**BIA 6311: Introduction to R for Data Analysis**

Xuan Pham, PhD

Helzberg School of Management

College of Business, Influence, and Information Analysis

Rockhurst University

Last Updated: July 31, 2019

# Workshop Road Map

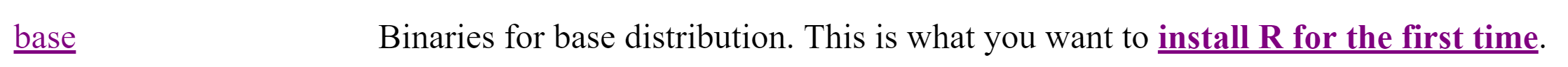
1. Getting Started
2. Basics of R
3. Data Types
4. Data Frames
5. Exploratory Data Analysis
6. R Packages
7. Charts
8. R Markdown
9. Resources

# GETTING STARTED

## DOWNLOAD AND INSTALL R AND R STUDIO

Please go here and download R: <http://rweb.crmda.ku.edu/cran/>

Choose your operating system (Windows or Mac) and then choose base distribution.



Go here to download R Studio: <https://www.rstudio.com/products/rstudio/download/#download>. Choose the zip file based on your operating system.

When you are done, click on the R Studio icon to launch it. The version of R that you downloaded is automatically attached to R Studio.

Alternatively, you can watch this video to follow along with the download and installation process: <https://www.youtube.com/watch?v=ii5SPVqWCgI>

## BASE R and PACKAGES

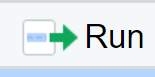
When you download R, it comes with the functionalities of what is called “base R.” Everyone has base R on his/her machine. If you want R to do something beyond its basic functionalities, you need to install and load packages.

R has thousands of packages. Each package is designed to do a different task: from plotting charts and graphs to performing linear regressions, from creating decision trees to teaching you the R programming language. To see the list of all the available R packages maintained by CRAN (the Comprehensive R Archive Network), click here: <https://cran.r-project.org/web/packages/available_packages_by_name.html>.

## RUNNING R SCRIPTS

Open a new R script: File 🡺 New File 🡺 R Script

To run the a line from your R script, put your mouse at the end of the line and do Ctrl + Enter (Windows) or Cmd + Enter (Mac).

Alternatively, you can also use the “Run” icon  .

## GETTING HELP

R has great built in help files. I highly recommend that you get familiar with reading the help files. It takes time to learn to read R documentation. Be patient.

?help Pull up the entire help file in the bottom right screen of R Studio.

?mean Pull up the specific help document about arithmetic mean

?plot Pull up the specific help document about Generic X-Y plotting

# BASICS OF R

## R AS A CALCULATOR

In its most basic form, R is a calculator. Try doing some calculations in the R Console window (bottom left in your R Studio environment).

Here are the basic mathematical signs:   
1 + 1                  add

2 -1                    subtract

4/2                     divide

2\*2                     multiply

2\*\*2   or 2^2      exponent

sqrt(4)               square root

## ASSIGNING A VARIABLE

Use <- sign to assign a variable. I do not recommend using =.

x <- 1    Here we are creating a variable called x, and x has a value of 1.

y <- 2    Here we are creating a variable called y, and y has a value of 2.

x <- 3    Here we are reassigning the variable x, and it now has a value of 3.

y <- x    Here we are reassigning the variable y, and it now has the value of the variable x (which is 3 since we just reassigned it above).

To see the value for a variable, simply type in the variable name and press enter.

Here are the rules for assigning variables in R:

1. Can contain any alphanumeric characters, periods (.), and underscores (\_)

2. Cannot start with a number or an underscore

3. Variables are case sensitive

Students often ask me for shortcut keys for the assignment operator. Here they are:

Windows: Alt + - (Alt key and the minus key)

Mac: Option + - (Option key and the minus key)

## REMOVING A VARIABLE

Removing a variable from the list of objects in R may free up memory space. There are two ways to remove a variable. rm is just a shortcut for remove.

remove(*variable name*)

rm(*variable name*)

## COMMENTS

Comments are an important component in writing any program. Comments allow you to explain what you are doing to yourself and other people who may use your program. R does not “act” on the comments you write. You specify an upcoming comment to R by using the # sign.

Here are some examples:

#Today is Saturday!

## FUNCTIONS

Functions are commands that do something useful in R. Base R has a set of functions. Each R package has its own functions. You can define your own function too, but that is an advanced topic.

date() #date and time function

c(1,2,3) #combine function

a <- c(1,2,3)

mean(a)

You can nest functions inside each other too.

mean(c(1,2,3))

# DATA TYPES

## CHECKING DATA TYPES

R stores many different data types. We will look at some common data types.

To check the date type of any variable, use the **class()** function.

        a. Positive and negative numbers with and without decimals

        b. Integers (i.e. whole numbers only)

        c. Zero

Examples:

> m <- 5

> class(m)

[1] "numeric"

> d <- 8.000000000001

> class(d)

[1] "numeric"

> e <- 0

> class(e)

[1] "numeric"

1. Integers: whole numbers

Example:

> j <- as.integer(4)

> class(j)

[1] "integer"

1. Character/string: Any combination of alphanumeric and special characters [ **~ ! @ # $ % ^ & \*( )** ].

Examples:

> a <- "Hello World"

> class(a)

[1] "character"

> b <- "!@#$%^&\*()"

> class(b)

[1] "character"

1. Date

    a. Date: year-month-day format

        b. POSIXct: year-month-day-time format. The time format is based on 24 hour clock (i.e. military time).

All dates in R are actually numeric values of seconds from the date January 1, 1970.

Example:

> c <- as.Date("2015-10-07")

> c

[1] "2015-10-07"

> class(c)

[1] "Date"

1. Logical: TRUE/FALSE

1 == 2         Does 1 equal 2?

1 != 2          Is 1 not equal 2?

1 < 2           Is 1 less than 2?

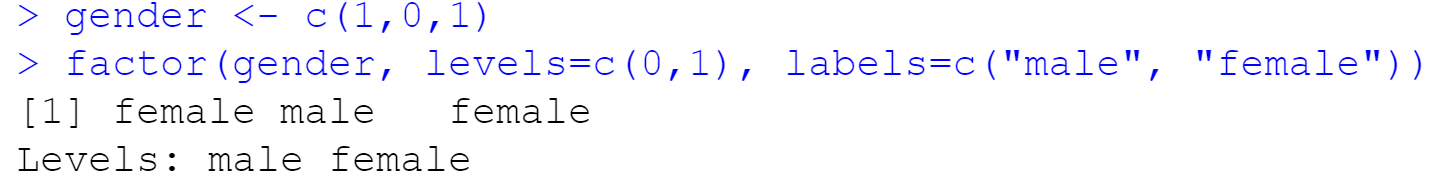
1 > 2           Is 1 greater than 2?

1 <= 2         Is 1 less than or equal to 2?

1 >= 2         Is 1 greater than or equal to 2?

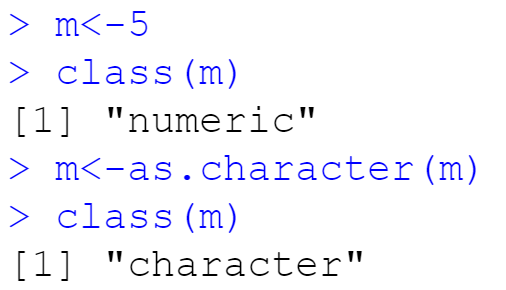
1. Factors: a set of numeric codes with character-valued levels. Factors are equivalent to

categorical variables.



## SET/CONVERT DATA TYPE

To set a data type for your variable OR to convert one data type to another type, use **as.** keyword.



One important rule to remember with data type conversion:

1. To convert a numeric into factor, it is a two-step process:
   1. Convert numeric into character
   2. Then convert character into factor

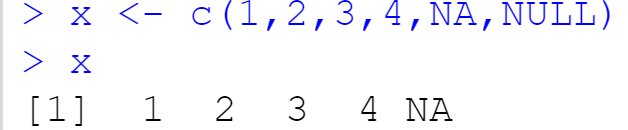
## MISSING DATA (NA & NULL)

R treats missing data in two different ways: NA and NULL. NA is equivalent to "missing" in R. NULL is equivalent to "nothing."

## VECTORS

A vector is a collection of elements that are of the same data type. In another word, each variable you have created so far is a vector.

To create a vector, we use the combine function, c().



y <- seq(1,10,1) #The sequence function seq() can also create a vector. Create a vector of values from 1 to 10 by an increment of 1.

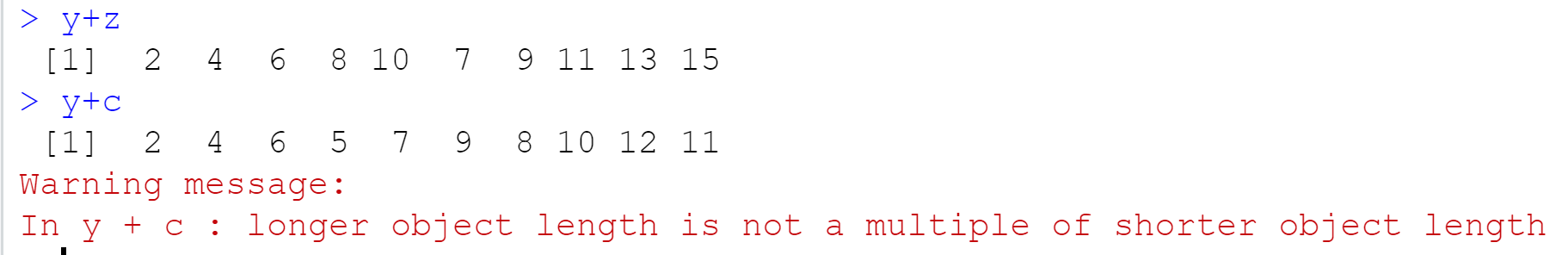
z <- seq(1,5,1)

c <- seq(1,3,1)

You can apply mathematical operations to vectors:

y + 1                            #Add 1 to each value in the x vector (specified above)

*R uses the principle of recycling in doing vector operations*. When two vectors of unequal lengths are operated together, the elements of the shorter vector are repeated until they have been matched up with every element of the longer vector. This usually works out when the longer vector is a multiple of the shorter vector. If the longer vector is not a multiple of the shorter vector, then R will issue a warning. In any case, be careful when you are trying to do mathematical calculations with two variables!

