Housing Price data exploration

July 23, 2017

This is the Exploration Data Analysis (EDA) part of kaggle competition - Housing price. The contents include: data overview and inspect, data cleaning and some feature engineering.

```
setwd('C:/Users/Bangda/Desktop/kaggle/housing-price')
library(tidyverse)
library(magrittr)
library(stringr)
library(e1071)
library(VIM)
library(mice)
library(gridExtra)
train <- read.csv('train.csv', header = TRUE, stringsAsFactors = FALSE)
test  <- read.csv('test.csv', header = TRUE, stringsAsFactors = FALSE)
dim(train)
```

```
## [1] 1460   81
```

```
dim(test)
```

```
## [1] 1459   80
```

We can see that there are many variables available to be the predictors, their meaning can be found here: https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data.

1. Data overview

Since there are too many variables, we won't display the structure of `train` and `test` right here. We can have a basic idea of the dataset: many of variables are categorical variables, like types or conditions etc. In addition, the categories of categorical variables are not too much (unlike the situation in Titanic data sets). And we can see some variables may related together from the variable names, some variables seems like can be "grouped".

Let's check the data types of `train` and `test`.

```
# get the data type of each column
train_var_class <- sapply(train, class)
test_var_class  <- sapply(test, class)
table(train_var_class)
```

```
## train_var_class
## character   integer
##        43        38
```

```
table(test_var_class)
```

```
## test_var_class
## character   integer
##        43        37
```

```
# numeric variables
train_var_class[train_var_class %in% c('integer', 'numeric')] %>% names()
```

```
##  [1] "Id"            "MSSubClass"    "LotFrontage"   "LotArea"
```

```
##  [5] "OverallQual"   "OverallCond"   "YearBuilt"     "YearRemodAdd" 
##  [9] "MasVnrArea"    "BsmtFinSF1"    "BsmtFinSF2"    "BsmtUnfSF"    
## [13] "TotalBsmtSF"   "X1stFlrSF"     "X2ndFlrSF"     "LowQualFinSF" 
## [17] "GrLivArea"     "BsmtFullBath"  "BsmtHalfBath"  "FullBath"     
## [21] "HalfBath"      "BedroomAbvGr"  "KitchenAbvGr"  "TotRmsAbvGrd" 
## [25] "Fireplaces"    "GarageYrBlt"   "GarageCars"    "GarageArea"   
## [29] "WoodDeckSF"    "OpenPorchSF"   "EnclosedPorch" "X3SsnPorch"   
## [33] "ScreenPorch"   "PoolArea"      "MiscVal"       "MoSold"       
## [37] "YrSold"        "SalePrice"
```

```r
# categorical variables
train_var_class[train_var_class %in% c('factor', 'character')] %>% names()
```

```
##  [1] "MSZoning"      "Street"        "Alley"         "LotShape"     
##  [5] "LandContour"   "Utilities"     "LotConfig"     "LandSlope"    
##  [9] "Neighborhood"  "Condition1"    "Condition2"    "BldgType"     
## [13] "HouseStyle"    "RoofStyle"     "RoofMatl"      "Exterior1st"  
## [17] "Exterior2nd"   "MasVnrType"    "ExterQual"     "ExterCond"    
## [21] "Foundation"    "BsmtQual"      "BsmtCond"      "BsmtExposure" 
## [25] "BsmtFinType1"  "BsmtFinType2"  "Heating"       "HeatingQC"    
## [29] "CentralAir"    "Electrical"    "KitchenQual"   "Functional"   
## [33] "FireplaceQu"   "GarageType"    "GarageFinish"  "GarageQual"   
## [37] "GarageCond"    "PavedDrive"    "PoolQC"        "Fence"        
## [41] "MiscFeature"   "SaleType"      "SaleCondition"
```

But when we check the actual value of some "numerical" variables we can find that some values are kinds of rank or grade, they might be also considered as categorical variables.

2. Missing data

(1) Detect missing data

```r
train_na_stat <- apply(train, 2, function(.data) sum(is.na(.data)))
train_na_stat[train_na_stat > 0]
```

```
##  LotFrontage        Alley   MasVnrType   MasVnrArea     BsmtQual 
##          259         1369            8            8           37 
##     BsmtCond BsmtExposure BsmtFinType1 BsmtFinType2   Electrical 
##           37           38           37           38            1 
##  FireplaceQu   GarageType  GarageYrBlt GarageFinish   GarageQual 
##          690           81           81           81           81 
##   GarageCond       PoolQC        Fence  MiscFeature 
##           81         1453         1179         1406
```

```r
test_na_stat  <- apply(test, 2, function(.data) sum(is.na(.data)))
test_na_stat[test_na_stat > 0]
```

```
##     MSZoning  LotFrontage        Alley    Utilities  Exterior1st 
##            4          227         1352            2            1 
##  Exterior2nd   MasVnrType   MasVnrArea     BsmtQual     BsmtCond 
##            1           16           15           44           45 
## BsmtExposure BsmtFinType1   BsmtFinSF1 BsmtFinType2   BsmtFinSF2 
##           44           42            1           42            1 
##    BsmtUnfSF  TotalBsmtSF BsmtFullBath BsmtHalfBath  KitchenQual 
##            1            1            2            2            1
```

```
##   Functional  FireplaceQu   GarageType  GarageYrBlt GarageFinish
##            2          730           76           78           78
##   GarageCars   GarageArea   GarageQual   GarageCond       PoolQC
##            1            1           78           78         1456
##        Fence  MiscFeature     SaleType
##         1169         1408            1
```

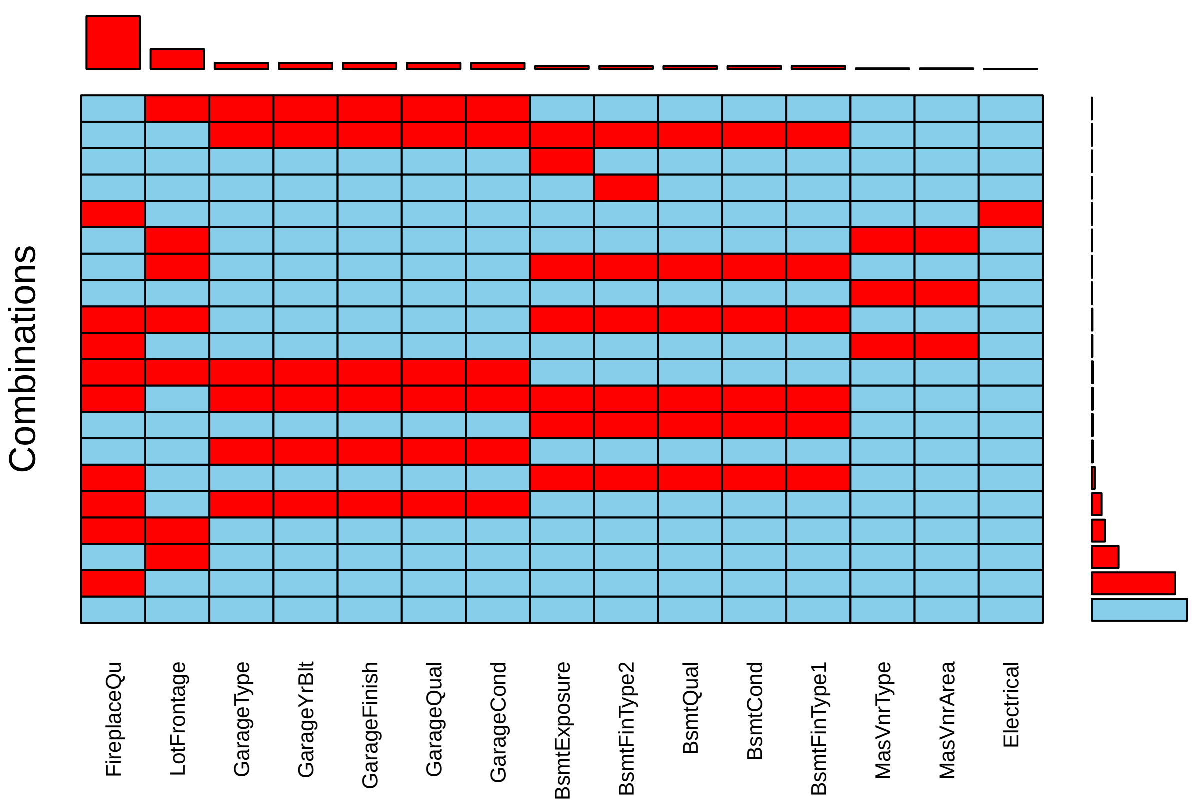
```r
# get the variables contain NA
train_na_variable <- names(train_na_stat[train_na_stat > 0])
test_na_variable  <- names(test_na_stat[test_na_stat > 0])
```

The missing rates of some variables are more than 50%, we can remove them directly since the information loss are significant. Therefore we remove *Alley*, *PoolQC*, *Fence MiscFeature* from the data set.

```r
train %<>%
  select(-Alley, -PoolQC, -Fence, -MiscFeature)
test %<>%
  select(-Alley, -PoolQC, -Fence, -MiscFeature)
# update
train_na_stat <- apply(train, 2, function(.data) sum(is.na(.data)))
test_na_stat  <- apply(test, 2, function(.data) sum(is.na(.data)))
train_na_variable <- names(train_na_stat[train_na_stat > 0])
test_na_variable  <- names(test_na_stat[test_na_stat > 0])
```

