# Homework 4 - Data Wrangling in R

Daniel Carpenter & NAME HERE

September 2022

## Table of contents

## Packages

```r
# Data Wrangling
library(tidyverse)

# Modeling
library(car)      # symbox
library(EnvStats) # boxcox function
library(mice)     # Predictive mean matching for missing values

# Aesthetics
library(cowplot)  # multiple ggplots on one plot with plot_grid()
```

# 1 - Data Quality Report

## 1 (a) - Read data

```r
# Read data
housingData <- read_csv('housingData.csv')
```

```
Rows: 1000 Columns: 74
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (38): MSZoning, Alley, LotShape, LandContour, LotConfig, LandSlope, Neig...
dbl (36): Id, MSSubClass, LotFrontage, LotArea, OverallQual, OverallCond, Ye...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# create three new variables
housingData <- housingData %>%
  dplyr::mutate(age              = YrSold - YearBuilt,
                ageSinceRemodel = YrSold - YearRemodAdd,
                ageofGarage     = YrSold - GarageYrBlt
                )
```

## 1 (b) - Numeric Housing Tibble

- Create a tibble named `housingNumeric` which contains all of the numeric variables from the original data.
- use the `dplyr::select` command along with the `is.numeric` function to complete this task.

```
# Convert df to a tibble
housingNumeric <- as_tibble(housingData) %>%

  # Only select numeric data
  # note would usually use command select_if(is.numeric)
  select(where(is.numeric))
```

## 1 (c) - Factor Housing Tibble

- create a tibble named `housingFactor` which contains all of the character variables from the original data.

```
housingFactor <- as_tibble(housingData) %>%

  # Change all character variables to factors
  # Keep only the changed data. Implicitly keeping only factor (prev. char vars)
  transmute_if(is.character, as.factor)
```

## 1 (d) - Use Glimpse

```
# NOT RUN
# glimpse(housingNumeric)
# glimpse(housingFactor)
```

## 1 (e) - Get Q1 and Q3

- create our own user-defined functions for extracting only first and 3rd quantile
- Explanation: Gets the quantiles of a vector using quantile function, but we use the `[]` brackets to retrieve the 2nd or 4th objects in the vector, which are `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]
}
```

## 1 (f) - Vectorized Summary Stats

- Function that will help apply several summary statistics to our data all at once
- Contains vector of functions with default to not evaluate if `na`

```r
# Vector of functions
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))
}

# Name of each functions within the vector
statNames <- c('n', 'unique', 'missing', 'mean', 'min', 'Q1', 'median', 'Q3', 'max', 'sd')
```

## 1 (g) - Apply Summary Stats

- Apply summary stats function with `summarize` function

```r
numericSummary <- housingNumeric %>%

  # Apply vector of functions using summarise
  summarise( across( where(is.numeric), ~myNumericSummary(.x) ) )
```

## 1 (h) - Add Stats Names

- Combine original data set and the names of each summary statistic

```r
numericSummary <- cbind(
  stat=c("n","unique","missing","mean","min","Q1","median","Q3","max","sd"),
  numericSummary
```

```
)

# glimpse(numericSummary) # uncomment to see effects
```

## 1 (i) - Pretty up data

### Transform data to make it ready for output format

```
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) %>%
  select(variable, n, missing, missing_pct, unique, unique_pct, everything())
```

### Show the output