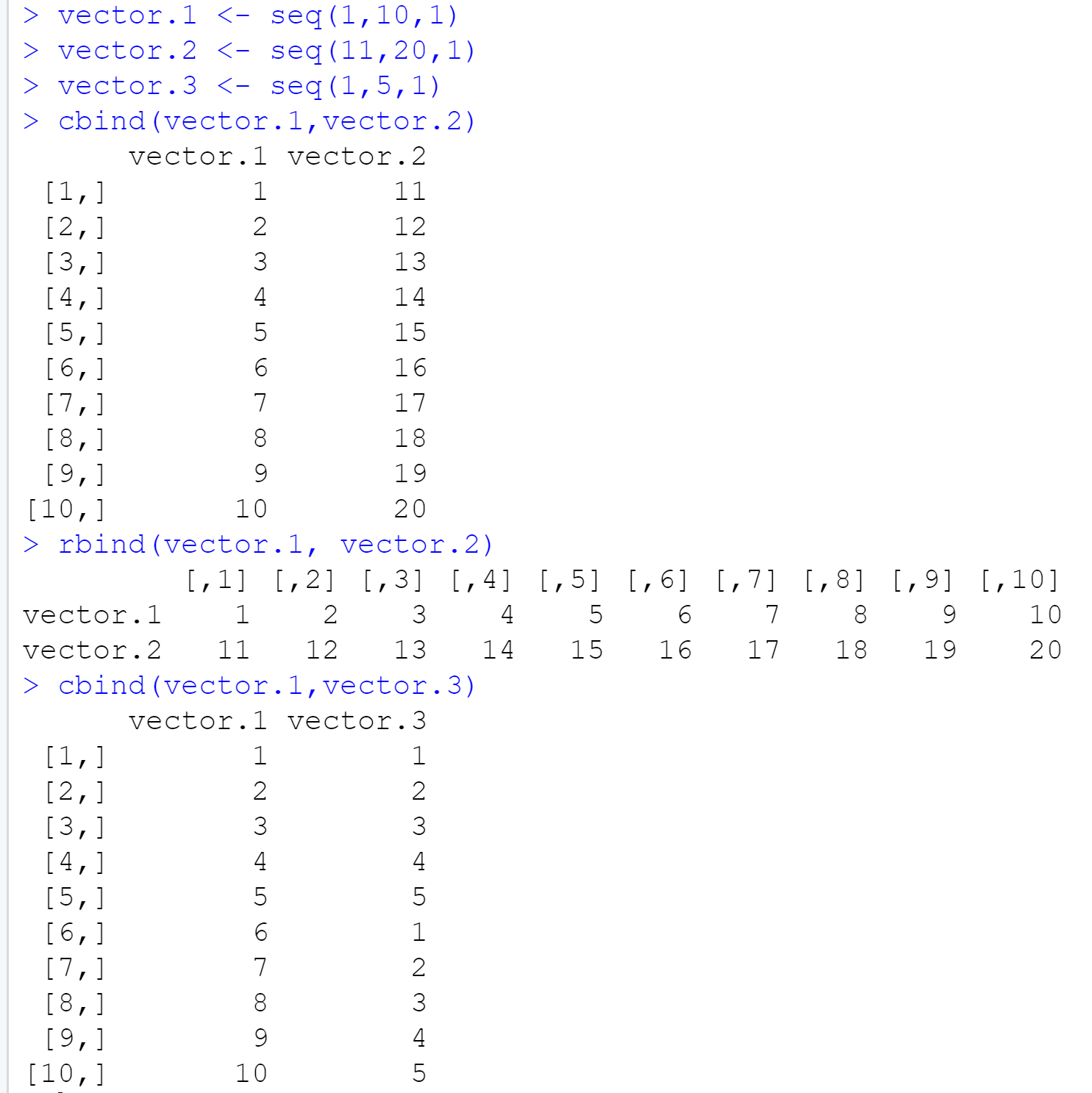


Two useful functions when working with vectors are cbind() and rbind()

cbind() combines two vectors together into columns

rbind() combines two vectors together into rows

Notice the vector recycling problem still exist.



# DATA FRAME

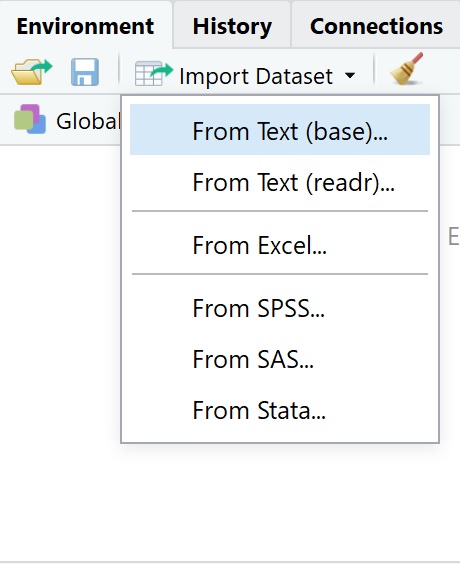
A data frame is a special data type. A data frame is the foundation of most data analysis in R. like a data set. It has rows and columns. Each column is equivalent to a variable. Each row is equivalent to an observation.

## Importing Data Frame

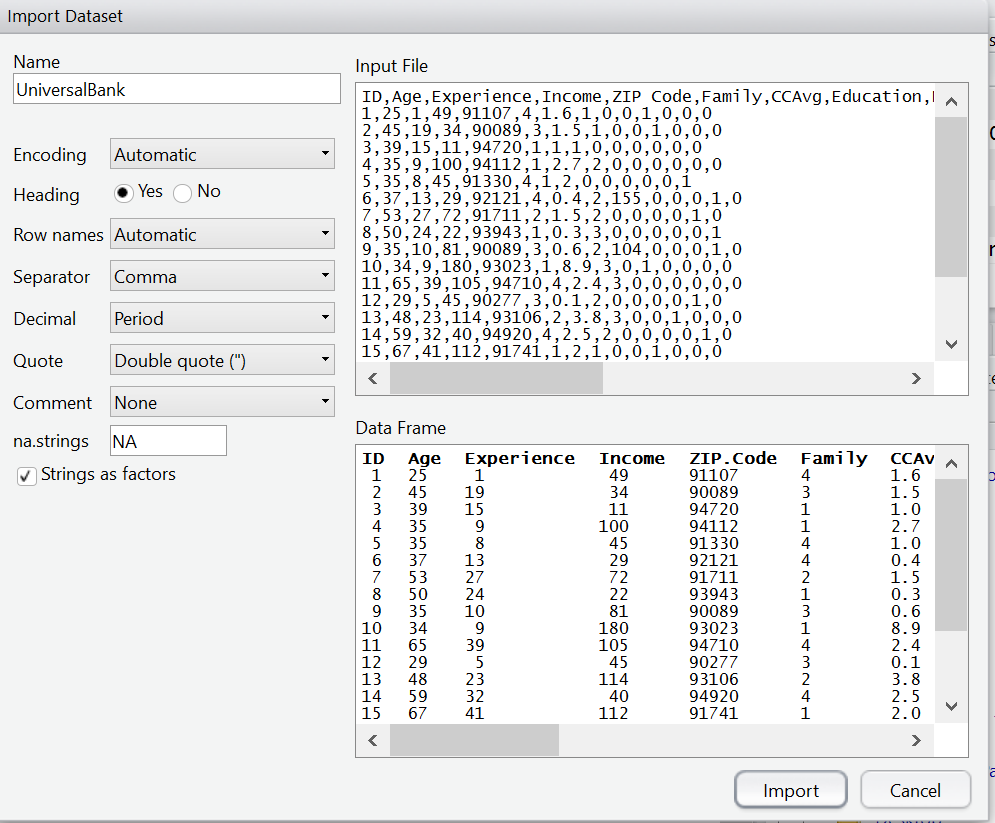
You need to first download the csv file titled “Universal Bank” from the course site.

To import the csv file into R, follow these steps:

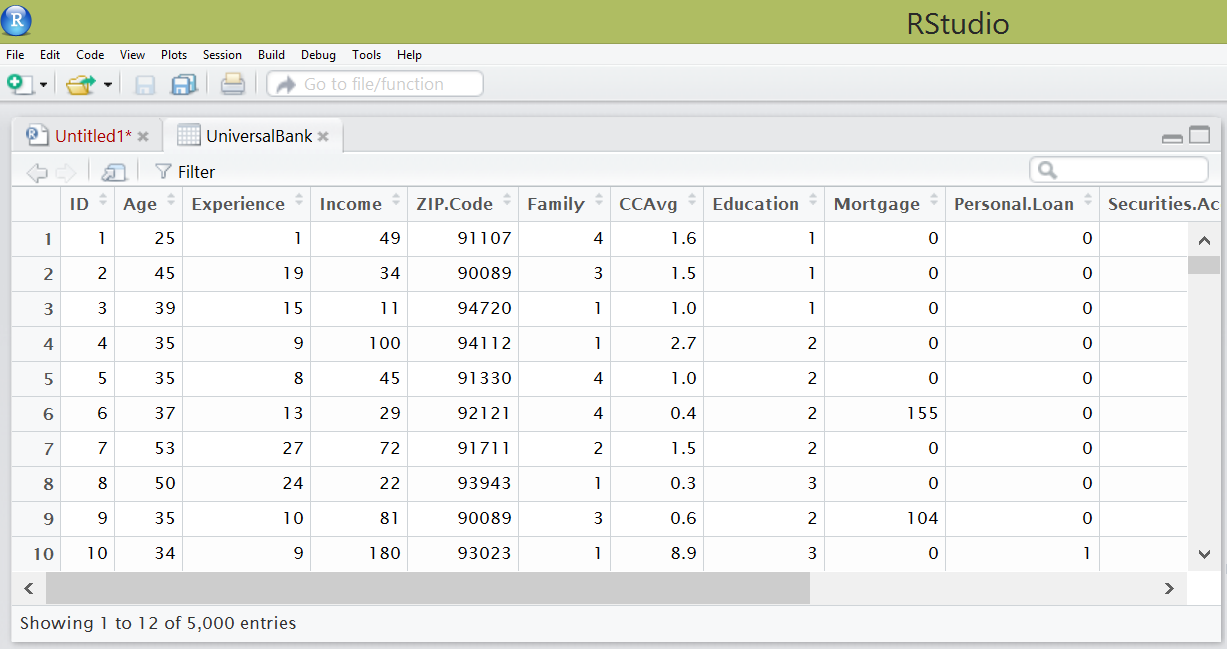
1. In the “Environment” window (top right of R Studio), click on the “Import Dataset” button.
2. Choose “From Text File” option.



1. Navigate to where your csv file is located and then click on it. You will then see this pop up window.



1. Click on the “Import” button.
2. You will see a preview of the Universal Bank dataset on the top left window of R Studio. Notice that this dataset has 5,000 rows.



Here is a description of each column in the Universal Bank dataset:

ID: Customer ID

Age: Customer’s age in years

Experience: Number of years of professional work experience

Income: Annual income in thousands of dollars ($000)

Zip code: Home address; zip code.

Family: Family size of customer

CC Avg: Average spending on credit cards per month in thousands of dollars ($000)

Education: Education level where 1 = Undergraduate; 2 = Graduate; and 3=Advanced/Professional.

