# PREDICTING HOME ZONING GROUPS USING HOME ATTRIBUTES

## Dasha Asienga

## Introduction

### Purpose of the Analysis

Real property can be grouped into different zoning classifications such as agricultural, commercial, industrial, residential high density, and residential low density. Such zoning relates to its general use and is important in determining property rights. Classifying real estate is important because it is one of the most fundamental ways to understand risk and reward, and to also understand potential for future investments. It also allows states to tax different properties and can sometimes be indicative of inequities within society. High density areas may have different living conditions and living standards as compared to low density areas, for example.

Some work has been done in trying to group or classify real estate property. Some previous work includes building models that can classify real estate according to quality using different characteristics of the property. Other models have been built to try and determine how much loan banks can viably lend to prospective home-owners and home-builders. Notably, most of the work is geared towards predicting risk or reward and answer questions such as, "Is this property a worthy investment?" and "Is this property good quality, taking all risk factors into account?". However, zoning groups are usually pre-assigned. For example, a piece of property is either agricultural or residential. Permitting some little overlap, most properties fall largely into a unique zoning classification.

Therefore, the purpose of this analysis is to investigate whether there are telling characteristics of what zoning group a piece of property belongs to. My key research question is – can we classify real estate into zoning groups based on other characteristics of the property? In particular, I'm interested in investigating whether variables such as price of the home, property condition, the square footage, and the number of bedrooms, to name a few, hold information on what zoning group a piece of property belongs to, which can help us gain insight on the housing industry.

### The Data Set

The data set used in this analysis is from the American Statistical Association and was compiled by Dean De Cock for use in data science education (Kaggle, 2022). It's an alternative for data scientists looking for a modernized and expanded version of the often cited Boston Housing data set (Kaggle, 2022). The homes included in the data set are from the Boston area, which has about 300,000 homes (Kaggle, 2022). The sample we will be using for our analysis contains 1,460 observations. There are 81 variables in the data set, with a mix of both categorical and quantitative variables.

My main categorical variable is the general zoning classification of the sale. It has 8 levels:

```
A    Agriculture
C    Commercial
FV   Floating Village Residential
I    Industrial
RH   Residential High Density
RL   Residential Low Density
RP   Residential Low Density Park
RM   Residential Medium Density
```

### Statistical Techniques

For this analysis, we will be using clustering and classification as our main statistical techniques.

Clustering, a form of unsupervised learning, refers to any set of methods used to find natural groups of objects. We will run different clustering algorithms to assess whether any of the natural groups recover our zoning class. Because we have many different variables, we will also be employing Principal Components Analysis (PCA), which is a dimension reduction technique, to visualize our clustering solution.

Classification, on the other hand, is a supervised learning technique that allows you to use predictor variables to make "rules" which classify observations according to categories of a categorical response variable, in this case, the zoning class. We will run different methods in an aim to maximize the accuracy of the model.

## Preliminary Analysis

```
test <- read.csv("Home Data/test.csv")
train <- read.csv("Home Data/train.csv")
data <- union_all(test, train)
```

### Subsetting the Data

The data loaded has 2,919 observations. Note that the final data set is a union of both the test and train data sets obtained from my source. The reason that there were 2 data sets from the source is that the data was originally intended for a data challenge predicting price.

As such, the test data does not contain any information on `SalePrice`.

We need to determine if sales price may be an important predictor. If so, then we will proceed to perform the entire analysis with the train data set from Kaggle, which contains 1460 observations.

Based on the density plot below colored by zoning class, there seems to be some reasonable separation between different zoning classes by sales price. Therefore, this may prove to be a useful classification variable.

```
ggplot(data, mapping = aes(x = SalePrice, fill = MSZoning)) +
  geom_density() +
  labs(title = "Density Plot of Sale Price by Zoning Groups") +
  theme_classic()
```


Let's begin by dropping all variables for which `SalePrice` is null. This leaves us with 1460 observations (identical to the train data set).

```
data <- data %>%
  filter(!is.na(SalePrice))
```

### Choosing Variables

As mentioned above, there are 81 variables with a mix of categorical and quantitative variables. Some categorical variables include the general zoning classification, type of road access, type of alley access, and the shape of the home. Some categorical variables are numerically coded as well, such as quality of the kitchen. Some quantitative variables include the valued price, lot size, number of bathrooms, number of bedrooms, and size of the garage.

We will use our univariate and bivariate analysis as well as other techniques to narrow down the number of variables since they may not all be useful in the data analysis.

First, let's drop the `ID` variable, which is an identifier variable, as well as quantitative variables with a lot of null values because our main methods, clustering and classification, do not work well with missing data. This will allow us to perform a complete-case analysis and preserve the number of observations we have by only dropping columns with a lot of null values instead. We will use the summary command to identify such variables.

```
data <- select(data, -c(Id, LotFrontage, MasVnrArea, GarageYrBlt))
dim(data)
```

```
## [1] 1460   77
```

We now have 77 variables in our data set.

Next, let's determine which variables in the data set are categorical. Note that the two main methods in this analysis will be clustering and classification. Clustering works only with quantitative variables, and while classification can work with categorical variables, they have to be numerically coded. Following that, let's remove the categorical variables that are not numerically coded, with the exception of zoning, our response variable.

```
data <- select(data, -c(Street, Alley, LotShape, LandContour, Utilities, LotConfig,
                        LandSlope, Neighborhood, Condition1, Condition2, BldgType,
                        HouseStyle, RoofStyle, RoofMatl, Exterior1st, Exterior2nd,
                        MasVnrType, ExterQual, ExterCond, Foundation, BsmtQual,
                        BsmtCond, BsmtExposure, BsmtFinType1, BsmtFinType2, Heating,
                        HeatingQC, CentralAir, Electrical, KitchenQual, Functional,
                        FireplaceQu, GarageType, GarageFinish, GarageQual, GarageCond,
                        PavedDrive, SaleType, SaleCondition, PoolQC, Fence, MiscFeature))
dim(data)
```

```
## [1] 1460   35
```

We now have 35 variables in our data set.

Finally, let's remove all the date variables – that is, variables encoding month and year – as we will not be using those in our analysis.

```
data <- select(data, -c(YearBuilt, YearRemodAdd, MoSold, YrSold))
dim(data)
```

```
## [1] 1460   31
```

Our final data set has 31 variables and 1460 observations.

Before we proceed, let's make sure that R is treating the numerical categorical variables as categorical.

```
data <- data%>%
  mutate(MSZoning = as.factor(MSZoning),
         MSSubClass = as.factor(MSSubClass),
         OverallQual = as.factor(OverallQual),
         OverallCond = as.factor(OverallCond))
```

Note that because the variables are on different scales, it will be necessary to scale our data when performing our clustering analysis to make sure that some variables don't dominate the solution.

```
head(data, 3)
```

```
##   MSSubClass MSZoning LotArea OverallQual OverallCond BsmtFinSF1 BsmtFinSF2
## 1         60       RL    8450           7           5        706          0
## 2         20       RL    9600           6           8        978          0
## 3         60       RL   11250           7           5        486          0
##   BsmtUnfSF TotalBsmtSF X1stFlrSF X2ndFlrSF LowQualFinSF GrLivArea BsmtFullBath
## 1       150         856       856       854            0      1710            1
## 2       284        1262      1262         0            0      1262            0
## 3       434         920       920       866            0      1786            1
##   BsmtHalfBath FullBath HalfBath BedroomAbvGr KitchenAbvGr TotRmsAbvGrd
## 1            0        2        1            3            1            8
## 2            1        2        0            3            1            6
## 3            0        2        1            3            1            6
##   Fireplaces GarageCars GarageArea WoodDeckSF OpenPorchSF EnclosedPorch
## 1          0          2        548          0          61             0
## 2          1          2        460        298           0             0
## 3          1          2        608          0          42             0
##   X3SsnPorch ScreenPorch PoolArea MiscVal SalePrice
## 1          0           0        0       0    208500
## 2          0           0        0       0    181500
## 3          0           0        0       0    223500
```

### Univariate Analysis

Let's begin by taking a look at the response variable, `MSZoning`, which has the following classes:

```
A    Agriculture
C    Commercial
FV   Floating Village Residential
I    Industrial
RH   Residential High Density
RL   Residential Low Density
RP   Residential Low Density Park
RM   Residential Medium Density
```

There are 8 zoning groups, but note below that there are only 5 classes recorded in the data set. It will be important to keep in mind that some of the classes have no data.

The class sizes are also very imbalanced, so we will need to be careful when using classification methods such as linear discriminant analysis, quadratic discriminant analysis, and k-nearest neighbors.

It seems that most of the homes in the data set are residential low density, followed by residential medium density, and then floating village residential. A few of them are residential high density and commercial.

```
tally(~MSZoning, data = data)
```

```
## MSZoning
## C (all)      FV      RH      RL      RM
##      10      65      16    1151     218
```

Let's now examine the categorical variables recording overall quality and overall condition of the house as well.

It makes sense that most of the houses have a quality rating between 4 and 8 with a few having a very high and very low quality rating.

```
tally(~OverallQual, data = data)
```

```
## OverallQual
##   1   2   3   4   5   6   7   8   9  10
##   2   3  20 116 397 374 319 168  43  18
```

We see similar patterns for the rating of overall condition of the houses as well, with most homes concentrated between a rating of 5 and 7.

```
tally(~OverallCond, data = data)
```

```
## OverallCond
##   1   2   3   4   5   6   7   8   9
##   1   5  25  57 821 252 205  72  22
```

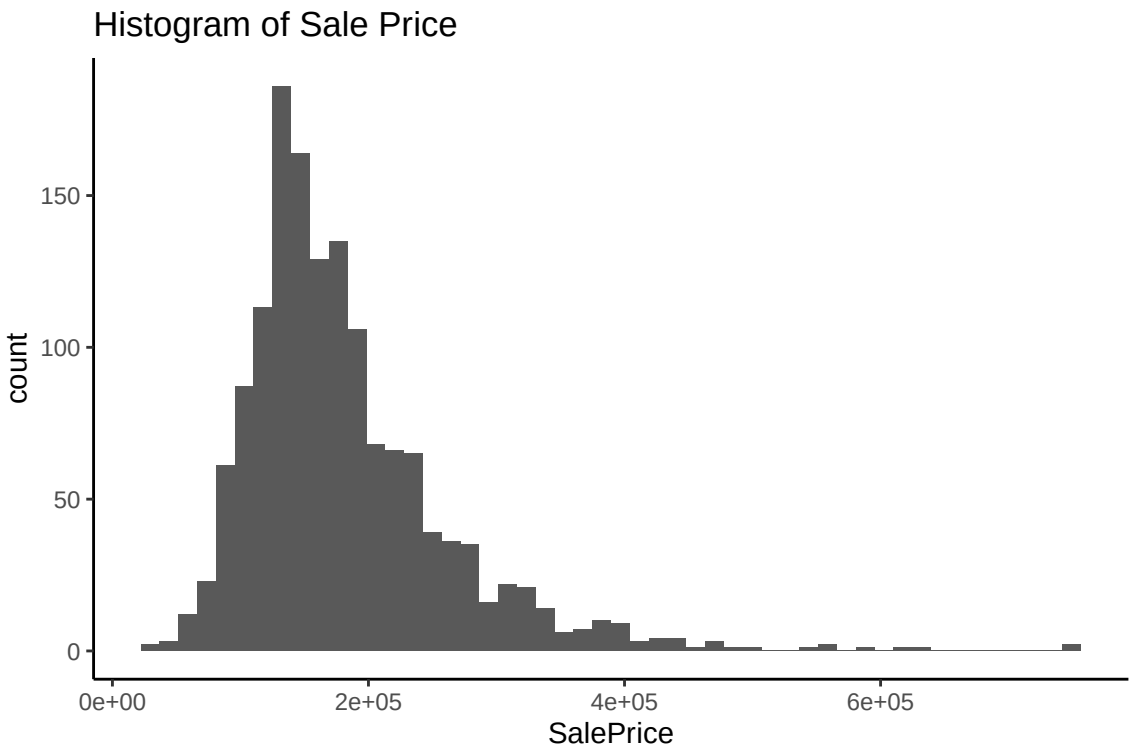
Most of the quantitative variables record information such as number of bedrooms, number of bathrooms, and square footage. To conclude my univariate analysis, I'll examine the distribution of a couple quantitative variables of interest: `SalePrice`, which records the valued price of the home, and `LotArea`, which records total square footage of the home. Before further analysis, it would seem that these 2 variables would be highly descriptive of a home's zoning group.