```
library(knitr)
options(digits=3)
options(scipen=99)
numericSummaryFinal %>% kable()
```

| variable | n | missing | missing_pct | unique | unique_pct | mean | min | Q1 | median | Q3 | max | sd |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id | 1000 | 0 | 0.0 | 1000 | 100.0 | 500.500 | 1 | 251 | 500 | 750.2 | 1000 | 288.819 |
| MSSubClass | 1000 | 0 | 0.0 | 13 | 1.3 | 57.185 | 20 | 20 | 50 | 70.0 | 190 | 41.875 |
| LotFrontage | 1000 | 207 | 20.7 | 102 | 10.2 | 68.745 | 21 | 58 | 68 | 80.0 | 313 | 23.198 |
| LotArea | 1000 | 0 | 0.0 | 760 | 76.0 | 10424.881 | 1477 | 7500 | 9422 | 11423.5 | 215245 | 9940.619 |
| OverallQual | 1000 | 0 | 0.0 | 10 | 1.0 | 5.979 | 1 | 5 | 6 | 7.0 | 10 | 1.310 |
| OverallCond | 1000 | 0 | 0.0 | 8 | 0.8 | 5.638 | 2 | 5 | 5 | 6.0 | 9 | 1.114 |
| YearBuilt | 1000 | 0 | 0.0 | 108 | 10.8 | 1969.836 | 1875 | 1954 | 1971 | 1998.0 | 2009 | 29.119 |
| YearRemodAdd | 1000 | 0 | 0.0 | 61 | 6.1 | 1984.108 | 1950 | 1967 | 1992 | 2002.0 | 2010 | 20.116 |
| MasVnrArea | 1000 | 4 | 0.4 | 249 | 24.9 | 95.418 | 0 | 0 | 0 | 146.2 | 1600 | 177.318 |
| BsmtFinSF1 | 1000 | 0 | 0.0 | 490 | 49.0 | 438.686 | 0 | 0 | 400 | 700.0 | 1880 | 405.837 |
| BsmtFinSF2 | 1000 | 0 | 0.0 | 107 | 10.7 | 44.296 | 0 | 0 | 0 | 0.0 | 1127 | 150.493 |
| BsmtUnfSF | 1000 | 0 | 0.0 | 598 | 59.8 | 535.078 | 0 | 208 | 441 | 779.2 | 2153 | 417.944 |
| TotalBsmtSF | 1000 | 0 | 0.0 | 549 | 54.9 | 1018.060 | 0 | 793 | 962 | 1223.5 | 3206 | 403.641 |
| X1stFlrSF | 1000 | 0 | 0.0 | 581 | 58.1 | 1131.251 | 334 | 868 | 1060 | 1327.2 | 3228 | 350.862 |
| X2ndFlrSF | 1000 | 0 | 0.0 | 306 | 30.6 | 346.279 | 0 | 0 | 0 | 735.0 | 1872 | 426.395 |
| LowQualFinSF | 1000 | 0 | 0.0 | 15 | 1.5 | 4.991 | 0 | 0 | 0 | 0.0 | 528 | 45.295 |
| GrLivArea | 1000 | 0 | 0.0 | 664 | 66.4 | 1482.521 | 334 | 1111 | 1442 | 1735.0 | 4316 | 490.566 |
| BsmtFullBath | 1000 | 0 | 0.0 | 3 | 0.3 | 0.427 | 0 | 0 | 0 | 1.0 | 2 | 0.509 |
| BsmtHalfBath | 1000 | 0 | 0.0 | 2 | 0.2 | 0.059 | 0 | 0 | 0 | 0.0 | 1 | 0.236 |
| FullBath | 1000 | 0 | 0.0 | 4 | 0.4 | 1.529 | 0 | 1 | 2 | 2.0 | 3 | 0.531 |

| variable | n | missing | missing_pct | unique | unique_pct | mean | min | Q1 | median | Q3 | max | sd |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| HalfBath | 1000 | 0 | 0.0 | 3 | 0.3 | 0.384 | 0 | 0 | 0 | 1.0 | 2 | 0.501 |
| BedroomAbvGr | 1000 | 0 | 0.0 | 7 | 0.7 | 2.865 | 0 | 2 | 3 | 3.0 | 6 | 0.791 |
| KitchenAbvGr | 1000 | 0 | 0.0 | 3 | 0.3 | 1.041 | 1 | 1 | 1 | 1.0 | 3 | 0.203 |
| TotRmsAbvGrd | 1000 | 0 | 0.0 | 11 | 1.1 | 6.410 | 2 | 5 | 6 | 7.0 | 12 | 1.562 |
| Fireplaces | 1000 | 0 | 0.0 | 4 | 0.4 | 0.618 | 0 | 0 | 1 | 1.0 | 3 | 0.642 |
| GarageYrBlt | 1000 | 53 | 5.3 | 94 | 9.4 | 1976.938 | 1906 | 1960 | 1977 | 1999.0 | 2009 | 23.592 |
| GarageCars | 1000 | 0 | 0.0 | 5 | 0.5 | 1.720 | 0 | 1 | 2 | 2.0 | 4 | 0.714 |
| GarageArea | 1000 | 0 | 0.0 | 353 | 35.3 | 458.329 | 0 | 319 | 470 | 572.0 | 1356 | 197.780 |
| WoodDeckSF | 1000 | 0 | 0.0 | 226 | 22.6 | 94.555 | 0 | 0 | 0 | 168.0 | 857 | 127.144 |
| OpenPorchSF | 1000 | 0 | 0.0 | 169 | 16.9 | 43.610 | 0 | 0 | 22 | 64.0 | 547 | 61.915 |
| EncPorchSF | 1000 | 0 | 0.0 | 122 | 12.2 | 40.641 | 0 | 0 | 0 | 0.0 | 508 | 82.139 |
| PoolArea | 1000 | 0 | 0.0 | 3 | 0.3 | 1.224 | 0 | 0 | 0 | 0.0 | 648 | 27.403 |
| MiscVal | 1000 | 0 | 0.0 | 14 | 1.4 | 27.210 | 0 | 0 | 0 | 0.0 | 3500 | 190.707 |
| MoSold | 1000 | 0 | 0.0 | 12 | 1.2 | 6.207 | 1 | 4 | 6 | 8.0 | 12 | 2.626 |
| YrSold | 1000 | 0 | 0.0 | 5 | 0.5 | 2007.919 | 2006 | 2007 | 2008 | 2009.0 | 2010 | 1.318 |
| SalePrice | 1000 | 0 | 0.0 | 477 | 47.7 | 174560.637 | 37300 | 130000 | 160000 | 205000.0 | 755000 | 69329.319 |
| age | 1000 | 0 | 0.0 | 115 | 11.5 | 38.083 | 1 | 10 | 37 | 55.0 | 135 | 29.109 |
| ageSinceRemodel | 1000 | 0 | 0.0 | 61 | 6.1 | 23.811 | 0 | 6 | 16 | 41.2 | 60 | 20.033 |
| ageofGarage | 1000 | 53 | 5.3 | 97 | 9.7 | 30.973 | 0 | 9 | 30 | 48.0 | 102 | 23.563 |

# 1 (j) - Factor Data Report

> # TODO

# 2 - Transformation

## 2 (a) - Fixing Skewed Data

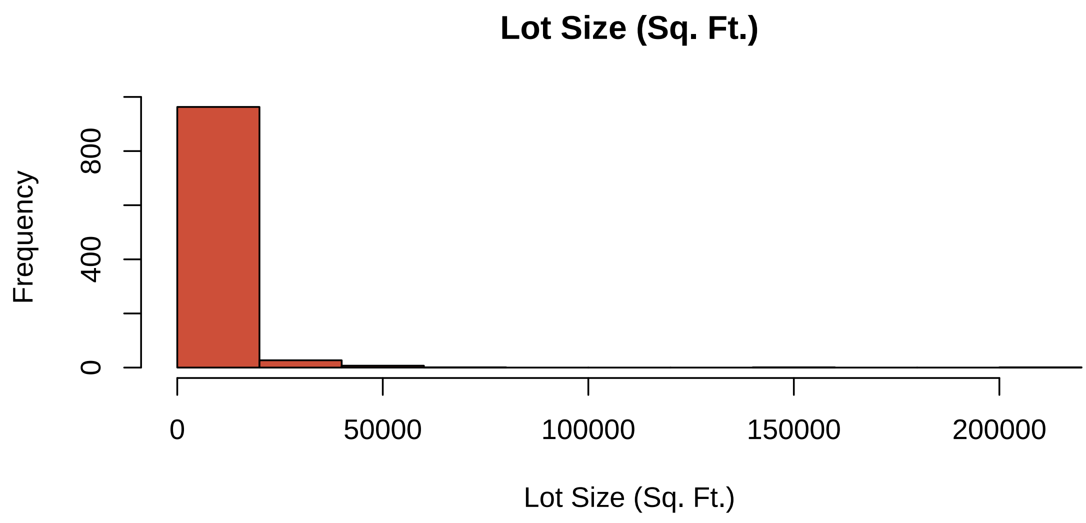
### Function to Convert Skewed Data to Normally Distributed Vector

