HarvardX PH125.9xData Science: Capstone Choose Your Own!(House Pricing)

Jose Quesada

27/12/2020

Executive Summary

First of all, I hope that in this difficult time you and your family are well and thank you for your time to see my final task, this course I started to have a good foundation and better understand the world of data science, I apologize for me English is not my native language. I did my best, I started doing this module in mid-December and I didn't have much time to dedicate to it. This project was not easy, I chose a dataset of 81 variables

Introduccion

This project consists of determining the value of a house according to its location, property characteristics and payment methods using the data set from kaggle House Prices - Advanced Regression Techniques https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data (kaggle competitions download -c house-prices-advanced-regression-techniques).

Description by Kaggle:

You have some experience with R or Python and machine learning basics. This is a perfect competition for data science students who have completed an online course in machine learning and are looking to expand their skill set before trying a featured competition. Competition Description

Ask a home buyer to describe their dream house, and they probably won't begin with the height of the basement ceiling or the proximity to an east-west railroad. But this playground competition's dataset proves that much more influences price negotiations than the number of bedrooms or a white-picket fence.

With 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa, this competition challenges you to predict the final price of each home. Practice Skills

```
Creative feature engineering
Advanced regression techniques like random forest and gradient boosting
```

Acknowledgments

The Ames Housing dataset was compiled by Dean De Cock for use in data science education. It's an incredible alternative for data scientists looking for a modernized and expanded version of the often cited Boston Housing dataset.

Importing Data.

Data set was already divided into training and test, the training file contains an additional column that would be the sale price, while the test set does not have this column, for the purposes of this exercise we combine both data sets to avoid staying with an unknown value at the time of cleaning and pre-processing

Files:

train.csv - the training set test.csv - the test set data_description.txt - full description of each column. sample_submission.csv - a benchmark submission from a linear regression on year and month of sale, lot square footage, and number of bedrooms

```
read_csv <- function(file){
    path_data <- "data"
    filename <- paste(path_data,file,sep="/")
    csv__ <- read.csv(filename)
    csv__
}

test_set <-read_csv('test.csv')
train_set<- read_csv('train.csv')


#Join datasets, For this project we going to join train and set data for the cleansing and EDA,
#later we going to split again by SalesPrices not null as train set and test set is null.
df<- bind_rows(train_set,test_set)
```

EDA

train_set:

```
* Dimensions: 1460, 81
* Memory Usage: 0.7 Mb
```

test_set:

```
* Dimensions: 1459, 80
* Memory Usage: 0.7 Mb
```

Comparing amount of columns between each dataset we can see that we have 1 more column in the train set vs the test set. **SalePrice** is the additional column in the train set and our **target value** for this model We going to use the train set to predict **SalePrice** on the test, first we going to make some EDA and data cleaning.

Total categorical columns: 43

Categorical Columns				
MSZoning	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	PoolQC	Fence
MiscFeature	SaleType	SaleCondition	MSZoning	Street
Alley	LotShape	LandContour	Utilities	LotConfig

Total numeric columns: 38

Numerical Columns			
Id	MSSubClass	LotFrontage	LotArea
OverallQual	OverallCond	YearBuilt	YearRemodAdd
MasVnrArea	BsmtFinSF1	BsmtFinSF2	BsmtUnfSF
TotalBsmtSF	X1stFlrSF	X2ndFlrSF	LowQualFinSF
GrLivArea	BsmtFullBath	BsmtHalfBath	FullBath
HalfBath	BedroomAbvGr	KitchenAbvGr	TotRmsAbvGrd
Fireplaces	GarageYrBlt	GarageCars	GarageArea
WoodDeckSF	OpenPorchSF	EnclosedPorch	X3SsnPorch
ScreenPorch	PoolArea	MiscVal	MoSold
YrSold	SalePrice	Id	MSSubClass

OverAll Missing Values

For this analysis we going to select just the columns that have missing values, **if they not in plot or table its because they not have missing values**. Im using is.na function to detect null values, i created the fuction missing_values where the imput is a data.table, this function get the name of each column where is a least 1 null values, we donde this using colsMean(is.na(df)) and created a summary table with the percentage of null values of each column.

Description of columns with Missing Values:

Missing Categorical Columns	
name	prc_na
PoolQC	0.9965742
MiscFeature	0.9640288
Alley	0.9321686
Fence	0.8043851
FireplaceQu	0.4864680
GarageFinish	0.0544707
GarageQual	0.0544707
GarageCond	0.0544707
GarageType	0.0537855
BsmtCond	0.0280918
BsmtExposure	0.0280918
BsmtQual	0.0277492
BsmtFinType2	0.0274066
BsmtFinType1	0.0270641
MasVnrType	0.0082220
MSZoning	0.0013703
Utilities	0.0006852
Functional	0.0006852
Exterior1st	0.0003426
Exterior2nd	0.0003426
Electrical	0.0003426
KitchenQual	0.0003426
SaleType	0.0003426

Missing Numerical Columns	
name	prc_na
SalePrice	0.4998287
LotFrontage	0.1664954
GarageYrBlt	0.0544707
MasVnrArea	0.0078794
BsmtFullBath	0.0006852
BsmtHalfBath	0.0006852
BsmtFinSF1	0.0003426
BsmtFinSF2	0.0003426
BsmtUnfSF	0.0003426
TotalBsmtSF	0.0003426
GarageCars	0.0003426
GarageArea	0.0003426

Handling Missing Values(extract from "https://en.wikipedia.org/wiki/Imputation_(statistics)")

Imputation In statistics, imputation is the process of replacing missing data with substituted values. When substituting for a data point, it is known as "unit imputation"; when substituting for a component of a data point, it is known as "item imputation". There are three main problems that missing data causes: missing data can introduce a substantial amount of bias, make the handling and analysis of the data more arduous, and create reductions in efficiency.[1] Because missing data can create problems for analyzing data, imputation is seen as a way to avoid pitfalls involved with listwise deletion of cases that have missing values. That is to say, when one or more values are missing for a case, most statistical packages default to discarding any case that has a missing value, which may introduce bias or affect the representativeness of the results. Imputation preserves all cases by replacing missing data with an estimated value based on other available information. Once all missing values have been imputed, the data set can then be analysed using standard techniques for complete data.[2] There have been many theories embraced by scientists to account for missing data but the majority of them introduce bias. A few of the well known attempts to deal with missing data include: hot deck and cold deck imputation; listwise and pairwise deletion; mean imputation; non-negative matrix factorization;[3] regression imputation; last observation carried forward; stochastic imputation; and multiple imputation.

Method to use. Mean substitution

Another imputation technique involves replacing any missing value with the mean of that variable for all other cases, which has the benefit of not changing the sample mean for that variable. However, mean imputation attenuates any correlations involving the variable(s) that are imputed. This is because, in cases with imputation, there is guaranteed to be no relationship between the imputed variable and any other measured variables. Thus, mean imputation has some attractive properties for univariate analysis but becomes problematic for multivariate analysis.

Regression

Regression imputation has the opposite problem of mean imputation. A regression model is estimated to predict observed values of a variable based on other variables, and that model is then used to impute values in cases where the value of that variable is missing. In other words, available information for complete and incomplete cases is used to predict the value of a specific variable. Fitted values from the regression model are then used to impute the missing values. The problem is that the imputed data do not have an error term included in their estimation, thus the estimates fit perfectly along the regression line without any residual variance. This causes relationships to be over identified and suggest greater precision in the imputed values than is warranted. The regression model predicts the most likely value of missing data but does not supply uncertainty about that value.

Stochastic regression was a fairly successful attempt to correct the lack of an error term in regression imputation by adding the average regression variance to the regression imputations to introduce error. Stochastic regression shows much less bias than the above-mentioned techniques, but it still missed one thing – if data are imputed then intuitively one would think that more noise should be introduced to the problem than simple residual variance.[5]

Identify associated columns by Name

By looking in the data_description file, we can determine that we have columns that show us measurements, condition and qualities of additional features of the houses, They are defined with NA when they do not have one of them. To determine if they are really null values, we must compare multiple columns, example if we have NA PoolQC and PoolArea equal to 0 is the NA is not a Missing value, because the house doesnt have a pool, if PoolQC is NA but the PoolArea is greater than 0, we have a missing value.

Im using key word to detect related columns, this is a manual process by looking the data_description.txt file, after this I am iterating in this list of words to detect the columns that contain this word and identifying what type of data it is (categorical or numerical), i created 3 empty variable, where im storing the result of each loop and using n as index.

Related Features		
name_features	dim_features	dtype
MasVnr		
MasVnr	MasVnrArea	numeric
MasVnr	MasVnrType	categorical
Bsmt		
Bsmt	BsmtCond	categorical
Bsmt	BsmtExposure	categorical
Bsmt	BsmtFinSF1	numeric
Bsmt	BsmtFinSF2	numeric
Bsmt	BsmtFinType1	categorical
Bsmt	BsmtFinType2	categorical
Bsmt	BsmtFullBath	numeric
Bsmt	BsmtHalfBath	numeric
Bsmt	BsmtQual	categorical
Bsmt	BsmtUnfSF	numeric
Bsmt	TotalBsmtSF	numeric
Fireplace		
Fireplace	FireplaceQu	categorical
Fireplace	Fireplaces	numeric
Pool		
Pool	PoolArea	numeric
Pool	PoolQC	categorical
Heating		
Heating	Heating	categorical
Heating	HeatingQC	categorical
Misc		
Misc	MiscFeature	categorical
Misc	MiscVal	numeric
Kitchen		
Kitchen	KitchenAbvGr	numeric
Kitchen	KitchenQual	categorical
Exter		
Exter	ExterCond	categorical
Exter	Exterior1st	categorical
Exter	Exterior2nd	categorical
Exter	ExterQual	categorical
Garage		
Garage	GarageArea	numeric
Garage	GarageCars	numeric
Garage	GarageCond	categorical
Garage	GarageFinish	categorical
Garage	GarageQual	categorical
Garage	GarageType	categorical
Garage	GarageYrBlt	numeric
Lot		
Lot	LotArea	numeric
Lot	LotConfig	categorical
Lot	LotFrontage	numeric
Lot	LotShape	categorical

Categorical to Numerical DATA

*Label Encoder**: It is used to transform non-numerical labels to numerical labels (or nominal categorical variables). Numerical labels are always between 0 and n_classes-1.) and after we going to label encoder the column(It is used to transform non-numerical labels to numerical labels (or nominal categorical variables). Numerical labels are always between 0 and n_classes-1.).

