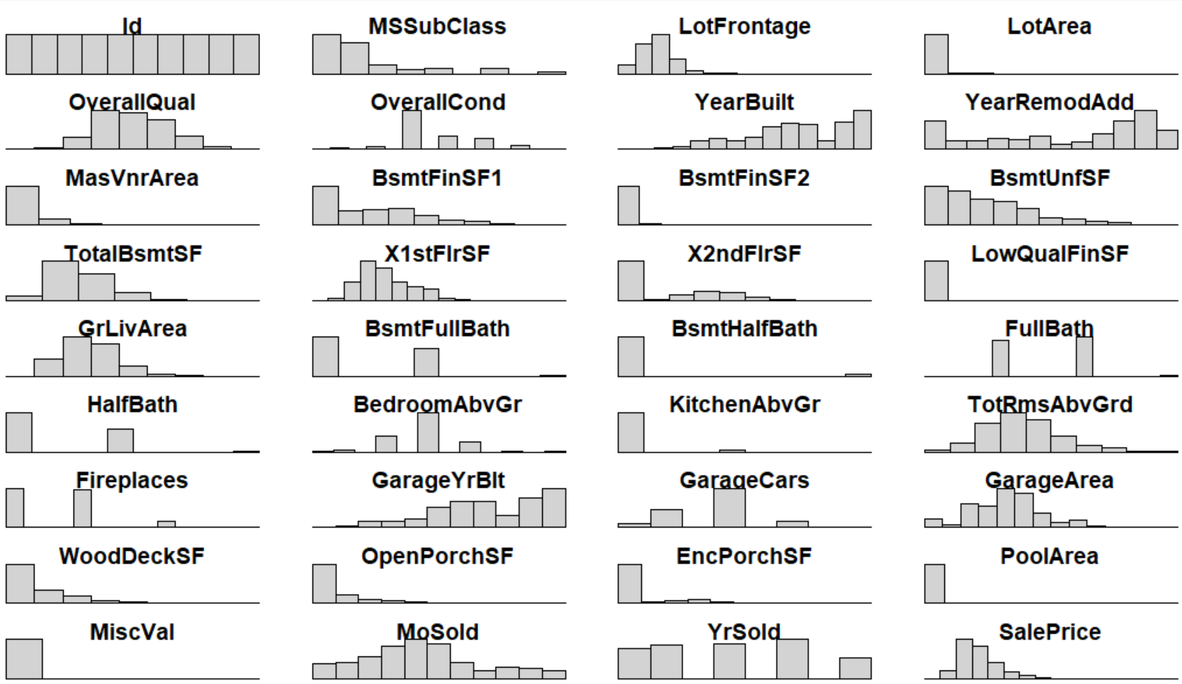
IDA Homework 4

Group 5

Bhavesh Kilaru/ Sujatha Sahu

1. Data Quality Report
   a.

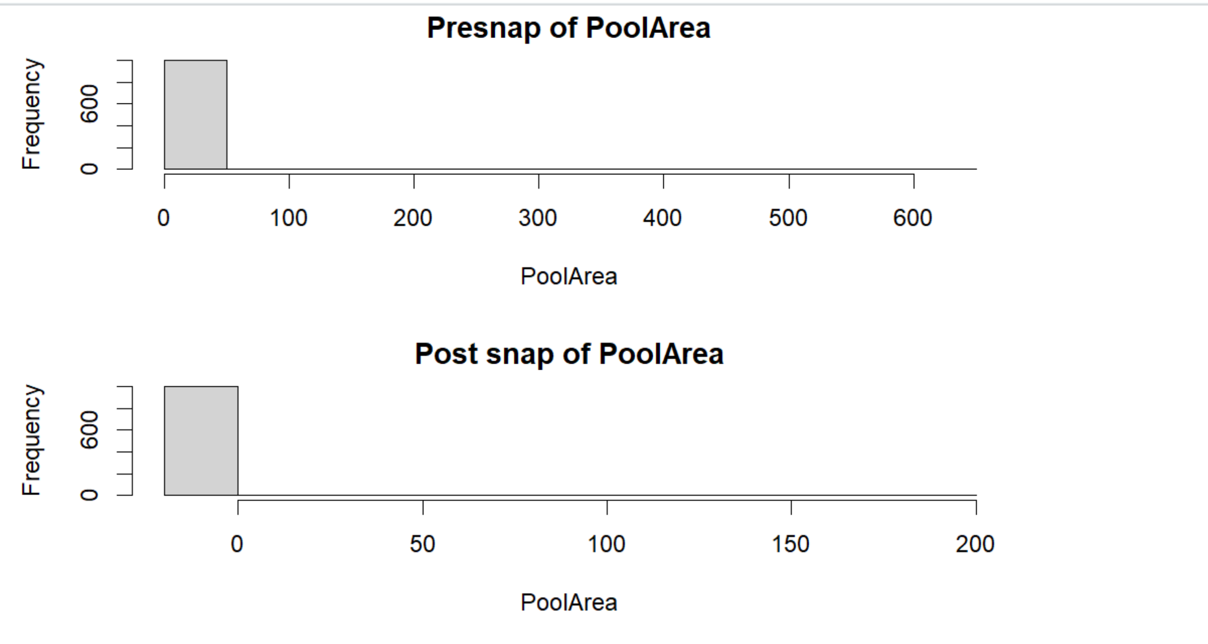
```
#### a
#### creating three new variables in housing data
```{r}
housingData <- housingData %>%
  dplyr::mutate(age = YrSold - YearBuilt,
  ageSinceRemodel = YrSold - YearRemodAdd,
  ageofGarage = YrSold - GarageYrBlt)
```
```

   b. Getting the numeric columns from the data

```
#### b
#### selecting the numeric columns from housingData data frame
```{r}
#getting the numeric columns
housingNumeric <- select_if(housingData, is.numeric) %>% as.tibble()
head(housingNumeric)
```
```

A tibble: 6 x 39

Id <int>	MSSubClass <int>	LotFrontage <int>	LotArea <int>	OverallQual <int>	OverallCond <int>	YearBuilt <int>	YearRemodAdd <int>
1	20	NA	11000	5	6	1966	1966
2	20	NA	36500	5	5	1964	1964
3	20	57	9764	5	7	1967	2003
4	70	NA	7500	6	7	1942	1950
5	20	80	9200	6	6	1965	1965
6	60	72	11317	7	5	2003	2003