`SalePrice` looks approximately normal with a slight right skew. There are a few houses that are priced very highly but most seem to have a mode at around 150,000. The lowest priced home is 34,900, while the highest priced home is 755,000. The median is 163,000 and the mean is 180,921.

```
favstats(~SalePrice, data = data)
```

```
##    min     Q1 median     Q3    max   mean      sd    n missing
##  34900 129975 163000 214000 755000 180921 79442.5 1460       0
```

```
ggplot(data, mapping = aes(x = SalePrice)) +
  geom_histogram(bins = 50) +
  labs(title = "Histogram of Sale Price") +
  theme_classic()
```

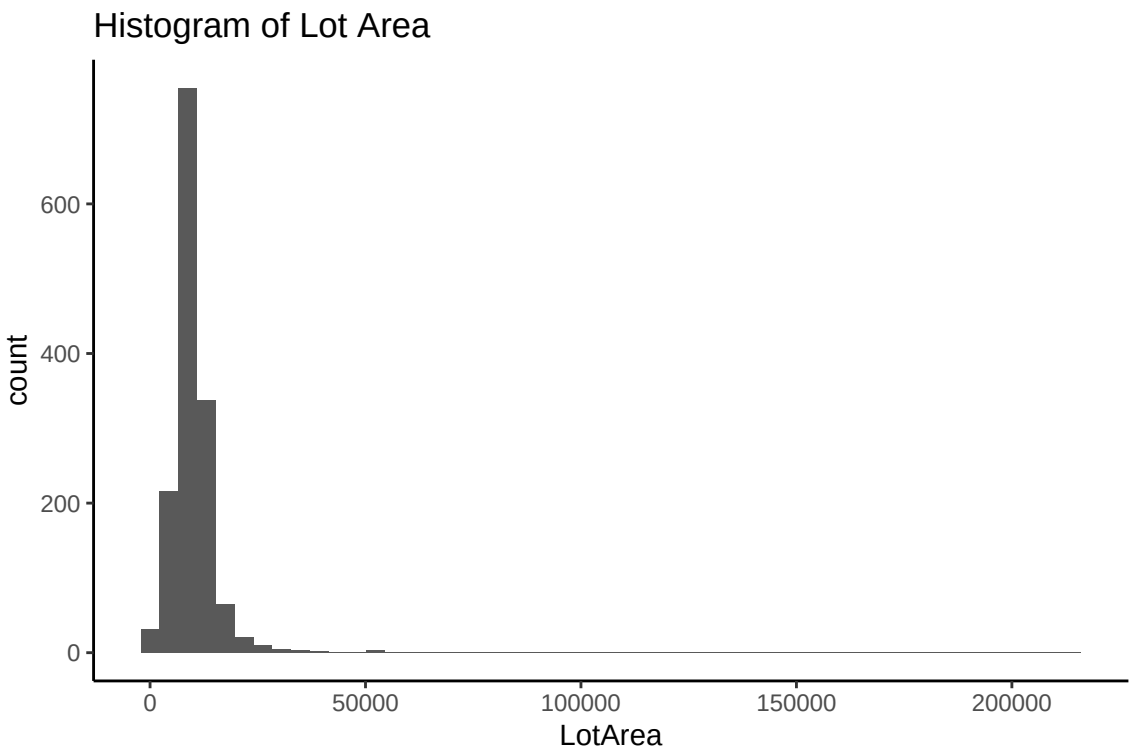



`LotArea` has a similar distribution with a very heavy right tail. The mean is 9,478.5 square feet, but the mean is 10,516.8 square feet. The smallest lot size is 1,300 square feet, while the largest is 215,245 square feet.

```
favstats(~LotArea, data = data)
```

```
##   min     Q1 median      Q3    max    mean      sd    n missing
##  1300 7553.5 9478.5 11601.5 215245 10516.8 9981.26 1460       0
```

```
ggplot(data, mapping = aes(x = LotArea)) +
  geom_histogram(bins = 50) +
  labs(title = "Histogram of Lot Area") +
  theme_classic()
```




It's important to note that there is a huge difference between minimum and maximum sale price and lot area, suggesting an observable difference between homes just by these 2 variables. They may be important in classifying homes into zoning groups.

As seen, with home data, it's very important to be careful about right skew distributions and it's unlikely that most of our variables are normally distributed. This will be important in case any methods require multivariate normality as a condition.

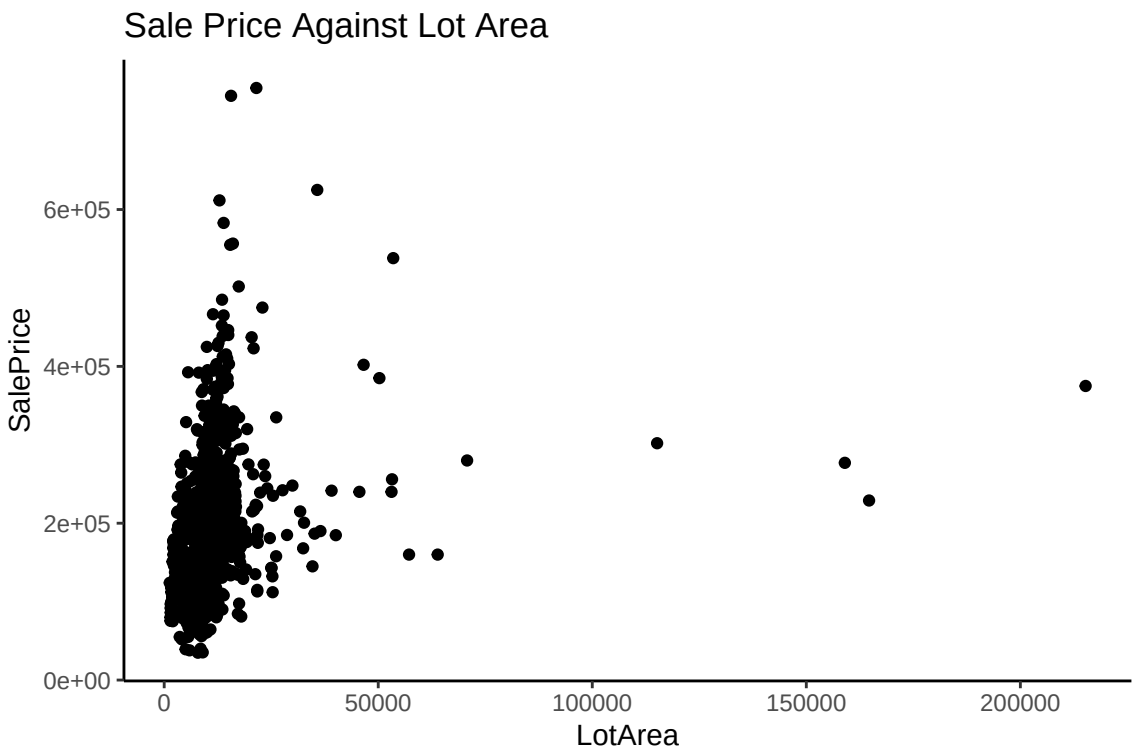
### Bivariate Analysis

Let's begin our bivariate analysis by creating a subset of quantitative variables to use for further analysis.

```
quant_data <- select(data, - c(MSZoning, MSSubClass, OverallQual, OverallCond))
```

In looking at a scatter plot of `LotArea` and `SalePrice`, it looks like there is some positive relationship, but there are undoubtedly a lot of outliers.

```
ggplot(data = data, mapping = aes(x = LotArea, y = SalePrice)) +
  geom_point() +
  labs(title = "Sale Price Against Lot Area") +
  theme_classic()
```




Let's examine if any of our variables are correlated. To do that, let's first subset our data into 3 subsets so that it's easier to analyze.

```
quant_data1 <- quant_data[ ,1:9]
quant_data2 <- quant_data[ ,10:18]
quant_data3 <- quant_data[ ,19:27]
```

```
cor(quant_data1)
```

```
##                 LotArea BsmtFinSF1  BsmtFinSF2   BsmtUnfSF TotalBsmtSF
## LotArea       1.00000000  0.2141031  0.11116975 -0.00261836   0.2608331
## BsmtFinSF1    0.21410313  1.0000000 -0.05011740 -0.49525147   0.5223961
## BsmtFinSF2    0.11116975 -0.0501174  1.00000000 -0.20929449   0.1048095
## BsmtUnfSF    -0.00261836 -0.4952515 -0.20929449  1.00000000   0.4153596
## TotalBsmtSF   0.26083313  0.5223961  0.10480954  0.41535961   1.0000000
## X1stFlrSF     0.29947458  0.4458627  0.09711745  0.31798744   0.8195300
## X2ndFlrSF     0.05098595 -0.1370790 -0.09926032  0.00446909  -0.1745120
## LowQualFinSF  0.00477897 -0.0645026  0.01480700  0.02816669  -0.0332454
## GrLivArea     0.26311617  0.2081711 -0.00963989  0.24025727   0.4548682
##                X1stFlrSF   X2ndFlrSF LowQualFinSF   GrLivArea
## LotArea       0.2994746  0.05098595   0.00477897  0.26311617
## BsmtFinSF1    0.4458627 -0.13707899  -0.06450260  0.20817113
## BsmtFinSF2    0.0971174 -0.09926032   0.01480700 -0.00963989
## BsmtUnfSF     0.3179874  0.00446909   0.02816669  0.24025727
## TotalBsmtSF   0.8195300 -0.17451195  -0.03324539  0.45486820
## X1stFlrSF     1.0000000 -0.20264618  -0.01424067  0.56602397
## X2ndFlrSF    -0.2026462  1.00000000   0.06335295  0.68750106
## LowQualFinSF -0.0142407  0.06335295   1.00000000  0.13468281
```

```
## GrLivArea     0.5660240  0.68750106   0.13468281  1.00000000
```

```
cor(quant_data2)
```

```
##              BsmtFullBath BsmtHalfBath   FullBath   HalfBath BedroomAbvGr
## BsmtFullBath    1.0000000   -0.1478710 -0.0645120 -0.0309050   -0.1506728
## BsmtHalfBath   -0.1478710    1.0000000 -0.0545358 -0.0123399    0.0465188
## FullBath       -0.0645120   -0.0545358  1.0000000  0.1363806    0.3632520
## HalfBath       -0.0309050   -0.0123399  0.1363806  1.0000000    0.2266515
## BedroomAbvGr   -0.1506728    0.0465188  0.3632520  0.2266515    1.0000000
## KitchenAbvGr   -0.0415025   -0.0379444  0.1331152 -0.0682625    0.1985968
## TotRmsAbvGrd   -0.0532752   -0.0238363  0.5547843  0.3434149    0.6766199
## Fireplaces      0.1379277    0.0289756  0.2436705  0.2036485    0.1075697
## GarageCars      0.1318812   -0.0208911  0.4696720  0.2191782    0.0861064
##              KitchenAbvGr TotRmsAbvGrd Fireplaces GarageCars
## BsmtFullBath   -0.0415025   -0.0532752  0.1379277  0.1318812
## BsmtHalfBath   -0.0379444   -0.0238363  0.0289756 -0.0208911
## FullBath        0.1331152    0.5547843  0.2436705  0.4696720
## HalfBath       -0.0682625    0.3434149  0.2036485  0.2191782
## BedroomAbvGr    0.1985968    0.6766199  0.1075697  0.0861064
## KitchenAbvGr    1.0000000    0.2560454 -0.1239362 -0.0506339
## TotRmsAbvGrd    0.2560454    1.0000000  0.3261145  0.3622886
## Fireplaces     -0.1239362    0.3261145  1.0000000  0.3007888
## GarageCars     -0.0506339    0.3622886  0.3007888  1.0000000
```

```
cor(quant_data3)
```

```
##               GarageArea  WoodDeckSF OpenPorchSF EnclosedPorch   X3SsnPorch
## GarageArea     1.0000000  0.22466631   0.2414347    -0.1217767  0.035086700
## WoodDeckSF     0.2246663  1.00000000   0.0586606    -0.1259889 -0.032770634
## OpenPorchSF    0.2414347  0.05866061   1.0000000    -0.0930793 -0.005842499
## EnclosedPorch -0.1217767 -0.12598889  -0.0930793     1.0000000 -0.037305283
## X3SsnPorch     0.0350867 -0.03277063  -0.0058425    -0.0373053  1.000000000
## ScreenPorch    0.0514118 -0.07418135   0.0743039    -0.0828642 -0.031435847
## PoolArea       0.0610473  0.07337821   0.0607621     0.0542026 -0.007991549
## MiscVal       -0.0273999 -0.00955123  -0.0185837     0.0183606  0.000353965
## SalePrice      0.6234314  0.32441344   0.3158562    -0.1285780  0.044583665
##               ScreenPorch    PoolArea      MiscVal  SalePrice
## GarageArea      0.0514118  0.06104727 -0.027399914  0.6234314
## WoodDeckSF     -0.0741814  0.07337821 -0.009551228  0.3244134
## OpenPorchSF     0.0743039  0.06076211 -0.018583739  0.3158562
## EnclosedPorch  -0.0828642  0.05420256  0.018360600 -0.1285780
## X3SsnPorch     -0.0314358 -0.00799155  0.000353965  0.0445837
## ScreenPorch     1.0000000  0.05130739  0.031945761  0.1114466
## PoolArea        0.0513074  1.00000000  0.029668651  0.0924035
## MiscVal         0.0319458  0.02966865  1.000000000 -0.0211896
## SalePrice       0.1114466  0.09240355 -0.021189580  1.0000000
```

Quickly overlooking the correlation matrices above, we notice that there exist some moderate to strong correlations between our variables. However, most of the correlations are weak.

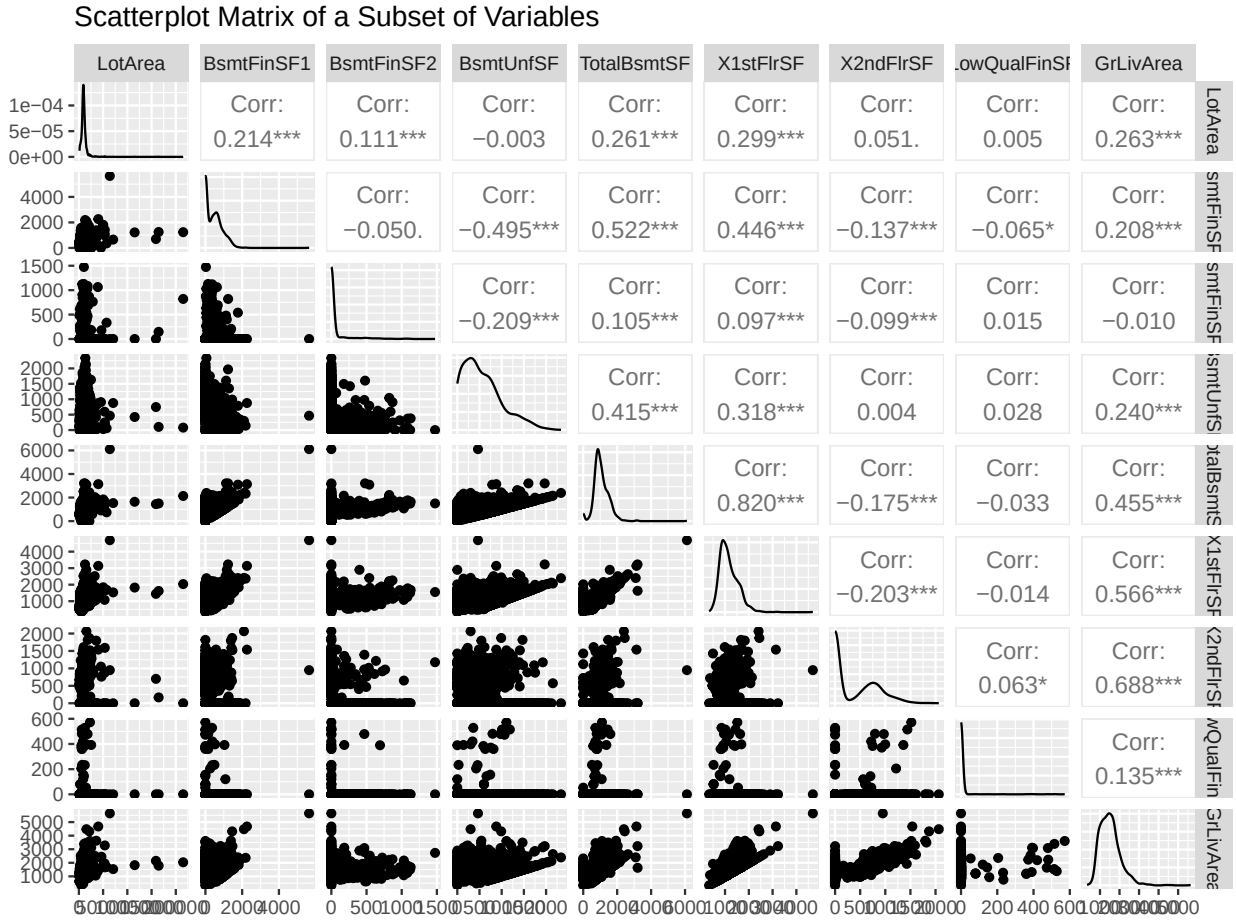
The existence of many weak correlations suggests that PCA may not be an appropriate secondary technique to visualize our final clusters. However, since PCA is not our primary technique, the existence of some

moderate to strong correlation suggests that we can attempt to use it for the purpose of visualizing our clusters.