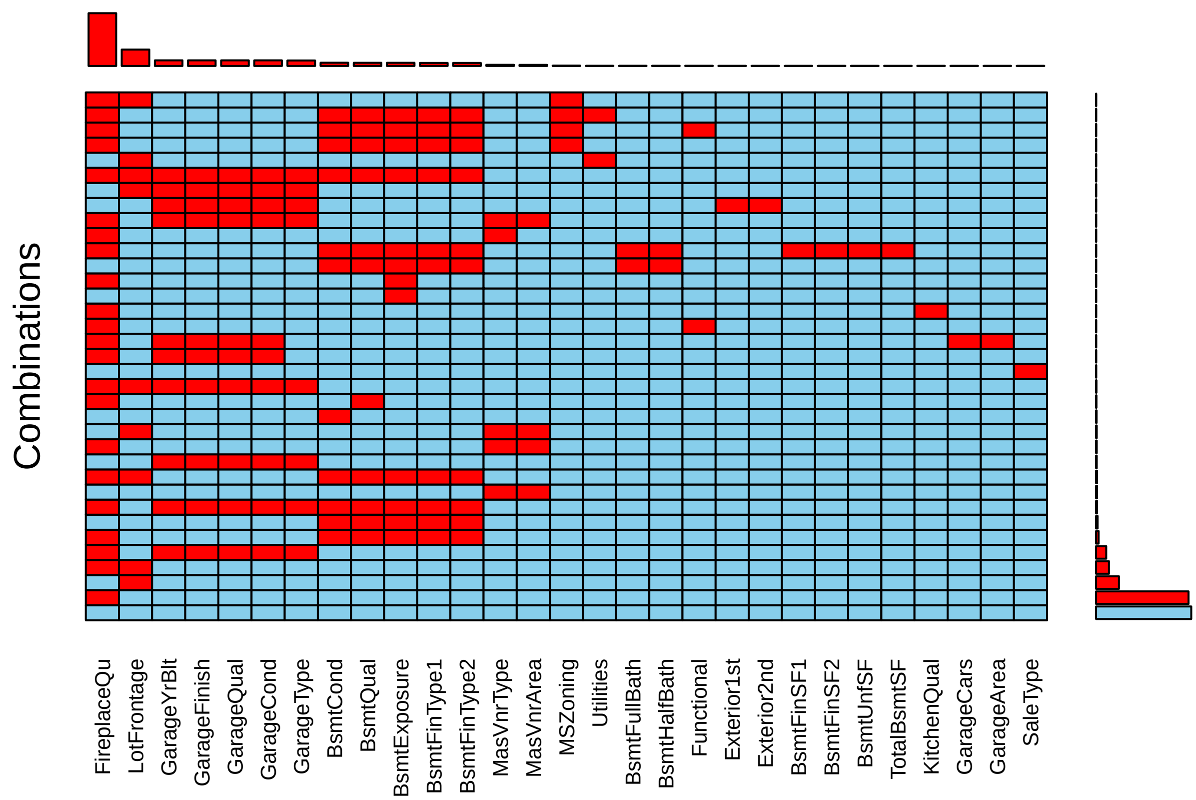
Then we visualize the missing data with the help of `aggr()` function in `VIM`.

```r
train[, colnames(train) %in% names(train_na_stat[train_na_stat > 0])] %>%
  aggr(prop = FALSE, combined = TRUE, sortVars = TRUE, cex.axis = .7)
```



```
##
## Variables sorted by number of missings:
##      Variable Count
##   FireplaceQu   690
##   LotFrontage   259
##    GarageType    81
##   GarageYrBlt    81
##  GarageFinish    81
##    GarageQual    81
##    GarageCond    81
##  BsmtExposure    38
##  BsmtFinType2    38
##      BsmtQual    37
##      BsmtCond    37
##  BsmtFinType1    37
##    MasVnrType     8
##    MasVnrArea     8
##    Electrical     1
```

```r
test[, colnames(test) %in% names(test_na_stat[test_na_stat > 0])] %>%
  aggr(prop = FALSE, combined = TRUE, sortVars = TRUE, cex.axis = .7)
```




```
##
## Variables sorted by number of missings:
##      Variable Count
##   FireplaceQu   730
##   LotFrontage   227
```

```
##   GarageYrBlt    78
##  GarageFinish    78
##    GarageQual    78
##    GarageCond    78
##    GarageType    76
##      BsmtCond    45
##      BsmtQual    44
##  BsmtExposure    44
##  BsmtFinType1    42
##  BsmtFinType2    42
##    MasVnrType    16
##    MasVnrArea    15
##      MSZoning     4
##     Utilities     2
##  BsmtFullBath     2
##  BsmtHalfBath     2
##    Functional     2
##   Exterior1st     1
##   Exterior2nd     1
##    BsmtFinSF1     1
##    BsmtFinSF2     1
##     BsmtUnfSF     1
##   TotalBsmtSF     1
##   KitchenQual     1
##    GarageCars     1
##    GarageArea     1
##      SaleType     1
```

We can also go with the "observations" side, we will count the missing variables for each observations.

```
train %>%
  apply(1, function(.row) sum(is.na(.row))) %>%
  sort(x = ., decreasing = TRUE) %>%
  '[' (1:10)
```

```
##  [1] 11 11 11 11 11 11 10  7  7  7
```

```
test %>%
  apply(1, function(.row) sum(is.na(.row))) %>%
  sort(x = ., decreasing = TRUE) %>%
  '[' (1:10)
```

```
##  [1] 12 12 11 11 11 11 11 11  8  8
```

So we know that the largest number of missing variables for observations is 11 in `train` and 12 in `test`. One of the practical rule is we can delete the observation if it has many missing variables. However, since the number of variables is much more greater than missing variables, we will not apply this rule in this case.

If we put variables contain `NA` along with the variable type we are suprised to find that almost all of them are categorical variables.

(2) Fill in missing data

One naive way to fill missing data for categorical data is using mode. So we will start from that and go back here later for more complicated methods like classification algorithms (decision trees) etc.

Since there is no built-in function to compute mode, we will define it by ourselves.

```
getMode <- function(x, na.rm = TRUE) {
  # get mode for character vector
  if (na.rm) {
    sort(table(x[!is.na(x)]), decreasing = TRUE)[1] %>% names()
  } else {
    sort(table(x), decreasing = TRUE)[1] %>% names()
  }
}
```

Gathering variables with `NA`,

```
train_na_set <- train[, colnames(train) %in% train_na_variable]
test_na_set  <- test[, colnames(test) %in% test_na_variable]
```

Then apply our filling rule: use median to fill missing data for numeric variables, use mode to fill missing data for categorical variables,

```
fillNA <- function(x) {

  if (sum(is.na(x)) == 0) return(x)

  if (class(x) %in% c('integer', 'numeric')) {
    x[which(is.na(x))] <- median(x, na.rm = TRUE)
  } else {
    x[which(is.na(x))] <- getMode(x, na.rm = TRUE)
  }
  x
}
```

Test,

```
table(train_na_set$MasVnrType)
```

```
##
##  BrkCmn BrkFace    None   Stone
##      15     445     864     128
```

```
train_na_set$MasVnrType %<>% fillNA()
table(train_na_set$MasVnrType)
```

```
##
##  BrkCmn BrkFace    None   Stone
##      15     445     872     128
```

Finally we apply the function on data set, since using `apply()` with has some data type issues, we use for - loop here.

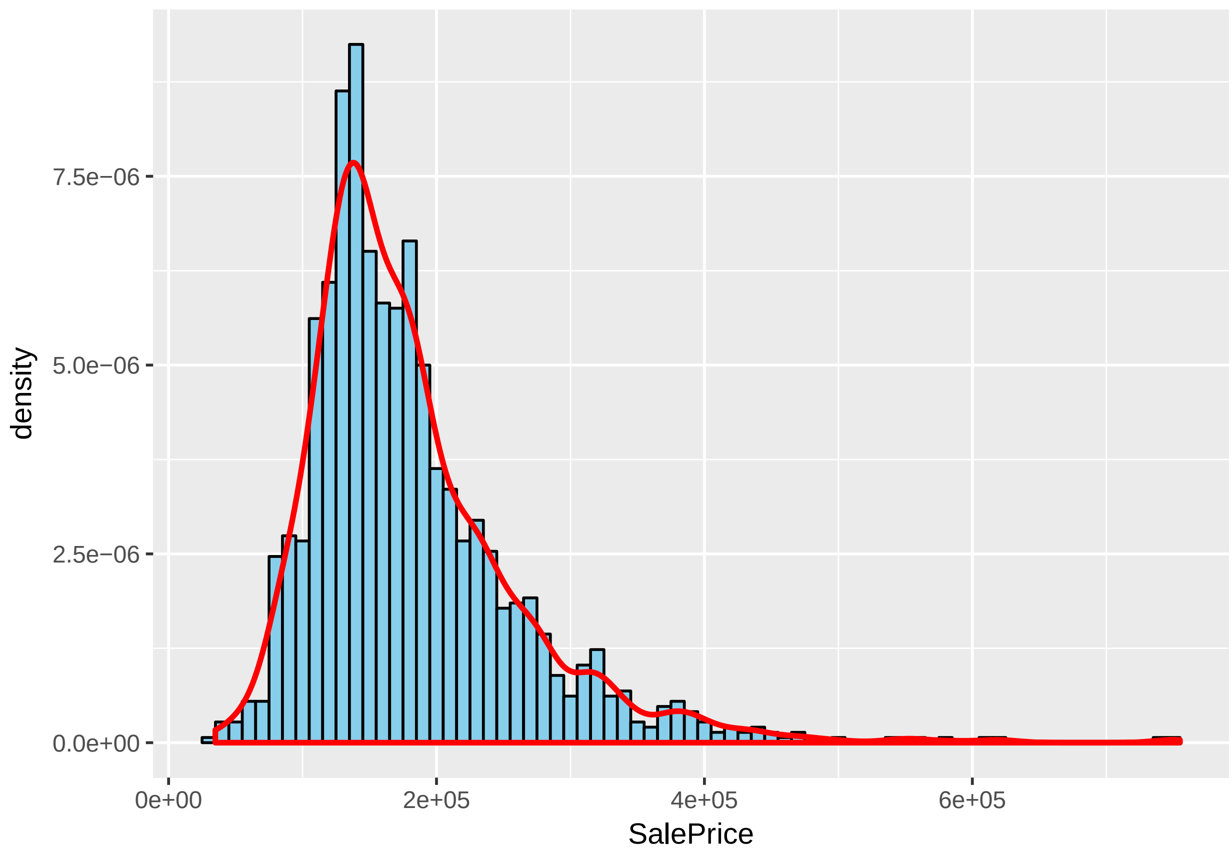
```
train_na_filled <- train
test_na_filled <- test
for (i in 1:ncol(train)) {
  train_na_filled[, i] <- fillNA(train[, i])
}
for (i in 1:ncol(test)) {
  test_na_filled[, i] <- fillNA(test[, i])
}
```

3. EDA and Feature Engineering

(1) Price

In order to satisfy the assumption of modeling (e.g. linear regression), we expect the *SalePrice* has normal distribution,

```
ggplot(train, aes(x = SalePrice)) +
  geom_histogram(aes(y = ..density..), binwidth = 10000,
                 color = 'black', fill = 'skyblue') +
  geom_density(aes(y = ..density..), size = 1, col = 'red')
```




```
skewness(train$SalePrice)
```

```
## [1] 1.879009
```

The histogram says the distribution of *SalePrice* is not perfectly normal, it's a little right skewed since there are some relative high price.

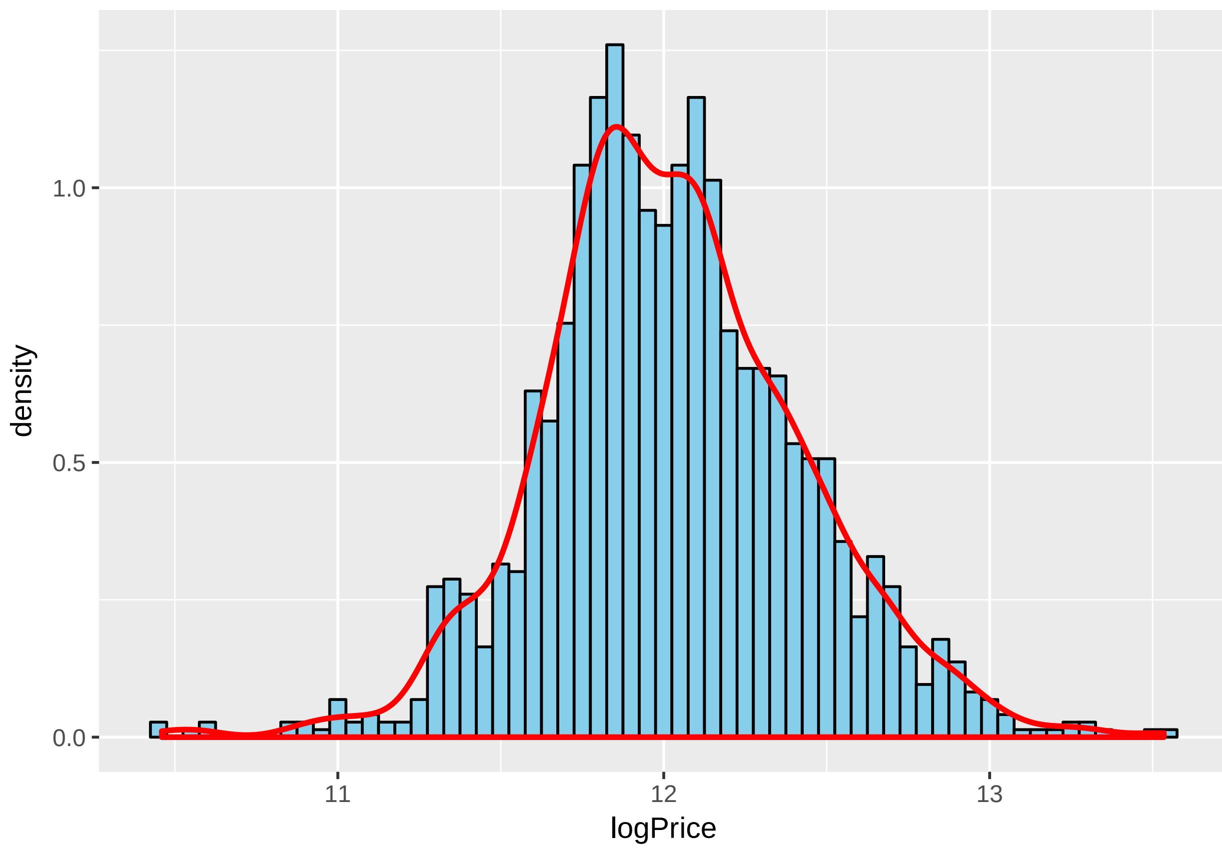
```
train %>%
  filter(SalePrice <= quantile(SalePrice, .97)) %>%
  ggplot(aes(x = SalePrice)) +
  geom_histogram(aes(y = ..density..), binwidth = 5000,
                 color = 'black', fill = 'skyblue') +
  geom_density(aes(y = ..density..), size = 1, col = 'red')
```


```
train %>%
  filter(SalePrice <= quantile(SalePrice, .97)) %$%
  skewness(SalePrice)
```

```
## [1] 0.7975751
```