After read the data description file i have identify Quality and Condition columns with values:

```
- Ex Excellent  (replace by: 5)
- Gd Good    (replace by: 4)
- TA Average (replace by: 3)
- Fa Fair (replace by: 2)
- Po Poor (replace by: 1)
- NA No (replace by: 0)
```

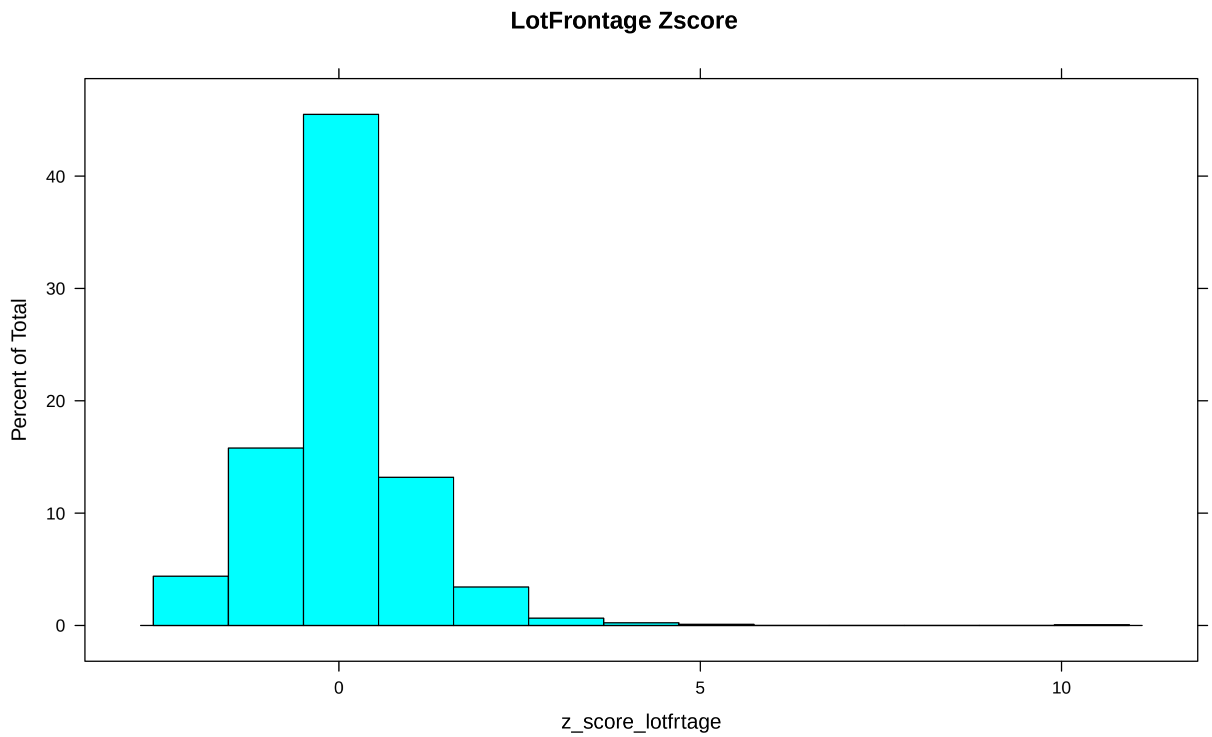
In each variable of the data set im replacing No existance feature for the string "None" or "No" + feature Name, and giving a score from 0 to No existant to 5 for Excellent quality or condition.

Variables

For the analysis and cleaning of the data I am going to divide into the different variables and related columns to be able to find null values and be able to replace it because it is because it is a null value because it does not have the characteristic or if it really is a null value where we have to do analysis correlation and use another column to be able to replace a grouped average and replace the values according to the case and try to convert the categorical data into numeric if possible, at the end of the cleaning of each variable (Pool, Basement, etc.) we will see the correlation of each column against the sale price and we will create a list of possible columns to discard for our model.

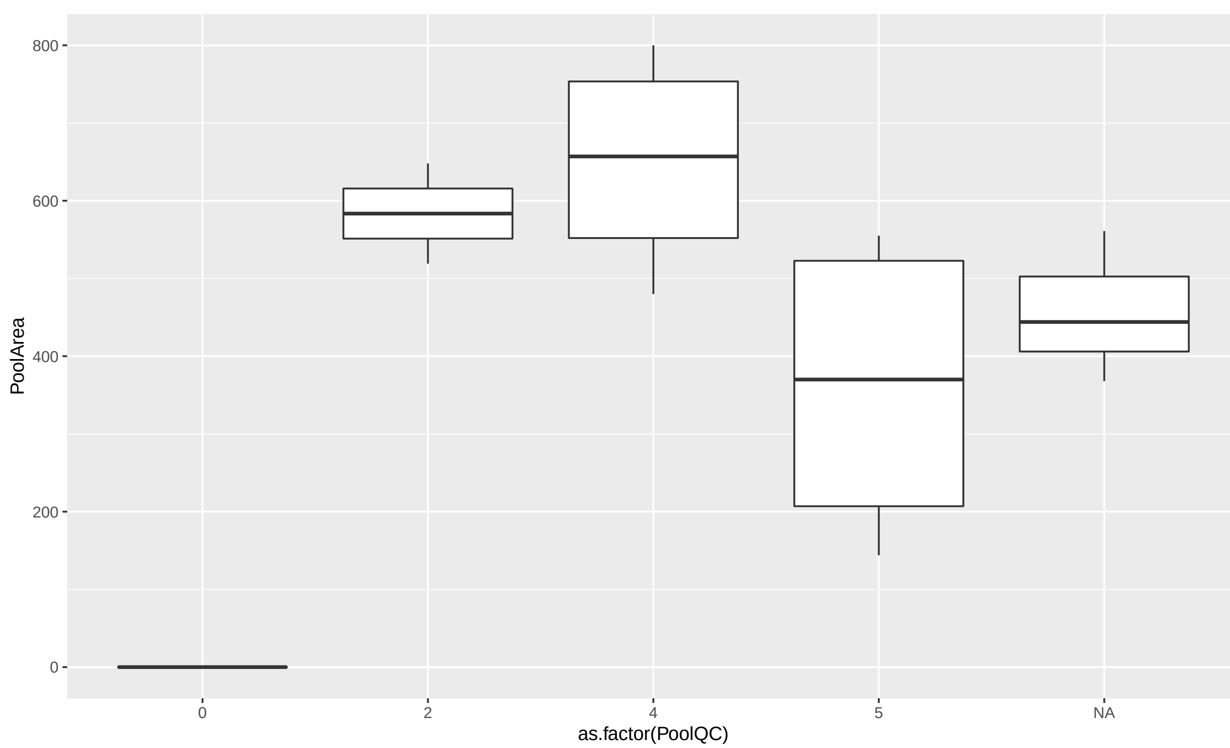
```
#HERE IM STORIGN COLUMNS TO DROP
cols_to_Drop <- NULL
```

LotFrontage This is a numeric column im going to calculate the z_score and understan the distribution, if is highly skewed distributionswe shoul use log transform to make it less skewed.



It is not necessary to apply a logarithmic transformation to it, now we going to calculate mean with values that are between -2.5 and 2.5 from median, and replace our missing values in this column

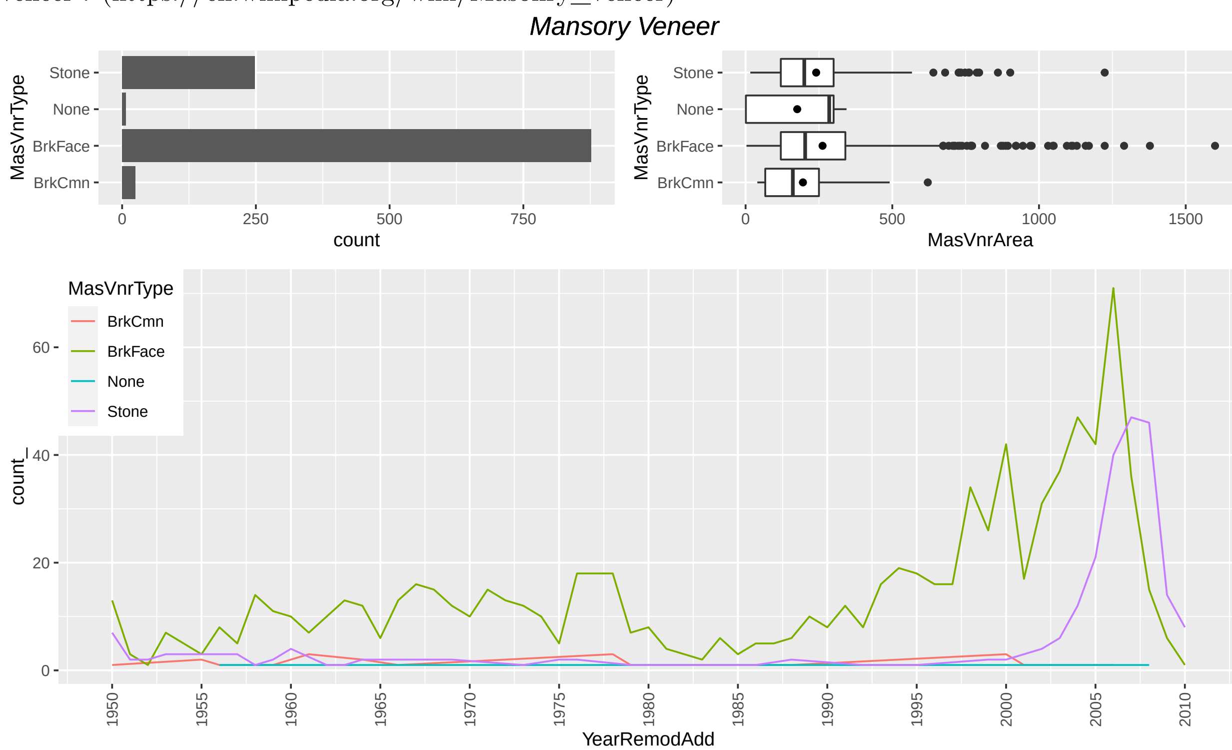
Pool First we going to replace NA in PoolArea to 0, then im replacing PoolQC to No Pool when Area is equal to 0 .



Just for looking into this graph im going to fill NA with 5.

Pool Missing Values		
name	prc_na	type
PoolArea	0	numerical
PoolQC	0	numerical

Masonry veneer. Masonry veneer walls consist of a single non-structural external layer of masonry, typically made of brick, stone or manufactured stone. Masonry veneer can have an air space behind it and is technically called “anchored veneer”. A masonry veneer attached directly to the backing is called “adhered veneer”. (https://en.wikipedia.org/wiki/Masonry_veneer)



At Overall we can see that break is used more than stone but the Year Vs count of each MasVnrType plot show us between 1950 and 2005 BrkFace was the predominant type of MasVnr and after 2005 was Stone, we going to get the mode in every yearn and fill Na values with mode by year and look how many null values we get.

