Math 136 - Lab 1

Intro to R and Descriptive Statistics

Dr. Jorge Basilio

March 10, 2020

Learning Objectives:

Learn why we are using R

Learn the basics of R

Learn how to enter data in R

Learn how to compute descriptive statistics in R

Learn about dataframes in R

DUE: Friday, March 13, by 11:59 PM

Section 0 - Intro to R

What is R?

R is a powerful, comprehensive, open-source software framework for doing professional statistics.

It is possible to download and install the software on computers, or to use it through a website-interface without downloading anything.

We will use R through a website-interface in the form of a Jupyter notebook or R Markdown document. In fact, what you are reading here is an R-Markdown document that will guide you through the first steps of getting familiar with R.

Why use R?

- It's free!
- Using it on web is free via CoCalc (hassle-free, no messy downloading needed)
- Downloading it is free (if you want to use it without internet)
- It's open source!
- No hidden algorithms (you can look at source code if you wish–and look "under the hood" so to speak)
- Lots of resources to get help
- It will prepare you for the future!
  - You will likely need to learn some basic programming no matter what you study in college
  - In Pyschology, for example, it's common to learn Python together with R when learning statistics
  - Data Analyst: hot new field where there's lots of jobs!
  - It truly and honestly answers the question student's always ask: "When will I ever use this???"

Watch this shot video introduction:

Getting Started

- Make an account at the CoCalc website page using the invitation email I sent. Then **login** to the free server. No nosy questions, just make up a username and set a password. Just be sure to use a modern web browser (Google Chrome, Mozilla Firefox, etc). You might need to update your browser if it is old. Also, be sure to use your Glendale email account using the invitation I sent. You need to do this so you can access the lab files.
- Navigate to the Labs folder and find the Lab_1 folder. Inside you will find a Juptyper Notebook which can run and execute R code titled **"Math136-Lab_1-YourLastName_YourFirstName-S20"**. Re-name the file with the obvious modifications.
- At the beginning of your Jupyter Notebook, double-click on the "Lab1" text. Replace the text "FirstName LastName" with your **actual** first and last name. Click "run cell" (looks like a play button) or hit `shift+enter`.
  - By looking at this document, you are encouraged to copy and paste lines of code and modify them :-)
- Have some fun and make a few calculations

Section 1 - Basics of R

Simple Calculations

You can perform all sorts of basic arithmetic using R:

```
2+2
```

```
## [1] 4
```

Of course we know 2+2 is 4. In the above, you'll see the gray box shows the **code**, `2+2` and in the box blow that you'll see the **output** `4`.

We can input much more complicated expressions as well. Just use parentheses liberally!

If we want to evaluate the expression $\sqrt{\frac{2+5\cdot 7}{10}}$ we need to know what the square root, multiplication, and division syntax used.

```
sqrt( (2+5*7) / 10)
```

```
## [1] 1.923538
```

You can add spaces so that it's easier to understand/read the code.

Assignments and Variables

Let's look at one more useful feature of R: **assignment**.

If you type the symbols less-than and hyphen, you type something that looks like an arrow pointing to the left: `<-`. You can use this arrow to assign values to variables.

For example, you can create a variable `x` and assign it the value `3` by typing this into a cell:

```
x <- 3
```

If you hit **run**, notice nothing happens. There's no output. This is because we've only defined the variable and stored the value of 3 to it. So that's all that R does. We have to be specific and tell R to show us the output by "calling" the variable.

All we do to **call** a variable is just type the name of it in a new line:

```
x <- 3
x
```

```
## [1] 3
```

Notice by typing `x` in the next line we DO get an output of `3`

You can name variable almost anything.

If you want to assign a variable a description or text instead of a number we use quotes:

```
fav_color <- "yellow"
fav_color # reminder: you need to type the name of the variable again to see it as an output
```

```
## [1] "yellow"
```

1. Create a variable named "my_feelings" and assign to it your feelings about statistics. Then have R output your feelings. Make sure that each cell begins with a comment that indicates the problem you are working on.

A Comment about Comments

Comments are very useful in programming and coding. They are parts of code that are NOT read or implemented by the program but are there to be helpful for humans–that is, for you or me to read :-)

In R, any text after a hashtag is not read by the program for that line.

For example,

```
banana <- 11 # define the variable 'banana'
banana
```

```
## [1] 11
```

If you go back up and read the examples, you'll see that I've already used comments!

I'm going to extensively use comments in the code cells to teach you important info that is not part of the code.

Writing regular text paragraphs

If you need to write regular text, then you need to create a different structure that understands plain text called "markdown."

To do this, click on the "Cell" menu, then under "Cell Type", select "Change to Markdown M". Double-click on this cell and you can now start writing.

2. Create a "markdown" cell and write a short paragraph telling me why you are taking this class.

Section 2 - Entering Data

Let's assume we have a simple data set of: 1,2,2,3,3,3,4,4,4,4,5,5,5,5,5,10.

