ISE 5103 Intelligent Data Analytics

Homework 4 - Data Wrangling in R

Daniel Carpenter & Sonaxy Mohanty

September 2022

Contents

Packages

```
# Data Wrangling
library(tidyverse)

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

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

1 - Data Quality Report

1 (a) - Read data

- Created a tibble `housingData` after adding three new variables: `age`, `ageSinceRemodel` and `ageofGarage` to the housing data

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

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

housingData
```

1 (b) - Numeric Housing Tibble

- Created a tibble named `housingNumeric` which contained all of the numeric variables from the original data using the `dplyr::select` command along with the `is.numeric` function

```
housingNumeric <- housingData %>%
  #selecting all the numeric data
  dplyr::select_if(is.numeric) %>%
  #converting the dataframe to tibble
  as_tibble()

housingNumeric
```

```
## # A tibble: 1,000 x 39
```

```
##       Id MSSub~1 LotFr~2 LotArea Overa~3 Overa~4 YearB~5 YearR~6 MasVn~7 BsmtF~8
##    <int>   <int>   <int>   <int>   <int>   <int>   <int>   <int>   <int>   <int>
##  1     1      20      NA   11000       5       6    1966    1966     200     740
##  2     2      20      NA   36500       5       5    1964    1964     621     812
##  3     3      20      57    9764       5       7    1967    2003       0     702
##  4     4      70      NA    7500       6       7    1942    1950       0     547
##  5     5      20      80    9200       6       6    1965    1965       0     892
##  6     6      60      72   11317       7       5    2003    2003     101       0
##  7     7      20      80    8480       5       6    1963    1963       0     630
##  8     8      70      65   11700       7       7    1880    2003       0       0
##  9     9      60      80    9760       6       6    1964    1964     360     674
## 10    10      60      93   10261       6       5    2000    2000     318       0
## # ... with 990 more rows, 29 more variables: BsmtFinSF2 <int>, BsmtUnfSF <int>,
## #   TotalBsmtSF <int>, X1stFlrSF <int>, X2ndFlrSF <int>, LowQualFinSF <int>,
## #   GrLivArea <int>, BsmtFullBath <int>, BsmtHalfBath <int>, FullBath <int>,
## #   HalfBath <int>, BedroomAbvGr <int>, KitchenAbvGr <int>, TotRmsAbvGrd <int>,
## #   Fireplaces <int>, GarageYrBlt <int>, GarageCars <int>, GarageArea <int>,
## #   WoodDeckSF <int>, OpenPorchSF <int>, EncPorchSF <int>, PoolArea <int>,
## #   MiscVal <int>, MoSold <int>, YrSold <int>, SalePrice <int>, age <int>, ...
```

1 (c) - Factor Housing Tibble

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

```
housingFactor <- housingData %>%

  #converting the character data into factors
  transmute_if(is.character, as.factor) %>%
  #converting the dataframe to tibble
  as_tibble()

housingFactor
```

```
## # A tibble: 1,000 x 38
##    MSZon~1 Alley LotSh~2 LandC~3 LotCo~4 LandS~5 Neigh~6 Condi~7 BldgT~8 House~9
##    <fct>   <fct> <fct>   <fct>   <fct>   <fct>   <fct>   <fct>   <fct>   <fct>
##  1 RL      <NA>  IR1     Lvl     CulDSac Gtl     NAmes   Norm    1Fam    1Story
##  2 RL      <NA>  IR1     Low     Inside  Mod     ClearCr Norm    1Fam    1Story
##  3 RL      <NA>  IR1     Lvl     other   Gtl     Sawyer  Feedr   1Fam    1Story
##  4 RL      <NA>  IR1     Bnk     Inside  Gtl     Crawfor Norm    1Fam    2Story
##  5 RL      <NA>  Reg     Lvl     Inside  Gtl     NAmes   Norm    1Fam    1Story
##  6 RL      <NA>  Reg     Lvl     Inside  Gtl     CollgCr Norm    1Fam    2Story
##  7 RL      <NA>  Reg     Lvl     Corner  Gtl     Sawyer  Norm    1Fam    1Story
##  8 RM      Pave  IR1     Lvl     Corner  Gtl     OldTown Norm    1Fam    2Story
##  9 RL      <NA>  Reg     Lvl     Inside  Mod     NAmes   Norm    1Fam    2Story
## 10 RL      <NA>  IR1     Lvl     Inside  Gtl     Gilbert Norm    1Fam    2Story
## # ... with 990 more rows, 28 more variables: RoofStyle <fct>,
## #   Exterior1st <fct>, Exterior2nd <fct>, MasVnrType <fct>, ExterQual <fct>,
## #   ExterCond <fct>, Foundation <fct>, BsmtQual <fct>, BsmtCond <fct>,
## #   BsmtExposure <fct>, BsmtFinType1 <fct>, BsmtFinType2 <fct>, Heating <fct>,
## #   HeatingQC <fct>, CentralAir <fct>, Electrical <fct>, KitchenQual <fct>,
## #   Functional <fct>, FireplaceQu <fct>, GarageType <fct>, GarageFinish <fct>,
```

```
## #   GarageQual <fct>, GarageCond <fct>, PavedDrive <fct>, PoolQC <fct>, ...
```

1 (d) - Use Glimpse

- Using `glimpse` function, we verified the output of `housingNumeric` and `housingFactor` tibbles

```
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`

```
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

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

```
# Vector of functions for summary of numeric variables
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))
}
```

1 (g) - Apply Summary Stats

- Applied the user-defined summary stats function to every variable in the `housingNumeric` data set and saving the new tibble as `numericSummary`

```
numericSummary <- housingNumeric %>%
  #applying the function myNumericSummary to every column across the housingNumeric dataset
  summarise(across(.cols=everything(), ~myNumericSummary(.x)))
```

1 (h) - Add Stats Names

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

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

#glimpse(numericSummary)
```

1 (i) - Pretty up data

- Transformed 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())
```

- Output

```
options(digits=3)
options(scipen=99)
numericSummaryFinal %>%
  #knitr::kable('pipe', table.attr = "style='width:40%;'")
  knitr::kable(format='latex', align=c('l', 'r', 'r','r', 'r', 'r', 'r', 'r',
                                'r', 'r', 'r', 'r', 'r'), booktabs=TRUE) %>%
  kable_styling(font_size=6)
```

1 (j) - Factor Data Report

- Created user-defined function to identify `first`, `second` or `least common` modes

```
getmodes <- function(v,type=1) {
tbl <- table(v)
m1<-which.max(tbl)
if (type==1) {
  #1st mode
return (names(m1))
}
else if (type==2) {
  #2nd mode
return (names(which.max(tbl[-m1])))
}
else if (type==-1) {
  #least common mode
return (names(which.min(tbl)))
}
```

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
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	174560.607	39300	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

```
else {
stop("Invalid type selected")
}
}
```

- Created another user-defined function to identify the `frequencies` of the first, second, or least common modes

```
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")
}
}
```