6 rows | 1-8 of 39 columns

c. Getting the factor columns from the data

```
#### c
#### creating a tibble housingFactor that contains all factor variables of housing data
```{r}
housingFactor <- housingData %>% transmute(across(where(is.character), as.factor)) %>% as.tibble()
head(housingFactor)
```
```

A tibble: 6 x 36

MSZoning <fctr>	Alley <fctr>	LotShape <fctr>	LandContour <fctr>	LotConfig <fctr>	LandSlope <fctr>	Condition1 <fctr>	BldgType <fctr>
RL	NA	IR1	Lvl	CulDSac	Gtl	Norm	1Fam
RL	NA	IR1	Low	Inside	Mod	Norm	1Fam
RL	NA	IR1	Lvl	other	Gtl	Feedr	1Fam
RL	NA	IR1	Bnk	Inside	Gtl	Norm	1Fam
RL	NA	Reg	Lvl	Inside	Gtl	Norm	1Fam
RL	NA	Reg	Lvl	Inside	Gtl	Norm	1Fam

6 rows | 1-8 of 36 columns

d. Using glimpse()

```
#### d
#### using glimse to look at the created tibble
```{r}
glimpse(housingFactor)
```
```

```
Rows: 1,000
Columns: 38
```

e. Explanation of function Q1 and Q3

```
#### e
#### functions to extracting Q1 and Q3
```{r}
Q1<-function(x,na.rm=TRUE) {
quantile(x,na.rm=na.rm)[2]
}
Q3<-function(x,na.rm=TRUE) {
quantile(x,na.rm=na.rm)[4]
}
```
```

The above function Q1 returns the 25th percentile of the column 'x' after eliminating the NA's. Since the array index starts from 1 in R, the value at index 2 of the quantile function returns the 25th percentile or 1st quantile.

Function Q3 returns the 75th percentile or third quantile of the column 'x' after removing NA's from the column.

f. Creating the function myNumericSummary

````
#### f
#### creating the function myNumericSummary
```{r}
myNumericSummary <- function(x){
c(length(x), n_distinct(x), sum(is.na(x)), mean(x, na.rm=TRUE),
min(x,na.rm=TRUE), Q1(x,na.rm=TRUE), median(x,na.rm=TRUE), Q3(x,na.rm=TRUE),
max(x,na.rm=TRUE), sd(x,na.rm=TRUE))
}
```
````

g. Using dplyr::summarize command together with myNumericSummary along with across() function.

````
#### g
#### creating a tibble numericSummary that contains the result of myNumericSummary function
#### applied on each column of housingNumeric tibble
```{r}
numericSummary <- housingNumeric %>% summarise(across(.cols = everything(), ~myNumericSummary(.x)))
glimpse(numericSummary)
```
````

h. Adding labels to summary statistics

````
#### h
#### adding a column stats that names each row of numericSummary
```{r}
numericSummary <-cbind( stat=c("n","unique","missing","mean","min","Q1","median","Q3","max","sd"),
                        numericSummary)
glimpse(numericSummary)
```
````

````
Rows: 10
Columns: 40
$ stat           <chr> "n", "unique", "missing", "mean", "min", "Q1", "median", "Q3", "max",~
$ Id             <dbl> 1000.0000, 1000.0000, 0.0000, 500.5000, 1.0000, 250.7500, 500.5000, 7~
$ MSSubClass     <dbl> 1000.00000, 13.00000, 0.00000, 57.18500, 20.00000, 20.00000, 50.00000~
$ LotFrontage    <dbl> 1000.00000, 102.00000, 207.00000, 68.74527, 21.00000, 58.00000, 68.00~
$ LotArea        <dbl> 1000.000, 760.000, 0.000, 10424.881, 1477.000, 7500.000, 9422.000, 11~
$ OverallQual    <dbl> 1000.000000, 10.000000, 0.000000, 5.979000, 1.000000, 5.000000, 6.000~
$ OverallCond    <dbl> 1000.000000, 8.000000, 0.000000, 5.638000, 2.000000, 5.000000, 5.0000~
````

i. Pivoting the data and converting to kable

````
#### i
#### pivoting the data
```{r}
numericSummaryFinal <- numericSummary %>%
  pivot_longer("Id":"ageofGarage", names_to = "variable", values_to = "value") %>%
  pivot_wider(names_from = stat, values_from = value) %>%
  mutate(missing_pct = 100*missing/n,
         unique_pct = 100*unique/n) %>%
  dplyr::select(variable, n, missing, missing_pct, unique, unique_pct, everything())

numericSummaryFinal
```
````

````
```{r}
options(digits=3)
options(scipen=99)
numericSummaryFinal %>% kable()
```
````

