Home Work One

Yi (Chris) Chen

September 19, 2017

Home Work One

- UNI: yc3356
- Name: YI CHEN
- Email: yc3356@columbia.edu

part 1: Loading and Cleaning in Data in R

i. load the data into a dataframe called housing

```
setwd("C:/Users/cheny/Desktop/study/statistical computing and intro to data science/homework")
housing <- read.csv('properties.csv')
```

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

```
total_row <- nrow(housing)
total_col <- ncol(housing)
paste('there are',as.character(total_row),'rows')
```

```
## [1] "there are 16319 rows"
```

```
paste('there are',as.character(total_col),'columns')
```

```
## [1] "there are 17 columns"
```

iii. Run this command, and explain, in words, what this does: apply(is.na(housing), 2, sum)

```
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
```

- explain:

1. **is.na** is the function indicates which elements are missing. If the element is missing it return TRUE otherwise it return FALSE. And *is.na(housing)* return a dataframe with same shape while the elements are all TRUE or FALSE.
2. **apply** is a function returns a list of values obtained by applying a function to margins of an array or matrix. In this example, **2** means the function will be applied over **columns**. And sum mean the function that will be applied.
3. When apply the sum function on TRUE & FALSE, the **TRUE is read as 1 while FALSE is read as 0**.
4. By this code, **we can find out how many missing elements exist in every column in the housing dataframe**.

iv. Remove the rows of the dataset for which the variable assessed valueequals 0.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
housing <- filter(housing,housing$assessed_value!=0)
# test the result
table(housing$assessed_value==0)
```

```
##
## FALSE
## 16253
```

- explain:

1. **dplyr** is a package provides a flexible grammar of dataframe manipulation.
2. **filter** is a function pick out the rows which fulfill the requirement. The first parameter is the data, the second parameter is the requirement.
3. as you can see, testing part show that none of the elements in the assessed_value equals to 0.

v. How many rows did you remove with the previous call?

```
number_of_removed_rows <- total_row - nrow(housing)
paste('there are',as.character(number_of_removed_rows),'rows have been
removed')

## [1] "there are 66 rows have been removed"
```

vi. Create a new variable in the dataset called logValue that is equal to the logarithm of

the property's assessed value. What are the **minimum, median, mean, and maximum** values of logValue?

```
housing$logValue <- log(housing$assessed_value)
housing %>%
        summarise(minimum = min(housing$logValue),
        median = median(housing$logValue),
        mean = mean(housing$logValue),
        maximum = max(housing$logValue))

##    minimum  median     mean  maximum
## 1 5.877736 13.2497 13.48347 20.03494
```

- explain:

1. **summarize** is a function in the **dplyr** package.

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 res units.

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

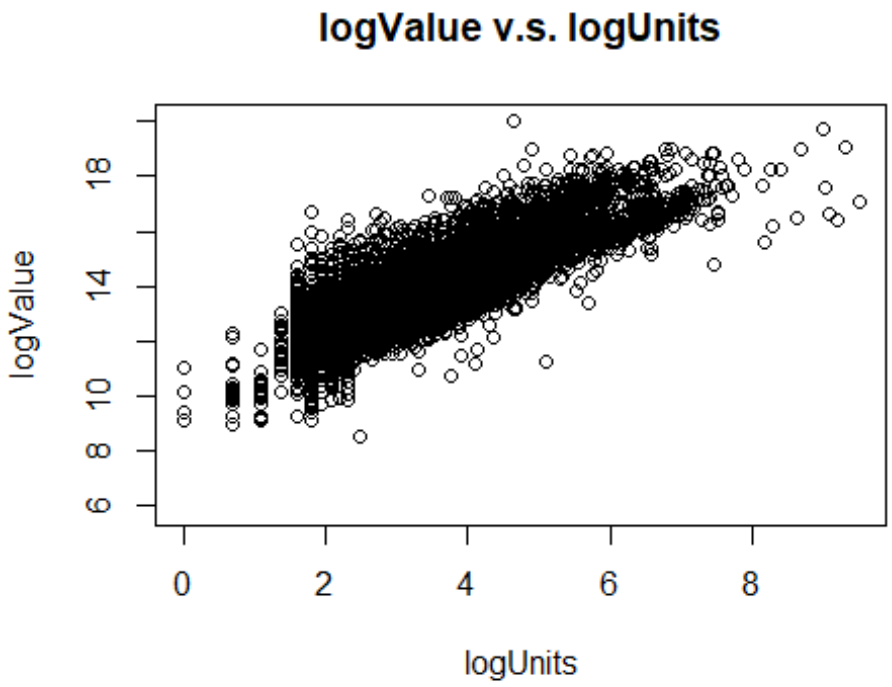
viii. Finally create a new variable in the dataset called *after2000 which equals TRUE if the property was built in or after 2000 and FALSE otherwise. You'll want to use the year built variable here. This can be done in a single line of code.