- User-defined function that will help apply several summary statistics to our data all at once

```
# Vector of functions for summary of categorical variables
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), getmodesCnt(x, type=-1))
}
```

- Applied the user-defined summary stats function to every variable in the `housingFactor` data set and saving the new tibble as `factorSummary`

```
factorSummary <- housingFactor %>%
  #applying the function myNumericSummary to every column across the housingNumeric dataset
  summarise(across(.cols=everything(), ~myFactorSummary(.x)))
```

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

```
factorSummary <- cbind(
  stat=c("n","unique","missing","firstmode","firstmode_freq","secondmode",
         "secondmode_freq","leastcommon","leastcommon_freq"),
  factorSummary)

#glimpse(factorSummary)
```

- Transformed data to make it ready for output format

variable	n	missing	missing_pct	unique	unique_pct	freqRatio	1st mode	1st mode freq	2nd mode	2nd mode freq	least common	least common freq
MSZoning	1000	0	0.0	4	0.4	5.32	RL	803	RM	151	RH	10
Alley	1000	938	93.8	3	0.3	1.82	Grvl	40	Pave	22	Pave	22
LotShape	1000	0	0.0	4	0.4	1.92	Reg	633	IR1	330	IR3	7
LandContour	1000	0	0.0	4	0.4	22.62	Lvl	905	Bnk	40	Low	26
LotConfig	1000	0	0.0	4	0.4	3.97	Inside	711	Corner	179	other	38
LandSlope	1000	0	0.0	3	0.3	19.71	Gtl	946	Mod	48	Sev	6
Neighborhood	1000	0	0.0	18	1.8	1.48	NAmes	167	CollgCr	113	Timber	20
Condition1	1000	0	0.0	6	0.6	17.08	Norm	871	Feedr	51	PosA	7
BldgType	1000	0	0.0	5	0.5	10.33	1Fam	837	TwnhsE	81	2fmCon	20
HouseStyle	1000	0	0.0	8	0.8	1.57	1Story	488	2Story	310	2.5Fin	5
RoofStyle	1000	0	0.0	3	0.3	4.32	Gable	795	Hip	184	other	21
Exterior1st	1000	0	0.0	8	0.8	1.87	VinylSd	328	HdBoard	175	CemntBd	36
Exterior2nd	1000	0	0.0	9	0.9	2.01	VinylSd	320	HdBoard	159	BrkFace	24
MasVnrType	1000	4	0.4	5	0.5	1.97	None	617	BrkFace	313	BrkCmn	8
ExterQual	1000	0	0.0	3	0.3	1.96	Avg	657	AboveAvg	336	BelowAvg	7
ExterCond	1000	0	0.0	3	0.3	8.54	Avg	880	AboveAvg	103	BelowAvg	17
Foundation	1000	0	0.0	4	0.4	1.12	CBlock	463	PConc	414	other	27
BsmtQual	1000	31	3.1	4	0.4	1.06	AboveAvg	488	Avg	459	BelowAvg	22
BsmtCond	1000	31	3.1	4	0.4	24.41	Avg	903	AboveAvg	37	BelowAvg	29
BsmtExposure	1000	32	3.2	5	0.5	4.77	No	668	Av	140	Mn	76
BsmtFinType1	1000	31	3.1	7	0.7	1.03	GLQ	273	Unf	265	LwQ	52
BsmtFinType2	1000	32	3.2	7	0.7	23.69	Unf	853	Rec	36	ALQ	11
Heating	1000	0	0.0	2	0.2	37.46	GasA	974	other	26	other	26
HeatingQC	1000	0	0.0	3	0.3	2.21	AboveAvg	664	Avg	300	BelowAvg	36
CentralAir	1000	0	0.0	2	0.2	14.62	Y	936	N	64	N	64
Electrical	1000	1	0.1	5	0.5	12.61	SBrkr	908	FuseA	72	FuseP	2
KitchenQual	1000	0	0.0	3	0.3	1.22	Avg	534	AboveAvg	439	BelowAvg	27
Functional	1000	0	0.0	6	0.6	35.54	Typ	924	Min2	26	Maj2	4
FireplaceQu	1000	466	46.6	4	0.4	1.04	AboveAvg	250	Avg	240	BelowAvg	44
GarageType	1000	53	5.3	7	0.7	2.15	Attchd	601	Detchd	280	2Types	3
GarageFinish	1000	53	5.3	4	0.4	1.49	Unf	434	RFn	291	Fin	222
GarageQual	1000	53	5.3	4	0.4	27.48	Avg	907	BelowAvg	33	AboveAvg	7
GarageCond	1000	53	5.3	4	0.4	29.36	Avg	910	BelowAvg	31	AboveAvg	6
PavedDrive	1000	0	0.0	3	0.3	14.71	Y	912	N	62	P	26
PoolQC	1000	998	99.8	3	0.3	1.00	Fa	1	Gd	1	Fa	1
Fence	1000	805	80.5	5	0.5	2.70	MnPrv	108	GdPrv	40	MnWw	8
MiscFeature	1000	966	96.6	3	0.3	16.00	Shed	32	Othr	2	Othr	2
SaleType	1000	0	0.0	2	0.2	33.48	WD	971	other	29	other	29

```
factorSummaryFinal <- factorSummary %>%
  pivot_longer("MSZoning":"SaleType", names_to = "variable",
               values_to = "value") %>%
  pivot_wider(names_from = stat, values_from = value) %>%

  #converting n, missing and unique variables from char to numeric datatype
  mutate(missing_pct = 100*as.numeric(missing)/as.numeric(n),
         unique_pct = 100*as.numeric(unique)/as.numeric(n),
         freqRatio = as.numeric(firstmode_freq)/as.numeric(secondmode_freq)) %>%
  select(variable, n, missing, missing_pct, unique, unique_pct, freqRatio,
         everything())
```

- Output

```
options(digits=3)
options(scipen=99)
factorSummaryFinal %>%
  knitr::kable(col.names = c("variable","n","missing","missing_pct", "unique",
                             "unique_pct","freqRatio","1st mode","1st mode freq",
                             "2nd mode", "2nd mode freq","least common",
                             "least common freq"),
               format='latex',
               align=c('l', 'r', 'r','r', 'r', 'r', 'r','r','r', 'r', 'r', 'r', 'r'),
               booktabs=TRUE) %>%
  kable_styling(font_size=4)
```