Let's now run a scatter plot matrix on each of our 3 subsets in order to visualize all the quantitative variables in our data set.

As speculated, a lot of the variables have extreme right skews, so it will be very important to note that the multivariate normal condition is violated. We also confirm some moderate relationships between the variables, which will allow us to visualize our clustering solution by reducing the dimension of our data set. The relationships also suggest that not all variables may be needed in the analysis, especially in cases where we may need to select a few variables to proceed with.

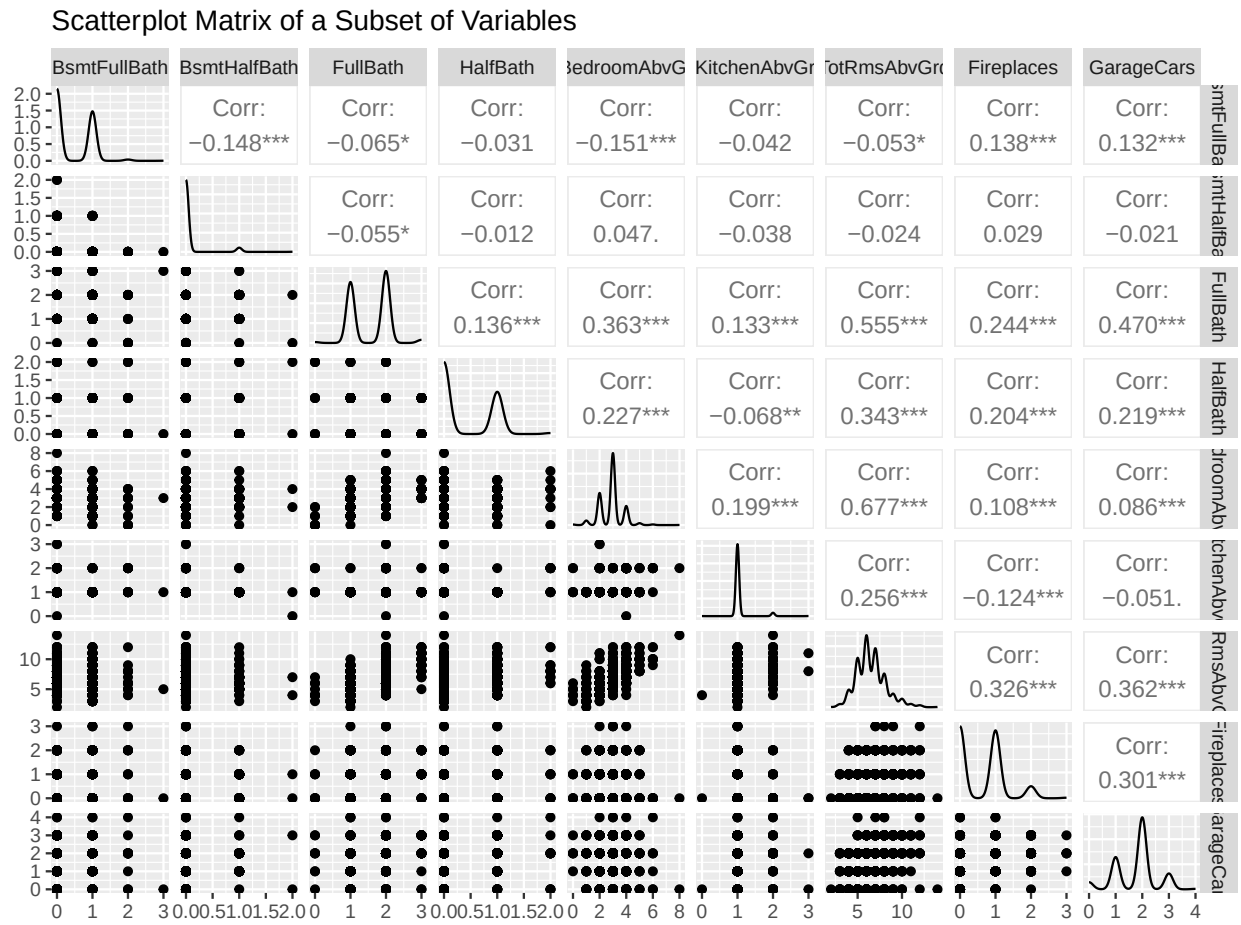
```
ggpairs(quant_data1, title = "Scatterplot Matrix of a Subset of Variables")
```




In the second scatter plot, we see that a lot of the variables describing attributes such as the number of bathrooms, bedrooms, fireplaces, and garage cars take on discrete values. As much as these are quantitative variables, they cannot take on negative or decimal values, and realistically, most homes fall within a small range. This makes it seem as though these are categorical variables, and may affect the way in which the clustering algorithm treats these variables. For example, most homes have 0, 1, 2, or 3 fireplaces. These variables, thus, have multimodal distributions.

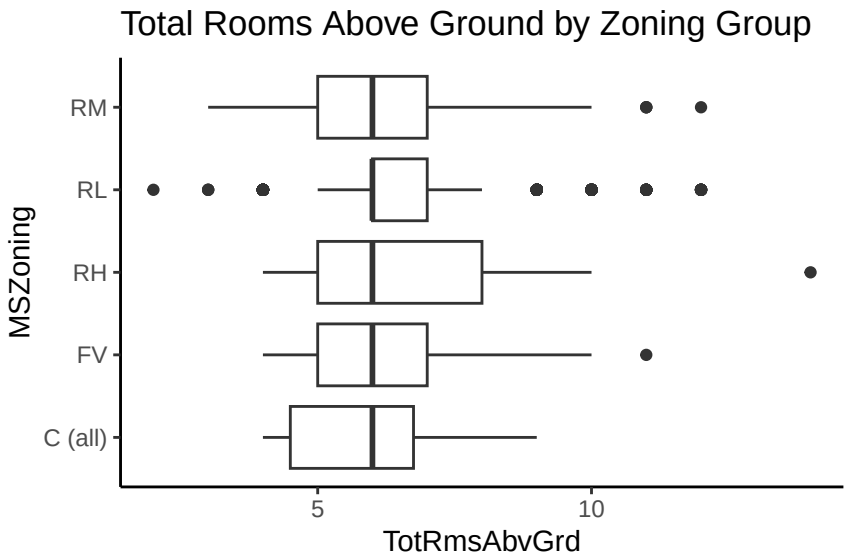
This is not a concern for classification, but may be a concern for clustering, particularly in computing the distance matrix.

```
ggpairs(quant_data2, title = "Scatterplot Matrix of a Subset of Variables")
```



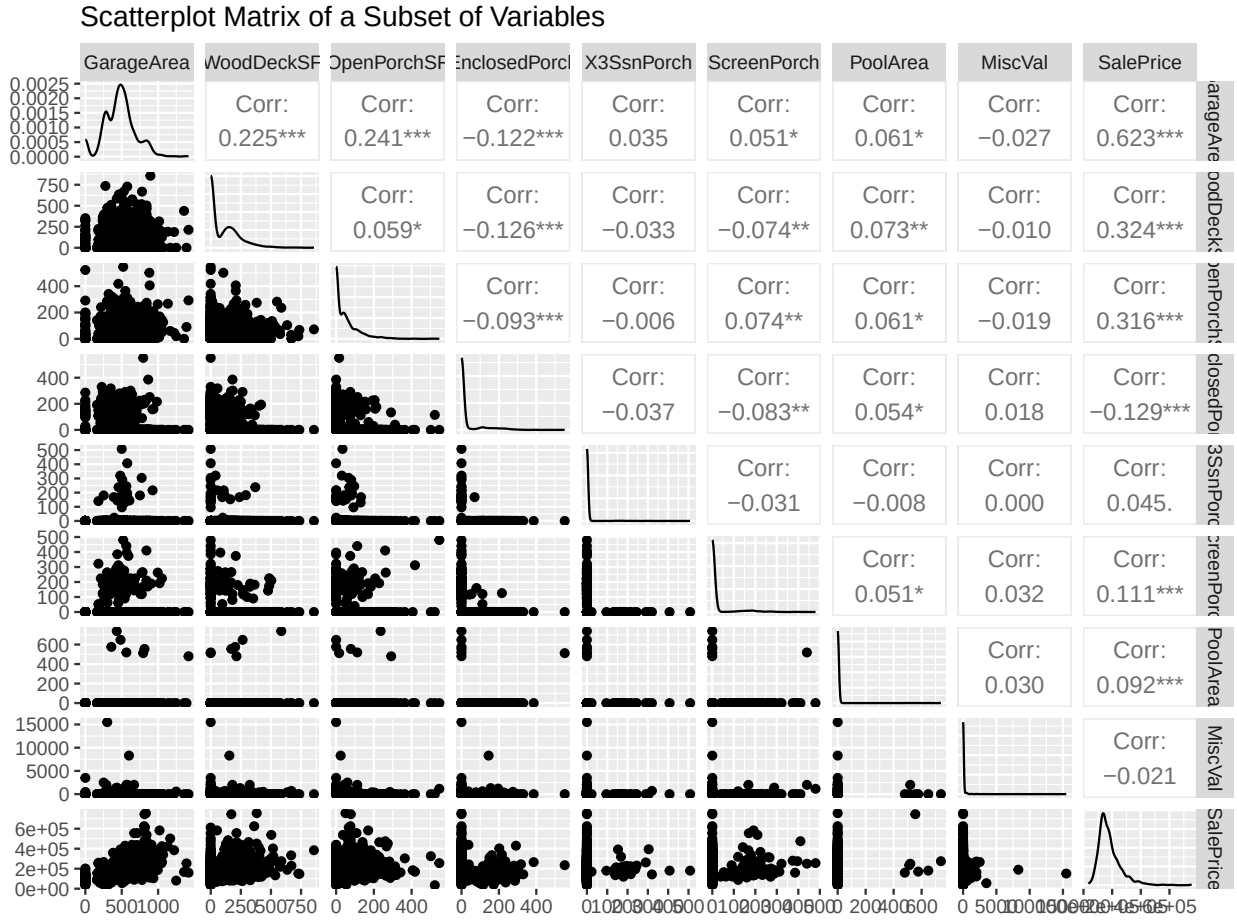

We can attempt to see if there is a good amount of separation in zoning characteristics based on some of these variables. For example, in looking at the number of rooms above ground across the 5 zoning groups, as much as there are some distributional differences, the median is the same across all zones. This indicates that there may not be much separation between the different zoning groups based on these variables.

```
gf_boxplot(MSZoning ~ TotRmsAbvGrd, data = data) +
  labs(title = "Total Rooms Above Ground by Zoning Group") +
  theme_classic()
```




Finally, the last subset reveals similar information to the first 2 scatter plot matrices above.

```
ggpairs(quant_data3, title = "Scatterplot Matrix of a Subset of Variables")
```




Overall, the key takeaways from my univariate and bivariate analysis are that:

i) many of the variables have extreme right skews,

ii) some variables only take very specific discrete values despite being quantitative,

iii) there exist some moderate relationships between the variables.

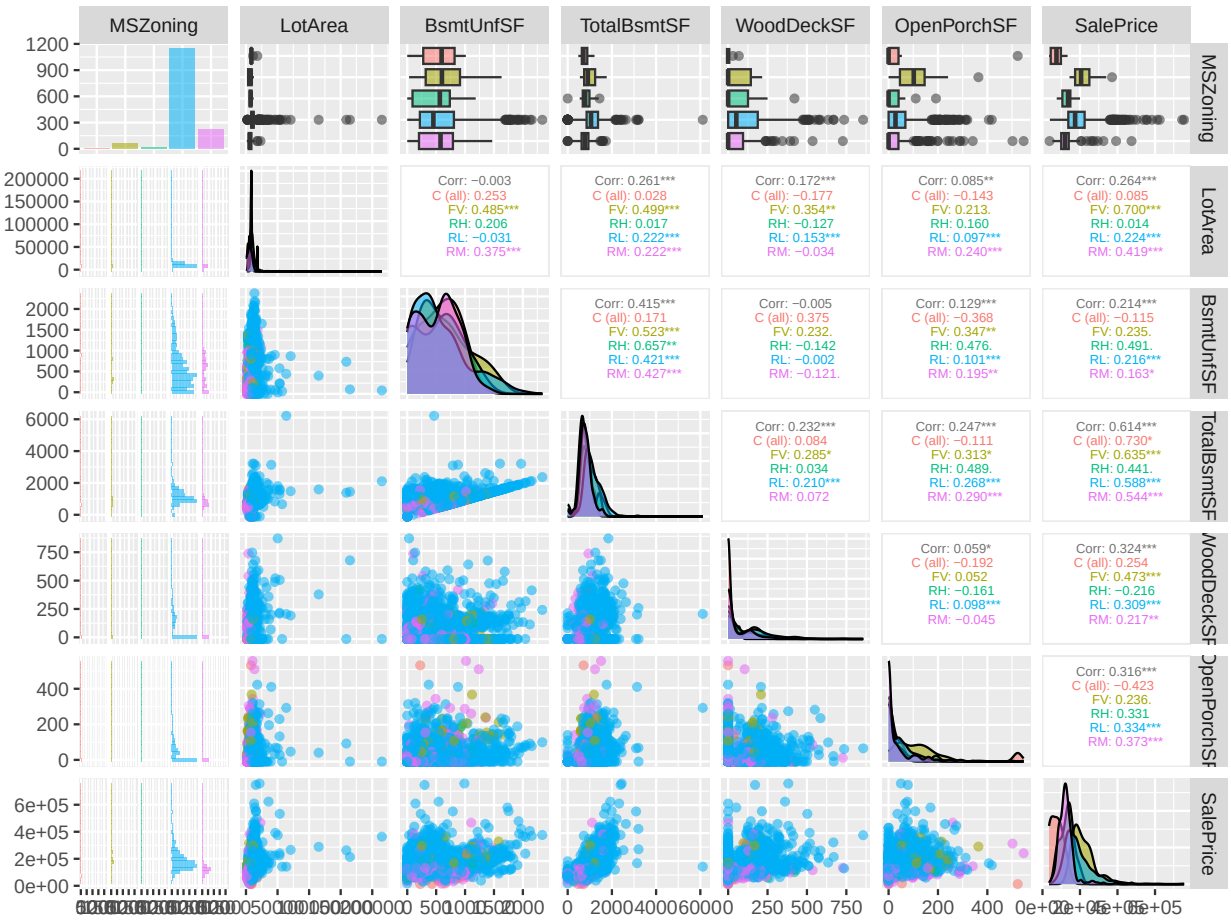
These will be important to consider as we begin the data analysis.

