Homework 1 Solutions

Cynthia Rush (cgr2130)

October 2, 2017

Part 1

i. Since **properties** is a **.csv** file I use **read.csv()** to import the data into R.

```
setwd("/Users/cynthiarush/Dropbox/Stats_Comp_2017/Homework/HW1")
housing <- read.csv("properties.csv", as.is = TRUE)
```

ii. The function **dim()** provides the dimension of its input object.

```
orig_dim <- dim(housing)
orig_dim
```

```
## [1] 16319    17
```

iii.

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

```
##       cartodb_id              bbl         tract_10         sba_name
##                0                0                0                0
##         ccd_name          cd_name        boro_name        city_name
##                0                0                0                0
##   tax_delinquency    ser_violation   assessed_value       owner_name
##                0                0                0                0
##        res_units       year_built        buildings standard_address
##              504              253              319                0
##   applied_filters
##                0
```

The command **is.na(housing)** creates a matrix of the same dimensions as **housing** with each element being TRUE or FALSE depending on whether or not the corresponding element in **housing** is an NA value. Then the full call **apply(is.na(housing), 2, sum)** counts the number of NA values each column of *housing*.

iv.

```
housing <- housing[housing$assessed_value != 0, ]
```

The call **housing$assessed_value != 0** returns a logical vector with TRUE where **housing$assessed_value** doesn't equal 0, therefore I filter using **housing$assessed_value != 0** to get only the rows where **assessed_value** doesn't equal 0. I reassign my **housing** dataframe, to be the filtered dataframe.

v.

```
new_dim <- dim(housing)
orig_dim[1] - new_dim[1]
```

```
## [1] 66
```

I removed 66 rows of my dataframe.

v.

```
housing$logValue <- log(housing$assessed_value)
summary(housing$logValue)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   5.878  12.480  13.250  13.480  14.350  20.030
```

vi.

```
housing$logUnits <- log(housing$res_units)
```

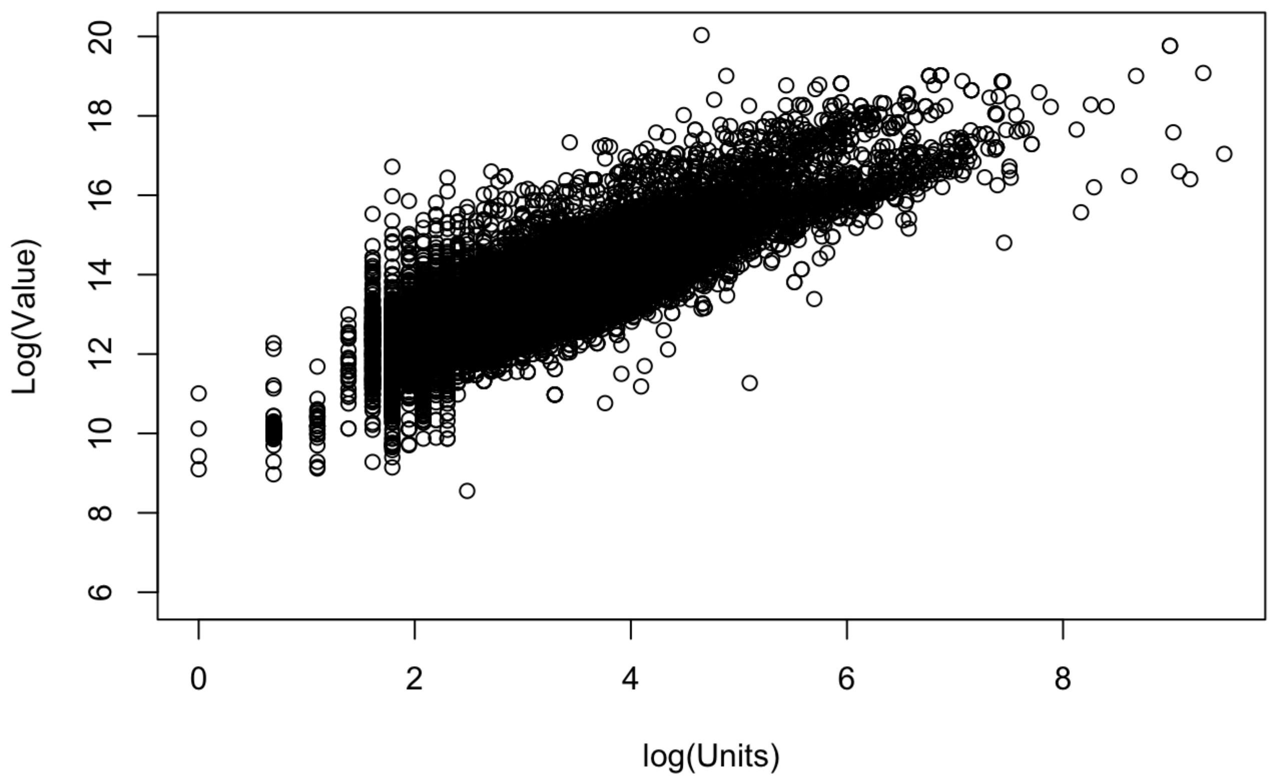
vii.

```
housing$after2000 <- housing$year_built >= 2000
```