2 - Transformation

2 (a) - Fixing Skewed Data

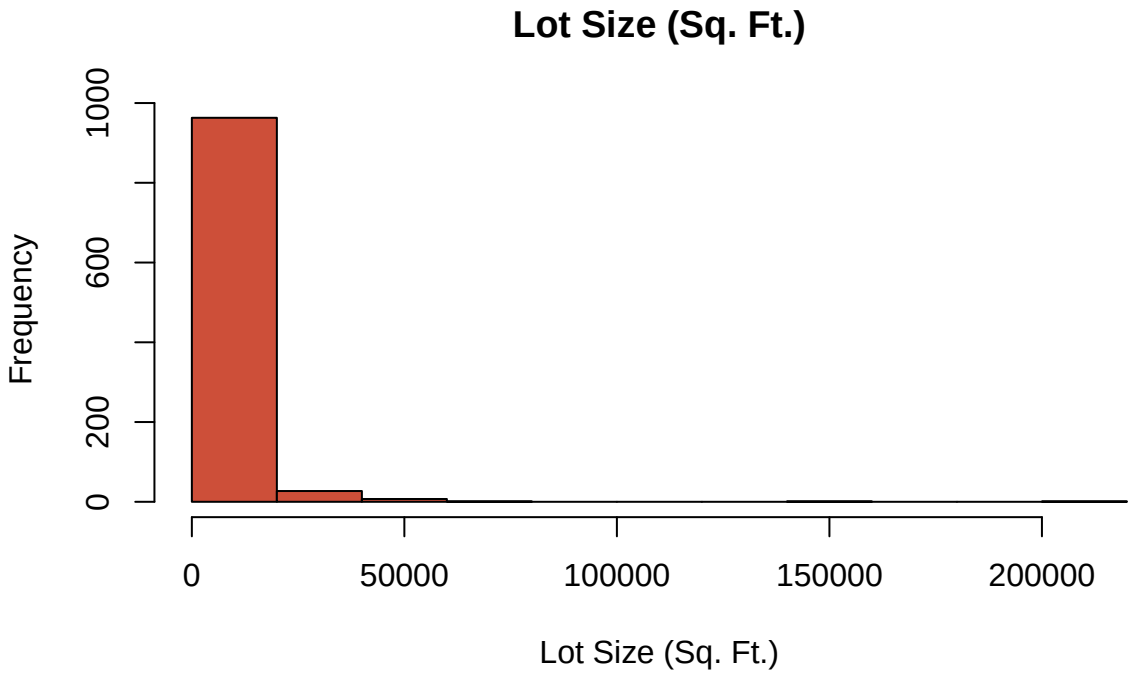
Function to Convert Skewed Data to Normally Distributed Vector

```
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'
```

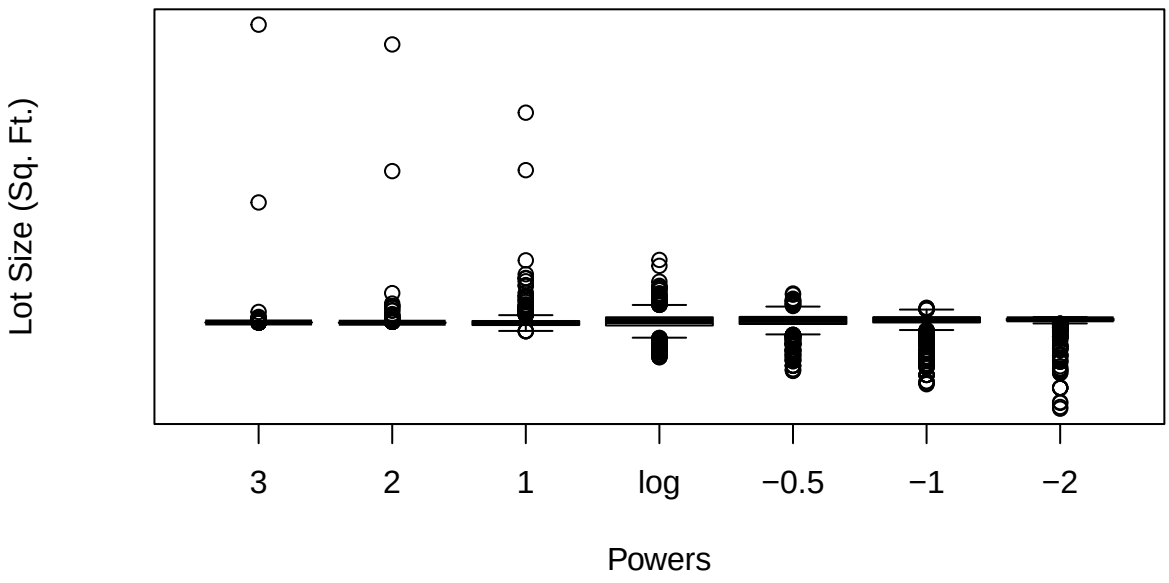
i. Fix LotArea in Housing Data Set

```


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

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

Lot area is highly skewed




```
# 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)
```

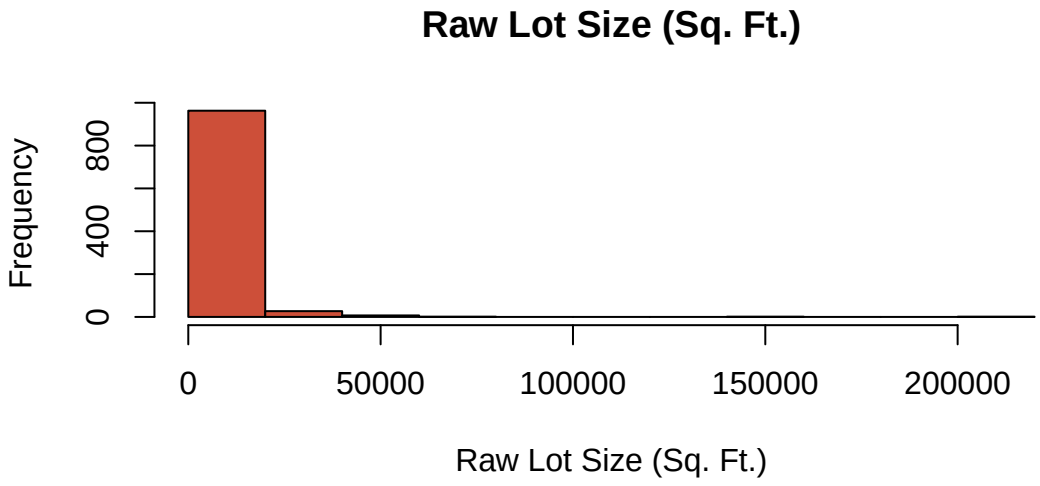


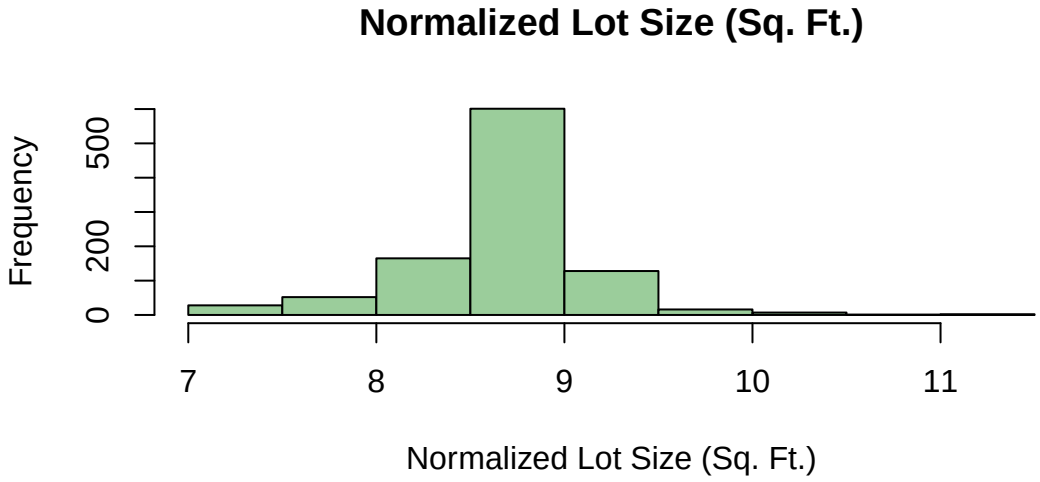
```
# Normalize the data and store in data
housingData <- housingData %>%
  mutate(normLotArea = normalizeDist(housingData$LotArea) )
```

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)
```