```
normalizeDist <- function(aVector) {

  # Get the optimal lambda. Used later for converting to normal distribution
  normLambda = boxcox(aVector, optimize = TRUE)$lambda

  # Now convert vector to normal distribution, using the optimal lambda
  normalizedVector <- (aVector ** normLambda - 1) / normLambda

  return(normalizedVector)
}

# Colors
goodCol = 'darkseagreen3'
badCol  = 'tomato3'
```

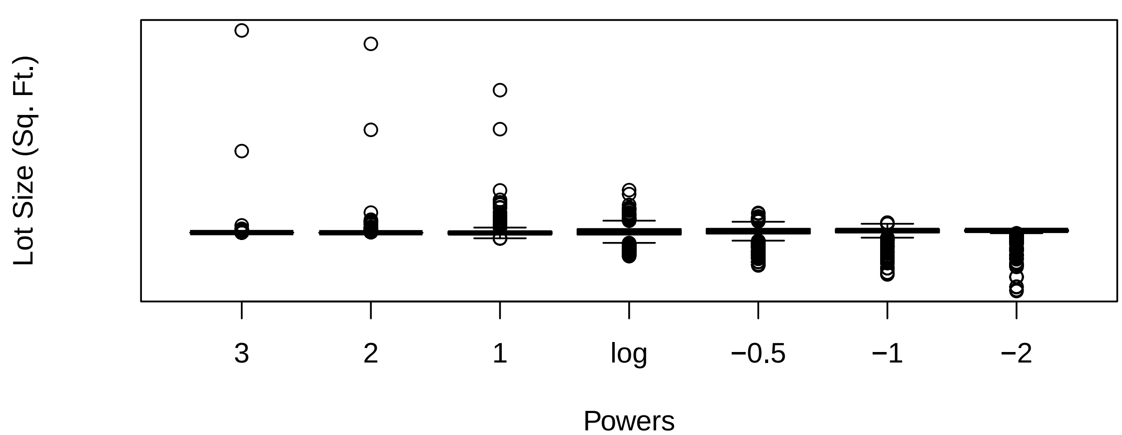
### i. Fix `LotArea` in Housing Data Set

### Lot area is highly skewed

```
varTitle = 'Lot Size (Sq. Ft.)'