Mansory Veneer		
name	prc_na	type
MasVnrArea	0	numerical
MasVnrType	0	categorical

Basement BsmtQual: Evaluates the height of the basement

```
Ex   Excellent (100+ inches)
Gd   Good (90-99 inches)
TA   Typical (80-89 inches)
Fa   Fair (70-79 inches)
```

```
Po   Poor (<70 inches
NA   No Basement
```

BsmtCond: Evaluates the general condition of the basement

```
Ex   Excellent
Gd   Good
TA   Typical - slight dampness allowed
Fa   Fair - dampness or some cracking or settling
Po   Poor - Severe cracking, settling, or wetness
NA   No Basement
```

BsmtExposure: Refers to walkout or garden level walls

```
Gd   Good Exposure
Av   Average Exposure (split levels or foyers typically score average or above)
Mn   Mimimum Exposure
No   No Exposure
NA   No Basement
```

BsmtFinType1: Rating of basement finished area

```
GLQ  Good Living Quarters
ALQ  Average Living Quarters
BLQ  Below Average Living Quarters
Rec  Average Rec Room
LwQ  Low Quality
Unf  Unfinshed
NA   No Basement
```

BsmtFinSF1: Type 1 finished square feet

BsmtFinType2: Rating of basement finished area (if multiple types)

```
GLQ  Good Living Quarters
ALQ  Average Living Quarters
BLQ  Below Average Living Quarters
Rec  Average Rec Room
LwQ  Low Quality
Unf  Unfinshed
NA   No Basement
```

BsmtFinSF2: Type 2 finished square feet

BsmtUnfSF: Unfinished square feet of basement area

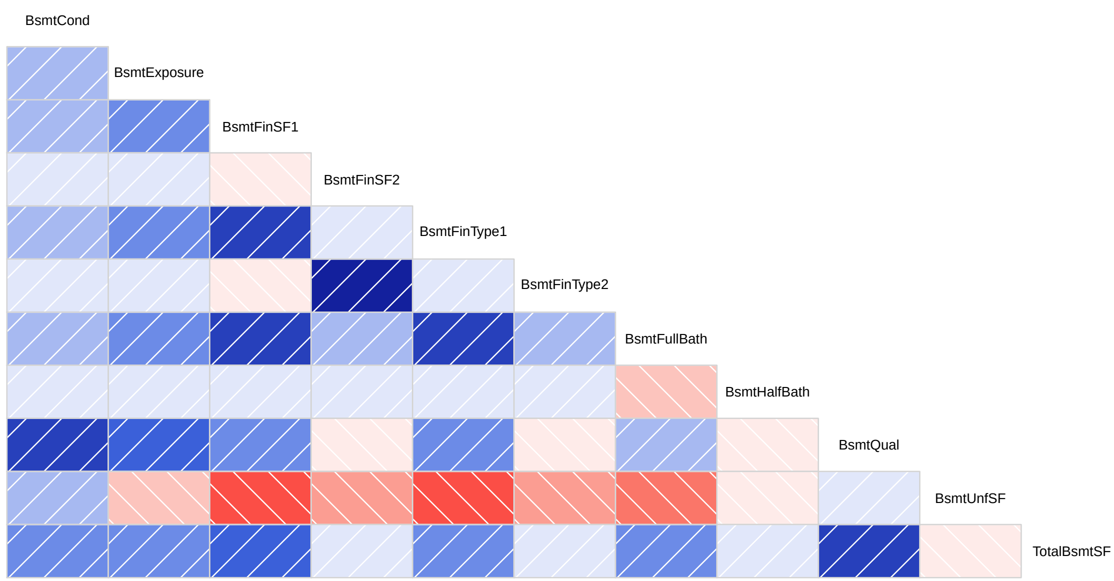
TotalBsmtSF: Total square feet of basement area

First we are going to replace the null values of columns Bsmt fSF, BsmtFinSF1, BsmtFinSF2 and TotalBsmtSF, by 0. After this we are going to replace the null values of the categorical columns by "No_Basement" and "No_Basement1" when BsmtFinSF1 is equal to 0 and "No_Basement2" when BsmtFinSF2 is equal to 0. After the first cleaning we are going to convert the BsmtCond columns, BsmtExposure, BsmtFinType1, BsmtFinType2 and BsmtQual, in numerical values, giving as a classification based on the descriptions that are in the file "data/data_description.txt", to be able to find correlations and finish replacing the null values in these columns, we also transform the BsmtUnfSF column into a percentage of the TotalBsmtSF.

Mansory Veneer		
name	prc_na	type
BsmtCond	0.0010277	numerical
BsmtQual	0.0006852	numerical
BsmtFinType2	0.0003426	numerical
BsmtExposure	0.0000000	numerical
BsmtFinSF1	0.0000000	numerical
BsmtFinSF2	0.0000000	numerical
BsmtFinType1	0.0000000	numerical
BsmtFullBath	0.0000000	numerical
BsmtHalfBath	0.0000000	numerical
BsmtUnfSF	0.0000000	numerical
TotalBsmtSF	0.0000000	numerical

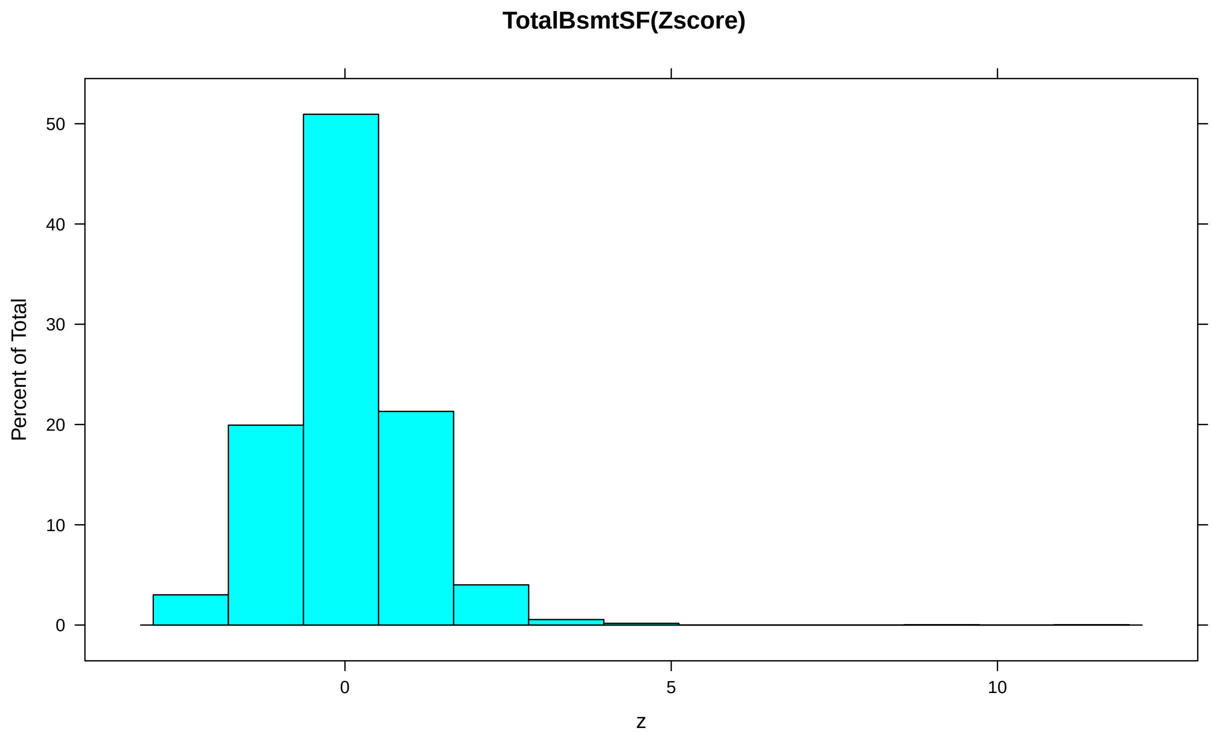
We are going to analyze variables that are highly correlated to replace the null values

Basement Dimensions

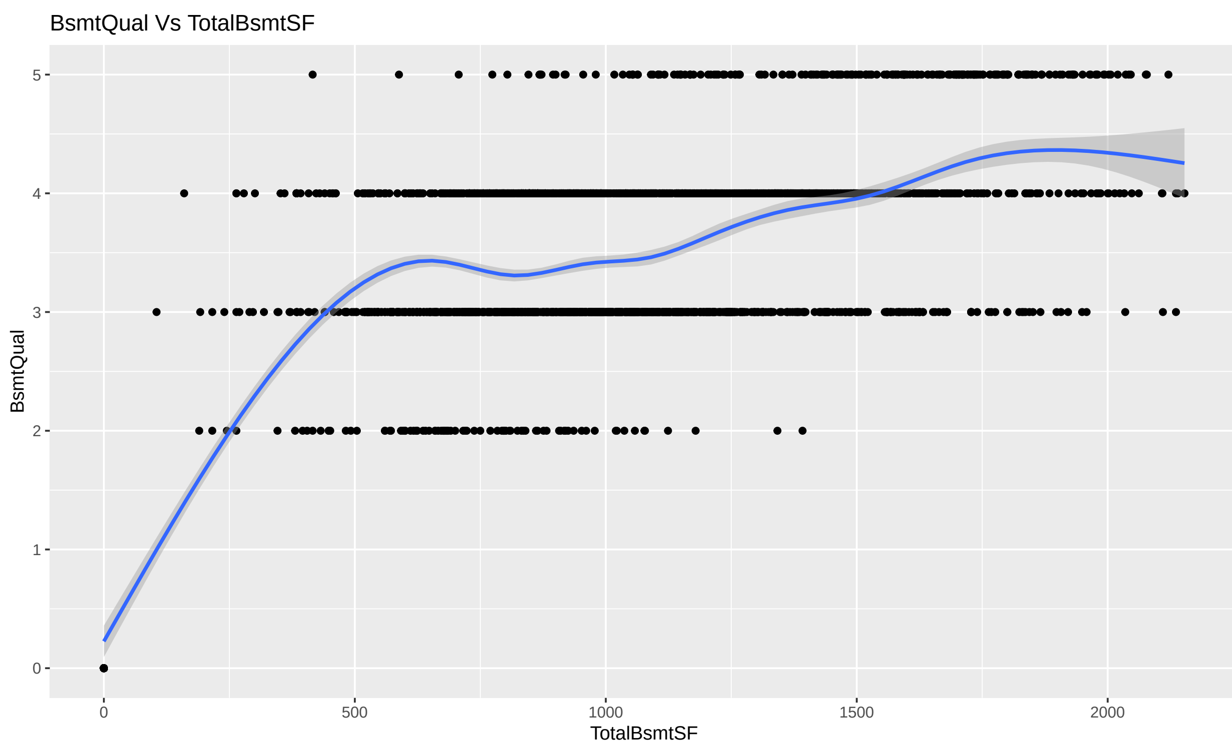


High Correlated Dim			
	Var1	Var2	value
1	BsmtFinType2	BsmtFinSF2	0.8288252
3	BsmtFinType1	BsmtFinSF1	0.7122094
5	BsmtFullBath	BsmtFinSF1	0.6394350
7	BsmtQual	BsmtCond	0.6344277
9	BsmtFullBath	BsmtFinType1	0.5877612
11	TotalBsmtSF	BsmtQual	0.5788890
13	TotalBsmtSF	BsmtFinSF1	0.5361229

To complete the null values for BsmtQual im making alinear models() BsmtQual ~ TotalBsmtSF), first calculate z socre of TotalBsmtSF the remove outliers.