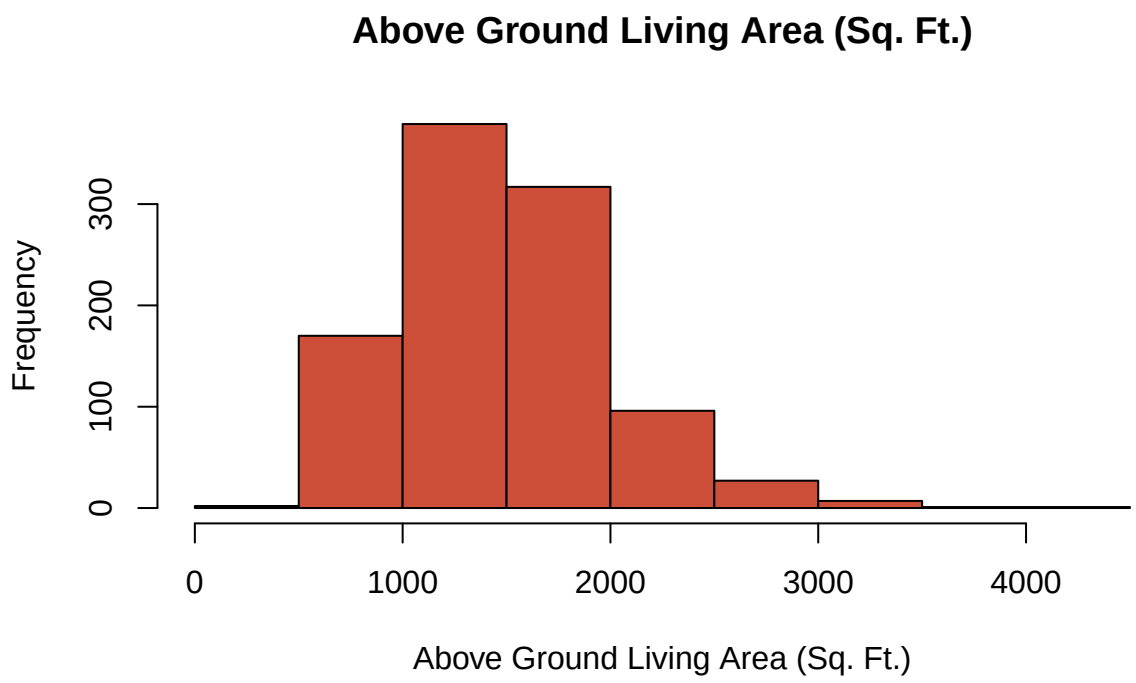

Raw Lot Size (Sq. Ft.)
Frequency
0
400
800
0
50000
100000
150000
200000
Raw Lot Size (Sq. Ft.)


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

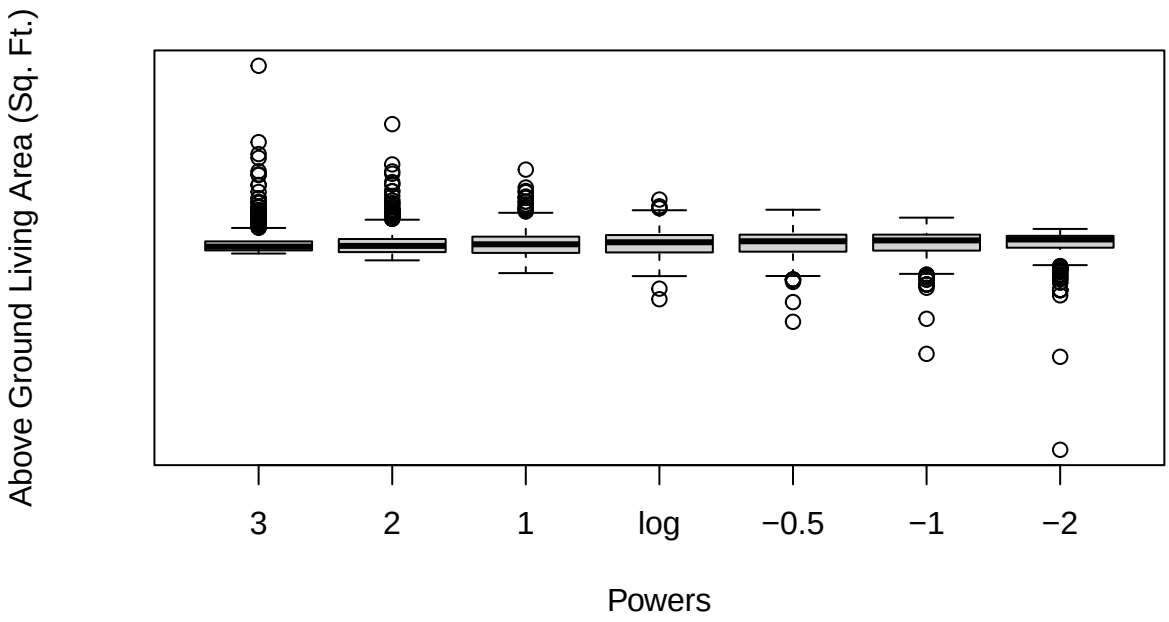
i. Fix `GrLivArea` in Housing Data Set

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

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

Above Ground Living Area is highly skewed




```
# 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)
```