j. Creating second part of data report

```
##### j
##### getting the modes
```{r}
getmodes <- function(v,type=1) {
  tbl <- table(v)
  m1<-which.max(tbl)
  if (type==1) {
    return (names(m1)) #1st mode
  }
  else if (type==2) {
    return (names(which.max(tbl[-m1]))) #2nd mode
  }
  else if (type==-1) {
    return (names(which.min(tbl))) #least common mode
  }
  else {
    stop("Invalid type selected")
  }
}
```
```

```
#### getting mode cnts
```{r}
getmodesCnt <- function(v,type=1) {
  tbl <- table(v)
  m1<-which.max(tbl)
  if (type==1) {
    return (max(tbl)) #1st mode freq
  }
  else if (type==2) {
    return (max(tbl[-m1])) #2nd mode freq
  }
  else if (type==-1) {
    return (min(tbl)) #least common freq
  }
  else {
    stop("Invalid type selected")
  }
}
```
```

```
##### creating the MyFactorSummary function to summarize factor columns
```{r}
myFactorSummary<-function(x){
  c(length(x), n_distinct(x), sum(is.na(x)),
    getmodes(x, type =1), getmodesCnt(x, type =1),
    getmodes(x, type =2), getmodesCnt(x, type =2),
    getmodes(x, type =-1), getmodes(x, type =-1))
}
```
```

```
#### creating a tibble FactorSummary that contains the result of myFactorSummary function
#### applied on each column of housingFactor tibble
```{r}
FactorSummary <- housingFactor %>% summarise(across(.cols = everything(), ~myFactorSummary(.x)))
glimpse(FactorSummary)
```
```

```
Rows: 9
Columns: 38
$ MSZoning     <chr> "1000", "4", "0", "RL", "803", "RM", "151", "RH", "RH"
$ Alley        <chr> "1000", "3", "938", "Grvl", "40", "Pave", "22", "Pave", "Pave"
$ LotShape     <chr> "1000", "4", "0", "Reg", "633", "IR1", "330", "IR3", "IR3"
$ LandContour  <chr> "1000", "4", "0", "Lvl", "905", "Bnk", "40", "Low", "Low"
$ LotConfig    <chr> "1000", "4", "0", "Inside", "711", "Corner", "179", "other", "other"
$ LandSlope    <chr> "1000", "3", "0", "Gtl", "946", "Mod", "48", "Sev", "Sev"
```

```
#### adding a column stats that names each row of FactorSummary
```{r}
FactorSummary <-cbind( stat=c("n","unique","missing","1st Mode", "1st Mode Freq", "2nd Mode",
                         "2nd Mode Freq", "Least Common Mode", "Least Common Mode Freq"),
                   FactorSummary)
glimpse(FactorSummary)
```
```

```
Rows: 9
Columns: 39
$ stat         <chr> "n", "unique", "missing", "1st Mode", "1st Mode Freq", "2nd Mode", "2nd ~
$ MSZoning     <chr> "1000", "4", "0", "RL", "803", "RM", "151", "RH", "RH"
$ Alley        <chr> "1000", "3", "938", "Grvl", "40", "Pave", "22", "Pave", "Pave"
$ LotShape     <chr> "1000", "4", "0", "Reg", "633", "IR1", "330", "IR3", "IR3"
$ LandContour  <chr> "1000", "4", "0", "Lvl", "905", "Bnk", "40", "Low", "Low"
$ LotConfig    <chr> "1000", "4", "0", "Inside", "711", "Corner", "179", "other", "other"
$ LandSlope    <chr> "1000", "3", "0", "Gtl", "946", "Mod", "48", "Sev", "Sev"
$ Neighborhood <chr> "1000", "18", "0", "NAmes", "167", "CollgCr", "113", "Timber", "Timber"
```

```
#### pivoting the data
```{r}
FactorSummaryFinal <- FactorSummary %>%
  pivot_longer("MSZoning":"SaleType", names_to = "variable", values_to = "value") %>%
  pivot_wider(names_from = stat, values_from = value) %>%
  mutate(missing_pct = 100* as.numeric(missing)/as.numeric(n),
         unique_pct = 100* as.numeric(unique)/as.numeric(n)) %>%
  dplyr::select(variable, n, missing, missing_pct, unique, unique_pct, everything())

FactorSummaryFinal
```
```

A tibble: 38 x 12

variable <chr>	n <chr>	missing <chr>	missing_pct <dbl>	unique <chr>	unique_pct <dbl>	1st Mode <chr>	1st Mode Freq <chr>
MSZoning	1000	0	0.0	4	0.4	RL	803
Alley	1000	938	93.8	3	0.3	Grvl	40
LotShape	1000	0	0.0	4	0.4	Reg	633
LandContour	1000	0	0.0	4	0.4	Lvl	905
LotConfig	1000	0	0.0	4	0.4	Inside	711
LandSlope	1000	0	0.0	3	0.3	Gtl	946
Neighborhood	1000	0	0.0	18	1.8	NAmes	167
Condition1	1000	0	0.0	6	0.6	Norm	871
BldgType	1000	0	0.0	5	0.5	1Fam	837
HouseStyle	1000	0	0.0	8	0.8	1Story	488

1-10 of 38 rows | 1-8 of 12 columns   Previous 1 2 3 4 Next

~~~~
#### converting data to kable
```{r}
options(digits=3)
options(scipen=99)
FactorSummaryFinal %>% kable()
```
~~~~

variable	n	missing	missing_pct	unique	unique_pct	1st Mode	1st Mode Freq	2nd Mode	2nd Mode Freq	Least Common Mode	Least Comm Mode Freq
MSZoning	1000	0	0.0	4	0.4	RL	803	RM	151	RH	RH
Alley	1000	938	93.8	3	0.3	Grvl	40	Pave	22	Pave	Pave
LotShape	1000	0	0.0	4	0.4	Reg	633	IR1	330	IR3	IR3
LandContour	1000	0	0.0	4	0.4	Lvl	905	Bnk	40	Low	Low
LotConfig	1000	0	0.0	4	0.4	Inside	711	Corner	179	other	other

2. Transformations
   a. Box cox

~~~~
#### a
#### plotting the histogram of all numeric columns in original data to check the skewness
```{r}
par(mar = c(0.75, 0.75, 0.75, 0.75))
par(mfrow=c(10,4))
housingNumeric <- select_if(housingData, is.numeric)
for(i in 1:ncol(housingNumeric)){
  hist(housingNumeric[, i], main = colnames(housingNumeric)[i], axes = F)
}
```
~~~~




```
##### Here I consider columns PoolArea, LotArea and BsmtUnfSFas they are continuous variables
```

````
##### Pool Area
#####taking pre snap of PoolArea attribute
```{r}
temp_PoolArea <- housingNumeric$PoolArea
```


##### box cox of PoolArea Attribute
##### adding a negligible value to PoolArea since boxcox is infeasible when value is zero
```{r}
b <- boxcox(housingNumeric$PoolArea+0.0001, optimize = T, lambda = c(-3, 3))
#changing the values of PoolArea using boxcox function
housingNumeric$PoolArea <- (housingNumeric$PoolArea^b$lambda - 1)/b$lambda
```


##### post snap of PoolArea
```{r}
par(mfrow = c(2, 1))

#plotting a pre snap of PoolArea attribute
hist(temp_PoolArea, xlab = 'PoolArea', main ='Presnap of PoolArea')

#plotting the post snap of PoolArea
hist(housingNumeric$PoolArea, xlab = 'PoolArea', main ='Post snap of PoolArea')
```
````



