A Data Analytic Case study using R

Data-set : AMES HOUSE PRICES

Data-set Info :: The Dataset contains information from the Ames Assessor's Office used in computing assessed values for residential(housing) properties sold in Ames(2006 to 2010).

Contributers of the Project

B. Kiran

Rukmani Sahu

GCS Living Stone

B. Hemant Reddy

Under the Guidance of :Dr. Shom Prasad Das, Prof. N. Sangita Acharya

Contents:

1. Problem Statement : Prediction of house prices of Ames based on the details of a residential properties of Ames

Explaination : The data set contains the details of residential attributes/properties of the house price of ames collected through the Ames Assessor's Office containing the information about the houses sold in from year 2006 to year 2010. Given the mentioned dataset we have to predict the prices of the houses.

Glimpse of Data-set :

1. Problem Type :: Regression Problem(Multiple Regression)

2. Number of Instances :: 1460

3. Number of attributes ::

- **Input Variables = 80**
- **Output Variable = 1**

4. Attributes type :: Multivariate type of variables

- **Number of Nominal Categorical variable = 23**
- **Number of Ordinal Categorical variable = 23**
- **Number of Discrete value variable = 14**
- **Number of Continuous value variable = 20**

5. Missing value :: There are presence of missing value

Data sets various attributes

A brief descriptive sight of what we would find in the data set.

- **SalePrice - the property's sale price in dollars. This is the target variable that you're trying to predict.**
- **MSSubClass: The building class**
- **MSZoning: The general zoning classification**
- **LotFrontage: Linear feet of street connected to property**
- **LotArea: Lot size in square feet**
- **Street: Type of road access**
- **Alley: Type of alley access**
- **LotShape: General shape of property**
- **LandContour: Flatness of the property**
- **Utilities: Type of utilities available**
- **LotConfig: Lot configuration**
- **LandSlope: Slope of property**
- **Neighborhood: Physical locations within Ames city limits**
- **Condition1: Proximity to main road or railroad**
- **Condition2: Proximity to main road or railroad (if a second is present)**
- **BldgType: Type of dwelling**
- **HouseStyle: Style of dwelling**
- **OverallQual: Overall material and finish quality**
- **OverallCond: Overall condition rating**
- **YearBuilt: Original construction date**
- **YearRemodAdd: Remodel date**
- **RoofStyle: Type of roof**
- **RoofMatl: Roof material**
- **Exterior1st: Exterior covering on house**
- **Exterior2nd: Exterior covering on house (if more than one material)**
- **MasVnrType: Masonry veneer type**
- **MasVnrArea: Masonry veneer area in square feet**
- **ExterQual: Exterior material quality**
- **ExterCond: Present condition of the material on the exterior**
- **Foundation: Type of foundation**
- **BsmtQual: Height of the basement**
- **BsmtCond: General condition of the basement**
- **BsmtExposure: Walkout or garden level basement walls**
- **BsmtFinType1: Quality of basement finished area**
- **BsmtFinSF1: Type 1 finished square feet**
- **BsmtFinType2: Quality of second finished area (if present)**
- **BsmtFinSF2: Type 2 finished square feet**
- **BsmtUnfSF: Unfinished square feet of basement area**
- **TotalBsmtSF: Total square feet of basement area**
- **Heating: Type of heating**
- **HeatingQC: Heating quality and condition**
- **CentralAir: Central air conditioning**
- **Electrical: Electrical system**
- **1stFlrSF: First Floor square feet**
- **2ndFlrSF: 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**
- **Bedroom: Number of bedrooms above basement level**
- **Kitchen: Number of kitchens**

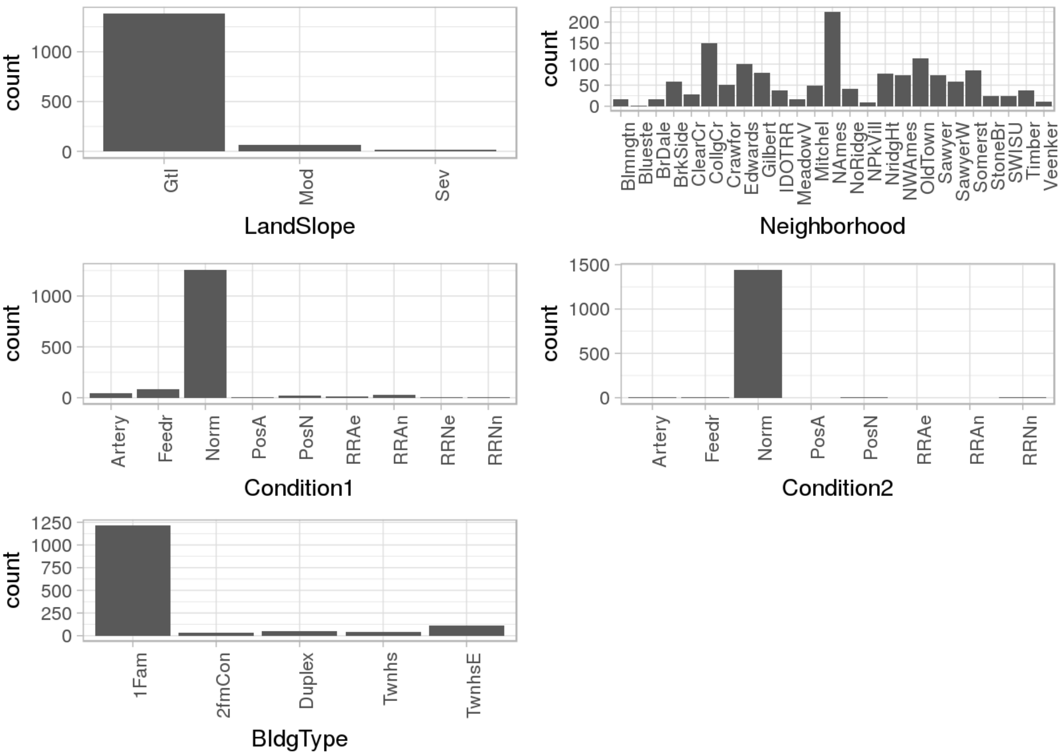
In [1]:
```
train_df = read.csv('train.csv',sep=",")
```

In [6]:
```
head(train_df)
```

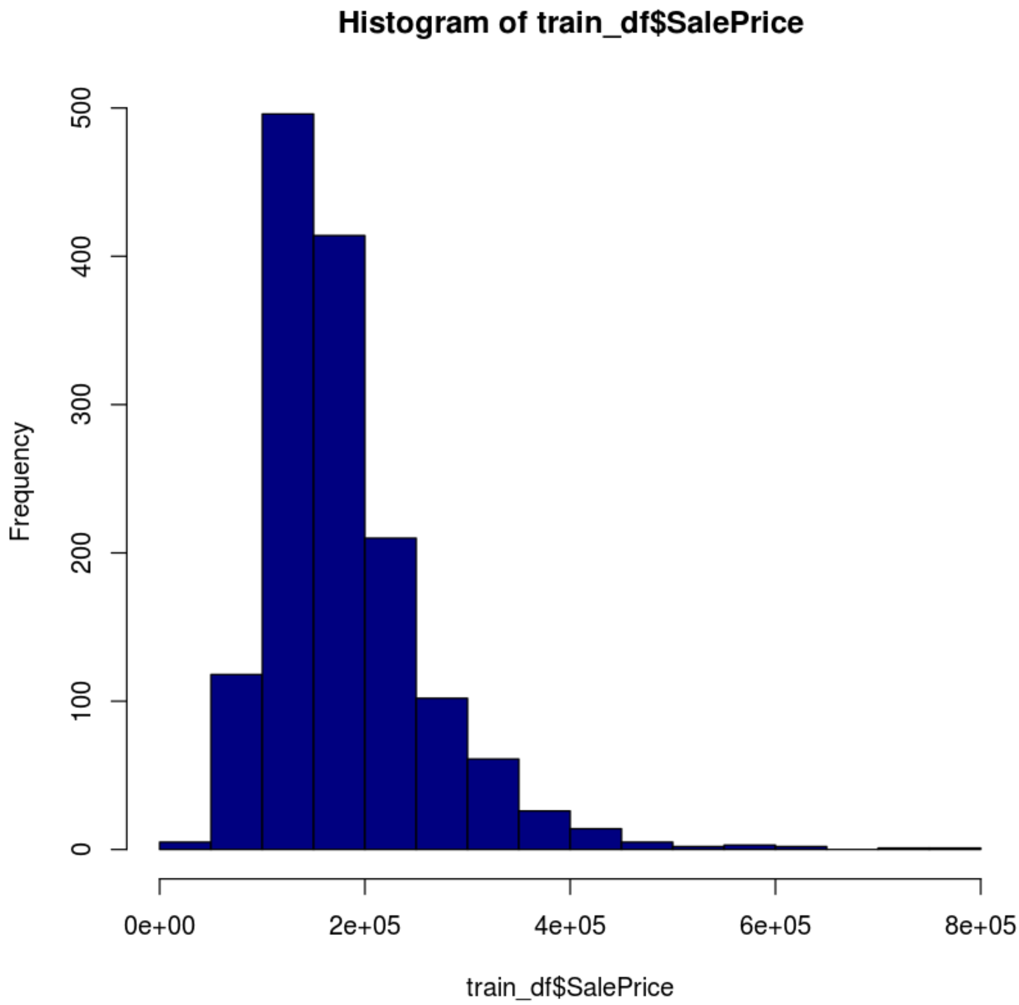
Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities
1	60	RL	65	8450	Pave	NA	Reg	Lvl	AllPub
2	20	RL	80	9600	Pave	NA	Reg	Lvl	AllPub
3	60	RL	68	11250	Pave	NA	IR1	Lvl	AllPub
4	70	RL	60	9550	Pave	NA	IR1	Lvl	AllPub
5	60	RL	84	14260	Pave	NA	IR1	Lvl	AllPub
6	50	RL	85	14115	Pave	NA	IR1	Lvl	AllPub

