FWF 610

Data Modeling and Visualization Using R

Alejandro Molina Moctezuma

2025-06-06

Table of contents

Welcome!

Introduction to Statistical Analysis in Agriculture and the Natural Resources

This book was created in order to host all of the materials for SNR610 class.

This book/course will provide graduate students with a theoretical framework for data modeling (linear models, additive models and multivariate models), data visualization, data management, and data interpretation. This course will also aim to teach a practical use of program R for data management, analysis and visualization. Topics include an overview of data management principles, followed by methods in R for data wrangling, linear models, polynomial regression, generalized linear models, generalized additive models, and multivariate models, particularly ordination methods.

Syllabus

Please check Canvas for an updated syllabus.

Objectives and structure

Specific Course Objectives

The objectives of this class are multiple.

First, this course is designed to review basic probability and statistics (Module 1).

Second, develop your understanding and ability to develop models that are usual in research in agriculture and natural resources.

Third, develop data management skills.

Fourth, develop coding expertise for statistical analysis and for other tools.

Fifth, develop the ability to design great publication ready plots in R. Finally, this course will cater to the students research needs, and some topics may be added in case there is a need.

Part I

Introduction

This section will introduce you to

- Program R
- Projects
- Quarto
- Models

We will also discuss multiple important concepts!

> **Note**
>
> Did you take ANSC/PLSC 571 with me in Spring 2025?
> Then, some of the activities/assignments in the first three weeks will seem pretty similar! This is OK! Take this as a time to reinforce the knowledge, and we will move to different topics soon!

Welcome to week 1

> **Important**
>
> If you are reading this file before class, it may not make a lot of sense to you. That's OK! We will go through it on the first class. More importantly, you will learn more about R on Friday.

Before we get into the weeds of data analysis, coding, and R, we should remember some basics! On Friday we will learn (or refresh) some basic R skills! If you have a lot of experience in R, just hang in there, we will get more and more complex topics as we go on! If you're brand new, also hang in there, this may seem overwhelming and very time consuming. It will get easier!

We will figure out theory and coding as we go. For all assignments, we will use Quarto. It allows you to write some neat reports while using code and plots.

R Projects

We will learn more about projects during class. We will have our first R assignment this Friday. For the time being, just trust me, and let's do the following:

1. Open R Studio, go to file > new project > new directory > new project
2. Name it SNR610, and use a directory that makes sense to you

3. Download the SNR610_W1_notes.qmd and the .csv files from CANVAS IN THE NEWLY CREATED PROJECT DIRECTORY

> **!** Important
>
> If you are working on this before class, you can (and should) stop now

Statistics refresher

Open R and make sure SNR 610 project is opened. Then open the SNR610_W1_notes.qmd file in R. We will continue there!

First of, we will discuss some topics.

I want you to take notes in the .qmd document

What is statistics?

What is a population and a sample? Give examples

Data

Variable and observation

Types of data:

Categorical: Nominal and Ordinal

Numerical: Discrete and Continuous

Now, let's go back to the html file and follow the instructions to explore some different types of data.

Data exploration

Download and read the butterfly file.

If you are comfortable with R basics, you can write the code for this exercise directly into Quarto. To add extra code chunks, write this symbols: "'{r}, if you are not comfortable with R, I recommend for the time being you open a new file and copy and paste the code there, and run it}

To read data, we use the following line:

```
data<-read.csv("data/ButterflyData.csv", stringsAsFactors = T)
```

This only works if the file is in the current working directory. Make sure you download the file to the projects folder!

Let's look at some of the data, the head command shows us the top 5 rows:

```
head(data)
```

```
  X Species Status Nparasites ForewingArea
1 1 Viceroy      1          2     964.8420
2 2 Monarch      3          0    1064.9463
3 3 Viceroy      2          2    1087.6290
4 4 Viceroy      2          3    1025.4572
5 5 Monarch      2          1     986.7462
6 6 Viceroy      2          2    1029.7973
```

So, we have data on three types of butterflies: Viceroy, Monarch, and Queen:

Figure 1: The three types of Butterflies that we have data on.

We have data on forewing area, number of parasites, species, and status. For status we have numbers 1, 2, and 3, which represent the following:

1: poor

2: OK

3: Good

What types of variables are there? Identify each variable type. What are the variables and what are the observations?

Let's look at the structure of the file:

```
str(data)
```

```
'data.frame':   134 obs. of  5 variables:
 $ X           : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Species     : Factor w/ 3 levels "Monarch","Queen",..: 3 1 3 3 1 3 1 3 3 1 ...
 $ Status      : int  1 3 2 2 2 2 3 2 2 2 ...
 $ Nparasites  : int  2 0 2 3 1 2 1 4 1 3 ...
 $ ForewingArea: num  965 1065 1088 1025 987 ...
```

Note that this data is a bit messy. We will learn how to clean data during the semester. But this is closer to what your data will look like initially!

What is a factor?

Should any other variable be a factor?

```
data$Status<-as.factor(data$Status)
```

What is a factor?

```
str(data)
```

You can look at the whole dataset using the following:

```
print(data)
```

Look at the data. Do you think there is an effect of # of parasites on wing growth? Do you think different species have different sizes? How can we tell?

Inferential statistics

We can use inferential statistics to answer these questions! But why? What are inferential statistics, and what is special about inferential statistics? Discuss

Thoughts:

More on that next week.

Plotting

Look at the data. Usually the best exploratory analysis is a visual one. We will learn about plotting during the course. We will learn about good plots, bad plots, and how to use R and ggplot to make some publication level plots.

For the purposes of this exercises, I want you to think about what type of plot you ant to make, and attempt to make it. You can use R, you can use excel, you can try to draw it. You can work in teams, but try to work mostly by yourself. We will revisit this dataset at the end of the course, and you will plot it again. Hopefully this will show some of the learning you have obtained!

Probability

To understand inferential statistics, we need to understand probability.

Discuss and write a definition for probability:

What is a probability distribution? Give one example

This is your file now!

Go to the top of the file, and replace the "author" name for yours. Try to Render the file, it will hopefully work and have all your notes :)

Friday

Introduction to R, Projects, and Quarto.

1 Intro to R, Projects, and Quarto

Introduction to R, Projects, and Quarto.

Next week topic will be:

1. Statistical inference and linear models in R

> **!** About assignments
>
> While assignments are usually due on Fridays before class (check CANVAS), please read the following:
>
> 1. On Fridays, I will stay past the end of the class in the classroom, in case you have any questions.
> 2. My office is in room 474 of the ANR building. I am available MWF after class until ~4 pm and I am happy to help you with any assignment questions. Please don't hesitate to ask! I have also availability on Tuesdays and Thursdays
> 3. I am ALWAYS available via email, and I will do my best effort to reply to any R questions about the assignments as soon as possible. Please don't hesitate to reach me.
> 4. Most of the assignments will have the following structure: half-tutorial, and then half questions and assignment. You only need to turn things in that actually say "assignment question". As we move on, there will be less and less tips and less "tutorials" and more questions.

1.1 Introduction to R

`R` is probably the most used language for statistical analysis in natural resources and agriculture. Other people use SAS (still very popular, but it is slowly becoming more unpopular) and python. R is designed for statistical analysis and data visualization, and in my opinion it is more intuitive, cleaner and faster than python for a majority of general uses in data analysis. It is **not** the only option, but it is an incredible resource for data analysis.

`R` is also free and open source, and there are incredibly smart people constantly working on developing packages that help you with your specific needs.

More importantly, whether you are going to continue your academic career, go into the professional world, or analyze your data the use of `R` is now a huge advantage in the job market!

The objectives for this course (R-wise) are the following:

1. Learn the logic and syntax of coding! This is the most important aspect. If you understand the logic, then you can run whatever you want. It might take some research, looking for some functions, asking online, but you can get there
2. Interpret results. Particularly for linear models and generalized linear models. Are the results significant? What is the effect size? What is the effect of the independent variable on the dependent variable? What do I report on my paper? All of these questions and more
3. Understand error messages. Why is there an error message, what does it mean, and how to fix it. Errors in your code are unavoidable, and while often times it's easy to find solutions online, sometimes it isn't, and it is always better when you can understand them yourself.
4. Get better and more efficient at it. Don't spend hours wrangling or moving data around before you can even attempt to analyze it.
5. Use the great visualization tools available to make great publication-quality plots

1.1.0.0.1 Let's get started!

Please contact me (or raise your hand if in the classroom) to get help.

Open R-Studio and go to file > new script.

This is usually how we start a new script where we will be writing code

At this point, you should see four panes:

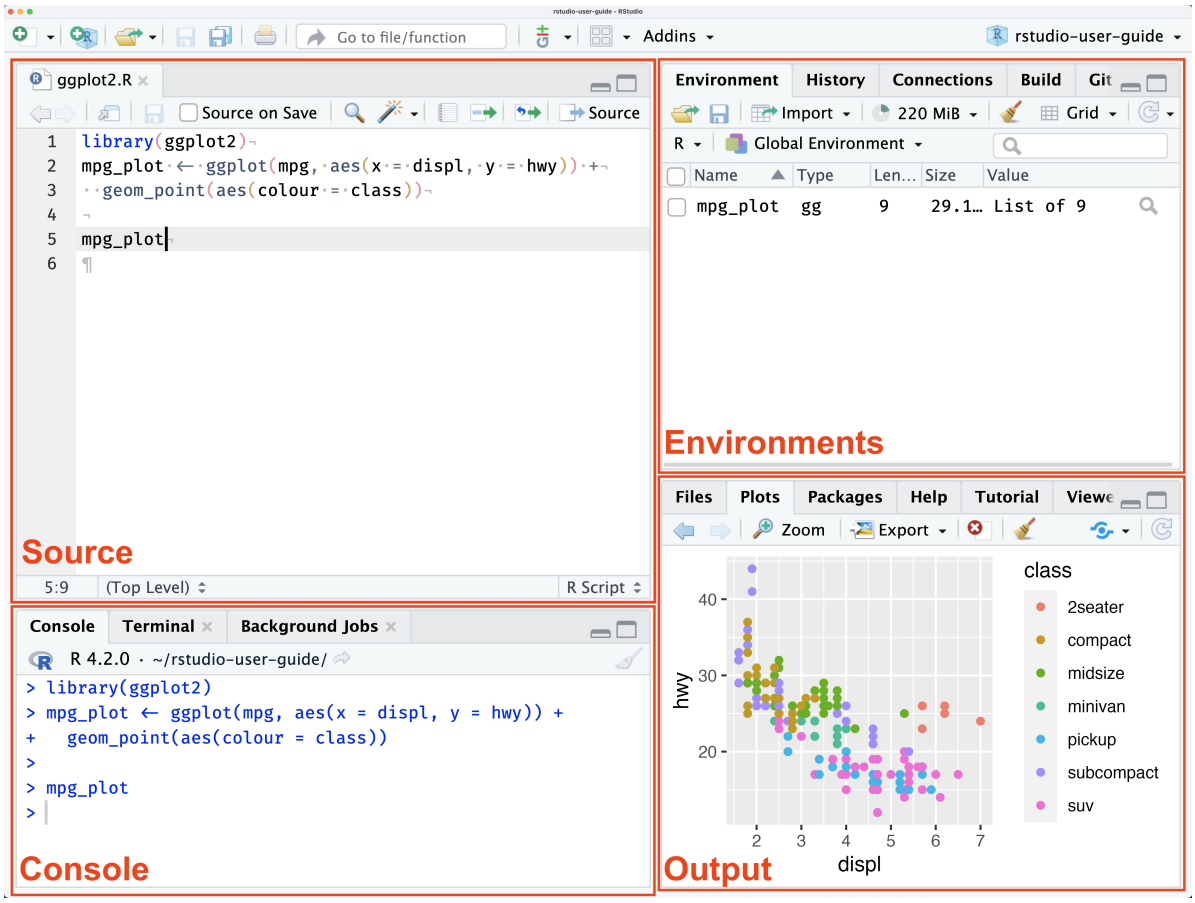


Figure 1.1: Four panes

Mine looks a bit different, but it is the same logic:

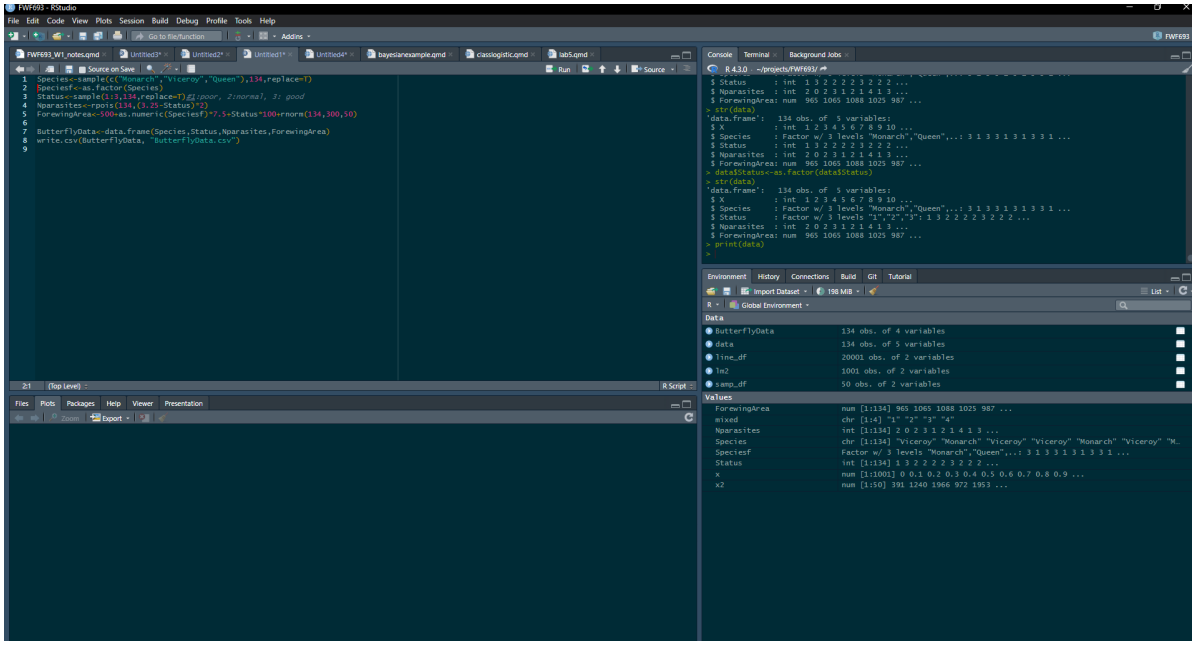


Figure 1.2: This is how my panes look

The different panes do different things. Let's go one by one on what they do:

1.1.0.0.2 Console

This is what the console looks like:

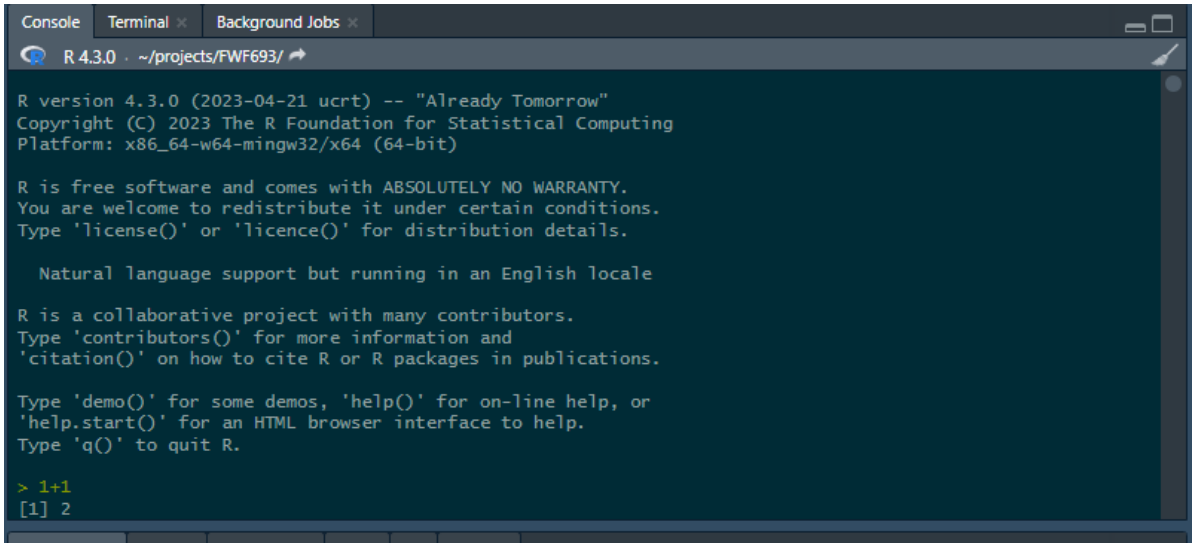


Figure 1.3: Console

You can type code here, and this is where the outputs are "printed". Notice in the figure how I wrote 1+1 and it gave me a result? Try it yourself. Try to use this "calculator" and multiply and divide.

Work at your own pace!

I want you to worry less about grades, and more about learning and acquiring skills.
If you have used R before, some of the steps in this assignment mat be too "basic". If that's the case feel free to skip them. Also, feel free to help other people that may be having issues or struggling. Collaborative learning can be super useful. And teaching/helping allows you to acquire even deeper knowledge.
At the same time, if you are brand new and are struggling, and this is taking a while to complete. Feel free to not do the last parts of the assignment (just let me know what you had issues with). Hopefully later in the course you won't feel the need to do this.
Graduate student life can be very busy. I want you to spend as much time as you can learning R and stats, because I believe it can be extremely beneficial for your career, but DO NOT sacrifice your learning in other areas (particularly research). Talk to me if you need more time, or have any issues

Try to run the following lines of code (if you are experienced with all of these commands, you can skip this step). You can copy and paste the code. Run them one by one, and try to understand what is happening:

```r
5+5
```

```r
8^5
```

```r
sqrt(9)
```

```r
c(5,7,9,15,50)
```

```r
mean(c(5,7,9,15,50))
```

```r
sd(c(5,7,9,15,50))
```

```r
1:10

-5:10

c(1:10,-5:10,25)
```

These are all basic commands!

1.1.0.0.3 Source

Now that you got familiar with what the console can do, it's time to remember the following: **DO NO WRITE YOUR CODE DIRECTLY INTO THE CONSOLE!** Using it as a quick calculator may be OK (as we just did), but for the majority of your work and analyses, you want to write scripts, and write your code there.

These scripts are written on the source window, which looks like this:

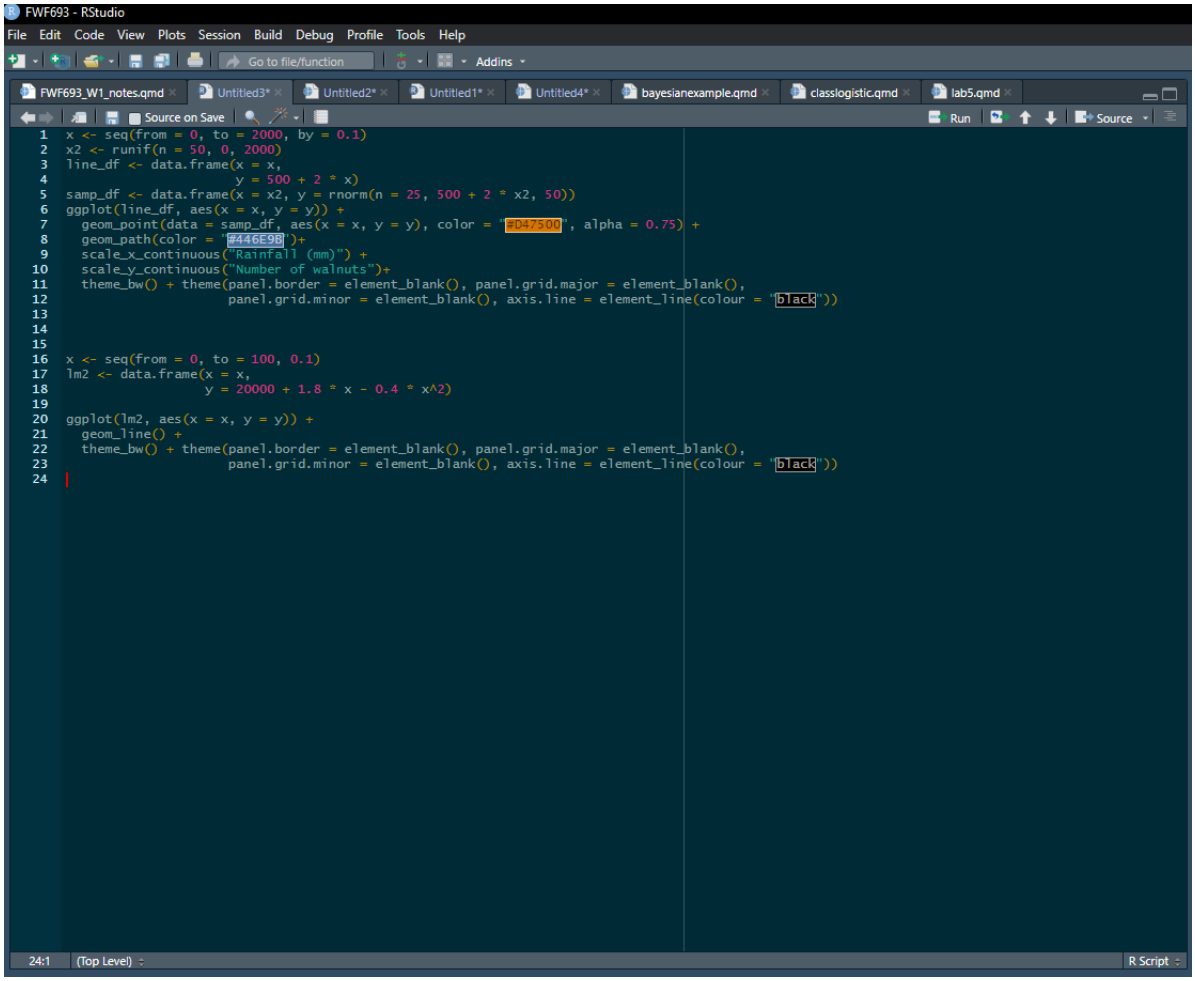


Figure 1.4: Source window

Here is where you can write all of your code and script. This is a code editor, so, we can re-run lines of code and modify/edit them for future use.

Try writing all of the code chunks you just ran, but instead of doing it in the console, write them in source window. To run each line of code you can use `ctrl` + `Enter` (at least on Windows, I am sure there must be a shortcut for macOS). You can also click run on top right corner of the window.

1.1.0.0.4 Creating and naming objects. Also; the environment and history pane

So far, you have been running r just as a calculator, but the strength of `R` is in running full scripts, models, and analyses. To do so, we need to use `R` less as a calculator and more as a programming software.

To create objects in R we use the `<-` operator. This is probably the most important operator in `R`. While you can create objects with `=` I would avoid this, as it can create some issues later on. This is how you create and name objects:

$$\underbrace{x}_{name} \underbrace{<-}_{symbol} \underbrace{17}_{value}$$

Here, we created an object called `x`, with a value of 17.

Try writing `x <- 17` in your script/code editor (not in the console!) and run it.

Now, let's create a new object called y

```
y<- x*2
```

Again run that.

As you can see, R is running these lines, but it is not printing the values.

If you look at the environment/history pane, you can see all of the objects that are loaded in R. This should include `x` and `y` and their values. You can also see their values by running the following code:

```
print(x)

print(y)
```

As you probably also noticed, y equals 34. This is because $y = 2x$ and $x = 17$. This is super useful! You don't need to use numbers every time, you can also use objects.

You can also print objects by simply running `y` or `x`:

```
y
```

1.1.0.1 Vectors

You can create vectors in different ways.

Way 1:

```
vector1<-c(4,6,1,11,50)
```

Run that and check the values of the vector. Does it make sense?

Vectors can be numeric, or other types:

```
Breed<-c("Holstein","Hereford","Longhorn","Longhorn","Longhorn","Hereford")
```

We can ask R about the class of an object.

```
class(vector1)

class(Breed)
```

We can also change classes:

```
Breed<-(as.factor(Breed))
Breed
class(Breed)
```

Here we changed from character to factor. See how it created a "level"? Essentially, R is making 3 groups and assigning each entry in the vector to one of the groups (that's a good way to think about it). It makes a lot of computational sense, and it is needed for modeling.

Now, let's try something fun. Let's go back to numeric vectors.

If you remember, x = 17 and y = 34. Run the following:

```
y-vector1
```

What happened when you did that? You are doing vector math! This can be incredibly useful!

Another way of creating a vector:

```
vector2<-1:100
vector3<-1001:1100
print(vector2)
print(vector3)
```

Run those lines, does it make sense?

If we want to extract a specific value, we can use `[]` as an index.

For example:

```
vector2[5]
```

Gives us the 5th value on vector2, which turns out is 5

Also, try the following:

```
vector3[5]
```

Which should give us the 5th value of vector3: 1005.

Now, try the following:

```
vector2+vector3
```

What happened? This is called vectorized math. Essentially, the first number in vector 2 was added to the fist number in vector 3, and then the second number on vector 2 was added to the second number on vector 3 and so on. This will be super useful for you!

There is another way of creating a vector:

```
vector4<-seq(1,100,4)
```

That can also be pretty useful! Do you understand what happened? You are creating a sequence of numbers! From 1

Actually, there are many, many, many ways of creating vectors. Which are simply a one dimensional matrix. Here are some examples that I use:

```
rnorm(50,0,5) #We sample 50 random values from a normal distribution with mean 0 and sd 5
rpois(100,10) #We sample 100 random values from a poisson distribution with lamba 10
runif(100,-10,10) #We sample 100 random individuals from a uniform distribution from -10 to
dpois(1:20,5) # The probability of obtaining a value of 1, 2, 3, ... ,20 if we were sampling
```

We won't have much time to go over probability distributions. If you are unfamiliar with them, as part of the class, the first reading will be chapter 4 and chapter 5 of An Intuitive, Interactive, Introduction to Biostatistics by Caitlin Ward and Collin Nolte . I would recommend you read those chapters before continuing (Ward and Nolte 2024)

> **Note**
>
> If you can't figure out what a "function" does. Using the help function might help. Use `?` to ask R about a function. Try running `?rnorm` and see if you can figure out what this

function does.
Again, no problem if you can't figure it out. By the end of the semester, you will! Also, what even is a function anyway?

1.1.1 Functions

`R` comes with tons of pre-built functions. You can also build your own, which can be a great way to do an analysis that you have to repeat multiple times. There are also `packages` that you can download to analyze your data. Essentially, a function is a code chunk that performs a task. However, it does require you to give it some information. That information is called `input`. Let's go back to the `runif` function. This function generates random deviates following a uniform distribution. However, to do that, it needs some information. The information that it needs is: `runif(n, min, max)` n is the number of deviates you want to generate. Min is the minimum potential value, and max is the maximum potential value (the limits of the uniform distribution. Try it:

```
runif(n=50, min=-20,max=20)
```

Let's try using `rnorm(n,mean,sd)` which generates n random variates using a normal distribution with mean "mean", and standard deviation "sd". Try it:

```
rnorm(20,0,1)
```

If you notice, we did not have to specify that 20 was n, 0 was the mean and 1 was the sd. `R` knew it already. `inputs` in `functions` have a pre-determined order.