For this model I am going to remove all the TotalBsmtSF that are at least 2.5 z score absolute from the average, this represents 98.8694758 percent of the data.



Linear Regression BsmtQual ~ TotalBsmtSF

```
MSE: 0.8381899
```

Now im replacing null values with BsmtQual predictions and removing the SE_(Error column) and predic(predictions column), then im making the model for replace null values at BsmtCond.

Bsmt Missing Values		
name	prc_na	type
BsmtCond	0	numerical
BsmtExposure	0	numerical
BsmtFinSF1	0	numerical
BsmtFinSF2	0	numerical
BsmtFinType1	0	numerical
BsmtFinType2	0	numerical
BsmtFullBath	0	numerical
BsmtHalfBath	0	numerical
BsmtQual	0	numerical
BsmtUnfSF	0	numerical
TotalBsmtSF	0	numerical

```
bsmt_cor <- cor_SalesPrice(df,names(df[grepl('Bsmt',names(df))]))
bsmt_cor<-high_cor_cols(bsmt_cor)
cols_to_Drop <- c(cols_to_Drop,names(df[,bsmt_cols&!names(df)%in% bsmt_cor$Var2]) )
bsmt_cor%>% kable() %>%
  kable_material(c("striped"))%>%
  kable_minimal()%>%
  add_header_above(c("Bsmt Columns to Keep"=3))
```

Bsmt Columns to Keep		
Var1	Var2	value
SalePrice	TotalBsmtSF	0.6135806
SalePrice	BsmtQual	0.5852072

Fireplace Fireplaces: Number of fireplaces

FireplaceQu: Fireplace quality

```
Ex   Excellent - Exceptional Masonry Fireplace
Gd   Good - Masonry Fireplace in main level
TA   Average - Prefabricated Fireplace in main living area or Masonry Fireplace in basement
Fa   Fair - Prefabricated Fireplace in basement
Po   Poor - Ben Franklin Stove
NA   No Fireplace
```

Im using similar approach as Pool NA's, if Fireplaces =0 the FireplaceQu = "No_Fireplace"

Fireplace Missing Values		
name	prc_na	type
FireplaceQu	0	numerical
Fireplaces	0	numerical

Garage GarageType: Garage location

```
2Types   More than one type of garage
Attchd   Attached to home
Basment  Basement Garage
BuiltIn  Built-In (Garage part of house - typically has room above garage)
CarPort  Car Port
Detchd   Detached from home
NA   No Garage
```

GarageYrBlt: Year garage was built

GarageFinish: Interior finish of the garage

```
Fin  Finished -> 3
RFn  Rough Finished ->2
Unf  Unfinished -> 1
NA   No Garage -> 0
```

GarageCars: Size of garage in car capacity

GarageArea: Size of garage in square feet

GarageQual: Garage quality

```
Ex   Excellent ->5
Gd   Good -> 4
TA   Typical/Average ->3
Fa   Fair -> 2
Po   Poor ->1
NA   No Garage ->0
```

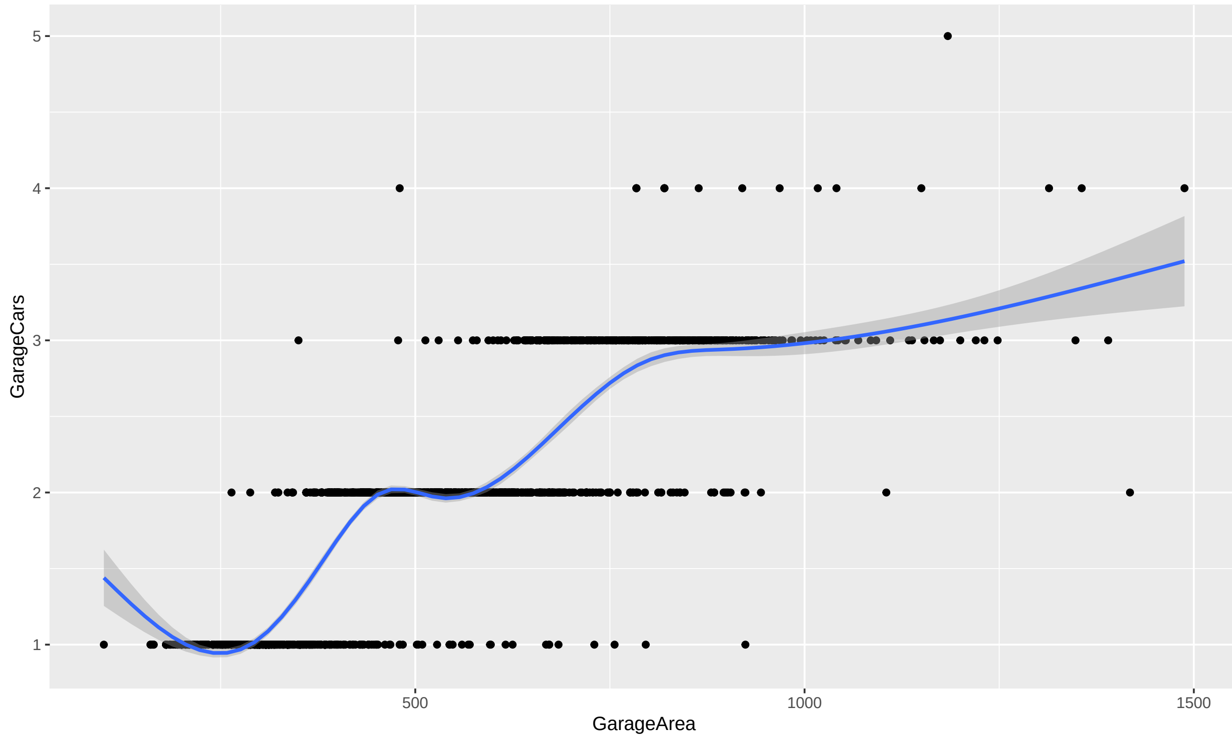
GarageCond: Garage condition

```
Ex    Excellent ->5
 Gd   Good -> 4
 TA   Typical/Average ->3
 Fa   Fair -> 2
 Po   Poor ->1
 NA   No Garage ->0
```

First im looking the correlation between numerical columns and ploting by GarageType.

Garage (Numerical Columns Corr)

	Var1	Var2	value
1	GarageCars	GarageArea	0.8454547
3	GarageYrBlt	GarageCars	0.5877121
5	GarageYrBlt	GarageArea	0.5558361



There clearly a postive correlation between Area and number of cars, im replaceing nan values in Garage to 0, when Garage Area is equal to 0, im replaced every Garage categorical column to "No_Garage" and GarageYrBlt to 0

Garage Missing Values Escenario	
	2127
GarageArea	360
GarageCars	1
GarageCond	NA
GarageFinish	NA
GarageQual	NA
GarageType	Detchd
GarageYrBlt	NA

By looking to the Garage columns after the first cleaning try, this table show us there still one row with NA values, im going to replace numerical column to 0 when more than 50% of the row has null values, and repet the same logic as before, replace every categorical to "No_Garage" and 0 in every numerical column where Area is equal to 0.

After replacing null Values i replaced quality and condition columns with numerical data using QC_Con_decoder.

```
garage_fn <- c("No_Garage"=0,"Unf"=1,"RFn"=2,"Fin"=3)
```

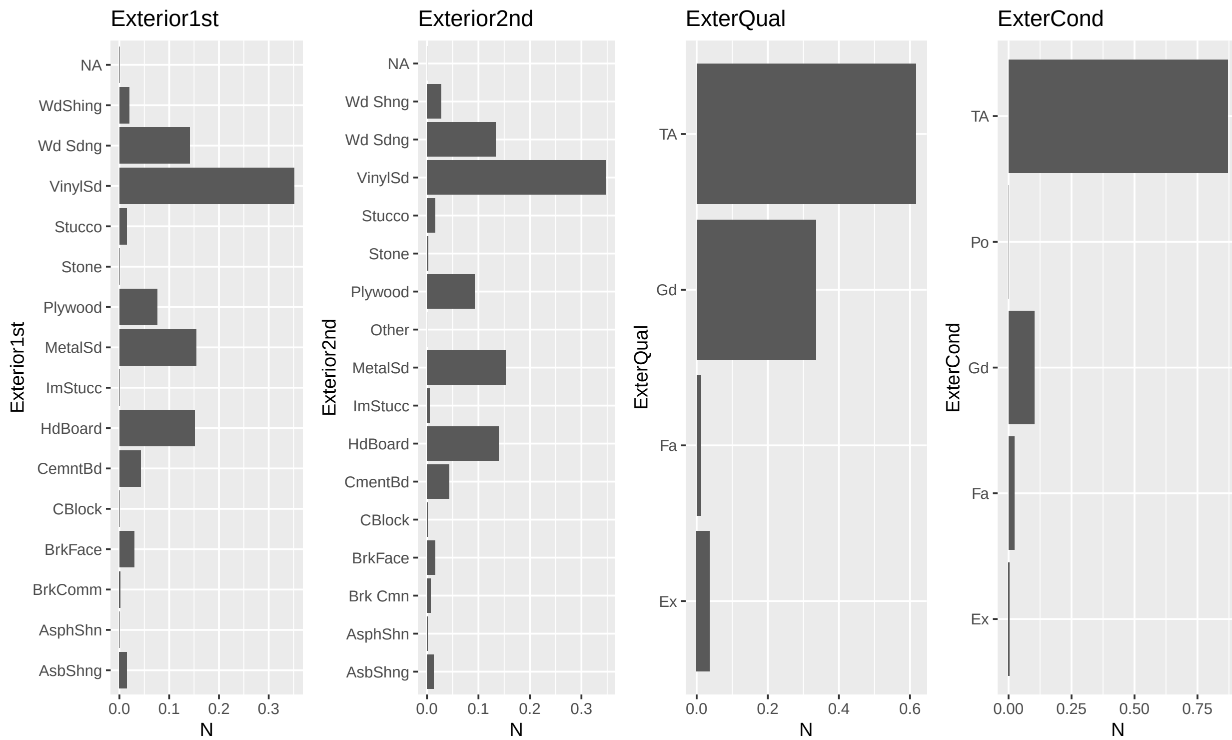
Garage Missing Values		
name	prc_na	type
GarageArea	0	numerical
GarageCars	0	numerical
GarageCond	0	numerical
GarageFinish	0	numerical
GarageQual	0	numerical
GarageType	0	categorical
GarageYrBlt	0	numerical