Based on the evaluation of the result, we can also take the logarithm

```
train %>%
  mutate(logPrice = log(SalePrice)) %>%
  ggplot(aes(x = logPrice)) +
  geom_histogram(aes(y = ..density..), binwidth = 0.05,
                 color = 'black', fill = 'skyblue') +
  geom_density(aes(y = ..density..), size = 1, col = 'red')
```



Now it looks better. Hence we can draw a conclusion that if the distribution is skewed, we can try to use logarithm transformation to fix it since it can "scale" the extreme value. Also don't forget to take exponential transformation to convert back.

```
train %<>% mutate(logSalePrice = log(SalePrice))
train_na_filled %<>% mutate(logSalePrice = log(SalePrice))
skewness(train$logSalePrice)
```

```
## [1] 0.1210859
```

(2) Variable cluster

We observed that some vaiables have same prefix, they are considered to describe different attributes of same object. For example, varaibles with prefix *Bsmt* are descriptions of basement. Our goal is try to find those groups.

```
# get the first 4 characters of names
prefix <- names(train) %>% str_sub(1, 3) %>% unique()
prefix_ref <- data.frame(prefix, stringsAsFactors = FALSE)
# count the variables with same prefix
prefix_ref$count <- sapply(prefix_ref$prefix,
                           function(pattern) str_detect(names(train), pattern) %>% sum())
# filter the count with at least 3
prefix_ref %<>%
  filter(count >= 3)
# get the variables that could have groups
grouped_var_idx <- sapply(
```

```r
  str_extract_all(paste0(names(train)),
                  paste(prefix_ref$prefix, collapse = '|')),
  function(num) length(num) > 0)
names(train)[grouped_var_idx]
```

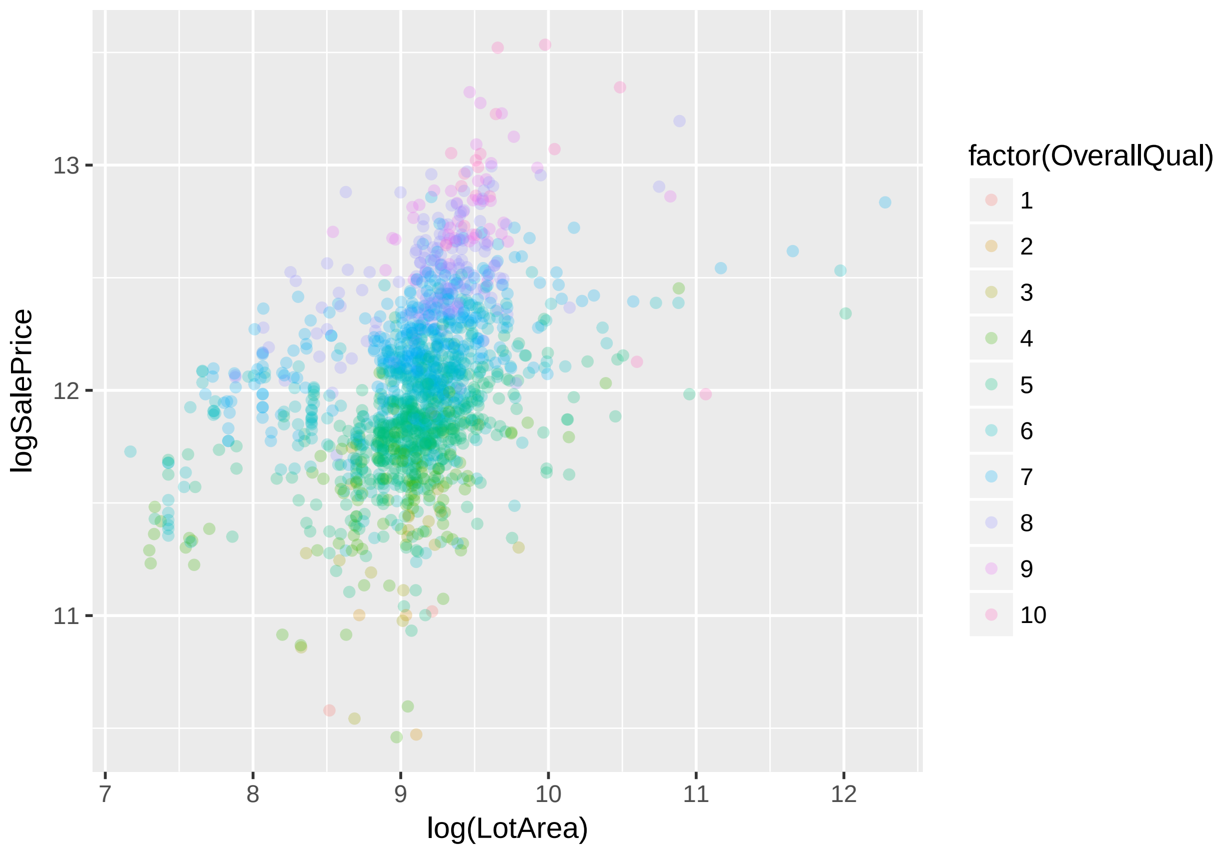
```
##  [1] "LotFrontage"   "LotArea"       "LotShape"      "LandContour"
##  [5] "LotConfig"     "Condition1"    "Condition2"    "OverallCond"
##  [9] "Exterior1st"   "Exterior2nd"   "ExterQual"     "ExterCond"
## [13] "BsmtQual"      "BsmtCond"      "BsmtExposure"  "BsmtFinType1"
## [17] "BsmtFinSF1"    "BsmtFinType2"  "BsmtFinSF2"    "BsmtUnfSF"
## [21] "TotalBsmtSF"   "BsmtFullBath"  "BsmtHalfBath"  "GarageType"
## [25] "GarageYrBlt"   "GarageFinish"  "GarageCars"    "GarageArea"
## [29] "GarageQual"    "GarageCond"    "SaleType"      "SaleCondition"
## [33] "SalePrice"     "logSalePrice"
```

Actually, we can check the descriptions of variables, and roughly classify them into 4 categories: (1) Interior, like *LotArea*, *Utilities*; (2) Exterior, like *MSSubClass*, *LotFrontage*; (3) Location and transporations, like *street*, *Condition1*; (4) Other attributes, like *YearBuilt*, *YearRemodAdd*.

(3) Interior

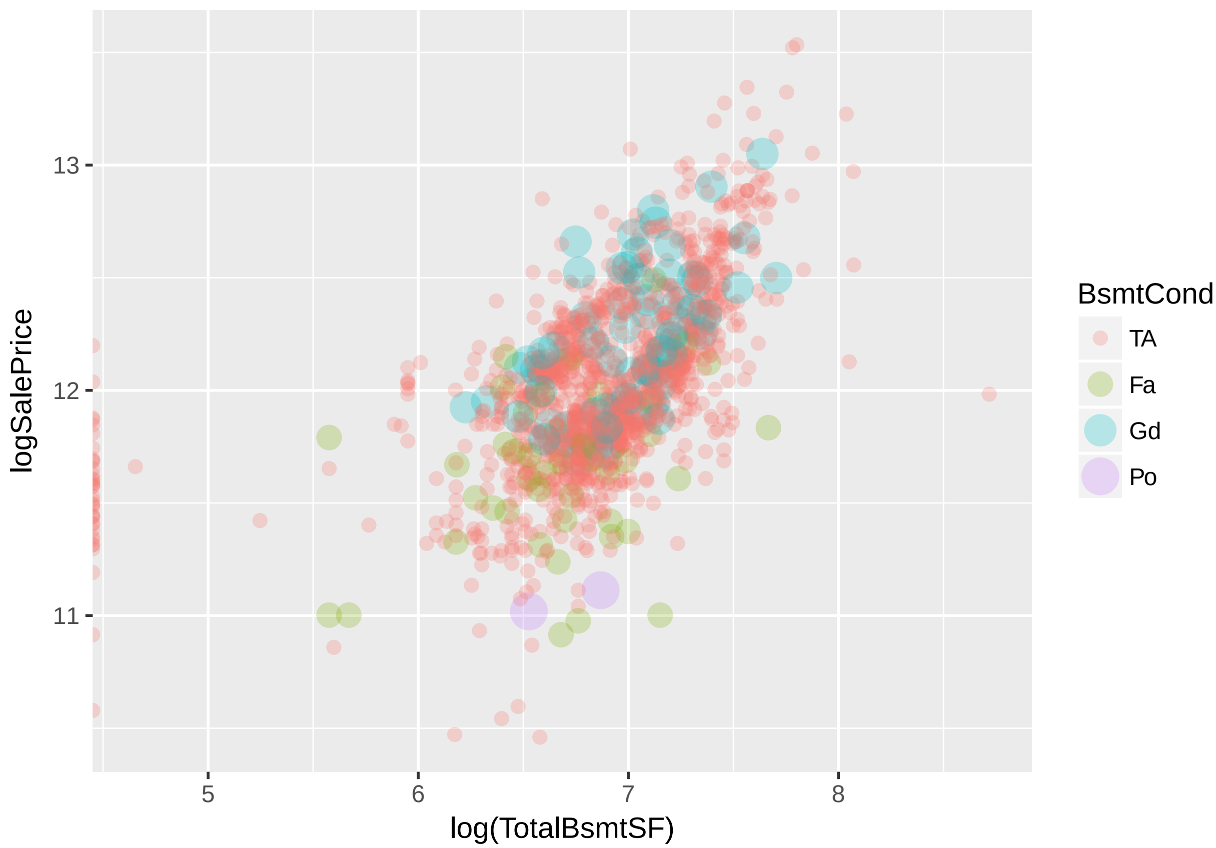
Typically *LotArea* is a key factor that everyone will consider when buying a house, we can see that there is a linear positive correlation between *LotArea* and *logSalePrice*. Adding *OverallQual* makes this statement more reasonable.

```r
ggplot(train_na_filled) +
  geom_point(aes(x = log(LotArea), y = logSalePrice,
                 col = factor(OverallQual)),
             alpha = I(1/4))
```



