Introduction to Biostatistics and R

Lecture 1

1 Outline

- Introduction to Biostatistics
- Introduction to `R`

2 Introduction to Biostatistics

2.1 *Datafication*

The field of statistics has grown in recent years primarily due to the *datafication* of the world.

> **Important**
>
> 98% of all stored information is digital. Collected data is increasing even at this very moment.

2.2 Why statistics?

> **Statistics**
>
> Statistics is a field of study concerned with:
>
> - The collection, organization, summarization, and analysis of data
> - The drawing of inferences about a body of data when only a part of the data is observed.

Data → Numbers → Information → Investigation → Communication

2.3 Statistical Thinking

How is statistical thinking different from mathematical thinking?

> **Statistical Thinking**
>
> Statistical thinking involves understanding and analyzing data while accounting for uncertainty!

2.4 Activity

Flip a coin 10 times. If you don't have a coin, search "coin flip" on Google.

> **Note**
>
> How many times did you get heads? Do you think the coin you flipped was fair?

2.5 Extensions of Statistics

2.5.1 Biostatistics

Biostatistics involves applying statistical concepts to data from the biological sciences, health sciences, and medicine.

2.5.2 Data Science

Data science is the study of how to extract useful information from data using quantitative methods and theories from many fields. The field focuses on large data sets that were not originally designed or collected to address the question of interest.

2.6 Sources of Data

Available data usually come from the following sources:

- Records
- Surveys
- Experiments
- Data Banks
- Prior Literature

2.7 Categories of Statistics

2.7.1 Descriptive Statistics

Descriptive statistics are used to describe properties of complex sets of numbers. Summary statistics are a good example of descriptive statistics.

2.7.2 Inferential Statistics

Inferential statistics are used to infer information from a smaller group (sample) to a more general group (population).

2.8 Random Variables

Random variables have values obtained arise as a result of chance factors, so that they cannot be exactly predicted in advance. Values of random variables resulting from measurement procedures are referred to as *observations/measurements*.

> **Note**
>
> Random variables could be classified as qualitative or quantitative.

2.9 Random Variable Types

2.9.1 Quantitative Variables

Quantitative variables are variables that can be measured or characterized with a numerical value.

> **Discrete Random Variables**
>
> A **discrete variable** is characterized by gaps or interruptions in the values that it can assume.
> Example: Customer counts at Cafe Rio, Number of missing teeth, Likert Scale scores

> **Continuous Random Variables**
>
> A **continuous variable** is characterized by gaps or interruptions in the values that it can assume.
> Example: Speed, Weight, Time

2.9.2 Qualitative Variables

Qualitative variables cannot be measured numerically, but can be described categorically. Discrete and qualitative variables are also known as *categorical variables*.

2.10 Data Types

2.10.1 Nominal Data

As the name implies, nominal data consist of "naming" observations or classifying them into various mutually exclusive and collectively exhaustive categories.

Examples: Assigned sex at birth (male,female); HHS Regions (HHS Regions 1-10)

> **!** Important
>
> Nominal data are typically qualitative in nature and does not account for any ordering in variable levels.

2.10.2 Ordinal Data

Ordinal data are for variables with values with inherent ordering.

Examples: Shirt size (Small, Medium, Large, XL); Socioeconomic status (Low, Medium, High)

> **!** Important
>
> Ordinal data can include discrete variables and some qualitative variables.

2.10.3 Interval Data

Interval data includes measurements that can be ordered and a distance metric can be defined between two measurements. Interval scales have equal intervals between values with arbitrary zero points.

Examples: Temperature, IQ

2.10.4 Ratio Data

Ratio data is similar to interval data, but with an absolute zero measurement defined as the "absence" of the variable being measured.