Frequency vs Sales range Plot

A few basic plots



In [2]:
```
hist(train_df$SalePrice,col = "navyblue",xlim=c(0,800000))
```



Applications in the real world::

1. Property value prediction

2. Real Estate market

- Predicting/Forecasting in buisness domains(Eg:Real Estate)

3. Corelational analysis(Eg: b)

4. Economical Status of a country's property

5. Optimization of resources

And so on

---

2. Data Exploration

Here we load data into the the data frame and then explore the aspects of dataset

In [1]:
```
#Reading the training data into data_

data_ = read.csv('train.csv',sep=',')
```

In [2]:
```
#data_ dimensions i.e No. of instances and no. of attributes

dim(data_)
```

1460 81

In [3]:
```
#viewing some data from the starting rows

head(data_,4)
```

Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities
1	60	RL	65	8450	Pave	NA	Reg	Lvl	AllPub
2	20	RL	80	9600	Pave	NA	Reg	Lvl	AllPub
3	60	RL	68	11250	Pave	NA	IR1	Lvl	AllPub
4	70	RL	60	9550	Pave	NA	IR1	Lvl	AllPub

2.1. Variable Identification

In [4]:
```
#Print column Names
cat("The column Names are:\n");
#print(colnames(data_));
cat(colnames(data_),sep=",  ");
```

```
The column Names are:
Id,  MSSubClass,  MSZoning,  LotFrontage,  LotArea,  Street,  Alley,  LotS
hape,  LandContour,  Utilities,  LotConfig,  LandSlope,  Neighborhood,  Co
ndition1,  Condition2,  BldgType,  HouseStyle,  OverallQual,  OverallCond,
YearBuilt,  YearRemodAdd,  RoofStyle,  RoofMatl,  Exterior1st,  Exterior2n
d,  MasVnrType,  MasVnrArea,  ExterQual,  ExterCond,  Foundation,  BsmtQua
l,  BsmtCond,  BsmtExposure,  BsmtFinType1,  BsmtFinSF1,  BsmtFinType2,  B
smtFinSF2,  BsmtUnfSF,  TotalBsmtSF,  Heating,  HeatingQC,  CentralAir,  E
lectrical,  X1stFlrSF,  X2ndFlrSF,  LowQualFinSF,  GrLivArea,  BsmtFullBat
h,  BsmtHalfBath,  FullBath,  HalfBath,  BedroomAbvGr,  KitchenAbvGr,  Kit
chenQual,  TotRmsAbvGrd,  Functional,  Fireplaces,  FireplaceQu,  GarageTy
pe,  GarageYrBlt,  GarageFinish,  GarageCars,  GarageArea,  GarageQual,  G
arageCond,  PavedDrive,  WoodDeckSF,  OpenPorchSF,  EnclosedPorch,  X3SsnP
orch,  ScreenPorch,  PoolArea,  PoolQC,  Fence,  MiscFeature,  MiscVal,  M
oSold,  YrSold,  SaleType,  SaleCondition,  SalePrice
```

In [5]:
```
#Using this function we find the type of the variable

class(data_$Id)
```

'integer'

In [6]:
```
#Data type of variable

sapply(data_,FUN = class)
```

Id	'integer'
MSSubClass	'integer'
MSZoning	'factor'
LotFrontage	'integer'
LotArea	'integer'
Street	'factor'
Alley	'factor'
LotShape	'factor'
LandContour	'factor'
Utilities	'factor'
LotConfig	'factor'
LandSlope	'factor'
Neighborhood	'factor'
Condition1	'factor'
Condition2	'factor'
BldgType	'factor'
HouseStyle	'factor'
OverallQual	'integer'
OverallCond	'integer'
YearBuilt	'integer'
YearRemodAdd	'integer'
RoofStyle	'factor'
RoofMatl	'factor'
Exterior1st	'factor'
Exterior2nd	'factor'
MasVnrType	'factor'
MasVnrArea	'integer'
ExterQual	'factor'
ExterCond	'factor'
Foundation	'factor'
BsmtQual	'factor'
BsmtCond	'factor'
BsmtExposure	'factor'
BsmtFinType1	'factor'
BsmtFinSF1	'integer'
BsmtFinType2	'factor'
BsmtFinSF2	'integer'
BsmtUnfSF	'integer'
TotalBsmtSF	'integer'
Heating	'factor'
HeatingQC	'factor'
CentralAir	'factor'
Electrical	'factor'
X1stFlrSF	'integer'
X2ndFlrSF	'integer'
LowQualFinSF	'integer'
GrLivArea	'integer'
BsmtFullBath	'integer'
BsmtHalfBath	'integer'
FullBath	'integer'
HalfBath	'integer'
BedroomAbvGr	'integer'
KitchenAbvGr	'integer'
KitchenQual	'factor'
TotRmsAbvGrd	'integer'

In [7]:
```
#Knowing Continuous variables:::
data_frame_with_continuous_variable <- data_ [ ,!sapply(data_, is.factor)]
print("Continuous Variable")
sapply(data_frame_with_continuous_variable, class)
```

```
[1] "Continuous Variable"
```

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

In [8]:
```
#Viewing Categorical variables:::
data_frame_with_categorical_variable <- data_ [ ,sapply(data_, is.factor)]
print("Categorical Variable\n")
sapply(data_frame_with_categorical_variable, class)
```

```
[1] "Categorical Variable\n"
```

MSZoning	'factor'
Street	'factor'
Alley	'factor'
LotShape	'factor'
LandContour	'factor'
Utilities	'factor'
LotConfig	'factor'
LandSlope	'factor'
Neighborhood	'factor'
Condition1	'factor'
Condition2	'factor'
BldgType	'factor'
HouseStyle	'factor'
RoofStyle	'factor'
RoofMatl	'factor'

2.2. Univariate Analysis

WITH THE HELP OF PACKAGE - ggplot2

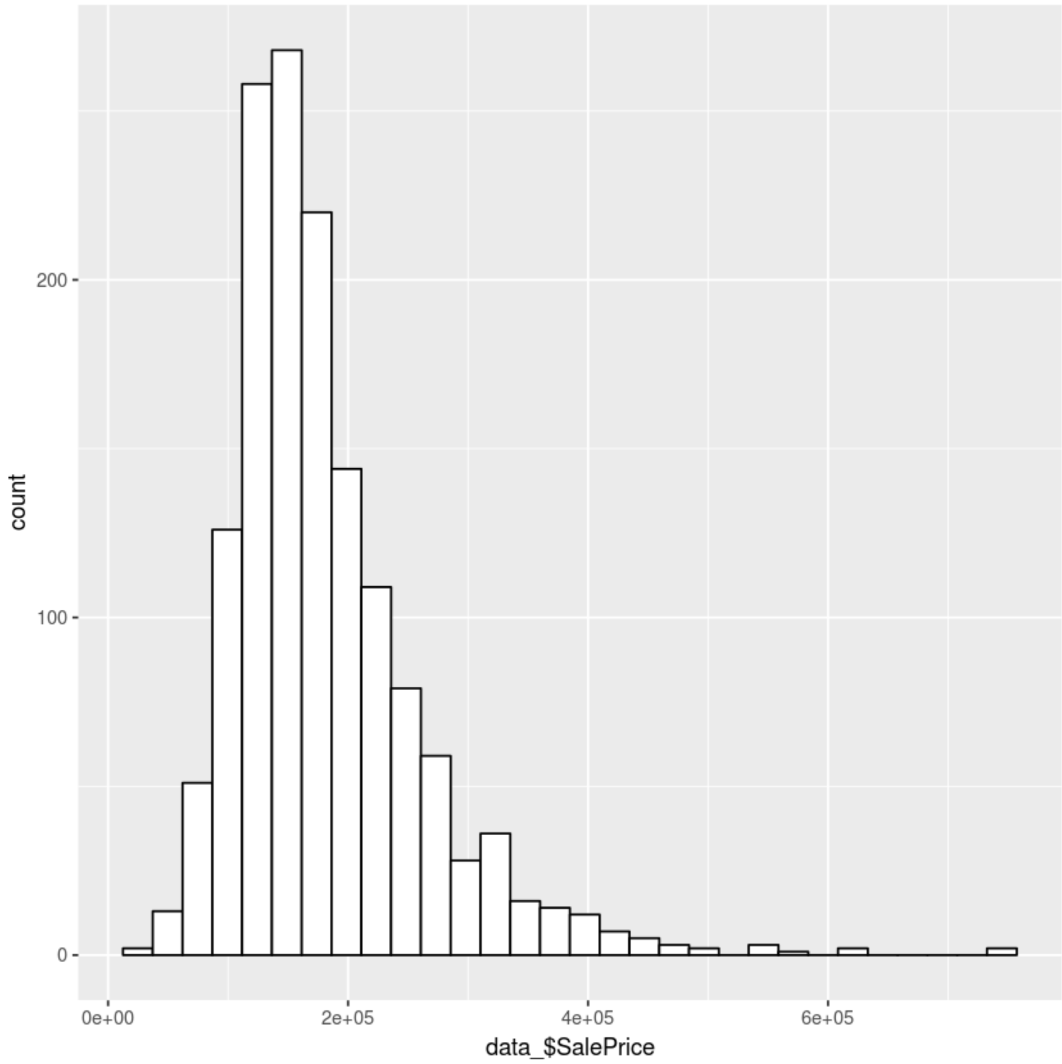
In [9]:
```
#Loading library ggplot2'

library(ggplot2)
```

2.1. Plotting SalePrice Frequency so as to know how the data is distributed

It is observed from the plot that the data set contains more information in the SalePrice range 1e+5 to 2e+5.