### Multivariate Analysis

In looking just at a subset of the data, we see significant overlap, but nevertheless, some differences between zoning groups across the different variables, suggesting that some of the variables may be useful in classifying or differentiating between different home zoning groups. Those with significant overlap may not appear to be useful at first glance, but perhaps in the presence of other variables, they may allow us to cluster and classify observations into different zoning groups more accurately. This may also allow us to see which zoning groups are most different from others and which are most similar to each other, and perhaps investigate that difference further.

This provides motivation for a clustering and classification analysis and perhaps we will be able to recover the zoning groups using these variables.

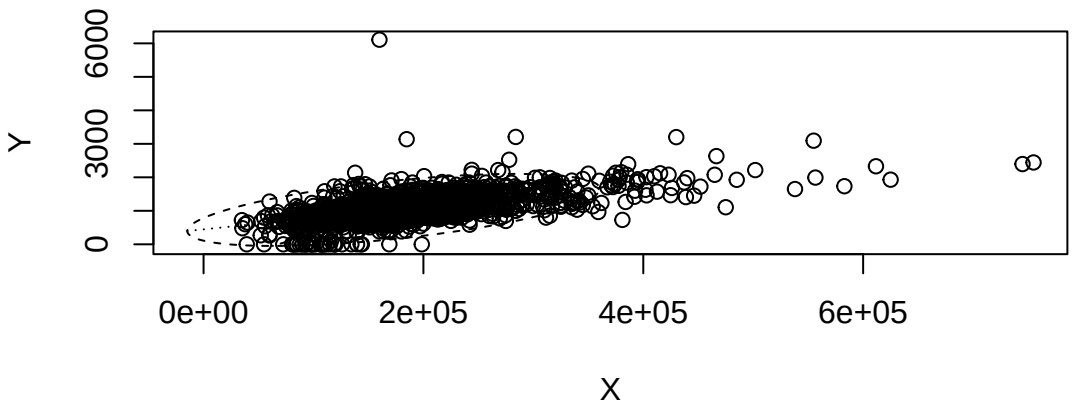
```
ggpairs(data, columns = c("MSZoning", "LotArea", "BsmtUnfSF", "TotalBsmtSF", "WoodDeckSF",
                          "OpenPorchSF", "SalePrice"),
        upper = list(continuous = wrap("cor", size = 2)),
        mapping = aes(color = MSZoning, alpha = 0.7,
                      title = "Scatterplot Matrix of Some Variables by Zoning Group"))
```




Finally, let's check for any multivariate outliers that may be present. Let's pick 2 variables: `SalePrice` and `TotalBsmtSF`, which describes the total basement square footage. In general, there are quite a few multivariate outliers, that is, a substantial number of observations lie outside the fence. This will be important to note, especially during clustering, as some clustering algorithms are easily influenced by outlying observations.

```
data_bvbox <- data %>%
  filter(!is.na(SalePrice),
         !is.na(TotalBsmtSF))

twoVars <- data_bvbox[, c("SalePrice", "TotalBsmtSF")]
bvbox(twoVars, mtitle = "Bivariate Boxplot")
```




**Preparation for Analysis**

For reference, the variable names in our final data set are included below.

```
names(data)
```

```
##  [1] "MSSubClass"    "MSZoning"      "LotArea"       "OverallQual"
##  [5] "OverallCond"   "BsmtFinSF1"    "BsmtFinSF2"    "BsmtUnfSF"
##  [9] "TotalBsmtSF"   "X1stFlrSF"     "X2ndFlrSF"     "LowQualFinSF"
## [13] "GrLivArea"     "BsmtFullBath"  "BsmtHalfBath"  "FullBath"
## [17] "HalfBath"      "BedroomAbvGr"  "KitchenAbvGr"  "TotRmsAbvGrd"
## [21] "Fireplaces"    "GarageCars"    "GarageArea"    "WoodDeckSF"
## [25] "OpenPorchSF"   "EnclosedPorch" "X3SsnPorch"    "ScreenPorch"
## [29] "PoolArea"      "MiscVal"       "SalePrice"
```

Here is a description of the final variables that may prove useful as we begin the data analysis.

```
  MSSubClass: Identifies the type of dwelling involved in the sale.
  MSZoning: Identifies the general zoning classification of the sale.
  LotArea: Lot size in square feet.
  OverallQual: Rates the overall material and finish of the house.
  OverallCond: Rates the overall condition of the house.
  BsmtFinSF1: Type 1 finished square feet.
  BsmtFinSF2: Type 2 finished square feet.
  BsmtUnfSF: Unfinished square feet of basement area.
  TotalBsmtSF: Total square feet of basement area.
  X1stFlrSF: First Floor square feet.
  X2ndFlrSF: Second floor square feet.
  LowQualFinSF: Low quality finished square feet (all floors).
  GrLivArea: Above grade (ground) living area square feet.
  BsmtFullBath: Basement full bathrooms.
  BsmtHalfBath: Basement half bathrooms.
  FullBath: Full bathrooms above grade.
```

```
HalfBath: Half baths above grade.
BedroomAbvGr: Bedrooms above grade (does NOT include basement bedrooms).
KitchenAbvGr: Kitchens above grade.
TotRmsAbvGrd: Total rooms above grade (does not include bathrooms).
Fireplaces: Number of fireplaces.
GarageCars: Size of garage in car capacity.
GarageArea: Size of garage in square feet.
WoodDeckSF: Wood deck area in square feet.
OpenPorchSF: Open porch area in square feet.
EnclosedPorch: Enclosed porch area in square feet.
X3SsnPorch: Three season porch area in square feet.
ScreenPorch: Screen porch area in square feet.
PoolArea: Pool area in square feet.
MiscVal: Dollar value of miscellaneous feature.
SalePrice: The property's sale price in dollars.
```

Note that "above grade" here means above the ground.

Let's now proceed onto the data analysis.

## Methods

### The Analysis

As described in the introduction, the purpose of this analysis is to use both unsupervised learning and supervised learning on a data set containing a sample of Boston homes to group homes into zoning groups based on other attributes of the home. For this analysis, we will be using clustering and classification as our main statistical techniques.

There are 31 variables in our final data set, with 27 being quantitative and 4 being categorical. Note that we will not include the 4 categorical variables when running our clustering algorithm. We will only use the 27 quantitative variables, and later assess whether our clustering solutions were able to recover the zoning groups, our primary focus. We will use all 31 variables to build a classification model predicting zoning group.

Because our variables have different scales, as observed in the preliminary analysis, we will scale/ standardize them in the clustering analysis.

### Agglomerative Hierarchical Clustering

In agglomerative hierarchical clustering, all observations start off in a cluster alone. The pair of clusters closest to each other are then merged. The distances between each of the current clusters is updated through linkage. This process is repeated until all observations are in one cluster. The ideal number of clusters is then chosen.

**Distance measure:** We will use Euclidean distance as the default when running this algorithm.

**Linkage:** Linkage is the way in which we update our distance matrix in hierarchical clustering. Chaining can be a problem for single linkage, which finds the smallest distance between clusters and is sensitive to outliers. We will use single linkage to check for outliers in our data set.

For this analysis, we will also use Ward's method, which is very common. This method merges clusters that result in the smallest Within Group Sum of Squares (WGSS), a metric used to determine what to merge.

**Choosing the Number of Clusters:** We will observe a dendogram of the clusters and choose a cut-off based on where the joins are large.

### K-Means Clustering

K-Means clustering, on the other hand, is a partitioning algorithm that uses an iterative process. We need to determine the number of clusters, $k$, that we wish to have. K-Means clustering then finds a set of $k$ clusters that minimize a criterion called the WGSS (Within Group Sum of Squares). The algorithm picks $k$ data points as starting cluster centers randomly and adjusts cluster centers and moves points between clusters until WGSS is minimized. Because it can get stuck with poor starting points, we will specify that the algorithm should try 10 different random starts.

**Distance measure:** We will use Euclidean distance as the default when running this algorithm.

**Choosing the Number of Clusters:** We will generate a WGSS elbow plot for K-Means. By looking at the elbow – that is, where the slope stops changing significantly –, we will be able to determine the number of clusters to set for our K-Means solution.

### Assessing the Strength and Validity of the Clustering Solution

For both algorithms, will use silhouette coefficients to assess the strength and validity of the clustering solution. In particular, we will look at the silhouette value per observation in each cluster, cluster average silhouette values, as well as the silhouette coefficient for the entire solution, which averages over all clusters. This will allow us to assess the structure of our solution. Values above 0.7 will indicate a strong structure, between 0.51 and 0.7 will indicate a reasonable structure, between 0.26 and 0.5 will indicate a weak, artificial structure, and below 0.25 will indicate no real structure.

We will also assess whether our solutions recovered the zoning groups in our response variable `MSZoning`.

### Principal Components Analysis

We will also run PCA on our data set to visualize our preferred clustering solution in the PC space using the first 2 principal components, which will hopefully explain most of the variability in the data set. In the preliminary analysis, we established that there are moderate to strong relationships between our variables, so PCA is appropriate. We will use the correlation matrix to run the PCA.

### Classification

After performing unsupervised learning on the data set, we will then move on to supervised learning methods, in particular, classification.

Of note, clustering uses unsupervised learning to find groups when we don't know what the groups are. In many ways, classification can validate a clustering solution. This is because classification uses supervised learning to find rules to assign observations to different groups, that is, the different groups are known and pre-specified.

For our purposes, the 30 predictor variables in our data set will be used to classify `MSZoning`, our response variable encoding the different zoning groups.

Different classification techniques make different decision rules to accomplish classification. In this analysis, we will use 4 different techniques: decision tree, random forest, bagging, and boosting.

### Tree Model

Classification trees are also known as decision trees, which create binary splits on predictor variables to cut the predictor space into hyper-cubes. Ideally, the end cubes only contain 1 class. Different hyper-cubes per class is allowed.

One can change the `CP` value if they want to grow a larger tree or prune a tree (make it smaller). It is initially set at 0.01. A user can also set `minsplit`, which determines the minimum number of observations that can be in a node in order to make a split, and `minbucket`, which determines the minimum number of elements that must be in a node after a split. Additionally, the user can determine whether to turn cross-validation on, as well as how many folds are desired, or whether to turn it off and instead use a holdout sample.

### Random Forest

Classification trees have 2 major problems: they are greedy (they don't think one step ahead) and unstable (they are extremely sensitive to changes in the data set). Random forests were designed to solve these problems by growing many trees, each built on a bootstrapped data set.

There are two main choices to make: `ntree`, which sets the number of trees to grow, and `mtry`, which sets the number of variables to be used at each split. The variables are randomly chosen at each split. Because of

variable selection, this method also allows us to assess which variables are most important in the classification process.

Because this process is run on bootstrapped data sets, the out-of-bootstrap (OOB) observations from each run form a natural test set.

### Bagging

Bagging is a special case of random forest that sets `mtry=p`, that is, the number of predictors in the data set. As such, it only combats the instability, but not the greediness, of classification trees. Because this a special case of random forests, then a user also needs to decide `ntree`, which sets the number of trees to grow.

Because this process is run on bootstrapped data sets, the out-of-bootstrap (OOB) observations from each run form a natural test set.

### Boosting

Boosting is a tree-based technique that builds a large number of small trees in sequence, and each tree depends on the one before. Boosting doesn't use the bootstrap, so there are no OOB observations. Instead, we will use the holdout sample approach to test our model. We will need to set the distribution to multinomial because our response is not binary coded – that is, there are 5 different classes in our response variable.

### Assessing the Classification Models

We will use the apparent error rate (AER) and the estimated true error rate (TER) as our main criteria for assessing the models. The AER tends to be over-optimistic because it is based on the training set, so the estimated TER will be our key statistic as it is based on the testing set.

## Results

### Agglomerative Hierarchical Clustering

First, let's create the distance matrix that will be used by the clustering algorithm. By default, this uses Euclidean distance. Note that we will only use the quantitative variables in the clustering analysis.

```
homes.dist <- dist(scale(quant_data))
```

Next, let's use single linkage to check for outliers in our solution. We immediately notice that there is a chaining problem. While the dendogram is hard to visualize, we see that almost all observations are clustered into 1 cluster, and there are a lot of clusters with just 1 observation. This is consistent even if we increase or decrease the tree height. Single linkage will not work for this data set.

```
hcsingle <- hclust(homes.dist, method = "single")
list(hcsingle)
```

```
## [[1]]
##
## Call:
## hclust(d = homes.dist, method = "single")
##
## Cluster method   : single
## Distance         : euclidean
## Number of objects: 1460
```

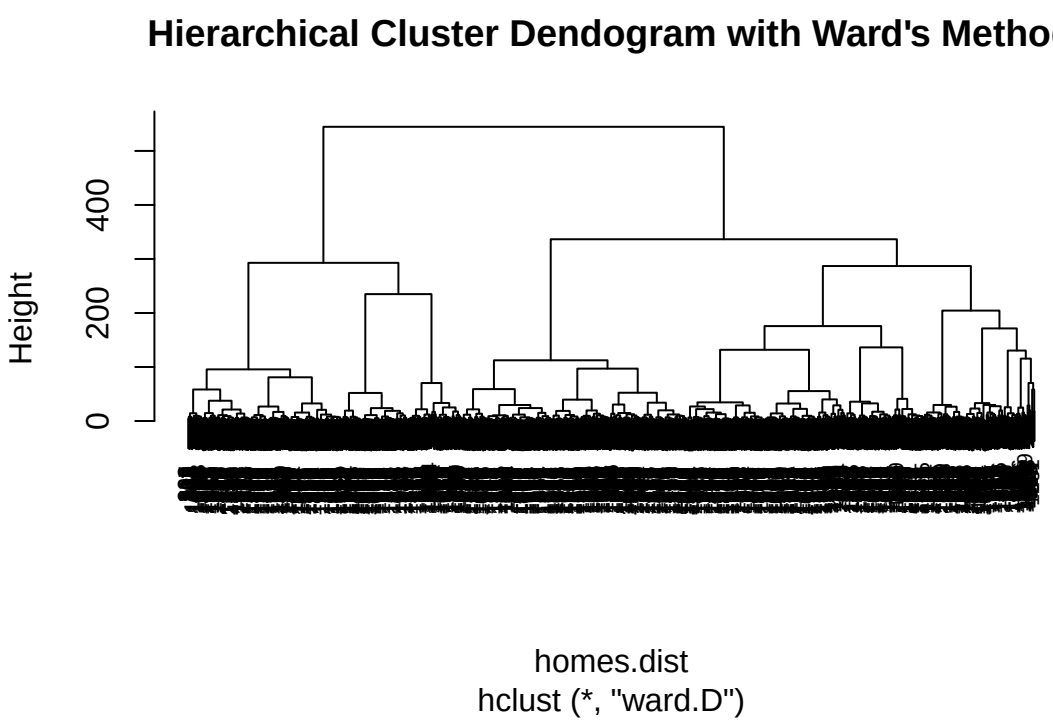
```
singleSol <- (cutree(hcsingle, k = 25))
summary(as.factor(singleSol))
```

```
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
## 1435    1    1    1    1    1    1    1    1    1    1    1    1    1    2    1
##   17   18   19   20   21   22   23   24   25
##    1    1    1    1    1    1    1    1    1
```

Let's attempt to use Ward's method instead, which may offer a better solution.