Mortgage: Value of house mortgage if any; in thousands of dollar ($000)

Personal loan: Did the customer accept a personal loan offered in the bank’s last campaign? 1=Yes; 0 = No.

Securities account: Does the customer have a securities account with the bank? 1 = Yes; 0 = No.

CD account: Does the customer have a certificate of deposit (CD) account with the bank? 1 = Yes; 0 = No.

Online: Does the customer use Internet banking facilities? 1 = Yes; 0 = No.

Credit card: Does the customer use a credit card issued by Universal Bank? 1 = Yes; 0 = No.

Take a look at the data frame. Can you identify a numeric variable? A factor?

## READ.CSV()

We can also use read.csv() to import a data frame.

> getwd() #We print the working directory.

[1] "C:/Users/PhamX/Documents"

> setwd("C:/Users/PhamX/Desktop") #Now we change the working directory to

Where we saved our Universal Bank dataset.

> my.first.data.frame <- read.csv("C:/Users/PhamX/Desktop/UniversalBank.csv", header=TRUE, sep=”,”)

#Here we import in UniversalBank.csv and called the object

“my.first.data.frame”

## EXPORTING CSV FILES FOR SHARING

To share your data frame with other users in csv format, you use the write.csv() function.

> write.csv(UniversalBank, file = "C:/Users/PhamX/Desktop/BankData.csv")

# EXPLORATORY DATA ANALYSIS

## Data frame dimensions

**dim()** provides the number of observations and variables.

> dim(UniversalBank)

[1] 5000 14

**names()** gives the column names

> names(UniversalBank)

**head()** provides a preview of the first six observations in a data frame (by default). You can specify the number of observations to be previewed as well.

> head(UniversalBank)

> head(UniversalBank,10)

**tail()** provides a preview of the last six observations in a data frame. You can also specify the number of observations to be previewed as well.

> tail(UniversalBank)

> tail(UniversalBank,10)

**str()** provides the structure of the data frame: number of observations; number of variables; name of each variable; data type of each variable; and the first ten values.

> str(UniversalBank)

**summary()** provides the minimum value, 1st quantile value, median, mean, 3rd quantile value, and maximum value for each variable.

**fivenum()** provides the minimum value, lower-hindge, median, upper-hinge, and maximum for the specified variable in the data frame.

Be careful when reading the output! For example, does a summary for ID column make sense?

> summary(UniversalBank)

> fivenum(UniversalBank)

## Accessing specific variables (columns) or observations (rows)

What if you just want to look at one column at a time? You need to separate the data frame and the column with a $ sign.

> summary(UniversalBank$Income)

Min. 1st Qu. Median Mean 3rd Qu. Max.

8.00 39.00 64.00 73.77 98.00 224.00

You can specify the number of row in the data frame to print. Remember R understands data frame's dimension as [row, column].

This shows the third row in the data frame:

> UniversalBank[3,]

ID Age Experience Income ZIP.Code Family CCAvg Education Mortgage Personal.Loan Securities.Account CD.Account Online CreditCard

3 3 39 15 11 94720 1 1 1 0 0 0 0 0 0

This shows the third row and columns 3 through 7 in the data frame.

> UniversalBank[3,3:7]

Experience Income ZIP.Code Family CCAvg

3 15 11 94720 1 1

## DESCRIPTIVE STATISTICS

These are the descriptive statistics that base R supports:

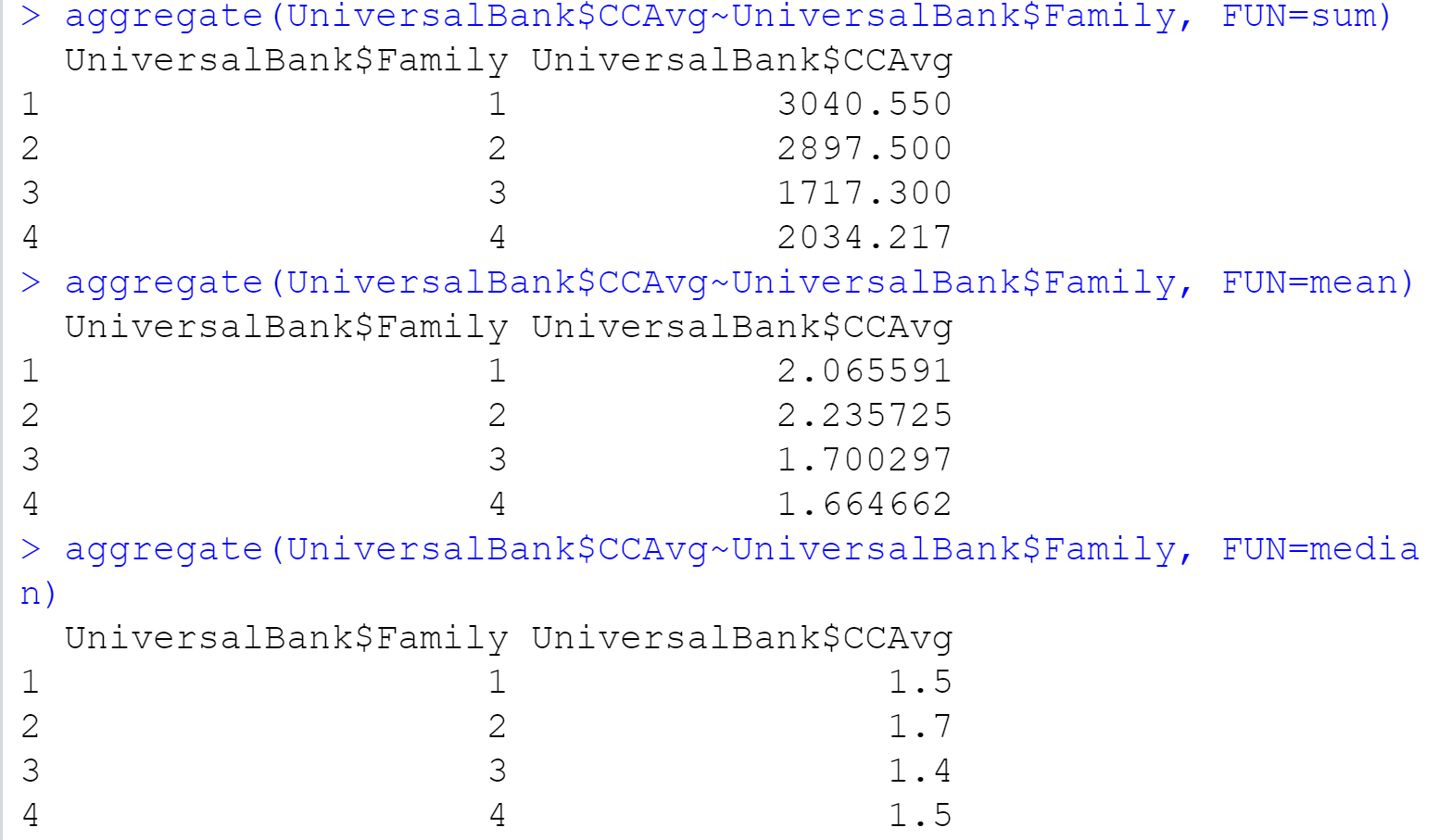
|  |  |  |
| --- | --- | --- |
| **Descriptive Statistic** | **R function** | **Example** |
| Mean | **mean()** | > mean(UniversalBank$Income)  [1] 73.7742 |
| Median | **median()** | > median(UniversalBank$Income)  [1] 64 |
| Weighted.Mean | **weighted.mean(x= , w=)** | > weighted.mean(x=UniversalBank$Income, w=UniversalBank$Experience)  [1] 72.55149 |
| Variance | **var()** | > var(UniversalBank$Income)  [1] 2119.104 |
| Standard Deviation | **sd()** | > sd(UniversalBank$Income)  [1] 46.03373 |
| Minimum | **min()** | > min(UniversalBank$Income)  [1] 8 |
| Maximum | **max()** | > max(UniversalBank$Income)  [1] 224 |

If you have missing values (NA) in your data set, all of the base R functions above with the exception of summary() will not produce valid results. You will need to add the option "na.rm = TRUE" to remove the missing values.

> mean(UniversalBank$Income, na.rm=TRUE)

## AGGREGATE

The aggregate() function is equivalent to PivotTable in Excel.

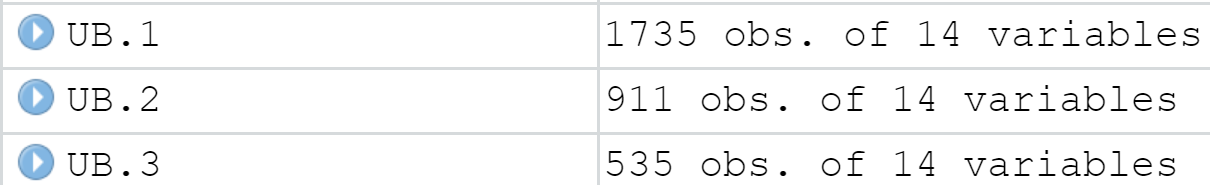


## SUBSET

> UB.1 <- subset(UniversalBank, Age<40)

> UB.2 <- subset(UniversalBank, Age<40 & Family <=2)

> UB.3 <- subset(UniversalBank, Age<40 & Family <=2 & Online==1)



## CORRELATION

The Pearson’s correlation coefficient ranges from -1 to 1. A coefficient closer to -1 indicates a strong negative correlation (X increases and Y decreases). A correlation coefficient closer to 1 indicates a strong positive correlation (X increases and Y increases). A correlation coefficient closer to 0 indicates no correlation between X and Y.

The **cor()** function calculates the Pearson’s correlation coefficient.

> cor(UniversalBank$Income, UniversalBank$Experience)

[1] -0.04657418

You can also look at the correlation of multiple variables at once.

> d<-cor(UniversalBank)

## DATA TRANSFORMATION

### Deleting Variable

There are two ways to delete a variable.

1. You can specify the variable name and then make the values “NULL.”

> UniversalBank$ID<-NULL

1. You can specify the column number with a minus sign to delete it.

> UniversalBank <- UniversalBank[-1]

### Creating New Variable

To create a new variable, you use an assignment statement.

Here we divide Income by the number of individuals in the family to create Income Per Capita.