Now im transforming GarageType to a Dummy Varibale. Dummy variables (or binary variables) are commonly used in statistical analyses and in more simple descriptive statistics. A dummy column is one which has a value of one when a categorical event occurs and a zero when it doesn't occur. In most cases this is a feature of the event/person/object being described.

```
#
#First join the feautre name with column value.
#The used spread to transpose the cols values as new binary columns (1,0)
df<-df %>%
  mutate(v = 1, GarageType=gsub(" ","_",paste0("Garage_",GarageType))) %>%
    spread(GarageType, v, fill = 0)
```

Exteriors

```
p1 <- df %>%
  group_by(Exterior1st) %>%
  dplyr::summarise(N=n()/nrow(df)) %>% ggplot(aes(x= Exterior1st,y=N)) +
  geom_bar(stat="identity") +
  labs(title="Exterior1st")+ coord_flip()
p2 <- df %>%
  group_by(Exterior2nd) %>%
  dplyr::summarise(N=n()/nrow(df)) %>% ggplot(aes(x= Exterior2nd,y=N)) +
  geom_bar(stat="identity") +
  labs(title="Exterior2nd")+ coord_flip()
p3 <- df %>%
  group_by(ExterQual) %>%
  dplyr::summarise(N=n()/nrow(df)) %>% ggplot(aes(x= ExterQual,y=N)) +
  geom_bar(stat="identity") +
  labs(title="ExterQual")+ coord_flip()
p4 <- df %>%
  group_by(ExterCond ) %>%
  dplyr::summarise(N=n()/nrow(df)) %>% ggplot(aes(x= ExterCond ,y=N)) +
  geom_bar(stat="identity") +
  labs(title="ExterCond")+ coord_flip()
grid.arrange(p1,p2,p3,p4,ncol=4)
```



It seems that both columns have the same values except for a few values that we can assume are misspelled, I am going to replace the data in the Exterior2nd column with the Exterior1st column WdShing, CemntBd, BrkComm

```
replace_exterior2 <- c("Wd Shng"="WdShing", "CmentBd"="CemntBd" ,"Brk Cmn"="BrkComm")
#replaceing NA with mode
df <- df%>% mutate(Exterior1st=ifelse(is.na(Exterior1st),mode(Exterior1st),Exterior1st),Exterior2nd=ife
  mutate(Exterior2nd=revalue(Exterior2nd,replace_exterior2))
#unique values between Exterior1st and Exterior2nd
exteriors<- unique(c(unique(df$Exterior1st),unique(df$Exterior2nd)))
```

There is a 90.8187736% where Exterior1st and Exterior2nd are same. Exterior Materials = VinylSd, MetalSd, Wd Sdng, HdBoard, BrkFace, WdShing, CemntBd, Plywood, AsbShng, Stucco, BrkComm, AsphShn, Stone, ImStucc, CBlock, Other

```
missing_values(df[,grepl("Ext",names(df))]) %>% kable() %>%
  kable_material(c("striped"))%>%
  kable_minimal()%>%
  add_header_above(c("Exterior Missing Values"=3))
```

Exterior Missing Values		
name	prc_na	type
ExterCond	0	categorical
Exterior1st	0	categorical
Exterior2nd	0	categorical
ExterQual	0	categorical

EXterior Categorical values 1st im going to transform ExterQual & ExterCond to numeric

```
evaluate_cond_qc <- c("Po"=1,"Fa"=2,"TA"=3,"Gd"=4,"Ex"=5)
df["ExterCond"] <- as.numeric(revalue(df$ExterCond,evaluate_cond_qc))
df["ExterQual"] <- as.numeric(revalue(df$ExterQual,evaluate_cond_qc))
```

Making exterior materials as dummy variable, we going to make a loop trought unique values between Exterior1st and Exterior2nd columns

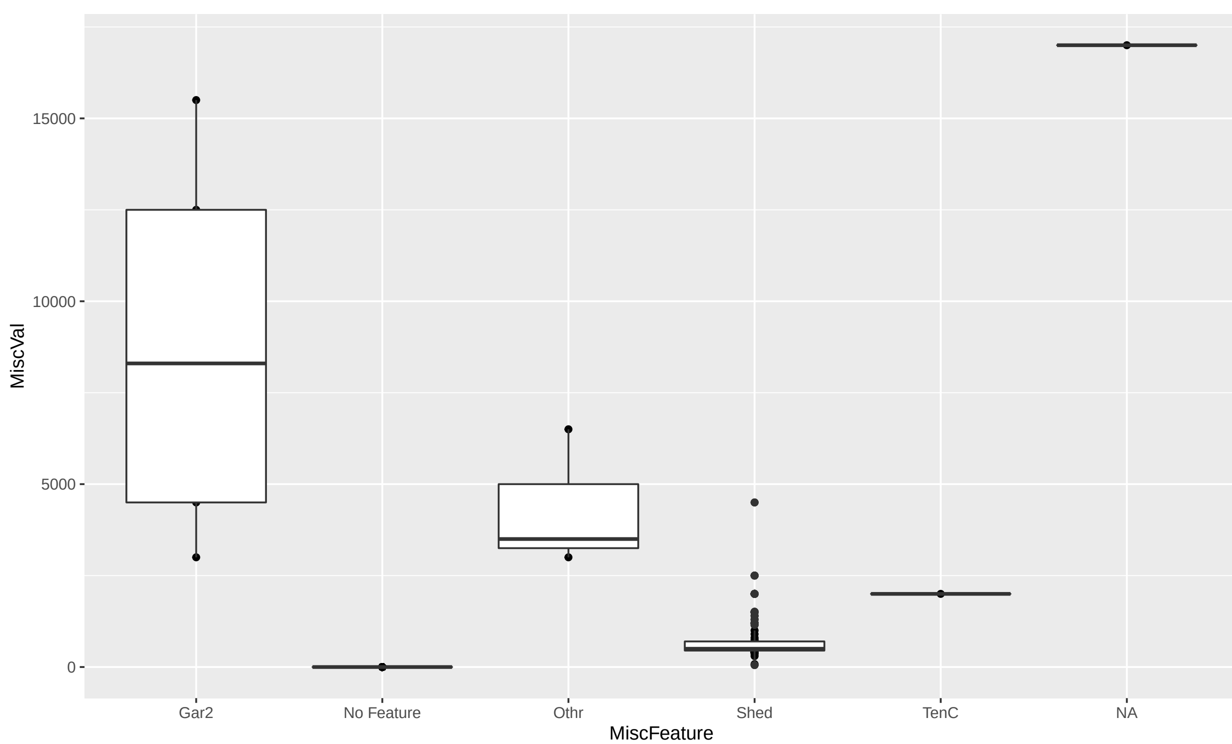
```
for (ex in exteriors){
  name_ = gsub(" ","_",sprintf("Exterior_Matertial_%s",ex))
  df[,name_] <- as.numeric(0)
  df[df$Exterior1st==ex,name_]<-1
  df[df$Exterior1st!=ex,name_]<-0

}
df <- df%>% select(-Exterior1st,-Exterior2nd)
```

MiscFeature MiscFeature: Miscellaneous feature not covered in other categories

```
Elev Elevator
Gar2 2nd Garage (if not described in garage section)
Othr Other
Shed Shed (over 100 SF)
TenC Tennis Court
NA   None
```

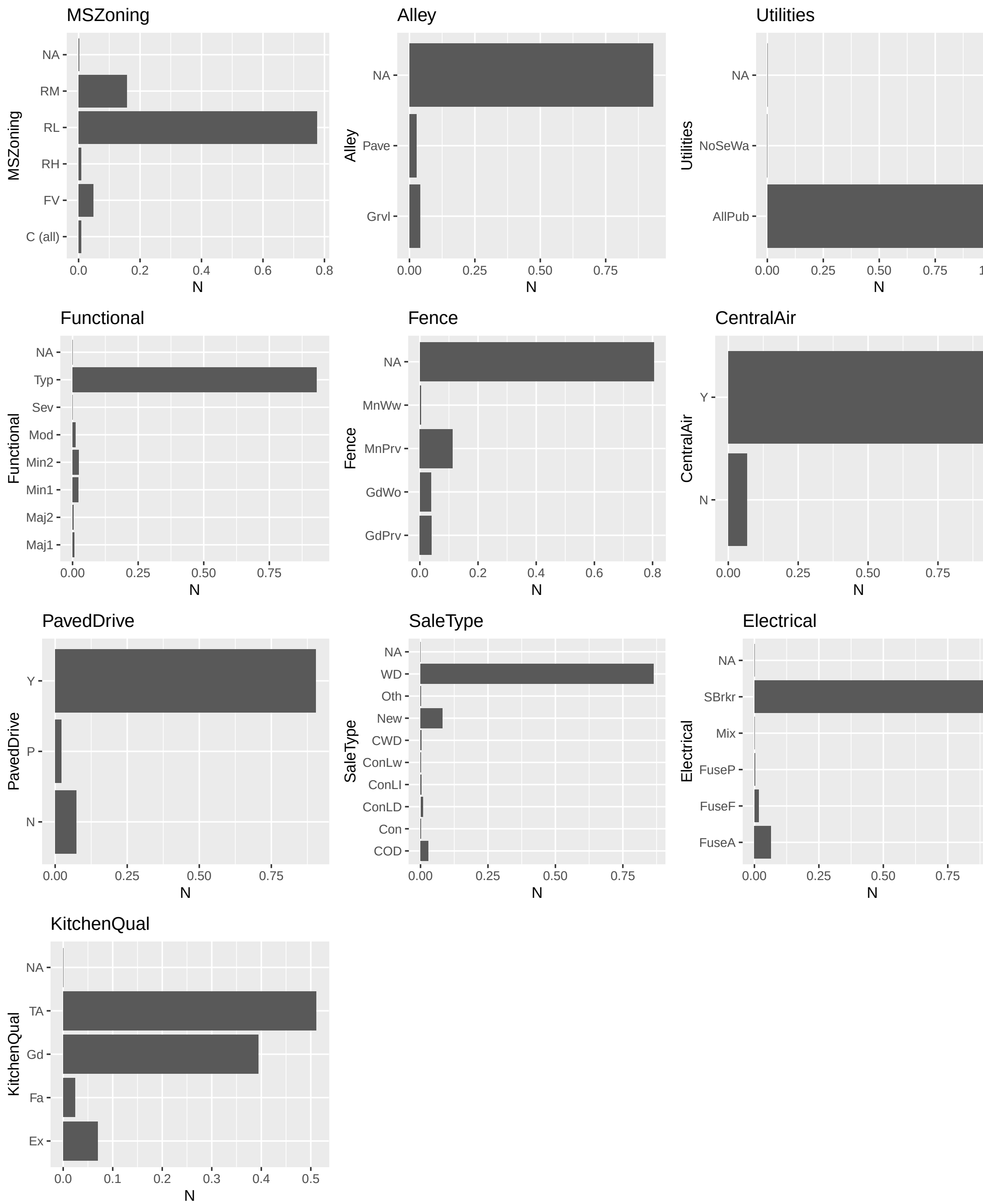
MiscVal: $Value of miscellaneous feature



