STAT GR 4206/5206 Homework_2

Christine Chong cc4190

May 31, 2017

Part 1 : Loading and Cleaning the Data in R

i. Load the data into a dataframe called housing.

Here we can use read.csv to obtain the data. Header is set to true because there is a row with headers. Seperation is set to "," to indicate to R that each line of the file would be seperated with ",".

```
housing <- read.csv("NYChousing.csv", header=TRUE, sep = ",")
```

ii. How many rows and columns does the dataframe have?

Here we can use nrow and ncol with housing as a parameter.

```
nrow(housing)
```

```
## [1] 2506
```

```
ncol(housing)
```

```
## [1] 22
```

iii. Run this command, and explain. in words, what this does :

```
apply(is.na(housing),2,sum)
```

```
##                              UID                     PropertyName
##                                0                                0
##                              Lon                              Lat
##                               15                               15
##                         AgencyID                             Name
##                                0                                0
##                            Value                          Address
##                               52                                0
##                   Violations2010                      REACNumber
##                                0                             1873
##                         Borough                               CD
##                                0                                0
##              CityCouncilDistrict                      CensusTract
##                               10                                0
##                    BuildingCount                        UnitCount
##                                0                                0
##                        YearBuilt                            Owner
##                                0                                0
##                     Rental.Coop                OwnerProfitStatus
##                                0                                0
##       AffordabilityRestrictions StartAffordabilityRestrictions
##                                0                                5
```

Is.na is a function that returns true if there is missing data. So is.na(housing) and is asking the program to look for missing values in the housing dataframe. The apply function here is asking the program, for each column, to run the is.na(housing) and add up all the missing values it finds.

iv. Remove the rows of the dataset for which the variable Value is NA.

Here we can create a new vector with rows that only have values for housing using **!is.na(housing$Value) == TRUE**.

```
newHousing <- housing[!(is.na(housing$Value) == TRUE),]
```

v. How many rows did you remove with the previous call? Does this agree with your result from (iii)?

Here we can see the difference in the rows by comparing the number of rows from the original dataset and the modified dataset

```
removed <- nrow(housing) - nrow(newHousing)
removed
```

```
## [1] 52
```

52 rows were removed with the previous call. This does agree with the results from iii.

vi. Create a new variable in the dataset called logValue that is equal to the logarithm of the property's Value. What are the minimum, median, mean, and maximum values of logValue?

Here we can use the **log** function to find the logs of the value of the property and then use the **min**, **median**, **mean**, and **max** function to find the minimum, median, mean and maximum values of these log values.

```
logValue <- log(newHousing$Value)
minLogValue <- min(logValue)
medLogValue <- median(logValue)
meanLogValue <- mean(logValue)
maxLogValue <- max(logValue)
minLogValue
```

```
## [1] 8.410053
```

```
medLogValue
```

```
## [1] 13.74818
```

```
meanLogValue
```

```
## [1] 13.6823
```

```
maxLogValue
```

```
## [1] 20.47182
```

vii. Create a new variable in the dataset called logUnits that is equal to the logarithm of the number of units in the property. The number of units in each piece of property is stored in the variable UnitCount.

Here we can use the **log** function to find the logs of the number of the units in the property.

```
logUnits <- log(newHousing$UnitCount)
```

viii. Finally create a new variable in the dataset called after1950 which equals TRUE if the property was built in or after 1950 and FALSE otherwise. You'll want to use the YearBuilt variable here. This can be done in a single line of code.