# See that LotArea is highly skewed
hist(housingData$LotArea,
      main = varTitle, xlab = varTitle, col = badCol)
```




```
# Look at the symbox to see where optimal may lie
symbox(housingData$LotArea, data=housingData, powers=c(3,2,1,0,-0.5,-1,-2),
       ylab = varTitle)
```

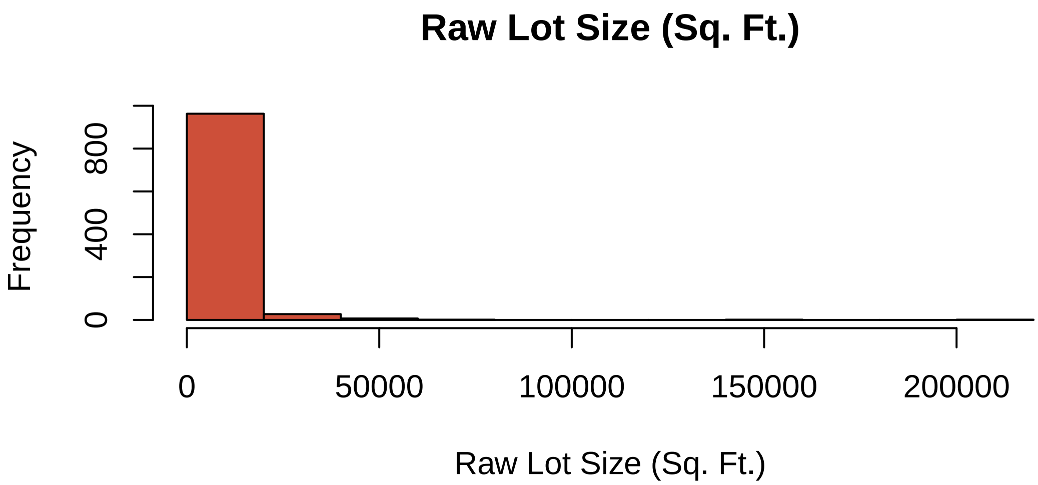


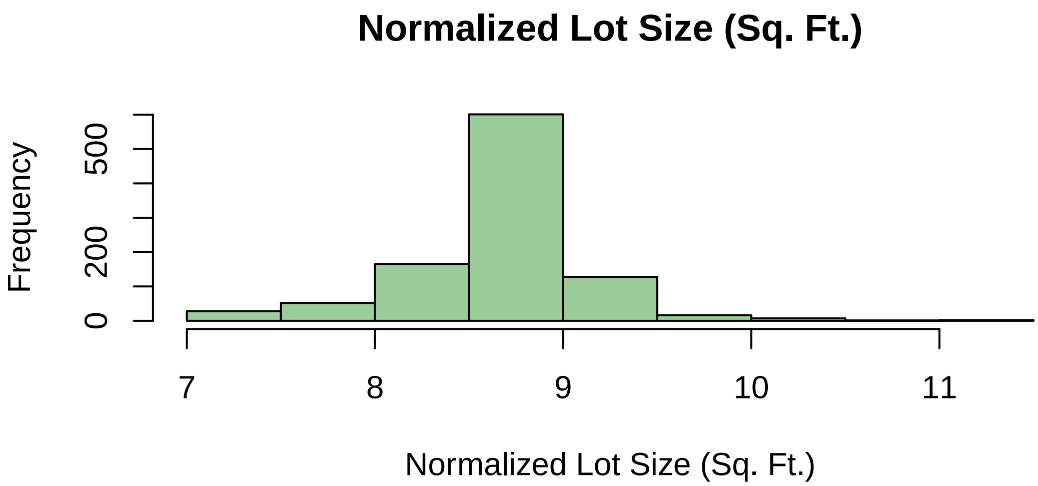
```
# Normalize the data and store in data
housingData <- housingData %>%
  mutate(normLotArea = normalizeDist(housingData$LotArea) )
```

#### See the normalized Lot Area variable

- You can see that the data is definitely more normal
- However, much of the data is near the median, which may or may not be fine, depending on the analysis

```
# Now see the results of the normalization
par(mfrow=c(2,1))

hist( housingData$LotArea,
      main = paste('Raw', varTitle), xlab = paste('Raw', varTitle),
      col = badCol )

hist( housingData$normLotArea,
      main = paste('Normalized', varTitle), xlab = paste('Normalized', varTitle),
      col = goodCol)
```

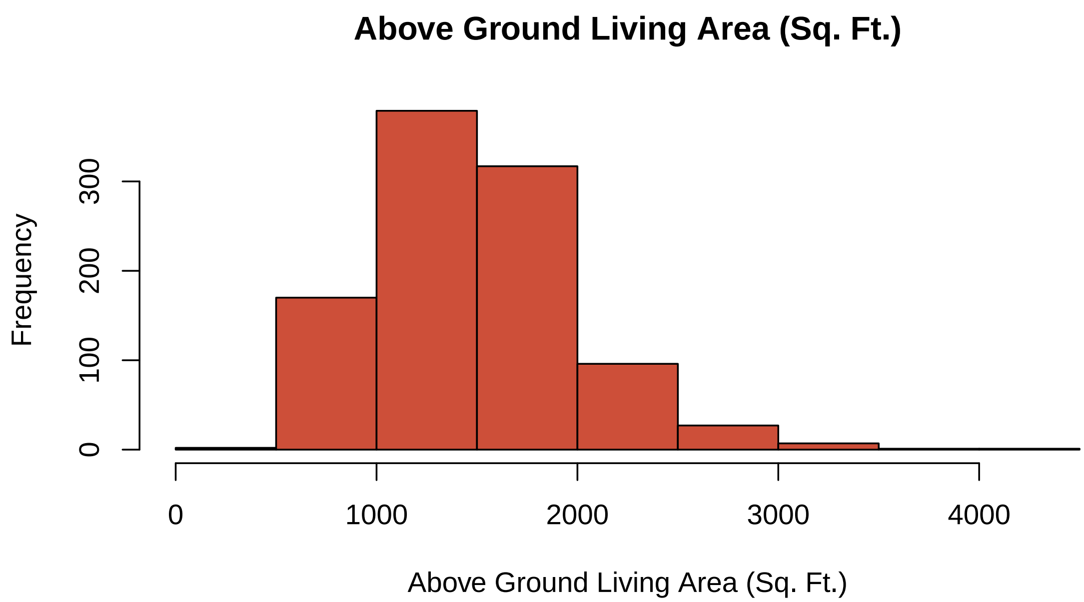

Raw Lot Size (Sq. Ft.)
Frequency
0
400
800
0
50000
100000
150000
200000
Raw Lot Size (Sq. Ft.)


Normalized Lot Size (Sq. Ft.)
Frequency
0
200
500
7
8
9
10
11
Normalized Lot Size (Sq. Ft.)

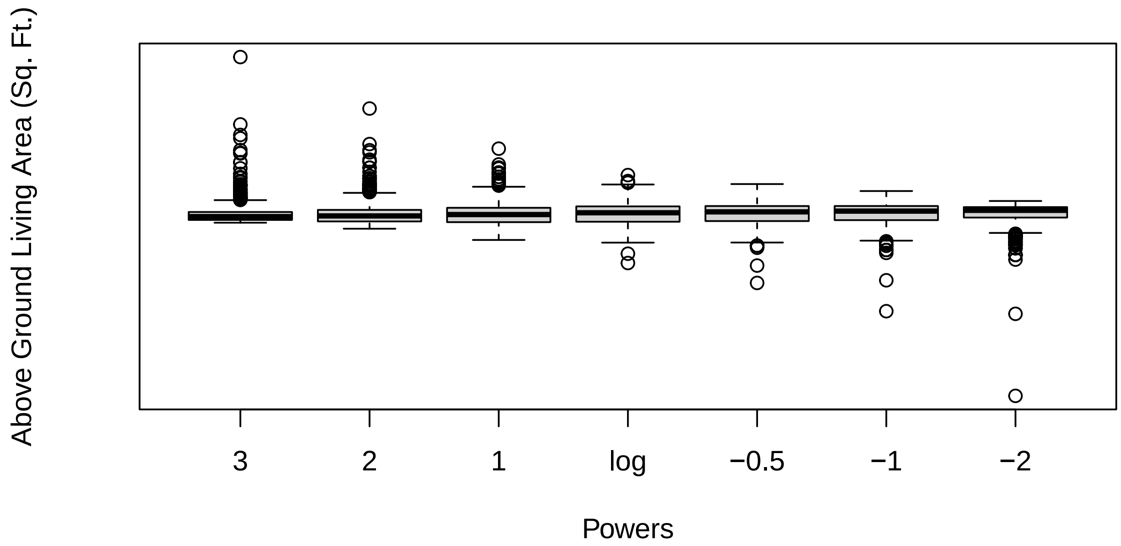
### i. Fix `GrLivArea` in Housing Data Set

#### Above Ground Living Area is highly skewed