In this case im using fill function from tidyverse fill(MiscFeature,.direction = "downup"), im sorting by descreasing Miscval and fill down miscfeature, by setting .direction to "downup" if the previous values is NULL it going to fill with the next one.

```
df <- df %>% arrange(-MiscVal) %>% fill(MiscFeature,.direction = "downup")
missing_values(df[,grepl("Misc",names(df))]) %>% kable() %>%
  kable_material(c("striped"))%>%
  kable_minimal()%>%
  add_header_above(c("Misc Missing Values"=3))
```

Misc Missing Values		
name	prc_na	type
MiscFeature	0	categorical
MiscVal	0	numerical



Other Columns

```
#Fence Replace NA with No Fence and Alley with no Alley
df <- df %>% mutate(Fence=ifelse(is.na(Fence),"No Fence",Fence),
                    Alley=ifelse(is.na(Alley),"No Alley",Alley))
#Fence Replace NA with No Fence
df <- df %>% mutate(Fence=ifelse(is.na(Fence),"No Fence",Fence))
```

```r
#Replace NA with mode
df$Functional <- ifelse(is.na(df$Functional),mode(df$Functional),df$Functional)
df$Utilities <- ifelse(is.na(df$Utilities),mode(df$Utilities),df$Utilities)
df$MSZoning <- ifelse(is.na(df$MSZoning),mode(df$MSZoning),df$MSZoning)
df$KitchenQual <- ifelse(is.na(df$KitchenQual),mode(df$KitchenQual),df$KitchenQual)
df$SaleType <- ifelse(is.na(df$SaleType),mode(df$SaleType),df$SaleType)
df$Electrical <- ifelse(is.na(df$Electrical),mode(df$Electrical),df$Electrical)


privacy_levels <- c("No Fence"=0,"MnWw"=1,"GdWo"=2,"MnPrv"=3,"GdPrv"=4)
#Encode Label
df$Fence= as.numeric(revalue(df$Fence,privacy_levels))

evaluate_cond_qc <- c("Po"=1,"Fa"=2,"TA"=3,"Gd"=4,"Ex"=5)
df["KitchenQual"] <- as.numeric(revalue(df$KitchenQual,evaluate_cond_qc))
df["HeatingQC"] <- as.numeric(revalue(df$HeatingQC,evaluate_cond_qc))
#CentraL Air(Y/N)
df$CentralAir <- as.numeric(ifelse(df$CentralAir=="Y",1,0))

#PAvedDrive Encoding
df$PavedDrive <- as.numeric(revalue(df$PavedDrive,c("N"=0,"P"=1,"Y"=2)))

missing_values(df)[1:4,]%>% kable() %>%
  kable_material(c("striped"))%>%
  kable_minimal()%>%
  add_header_above(c("Missing Values"=3))
```

Missing Values		
name	prc_na	type
SalePrice	0.4998287	numerical
Alley	0.0000000	categorical
BedroomAbvGr	0.0000000	numerical
BldgType	0.0000000	categorical

DATA CLEANING DONE!!!!!

Adding New Columns

Infrastructure

I will create a column that totals the bathrooms in the house, another that says if it has a second floor.

```r
df$TotalBathRooms <- df$BsmtFullBath +  (df$BsmtHalfBath * 0.5) + df$FullBath + (df$HalfBath * 0.5)
df$Second_Floor<- ifelse(df$X2ndFlrSF>0,1,0) #0=No Second Floor, 1= Second Floor
```

Age

For purposes of age I will use the year of remodeling against the year of sale, and I will also create a variable that says if it was remodeled or not. In general, the entire infrastructure is not renewed

```r
df$Remod <- ifelse(df$YearBuilt==df$YearRemodAdd, 0, 1) #0=No Remodeling, 1=Remodeling
df$Age <- as.numeric(df$YrSold)-df$YearRemodAdd
```

```r
df$New <- ifelse(df$YrSold==df$YearBuilt, 1, 0)  #0=No, 1= Yes


cols_to_Drop <- c(cols_to_Drop,"Yrsold","MoSold","YearBuilt","Id")
```

PreProcessing SalesPrice, Squeare Feet & Area Columns

```r
sf_area<- setdiff(names(df[,grep("SF|Area|SalePrice",names(df))]), cols_to_Drop)
t(head((df[,sf_area]),10))
```

```
##                   1      2     3      4     5     6    7     8    9   10
## GarageArea     1154    301   312    600   495   626  624     0  286  462
## GrLivArea      5095    958  1329   2620  1312  1823  960  1092  930 1731
## LotArea       39290  12772 14267  18890 11355 12192 9750  5600 9520 9370
## LowQualFinSF      0      0     0      0     0     0    0     0    0    0
## MasVnrArea     1224      0   108      1   125     0    0     0  115    0
## OpenPorchSF     484      0    36     24   304    36    0     0    0   85
## PoolArea          0      0     0      0     0     0    0     0    0    0
## SalePrice        NA 151500    NA 190000    NA    NA   NA 55000   NA   NA
## TotalBsmtSF    5095    958  1329   1361  1312   928  960     0  911  836
## WoodDeckSF      546      0   393    155     0   192    0     0  134  307
## X1stFlrSF      5095    958  1329   1361  1312   928  960   372  930  844
## X2ndFlrSF         0      0     0   1259     0   895    0   720    0  887
```

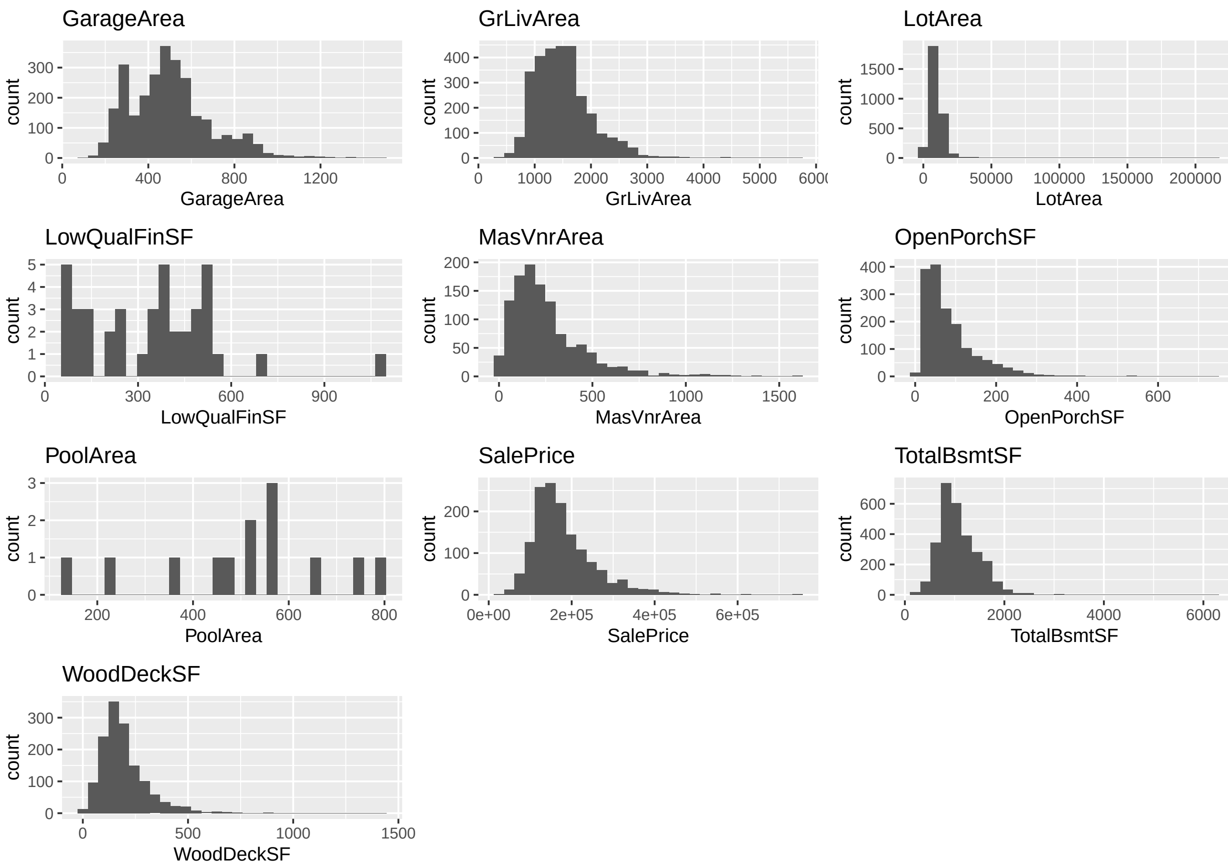
By Looking in the top 10 row of the dataset, it seems that GrLivArea is the sum of X1stFlrSF and X2ndFlrSF 100% of the time this happens, i going to append X1stFlrSF and X2ndFlrSF to our list of columnst to drop and add a new columns that show if the house have a second floord.

```r
cols_to_Drop <- c(cols_to_Drop,"X1stFlrSF", "X2ndFlrSF")
df<- df[, !names(df)%in%cols_to_Drop]
```

```r
sf_area<- setdiff(names(df[,grep("SF|Area|SalePrice",names(df))]), cols_to_Drop)

graph_dist<- function(df,feature){

  p <-  df %>% filter(!!as.name(feature)>0)%>% ggplot(aes(x= !!as.name(feature))) +
    geom_histogram() +
    labs(title=feature)
  p}
p1<- graph_dist(df,sf_area[1])
p2<- graph_dist(df,sf_area[2])
p3<- graph_dist(df,sf_area[3])
p4<- graph_dist(df,sf_area[4])
p5<- graph_dist(df,sf_area[5])
p6<- graph_dist(df,sf_area[6])
p7<- graph_dist(df,sf_area[7])
p8<- graph_dist(df,sf_area[8])
p9<- graph_dist(df,sf_area[9])
p10<- graph_dist(df,sf_area[10])
p<-arrangeGrob(p1,p2,p3,p4,p5,p6,p7,p8,p9,p10)
grid.arrange(p)
```




```
sf_area_Cor<-cor_SalesPrice(df,sf_area)
sf_area_Cor<-high_cor_cols(sf_area_Cor)
cols_to_Drop <- c(cols_to_Drop,names(df[,!names(df) %in% unique(sf_area_Cor$Var2) & names(df)%in%sf_area
sf_area_Cor%>% kable() %>%
  kable_material(c("striped"))%>%
  kable_minimal()%>%
  add_header_above(c("SF & Area Columns to Keep"=3))
```

