Housing Price

jianwen wu

4/27/2019

1. Set Up

```
library(tidyverse)
library(e1071)
library(recipes)
library(tidyquant)
library(tidyverse)
library(stringr)
library(forcats)
```

```
train_df <- read_csv("data/training.csv")

test_df <- read_csv("data/validation.csv")
```

2. Data Understanding

2.1 structure of data

```
train_df %>%
  map(
    .f = function(x){
    class(x)
  }
) %>%
  as_tibble() %>%
  gather()
```

```
## # A tibble: 81 x 2
##    key         value
##    <chr>       <chr>
##  1 Id          numeric
##  2 MSSubClass  numeric
##  3 MSZoning    character
##  4 LotFrontage numeric
##  5 LotArea     numeric
##  6 Street      character
##  7 Alley       character
##  8 LotShape    character
##  9 LandContour character
## 10 Utilities   character
## # ... with 71 more rows
```

3. Data Processing

Step 1

We know the numerical variables below are actually categorical variables. We need to transform variables below into factors.

- MSSubClass
- OverallCond
- YrSold
- MoSold

Step 2

According to the data description, NA refers to "None" for the variables below. We are going to fill NA with None for those variables

For example, PoolQC with NA means No PoolQC

- PoolQC
- MiscFeatur
- Alley
- Fence
- FireplaceQu
- GarageType
- GarageFinish
- GarageQual
- GarageCond
- BsmtQual
- BsmtCond
- BsmtExposure
- BsmtFinType1
- BsmtFinType2

Step 3

We are going to fill NA with median of LotFrotage group by Neighborhood

Step 4

We are going to fill NA with 0 for the variables below:

- GarageYrBlt
- GarageArea
- GarageCars
- BsmtFinSF1
- BsmtFinSF2
- BsmtUnfSF
- TotalBsmtSF
- BsmtFullBath
- BsmtHalfBath
- MasVnrArea

Step 5

- Electrical - only has one NA value, I am going to assign it to SBrkr. Since this variable contains 91% SBrkr.

```
data_process <- function(data){

  #combine train and test ----
  #create factor ----

  df <- data %>%
      mutate(MSSubClass = as.factor(MSSubClass),
             OverallCond = as.factor(OverallCond),
             YrSold = as.factor(YrSold),
             MoSold = as.factor(MoSold))
  # fill na with none ----
  NA_cols_None <- c("PoolQC",
                    "MiscFeature",
                    "Alley",
                    "Fence",
                    "FireplaceQu",
                    "GarageType",
                    "GarageFinish",
                    "GarageQual",
                    "GarageCond",
                    "BsmtQual",
                    "BsmtCond",
                    "BsmtExposure",
                    "BsmtFinType1",
                    "BsmtFinType2",
                    "MasVnrType",
                    "MSSubClass")

  df[,NA_cols_None][is.na(df[,NA_cols_None])] <- "None"

  # fill na with median ----
  df <- df %>%
    group_by(Neighborhood) %>%
    mutate(LotFrontage = replace_na(LotFrontage, replace = median(LotFrontage, na.rm = T))) %>%
    ungroup()


  # fill na with 0 ----
  NA_cols_0 <- (c(
    "GarageYrBlt",
    "GarageArea",
    "GarageCars",
    "BsmtFinSF1",
    "BsmtFinSF2",
    "BsmtUnfSF",
    "TotalBsmtSF",
    "BsmtFullBath",
    "BsmtHalfBath",
    "MasVnrArea"
  ))
```

```
  df[,NA_cols_0][is.na(df[,NA_cols_0])] <- 0

  #fill na with most frequently ----
  df <- df %>%
    mutate(Electrical = str_replace_na(string = Electrical, replacement = "SBrkr"))



  return(df)
}
```

```
train_df <- data_process(train_df)

#check the missing value again
train_df %>%
  map(.f = function(x){
    sum(is.na(x))
  })  %>%
  as_tibble() %>%
  gather(key = "Varible", value =  "Number_of_Missing_Value") %>%
  arrange(desc(Number_of_Missing_Value)) %>%
  mutate(Percentage = round(Number_of_Missing_Value / nrow(train_df) * 100,3))
```

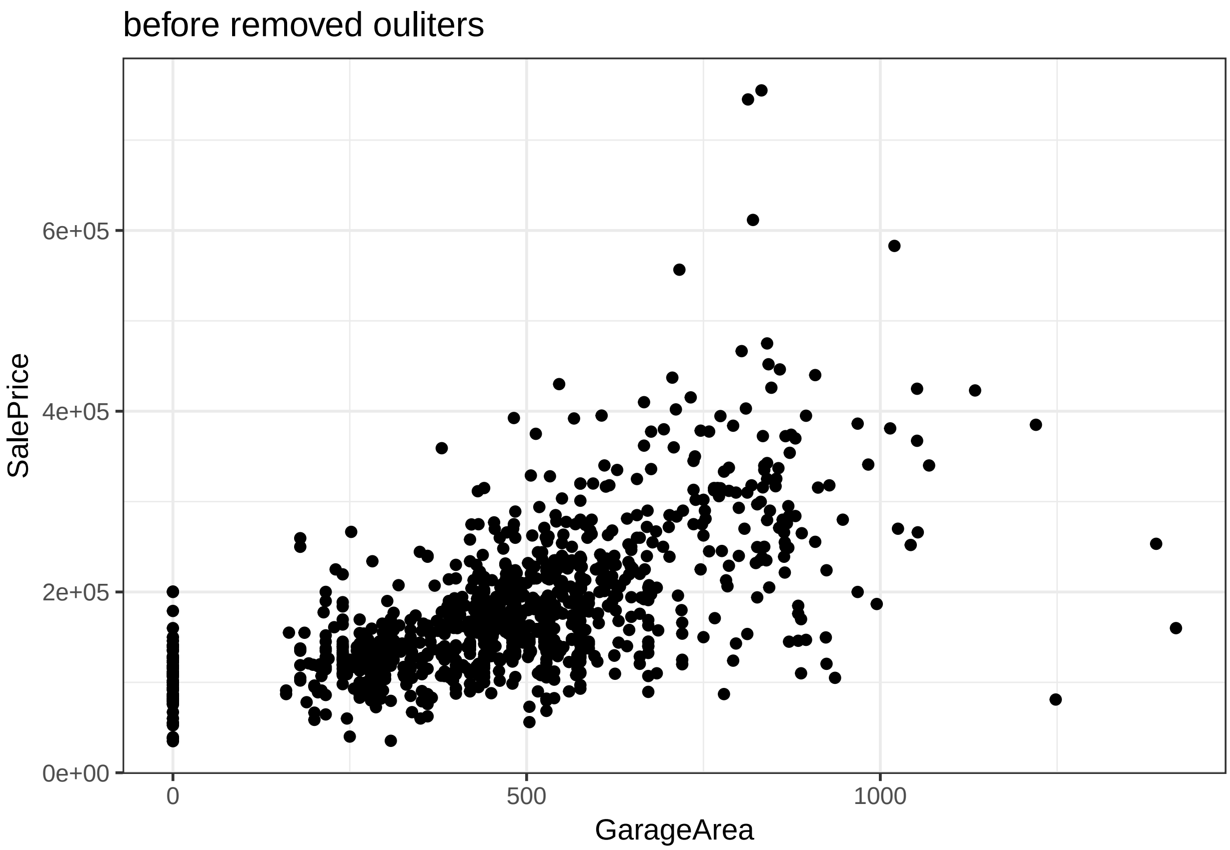
```
## # A tibble: 81 x 3
##    Varible     Number_of_Missing_Value Percentage
##    <chr>                         <int>      <dbl>
##  1 Id                                0          0
##  2 MSSubClass                        0          0
##  3 MSZoning                          0          0
##  4 LotFrontage                       0          0
##  5 LotArea                           0          0
##  6 Street                            0          0
##  7 Alley                             0          0
##  8 LotShape                          0          0
##  9 LandContour                       0          0
## 10 Utilities                         0          0
## # ... with 71 more rows
```