This solution is much better. We can visualize the clusters better and see where the joins are large. We see that there are 2 main clusters of homes at the top of the dendogram, which are then split into 3 major sub-classes each. Based on where the joins are large, I would choose $k = 2$ or $k = 6$ as an appropriate number of home classes. We will assess the fit for both of these $k$ to pick the best agglomerative solution.

```
hcward <- hclust(homes.dist, method = "ward.D")
plot(hcward, cex = 0.7, main = "Hierarchical Cluster Dendogram with Ward's Method")
```




Having chosen $k = 2$ or $k = 6$, let's look at the summary solution.

Using $k = 2$, observe that one cluster has 472 homes and another cluster has 988 homes. Using $k = 6$, we notice that the clusters are all of vastly different sizes. The largest cluster has 409 observations with another cluster of similar size with 389 observations. The smallest cluster has only 94 homes.

```
wardSol <- (cutree(hcward, k = 2))
summary(as.factor(wardSol))
```

```
##   1   2
## 472 988
```

```
wardSol2 <- (cutree(hcward, k = 6))
summary(as.factor(wardSol2))
```

```
##   1   2   3   4   5   6
## 267 190  94 409 111 389
```

Let's assess the strength and validity of a solution with $k = 2$ and $k = 6$. A 2-cluster solution has a Silhouette coefficient of 0.09 while a 6-cluster solution has a Silhouette coefficient of 0.07. Both solutions have no real structure.

Visualizing the 2-cluster solution, we observe that cluster 1 has a stronger silhouette values than cluster 2. However, both clusters have no real structure, suggesting that the homes do not fit well into their clusters.

```
wardSil <- silhouette(wardSol, homes.dist)
wardSil2 <- silhouette(wardSol2, homes.dist)
```

```
wardSil <- as.data.frame(wardSil)
mean(wardSil$sil_width)
```

```
## [1] 0.0880959
```

```
mean(wardSil$sil_width[wardSil$cluster == 1])
```

```
## [1] 0.211404
```

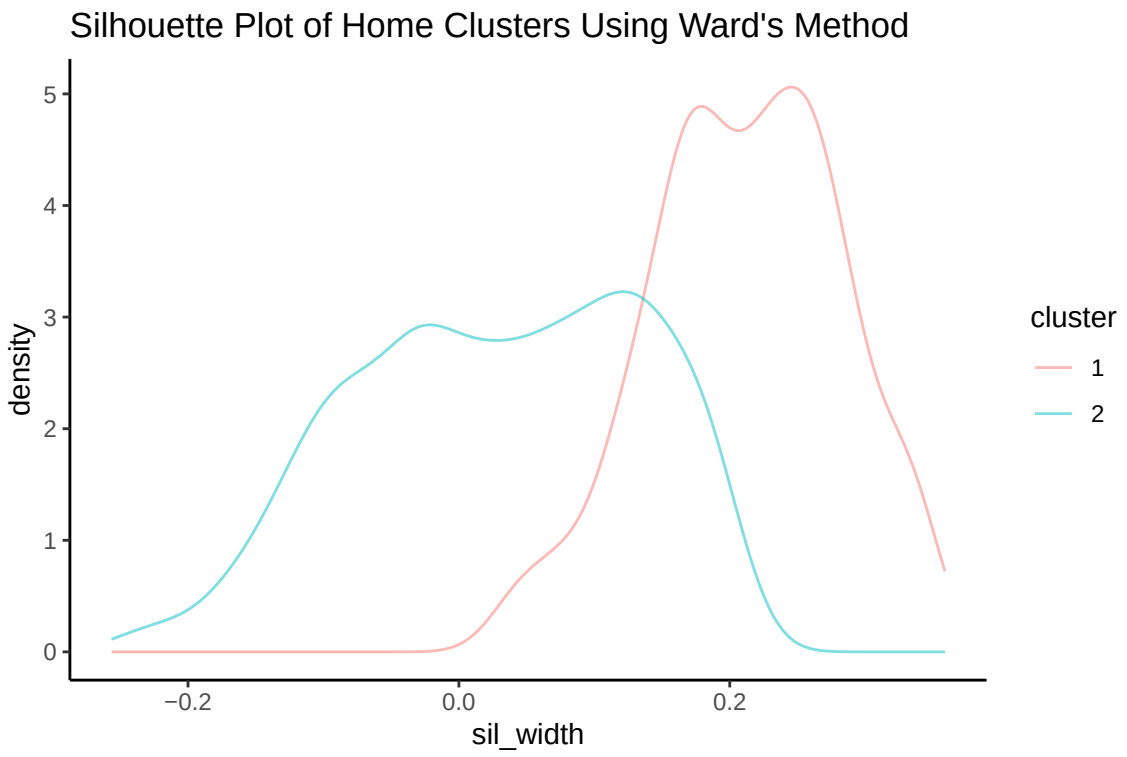
```
mean(wardSil$sil_width[wardSil$cluster == 2])
```

```
## [1] 0.0291876
```

```
wardSil2 <- as.data.frame(wardSil2)
mean(wardSil2$sil_width)
```

```
## [1] 0.0702793
```

```
wardSil <- mutate(wardSil, cluster = factor(cluster))
gf_dens(~ sil_width, color = ~ cluster, data = wardSil,
        title = "Silhouette Plot of Home Clusters Using Ward's Method")  +
  theme_classic()
```




Overall, the agglomerative clustering solutions have no real structure. Let's try running K-Means clustering next.

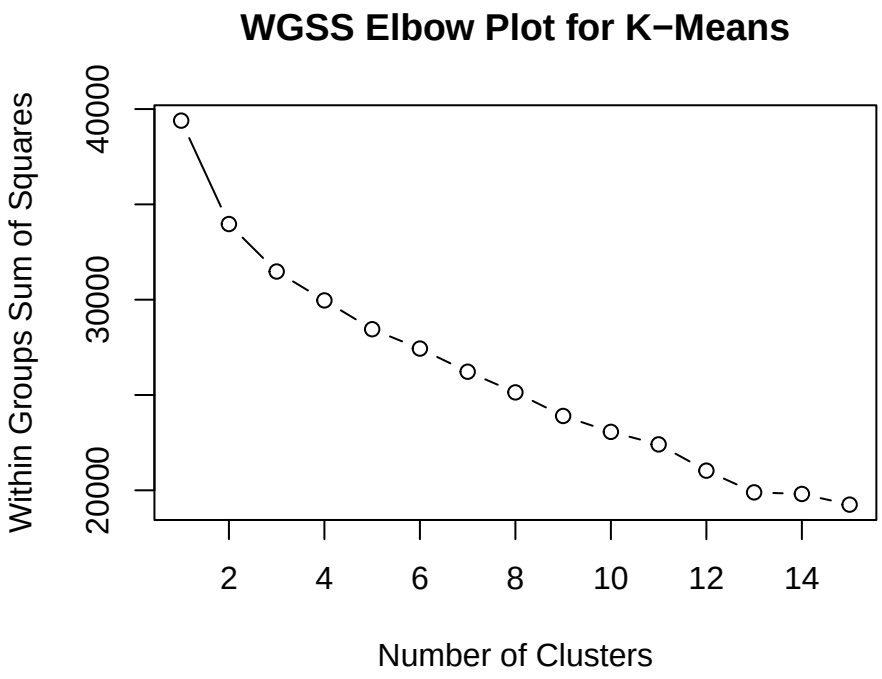
**K-Means Clustering**

First, let's begin by determining the number of clusters, $k$, that we will use for the K-Means clustering algorithm by generating the WGSS elbow plot. It's difficult to see where the slope changes, but it seems the sharpest change occurs at $k = 2$.

```
set.seed(240)
nclustmax <- 15

wss <- rep(0, nclustmax)
for(i in 1:nclustmax){
  wss[i] <- sum(kmeans(scale(quant_data),
                       centers = i, nstart = 10)$withinss)
}

plot(1:nclustmax, wss, type = "b",
     xlab = "Number of Clusters",
     ylab = "Within Groups Sum of Squares",
     main = "WGSS Elbow Plot for K-Means")
```




After trying a few other possible options for $k$, we decided to proceed with $k = 2$. This is consistent with our choice for $k$ in the agglomerative clustering solution.

Now, let's run K-Means clustering with $k = 2$. We will use 10 random starts to ensure that our algorithm doesn't get stuck.

Observe that the clusters are not too different in size. Cluster 1 has 628 homes while cluster 2 has 832 homes.

```
set.seed(240)

Ksol1 <- kmeans(scale(quant_data),
                centers = 2, nstart = 10)
list(Ksol1$size)
```

```
## [[1]]
## [1] 628 832
```

Finally, let's assess the strength and validity of this solution. In looking at the summary, cluster 1 has a silhouette coefficient of 0.05 and cluster 2 has a silhouette coefficient of 0.24. Cluster 1 has no real structure. Cluster 2 has a borderline weak structure, which suggests that it may be artificial. Some homes even have negative silhouette values, so they don't fit in their clusters at all.

The overall silhouette coefficient is 0.16, suggesting that the observations fit in these clusters better than they fit into the agglomerative clusters, but nevertheless, there is still no real structure.

```
kmeansSil <- silhouette(Ksol1$cluster,
                        dist(scale(quant_data)))
```

```
kmeansSil <- as.data.frame(kmeansSil)
mean(kmeansSil$sil_width)
```

```
## [1] 0.157964
```

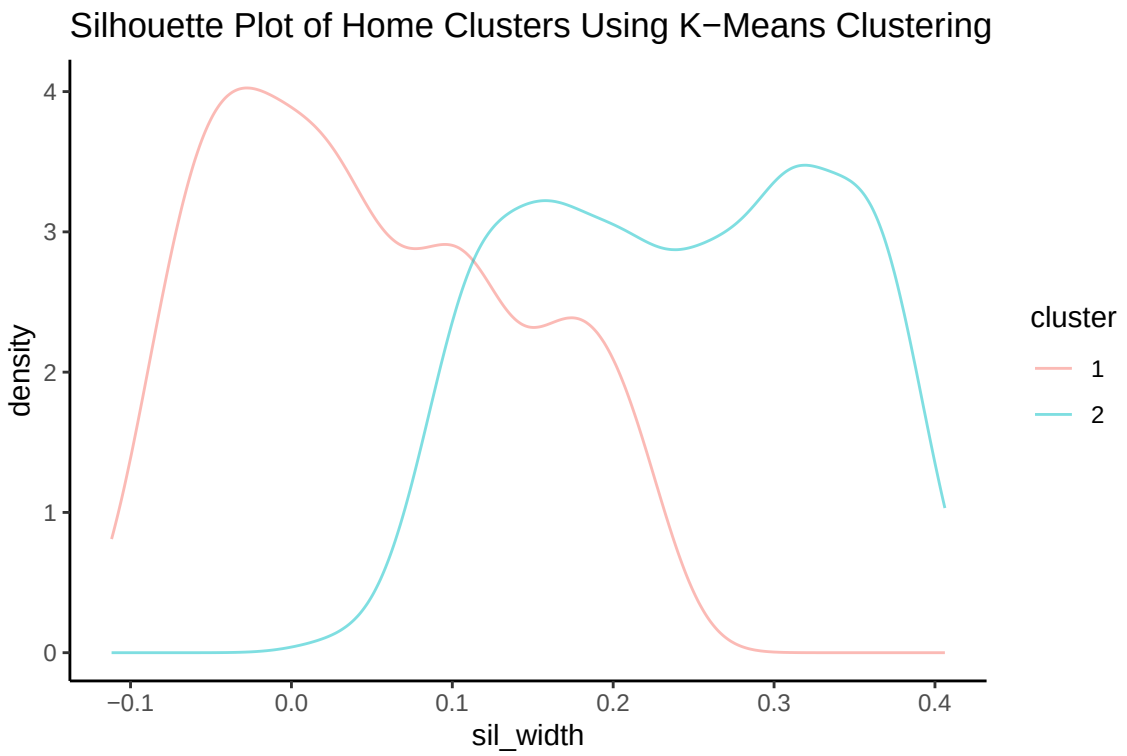
```
mean(kmeansSil$sil_width[kmeansSil$cluster == 1])
```

```
## [1] 0.0494791
```

```
mean(kmeansSil$sil_width[kmeansSil$cluster == 2])
```

```
## [1] 0.239849
```

```
kmeansSil <- mutate(kmeansSil, cluster = factor(cluster))
gf_dens(~ sil_width, color = ~ cluster, data = kmeansSil,
        title = "Silhouette Plot of Home Clusters Using K-Means Clustering") +
  theme_classic()
```




**Preferred Clustering Solution**

We also attempted to run model-based clustering using *mclust*. This method uses Gaussian finite mixture modeling – that is, the algorithm basically assumes it is looking for the clusters to be multivariate normal distributions. The best solution maximizes BIC – the Bayesian Information Criterion. Unfortunately, running model-based clustering on this data set put all the observations into one cluster, so we were not able to perform any further cluster validation analysis.