SF & Area Columns to Keep		
Var1	Var2	value
SalePrice	GrLivArea	0.7086245
SalePrice	GarageArea	0.6234314
SalePrice	TotalBsmtSF	0.6135806
SalePrice	MasVnrArea	0.4726145

To support the histograms in this section, I will calculate the skewness and kurtosis, if the absolute value of the skewness is greater than one we will perform a logarithmic transformation

```
sf_area
```

```
## [1] "GarageArea"   "GrLivArea"    "LotArea"      "LowQualFinSF" "MasVnrArea"
## [6] "OpenPorchSF"  "PoolArea"     "SalePrice"    "TotalBsmtSF"  "WoodDeckSF"
```

```r
feature <- NULL
skew <- NULL
kurt_<-NULL
mean_ <- NULL
median_ <- NULL
n<-0
for(col in sf_area){
  cat(col)
  df_ = df
  if(col=="SalePrice"){
    df_ = df %>% filter(!is.na(SalePrice))
  }
  n=1+n
  feature[n]<- col
  skew[n]<- skewness(df_[,col])
  kurt_[n]<- kurtosis(df_[,col])
  mean_[n]<-mean(df_[,col])
  median_[n]<-median(df_[,col])
}
```

```
## GarageAreaGrLivAreaLotAreaLowQualFinSFMasVnrAreaOpenPorchSFPoolAreaSalePriceTotalBsmtSFWoodDeckSF
```

```r
dist_summary <-data.table(feature,skew,kurt_,mean_,median_)
dist_summary %>% kable() %>%
  kable_material(c("striped"))%>%
  kable_minimal()%>%
  add_header_above(c("Missing Values"=5))
```

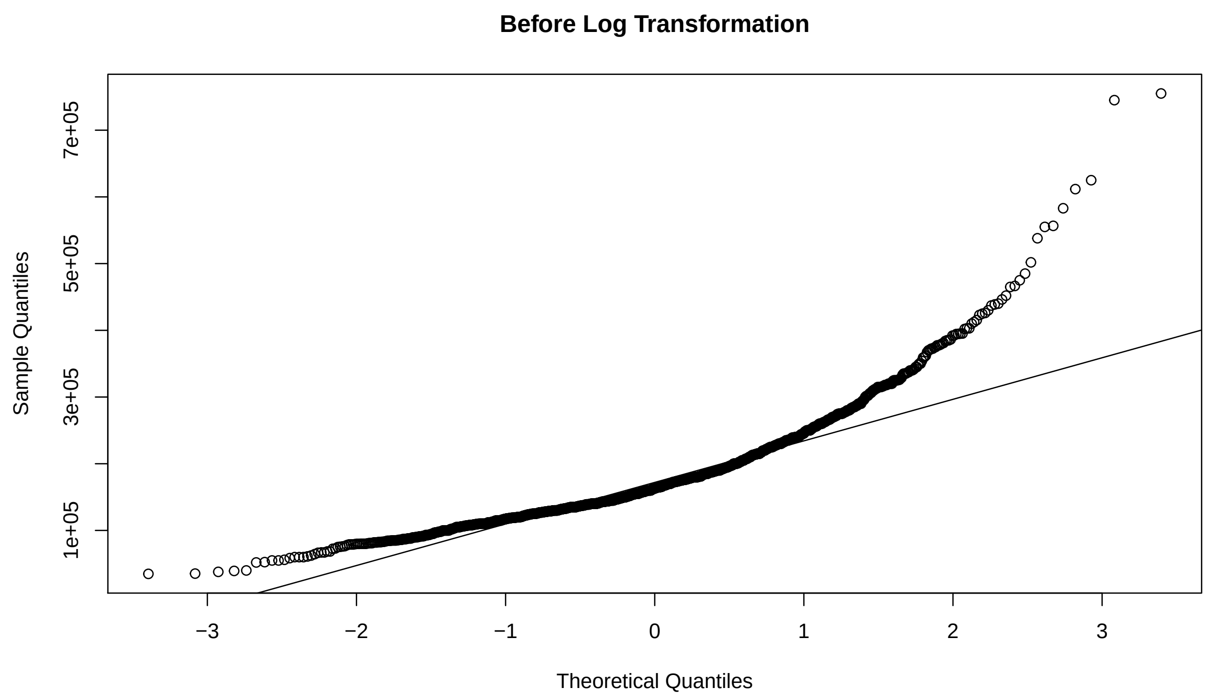
Missing Values				
feature	skew	kurt_	mean_	median_
GarageArea	0.2368571	3.929373	4.725892e+02	480
GrLivArea	1.2693577	7.112492	1.500760e+03	1444
LotArea	12.8224314	267.496632	1.016811e+04	9453
LowQualFinSF	12.0887610	177.631256	4.694416e+00	0
MasVnrArea	2.6135921	12.318376	1.013960e+02	0
OpenPorchSF	2.5351137	13.916572	4.748681e+01	26
PoolArea	16.8983279	301.119801	2.251799e+00	0
SalePrice	1.8809407	9.509812	1.809212e+05	163000
TotalBsmtSF	1.1568941	12.105153	1.051417e+03	989
WoodDeckSF	1.8424328	9.727953	9.370983e+01	0

So, when is the skewness too much?

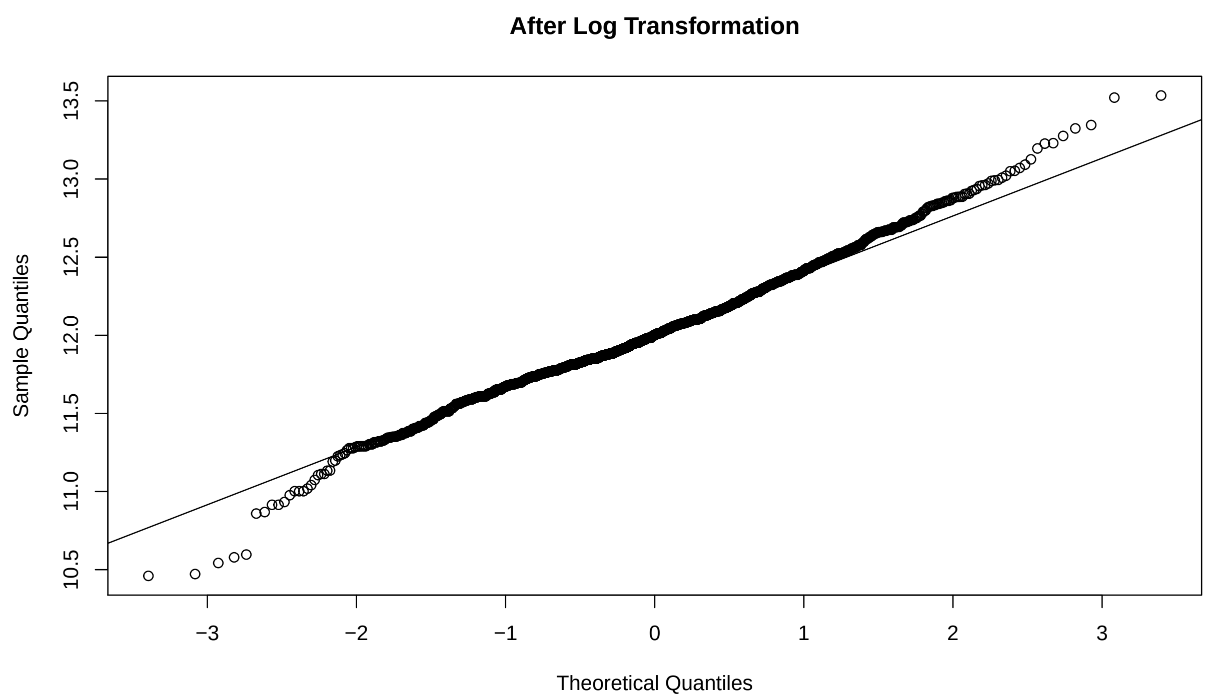
The rule of thumb seems to be:

```
If the skewness is between -0.5 and 0.5, the data are fairly symmetrical.
If the skewness is between -1 and -0.5(negatively skewed) or between 0.5 and 1(positively skewed), the
If the skewness is less than -1(negatively skewed) or greater than 1(positively skewed), the data are h
```

```r
qqnorm(df$SalePrice,main = "Before Log Transformation")
qqline(df$SalePrice)
```




```
qqnorm(log(df$SalePrice),main = "After Log Transformation")
qqline(log(df$SalePrice))
```



Leftover categorical data

```
#Selecting character columns
categorical_columns<- colnames(df %>% select(which(sapply(.,is.character))))
#Number of character columns
n_cat_cols <- length(categorical_columns)
#Display character columns in table

matrix(categorical_columns,9,byrow=TRUE) %>%kable()%>%
  kable_material(c("striped"))%>%
  kable_minimal()
```

Alley	BldgType	Condition1
Condition2	Electrical	Foundation
Functional	Heating	HouseStyle
LandContour	LandSlope	LotConfig
LotShape	MasVnrType	MiscFeature
MSZoning	Neighborhood	RoofMatl
RoofStyle	SaleCondition	SaleType
Street	Utilities	Alley
BldgType	Condition1	Condition2

One-hot encoding is the process of converting a categorical variable with multiple categories into multiple variables, each with a value of 1 or 0.

```
#
#First join the feautre name with column value.
#The used spread to transpose the cols values as new binary columns (1,0)
column_dummy_name <- function(x,colname_){
   gsub(" ","_",paste(colname_,x,sep="_"))

}

for(col in categorical_columns){
  df[col] <-apply(df[col],2,FUN=column_dummy_name,colname_=col)
  df<-df %>% mutate(v = 1) %>%
  spread(!!as.name(col), v, fill = 0)}
colnames(df)<- sapply(colnames(df),function(X){gsub("\\(","",X)})
colnames(df)<- sapply(colnames(df),function(X){gsub("\\)","",X)})
colnames(df)<- sapply(colnames(df),function(X){gsub("\\&","",X)})
```

Model

Spliting the Data set

Data set was already divided into training and test, the training file contains an additional column that would be the sale price, while the test set does not have this column, for the purposes of this exercise we combine both data sets to avoid staying with an unknown value at the time of cleaning and pre-processing.