Step 7

we want to see are there outlier in the variable below,

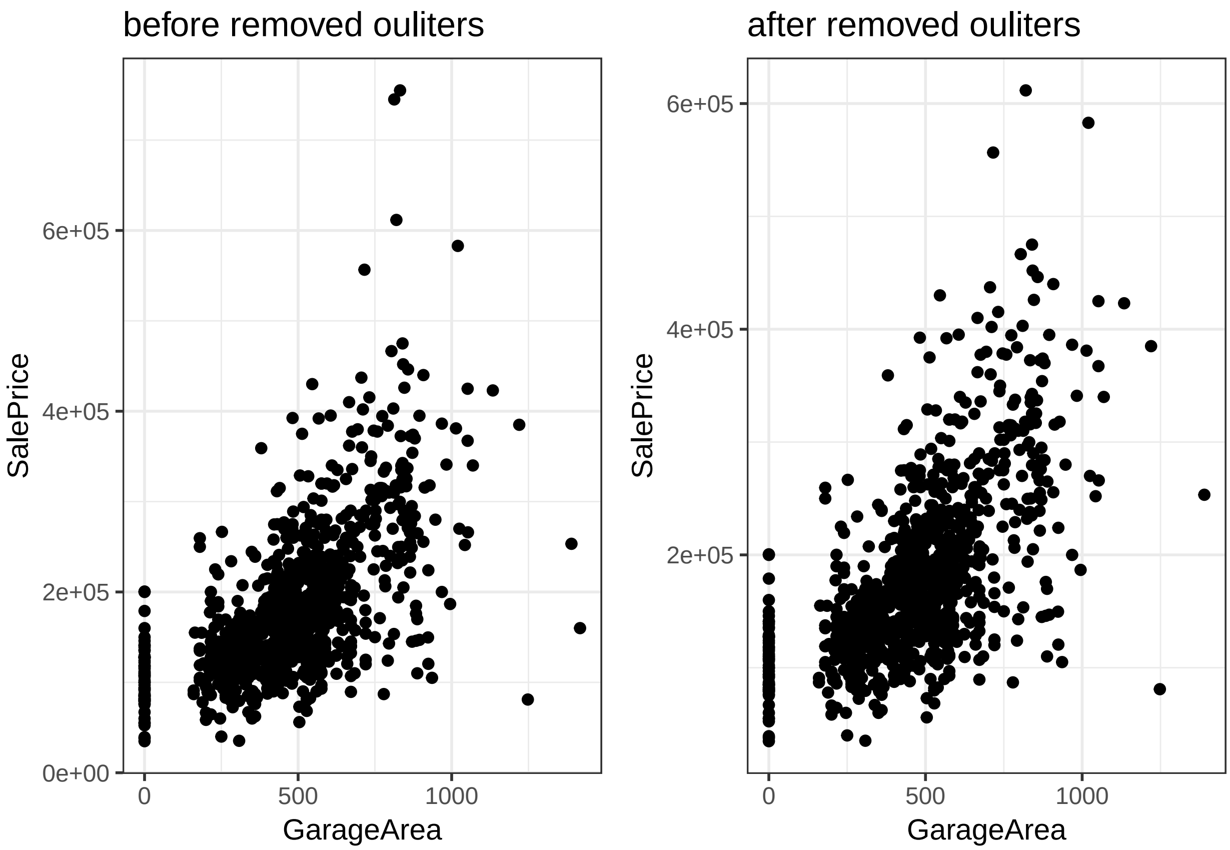
- GarageArea

```
p1_outlier <- train_df %>%
  ggplot(aes(GarageArea, SalePrice)) +
  geom_point() +
  theme_bw() +
  labs(title = "before removed ouliters")

p1_outlier
```



It seem like it's more safe to remove the outliers. Those outliers are huge and bad.

```
# removing outliers recomended by author
train_df <- train_df %>%
  filter(GrLivArea < 4000)

p2_outlier <- train_df %>%
  ggplot(aes(GarageArea, SalePrice)) +
  geom_point() +
  theme_bw() +
  labs(title = "after removed ouliters")

cowplot::plot_grid(p1_outlier, p2_outlier)
```

