Introduction to R

Nelson Lim, Samantha Sangenito
Fels Institute of Government
University of Pennsylvania

Bottom Line Up Front (BLUF)

This handout introduces R. You can use R as a software like Microsoft Excel. But R can do a lot more. This handout will help you start reasoning with data.

Proper Mindset for Learning R

Learning R is not like learning how to use Microsoft Word. You are learning to compose a set of instructions to communicate with a computer; you are not learning to execute a series of standardized tasks. To accomplish this, you need to adopt a proper mindset.

Be curious

Learning R is rewarding but difficult. You will not immediately experience a tangible reward for your hard work, even though the reward is there waiting for you. When you successfully execute your R code, you experience a feeling of satisfaction that only creators like composers, poets, or writers experience.

By learning R, you are training your brain to do something new. Neurologists tell us that when we are learning a new skill, your brain is making new connections among neurons. The dense networks of neurons are associated with creativity. So be curious.

Be patient

You learn R just like you learn a foreign language. You learn by using it. You need to learn its grammar, vocabulary, and expressions. In the beginning, you will find it awkward and frustrating. Moreover, unlike communicating with a human, you are communicating with a computer. Computers are very literal and adhere to strict rules. For a computer ‘ is not the same as ’. Similarly, a computer doesn't know that MAINE and Maine are the same. A computer will not follow your instruction, until you give the instruction exactly the way it is expecting.

Because it is difficult, you may often feel like giving up. You will feel that you are wasting your time. You will come up with reasons to give up. Resist those negative thoughts. We, Samantha and I, have introduced R to students with little or no background in computer programming and statistics. In fact many of them have internalized that they are “not good at math”. But we have witnessed them learn R and do incridible projects. So don't give up; be patient.

Be persistent

It is extremely unlikely that the code you write will run without generating errors. Some of the errors are difficult to comprehend. Error messages in R are infamous for their uselessness. They are often written by advanced users for advanced users. For us they often are impossible to comprehend. Google is an invaluable resource to find solutions, because R users are very active and

happy to assist each other. But you may not find what you are looking for right away. So you need to be persistent.

Resources

There are many useful resources to learn R. In fact, there are too many resources that they can be overwhelming. So it is wise to invest in reliable references. Here are three references that you should acquire for your learning journey.

1. *R in Action: Data Analysis and Graphics with R* by Robert I. Kabacoff. If you are going to buy only one book to learn R, get this book. Even though it doesn't cover the latest features of R, it has everything you need to get you started.
2. *R for Data Science* by Hadley Wickham and Garrett Grolemund. This book is written by one of the most influential R programmers, Hadley Wickham. He has developed many popular R packages. In this book, he introduces a new paradigm for R users. Therefore, this book is a unique and innovative book.
3. *Quantitative Social Science: An Introduction* by Kosuke Imai. This book offers a rigorous introduction to the latest thinking on how to reason with data. It also teaches you how to use R.

Important House Keeping Items

This handout assumes that you have installed the latest version of R and R Studio. If you have not installed those software packages, please do so now. You can download the packages by googling *R* and *R Studio*.

If you need further assistance, you can watch our tutorial videos showing how to install these software packages. Samantha recorded a video. You can also get instructional videos from *youtube*. Just search by entering key words: *how to install R*.

R is command-driven

Most of the software packages you have used are menu-based. R is command-driven. You have to type a series of instructions (commands) in text, which is called a script, and ask R to execute it. Samantha recorded a video showing how you compose a script and execute the commands in R Studio.

Installing and Loading Packages

The standard (or core) R you installed already is a very powerful software. You can expand its power by using packages that are developed and freely shared by R users. R studio provides easy ways to install packages. You can also install packages using `install.packages()`. For example, `install.packages("ggplot2")`. You will only need to install a package once. Once a package is installed, you can load a package using `library()`. For example, `library(ggplot2)`.

You will get an error message if you try to use a command (or a function) from a package without loading the associated package first. **This is one of the most common errors you will get as a new user of R. Make sure to load the packages you need at the front of your script.**

Setting a Working Directory

This is another one of the most common errors. You need to master this concept to avoid unnessary heart ache.

Your working directory is a location on your computer where R will look for your data and scripts. You will need to tell R what your working directory is everytime you open RStudio, otherwise R will just guess. **You don't want R to guess.**

To find out where R has currently set as your working directory, you write this code: `getwd()`.