In [10]:
```
ggplot(data = data_)+ geom_histogram(mapping = aes(data_$SalePrice), color =
```

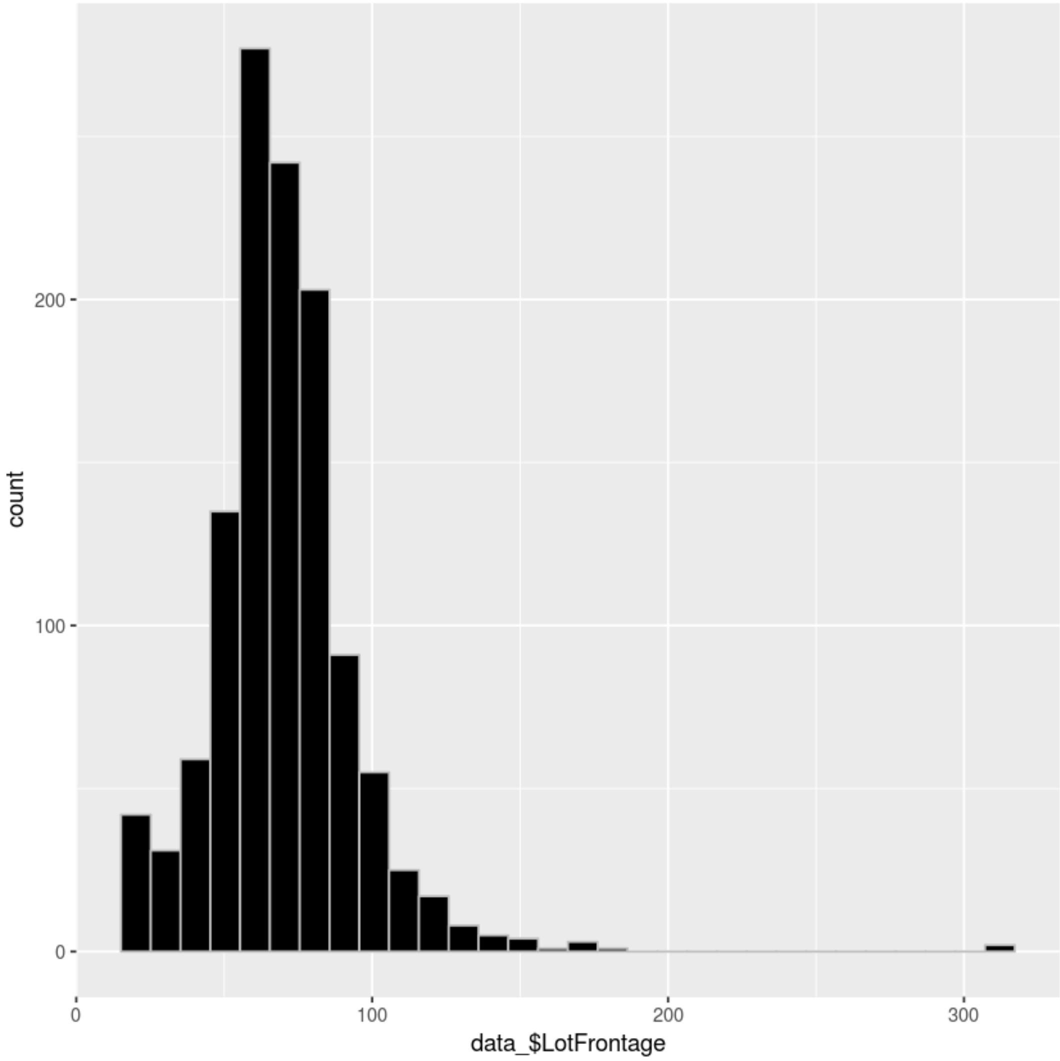
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



2.2. Plotting LotFrontage Frequency so as to know how the data is distributed

In [10]:
```
ggplot(data = data_ ) +  geom_histogram(mapping = aes(data_$LotFrontage),fi
```

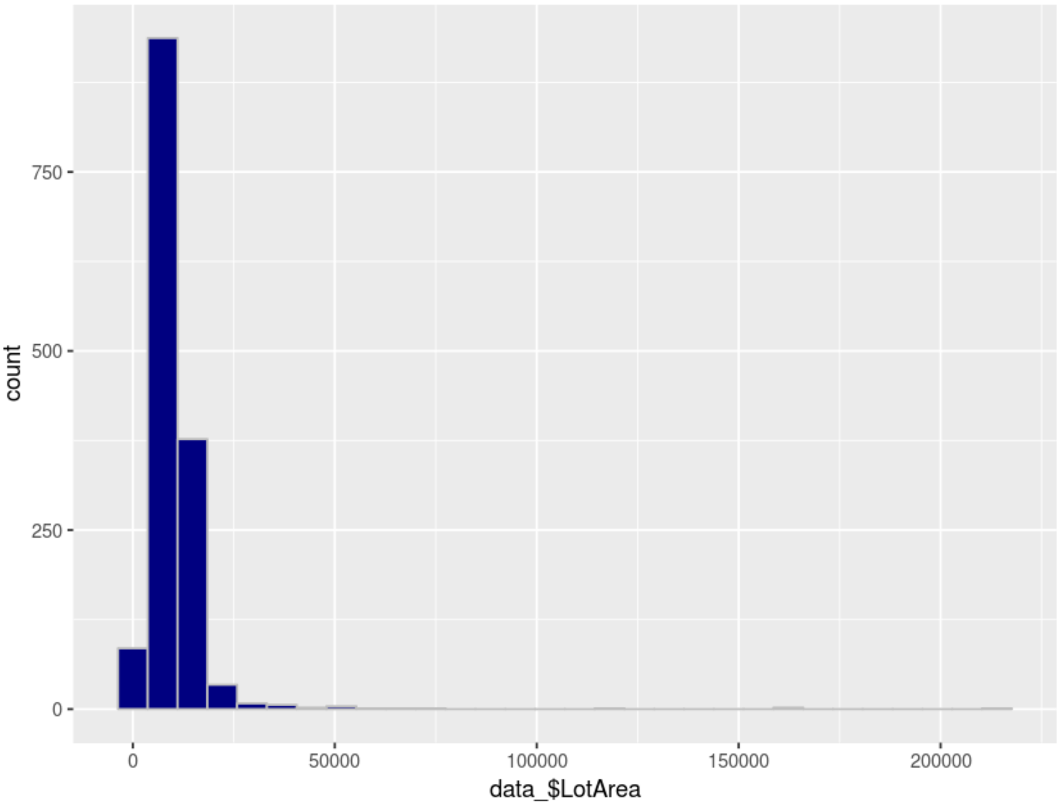
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
Warning message:
“Removed 259 rows containing non-finite values (stat_bin).”
```



2.3. Plotting LotArea Frequency so as to know how the data is distributed

In [11]:
```
ggplot(data = data_ ) +
    geom_histogram(mapping = aes(data_$LotArea),fill="navyblue",color="grey'
        theme(aspect.ratio =.75)
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

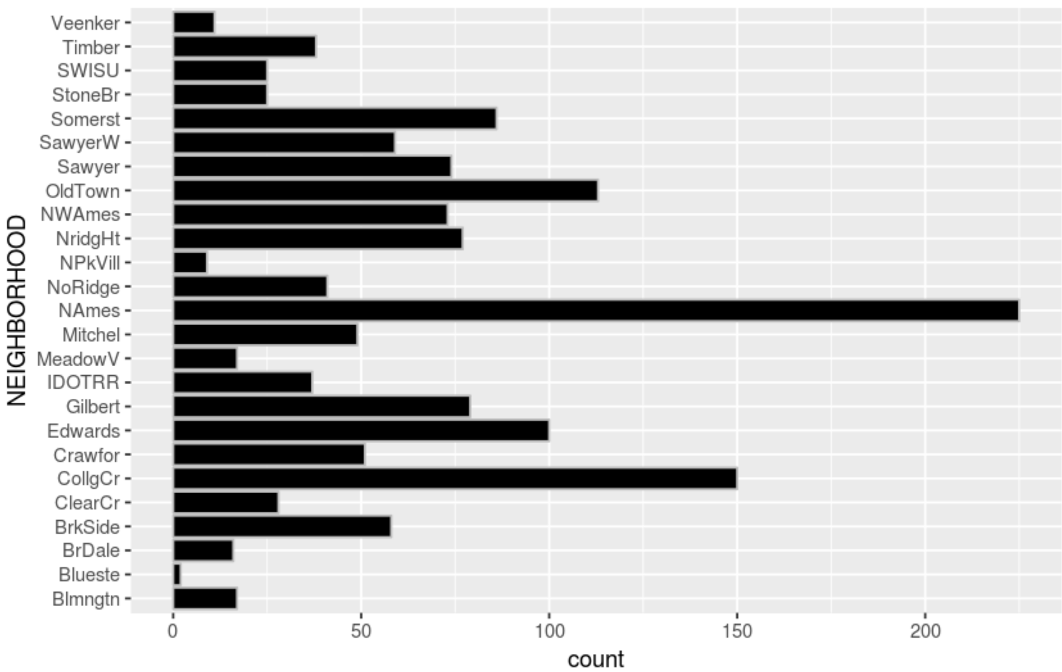


2.4. Plotting Frequency distribution and SalePrice distribution among Neighborhood cities

2.4.1. BAR GRAPH

It is observed from the plot that the data set contains more houses located in the neighborhood of "NAmes" city.

