ITEC 621 Exercise 1 - R Refresher

J. Alberto Espinosa

January 15, 2023

Table of Contents

General Instructions

Download the R Markdown template for this exercise **Ex1_R_YourLastName.Rmd** and save it with your own last name **exactly**. Then open it in R Studio and complete all the exercises and answer the questions below in the template. Run the code to ensure everything is working fine. When done, knit your R Markdown file into a **Word document** and submit it. No need to submit the **.Rmd**, file just the Word knitted file. If for some reason you can't knit a Word file, you can knit to a PDF file, or to an HTML file and then save it as a PDF. Some LMS systems don't accept HTML submissions.

This exercise is similar to HW0 in KSB-999, which you were requred to complete befor starting this course. So, if you already did that, this should be an easy exercise and a good warm up refresher. If you didn't do it, this is you opportunity to catch up. This course moves fast and it assumes that you have some familiarity with R.

R Markdown Overview (please read carefully)

R Markdown is a package that allows you to write R code and prepare an analytics report in a single file. To use R Markdown, you first need to install it in your computer using the command `install.packages("rmarkdown")`. If you have not done this yet, go to the **R Console** and install R Markdown. Once you have done this, you can create R Markdown files from the File -> New File menu.

When you create an R Markdown file, it will look like text comingled with R code. You will see a button option named **Knit** in your tool bar (if you don't, then R Markdown is not installed). Once you are done with all the coding, click on the **Knit** button and R Markdown will knit a Word, HTML, PDF or PowerPoint document for you, depending on the output type you specified, with all your typed text and R results.

Important: This is a business course and, as such you are required to submit all exercises, homework and project reports with a professional, businesslike appearance, free of grammatical errors and typos, and with well articulated interpretation narratives. **No knitting, improper knitting and submissions with writing and formatting issues will have up to 3-point (out of 10) deductions for exercises and up to 10-point (out of 100) deductions for homework.**

R Markdown contains three main types of content:

1. The **YAML** (YAML Ain't Markup Language) header, which is where you place the title, author, date, type of output, etc. It is at the top of the R Markdown file and starts and ends with `---`. I suggest using an output type `word_document`. HTML works well, but blackboard will not read HTML files submitted by students (for security reasons).

2. **Markup** sections, which is where you type any text you wish, which will show up as typed text. You will learn these later.

3. **Code chunks**: which is where you write your R code. An R code chunk starts with a ```**{r}** and ends with a ```.

I recommend that you first write your R code in an R Script file to try your R code first. Once you are satisfied that the R code is working fine, then you can copy/paste the respective code segments to the corresponding R Markdown code chunks.

Your knitted file must:

- Display all your R commands (leave `echo=T` in the global options; `echo=F` suppresses the R code)
- Display the resulting **R output results**
- Contain anay necessary text and explanations, as needed; and
- Be formatted for good readability and in a business like manner
- Be in the same order as the questions and with the corresponding question numbers

1. Basic R Concepts

1.1 Write a simple R function named **area()** that takes 2 values as parameters (x and y, representing the two sides of a rectangle) and returns the product of the two values (representing the rectangle's area). Then use this function to diaplay the area of a rectangle of sides 6x4. Then, use the functions `paste()`, `print()` and `area()` to output this result: **The area of a rectangle of sides 6x4 is 24**, where 24 is calculated with the `area()` function you just created

```
area <- function(x,y) {return(x*y)}
area(4,6)

## [1] 24

print(paste("The area of a 4x6 rectanlge is", area(4,6)))
```

```
## [1] "The area of a 4x6 rectanlge is 24"
```

1.2 Write a simple **for loop** for i from 1 to 10. In each loop cycle, compute the area of a rectangle of sides i and i*2 (i.e., all rectangles have one side double the lenght than the other) and for each of the 10 rectangles display "The area of an 1 x 2 rectangle is 2" for i=1, "The area of an 2 x 4 rectangle is 8", and so on.

```
for (i in 1:10) {
  print(paste("The area of a", i, "x", i*2,
              "rectangle is", area(i, 2*i)))
  }

## [1] "The area of a 1 x 2 rectangle is 2"
## [1] "The area of a 2 x 4 rectangle is 8"
## [1] "The area of a 3 x 6 rectangle is 18"
## [1] "The area of a 4 x 8 rectangle is 32"
## [1] "The area of a 5 x 10 rectangle is 50"
## [1] "The area of a 6 x 12 rectangle is 72"
## [1] "The area of a 7 x 14 rectangle is 98"
## [1] "The area of a 8 x 16 rectangle is 128"
## [1] "The area of a 9 x 18 rectangle is 162"
## [1] "The area of a 10 x 20 rectangle is 200"
```

2. Data Manipulation

2.1 Copy the **Credit.csv** data file to your working directory (if you haven't done this yet). Then read the **Credit.csv** data file into a data frame object named **Credit** (Tip: use the `read.table()` function with the parameters `header=T, sep=",", row.names=1`). Then, list the first 5 columns of the top 5 rows (Tip: use `Credit[1:5,1:5]`)

```
Credit <- read.table("Credit.csv", header=T, sep=",", row.names=1)
Credit[1:5,1:5]

##    Income Limit Rating Cards Age
## 1  14.891  3606    283     2  34
## 2 106.025  6645    483     3  82
## 3 104.593  7075    514     4  71
## 4 148.924  9504    681     3  36
## 5  55.882  4897    357     2  68
```

2.2 Using the `class()` function, display the object class for the Credit data set, and for Gender (i.e., `Credit$Gender`), Income and Cards

```
class(Credit)

## [1] "data.frame"

class(Credit$Gender)

## [1] "character"

class(Credit$Income)
```

```
## [1] "numeric"

class(Credit$Cards)

## [1] "integer"
```

2.3 Create a vector named **income.vect** with data from the Income column. Then use the `head()` function to display the first 6 values of this vector.

```
income.vect <- Credit$Income
head(income.vect)

## [1]  14.891 106.025 104.593 148.924  55.882  80.180
```

3. Basic Descriptive Analytics

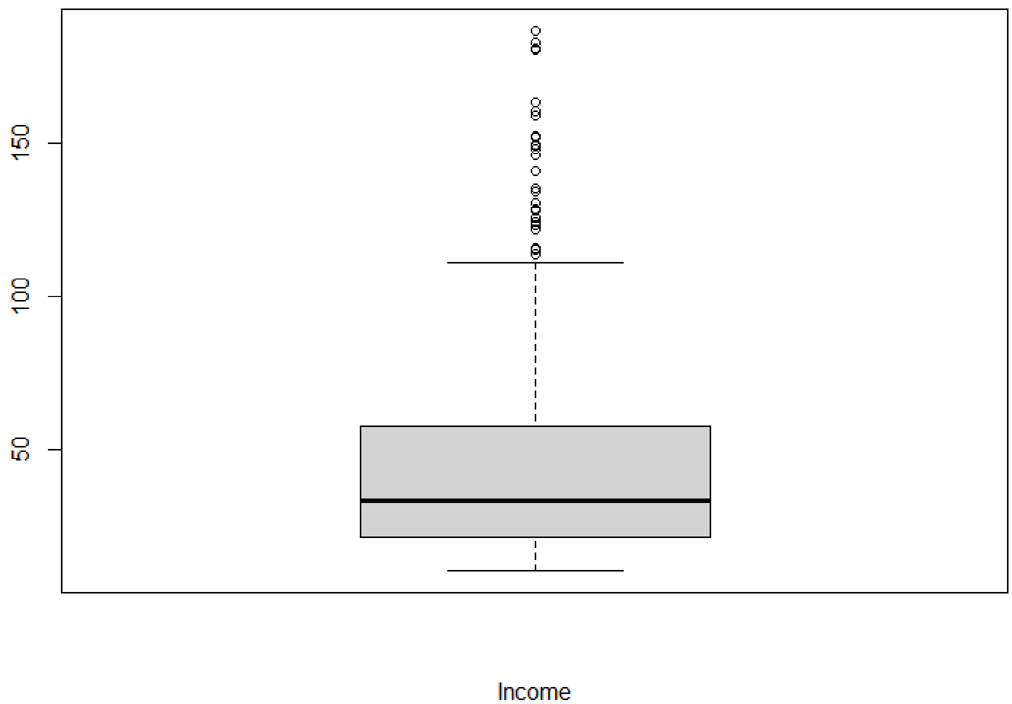
3.1 Compute the mean, minimum, maximum, standard deviation and variance for all the values in this income vector. Store the respective results in variables name mean.inc, min.inc, etc. Then, use the `c()` function to create a vector called **income.stats** with 5 values you computed above. Then use the `names()` function to give the corresponding names "Mean", "Min", "Max", "StDev", and "Var". Then display Income.stats vector, but wrap it within the `round()` function with a parameter `digits = 2` to display only 2 decimals.

Technical Note: The `names()` needs to create a vector with the respective names above, which need to correspond to the values in **incom.vect**. Therefore, you need to use the `c()` function to create a vector with these 5 names.

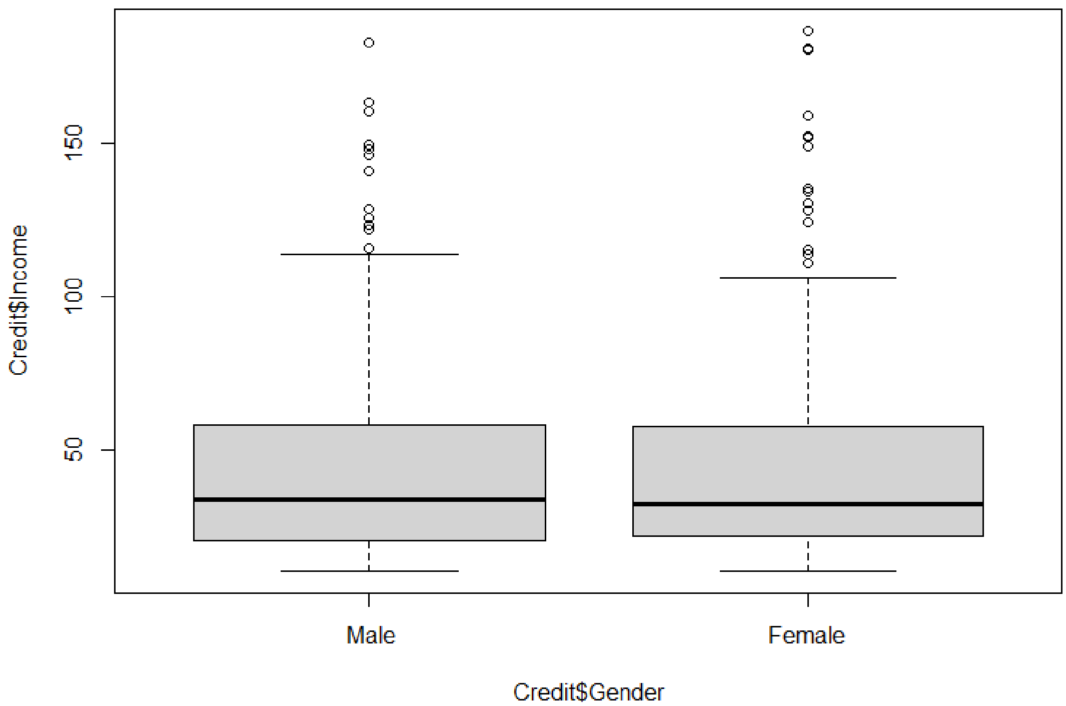
```
mean.inc <- mean(income.vect)
min.inc <- min(income.vect)
max.inc <- max(income.vect)
sd.inc <- sd(income.vect)
var.inc <- var(income.vect)

income.stats <- c(mean.inc, min.inc, max.inc, sd.inc, var.inc)
names(income.stats) <- c("Mean","Min","Max","StDev", "Var")
round(income.stats, digits = 2)

##    Mean     Min     Max   StDev     Var
##   45.22   10.35  186.63   35.24 1242.16
```

3.2 Display a boxplot for the predictor Income. Tip: you can do this 2 ways. First you can `attach()` the Credit data set (which loads the data set in the work environment) and then do a `boxplot()` for **Income**. Or, do it without attaching, but using the table prefix (i.e., **Credit$Income). Use the **xlab** attribute to name include the label "Income". Then display similar boxplots but this time broken down by **Gender** (i.e., `Credit$Income ~ Credit$Gender`).

```
boxplot(Credit$Income, xlab="Income")
```

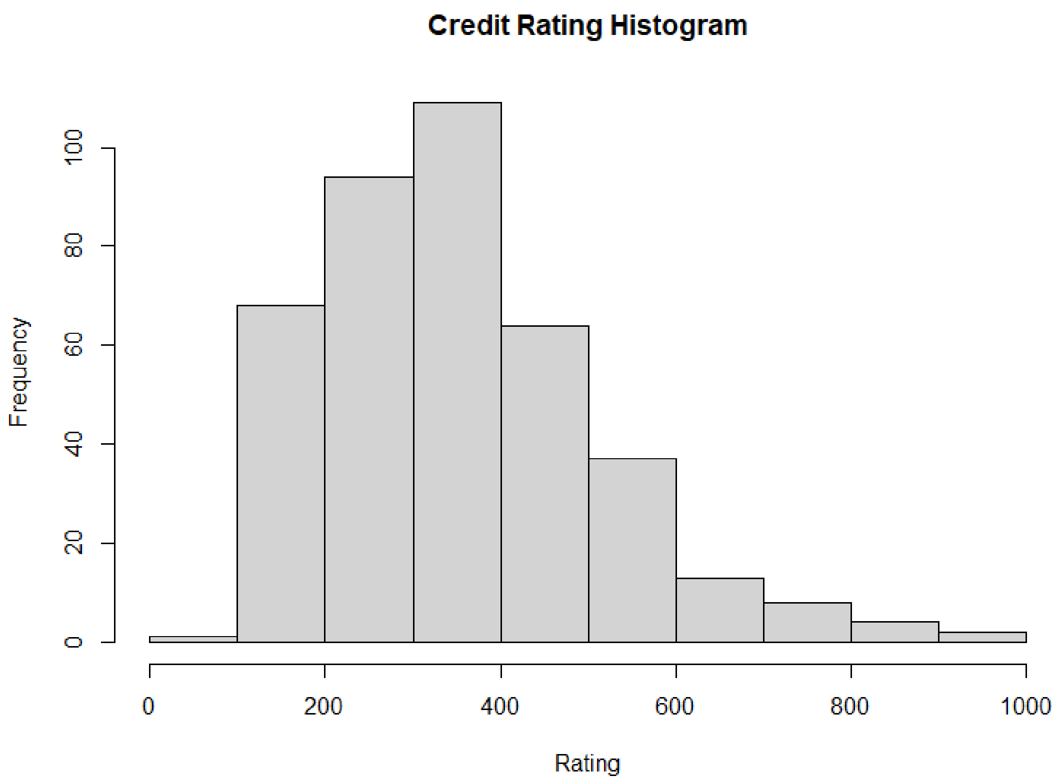



```
boxplot(Credit$Income ~ Credit$Gender)
```

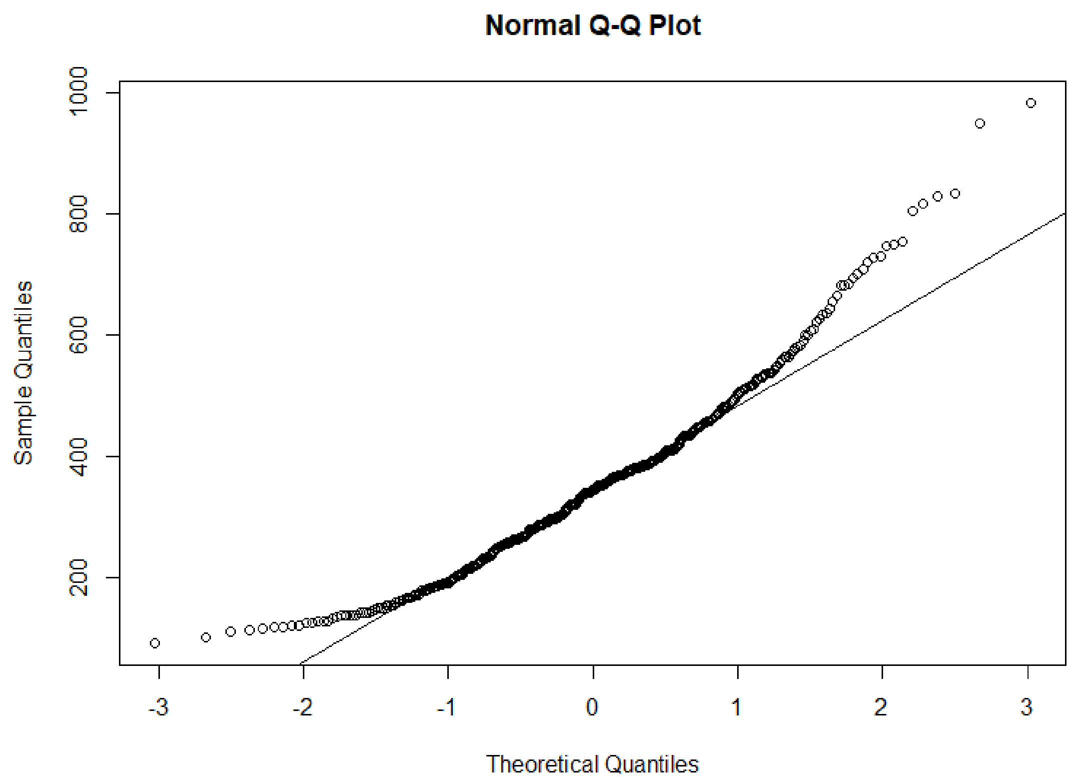


3.3 Display a histogram for the variable **Rating**, with the main title "Credit Rating Histogram" (`main=`) and X label "Rating" (`xlab=`). Then draw a QQ Plot for **Rating** (Tip: use the `qqnorm()` function first to draw the data points and then use the `qqline()` function to layer the QQ Line on top).

```
hist(Credit$Rating, main="Credit Rating Histogram", xlab="Rating")
```




```
qqnorm(Credit$Rating)
qqline(Credit$Rating)
```



3.4 Briefly answer **in your own words**: Do you think that this data is somewhat normally distributed? Why or why not? In your answer, please refer to both, the Histogram and the QQ Plot.

```
# The data is somewhat normal in the middle, but the qqplot deviates from the
qqline providing some indication of non-normality at the tails. The histogram
shows some skewness to the right indicating some departure from normality,
but it has a bell shape in the center of the data, which is consistent with
the QQ Plot.
```

4. Basic Predictive Analytics

4.1 First, enter the command `options(scipen = 4)` to minimize the display values with scientific notation. Then, create a simple linear regression model object with the `lm()` function to fit credit **Rating** as a function of **Income** and save the results in an object named **lm.rating**. Then display the model summary results with the `summary()` function. Tip: use the formula `Rating ~ Income, data = Credit` inside the `lm()` function.

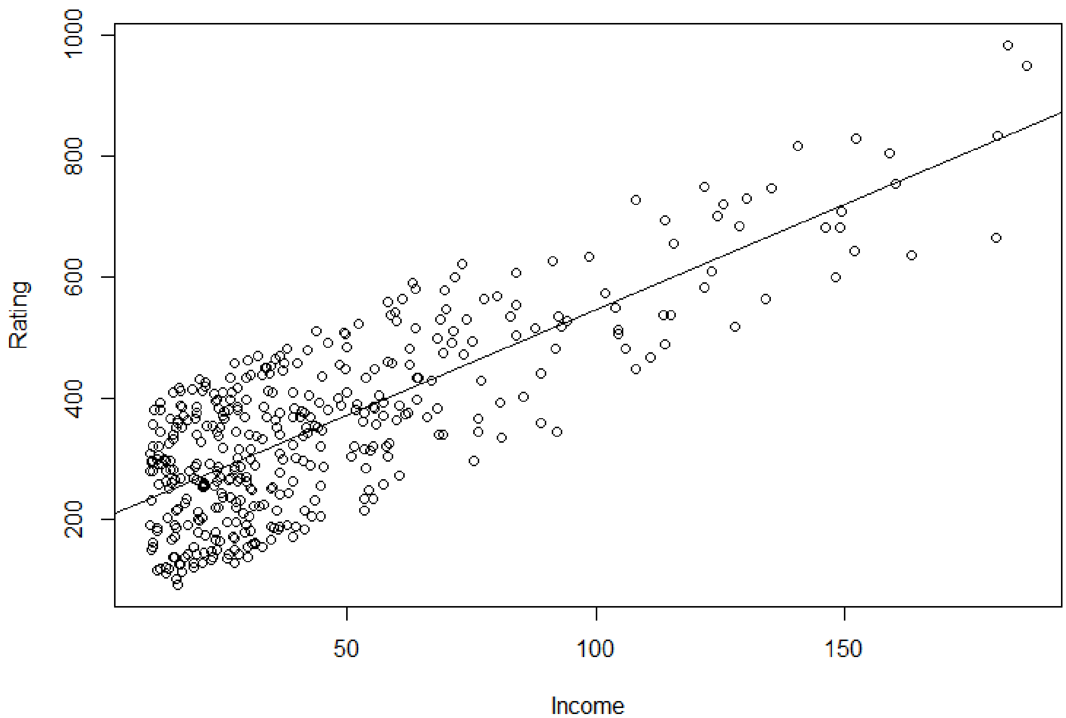
```
options(scipen = 4)
lm.rating <- lm(Rating ~ Income, data = Credit)
summary(lm.rating)

##
## Call:
## lm(formula = Rating ~ Income, data = Credit)
```

```
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -173.855  -79.417   -0.384   79.747  171.955
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 197.8411     7.7089   25.66   <2e-16 ***
## Income        3.4742     0.1345   25.83   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 94.71 on 398 degrees of freedom
## Multiple R-squared:  0.6263, Adjusted R-squared:  0.6253
## F-statistic:   667 on 1 and 398 DF,  p-value: < 2.2e-16
```

4.2 Now, plot Credit Rating (Y axis) against Income (X axis), with respective labels "Income" and "Credit Rating". Tip: feed the same formula you used in the `lm()` function above, but using the `plot()` function instead. Then draw a regression line by feeding **lm.rating** into the `abline()` function.

Note: see how I added the `fig.width` and `fig.height` parameters in the `{r` code chunk header below to control the size of the figures.

```
plot(Rating ~ Income, data = Credit)
abline(lm.rating)
```



4.3 Write a simple linear model to predict credit ratings using these predictors: **Income**, **Limit**, **Cards**, **Married** and **Balance**. Name the resulting model **lm.rating.5**. Then display the regression using the `summary()` function. No need to answer, but what do you think are the most influential predictors of credit rating?

```
lm.rating.5 <- lm(Rating ~ Income + Limit + Cards +
                           Married + Balance,
                  data = Credit)

summary(lm.rating.5)

##
## Call:
## lm(formula = Rating ~ Income + Limit + Cards + Married + Balance,
##     data = Credit)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -24.0051  -7.0024  -0.9291   6.3789  26.2751
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 27.1070066  2.1867611  12.396  < 2e-16 ***
## Income       0.0975008  0.0335195   2.909  0.00383 **
## Limit        0.0641536  0.0009004  71.247  < 2e-16 ***
```

```
## Cards        4.7108256  0.3762419  12.521  < 2e-16 ***
## MarriedYes   2.1217503  1.0441007   2.032  0.04281 *
## Balance      0.0084355  0.0031308   2.694  0.00735 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.14 on 394 degrees of freedom
## Multiple R-squared:  0.9958, Adjusted R-squared:  0.9957
## F-statistic: 1.85e+04 on 5 and 394 DF,  p-value: < 2.2e-16
```

```
# All predictors are statistically significant (i.e., they have asterisks
next to them and the p-values are smaller than 0.05). Also, all predictors
are positive, so they all have a positive influence on credit rating. Limit
and Cards are the most significant and the number of Cards seems to have the
strongest effect.
```