When I **run** the above code, R shows me what my current working directory is.

```
getwd()
```

```
## [1] "C:/Users/nellim/Box Sync/Introduction to R"
```

Your Turn: Run the above code in your R console to find out your working directory.

Check out the pathway that prints out in your console. This pathway is *unique* to your computer. Notice that R uses forward slash.

It is a best practice to create a folder for your R data and tell R to use that folder every time you open R using `setwd()`.

Here's how I set the pathway that I would like to use as the working directory.

```
setwd("C:/Users/nellim/Box Sync/Introduction to R")
```

Your turn: Can you edit my setwd() code to set your own working directory?

Types of data in R

You can use R to work with many types of data. The most common type of data you will work with is something called a dataframe. This is very similar to an Excel spreadsheet in appearance.

Creating a dataframe

There are two ways to create a dataframe in R, you can either make one yourself or read in a premade data file. Most of the time we will read in an external file. This tutorial will show you how to do both.

Making a dataframe

Let's make one so you can see.

```
# beatles is the name of the dataframe
beatles <- data.frame(
  name = c("John", "Paul", "George", "Ringo"),
  birth = c(1940, 1942, 1943, 1940),
  instrument = c("guitar", "bass", "guitar", "drums")
)
```

Note the sentence starts with #. It is a comment. When you start a sentence with #, R ignores it. You should write a lot of comments in your script. This is an essential habit of a good analyst.

Note also how we *assign* the dataframe to be `beatles` using `<-`. This symbol is pronounced "gets." Note also that we created a dataframe with 3 columns: name, birth, and instrument, then populated those columns with information inside the function: `c()`. Now you can ask R to get this dataframe using that name `beatles`.

Columns in the dataframe are called *variables*; rows in the dataframe are called *observations*.

You can see what is in your dataframe with View()

```
View(beatles)
```

You can also *print* what is in your dataframe by executing the name of the dataframe.

```
beatles
```

```
##     name birth instrument
## 1   John  1940     guitar
## 2   Paul  1942       bass
## 3 George  1943     guitar
## 4  Ringo  1940      drums
```

You should only use this option to view the dataframe, when it is relatively small. You don't want to do this for large dataframe. For a large dataframe, you can print the first top 10 observations of the dataframe by executing `head(mydata, n=10)`. `mydata` is a generic name of the dataframe.

Your turn: Make your own dataframe that contains information on the band Aerosmith from the following data on instrument, birth year, and marital status for each member:

```
Steven Tyler, lead vocals, born 1948, single
Joe Perry, guitar, born 1950, married
Tom Hamilton,  bass, 1951, married
Joey Kramer, drums, 1950, married
Brad Whitford, guitar, born 1952, married
```

Saving and loading a dataframe

Now that we have created a data frame, we can save it.

```
# the format for the command to save a data frame in a file is:
# save(xxx, file = "yyy.RData")
# xxx = the object name
# yyy = the file name
save(beatles, file = "beatles.RData")
```

Once you save a file, you can load it when you need it.

```
load("beatles.RData")
```

Managing data frames

Now that you have a data frame, you can select different elements of the data frame to examine and change. Each element of the data frame has an "address". **The "address" structure is one of most important concepts to master as a new user of R. You may find it confusing at first.**

Table 1: "Address" of each element in a data frame

	[,1]	[,2]	[,3]
[1,]	[1,1]	[1,2]	[1,3]
[2,]	[2,1]	[2,2]	[2,3]
[3,]	[3,1]	[3,2]	[3,3]

Using these "addresses" to select the elements that you are interested.

```
# you can select a column of a data frame
# for example, select names of the Beatles
beatles$name
```

```
## [1] John   Paul   George Ringo
## Levels: George John Paul Ringo
```

```
# you can do the same task differently
beatles[,1]
```

```
## [1] John   Paul   George Ringo
## Levels: George John Paul Ringo
```

```
# note how the variable "name", the first column of the data frame,
# beatles, is selected without using the variable name

# you can also select a particular member of the Beatles
beatles[1,]
```

```
##   name birth instrument
## 1 John  1940     guitar
```

```
beatles[beatles$name=="John",]
```

```
##   name birth instrument
## 1 John  1940     guitar
```

Your turn:

(1) Print the names of Aerosmith's members

(2) Print information about a member of Aerosmith

Reading in a data file

In most cases, you are not going to manually enter the data like you did above. The data you need are already stored in a file. The function for reading in a data file will differ depending on the type of file you are reading. In this example, we will use a comma-separated values (csv). You can import a csv file using `read.csv`.