For validate the training model I have splited it into 25% for testing and 75% for training

```r
pre_train_set<- df[!is.na(df$SalePrice),]
indx  <- createDataPartition(y = pre_train_set$SalePrice, times = 1, p = 0.25, list = FALSE)
train_set<- pre_train_set[-indx,]
X_train<- train_set[,names(train_set)!="SalePrice"]
Y_train <- train_set$SalePrice
test_set<- pre_train_set[indx,]
X_test<- test_set[,names(train_set)!="SalePrice"]
Y_test<- test_set$SalePrice
```

1093 367 ## Random Forests

The basic algorithm for a regression random forest can be generalized to the following:

1. Given training data set
2. Select number of trees to build (ntrees)
3. for i = 1 to ntrees do
4. Generate a bootstrap sample of the original data
5. Grow a regression tree to the bootstrapped data
6. for each split do
7. | Select m variables at random from all p variables
8. | Pick the best variable/split-point among the m
9. | Split the node into two child nodes
10. end
11. Use typical tree model stopping criteria to determine when a tree is complete (but do not prune)
12. end

Advantages & Disadvantages

Advantages:

```
Typically have very good performance
Remarkably good "out-of-the box" - very little tuning required
Built-in validation set - don't need to sacrifice data for extra validation
No pre-processing required
Robust to outliers
```

Disadvantages:

```
Can become slow on large data sets
Although accurate, often cannot compete with advanced boosting algorithms
Less interpretable
```

randomForest also allows us to use a validation set to measure predictive accuracy if we did not want to use the OOB samples. Here we split our training set further to create a training and validation set. We then supply the validation data in the xtest and ytest arguments.

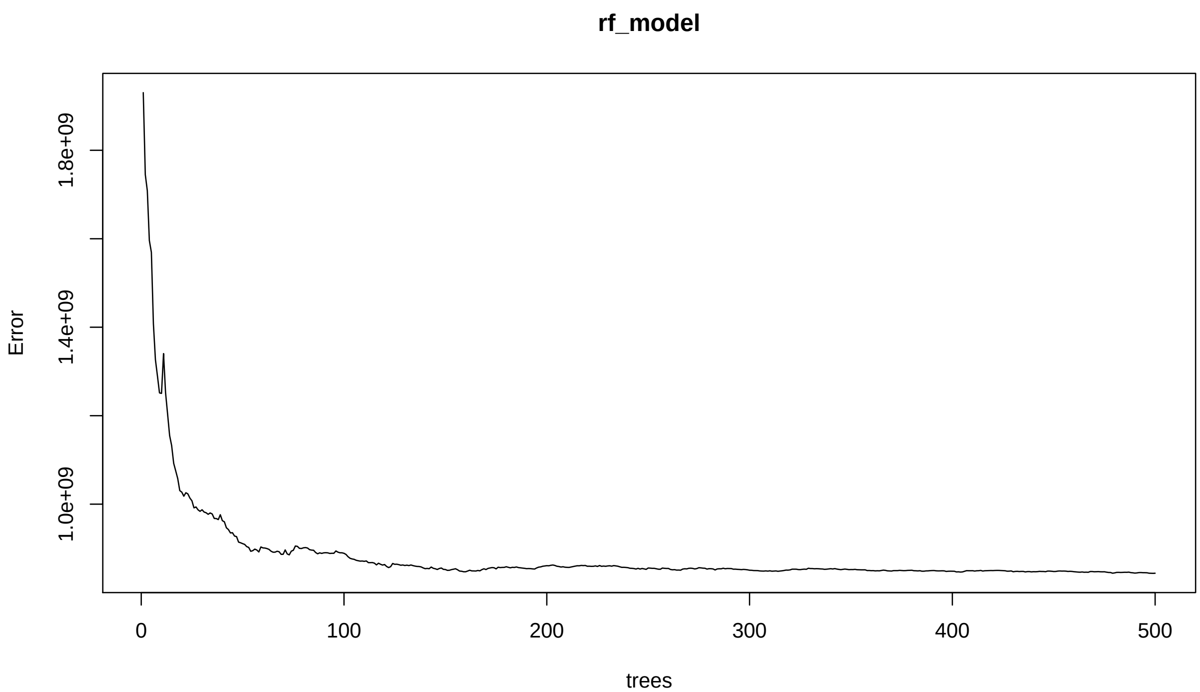
```r
set.seed(250000)

rf_model<-randomForest(
  formula = SalePrice ~ .,
  data    = train_set)

summary(rf_model)
```

```
##                  Length Class  Mode
## call                3   -none- call
## type                1   -none- character
## predicted        1093   -none- numeric
## mse               500   -none- numeric
## rsq               500   -none- numeric
## oob.times        1093   -none- numeric
## importance        214   -none- numeric
## importanceSD        0   -none- NULL
## localImportance     0   -none- NULL
## proximity           0   -none- NULL
## ntree               1   -none- numeric
## mtry                1   -none- numeric
## forest             11   -none- list
## coefs               0   -none- NULL
## y                1093   -none- numeric
## test                0   -none- NULL
## inbag               0   -none- NULL
## terms               3   terms  call
```

```
plot(rf_model)
```



number of trees with lowest MSE 499 RMSE of this optimal random forest: 2.9044694×10^4

```
rfImportance <- rf_model$importance
varsSelected <- length(which(rfImportance!=0))
varsNotSelected <- length(which(rfImportance==0))

cat('Raandome Forest uses', varsSelected, 'variables in its model, and did not select', varsNotSelected
```

```
## Raandome Forest uses 204 variables in its model, and did not select 10 variables.
```

```r
rf_mode_pred <- predict(rf_model,X_test)
original<- Y_test
d = original-rf_mode_pred
mse = mean((d)^2)
mae = mean(abs(d))
rmse = sqrt(mse)
R2 = 1-(sum((d)^2)/sum((original-mean(original))^2))

cat(" MAE:", mae, "\n", "MSE:", mse, "\n",

    "RMSE:", rmse, "\n", "R-squared:", R2)
```

```
##  MAE: 18559.74
##  MSE: 1189236898
##  RMSE: 34485.31
##  R-squared: 0.8518705
```

Lasso Regression

Multiple linear regression is a statistical method that attempts to model the relationship between a continuous variable and two or more independent variables by fitting a linear equation. Three of the limitations that appear in practice when trying to use this type of model (adjusted by ordinary least squares) are:

```
They are adversely affected by the incorporation of correlated predictors.

They do not select predictors, all predictors are incorporated into the model even if they do not provic

They cannot be adjusted when the number of predictors is greater than the number of observations.
```

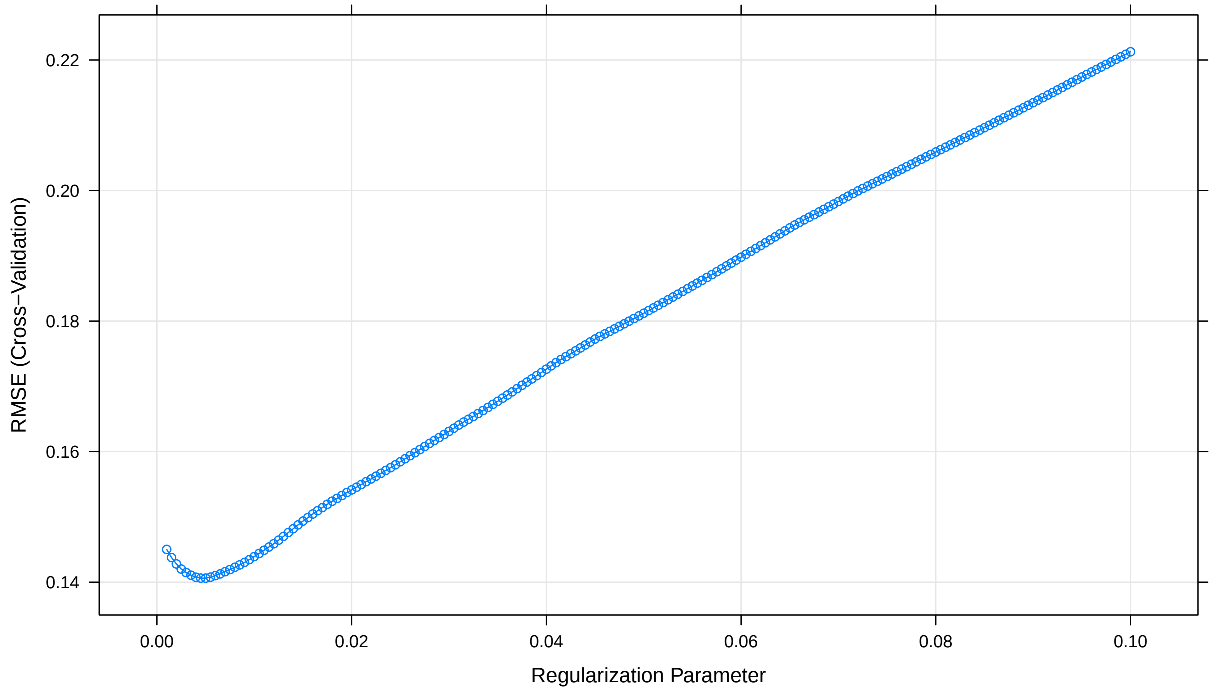
Some of the strategies that can be applied to mitigate the impact of these problems are:

```
Subset selection: use an iterative process that discards the least relevant predictors.

Ridge, Lasso or Elastic Net regularization: these methods force the coefficients of the model to tend to

Dimensionality reduction: they create a reduced number of new predictors (components) from linear or nor
```

```r
set.seed(2500000)
cv <-trainControl(method="cv", number=15)
lasso<- train(x= X_train
              , y= log(Y_train)
              , method='glmnet'
              , trControl= cv
              , tuneGrid= expand.grid(alpha = 1, lambda = seq(0.001,0.1,by = 0.0005)))
plot(lasso)
```




```
t(lasso$bestTune %>% select(alpha,lambda))%>%kable() %>%
  kable_styling(font_size = 10) %>%
  add_header_above(c("Parameters"=2))
```

Parameters	
	9
alpha	1.000
lambda	0.005

```
## Lasso uses 75 variables in its model, and did not select 139 variables.
```

```
##  MAE: 15865.23
##  MSE: 620986308
##  RMSE: 24919.6
##  R-squared: 0.9226509
```

Conclusion

I would have liked to have had more time to study more algorithms, after several days cleaning data and trying to understand what would be the best way to present it, when I sit down to read about the different algorithms that exist, I realize that with xgboost or randome forest regressor could have more easily analyzed the 81 variables presented by this project. Both algorithms are very powerful.