```
varTitle = 'Above Ground Living Area (Sq. Ft.)'

# See that LotArea is highly skewed
hist(housingData$GrLivArea,
      main = varTitle, xlab = varTitle, col = badCol )
```




```
# Look at the symbox to see where optimal may lie
symbox(housingData$GrLivArea, data=housingData, powers=c(3,2,1,0,-0.5,-1,-2),
       ylab = varTitle)
```

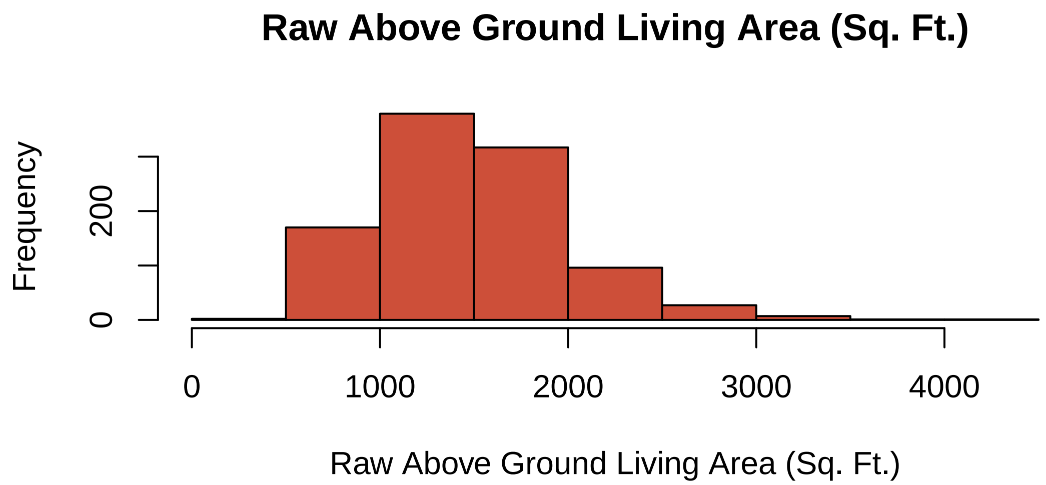


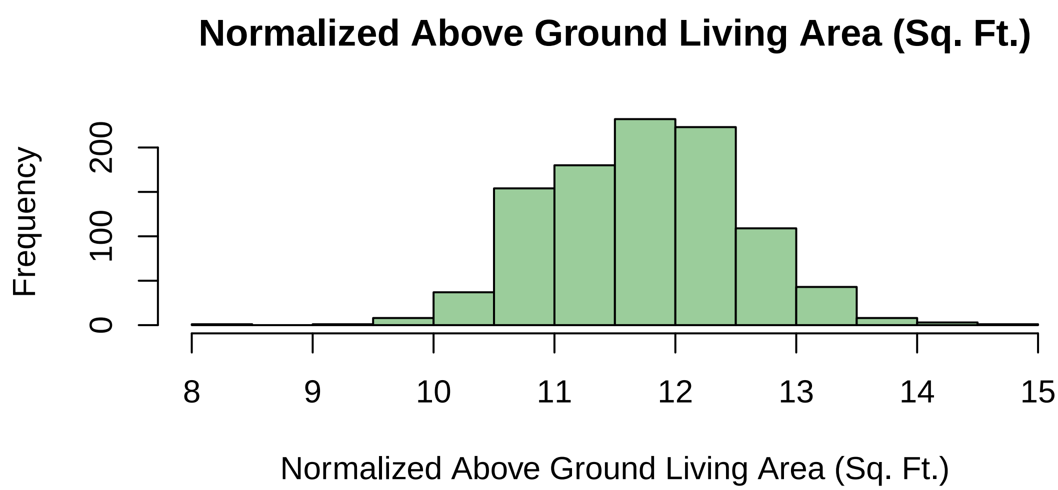