Part 2: EDA

i.

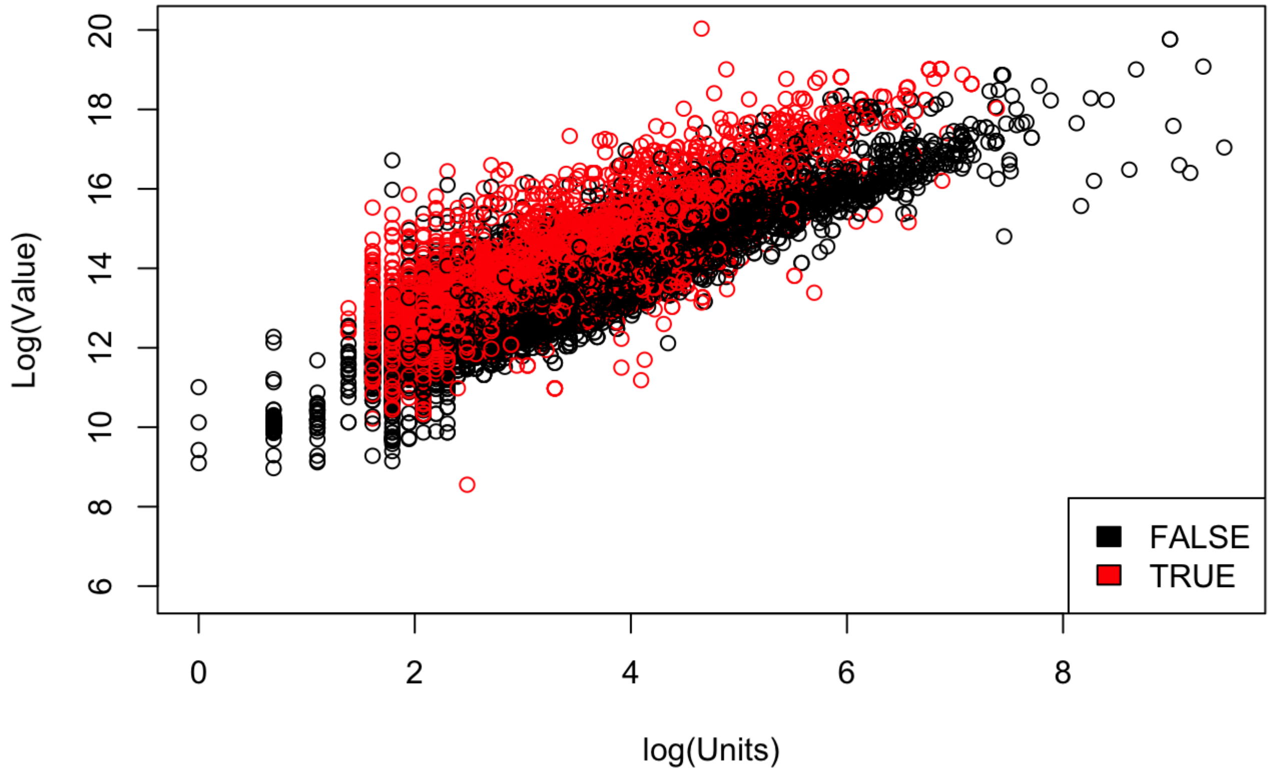
```
plot(housing$logUnits, housing$logValue, xlab = "log(Units)", ylab = "Log(Value)")
```