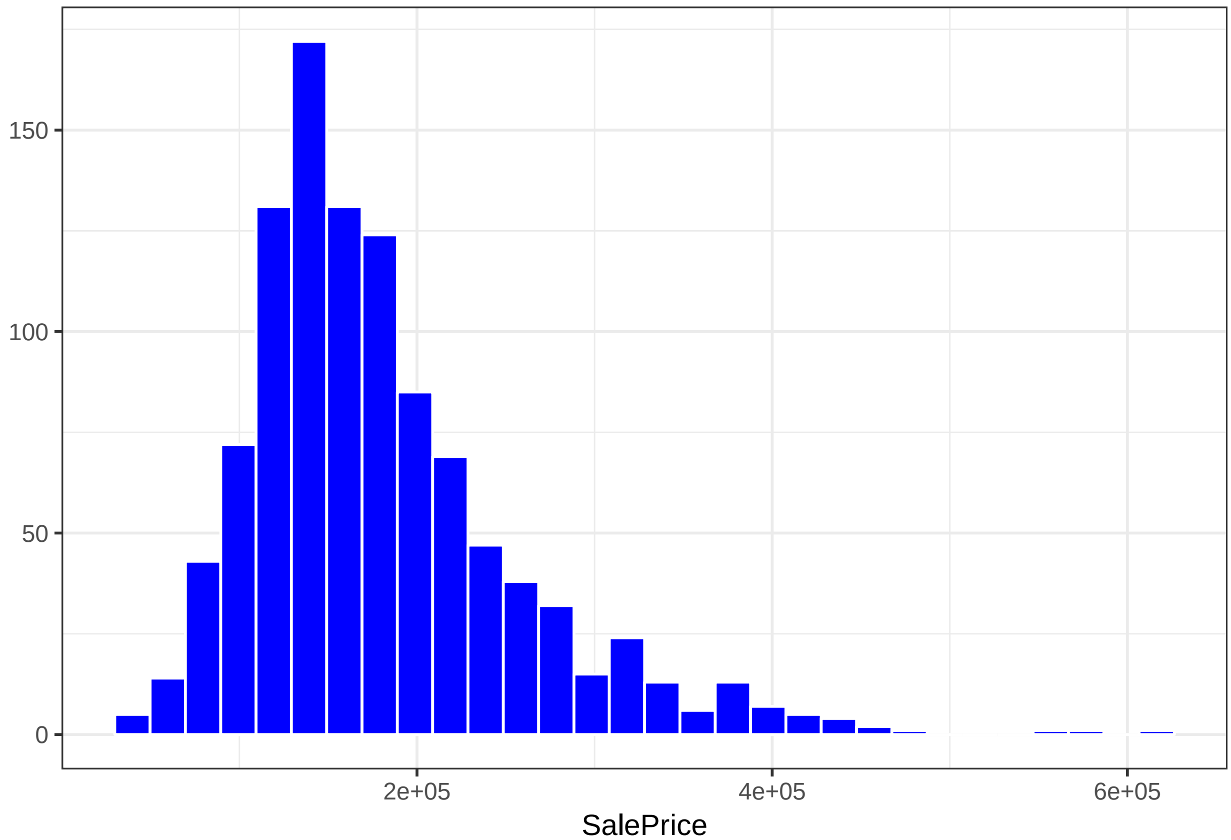


There might be more outliers in other variables, but removing all of them might affect our model. such as less observation, and there might be outliers in our test data as well. Therefore, we just removed the outlier for variable "GrlivArea".

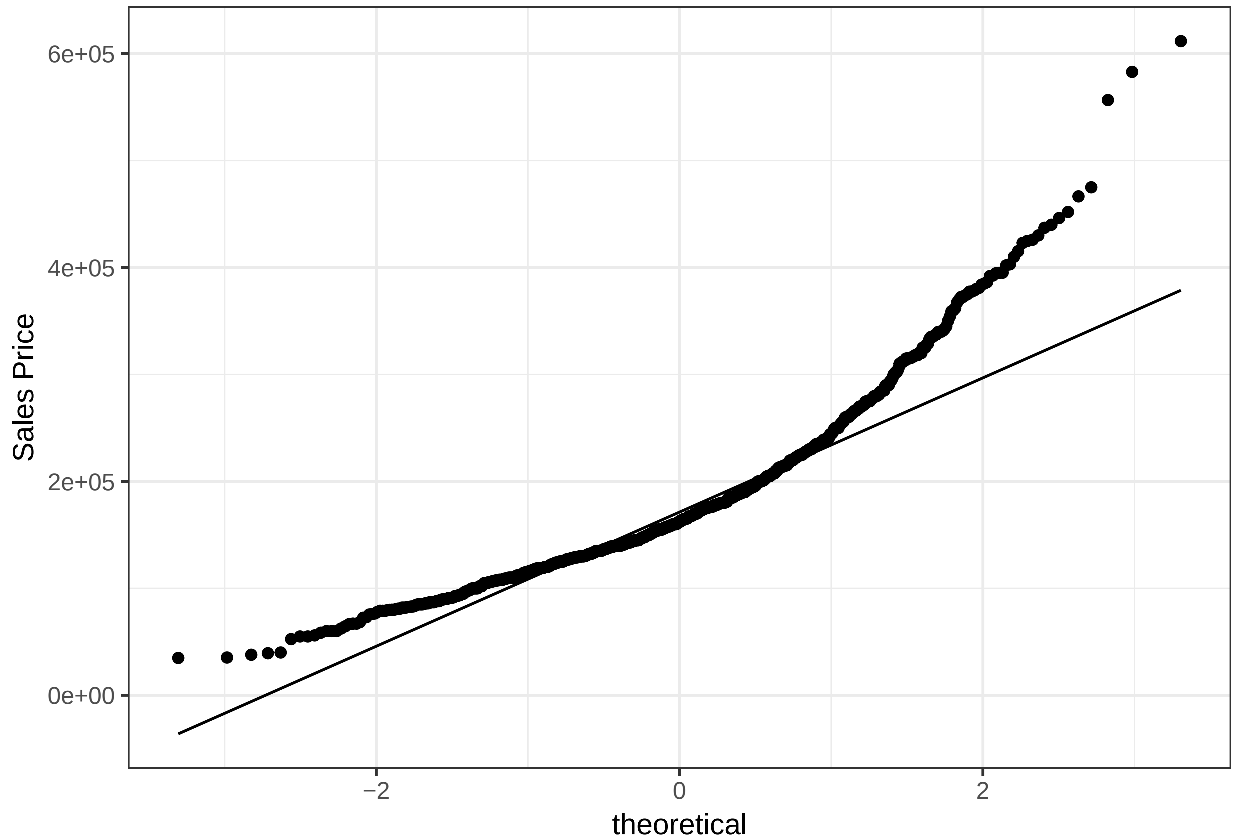
Step 8

- Skewness of Target Varible - SalePrice

```r
#Histogram of SalePrice
p1_saleprice_Hist <- train_df %>%
  ggplot(aes()) +
  geom_histogram(aes(SalePrice), fill = "blue",
                 color = "white", bins = 30)   +
  theme_bw() +
  labs(y = NULL)

p1_saleprice_Hist
```




```
#QQ plot of SalePrice

p1_saleprice_QQ  <- train_df %>%
  ggplot(aes(sample = SalePrice)) +
  stat_qq() +
  stat_qq_line() +
  theme_bw() +
  labs(y = "Sales Price")



# p1_saleprice_QQ <- car::qqPlot(train_df$SalePrice,
#             ylab = "SalesPrice")

p1_saleprice_QQ
```




```r
#Skewness of SalePrice
glue::glue("The Skewness is: {round(skewness(
           train_df$SalePrice),4)}")
```

```
## The Skewness is: 1.4251
```

The target variable SalePrice is right skewed. We need to transform the target variable SalePrice into more normally distributed.