As such, although all solutions have no real structure, the K-Means solution is our preferred model because it has the highest silhouette value of 0.16.

At the moment, our clustering output is on standardized variables. Therefore, we need to save the clusters to the original data set to perform any further analysis. Let's first create a data set that encodes our original variables and the clusters obtained from the k-means clustering solution.

```
homes_sub <- mutate(data, cluster = factor(Ksol1$cluster))
```

Now, let's examine if any zoning groups were recovered by comparing them against our clusters.

```
tally(MSZoning ~ cluster, data = homes_sub)
```

```
##          cluster
## MSZoning   1   2
##   C (all)  1   9
##   FV      35  30
##   RH       4  12
##   RL     556 595
##   RM      32 186
```

It seems that most of the zoning groups were split across the 2 clusters with no real pattern. The clustering solution did not recover any zoning groups.

### Principal Components Analysis

Finally, let's visualize our clustering solution. Because we have 27 quantitative variables in our data set, we will turn to PCA to reduce the dimensionality of our data set and visualize our clusters in the principal components space.

We will use the correlation matrix because our variables have varying scales. Similar to the clustering solution, the PCA solution is very weak. The first 2 PC's explain only 33.4% of the variation in the data set. I'm skeptical to use these principal components to visualize the clustering solution.

```
homePCAs <- princomp(quant_data, cor = TRUE)
summary(homePCAs)
```

```
## Importance of components:
##                           Comp.1   Comp.2    Comp.3    Comp.4    Comp.5
## Standard deviation     2.453345 1.731288 1.3798466 1.2401455 1.1262891
## Proportion of Variance 0.222922 0.111013 0.0705177 0.0569615 0.0469825
## Cumulative Proportion  0.222922 0.333936 0.4044533 0.4614148 0.5083973
##                            Comp.6    Comp.7   Comp.8    Comp.9   Comp.10
## Standard deviation     1.0532871 1.0357555 1.020498 1.0035114 0.9834769
## Proportion of Variance 0.0410894 0.0397329 0.038571 0.0372976 0.0358232
## Cumulative Proportion  0.5494867 0.5892197 0.627791 0.6650882 0.7009115
##                           Comp.11   Comp.12   Comp.13   Comp.14   Comp.15
## Standard deviation     0.9774124 0.9588082 0.9439544 0.9260633 0.8781875
## Proportion of Variance 0.0353828 0.0340486 0.0330019 0.0317627 0.0285635
## Cumulative Proportion  0.7362942 0.7703429 0.8033447 0.8351074 0.8636709
##                           Comp.16   Comp.17   Comp.18   Comp.19   Comp.20
## Standard deviation     0.8085386 0.7975094 0.7843128 0.6991988 0.5886472
## Proportion of Variance 0.0242124 0.0235563 0.0227832 0.0181066 0.0128335
## Cumulative Proportion  0.8878833 0.9114396 0.9342228 0.9523295 0.9651630
##                           Comp.21    Comp.22    Comp.23    Comp.24    Comp.25
## Standard deviation     0.5438026 0.46184473 0.43664254 0.37437977 0.31742724
## Proportion of Variance 0.0109526 0.00790002 0.00706136 0.00519112 0.00373185
## Cumulative Proportion  0.9761156 0.98401567 0.99107703 0.99626815 1.00000000
##                            Comp.26     Comp.27
## Standard deviation     8.09644e-08 7.08429e-08
## Proportion of Variance 2.42786e-16 1.85879e-16
## Cumulative Proportion  1.00000e+00 1.00000e+00
```

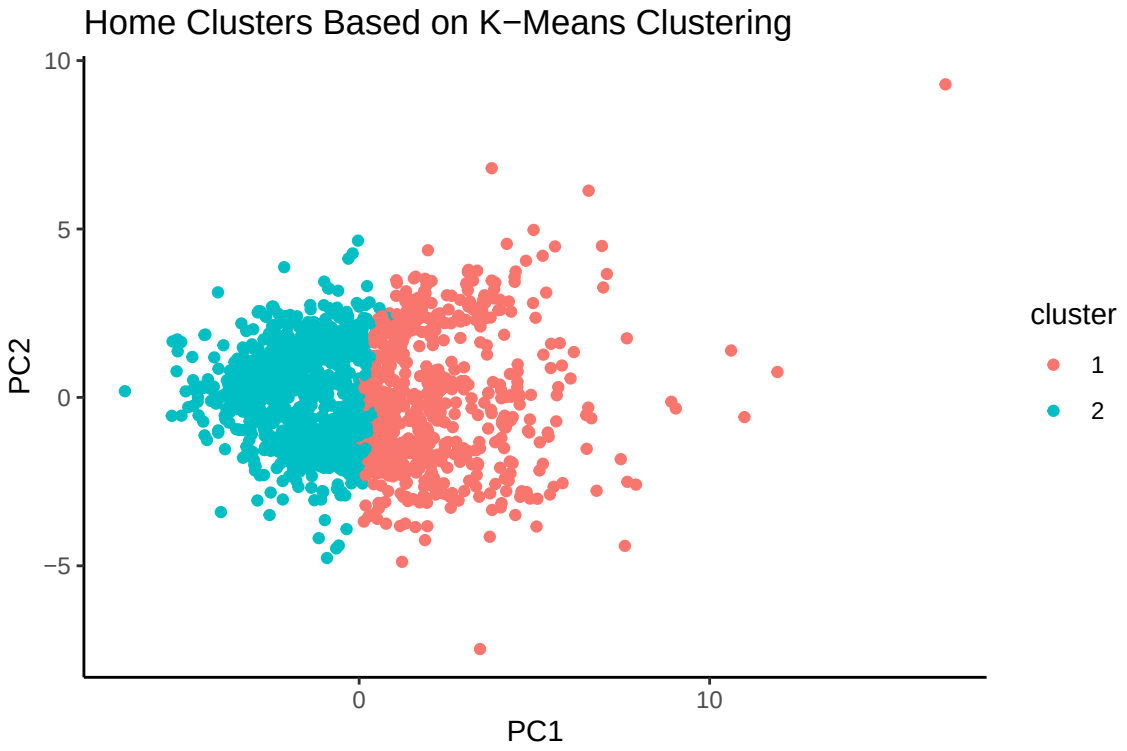
Because we are using PCA as a secondary technique, let's proceed to visualize our solution on the PC space with caution.

First, let's encode the scores in the data set that already contains the k-means clusters, so that we can use them to plot cluster comparisons. We will only include the scores for $PC_1$ and $PC_2$ because the principal components are ordered.

```
homes_sub <- mutate(homes_sub, PC1 = homePCAs$scores[, 1],
                    PC2 = homePCAs$scores[, 2])
```

Now, let's plot the clusters.

```
gf_point(PC2 ~ PC1, data = homes_sub, color = ~ cluster) %>%
  gf_labs(title = "Home Clusters Based on K-Means Clustering") +
  theme_classic()
```




Interestingly, there is a pretty distinct separation of homes between cluster 1 and cluster 2, and the separation seems to occur along $PC_1 = 0$. The observations appear to split between negative and positive values for $PC_1$. However, not each observation fits well in each cluster. Additionally, it is difficult to interpret what is characteristic of the 2 groups because none of the categorical variables in our data set has just 2 levels. It would be interesting to investigate that separation further by perhaps looking into the interpretability of $PC_1$, but that is not the focus of this analysis.

Of importance to note, however, is that $PC_1$ and $PC_2$ only explain ~34% of the variation in the original data set, and our final clustering solution had no real structure.

### Classification Analysis

We were not able to recover any zoning groups through a clustering analysis. Let's attempt to run a classification analysis and investigate whether we can build a predictive model for `MSZoning` instead.

For reference, the composition of class sizes in the data set is as follows:

```
tally(~MSZoning, data = data)
```

```
## MSZoning
## C (all)      FV      RH      RL      RM
##      10      65      16    1151     218
```

The root node error is determined by finding the error rate if we classified everything into the largest class. In this case, it would be $\frac{1460-1151}{1460} = 0.2116 = 21.16\%$. We want our models to have a lower error rate than this.

### Tree Model

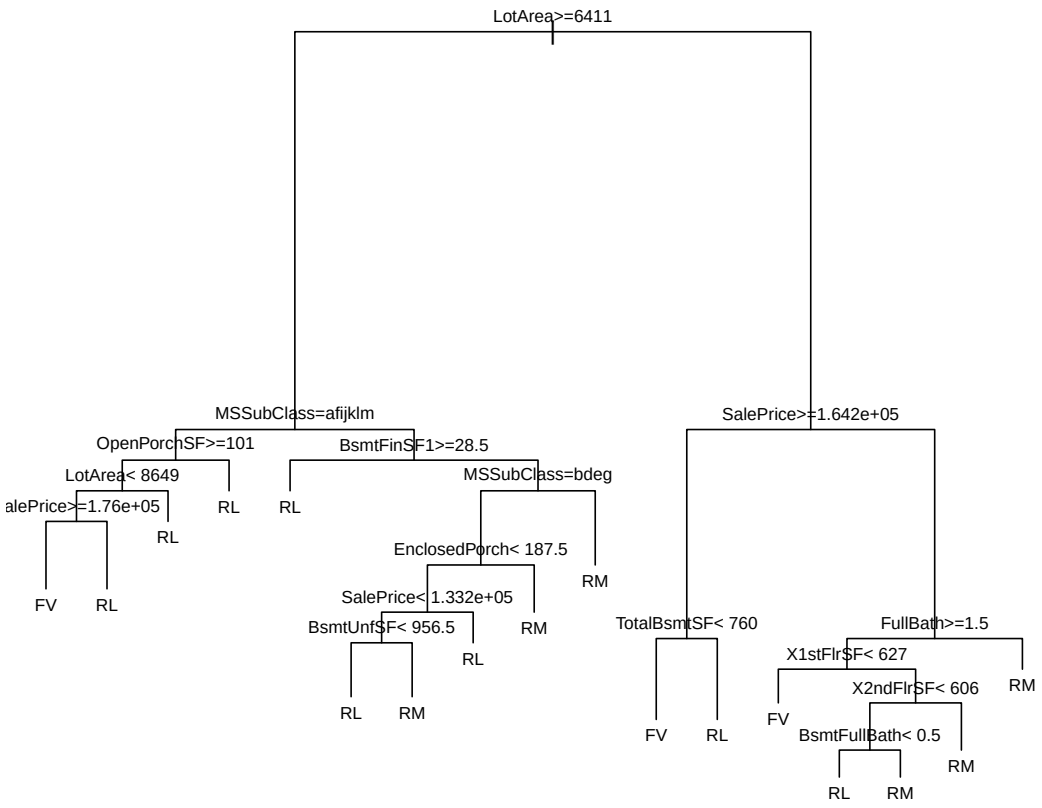
After trying out a few options, we will set a basic stopping criteria of `minbucket = 9` and `minsplit = 3`, meaning that the minimum number of observations in order to make a split is 9, and each bucket will contain at least 3 observations after a split. Smaller values allow for a more accurate solution. Note that we will leave the CP criteria at its default stopping level of 0.01. We will also use Leave One Out Cross Validation (LOOCV)/ Jackknife Cross Validation to test our model. That is, we will set `xval = 1460` (the number of observations).

Interestingly, only 12 variables were used in the tree construction: `BsmtFinSF1`, `BsmtFullBath`, `BsmtUnfSF`, `EnclosedPorch`, `FullBath`, `LotArea`, `MSSubClass`, `OpenPorchSF`, `SalePrice`, `TotalBsmtSF`, `X1stFlrSF`, and `X2ndFlrSF`.

```
set.seed(240)
g.control <- rpart.control(minsplit = 9, minbucket = 3, xval = 1460)
g.treeorig <- rpart(MSZoning ~ ., data = data, method = "class", control = g.control)
printcp(g.treeorig)
```

```
##
## Classification tree:
## rpart(formula = MSZoning ~ ., data = data, method = "class",
##     control = g.control)
##
## Variables actually used in tree construction:
##  [1] BsmtFinSF1    BsmtFullBath  BsmtUnfSF     EnclosedPorch FullBath
##  [6] LotArea       MSSubClass    OpenPorchSF   SalePrice     TotalBsmtSF
## [11] X1stFlrSF     X2ndFlrSF
##
## Root node error: 309/1460 = 0.2116
##
## n= 1460
##
##         CP nsplit rel error xerror    xstd
## 1 0.19094      0    1.0000 1.0000 0.05051
## 2 0.10032      1    0.8091 0.8155 0.04673
## 3 0.03883      2    0.7087 0.7184 0.04440
## 4 0.01294      3    0.6699 0.6990 0.04390
## 5 0.01187      7    0.6181 0.7540 0.04529
## 6 0.01079     13    0.5340 0.7638 0.04552
## 7 0.01000     16    0.5016 0.7540 0.04529
```

```
plot(g.treeorig)
text(g.treeorig, cex = 0.7)
```




There are 17 different end nodes, which is quite large. `RL` and `RM` seem to be the hardest to classify and require a lot of different rules. This makes sense because they are the largest classes. However, `C` and `RH` have not been included in any of the end nodes of this tree. This will be important to note when assessing the solution.

This model has an apparent error rate (AER) of 0.106139 – 10.6% – and an estimated true error rate (TER) of 0.159546 – 16%. This is quite good to begin with.

```
0.2116 * 0.5016
```

```
## [1] 0.106139
```

```
0.2116 * 0.7540
```

```
## [1] 0.159546
```

### Random Forest Model

Now, let's fit a random forest model that will combat some of the limitations of a tree model.

After trying a few options, will set `mtry = 6` and `ntree = 1000`, meaning that each split, 6 variables are chosen at random, and 1000 trees will be grown in this forest. This model will also assess the importance of variables as well as calculate the proximity measure among the rows. All variables were used to fit the model.

From the confusion matrix, this model has an AER of 0%, and from the R output, the estimated TER is 12.33%. This is much lower than the estimated TER of the classification tree, so this model is performing much better on the data set. The estimated TER is also reasonably low, so this model is promising.

Of note, `RH` and `C`, our smallest classes, both have a class error rate of 100%, which is a concern since those make up 2 out of the 5 classes present. Perhaps the small relative class sizes is impacting the ability of the model to classify these classes.