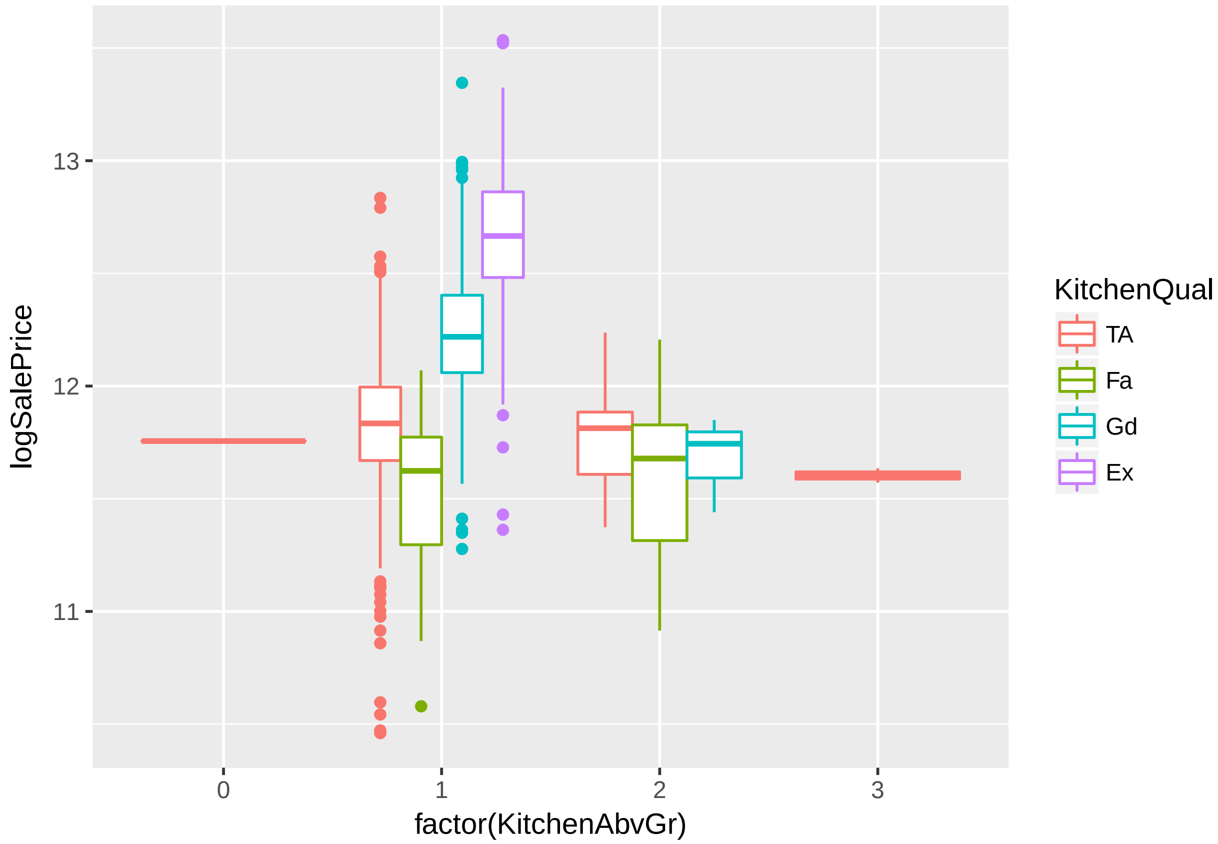
We can also get similar conculsion on the condition of basement, hence condition of basement is also a key factor count for housing price.

```
train_na_filled %>%
  mutate(BsmtCond = factor(BsmtCond, levels = c('TA', 'Fa', 'Gd', 'Po'))) %>%
  ggplot() +
  geom_point(aes(x = log(TotalBsmtSF), y = logSalePrice, col = BsmtCond,
                 size = BsmtCond), alpha = I(1/4))
```

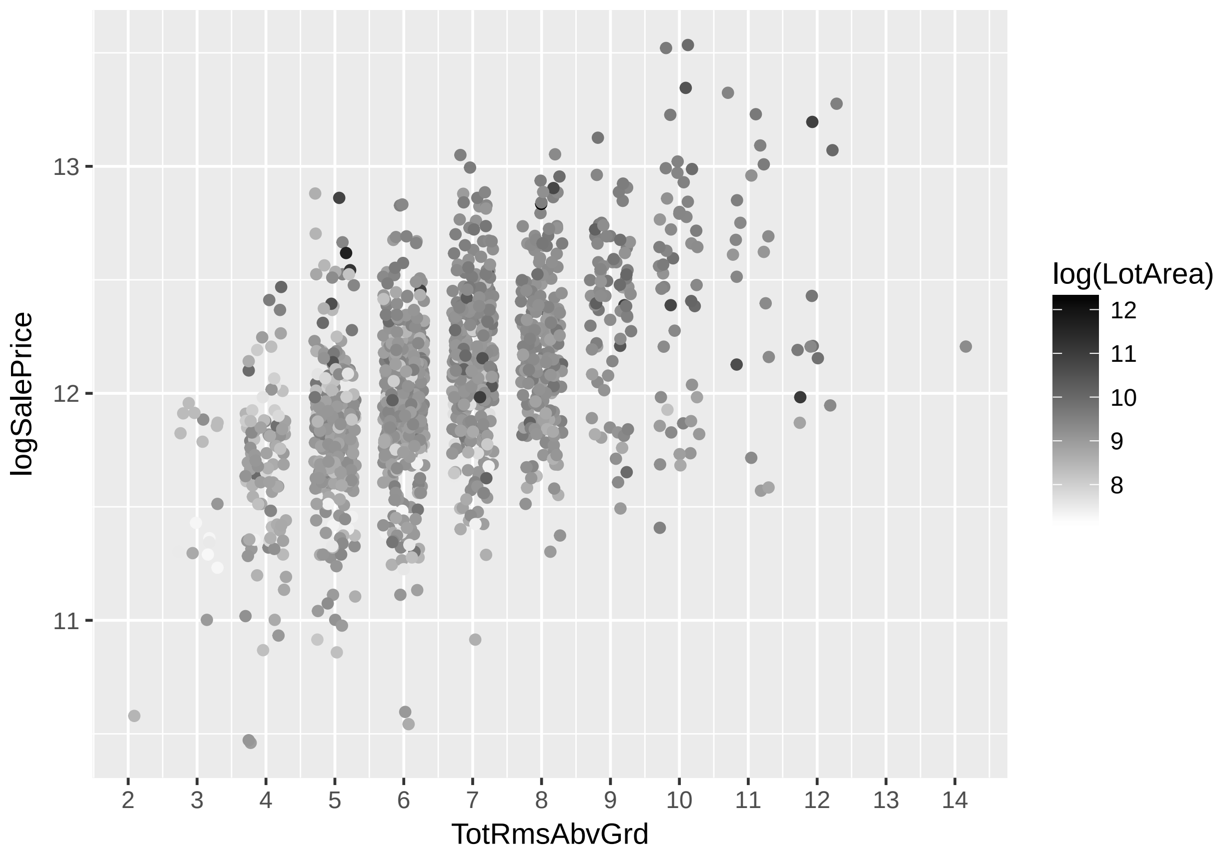


Then check the number and conditions of Bedrooms (above ground), Kitchen and overall rooms, we illustrate those variables along with *LotArea*.

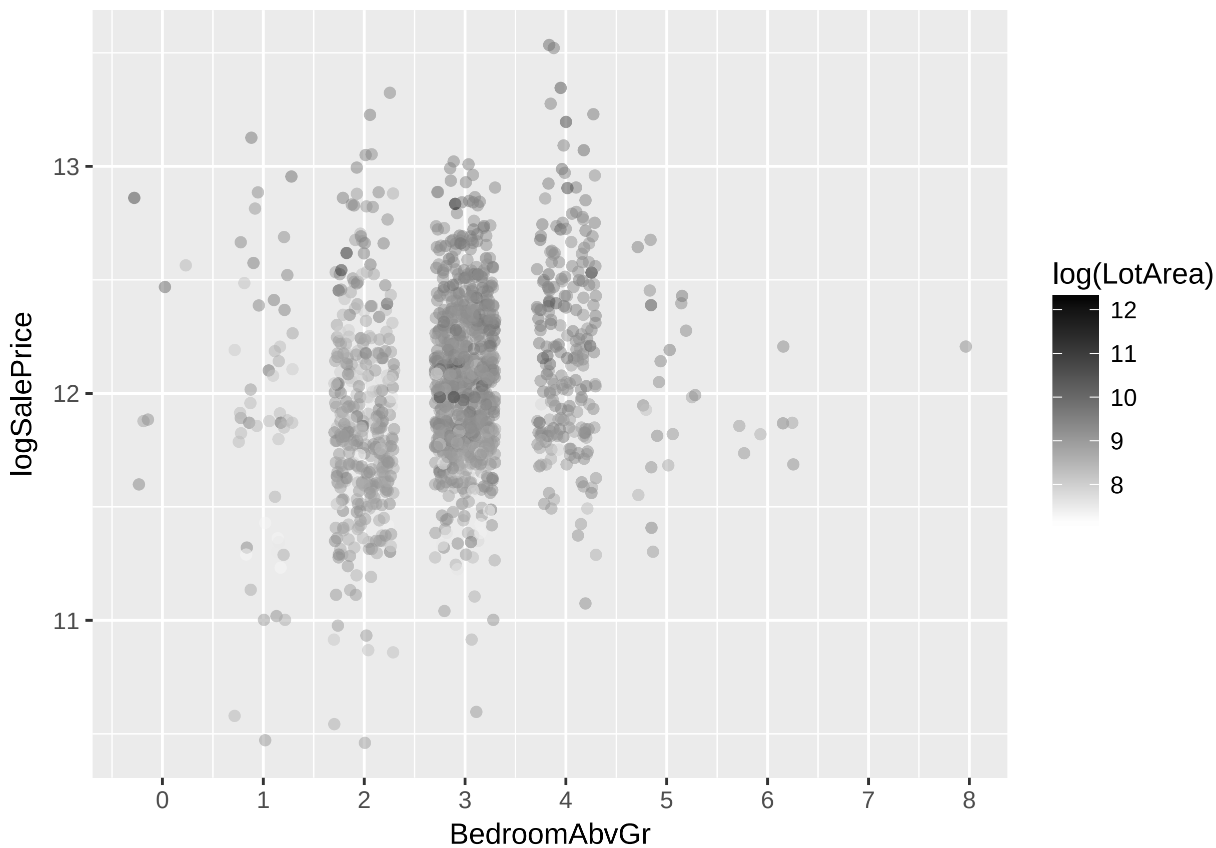
```
train_na_filled %>%
  mutate(KitchenQual = factor(KitchenQual, levels = c('TA', 'Fa', 'Gd', 'Ex'))) %>%
  ggplot(aes(x = factor(KitchenAbvGr), y = logSalePrice)) +
  geom_boxplot(aes(col = KitchenQual))
```




```
ggplot(train_na_filled, aes(x = TotRmsAbvGrd, y = logSalePrice)) +
  geom_jitter(aes(col = log(LotArea)), width = .3) +
  scale_x_continuous(breaks = 2:14) +
  scale_colour_gradient(low = "white", high = "black")
```




```
ggplot(train_na_filled) +
  geom_jitter(aes(x = BedroomAbvGr, y = logSalePrice, col = log(LotArea)),
              width = .3, alpha = I(1/2)) +
  scale_x_continuous(breaks = 0:8) +
  scale_colour_gradient(low = "white", high = "black")
```