Here we are reading in the csv file, `Philly_schools.csv`. This dataset contains information on public schools in Philadelphia. Each row represents 1 school.

```
# importing data
library(foreign)
schools <- read.csv("Philly_schools.csv", stringsAsFactors = FALSE)
# we print top 3 observations to take a look
head(schools, n=3)
```

```
##   School_code Attendance Enrollment New_student Withdrawals
## 1        1010      83.80     1155.0       177.0       232.0
## 2        1020      80.62      854.4       146.0       180.8
## 3        1030      89.82      419.0         8.8        38.4
##   African_American White Asian Latino Other Pacific_Islander
## 1            91.94  1.70  4.30   1.36  0.66                0
## 2            97.50  0.56  0.62   0.78  0.52                0
## 3            90.32  3.56  1.14   4.30  0.74                0
##   Low_income_family                SCHOOL_NAME_1
## 1             85.90      JOHN BARTRAM HIGH SCHOOL
## 2             87.70 WEST PHILADELPHIA HIGH SCHOOL
## 3             85.88     HIGH SCHOOL OF THE FUTURE
##                   SCHOOL_NAME_2            ADDRESS SCHOOL_ZIP ZIP_PLUS_4
## 1            BARTRAM, JOHN HIGH   2401 S. 67TH ST.      19142         NA
## 2 WEST PHILADELPHIA HIGH SCHOOL  4901 CHESTNUT ST.      19139         NA
## 3     HIGH SCHOOL OF THE FUTURE 4021 PARKSIDE AVE.      19104         NA
##           CITY STATE_CD PHONE_NUMBER SCH_START_GRADE SCH_TERM_GRADE
## 1 PHILADELPHIA       PA   2154926450               9             12
```

```
## 2 PHILADELPHIA       PA   2154712902              9             12
## 3 PHILADELPHIA       PA   2158235502              9             12
##                             HPADDR SCHOOL_LEVEL_NAME Drugs Morals
## 1                                        HIGH SCHOOL 10.00   1.00
## 2 www.philasd.org/schools/westphila       HIGH SCHOOL  5.75   0.50
## 3      www.philasd.org/schools/hsof       HIGH SCHOOL  0.75   0.75
##   Assaults Weapons Thefts Total_suspensions One_suspension Two_suspensions
## 1    31.00   10.50   3.25            176.25          137.0           28.25
## 2    25.00    4.75   3.00            249.50          150.0           58.50
## 3    14.25    1.00   6.00             80.75           59.5           15.25
##   Three_suspensions Three_plus_suspensions Teacher_attendance
## 1              5.25                   5.75              93.95
## 2             24.00                  17.00              92.75
## 3              4.50                   1.50              95.45
##   Special_education Gifted_education English_second_language
## 1             21.08             0.82                    8.96
## 2             20.06             0.68                    2.18
## 3             13.54             2.88                    1.60
##   Average_salary
## 1       72345.82
## 2       63536.71
## 3       63622.83
```

Note that we use a command `read.csv` from a package called `foreign`. We also turn on an option `stringsAsFactors = FALSE`. This code tells R not to convert character variables into factors. Just make a mental note of this for now. We talk more about this below.

Working with Variables

Conceptually there are two types of variables: categorical and continuous. (We are over simplifying for this handout.) Categorical variables are associated with, you guess it, categories like gender, race, college major, academic degrees, ranks. Continous variables are associated with measures with many (or infinite) values like age, income, wage, or body weight.

There are three main classifications for variables in R:character, factor, and numeric. The first two classes are associated with categorical data. There are two types of categorical data: nominal and ordinal. We will discuss the types later.

Table 2: Relationship between conceptual types and classification of variable

Types	Sub-type	Classification
Categorical data	Nominal	Character
	Ordinal	
	Nominal	Factor
	Orinal	
Continuous		Numeric

Character variables

Character variables are just text. In the schools dataframe, the SCHOOL_NAME_1 variable is a character. It is just the name of the school.

We can check to see if a variable is a character by executing this code: `class(schools$SCHOOL_NAME_1)`.

```
# what is the "class" of SCHOOL_NAME_1
class(schools$SCHOOL_NAME_1)
```

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

Here we are asking R to give us the class type of the column for school name inside the dataframe, schools. This `dataframe$columnname` syntax will become quite familiar to you in time!