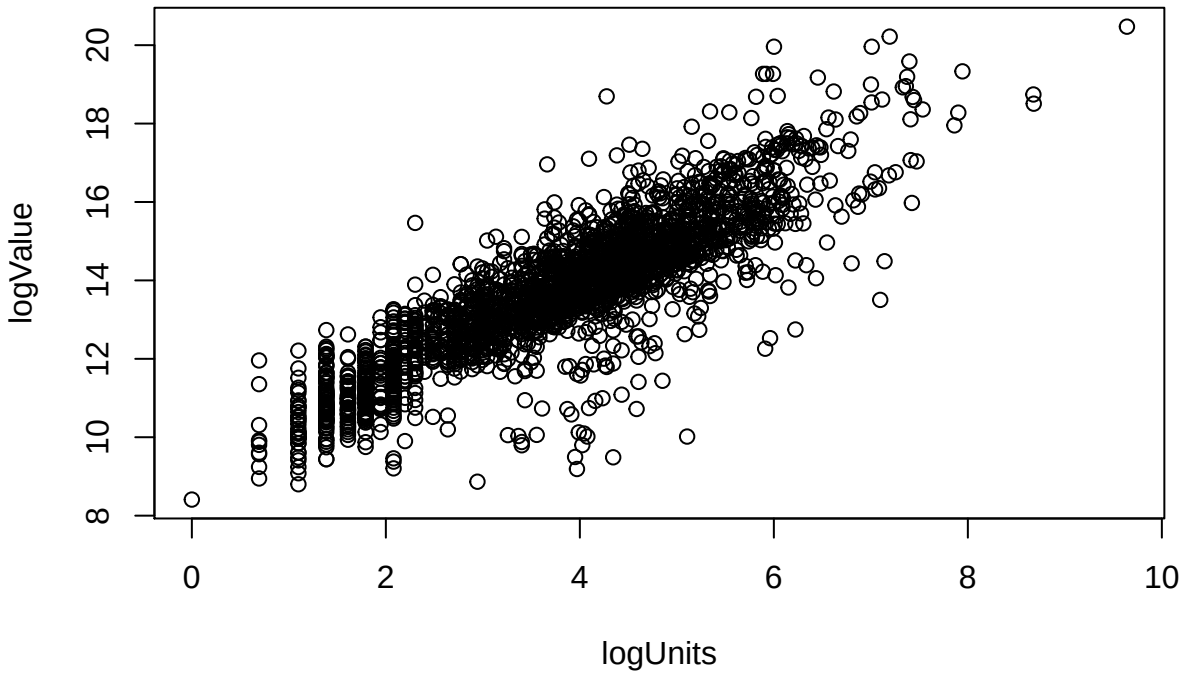
Here we can use an **ifelse** statement to create a column of **TRUES** and **FALSES** to indicated whether the house was built in or after 1950.

```
housing$after1950 <- ifelse(housing$YearBuilt >= 1950, TRUE, FALSE)
```

Part 2: EDA

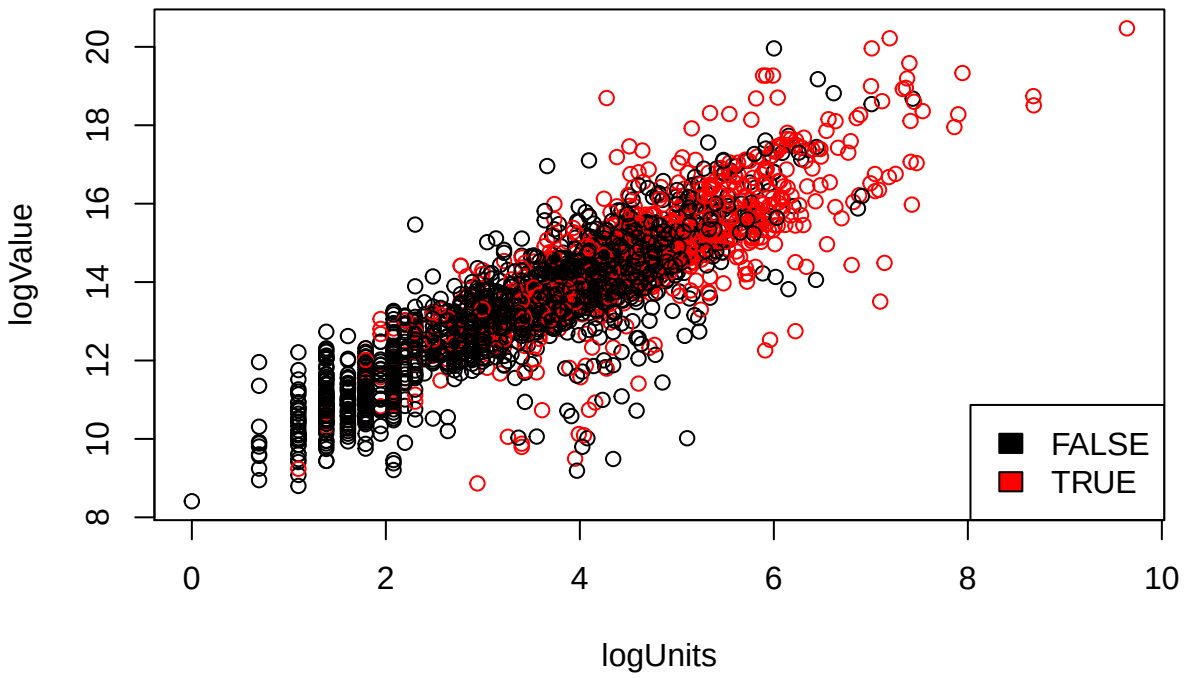
i.Plot property logValue against property logUnits. Name the x and y labels of the plot appropriately. logValue should be on the y-axis. Here we can use the **plot** function, mapping the logUnits and logValue.

```
plot(logUnits, logValue, xlab = "logUnits",
     ylab = "logValue")
```