```
housing$after2000 <- as.numeric(housing$year_built) >= 2000
```

Part 2:EDA

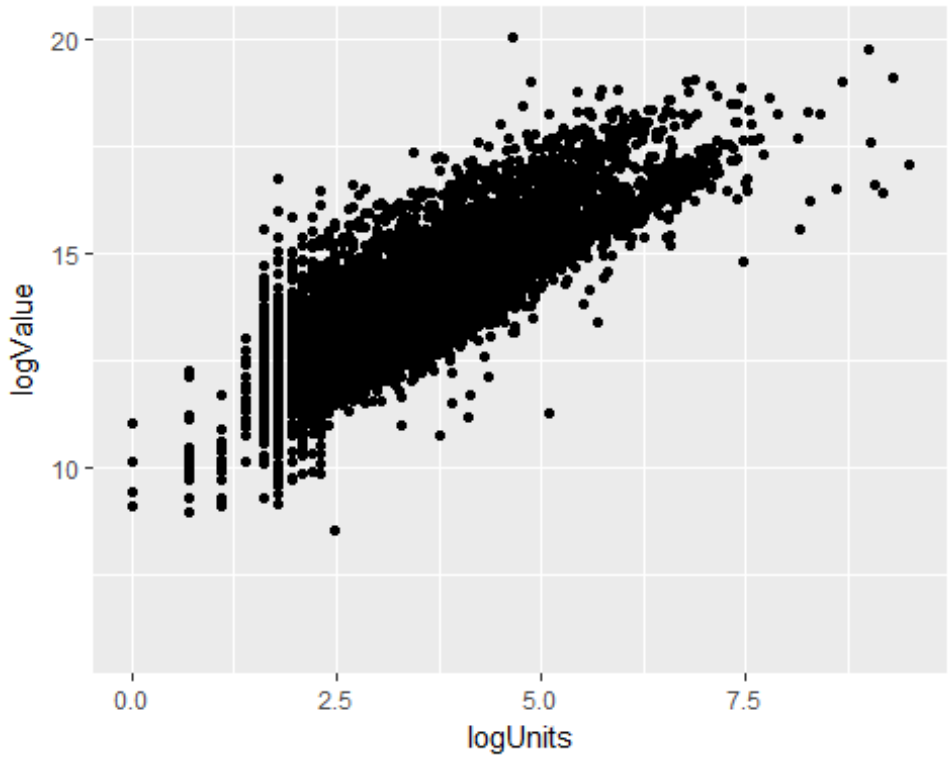
i. Plot property logValue against property logUnits. Name the x and y labels of the plot appropriately. logValue should be on the y-axis.