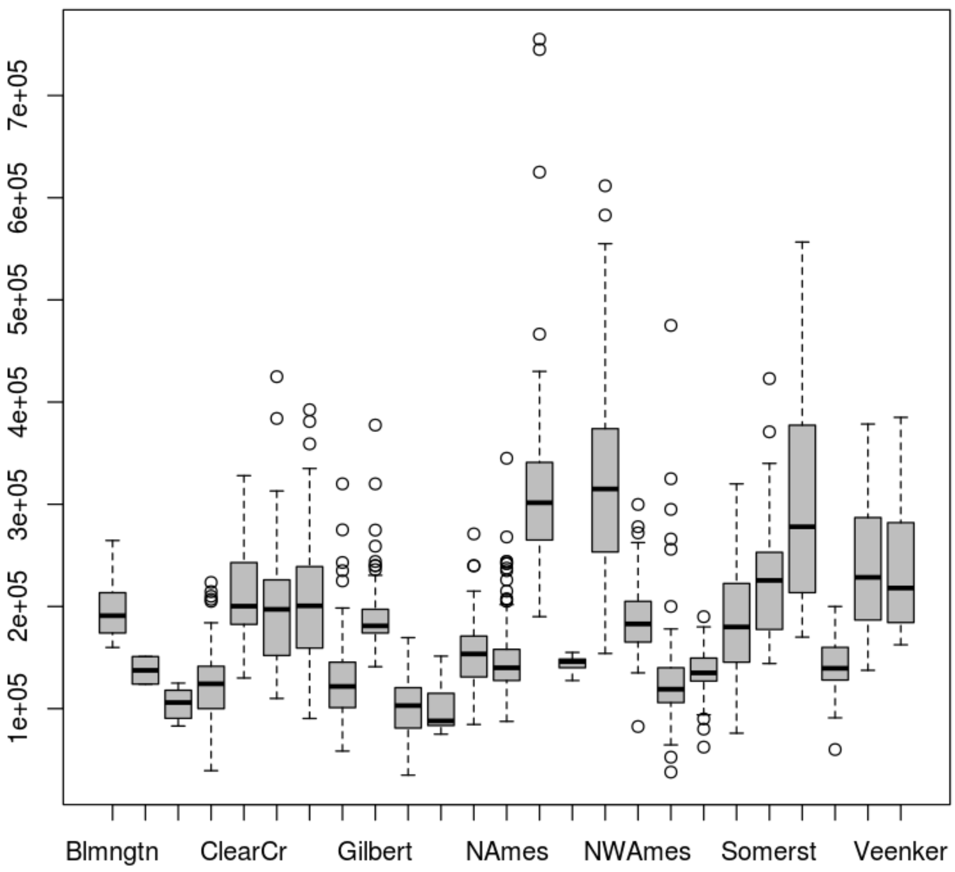
In [12]:
```
ggplot(data = data_) +
    geom_bar(mapping = aes(data_$Neighborhood), fill="black",color="grey")+
        theme(aspect.ratio =.65) +
            labs(x="NEIGHBORHOOD") +
                #theme(axis.text.x = element_text(angle=90, hjust=1)) +
                     coord_flip();
```



2.4.2. BOX PLOT

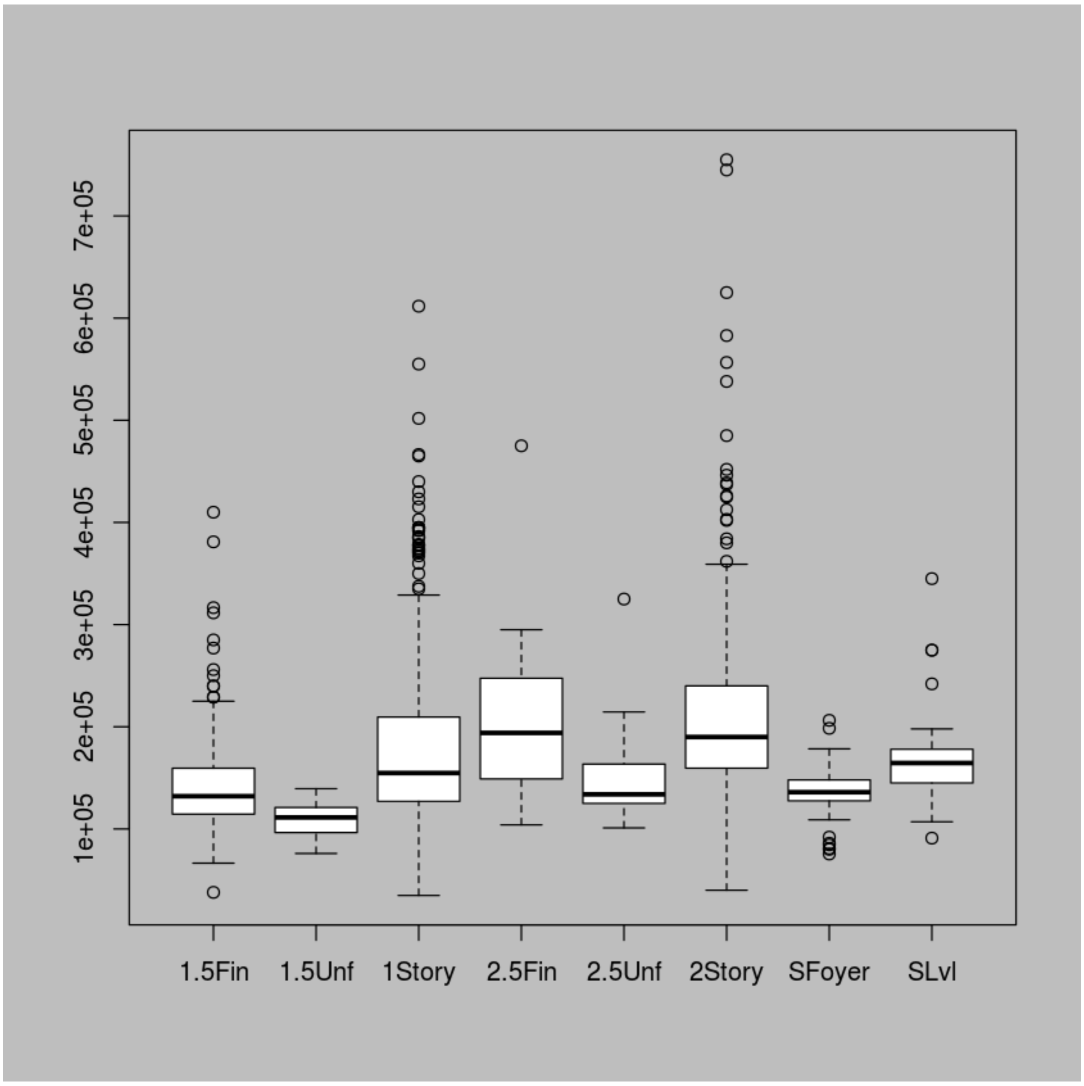
This plot shows the SalePrice range and details of each Neighborhood

In [13]:
```
boxplot(SalePrice~Neighborhood,data = data_,col = "grey",border = "black")
```



Plotting Box plot for HouseStyle so as to know how the data is distributed

In [14]:
```
boxplot( SalePrice~HouseStyle,
        data = data_,
        col = "white",
        border = "black", par(bg = 'grey') )
```



Note: We don't conclude our exploration yet. The rest of the exploration will be done after the missing value imputation.

3. Missing Value

-By using missForest package

This package is selected because it is flexible for imputing missing value of categorical and continuous variable at one go.

This imputes the missing value by predicting the possibility of value

3.1 Checking for missing Values

In [15]:
```
## Loading missForest package for missing value treatment

library(missForest)
```

```
Loading required package: randomForest
randomForest 4.6-12
Type rfNews() to see new features/changes/bug fixes.

Attaching package: 'randomForest'

The following object is masked from 'package:ggplot2':

    margin

Loading required package: foreach
Loading required package: itertools
Loading required package: iterators
```

In [16]:
```
#Checking For the Columns for presence of Missing Values

no_of_missing_values = colMeans(is.na(data_))
names(no_of_missing_values) <- colnames(data_)

#print(no_of_missing_valuesof_missing_values)
sort(no_of_missing_values,decreasing = TRUE)[1:20]
```

PoolQC	0.995205479452055
MiscFeature	0.963013698630137
Alley	0.937671232876712
Fence	0.807534246575342
FireplaceQu	0.472602739726027
LotFrontage	0.177397260273973
GarageType	0.0554794520547945
GarageYrBlt	0.0554794520547945
GarageFinish	0.0554794520547945
GarageQual	0.0554794520547945
GarageCond	0.0554794520547945
BsmtExposure	0.026027397260274
BsmtFinType2	0.026027397260274
BsmtQual	0.0253424657534247
BsmtCond	0.0253424657534247
BsmtFinType1	0.0253424657534247
MasVnrType	0.00547945205479452

3.2. Missing Value Treatment

3.2.1. Treating Missing values of those columns whose no. of missing values is more than 80%

These attributes without missing values might have been useful but since the missing values overempowers the dataset it is better to remove them because imputing large no. of values by considering a little data would be of less significance