```
# Normalize the data and store in data
housingData <- housingData %>%
  mutate(normYearBuilt = normalizeDist(housingData$GrLivArea) )
```

#### See the normalized Lot Area variable

- You can see that the data is definitely more normal

```
# Now see the results of the normalization
par(mfrow=c(2,1))

hist( housingData$GrLivArea,
      main = paste('Raw', varTitle), xlab = paste('Raw', varTitle),
      col = badCol )

hist( housingData$normYearBuilt,
      main = paste('Normalized', varTitle), xlab = paste('Normalized', varTitle),
      col = goodCol )
```


Raw Above Ground Living Area (Sq. Ft.)
Frequency
0
200
0
1000
2000
3000
4000
Raw Above Ground Living Area (Sq. Ft.)


Normalized Above Ground Living Area (Sq. Ft.)
Frequency
0
100
200
8
9
10
11
12
13
14
15
Normalized Above Ground Living Area (Sq. Ft.)

## 2 (b) - Impute Missing Values

### Function to plot comparison of imputation methods

Highlights of function include:

- Histogram of actual data, with mean line on x-axis
- Histogram of imputed data, with mean line on x-axis
- Regression of actual and imputed data, spread across trival x-axis. Goal is to show variation in data

```
seeImputation <- function(df, df.meanInputed,
                          imputationMethod) {

  # Min/Max ranges so actual and imputed histograms align
  yMin = min(df.meanInputed$y)
  yMax = max(df.meanInputed$y)

  # Non Altered data -------------------------------------------------

  meanVal = mean(df$y, na.rm=T) # mean of the non altered data

  # Create the plot
  p1 <- df %>%
    ggplot(aes(x = y)) +

    # Histogram
    geom_histogram(color = 'grey65', fill = 'grey95') +

    # The mean value line
    geom_vline(xintercept = meanVal, color = 'tomato3') +

    # Text associated with mean value
    annotate("text",
             label = "Mean Value",
             x = meanVal, y = 100,
             size = 5, colour = "tomato3" ) +

    # Labels
    labs(title = 'Data with Missing Values',
         y     = 'Frequency',
```

```r
        x      = '' ) +

  xlim(yMin, yMax) + # min and max range of x axis (for equal comparison)
  theme_minimal() # Theme


# Imputed data -------------------------------------------------

meanValImpute = mean(df.meanInputed$y, na.rm=T)

# Create the plot
p2 <- df.meanInputed %>%
  ggplot(aes(x = y)) +

  # Histogram
  geom_histogram(color = 'grey65', fill = 'grey95') +

  # The mean value line
  geom_vline(xintercept = meanVal, color = 'tomato3') +

  # Text associated with mean value
  annotate("text",
           label = "Mean Value",
           x = meanValImpute, y = 100,
           size = 5, colour = "tomato3" ) +

  # Labels
  labs(title = 'Data without Missing Values',
           subtitle = paste('Using', imputationMethod, 'Imputation Method'),
           y = 'Frequency',
           x = 'Linear feet of street connected to property') +

  xlim(yMin, yMax) + # min and max range of x axis (for equal comparison)
  theme_minimal() # Theme

# Variation scatter ----------------------------------------------------------

p3 <- df.meanInputed %>% ggplot(aes(x=x, y=y, color=missing)) +

  # Add points
  geom_point(alpha = 0.5) +
```

```
    # Colors, limits, labels, and themes
    scale_color_manual(values = c('grey80', badCol),
                       labels = c('Actuals', 'Imputed') ) +
    ylim(0, quantile(df.meanInputed$y, 0.99)) + # lower 99% of dist
    labs(title   = 'Variation of Actuals vs. Imputed Data',
         x       = 'x',
         y       = 'Lot Frontage',
         caption =paste0('\nUsing housing.csv data',
                         '\nOnly showing lower 99% of distribution for viewing')
         ) +
    theme_minimal() + theme(legend.position = 'bottom',
                            legend.title    = element_blank())


  # Combine the plots for the final returned output
  combinedPlots <- plot_grid(p1, p2, p3,
                             ncol = 1, label_size = 12,
                             rel_heights = c(1, 1.1, 1.75))
  return(combinedPlots)
}
```

### Create Reusable data set `df`

```
# How much is missing?
missing <- is.na(housingData$LotFrontage)
paste('There are', sum(missing), 'missing values')
```