> UniversalBank$Inc\_Per\_Capita <- UniversalBank$Income/UniversalBank$Family

> UniversalBank$CC\_type <- ifelse(UniversalBank$CCAvg <= 1, “low”, “high”)

> UniversalBank$Education <- factor(UniversalBank$Education, levels=c(1,2,3), labels=c("BA", "MA", "Professional"))

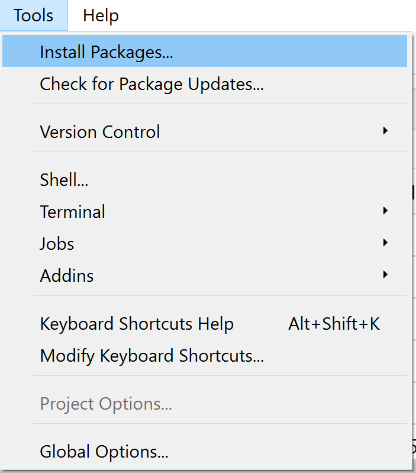
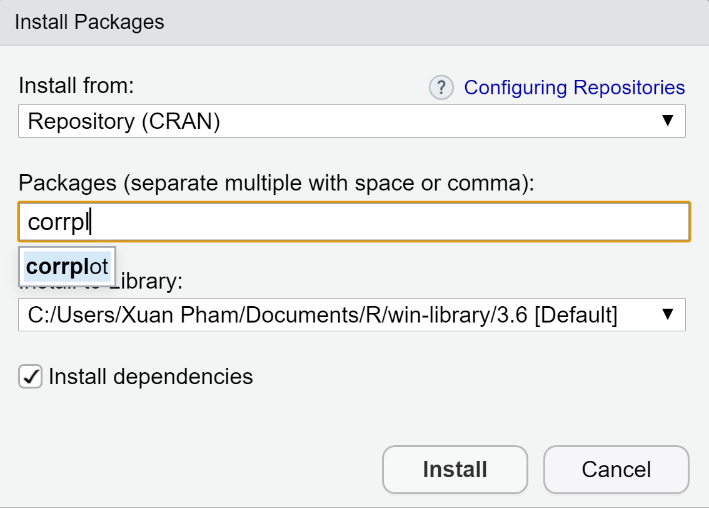
If you want to override the old variable with the new variable, set the names the same.

If you give the new variable a different name, R will preserve the old variable.

# R PACKAGES

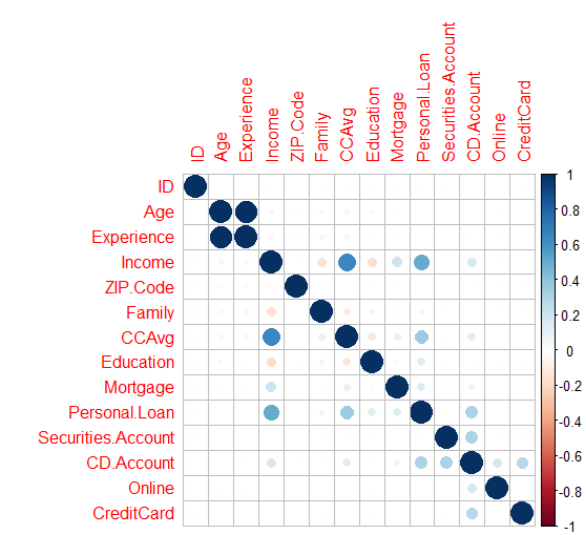
Remember we were looking at that big correlation coefficient matrix earlier? It’d be nice if we can use an R package to make it easier to interpret.

## corrplot package

> library(corrplot) #load the package to use

> corrplot(d) #use the function corrplot from this package



What if we want to see the actual correlation coefficients?

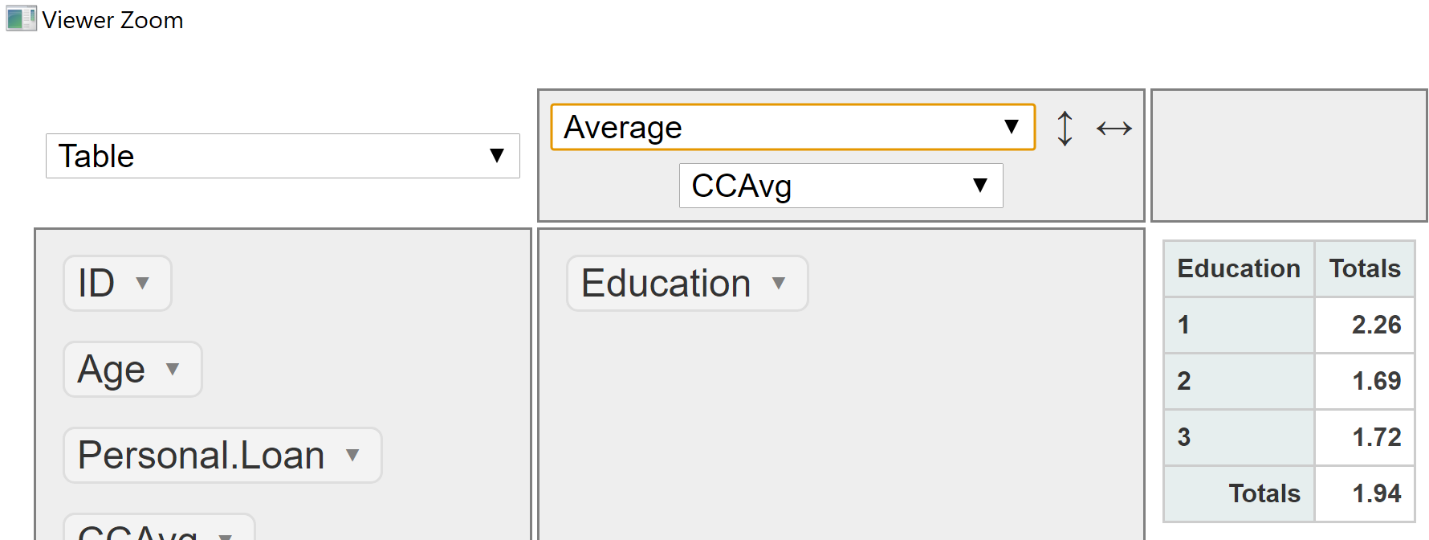
corrplot(d, method = “number”)

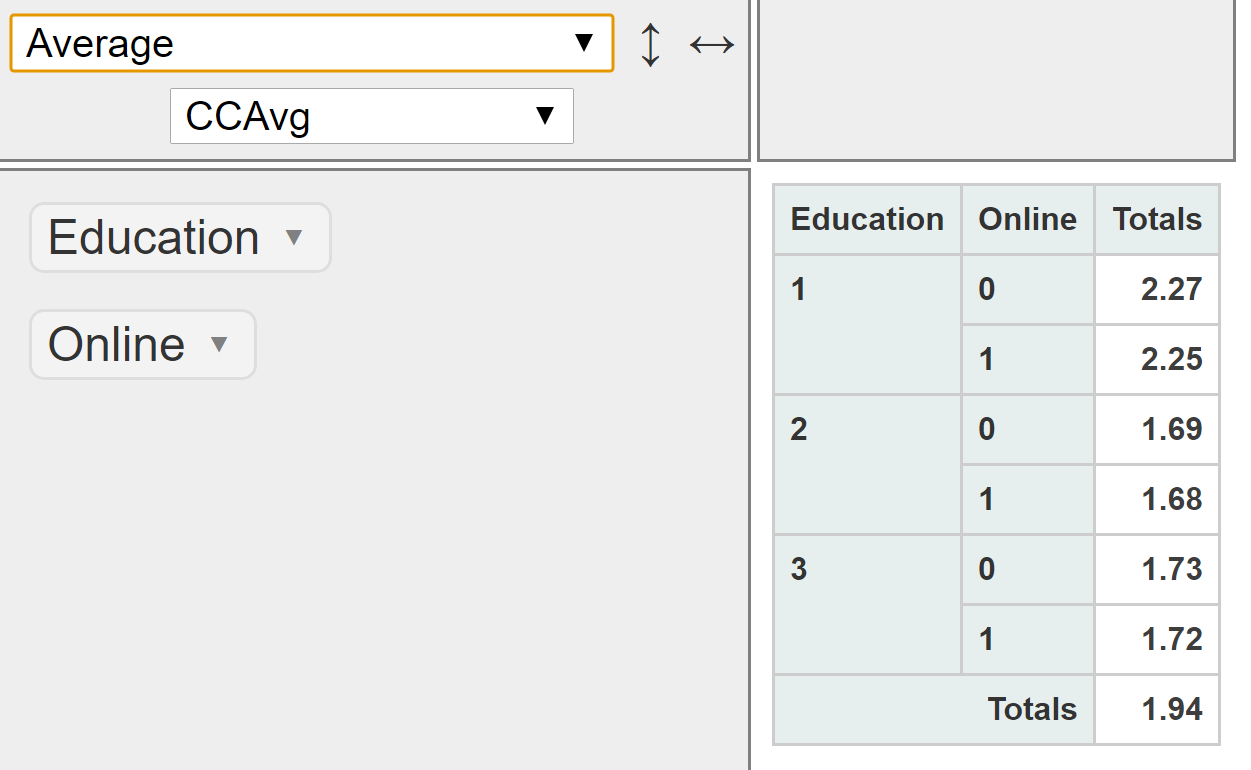
## rpivotTable

Install and load the rpivotTable package. Run this line

rpivotTable(UniversalBank)

Now you can use PivotTable in R just like in Excel!





# CHARTS

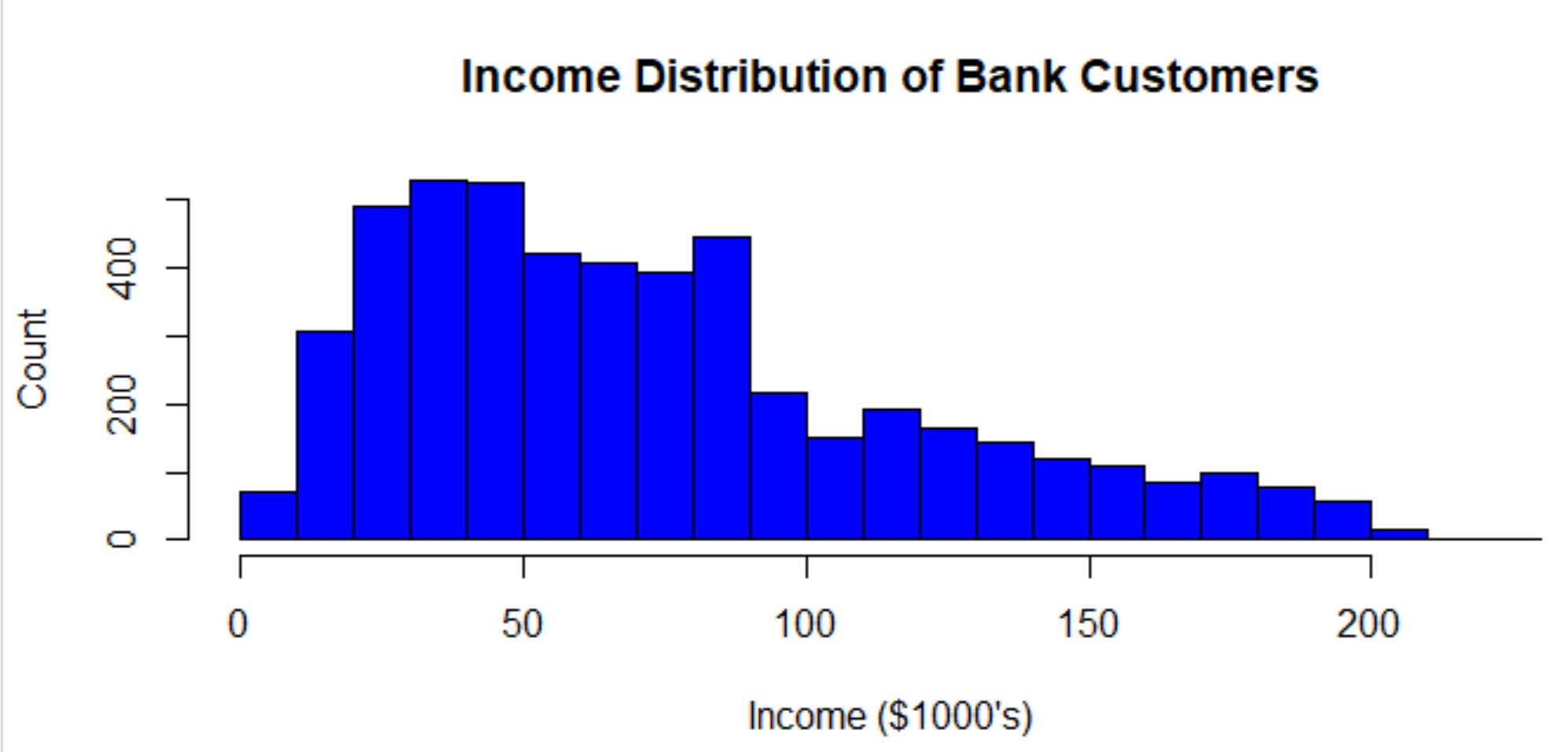
Plotting in R can mostly be done in two ways: 1) with base R functions and 2) with the ggplot2 package. Be sure to install and load ggplot2 package before proceeding.