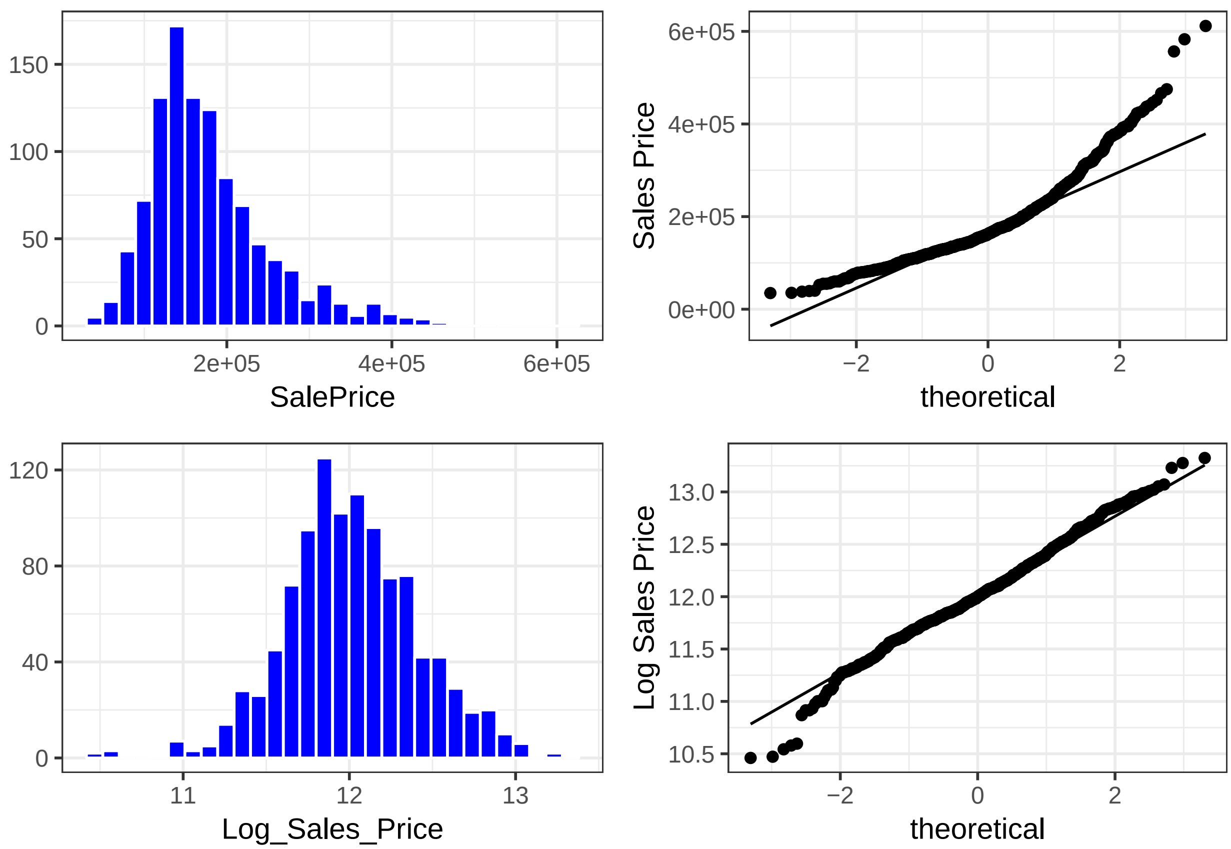
We are going to use log-transformation

```r
#Create a log SalePrice variable
train_df$Log_Sales_Price <- log(train_df$SalePrice)
train_df$Log_Sales_Price <- log(train_df$SalePrice)

#Histogram of log SalePrice

p2_log_saleprice_Hist <- train_df %>%
  ggplot(aes()) +
  geom_histogram(aes(Log_Sales_Price), fill = "blue",
                 color = "white", bins = 30)   +
  theme_bw() +
  labs(y = NULL)

#QQ plot of log SalePrice

p2_log_saleprice_QQ  <- train_df %>%
  ggplot(aes(sample = Log_Sales_Price)) +
  stat_qq() +
```

```r
  stat_qq_line() +
  theme_bw() +
  labs(y = "Log Sales Price")
# p2_saleprice_QQ <- car::qqPlot(log(train_df$Log_Sales_Price),
#               ylab = "Log SalesPrice")


#Skewness of log SalePrice
glue::glue("The Skewness is: {round(skewness(train_df$Log_Sales_Price),4)}")
```

```
## The Skewness is: -0.0337
```

```r
cowplot::plot_grid(p1_saleprice_Hist, p1_saleprice_QQ,
                   p2_log_saleprice_Hist, p2_log_saleprice_QQ,
                   ncol = 2)
```



Now, the target variable SalePrice are more normally distributed.

4.Modeling

Bar plot - Categrical Variables vs Log-SalePrice

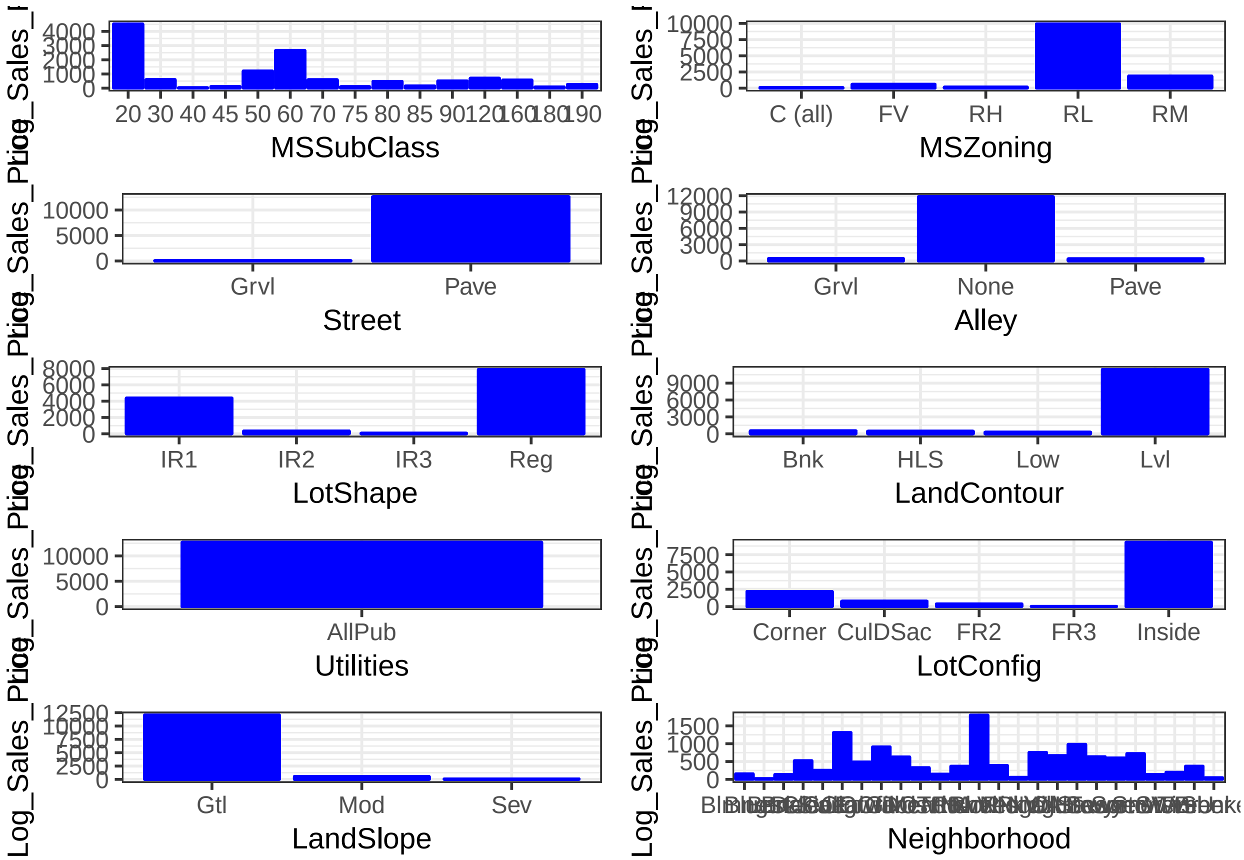
```r
df_fac <- train_df %>%
  mutate_if(is.factor, as.character) %>%
  select_if(is.character)

n <- as.list(rep("", 47))
```

```r
p <- as.list(rep("", 47))

for (i in 1:47){

  n[[i]]<- print(colnames(df_fac[i]),quote=FALSE)

  p[[i]]<- ggplot(data=train_df, aes_string (x= n[[i]], y="Log_Sales_Price")) +
    geom_bar(stat="identity", color = "blue") +
    theme_bw()
  }
```

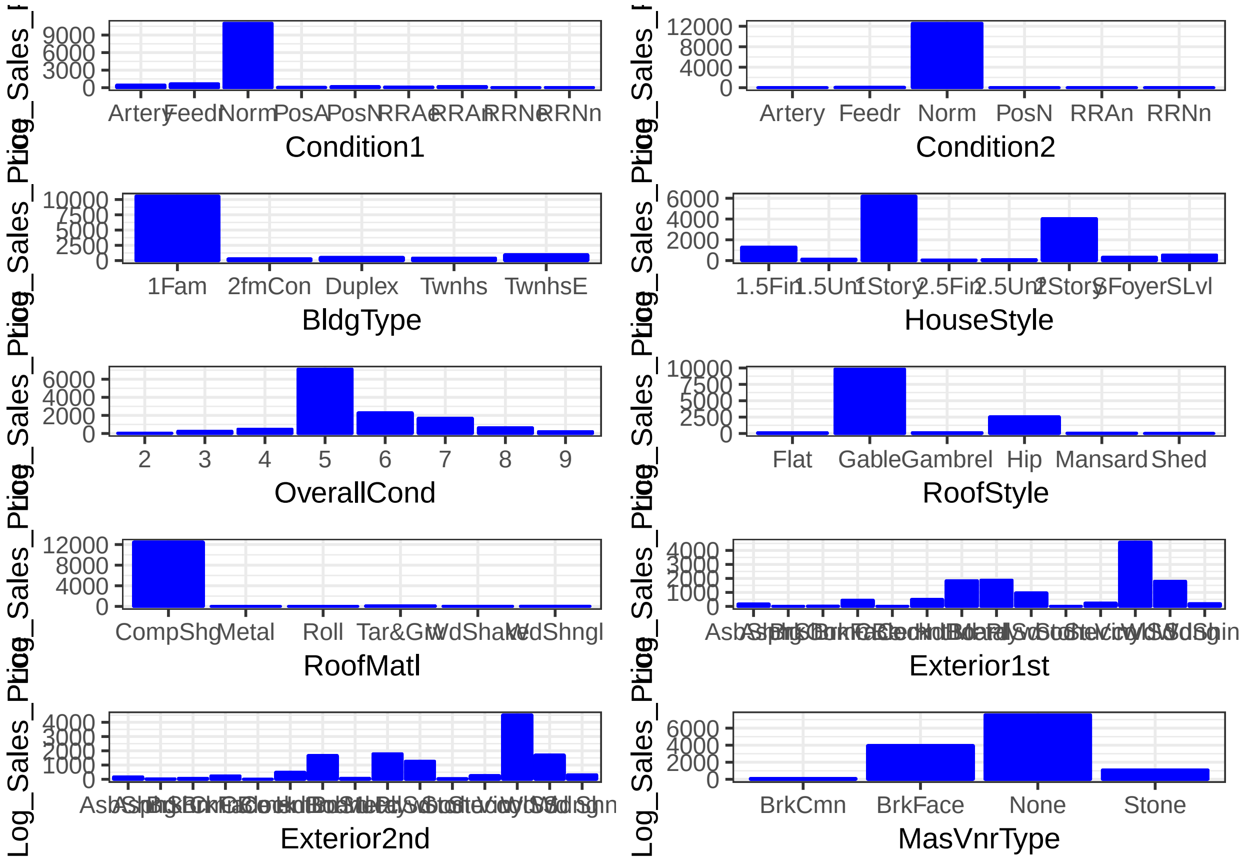
```
## [1] MSSubClass
## [1] MSZoning
## [1] Street
## [1] Alley
## [1] LotShape
## [1] LandContour
## [1] Utilities
## [1] LotConfig
## [1] LandSlope
## [1] Neighborhood
## [1] Condition1
## [1] Condition2
## [1] BldgType
## [1] HouseStyle
## [1] OverallCond
## [1] RoofStyle
## [1] RoofMatl
## [1] Exterior1st
## [1] Exterior2nd
## [1] MasVnrType
## [1] ExterQual
## [1] ExterCond
## [1] Foundation
## [1] BsmtQual
## [1] BsmtCond
## [1] BsmtExposure
## [1] BsmtFinType1
## [1] BsmtFinType2
## [1] Heating
## [1] HeatingQC
## [1] CentralAir
## [1] Electrical
## [1] KitchenQual
## [1] Functional
## [1] FireplaceQu
## [1] GarageType
## [1] GarageFinish
## [1] GarageQual
## [1] GarageCond
## [1] PavedDrive
## [1] PoolQC
## [1] Fence
## [1] MiscFeature
```

```
## [1] MoSold
## [1] YrSold
## [1] SaleType
## [1] SaleCondition
```

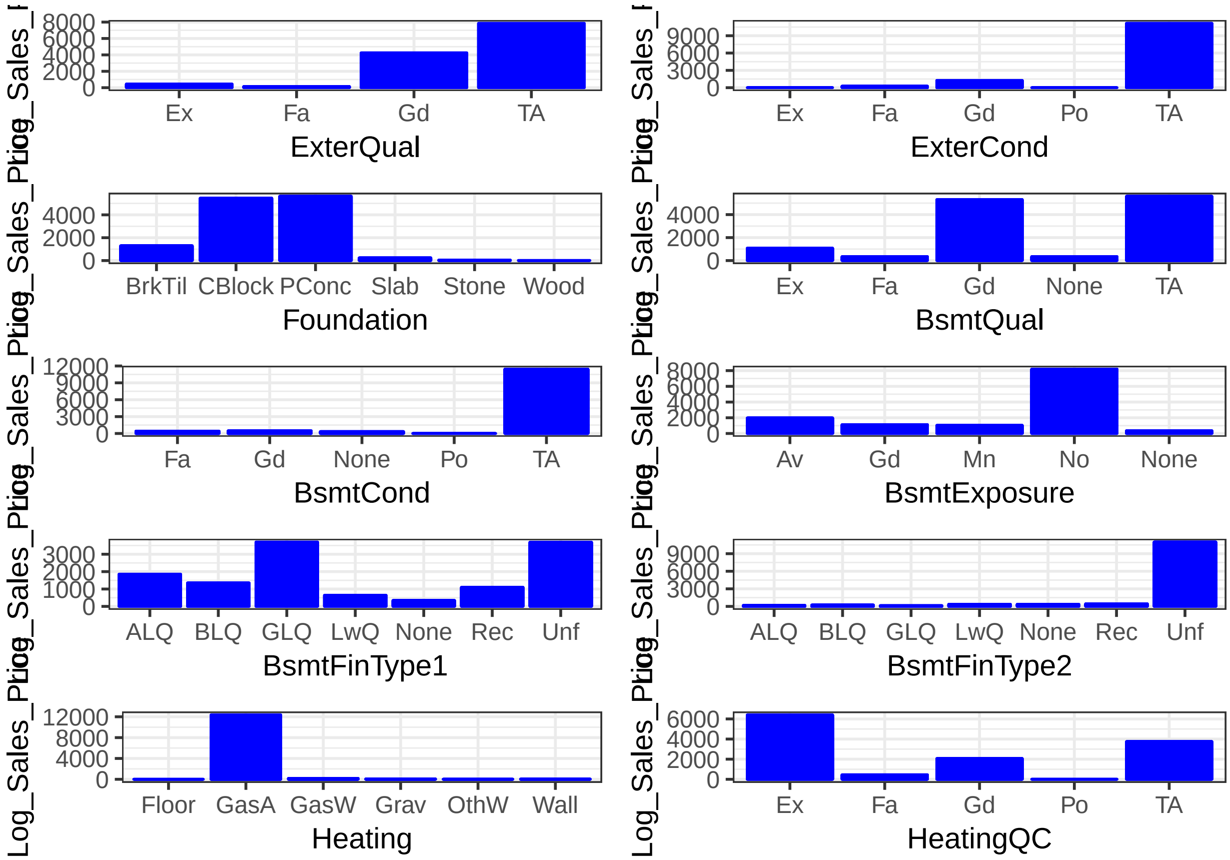
```
cowplot::plot_grid(p[[1]], p[[2]], p[[3]], p[[4]], p[[5]], p[[6]], p[[7]], p[[8]], p[[9]], p[[10]], ncol
```



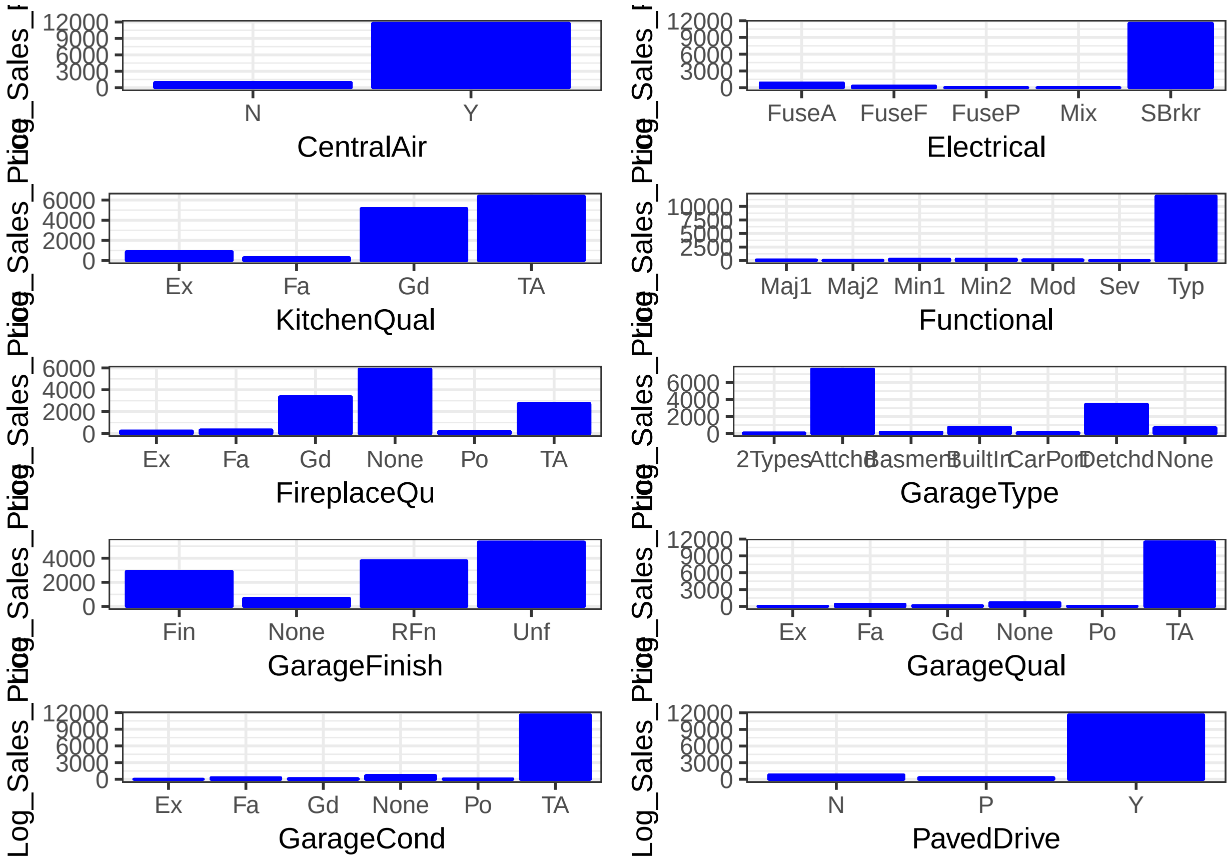

```
cowplot::plot_grid(p[[11]], p[[12]], p[[13]], p[[14]],p[[15]], p[[16]], p[[17]], p[[18]], p[[19]], p[[2
                   ncol = 2)
```