```
plot(y = housing$logValue,x = housing$logUnits, ylab = 'logValue',xlab =
'logUnits',main = 'logValue v.s. logUnits')
```



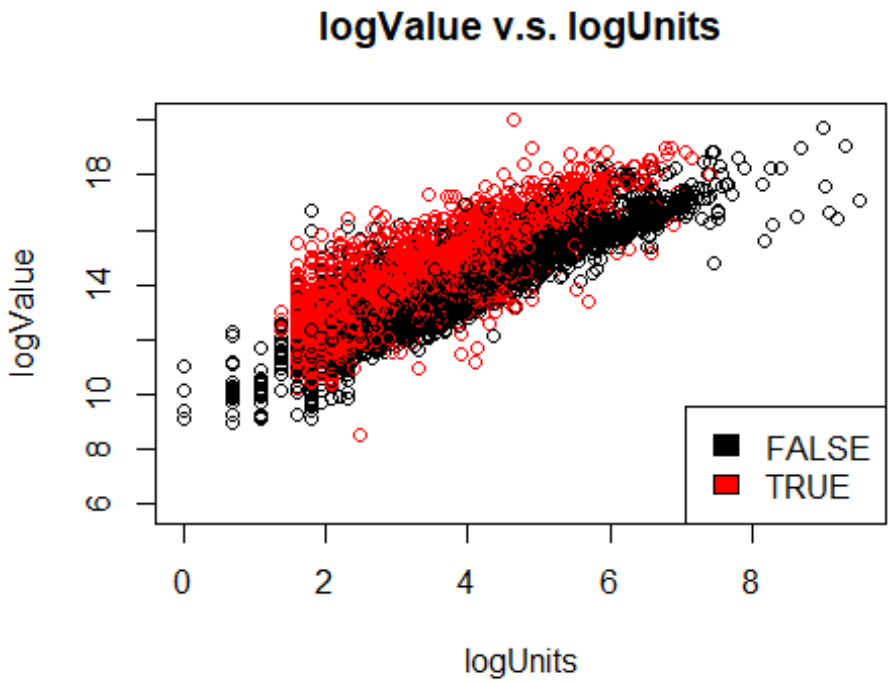
- also can use ggpolt2 package to draw the picture

```
library(ggplot2)
picture_one <- ggplot(data = housing, mapping = aes(y = logValue,x =
logUnits))             + geom_point()
picture_one

## Warning: Removed 447 rows containing missing values (geom_point).
```



ii. Make the same plot as above, but now include the argument col = factor(housing$after2000). Describe this plot and the covariation between the two variables. What does the coloring in the plot tell us?

```
plot(y = housing$logValue,x = housing$logUnits, ylab = 'logValue',xlab =
'logUnits',main = 'logValue v.s. logUnits',col = factor(housing$after2000))
legend("bottomright", legend = levels(factor(housing$after2000)), fill
= unique(factor(housing$after2000)))
```



- exlpain:

1. As we can see in the plot, given a specific value of logUnits the value of logValue is much higher after 2000 than before 2000.
2. In reality, this means if two properities have the same number of units the properities which are builted later will have higher value. This is reasonable because, new properities are more likely to have a higher quality or nicer apperance. People like the new properities more, thus the price are tends to be higher.

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 2000 (v) for properties built before 2000? You will need to add the argument use = "pairwise.complete.obs" to handle NA values.

```
# the whole data
whole <- housing %>% select(logValue,logUnits) %>%
summarise(correlation_of_whole = cor(logValue,logUnits,use =
"pairwise.complete.obs"))
print(whole)

##   correlation_of_whole
## 1            0.8431877

# just Manhattan
Manhattan <-filter(housing,housing$boro_name == 'Manhattan') %>%
select(logValue,logUnits) %>% summarise(correlation_of_Manhattan =
```

```r
cor(logValue,logUnits,use = "pairwise.complete.obs"))
print(Manhattan)
```

```
##   correlation_of_Manhattan
## 1                0.8592745
```

```r
# just Brooklyn
Brooklyn <-filter(housing,housing$boro_name == 'Brooklyn') %>%
select(logValue,logUnits) %>% summarise(correlation_of_Brooklyn =
cor(logValue,logUnits,use = "pairwise.complete.obs"))
print(Brooklyn)
```

```
##   correlation_of_Brooklyn
## 1               0.8579328
```

```r
# for properties built after 2000
after_2000 <-filter(housing,housing$after2000 == TRUE) %>%
select(logValue,logUnits) %>% summarise(correlation_after_2000 =
cor(logValue,logUnits,use = "pairwise.complete.obs"))
print(after_2000)
```

```
##   correlation_after_2000
## 1              0.8337845
```

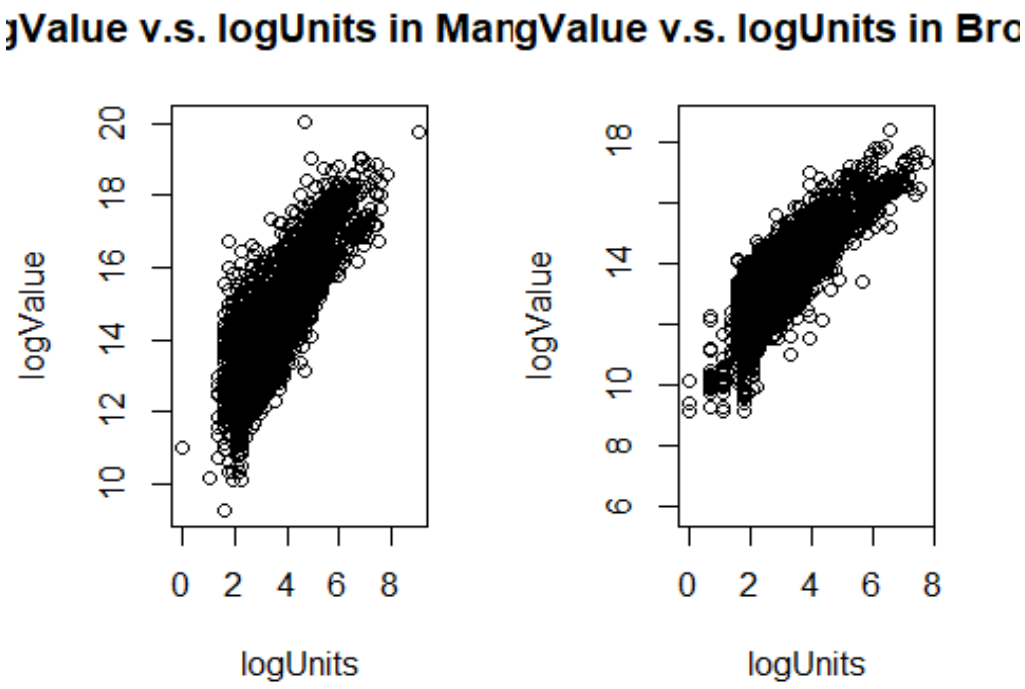
```r
#for properties built before 2000
befor_2000 <-filter(housing,housing$after2000 == FALSE) %>%
select(logValue,logUnits) %>% summarise(correlation_befor_2000 =
cor(logValue,logUnits,use = "pairwise.complete.obs"))
print(befor_2000)
```

```
##   correlation_befor_2000
## 1              0.8927153
```

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.

```r
#generate the data
Manhattan <-filter(housing,housing$boro_name == 'Manhattan') %>%
select(logValue,logUnits)

Brooklyn <-filter(housing,housing$boro_name == 'Brooklyn') %>%
select(logValue,logUnits)

par(mfrow=c(1,2)) ## draw two pictures together

plot(y = Manhattan$logValue,x = Manhattan$logUnits, ylab = 'logValue',xlab =
'logUnits',main = 'logValue v.s. logUnits in Manhattan')
```