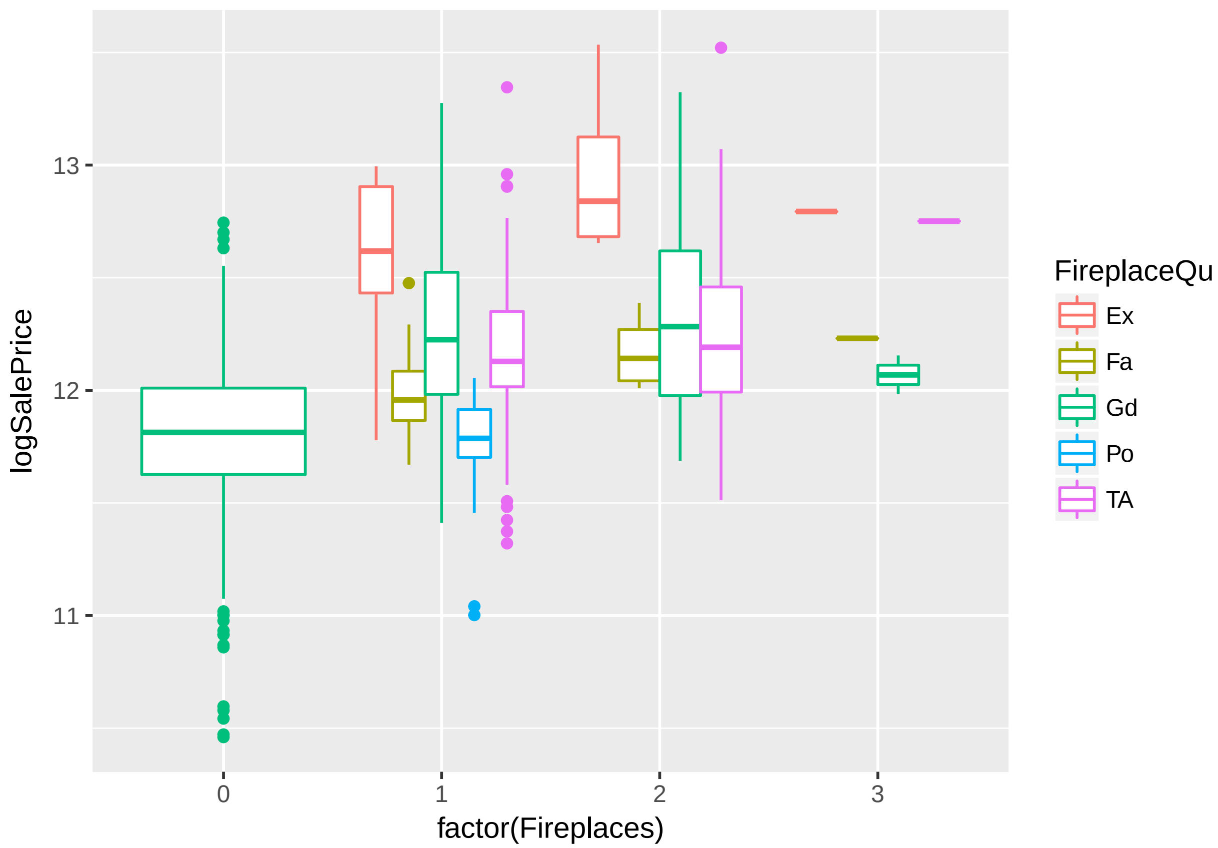


From the plots with *KitchenAbrGrd* and *KitchenQual* we find that many houses have just one kitchen, therefore the number of kitchen doesn't has too much contribution on *logSalePrice*. However *logSalePeice* shows significant differences on different levels of *KitchenQual*.

We see that *logSalePrice* increases as the total number of room increases in general, and similar pattern could be found on number of bedroom but not very clear.

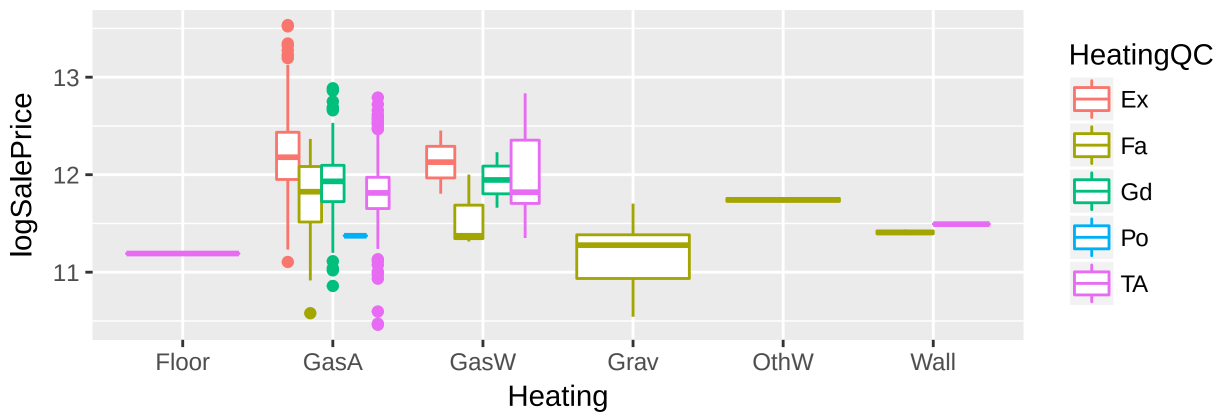
And what about *Fireplaces* and *FireplaceQu*?

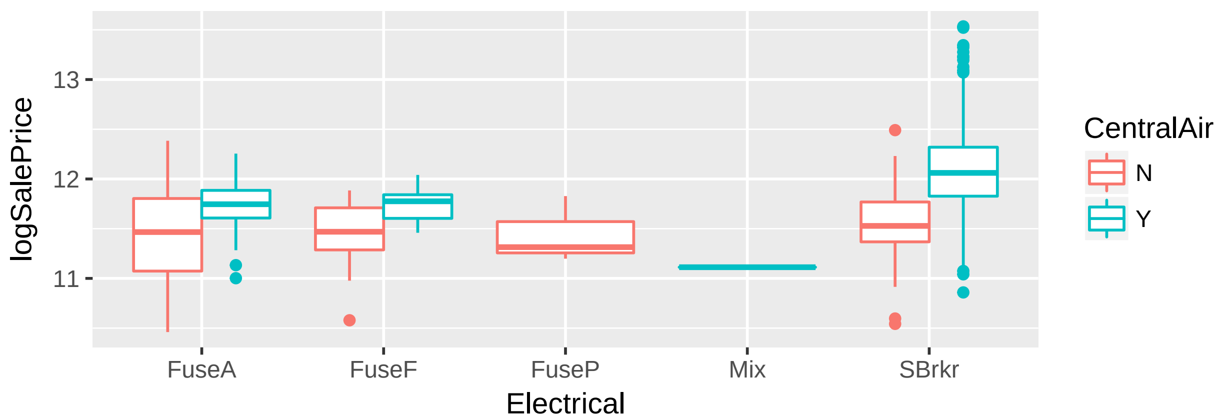
```
ggplot(train_na_filled) +
  geom_boxplot(aes(x = factor(Fireplaces), y = logSalePrice,
                   col = FireplaceQu))
```



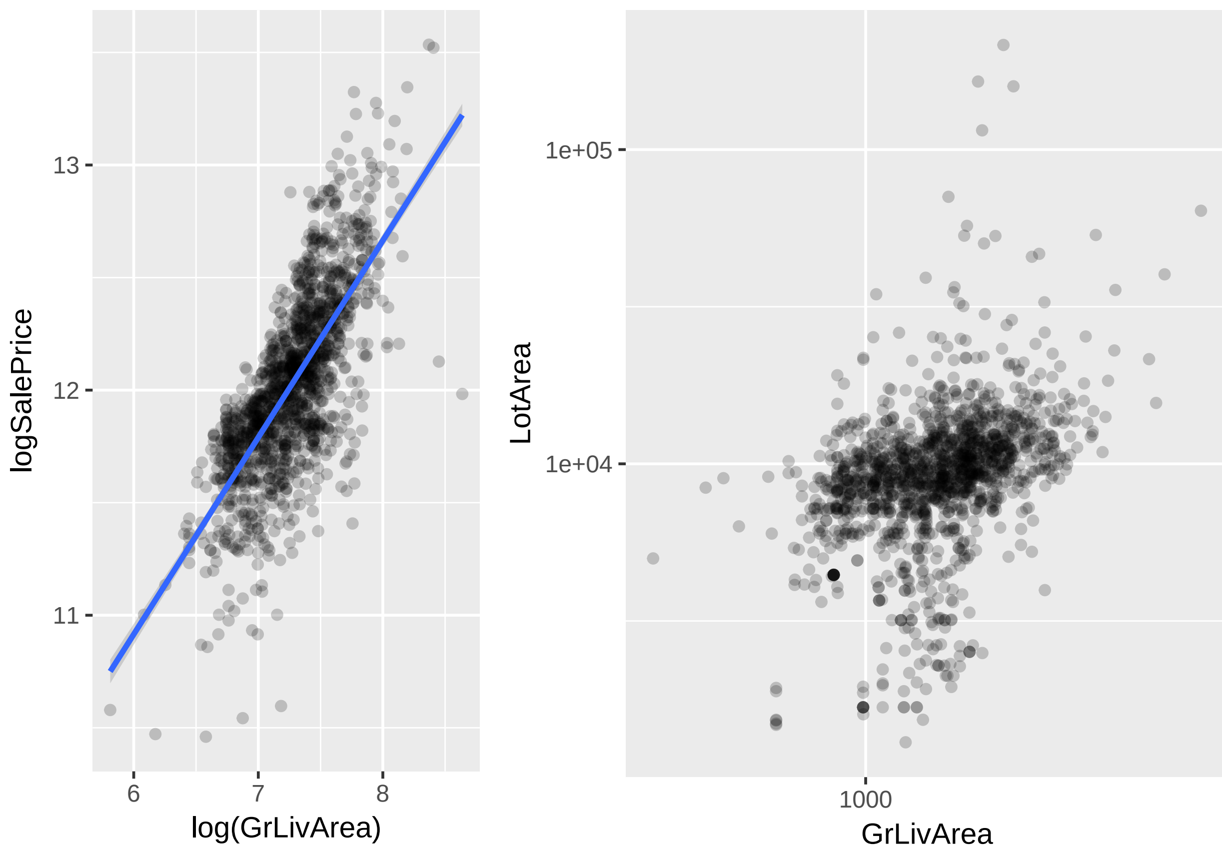
Electrical and *Heating* are also important factors in common sense, therefore we check them also as well as *GrLivArea*.

```
pHeat <- ggplot(train_na_filled) +
  geom_boxplot(aes(x = Heating, y = logSalePrice,
                   col = HeatingQC))
pElecAir <- ggplot(train_na_filled) +
  geom_boxplot(aes(x = Electrical, y = logSalePrice,
                    col = CentralAir))
grid.arrange(pHeat, pElecAir, nrow = 2)
```