We will explore a lot more functions in the next section.

1.1.2 Matrices and Data Frames

Our data usually have a different structure than just a single vector. We usually have excel files, databases, and dataframes with multiple columns and multiple rows. We will mostly work with dataframes during the course. Later on we will incorporate lists, and matrices with more than 2 dimensions though.

Matrices and dataframes in r have different characteristics. An important one is that **Matrices in R CANNOT contain different types of data. They all have to be either numerical, character, or logical or other.** While matrices are super useful, we will focus on using dataframes first. Dataframes can contain multiple types of data.

Now, let's work on some real data.

1.2 Reading data into R

You can import data of various formats into R; they include data tables in the form of `.dbf`, `.csv`, and `.xlsx` files or even spatial data such as vectors `.shp` or rasters `.nc`.

But the most common type of data files imported into R are probably `.csv` files.

You can import the dataset using the function `read.csv()`

Type this into R:

```
read.csv('C:/Users/amolina6/Documents/projects/FWF690/ButterflyData.csv')
```

The text within the brackets, and contained in quotation marks, is the file path (directory and file name) of the file you want to import. This is my directory. Yours would most likely be different. Perhaps something like…

IF MAC:

```
read.csv('~/FWF690/ButterflyData.csv')
```

IF WINDOWS:

```
read.csv('C:/Users/YourUserID/FWF690/ButterflyData.csv')
```

How do you find the file path?

For mac, open Finder, click to the folder where you saved your data, find your file, right-click (or ctrl-click) it, and then click on GetInfo.

For Windows, you can search documents, or you can look at the latest downloads on your browser.

When we read a datafile, we need to create an "R object". We can do it this way:

$$\underbrace{data}_{Object\ name} \quad \underbrace{<-}_{arrow} \quad \underbrace{read.csv}_{function} \underbrace{('C:/.../eel.csv')}_{Data\ location}$$

Object name is the name you are giving to the new object you are creating. `read.csv` is the function, you can also use `read_csv` from package `readR`

You can also do the following to import data into r:

1. You can change your working directory, to the folder where your data is located, and then you don't have to write the flepath. Check this website: https://dzchilds.github.io/eda-for-bio/working-directories-and-data-files.html (Childs 2024)
2. Create a project. If you create a project, the location of said project (folder) will be the working directory. If you download files directly into the folder, you can read them without specifying a working directory.

I highly recommend you create a project. This will make your life easier and that's what I do. Make sure to go to file > new project > new directory, and **select a location in your computer that will be easy for you to access.** At this point I would recommend you download **ALL** files directly into the project folder. Again, this is what I do, so I read files like this:

```
data<-read.csv("data/ButterflyData.csv", stringsAsFactors = T)
# raise your hand if you are  struggling to read this.
```

This file will only open if it's saved in the same working directory where you are currently working or in the same project.

> **Tip**
>
> Did you realize the **#** symbol in that last code chunk? This is a very useful symbol. It is used to comment the code, and everything after the symbol won't run. It's a good idea to comment all of your scripts.

Going back to our dataset, **head** shows us the top rows:

```
head(data)
```

```
  X Species Status Nparasites ForewingArea
1 1 Viceroy      1          2     964.8420
2 2 Monarch      3          0    1064.9463
3 3 Viceroy      2          2    1087.6290
4 4 Viceroy      2          3    1025.4572
5 5 Monarch      2          1     986.7462
6 6 Viceroy      2          2    1029.7973
```

We can use **summary** to obtain a summary of the dataset:

```
summary(data)
```

```
       X             Species       Status        Nparasites
 Min.   :  1.00   Monarch:41   Min.   :1.000   Min.   : 0.000
 1st Qu.: 34.25   Queen  :49   1st Qu.:1.000   1st Qu.: 0.000
 Median : 67.50   Viceroy:44   Median :2.000   Median : 2.000
 Mean   : 67.50                Mean   :2.075   Mean   : 2.246
 3rd Qu.:100.75                3rd Qu.:3.000   3rd Qu.: 3.000
 Max.   :134.00                Max.   :3.000   Max.   :10.000
  ForewingArea
 Min.   : 806.5
 1st Qu.: 949.4
 Median :1027.7
 Mean   :1022.7
 3rd Qu.:1087.1
 Max.   :1209.6
```

We can also ask it how many columns and how many rows it has. This is a great way to "count" observations. We use `ncol()` and `nrow()` to do this:

```
nrow(data)
```

```
[1] 134
```

1.2.0.1 Indices in dataframes

We can index dataframes using `[,]`, and you should think of it as `[rows,columns]` or, in the case of most dataframes: `[observation,variable]`. So, `[5,]` is the fifth observation:

```
data[5,]
```

```
  X Species Status Nparasites ForewingArea
5 5 Monarch      2          1     986.7462
```

While `[,5]` is the fifth variable, in this case "Forewing Area":

```
data[,5]
```

```
  [1]  964.8420 1064.9463 1087.6290 1025.4572  986.7462 1029.7973 1092.1279
  [8] 1030.1300  951.0215  952.0203  972.7687  923.5299  965.2786 1067.0532
 [15]  806.4759  933.6333  984.8791  900.8961  896.5896 1073.4148 1031.5598
 [22] 1122.3702  923.2774 1010.8952 1183.9889 1085.7107 1159.4511 1023.0000
```

```
 [29]  861.2154  906.4139 1046.9687 1165.4515 1037.9212  900.6176 1093.4250
 [36] 1050.7947  922.8024  929.6439 1045.5846  832.4090  984.0448 1059.0467
 [43]  886.6116  928.6071  992.8063 1117.8374 1012.8004 1057.8563 1174.0423
 [50] 1078.7093  982.5831 1141.2504 1117.7142 1049.4940  945.7766 1193.0925
 [57]  993.9152  900.3906  977.3681 1008.7073  937.7873  947.1193 1055.6590
 [64] 1134.0258 1092.5467  877.6170 1067.0919 1165.0152 1097.8645 1125.9920
 [71]  895.4531 1055.2484 1076.5924 1071.7679 1209.6463 1125.4865  968.7989
 [78] 1138.3152 1154.0843  936.4563  940.2080 1065.3797  884.6532  843.2516
 [85] 1000.2088  955.3850 1047.8846 1170.0051 1175.3559 1162.4729 1005.8265
 [92] 1013.7478  919.3033 1051.0724  942.7679 1178.7499  978.6767 1118.7880
 [99]  949.3146 1003.0768  967.5269 1165.6341 1016.6392 1100.9496 1106.7751
[106]  873.7377 1045.2197 1187.2821 1062.3207  936.8004  866.6742  921.0374
[113] 1111.3470 1019.4120  997.5843  878.4902  949.6300 1018.3885 1050.6530
[120]  891.1566 1050.1545  969.3831 1061.6462  998.0876 1069.8626 1103.7155
[127] 1075.5635 1083.0892 1080.7179 1025.5224 1066.6942  912.5556 1089.3433
[134] 1101.9849
```

Finally, you can get the fifth observation from the fifth variable:

```
data[5,5]
```

```
[1] 986.7462
```

Pretty useful, right? Variables usually also have names, and we can use the `$` operator to obtain specific variables (or columns) by name rather than by number. For example:

```
data$ForewingArea
```

```
  [1]  964.8420 1064.9463 1087.6290 1025.4572  986.7462 1029.7973 1092.1279
  [8] 1030.1300  951.0215  952.0203  972.7687  923.5299  965.2786 1067.0532
 [15]  806.4759  933.6333  984.8791  900.8961  896.5896 1073.4148 1031.5598
 [22] 1122.3702  923.2774 1010.8952 1183.9889 1085.7107 1159.4511 1023.0000
 [29]  861.2154  906.4139 1046.9687 1165.4515 1037.9212  900.6176 1093.4250
 [36] 1050.7947  922.8024  929.6439 1045.5846  832.4090  984.0448 1059.0467
 [43]  886.6116  928.6071  992.8063 1117.8374 1012.8004 1057.8563 1174.0423
 [50] 1078.7093  982.5831 1141.2504 1117.7142 1049.4940  945.7766 1193.0925
 [57]  993.9152  900.3906  977.3681 1008.7073  937.7873  947.1193 1055.6590
 [64] 1134.0258 1092.5467  877.6170 1067.0919 1165.0152 1097.8645 1125.9920
 [71]  895.4531 1055.2484 1076.5924 1071.7679 1209.6463 1125.4865  968.7989
 [78] 1138.3152 1154.0843  936.4563  940.2080 1065.3797  884.6532  843.2516
 [85] 1000.2088  955.3850 1047.8846 1170.0051 1175.3559 1162.4729 1005.8265
```

```
 [92] 1013.7478  919.3033 1051.0724  942.7679 1178.7499  978.6767 1118.7880
 [99]  949.3146 1003.0768  967.5269 1165.6341 1016.6392 1100.9496 1106.7751
[106]  873.7377 1045.2197 1187.2821 1062.3207  936.8004  866.6742  921.0374
[113] 1111.3470 1019.4120  997.5843  878.4902  949.6300 1018.3885 1050.6530
[120]  891.1566 1050.1545  969.3831 1061.6462  998.0876 1069.8626 1103.7155
[127] 1075.5635 1083.0892 1080.7179 1025.5224 1066.6942  912.5556 1089.3433
[134] 1101.9849
```

And if you want the fifth observation for this variable:

```
data$ForewingArea[5]
```

```
[1] 986.7462
```

Note than in this case `[]` is one dimensional. That is because `data$ForewingArea` returns a one dimensional (AKA vector) matrix with the values for this variable.

And if you want to obtain the first four columns of the dataframe you can do the following:

```
data[1:5,]
```

```
  X Species Status Nparasites ForewingArea
1 1 Viceroy      1          2     964.8420
2 2 Monarch      3          0    1064.9463
3 3 Viceroy      2          2    1087.6290
4 4 Viceroy      2          3    1025.4572
5 5 Monarch      2          1     986.7462
```

Look at that last result. The X column doesn't make sense. However, it is super easy to remove it. we can use `[,-1]` to remove the first column:

```
data<-data[,-1]
data
```

```
    Species Status Nparasites ForewingArea
1   Viceroy      1          2     964.8420
2   Monarch      3          0    1064.9463
3   Viceroy      2          2    1087.6290
4   Viceroy      2          3    1025.4572
5   Monarch      2          1     986.7462
```

6	Viceroy	2	2	1029.7973
7	Monarch	3	1	1092.1279
8	Viceroy	2	4	1030.1300
9	Viceroy	2	1	951.0215
10	Monarch	2	3	952.0203
11	Queen	1	9	972.7687
12	Queen	1	2	923.5299
13	Queen	1	7	965.2786
14	Monarch	3	0	1067.0532
15	Monarch	1	8	806.4759
16	Monarch	1	4	933.6333
17	Monarch	2	1	984.8791
18	Viceroy	1	5	900.8961
19	Viceroy	1	1	896.5896
20	Monarch	3	1	1073.4148
21	Monarch	3	2	1031.5598
22	Monarch	3	0	1122.3702
23	Queen	1	6	923.2774
24	Queen	2	4	1010.8952
25	Viceroy	3	0	1183.9889
26	Queen	2	2	1085.7107
27	Viceroy	3	0	1159.4511
28	Viceroy	2	0	1023.0000
29	Queen	1	3	861.2154
30	Viceroy	1	4	906.4139
31	Queen	2	4	1046.9687
32	Queen	3	0	1165.4515
33	Queen	3	0	1037.9212
34	Monarch	1	6	900.6176
35	Monarch	3	1	1093.4250
36	Monarch	3	0	1050.7947
37	Viceroy	1	10	922.8024
38	Viceroy	2	1	929.6439
39	Viceroy	2	4	1045.5846
40	Queen	1	2	832.4090
41	Queen	1	3	984.0448
42	Monarch	2	2	1059.0467
43	Monarch	1	0	886.6116
44	Queen	1	9	928.6071
45	Queen	2	2	992.8063
46	Viceroy	3	1	1117.8374
47	Monarch	1	6	1012.8004
48	Monarch	3	0	1057.8563

49	Queen	3	0	1174.0423
50	Queen	3	0	1078.7093
51	Queen	2	0	982.5831
52	Monarch	3	0	1141.2504
53	Viceroy	3	2	1117.7142
54	Viceroy	2	0	1049.4940
55	Viceroy	1	3	945.7766
56	Queen	3	0	1193.0925
57	Viceroy	2	2	993.9152
58	Queen	1	5	900.3906
59	Queen	2	2	977.3681
60	Queen	2	3	1008.7073
61	Monarch	1	5	937.7873
62	Viceroy	1	9	947.1193
63	Viceroy	3	0	1055.6590
64	Viceroy	3	1	1134.0258
65	Viceroy	3	0	1092.5467
66	Monarch	1	2	877.6170
67	Queen	3	0	1067.0919
68	Queen	3	2	1165.0152
69	Monarch	3	0	1097.8645
70	Queen	3	0	1125.9920
71	Viceroy	1	3	895.4531
72	Monarch	2	5	1055.2484
73	Queen	3	2	1076.5924
74	Queen	3	0	1071.7679
75	Queen	3	0	1209.6463
76	Monarch	3	2	1125.4865
77	Viceroy	2	4	968.7989
78	Queen	3	0	1138.3152
79	Monarch	3	0	1154.0843
80	Viceroy	1	3	936.4563
81	Viceroy	1	1	940.2080
82	Queen	2	7	1065.3797
83	Viceroy	1	4	884.6532
84	Viceroy	1	4	843.2516
85	Viceroy	2	2	1000.2088
86	Viceroy	1	0	955.3850
87	Queen	2	3	1047.8846
88	Monarch	3	1	1170.0051
89	Monarch	3	0	1175.3559
90	Queen	3	0	1162.4729
91	Viceroy	2	4	1005.8265

92	Monarch	2	4	1013.7478
93	Queen	1	8	919.3033
94	Queen	2	1	1051.0724
95	Viceroy	2	3	942.7679
96	Queen	3	1	1178.7499
97	Monarch	2	2	978.6767
98	Monarch	3	1	1118.7880
99	Queen	2	0	949.3146
100	Viceroy	2	2	1003.0768
101	Viceroy	1	6	967.5269
102	Queen	3	0	1165.6341
103	Queen	2	2	1016.6392
104	Monarch	3	1	1100.9496
105	Monarch	2	0	1106.7751
106	Queen	2	0	873.7377
107	Monarch	3	1	1045.2197
108	Queen	3	0	1187.2821
109	Viceroy	2	1	1062.3207
110	Viceroy	1	5	936.8004
111	Viceroy	1	3	866.6742
112	Viceroy	2	4	921.0374
113	Queen	3	1	1111.3470
114	Viceroy	2	3	1019.4120
115	Queen	2	4	997.5843
116	Monarch	1	5	878.4902
117	Queen	1	3	949.6300
118	Queen	2	2	1018.3885
119	Monarch	2	2	1050.6530
120	Monarch	1	9	891.1566
121	Monarch	2	1	1050.1545
122	Queen	2	1	969.3831
123	Monarch	2	0	1061.6462
124	Queen	2	1	998.0876
125	Monarch	3	0	1069.8626
126	Queen	3	0	1103.7155
127	Monarch	3	0	1075.5635
128	Viceroy	2	3	1083.0892
129	Queen	2	1	1080.7179
130	Viceroy	2	1	1025.5224
131	Queen	2	0	1066.6942
132	Monarch	1	7	912.5556
133	Queen	3	1	1089.3433
134	Viceroy	3	1	1101.9849

And now it's gone! Be very careful! Just run that line once. If you were to run it again you would remove the new "first" column (Species).

> **Note**
>
> While this worked great, the tidyverse (https://www.tidyverse.org/) is better for data transformation and cleaning than baseR (in my opinion). We will work on that later, but the R for Data Science book is available online for free: https://r4ds.hadley.nz/data-visualize I do recommend trying to master base R before moving into tidy or data.tables.

Finally, we can also use indices to obtain certain data that we are interested in. We can use `logical` operators for this. For example:

```
dataMonarch<-data[data$Species=="Monarch",]
head(dataMonarch)
```

```
   Species Status Nparasites ForewingArea
2  Monarch      3          0    1064.9463
5  Monarch      2          1     986.7462
7  Monarch      3          1    1092.1279
10 Monarch      2          3     952.0203
14 Monarch      3          0    1067.0532
15 Monarch      1          8     806.4759
```

This gives us a new dataframe that only contains monarchs.

These are called "logical operators". Here are the ones I use the most often:

Operator	Description
<	less than
<=	less than or equal to
>	more than
>=	more than or equal to
==	equal to
!=	different than
!x	not x
x & y	x AND y
isTRUE(x)	test if X is true
X%in%Y	is X in Y?

Another cool thing about indices, is that we can use functions on specific columns (or variables) of a dataframe. For example:

```r
mean(dataMonarch$ForewingArea)
```

```
[1] 1030.861
```

Gives us the mean of the Forewing Area for Monarchs.

Finally, we can also add columns using the `$` operator. The forewing area is in squared millimeters, if we wanted to calculate it in squared centimeters we could do the following:

```r
dataMonarch$FAcm<-dataMonarch$ForewingArea*0.01
head(dataMonarch)
```

```
   Species Status Nparasites ForewingArea      FAcm
2  Monarch      3          0    1064.9463 10.649463
5  Monarch      2          1     986.7462  9.867462
7  Monarch      3          1    1092.1279 10.921279
10 Monarch      2          3     952.0203  9.520203
14 Monarch      3          0    1067.0532 10.670532
15 Monarch      1          8     806.4759  8.064759
```

1.3 Projects

I really recommend you create a project for each specific research project you have. Common workflows have tons of problems, the main one is that R uses a global workspace. Without projects you are running and working on different analyses in the same global environment. As your code gets more complex, this will get more and more dangerous.

1.4 Quarto

You may or may not have heard of RMarkdown or Markdown. For your assignments, you will be using a similar program called Quarto.

Quarto is a multi-language, next-generation version of R Markdown from Posit. Just like Markdown and RMarkdown, Quarto is a plain text file editor. It has tons of advantages:

1. It allows to annotate your code
2. You can embed, run code and show plots.
3. This whole document was written using Quarto, by the end of the semester, I hope you are able to write full reports on Quarto

4. You can mix it with Github and have version control
5. Create a yearly report. You can write the report, and have sections that are data-dependent. As new data comes in, the report auto-updates.

If you want to learn about how to format in Quarto, you can use the following cheat sheet for RMarkdown (it uses the same syntax): https://posit.co/blog/the-r-markdown-cheat-sheet/

For me, the most important aspect of Quarto is embedding code (I call these "call chunks"). To do this you use the following symbol: three times. This way:

```
```{r}
```

The ` symbol is on the top left section of your keyboard. Next to the 1, and under the ~.

You can also go to insert > executable cell > R.

While I recommend you use scripts in your own research, I will have you use Quarto for your assignments in this class. This will print a "report" with all your answers, code and results, which will be easier for me to grade.

To create a new quarto file, go to File > New File > Quarto Document > Write a title and your name > Create.

Make sure the header has the following information, with the following header (this will make it pretty, and upload all the information):

```
title: "Title of document"
author: "Your name"
format:
  html:
    self-contained: true
editor: visual
```

You can copy and paste it.

⚠ Question 1 5pts

1. Create a new Quarto file (html). This is the file that you will upload to Canvas.
2. Write an explanation of what you think a statistical model is
3. Inside the Quarto file, using a code chunk, load the dataset, and then remove the first column `X` as we did earlier.
4. Look at the whole dataset
5. If you did your plot in R, try to embed it using code. If you used a different software option (or drew it), upload it to canvas as an independent file.
```

Question 2 5pts

1. In the same file, create an object with only monarchs, one object with only viceroy, and one with only queen.
2. Estimate the mean forewing area for each group. Do you think this differences are significant? Can we tell if the differences are because they are different species or because of some other variable (e.g., parasites)
3. Estimate the standard deviation of the forewing area for each group
4. Create a new object in which you only have observations that have a forewing area lower than 1000, AND 2 or more parasites.

Question 3 5pts

You came up with a new population butterfly condition index (BCI). The equation is fairly simple:

$$BCI_i = \frac{x_i}{\overline{x}}$$

where BCI_i is the index for individual i, x_i is the forewing area for individual i, and $\overline{x}$ is the sample mean. You need to estimate this for each individual, and you have to use a different $\overline{x}$ for each species (each species has its own mean forewing area).
Write the code needed to estimate this BCI for each individual.
tip: In the three objects you created (one for each species) add a column that estimates this BCI. Report he maximum and the minimum BCI (functions `max` and `min`) for each species.

Click Render, and upload the resulting html file to CANVAS. If you can't get it to work contact me or talk to me after class.

Finally, please let me know if this assignment and these types of exercises are useful! :)

Assignment 1

To create a new quarto file, go to File > New File > Quarto Document > Write a title and your name > Create.

Make sure the header has the following information, with the following header (this will make it pretty, and upload all the information):

```
title: "Title of document"
author: "Your name"
format:
  html:
    embed-resources: true
editor: visual
```

You can copy and paste it.

Question 1 5pts

1. Create a new Quarto file (html). This is the file that you will upload to Canvas.
2. Write an explanation of what you think a statistical model is
3. Inside the Quarto file, using a code chunk, load the dataset, and then remove the first column `X` as we did earlier.
4. Look at the whole dataset
5. If you did your plot in R, try to embed it using code. If you used a different software option (or drew it), upload it to canvas as an independent file.

Question 2 5pts

1. In the same file, create an object with only monarchs, one object with only viceroy, and one with only queen.
2. Estimate the mean forewing area for each group. Do you think this differences are significant? Can we tell if the differences are because they are different species or because of some other variable (e.g., parasites)
3. Estimate the standard deviation of the forewing area for each group
4. Create a new object in which you only have observations that have a forewing area

lower than 1000, AND 2 or more parasites.

Question 3 5pts

You came up with a new population butterfly condition index (BCI). The equation is fairly simple:

$$BCI_i = \frac{x_i}{\overline{x}}$$

where BCI_i is the index for individual i, x_i is the forewing area for individual i, and $\overline{x}$ is the sample mean. You need to estimate this for each individual, and you have to use a different $\overline{x}$ for each species (each species has its own mean forewing area).
Write the code needed to estimate this BCI for each individual.
tip: In the three objects you created (one for each species) add a column that estimates this BCI. Report he maximum and the minimum BCI (functions `max` and `min`) for each species.

Click Render, and upload the resulting html file to CANVAS. If you can't get it to work contact me or talk to me after class.

Finally, please let me know if this assignment and these types of exercises are useful! :)

Readings 1

Please Read Chapter 4 and 5 of (Ward and Nolte 2024).

Part II

The Central Limit Theorem and confidence intervals

In this section we are going to learn the following:

1. Plotting and using ggplot. I commonly use ggplot, although still use "baseR" quite often to plot. But the reality is that the ggplot package **is probably the best tool and package to plot in R.** We will learn to make plots using ggplot and we will use it throughout the course.
2. Data simulation.
3. The central theorem and confidence intervals
4. Introduction to linear models

This is a "longer" assignment. But the next one will be shorter. You can take as long as you need on it

You can download the quarto document (from Canvas) and use it to answer the questions, copy code, modify code, etc.

Remember!

> **Note**
>
> If you are comfortable with R basics, you can write the code for this exercise directly into Quarto. To add extra code chunks, write this symbols: "'{r}, if you are still not comfortable with R, I recommend for the time being you open a new file and copy and paste the code there, and run it. You can submit this as a qmd file or as an r-script. Whatever you feel more comfortable with at the moment (.qmd + .html preferred)
> You can also go to insert > executable cell > R and that should insert an r chunk

> **Why is my data not loading?**
>
> Make sure you download the data file, and also save the quarto file you're working on in the class folder!

Part III

2 Exploring the CLT

The applet on Ward and Nolte's book (Ward and Nolte 2024) is a great example of the central limit theorem. If you still have questions about this topic I recommend you check the applet on section 6.3 and the one on section 7.2.

We are going to do a little experiment regarding the Central Limit Theorem and confidence intervals. We are going to:

1. Simulate a population. This population can be anything, and can have any distribution.
2. Take a sample of size n = 30
3. Estimate the mean, and CI of the sample
4. Compare it to the population. Did we get it right? Is the real mean encompassed in the population?
5. Do it again… and again… and again
6. Do it 1000 times
7. Plot this using ggplot

This may seem like a lot… but it is not! It can be done fairly easily using R. And at this point I will still guide you step by step.

> If you feel like you are falling behind
>
> I cannot stress this enough. If you feel like you are falling behind, I can meet you MWF before or after class, or you can meet with me in my office to go over coding, any topic, or any issues you may be having.

Let start

2.0.0.1 Step 1: Simulate your data

This may be the most complicated step. We haven't really talked about distributions, and you're not expected to know this at this point. If you are not comfortable simulating your own data, I am providing some examples below. Choose the one you like the most, and use it to simulate your data. If you decide to use one of the provided examples, then you can skip this, and go directly to the Section 2.0.0.2 section. If you want to simulate your own population, look at the examples.

> **Warning**
>
> Be careful! This data is simulated, so each time you run the code it will generate a new set of data. Make sure to simulate your population only once!

If you want to simulate the example, let's do the following:

1. Decide what system you are trying to simulate
2. Create an object called "data" in which you will simulate 10,000 datapoints generating 10,000 random deviate using the distribution of your choice
3. Obtain the population parameters (mean and sd). Call them mu and sd.

2.0.0.2 Examples

I have different examples. In order to simulate the data, simply run the example that you want. Only run one example (the one you want) or they will overwrite themselves. These have different distributions

Please be aware that I am not super knowledgeable about these systems, so the units may be completely off. I tried my best though!

2.0.0.2.1 Example 1: bats

This data-set is on the wing aspect ratio of the pygmy round-eared bat.

```
data<-rnorm(10000, mean = 5.5, sd=0.75) #your population, 10,000 individuals total
mu<-mean(data) #real population parameter
sigma<-sd(data)
```

2.0.0.2.2 Example 2: frogs

This data-set is on the presence-absence data of Chytridiomycosis in a population of *Lithobates clamitans*This is a binomial distribution. 1 is presence and 0 is absence

```
data<-rbinom(1000,1,0.23) #your population, 10,000 individuals tota.
mu<-mean(data) #real population parameter
sigma<-sd(data)
```

2.0.0.2.3 Example 3: grapes

This dataset is on the number of wild grapes in 10,000 vines in a meadow (you can also make this wine grapes in a vineyard if you are into that).

```
data<-rpois(1000,40) #your population, 10,000 individuals total.
mu<-mean(data) #real population parameter
sigma<-sd(data)
```

2.0.0.2.4 Example 4: Beef

Annual live weight (LW) meat production in 10,000 hectares (n = 10,000), each hectare represents one datapoint.

```
data<-rnorm(10000, mean = 410.5, sd=60) #your population, 10,000 individuals total
mu<-mean(data) #real population parameter
sigma<-sd(data)
```

> ⚠ Question 1
>
> Come up with some potential systems/examples that I can simulate or use for future students. Coming up with examples is probably the most challenging aspect of teaching this course! You don't have to come up with actual numbers. Just ideas!