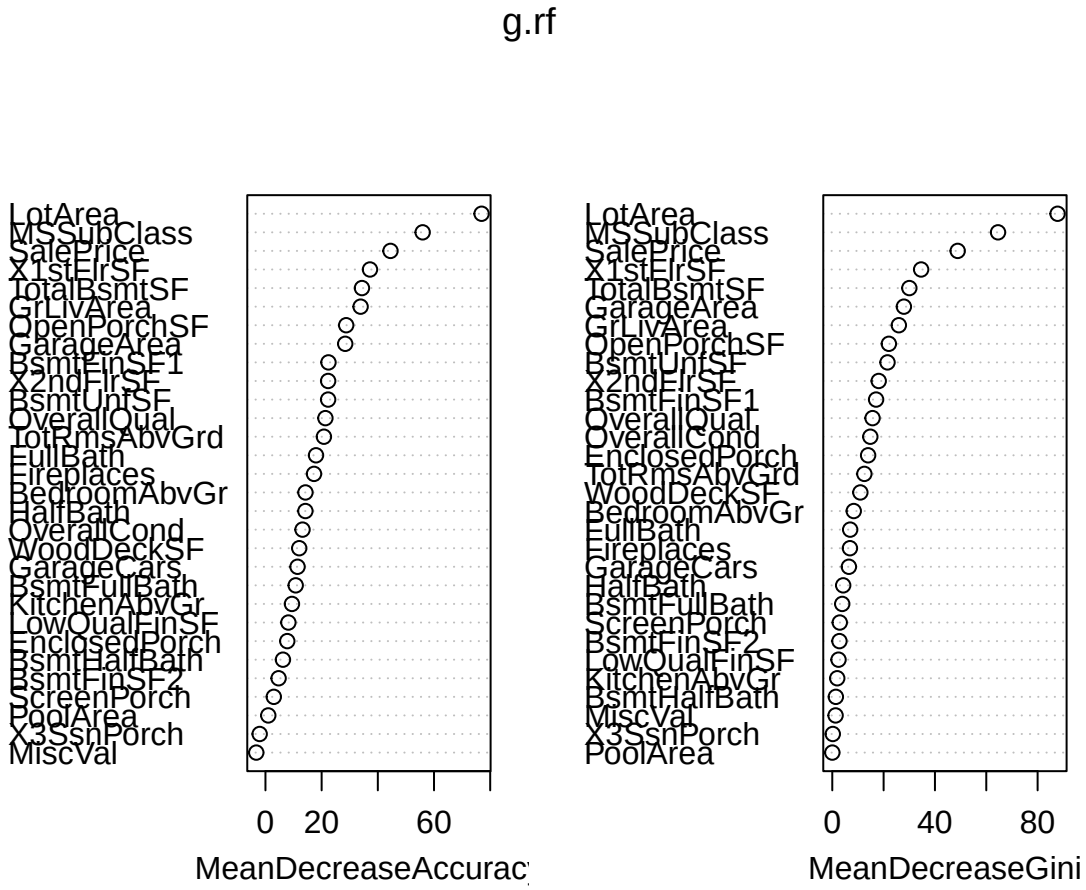
```
set.seed(240)
g.rf <- randomForest(as.factor(MSZoning) ~ ., data = data, mtry = 6, ntree = 1000,
                     importance = T, proximity = T)
g.rf
```

```
##
## Call:
##  randomForest(formula = as.factor(MSZoning) ~ ., data = data,      mtry = 6, ntree = 1000, importanc
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 12.33%
## Confusion matrix:
##         C (all) FV RH   RL  RM class.error
## C (all)       0  0  0    8   2   1.0000000
## FV            0 25  0   40   0   0.6153846
## RH            0  0  0   10   6   1.0000000
## RL            0  0  0 1119  32   0.0278019
## RM            0  0  0   82 136   0.3761468
```

```
table(data$MSZoning, predict(g.rf, data))
```

```
##
##           C (all)   FV   RH   RL   RM
##   C (all)      10    0    0    0    0
##   FV            0   65    0    0    0
##   RH            0    0   16    0    0
##   RL            0    0    0 1151    0
##   RM            0    0    0    0  218
```

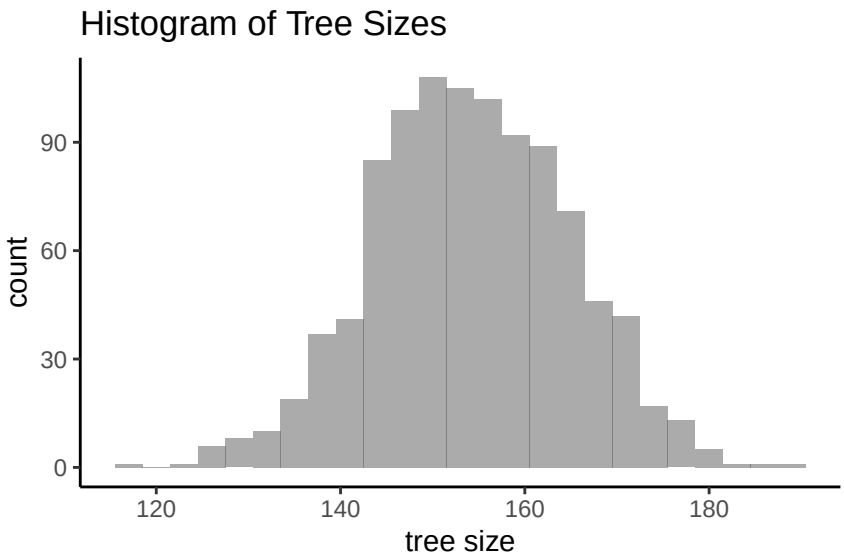
```
varImpPlot(g.rf)
```




We also tested for variable importance, as seen in the plots above. Variable importance is measured by how often a variable was used in the decision whenever it was randomly picked, as measured by 2 indexes: mean decrease accuracy and mean decrease gini. From the plot, the most important variable is `LotArea` on both indexes, followed by `MSSubClass` and `SalePrice` respectively.

Let's examine the distribution of the size of the trees in this forest.

```
gf_histogram(~ treesize(g.rf), title = "Histogram of Tree Sizes", xlab = "tree size") +
  theme_classic()
```




Most of the trees are quite large, with a mode at ~150. However, we don't notice anything of concern in the tree-size distribution.

### Bagging

Bagging is a special case of a random forest with `mtry` = the number of variables. In this case, `mtry = 30`.

From the confusion matrix, this model has an AER of 0%, and from the R output, the estimated TER is 11.85%. This is the lowest estimated TER so far. `C` has a better class error rate of 90% but `RH` still has a class error rate of 100%. The distribution of tree sizes is not concerning, however, the tree sizes are now a bit smaller, with a mode at ~110. Bagging has improved on the random forests solution.

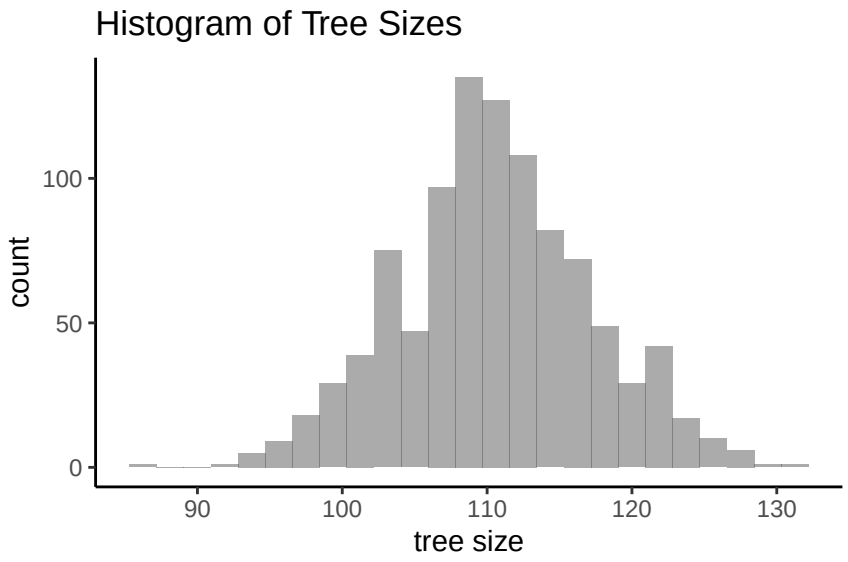
```
set.seed(240)
bf.rf <- randomForest(as.factor(MSZoning) ~ ., data = data, mtry = 30, ntree = 1000,
                      importance = T, proximity = T)
bf.rf
```

```
##
## Call:
##  randomForest(formula = as.factor(MSZoning) ~ ., data = data,      mtry = 30, ntree = 1000, importan
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 30
##
##         OOB estimate of  error rate: 11.85%
## Confusion matrix:
##         C (all) FV RH   RL  RM class.error
## C (all)       1  0  0    7   2   0.9000000
## FV            0 35  0   30   0   0.4615385
## RH            0  0  0   10   6   1.0000000
## RL            0  7  0 1100  44   0.0443093
## RM            0  2  0   65 151   0.3073394
```

```
table(data$MSZoning, predict(bf.rf, data))
```

```
##
##           C (all)   FV   RH   RL   RM
##   C (all)      10    0    0    0    0
##   FV            0   65    0    0    0
##   RH            0    0   16    0    0
##   RL            0    0    0 1151    0
##   RM            0    0    0    0  218
```

```
gf_histogram(~ treesize(bf.rf), title = "Histogram of Tree Sizes", xlab = "tree size") +
  theme_classic()
```




### Boosting

Finally, boosting is a classification method that builds a large number of trees in sequence, and each tree depends on the one before.

Because there are no out of bag observations, we need to first use the holdout sample approach to split the data into a train set and a test set. We will build our boosting model using the train set.

```
set.seed(240)

n <- nrow(data)
train_index <- sample(1:n, 0.75 * n)
test_index <- setdiff(1:n, train_index)

data_train <- data[train_index, ] #could use slice too
data_test <- data[test_index, ]

tally(~ MSZoning, data_train)
```

```
## MSZoning
## C (all)      FV      RH      RL      RM
##       9      52      11     867     156
```

```
set.seed(240)
data.boost <- gbm(MSZoning ~ ., data = data_train,
                  distribution = "multinomial",
                  n.trees = 5000,
                  interaction.depth = 2,
                  shrinkage = 0.1) #n.trees = B, interaction.depth = d
```

Now, let's test our model using the train set.

```
boost_estimate <- predict(data.boost,
                          newdata = data_test,
                          n.trees = 5000, type = "response")

pred_data <- apply(boost_estimate, 1, which.max)
tally(~ pred_data)
```

```
## pred_data
##   1   2   3   4   5
##   2  14   2 290  57
```

```
table(data_test$MSZoning, pred_data)
```

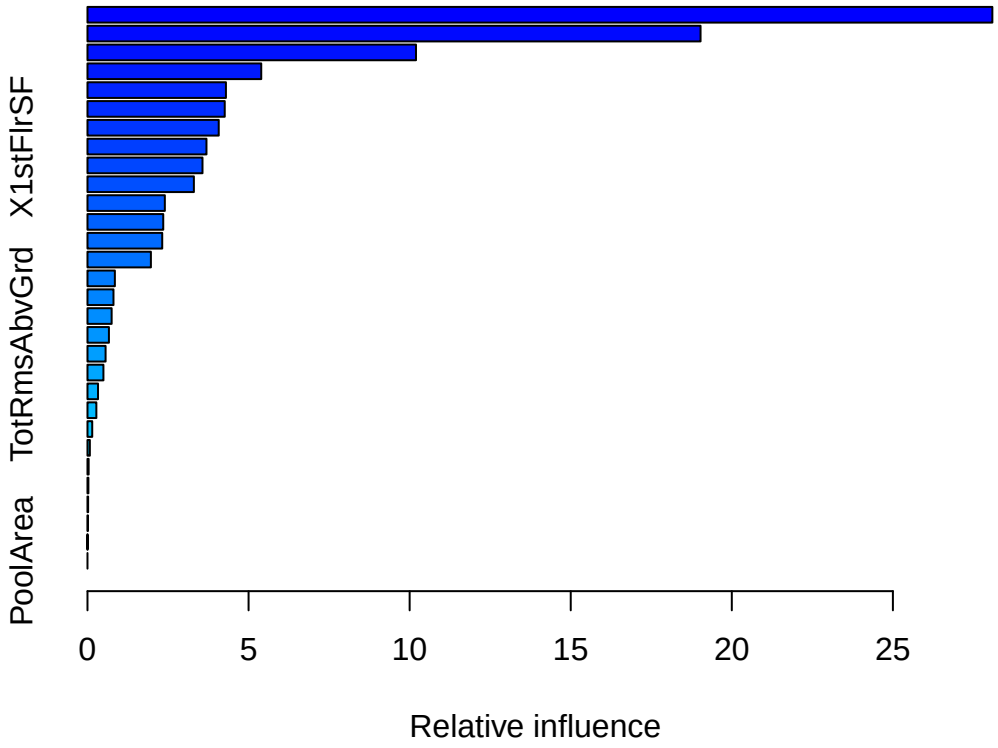
```
##           pred_data
##              1   2   3   4   5
##   C (all)    0   0   0   0   1
##   FV         0   8   0   5   0
##   RH         0   0   0   4   1
##   RL         1   6   0 265  12
##   RM         1   0   2  16  43
```

From the confusion matrix above, the estimated TER is 0.134247 – 13.42%.

```
49/365
```

```
## [1] 0.134247
```

```
summary(data.boost)
```




```
##                          var     rel.inf
## MSSubClass       MSSubClass 28.08778052
## LotArea             LotArea 19.03009276
## SalePrice         SalePrice 10.19710989
## GarageArea       GarageArea  5.39333406
## OpenPorchSF     OpenPorchSF  4.29765661
## OverallQual     OverallQual  4.26077361
## TotalBsmtSF     TotalBsmtSF  4.07407066
## X1stFlrSF         X1stFlrSF  3.69322318
## EnclosedPorch EnclosedPorch  3.56972275
## BsmtUnfSF         BsmtUnfSF  3.30256949
## BsmtFinSF1       BsmtFinSF1  2.40129936
## GrLivArea         GrLivArea  2.35188487
## OverallCond     OverallCond  2.31988151
## X2ndFlrSF         X2ndFlrSF  1.96866028
## BedroomAbvGr   BedroomAbvGr  0.84999541
## WoodDeckSF       WoodDeckSF  0.80674140
## Fireplaces       Fireplaces  0.75021909
## FullBath           FullBath  0.66523116
## TotRmsAbvGrd   TotRmsAbvGrd  0.56221855
## LowQualFinSF   LowQualFinSF  0.49482151
## GarageCars       GarageCars  0.32704613
## BsmtFinSF2       BsmtFinSF2  0.27582667
```

```
## KitchenAbvGr   KitchenAbvGr  0.14750275
## ScreenPorch     ScreenPorch  0.07346694
## MiscVal             MiscVal  0.03541687
## HalfBath           HalfBath  0.02850372
## BsmtFullBath   BsmtFullBath  0.01974283
## BsmtHalfBath   BsmtHalfBath  0.01372282
## X3SsnPorch       X3SsnPorch  0.00148457
## PoolArea           PoolArea  0.00000000
```

It's difficult to read the plot above, so we will use the table to examine the relative importance of different variables in building the boosting model. `MSSubClass`, `LotArea`, and `SalePrice` have the highest relative influence respectively. These were the 3 most important variables in the random forests solution as well.