## Histogram

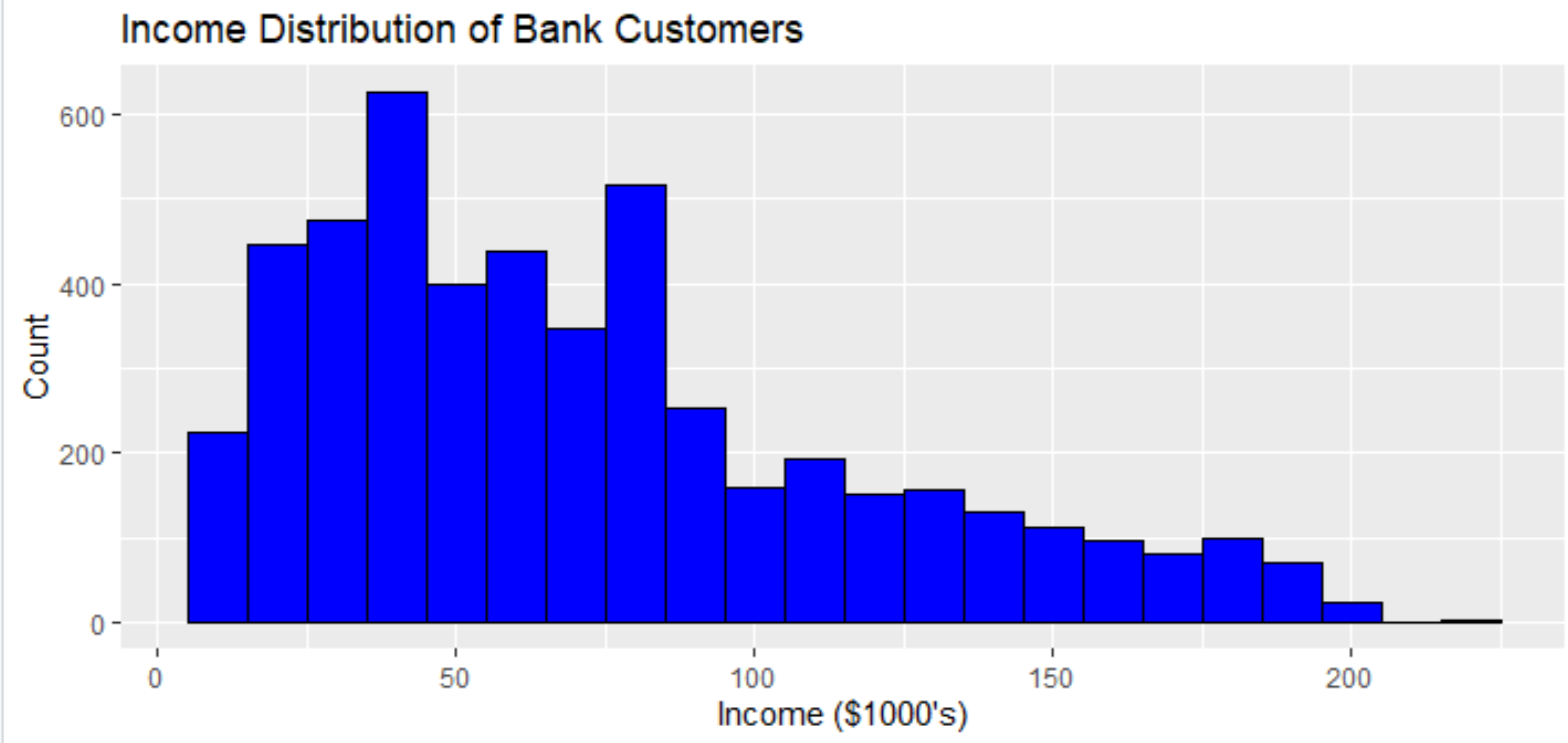
hist(UniversalBank$Income)

hist(UniversalBank$Income, breaks=20)

hist(UniversalBank$Income, breaks=20, main = "Income Distribution of Bank Customers", xlab="Income ($1000's)", ylab="Count", col="blue")



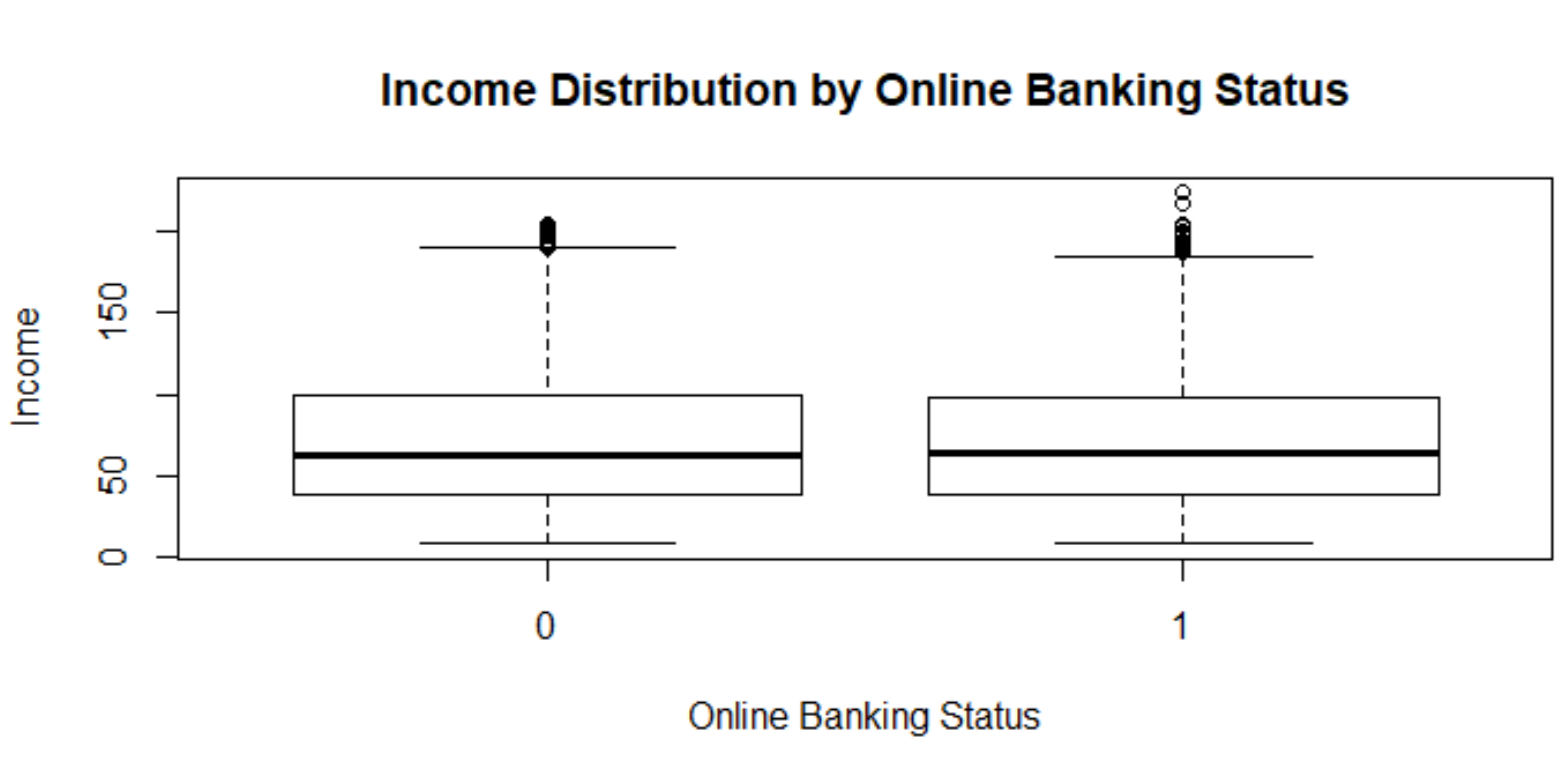
ggplot(data=UniversalBank) + geom\_histogram(aes(x=Income), binwidth=10, fill="blue", color="black") + labs(x="Income ($1000's)", y="Count", title ="Income Distribution of Bank Customers")



## Boxplot

boxplot(UniversalBank$Income)

boxplot(UniversalBank$Income~UniversalBank$Online)