ii. Make the same plot as above, but now include the argument col = factor(housing$after1950). Describe this plot and the covariation between the two variables. What does the coloring in the plot tell us? Here we can add and graph another factor to the map using the **TRUE** and **FALSE** from evaluating whether a house was built in or after 1950 .

```
plot(logUnits, logValue, xlab = "logUnits",
     ylab = "logValue",
     col = factor(housing$after1950))
legend("bottomright", legend = levels(factor(housing$after1950)), fill
= unique(factor(housing$after1950)))
```



The coloring on the plots tell us that the houses represented by black circles were built before 1950 and the houses represented by red circles were built after 1950.

iii. The cor() function calculates the correlation coefficient between two variables. What is the correlation between property logValue and property logUnits in (i) the whole data, (ii) just Manhattan (iii) just Brooklyn (iv) for properties built after 1950 (v) for properties built before 1950?

For the whole data :

```
cor(logUnits, logValue)
```

```
## [1] 0.8727348
```

For Manhattan :

```
manHousing <- newHousing[newHousing$Borough == "Manhattan", ]
logManUnits <- log(manHousing$UnitCount)
logManValue <- log(manHousing$Value)
cor(logManUnits, logManValue)
```

```
## [1] 0.8830348
```

For Brooklyn :

```
brookHousing <- newHousing[newHousing$Borough == "Brooklyn", ]
logBrookUnits <- log(brookHousing$UnitCount)
logBrookValue <- log(brookHousing$Value)
cor(logBrookUnits, logBrookValue)
```

```
## [1] 0.9102601
```

Properties before 1950 :

```
before1950Housing <- newHousing[newHousing$YearBuilt < 1950, ]
logBefore1950Units <- log(before1950Housing$UnitCount)
logBefore1950Value <- log(before1950Housing$Value)
cor(logBefore1950Units, logBefore1950Value)
```

```
## [1] 0.8643297
```