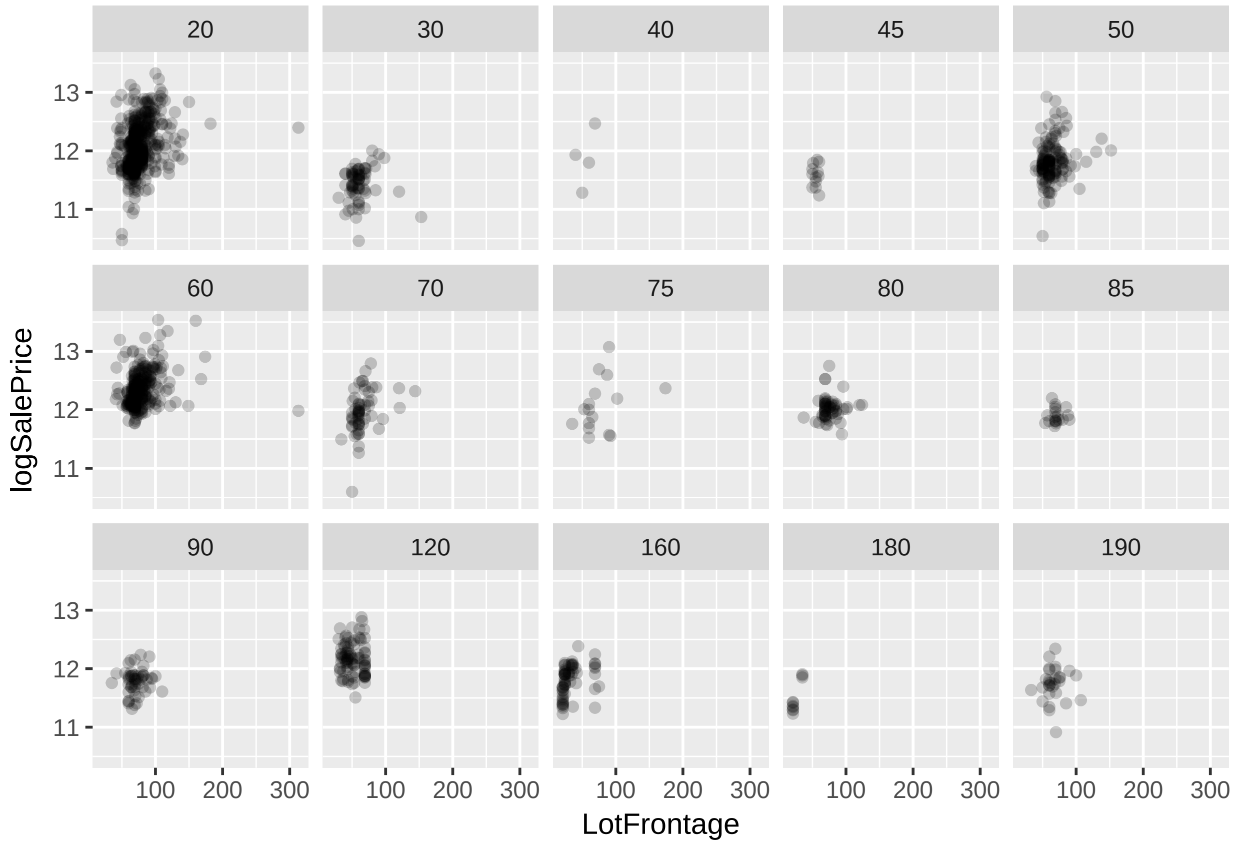

```
pGrLiv <- ggplot(train_na_filled, aes(x = log(GrLivArea), y = logSalePrice)) +
  geom_point(alpha = I(1/5)) +
  geom_smooth(method = 'lm', level = .9)
pGrLivLot <- ggplot(train_na_filled) +
  geom_point(aes(x = GrLivArea, y = LotArea), alpha = I(1/5)) +
  scale_x_log10() +
  scale_y_log10()
grid.arrange(pGrLiv, pGrLivLot, ncol = 2, widths = c(2, 3))
```



(4) Exterior

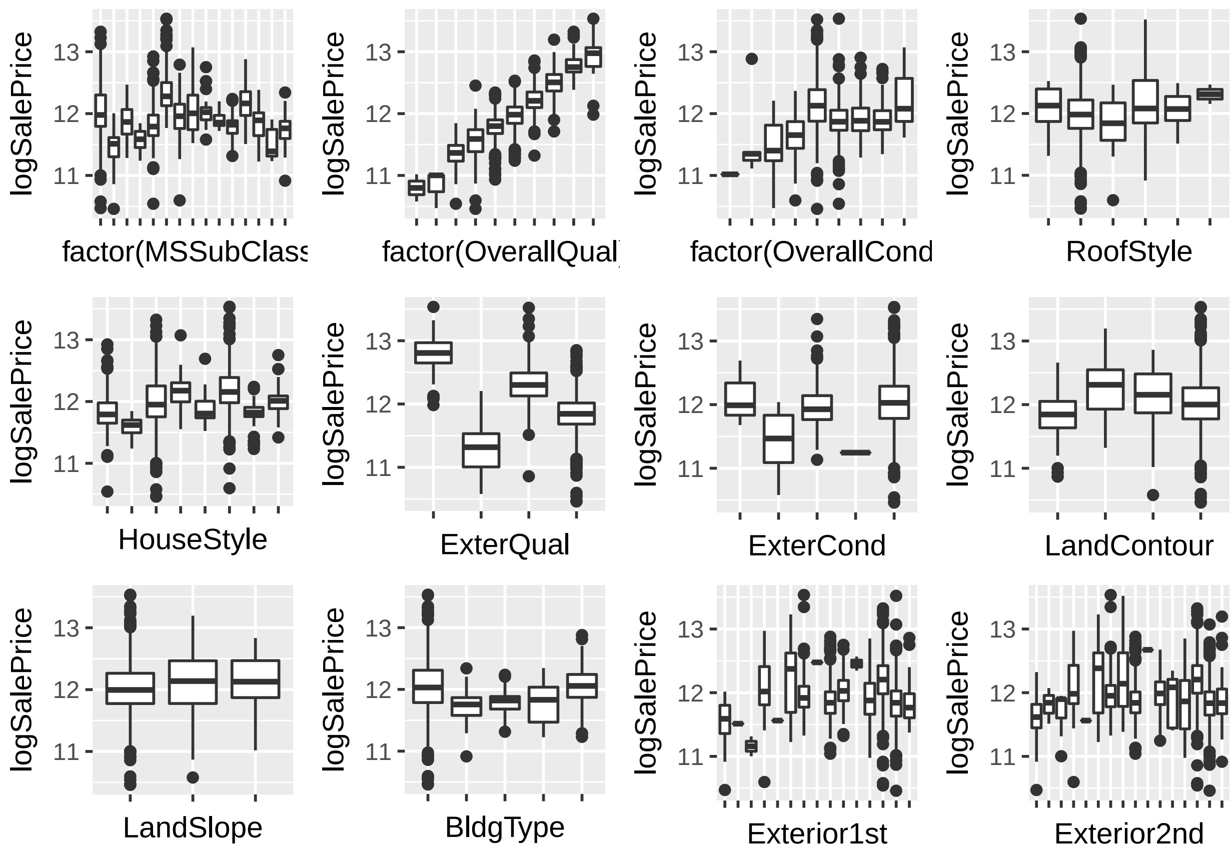
First we illustrate the relationships between various "Exterior" variables with *logSalePrice*,

```
ggplot(train_na_filled) +
  geom_point(aes(x = LotFrontage, y = logSalePrice), alpha = I(1/5)) +
  facet_wrap(~ MSSubClass, ncol = 5)
```




```
pMSSubClass <- ggplot(train_na_filled) +
  geom_boxplot(aes(x = factor(MSSubClass), y = logSalePrice)) +
  theme(axis.text.x = element_blank())
pOverallQu <- ggplot(train_na_filled) +
  geom_boxplot(aes(x = factor(OverallQual), y = logSalePrice)) +
  theme(axis.text.x = element_blank())
pOverallConv <- ggplot(train_na_filled) +
  geom_boxplot(aes(x = factor(OverallCond), y = logSalePrice)) +
  theme(axis.text.x = element_blank())
pRfStl <- ggplot(train_na_filled) +
  geom_boxplot(aes(x = RoofStyle, y = logSalePrice)) +
  theme(axis.text.x = element_blank())
pHsStl <- ggplot(train_na_filled) +
  geom_boxplot(aes(x = HouseStyle, y = logSalePrice)) +
  theme(axis.text.x = element_blank())
pExQu <- ggplot(train_na_filled) +
  geom_boxplot(aes(x = ExterQual, y = logSalePrice)) +
  theme(axis.text.x = element_blank())
pExCond <- ggplot(train_na_filled) +
  geom_boxplot(aes(x = ExterCond, y = logSalePrice)) +
  theme(axis.text.x = element_blank())
pEx1 <- ggplot(train_na_filled) +
  geom_boxplot(aes(x = Exterior1st, y = logSalePrice)) +
  theme(axis.text.x = element_blank())
pEx2 <- ggplot(train_na_filled) +
  geom_boxplot(aes(x = Exterior2nd, y = logSalePrice)) +
```

```
  theme(axis.text.x = element_blank())
pLdConto <- ggplot(train_na_filled) +
  geom_boxplot(aes(x = LandContour, y = logSalePrice)) +
  theme(axis.text.x = element_blank())
pLdSlp <- ggplot(train_na_filled) +
  geom_boxplot(aes(x = LandSlope, y = logSalePrice)) +
  theme(axis.text.x = element_blank())
pBldgTp <- ggplot(train_na_filled) +
  geom_boxplot(aes(x = BldgType, y = logSalePrice)) +
  theme(axis.text.x = element_blank())
grid.arrange(pMSSubClass, pOverallQu, pOverallConv, pRfStl,
             pHsStl, pExQu, pExCond, pLdConto, pLdSlp, pBldgTp, pEx1,
             pEx2, ncol = 4)
```