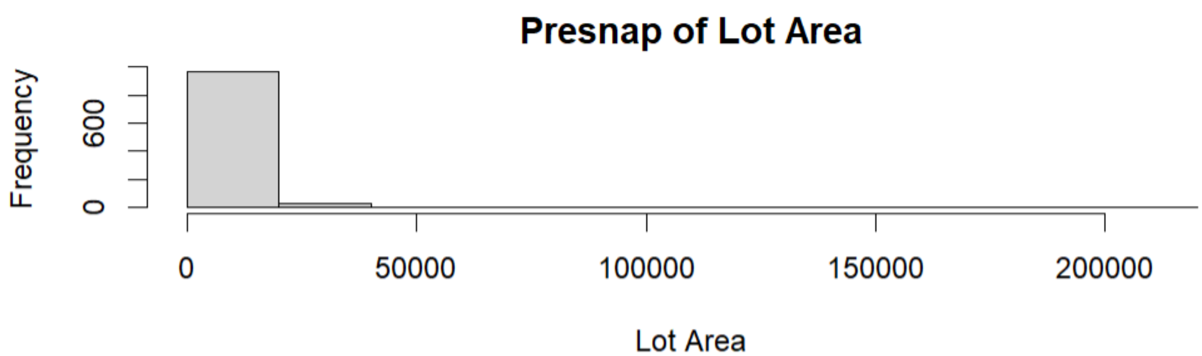

````
since most of the data is concentrated at zero, PoolArea haven't changed even after applying the
BoxCox function
````

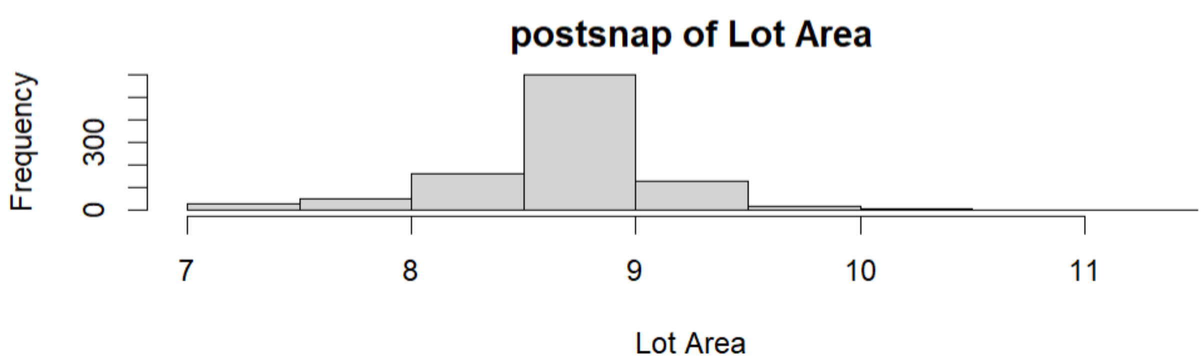
```
#### Lot AREA
#####getting pre snap of Lot area
```{r}
temp_LotArea <- housingNumeric$LotArea
```
boxcox of Lot area
```{r}
b <- boxcox(housingNumeric$LotArea, optimize = T, lambda = c(-3,3))

#changing the values of LotArea using boxcox function
housingNumeric$LotArea <- (housingNumeric$LotArea^b$lambda -1)/b$lambda
```
plotting pre snap and post snap of Lot area
```{r}
par(mfrow=c(2, 1))

#plotting the pre snap
hist(temp_LotArea, xlab = 'Lot Area', main ='Presnap of Lot Area')

#plotting the post snap
hist(housingNumeric$LotArea, xlab = 'Lot Area', main ='postsnap of Lot Area')
```
```





```
#### BsmtUnfSF

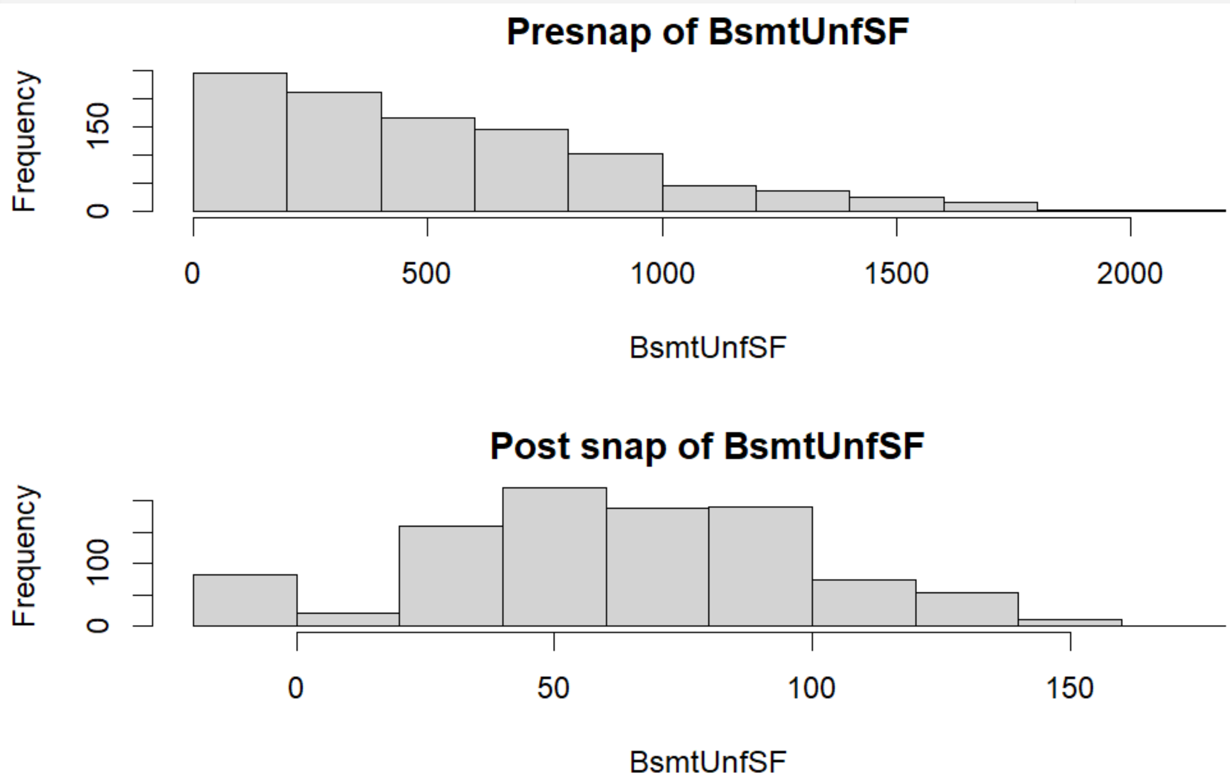
##### taking a pre snap of BsmtUnfSF
```{r}
temp_BsmtUnfSF <- housingNumeric$BsmtUnfSF
```

box cox of BsmtUnfSF Attribute
adding a negligible value to BsmtUnfSF since boxcox is infeasible when value is zero
```{r}
b <- boxcox(housingNumeric$BsmtUnfSF+0.0001, optimize = T, lambda = c(-3, 3))
#changing the values of MSsubClass using boxcox function
housingNumeric$BsmtUnfSF <- (housingNumeric$BsmtUnfSF^b$lambda - 1)/b$lambda
```

post snap of MSSub Class
```{r}
par(mfrow = c(2, 1))