I plot a scatterplot with the **plot()** command and add argument **xlab =** and **ylab =** for the labels.

ii.

```
plot(housing$logUnits, housing$logValue, col = factor(housing$after2000), xlab = "log(Units)"
, ylab = "Log(Value)")
legend("bottomright", legend = levels(factor(housing$after2000)), fill = unique(factor(housin
g$after2000)))
```



There appears to be a pretty strong linear reltionship between **logValue** and **logUnits**. When colored according to the **after2000** variable, it is clear that newer buildings (those built after 2000) tend to be more expensive and have more units than older buildings.

iii.

```
cor(housing$logValue, housing$logUnits, use = "pairwise.complete.obs")
```

```
## [1] 0.8431877
```

```
cor(housing$logValue[housing$boro_name == "Manhattan"], housing$logUnits[housing$boro_name ==
 "Manhattan"], use = "pairwise.complete.obs")
```

```
## [1] 0.8592745
```

```
cor(housing$logValue[housing$boro_name == "Brooklyn"], housing$logUnits[housing$boro_name ==
"Brooklyn"], use = "pairwise.complete.obs")
```

```
## [1] 0.8579328
```

```
cor(housing$logValue[housing$after2000], housing$logUnits[housing$after2000], use = "pairwis
e.complete.obs")
```

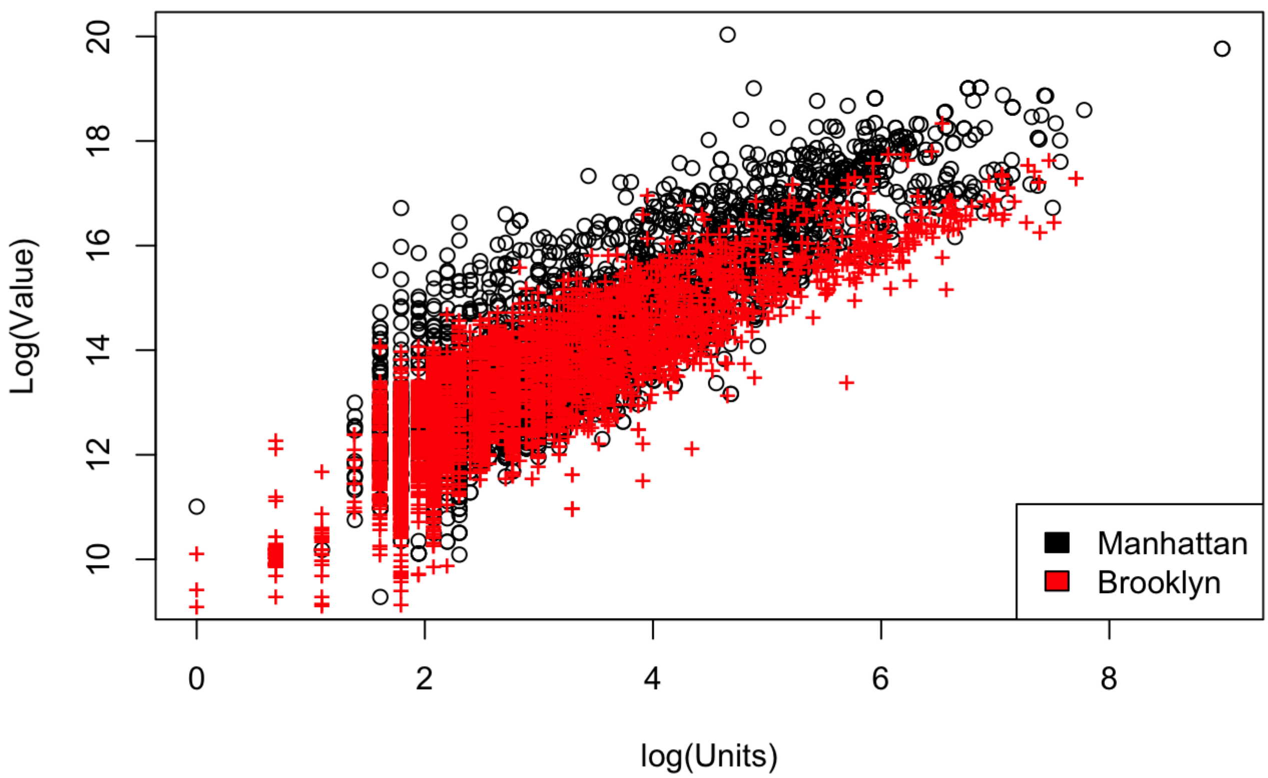
```
## [1] 0.8337845
```

```
cor(housing$logValue[!housing$after2000], housing$logUnits[!housing$after2000], use = "pairwi
se.complete.obs")
```

```
## [1] 0.8927153
```

iv.

```
plot(housing$logUnits[housing$boro_name == "Manhattan"], housing$logValue[housing$boro_name =
= "Manhattan"], xlab = "log(Units)", ylab = "Log(Value)")
points(housing$logUnits[housing$boro_name == "Brooklyn"], housing$logValue[housing$boro_name
== "Brooklyn"], col = "red", pch = "+")
legend("bottomright", legend = c("Manhattan", "Brooklyn"), fill = c("black", "red"))
```



v.

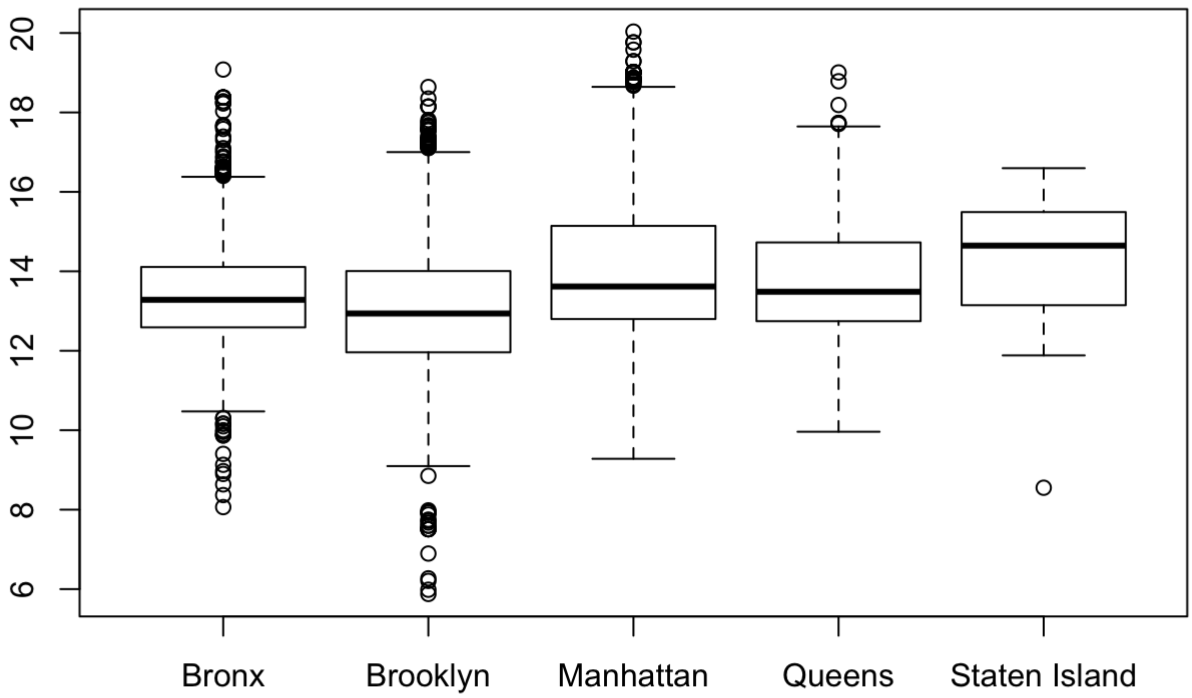
```
median(housing$assessed_value[housing$boro_name == "Manhattan"])
```

```
## [1] 820350
```

The code calculates the median property value for all properties in Manhattan.

vi.

```
boxplot(housing$logValue ~ housing$boro_name)
```



vii.

```
tapply(housing$assessed_value, housing$boro_name, median)
```

```
##         Bronx      Brooklyn     Manhattan        Queens Staten Island
##        587250        416014        820350        719100       2296350
```

We use **tapply()** which splits the property value into groups based on **boro_name** and then calculated the median within each group.