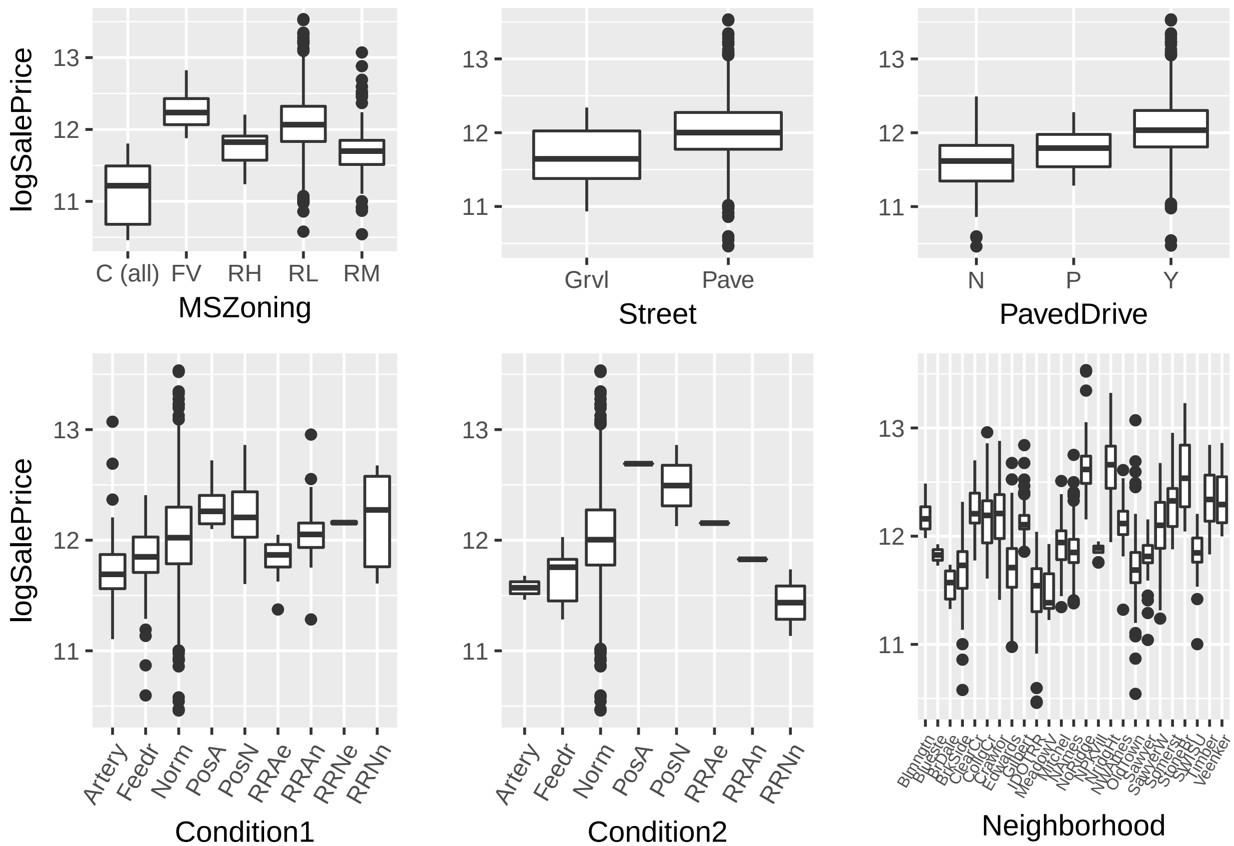
Almost all boxplots show the differences of distribution of *logSalePrice* on different levels of "Exterior" variables. We can include all of them into models, but this might lead to collinearity.

(5) Location and transporation

Next we consider the location and transporation condition of house. Again, we use boxplots to display the data.

```
pMSzoning <- ggplot(train_na_filled) +
  geom_boxplot(aes(x = MSZoning, y = logSalePrice))
pNeighbor <- ggplot(train_na_filled) +
  geom_boxplot(aes(x = Neighborhood, y = logSalePrice)) +
```

```
  theme(axis.text.x = element_text(angle = 60, hjust = 1, size = 5.5)) +
  ylab('')
pStreet <- ggplot(train_na_filled) +
  geom_boxplot(aes(x = Street, y = logSalePrice)) +
  ylab('')
pPDrive <- ggplot(train_na_filled) +
  geom_boxplot(aes(x = PavedDrive, y = logSalePrice)) +
  ylab('')
pCond1 <- ggplot(train_na_filled) +
  geom_boxplot(aes(x = Condition1, y = logSalePrice)) +
  theme(axis.text.x = element_text(angle = 60, hjust = 1))
pCond2 <- ggplot(train_na_filled) +
  geom_boxplot(aes(x = Condition2, y = logSalePrice)) +
  theme(axis.text.x = element_text(angle = 60, hjust = 1)) +
  ylab('')
grid.arrange(pMSZoning, pStreet, pPDrive,
             pCond1, pCond2, pNeighbor, nrow = 2,
             heights = c(2, 3))
```

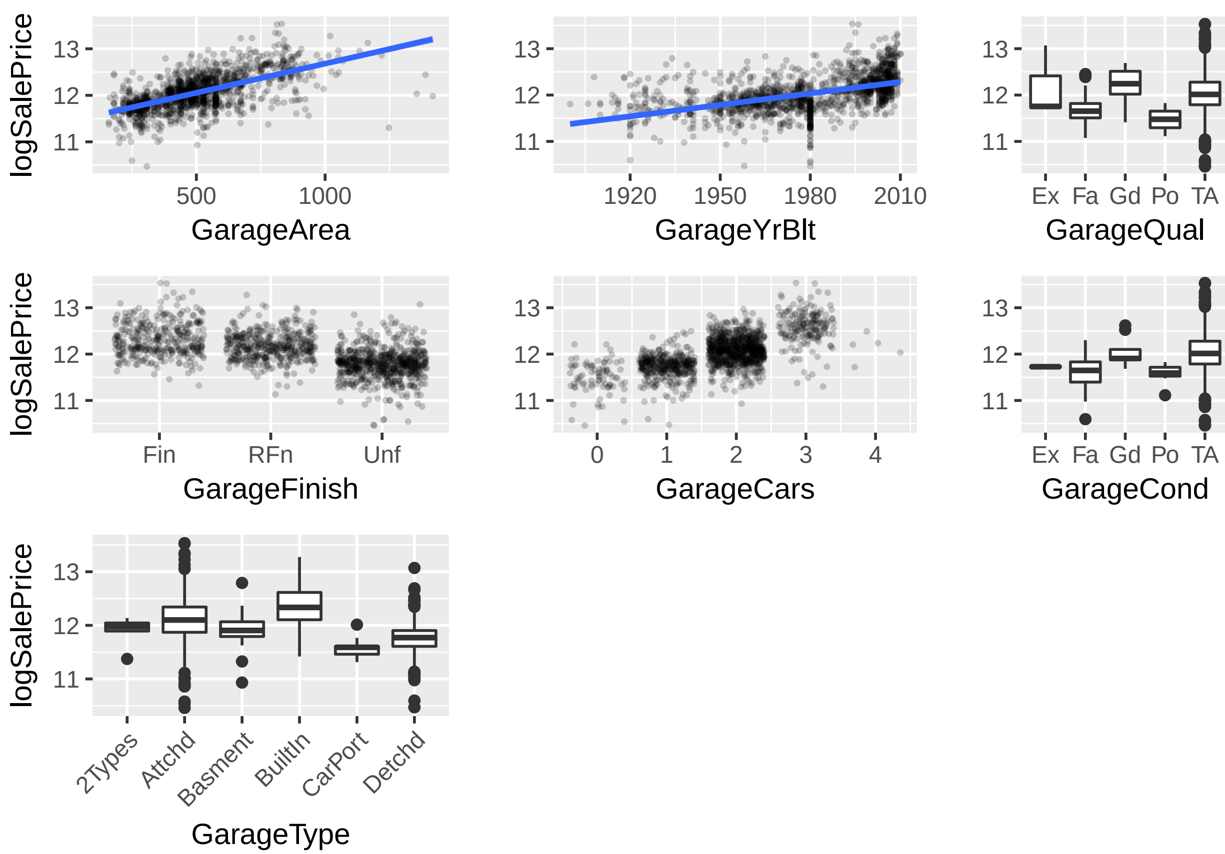


As a result we can claim that the location and transportion are highly correlate with the price of hoses.

Then check *Garage_*

```
pGargTp <- ggplot(train_na_filled) +
  geom_boxplot(aes(x = GarageType, y = logSalePrice)) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
pGargYBt <- ggplot(train_na_filled, aes(x = GarageYrBlt, y = logSalePrice)) +
```

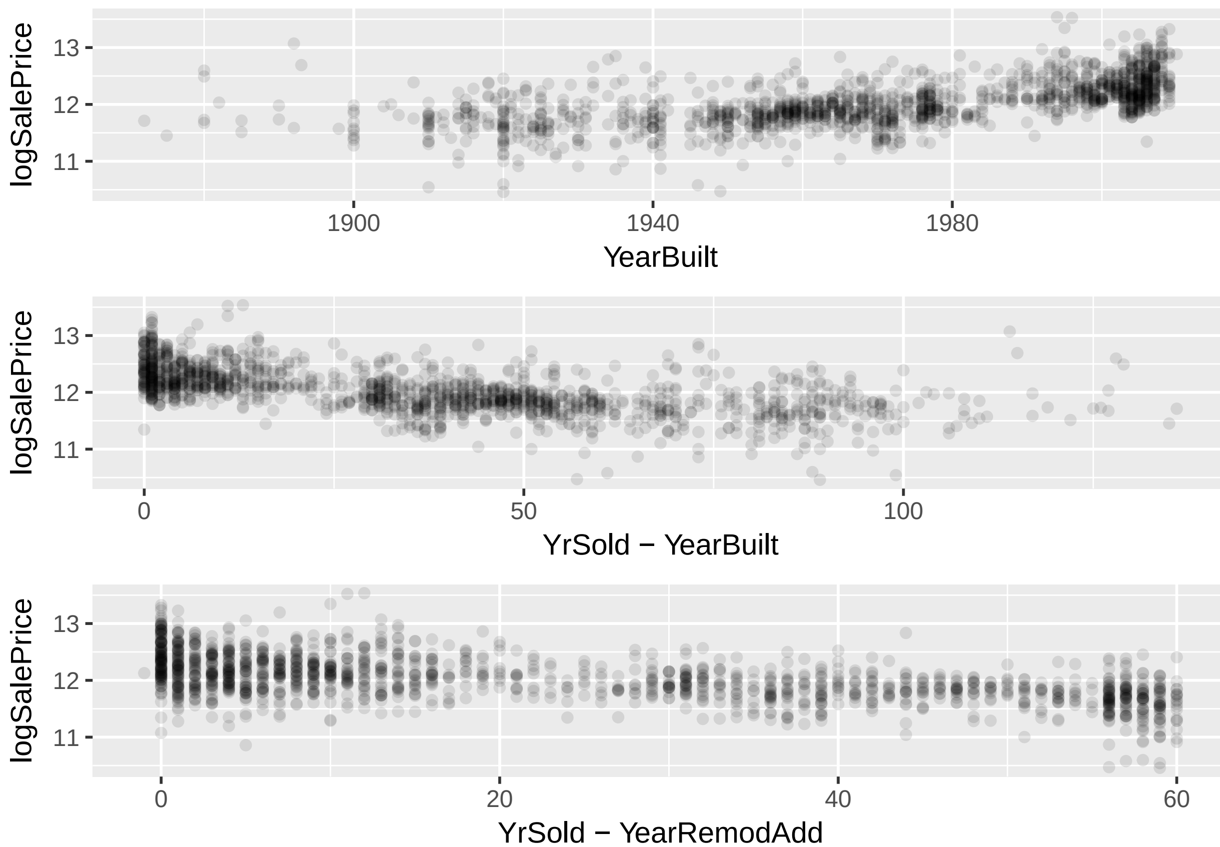
```
  geom_point(alpha = I(1/5), size = .5) +
  geom_smooth(method = 'lm') +
  ylab('')
pGargFns <- ggplot(train_na_filled, aes(x = GarageFinish, y = logSalePrice)) +
  geom_jitter(alpha = I(1/5), size = .5)
pGargCars <- ggplot(train_na_filled, aes(x = GarageCars, y = logSalePrice)) +
  geom_jitter(alpha = I(1/5), size = .5) +
  ylab('')
pGargAr <- ggplot(train_na_filled %>% filter(GarageArea > 0),
       aes(x = GarageArea, y = logSalePrice)) +
  geom_point(alpha = I(1/5), size = .5) +
  geom_smooth(method = 'lm')
pGargQu <- ggplot(train_na_filled, aes(x = GarageQual, y = logSalePrice)) +
  geom_boxplot() +
  ylab('')
pGargCond <- ggplot(train_na_filled, aes(x = GarageCond, y = logSalePrice)) +
  geom_boxplot() +
  ylab('')
grid.arrange(pGargAr, pGargYBt, pGargQu,
             pGargFns, pGargCars,
             pGargCond, pGargTp,
             nrow = 3, heights = c(3, 3, 4),
             widths = c(3, 3, 2))
```



We can see that the distribution of *logSalePrice* is similar on *GarageQual* and *GarageCond*. And generally speaking, *GarageArea* should has high positive correlation with *GarageCars*.