In [17]:
```
data_change <- data_[,! colMeans(is.na(data_))>.8 ]
```

In [18]:
```
names(data_change)
```

'Id' 'MSSubClass' 'MSZoning' 'LotFrontage' 'LotArea' 'Street' 'LotShape' 'LandContour' 'Utilities' 'LotConfig' 'LandSlope' 'Neighborhood' 'Condition1' 'Condition2' 'BldgType' 'HouseStyle' 'OverallQual' 'OverallCond' 'YearBuilt' 'YearRemodAdd' 'RoofStyle' 'RoofMatl' 'Exterior1st' 'Exterior2nd' 'MasVnrType' 'MasVnrArea' 'ExterQual' 'ExterCond' 'Foundation' 'BsmtQual' 'BsmtCond' 'BsmtExposure' 'BsmtFinType1' 'BsmtFinSF1' 'BsmtFinType2' 'BsmtFinSF2' 'BsmtUnfSF' 'TotalBsmtSF' 'Heating' 'HeatingQC' 'CentralAir' 'Electrical' 'X1stFlrSF' 'X2ndFlrSF' 'LowQualFinSF' 'GrLivArea' 'BsmtFullBath' 'BsmtHalfBath' 'FullBath' 'HalfBath' 'BedroomAbvGr' 'KitchenAbvGr' 'KitchenQual' 'TotRmsAbvGrd' 'Functional' 'Fireplaces' 'FireplaceQu' 'GarageType' 'GarageYrBlt' 'GarageFinish' 'GarageCars' 'GarageArea' 'GarageQual' 'GarageCond' 'PavedDrive' 'WoodDeckSF' 'OpenPorchSF' 'EnclosedPorch' 'X3SsnPorch' 'ScreenPorch' 'PoolArea' 'MiscVal' 'MoSold' 'YrSold' 'SaleType' 'SaleCondition' 'SalePrice'

3.2.2. Treating Missing Values less than 70%

By Predictive Imputation

We use package "missForest" to impute missing values because our data contains a mixed type of variables

In [19]:
```
#VIEWING THE MISSING VALUE AMOUNT IN DATA SET BEFORE TREATMENT
sort(colMeans(is.na(data_change)),decreasing = TRUE)[1:17]
```

FireplaceQu	0.472602739726027
LotFrontage	0.177397260273973
GarageType	0.0554794520547945
GarageYrBlt	0.0554794520547945
GarageFinish	0.0554794520547945
GarageQual	0.0554794520547945
GarageCond	0.0554794520547945
BsmtExposure	0.026027397260274
BsmtFinType2	0.026027397260274
BsmtQual	0.0253424657534247
BsmtCond	0.0253424657534247
BsmtFinType1	0.0253424657534247
MasVnrType	0.00547945205479452
MasVnrArea	0.00547945205479452
Electrical	0.000684931506849315
Id	0
MSSubClass	0

In [20]:
```
#Viewing the missing data
tail(data_change[,colSums(is.na(data_change))>0])
cat("\n\nFocusing on a particular column::::\n\n")
tail(data_change['FireplaceQu'])
```

	LotFrontage	MasVnrType	MasVnrArea	BsmtQual	BsmtCond	BsmtExposure	BsmtFinType
1455	62	None	0	Gd	TA	No	GLQ
1456	62	None	0	Gd	TA	No	Unf
1457	85	Stone	119	Gd	TA	No	ALQ
1458	66	None	0	TA	Gd	No	GLQ
1459	68	None	0	TA	TA	Mn	GLQ
1460	75	None	0	TA	TA	No	BLQ

```
Focusing on a particular column::::
```

	FireplaceQu
1455	NA
1456	TA
1457	TA
1458	Gd
1459	NA
1460	NA

In [21]:
```
#Imputing the data set using pmm

data_imp <- missForest(data_change, maxiter = 5,ntree = 50)
```

```
  missForest iteration 1 in progress...done!
  missForest iteration 2 in progress...done!
  missForest iteration 3 in progress...done!
```

In [22]:
```
## The error that missForest pmm(Predictive Means Matching) has gone througl

data_imp$OOBerror
```

NRMSE	0.000625541665680956
PFC	0.0503280868151553

In [23]:
```
#Creating a new dataFrame for these furnished data

imputed_data_ <- data_imp$ximp
```

In [24]:
```
#Viewing the missing data

tail(imputed_data_[,colSums(is.na(imputed_data_))>0])

cat("\n\nRevisiting that particular column::::\n\n")
tail(imputed_data_['FireplaceQu'])
```

1455
1456
1457
1458
1459
1460

```
Revisiting that particular column::::
```

	FireplaceQu
1455	Gd
1456	TA
1457	TA
1458	Gd
1459	Gd
1460	Gd

In [25]:
```
colSums( is.na(imputed_data_))
```

Id	0
MSSubClass	0
MSZoning	0
LotFrontage	0
LotArea	0
Street	0
LotShape	0
LandContour	0
Utilities	0
LotConfig	0
LandSlope	0
Neighborhood	0
Condition1	0
Condition2	0
BldgType	0
HouseStyle	0
OverallQual	0

4. Exploration (Part2)

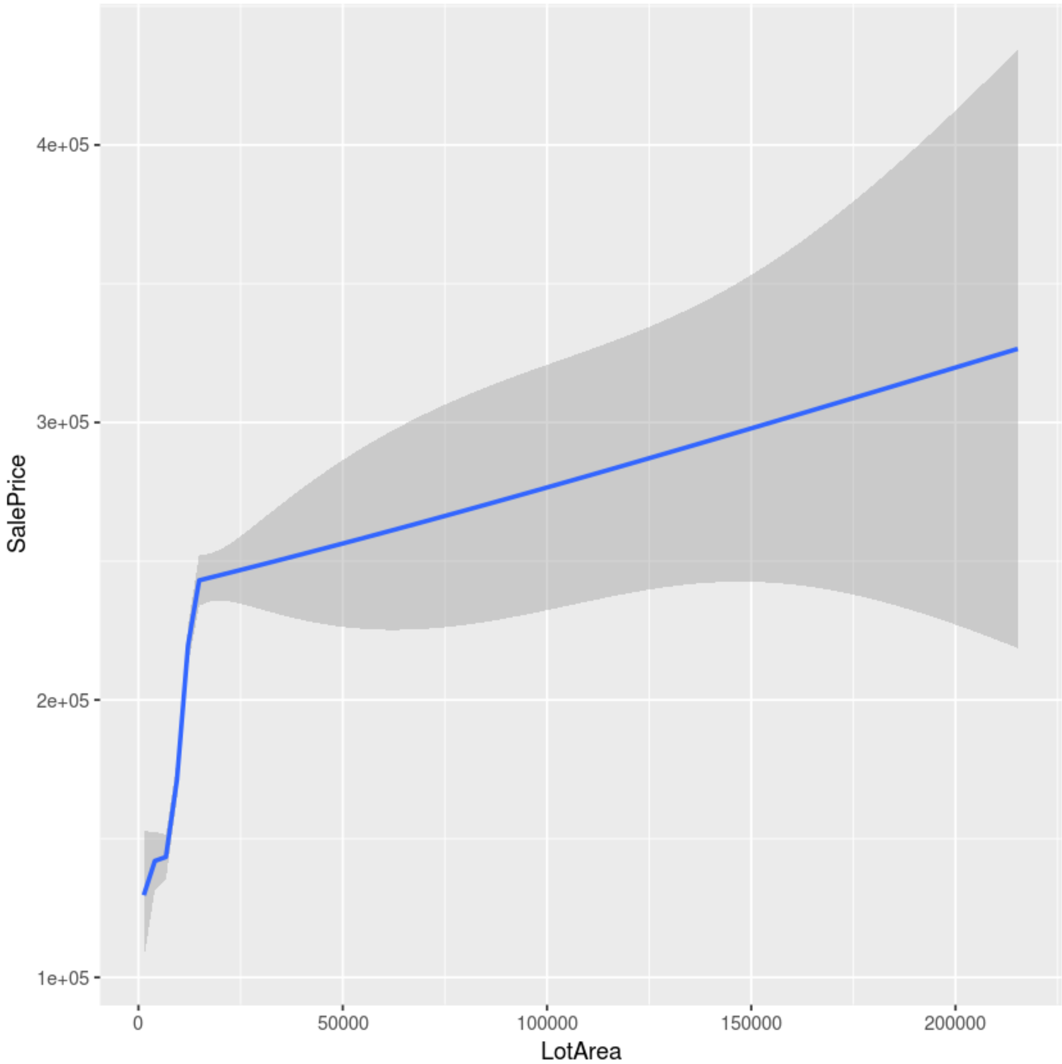
4.1 Bivariate Analysis

Effect of lotarea to Saleprice

Here we could see that SalePrice is linearly increasing with increase in LotArea (after 13000 Sq. Units).
So we infer that SalePrice is linearly proportional to LotArea

In [26]:
```
ggplot(data = data_ ) + geom_smooth( mapping = aes( x=LotArea, y=SalePrice)
```

```
`geom_smooth()` using method = 'gam'
```