2.0.0.3 Step 2: Take a sample

Before taking a sample, write down or make a mental note of the real population mean:

```
mu
```

```
[1] 410.3065
```

We are going to simulate taking a sample of size 40. It is actually pretty straightforward to take a sample:

```
n<-40
sample1<-sample(data,n)
sample1_mean<-mean(sample1)
s<-sd(sample1)
```

Now, let's calculate the standard error. Remember, SE is:

$$SE = \frac{s}{\sqrt{n}}$$

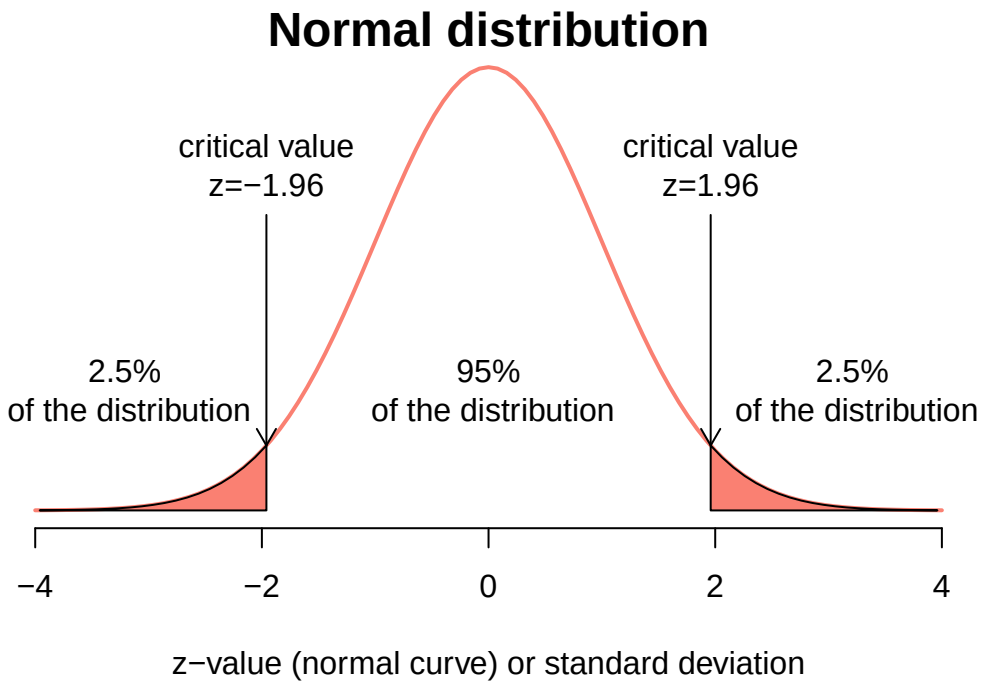
Complete the code to estimate SE:

```
SE1<-s/sqrt(n)
```

If you remember the class, we said that usually to estimate confidence intervals we use the following equation:

$$LowerCI = \overline{y} - 1.96SE$$
$$UpperCI = \overline{y} + 1.96SE$$

But why? Look at the following plot:



So, our confidence interval is called a 95% confidence interval, and has an $\alpha = 0.05$ (think of this as a 5% chance of the parameter not included within the confidence interval). We can actually use any CI value we want, but 95% is usually used. It is OK to use a critical value of 2 or one of 1.96

We can also estimate the critical values for any alpha we want. For this, we use the following code:

```r
qnorm(0.025) #for lower CI
```

```
[1] -1.959964
```

```r
qnorm(0.975) #for highert CI
```

```
[1] 1.959964
```

2.0.0.3.1 Assignment question 2:

> **Question 2**
>
> See the last code chunk, and answer: Why do we use 0.025 and 0.975 instead of 0.95? If you can't figure it out, that's ok, but we should meet at some point so I can explain it .

We can also use the critical value (in this case $\alpha = 0.05$) to find our critical values with the following function:

```r
alpha<-0.05
qnorm(alpha/2)
```

```
[1] -1.959964
```

```r
qnorm(1-alpha/2)
```

```
[1] 1.959964
```

Try a different critical value.

Also, while I am estimating the critical values for both lower and upper CI's, you only need to do one because they are symmetric. We also will very rarely use the normal distribution and will be using a different distribution called "t distribution", and can estimate the critical values using `qt()`. But for now, lets stick to the normal distribution.

Ok, let's go back to our sample:

We can estimate the CI's now, and check whether the real parameter mean is included within them. Remember, you estimated the real mean and named it mu:

2.0.0.3.2

The following code is used to estimate confidence intervals.

```
UCI1<-sample1_mean+1.96*SE1
LCI1<-sample1_mean-1.96*SE1
```

And you can use the following code to check whether the real mean (mu) is included within the confidence intervals you estimated!

```
if(mu<UCI1 & mu> LCI1){
  print(paste0("Nice! The real parameter ", mu, " is included within the confidence interval
} else {print(paste0("Bummer! The real parameter ", mu, " is NOT included within the confide
```

```
[1] "Nice! The real parameter 410.306498411648 is included within the confidence interval. LC
```

In class I mentioned examples in which I would sample the same population multiple times. Let's do that, let's start by taking a second sample and estimating the confidence interval:

```
sample2<-sample(data,n)
sample2_mean<-mean(sample2)
s2<-sd(sample2)

SE2<-s2/sqrt(n)

UCI2<-sample2_mean+1.96*SE2
LCI2<-sample2_mean-1.96*SE2
if(mu<UCI2 & mu> LCI2){
  print(paste0("Nice! The real parameter ", mu, " is included within the confidence interval
} else {print(paste0("Bummer! The real parameter ", mu, " is NOT included within the confide
```

```
[1] "Nice! The real parameter 410.306498411648 is included within the confidence interval. LC
```

Run the code again for sample 3

Now, I want you to do this 100 times (take a sample, estimate the CI, check if the real mean is included). This is just like in the applet from the book you read this week..

This might seem like a lot of work, but we can use a thing called `for loops`. Keep reading!

A `for` loop allows us to iterate over a sequence. For example:

```r
for (x in 1:5){
  print(x*2)
}
```

```
[1] 2
[1] 4
[1] 6
[1] 8
[1] 10
```

Some important aspects of that last for loop:

$$\underbrace{for}_{initiate\ loop}\ \underbrace{(x\ in\ 1:5)}_{vector}\underbrace{print(x*2)}_{code\ block}$$

1. You start the for loop using `for`
2. You give a vector of values. During the `for` loop, x will take the first value in the vector and run the code block. After this, x will take the second value and run the code block. It will continue doing this until it reaches the last value. This is the "last iteration". After the last iteration, it will exit the for loop.

Let's use a `for` loop to take100 samples.

First, let's create some empty vectors to store our results. I like to create vectors before running the `for` loop because it is computationally faster than concatenating results as the loop goes. We will create vectors of size 100:

```r
experiment<-1:100
means<-rep(NA,100) #For the means
s<-rep(NA,100) #sd
SE<-rep(NA,100) #SE
LCI<-rep(NA,100) # Lower CI
UCI<-rep(NA,100) # Upper CI
inclusive<-rep(NA,100) #is the real parameter included?
alpha<-0.05 #our alpha
lower<-qnorm(alpha/2) #lower critical value
upper<-qnorm(1-alpha/2) #upper critical value
```

We can now run the code:

```
for(i in 1:100){
sample<-sample(data,n)
means[i]<-mean(sample)
s[i]<-sd(sample)
SE[i]<-s[i]/sqrt(n)
LCI[i]<-means[i]+SE[i]*lower
UCI[i]<-means[i]+SE[i]*upper
if(mu<UCI[i] & mu> LCI[i]){
  inclusive[i]<-T} else{
    inclusive[i]<-F
  }
}
```

That is pretty cool. You can run for loops with thousands of iterations and it will run in seconds!

We would expect that about 95 of our runs had the real parameter included within the estimated CI's. There is a super easy way to estimate this:

```
sum(inclusive)
```

```
[1] 91
```

That's it! It will give you the number of "successes".

At the end of the document, I have mroe information on for loops, and a "challenge" if you want to create one.

Let's create a dataframe in which we store all of these results.

```
df<-data.frame(experiment,means,s,SE,LCI,UCI,inclusive)
```

Now, let's plot it. We will use `ggplot`. `ggplot` is a package, and packages need to be installed before you are able to use them. Similar to computer software, you **only need to install a package once in a computer**. Let's install `ggplot` (if you already have ggplot installed, you can skip this step

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

After you have installed a package, you need to load it before using it using the `library` command. Again, think of this as any software in your laptop. After you install it, you need to open the program before you run it:

```
library(ggplot2)
```

```
Warning: package 'ggplot2' was built under R version 4.3.3
```

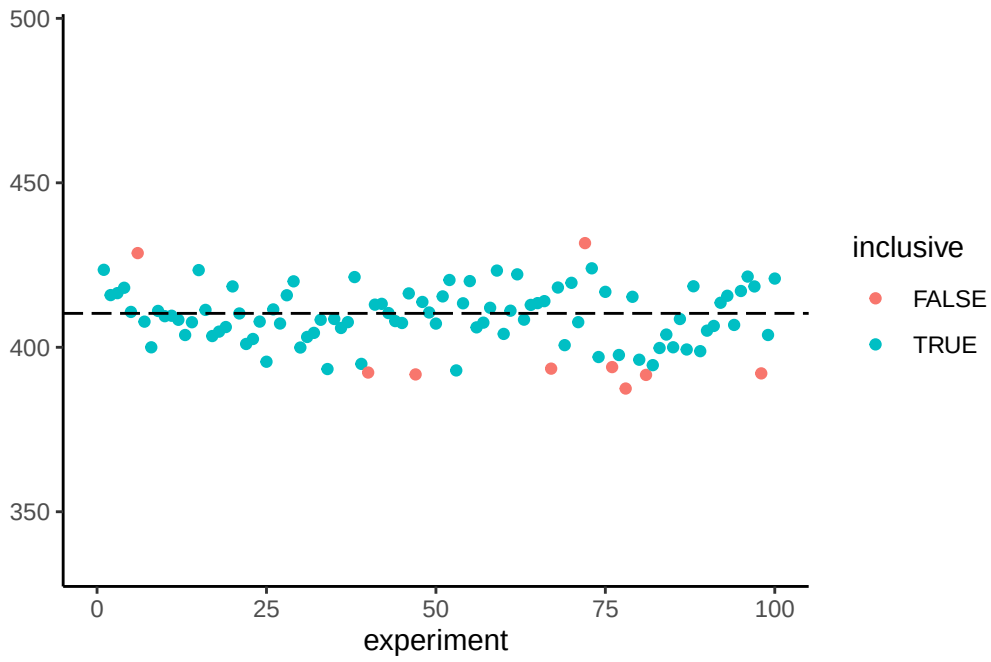
This is how ggplot works:

$$\underbrace{ggplot}_{initiate\ plot}\ (\underbrace{data = df}_{data\ frame},\ \underbrace{aes(x = , y =)}_{plot\ attributes}) + \underbrace{geom\_line()}_{geometry\ line} + \underbrace{geom\_point()}_{geometry\ point} + \underbrace{theme()}_{customize}$$

In that example we have two geometries: lines and points. So, in that case we would be plotting lines and points (easy, right?). If you want to see all the available geometries, visit: https://ggplot2.tidyverse.org/reference#geoms

We will use ggplot to plot all of the points:

```
ggplot(df, aes(x = experiment, y = means,colour=inclusive)) +
   geom_point() +
   geom_hline(yintercept = mu, color = "black", linetype = "longdash") +
   scale_y_continuous(name = "", limits = c(min(df$LCI) - min(df$LCI)/10 , max(df$UCI + max(
   scale_x_continuous(limits = c(0, 100)) +
   theme_classic()
```



Three things you may be thinking about this plot:

1. Plotting it seems harder than I made it seem! There are more lines there
2. It's a lot of code, and I'm not sure how good it looks
3. It's missing the confidence intervals!

I agree with all of that. However, as class goes by I hope you will understand all the sections of the plot and make nicer plots than this!

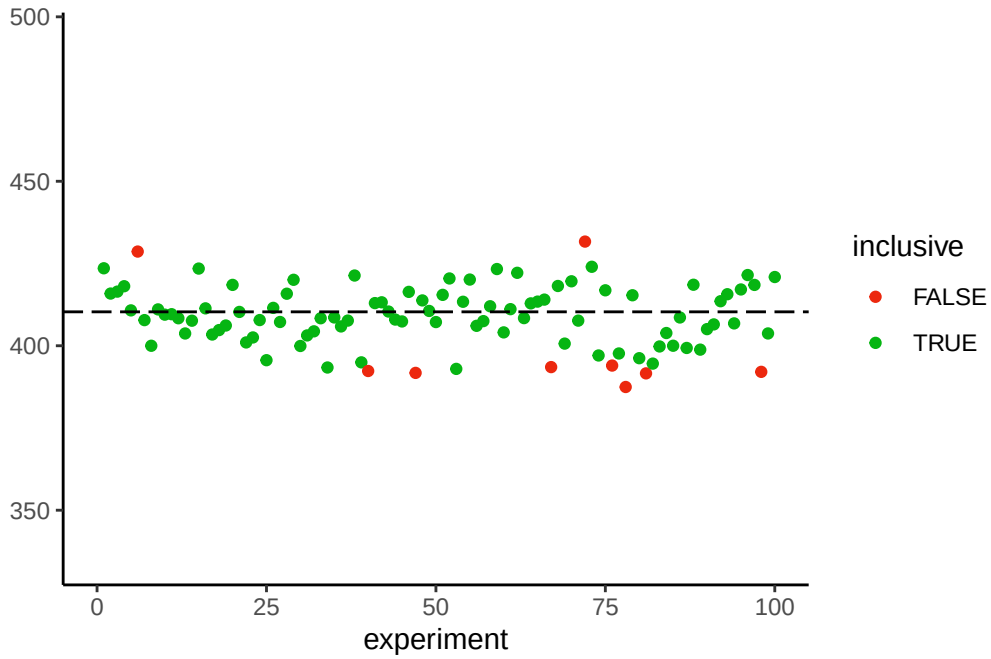
For the time being, let's inspect each element:

1. `ggplot(df, aes(x = experiment, y = means,colour=inclusive))` initiate the plot, provides the data, and provides values for x, y, and for the color
2. `geom_point` draws the geometry (point) with our provided elements
3. `geom_hline` is drawing a horizontal line. We need to provide an actual value (mu in this case). Therefore it represents the real parameter
4. `scale_y_continuous` and `scale_x_continuous` provide information for the axes. You won't use this 99% of the time, and no need to worry about it right now. In this case I created y axis to be able to easily add the confidence intervlas in one of the following steps.
5. `theme_classic` is providing a theme. Try running it without a pre-defined theme. I don't like the ggplot standard theme

Finally, please notice how we add elements using `+`.

Now, I don't like the colors of the dots. I am going to change them:

```
ggplot(df, aes(x = experiment, y = means,colour=inclusive)) +
   geom_point() +
   geom_hline(yintercept = mu, color = "black", linetype = "longdash") +
   scale_y_continuous(name = "", limits = c(min(df$LCI) - min(df$LCI)/10 , max(df$UCI + max(d
   scale_x_continuous(limits = c(0, 100)) +
   theme_classic()+
  scale_color_manual(values=c("#ec280e", "#06b512"))
```

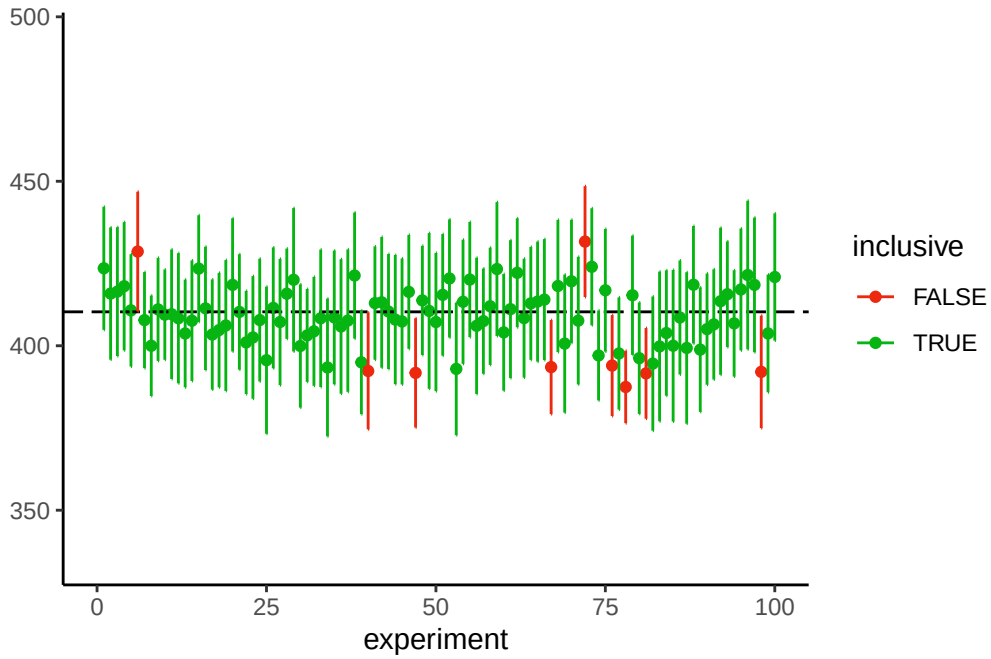


I like those colors better. There are many ways to choose colors, but I like giving R the HEX code of the color I want. I usually use a color picker: https://htmlcolorcodes.com/color-picker/ to pick the color, and that supplies me with the HEX code.

> ⚠ Question 3
>
> Change the plot colors

Finally, I want to add the CI's:

```
ggplot(df, aes(x = experiment, y = means,colour=inclusive,ymin = LCI, ymax=UCI)) +
   geom_point() +
   geom_hline(yintercept = mu, color = "black", linetype = "longdash") +
   scale_y_continuous(name = "", limits = c(min(df$LCI) - min(df$LCI)/10 , max(df$UCI + max(
   scale_x_continuous(limits = c(0, 100)) +
   theme_classic()+
  scale_color_manual(values=c("#ec280e", "#06b512"))+
  geom_errorbar(width = 0)
```



Notice how I added `ymin` and `ymax` to `aes()`? I am providing more plot attributes. I also added `geom_errorbar(width = 0)`.

3 One sample T-test

Finally! We will work on a one-sample t-test.

Example: The Jackson Laboratory (Bar Harbor, ME) shipped you some lab mice from the C57BL/6J strain. You chose this specific strain because their mean mass at 21 weeks is 20.2 grams, which is important for your experiments. However, you suspect your mice set might have a different weight, so, you take a random sample of 21 individuals to check.

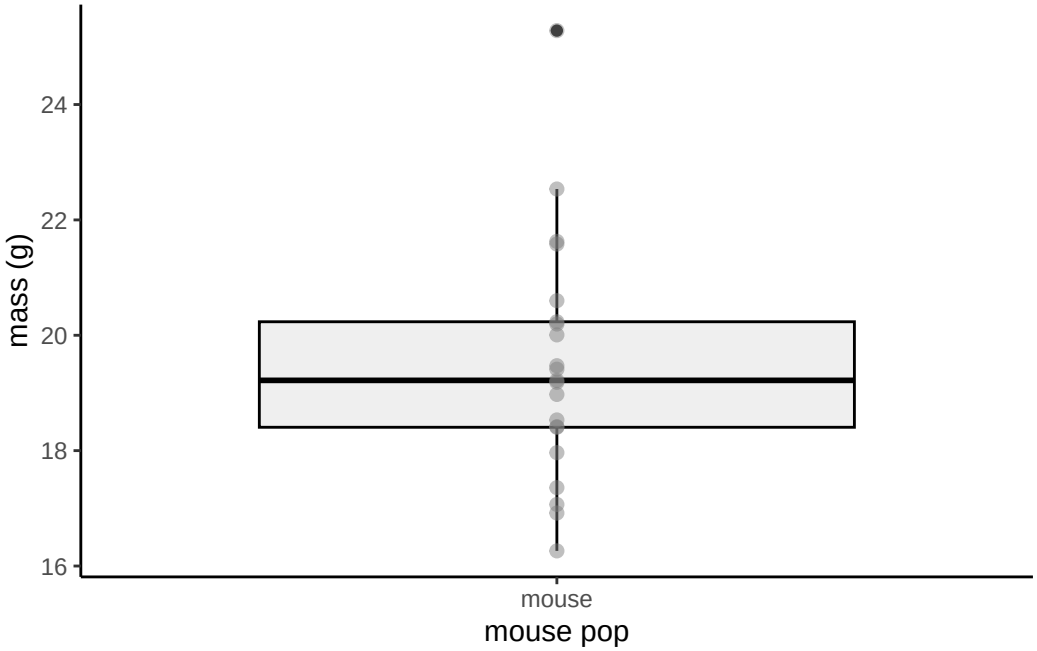
Download the "mousedata.csv" file and load it in an object named mousedata and inspect it

```
mousedata<-read.csv("data/mousedata.csv")
mousedata<-mousedata$x
mousedata
```

```
 [1] 18.41489 17.96535 20.23447 16.91896 19.21828 25.28136 17.35877 18.97449
 [9] 21.58163 16.26229 18.40497 20.59939 19.41167 20.00668 18.53189 17.06733
[17] 19.18580 22.53532 21.62798 20.19551 19.47198
```

Let's plot the data:

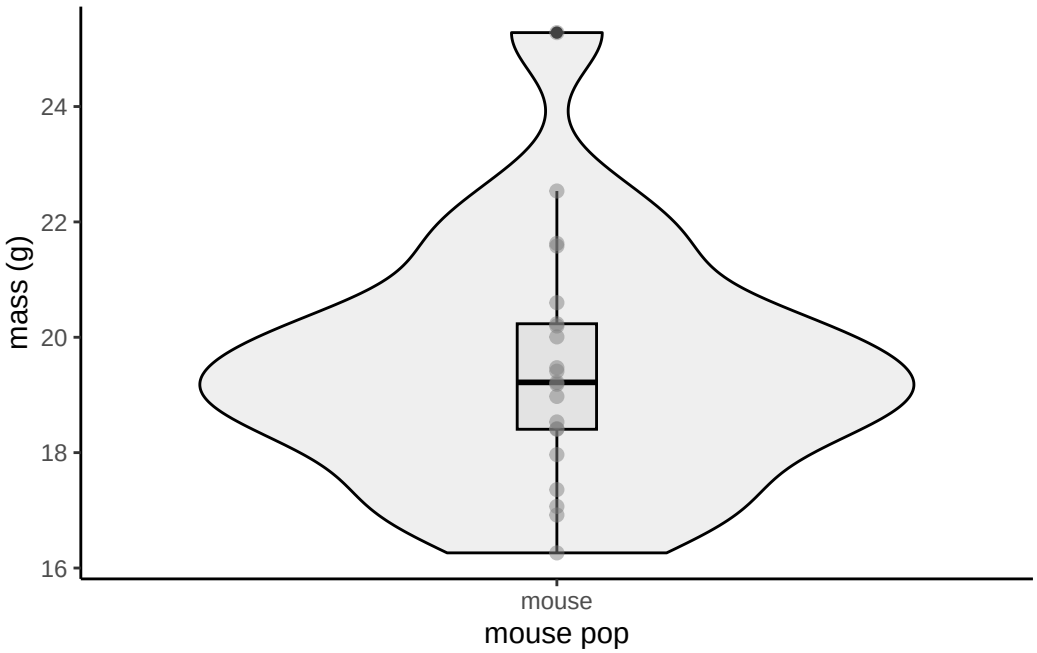
```
df1<-data.frame(mousedata,mouse="mouse")
ggplot(data=df1,aes(x=mouse,y=mousedata))+
  geom_boxplot(fill=gray(0.7,0.2),color="black")+
  geom_point(size=2,col=gray(0.5,0.5))+
   theme_bw() + theme(panel.border = element_blank(), panel.grid.major = element_blank(),
                      panel.grid.minor = element_blank(), axis.line = element_line(colour = "
  ylab("mass (g)")+
  xlab("mouse pop")
```



Not a very good looking plot. And we are not comparing groups, so, a bit pointless. But you will be plotting the next example. It also brings an important issue:

Boxplot is probably the most commonly used chart type to compare distribution of several groups. However, you should keep in mind that in a boxplot data distribution is hidden behind each box. For instance, a normal distribution could look exactly the same as a bimodal distribution. Please consider a violin plot or a ridgline chart instead. For reference, here is how to do a violin plot:

```
df1<-data.frame(mousedata,mouse="mouse")
ggplot(data=df1,aes(x=mouse,y=mousedata))+
  geom_violin(fill=gray(0.7,0.2),color="black")+
  geom_boxplot(fill=gray(0.7,0.2),color="black",width=0.1)+
  geom_point(size=2,col=gray(0.5,0.5))+
   theme_bw() + theme(panel.border = element_blank(), panel.grid.major = element_blank(),
                      panel.grid.minor = element_blank(), axis.line = element_line(colour = "
  ylab("mass (g)")+
  xlab("mouse pop")
```



After plotting, the question is: is the average mass of your mice equal to 20.2 grams?

You are not measuring all the individuals, only 21

The relevant hypotheses would be:

- $$Ho : \mu_1 = 20.2$$
- $$Ha : \mu_1 \neq 20.2$$

Hopefully you understand this based on the readings, and some previous stats classes. If it's not clear, talk to me.

We will be doing a t-test. In a t-test we are essentially estimating the probability of observing a result as (or more) extreme as what we observed given that Ho was true. Essentially, if the actual mean of your mice is 65-g, what are the chances ob obtaining the sample you did. The further your sample mean is 65, the lower the chances of observing that result.

We will also estimate a "critical value" that corresponds to an α of 0.05. While now we can estimate p directly with r, you may remember your stats courses where you were tasked with finding a critical value, in order to decide whether to reject or fail to reject the null hypothesis.

We cannot use 1.96 because our sample is small (and we are doing a t-test). Therefore, estimating a critical value from t distribution is better.

We will use the following code to obtain our critical value:

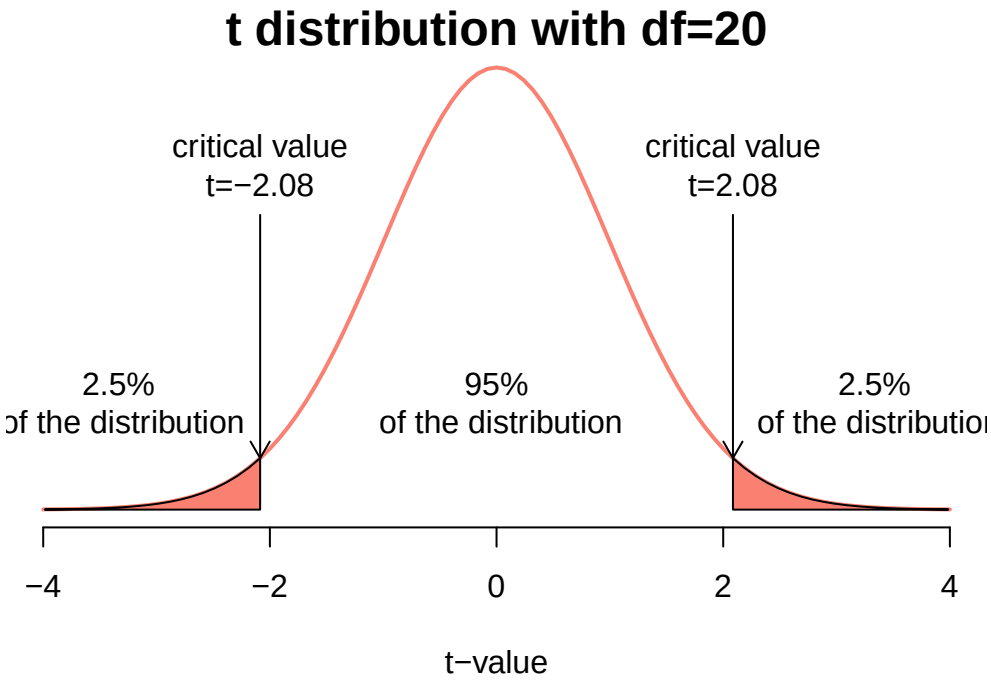
```
alpha<-0.05
qt(c(alpha/2, 1-alpha/2), df=21-1)
```

```
[1] -2.085963  2.085963
```

Our degrees of freedom are n-1. And our critical values are -2.08 and 2.08. Any value lower than -2.08 or higher than 2.08 will be significant (i.e., we reject the null hypothesis)

> **!** Think about it
>
> Why are degrees of freedom n-1?

Essentially, our critical values show the following:



This is the t-distribution for our example.

If you are curious as to what happens to the t distribution as df goes up, try calculatin it for 10,000 df. It actually becomes 1.96 and is no different than a normal distribution.

Now, let's solve our problem. Remember, our linear model looks like:

$$y_i = \beta_0 + \epsilon_i \qquad \text{where} \qquad \epsilon_i \sim \mathrm{N}(0, \sigma)$$

3.0.1 Linear model

Running this dataset as a linear model is very straightforward. I talked about this in class.

```
model1<-lm(mousedata~1)
summary(model1)
```

```
Call:
lm(formula = mousedata ~ 1)

Residuals:
    Min      1Q  Median      3Q     Max
-3.2258 -1.0831 -0.2698  0.7464  5.7933

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  19.4880     0.4571   42.63   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.095 on 20 degrees of freedom
```

> **Question 4**
>
> Interpret the results. What is the estimated mean? What are the confidence intervals? Do you think this population is significanlty different from 20.2?

You can also run a linear model like this:

```
model1alternative<-lm(mousedata-20.2~1)
summary(model1alternative)
```

```
Call:
lm(formula = mousedata - 20.2 ~ 1)
```

```
Residuals:
    Min      1Q  Median      3Q     Max
-3.2258 -1.0831 -0.2698  0.7464  5.7933

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.7120     0.4571  -1.557    0.135

Residual standard error: 2.095 on 20 degrees of freedom
```

What is different in this case?

In this case, we can actually interpret the intercept. Try to include an interpretation of the intercept estimate and P-value as part of question 4.

If you run the test using this alternative way, the t value of the intercept actually corresponds to the t value of a t-test. Essentially, r is running a t-test. Hopefully, this is the way you will be running t-tests in the future. In this case -5.33 is lower than the critical value. What does that mean?

3.0.2 Traditional way

We can also estimate the actual t-statistic using an equation:

$$t = \frac{\bar{y} - \mu_0}{\frac{s}{\sqrt{n}}} \qquad \text{where} \qquad \frac{s}{\sqrt{n}} = \frac{\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})^2}}{\sqrt{n}}$$

Let's do this. It looks super complicated, but R is doing all the heavy lifting.

1- Obtain the mean of the mouse mass

```
y_bar<-mean(mousedata)
```

2- Obtain the standard error

```
se_y <- sd(mousedata)/sqrt(length(mousedata))
```

3- Calculate t statistic

```
t_stat <- (y_bar - 21)/se_y
t_stat
```

```
[1] -3.307546
```

And that's it!

This should be the same as the linear model. We can compare this to the critical value

```
alpha<-0.05
qt(c(alpha/2, 1-alpha/2), df=21-1)
```

```
[1] -2.085963  2.085963
```

and you are done. Again, the linear model function does the same :)

Now, we can also run a ttest using the `t.test` function

> **Note**
>
> I would recommend you use linear models rather than running t-tests.

```
t.test(mousedata,mu=20.2)
```

```
    One Sample t-test

data:  mousedata
t = -1.5575, df = 20, p-value = 0.135
alternative hypothesis: true mean is not equal to 20.2
95 percent confidence interval:
 18.53451 20.44159
sample estimates:
mean of x
 19.48805
```

Again, this result should be the same exact one.

4 Two sample T-test

For this example, we will only run a linear model using `lm` and a t-test using `t.test`.

First off, download the file named tunadata and save it in the same folder as your .qmd file:

```
tunadata<-read.csv("data/tunadata.csv")
head(tunadata)
```

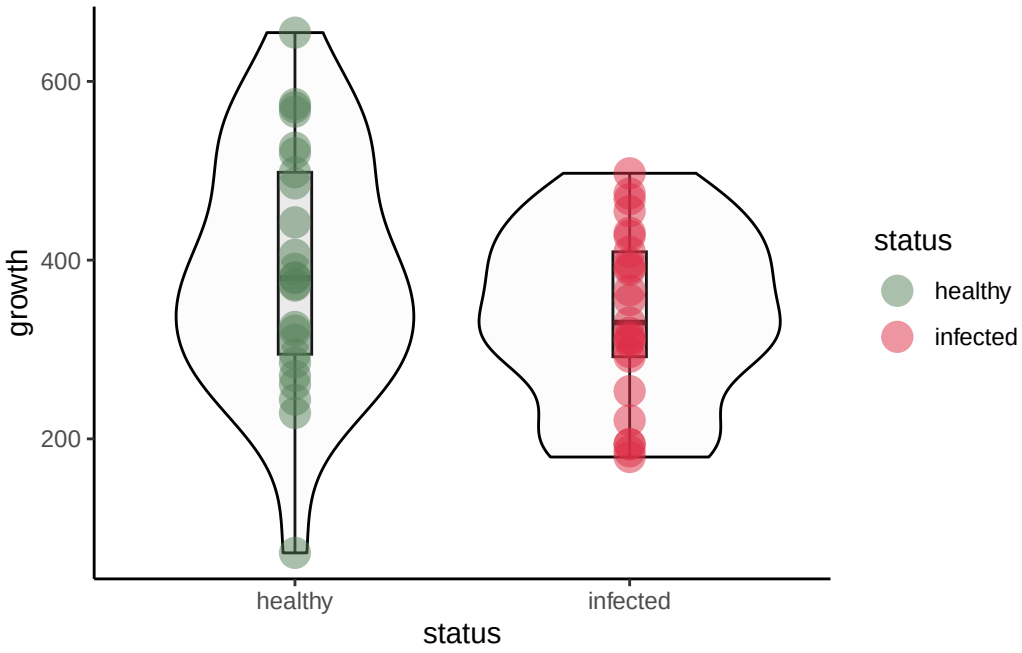
```
  growth   status
1  474.6 infected
2  313.4 infected
3  194.2 infected
4  442.3  healthy
5   72.4  healthy
6  406.0  healthy
```

Here we have data of . We have measured the "growth" (total length) of the tuna, and recorded whether they are healthy, or they are infected by a parasite that you suspect affects growth.

> ⚠ Question 5
>
> What are the null and alternative hypotheses?

Now, let's plot the data:

```
ggplot(data = tunadata, aes(x = status, y = growth)) +
  geom_boxplot(fill=gray(0.7,0.25),color="black",width=0.1) +
  geom_violin(fill=gray(0.9,0.1),color="black")+
 theme_bw() + theme(panel.border = element_blank(), panel.grid.major = element_blank(),
                    panel.grid.minor = element_blank(), axis.line = element_line(colour = "
  geom_point(aes(color = status), size = 5, alpha = 0.5)+
  scale_color_manual(values=c("#57825a","#dd2e46"))
```



Now, let's run a linear model.

Remember, the equation is:

$$y_i = \beta_0 + \beta_1 x_{1,i} + \epsilon_i \qquad \text{where} \qquad \epsilon_i \sim \mathrm{N}(0, \sigma)$$

You will use this equation:

```
model2<-lm(growth~status,data=tunadata)
summary(model2)
```

```
Call:
lm(formula = growth ~ status, data = tunadata)

Residuals:
    Min      1Q  Median      3Q     Max
-317.76  -83.56   -9.86   91.37  264.44

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)      390.16      23.55  16.569   <2e-16 ***
statusinfected   -51.76      33.30  -1.554    0.127
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 117.7 on 48 degrees of freedom
Multiple R-squared:  0.04791,   Adjusted R-squared:  0.02808
F-statistic: 2.416 on 1 and 48 DF,  p-value: 0.1267
```

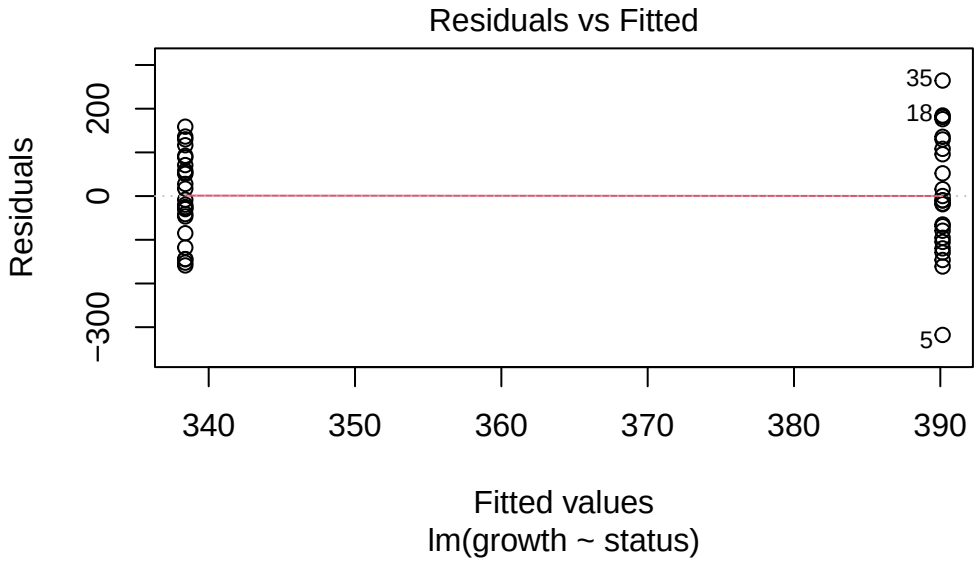
4.0.0.0.1 Assignment question 9:

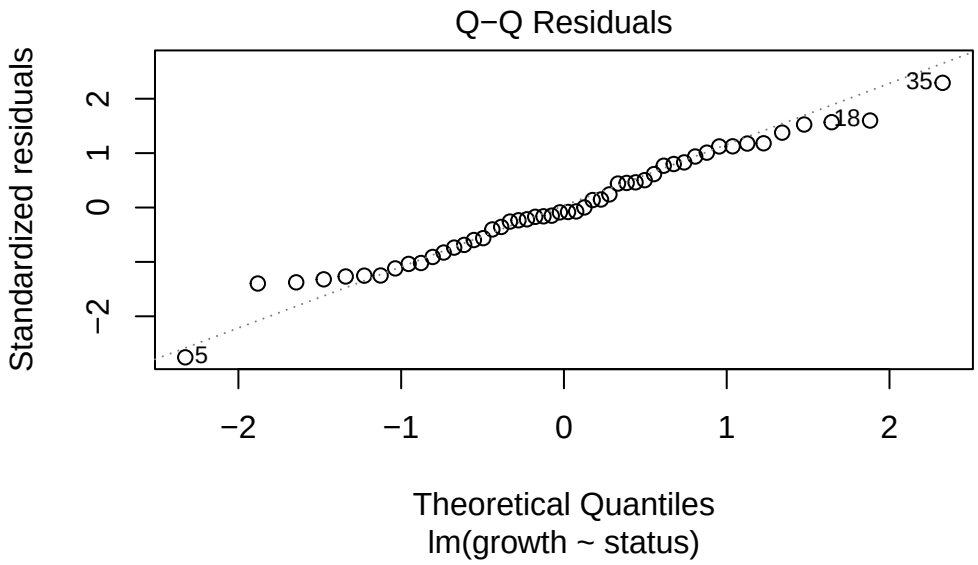
Using this output, answer the following:

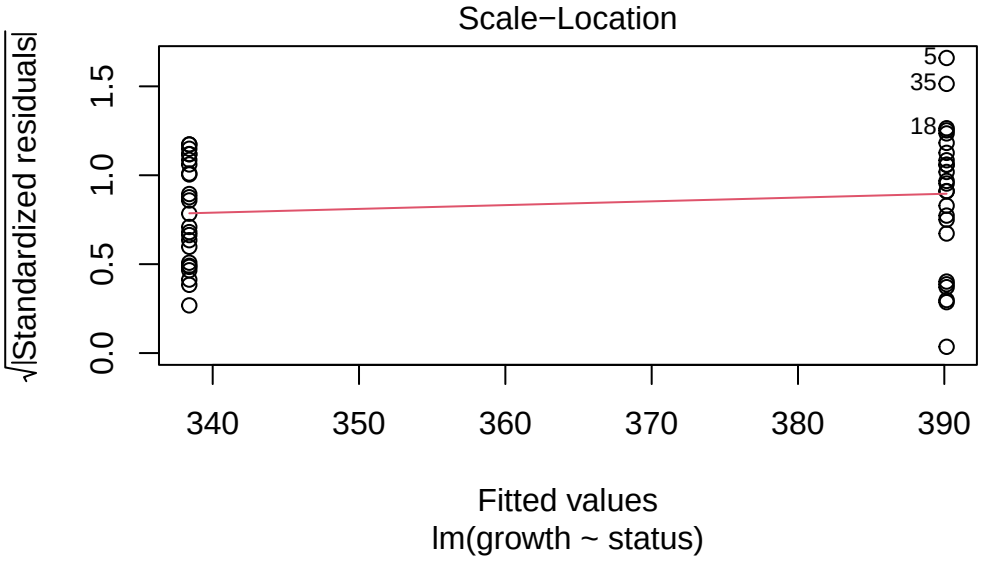
1. Where there any differences? (we will discuss this on Wednesday)
2. What is the estimated mean for infected individuals? Calculate it using only the coefficients!

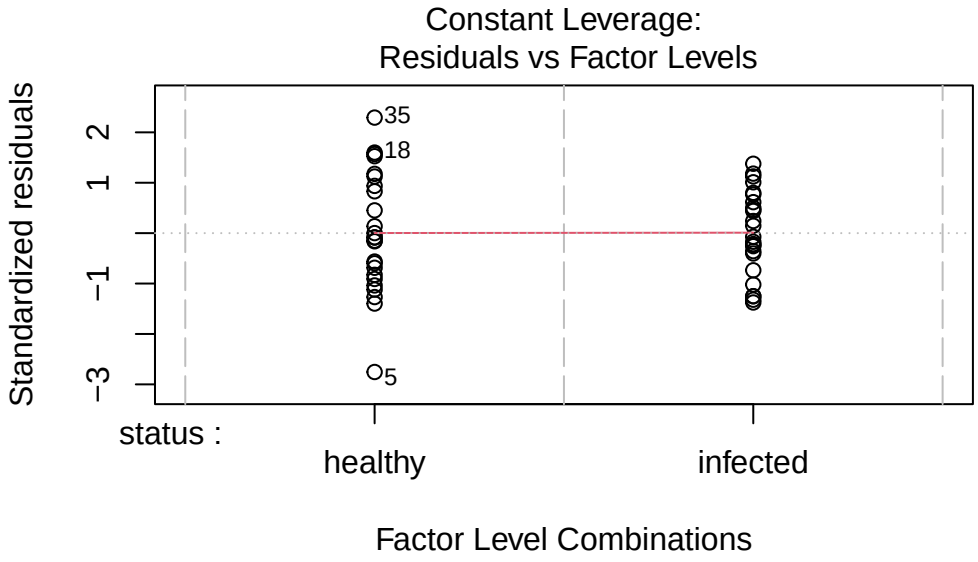
We are not focusing yet on assumptions. But usually, we can check most of them by running:

```
plot(model2)
```




Q−Q Residuals
Standardized residuals
−2
0
1
2
35
18
5
−2
−1
0
1
2
Theoretical Quantiles
lm(growth ~ status)


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
5
35
18
340
350
360
370
380
390
Fitted values
lm(growth ~ status)



In this case they look good.

Finally, to run a 2-sample t-test we need to know whether the variance is equal. Do the following to test it:

```
yH <- tunadata$growth[tunadata$status== 'healthy']
yI <- tunadata$growth[tunadata$status== 'infected']
```

```
var.test(yH,yI)
```

```
    F test to compare two variances

data:  yH and yI
F = 1.9821, num df = 24, denom df = 24, p-value = 0.1004
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.8734287 4.4978255
sample estimates:
ratio of variances
          1.982052
```

Are the variances equal? or not? Remember, **usually** a p value lower than 0.05 means difference.

Also, read the alternative hypotehsis: `true ratio of variances is not equal to 1` check the 95% confidence interval, does it include 1?

Finally, let's run the t-test. This code chunk is incomplete. You have to specify whether the variance is equal or not. Use the results from the variance test to make your decision.

```
t.test(growth ~ status, data = tunadata,
       var.equal = , paired = FALSE,
       alternative = )
```

Question 6

1. Download the teporingos2pops.csv file, run a linear model on it and plot a boxplot with points of different color. For fun, try to also have the boxplot be a different color. IT doesn't have to look good, just explore and play with ggplot.
2. Describe what a boxplot is. We will talk more about it during our "plotting and visualizing" classes, but if you don't know what it is, this is a good time to research it and try to describe it

4.1 Supplementary information. For loops and a "challenge"

This is not a part of the assignment, but it is a useful guide. There is also a cool challengfe at the end if you want to attempt it.

For loops aren't the only kind of loops. If you remember, the for loops looked like this (see this section on html or a fully rendered document):

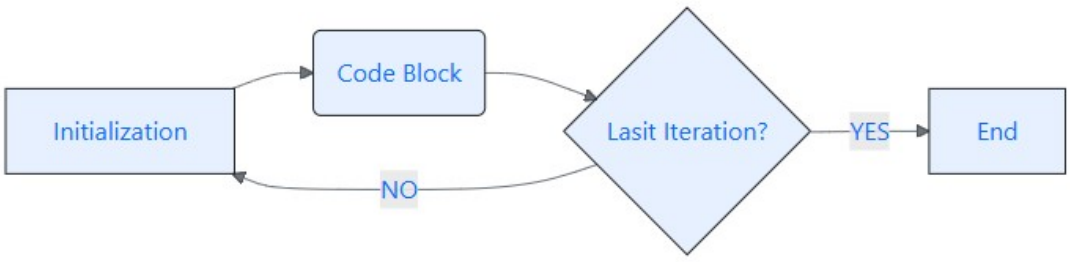
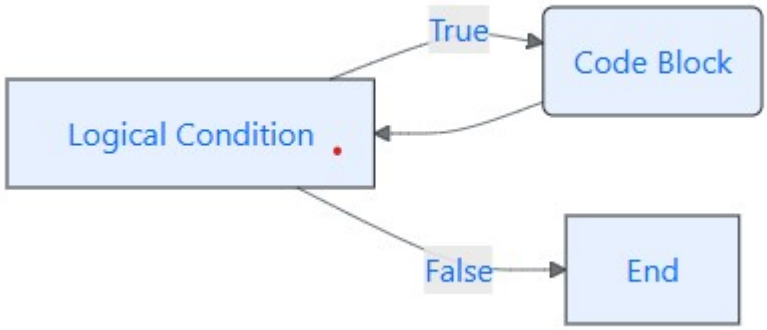


Figure 4.1: for loop

There are other types of loops



For example, a while-loop is:

And a repeat loop is:

Some of the important functions to know.

- `for` iterates over a sequence
- `while` executes a set of statement as long as a logical condition is true
- `break` stops the loop before it loops through all items
- `next` skip to next iteration before terminating the code chunk
- `if...else` might also be good to know to set logical conditions

4.1.0.1 Challenge

You are going to flip a **fair** coin 10,000,000 times. However, if you get lucky and get 20 **straight** tails, you get to stop flipping a coin (and you get 10 extra credit points!) You need the record in which throw you got the 20th straight tail, and print it. If you are able to do all of this, you get the credit!

Supplementary Readings

Please Read Chapter 6 of (Ward and Nolte 2024).

Part IV

Introduction to Models

What is a model?

A model is a simplification (or abstraction) of reality that helps us describe, understand or predict a system. Statistical models, help us describe relationships among variables. Which allows us to describe systems, and more specifically, it allows us to do:

- Inference
- Prediction
- Exploration

Because biological systems have uncertainty, we need models in order to answer hypotheses.

The Lego models

Notre-Dame

Take for example, the following lego model:

Figure 4.2: Notre Dame model.

That Lego model was obtained from https://www.nytimes.com/2024/06/01/world/europe/lego-notre-dame-cathedral.html. It clearly represents reality. But it obviously is NOT the real Notre-Dame. Most importantly, some things are obviously wrong. Some details are missing, the trees look different, the color is off, the aging doesnt show, it is missing windows, and bells, and many other things.

However, if you showed this to someone that has never seen the real Notre-Dame Cathedral, they would have a very good idea of what it looks like.

In biology and other sciences we rarely have access to the real buildings. We only have access to the Lego models. So, we derive our understanding of the natural world, from a series of Lego models that represent different hypothesis. Some are very complex and intricate, some are very simple, some are very hard to understand. Some try to show you the shape of a building, others the usefulness, others, the colors, and others how future real buildings might look like. But most importantly, they are all wrong. One way or another.

Because we do not have access to the real world buildings, and we only have access to the Lego

models, it is important to know that 1- all models are wrong, 2- some of them are useful, and 3- it is important to know why they are wrong. This is the basis for George Box's quote:

George Box and Norman Drapper said

"Remember that **all models are wrong**: the practical questions is how wrong do they have to be to not be useful" (Box and Draper 1987)
Remember, all of your models (ANOVAS, Linear Models, etc) will be wrong! But they can still be useful!

Think about it

You have an activity, in which a group of people are trying to recreate a house (think, a typical suburban American house).
A group of 20 people are given a bag with 30 simple pieces of Lego and are asked to build a typical suburban American house. They have to use all pieces.
A group of 20 people are given a bag with 20,000 pieces of Lego. Some are very unique pieces. They are also asked to build a typical suburban American house.
Think about the following:
1) Which of the two groups will have higher bias (difference between reality and model)?
2) Which will have higher variance (differences among models)?

5 Linear Models

So far we have run a categorical linear model with no covariates (also known as a one-sample t-test):

$$y_i \sim \beta_0 + \epsilon_i$$
$$\text{where } \epsilon \sim normal(0, \sigma)$$

That is the first model you ran in the last assignment (assignment 2).

That equation reads as the ith observation of the response variable (y) is given by the intercept β_0 plus some "observation" error that is normally distributed.

You also ran a couple of 2 sample test (AKA 2 sample t-test) which have the following structure:

$$y_i \sim \beta_0 + \beta_1 x_i + \epsilon_i$$
$$\text{where } \epsilon \sim normal(0, \sigma)$$

That reads as the ith observation of the response variable (y) is given by the intercept β_0 plus a coefficient (β_1) multiplied by an explanatory variable, plus some "observation" error that is normally distributed.

Let's actually look at the tuna example that you should have run.

$$Growth_i \sim \beta_0 + \beta_1(\text{status infected}) + \epsilon_i$$
$$\text{where } \epsilon \sim normal(0, \sigma)$$

Where status (infected) is treated as:

Status	Value of x
Healthy	0
Infected	1

So, for healthy individuals, we can obtain the predicted value as:

$$y_i \sim \beta_0 + \epsilon_i$$
$$\text{where } \epsilon \sim normal(0, \sigma)$$

While for the infected individuals, the predicted value is obtained using the following:

$$y_i \sim \beta_0 + \beta_1 + \epsilon_i$$
$$\text{where } \epsilon \sim normal(0, \sigma)$$

So, when we run the model in R (you did this last week!) We get the following:

```
Call:
lm(formula = growth ~ status, data = tunadata)

Residuals:
    Min      1Q  Median      3Q     Max
-317.76  -83.56   -9.86   91.37  264.44

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)      390.16      23.55  16.569   <2e-16 ***
statusinfected   -51.76      33.30  -1.554    0.127
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 117.7 on 48 degrees of freedom
Multiple R-squared:  0.04791,   Adjusted R-squared:  0.02808
F-statistic: 2.416 on 1 and 48 DF,  p-value: 0.1267
```

So, the mean value for healthy individuals (or expected value for a healty individuals is β_0 or 390.16 , while the mean value (or expected value) for an infected individual is $\beta_0 + \beta_1$ or 390.16 - 51.76 = 338.4.

Other good information about this test can be found in the bottom right corner. It's a p-value. This is the **MODEL p-value.** Essentially, the model p-value in R compares two hypotheses:

$$\text{Ho: } y_i \sim \beta_0 + \epsilon_i$$
$$\text{Ha: } y_i \sim \beta_0 + \beta_1 + \epsilon_i$$

Essentially, it always compares a "null model" where only an intercept is estimated with whatever model you ran. Later in class we will learn how to run and compare multiple models, and that is a great way to get away from the limitations of p-values and hypothesis testing!

Only in this case (a simple regression with only one predictor), the **model p-value** and the **p-value** for the coefficient will be the same. Check the results and look at the coefficients. Each coefficient has a P-value. For each coefficient, the hypothesis checked are:

$$\text{Ho: coefficient} = 0$$
$$\text{Ha: coefficient} \neq 0$$

Essentially, if there is an effect of the coefficient, then the result will be significant, **and therefore** this model will be better than a null model **(no effect model)**.

As we move to more complex models, we will get multiple coefficient p-values, and an overall model p-value.

Stop!

At this time, you should have understood all of the concepts I explained. If you didn't, now is the time to raise your hand, shoot me an email, read info online, or meet me

Finally, there is something I want you to think about:

⚠ Question 1

According to the results of the model, there is not a significant effect of status on tuna growth. I am going to give you a hint, there is a **huge** effect of infection on tuna growth. However, this test wasn't able to identify it. Why? hint: think both biologically and statistically. Write an answer

5.0.1 Assignment 3

In this assignment we will look at a model with **more** than two groups in a categorical variable, ANOVA, at a model with one continuous predictor, and at models with both categorical and continuous predictors (ANCOVA). Some of these topics we haven't seen in class yet.

You have two weeks to turn this assignment in, as there will be no assignment next week! :)

We will also look at the always scary subject of model assumptions.

i What test should I use when...?

Based on your experience in past courses, you may be used to statistics being treated almost as a cookbook, or manual, in which depending on the situation you choose a test (t-test, ANOVA, chi-square, etc). That is not how most graduate-level courses teach statistics, for multiple reasons.
The most important is: the idea of this course is for you to become a modeler that can design models to answer any question without being strapped to some very-specific tests. I also strongly believe that **philosophically** statistics should be taught this way. I believe it equips students with an understanding of what makes a model good, what makes it useful, what makes it significant, and what decisions we can make knowing all of that information.

5.0.1.1 Assumptions

There are four assumptions of a linear model:

1. Linearity
2. Normality
3. Homoscedasticity
4. Independence

Let's test each one with the Tuna data from the last assignment. Read the tuna csv and run a model in which the response variable is growth and the explanatory variable is status. Name it model 2 (just like you did last time, it might even be pre-loaded if you haven't closed RStudio)

5.0.2 1. Linearity

This essentially means that a regression model assumes that the average outcome is linearly related to each term in the model when holding all others fixed. Those are a lot of words. Essentially, when you plot x against y, it should look like a line.

How do we test linearity?

Well, if a model has only categorical variables, then it is automatically linear. So, no need to check this assumption in this case.

Raise your hand if you don't understand why. We will actually discuss this in the next two weeks.

> **Question 2**
>
> Why do you think that all models with *only* categorical variables are always linear?

While you do not need to test linearity if there are only categorical variables, we will do it regardless.

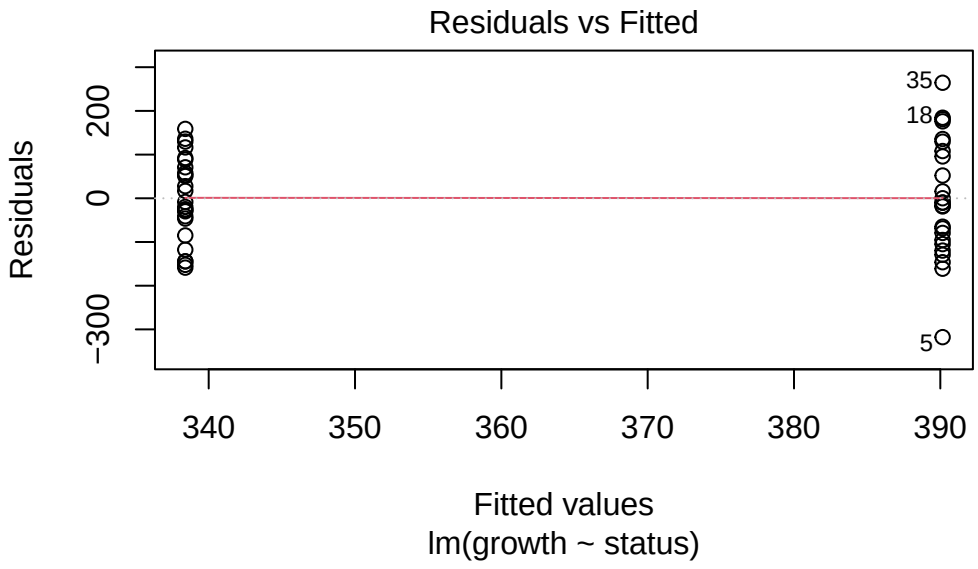
We can test linearity by plotting residuals against the fitted values. To get the residuals, we use:

```
#to get residuals
res<-model2$residuals
fv<-model2$fitted.values
```

And then, we could plot them.

However, r has a very nice function, that automatically plots this interaction:

```
plot(model2, which=1)
```



This function plots the residuals and the fitted values. If the red line is around the dotted line (at y=0) then, the model is linear. We will talk about what to do when they are not linear on future classes. In this case it is as linear as can be (again, categorical variables are always completely linear).

5.0.2.0.1 What to do when it is not linear?

We will talk about this in future classes. But here is what you can do: usually we can transform the data (log-transform or square root), or add a polynomial term to it in order to fix non-linearity. Oftentimes we should be running a **generalized linear model (glm)** rather than a linear model. We will learn more about that in a couple weeks!

5.0.3 2. Normality

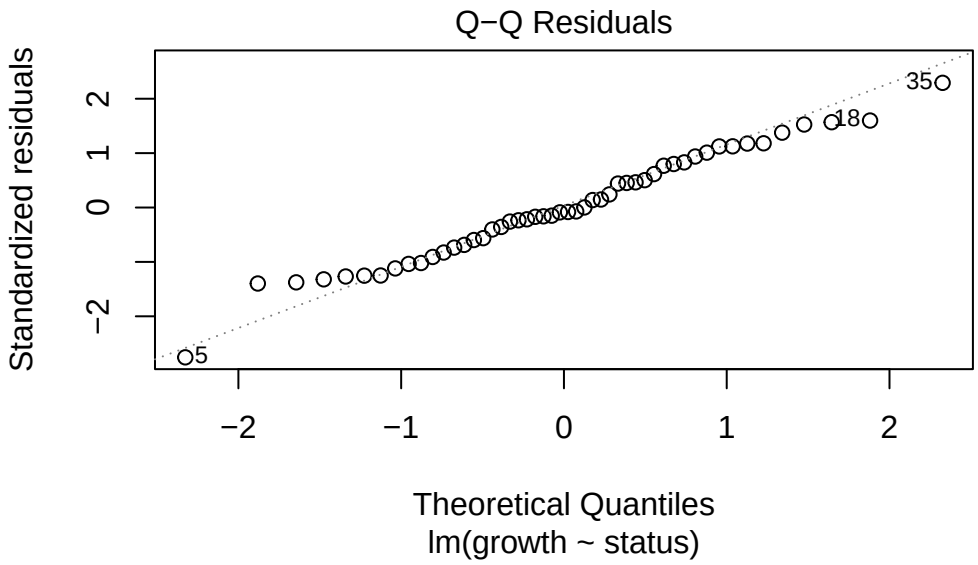
This is a bit of an interesting one. Before we start, remember, it is **the residuals** from the model that should be analyzed, not the distribution of the data per-se (which usually is non-normal).

Even when the residuals are not normal, you can probably get away with it as long as it's not too bad. If the residuals are completely non normal, you may have to run a **generalized linear model (glm)** rather than a linear model. You can also transform your data or run a box-cox transformation.

How do you test for it? You use a qq-plot. It essentially compares your observed standardized residuals, to the standardized residuals that would be expected under normality (theoretical quantiles).

You can test normality using this plot:

```
plot(model2, which=2)
```



However, I prefer to use one function in the `car` package:

Download the package (use `install.packages()` outside of your qmd file. We will be using this package a lot.

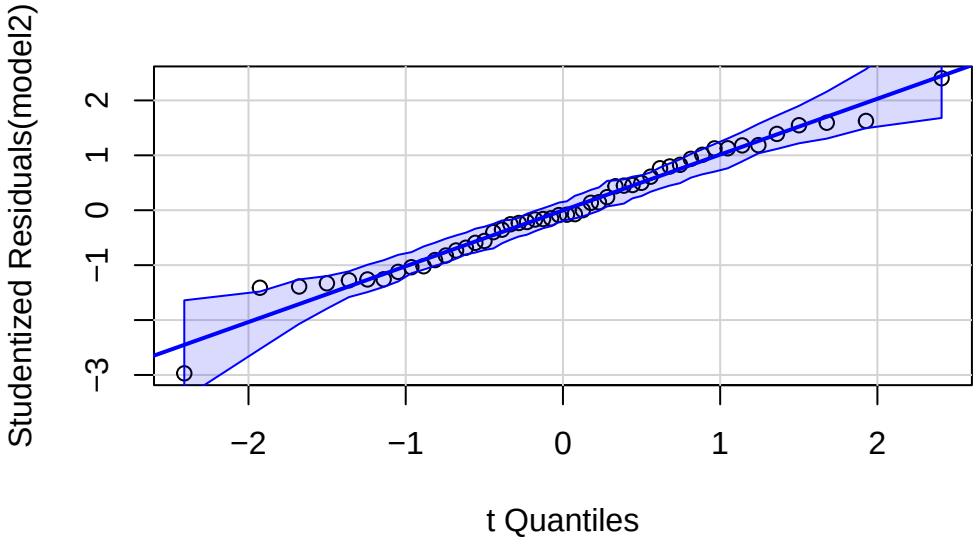
After you have downloaded the package, run:

```
library(car)
```

```
Loading required package: carData
```

And you can run this plot:

```
qqPlot(model2, id=F)
```



This adds confidence itnervals to the plot. As long as the majority of the points fall within the CI's you are fine. Even if some do fall out, it's usually OK. Normality is a very forgiving assumption.

I **DO NOT** recommend to test for normality using a Shapiro-Wilk test. My reasons are explained incredibly by Allen Downey here: https://www.allendowney.com/blog/2023/01/28/never-test-for-normality/ Also, Allen Downey is incredibly insightful, and has some very interesting posts, you should read some of his other work,.

Anyway, in his post, he explains the following:

`...the question you really care about, which is whether the Gaussian model is a good enough model of the data for your purposes... That's a modeling decision, and no statistical test can help. In the original article, I suggested some methods that might.`

I suggest a qq-plot as an easy solution. He provides some other potential solutions (he doesn't love qq-plots). However, all of his methods check whether a particular distribution is a good model for a dataset. And that's not a statistical test, it is a modeling decision. **Meaning:** There is no test that will give you a check-mark. They are all visual tests, and it is up to you to decide if it's good enough for your model. In my experience, students **hate** this answer. We tend to want specific answers ("just tell me if it's normal and what test I should run"). Unfortunately, life (and data) is way more complicated that that. Fortunately, that makes it fun! . Also, as a result, modeling and statistics are life-long ventures in which you never stop learning and updating your priors, posteriors, beliefs, and philosophies! Ok… if you are still

reading this, thank you! I went on a long tangent, and it's time to get back to the assumptions. Again, if you don't like the distribution you can run a different model (maybe a glm), or do a box-cox transformation.

Anyway, in our example the qqplots look great, so we can move on!

5.0.4 3. Homoscedasticity

Remember, our linear model is:

$$y_i \sim \beta_0 + \beta_1 x_i + \epsilon_i$$
$$\text{where } \epsilon \sim normal(0, \sigma)$$

And homoscedasticity essentially means that the σ stays the same independently of the value of x (equal variance).

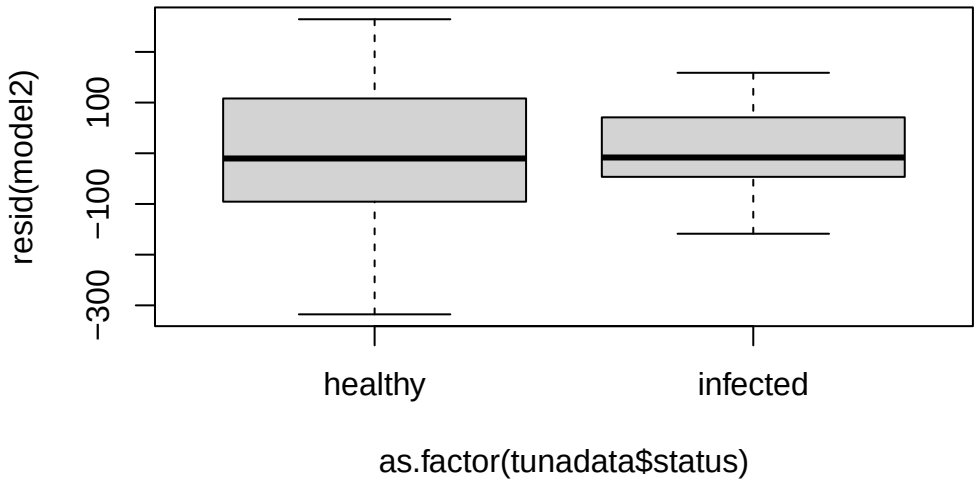
If we don't meet this assumption we might be working with count data or binomial data (and thus, should do a **glm)** or we can do a log-transformation if we are working with continuous data.

To test for homscedasticity, I recommend plotting the residuals against the predictors.

If your predictor variable is continuous do a scatterplot.

If it is discrete you use boxplot:

```
plot(resid(model2)~as.factor(tunadata$status))
```



mmm… something is up here. While I believe that you could justify this, and say that the assumption is met, I think this telling me something about the data. The infected had a lot less variability. Why might that be? Could this information help you answer question 1?

If we did a test of equal variance:

```
var.test(growth~status,data=tunadata)
```

```
    F test to compare two variances

data:  growth by status
F = 1.9821, num df = 24, denom df = 24, p-value = 0.1004
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.8734287 4.4978255
sample estimates:
ratio of variances
          1.982052
```

We would conclude that the assumption is met . However, I think that in this particular case we would not completely trust that the variance is equal. We will discuss this in class.

5.0.4.1 4. Independence

We will discuss pseudo-replication and temporal and spatial autocorrelation later on. But this is not a statistical question. This is an experimental design question. It depends on how the data were collected, and that might be information more suited to an experimental design class. In this case, the data is independent, but there's no way to tell from the dataset itself.

OK. At this point I expect you to be even more confused than before. That's OK and normal! Part of the process.

5.1 Model with a categorical variable with more than 2 groups

Let's actually work on the data and run models.

You are welcome to check for assumptions for every model, but if your head is spinning after reading the assumptions section, then you can skip it and not run assumptions.

Teporingos part two: Remember the last teporingos dataset you worked with last week? Well, turns out that there were a total of four sites (or populations) sampled! However, they were saved in different files!

So, you have to load two datasets: `teporingos2pops.csv` and `teporingosnew.csv` and somehow fuse them. We will do it using basic `r`. In a couple weeks you will be introduced to a different way to manage data: the `tidyverse`.

Read both datasets:

```
pops1<-read.csv("data/teporingos2pops.csv")
pops2<-read.csv("data/teporingosnew.csv")
```

And let's combine them:

```
allpops<-rbind(pops1,pops2)
head(allpops)
```

```
     site     mass
1 texcoco 400.3696
2 texcoco 407.5110
3 texcoco 413.9215
4 texcoco 412.7282
5 texcoco 424.2921
6 texcoco 418.2353
```

This code works because both files had the same structure, so, we just bind the rows.

Now, let's run the model:

```
modelt<-lm(mass~site,data=allpops)
summary(modelt)
```

```
Call:
lm(formula = mass ~ site, data = allpops)

Residuals:
    Min      1Q  Median      3Q     Max
-42.180 -10.463  -1.370   7.119  49.499

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   448.857      2.811 159.702  < 2e-16 ***
sitetexcoco   -23.023      4.216  -5.461 3.85e-07 ***
siteTlaloc      2.093      4.216   0.496 0.620713
sitetopilejo  -15.184      4.444  -3.417 0.000938 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 15.39 on 94 degrees of freedom
Multiple R-squared:  0.3267,    Adjusted R-squared:  0.3052
F-statistic:  15.2 on 3 and 94 DF,  p-value: 3.837e-08
```

Wow! We ran this model **the same** way we did the one with 2 groups. That is super useful!

You get more coefficients this time. Four of them. Let's explore the model equation (next week I will ask you to provide the model equations):

$$y_i \sim \beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i} + \beta_3 x_{3,i} + \epsilon_i$$
$$\text{where } \epsilon \sim normal(0, \sigma)$$

The betas (4 values) are the coefficient estimates. And the x values (there are 3 different values) are:

Site	Value of x_1	Value of x_2	Value of x_3
Popocatepetl	0	0	0
Texcoco	1	0	0
Tlatelolco	0	1	0
Topilejo	0	0	1

Question 3

Interpret the results from the last model, including the p-values of the coefficient and of the model. Calculate the expected mass of an individual from Popocatepetl and one from Tlatelolco by "hand" (you can use r, but input the specific numbers and write the equation, ask me if you don't understand the question)

After we run the model, we need to know whether there is an effect of site on weight. We can use an ANOVA for that. There are many ways to run ANOVAS in R. This is my go to (from package `car`. First, let's explore what out hypotheses are:

$$\text{Ho: } \mu_1 = \mu_2 = \mu_3 = \mu_4$$
$$\text{Ha: At least one } \mu \text{ different}$$

So, if p<0.05, we can reject the null hypothesis. Let's run it:

```
Anova(modelt)
```

```
Anova Table (Type II tests)

Response: mass
          Sum Sq Df F value    Pr(>F)
site       10809  3  15.203 3.837e-08 ***
Residuals  22276 94
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

That is cool. It actually tests if the four sites are the same!

Question 4

Interpret the p-value obtained and make a "statistical decision"

So, we now know that at least one is different. But which one? In this course we will be doing a lot of pairwise comparisons. And we will be using the package `emmeans`. This is a fantastic package, because it lets you do pairwise comparisons in very complex models. For this model, we use a simple Tukey test. However, we essentially use the same code to use emmeans independently of model complexity:

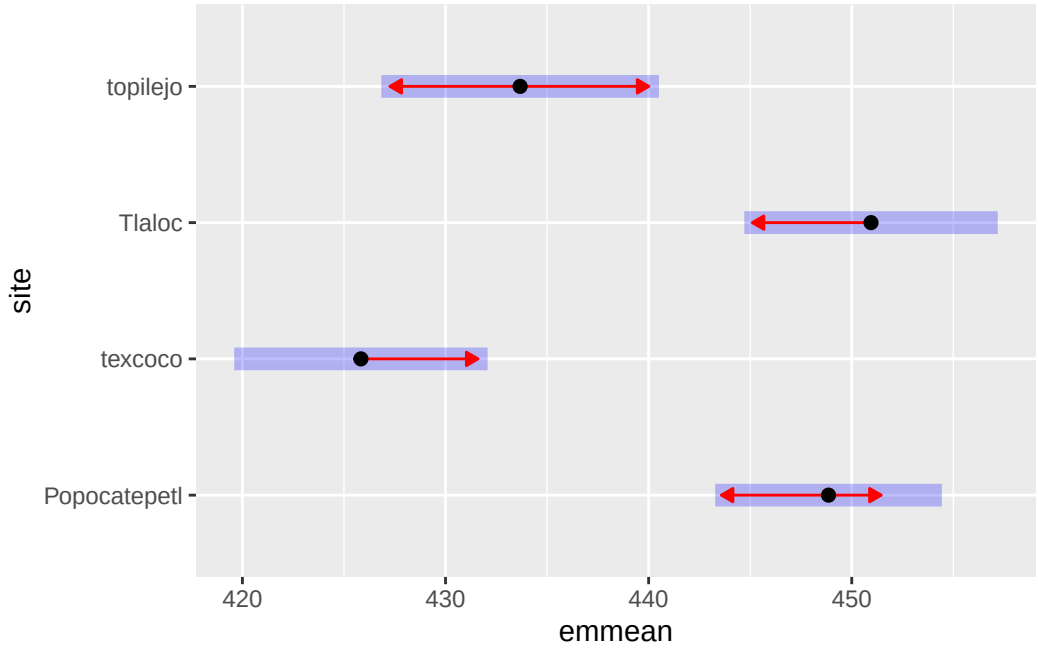
```
library(emmeans)
```

```
Warning: package 'emmeans' was built under R version 4.3.2
```

```
emm.s<-emmeans(modelt, "site")
pairs(emm.s)
```

```
 contrast                estimate   SE df t.ratio p.value
 Popocatepetl - texcoco     23.02 4.22 94   5.461  <.0001
 Popocatepetl - Tlaloc      -2.09 4.22 94  -0.496  0.9597
 Popocatepetl - topilejo    15.18 4.44 94   3.417  0.0051
 texcoco - Tlaloc          -25.12 4.44 94  -5.652  <.0001
 texcoco - topilejo         -7.84 4.66 94  -1.682  0.3389
 Tlaloc - topilejo          17.28 4.66 94   3.707  0.0020

P value adjustment: tukey method for comparing a family of 4 estimates
```

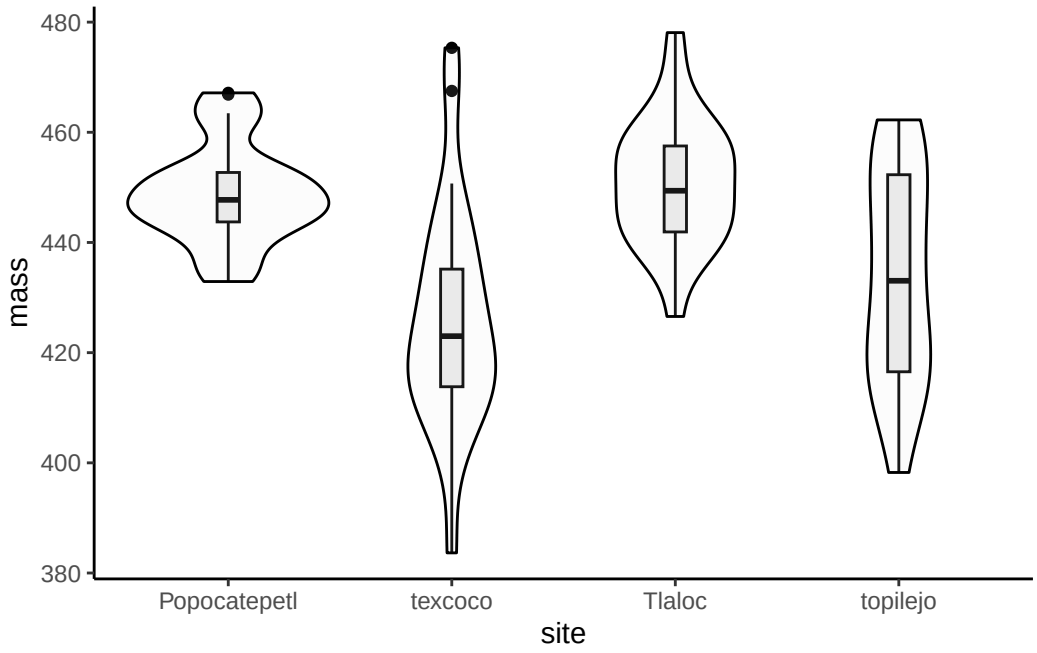
```
plot(emm.s, comparisons = T)
```




```
library(ggplot2)
```

```
Warning: package 'ggplot2' was built under R version 4.3.3
```

```
ggplot(data = allpops, aes(x = site, y = mass)) +
  geom_boxplot(fill=gray(0.7,0.25),color="black",width=0.1) +
  geom_violin(fill=gray(0.9,0.1),color="black")+
 theme_bw() + theme(panel.border = element_blank(), panel.grid.major = element_blank(),
                    panel.grid.minor = element_blank(), axis.line = element_line(colour = "
```




```
emmeans(modelt,"site")
```

```
 site         emmean   SE df lower.CL upper.CL
 Popocatepetl    449 2.81 94      443      454
 texcoco         426 3.14 94      420      432
 Tlaloc          451 3.14 94      445      457
 topilejo        434 3.44 94      427      441

Confidence level used: 0.95
```

This test compares all of the sites.

> ⚠ Question 5
>
> Which sites are different?

5.2 Model with a continuos variable

We will now run a model with a single continuous variable. Essentially testing whether the explanatory variable has an effect on the response variable.

First, let's run read the dataset:

```r
foodav<-read.csv("data/foodav.csv")
head(foodav)
```

```
  site Foodavailability ReproductiveEffort
1    1         3.274107           4.664454
2    2         4.674298           6.572521
3    3         5.907927           7.804491
4    4         3.109825           4.624609
5    5         2.120704           3.660976
6    6         0.984446           2.189319
```

This dataset cotains data on 63 sampled sites of *Poeciliopsis baenschi* . This is a small viviparous fish, and you are studying whether the food availability at each site has an effect on the mean reproductive effort (measured as wet weight in grams) of this species.

The way we run this model is the same as before:

```r
modelfish<-lm(ReproductiveEffort~Foodavailability,data = foodav)
summary(modelfish)
```

```
Call:
lm(formula = ReproductiveEffort ~ Foodavailability, data = foodav)

Residuals:
     Min       1Q   Median       3Q      Max
-0.37653 -0.25549 -0.04903  0.23662  0.45750

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)       0.89490    0.07262   12.32   <2e-16 ***
Foodavailability  1.19387    0.01590   75.08   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2652 on 61 degrees of freedom
Multiple R-squared:  0.9893,    Adjusted R-squared:  0.9891
F-statistic:  5636 on 1 and 61 DF,  p-value: < 2.2e-16
```

And the output looks really similar!

> ⚠ Question 6
>
> Look at the output and answer:
> Is there an effect of food availability on reproductive effort? What are your statistical hypotheses and what is your conclusion?

Because our variable is continuous, x takes different values. Let's remember our equation for this model:

$$y_i \sim \beta_0 + \beta_1 x_i + \epsilon_i$$
$$\text{where } \epsilon \sim normal(0, \sigma)$$

Or, in this specific case:

$$Reproductive\ effort_i \sim \beta_0 + \beta_1 Food\ availability_i + \epsilon_i$$

So, for a site with a food availability index of one, our estimated reproductive effort would be:

```
0.89490 + 1.19387*1
```

```
[1] 2.08877
```

> ⚠ Question 7
>
> What is the food availability of a site with a food index of 3.87?

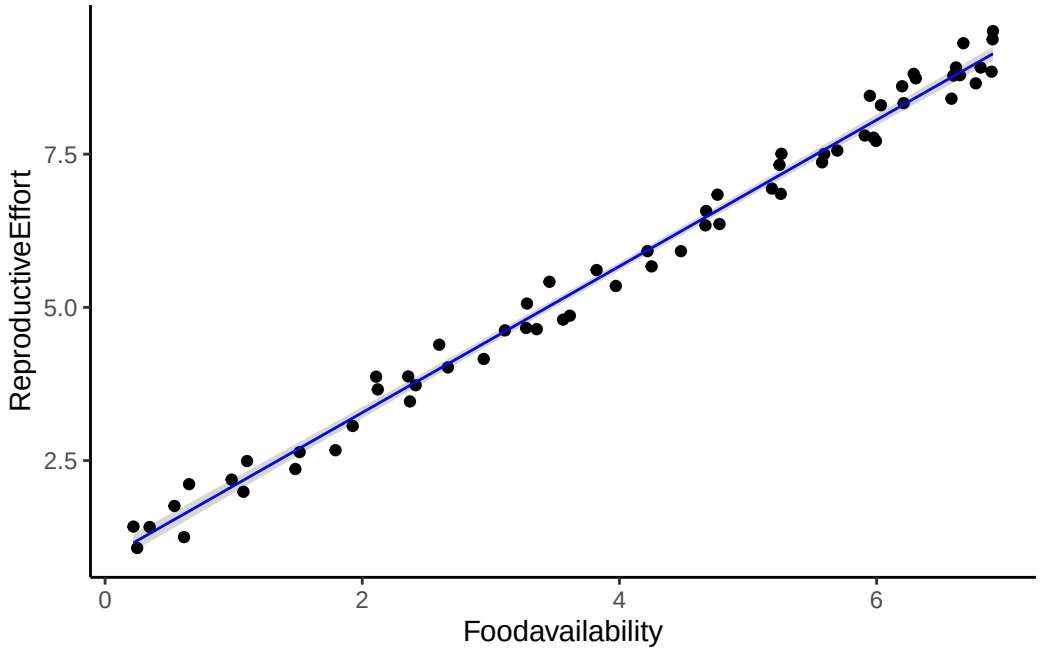
You can estimate the food availability and a confidence interval (crit.value * std. error) for an infinity of values. However, the `predict.lm` function does it for you:

```
predictedv<-predict.lm(modelfish,foodav,interval="co")
foodav<-cbind(foodav,predictedv)
```

And that can be super useful for plotting!:

```
library(ggplot2)
```

```
ggplot(foodav, aes(x = Foodavailability, y = ReproductiveEffort,ymin=lwr,ymax=upr)) +
   geom_point() +
   geom_line(aes(y=fit),color="blue") +
   geom_ribbon(alpha=0.2)+
   theme_classic()
```



Pretty cool! And kind of easy, right?

5.3 Model with categorical and continuous variables

In this case, you are working for a food company that is developing a drug that lowers sugar consumption in rats.

You are testing 3 doses: control, mid, and high. You do this in rats that are being fed ad-libitum. You also record the daily consumption of food "pre-trial", and finally the daily consumption of food "post trial".

Read the drugZ csv:

```
drugrat<-read.csv("data/drug_rat.csv")
```

Where FC is final consumption, IC is initial consumption.

and run the following model:

```
modeld<-lm(FC~IC+Dose,data=drugrat)
summary(modeld)
```

```
Call:
lm(formula = FC ~ IC + Dose, data = drugrat)

Residuals:
    Min      1Q  Median      3Q     Max
-2.2610 -0.6360  0.0000  0.6514  2.2876

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.51804    0.38892   1.332   0.1849
IC           0.93384    0.05647  16.537   <2e-16 ***
Dosedose1   -0.44007    0.20011  -2.199   0.0294 *
Dosedose2   -2.09915    0.20162 -10.412   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9893 on 146 degrees of freedom
Multiple R-squared:  0.7636,    Adjusted R-squared:  0.7588
F-statistic: 157.2 on 3 and 146 DF,  p-value: < 2.2e-16
```

⚠ Questions 8-12

9. Write the model equation (using betas). You can ask me how to write equations in Quarto!
10. Interpret the model output
11. Run an Anova (and a Tukey test using emmeans if necessary) to test whether there is an effect of dose on final consumption. Make sure to state the hypotheses!
12. Plot the points, and predicted values (it is OK if you struggle with this, we will work on this together on Friday)

6 Model Selection and multi-model inference

6.1 Complexity of Linear Models

So far we have only explored two types of linear models (we are currently starting to talk about Generalized Linear Models:

- Simple Linear Models
- Multiple Linear models

As we progress in the class we will look at more complex (both theoretically, and coding-wise) models. We will look into **Generalized** Linear Models (**GLM**), mixed-effects models, generalized additive models, and multivariate models. Time (and interest) dependent we might go into neural networks, machine learning and AI.

In order to understand those models, you need to understand linear models.

Also, so far we have been running a single model for each dataset. In today's activity you will be tasked with running multiple models for a single dataset. And then comparing models using Maximum Likelihood. These will allow us to explore multiple models.

Before continuing: while we have been running multiple models, we haven't had time to stop and think about how `r` is computing the models. How does `r` finds a value for each coefficient β? It has been using least squares to find these values. Essentially, it finds values of β that minimizes the sum of squares of the differences between predicted and observed values. Think of it as finding the line that minimizes residuals! While these are very useful methods, they are not the only ones!

We will start using some methods that use Likelihood. During Wednesday class we talked a bit about likelihood and probability. At this point you should be able to understand the difference between the two (even if you can't describe it). If you find yourself using (or interested in using) Likelihood and Maximum Likelihood Methods, do yourself a favor and download the book by Burnham and Anderson: Model Selection and Multimodel Inference. Which is free to download through the University of Tennessee Libraries! (Burnham and Anderson 2010) It is the most useful book for multi-model inference!

We will discuss the following paper next week: (Tredennick et al. 2021) available here!

While I am a huge proponent and user of AIC as an inferential method, the authors clearly disagree with me, and they make some very good points!

> **Note**
>
> We will talk more about Likelihood next week. Last Wednesday we discussed the differences between Likelihood and Probability, hopefully that is still partially clear in your mind

---

Likelihood description. Hopefully this will help with understanding likelihood.

Likelihood theory is a paradigm underlying both frequentist and Bayesian statistics, and you need to understand it if you ever want to go deeper into data science and data analyses. This is similar to the example I talked about in class.

The theory underlying likelihood deals with a probabilistic model given the parameters (θ). Remember, the parameters are "the real" values that we are trying to estimate! And in this case I will describe our observations as y. In probability we usually do the following: $P(y|\theta)$. Essentially, what is the probability of of observing the outcomes represented by y, given θ.

In a coin-toss example, θ is the probability of a success (in this case, a success is considered a tail). And we can calculate the probability of observing three straight tails (think of this as $y_1 = 1, y_2 = 1, y_3 = 1$). In this case we know θ. It is 0.5. So, the probability of observing three straight tails is 0.125. We can also estimate the probability of observing 1,10 or 1,000,000 straight tails. Or 2 tails and three heads, or any combination of results. Because we know θ.

However, in the real world, we do not know θ. That is what we are trying to estimate. Think of θ not as a coin-toss now, but as ANY of the following examples: 1) probability that a cow will be pregnant given that it was given a specific drug, 2) probability of survival for a turkey during a season, 3) probability that a food product will spoil after 48 hours at room temperature following a new packing procedure, 4) probability of a fish passing through a fish ladder it approached, 5) probability of a manufacturing defect in a new wood processing method.

In these cases, we do not know the probability! That is what we are trying to estimate, however, we do know the outcomes! We sample our population and know the results. Going back to the coin-toss example, assume we don't know the probability of getting tails in a coin-toss. We don't know if the coin is fair or not. However, you do an experiment and toss the coin 10 times and record the results. You replicate this experiment 25 times. Now you have 25 observations (y_i), where each y is the number of successes observed in each experiment i. However, we can estimate the probability of observing what we did if θ was 0.01, and the probability of observing wat we did if it was 0.02, and if it was 0.03, or 0.04, and so on. This allow us to find a parameter θ that would maximize the following function:

$$\mathcal{L}(\theta|y) = P(y|\theta)$$

Essentially, maximum likelihood is a method that answers the following: "what value of theta would maximize the probability of observing what you actually observed?" By definition, the Likelihood function is conditional on the observed data, and is a function of the unknown parameter θ.

6.2 Testing different models

The beauty of model selection and multi-model inference is that we are not constrained by the limits of hypothesis testing. More often that not, we already know the answer to a hypothesis test. We already know that there are differences among populations, or that there is an effect of x on y. Using multi-model inference or other model selection methods we can test different hypotheses.

To do this, we will use different methods. First, we will simply compare the r^2 values of each model. We will also use an information criterion, and finally we will use model cross-validation.

Download the `grasslanddata.csv` file. And explore it.

This dataset has 6 variables:

Response variable:

KgDMHA: Pasture mass. Measured in kilograms of fry matter per hectare

Abiotic explanatory variables:

Water, salinity, and nitrogen

Biotic variables:

Graze (average number of grazing animals observed in the area)

Pests: Proportion of grassland where pests are found. Divided in low (low proportion), half (about half of it has pests), and most.

You will be running all of the following models (keep reading before you start running models!):

1. Null model
2. Model with only effect of irrigation
3. Model with with a quadratic effect of irrigation

4. Model with all abiotic effects (additive)
5. Model with all abiotic effects (interactive)
6. Model with all biotic effects (additive)
7. Model with all biotic effects (interactive)
8. Model with all effects (additive)
9. Run 1 model not described in this list
10. Model with all effects (interactive)

    1. This is called a "global" model. It is your most complex model
    2. This is the model we use to test the assumptions
    3. No need to plot this model

You will be running all the models except #10. I will help with that one .

```
grasslanddata<-read.csv("data/grasslanddata.csv")
```

The first model we run is the global model, that takes all 5 variables, and all the potential interactions:

$$\begin{aligned} y \sim \beta_0 + \beta_1 x_1 + \\ \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \\ \beta_6 x_1 x_2 + \beta_7 x_1 x_3 + \beta_8 x_2 x_3 + \\ \beta_9 x_1 x_4 + \beta_{10} x_2 x_4 + \beta_{11} x_3 x_4 + \\ \beta_{12} x_1 x_5 + \beta_{13} x_2 x_5 + \beta_{14} x_3 x_5 + \\ \beta_{15} x_4 x_5 + \beta_{16} x_1 x_2 x_3 + \beta_{17} x_1 x_2 x_4 + \\ \beta_{18} x_1 x_3 x_4 + \beta_{19} x_2 x_3 x_4 + \beta_{20} x_1 x_2 x_5 + \\ \beta_{21} x_1 x_3 x_5 + \beta_{22} x_2 x_3 x_5 + \beta_{23} x_1 x_4 x_5 + \\ \beta_{24} x_2 x_4 x_5 + \beta_{25} x_3 x_4 x_5 + \beta_{26} x_1 x_2 x_3 x_4 + \\ \beta_{27} x_1 x_2 x_3 x_5 + \beta_{28} x_1 x_2 x_4 x_5 + \\ \beta_{29} x_1 x_3 x_4 x_5 + \beta_{30} x_2 x_3 x_4 x_5 + \beta_{31} x_1 x_2 x_3 x_4 x_5 \end{aligned}$$

Woah! That is a lot. You only have to write the equations for three models, just be sure to udnerstand where they are coming from! Now, let's run the model:

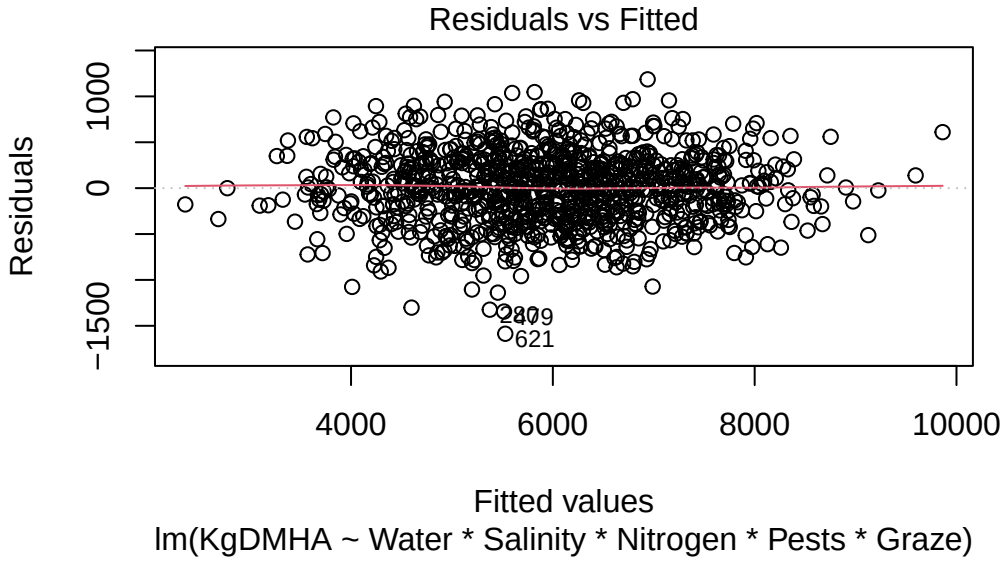
$$\frac{123}{200}$$

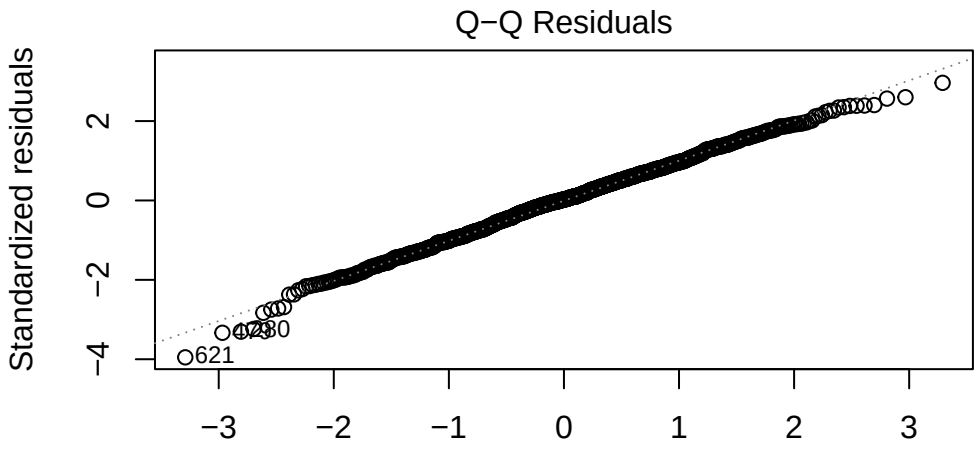
```
model10<-lm(KgDMHA~Water*Salinity*Nitrogen*Pests*Graze,data = grasslanddata)
#summary(model10)
AIC(model10)
```

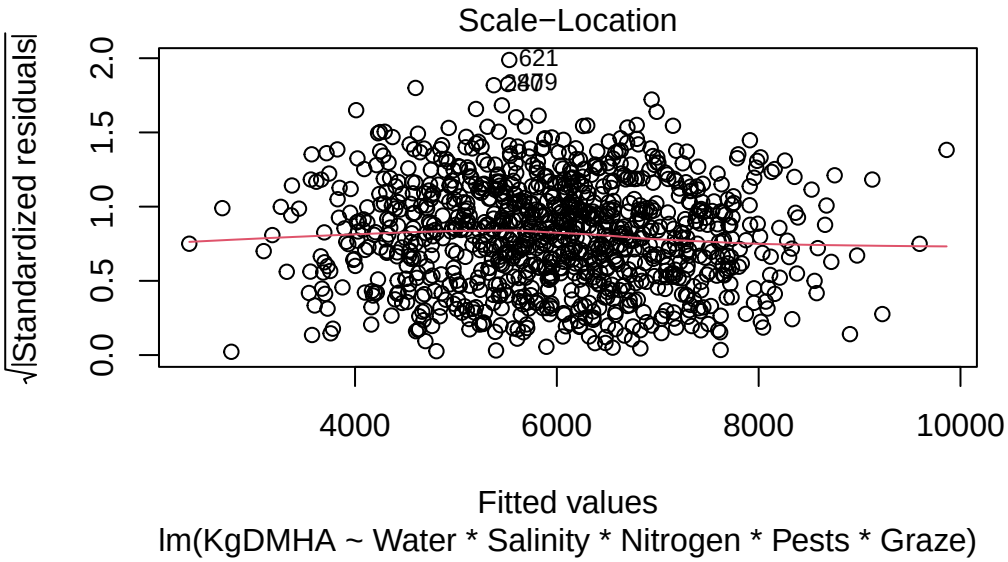
```
[1] 14894.11
```

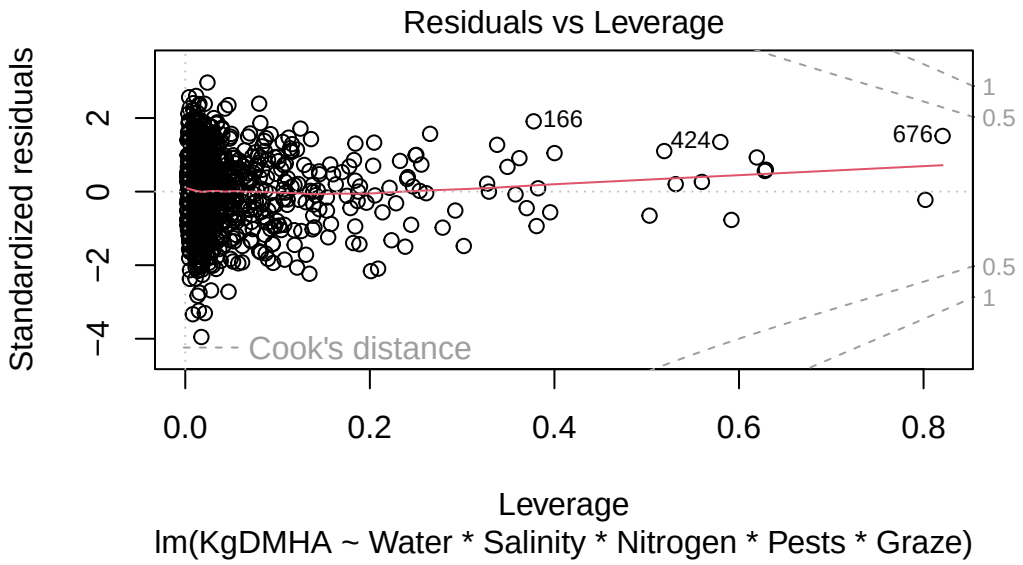
Using `plot(lm)` will plot some exploratory plots that allow you to test assumptions. Essentially, plot 1 and 2 are enough to check the 3 main assumptios.

```
plot(model10)
```




Q−Q Residuals
Standardized residuals
−4
−2
0
2
621
−3
−2
−1
0
1
2
3
Theoretical Quantiles
lm(KgDMHA ~ Water * Salinity * Nitrogen * Pests * Graze)


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
621
4000
6000
8000
10000
Fitted values
lm(KgDMHA ~ Water * Salinity * Nitrogen * Pests * Graze)



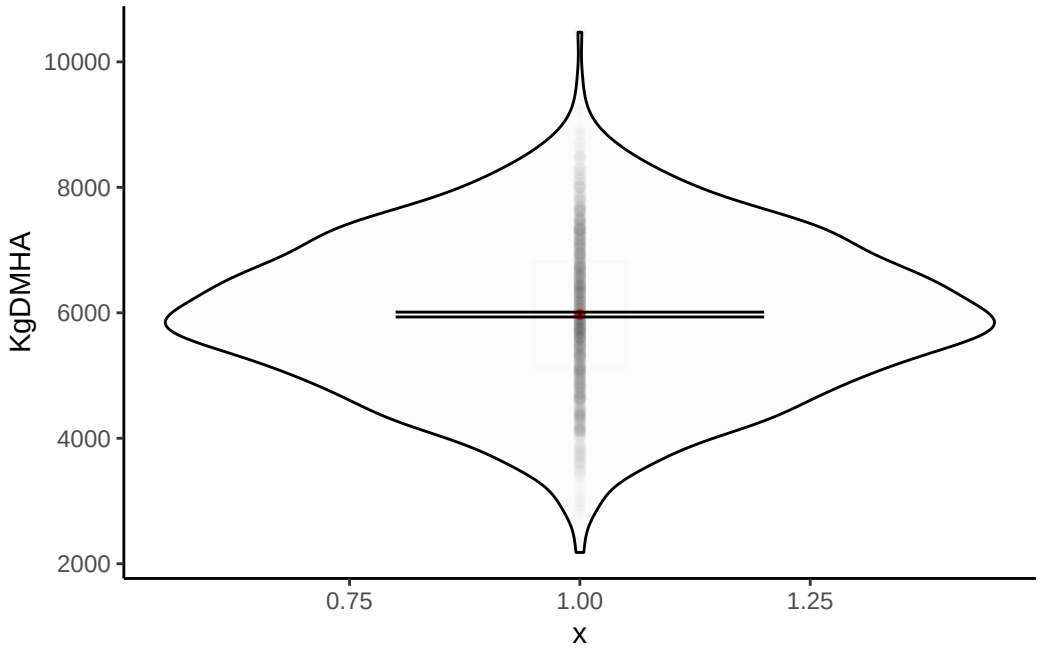

```
#plot(resid(model10)~grasslanddata$KgDMHA)
```

While some of the plots might not look great, in reality this is how most data looks like. In this case values over 8,000 kgs and under 5,000 kgs are rare, which makes the variance seem smaller in those values. However, it is usually pretty obvious when you need to transform your data and we will work on that later. In this case, we are meeting the assumptions, so we can run linear models without any transformations.

Now that we ran it, we can run all of the other models, make sure to follow the outlined steps

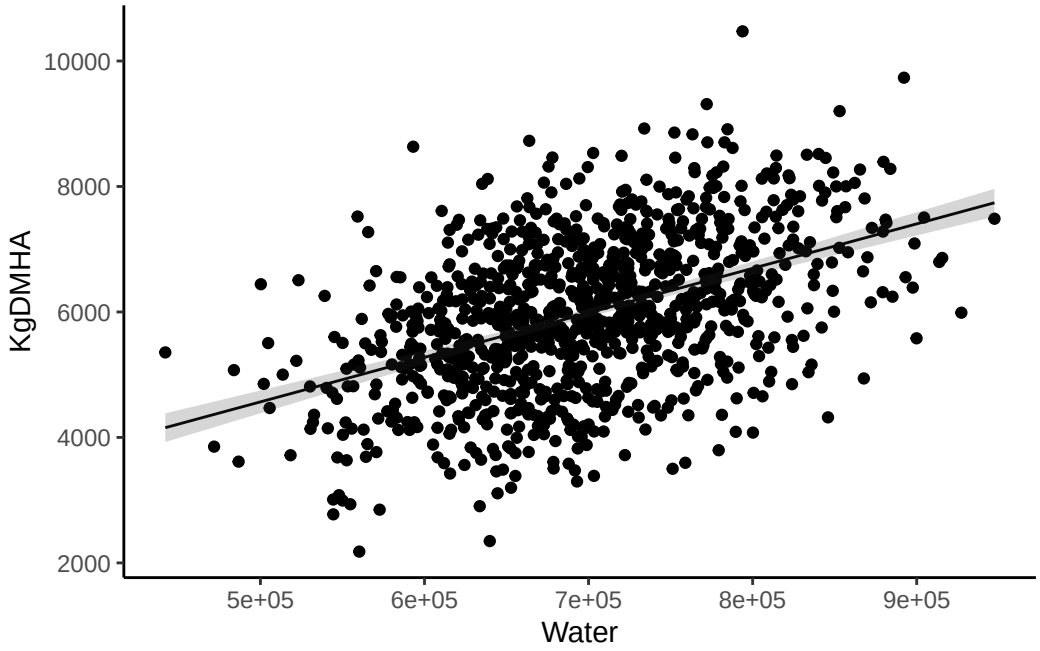
```
Call:
lm(formula = KgDMHA ~ 1, data = grasslanddata)

Residuals:
    Min      1Q  Median      3Q     Max
-3792.2  -854.7   -17.2   847.3  4500.8

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   5972.2       38.9   153.5   <2e-16 ***
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1230 on 999 degrees of freedom

Warning: package 'ggplot2' was built under R version 4.3.3

Warning: The `fun.y` argument of `stat_summary()` is deprecated as of ggplot2 3.3.0.
i Please use the `fun` argument instead.
```




```
Call:
lm(formula = KgDMHA ~ Water, data = grasslanddata)

Residuals:
    Min      1Q  Median      3Q     Max
-3210.3  -705.0   -15.5   782.0  3821.6

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 1.020e+03  3.050e+02   3.344 0.000856 ***
Water       7.094e-03  4.340e-04  16.344  < 2e-16 ***
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1093 on 998 degrees of freedom
Multiple R-squared:  0.2112,    Adjusted R-squared:  0.2104
F-statistic: 267.1 on 1 and 998 DF,  p-value: < 2.2e-16
```



Model 6

```
Call:
lm(formula = KgDMHA ~ Graze + Pests, data = grasslanddata)

Residuals:
    Min      1Q  Median      3Q     Max
-2556.9  -629.3    22.9   608.7  2854.6

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 8508.848    105.262   80.83  < 2e-16 ***
Graze        -57.008      2.446  -23.31  < 2e-16 ***
Pestslow     844.351    112.874    7.48 1.62e-13 ***
Pestsmost   -892.254     62.145  -14.36  < 2e-16 ***
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 909 on 996 degrees of freedom
Multiple R-squared:  0.4557,    Adjusted R-squared:  0.4541
F-statistic:   278 on 3 and 996 DF,  p-value: < 2.2e-16
```

Question 4 10 pts

For the models do the following:

1. Write the equation (only use y, x, and $\beta's$).
    1. *Only do this for models* 1, 2, 3, 4, and 6.
    2. To write equations, write two dollar signs $$$ equation $$ $
    3. Google "Rmarkdown math" for help with how to write equations
    4. You can also right click on any equation on my assignment and select show math as Tex command to see how I wrote the equation.
2. Run the model
3. Print the output
4. Plot the model (only models with one effect or one categorical and one continuous effect)
    1. What do I mean with plot the model? Plot the data (observed), and the predicted values (the line), as well as confidence intervals)
    2. You don't **NEED** to plot the models with three or more effects, but I encourage you to try to think or find a way to plot them

Some important coding tips:

- Additive models have a `+`
- Interactive models have a `*`
- You can mix interactive and additive effects. The order is important though!
- Run quadratic models using: `poly(x,2)` You can run any polynomial term using `poly(x,degree)`. We haven't talked about this!

Be careful with naming your model!

Give each model a different name, and make it a name that makes sense! We will reference the models back!

6.3 Model selection

We will use three different methods to compare models and select our best model.

> Think about it
>
> We will discuss this on Monday and Wednesday: what makes a model "better"? Start thinking about it

6.3.1 R^2

We haven't talked about correlation and coefficient of determination

Correlation: The amount of linear association between two variables. It's measured as r

The **R-squared value, denoted by $\mathbf{R^2}$**, is the square of the correlation. It measures the proportion of variation explained by the model. Essentially, if our predicted model explained and landed perfectly on each observation, then In each of the models we ran, we can obtain a

An R^2 value gives us the proportion of the data variance that was explained by the model.

> Think about it
>
> We will discuss this on Monday: is a higher r^2 always better? When is it not better?

> Assignment question 5:
>
> Check the outputs of the models you ran, and find the one with the highest R^2. Do you think it is the *best* model? You should use the adjusted R^2

6.3.2 AICc

We will talk more about AICc later in class. For the time being what you need to know is that it is an information criterion that uses maximum likelihood. While we can use maximum likelihood to estimate expected values and obtain estimates, we can also use it to compare models. In this case AIC compares models, and the model with the lowest AIC is the best. I attempt an explanation here, but if you don't understand it, hopefully next classes will make it clearer!

AIC has the following equation:

$$AIC = -2log(\mathcal{L}(\hat{\theta}|data)) + 2K.$$

Which may look complicated. But it is an actually pretty simple equation with two important parts:

1) The expression $log(\mathcal{L}(\hat{\theta}|data)$ is the numerical value of the log-likelihood at its maximum point (see Likelihood description or Burnham and Anderson). This maximum point on the log-likelihood function corresponds to the values of the maximum likelihood estimates.

What does this mean? A good way of thinking about it is: "What is the likelihood of observing the data you did, if the model was real?" The Higher the Likelihood the "evidence" of a better model (or better fit of the data to the model). If you notice the equation is -2 times the log-likelihood. Essentially meaning: a lower value is a better model.

2) The expression $2K$ is simply 2 times the number of parameters (K). It is interpreted as a penalty for increasing the number of parameters. It increases parsimony

Essentially this means, AICc chooses the model with the highest Likelihood, while penalizing higher complexity. There are two reasons for this:

1) Every time you add a parameter the Likelihood goes up.

2) While higher complexity tends to decrease bias, it increases variance. The optimum model complexity is found looking at both the Likelihood and the complexity.

> Think about it
>
> Why does adding parameters result in higher Likelihood?

Because AIC has a negative log-likelihood term and a positive K term **LOWER VALUES ARE BETTER. The model with the lowest AIC is "the best model".**

Importantly, the AIC value doesn't tell us anything by itself. It is purely a comparative value, where lower is better. We need to estimate ΔAIC_i which is the difference between the best model and each model i.

ΔAIC_i	Level of Empirical Support of Model i
0-2	Substantial
2-4	Less
4-7	Considerably Less
>10	Essentially none

Ok, let's now use AIC to analyze our models.

We are using a corrected version of AIC that is ideal for small samples (I recommend you always use this one):

$$AIC_c = AIC + \frac{2K(K+1)}{n-K-1}$$

First save all the models in a list. I have some sample code here, but do it with your models (you already run them!)

```
CandidateModels<-list("null"=model1,"model2"=model2...)
```

Now, download and load the `AICcmodavg` package, and run:

```
selectionTable <- aictab(cand.set = CandidateModels)
```

And that's it! What model or models are best? This is super easy to run!

> **Assignment question 6:**
>
> Present your selection table, and describe what model was the best.

You probably have also heard of likelihood ratios. This essentially compared the Likelihood of two models. You can run them pretty easily in R. But we will leave that for some other time. I am terrible at estimating how much time my assignments take students, but I think that this lab is already very time consuming!

If you are curious about how to run them, check this documentation by the `AICcmodavg` package: vignette

6.4 Cross-Validation

Oftentimes a criticism when using linear models to predict values, is that we are using the same dataset to explore the data, to create model, to make inference, and to predict the results.

A way to deal with this, is by using cross-validation, in cross-validation we use a portion of the dataset to "train" the data, and a portion to "test" it.

> **Think about it**
>
> Why is it useful to separate your data-set in two to test your predictions useful?

We will do that for the following three models:

```
KgDMHA~ Water + Salinity + Nitrogen
KgDMHA~ Pests + Graze
KgDMHA~ Water + Salinity + Nitrogen + Pests + Graze
```

⚠ Assignment question 7:

Follow these seven steps:
1) Randomly split your dataset, with ~75% of the data for the training and ~25% of the data for the test
2) With the training dataset, run each of the three models. You should obtain an output for each model
3) Obtain an AICc value for each of the three models.

```
AIC(model)
```

4) In the test dataset, add three new columns called predictmodel1, predictmodel2, precitmodel3. Populate the columns with the predicted values based on each of the three models you ran. You can use the coefficients, or the predict function to obtain those values. It's easiest to use the `predict.lm` function. Let me know if you need help with this! Or check the last assignment!
5) Use the following equation:

$$RMSE = \sqrt{\frac{\sum(y_i - \hat{y}_i^2)}{n}}$$

RMSE is the root mean square error. Essentially, a measure of the distance between predictor and observation.
Estimate RMSE for EACH of the three models. Lower RMSE is better!
6) According to the RMSE, what model is the best?
7) Compare the RMSE with the AICc and with the R-squared from the models. Do all three metrics agree on the best model?

While this is great, it's painful and costly to separate your data! You want to use 100% of your data to make predictions! Also, oftentimes our datasets aren't big enough to separate them in two. In order to deal with this, we will use a cross-validation method that requires only one dataset. We will use a particular kind of cross validation known as K-fold. The conceptual steps of K-fold cross validation for model selection are as follows:

1. Randomly divide the data set into k number of groups (preferably equal size)
2. Fit a model to all but one of the groups

3. Calculate a metric such as RMSE using the observations from the k-th group that was not used to train the model
4. Repeat this process k-number of times using each group
5. Calculate the overall RMSE as the average of each calculated above
6. Repeat this process for each separate model you wish to compare
7. Compare the metric to the estimated metric from other possible models to select the 'best' model

The point is, you are testing your model in data that was not used to develop the model. Does that make sense? If not, wait until Monday or raise your hand! :)

It is easy to do:

```
library(caret)
```

```
Warning: package 'caret' was built under R version 4.3.3
```

```
Loading required package: lattice
```

```
ctrl <- trainControl(method= 'cv', number= 10)

abiotic <- train(KgDMHA~ Water + Salinity + Nitrogen, data= grasslanddata, trControl= ctrl,me
```

We can get the RMSE for each model using:

```
print(abiotic)
```

```
Linear Regression

1000 samples
   3 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 900, 900, 900, 900, 900, 900, ...
Resampling results:

  RMSE      Rsquared   MAE
  959.1829  0.3919653  771.6586

Tuning parameter 'intercept' was held constant at a value of TRUE
```

And we can get the predictive values from the training set using:

```
summary(abiotic$finalModel)
```

```
Call:
lm(formula = .outcome ~ ., data = dat)

Residuals:
     Min       1Q   Median       3Q      Max
-2991.99  -652.64    27.78   654.49  3105.87

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  4.373e+02  3.162e+02   1.383    0.167
Water        7.154e-03  3.805e-04  18.802   <2e-16 ***
Salinity    -1.189e+02  9.003e+00 -13.207   <2e-16 ***
Nitrogen     4.556e+02  3.969e+01  11.479   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 958.3 on 996 degrees of freedom
Multiple R-squared:  0.395, Adjusted R-squared:  0.3931
F-statistic: 216.7 on 3 and 996 DF,  p-value: < 2.2e-16
```

Assignment question 8:

Report the three RMSE values.
In future assignments you might end up running multiple models. We will use a combination of tidy, loops and other tools to test how to run these models and compare them relatively quickly.

7 Generalized Linear models

Just as a reminder... We use GLM's when the predictor and response variables **DO NOT** have an underlying normal distribution of the residuals or lack an underlying linear relationship.

In **Linear Models** we model the response as a function of the predictors.

In GLM, we model **a function** of the response as a function of the predictors. That function is called the link function. So, we are not modeling the response variable (usually "y") but a link function of the response. So, instead of modeling:

$$\hat{y} = \beta_0 + \beta_1 x_1$$

(where $\hat{y}$ represents the **expected** value of y) we model:

$$logit(p) = \beta_0 + \beta_1 x_1$$

or

$$log(\lambda) = \beta_0 + \beta_1 x_1$$

We use logit for logistic regression (aka binomial distribution), and we use log for Poisson distributed data.

> ⚠ Think about it
>
> How to interpret the $\beta's$?
> Everything to the right of the = sign is **THE SAME.** This means, the link function allows us to estimate link(y) as a linear model!
> In the simple linear model the β_0 is the expected (or predicted) value of y when x is 0. The β_1 is how much y changes for a one unit increase in x. This is called the slope and it is very useful! it can tell us how strong the effect of x is on y (effect size). We can analyze whether this effect is biologically important. We can analyze whether it is **significantly** different from 0. All in all, the $\beta's$ are super informative. And as the paper we read said, inference is about the $\beta's$!
> What happens when we have generalized models? The $\beta's$ represent effects **NOT** on the response variable, but on the **link function** of the variable. So β_1 represents how much

$log(\lambda)$ changes for a one unit increase in x. But it doesn't tell us how much λ changes. In order to figure this out, we need to do an inverse link equation:
$\lambda = e^{\beta_0+\beta_1 x_1}$
For this reason is important to:

1. Know the link functions
2. Know the **inverse** link functions
3. Know these functions in R

Here is a table with the link functions for both Binomial and Poisson distributions.

Save it somewhere important!

Distribution	link name	link equation	inverse link eq	link function R	inverse link in R
Binomial	logit	$\mu = log(\frac{p}{1-p})$	$p = \frac{e^\mu}{1+e^\mu}$	`qlogis()`	`plogis()`
Poisson	log	$\mu = log(\lambda)$	$\lambda = e^\mu$	`log()`	`exp()`

7.1 This assignment

We went through the Poisson example in class. You can also download the presentation file (from Canvas) and access all of the code we used. During this assignment we will focus on the following:

1. Logistic regression
2. Questions where you are presented with data, and a research question and you have to analyze the data. You need to decide what test or model to use, and interpret the results.

7.2 Logistic Regression

We use the logistic regression when we have binomial data. Remember, in binomial data the response variable is binary, our responses are limited to 0's and 1's. Which is which depends on you, but usually 1 is seen as a "success" or "positive".

Some examples:

1. Presence/absence
2. Alive/Dead
3. Homozygous/Heterozygous
4. Mature/non mature

5. Male/Female
6. pregnant / not pregnant
7. Healthy / Disease

> ⚠ Think about it
>
> What is the response variable?
> We usually obtain the **mean** of the response variable. In the grasslandexample, the response variable was **KgDMHA,** or Kilograms of Dry Matter per Hectare.
> In the logistic regression, we are trying to find the **probability** of an event happening. Look at the examples before this box. If the binary outcomes are pregnant or not pregnant, the response variable is the probability of being pregnant.
> If we were exploring whether cortisol has an effect on succesful pregnancies in mice, and the model was `glm(pregnancy~cortisol, data=data, family=binomial(link="logit")` then, what we are trying to estimate are 1) whether cortisol has an effect on the probability of being pregnant (inference) **and** 2) what is the probability of a mouse being pregnant given a specific cortisol measurement. This explains why it is not linear. Probabilities go from 0 to 1.

The link equation is the "log of odds", also known as logit. This equation allows us to move from a system were the values go from 0 to 1 (probability) to one where we can theoretically go from -INF to INF.

7.2.0.1 Log of odds? What is that? And what are odds?

The logit link function is: $\mu = log(\frac{p}{1-p})$ or also known as "*the log of odds*". This works because **any** probability can be converted to log odds by finding the **odds** and taking the log of that. **James Jaccard** calls log odds "counterintuitive and challenging to interpret". They are not as easy to interpret as the "log" we use in Poisson. However, I don't think you absolutely need to understand the transformation, you only need to understand why it is useful!

First off, the odds is $\frac{p}{1-p}$. It is simply dividing the probability of an event occurring divided by the probability of it not occurring. Let's look at some examples:

In a coin toss with 0.5 probability, the odds ratio is: $\frac{0.5}{1-0.5} = 1$. We can interpret this as the odds of getting a success (or head) is 1:1 (think 1 to 1, or 50-50)

Let's imagine the probability of getting a 5 after rolling one die. The probability of this event is $\frac{1}{6}$, so, the odds are:

```
(1/6)/(5/6)
```

```
[1] 0.2
```

The odds are 0.2. This means you will see 0.2 successes for every failure (or one success for every 5 failures).

Finally, let's think of the odds of me (Alejandro) seeing a car accident during my daily drive to and from campus). I know this is biased, and one semester I will actually sample this for an exercise, but my very biased estimate says the probability is 0.8 (80% chance of seeing one). If this was true, then the odds would be:

```
0.8/(1-0.8)
```

```
[1] 4
```

Four. Again, think of this as four to one. If I drove five times, I would see an accident 4 times, and "no accidents" one time.

OK, that was a lot of time spent on odds. And the reason I did that, is that my brain struggles thinking in odds rather than probabilities. Which is why we want to present the results in a probabilistic scale. However, this concept is important to understand how it is estimated.

Odds are always "positive". But the transformation need to potentially be from -inf to inf. Similar to what we do with counts, then we take the log of the odds.

Let's imagine the probability of winning the lottery is 0.000000000065789. Then, the log odds would be:

```
log(6.5789e-11/(1-6.5789e-11))
```

```
[1] -23.44457
```

-23.44. And if we had an event with high odds (like the accident one):

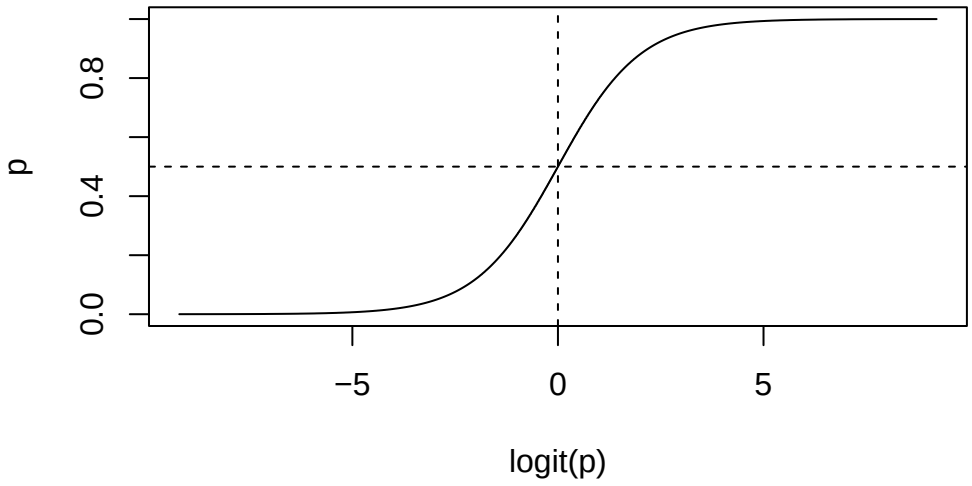
```
log(0.8/(1-0.8))
```

```
[1] 1.386294
```

Here you can see how values higher than 1 (p higher than 0.5) will be positive, and odds lower than 1 (p lower than 0.5) will be negative.

Actually, we can plot the relationship between p (probability) and logit(pi):

```r
pi = seq(0.0001,1-0.0001, by=0.0001)
plot(pi~qlogis(pi), type="l",xlab="logit(p)",ylab="p"); abline(v=0, lty="dashed"); abline(h=0
```



What I want you to take away from this whole section is the following:

> **Take home message**
>
> 1- logit(p) ranges from -Inf to +Inf as pi increases from 0 to 1 Take home message
> 2- logit(p) takes on a full range of values which allow the modeling algorithm to explore a full range of coefficient values in the systematic component of the model In other words, $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_m x_m$ can be unconstrained and can take values from -Inf to + Inf
> 3- logit(p) is `qlogis(p)` in r

7.2.0.2 Let's actually do a regression

This dataset (parasitecod.csv) was obtained from the following book:

Zuur, A., Ieno, E. N., Walker, N., Saveliev, A. A. & Smith, G. M. *Mixed Effects Models and Extensions in Ecology with R.* (Springer New York, 2009).

It's a highly recommended book (I showed it to you during the first week!)

Load it into R and let's run a glm.

```
cod<-read.csv("data/parasitecod.csv")
library(dplyr)
cod<-cod%>%mutate(across(c(Year,Sex,Stage,Area),as.factor))
```

Before running the glm, I changed some categorical data to "factor" in R. This step is really important, because if you don't do it, it will take the data as continuous!

Now, let's run the model!

```
codmodel1<-glm(Prevalence~Length+Year+Area,data=cod,family = binomial(link="logit"))
summary(codmodel1)
```

```
Call:
glm(formula = Prevalence ~ Length + Year + Area, family = binomial(link = "logit"),
    data = cod)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.465947   0.269333  -1.730 0.083629 .
Length       0.009654   0.004468   2.161 0.030705 *
Year2000     0.566536   0.169715   3.338 0.000843 ***
Year2001    -0.680315   0.140175  -4.853 1.21e-06 ***
Area2       -0.626192   0.186617  -3.355 0.000792 ***
Area3       -0.510470   0.163396  -3.124 0.001783 **
Area4        1.233878   0.184652   6.682 2.35e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1727.8  on 1247  degrees of freedom
Residual deviance: 1537.6  on 1241  degrees of freedom
  (6 observations deleted due to missingness)
AIC: 1551.6

Number of Fisher Scoring iterations: 4
```

Remember that those coefficients (think of each coefficient as a β, so this model has 7 $\beta's$) are for the link function:

! Assignment question 1

Look at the summary. And before continuing make sure you understand it. If you don't, now is the time to raise your hand.
You need to calculate the expected probability (we'll call this π) of an individual being infected for each of the following two cases:

1. An individual of length 50 from Area 1, in 1999
2. An individual of length 50 from Area 3, in 2001

Remember that:

$$log(\frac{\pi_i}{1-\pi_i}) = \beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i} + ... + \beta_m x_{m,i}$$

And we are trying to solve for π
Please do not use the function "predict". You can use algebra to solve this problem, or you can use the `qlogis` and `plogis()` functions.

We can use the packages `car` and `emmeans` to run an ANOVA and look for differences among some of the explanatory variables

First we run:

```
Anova(codmodel1)
```

```
Analysis of Deviance Table (Type II tests)

Response: Prevalence
       LR Chisq Df Pr(>Chisq)
Length    4.713  1    0.02993 *
Year     54.421  2  1.523e-12 ***
Area    143.079  3  < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

And see that there are significant effects of all three factors.

Let's explore the effects of Year running pairwise comparisons using `emmeans`:

```
emmeans(codmodel1, ~Year) %>% contrast("pairwise")
```

```
 contrast            estimate    SE  df z.ratio p.value
```

```
 Year1999 - Year2000   -0.567 0.170 Inf  -3.338  0.0024
 Year1999 - Year2001    0.680 0.140 Inf   4.853  <.0001
 Year2000 - Year2001    1.247 0.179 Inf   6.958  <.0001

Results are averaged over the levels of: Area
Results are given on the log odds ratio (not the response) scale.
P value adjustment: tukey method for comparing a family of 3 estimates
```

Please note that 1) results are averaged over the levels of Area, and 2) results are given on the log odds ratio scale. This is **not** the response scale. The response scale is probabilistic, so it goes from 0 to 1 (and therefore there could not be a difference of 1.247

> **!** Assignment question 2
>
> Run a pairwise comparison for Area, and a pairwise comparison for Year and Area

Finally, let's plot it!

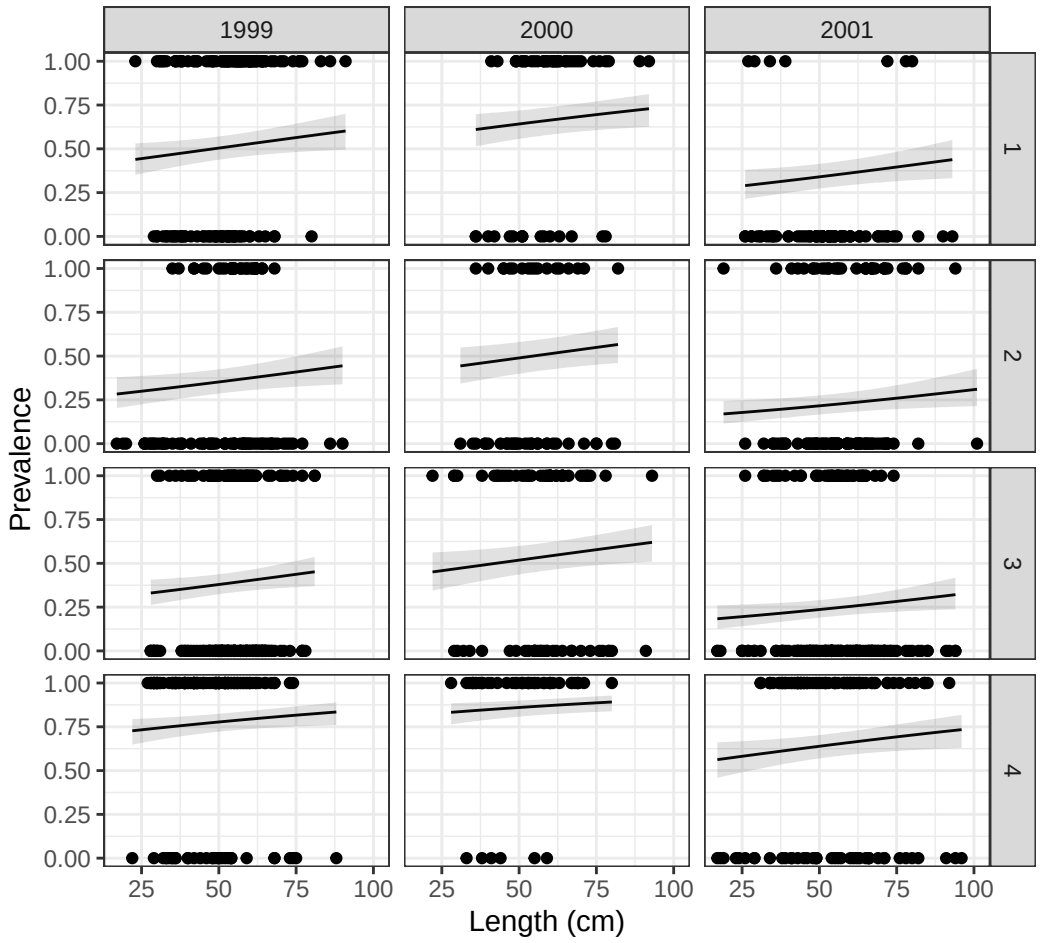
First, let's predict the values. Predict also gives us values on the log odds scale, so, some transformation is needed:

```
predictedmodel<-predict.glm(codmodel1,cod,se.fit = T)
ci_lwr <- with(predictedmodel, plogis(fit - 1.96*se.fit))
ci_upr <- with(predictedmodel, plogis(fit + 1.96*se.fit))
cod2<-cbind(cod,predictedmodel)
cod2$fit2<-plogis(cod2$fit)
cod2$lwr<-ci_lwr
cod2$upr<-ci_upr
```

And finally, let's plot it!

These type of models can be pretty tough to plot. One option is to use a grid, in which each column is a year, and each row is an area:

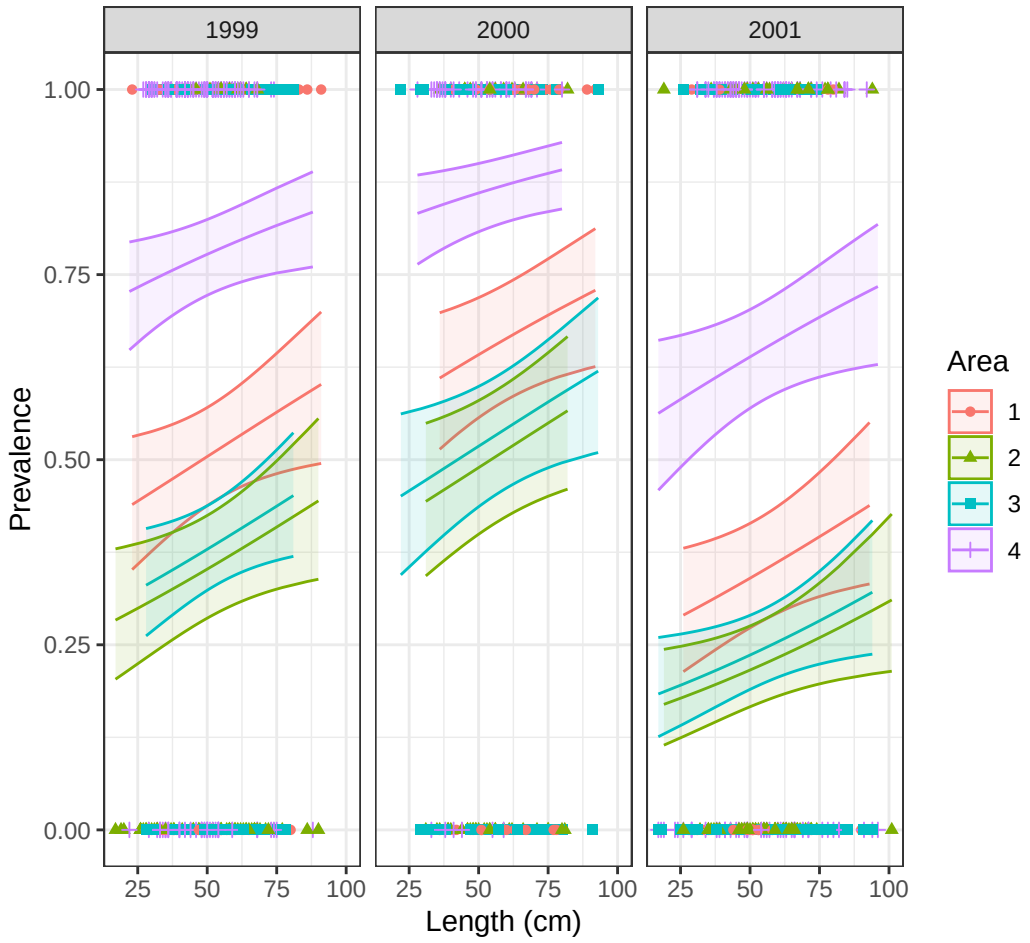
```
ggplot(cod2,aes(x=Length,y=Prevalence,ymin=lwr,ymax=upr))+
  geom_point()+
  geom_line(aes(y=fit2))+
  geom_ribbon(alpha=0.15)+
  xlab("Length (cm)")+
  ylab("Prevalence")+
  theme_bw()+
  facet_grid(Area~Year)
```



While tough to interpret, we can see some patterns. For example, fish from 2001 have a lower prevalence probability!

Another way to plot it, is to choose one categorical variable to be presented as colors, and the other one to be at the grid level:

```
ggplot(cod2,aes(x=Length,y=Prevalence,ymin=lwr,ymax=upr,color=Area,shape=Area,fill=Area))+
  geom_point()+
  geom_line(aes(y=fit2))+
  geom_ribbon(alpha=0.1)+
  xlab("Length (cm)")+
  ylab("Prevalence")+
  theme_bw()+
  facet_grid(~Year)
```



In here, we can see two patterns: 2001 has lower probability of prevalence, while area 4 has a higher probability. Finally, size also has an effect.

! Assignment question 3

Look at the cod data and come up with 3 "biological hypotheses" that you can run as models. Run the three models
Use model selection or any method that you want to compare the 4 models (the model I ran; codmodel1; and the models you ran). And select the best model.