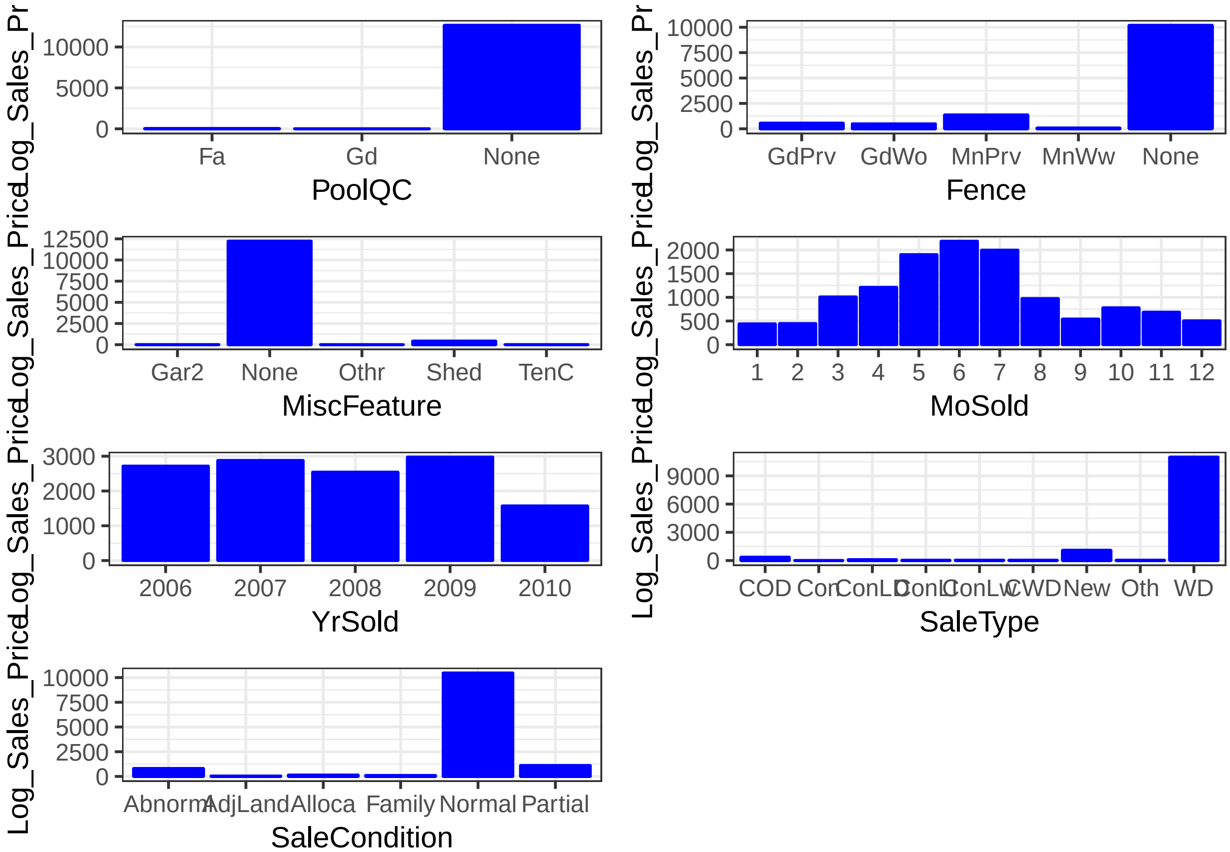

```
cowplot::plot_grid(p[[21]], p[[22]], p[[23]], p[[24]],p[[25]], p[[26]], p[[27]], p[[28]], p[[29]], p[[3
```




```
cowplot::plot_grid(p[[31]], p[[32]], p[[33]], p[[34]],p[[35]], p[[36]], p[[37]], p[[38]], p[[39]], p[[40
```




```
cowplot::plot_grid(p[[41]], p[[42]], p[[43]], p[[44]], p[[45]], p[[46]], p[[47]], ncol = 2)
```