### Preferred Classification Model

Note that we cannot run Linear Discriminant Analysis (LDA) or Quadratic Discriminant Analysis (QDA) on our entire data set because our variables violate the univariate normal condition, and so certainly, the entire data set is not multivariate normal.

We also fit a K-NN Classification model, which classifies observation based on a majority rule of the classes of the observation's $k$ neighbors. This model had an AER of 11.3%. Estimated TERs are expected to be higher than the AER, so we did not proceed further with this model. It likely was not also a good fit because of the highly varying class sizes.

Therefore, the final classification model of choice to predict `MSZoning` is the bagging model, which has an AER of 0% and an estimated TER is 11.85%.

This model allows us to classify homes into their respective zones correctly 11.85% of the time using the variables in the data set. We would hope for a model with a lower error rate, but this is quite good for such a data set.

### Reattempting Clustering with 2 Variables

Now that we have run classification, let's attempt to run one final clustering solution with the 3 most important variables from both the random forests output and the bagging output: `LotArea`, `SalePrice`, and `MSSubClass`. Note that `MSSubClass` is a categorical variable, so we will only use `LotArea` and `SalePrice`, and set $k = 2$.

Before, we used all the variables in the data set to find natural groups and assess whether any zoning groups were recovered. Now that we have some insight on what variables play the most important role in classifying homes into their respective zoning groups, let's re-examine whether we can obtain a better 2-cluster solution.

```
set.seed(240)

Ksol2 <- kmeans(scale(select(quant_data, c(LotArea, SalePrice))),
                centers = 2, nstart = 10)
list(Ksol2$size)
```

```
## [[1]]
## [1]  317 1143
```

Notice that the class sizes are very different, with cluster 1 having 317 observations and cluster 2 having 1,143 observations.

Let's assess whether this solution is better than our preferred solution that has all the variables.

```
kmeansSil2 <- silhouette(Ksol2$cluster,
                         dist(scale(select(quant_data, c(LotArea, SalePrice)))))
```

In looking at the summary, cluster 1 has a silhouette coefficient of 0.217 and cluster 2 has a silhouette coefficient of 0.662. Cluster 1 has no real structure. However, cluster 2 now has a reasonable structure. Of note, there are still some observations with negative silhouette values in both clusters, suggesting that those observations do not fit in either of the clusters at all. Most observations do not fit too well into cluster 1 either, but it seems that a number of observations fit very well into cluster 2.

The overall silhouette coefficient is 0.57, suggesting a reasonable structure around the clustering solution. This is much higher than the silhouette value (0.16) of the preferred solution with all the variables.

However, it's important to note that 80% of the observations were placed into 1 cluster, suggesting very little separation between the observations overall. This almost suggests that clustering may not be the best technique for this data set because the distances between observations are not large enough.

```
kmeansSil2 <- as.data.frame(kmeansSil2)
mean(kmeansSil2$sil_width)
```

```
## [1] 0.565355
```

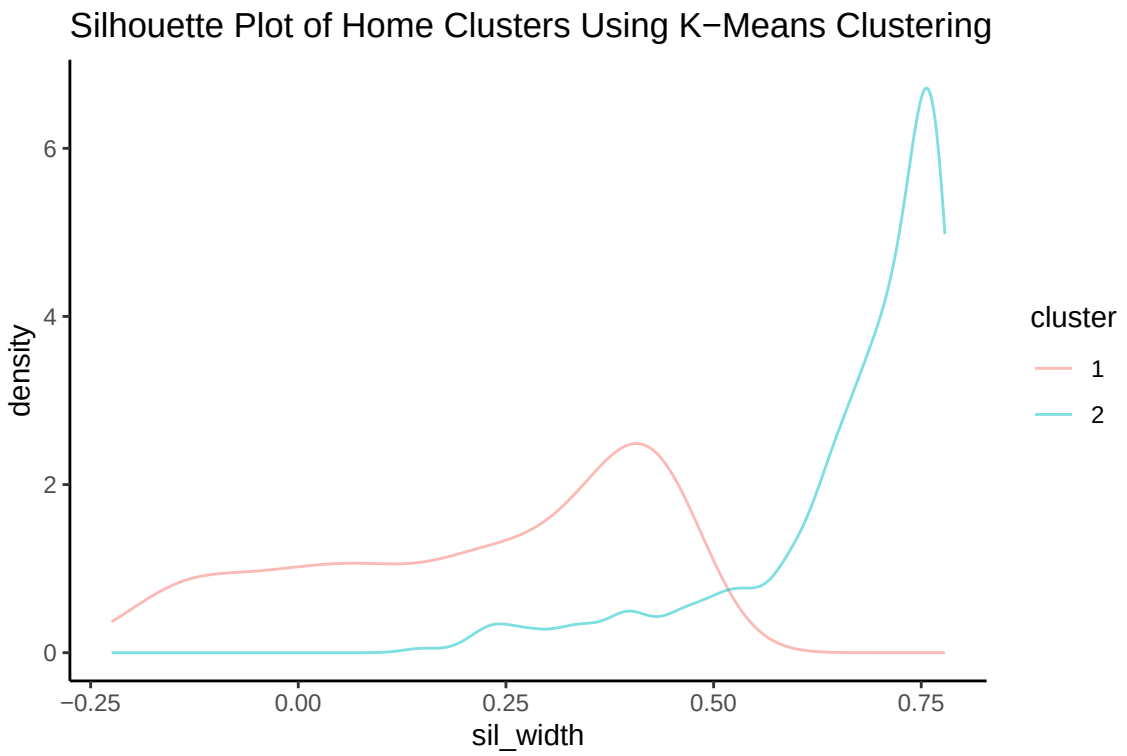
```
mean(kmeansSil2$sil_width[kmeansSil2$cluster == 1])
```

```
## [1] 0.216932
```

```
mean(kmeansSil2$sil_width[kmeansSil2$cluster == 2])
```

```
## [1] 0.661986
```

```
kmeansSil2 <- mutate(kmeansSil2, cluster = factor(cluster))
gf_dens(~ sil_width, color = ~ cluster, data = kmeansSil2,
        title = "Silhouette Plot of Home Clusters Using K-Means Clustering") +
  theme_classic()
```



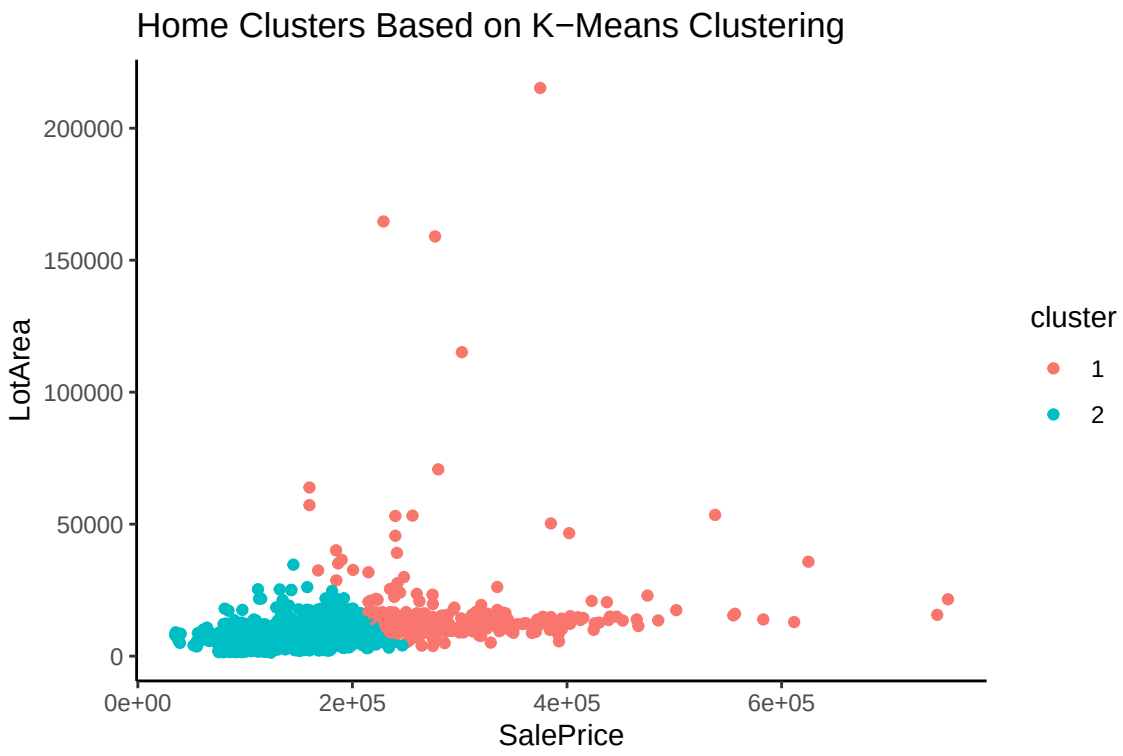

Before we visualize this solution, let's assess whether any of the zoning groups were recovered. We see much better separation in this solution. All homes of zone `C` and `RH` belong in cluster 2 – note that these were the hardest zones to classify in the classification analysis. Most homes of type `RM` are also classified into cluster 2. Finally, there seems to be an uneven split between `FV` and `RL` across the 2 clusters. It would be worth investigating why that may be the case in a future analysis, but perhaps what we can initially draw from this is that commercial homes, residential high density homes, and residential medium density homes are most similar. Some floating village residential and some residential low density homes are quite similar to each other, while most of both of these zones tend to be clustered with the former 3 zones. This is particularly with regard to the lot area and sales price of the property.

```
homes_sub2 <- mutate(data, cluster = factor(Ksol2$cluster))
tally(MSZoning ~ cluster, data = homes_sub2)
```

```
##          cluster
## MSZoning    1   2
##   C (all)   0  10
##   FV       20  45
##   RH        0  16
##   RL      290 861
##   RM        7 211
```

Finally, let's visualize this solution on a two-dimensional plane.

```
gf_point(LotArea ~ SalePrice, data = homes_sub2, color = ~ cluster) %>%
  gf_labs(title = "Home Clusters Based on K-Means Clustering") +
  theme_classic()
```




Once again, we see a very distinct separation, this time along `SalePrice = ~200,000`. Evidently, many observations do not fit well into cluster 1. However, most of the observations are closely clustered together, so it's not easy to make out any distinct clusters. This may warrant deeper investigation in future.

## Conclusion

The purpose of this analysis was to assess whether characteristics of a home hold enough information on the zoning group that the home belongs to. As such, the key research question was: can we classify homes into their respective zoning groups using various home attributes such as the number of bedrooms, the lot area, the valued sales price, and the number of fireplaces. To answer this research question, we ran a clustering analysis – which is a form of unsupervised learning – and a classification analysis – which is a form of supervised learning – after performing preliminary analysis.

Our preferred clustering solution was a k-means solution that had just 2 variables: `SalePrice` and `LotArea`. These variables were also 2 of the 3 most important variables in the classification analysis. This solution had a silhouette coefficient of 0.57. Examining the solution further, it seemed that commercial homes, residential high density homes, and residential medium density homes are most similar. Some floating village residential and some residential low density homes are quite similar to each other, while most of both of these zones tend to be clustered with the former 3 zones. Nevertheless, most observations fit into one cluster, and visualizing the solution revealed that the observations were clustered together with little, if any, observable difference. The silhouette coefficient alone, without examining the solution further, is not an indication that this clustering solution is reasonable. In fact, it seems that the distances between the observations are not large enough to make clustering a suitable technique for this data set.

We also ran a classification analysis using a variety of methods. Using all the 30 final variables when performing clustering resulted in very weak solutions, with the best solution having a silhouette coefficient of 0.16. On the contrary, quite the opposite occurred during the classification analysis. Our best model was obtained through bagging, which uses all the variables at every iteration, and had an estimated true error rate of 11.85%. This model is quite strong, accurately classifying homes into their zoning groups 88.15% of the time. Note that homes of zone `C` and `RH` were hardest to classify, likely because they had very few observations in comparisons to the other zones.

However, there are some limitations with both the data and my analysis. Unfortunately, I don't have much information on how the data was pulled and compiled, and whether it is representative enough or there were some biases in the data collection process. Ideally, we should be able to say something about home zoning in Boston, but we can't make that claim with full confidence. The data was also presumably last updated 7 years ago, so any conclusions would not be recent and necessarily transferable to the present day.

Additionally, I do not have expert knowledge on home attributes and I had a very focused research question. There were 81 variables in this data set and there are so many potential questions to answer. Future analysis could further my work by looking into, and perhaps better understanding, what variables are most important for classifying homes into zoning groups. More expert knowledge on which variables are most influential and a richer data set, perhaps with data for all 8 home zoning groups and with more observations, may also have provided a better solution.

With regard to improving my analysis, a future researcher could use optimization tools and techniques to pick the best hyper-parameters for my classification models. I tried out a few different options before settling on the lowest error rate I could yield, but a more automated approach would allow for more confidence in the final model.

Overall, it was quite interesting to investigate whether home attributes are predictive of home zoning groups. While the final classification model is promising, in future, I would perhaps further subset the variables and only use a few for the clustering solution. Nevertheless, it's interesting that we did not obtain a strong clustering solution despite the fact that there were varying types of homes in the data set. This suggests that there either was not enough variability in the data set, that is, the sample of homes recorded are very similar, or that there is an external, confounding variable beyond the physical home that can explain different home zones and different living conditions and neighborhoods.

In conclusion, this is a very important question, especially with regard to understanding the societal differences in the housing sector, and this analysis began to uncover some of the distinctive characteristics of different zones. This is a positive step towards using data-driven approaches to answer such questions with real-world implications.

## Citations

Housing prices competition for Kaggle learn users. (n.d.). Retrieved November 29, 2022, from https://www.kaggle.com/competitions/home-data-for-ml-course/data.