(6) Other attributes

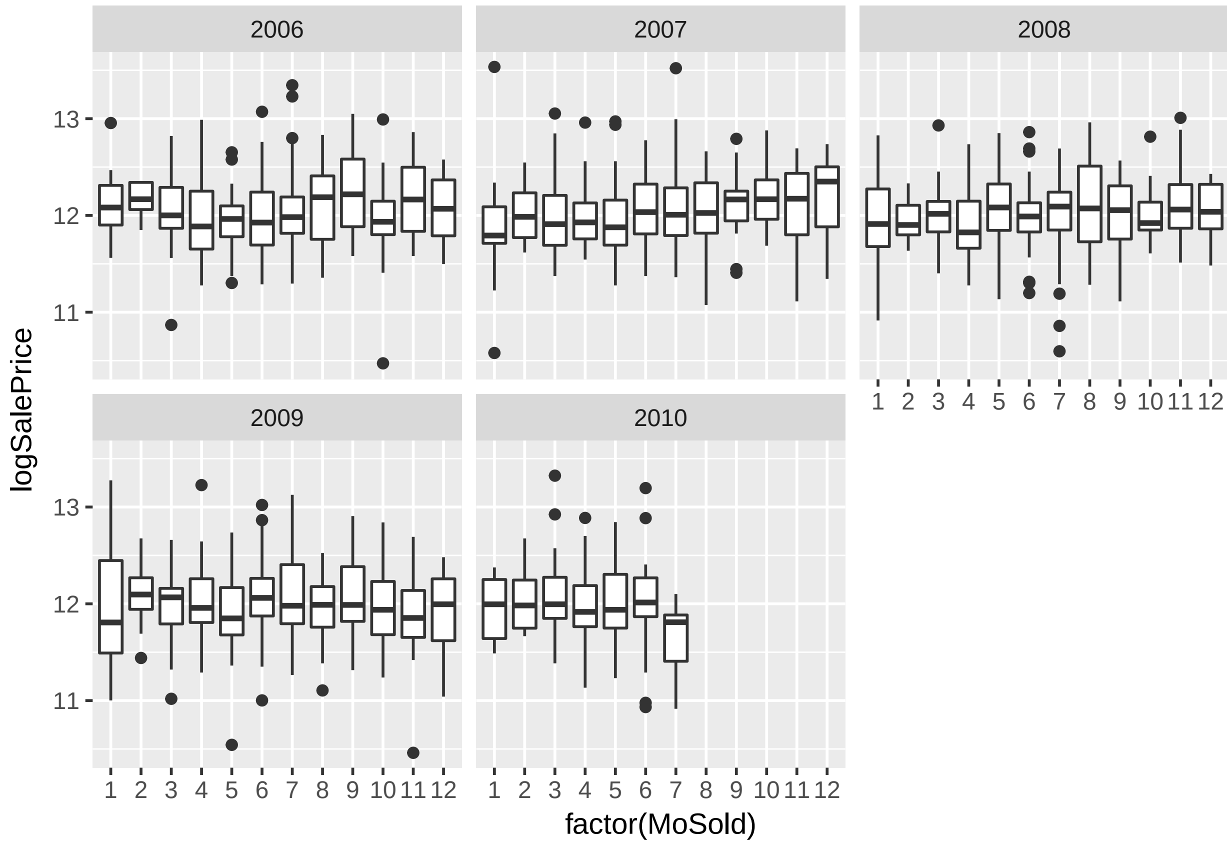
```
pYrBlt <- ggplot(train_na_filled) +
  geom_point(aes(x = YearBuilt, y = logSalePrice), alpha = I(1/10))
pHsAge <- ggplot(train_na_filled) +
  geom_point(aes(x = YrSold - YearBuilt, y = logSalePrice), alpha = I(1/10))
pRemd <- ggplot(train_na_filled, aes(x = YrSold - YearRemodAdd, y = logSalePrice)) +
  geom_point(alpha = I(1/10))
grid.arrange(pYrBlt, pHsAge, pRemd, nrow = 3, heights = c(2, 2, 2))
```



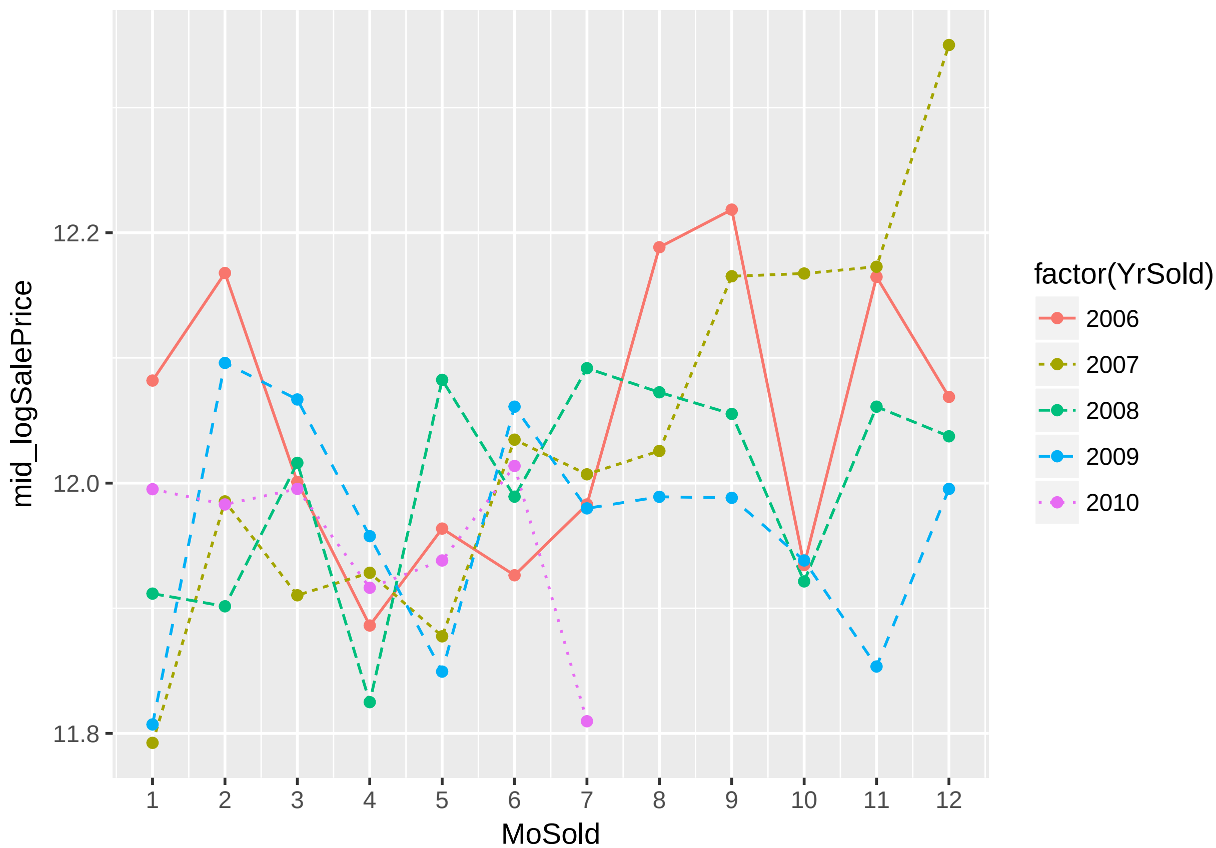
The scatter plots between *Year* attributes and *logSalePrice* express that the "Age" of house has linear correlation with its value in general, the variation of price of houses built prior to 1900 is higher.

Then we will check *MoSold* and *YrSold* to see if they related with the price of houses. We bet that it might exits seasonal trend

```
ggplot(train_na_filled) +
  geom_boxplot(aes(x = factor(MoSold), y = logSalePrice)) +
  facet_wrap(~ YrSold)
```




```r
train_na_filled %>%
  group_by(MoSold, YrSold) %>%
  summarise(mid_logSalePrice = median(logSalePrice)) %>%
  ggplot(aes(x = MoSold, y = mid_logSalePrice)) +
  geom_line(aes(group = YrSold,
                col = factor(YrSold),
                linetype = factor(YrSold))) +
  geom_point(aes(group = YrSold,
                col = factor(YrSold),
                linetype = factor(YrSold))) +
  scale_x_continuous(breaks = 1:12)
```



Also we check the relationship between *SaleType*, *SaleCondisiton* and *MoSold*.

```
options(digits = 2)
# prop of different SaleType above SaleCondition
table(train_na_filled$SaleType, train_na_filled$SaleCondition) /
  rowSums(table(train_na_filled$SaleType, train_na_filled$SaleCondition))
```

```
##
##         Abnorml AdjLand Alloca Family Normal Partial
##   COD    0.5581  0.0000 0.0000 0.0000 0.4419  0.0000
##   Con    0.0000  0.0000 0.0000 0.0000 1.0000  0.0000
##   ConLD  0.2222  0.0000 0.0000 0.0000 0.6667  0.1111
##   ConLI  0.2000  0.0000 0.0000 0.0000 0.8000  0.0000
##   ConLw  0.0000  0.0000 0.0000 0.0000 1.0000  0.0000
##   CWD    0.2500  0.0000 0.0000 0.2500 0.5000  0.0000
##   New    0.0000  0.0000 0.0000 0.0000 0.0000  1.0000
##   Oth    1.0000  0.0000 0.0000 0.0000 0.0000  0.0000
##   WD     0.0552  0.0032 0.0095 0.0150 0.9155  0.0016
```

```
# prop of different MoSold above SaleCondition
table(train_na_filled$MoSold, train_na_filled$SaleCondition) /
  rowSums(table(train_na_filled$MoSold, train_na_filled$SaleCondition))
```

```
##
##      Abnorml AdjLand Alloca Family Normal Partial
##   1   0.1207  0.0000 0.0172 0.0345 0.7414  0.0862
##   2   0.0769  0.0000 0.0000 0.0000 0.8846  0.0385
##   3   0.0755  0.0000 0.0283 0.0189 0.8113  0.0660
```

```
##    4    0.0780   0.0071 0.0000 0.0142 0.8369  0.0638
##    5    0.0588   0.0049 0.0000 0.0049 0.8824  0.0490
##    6    0.0237   0.0040 0.0198 0.0119 0.8814  0.0593
##    7    0.0684   0.0000 0.0000 0.0085 0.8291  0.0940
##    8    0.0984   0.0000 0.0082 0.0246 0.7377  0.1311
##    9    0.0476   0.0000 0.0159 0.0317 0.7302  0.1746
##    10   0.1011   0.0112 0.0000 0.0112 0.7528  0.1236
##    11   0.0886   0.0000 0.0127 0.0127 0.7595  0.1266
##    12   0.1017   0.0000 0.0000 0.0169 0.7627  0.1186
```

Saving data.

```r
save.image("C:/Users/Bangda/Desktop/kaggle/housing-price/eda.RData")
```