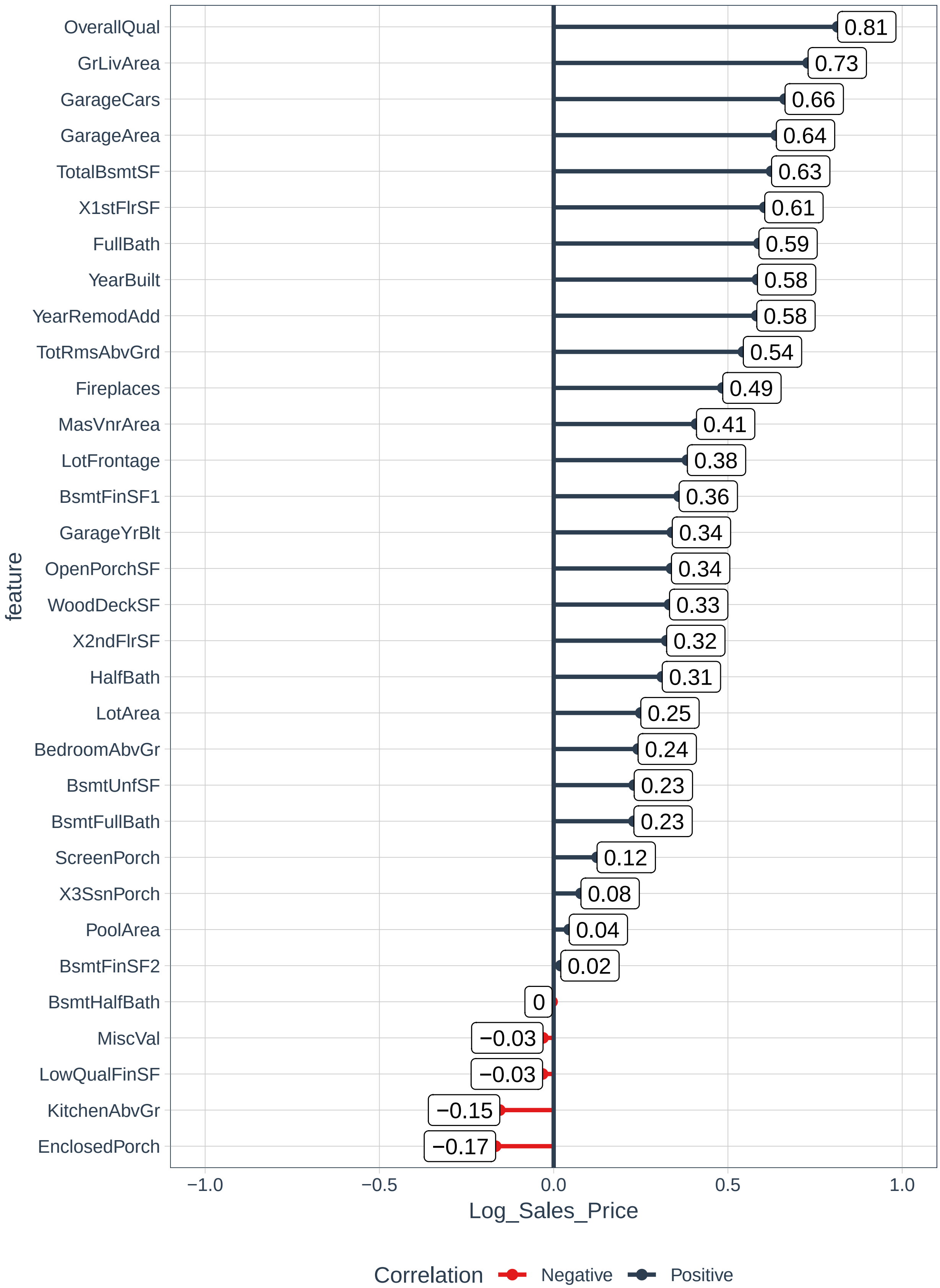


Correlation for Numerical Variables

```
get_cor <- function(data, target, use = "pairwise.complete.obs",
                    fct_reorder = FALSE, fct_rev = FALSE) {

    feature_expr <- enquo(target)
    feature_name <- quo_name(feature_expr)

    data_cor <- data %>%
        mutate_if(is.character, as.factor) %>%
        mutate_if(is.factor, as.numeric) %>%
        cor(use = use) %>%
        as.tibble() %>%
        mutate(feature = names(.)) %>%
        select(feature, !! feature_expr) %>%
        filter(!(feature == feature_name)) %>%
        mutate_if(is.character, as_factor)

    if (fct_reorder) {
        data_cor <- data_cor %>%
            mutate(feature = fct_reorder(feature, !! feature_expr)) %>%
            arrange(feature)
    }

    if (fct_rev) {
        data_cor <- data_cor %>%
```

```
            mutate(feature = fct_rev(feature)) %>%
            arrange(feature)
    }

    return(data_cor)

}

plot_cor <- function(data, target, fct_reorder = FALSE, fct_rev = FALSE,
                     include_lbl = TRUE, lbl_precision = 2, lbl_position = "outward",
                     size = 2, line_size = 1, vert_size = 1,
                     color_pos = palette_light()[[1]], color_neg = palette_light()[[2]]) {

    feature_expr <- enquo(target)
    feature_name <- quo_name(feature_expr)

    data_cor <- data %>%
        get_cor(!! feature_expr, fct_reorder = fct_reorder, fct_rev = fct_rev) %>%
        mutate(feature_name_text = round(!! feature_expr, lbl_precision)) %>%
        mutate(Correlation = case_when(
            (!! feature_expr) >= 0 ~ "Positive",
            TRUE                   ~ "Negative") %>% as.factor())

    g <- data_cor %>%
        ggplot(aes_string(x = feature_name, y = "feature", group = "feature")) +
        geom_point(aes(color = Correlation), size = size) +
        geom_segment(aes(xend = 0, yend = feature, color = Correlation), size = line_size) +
        geom_vline(xintercept = 0, color = palette_light()[[1]], size = vert_size) +
        expand_limits(x = c(-1, 1)) +
        theme_tq() +
        scale_color_manual(values = c(color_neg, color_pos))

    if (include_lbl) g <- g + geom_label(aes(label = feature_name_text), hjust = lbl_position)

    return(g)

}

train_df %>%
  select_if(is.numeric) %>%
  select(-Id, -SalePrice) %>%
  plot_cor(Log_Sales_Price,fct_reorder = T)
```

```
## Warning: `as.tibble()` is deprecated, use `as_tibble()` (but mind the new semantics).
## This warning is displayed once per session.
```



From the Numerical Variables, we are going to choose any numerical variables have have positive correlation with saleprice more than 0.5, and negative correlation with saleprice less than -0.50.

```
## # A tibble: 6 x 2
##   feature     Log_Sales_Price
##   <fct>                 <dbl>
## 1 OverallQual           0.815
## 2 GrLivArea             0.730
## 3 GarageCars            0.664
## 4 GarageArea            0.639
## 5 TotalBsmtSF           0.625
## 6 X1stFlrSF             0.606
```

- OverallQual
- GrLivArea
- GarageCars
- GarageArea
- TotalBsmtSF
- X1stFlrSF
- FullBath
- YearBuilt

```r
train_df %>%
  pull(Utilities) %>%
  table
```

```
## .
## AllPub
##   1056
```

```r
#we should remove Utilities
rec_obj <- recipe(Log_Sales_Price ~ . , data = train_df %>%
                    select(-Id)) %>%
  step_zv(all_predictors()) %>%
  step_scale(all_numeric()) %>%
  step_center(all_numeric()) %>%
  prep()


train_model <- bake(rec_obj, new_data = train_df)



lm_fit <- lm(Log_Sales_Price ~ .,data = train_df %>%
               mutate_if(is.character, as.factor) %>%
               select(-Utilities, -SalePrice, -Id)

               )
```

Variables Selections

```r
library(olsrr)
ols_step_forward_p(model = lm_fit, data  = train_df %>%
                mutate_if(is.character, as.factor) %>%
                select(-Utilities, -SalePrice, -Id), detail = T, penter = 0.05)

ols_step_forward_aic(model = lm_fit, data  = train_df %>%
```

```
                mutate_if(is.character, as.factor) %>%
                select(-Utilities, -SalePrice, -Id), detail = T)
```