Alternative way to select a variable from a data frame

The attach() function allows us to simplify our code by removing the need to refer to the name of the dataframe

For example: ‘

```
attach(schools)
class(schools$SCHOOL_NAME_1)
```

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

Notice that the function is working without the dataframe$column name syntax.

Factor Variables

Factors are a special type of variable. They will appear in a dataframe as either character or numeric, but they have some additional information associated with them that is hidden to us, but that R uses. Factor levels control how variables are graphed, displayed in tables, and matter for interpreting regressions. We'll discuss factors in greater detail later on.

`SCHOOL_LEVEL_NAME` is currently a character variable.

```
# what is the class of SCHOOL_LEVEL_NAME
class(schools$SCHOOL_LEVEL_NAME)
```

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

Now we can change this variable to a factor variable.

```
# change a character variable into a factor
schools$SCHOOL_LEVEL_NAME <- as.factor(schools$SCHOOL_LEVEL_NAME)
```

Check that this variable is now a factor

```
class(schools$SCHOOL_LEVEL_NAME)
```

```
## [1] "factor"
```

You can examine the levels for a factor variable.

```
levels(schools$SCHOOL_LEVEL_NAME)
```

```
## [1] "CAREER AND TECHNICAL HIGH SCHL" "ELEMENTARY SCHOOL"
## [3] "HIGH SCHOOL"                    "MIDDLE SCHOOL"
```

We'll see how R uses these levels later on in this script.

Numeric Variables

Numeric variables are just numbers. The variable, `Average_salary`, is the average salary of teachers in each school.

```
# what is the class of Average_salary
class(schools$Average_salary)
```

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

Basic Data Manipulation

To get us familiar with working in R, we will learn some basic functions commonly applied to character and numeric variables

Working with character data

Print an alphabetized list with `sort()`

```
listofschools <- sort(schools$SCHOOL_NAME_1)
# the list is too long
# we will only print the names of the first 10 schools
head(listofschools, n=10)
```

```
##  [1] "A. D. HARRINGTON SCHOOL"       "A. L. FITZPATRICK SCHOOL"
##  [3] "A.S. JENKS ACADEMICS PLUS SCH" "ABIGAIL VARE SCHOOL"
##  [5] "ABRAHAM LINCOLN HIGH"          "ACADEMY AT PALUMBO"
##  [7] "ADD B ANDERSON SCHOOL"         "ALAIN LOCKE SCHOOL"
##  [9] "ALBERT M. GREENFIELD SCHOOL"   "ALEXANDER ADAIRE SCHOOL"
```

The table function provides us a count of character variables

```
table(schools$SCHOOL_LEVEL_NAME)
```

```
##
## CAREER AND TECHNICAL HIGH SCHL              ELEMENTARY SCHOOL
##                              5                            147
##                    HIGH SCHOOL                   MIDDLE SCHOOL
##                             40                             17
```

Working with numeric data

With numeric data, you can compute simple summary (or descriptive) statistics using `mean()` and `summary`.

```
# mean value of number of suspensions in Philly public schools
mean(schools$Total_suspensions)
```

```
## [1] 75.26555
```

```
# alternative way to get similar summary statistics
summary(schools$Total_suspensions)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   25.75   45.75   75.27   81.50  667.00
```

Creating new variables using arithmetic operators

```
# creating a new variable, NewEnrollmentProp, using two existing variables: New_student and Enro
schools$NewEnrollmentProp <- (schools$New_student / schools$Enrollment)*100
summary(schools$NewEnrollmentProp)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##  0.08123  7.50800 11.29000 10.69000 14.22000 38.16000
```

Aggregating numeric variables by categories of character variable

```
class(schools$Total_suspensions)
```

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

```
class(schools$SCHOOL_LEVEL_NAME)
```

```
## [1] "factor"
```

```r
#compute average number of total suspension by school level
aggregate(schools$Total_suspensions, by=list(schools$SCHOOL_LEVEL_NAME), FUN=mean)
```

```
##                          Group.1         x
## 1 CAREER AND TECHNICAL HIGH SCHL 102.15000
## 2              ELEMENTARY SCHOOL  48.58673
## 3                    HIGH SCHOOL 161.10625
## 4                  MIDDLE SCHOOL  96.07353
```

Your turn:

(1) Look in the dataframe schools, and find two examples of categorical and continuous variables.