```
plot(y = Brooklyn$logValue,x = Brooklyn$logUnits, ylab = 'logValue',xlab =
'logUnits',main = 'logValue v.s. logUnits in Brooklyn')
```



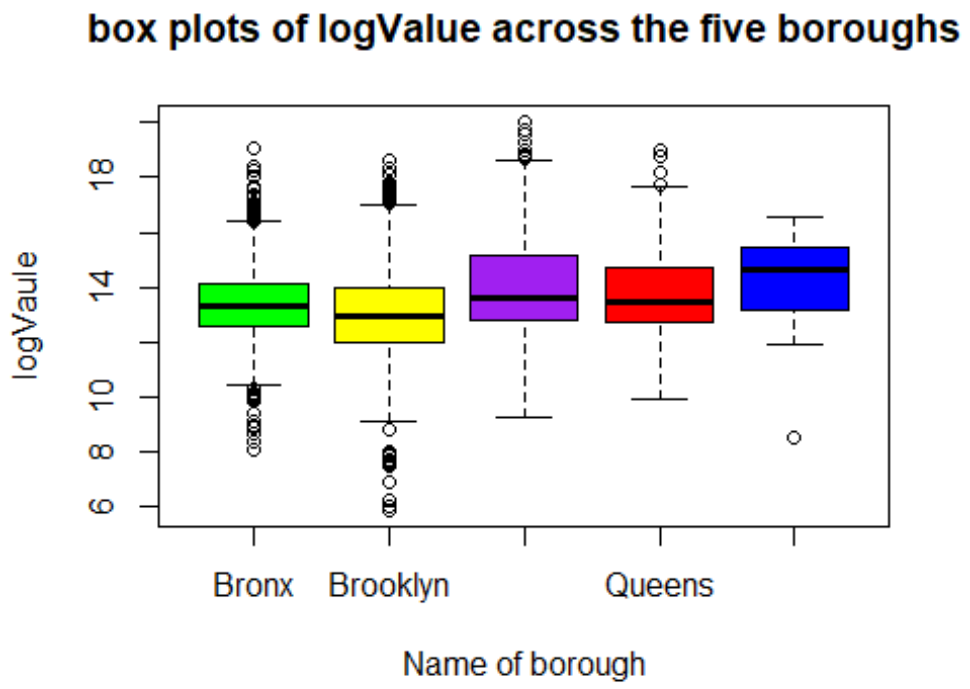
v. Consider the following block of code. Give a single line of R code which gives the same final answer as the block of code. There are a few ways to do this.

- explain: this code is aiming at finding all the assessed_value of 'Manhattan', and retrun the **median value of the assessed_value of Manhattan house**.

```
# solution one
print(filter(housing, housing$boro_name == 'Manhattan') %>%
select(assessed_value) %>% summarise( median_value =
median(assessed_value,na.rm=TRUE)))

##   median_value
## 1       820350

# solution two
paste('the median value is: ', as.character(median(housing[housing$boro_name
== 'Manhattan','assessed_value'],na.rm = TRUE)))

## [1] "the median value is:  820350"
```

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

```
# draw the pictures
boxplot(housing$logValue ~ housing$boro_name, data = housing,
   ylab = "logVaule",
   xlab = "Name of borough",
   main = "box plots of logValue across the five boroughs",
   col = c("green","yellow","purple",'red','blue')
)
```



vii. For five boroughs, what are the median property values? (Use assessed value here,

not logValue.

```
Manhattan <-filter(housing,housing$boro_name == 'Manhattan') %>%
select(assessed_value)
Bronx<-filter(housing,housing$boro_name == 'Bronx') %>%
select(assessed_value)
Brooklyn<-filter(housing,housing$boro_name == 'Brooklyn') %>%
select(assessed_value)
Queens<-filter(housing,housing$boro_name == 'Queens') %>%
select(assessed_value)
Staten_Island<-filter(housing,housing$boro_name == 'Staten Island') %>%
select(assessed_value)

Manhattan_median <- median(Manhattan$assessed_value)
Bronx_median <- median(Bronx$assessed_value)
Brooklyn_median <- median(Brooklyn$assessed_value)
```

```r
Queens_median <- median(Queens$assessed_value)
Staten_Island_median <- median(Staten_Island$assessed_value)

result <-
data.frame(Manhattan_median,Bronx_median,Brooklyn_median,Queens_median,Staten
_Island_median)
print(result)
```

```
##   Manhattan_median Bronx_median Brooklyn_median Queens_median
## 1           820350       587250          416014        719100
##   Staten_Island_median
## 1              2296350
```