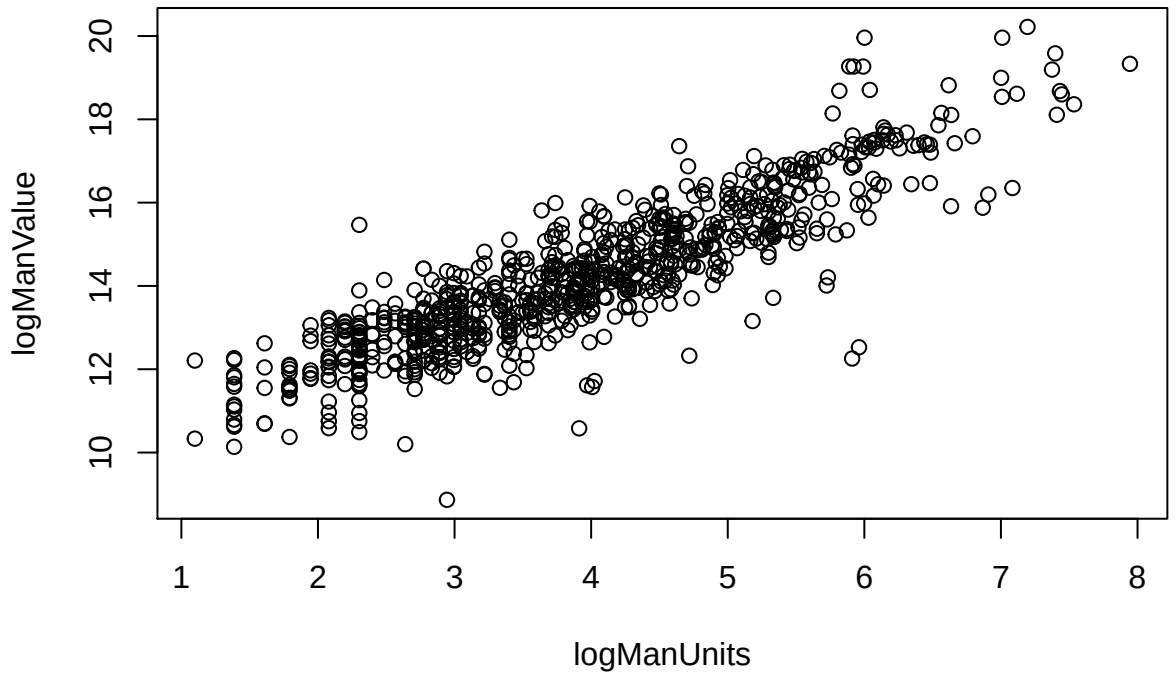
Properties after 1950 :

```
after1950Housing <- newHousing[newHousing$YearBuilt >= 1950, ]
logAfter1950Units <- log(after1950Housing$UnitCount)
logAfter1950Value <- log(after1950Housing$Value)
cor(logAfter1950Units, logAfter1950Value)
```

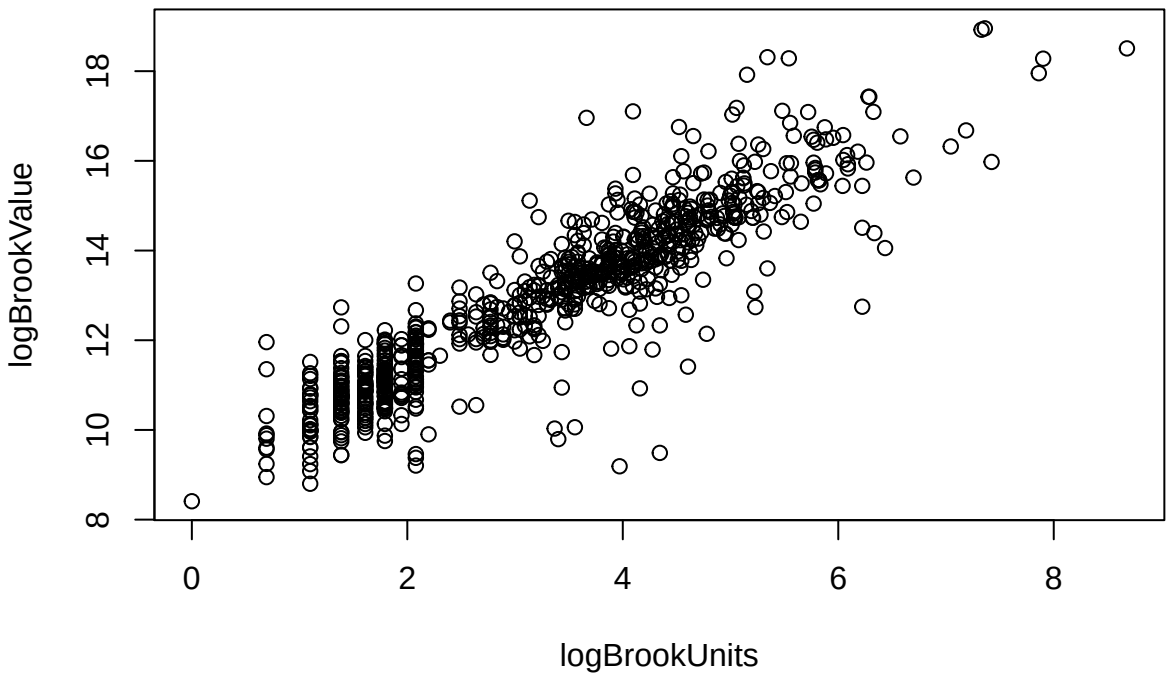
```
## [1] 0.721735
```

iv. Make two plots showing property logValue against property logUnits for Manhattan and Brooklyn. (If you can fit the information into one plot, clearly distinguishing the two boroughs, that's OK too.) Here we can use the plot function again with their respective parameters.

```
plot(logManUnits, logManValue, xlab = "logManUnits",
     ylab = "logManValue")
```




```
plot(logBrookUnits, logBrookValue, xlab = "logBrookUnits",
     ylab = "logBrookValue")
```