```
[1] "There are 207 missing values"
```

```
# Create a data frame to easily reference the lot frontage for other inputation
df <- data.frame(x = rexp(nrow(housingData)), # meaningless x to help show varation
                 y = housingData$LotFrontage,
                 missing = missing)
```

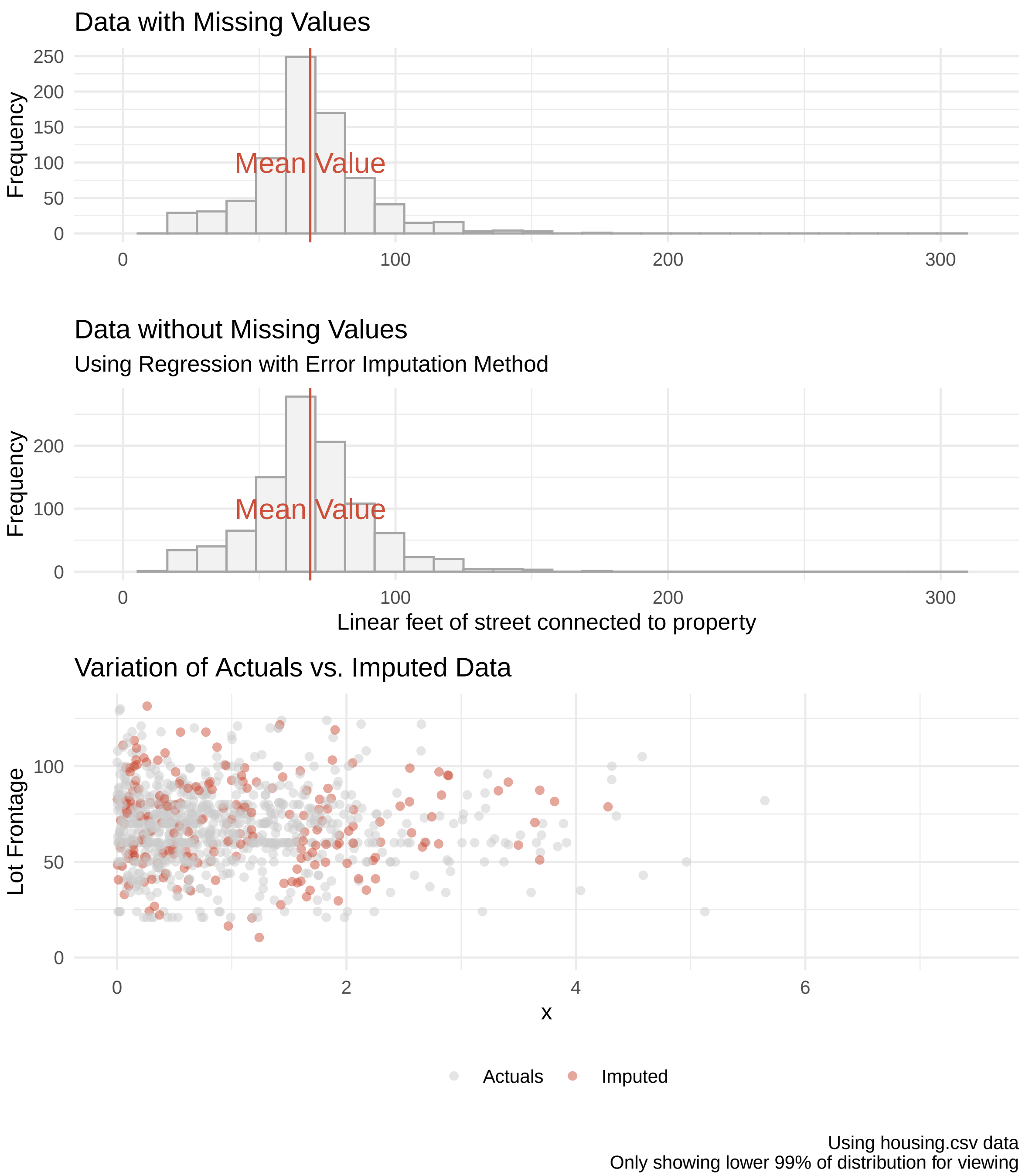
i **Mean Value Imputation**