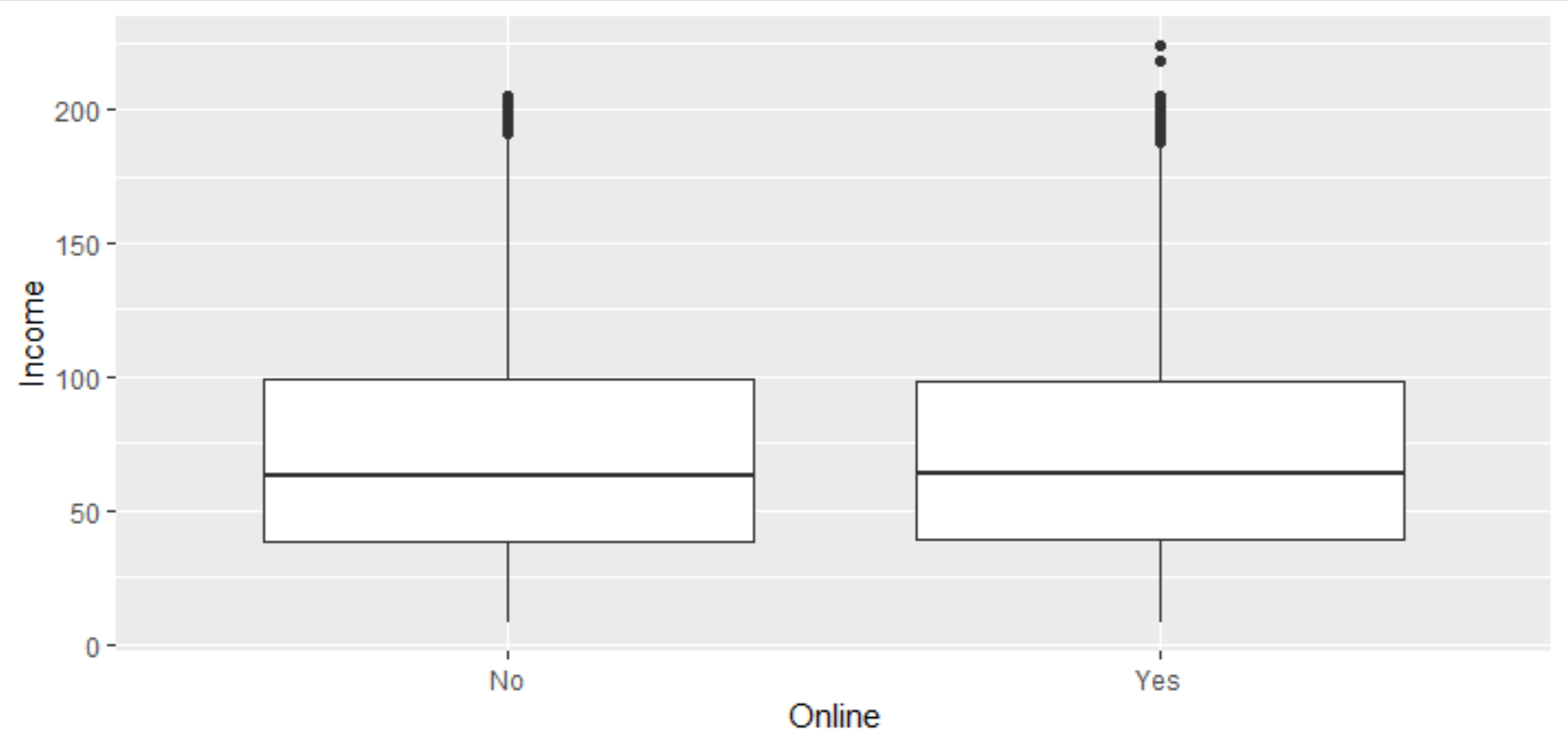


The boxplot in ggplot requires groups to be recoded into factors first.

UniversalBank$Online<-factor(UniversalBank$Online,levels=c(0,1), labels=c("No","Yes"))

> ggplot(UniversalBank, aes(y=Income, x=Online)) + geom\_boxplot()

> ggplot(UniversalBank, aes(y=Income, x=Online)) + geom\_boxplot(title="Boxplots of Income Distribution by Online Banking Status")

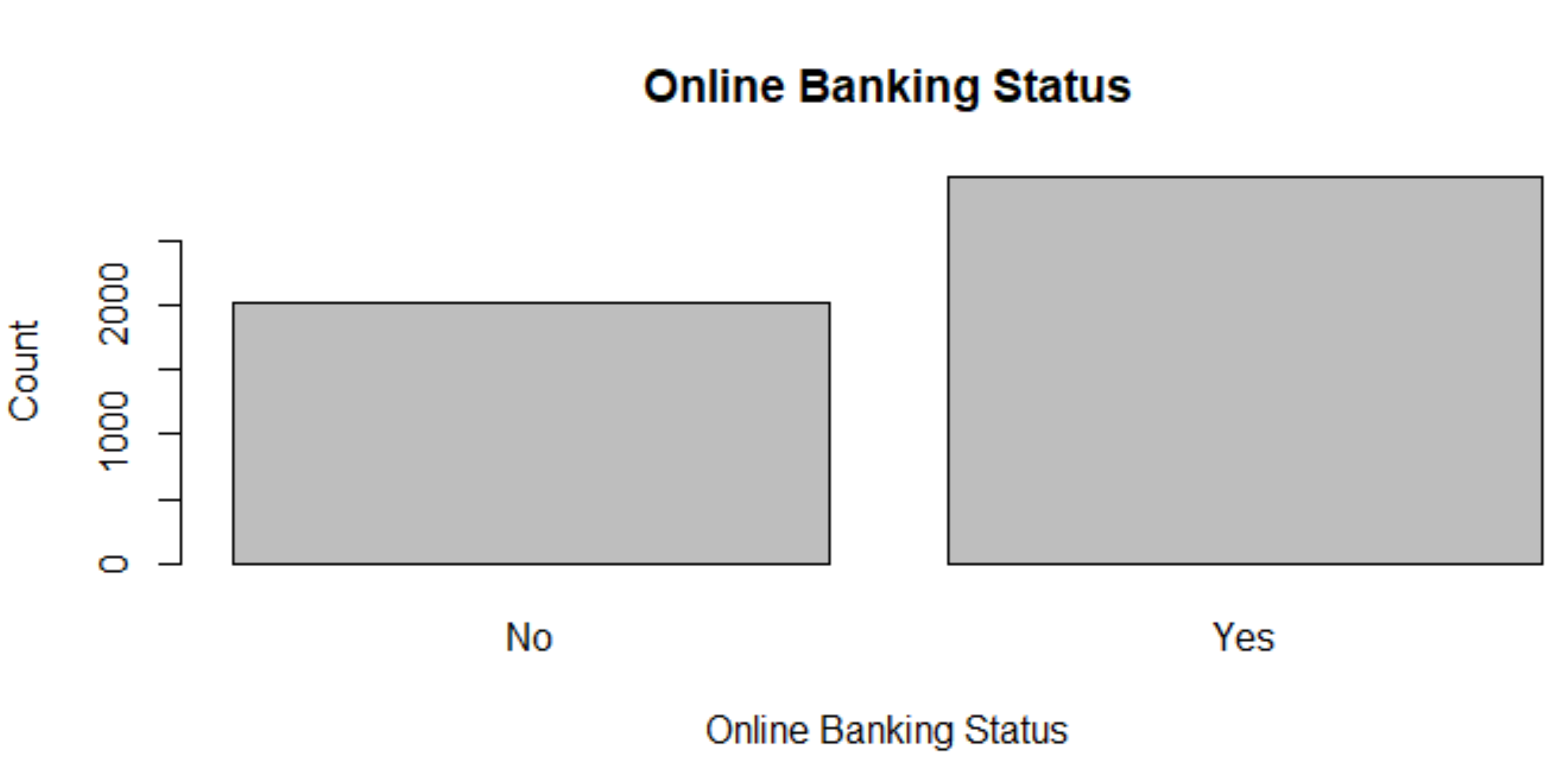


## Bar Chart

Before you can create a bar chart with base R, you must create a frequency table.

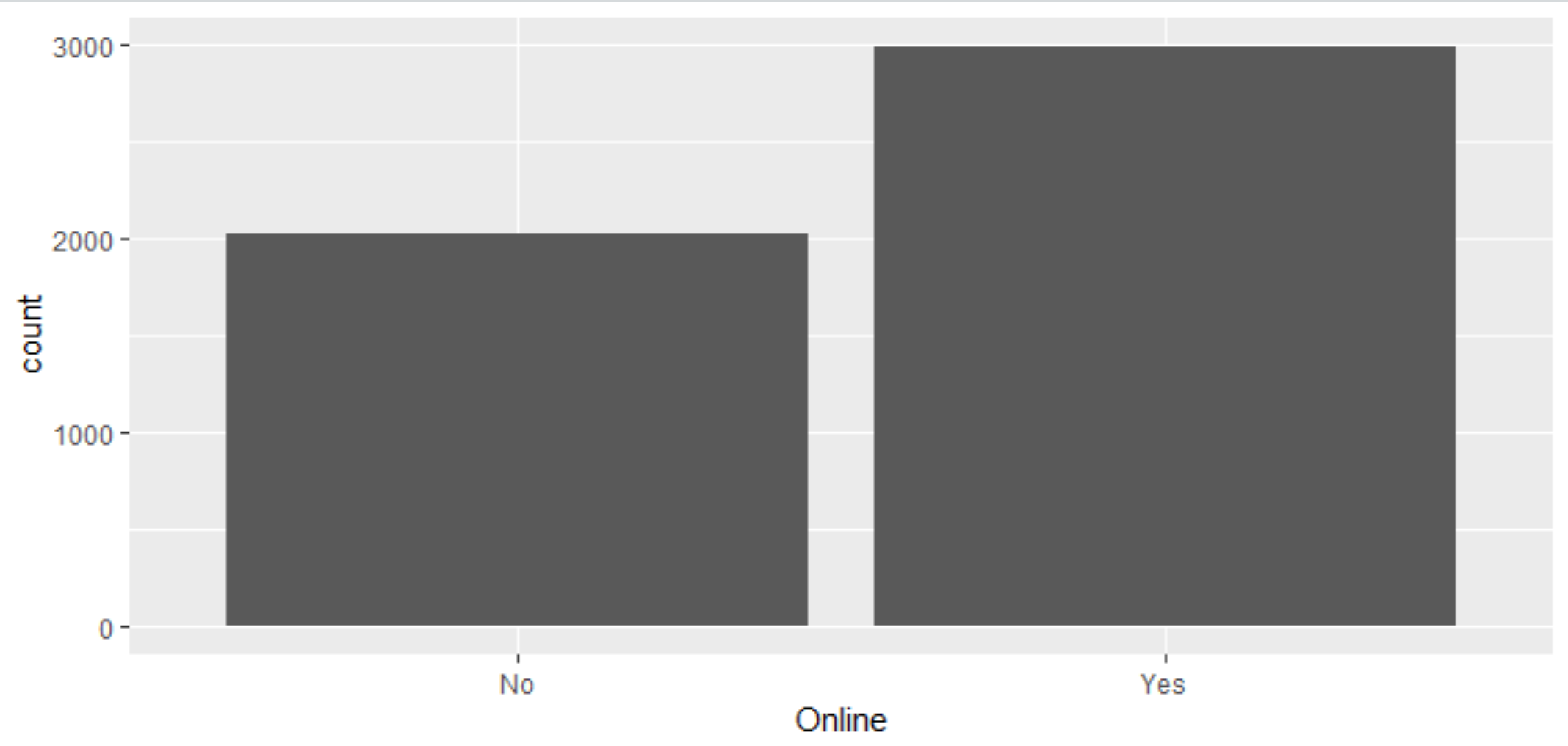
counts <- table(UniversalBank$Online)

barplot(counts, xlab = "Online Banking Status", ylab = "Count", main = "Online Banking Status")



