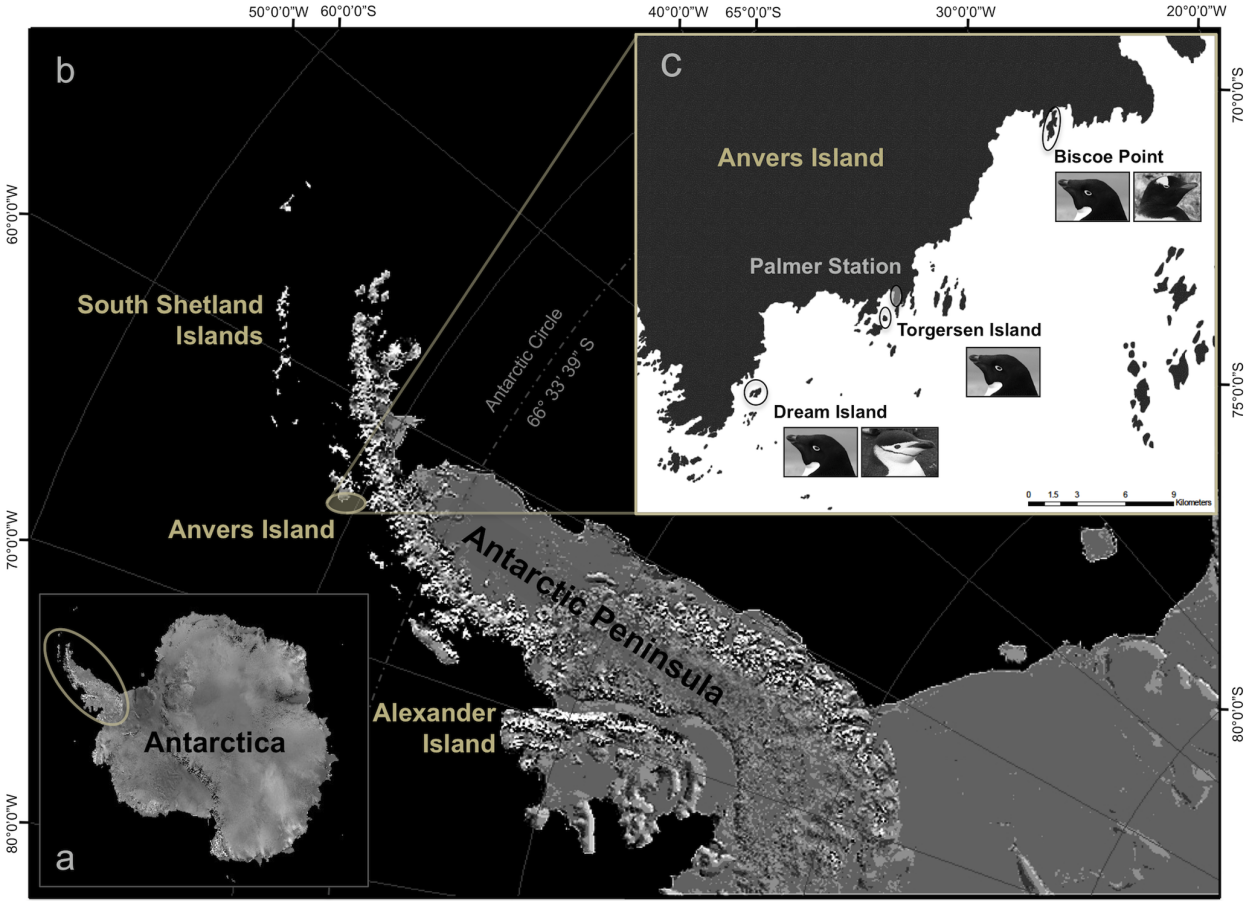
```
## Warning: Removed 2 rows containing non-finite outside the scale range
## (‘stat_bin()‘).
```




```
# Look at how many penguins exist per island and species.
# desc() dictates the *descending* order
data %>%
  count(island, species) %>%
  arrange(desc(n))
```

```
## # A tibble: 5 x 3
##   island    species                                      n
##   <chr>     <chr>                                    <int>
## 1 Biscoe    Gentoo penguin (Pygoscelis papua)          124
## 2 Dream     Chinstrap penguin (Pygoscelis antarctica)   68
## 3 Dream     Adelie Penguin (Pygoscelis adeliae)         56
## 4 Torgersen Adelie Penguin (Pygoscelis adeliae)         52
## 5 Biscoe    Adelie Penguin (Pygoscelis adeliae)         44
```


Chinstrap
Gentoo
Adélie
Photos: K Gorman


50°0'0"W
60°0'0"S
40°0'0"W
65°0'0"S
30°0'0"W
20°0'0"W
60°0'0"W
70°0'0"W
80°0'0"W
70°0'0"S
75°0'0"S
80°0'0"S
b
c
a
Anvers Island
Biscoe Point
Palmer Station
Torgersen Island
Dream Island
South Shetland Islands
Antarctic Circle
66° 33' 39" S
Anvers Island
Antarctic Peninsula
Alexander Island
Antarctica
0 1.5 3 6 9
Kilometers