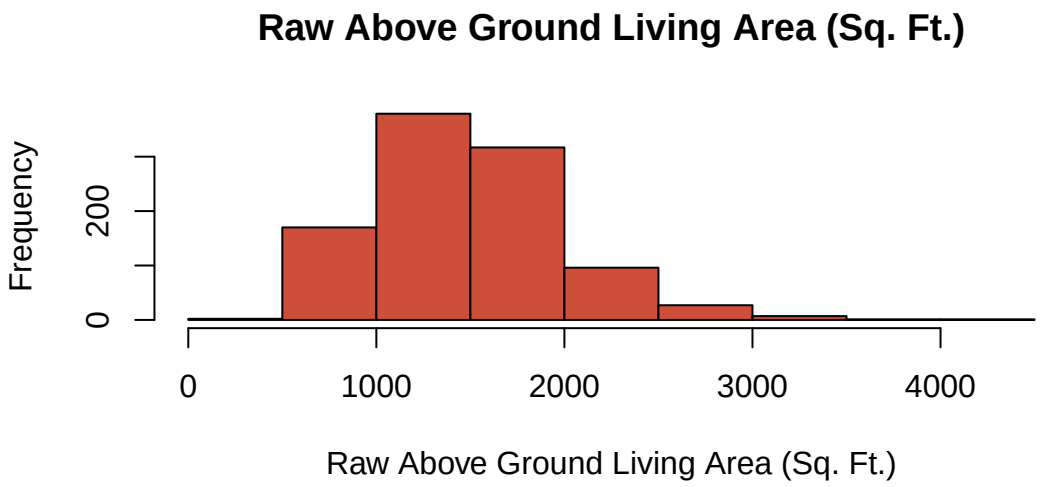


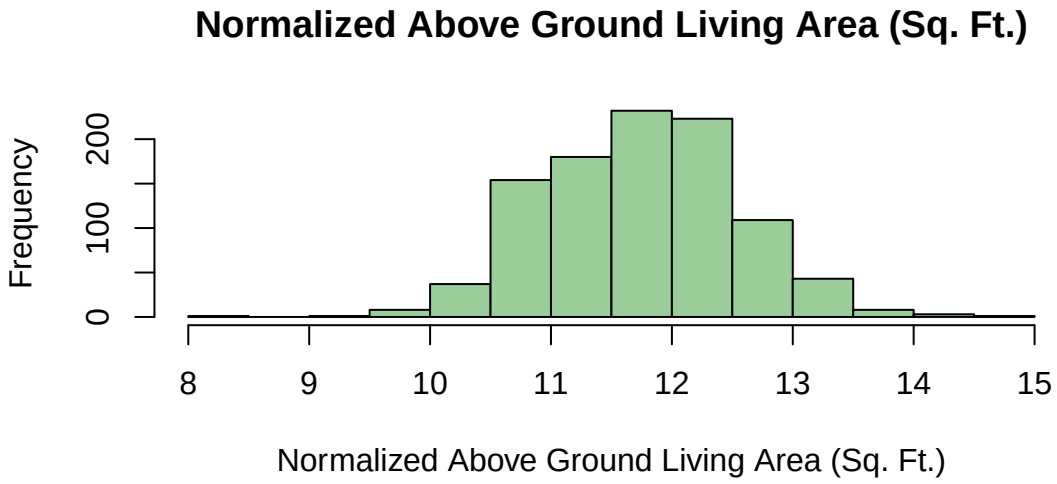

```
# Normalize the data and store in data
housingData <- housingData %>%
  mutate(normYearBuilt = normalizeDist(housingData$GrLivArea) )
```

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',
         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`

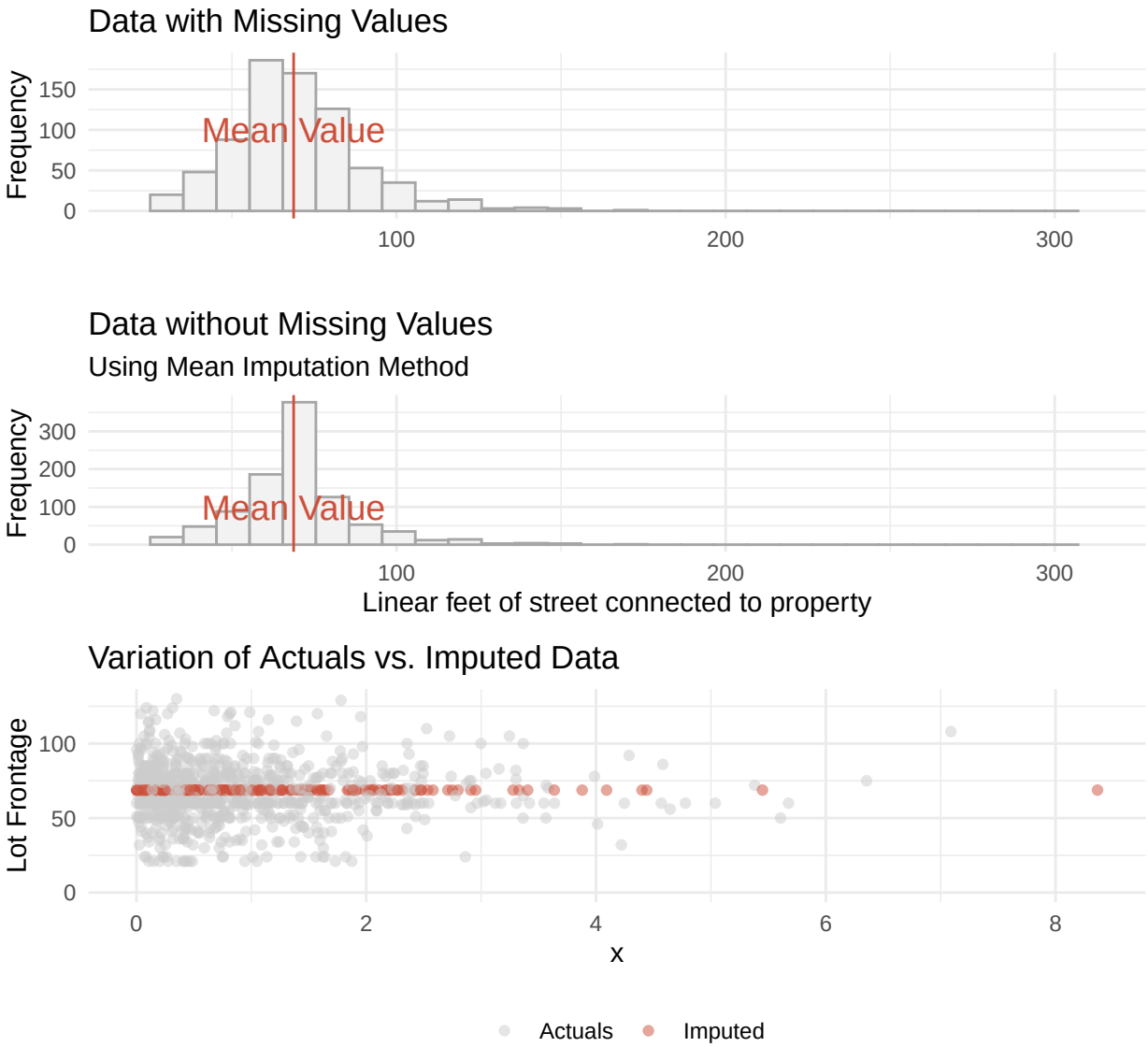
```
# How much is missing?
missing <- is.na(housingData$LotFrontage)
paste('There are', sum(missing), 'missing values')
```

```
## [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')
```