v. Consider the following block of code. Give a single line of R code which gives the same final answer as the block of code. There are a few ways to do this. Here we can use the **tapply** function with the parameters value and borough to the newhousing subset(to remove the NAs). Then we can use the median function to find the median of housing prices between those in Manhattan, and those not in Manhattan.

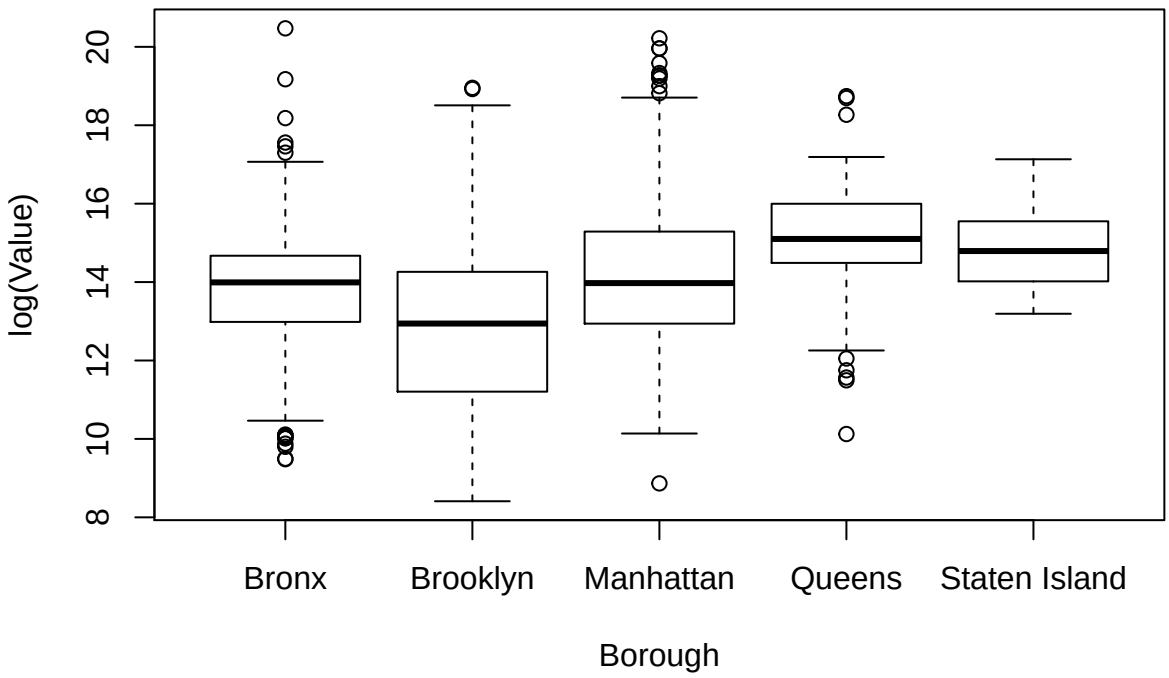
```
tapply(newHousing$Value, newHousing$Borough == "Manhattan", median)
```

```
##    FALSE    TRUE
##   823760 1172362
```

vi. Make side-by-side box plots comparing property logValue across the five boroughs.

Here we can use the **boxplot** function with the parameters of the log of the Values and the Boroughs in order to compare the property logValues.

```
boxplot(log(Value) ~ Borough, data=newHousing, ylab="log(Value)", xlab="Borough")
```



vii. For five boroughs, what are the median property values? (Use Value here, not logValue.)

Here we can use the **tapply** function to find the medians of the values of the houses per borough.

```
tapply(newHousing$Value, newHousing$Borough, median)
```

```
##         Bronx      Brooklyn     Manhattan        Queens Staten Island
##       1192950        417610       1172362       3611700       2654100
```