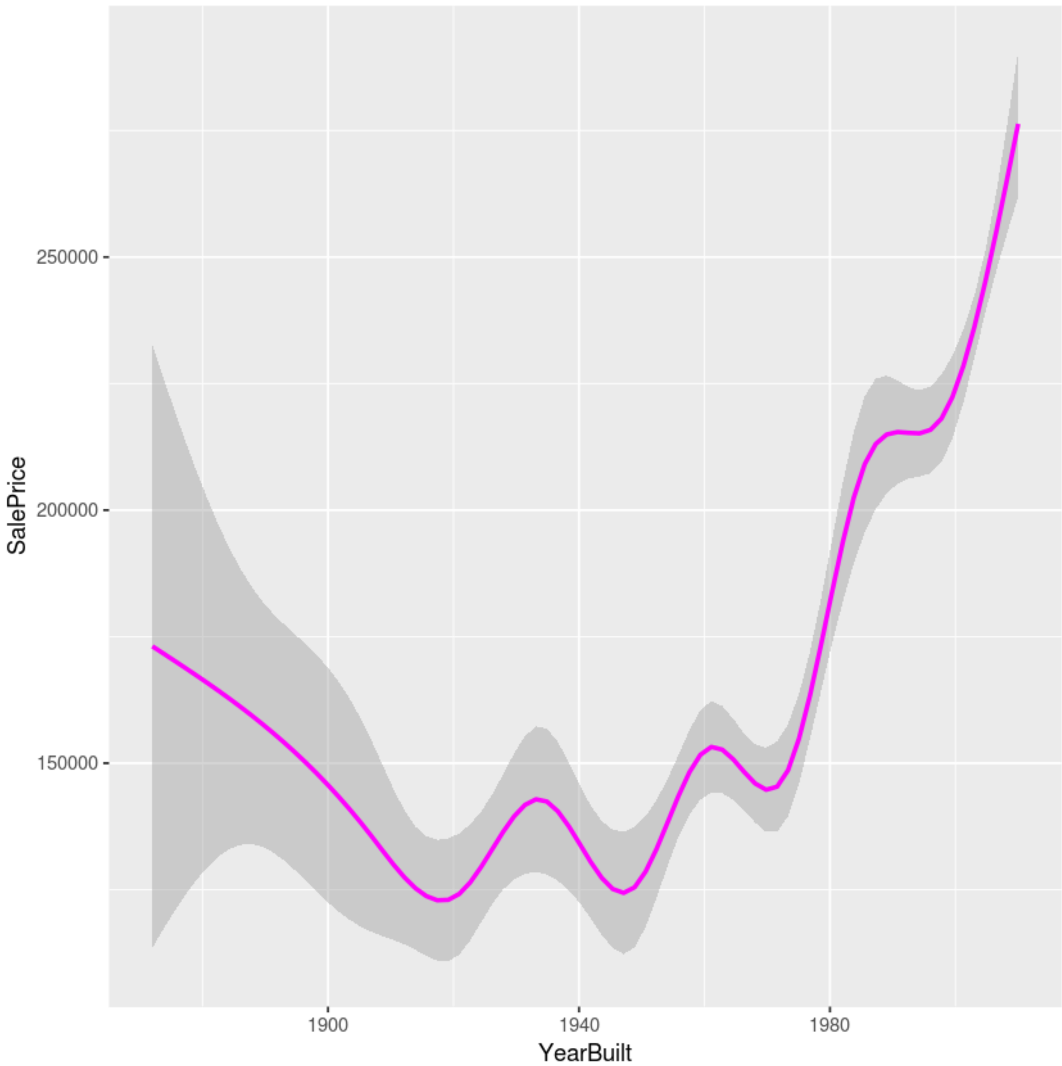


Trace of Sale price over the years

This plot shows that the SalePrice is non-linearly changing with BuiltYear .
There has been an exponential growth after year 1980

In [27]:
```
ggplot(data = data_) + geom_smooth( mapping = aes( x=YearBuilt, y=SalePrice
```

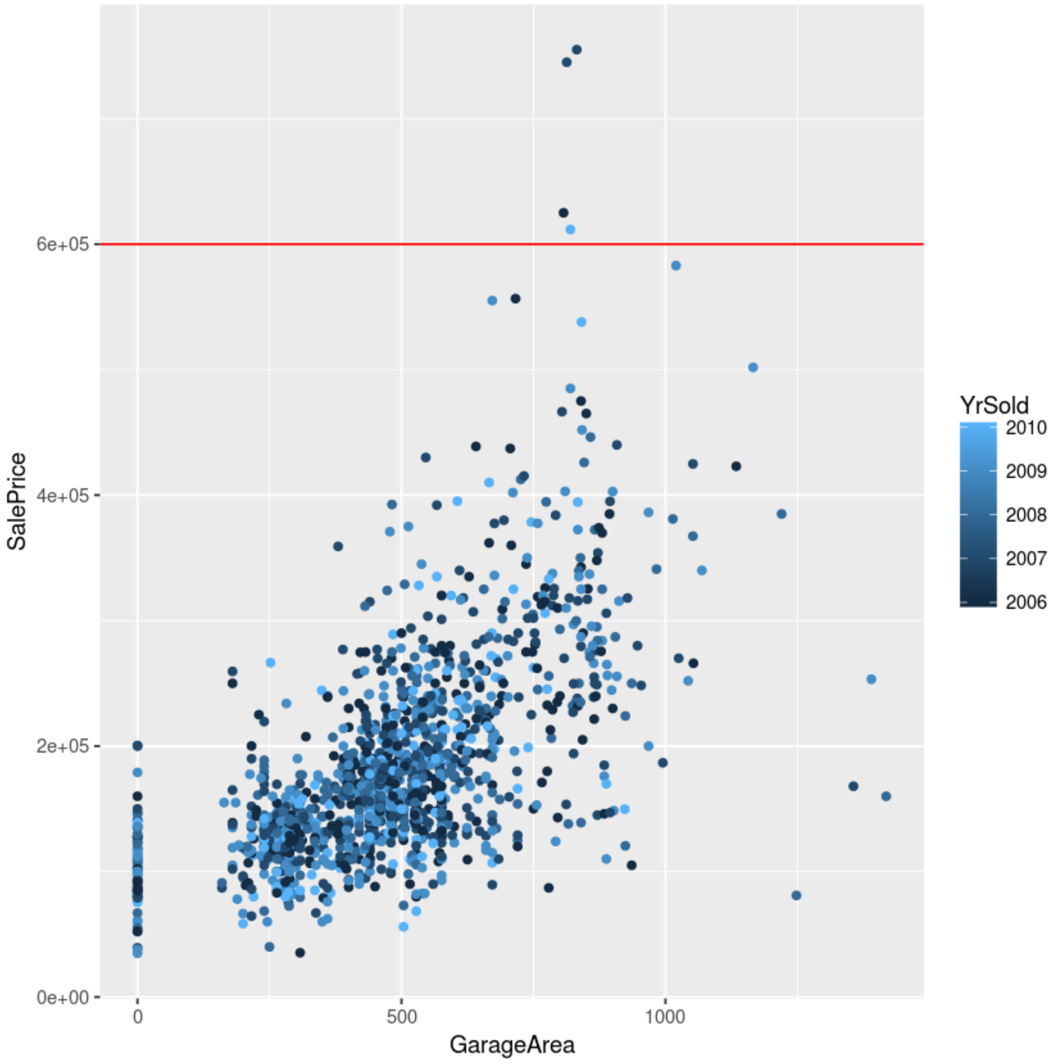
```
`geom_smooth()` using method = 'gam'
```



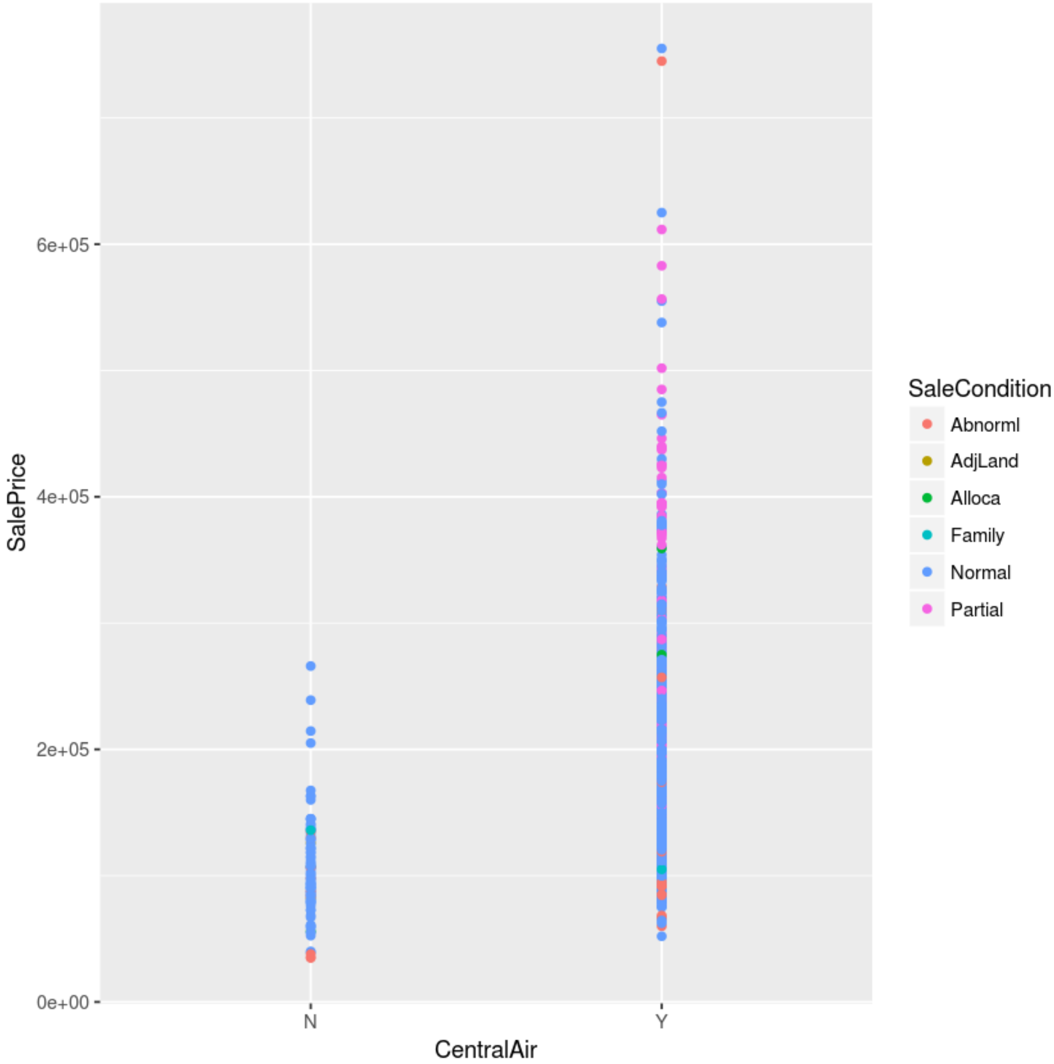
4.2 Multivariate analysis to get a comprehensive understanding