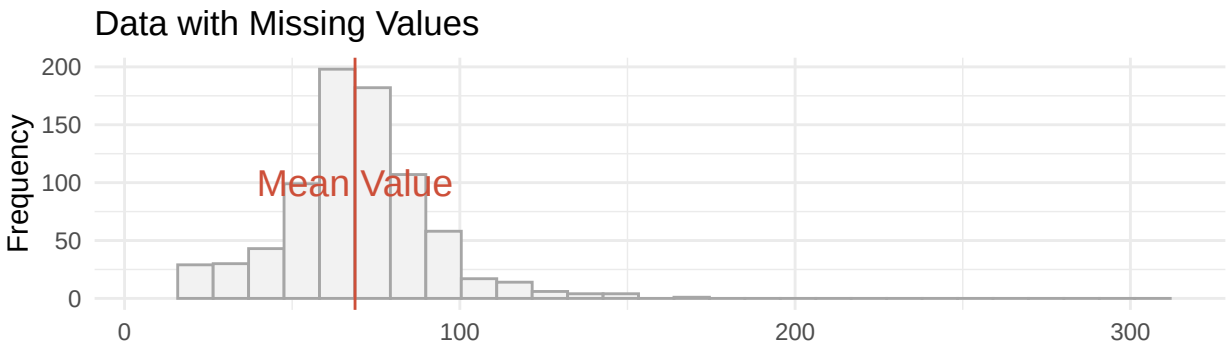


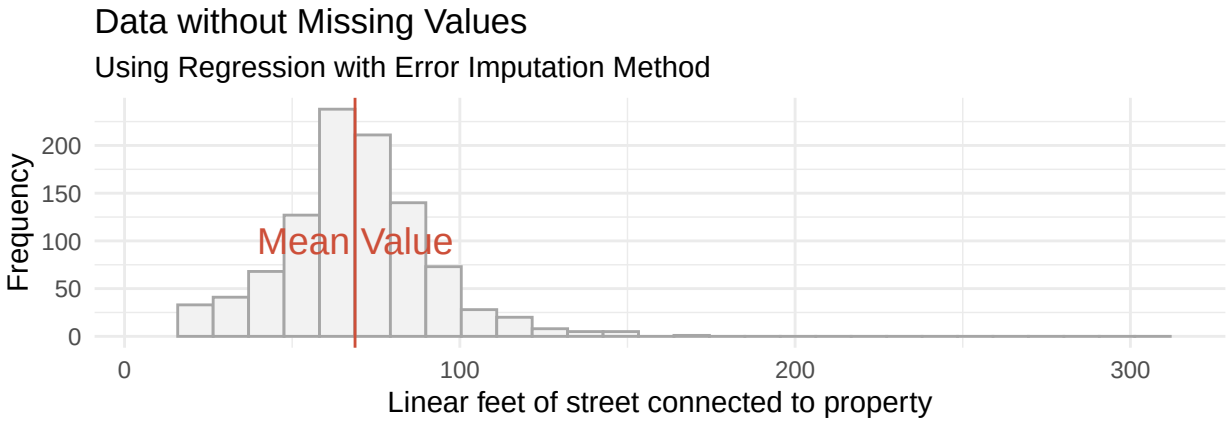
Using housing.csv data
Only showing lower 99% of distribution for viewing