(2) Find the average teacher salary in Philadelphia

(3) Find the average teacher salary in Philadelphia by school level

---

Graphing different Types of variables

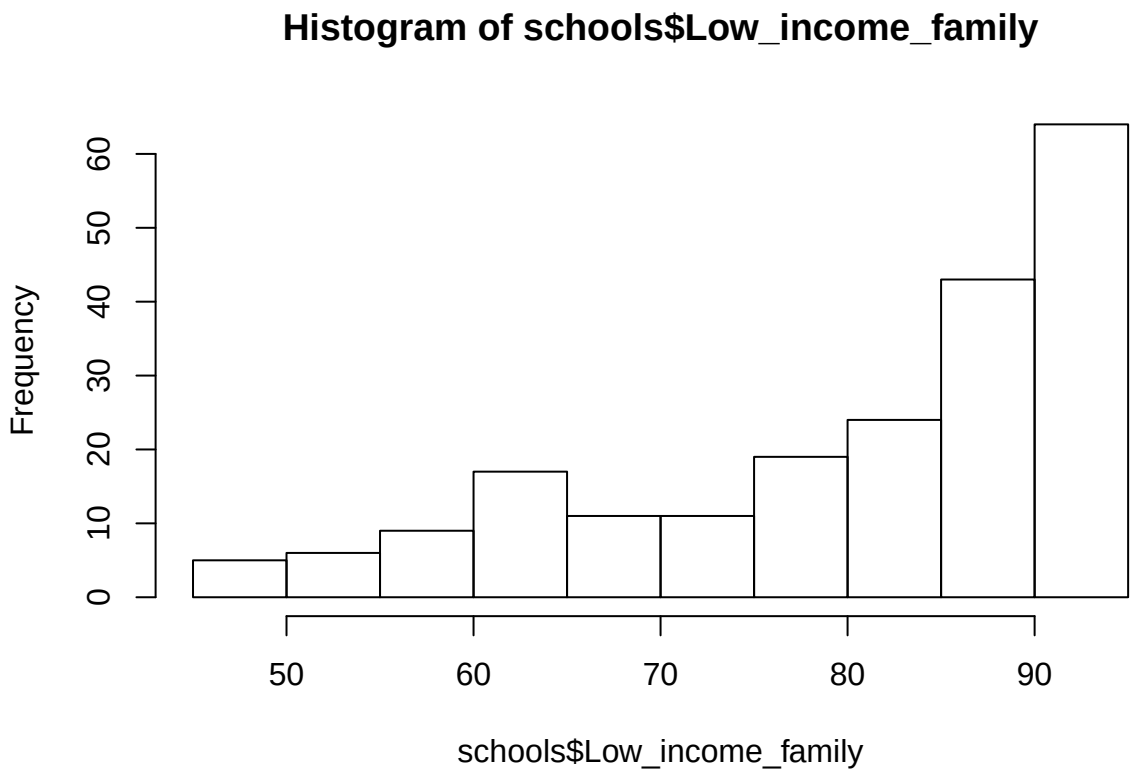
R is wellknown for its capability to produce effective vitualization of the data. We will show you how to use core graphical commands in R. In the Spring, we will show you how to use *ggplot* and other packages that are more powerful.

Displaying the distribution of continuous variables

Continuous and Categorical variables naturally lend themselves to different types of graphs.

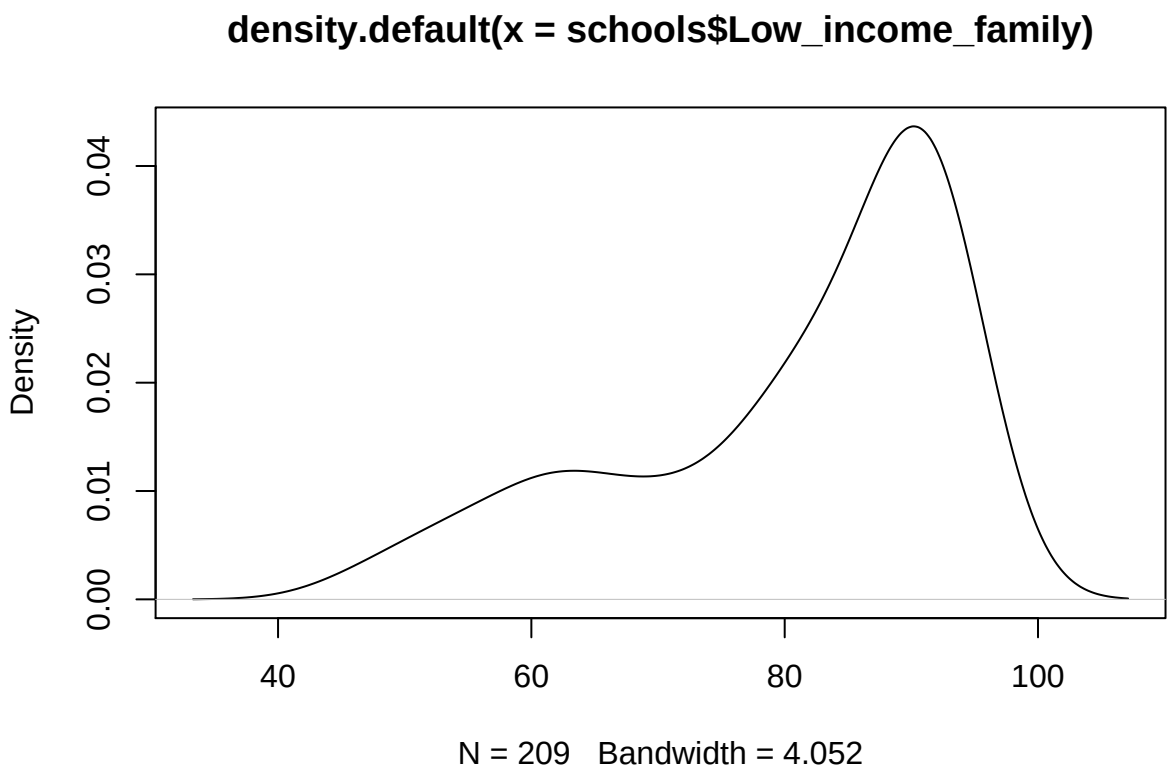
Histrogram: Base R offers a frequency chart of numeric variables

```r
hist(schools$Low_income_family)
```



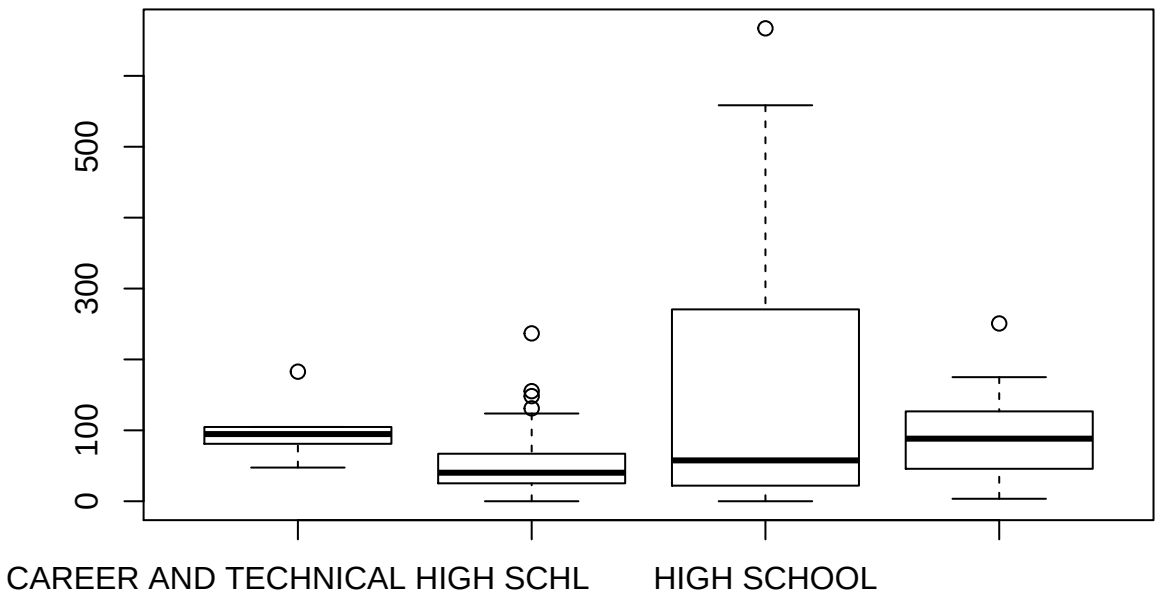
A Density plot displays similar information

```
plot(density(schools$Low_income_family))
```



Boxplots are useful for understanding the spread of the data

```
boxplot(schools$Total_suspensions ~ schools$SCHOOL_LEVEL_NAME)
```



Notice anything funny about this graph? What is causing it?

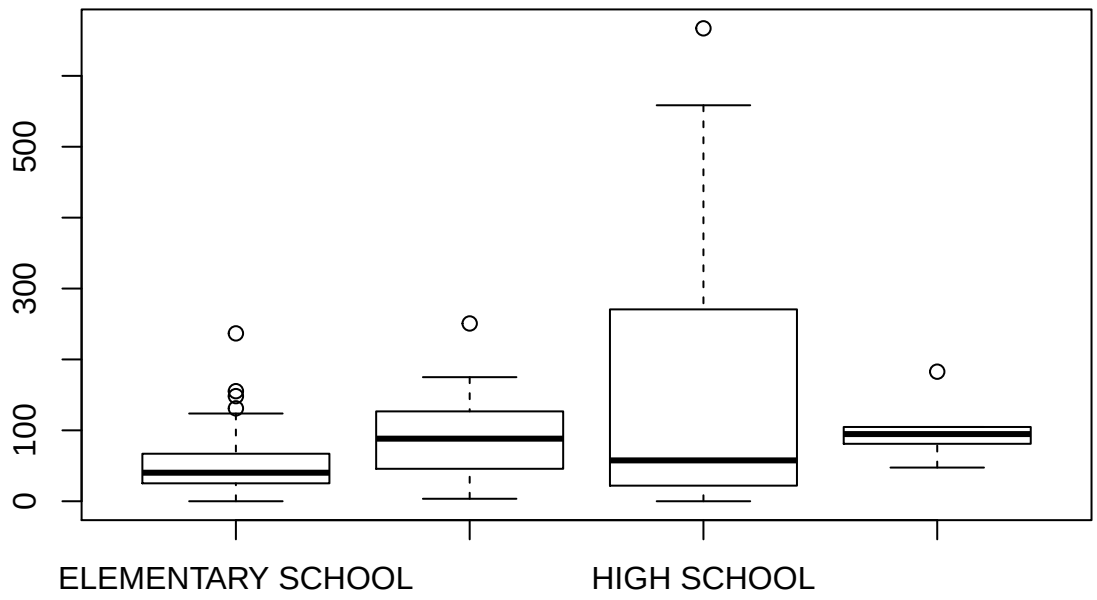
To change the levels:

```
schools$SCHOOL_LEVEL_NAME<-factor(schools$SCHOOL_LEVEL_NAME, levels = c("ELEMENTARY SCHOOL", "MI
levels(schools$SCHOOL_LEVEL_NAME)
```

```
## [1] "ELEMENTARY SCHOOL"              "MIDDLE SCHOOL"
## [3] "HIGH SCHOOL"                    "CAREER AND TECHNICAL HIGH SCHL"
```

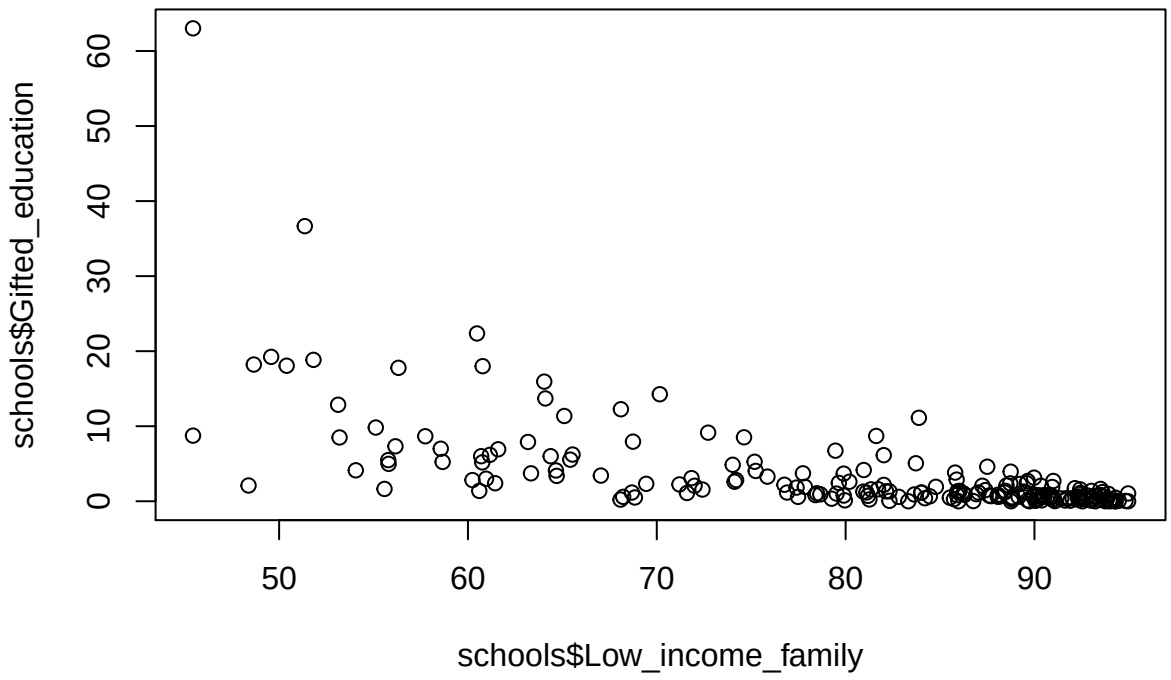
Now everything looks right!

```
boxplot(schools$Total_suspensions ~ schools$SCHOOL_LEVEL_NAME)
```

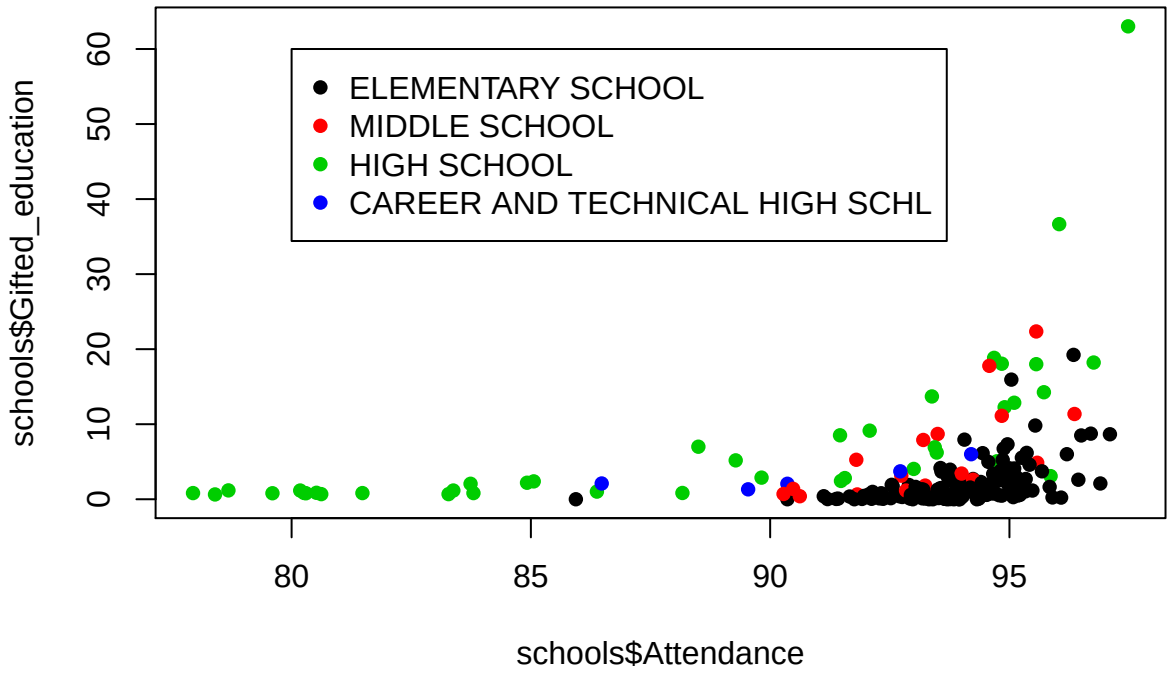


Scatter plots to display associations among continuous variables

```
plot(schools$Low_income_family, schools$Gifted_education)
```




```
plot(schools$Attendance, schools$Gifted_education, col = schools$SCHOOL_LEVEL_NAME, pch=16)
legend(x=80, y=60, legend=levels(schools$SCHOOL_LEVEL_NAME), col=c(1:4), pch=16)
```



_____

Your turn: Find something interesting to you in the data and plot it!

_____

_____

Your turn: Now you will analyze a new dataset using your new skills!

1. Read in the csv file, "acs_orientation.csv". This file contains information on the gender, years of education, race, and income of 500 residents of Philadelphia.

2. Find the number of males and females in this dataset

3. Find the average salary of this dataset

4. Find the average salary of this dataset by gender

5. Find the average salary of this dataset by race

6. Plot the relationship between years of education and income