#plotting a pre snap of BsmtUnfSF attribute
hist(temp_BsmtUnfSF, xlab = 'BsmtUnfSF', main ='Presnap of BsmtUnfSF')

#plotting the post snap of BsmtUnfSF
hist(housingNumeric$BsmtUnfSF, xlab = 'BsmtUnfSF', main ='Post snap of BsmtUnfSF')
```
```



b. Imputing missing values in attribute 'LotFrontage'

i. Mean value imputation

```
##### mean imputation of missing values in LotFrontage Column
```{r}
#creating a temporary variable to store the original housing data
data_imp_mean <- housingData

#getting pre count
print(paste("The number of missing values before mean impuatation is ",
sum(is.na(data_imp_mean$LotFrontage))))

#getting the position of missing values of LotFrontage attribute
missing_values <- which(is.na(data_imp_mean$LotFrontage))

#imputing missing values of LotFrontage attribute with mean
data_imp_mean$LotFrontage[missing_values] <- mean(data_imp_mean$LotFrontage, na.rm = T)

#getting post count
print(paste("The number of missing values after mean impuatation is ",
sum(is.na(data_imp_mean$LotFrontage))))
```
```

```
[1] "The number of missing values before mean impuatation is  207"
[1] "The number of missing values after mean impuatation is  0"
```

ii. Regression with error imputation

```
##### ii
##### regression with error
```{r}
#getting a copy of original housing data
data_imp_RE <- housingData

#getting pre count
print(paste("The number of missing values before mean impuatation is ",
sum(is.na(temp_HD$LotFrontage))))
```
```

```
[1] "The number of missing values before mean impuatation is  207"
```

```
```{r}
#fitting the linear regression model
fit<-lm(LotFrontage ~ LotArea + SalePrice, data_imp_RE)

#getting the summary of the model
f<-summary(fit)
print (f)
```
```

```
Call:
lm(formula = LotFrontage ~ LotArea + SalePrice, data = data_imp_RE)
```

```{r}
# extract the coefficients
c<-f[[4]]

# extract the model standard error
se<-f[[6]]
```

```{r}
#getting the position of missing values of LotFrontage attribute
missing <- which(is.na(data_imp_RE$LotFrontage))

#data_imp_RE[missing,"LotFrontage"]<- (c[1] + c[2]*data_imp_RE[missing,"LotArea"] + c[3] *
data_imp_RE[missing,"SalePrice"] )

#predicting missing values using the linear model fitted and adding error to it
data_imp_RE[missing,"LotFrontage"]<- predict(fit, housingData[missing, c('LotArea', 'SalePrice')]) +
 rnorm(length(missing),0,se)
```

```{r}
#getting post count
print(paste("The number of missing values after mean impuatation is ",
sum(is.na(data_imp_RE$LotFrontage))))
```

```
[1] "The number of missing values after mean impuatation is  0"
```

iii. Predictive Mean Matching (PMM)

```
##### iii
#####  predictive mean matching
```{r}
#getting a copy of original housing data
temp_HD <- housingData

#getting pre count
print(paste("The number of missing values before mean impuatation is ",
sum(is.na(temp_HD$LotFrontage))))

# Impute missing values using PMM
imp_single <- mice(temp_HD, m = 1, method = "pmm")

#getting only the complete cases from above 'imp_single' mids class
data_imp_pmm <- complete(imp_single)