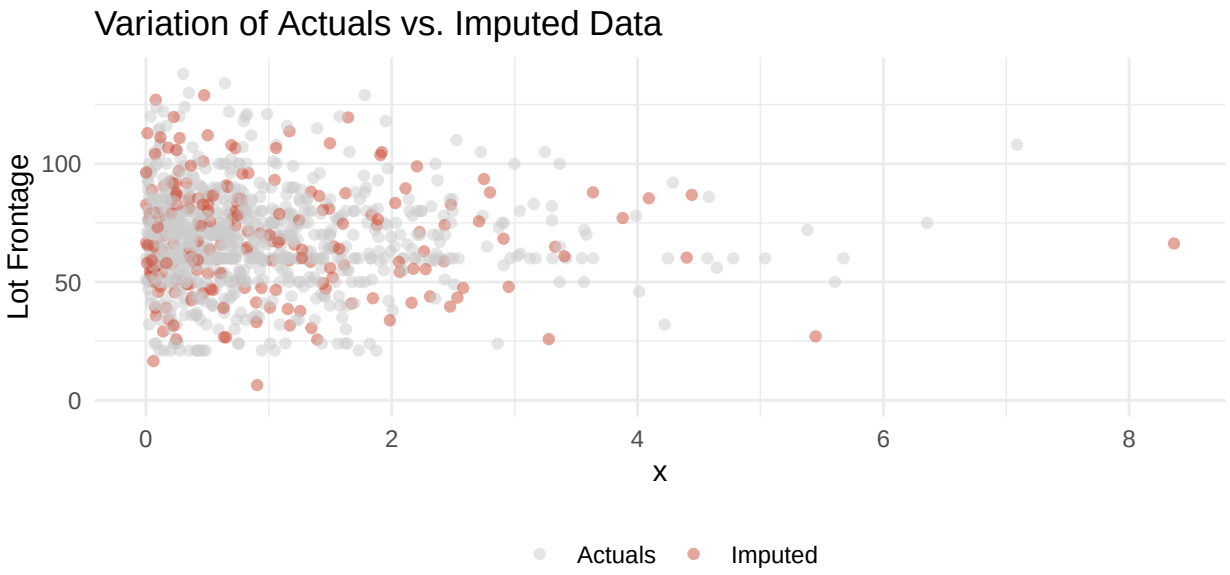
ii Regression with Error Imputation

- 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')
```





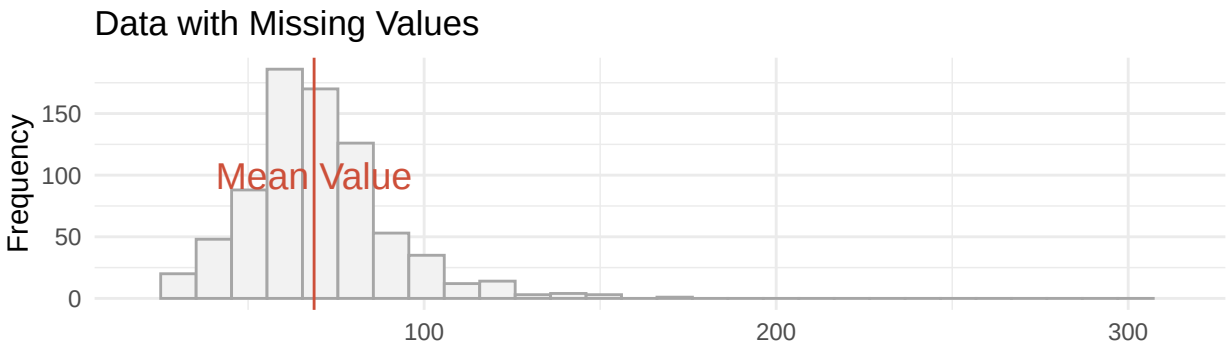


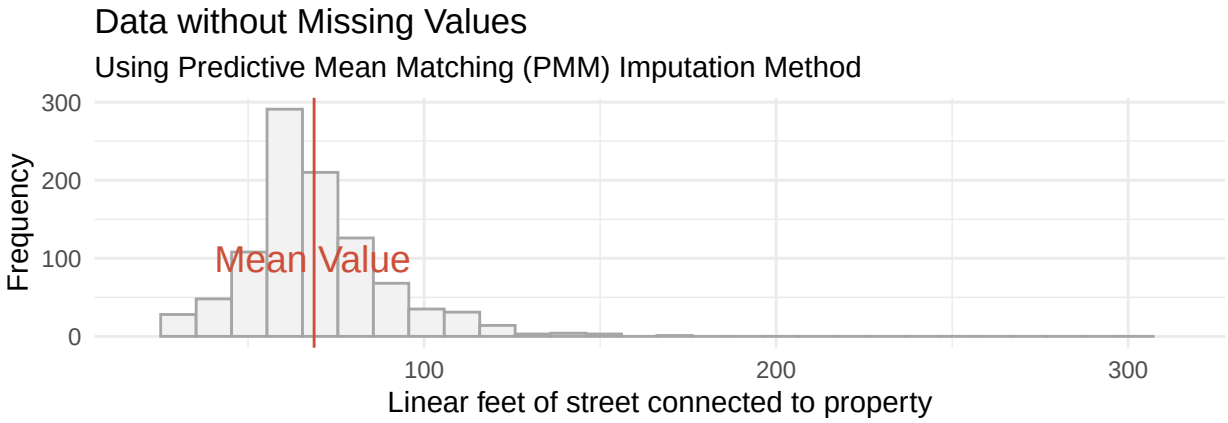
Using housing.csv data
Only showing lower 99% of distribution for viewing

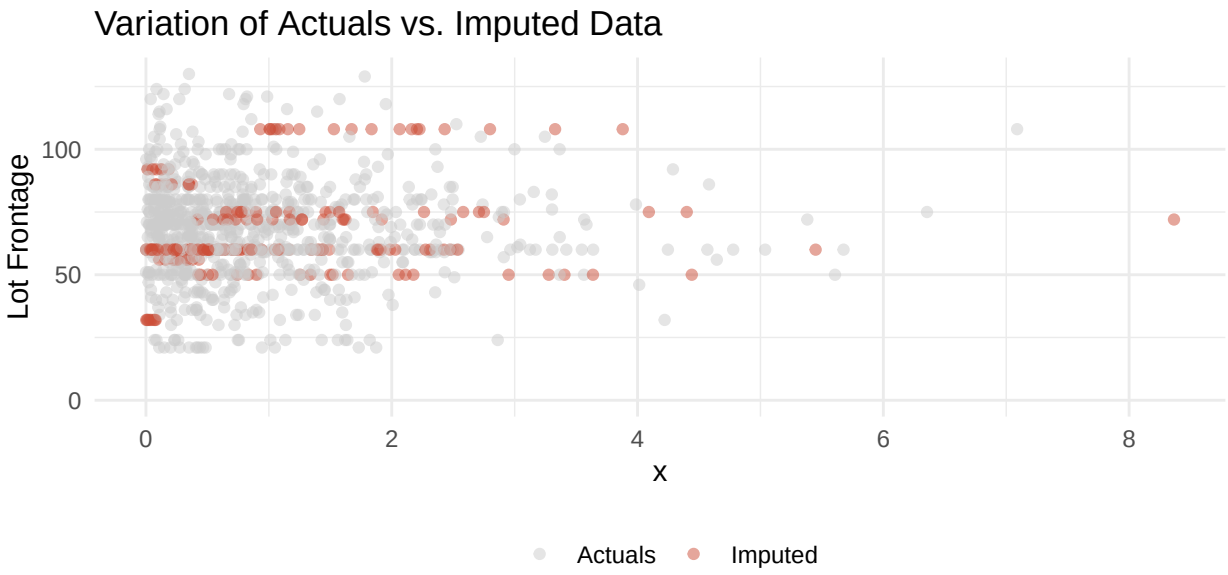
iii Predictive Mean Matching Imputation

- 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.pmmImputed,
              imputationMethod = 'Predictive Mean Matching (PMM)')
```







Using housing.csv data
Only showing lower 99% of distribution for viewing

2 (c) - Dummy Variable Manipulation

Collapse the factor levels in the Exterior1st down to only five levels – the first four levels should be the most frequent levels and all other levels should be collapsed into a single "Other" level.

Interpretation

- Now there are only 5 levels for the `Exterior1st` variable
- 4 most frequent left untouched, and the others grouped in `Other`

```
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
```

```
length(unique(housingData$Exterior1st))
```

```
## [1] 5
```

2 (d) - More fun with Factors

i - Get average SalePrice for each Neighborhood using tidyverse

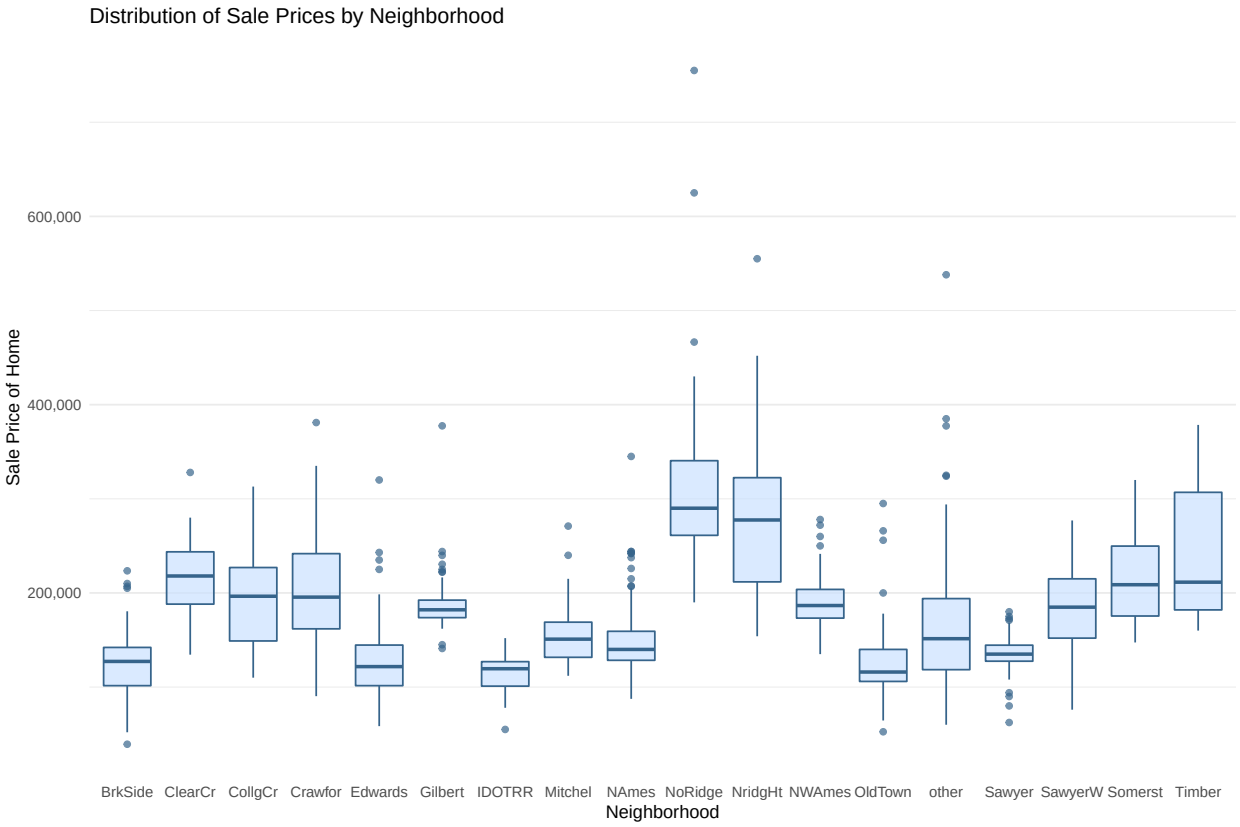
- Note sorted descending

```
housingData %>%
  group_by(Neighborhood) %>%
  summarise(AvgSalePrice = mean(SalePrice)) %>%
  arrange(desc(AvgSalePrice)) %>% # sort descending on sale price
  kable() # output as a kable table
```

Neighborhood	AvgSalePrice
NoRidge	328794
NridgHt	283057
Timber	241940
ClearCr	218265
Somerst	211678
Crawfor	209766
CollgCr	194942
NWAmes	191823
Gilbert	189466
SawyerW	183971
other	170248
Mitchel	154788
NAmes	146669
Sawyer	134708
Edwards	128772
OldTown	126023
BrkSide	124844
IDOTRR	114319

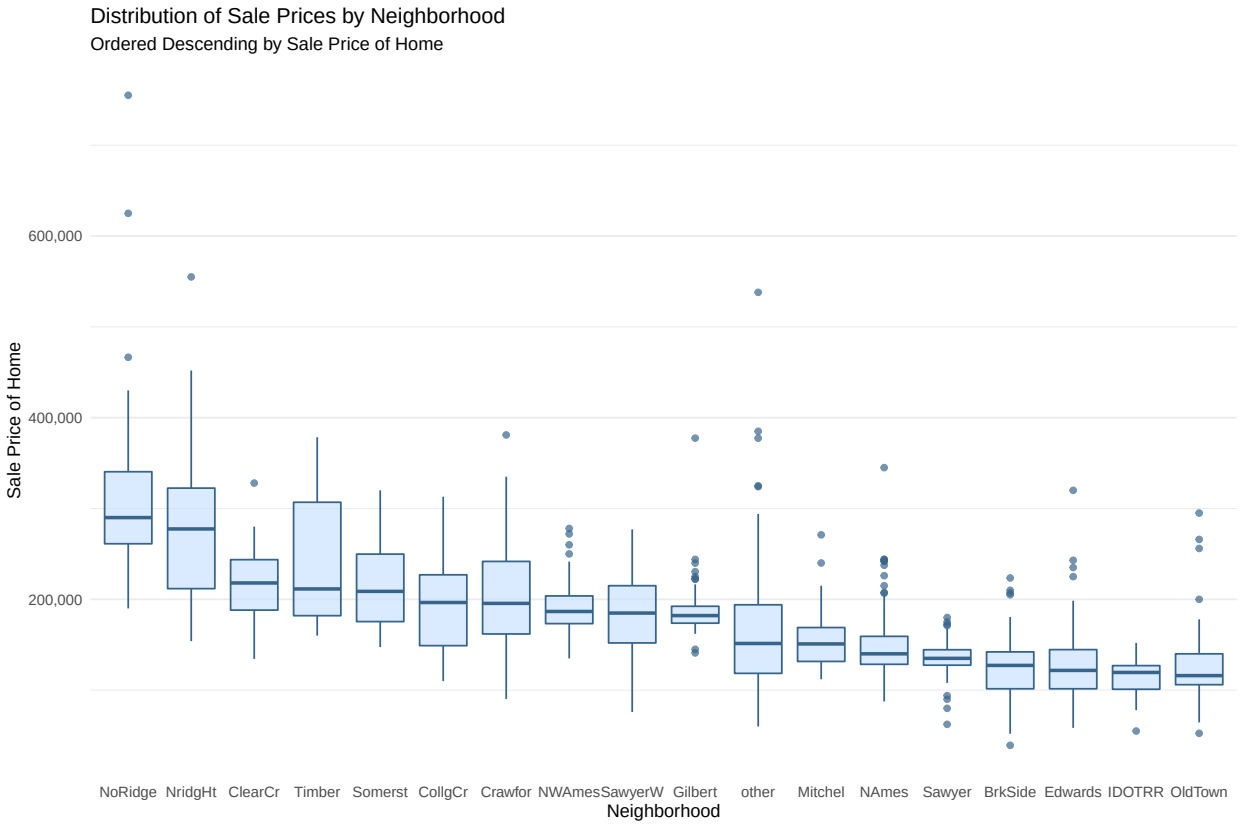
ii - Boxplots of Avg. Sale Price by Neighborhood

```
housingData %>%
  ggplot(aes(x = Neighborhood,
             y = SalePrice) ) +

  # Boxplots
  geom_boxplot(color = 'steelblue4', fill = 'lightsteelblue1', alpha = 0.7) +

  # Theme, y scale format, and labels
  theme_minimal() + theme(panel.grid.major.x = element_blank()) +
  scale_y_continuous(labels = comma) +
  labs(title = 'Distribution of Sale Prices by Neighborhood',
       x     = 'Neighborhood',
       y     = 'Sale Price of Home')
```



`iii-iv` - Boxplots Reordered by Descending Sale Price

```
housingData %>%
  ggplot(aes(x = fct_reorder(Neighborhood, desc(SalePrice) ),
             y = SalePrice) ) +

  # Boxplots
  geom_boxplot(color = 'steelblue4', fill = 'lightsteelblue1', alpha = 0.7) +

  # Theme, y scale format, and labels
  theme_minimal() + theme(panel.grid.major.x = element_blank()) +
  scale_y_continuous(labels = comma) +
  labs(title = 'Distribution of Sale Prices by Neighborhood',
       subtitle = 'Ordered Descending by Sale Price of Home',
       x     = 'Neighborhood',
       y     = 'Sale Price of Home')
```



References

1. https://dplyr.tidyverse.org/reference/across.html
2. https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/options
3. https://cran.r-project.org/web/packages/kableExtra/vignettes/awesome_table_in_pdf.pdf