! Assignment question 4

For the best model from question 2 (if the best model is the one I ran, then select the second best model) please do the following:

1. Interpret the summary output
2. Run an ANOVA (and if needed) a pairwise comparison
3. plot the model. If the model seems too complex to plot, ask for my help, and we can figure it out together

7.3 Testing your knowledge

Moving forward, for each assignment I will ask you 1 or 2 questions in which you will have to apply knowledge from previous classes and assignments.

! Assignment question 5

Data: `welldata.csv`
1. wellID - ID of well
2. fluoride - mg/L of fluoride in a sample

Research question:
You are tasked with researching whether the mean content of fluoride in a large rural area with >2,000 private wells might be over the EPA recommendation of 4.0 mg/L. You sample 28 wells.

! Assignment question 6

Data: `parasitecod.csv`
1. Intensity: Number of parasites present
2. Prevalence: 1-parasite present, 0- no parasite
3. Year
4. Depth (in meters)
5. Weight (g)
6.Length (cm)
7. Sex
8. Stage
9. Age
10. Area

Be aware! The weight, length, stage and age covariates are highly co-linear! Use only one at a time.
Explore 3 different hypotheses (make sure to include a null model as part of your hypotheses), but use intensity as your response variable.

Be aware! The `intensity` response variable is missing some data! Thee are cases were the number of parasites were not counted. You need to deal with this before you can run the models.

References

Box, George E. P., and Norman Richard Draper. 1987. *Empirical model-building and response surfaces*. 7. Dr. Wiley series in probability and mathematical statistics. Applied probability and statistics. New York: Wiley.

Burnham, Kenneth P., and David Ray Anderson. 2010. *Model selection and multimodel inference: a practical information-theoretic approach*. 2. ed. New York, NY: Springer.

Childs, Dylan. 2024. "Chapter 10 Working Directories and Data Files." In. https://dzchilds.github.io/eda-for-bio/working-directories-and-data-files.html.

Scott, James. n.d. *Data Science in r: A Gentle Introduction*. https://bookdown.org/jgscott/DSGI/.

Tredennick, Andrew T., Giles Hooker, Stephen P. Ellner, and Peter B. Adler. 2021. "A Practical Guide to Selecting Models for Exploration, Inference, and Prediction in Ecology." *Ecology* 102 (6): e03336. https://doi.org/10.1002/ecy.3336.

Ward, Caitlin, and Collin Nolte. 2024. *An Intuitive, Interactive, Introduction to Biostatistics*. https://collinn.github.io/sbi/index.html#acknowledgements.

Wickham, Hadley. 2014. "Tidy Data." *The Journal of Statistical Software* 59 (10). http://www.jstatsoft.org/v59/i10/.

Wickham, Hadley, Mine Çetinkaya-Rundel, and Garrett Grolemund. 2023. *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. Second edition. Beijing ; Sebastopol, CA: O'Reilly.

A Installing R and RStudio

A.1 Install R and RStudio

A.1.1 Install R

R can be downloaded from the following page https://cran.r-project.org/, follow the instructions to download it.

A.1.1.1 Windows

To install R on Windows, click the "Download R for Windows" link. Then click the "base" link. Next, click the first link at the top of the new page.

A.1.1.2 Mac

To install R in a Mac the "Download R for Mac" link. Next, click on the `R-4.4.2` package link.

A.1.1.3 Chromebook

Chromebooks don't allow you to install R or the RStudio program. However, there is an online platform that will allow you to run RStudio. If you go to https://posit.cloud , you should be able to create an account, and in your workspace you can open a new RStudio project. Your window will look the same as what we are working with, though there may be some quirks to getting

A.1.2 Install RStudio

Go to https://posit.co/download/rstudio-desktop/

Click on the Download RStudio Desktop link. If you are using a Mac, you will need to scroll down a bit to see the download link.

A.2 Explore R and RStudio

RStudio is a program, not different than Word. It works as a "wrapper" or "editor". You will write the code in RStudio, and it will run it in program R. You will *rarely* or maybe **never** open program R. You will do everything from RStudio!

> **Tip**
>
> RStudio needs R to function, so you need to download both programs!

After you open RStudio, this is what you will see:

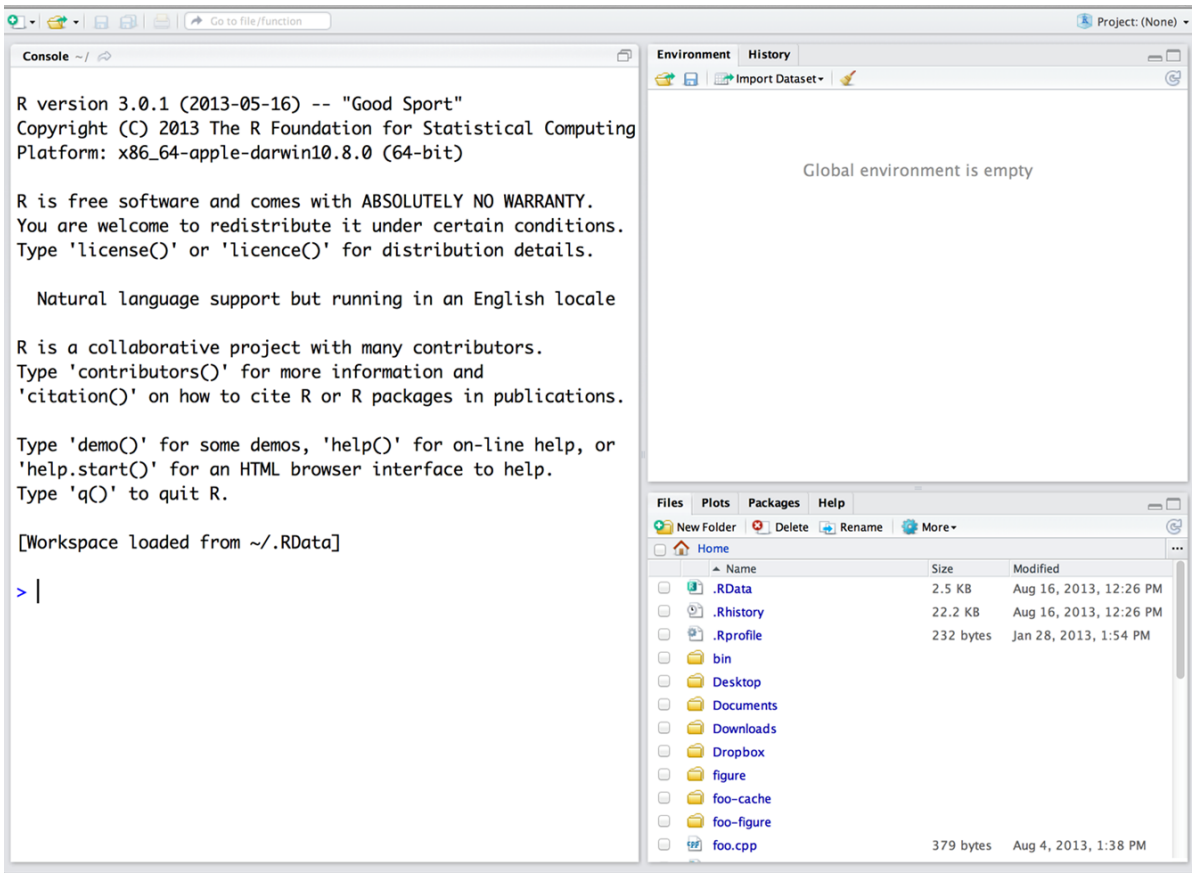


Figure A.1: RStudio Window

Now, you can code in R!

A.2.1 Let's check R and RStudio!

Let's do a couple of things in R. Run the following lines (or similar, you can use different numbers!) on the **Console** (big screen on the left!)

First, let's do basic math:

```
5+5
10*7
89-5
90/3
```

As you can see, we can use R as a calculator

Now, let's create a vector

```
1:10
```

```
120:170
```

> **Try it!**
>
> Now, try to create a vector from 200 to 300

Finally, let's roll a die!

We will roll a "D20" die (this is a die with 20 sides). First, we need to create this die. We can create objects in r using the following symbol: `<-` . Objects will save data, and we can use the object name to extract the data. We can create our dice using the following:

$$\underbrace{D20}_{Object\ name} \quad \underbrace{<-}_{arrow} \quad \underbrace{1:20}_{Data}$$

The arrow is created with these two symbols: `<` and `-` .

```
D20<-1:20
```

Now, if you do the following:

```
D20
```

You extract the data from the object. Pretty cool!

Now, let's roll our die!

```
sample(x=D20, size=1)
```

Let's now roll 3 D20 dice:

```
sample(x=D20, size=3, replace=TRUE)
```

> **Think about it!**
>
> What do you think the `replace=TRUE` does?

B Data Wrangling

Data wrangling might be very important for you in your career. While this topic doesn't fall neatly in any of our topics, I usually decide to have a break around week 5 and look at this.

This assignment was written based on the following literature:

1. R for Data Science (2e) written by Hadley Wickham, Mine Çetinkaya-Rundel, and Garrett Grolemund. Wickham, Çetinkaya-Rundel, and Grolemund (2023) And available online (for free!) at: https://r4ds.hadley.nz/ Please be aware that this book is the **second** edition and has changed dramatically from the first edition. I highly recommend it! Even if you have read the first edition
2. The original "Tidy Data" paper. (Wickham 2014) http://www.jstatsoft.org/v59/i10/.
3. Data Science in R: A Gentle Introduction. (Scott, n.d.) https://bookdown.org/jgscott/DSGI/
4. The Tidy Data vignette: https://tidyr.tidyverse.org/articles/tidy-data.html
5. The dplyr vignette: https://dplyr.tidyverse.org/
6. Some videos like the following: https://www.youtube.com/watch?v=1ELALQlO-yM&ab_channel=PositPBC

I hope this is a useful assignment for you, and I highly recommend you check all of those tools! Hadley Wickham is the chief scientist at RStudio and he is the creator of ggplot, dplyr, and many other great tools.

The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures. The tidyverse is "opinionated" in that it is fairly rigid about input types and output types, and most functions are designed to accomplish one specific task. This makes it easier for users to avoid some of the sneakier bugs that are hard to notice (for example: different output types of the apply functions). It also tends to make analyses more reproducible.

The packages are:

1. ggplot2: You have been using this already!
2. dplyr: In my mind the most important package in the tidyverse. Plenty of tools
3. tidyr: Make tidy data: every variable goes in a column, and every column is a variable
4. readr: I rarely use this one, but you can read csv's and other dataframes into "tibbles" (tidy's idea of a data frame)
5. purrr: Provides a set of tools for working with vectors and functions

6. tibble: Modern re-imagining of the data frame, keeping what time has proven to be effective, and throwing out what it has not
7. stringr: Helps when working with strings (aka words and letters)
8. forcats: Solves some issues with factors and provides tools to work with them (remember, a factor is a "grouping variable"

You can install all of them using the following:

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

You can also load ALL of them using the following:

```
library(tidyverse)
```

> ⚠ What are all these conflicts?
>
> When you load the package, you will see a lot of conflicts. This is normal in `r` and it becomes pretty common the more packages you use.
> Sometimes, different packages name their functions the same (for example `some()` is a function on dplyr, as well as a basic function in the package car).
> If you want to force `r` to use the function in the package `car` you can code it the following way: `car::some()` and if you want to use the purrr versions use: `purrr::some()`
> I recommend doing this, particularly if you are collaborating on a project, using version control, creating websites or apps.

While loading all the packages at once is useful, I recommend you load the specific packages you want!

B.1 Additional tidy packages

I also recommend you use the package `lubridate` if you ever work with dates!

B.2 My thoughts

Before continuing I do want to admit to the fact that I generally prefer a combination of baseR and the tidyverse, and I will continue supplying code in `baseR`. I also really enjoy using `data.table` which is faster and has some great functionality, and I recommend you check it out after you master tidy. However, I do recognize that a majority of users prefer the tidyverse, and have fully transitioned to using it.

Particularly, most users find that the syntax can be easier to understand and write, as it is written from left to right (instead of inside-out). You are free to use any package or any syntax that works best for you.

While early in the semester I had some sections in which I expected you to run some specific code (with the idea of showcasing some of `baseR` syntax), going forward the focus will be in the results. If you identify the analysis needed, and you do it, it will be correct! No matter hoe you get there!

> Tibble, data.frame, what's the difference?
>
> I don't want you to focus on this too much. These two types of objects have different structures, and the tidyverse uses tibble. There isn't anything too fundamentally different between them, so for the time being, just think of them as equal. I will continue to refer to them as data.frame

B.3 Data Wrangling

There are three main parts for data analysis:

1. Managing (or Wrangling) the data
2. Analyzing it (vi visualization and modeling)
3. Communicating the results

I like this figure from the **first edition** of R for Data Science: https://r4ds.had.co.nz/index.html

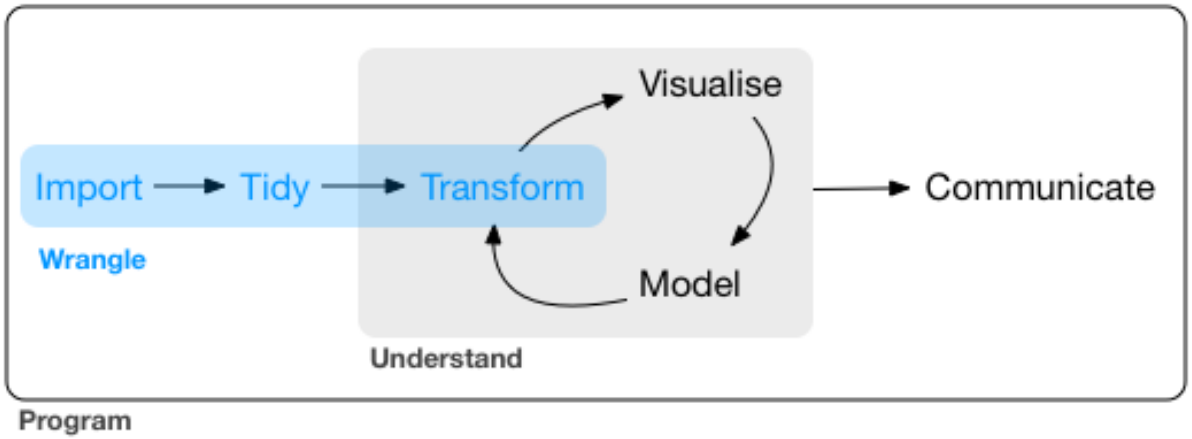


Figure B.1: Data wrangling (read R for Data Science)

We have been working on visualizing and modeling data (mostly modeling though). Now, we need to focus on *wrangling* data.

As you see in the image, there are three main important sections when wrangling data:

1. Import
2. Tidy
3. Transform

B.3.1 Import

> read.csv vs read_csv
>
> In the last ~5 years there has been a migration from `read.csv()` to `read_csv()`. Main differences are: 1) `read.csv()` doesn't require a package and creates a data.frame, while `read_csv()` creates a tibble

You have so far been importing data using the `read.csv` function by saving the file in the same directory as the `quarto` file. However you should be able to do all of the following:

- Create a project and load files from within the project
- Add a folder to the project and read files from the folder, not the `root` of the directory
- Open an R script (not quarto, not in a project) and import a dataset
- Change the working directory and check the working directory from within `R`

If you can do all of that, congrats! You have 1/3 of the skills needed to wrangle data in `R` and you can **skip the first assignment question.**

If you don't know how to do these things, here is your first **assignment question**: go to https://www.statology.org/import-csv-into-r/ and follow the first two methods and make sure you can do all these things

B.3.2 Tidy and transform

Let's talk about probably the biggest change using tidy.

For this we will use a "pre-loaded" data set in R. The `iris` dataset. This dataset is included with `baseR` and you can call it using the `data()` function:

```
data(iris)
summary(iris)
```

```
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
 Median :5.800   Median :3.000   Median :4.350   Median :1.300
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
       Species
 setosa    :50
 versicolor:50
 virginica :50
```

If we wanted to get a dataframe with only *setosa* species, add a new column where we multiply the width of the pedal by 10 (to change units) we would do the following in baseR:

```
Setosa<-iris[iris$Species=="setosa",]
Setosa$widthm<-Setosa$Sepal.Width*10
mean(Setosa$widthm)
```

```
[1] 34.28
```

If we used the tidyverse

```
library(tidyverse)
```

```
Warning: package 'tidyverse' was built under R version 4.3.2
```

```
Warning: package 'ggplot2' was built under R version 4.3.3
```

```
Warning: package 'lubridate' was built under R version 4.3.2
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.2     v readr     2.1.4
v forcats   1.0.0     v stringr   1.5.0
v ggplot2   3.5.0     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.0
v purrr     1.0.1
```

```
-- 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 becor
```

We would do the following:

```
iris %>% filter(Species=="setosa") %>% mutate(widthm = Sepal.Width*10)
```

```
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species widthm
1           5.1         3.5          1.4         0.2  setosa     35
2           4.9         3.0          1.4         0.2  setosa     30
3           4.7         3.2          1.3         0.2  setosa     32
4           4.6         3.1          1.5         0.2  setosa     31
5           5.0         3.6          1.4         0.2  setosa     36
6           5.4         3.9          1.7         0.4  setosa     39
7           4.6         3.4          1.4         0.3  setosa     34
8           5.0         3.4          1.5         0.2  setosa     34
9           4.4         2.9          1.4         0.2  setosa     29
10          4.9         3.1          1.5         0.1  setosa     31
11          5.4         3.7          1.5         0.2  setosa     37
12          4.8         3.4          1.6         0.2  setosa     34
13          4.8         3.0          1.4         0.1  setosa     30
14          4.3         3.0          1.1         0.1  setosa     30
15          5.8         4.0          1.2         0.2  setosa     40
16          5.7         4.4          1.5         0.4  setosa     44
17          5.4         3.9          1.3         0.4  setosa     39
18          5.1         3.5          1.4         0.3  setosa     35
19          5.7         3.8          1.7         0.3  setosa     38
20          5.1         3.8          1.5         0.3  setosa     38
21          5.4         3.4          1.7         0.2  setosa     34
22          5.1         3.7          1.5         0.4  setosa     37
23          4.6         3.6          1.0         0.2  setosa     36
24          5.1         3.3          1.7         0.5  setosa     33
25          4.8         3.4          1.9         0.2  setosa     34
26          5.0         3.0          1.6         0.2  setosa     30
27          5.0         3.4          1.6         0.4  setosa     34
28          5.2         3.5          1.5         0.2  setosa     35
29          5.2         3.4          1.4         0.2  setosa     34
30          4.7         3.2          1.6         0.2  setosa     32
31          4.8         3.1          1.6         0.2  setosa     31
32          5.4         3.4          1.5         0.4  setosa     34
```

```
33          5.2         4.1          1.5         0.1  setosa     41
34          5.5         4.2          1.4         0.2  setosa     42
35          4.9         3.1          1.5         0.2  setosa     31
36          5.0         3.2          1.2         0.2  setosa     32
37          5.5         3.5          1.3         0.2  setosa     35
38          4.9         3.6          1.4         0.1  setosa     36
39          4.4         3.0          1.3         0.2  setosa     30
40          5.1         3.4          1.5         0.2  setosa     34
41          5.0         3.5          1.3         0.3  setosa     35
42          4.5         2.3          1.3         0.3  setosa     23
43          4.4         3.2          1.3         0.2  setosa     32
44          5.0         3.5          1.6         0.6  setosa     35
45          5.1         3.8          1.9         0.4  setosa     38
46          4.8         3.0          1.4         0.3  setosa     30
47          5.1         3.8          1.6         0.2  setosa     38
48          4.6         3.2          1.4         0.2  setosa     32
49          5.3         3.7          1.5         0.2  setosa     37
50          5.0         3.3          1.4         0.2  setosa     33
```

So, there are two main differences. `tidy` reads "left to right", so, you filter the data, and the mutate it. But maybe the main difference is the use of a piping operator `%>%`.

R pipes are a way to chain multiple operations together in a concise and expressive way. They are so popular thar `R` introduced them to their base use as: `|>`.

Now, imagine you want to

1) add the widthm column to the original dataset

2) estimate all of the following for each of the three species **and** each of the columns:

Mean, median, maximum value, and minimum value.

That seems like a lot of coding using `baseR`! However, using pipes and the powerful dplyr, we can do the following:

```
iris %>% group_by(Species) %>% mutate(widthm = Sepal.Width*10) %>% summarise_all(list(mean=me
```

```
# A tibble: 3 x 21
  Species  Sepal.Length_mean Sepal.Width_mean Petal.Length_mean Petal.Width_mean
  <fct>                <dbl>            <dbl>             <dbl>            <dbl>
1 setosa                5.01             3.43              1.46            0.246
2 versico~              5.94             2.77              4.26            1.33
3 virgini~              6.59             2.97              5.55            2.03
# i 16 more variables: widthm_mean <dbl>, Sepal.Length_median <dbl>,
```

```
#   Sepal.Width_median <dbl>, Petal.Length_median <dbl>,
#   Petal.Width_median <dbl>, widthm_median <dbl>, Sepal.Length_max <dbl>,
#   Sepal.Width_max <dbl>, Petal.Length_max <dbl>, Petal.Width_max <dbl>,
#   widthm_max <dbl>, Sepal.Length_min <dbl>, Sepal.Width_min <dbl>,
#   Petal.Length_min <dbl>, Petal.Width_min <dbl>, widthm_min <dbl>
```

B.3.3 Wide and long data

One of the most important transformation is changing your data from wide to long format (or long to wide).

While we generally have this structure in our data:

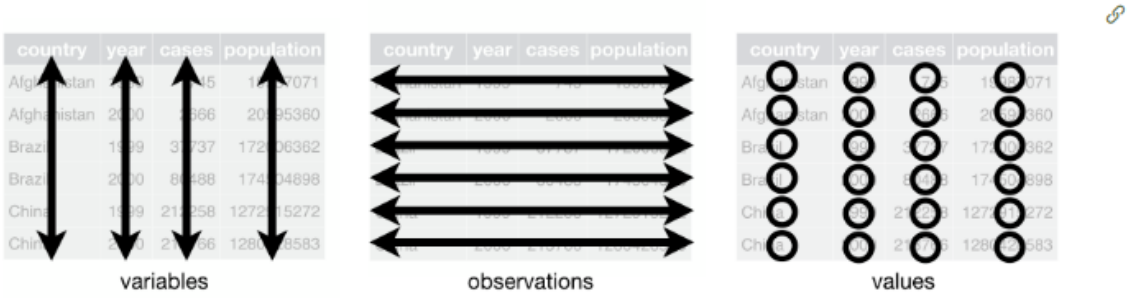


Figure B.2: Variable, observations, and values

If we have some repeated measures (think, an individual gets measured weekly), we can have each week be a column, or we can have a column named "week" and a column for the "value".

Think about the following example: You are traveling to four sites and measuring the diameter at breast height for 3 species of tree. You do this at four sites. There are two ways you can present these data:

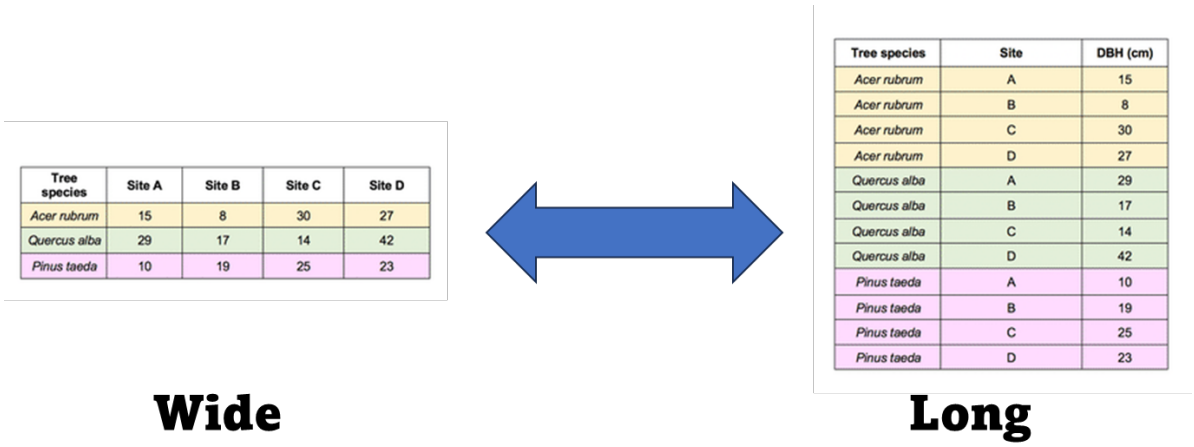


Figure B.3: Wide and Long data

Both can be useful! And we can go back and from using R.

```
tree<-read.csv("data/trees.csv")
head(tree)
```

```
       Species Site.A Site.B Site.C Site.D
1  Acer rubrum     15      8     30     27
2 Quercus alba     29     17     14     42
3  Pinus taeda     10     19     25     23
```

Let's make the dataset longer! We need to make all columns (except species) into two columns, one for the site, one for the value:

```
treelong<- tree%>% pivot_longer(!Species,names_to = "site", values_to = "meanDBH")
head(treelong)
```

```
# A tibble: 6 x 3
  Species      site   meanDBH
  <chr>        <chr>    <int>
1 Acer rubrum  Site.A      15
2 Acer rubrum  Site.B       8
3 Acer rubrum  Site.C      30
4 Acer rubrum  Site.D      27
5 Quercus alba Site.A      29
6 Quercus alba Site.B      17
```

And let's make this "long" dataset into a wide form:

```
treewide<- treelong%>% pivot_wider(names_from = "site", values_from = "meanDBH")
treewide
```

```
# A tibble: 3 x 5
  Species      Site.A Site.B Site.C Site.D
  <chr>         <int>  <int>  <int>  <int>
1 Acer rubrum      15      8     30     27
2 Quercus alba     29     17     14     42
3 Pinus taeda      10     19     25     23
```

Easy! This will be useful for you in the future!

Please read careful before starting to work on the assignment: For this week's assignment, you need to go to: https://bookdown.org/jgscott/DSGI/data-wrangling.html and read chapter 6. Reality is, this "tutorial" and this book as well as the *R for Data Science* book are better written and have better information than what I could do for this class, and I think it will be more useful to you!. **If you are new** to `R` **and the** `tidyverse` your assignment I recommend you open a new r script and follow the "tutorial". You can ask me questions. The assignment will be to just let me know in Canvas whether you like this way (or baseR) better. **If you are experienced** in tidy, and you know the basics, then, I recommend you read chapter 17 https://bookdown.org/jgscott/DSGI/probability-models.html. (Actually, everyone should read it, if you have time). If you understand probability, then you understand statistics, whether they are frequentist, Bayesian, multivariate, etc. You can also read any other chapter if you find it more useful. Your assignment is telling me what chapter you worked on.