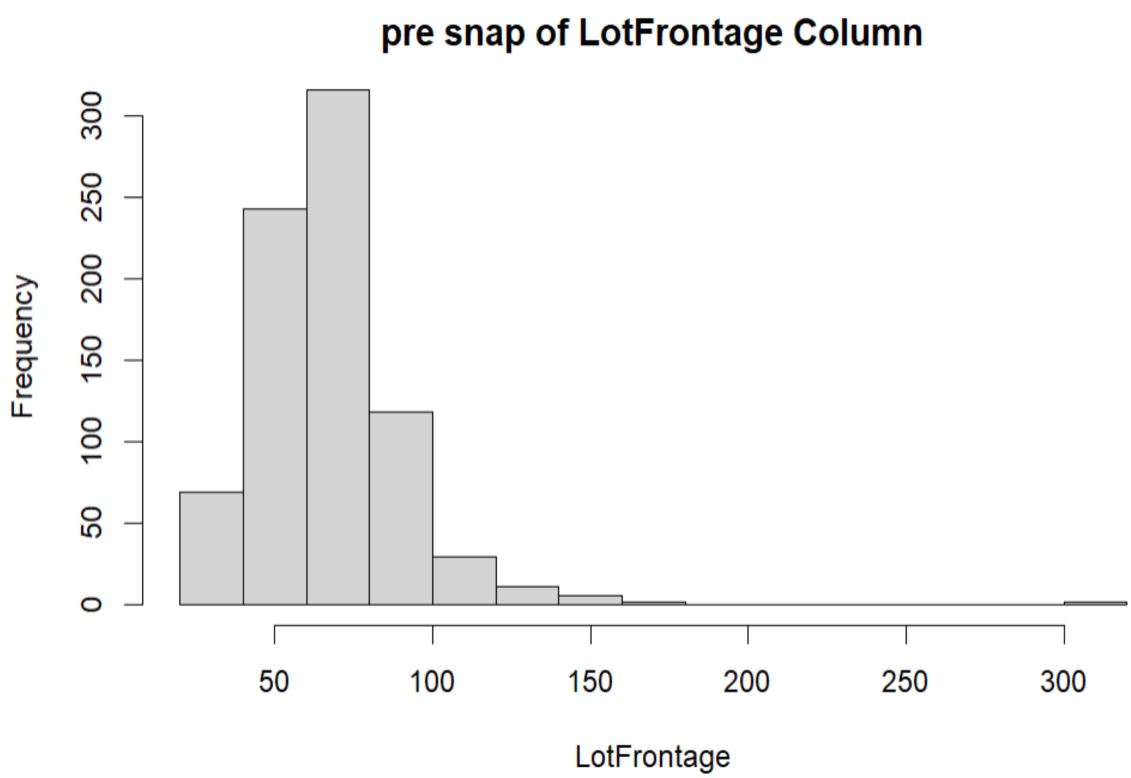
#getting post count
print(paste("The number of missing values after mean impuatation is ",
sum(is.na(data_imp_pmm$LotFrontage))))
```
```

~~~~
[1] "The number of missing values before mean impuatation is  207"

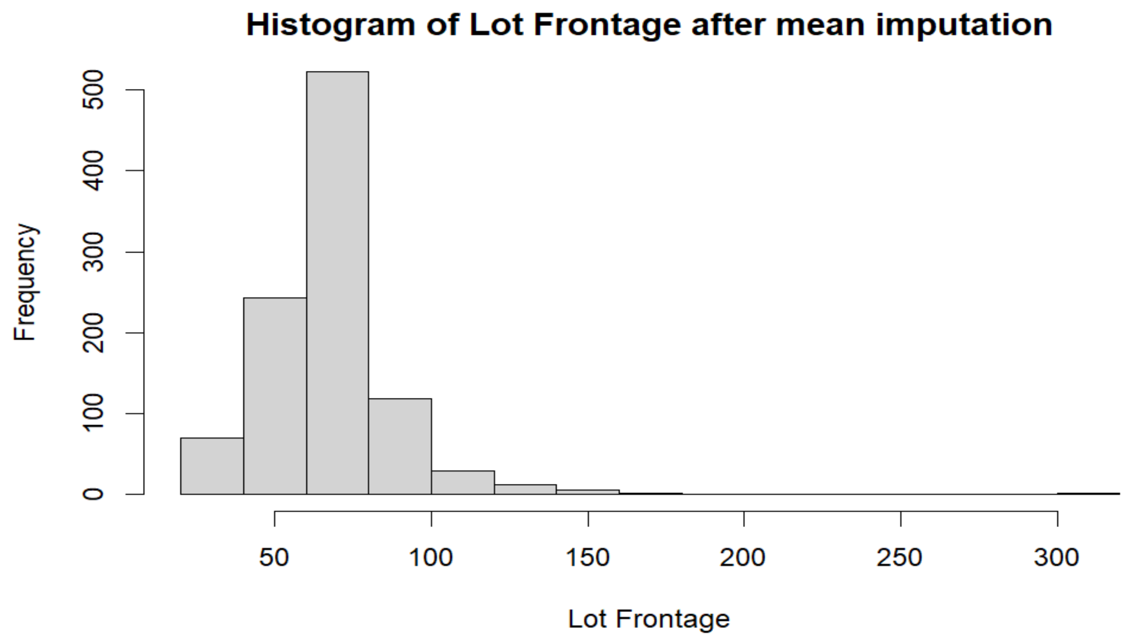
 iter imp variable
  1   1  LotFrontage*  MasVnrArea*  GarageYrBlt*
  2   1  LotFrontage*  MasVnrArea*  GarageYrBlt*
  3   1  LotFrontage*  MasVnrArea*  GarageYrBlt*
  4   1  LotFrontage*  MasVnrArea*  GarageYrBlt*
  5   1  LotFrontage*  MasVnrArea*  GarageYrBlt*
Warning: Number of logged events: 68
[1] "The number of missing values after mean impuatation is  0"
~~~~

iv. Getting the visual depictions how the data transformed before and after imputation

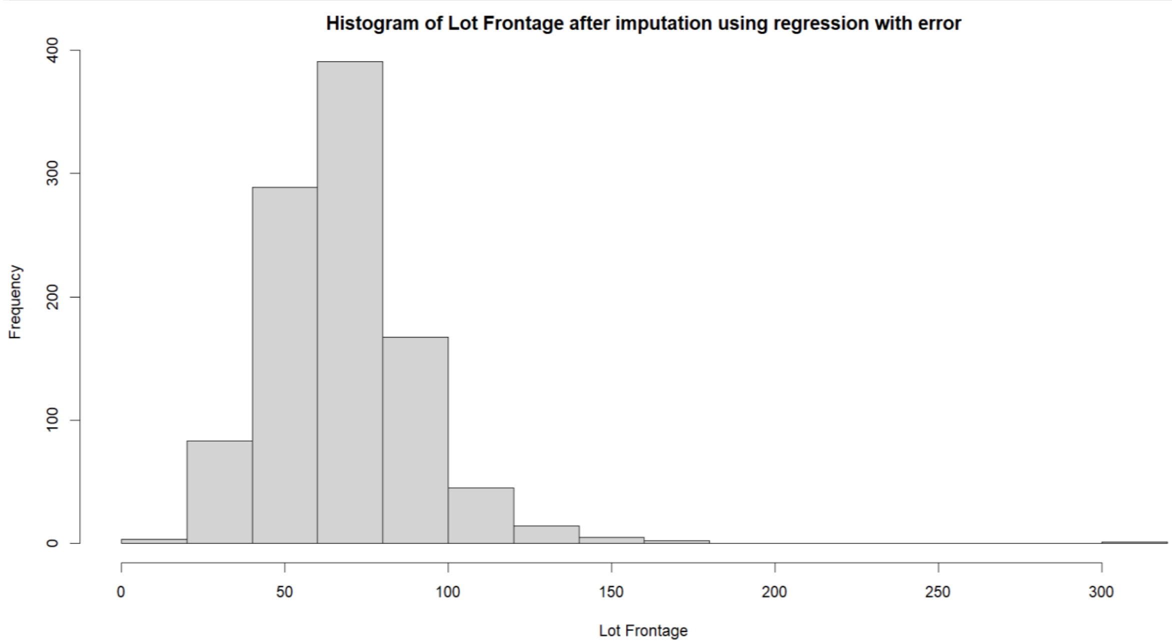
~~~~
##### Histogram of Lot Frontage after imputation using predictive mean matching
```{r}
#pre snap of lot frontage
hist(housingData$LotFrontage,xlab = 'LotFrontage',
     main = "pre snap of LotFrontage Column")
```
~~~~