You do not create a frequency table before making a bar chart with ggplot

ggplot(data=UniversalBank) + geom\_bar(aes(x=Online))



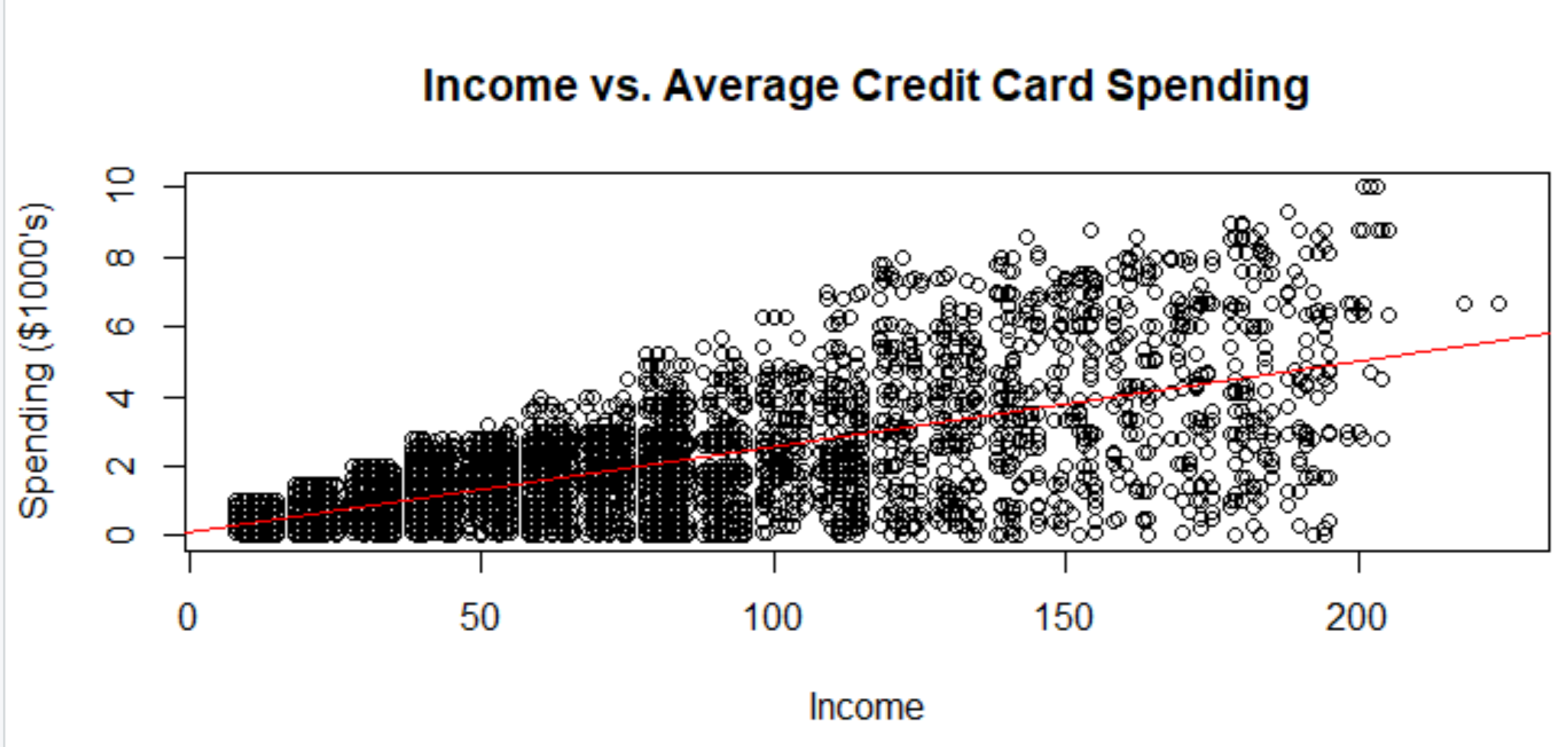
## Pie Chart

Pie charts are not visually accurate representation of proportions. Don’t use pie charts. Just say no!

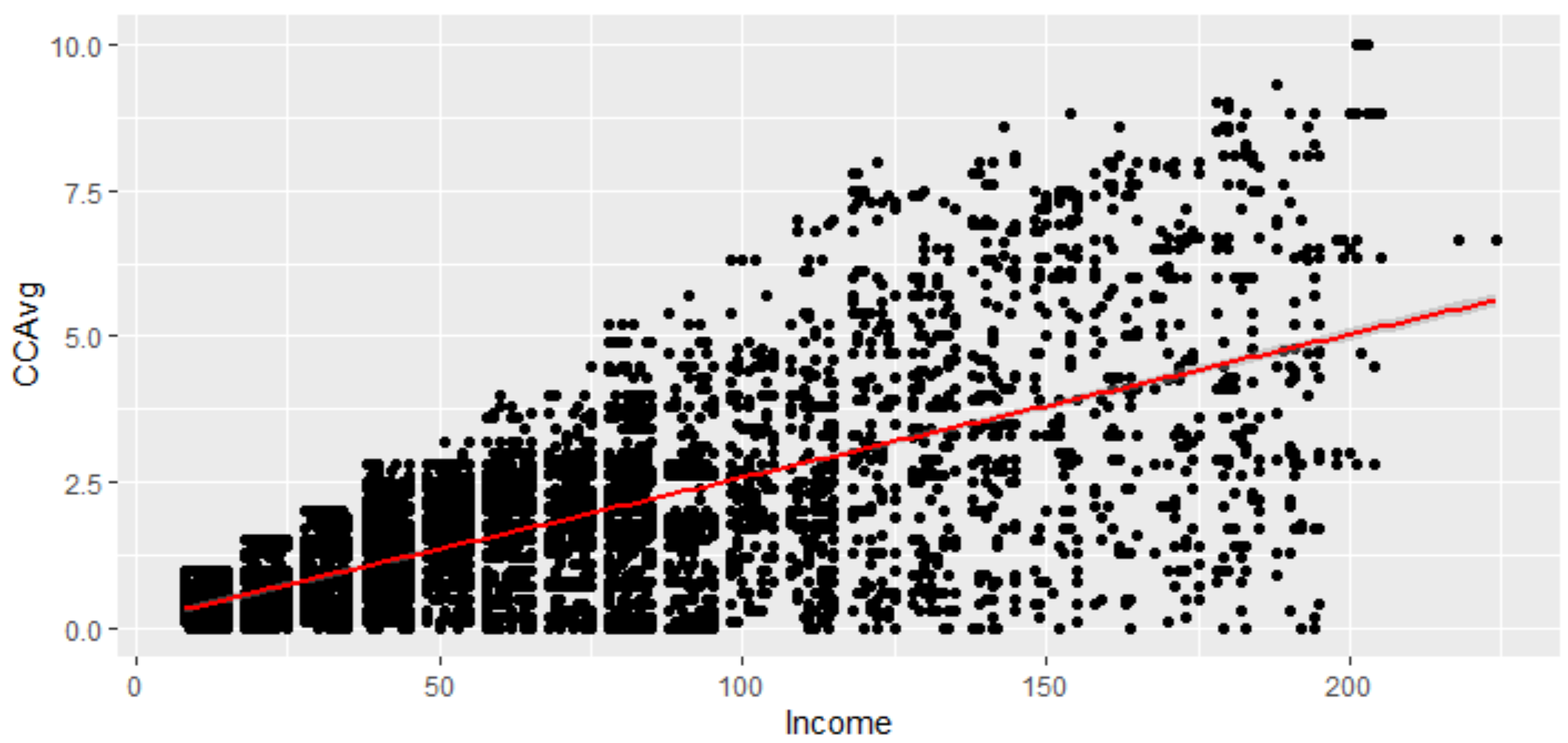
## Scatterplot and Trend Line

plot(UniversalBank$Income, UniversalBank$CCAvg, main = "Income vs. Average Credit Card Spending", xlab = "Income", ylab = "Spending ($1000's)")

abline(lm(UniversalBank$CCAvg~UniversalBank$Income), col = "red")



ggplot(UniversalBank, aes(x=Income, y=CCAvg)) + geom\_point() + geom\_smooth(method="lm", color="red", size=1)

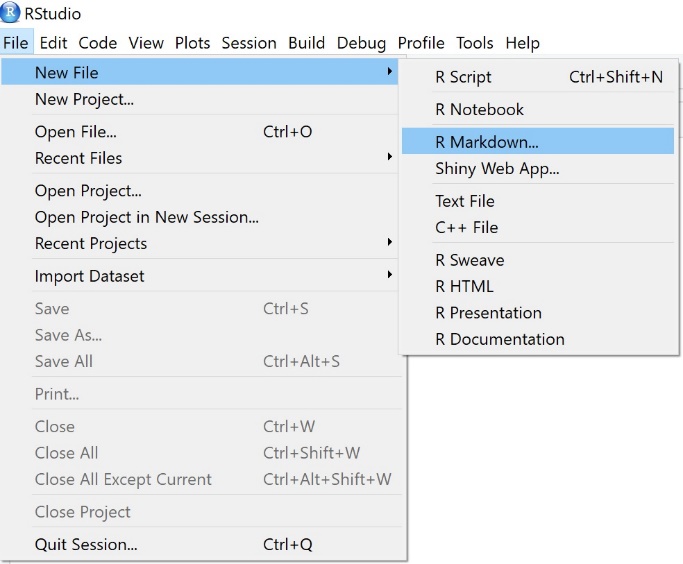


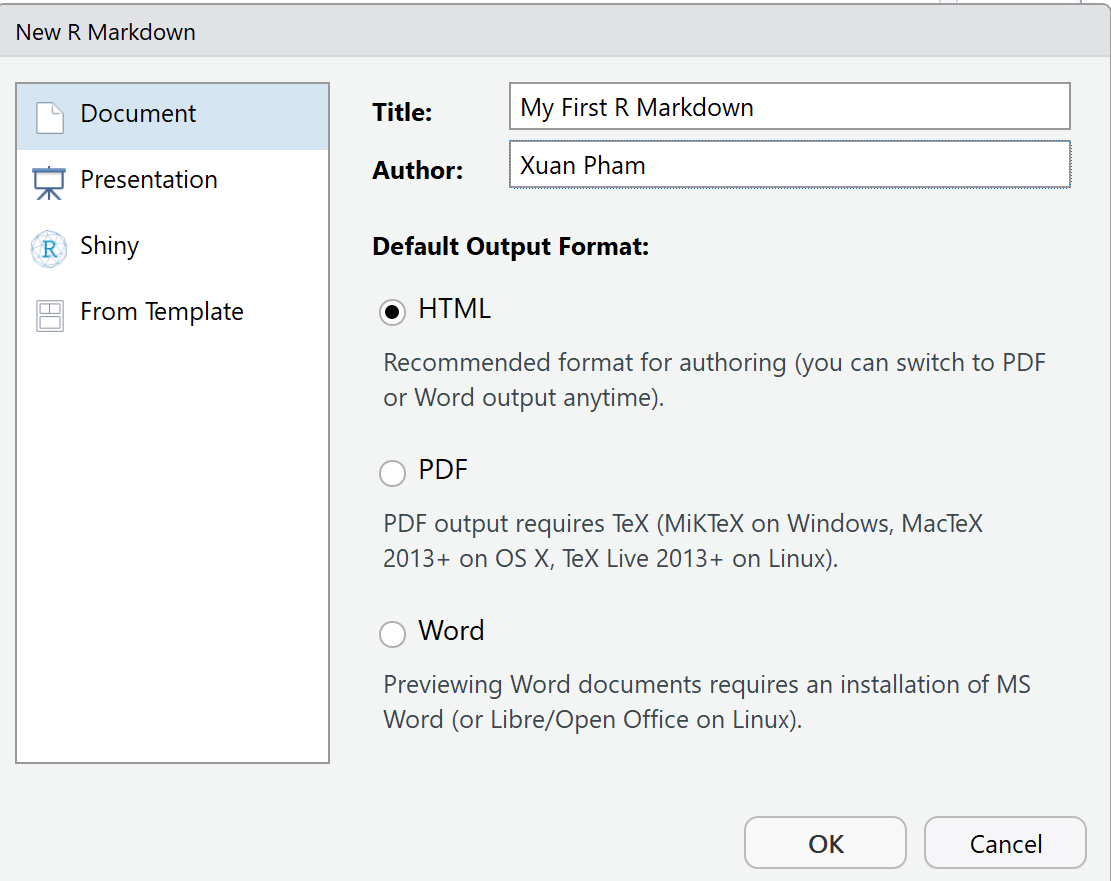
# CREATING DYNAMIC R DOCUMENTS (R MARKDOWN)

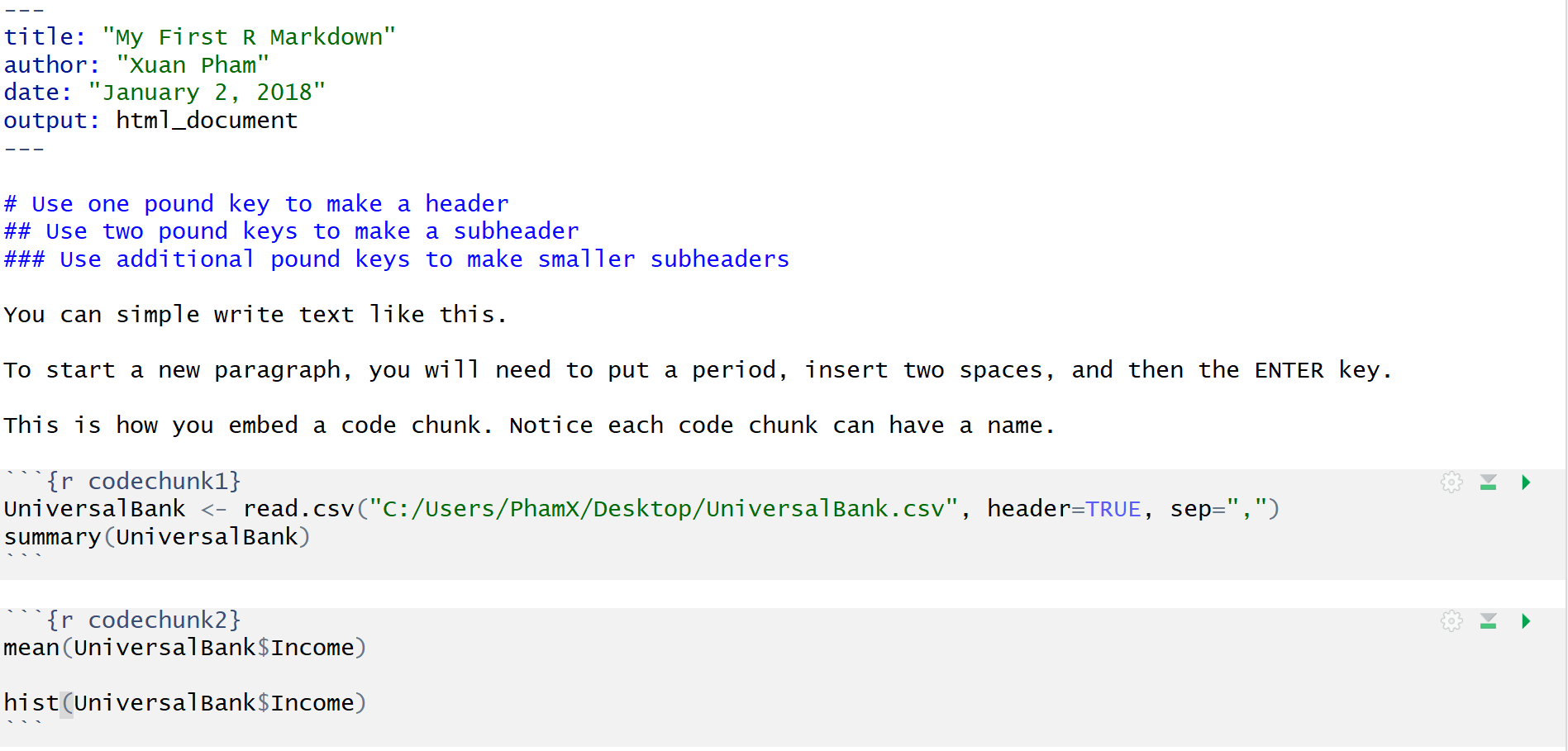
An R markdown document is an R script and a word processor all in once. The purpose of an R markdown document is to make your work reproducible. Anyone should be able to look at your R markdown file and be able to replicate your outputs. Here is more information: <http://rmarkdown.rstudio.com/>

Lessons about R markdown are available here: <http://rmarkdown.rstudio.com/lesson-2.html>

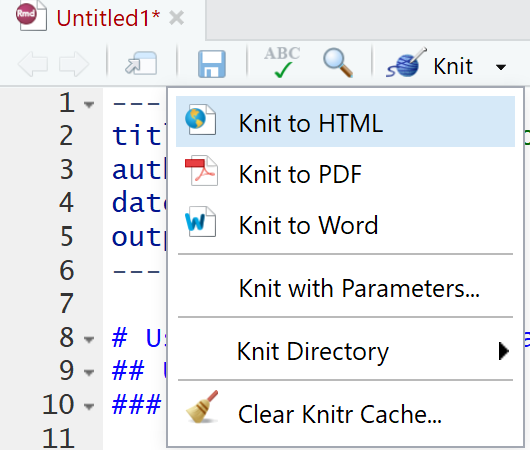
Let’s make an R markdown document.

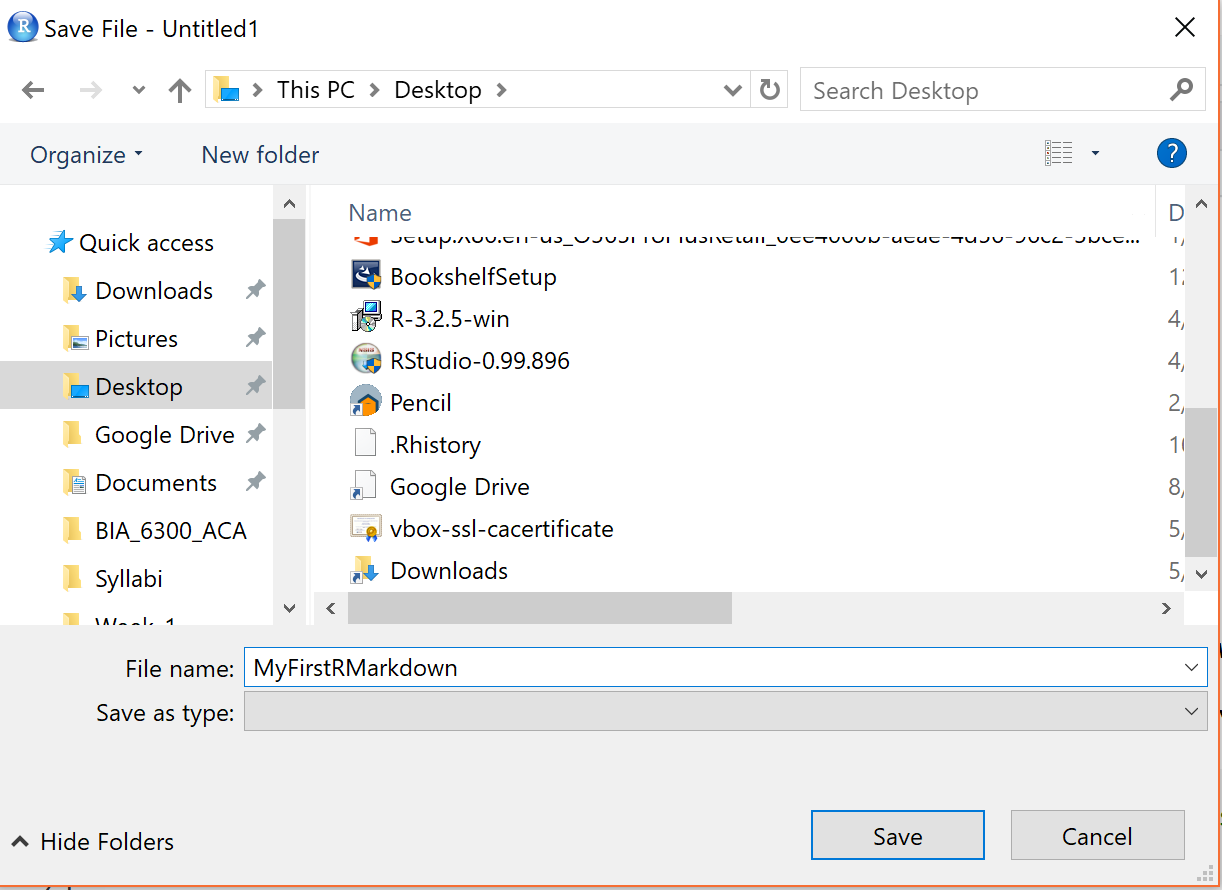






**KNITTING AN R MARKDOWN FILE**





# RESOURCES

R-bloggers: <https://www.r-bloggers.com/>

Quick R: <http://www.statmethods.net/>

Stack Overflow: <http://stackoverflow.com/>

Cross Validated: <http://stats.stackexchange.com/>

Various reference manuals for R packages (available via CRAN): <https://cran.r-project.org/web/packages/available_packages_by_name.html>