````
---
title: "Chapter 1"
output:
  html_document
---

# Chapter 1 Data in R

This .rmd file is where you will be doing all of your work this semester. We will generally be following the book
*Fundations of Statistics with R*. After each chapter, you will turn in your .rmd file for a grade.

Using RMarkdown is easy. Above you will see a 'Knit' button. Try pressing the `Knit` command and see what happens.

<https://media.csuchico.edu/media/Math+350+Chapter+1+Part+1/1_whzfu54r>


## Section 1.1 Arithmetic and variable assignment

The following code chunck in R demonstrates how R can be used as a calculator. Click the green arrow on the right to
run the code below.

```{r}
1+1
4-3
3*7
8/3
2^3
pi^2
2*exp(1)
```

There are many built in functions in R such as `log` or `exp`. Using these functions in R is not much different than
using them on your calculator. The function `log` has two arguements, one is required and one is set to a default,
$log_{b}(x)$ where *b* is the base. The default is *b* which is set to `exp(1)` (i.e. the natural logarithm). Below
are some examples of built in functions in R.

```{r}
exp(2)
log(8)
log(8,base=2)
```

### Storing results

Most of the time we will want to save our results. To do so we use the `<-` notation as shown below:

```{r}
height <- 62 # in inches
height <- height + 2
height <- 3 * height

```

To see the final value of height above we can either type the word or put the last line in parentheses.

```{r}


(height <- 3 * height)
```

## 1.3 Vectors

A vector is a list of values in order to be able to work with them. For us they will usually represent data collected
on a charactertic of the population. In general, we want to give the vector a name so that we can call it later when
needed.

```{r}
##Vector of the first 5 primes
primes_5 <- c(2,3,5,7,11)


##Vector of the numbers 1 through 10
one_ten<- c(1,2,3,4,5,6,7,8,9,10)
one_ten<-1:10
one_ten<-seq(1,10,by=1)

## We can use the "rep" function to repeat a number of sequence of numbers

x<-rep(c(2,3),times=c(2,2))
x #viewing the results

(y<-rep(c(2, 3), length.out = 3))

x<-rep(2,3)
````
````

```
x #viewing the results

x<-rep(c(2,3),4)
x #viewing the results

##The vector does not have to be numerical if we use quotes around the word

rep(c("Bryan", "Darrin"), each = 2)# tick marks or quotes will work

```

1.4 Indexing vectors

<https://media.csuchico.edu/media/Math+350+Chapter+1+Part+2/1_q7zivi6q>

We can index vectors to pull off values at a particular position on a vector.

```{r}
##Returns first and second number of the vector we called "primes" above
primes_5 #the entire c vector
primes_5[1] #returns the element at teh first index in vector c
primes_5[2] #returns the element at eht second index in vector c index start at 1!!!

##Returns first three numbers of a vector

x<-c(1,2,3,4,5,6) #assigning a vector using C function
x[1:3] #prints the first three elements in the vector x
x #prints all the elements in the vector x
```

We can change a vector to TRUE and FALSE.

```{r}


##first Identify the numbers in primes that are greater than 6

primes_5 #lists all the numbers in the vector
primes_5>6 #R studio goes threw vector c and assigns true or false values based on the element in the vector
as.numeric(primes_5>6)# converts the trues to 1s and false to 0s


## And count the number of primes greater than 6
mean(primes_5>6)
sum(primes_5>6)
```

Built in data sets in R

R comes with many built in data sets such as **rivers**. Rivers contain data on the lengths in miles of 141 major rivers in North America.

```{r}
## To see the data set

rivers

## To see some of the data

head(rivers,n=10)

##Another data set

discoveries

##To make a table of discoveries
table(discoveries)

##To make a historgraph of discoveries
hist(discoveries,col="red")
```

1.5 Data Types

All data in R has a type such as numeric for real numbers or integer for a variable that is all integers. A character variable refers to a variable where observations are recorded into categories. Read the section in the book to learn about the other types of variables.

Missing Data

Missing data comes up frequently in data. R denotes a missing value with NA. Let's consider an example using the built in data set `airquality` and the variable `Ozone` which shows the daily Ozone levels (ppb) in New York during the summer of 1973.

```{r}
```

```
airquality$Ozone
```

We can see that there are multiple missing values.

Suppose we want to find the average Ozone level. We would use the function **mean** in R like so:

```{r}
mean(airquality$Ozone) #we have na values that we need to remove default is NA values are counted!
```

The mean is the sum of all the values divided by the number of observations. Because of the NAs, R doesn't know how to sum the values. Therefore, we need to tell R that we want to ignore the missing values. The following code lets R know that it can skip any observation with a missing value.

```{r}
mean(airquality$Ozone,na.rm=TRUE) #we have na values that we need to remove

```

The default of the mean function is to include all values. When we type na.rm=TRUE it changes the default so that R knows to remove NAs.