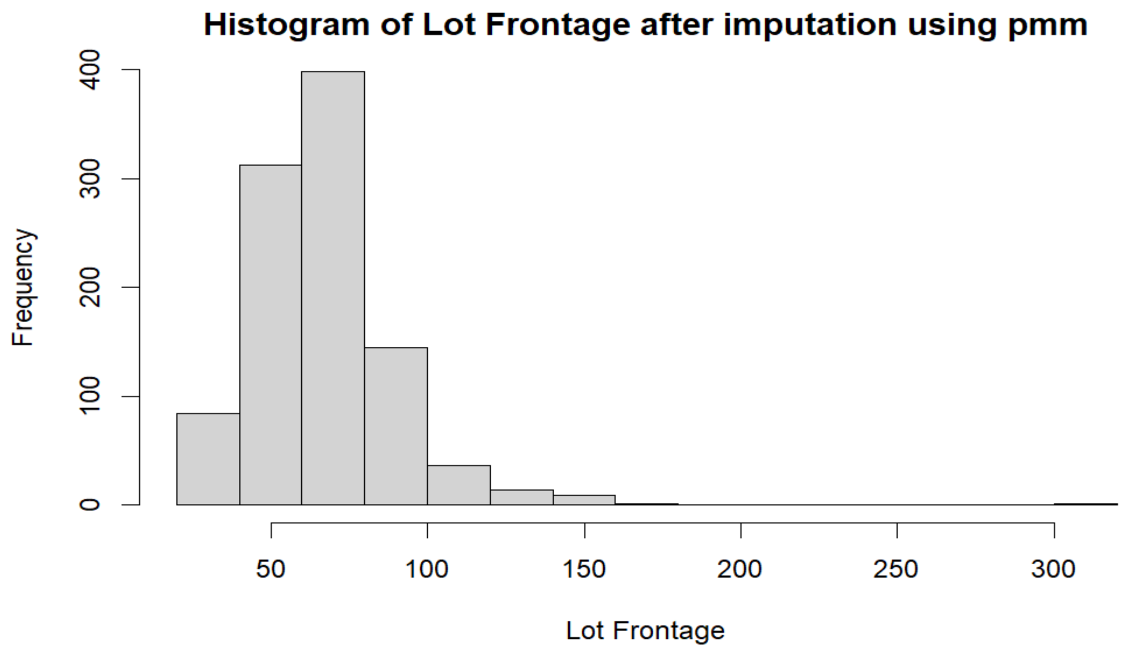

~~~~
```{r}
#post snap of LotFrontage after after mean imputation
hist(data_imp_mean$LotFrontage, xlab = "Lot Frontage",
     main = "Histogram of Lot Frontage after mean imputation")
```
~~~~




```{r}
#post snap of LotFrontage after imputation using regression with error
hist(data_imp_RE$LotFrontage, xlab = 'Lot Frontage',
     main = 'Histogram of Lot Frontage after imputation using regression with error')
```




```{r}
#post snap of LotFrontage after imputation using PMM
hist(data_imp_pmm$LotFrontage, xlab = 'Lot Frontage',
     main = 'Histogram of Lot Frontage after imputation using pmm')
```



c. Collapsing the factor levels of 'Exterior1st' attribute to five

```
##### c
##### getting the pre count of each factor in attribute Exterior1st
#####in descending order
```{r}
#summary(housingData$Exterior1st)
fct_count(housingData$Exterior1st, sort = T)
```
```

A tibble: 8 x 2

f <fctr>	n <int>
VinylSd	328
HdBoard	175
MetalSd	153
Wd Sdng	141
Plywood	73
other	52
BrkFace	42
CemntBd	36

8 rows

```
##### collapsing the columns so that the column Exterior1st have only five factors
```{r}
housingData$Exterior1st <- fct_collapse(housingData$Exterior1st,
                                        Other = c("BrkFace", "CemntBd", "other",
"Plywood"))

#getting the post count for each factor level in descending order
fct_count(housingData$Exterior1st, sort = T)
```
```

A tibble: 5 x 2

f <fctr>	n <int>
VinylSd	328
Other	203
HdBoard	175
MetalSd	153
Wd Sdng	141

5 rows

d. More fun with factors

i. Computing average Sales price for each Neighborhood level

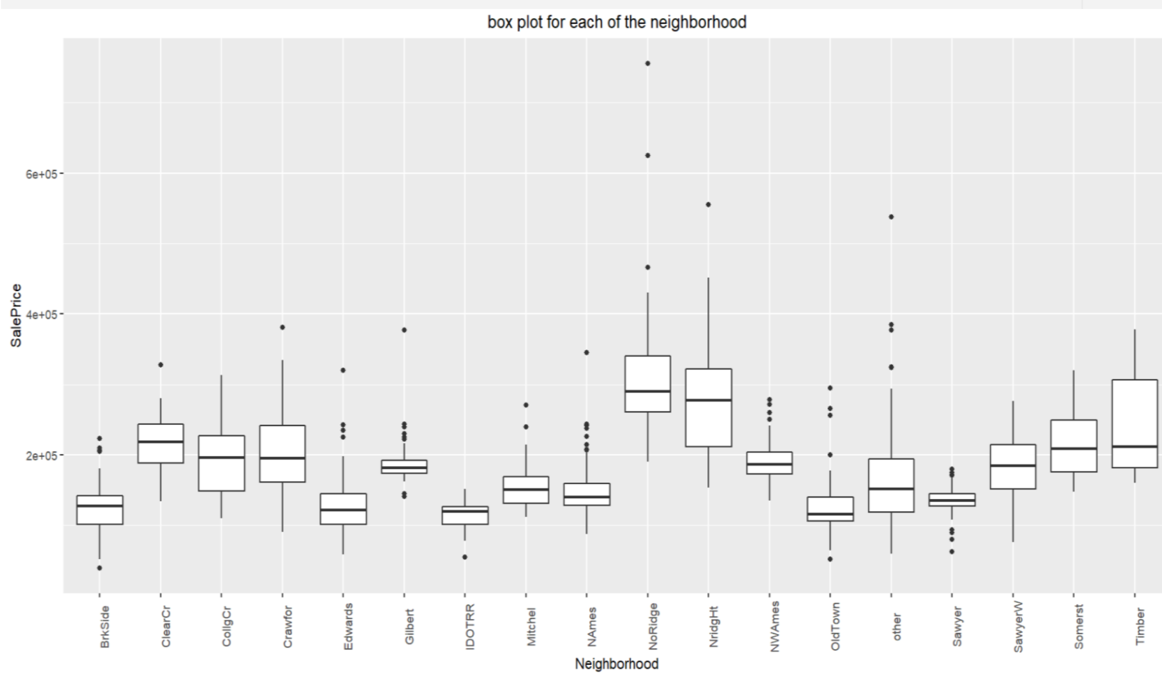
```
##### i
##### calculating average sales price for each neighborhood
```{r}
housingData %>% group_by(Neighborhood) %>%
  summarise(Mean = mean(SalePrice))
```
```