```
# Create copy of the data with NAs
df.meanInputed <- df

# Conduct Mean imputation
df.meanInputed[missing,"y"] <- mean(df.meanInputed$y, na.rm=T)

# Compare missing vs. non missing for given imputation method
seeImputation(df, df.meanInputed, imputationMethod = 'Mean')
```

### ii Regression with Error Imputation

- Output seems to capture appropriate variance of the actual data.
- It is clear that the mean does not change

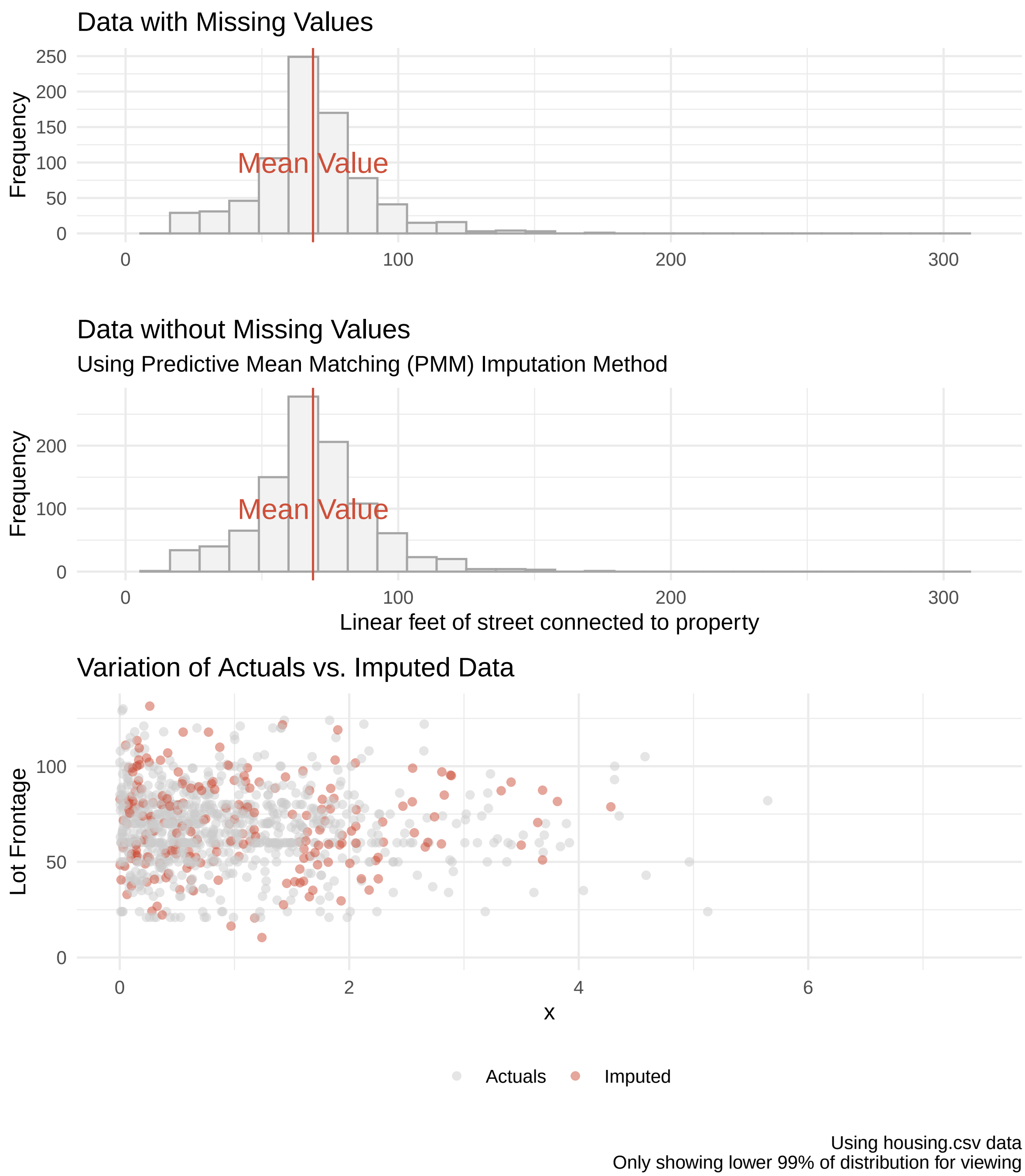
```
fit <- lm(y ~ x, data = df)    # fit a linear model to the data
f   <- summary(fit)

c  <- f[[4]] # extract the coefficients
se <- f[[6]] # extract the model standard error

# Regression with NO error
dfReg.imp <- df
dfReg.imp[missing,"y"]<- (c[1] + c[2] * dfReg.imp[missing,"x"])

# Imputation by Regression with error. Note se = standard error of model
df.regErrorImputed <- dfReg.imp %>%
  mutate(y = y + if_else(missing, rnorm(n(), 0, se), 0))

# Compare missing vs. non missing for given imputation method
seeImputation(df, df.regErrorImputed, imputationMethod = 'Regression with Error')
```


Data with Missing Values
Frequency
250
200
150
100
50
0
Mean Value
0
100
200
300
Data without Missing Values
Using Regression with Error Imputation Method
Frequency
200
100
0
Mean Value
0
100
200
300
Linear feet of street connected to property
Variation of Actuals vs. Imputed Data
Lot Frontage
100
50
0
0
2
4
6
x
Actuals
Imputed
Using housing.csv data
Only showing lower 99% of distribution for viewing

### iii Predictive Mean Matching Imputation

- Output seems to capture appropriate variance of the actual data.
- It is clear that the mean does not change much, if at all.

```
df.pmmImputed <- df # copy data with missingness

# imputation by PMM
df.pmmImputed[missing,"y"] <- mice.impute.pmm( df.pmmImputed$y,
                                              !df.pmmImputed$missing,
                                               df.pmmImputed$x)

# Compare missing vs. non missing for given imputation method
seeImputation(df, df.regErrorImputed,
              imputationMethod = 'Predictive Mean Matching (PMM)')
```


Data with Missing Values
Frequency
250
200
150
100
50
0
Mean Value
0
100
200
300
Data without Missing Values
Using Predictive Mean Matching (PMM) Imputation Method
Frequency
200
100
0
Mean Value
0
100
200
300
Linear feet of street connected to property
Variation of Actuals vs. Imputed Data
Lot Frontage
100
50
0
0
2
4
6
x
Actuals
Imputed
Using housing.csv data
Only showing lower 99% of distribution for viewing

## 2 (c) - Dummy Variable Manipulation

```r
housingData <- housingData %>%

  # lumps all levels except for the n most frequent
  mutate(Exterior1st = fct_lump_n(Exterior1st, n=4))

# See that there are only 5 levels now
unique(housingData$Exterior1st)
```

```
[1] Other   Wd Sdng VinylSd HdBoard MetalSd
Levels: HdBoard MetalSd VinylSd Wd Sdng Other
```

## 2 (d) - More fun with Factors