1.6 Data Frames

<https://media.csuchico.edu/media/Math+350+Chapter+1+Part+3/1_zsrnmvdu>

So far we have looked at data that came in single vectors. However, most data have several characteristics (variables) recorded on each observation. Let's consider the built in data file called mtcars consists of 32 observations and 11 variables.

```{r}
mtcars #displays the values as a vector
View(mtcars) #displays as a table
?mtcars #if you have any questions use the ? to explore
```

Before we start to play with the data we should find out more information about it. We might want to know how many observations (in this case cars) are in the data set and what characteristics (variables) have been collected on each car.

```{r}
str(mtcars)

## gives names of the data set
names(mtcars)
```

We can extend our indexing idea to data frames. We can think of the data frame as a matrix with rows and columns. We still use the square brackets to index the observations, however, we need to give a row number and a column number. The row number will be first and the column number second. For instance, if we want to know the value in the third row and 6th column of mtcars we would type:

```{r}
mtcars[3,6]
```

Most of the time we want all values in a particular row or a particular column. We can do this by leaving one blank.

```{r}
## To get the values in the third column (the variable disp)
mtcars[,3]
mtcars$disp
mtcars[3,3] #gets the displacement variable for the Datsun

## To get the values in the third row (for the Datsun)
mtcars[3,]


```

If we wanted information on a particular variable within the data set we need to tell R what dataset we want and the variable within the data set that we are interested in. To do this we use the $ to connect the data set with the variable. For instance, suppose we are interested in the variable mpg.

```{r}
mtcars$mpg
mtcars[,1:3] #first three coloumns
mtcars[,c(1,3,5)] #using a vector to decide the columns we want
```

1.8 Packages

<https://media.csuchico.edu/media/Math+350+Chapter+1+Part+4/1_12o3ugqg>

Packages provide different features of R that aren't available in the "Base R". R packages are available freely and provide additional functions and data to work with. Installing packages is extremely easy. You can think of packages like apps on your phone. There are 2 steps to using an R package, the first is to install it and the second is to load it. This is similar to an app on your phone in that first you must download an app to your phone. When you want to use the app you generally have to open it first. You only have to install a package once, however, whenever you want to use the package you have to load it. We use the function "install.package" to download the package to your computer and we use the function "library" whenever we want to use the package. The following code installs the package "fosdata" which is the R package that includes many of the data sets we will use in this course.

```{r}
#install.packages("remotes")
#remotes::install_github("speegled/fosdata")

##open a package we use library
library(dadjoke)
?dadjoke
```

Now that this package is installed, we will not have to install it again (unless you switch computers), however, we won't be able to use the package unless we load it (just like you can't use your Starbucks app unless you open it on your phone):

```{r}
#library(fosdata)
```

End of Chapter Problems

1. Let x<-c(1,2,3) and y<-c(6,5,4). What happens when you run `x*2` b) `x*y` and c) `x[1] * y[2]`?

```{r}
x<-c(1,2,3)
y<-c(6,5,4)

x*2 #I predicted it x would be (2,4,6) how ever I am not sure because we aren;t saving to x... or are we?
x

y
x*y #I predicted x would be (6,10,12)
x
y

x[1] * y[2] #I predicted x would be 5
x
y


```

2. Use `seq` to create a vector `p` of numbers from 0 to 1 spaced by 0.2.

```{r}
p <- seq(from=0, to=1,by=0.2)
p
```

3. Use R to calculate the sum of the squares of all numbers from 1 to 100: $1^{2}+2^{2}+\cdots +100^{2}$

```{r}



x <-sum(c(1:100)* c(1:100))
x
```

4. What is the sum of the first 100 positive integers? The formula for the sum of integers 1 through n is $n(n+1)/2$. Define $n=100$ and then use R to compute the sum of 1 through 100 using the formula. What is the sum?

```{r}

n <-100
n*(n+1)/2
n
```

5. Consider the built-in data frame `airquality`.

  a. How many observations are there? How many variables?

```{r}
str(airquality)# 153 observations and 6 variables in this data set
```

  b. What are the names of the variables?

```{r}
names(airquality)#this code answers the question!
```

  c. What type of data is each variable?

```{r}

```

6. Consider the `mtcars` data set.

 a. How many cars have 4 forward gears?

```{r}
names(mtcars)
mtcars$gear
sum(mtcars$gear==4)#set it to true to false
?mtcars


```

 b. What subset of `mtcars` does mtcars[mtcars$disp > 150 & mtcars$mpg > 20,]
describe?
The data set contains cars with a displacement value greater than 150 and an mpg greater than 20. It is pulling out the specific rows since it is indexed before the comma.

 c. How many cars have 4 forward gears and manual transmission? Note that manual transmission is 1 and automatic is 0 and the variable name is `am`.

```{r}
sum(mtcars$gear==4&mtcars$am==1)

```

 d. Find the mean mpg of the cars with 2 carburetors.

```{r}
mean(mtcars$carb==2) #my first wack at it
mean(mtcars$mgp[mtcars$carb==2]) #what ended up working

```