Neighborhood <chr>	Mean <dbl>
BrkSide	124844.4
ClearCr	218265.1
CollgCr	194941.7
Crawfor	209765.6
Edwards	128771.5
Gilbert	189466.2
IDOTRR	114319.0
Mitchel	154788.5
NAmes	146669.3
NoRidge	328793.8

Neighborhood <chr>	Mean <dbl>
NridgHt	283057.1
NWAmes	191823.1
OldTown	126023.4
other	170247.7
Sawyer	134707.9
SawyerW	183970.7
Somerst	211678.4
Timber	241940.0

ii. Creating a parallel boxplot chart for each of the neighborhoods

```
##### ii
##### making the boxplots of Saleprice for each neighborhood
```{r}
ggplot(housingData, aes(x= Neighborhood, y = SalePrice)) +
  #Adding the boxplot layer
  geom_boxplot() +
  #adding title
  labs(title = "box plot for each of the neighborhood", xlab = 'Neighborhood') +
  #setting the title of the plot to middle and alligning the xlabs vertically
  theme(axis.text.x = element_text(angle = 90), plot.title = element_text(hjust = 0.5))
```
```



iii. Using forcats to reorder the factor levels of the neighborhood based on median price per neighborhood

```
##### iii
##### using forcats, ordering the factors levels of neighborhooding based on the decsending order of Median of Saleprice
```{r}
housingData$Neighborhood <-fct_reorder(housingData$Neighborhood,
                                        housingData$SalePrice, .fun = median, .desc = T)
```
```

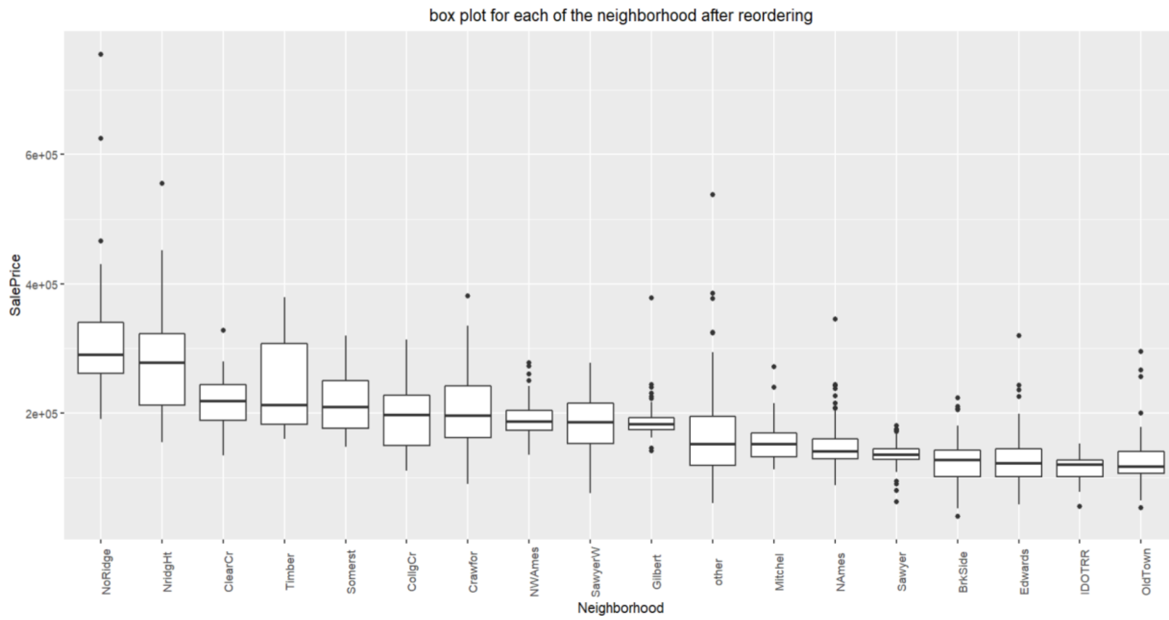
```
##### displaying the first five rows based on above transformation
```{r}
head(housingData %>% group_by(Neighborhood) %>%
  summarise(Median = median(SalePrice)))
```
```

A tibble: 6 x 2

Neighborhood <fctr>	Median <dbl>
NoRidge	290000
NridgHt	277500
ClearCr	218000
Timber	211450
Somerst	208750
CollgCr	196500

iv. Plotting the parallel boxplot after reordering the factor levels of neighborhood based on median of sale price in decreasing order.

```
##### iv
#### making boxplot of saleprice for each neighborhood which is sorted on descending order of median
```{r}
ggplot(housingData, aes(x= Neighborhood, y = SalePrice)) +
  #Adding the boxplot layer
  geom_boxplot() +
  #adding title
  labs(title = "box plot for each of the neighborhood after reordering", xlab = 'Neighborhood') +
  #setting the title of the plot to middle and alligning the xlabs vertically
  theme(axis.text.x = element_text(angle = 90), plot.title = element_text(hjust = 0.5))
```
```


box plot for each of the neighborhood after reordering
SalePrice
6e+05
4e+05
2e+05
NoRidge
NridgHt
ClearCr
Timber
Somerst
CollgCr
Crawfor
NWAmes
SawyerW
Gilbert
other
Mitchel
NAmes
Sawyer
BrkSide
Edwards
IDOTRR
OldTown
Neighborhood