Over Priced houses

In [28]:
```
# we can see some over priced houses of sale condition
ggplot(data = imputed_data_) +
    geom_point( mapping = aes( x =GarageArea, y=SalePrice, color=YrSold) ) +
                   geom_hline(yintercept = 6e+05,color="red")
```



In [29]:
```
ggplot(data = imputed_data_) +
        geom_point( mapping = aes( x = CentralAir, y=SalePrice, color=SaleC(
```



4.3. Conclusions and Insights from Explorations

1. Univariate analysis provided us with features and their distributions
2. Multivariate Bivariate analysis gave us some relations b/w variables although we did not focus more here.

---

5. Fitting Model and Feature selection

Using randomForest model for predicting the house prices

Note: 1. We proceed forward without normalization because Random forest model fits the same for normalized data and un-normalized data.. i.e. It does not get affected by monotonic transformation of input variables.

We do the Primary fitting on two aspects of same dataset.

1. UNMODELED DATA : The data is neither transformed nor modelled
2. MODELED DATA : Here the categorical variables are transformed into attributes(binary)

In [31]:
```
#Loading the library for Random Forest model

library(randomForest)
```

In [30]:
```
#Creating a new dataframe to work on for modelfitting

df <- imputed_data_
```

Defining our own functions for Error calculation

Our error evaluation metric is RMSE of Logarthimic values. i.e:

RMSE_of_Log = Square-root of mean of square of [log(actual) - log(predicted)]

In [32]:
```
#PRE-REQUIRED FUNCTION FOR ERROR CALCULATION

#SQUARE FUNCTION
sqr <- function(i){
    return (i*i);
}

#RMSE
rmse <- function(predicted_values,dataframe,col_name){
    squared_sum = 0.0
    for (i in 1:length(predicted_values)){
        squared_sum = squared_sum + sqr(predicted_values[i]-dataframe[i,col_
    }
    rmse <- sqrt( squared_sum/nrow(dataframe) )
    return (rmse)
}

#Log-RMSE
lmse <- function(predicted_values,dataframe,col_name){
    squared_sum = 0.0
    for (i in 1:length(predicted_values)){
        squared_sum = squared_sum + sqr(log(predicted_values[i])-log(datafra
    }
    rmse <- sqrt( squared_sum/nrow(dataframe) )
    return (rmse)
}
```

5.1. Random Forest on UN-MODELLED DATA

In [35]:
```
#MODEL FITTING

rf_model <- randomForest(SalePrice ~ ., imputed_data_, do.trace=10, ntree=1(
```

```
     |      Out-of-bag   |
Tree |      MSE  %Var(y) |
  10 | 1.295e+09    20.53 |
  20 | 1.047e+09    16.60 |
  30 | 9.442e+08    14.97 |
  40 | 9.084e+08    14.40 |
  50 | 8.655e+08    13.72 |
  60 | 8.175e+08    12.96 |
  70 | 8.168e+08    12.95 |
  80 | 8.051e+08    12.77 |
  90 | 8.076e+08    12.81 |
 100 | 8.053e+08    12.77 |
 110 | 8.024e+08    12.72 |
 120 | 7.961e+08    12.62 |
 130 | 7.862e+08    12.47 |
 140 | 7.76e+08    12.30 |
 150 | 7.733e+08    12.26 |
 160 | 7.715e+08    12.23 |
 170 | 7.678e+08    12.17 |
 180 | 7.641e+08    12.11 |
```

In [36]:
```
#MODEL PREDICTION

x_prediction <- predict(rf_model, imputed_data_)
```

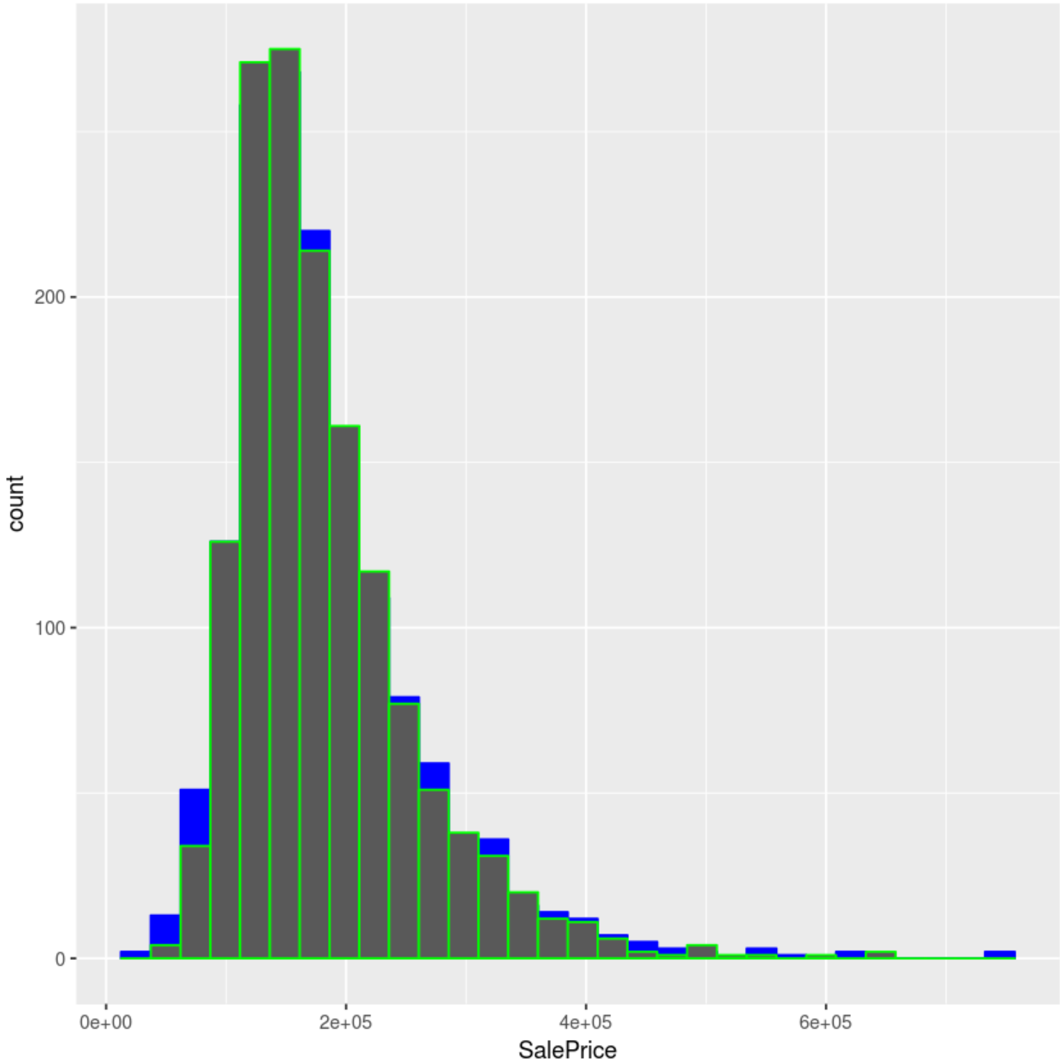
Plotting the Prediction value

Here the blue bars are actual distribution
And the grey shaded are the prediction distribution

We could observe the predictions are quite similarly distributed

In [37]:
```
#PLOT
ggplot(data = imputed_data_) +geom_histogram(mapping= aes(SalePrice),color='
```