Examples: Weight, Height

2.11 Exercise

2.11.1 Question

Identify the type of data/variable for the following:

- BMI
- Satisfaction Scale (Unsatisfied, Moderately Satisfied, Satisfied, Very Satisfied)
- Eye Color
- Credit Rating

2.11.2 Answers

- BMI (**ratio**)
- Satisfaction Scale (Unsatisfied, Moderately Satisfied, Satisfied, Very Satisfied) (**ordinal**)
- Eye Color (**nominal**)
- Credit Score (**interval**)

2.12 Population vs. Sample

Population

A population is the largest collection of entities for which we have an interest at a particular time. Measurements of some variable from these entities would generate a population of values for that variable.
An exact value calculated from a population is referred to as a **parameter**.

Sample

A sample is a part of the population.
An exact value calculated from a sample is referred to as a **statistic**.

2.13 Population vs. Sample

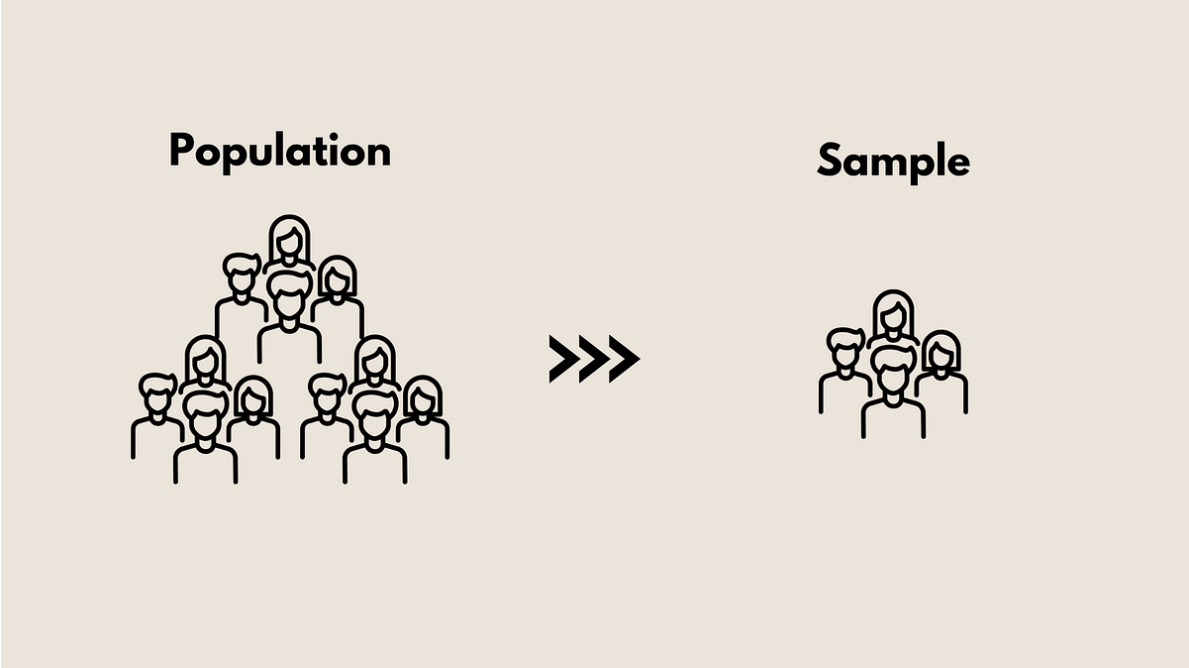


Figure 1: Taken from: https://medium.com/@ritusantra/population-v-s-sample-f17c40967257

2.14 How to Sample

Sampling can be grouped into two broad categories: probability-based/random sampling and convenience sampling.

2.14.1 Random Samples

A sample is a random sample when the probability with which every respondent was sampled is known. These probabilities are not necessarily equal. Types of random sampling include:

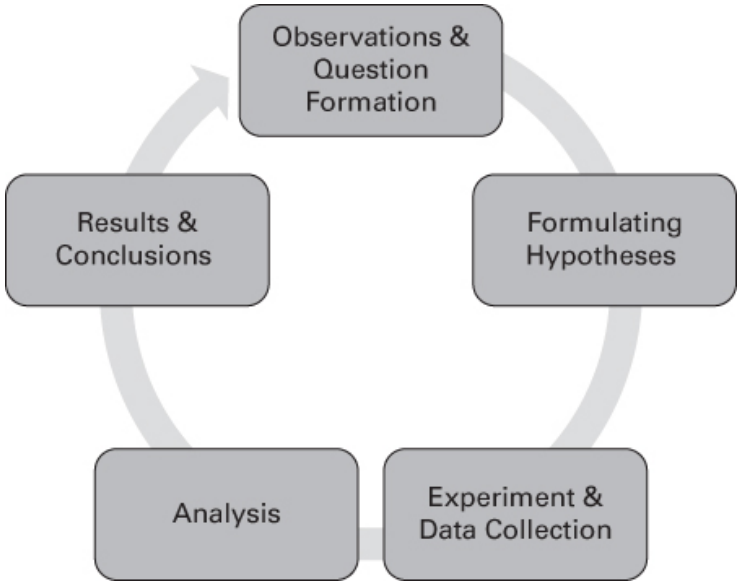
- Simple random sampling
- Stratified random sampling
- Cluster sampling

2.14.2 Convenience Samples

A sample is a convenience sample when the respondents are selected based on ease of access or availability. This could be a potential source of bias.

2.15 Scientific Method

The scientific method is a process by which scientific information is collected, analyzed, and reported in order to produce unbiased and replicable results in an effort to provide an accurate representation of observable phenomena.



3 Introduction to `R`

3.1 Installation

You can install `R` and `RStudio` on your personal computers and laptops by following the instructions on this page: https://posit.co/download/rstudio-desktop/

`RStudio` is currently installed on the classroom computers.

3.2 Basic Programming Terminology

- Source Code: A text listing of commands to be compiled or assembled into an executable computer program.
- Running Code: The act of telling `R` to perform an act by giving it commands through source code.
- Console Pane: Where `R` commands are entered

> **!** Important
>
> There are different types of programming data types such as integers, doubles/numerics, logicals, and characters.
>
> - Integers (int) have values like -1,0,2
> - Numerics (dbl, num) are numbers including integers and decimals,
> - Logicals (logi) are either `TRUE` or `FALSE`
> - Characters (chr)are text variables such as "Hello, World", "Female", "Yes"

3.3 Basic Programming Terminology

3.3.1 Vectors

Vectors are a series of values. These can be created using the `c()` function, known as the combine/concatenate function.

```
c(1,2,3)
```

```
[1] 1 2 3
```

```
c("A","B","C")
```

```
[1] "A" "B" "C"
```

3.3.2 Variables

You can store different data types to variables using the `<-` sign.

```
var1 <- c(1,2,3)
var1
```

```
[1] 1 2 3
```

```
var1+2
```

```
[1] 3 4 5
```

```
var2 <- c("A","B","C")
var2
```

```
[1] "A" "B" "C"
```

3.3.3 Factors

Categorical data are commonly represented in R as factors.

```
factor(c("18-39","40-59","60+"), levels=c("18-39","40-59","60+"))
```

```
[1] 18-39 40-59 60+
Levels: 18-39 40-59 60+
```

```
likert <- c("Disagree", "Strongly Disagree","Agree","Neutral","Strongly Agree")
likert
```

```
[1] "Disagree"          "Strongly Disagree" "Agree"
[4] "Neutral"           "Strongly Agree"
```

```
factor(likert, levels=c("Strongly Disagree","Disagree", "Neutral","Agree","Strongly Agree"))
```

```
[1] Disagree          Strongly Disagree Agree             Neutral
[5] Strongly Agree
Levels: Strongly Disagree Disagree Neutral Agree Strongly Agree
```

3.4 Basic Programming Technology

3.4.1 Data Frames

Data frames are rectangular spreadsheets. Data are typically imported as data frames.

> **Note**
>
> Rows correspond to observations and the columns correspond to variables.

Example:

```
head(mtcars)
```

```
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

3.4.2 Conditionals

You can test for mathematical relations such as equality/inequality, resulting in a `TRUE` or `FALSE` value.

```
2+1==3
```

```
[1] TRUE
```

```
3+5 <=1
```

```
[1] FALSE
```

```
x <- c(1,2,3,4,5)
x < 3
```

```
[1]  TRUE  TRUE FALSE FALSE FALSE
```

3.4.3 Functions

Functions are commands in R. `c()` is a function that combines different values. You can create a function or use functions built for R.

Example: `seq()` is a function that generates a sequence of numbers. The resulting sequence of numbers can be changed by changing the attributes of the function

```
seq(1,5)
```

```
[1] 1 2 3 4 5
```

```
seq(1,5,by=2)
```

```
[1] 1 3 5
```

```
seq(1,5,length.out=10)
```

```
 [1] 1.000000 1.444444 1.888889 2.333333 2.777778 3.222222 3.666667 4.111111
 [9] 4.555556 5.000000
```

To learn more about what a specific R function does, type `?function_name` in the console. (Ex:`?seq`, `?c`)

3.5 Errors, Warnings, and Messages

3.5.1 Errors

When you input a legitimate error, R will warn you using a sentence starting with "Error in" and includes a sentence explaining what went wrong.

```
add(1,2,3)
```

```
Error in add(1, 2, 3): could not find function "add"
```

```
c("A","B")+1
```

```
Error in c("A", "B") + 1: non-numeric argument to binary operator
```

> **Note**
>
> If the text starts with "Error", figure out what's causing it. Think of errors as a red traffic light: stop and assess for anything wrong in the code (missing parenthesis, adding characters to numbers, non-existent functions, etc.)

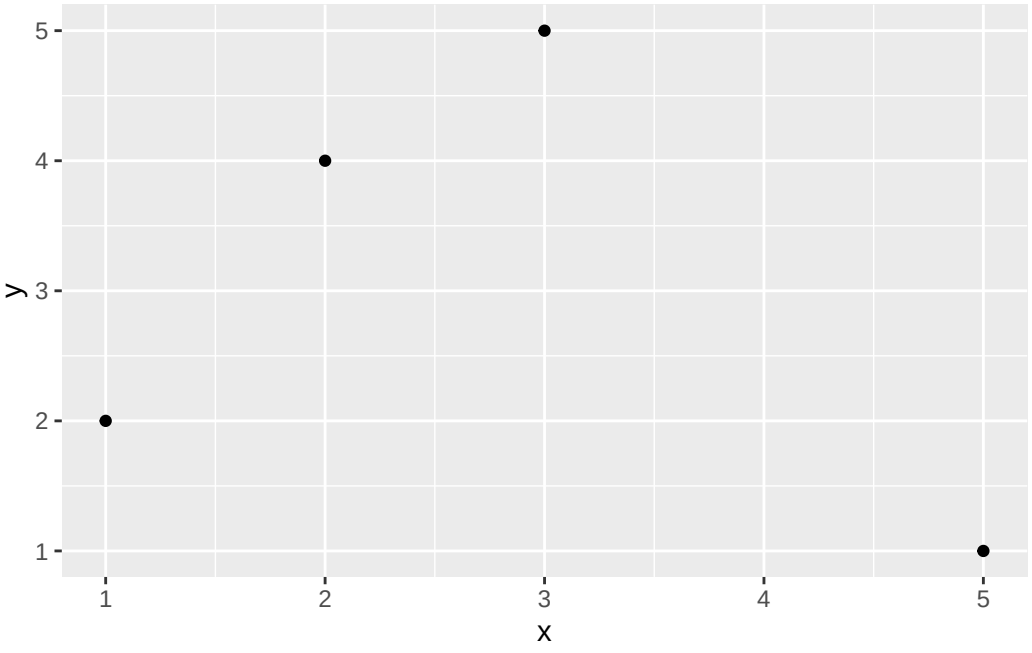
> **Important**
>
> Encountering errors in your code is normal even for an experienced coder. Don't be afraid to review your code if you made an error! This process is called "debuggging".

3.5.2 Warnings

When `R` produces a warning, your code will still be implemented albeit with some caveats.

```
for_plotting <- data.frame(x=c(1,2,3,4,5), y=c(2,4,5,NA,1))
library(ggplot2)
ggplot(data=for_plotting,aes(x=x,y=y)) + geom_point()
```

```
Warning: Removed 1 row containing missing values or values outside the scale range
(`geom_point()`).
```



The plot is displayed, but a warning was raised regarding a missing value. Note that the data frame `for_plotting` has a missing value for `y` denoted by `NA`.

3.5.3 Messages

If the output text does not start with an error or a warning, it's just a friendly message from `R` telling you about the output. These messages are typically seen when loading `R` packages or data sets from external sources.

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v lubridate 1.9.4     v tibble    3.3.0
v purrr     1.1.0     v tidyr     1.3.1
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to beco
```

3.6 `R` packages

`R` packages extend the functionality of `R` by providing additional functions, data, and documentation. These packages are written by `R` users around the world and can be downloaded for free!

> **Note**
>
> Think of `R` as a new phone. `R` packages are apps that you can download to use your phone in many different ways.

3.7 Installing and Loading `R` Packages

Like apps on a phone, `R` packages also need to be installed. These packages can be installed by running the following code snippet `install.packages("PackageName")`. For example, to install the package `tidyverse` used for data manipulation and cleaning, you can run the following code:

```
install.packages("tidyverse")
```

To load this package in R, you can use the following syntax:

```
library(tidyverse)
```

> **!** Important
>
> You must have an active internet connection to install R packages to your device.

3.8 Exercise

3.8.1 Exercise

Install and load the following packages: `readxl`, `nycflights23` and `knitr`.

3.8.2 Answer

```
install.packages("readxl")
install.packages("nycflights23")
install.packages("knitr")
```

```
library(readxl)
library(nycflights23)
library(knitr)
```

3.9 Exploring Data Sets

The `nycflights23` package includes some data sets saved as data frames. These data sets are related to all domestic flights departing from one of New York City's three main airports in 2023: Newark Liberty International (EWR), John F. Kennedy International (JFK), and LaGuardia Airport (LGA).

One of the data sets in this package is the `flights` data set.

```
flights
```

```
# A tibble: 435,352 x 19
    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
 1  2023     1     1        1           2038       203      328              3
 2  2023     1     1       18           2300        78      228            135
 3  2023     1     1       31           2344        47      500            426
 4  2023     1     1       33           2140       173      238           2352
 5  2023     1     1       36           2048       228      223           2252
 6  2023     1     1      503            500         3      808            815
 7  2023     1     1      520            510        10      948            949
 8  2023     1     1      524            530        -6      645            710
 9  2023     1     1      537            520        17      926            818
10  2023     1     1      547            545         2      845            852
# i 435,342 more rows
# i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
#   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
#   hour <dbl>, minute <dbl>, time_hour <dttm>
```

> **Note**
>
> A tibble is a special type of data frame! The dimensions show the `number of rows x number of columns`. Each row corresponds to an observation, while each column corresponds to the variables describing each observation.

3.10 Exploring the `flights` data set.

You can use the following functions to explore a data set.

3.10.1 `View()`

`View()` brings up RStudio's built in data viewer. That is, if you want to view data like an Excel sheet.

```
View(flights)
```

3.10.2 `glimpse()`

The `glimpse()` function from the package `dplyr` (part of `tidyverse`) provides us with a different view of the data set. It includes the *data type* of each variable defined by the columns of the data frame.

```r
glimpse(flights)
```

```
Rows: 435,352
Columns: 19
$ year           <int> 2023, 2023, 2023, 2023, 2023, 2023, 2023, 2023, 2023, 2~
$ month          <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
$ day            <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
$ dep_time       <int> 1, 18, 31, 33, 36, 503, 520, 524, 537, 547, 549, 551, 5~
$ sched_dep_time <int> 2038, 2300, 2344, 2140, 2048, 500, 510, 530, 520, 545, ~
$ dep_delay      <dbl> 203, 78, 47, 173, 228, 3, 10, -6, 17, 2, -10, -9, -7, -~
$ arr_time       <int> 328, 228, 500, 238, 223, 808, 948, 645, 926, 845, 905, ~
$ sched_arr_time <int> 3, 135, 426, 2352, 2252, 815, 949, 710, 818, 852, 901, ~
$ arr_delay      <dbl> 205, 53, 34, 166, 211, -7, -1, -25, 68, -7, 4, -13, -14~
$ carrier        <chr> "UA", "DL", "B6", "B6", "UA", "AA", "B6", "AA", "UA", "~
$ flight         <int> 628, 393, 371, 1053, 219, 499, 996, 981, 206, 225, 800,~
$ tailnum        <chr> "N25201", "N830DN", "N807JB", "N265JB", "N17730", "N925~
$ origin         <chr> "EWR", "JFK", "JFK", "JFK", "EWR", "EWR", "JFK", "EWR",~
$ dest           <chr> "SMF", "ATL", "BQN", "CHS", "DTW", "MIA", "BQN", "ORD",~
$ air_time       <dbl> 367, 108, 190, 108, 80, 154, 192, 119, 258, 157, 164, 1~
$ distance       <dbl> 2500, 760, 1576, 636, 488, 1085, 1576, 719, 1400, 1065,~
$ hour           <dbl> 20, 23, 23, 21, 20, 5, 5, 5, 5, 5, 5, 6, 5, 6, 6, 6, 6,~
$ minute         <dbl> 38, 0, 44, 40, 48, 0, 10, 30, 20, 45, 59, 0, 59, 0, 0, ~
$ time_hour      <dttm> 2023-01-01 20:00:00, 2023-01-01 23:00:00, 2023-01-01 2~
```

3.10.3 `kable()`

The `kable()` function is part of the package `knitr`. In this example, we will use another data set in the `nycflights23` package: the `airlines` data set.

```r
kable(airlines)
```

carrier	name
9E	Endeavor Air Inc.
AA	American Airlines Inc.
AS	Alaska Airlines Inc.
B6	JetBlue Airways
DL	Delta Air Lines Inc.
F9	Frontier Airlines Inc.
G4	Allegiant Air
HA	Hawaiian Airlines Inc.

carrier	name
MQ	Envoy Air
NK	Spirit Air Lines
OO	SkyWest Airlines Inc.
UA	United Air Lines Inc.
WN	Southwest Airlines Co.
YX	Republic Airline

3.10.4 `$`

The `$` operator allows us to extract and then explore a single variable within a data frame.

```
airlines$name
```

```
 [1] "Endeavor Air Inc."      "American Airlines Inc." "Alaska Airlines Inc."
 [4] "JetBlue Airways"        "Delta Air Lines Inc."   "Frontier Airlines Inc."
 [7] "Allegiant Air"          "Hawaiian Airlines Inc." "Envoy Air"
[10] "Spirit Air Lines"       "SkyWest Airlines Inc."  "United Air Lines Inc."
[13] "Southwest Airlines Co." "Republic Airline"
```

3.11 Exercise

3.11.1 Exercise

Can you provide me with two qualitative variables and two quantitative variables in the dataset `planes` in the `nycflights23` package?

3.11.2 Answer

```
glimpse(planes)
```

```
Rows: 4,840
Columns: 9
$ tailnum      <chr> "N101DQ", "N101DU", "N101HQ", "N101NN", "N102DN", "N102DU~
$ year         <int> 2020, 2018, 2007, 2013, 2020, NA, 2007, 2013, 1998, NA, 2~
$ type         <chr> "Fixed wing multi engine", "Fixed wing multi engine", "Fi~
$ manufacturer <chr> "AIRBUS", "C SERIES AIRCRAFT LTD PTNRSP", "EMBRAER-EMPRES~
$ model        <chr> "A321-211", "BD-500-1A10", "ERJ 170-200 LR", "A321-231", ~
```

```
$ engines      <int> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, ~
$ seats        <int> 199, 133, 80, 379, 199, 133, 80, 379, 182, 133, 199, 80, ~
$ speed        <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
$ engine       <chr> "Turbo-fan", "Turbo-fan", "Turbo-fan", "Turbo-fan", "Turb~
```

Columns made up of characters are categorical variables. Quantitative variables are those marked as integers.

> **!** Important
>
> `NA` means that the data point is missing. This is not the same as "NA", which is a character.

3.12 Exercise

3.12.1 Exercise

Explore the data set `iris`.

- How many observations does `iris` have?
- How many variables does `iris` have?
- Use `glimpse()` to determine the type of data of each column of `iris`.
- Use the `$` operator to extract the species variable in `iris`

3.12.2 Answer

```
glimpse(iris)
```

```
Rows: 150
Columns: 5
$ Sepal.Length <dbl> 5.1, 4.9, 4.7, 4.6, 5.0, 5.4, 4.6, 5.0, 4.4, 4.9, 5.4, 4.~
$ Sepal.Width  <dbl> 3.5, 3.0, 3.2, 3.1, 3.6, 3.9, 3.4, 3.4, 2.9, 3.1, 3.7, 3.~
$ Petal.Length <dbl> 1.4, 1.4, 1.3, 1.5, 1.4, 1.7, 1.4, 1.5, 1.4, 1.5, 1.5, 1.~
$ Petal.Width  <dbl> 0.2, 0.2, 0.2, 0.2, 0.2, 0.4, 0.3, 0.2, 0.2, 0.1, 0.2, 0.~
$ Species      <fct> setosa, setosa, setosa, setosa, setosa, setosa, setosa, s~
```

```
iris$Species
```

```
  [1] setosa     setosa     setosa     setosa     setosa     setosa
  [7] setosa     setosa     setosa     setosa     setosa     setosa
 [13] setosa     setosa     setosa     setosa     setosa     setosa
 [19] setosa     setosa     setosa     setosa     setosa     setosa
 [25] setosa     setosa     setosa     setosa     setosa     setosa
 [31] setosa     setosa     setosa     setosa     setosa     setosa
 [37] setosa     setosa     setosa     setosa     setosa     setosa
 [43] setosa     setosa     setosa     setosa     setosa     setosa
 [49] setosa     setosa     versicolor versicolor versicolor versicolor
 [55] versicolor versicolor versicolor versicolor versicolor versicolor
 [61] versicolor versicolor versicolor versicolor versicolor versicolor
 [67] versicolor versicolor versicolor versicolor versicolor versicolor
 [73] versicolor versicolor versicolor versicolor versicolor versicolor
 [79] versicolor versicolor versicolor versicolor versicolor versicolor
 [85] versicolor versicolor versicolor versicolor versicolor versicolor
 [91] versicolor versicolor versicolor versicolor versicolor versicolor
 [97] versicolor versicolor versicolor versicolor virginica  virginica
[103] virginica  virginica  virginica  virginica  virginica  virginica
[109] virginica  virginica  virginica  virginica  virginica  virginica
[115] virginica  virginica  virginica  virginica  virginica  virginica
[121] virginica  virginica  virginica  virginica  virginica  virginica
[127] virginica  virginica  virginica  virginica  virginica  virginica
[133] virginica  virginica  virginica  virginica  virginica  virginica
[139] virginica  virginica  virginica  virginica  virginica  virginica
[145] virginica  virginica  virginica  virginica  virginica  virginica
Levels: setosa versicolor virginica
```

`iris` has 150 observations and 5 variables: Sepal Length, Sepal Width, Petal Length, Petal Width, and Species. Species is a **factor**, while all the other variables are numeric (dbl) in nature.

4 Summary

4.1 Summary

- Introduced biostatistics and its importance
- Introduced `R`
- Explored data sets

4.2 What's next?

We will be using `R` to work with data and perform statistical analysis. We will also explore how to use `R` to explore and describe data from external sources (.csv files).