When entering data "by hand" we use the following code:

```
my_cool_data <- c(1,2,2,3,3,3,4,4,4,4,5,5,5,5,5,10) # define and store values to my_cool_data
```

Again, it seems like nothing happened. But something did happen. The program has now stored all the numbers to a **vector** (i.e. list) variable that we named `my_cool_data`.

If we want to see our data, we have to type the name again in a new line:

```
my_cool_data <- c(1,2,2,3,3,3,4,4,4,4,5,5,5,5,5,10) # define and store values to my_cool_data
my_cool_data
```

```
##  [1]  1  2  2  3  3  3  4  4  4  4  5  5  5  5  5 10
```

Now `my_cool_data` is stored and ready to be studied.

Mean, Median, Standard Deviation, and Variance

We calculate the mean of the data set `my_cool_data` as follows:

```
mean(my_cool_data) # mean
```

```
## [1] 4.0625
```

Notice the parentheses after the mean that sandwich the data name.

This is because `mean()` is a **function**. It needs an **input** (in the example above the input is the vector `my_cool_data`) and it gives us an **output** (the mean of the data).

We can compute the mean more explicitly:

```
# explicit mean
# recall: mean = sum of values divided by # of values
num <- sum(my_cool_data) # numerator is sum of values
denom <- length(my_cool_data) # length() function tells us number of values in data set (vectors)
mean_hand <- num/denom
mean_hand
```

```
## [1] 4.0625
```

Luckily, we don't need to compute things the long way. We'll use the functions already programmed into `R`.

Similarly, we can calculate the median, standard deviation, and variance as follows:

```
median(my_cool_data) # median
```

```
## [1] 4
```

```
sd(my_cool_data) # standard deviation
```

```
## [1] 2.015564
```

```
var(my_cool_data) # variance
```

```
## [1] 4.0625
```

Five Number Summary

We can also have the *5 number summary*

```
# 5 number summary
summary(my_cool_data)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   3.000   4.000   4.062   5.000  10.000
```

Qualitative Data

```
# A dataset consisting of the 5 values: Yes, No, Yes, Yes, Maybe
acat = c("Yes", "No", "Yes", "Yes", "Maybe")
         # notice that you must use quotes to enclose categorical values
table(acat)           # make a frequency table
```

```
## acat
## Maybe    No   Yes
##     1     1     3
```

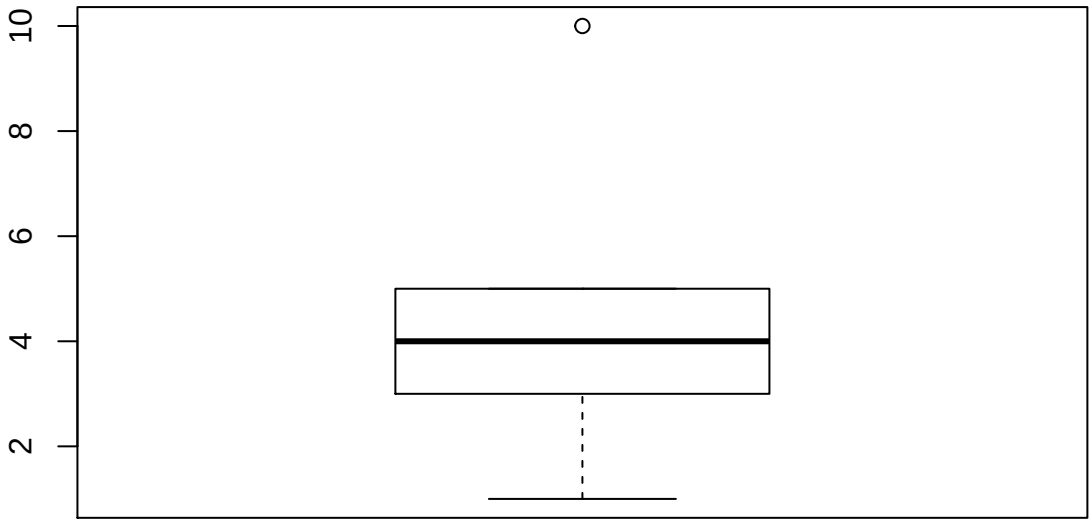
3. Create an R variable for each of the following data sets. Compute summary stats for each quantitative variable, and make a frequency table for each categorical variable.
   [a] $P = \{\text{blue, pink, blue, green, green, blue, pink, blue}\}$
   [b] $Q = \{3.9, 0, -4.6, -3.3, 2.2, 3.6, -2.9, -0.4, 0.9, 1.5\}$
   [c] $R = \{0, 1, 2, 3, a, b, c\}$

Section 3 - Data Visualization

Box Plots

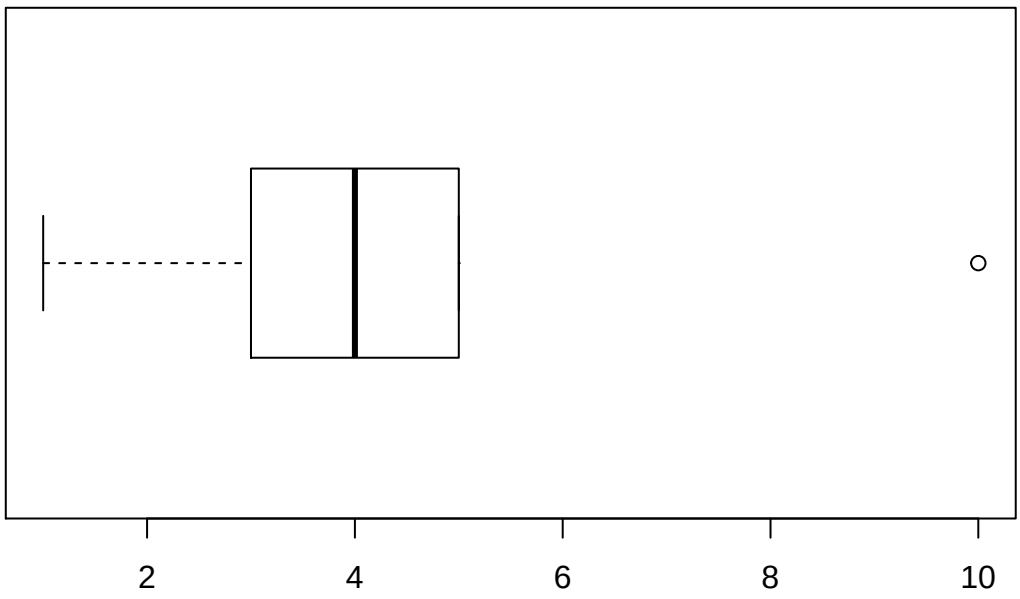
If we want to see the box plot, we use:

```
# box plot
boxplot(my_cool_data)
```



Notice that by default, the box-plot is vertical. We like it to look horizontal so we add the following code `horizontal=TRUE` and use a comma after the data name.

```
boxplot(my_cool_data, horizontal=TRUE)
```



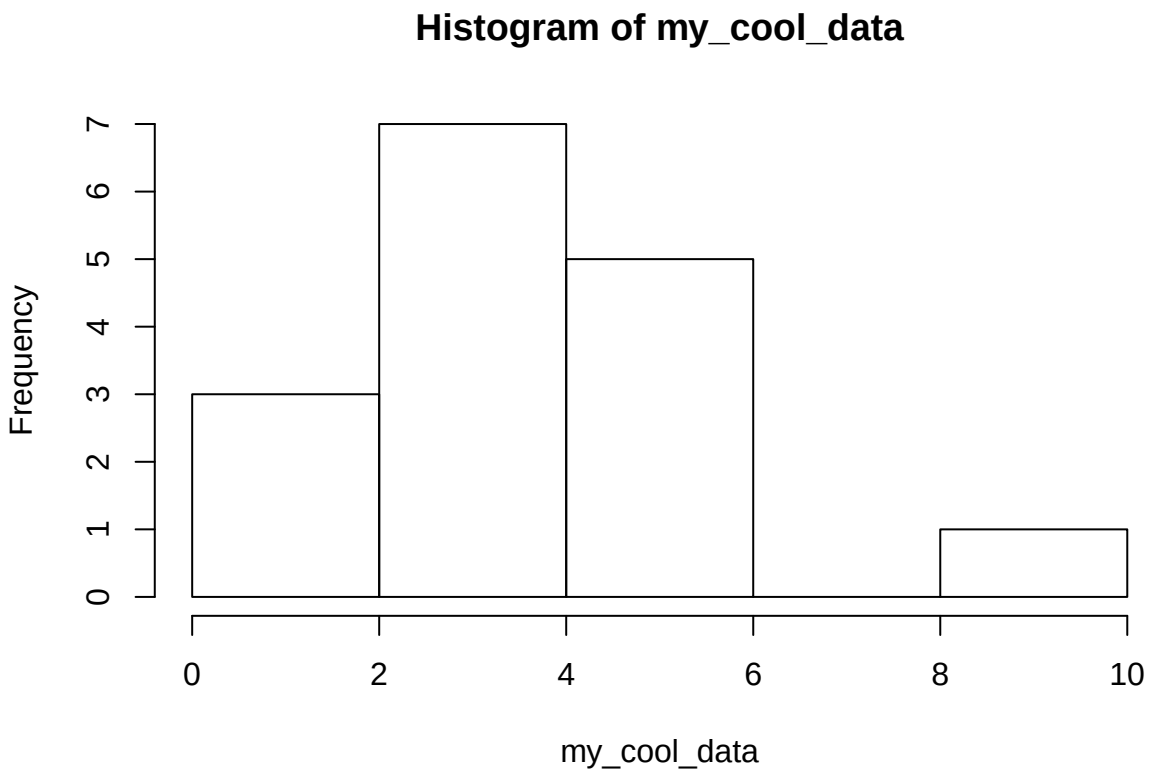
Notice the box plot looks a little different than what we discussed in class. This is because the value of 10 is considered an **outlier**.

See the textbook for how outliers are determined using the IQR (inter-quartile range).

Histograms, Dot plots, Stemplots, and Pie charts

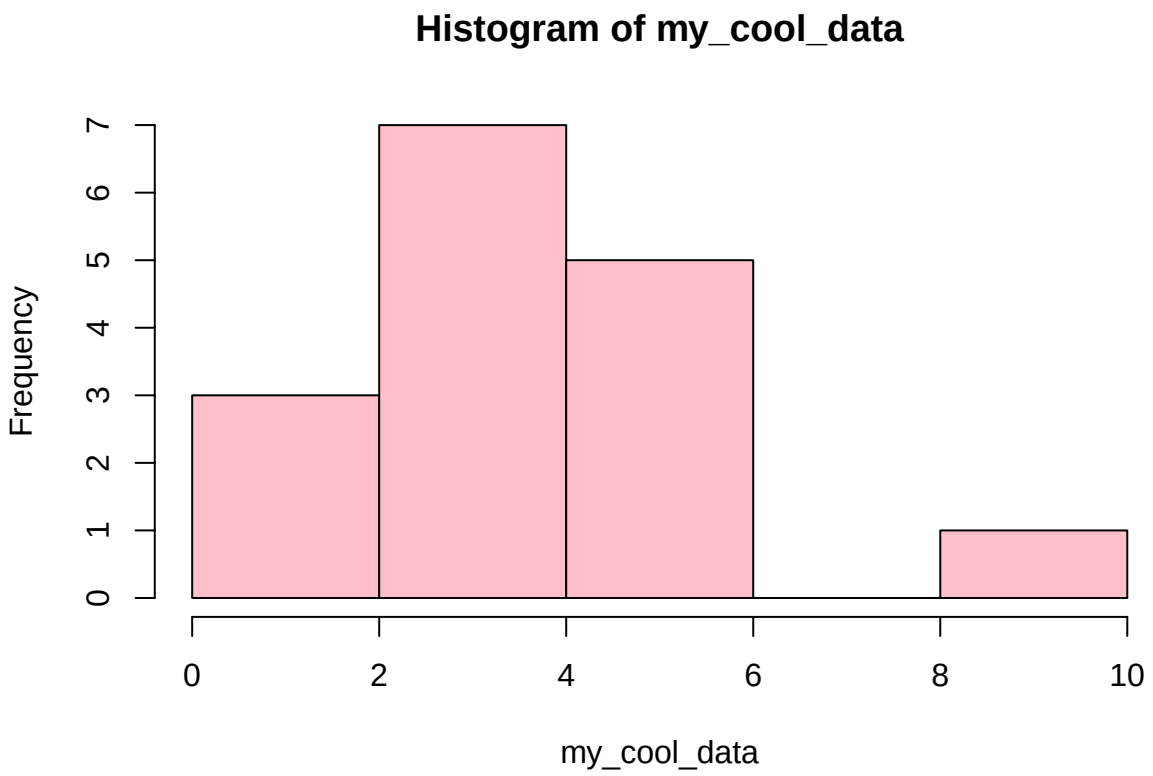
A histogram is easy to produce using:

```
# histogram
hist(my_cool_data)
```



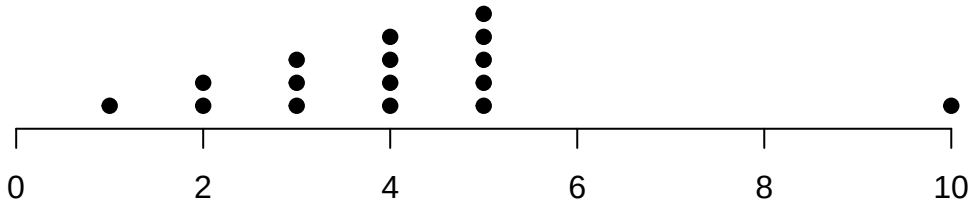
We can easily customize this to add some color:

```
# histogram
hist(my_cool_data, col="pink")
```



A dot plot is easy to produce using:

```
# Dot plots
library(plotrix) # need to use a special package
dotplot.mtb(my_cool_data) # dot plot
```



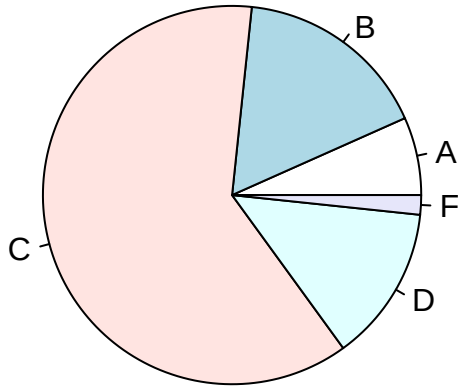
A stem-leaf plot is easy as well. To make it more interesting, we'll introduce a new data set called "ruth".

```
ruth <- c(22, 25, 34, 35, 41, 41, 46, 46, 46, 47, 49, 54, 54, 59, 60)
stem(ruth) # stem-leaf plot
```

```
##
##   The decimal point is 1 digit(s) to the right of the |
##
##   2 | 25
##   3 | 45
##   4 | 1166679
##   5 | 449
##   6 | 0
```

Finally, pie charts require a bit more to set-up.

```r
# Pie Charts
  # Notes:
  # "labels= " will create labels that we defined above
  # "main=" will create a title for the pie chart
grades <- c(4, 10 , 37, 8, 1) # the data for each "slice" of the pie
lbls <- c("A", "B", "C", "D", "F") # the labels to apply to each slice (in order)
pie(grades, labels = lbls, main="Final Exam Grades for a Statistics Course")
```



4. The table shows the movement of Walt Disney stock for 30 randomly selected trading days. "Up" means the stock price increased in value for the day, "Down" means the stock price decreased in value for the day, and "No Change" means the stock price closed at the same price it closed for the previous day.

Down

Up

Up

Down

Down

Up

Down

Up

Down

Up

Down

Up

Down

Down

Up

Up

Up

Up

Down

Down

Down

Up

Down

Up

No Change

Up

Down

Down

No Change

Down

Create a pie chart with labels and a title.

Putting it together

5. Now, consider the following data set on the "Freshman 15" for September:
   72, 97, 74, 93, 68, 59, 64, 56, 70, 58, 50, 71, 67, 56, 70, 61, 53, 92, 57, 67, 58, 49, 68, 69, 87, 81, 60, 52, 70, 63, 56, 68, 68, 54, 80, 64, 57, 63, 54, 56, 54, 73, 77, 63, 51, 59, 65, 53, 62, 55, 74, 74, 64, 64, 57, 64, 60, 64, 66, 52, 71, 55, 65, 75, 42, 74, 94
   Do the following:
   [a] Store the above data to the variable named "fresh_15". *Hint: use copy-paste (short-cut* `command+c` *(mac) or* `control+c` *(pc))*
   [b] Find the Mean, Median, Standard Deviation, and Variance.
   [c] Find the Five Number Summary and Box-plot
   [d] Find the Histogram, Dot Plot, and Stem-Leaf Plot.
   [e] Explain the similarities and differences between the Histogram and Dot Plot. **Note: Create a "markdown" cell so you can type your answer in regular text.**

Section 4 - Data Frames

How to input a spreadsheet of data

A spreadsheet or table of raw data can be created manually via keyboard input, or by reading in data files written in various standard formats. The structure used in R to represent such tables is called a **dataframe**.

Consider, for example, the following dataset:

Age

Sex

Class year

SAT score

Financial aid?

18
F
1
1014
N
20
F
3
1222
Y
17
M
1
1141
Y
17
F
1
1082
N
19
M
2
1261
Y
18
F
2
1288
N
20
F
1
1002
N
21

M

3

1078

N

We will input each column of data as a separate variable first, after which we will organize them into a dataframe.

The dataframe can be given any convenient name, e.g., "mydata"

```
# First create each column as a separate variable: we'll use the names
# "age", "sex", etc., for the names of my variables
age <- c(18, 20, 17, 17, 19, 18, 20, 21)
sex <- c("f", "f", "m", "f", "m", "f", "f", "m")
year <- c(1, 3, 1, 1, 2, 2, 1, 3)
sat_score <- c(1014, 1222, 1141, 1082, 1261, 1288, 1002, 1078)
f_aid <- c("n", "y", "y", "n", "y", "n", "n", "n")

# Next, we'll combine the variables into a dataframe that we will call "mydata"
mydata <- data.frame(age, sex, year, sat_score, f_aid)
```

Let's see how it looks:

```
# Let's print out the dataframe and see if it is what I expect
mydata
```

```
##   age sex year sat_score f_aid
## 1  18   f    1      1014     n
## 2  20   f    3      1222     y
## 3  17   m    1      1141     y
## 4  17   f    1      1082     n
## 5  19   m    2      1261     y
## 6  18   f    2      1288     n
## 7  20   f    1      1002     n
## 8  21   m    3      1078     n
```

Now we can compute summary stats, make histograms, boxplots, piecharts, etc.

Data Frames

Once a dataframe is created, it is easy to make various displays, and to compute summary statistics for variables in the dataframe.

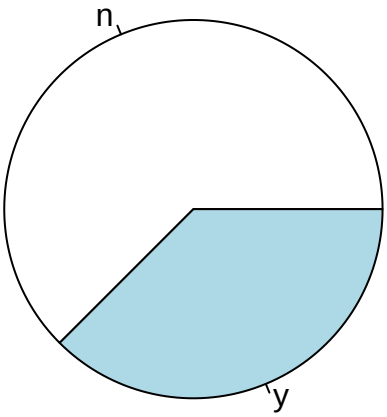
The following examples show how to do this for variables in the dataframe created above.

The basic structure is as follows: `dataframe_name$variable_name` where the dollar sign `$` is important and how we tell R to look at one specific variable.

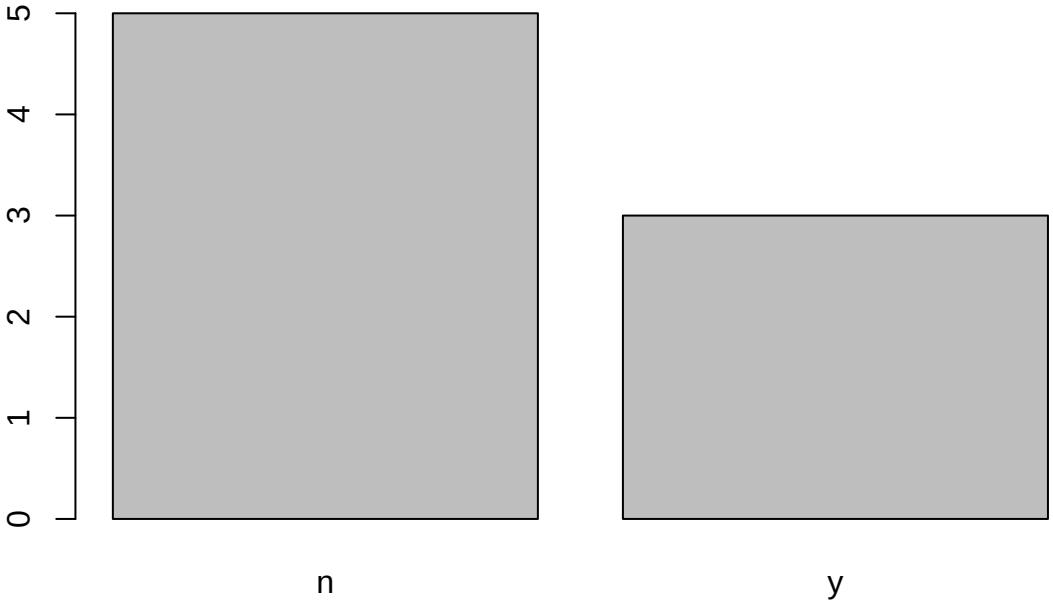
```
# creates frequency table using "f_aid" from "mydata"
table(mydata$"f_aid")
```

```
##
## n y
## 5 3
```

```
# pie chart of "f_aid"
pie(table(mydata$"f_aid"))
```



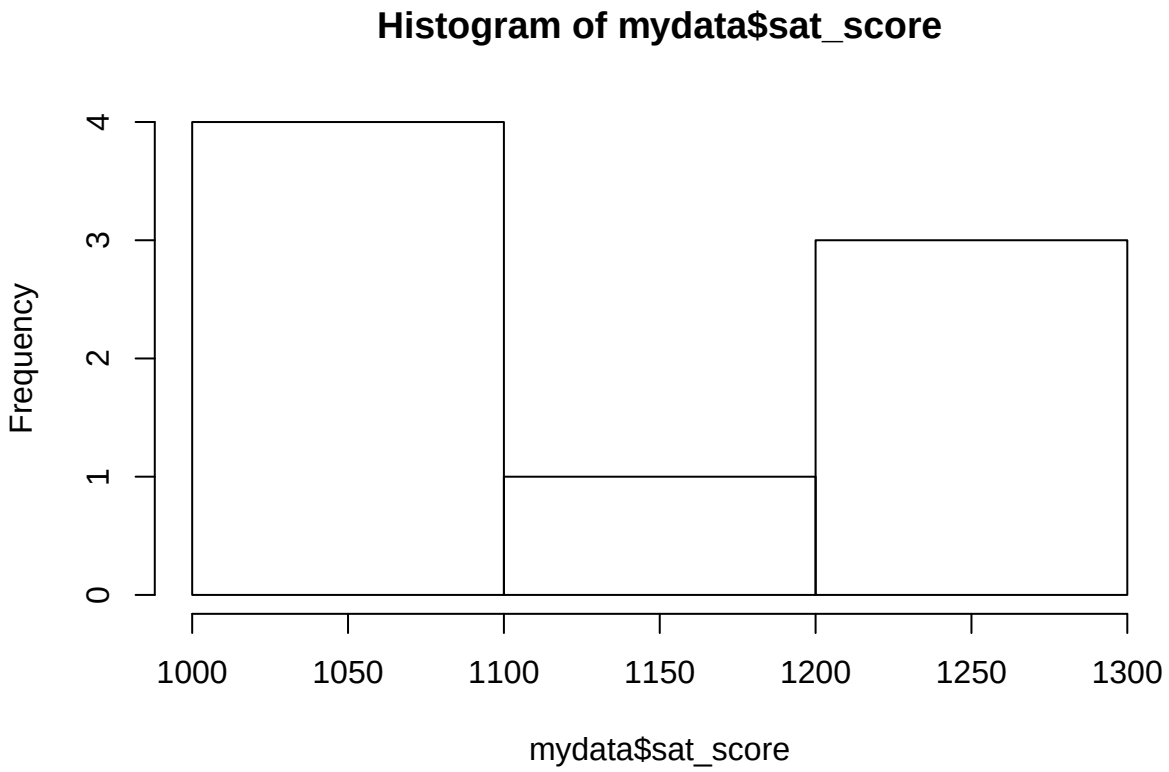

```
# bar graph of "f_aid"
barplot(table(mydata$"f_aid"))
```



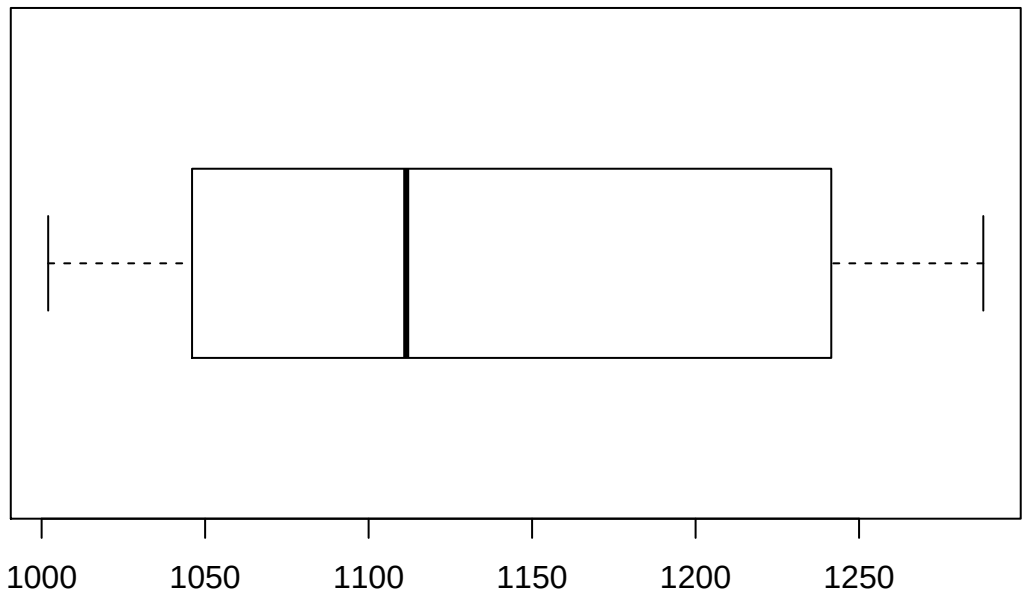

```
# 5-number summary & mean of "sat_score"
summary(mydata$"sat_score")
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1002    1062    1112    1136    1232    1288
```

```
# plot histogram
hist(mydata$"sat_score")
```

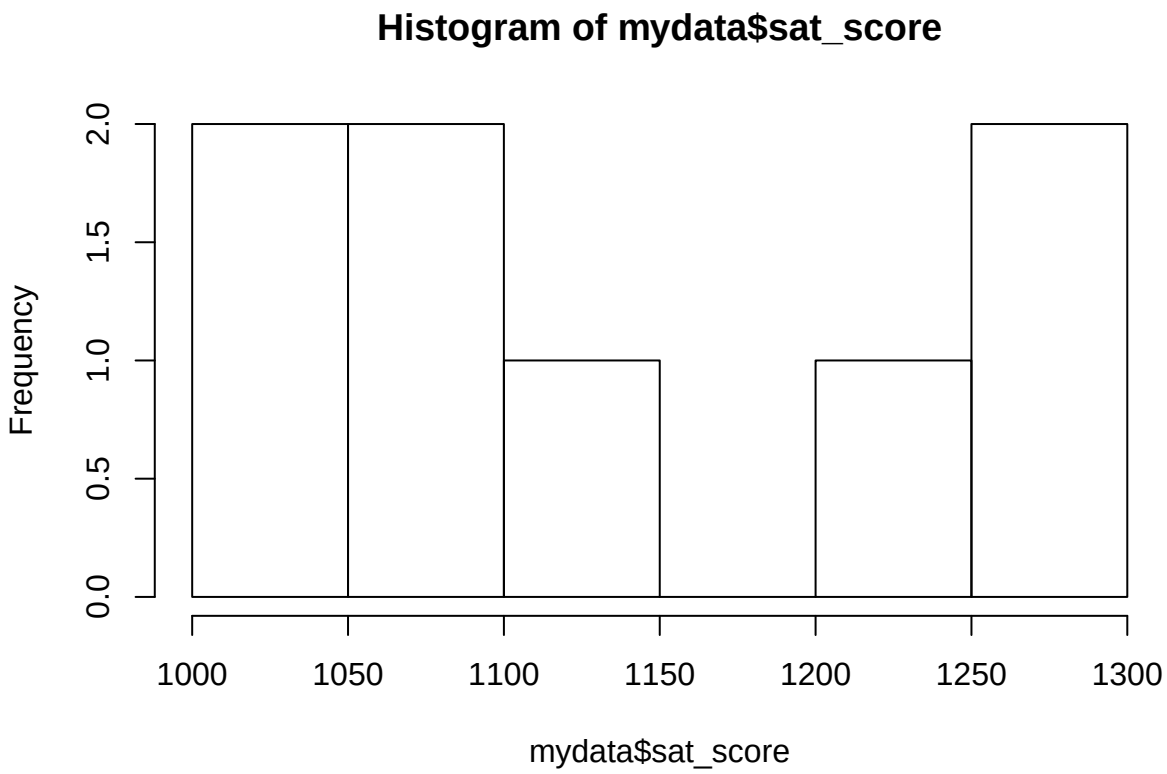



```
# box plot
boxplot(mydata$"sat_score", horizontal=TRUE)
```

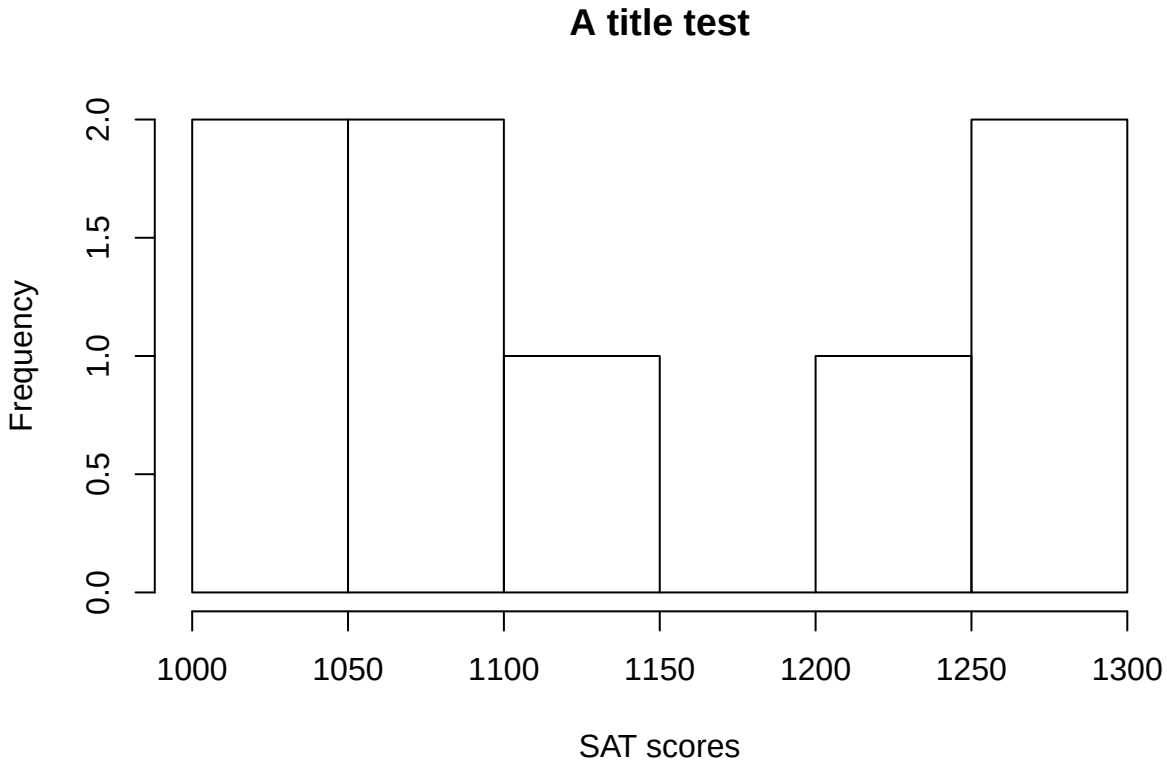


If we want to change the class widths using R in the histogram we can do it using the `breals=` command:

```
# A simple way to set the histogram scale is to specify
# the number of bars to use, like this
hist(mydata$"sat_score", breaks=6)  # histogram with 6 equal-width bars
```




```r
# It is also easy to customize the plot title, axes labels, etc., like this
hist(mydata$"sat_score", breaks=6, xlab="SAT scores", main="A title test")
```



6. The following table contains data on the employment status of a sample of college students:
   Age

Major
Employment
Work hours
19
Business
Part time
35
19
English
Part time
30
34
Business
Unemployed
0
20
Psychology
Part time
19
20
Psychology
Part time
32
21
History
Unemployed
0
21
Business
Part time
20
21
History
Part time
15
23
Psychology
Full time
36
41
Business
Full time
50
30
Physics
Unemployed
0

Create a dataframe via keyboard input to represent these data. Print your dataframe and verify that it is correct.

7. For each variable in the dataset above, make a display (or two!) and compute summary statistics.

Conclusion

This completes a basic introduction to R and also the basics of descriptive statistics.

In the next lab, we will continue descriptive statistics by studying the full data set of "freshman_15_full". Since it is tedius to type in data by "keyboard", in the next lab we will learn how to **import** a spreadsheet into R saving us lots of time!

You've done it! You've now finished Lab_1! ;-)