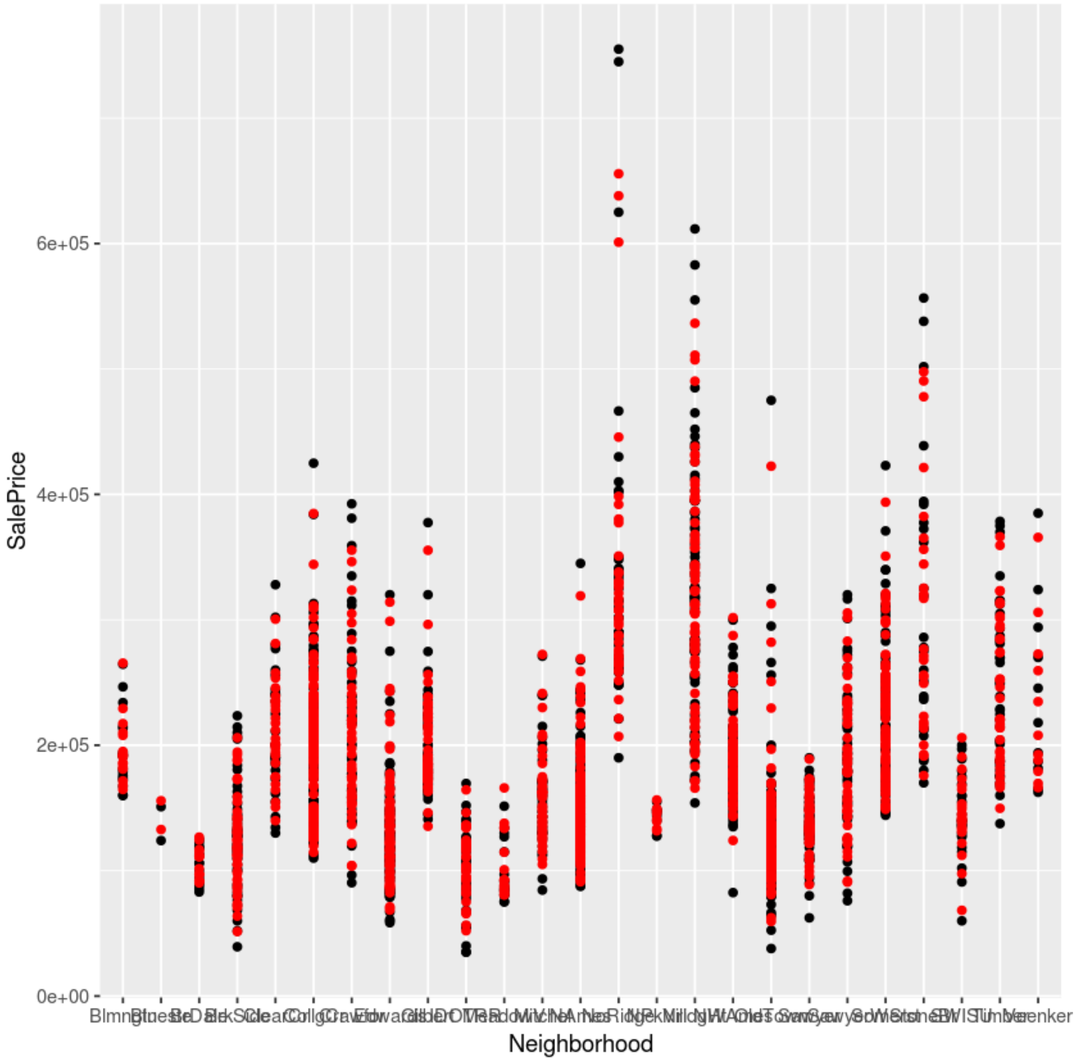
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Plotting the predicted values with respect to a specific attribute

If we could recall, we claimed the neighborhood plays a major role in determining the Salesprice. This is evident from the below plot which reveals that the distribution of Sales price is in alignment with the NEIGHBORHOOD Attribute..

In [38]:
```
ggplot(data = imputed_data_) +geom_point(mapping= aes(x =Neighborhood,y=Sale
```



Evaluation of error

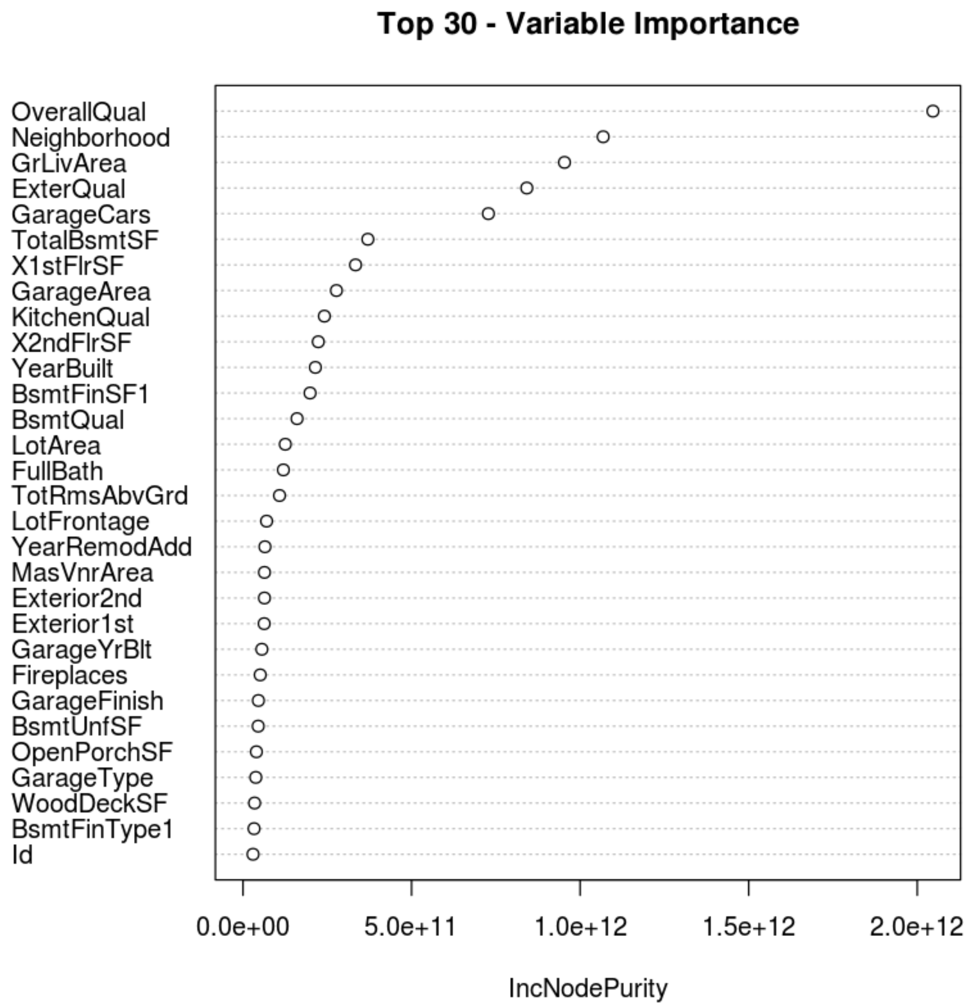
In [39]:
```
#RMSE CALCULATION

cat(err_1<-rmse(x_prediction,imputed_data_,'SalePrice'))
```
```
11245.63
```

In [40]:
```
#LMSE CALCULATION

cat(err_1<-lmse(x_prediction,imputed_data_,'SalePrice'))
```
```
0.06206722
```

Plotting the variable importance for the features in the model

The increasing node purity is used for rating features
It is analogous to finding intra-clusterdistance and inter-clusterdistance in clustering

In [45]:
```
# Important variable plot

varImpPlot(rf_modellmodel,
           sort = T,
           n.var=30,
           main="Top 30 - Variable Importance")
```



5.2. RANDOM FOREST on MODELLED DATA

We model the data using the model.matrix() method/function in R

In [ ]:
```
#MODELLED_DATA COLUMN SELECTION

cnames <- colnames(df)
cnames <- cnames[!cnames %in% "Id"]

#Formula for modelling Categorical variable into patterns

formula_for_modelling <- as.formula(paste ("~",paste(cnames,collapse = " + '

#CREATION OF MODELLED_DATA

modeled_df <- model.matrix(formula_for_modelling, data = df)

##FINALIZING MODELED_DATA

#selecting column names of modeled_data
cnames <- colnames(modeled_df)
cnames <- cnames[! cnames %in% "(Intercept)"]

#Adjusting the space between certain attribute's names::::::::::::::
cnames_ok = c()
for ( i in cnames){
    str = paste(unlist(strsplit(i," ")),collapse = "")
    cnames_ok <- c(cnames_ok,str )
}

#selecting attribute names without spaces
selected_attr = c()
for ( i in cnames_ok){
    if ( i %in% cnames && !i %in% c('RoofMatlTar&Grv')){
      selected_attr <- c(selected_attr,i)
    }
}


modeled_df <- modeled_df[,selected_attr]
```

In [51]:
```
#MODEL FITTING

rf_model2 <- randomForest(SalePrice ~ ., modeled_df, do.trace=10, ntree=100(
```

```
     |      Out-of-bag   |
Tree |      MSE  %Var(y) |
  10 | 1.198e+09    19.00 |
  20 | 1.221e+09    19.37 |
  30 | 9.903e+08    15.70 |
  40 | 9.272e+08    14.70 |
  50 | 9.452e+08    14.99 |
  60 | 9.186e+08    14.57 |
  70 | 8.987e+08    14.25 |
  80 | 8.809e+08    13.97 |
  90 | 8.719e+08    13.82 |
 100 | 8.908e+08    14.12 |
 110 | 8.852e+08    14.04 |
 120 | 8.854e+08    14.04 |
 130 | 8.742e+08    13.86 |
 140 | 8.67e+08    13.75 |
 150 | 8.574e+08    13.60 |
 160 | 8.491e+08    13.46 |
 170 | 8.493e+08    13.47 |
```

In [52]:
```
#PREDICTION

x_prediction2 <- predict(rf_model2, modeled_df)
```

Evaluation of error

In [53]:
```
#RMSE CALCULATION
#RMSE
cat(err_2 <-rmse(x_prediction2,modeled_df,'SalePrice'))
```

```
11665.76
```

In [54]:
```
#RMSE
cat(err_2 <-lmse(x_prediction2,modeled_df,'SalePrice'))
```

```
0.06360193
```

Comparision between the results of model1 and model2

We could observe that the results in case of modeled data are less accurate than expected.
This negative results are the consequence of increasing the data attributes with the same no. of instances.

In [55]:
```
# COMPARISION
cat(err_1," ",err_2)
```

```
0.06231549   0.06360193
```

In [71]:
```
#Reason for difference in results
dim(modeled_df)
```

1460 232

In [ ]: