HarvardX: PH125.9x Data Science: Capstone - "Choose Your Own" Project: Predicting house prices

Martin Weihrauch

11 April, 2019

Contents

1 Introduction

This is my project submission to satisfy the *HarvardX: PH125.9x Data Science: Capstone - "Choose Your Own" Project* course requirements to obtain a verified *Professional Certificate in Data Science* from HarvardX on the *edX* MOOC online platform. In order to satisfy the course requirements I had to choose my own data science project, obtain a dataset, come up with an analysis goal, do exploratory data analysis and dataset wrangling, build a machine learning algorithm and validate it. In order to practice my data analysis and wrangling skills, I chose a dataset that is available on *Kaggle*, a well-known, crowd-sourced online platform dedicated to train and challenge data scientists from all around the world to solve data science and machine learning problems. On *Kaggle* I chose the *House Prices: Advanced Regression Techniques* challenge, which comes with the following short 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."

The challenge of this dataset thus comes from the large amout of variables to explore, analyse, wrangle, and ultimately use to build a machine learning algorithm to predict house prices, as in no way are all the variables perfectly prepared and "clean". Upon exploratory data analysis it quickly becomes clear that there are plenty of missing values to deal with and collinearity (a correlation between predictor variables) of different variables to avoid.

I therefore chose to conduct a thorough exploratory data analysis and to systematically analyze each and every explanatory variable ("feature") of the dataset. After much exploration and data wrangling I then

started to develop my machine learning algoritm with the help of the `caret` package. After having established a few baseline models, I continued to tune the various hyperparameters of my algorithm, which I accompanied with visualizations. My project report culminates with my final tuned algorithm predicting the house prices of the `test` subset, writing a submission file for score submission at *Kaggle* and reporting the final results obtained. I end with a discussion and conclusion of my experience throughout this self-chosen project.

Dataset Preparation The house prices dataset can be found on *Kaggle* or, alternatively, in my **Github repository**. We will work with the `test.csv` and `train.csv` files. There is also a `data_description.txt` with descriptions about all the different parameters in the dataset. The desired outcome variable "SalePrice" is not present in `test`.

2 Exploratory data analysis and dataset manipulations

We begin by inspecting some basic properties of the dataset. The desired outcome column is named "SalePrice" and denotes a houses' sale price in dollars. In the train subset we are dealing with 1460 different houses and 81 different features of them (including their Id), such as the year they were built in or their overall condition. We also notice that several features contain missing values.

```
dplyr::glimpse(train)
## Observations: 1,460
## Variables: 81
## $ Id            <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 1...
## $ MSSubClass    <int> 60, 20, 60, 70, 60, 50, 20, 60, 50, 190, 20, 60,...
## $ MSZoning      <fct> RL, RL, RL, RL, RL, RL, RL, RL, RM, RL, RL, RL, ...
## $ LotFrontage   <int> 65, 80, 68, 60, 84, 85, 75, NA, 51, 50, 70, 85, ...
## $ LotArea       <int> 8450, 9600, 11250, 9550, 14260, 14115, 10084, 10...
## $ Street        <fct> Pave, Pave, Pave, Pave, Pave, Pave, Pave, Pave, ...
## $ Alley         <fct> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ...
## $ LotShape      <fct> Reg, Reg, IR1, IR1, IR1, IR1, Reg, IR1, Reg, Reg...
## $ LandContour   <fct> Lvl, Lvl, Lvl, Lvl, Lvl, Lvl, Lvl, Lvl, Lvl, Lvl...
## $ Utilities     <fct> AllPub, AllPub, AllPub, AllPub, AllPub, AllPub, ...
## $ LotConfig     <fct> Inside, FR2, Inside, Corner, FR2, Inside, Inside...
## $ LandSlope     <fct> Gtl, Gtl, Gtl, Gtl, Gtl, Gtl, Gtl, Gtl, Gtl, Gtl...
## $ Neighborhood  <fct> CollgCr, Veenker, CollgCr, Crawfor, NoRidge, Mit...
## $ Condition1    <fct> Norm, Feedr, Norm, Norm, Norm, Norm, Norm, PosN,...
## $ Condition2    <fct> Norm, Norm, Norm, Norm, Norm, Norm, Norm, Norm, ...
## $ BldgType      <fct> 1Fam, 1Fam, 1Fam, 1Fam, 1Fam, 1Fam, 1Fam, 1Fam, ...
## $ HouseStyle    <fct> 2Story, 1Story, 2Story, 2Story, 2Story, 1.5Fin, ...
## $ OverallQual   <int> 7, 6, 7, 7, 8, 5, 8, 7, 7, 5, 5, 9, 5, 7, 6, 7, ...
## $ OverallCond   <int> 5, 8, 5, 5, 5, 5, 5, 6, 5, 6, 5, 5, 6, 5, 5, 8, ...
## $ YearBuilt     <int> 2003, 1976, 2001, 1915, 2000, 1993, 2004, 1973, ...
## $ YearRemodAdd  <int> 2003, 1976, 2002, 1970, 2000, 1995, 2005, 1973, ...
## $ RoofStyle     <fct> Gable, Gable, Gable, Gable, Gable, Gable, Gable,...
## $ RoofMatl      <fct> CompShg, CompShg, CompShg, CompShg, CompShg, Com...
## $ Exterior1st   <fct> VinylSd, MetalSd, VinylSd, Wd Sdng, VinylSd, Vin...
## $ Exterior2nd   <fct> VinylSd, MetalSd, VinylSd, Wd Shng, VinylSd, Vin...
## $ MasVnrType    <fct> BrkFace, None, BrkFace, None, BrkFace, None, Sto...
## $ MasVnrArea    <int> 196, 0, 162, 0, 350, 0, 186, 240, 0, 0, 0, 286, ...
## $ ExterQual     <fct> Gd, TA, Gd, TA, Gd, TA, Gd, TA, TA, TA, TA, Ex, ...
## $ ExterCond     <fct> TA, TA, TA, TA, TA, TA, TA, TA, TA, TA, TA, TA, ...
## $ Foundation    <fct> PConc, CBlock, PConc, BrkTil, PConc, Wood, PConc...
## $ BsmtQual      <fct> Gd, Gd, Gd, TA, Gd, Gd, Ex, Gd, TA, TA, TA, Ex, ...
## $ BsmtCond      <fct> TA, TA, TA, Gd, TA, TA, TA, TA, TA, TA, TA, TA, ...
## $ BsmtExposure  <fct> No, Gd, Mn, No, Av, No, Av, Mn, No, No, No, No, ...
## $ BsmtFinType1  <fct> GLQ, ALQ, GLQ, ALQ, GLQ, GLQ, GLQ, ALQ, Unf, GLQ...
## $ BsmtFinSF1    <int> 706, 978, 486, 216, 655, 732, 1369, 859, 0, 851,...
## $ BsmtFinType2  <fct> Unf, Unf, Unf, Unf, Unf, Unf, Unf, BLQ, Unf, Unf...
## $ BsmtFinSF2    <int> 0, 0, 0, 0, 0, 0, 0, 32, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ BsmtUnfSF     <int> 150, 284, 434, 540, 490, 64, 317, 216, 952, 140,...
## $ TotalBsmtSF   <int> 856, 1262, 920, 756, 1145, 796, 1686, 1107, 952,...
## $ Heating       <fct> GasA, GasA, GasA, GasA, GasA, GasA, GasA, GasA, ...
## $ HeatingQC     <fct> Ex, Ex, Ex, Gd, Ex, Ex, Ex, Ex, Gd, Ex, Ex, Ex, ...
## $ CentralAir    <fct> Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, ...
## $ Electrical    <fct> SBrkr, SBrkr, SBrkr, SBrkr, SBrkr, SBrkr, SBrkr,...
## $ X1stFlrSF     <int> 856, 1262, 920, 961, 1145, 796, 1694, 1107, 1022...
```

```
## $ X2ndFlrSF     <int> 854, 0, 866, 756, 1053, 566, 0, 983, 752, 0, 0, ...
## $ LowQualFinSF  <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ GrLivArea     <int> 1710, 1262, 1786, 1717, 2198, 1362, 1694, 2090, ...
## $ BsmtFullBath  <int> 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, ...
## $ BsmtHalfBath  <int> 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ FullBath      <int> 2, 2, 2, 1, 2, 1, 2, 2, 2, 1, 1, 3, 1, 2, 1, 1, ...
## $ HalfBath      <int> 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, ...
## $ BedroomAbvGr  <int> 3, 3, 3, 3, 4, 1, 3, 3, 2, 2, 3, 4, 2, 3, 2, 2, ...
## $ KitchenAbvGr  <int> 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, ...
## $ KitchenQual   <fct> Gd, TA, Gd, Gd, Gd, TA, Gd, TA, TA, TA, TA, Ex, ...
## $ TotRmsAbvGrd  <int> 8, 6, 6, 7, 9, 5, 7, 7, 8, 5, 5, 11, 4, 7, 5, 5,...
## $ Functional    <fct> Typ, Typ, Typ, Typ, Typ, Typ, Typ, Typ, Min1, Ty...
## $ Fireplaces    <int> 0, 1, 1, 1, 1, 0, 1, 2, 2, 2, 0, 2, 0, 1, 1, 0, ...
## $ FireplaceQu   <fct> NA, TA, TA, Gd, TA, NA, Gd, TA, TA, TA, NA, Gd, ...
## $ GarageType    <fct> Attchd, Attchd, Attchd, Detchd, Attchd, Attchd, ...
## $ GarageYrBlt   <int> 2003, 1976, 2001, 1998, 2000, 1993, 2004, 1973, ...
## $ GarageFinish  <fct> RFn, RFn, RFn, Unf, RFn, Unf, RFn, RFn, Unf, RFn...
## $ GarageCars    <int> 2, 2, 2, 3, 3, 2, 2, 2, 2, 1, 1, 3, 1, 3, 1, 2, ...
## $ GarageArea    <int> 548, 460, 608, 642, 836, 480, 636, 484, 468, 205...
## $ GarageQual    <fct> TA, TA, TA, TA, TA, TA, TA, TA, Fa, Gd, TA, TA, ...
## $ GarageCond    <fct> TA, TA, TA, TA, TA, TA, TA, TA, TA, TA, TA, TA, ...
## $ PavedDrive    <fct> Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, ...
## $ WoodDeckSF    <int> 0, 298, 0, 0, 192, 40, 255, 235, 90, 0, 0, 147, ...
## $ OpenPorchSF   <int> 61, 0, 42, 35, 84, 30, 57, 204, 0, 4, 0, 21, 0, ...
## $ EnclosedPorch <int> 0, 0, 0, 272, 0, 0, 0, 228, 205, 0, 0, 0, 0, 0, ...
## $ X3SsnPorch    <int> 0, 0, 0, 0, 0, 320, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ ScreenPorch   <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 176, 0, 0, 0...
## $ PoolArea      <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ PoolQC        <fct> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ...
## $ Fence         <fct> NA, NA, NA, NA, NA, MnPrv, NA, NA, NA, NA, NA, N...
## $ MiscFeature   <fct> NA, NA, NA, NA, NA, Shed, NA, Shed, NA, NA, NA, ...
## $ MiscVal       <int> 0, 0, 0, 0, 0, 700, 0, 350, 0, 0, 0, 0, 0, 0, 0,...
## $ MoSold        <int> 2, 5, 9, 2, 12, 10, 8, 11, 4, 1, 2, 7, 9, 8, 5, ...
## $ YrSold        <int> 2008, 2007, 2008, 2006, 2008, 2009, 2007, 2009, ...
## $ SaleType      <fct> WD, WD, WD, WD, WD, WD, WD, WD, WD, WD, WD, New,...
## $ SaleCondition <fct> Normal, Normal, Normal, Abnorml, Normal, Normal,...
## $ SalePrice     <int> 208500, 181500, 223500, 140000, 250000, 143000, ...
```

We inspect the desired outcome "SalePrice" and can see that house prices range from 34900 to 755000 Dollars in the training subset. The median house price is 163000 Dollars.

```
summary(train$SalePrice)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   34900  129975  163000  180921  214000  755000
```

To facilitate feature engineering and data cleaning we temporarily merge `train` and `test` into `dataset`. We will now systematically analyse each feature of the dataset. Whenever we work with the "SalePrice"" variable, we will subset `dataset` with the `train$Id`, as `test` has no entries for it.

```
test$SalePrice <- 0 # Temporarily add a "SalePrice" column with zeros to `test`.
dataset <- rbind(train, test) # Merge `train` and `test` into one `dataset`.
```

2.1 Systematic analysis and manipulation of all dataset features

We will systematically analyse each individual feature of the dataset, deal with missing values, engineering them where deemed useful, and analysing their potential influence on the outcome variable "SalePrice". Some groups of features will be dealt with simultaneously.

2.1.1 MSSubClass: Identifies the type of dwelling involved in the sale.

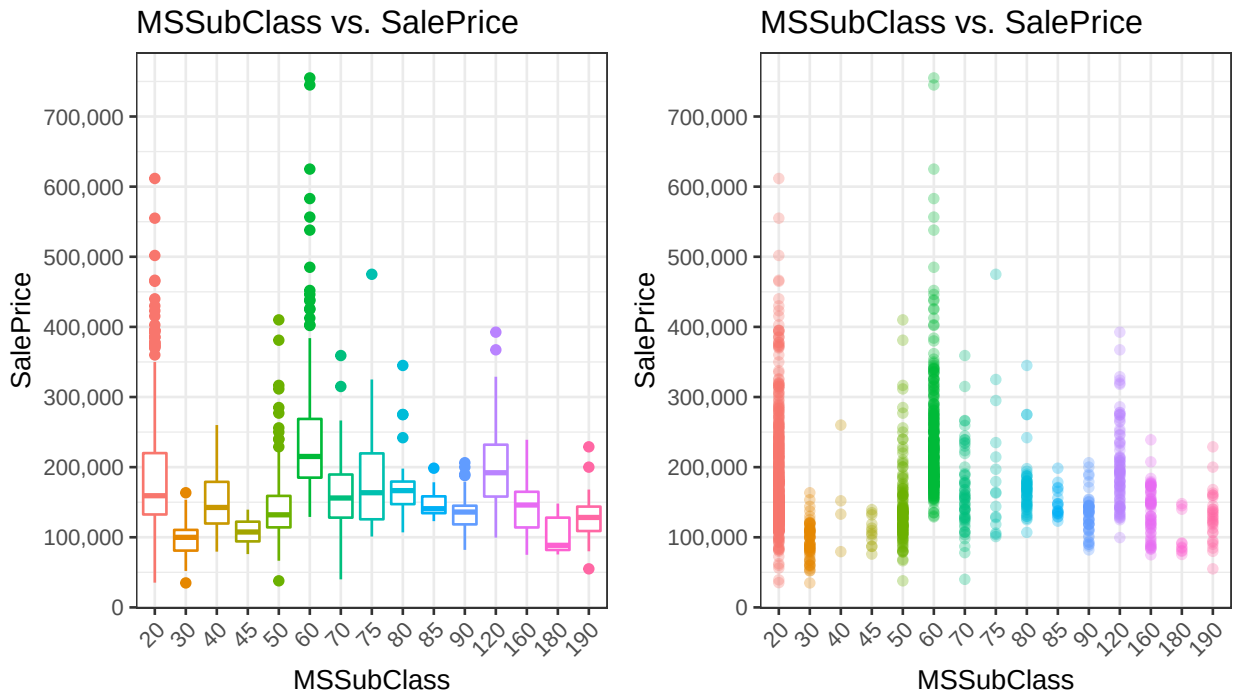
There are no missing values in MSSubClass.

```
summary(dataset$MSSubClass)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   20.00   20.00   50.00   57.14   70.00  190.00
```

MSSubClass should be a factor variable and we therefore change it accordingly.

```
dataset$MSSubClass <- as.factor(dataset$MSSubClass)
```

We plot "MSSubClass" (in the `train$Id` subset of `dataset`) versus "SalePrice" and observe that some of the most expensive houses are from the "20", "50", and "60" category of MSSubClass. "20" are "1-STORY 1946 & NEWER ALL STYLES", "50" are "1-1/2 STORY FINISHED ALL AGES", and "60" are "2-STORY 1946 & NEWER". From the scatterplot we can see that only a few houses are in "MSSubClass" "40" and "180". "MSSubClass" is therefore a variable that denotes type and age of houses. A somewhat troubling part about this feature is the slight underrepresentation of some of its categories, which could be detrimental for predictive purposes.

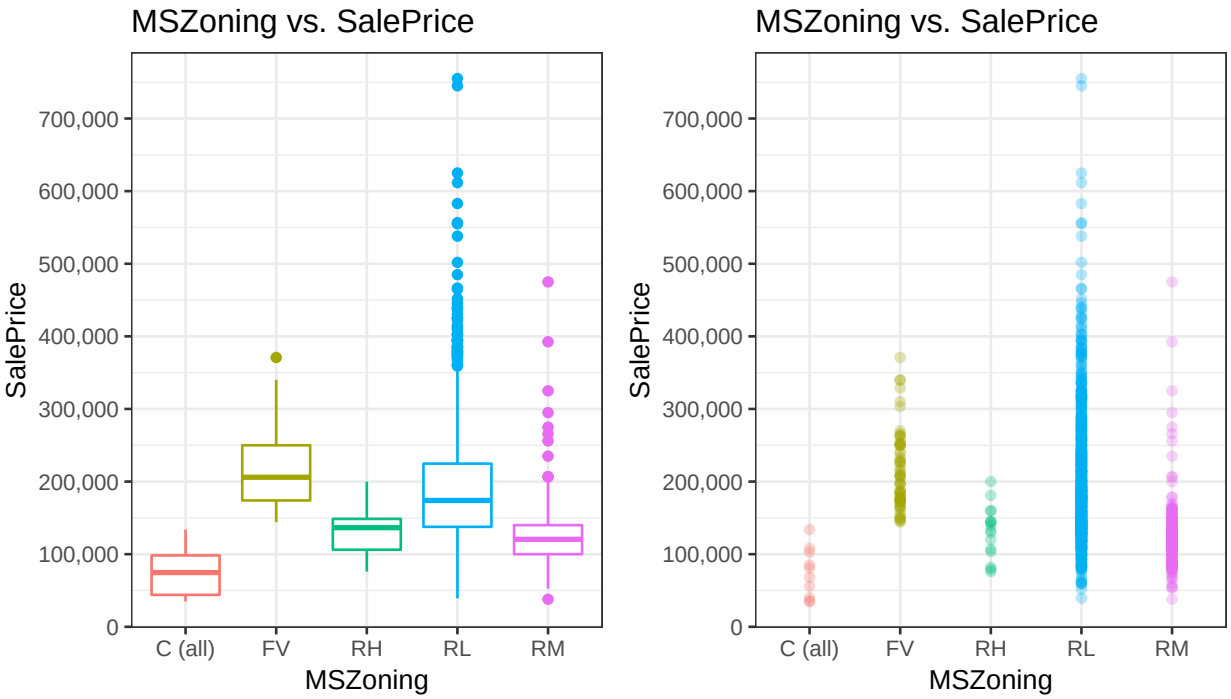


2.1.2 MSZoning: Identifies the general zoning classification of the sale.

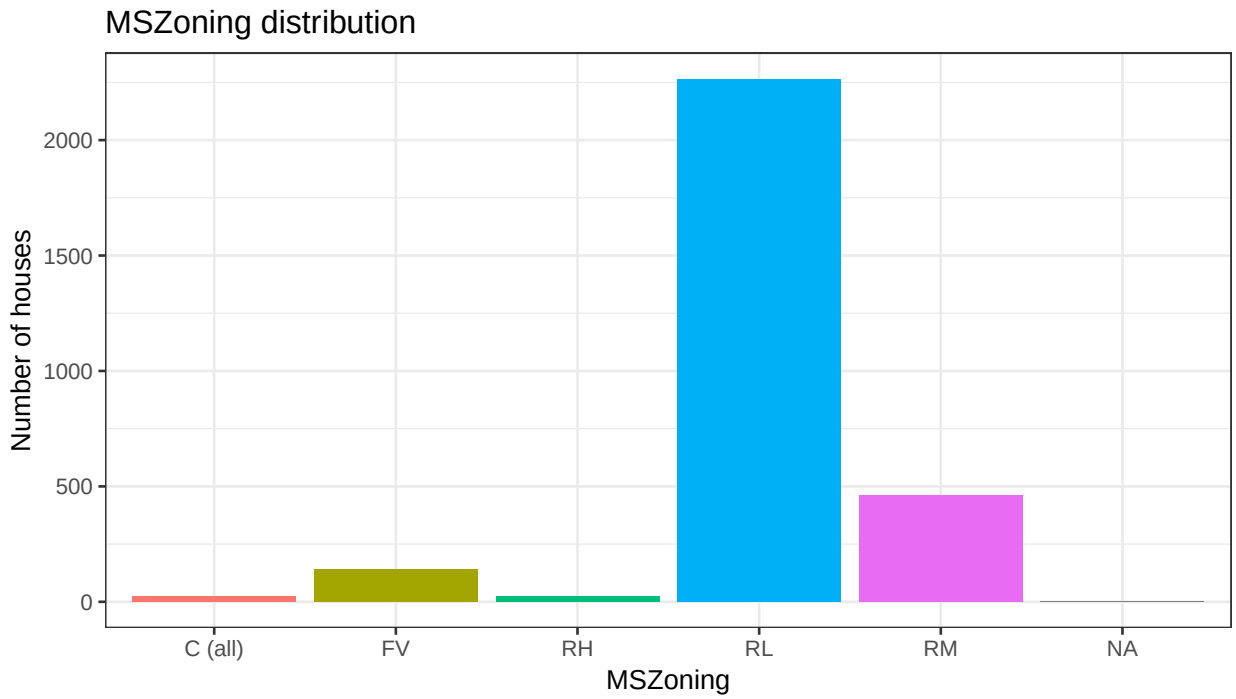
There are some missing values in "MSZoning".

```
summary(dataset$MSZoning)
## C (all)      FV      RH      RL      RM    NA's
##      25     139      26    2265     460       4
```

We plot "MSZoning" versus "SalePrice" and observe that the most expensive houses belong to the "RL" category, while "C (all)" contains less valuable houses. Residential low ("RL") and medium ("RM") density can be predictive of a higher "SalePrice", although floating village residential ("FL") has an even higher median sale price.



Next we are dealing with missing values in the "MSZoning" column. From the plot below we can see that "RL", or "residential low-density"" is clearly the most common value.



Does "MSZoning" depend on "MSSubClass"? Below we can also see that for "MSSubClass" of "20", "residential low-density" is clearly the most common value, while it is less clear-cut for values of "30" and "70".

```
dataset %>% select(MSSubClass, MSZoning) %>%
  group_by(MSSubClass, MSZoning) %>%
  filter(MSSubClass %in% c(20, 30, 70)) %>%
  count()
## # A tibble: 16 x 3
## # Groups:   MSSubClass, MSZoning [16]
##    MSSubClass MSZoning     n
##    <fct>      <fct>    <int>
##  1 20         C (all)      3
##  2 20         FV          34
##  3 20         RH           4
##  4 20         RL        1016
##  5 20         RM          20
##  6 20         <NA>         2
##  7 30         C (all)      8
##  8 30         RH           2
##  9 30         RL          61
## 10 30         RM          67
## 11 30         <NA>         1
## 12 70         C (all)      4
## 13 70         RH           3
## 14 70         RL          57
## 15 70         RM          63
## 16 70         <NA>         1
```

As it is not entirely clear which value we should impute, we use kNN-based missing value imputation to fill the missing values instead. The kNN-based model predicts the following values, which we use for imputation.

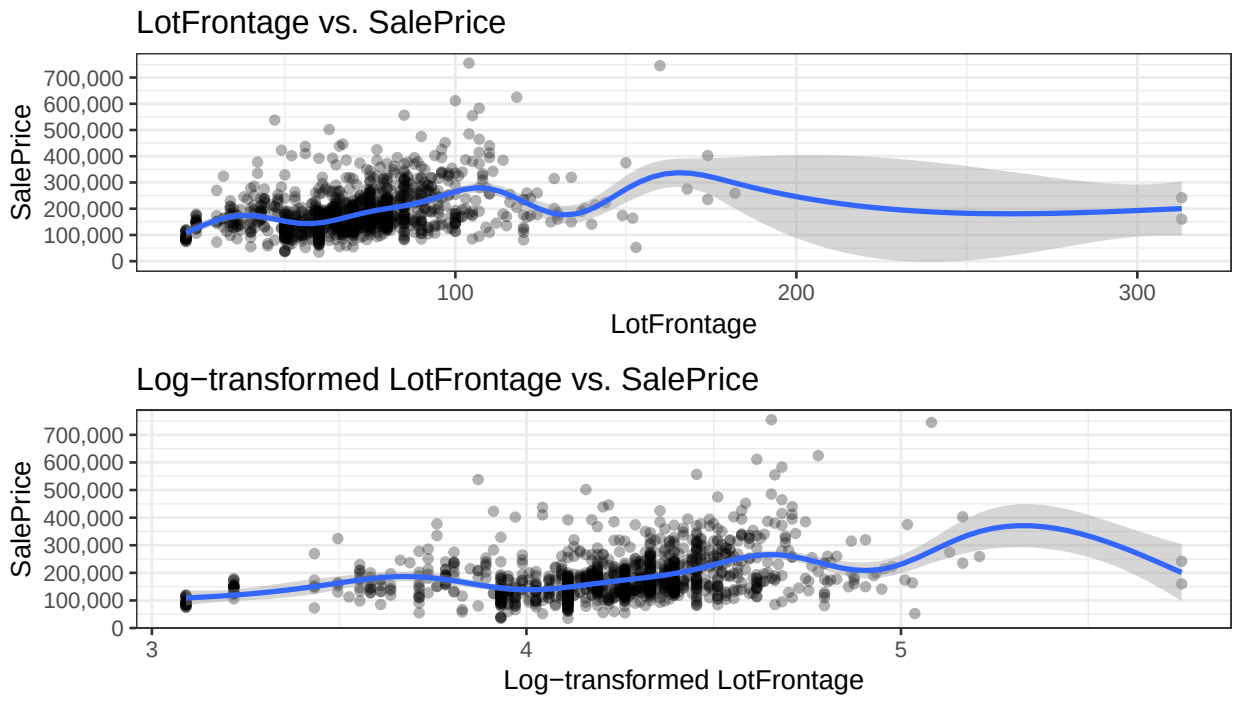
```
# Build a kNN-model with a k of 9 for MSZoning.
knn_model <- kNN(dataset, variable = "MSZoning", k = 9)

# Predicted MSZoning values.
knn_model[knn_model$MSZoning_imp == TRUE, ]$MSZoning
## [1] RM RL RL RL
## Levels: C (all) FV RH RL RM

# Missing value imputation.
dataset$MSZoning[which(is.na(dataset$MSZoning))] <-
       knn_model[knn_model$MSZoning_imp == TRUE, ]$MSZoning
```

2.1.3 LotFrontage: Linear feet of street connected to property

There are a lot of missing values in "LotFrontage".

```
summary(dataset$LotFrontage)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   21.00   59.00   68.00   69.31   80.00  313.00     486
```

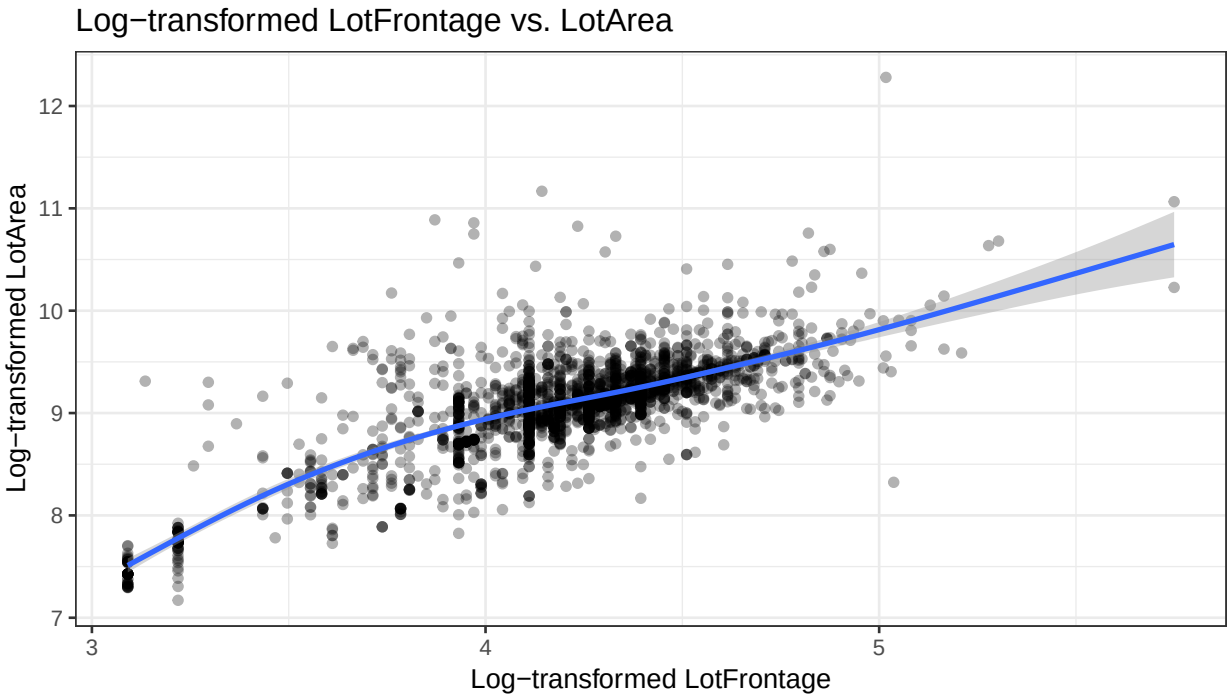
We plot "LotFrontage" against "SalePrice" (regular and log-transformed). From the plots we can observe that "LotFrontage" doesn't seem to influence "SalePrice" a lot. Also, there are two houses with very large "LotFrontage" values, but comparatively low "SalePrice".



Still, "LotFrontage" is decently correlated with "SalePrice" , as it can be interpreted as a measure of property size.

```
cor(na.omit(dataset$LotFrontage[train$Id]),
    dataset[train$Id,]$SalePrice[-which(is.na(dataset$LotFrontage))])
## [1] 0.3517991
```

Does "LotFrontage" correlate well with "LotArea"? We plot the log-transformed "LotFrontage" against "LotArea". Indeed, we find that "LotFrontage" correlates well with "LotArea", although there are some houses with a larger deviation.



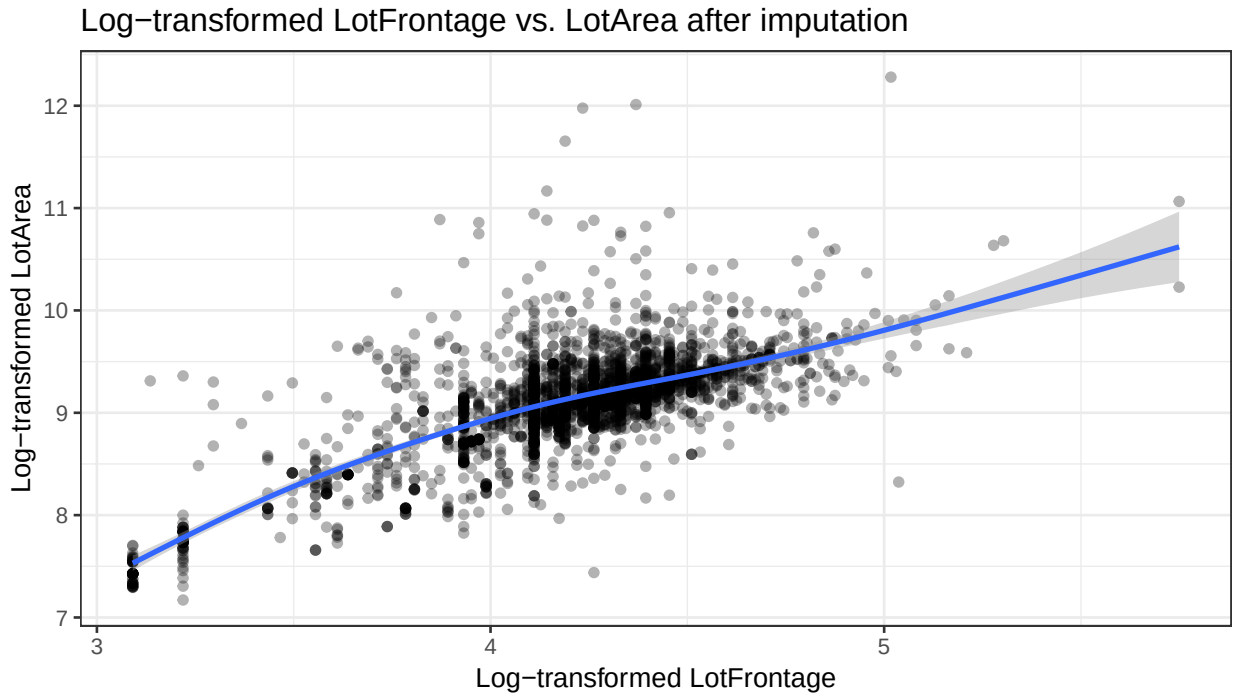
We calculate the correlation between "LotFrontage" and "LotArea." The correlation is quite strong, at almost 50% as seen below. This correlation is probably not stronger due to quite a few houses having noticably larger "LotAreas" while having lower "LotFrontage" values and vice-versa. This might potentially throw off predictions of "LotFrontage" based on "LotArea" a lot.

```
cor(na.omit(dataset$LotFrontage), dataset$LotArea[-which(is.na(dataset$LotFrontage))])
## [1] 0.4898956
```

Therefore we use kNN-based imputation for "LotFrontage". Plotted below is "LotFrontage" after kNN-based missing value imputation. In addition, the feature encoding is changed to numeric.

```
knn_model <- kNN(dataset, variable = "LotFrontage", k = 9)

# Predicted LotFrontage values.
# knn_model[knn_model$LotFrontage_imp == TRUE, ]$LotFrontage

# We impute the values and plot LotFrontage after imputation.
dataset$LotFrontage[which(is.na(dataset$LotFrontage))] <-
  knn_model[knn_model$LotFrontage_imp == TRUE, ]$LotFrontage
```

```
dataset %>%
  ggplot(aes(x = log1p(LotFrontage), y = log1p(LotArea))) +
  geom_point(alpha = 0.3) +
  geom_smooth(method = "gam", formula = y ~ s(x)) +
  ggtitle("Log-transformed LotFrontage vs. LotArea after imputation") +
  xlab("Log-transformed LotFrontage") +
  ylab("Log-transformed LotArea") +
  theme_bw()
```




```
# We change variable encoding to numeric
dataset$LotFrontage <- as.numeric(dataset$LotFrontage)
```

2.1.4 LotArea: Lot size in square feet

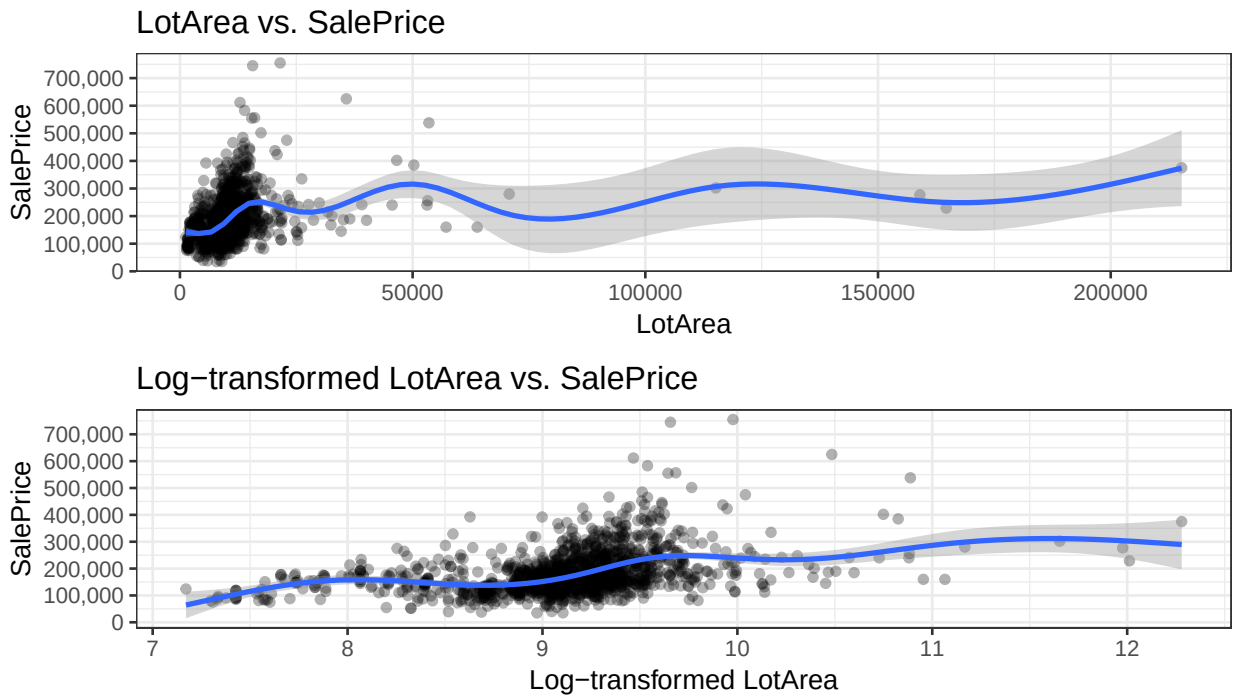
There are no missing values in "LotArea".

```
summary(dataset$LotArea)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1300    7478    9453   10168   11570  215245
```

"LotArea" is encoded as an integer value, but it should be numeric and is changed accordingly.

```
dataset$LotArea <- as.numeric(dataset$LotArea)
```

We plot "LotArea" against "SalePrice" (regular and log-transformed). As there seem to be some outliers, the log-transformation of "LotArea" helps us visualize the relationship of "LotArea" with "SalePrice" better.



"LotArea" is positively correlated with "SalePrice".

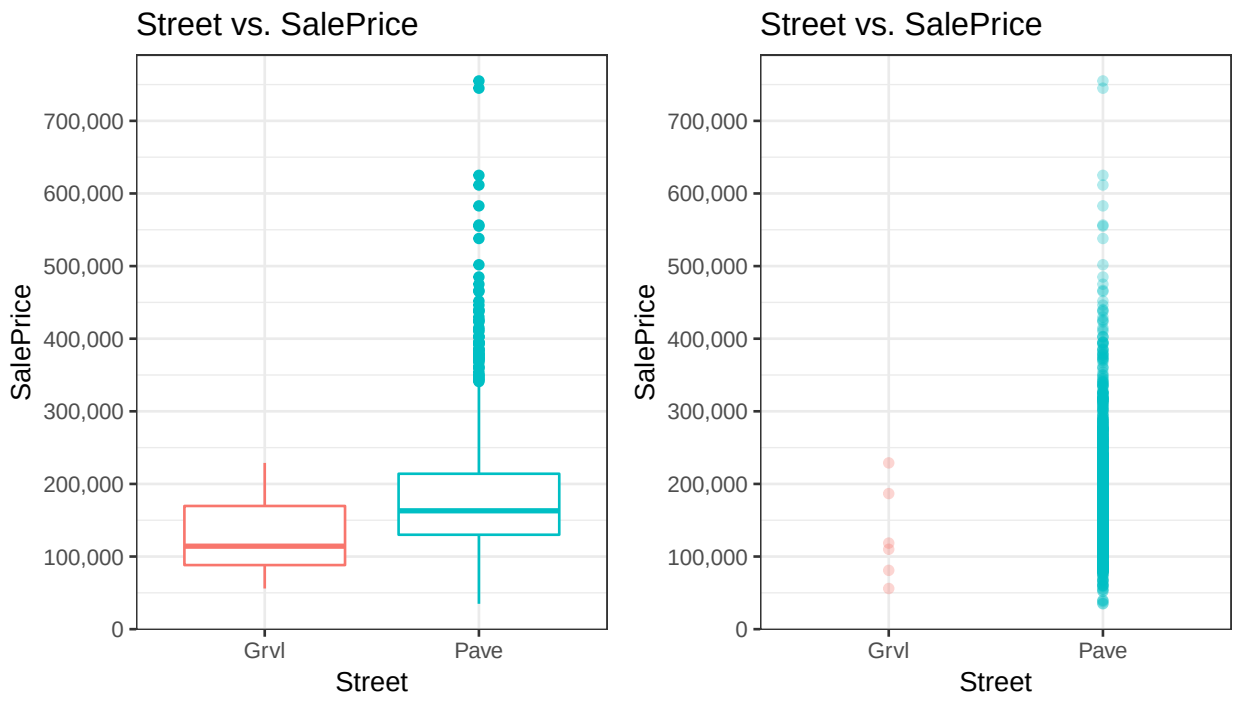
```
cor(dataset[train$Id, ]$LotArea, dataset[train$Id, ]$SalePrice)
## [1] 0.2638434
```

2.1.5 Street: Type of road access to property

There are no missing values in "Street".

```
summary(dataset$Street)
## Grvl Pave
##   12 2907
```

We plot "Street" against "SalePrice" and observe that the type of road access to a property seems to matter quite a bit in terms of "SalePrice". However, only a few houses have "gravel" values in "Street". "Street" can take either "Gravel" or "Paved" as its value and might indicate a more rural or urban setting, respectively.



2.1.6 Alley: Type of alley access to property

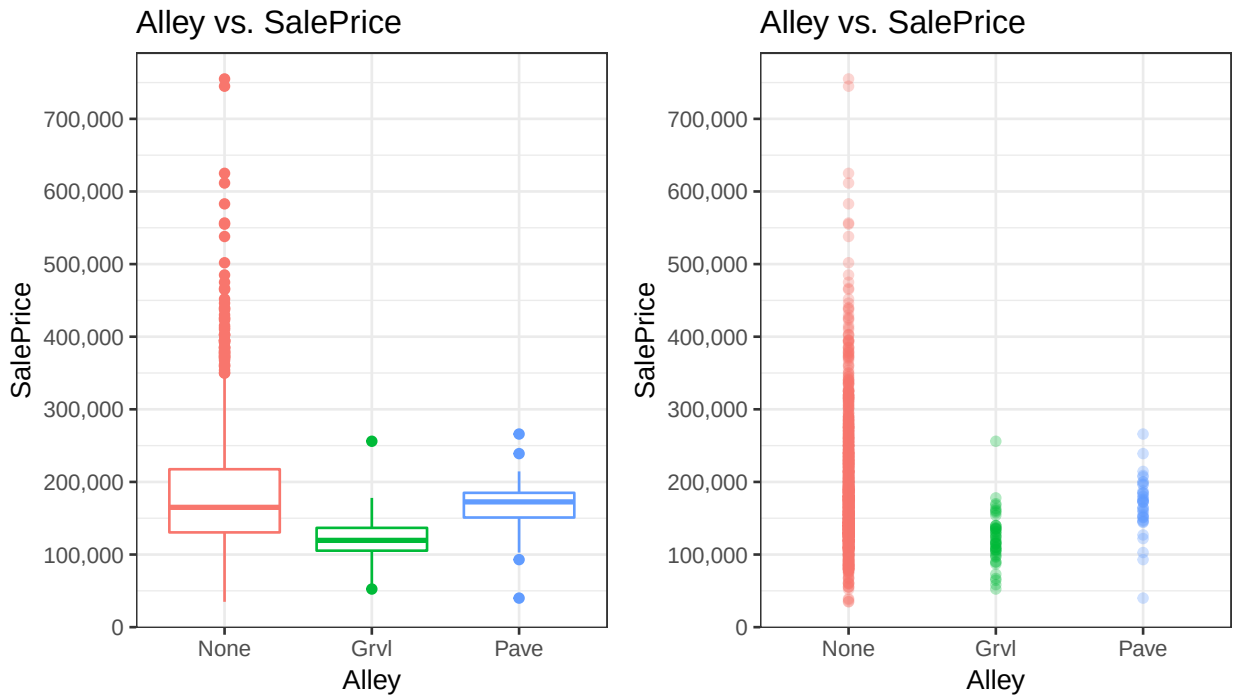
There is a very large amount of missing values in Alley. We know from the data description that "No alley access" was, rather unfortunately, encoded as "NA". These "NA" entries are producing false missing values.

```
summary(dataset$Alley)
## Grvl Pave NA's
##  120   78 2721
```

We fix these wrong "NA" entries by replacing them with "None".

```
dataset$Alley <- str_replace_na(dataset$Alley, replacement = "None")
dataset$Alley <- factor(dataset$Alley, levels = c("None", "Grvl", "Pave"))
```

As with "Street", we can see that having a "paved" type of "Alley" access to the property is indicative of a higher "SalePrice". Most houses don't have any "Alley" access, including all of the more expensive houses as well.

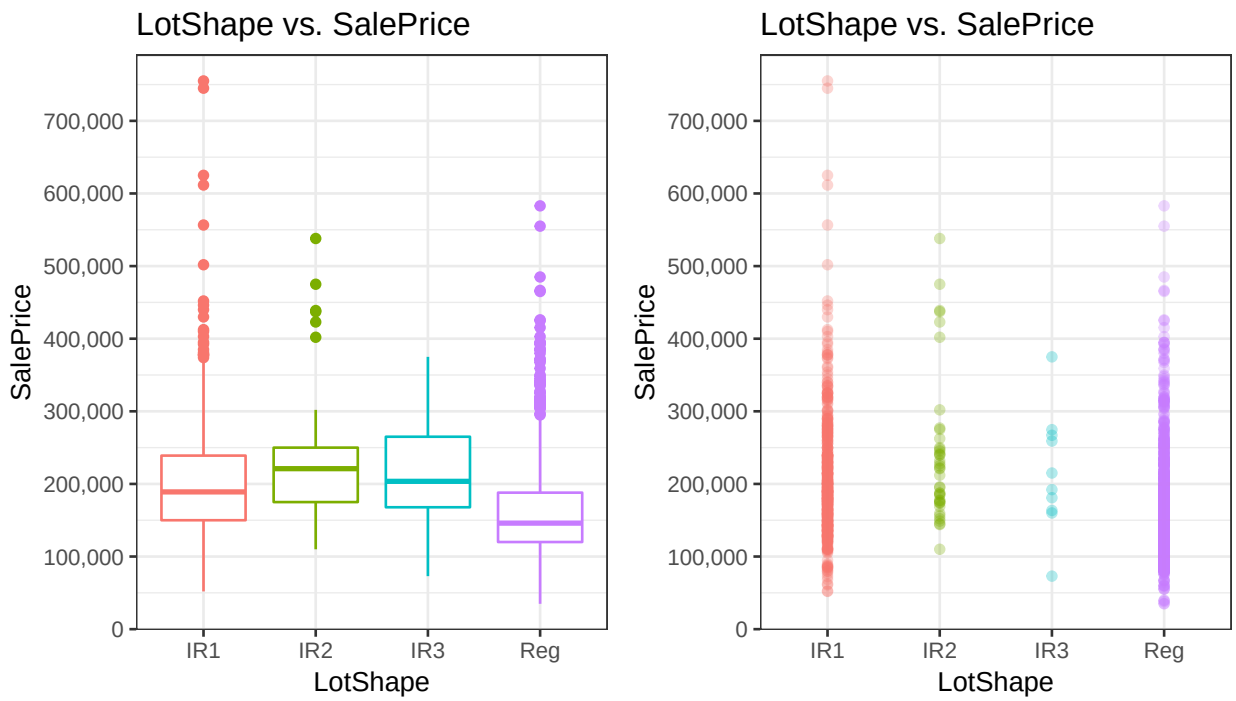


2.1.7 LotShape: General shape of property

There are no missing values in LotShape.

```
summary(dataset$LotShape)
##  IR1  IR2  IR3  Reg
##  968   76   16 1859
```

We plot "LotShape" against "SalePrice" and observe that some of the more expensive houses have a slightly irregular "IR1" "LotShape." A "regular" "Reg" "LotShape" is indicative of a lower "SalePrice", but the category still contains many houses with larger "SalePrice" as well. Only a small number of houses has a really irregular "IR3" "LotShape." Overall, "LotShape" doesn't seem to influence "SalePrice" too much.

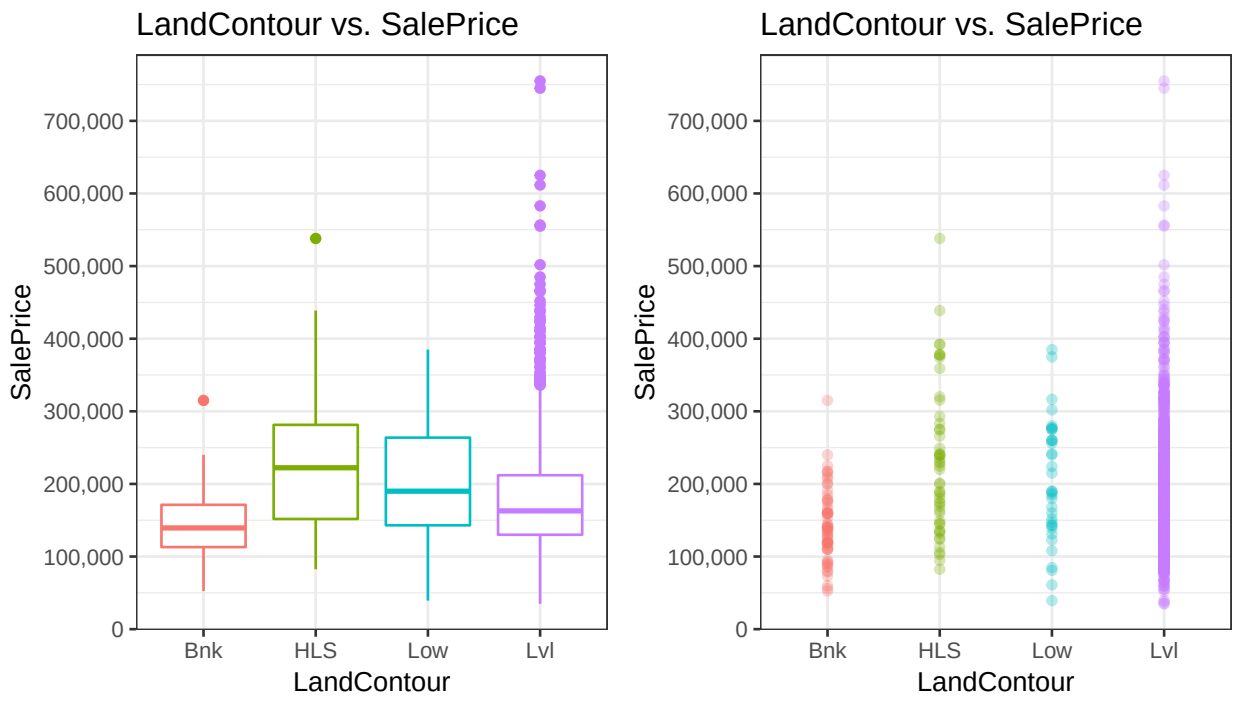


2.1.8 LandContour: Flatness of the property

"LandContour" contains no missing values.

```
summary(dataset$LandContour)
##  Bnk  HLS  Low  Lvl
##  117  120   60 2622
```

We plot "LandContour" against "SalePrice" and obsere that a "Banked - Quick and significant rise from street grade to building entry" entry in "LandContour" can be indicative of a lower "SalePrice". The most expensive houses are on level, nearly flat terrain. This feature will help to distinguish some of the lower priced houses from the higher priced ones.

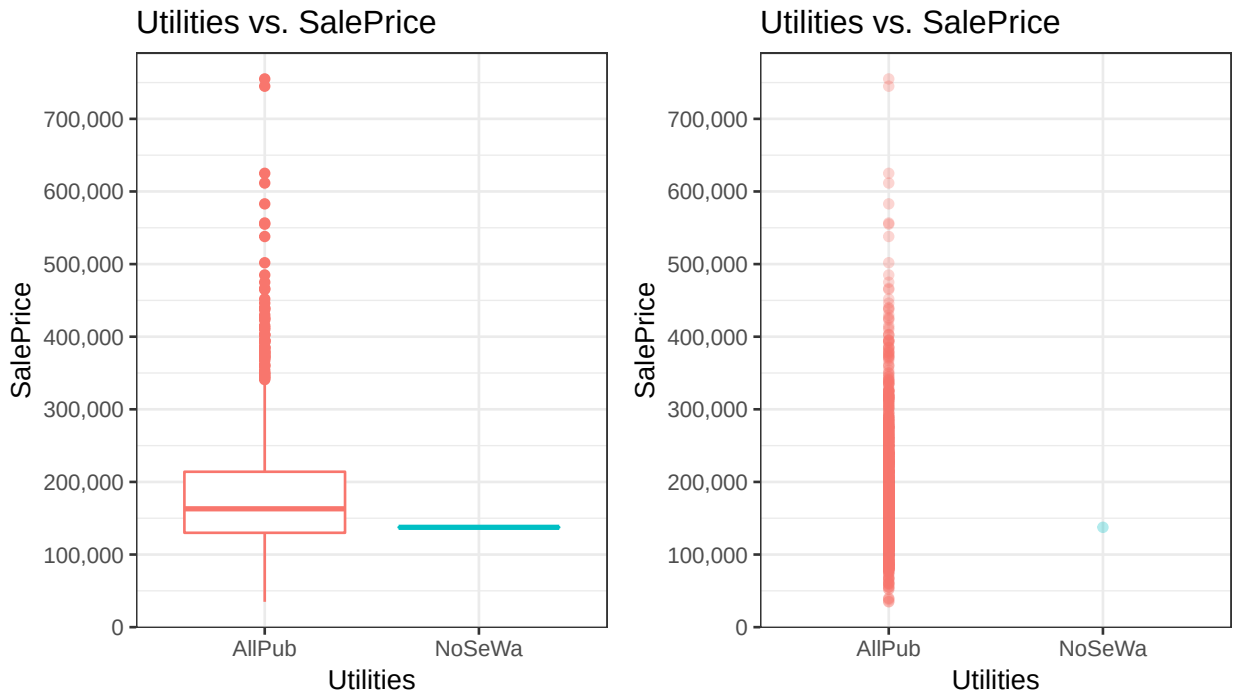


2.1.9 Utilities: Type of utilities available

There are a few missing values in "Utilities".

```
summary(dataset$Utilities)
## AllPub NoSeWa   NA's
##   2916      1      2
```

We plot "Utilities" against "SalePrice" and immediately notice that only a single house has an "NoSeWa" entry.



This feature will not be helpful in "SalePrice" prediction, as "NoSeWa" is woefully underrepresented. We will therefore drop this feature from the dataset.

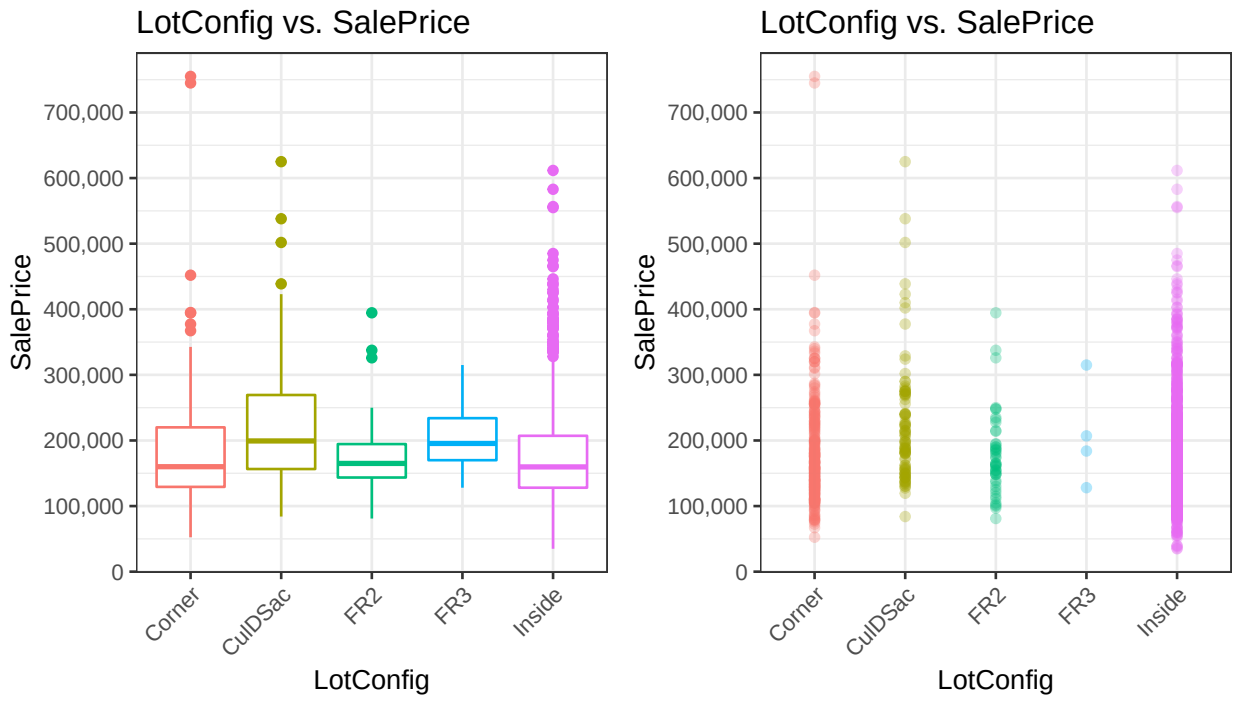
```
dataset <- subset(dataset, select = -Utilities)
```

2.1.10 LotConfig: Lot configuration

There are no missing values in "LotConfig".

```
summary(dataset$LotConfig)
##  Corner CulDSac     FR2     FR3  Inside
##     511     176      85      14    2133
```

We plot "LotConfig" against "SalePrice" and observe that there doesn't seem to be a strong correlation between them. The "FR3" category might be a little bit underrepresented.

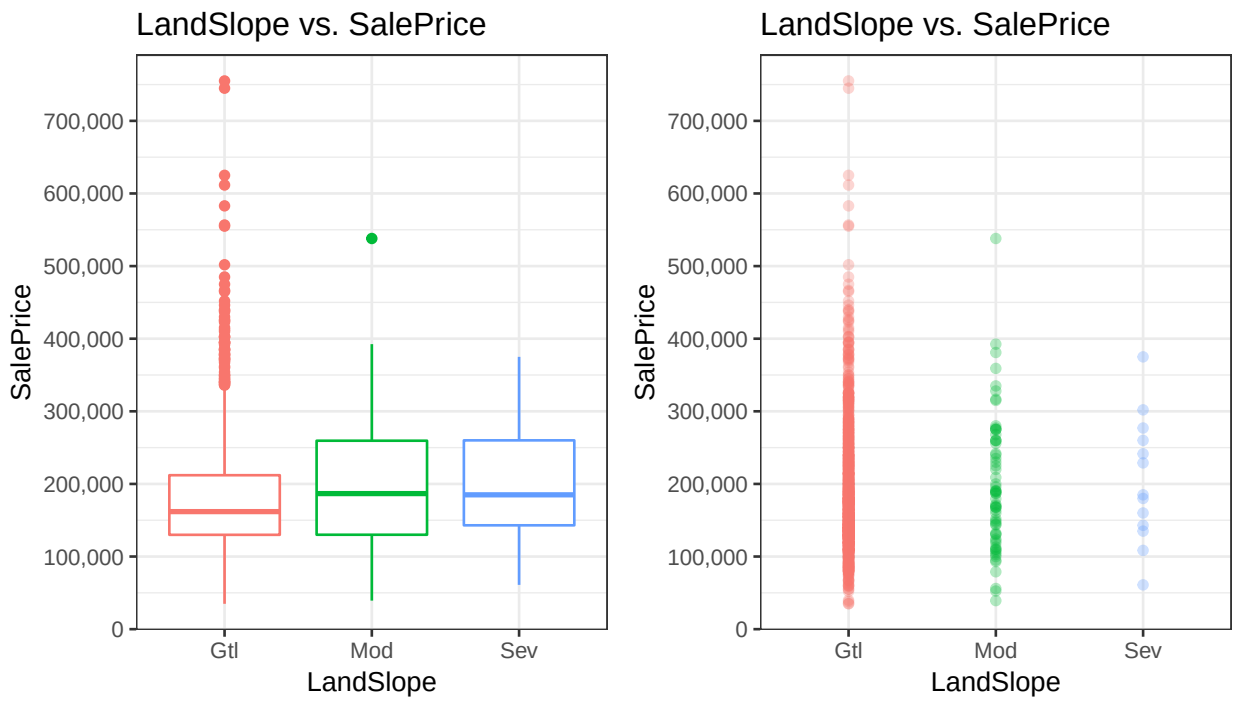


2.1.11 LandSlope: Slope of property

There are no missing values in "LandSlope".

```
summary(dataset$LandSlope)
##  Gtl  Mod  Sev
## 2778  125   16
```

We plot "LandSlope" against "SalePrice" and we can see that a gentle "LandSlope" can be predictive of a higher "SalePrice", as this category contains most of the very expensive houses. The medians do not differ very much, however.

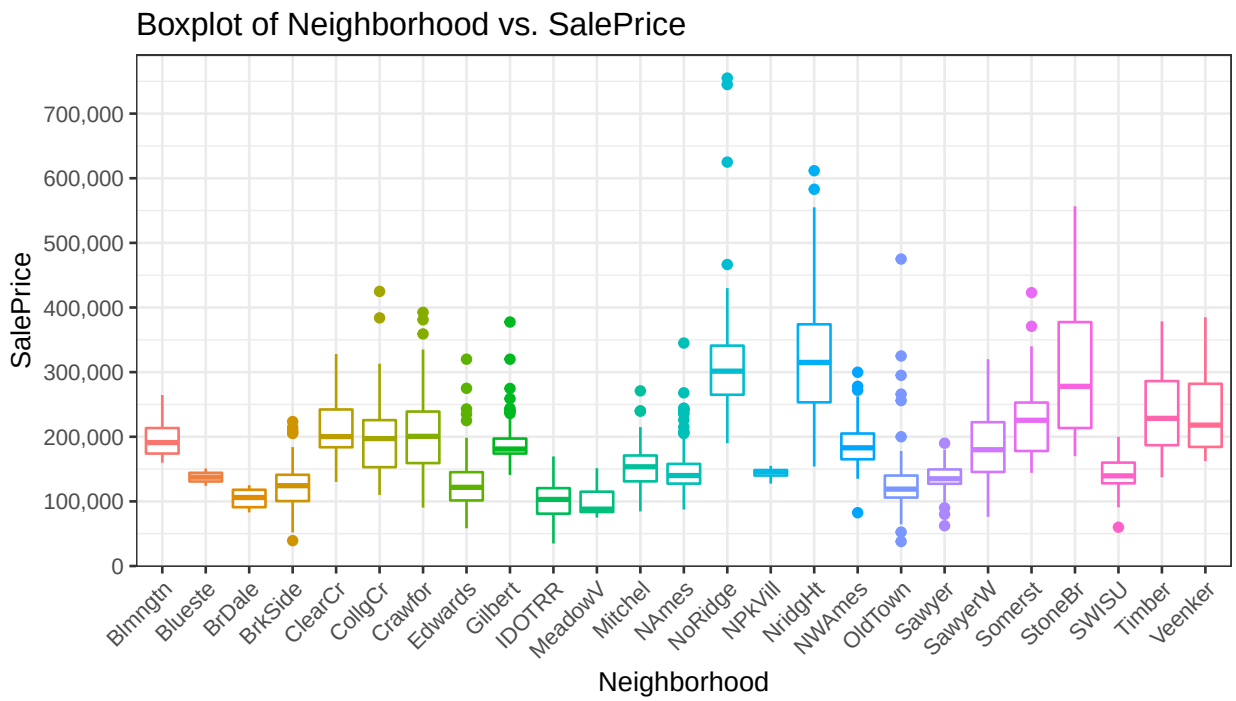


2.1.12 Neighborhood: Physical locations within Ames city limits

There are no missing values in "Neighborhood".

```
summary(dataset$Neighborhood)
## Blmngtn Blueste  BrDale BrkSide ClearCr CollgCr Crawfor Edwards Gilbert
##      28      10      30     108      44     267     103     194     165
##  IDOTRR MeadowV Mitchel   NAmes NoRidge NPkVill NridgHt  NWAmes OldTown
##      93      37     114     443      71      23     166     131     239
##  Sawyer SawyerW Somerst StoneBr   SWISU  Timber Veenker
##     151     125     182      51      48      72      24
```

We plot "Neighborhood" against "SalePrice" and observe that "Northridge" and "Northridge High"" are where some of the most expensive houses are situated. The respective "Neighborhood" is a strong indicator for a houses' "SalePrice".

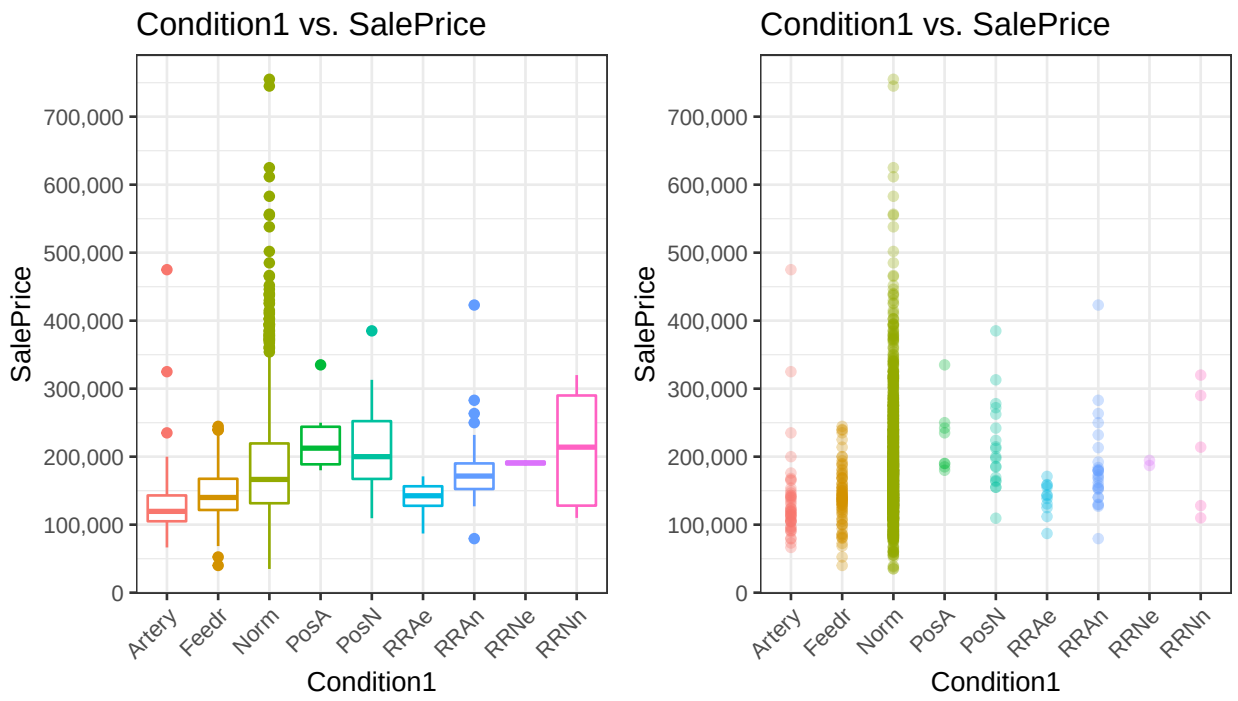


2.1.13 Condition1: Proximity to various conditions

There are no missing values in "Condition1".

```
summary(dataset$Condition1)
## Artery  Feedr   Norm   PosA   PosN   RRAe   RRAn   RRNe   RRNn
##     92    164   2511     20     39     28     50      6      9
```

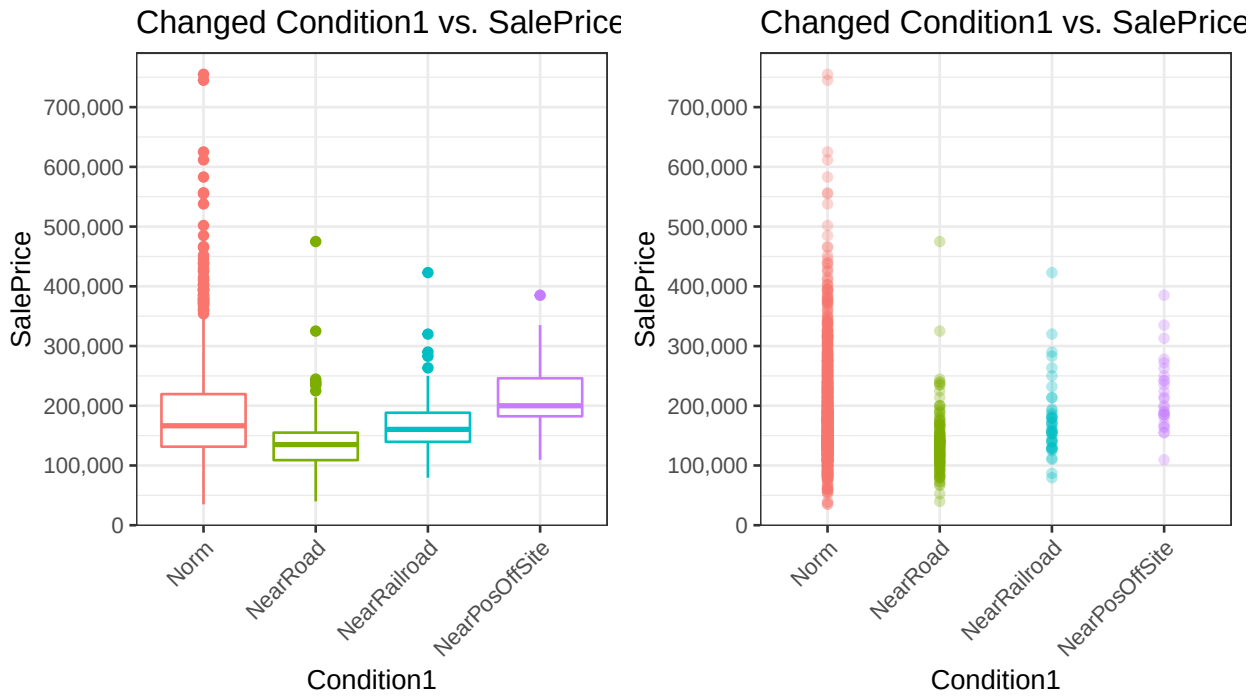
We plot "Condition1" against "SalePrice" and observe that adjacency to an artery or feeder street, as well as to a railroad may indicate a lower "SalePrice". Adjacency to a positive off-site feature can be indicative of an increased value. We also note that some categories are rather sparsely populated.



We will combine the two different road-associated categories ("Artery", "Feedr") and the four different rail-associated categories ("RRAe,"RRAn","RRNe","RRNn") into two distinct categories. We will also combine the two positive off-site features.

```
# Before the conversion, we convert the factor into a character vector.
dataset$Condition1 <- as.character(dataset$Condition1)

# We then merge the categories as described.
dataset$Condition1[dataset$Condition1 %in% c("Artery", "Feedr")] <- "NearRoad"
dataset$Condition1[dataset$Condition1 %in% c("RRAe",
                                             "RRAn", "RRNe", "RRNn")] <- "NearRailroad"
dataset$Condition1[dataset$Condition1 %in% c("PosA", "PosN")] <- "NearPosOffSite"

# We convert back into a factor with appropriate levels.
dataset$Condition1 <- factor(dataset$Condition1, levels = c("Norm",
                                    "NearRoad", "NearRailroad", "NearPosOffSite"))
```

The plots of the changed Condition1 feature show that adjacency to a road or railroad tends to decrease "SalePrice", while adjacency to a positive off-site leads to an increase. Almost all expensive houses are in the "Normal" category.



2.1.14 Condition2: Proximity to various conditions (if more than one is present)

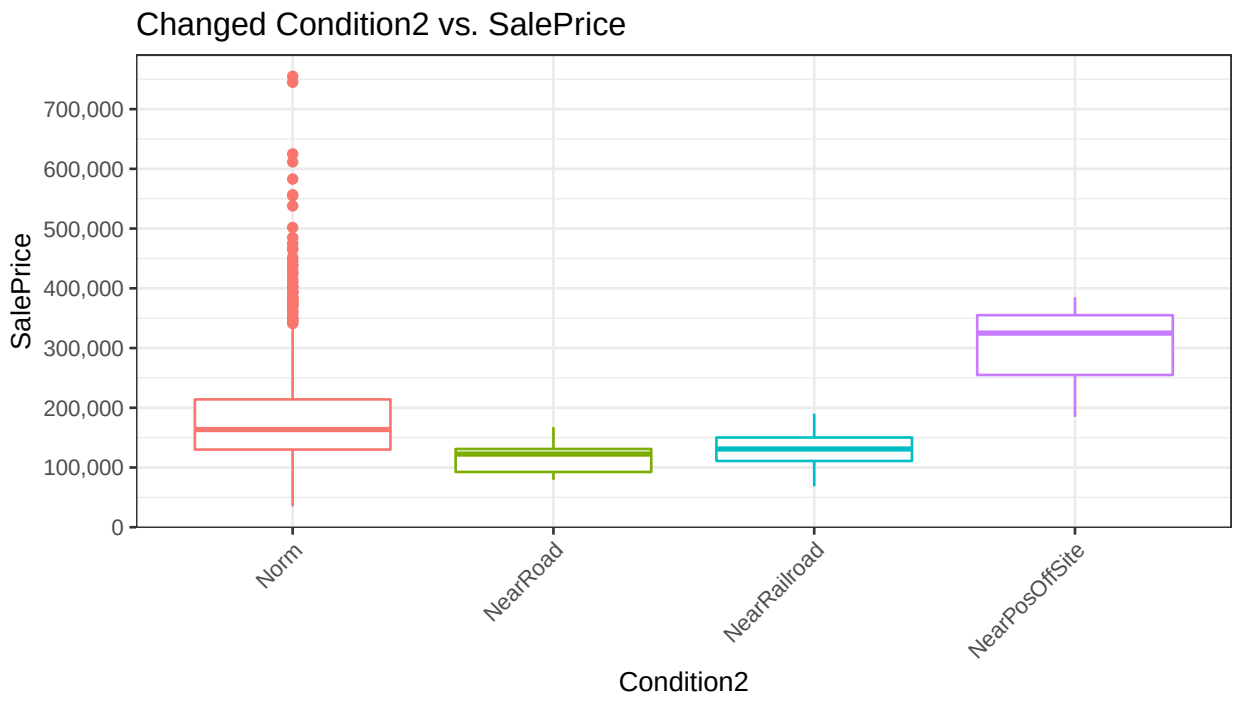
Some houses are adjacent to more than one condition, else they received a "Normal" entry in "Condition2". There are no missing values in "Condition2". However, there is no "RRNe" entry.

```
summary(dataset$Condition2)
## Artery  Feedr   Norm   PosA   PosN   RRAe   RRAn   RRNn
##      5     13   2889      4      4      1      1      2
```

We will apply the same changes to "Condition2" as we did to "Condition1".

```
# Before the conversion, we convert the factor into a character vector.
dataset$Condition2 <- as.character(dataset$Condition2)

# We then merge the categories as described.
dataset$Condition2[dataset$Condition2 %in% c("Artery", "Feedr")] <- "NearRoad"
dataset$Condition2[dataset$Condition2 %in% c("RRAe", "RRAn", "RRNn")] <- "NearRailroad"
dataset$Condition2[dataset$Condition2 %in% c("PosA", "PosN")] <- "NearPosOffSite"

# We convert back into a factor with appropriate levels.
dataset$Condition2 <- factor(dataset$Condition2, levels = c("Norm",
                                    "NearRoad", "NearRailroad", "NearPosOffSite"))
```

We plot the changed "Condition2" against "SalePrice". Clearly, being near a positive off-site raises the sale price, while adjacency to a busy road or railroad reduces it.



Are there houses which are adjacent to both a road and a railroad simultaneously? If so, what is their average "SalePrice"?

```
dataset[train$Id, ] %>%
  filter(Condition1 %in% c("NearRoad", "NearRailroad"),
         Condition2 %in% c("NearRoad", "NearRailroad")) %>%
          summarize(avg.value = mean(SalePrice))
##   avg.value
## 1  121700.4
```

Indeed, the average "SalePrice" of houses with a "NearRoad" as well as a "NearRailRoad" entry is rather low. We will merge "Condition1" and "Condition2" into a single feature, as "Condition2" has very few entries in general. Prior to this, we will merge "NearRoad" and "NearRailRoad" into a single category, "NearNegOffSite", which simply indicates adjacency to a negative off-site feature (e.g. a road or railroad).

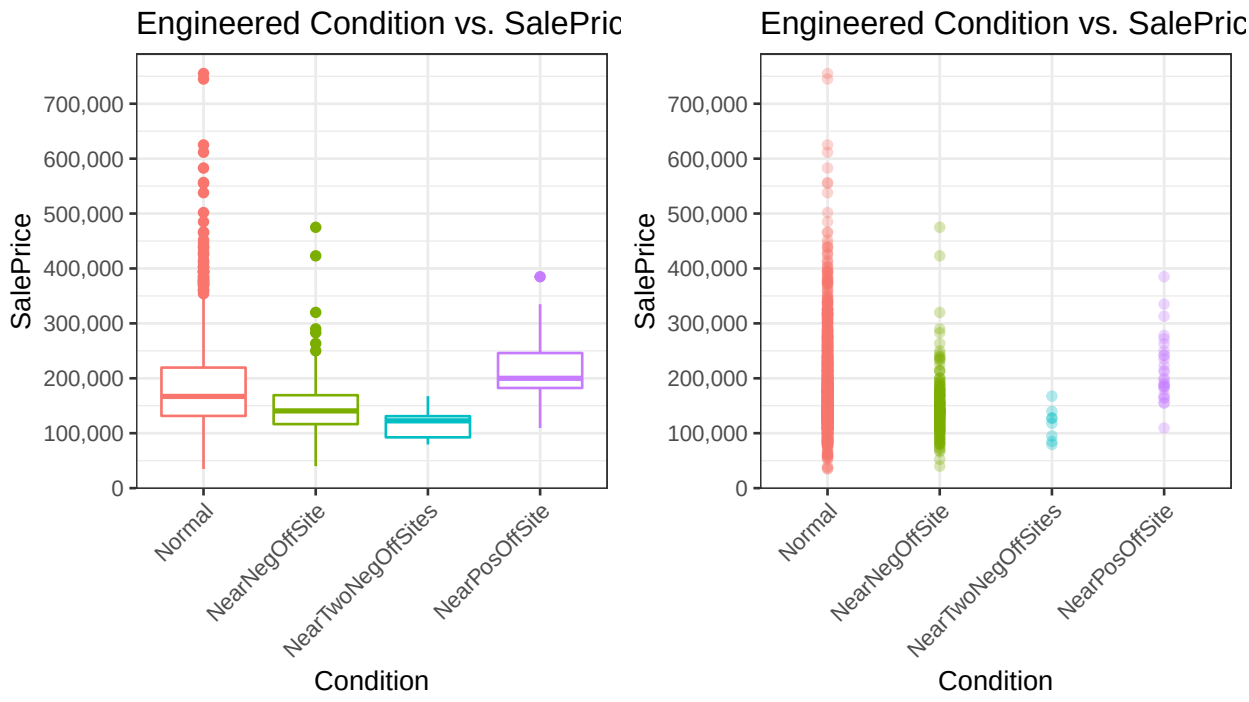
```
# Prior to modification convert to character and combine the entries into one.
dataset$Condition1 <- as.character(dataset$Condition1)
dataset$Condition1[which(dataset$Condition1 %in% c("NearRoad",
                                 "NearRailroad"))] <- "NearNegOffSite"
```

Next, we will determine which houses are nearby a second negative or positive off-site and create respective categories. As there is only a single house which is near a positive as well as a negative off-site, we will assign it to the "Normal" group, assuming that the effects cancel each other out. As there are only two houses near two positive off-sites, we assign them to "NearPosOffSite", instead of creating another sparsely populated category. The merged feature "Condition" is added to the `dataset` and "Condition1" and "Condition2" are dropped.

```
# We convert to character prior to editing.
dataset$Condition2 <- as.character(dataset$Condition2)
```

```
# We assign the respective entries as described.
dataset$Condition1[which(dataset$Condition2 %in% c("NearRoad",
                                    "NearRailRoad"))] <- "NearTwoNegOffSites"
index <- which(dataset$Condition2 %in% c("NearPosOffSite"))
dataset$Condition1[index[which(dataset$Condition1[index] == "NearPosOffSite")]] <-
  "NearPosOffSite"
dataset$Condition1[index[which(dataset$Condition1[index] == "NearNegOffSite")]] <-
  "Normal"
dataset$Condition1[dataset$Condition1 == "Norm"] <- "Normal"

# We convert back into factor and fix levels.
dataset$Condition1 <- factor(dataset$Condition1, levels = c("Normal",
                       "NearNegOffSite", "NearTwoNegOffSites", "NearPosOffSite"))

# We rename the engineered feature and remove the previous ones.
dataset$Condition <- dataset$Condition1
dataset <- subset(dataset, select = -c(Condition1, Condition2))
```

We then plot the engineered, merged "Condition" against "SalePrice". From the plots of the engineered "Condition" feature, we can see that being near one or even two negative off-sites (e.g. a road and/or a railroad) might indicate a lower "SalePrice". Adjacency to a positive off-site (e.g. a park or greenbelt etc...) is associated with a higher "SalePrice", however all really expensive houses are in the "Normal" category.

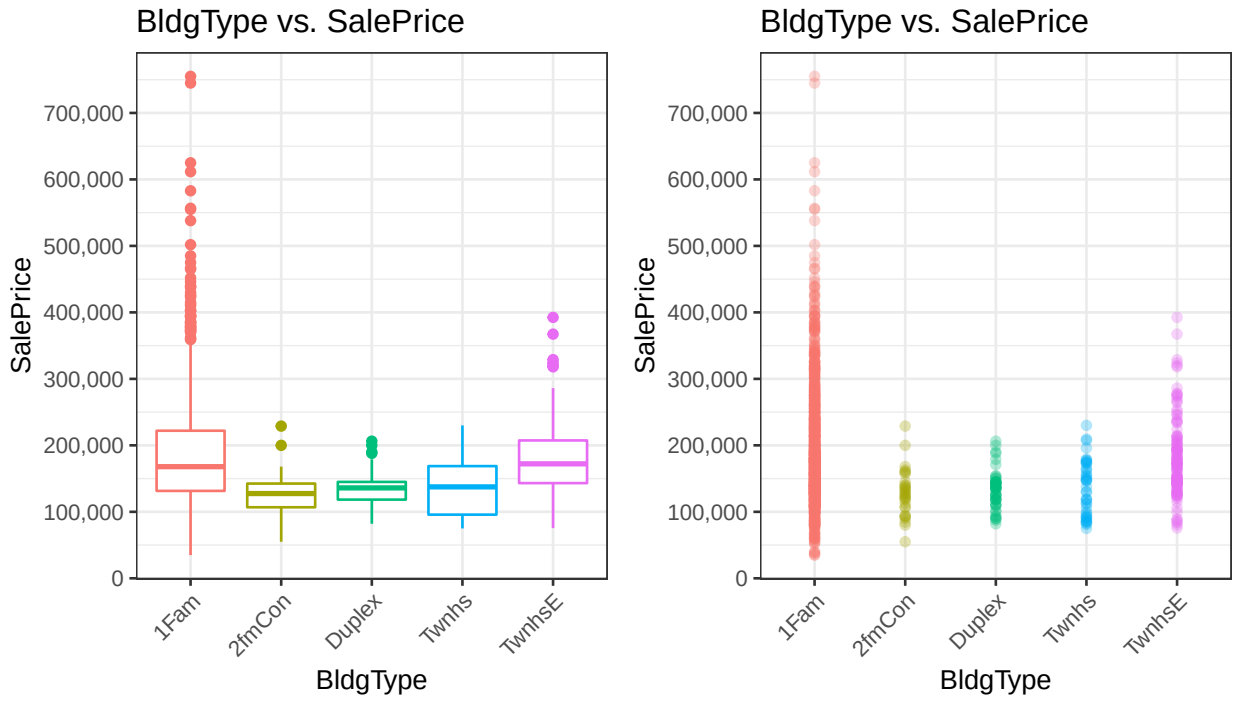


2.1.15 BldgType: Type of dwelling

There are no missing values in "BldgType".

```
summary(dataset$BldgType)
##   1Fam 2fmCon Duplex  Twnhs TwnhsE
##   2425     62    109     96    227
```

We plot "BldgType" against "SalePrice" and observe that the most expensive houses are all detached single-family houses ("1Fam"). Even then, there must be other distinguishing features behind those, as the median sale price of "1Fam" houses isn't noticably higher compared to the other categories.

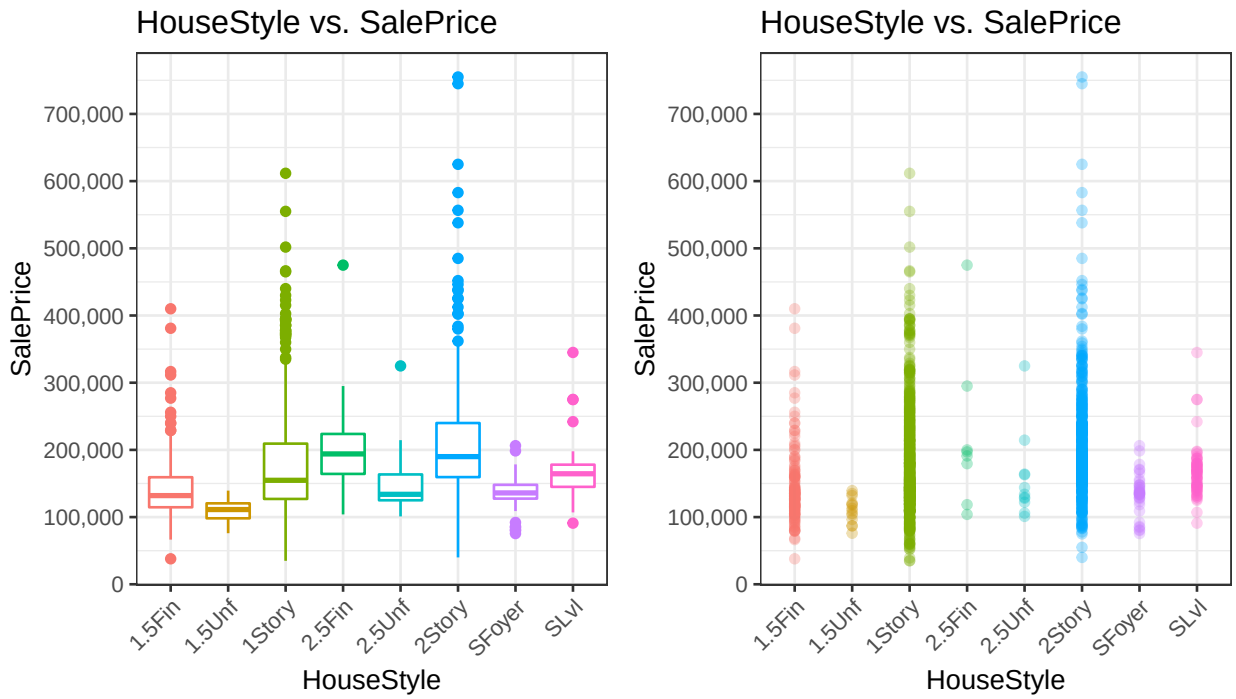


2.1.16 HouseStyle: Style of dwelling

There are no missing values in "HouseStyle".

```
summary(dataset$HouseStyle)
## 1.5Fin 1.5Unf 1Story 2.5Fin 2.5Unf 2Story SFoyer   SLvl
##    314     19   1471      8     24    872     83    128
```

We plot "HouseStyle" against "SalePrice" and observe that HouseStyle doesn't reveal very much about "SalePrice", but unfinished 1.5 and 2.5 storey houses tend to have lower "SalePrice" compared to completed ones. Unsurprisingly, the most expensive houses are in the "2Story" category, while another large set of expensive houses are in the "1Story" category.



2.1.17 OverallQual: Rates the overall material and finish of the house

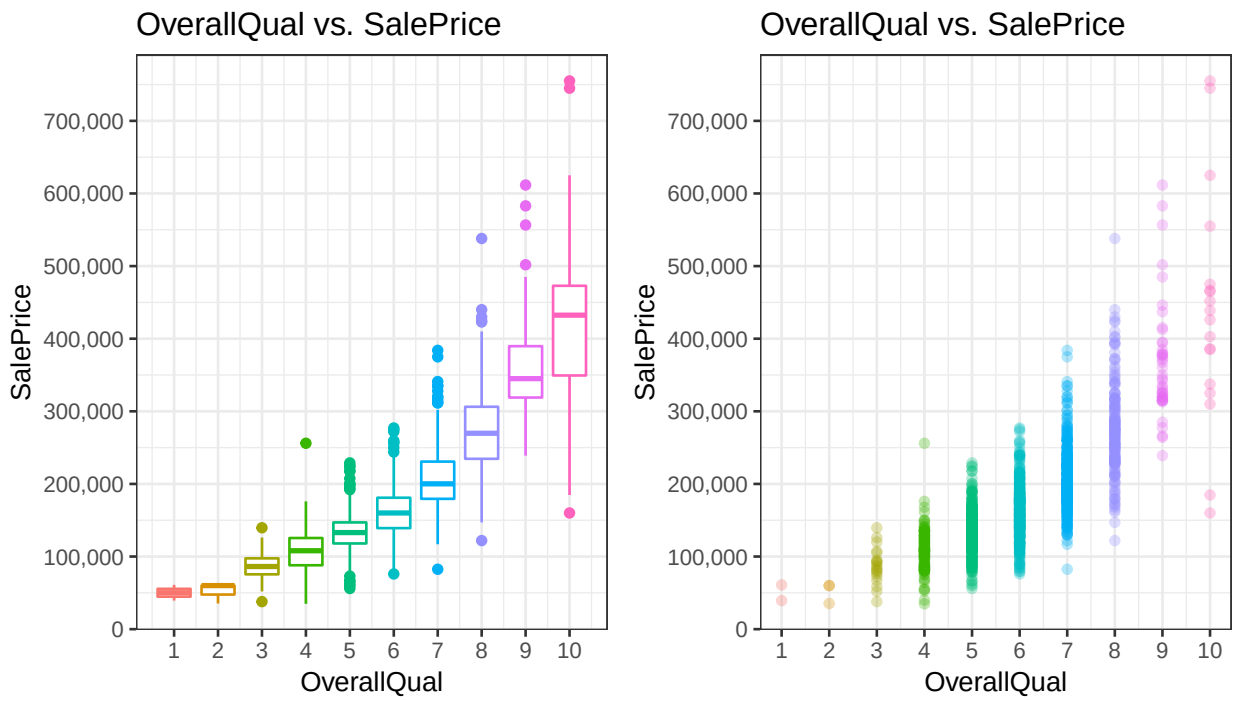
There are no missing values in "OverallQual".

```
summary(dataset$OverallQual)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   5.000   6.000   6.089   7.000  10.000
```

"OverallQual" should be a numeric variable, not an integer.

```
dataset$OverallQual <- as.numeric(dataset$OverallQual)
```

We plot "OverallQual" against "SalePrice" and notice its tremendous influence on a houses' "SalePrice". In the highest category, 10, there are 2 houses with unexpectedly low, and 2 houses with extremely high "SalePrice". This increases the range a lot, but is most likely due to other characteristics of these houses being less valuable. "OverallQual" will be very important for predicting a houses value.



There are only a few houses in the lowest "OverallQual" categories 1 and 2. Additionally, the higher quality levels show quite a large range of values.

2.1.18 OverallCond: Rates the overall condition of the house

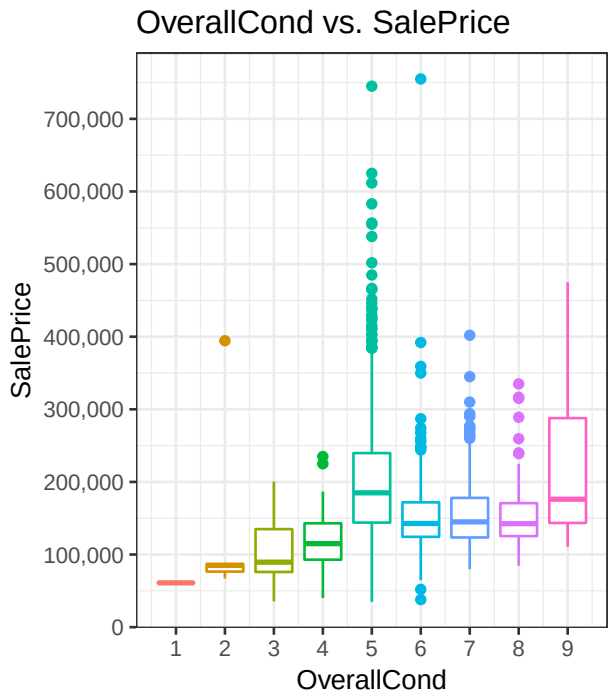
There are no missing values in "OverallCond".

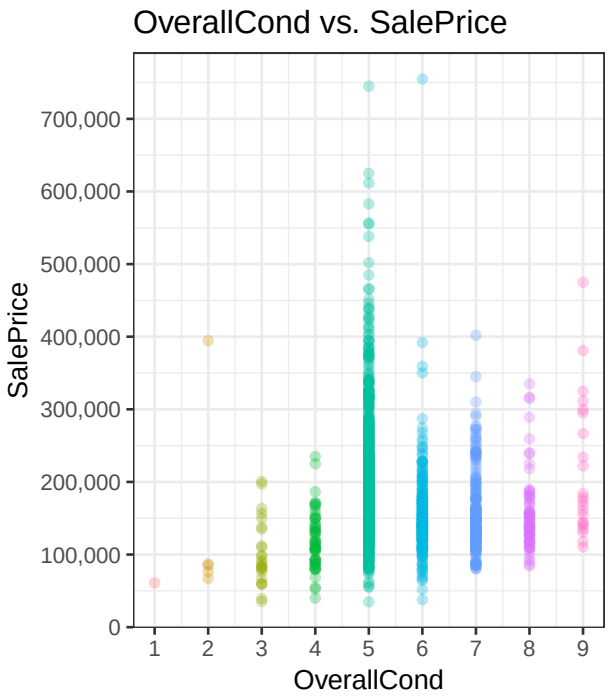
```
summary(dataset$OverallCond)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   5.000   5.000   5.565   6.000   9.000
```

"OverallCond" should be a numeric variable, not an integer.

```
dataset$OverallCond <- as.numeric(dataset$OverallCond)
```

We plot "OverallCond" against "SalePrice". Contrary to intuition, "OverallCond" is much less predictive of a houses' "SalePrice" and there is not a single house with a "very excellent" condition. We observe that the most expensive houses fall between 4 - 9, with 5 having most of them. "OverallCond" below 5 holds houses with lower "SalePrice". The lower two levels hold only very few houses.



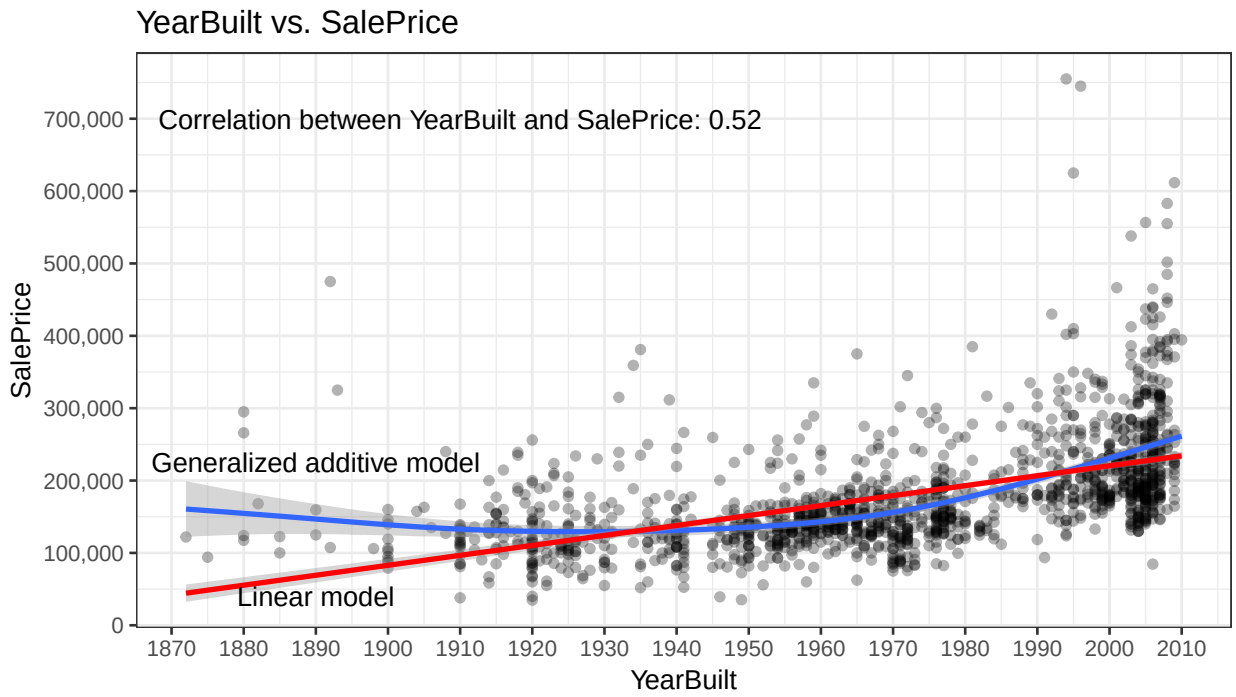


2.1.19 YearBuilt: Original construction date

There are no missing values in "YearBuilt".

```
summary(dataset$YearBuilt)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1872    1954    1973    1971    2001    2010
```

We plot "YearBuilt" against "SalePrice" and observe that "SalePrice" is influenced by "YearBuilt", especially houses built after the 1950's tend to go up in value as visualized by the models. It also becomes evident that there are relatively few entries before 1920.



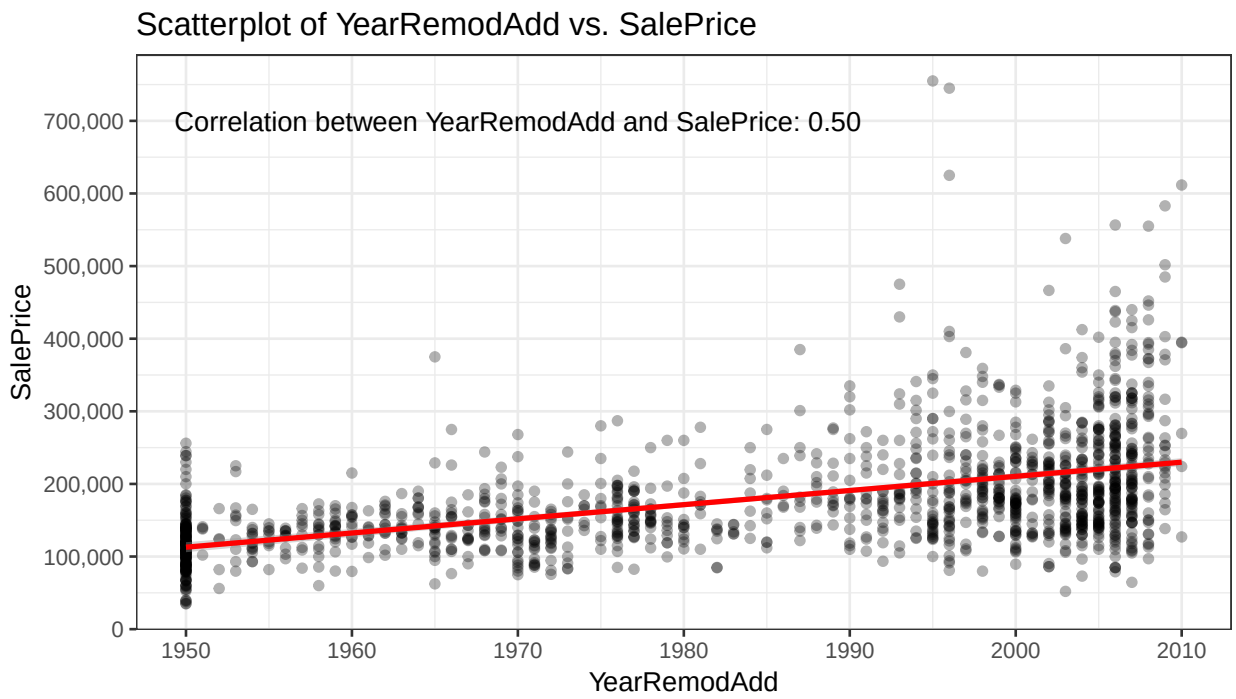

```
# There is a large positive correlation between "YearBuilt" and "SalePrice."
cor(dataset[train$Id, ]$YearBuilt, dataset[train$Id, ]$SalePrice)
## [1] 0.5228973
```

2.1.20 YearRemodAdd: Remodel date (same as construction date if no remodeling or additions)

There are no missing values in "YearRemodAdd".

```
summary(dataset$YearRemodAdd)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1950    1965    1993    1984    2004    2010
```

We plot "YearRemoddAdd" against "SalePrice" and observe that "YearRemodAdd" has a very similar correlation with "SalePrice" compared with "YearBuilt." We also note that remodellings were only recorded starting 1950.



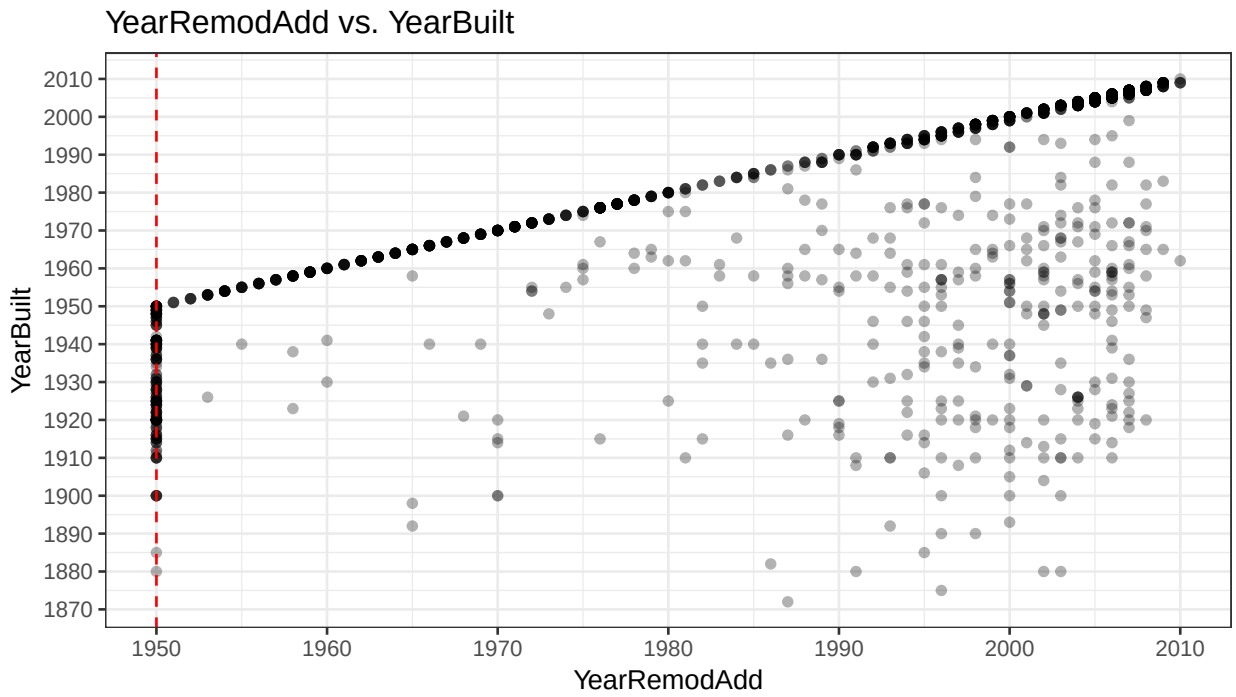
There is a large positive correlation between "YearRemodAdd" and "SalePrice".

```
cor(dataset[train$Id, ]$YearRemodAdd, dataset[train$Id, ]$SalePrice)
## [1] 0.507101
```

There is also a large correlation between "YearRemodAdd" and "YearBuilt".

```
cor(dataset$YearRemodAdd, dataset$YearBuilt)
## [1] 0.6122346
```

"YearRemodAdd" correlates $> 61\%$ with "YearBuilt." We plot the two variables against each other. From the plot it seems that for all houses built before 1950, even if there actually wasn't any remodelling done, they received an entry in "YearRemodAdd" at 1950 (see red dashed line).



Most houses have identical "YearBuilt" and "YearRemodAdd" values, since no remodellings were actually done. For predictive purposes, this variable isn't very helpful and is therefore dropped.

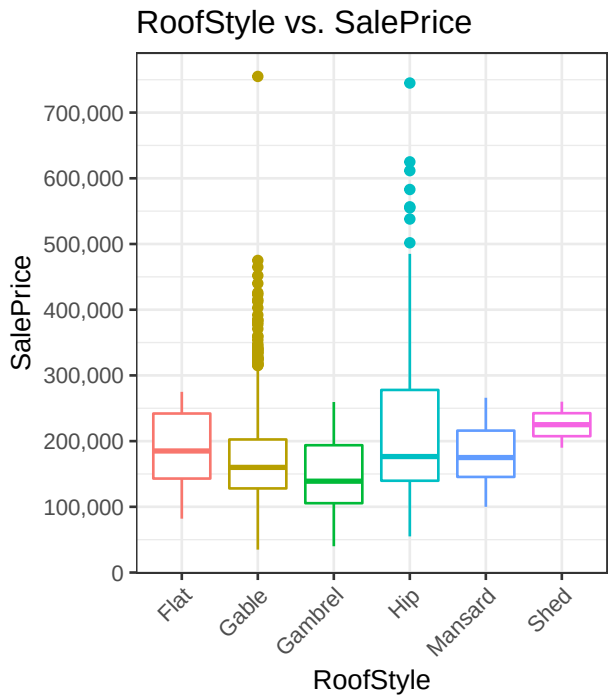
```
dataset <- subset(dataset, select = -YearRemodAdd)
```

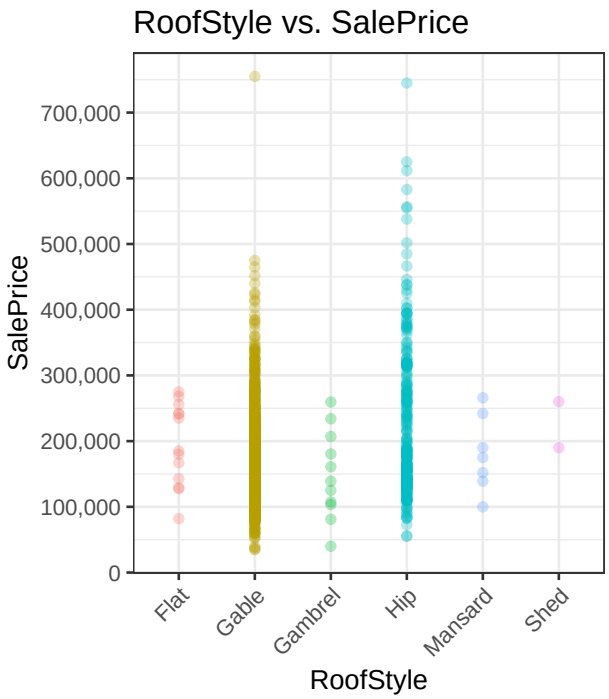
2.1.21 RoofStyle: Type of roof

There are no missing values in "RoofStyle".

```
summary(dataset$RoofStyle)
##    Flat   Gable Gambrel     Hip Mansard    Shed
##      20    2310      22     551      11       5
```

We plot "RoofStyle" against "SalePrice" and observe that the style of roof is not very predictive of "SalePrice", but most expensive houses are either in the "Gable" or "Hip" category. The "Shed" category is perhaps somewhat underrepresented.



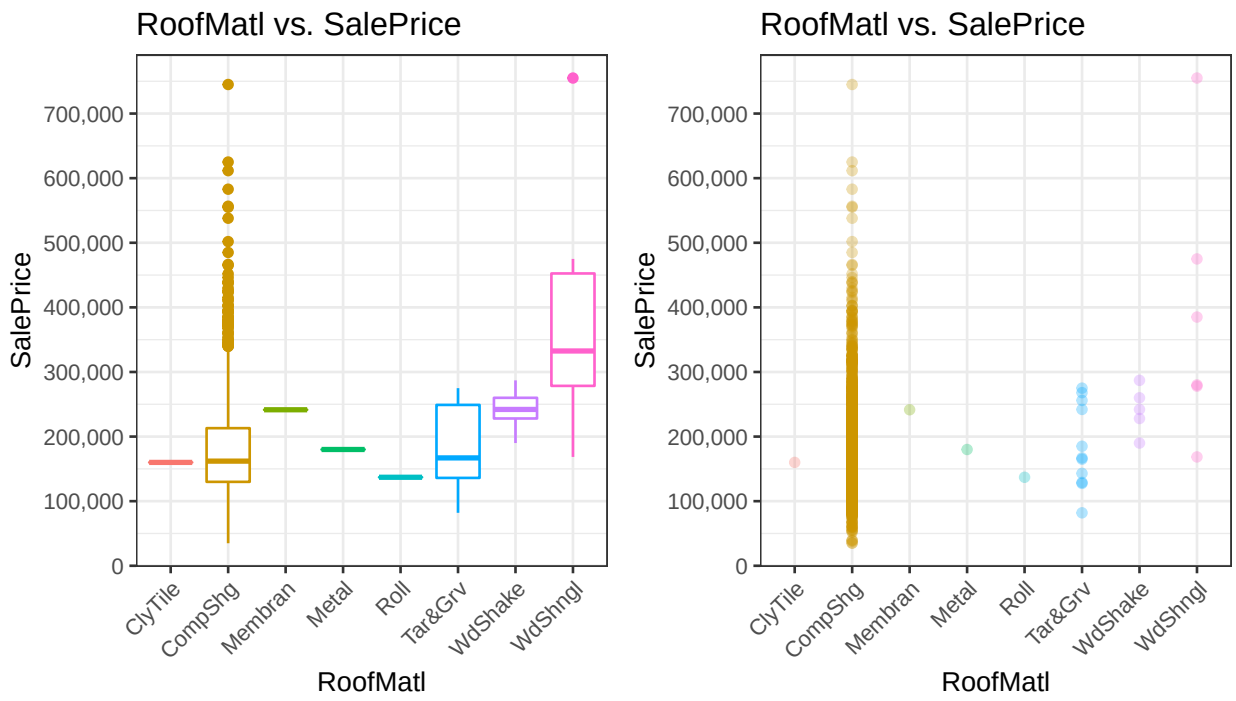


2.1.22 RoofMatl: Roof material

There are no missing values in "RoofMatl".

```
summary(dataset$RoofMatl)
## ClyTile CompShg Membran   Metal    Roll Tar&Grv WdShake WdShngl
##       1    2876       1       1       1      23       9       7
```

We plot "RoofMatl" against "SalePrice". From the plots we can see that there are very few houses with roof materials differing from the standard composite shingle. Most categories are populated by very few houses and are widely spread. It might be difficult to derive much predictive value from this feature as there are not enough data points in most categories.

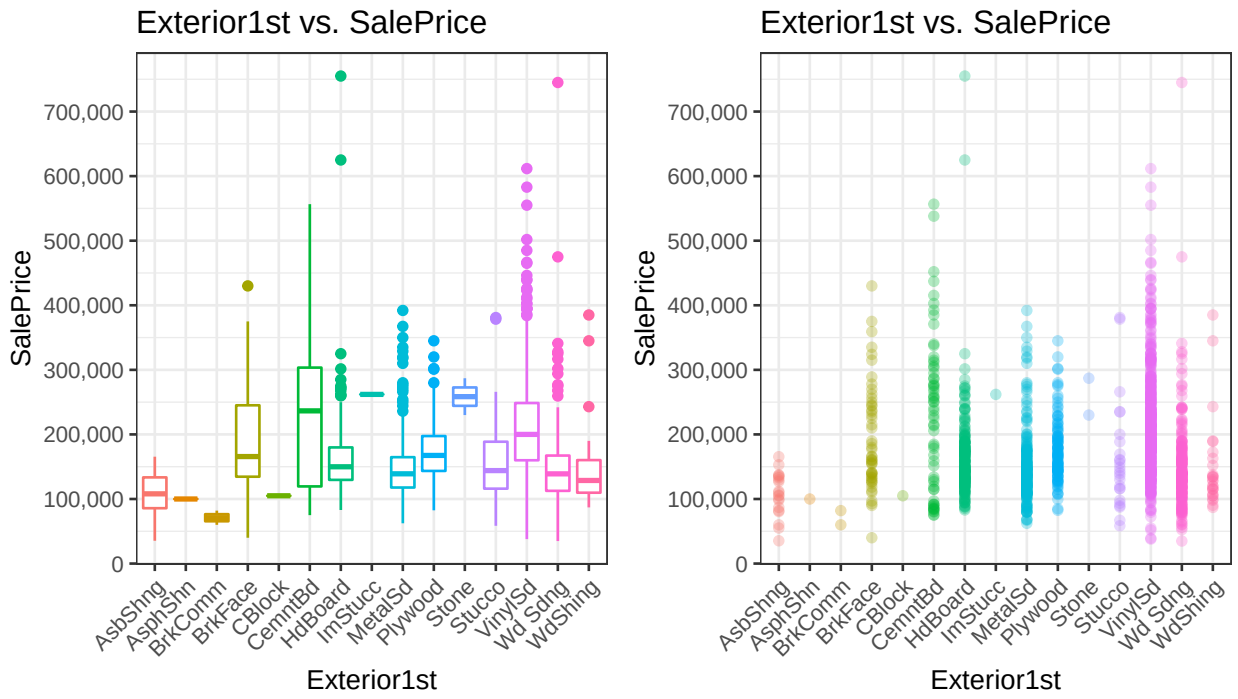


2.1.23 Exterior1st: Exterior covering on house

There is one missing value in "Exterior1st".

```
summary(dataset$Exterior1st)
## AsbShng AsphShn BrkComm BrkFace  CBlock CemntBd HdBoard ImStucc MetalSd
##      44       2       6      87       2     126     442       1     450
## Plywood   Stone  Stucco VinylSd Wd Sdng WdShing    NA's
##     221       2      43    1025     411      56       1
```

We plot "Exterior1st" against "SalePrice". From the plots it becomes apparent that asbestos shingles and asphalt shingles are mostly used with cheaper houses. Before we deal with the missing value, we take a look at "Exterior2nd" as well.

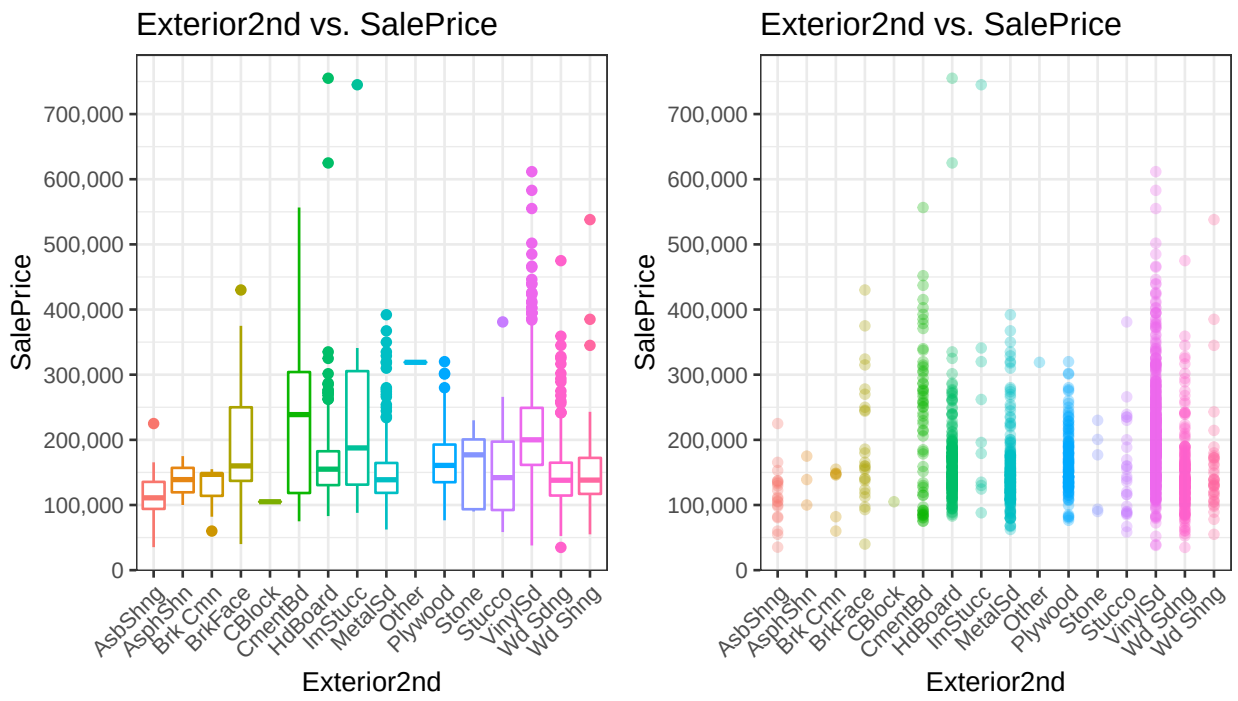


2.1.24 Exterior2nd: Exterior covering on house (if more than one material)

There is one missing value in "Exterior2nd".

```
summary(dataset$Exterior2nd)
## AsbShng AsphShn Brk Cmn BrkFace  CBlock CmentBd HdBoard ImStucc MetalSd
##      38       4      22      47       3     126     406      15     447
##   Other Plywood   Stone  Stucco VinylSd Wd Sdng Wd Shng    NA's
##       1     270       6      47    1014     391      81       1
```

From the plots it becomes apparent that asbestos shingles and asphalt shingles are mostly used with cheaper houses.



There is a missing value in "Exterior1st" and "Exterior2nd", is it the same house? It turns out that it is.

```
# THe value is missing for the same house.
dataset[which(is.na(dataset$Exterior1st)), ]$Id
## [1] 2152
dataset[which(is.na(dataset$Exterior2nd)), ]$Id
## [1] 2152
```

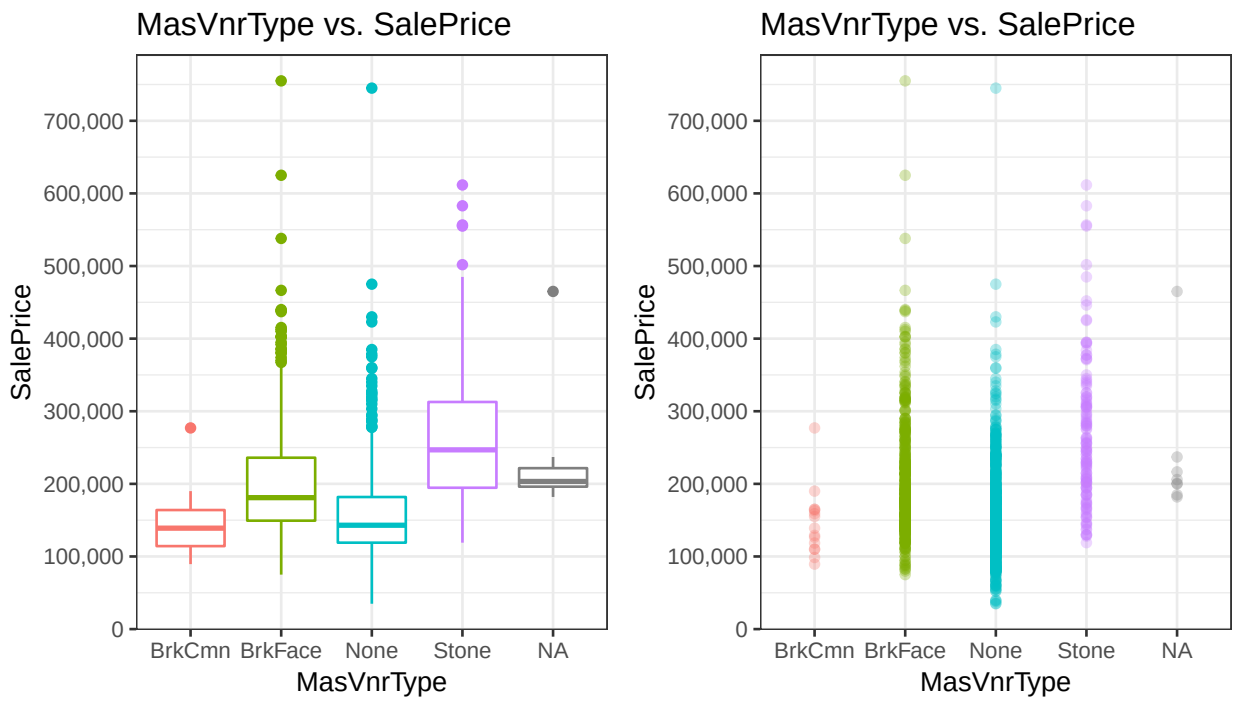
We use kNN-based imputation for "Exterior1st" and "Exterior2nd". The model predicts "Stucco" for this house.

```
# kNN-based model for Exterior1st and 2nd.
knn_model <- kNN(dataset, variable = c("Exterior1st", "Exterior2nd"), k = 9)

# Predicted Exerior1st value.
knn_model[knn_model$Exterior1st_imp == TRUE, ]$Exterior1st
## [1] Stucco
## 15 Levels: AsbShng AsphShn BrkComm BrkFace CBlock CemntBd ... WdShing

# Predicted Exerior2nd value.
knn_model[knn_model$Exterior2nd_imp == TRUE, ]$Exterior2nd
## [1] Stucco
## 16 Levels: AsbShng AsphShn Brk Cmn BrkFace CBlock CmentBd ... Wd Shng

# Missing value imputation.
dataset$Exterior1st[2152] <- knn_model[knn_model$Exterior1st_imp == TRUE, ]$Exterior1st
dataset$Exterior2nd[2152] <- knn_model[knn_model$Exterior2nd_imp == TRUE, ]$Exterior2nd
```

2.1.25 MasVnrType: Masonry veneer type

There are some missing values in "MasVnrType".

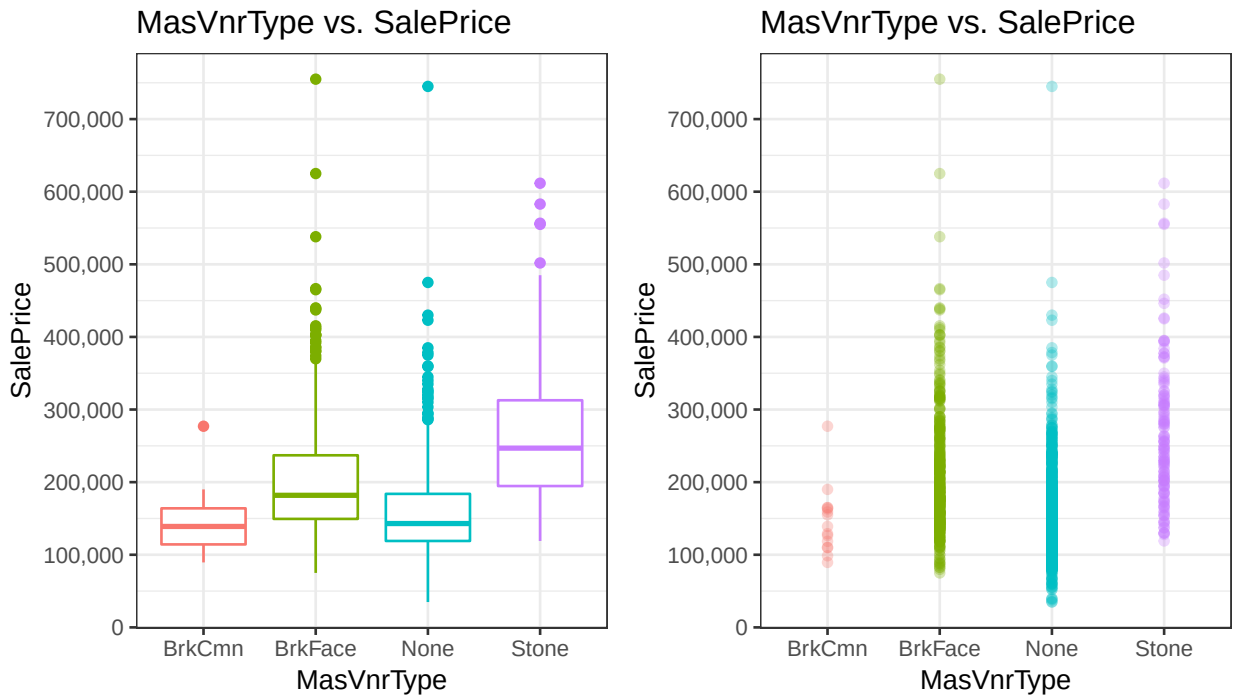
```
summary(dataset$MasVnrType)
##  BrkCmn BrkFace    None   Stone    NA's
##      25     879    1742     249      24
```



We will use kNN-based predictions to impute the missing values and re-plot the variable against "SalePrice".

```
# kNN-model.
knn_model <- kNN(dataset, variable = "MasVnrType", k = 9)

# Predicted "MasVnrType" values.
knn_model[knn_model$MasVnrType_imp == TRUE, ]$MasVnrType
##  [1] None    None    None    None    None    None    BrkFace BrkFace
##  [9] BrkFace BrkFace None    None    BrkFace None    None    None
## [17] None    Stone   None    Stone   None    None    BrkFace None
## Levels: BrkCmn BrkFace None Stone

# Missing value imputation of MasVnrType
dataset[which(is.na(dataset$MasVnrType)), ]$MasVnrType <-
              knn_model[knn_model$MasVnrType_imp == TRUE, ]$MasVnrType
```

From the plots we can see that "Stone" can be predictive of a higher "SalePrice", as it has the highest median value. "BrkCmn" is less populated with values, but shows the lowest median value.



2.1.26 MasVnrArea: Masonry veneer area in square feet

There are some missing values in "MasVnrArea", presumably the same houses that had missing values in "MasVnrType".

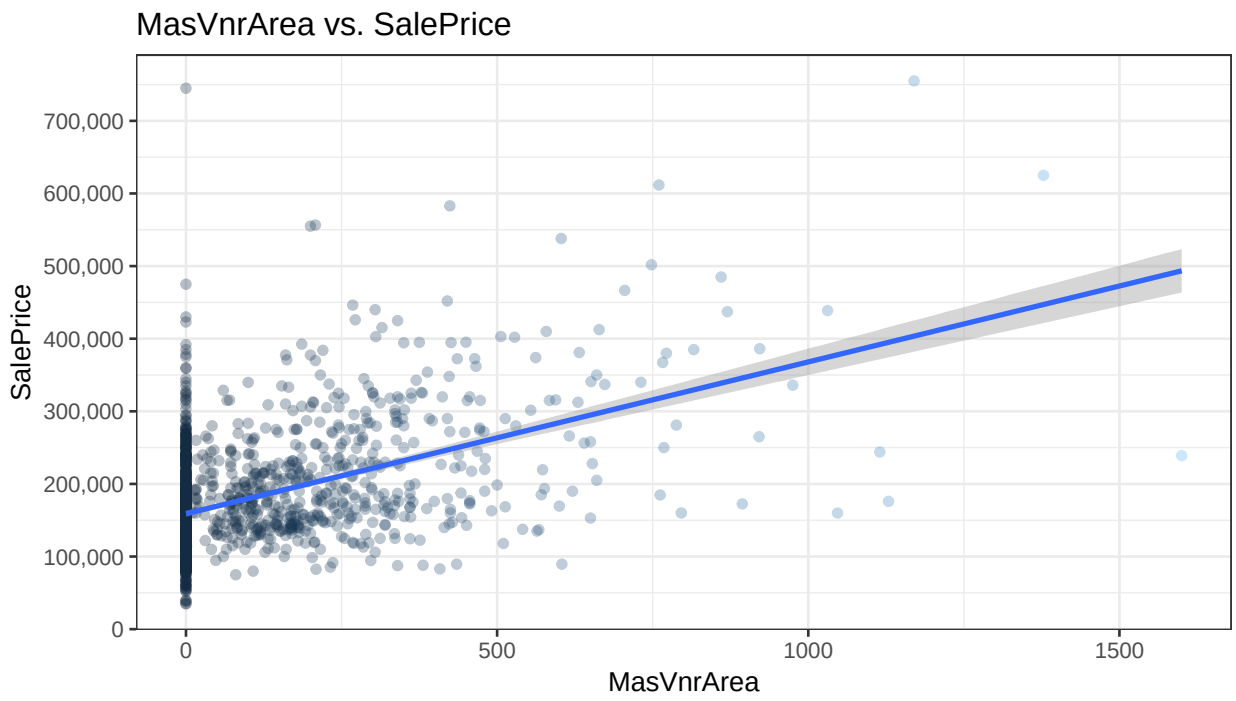
```
summary(dataset$MasVnrArea)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##     0.0     0.0     0.0   102.2   164.0  1600.0      23
```

"MasVnrArea" should be a numeric variable.

```
dataset$MasVnrArea <- as.numeric(dataset$MasVnrArea)
```

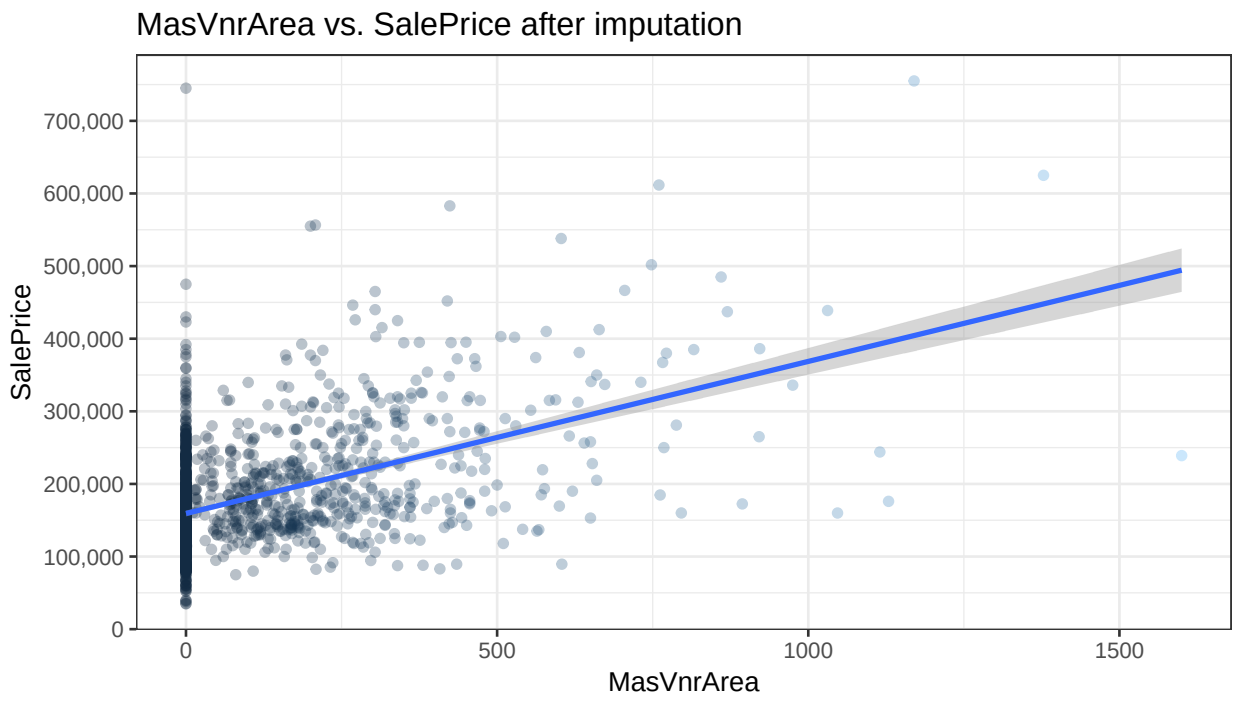
From the plot we can see that many houses actually have exactly zero "MasVnrArea", which makes sense as there's a level "None" in "MasVnrType".

```
## Warning: Removed 8 rows containing non-finite values (stat_smooth).
## Warning: Removed 8 rows containing missing values (geom_point).
```



We will use kNN-based predictions to impute the missing values.

```
# kNN-model.
knn_model <- kNN(dataset, variable = "MasVnrArea", k = 9)

# Predicted "MasVnrArea" values.
knn_model[knn_model$MasVnrArea_imp == TRUE, ]$MasVnrArea
##  [1]   0   0   0   0   0   0 304   0 126 140   0   0 196  95   0   0   0
## [18] 164   0 132   0 215   0

# Missing value imputation of MasVnrArea
dataset[which(is.na(dataset$MasVnrArea)), ]$MasVnrArea <-
  knn_model[knn_model$MasVnrArea_imp == TRUE, ]$MasVnrArea
```



It is however still possible that there are non-zero "MasVnrArea" values where "MasVnrType" is equal to "None". We will check for this and set all of these cases to zero. Indeed, several houses have an entry in "MasVnrArea" while having "MasVnrType" of "None". As it is unclear which value might be wrong, we will set "MasVnrArea" to zero in these cases. The opposite might also be possible, where a house has a type other than "None", but has an area value larger than zero. Fot hose cases we will change "MasVnrType" to "None".

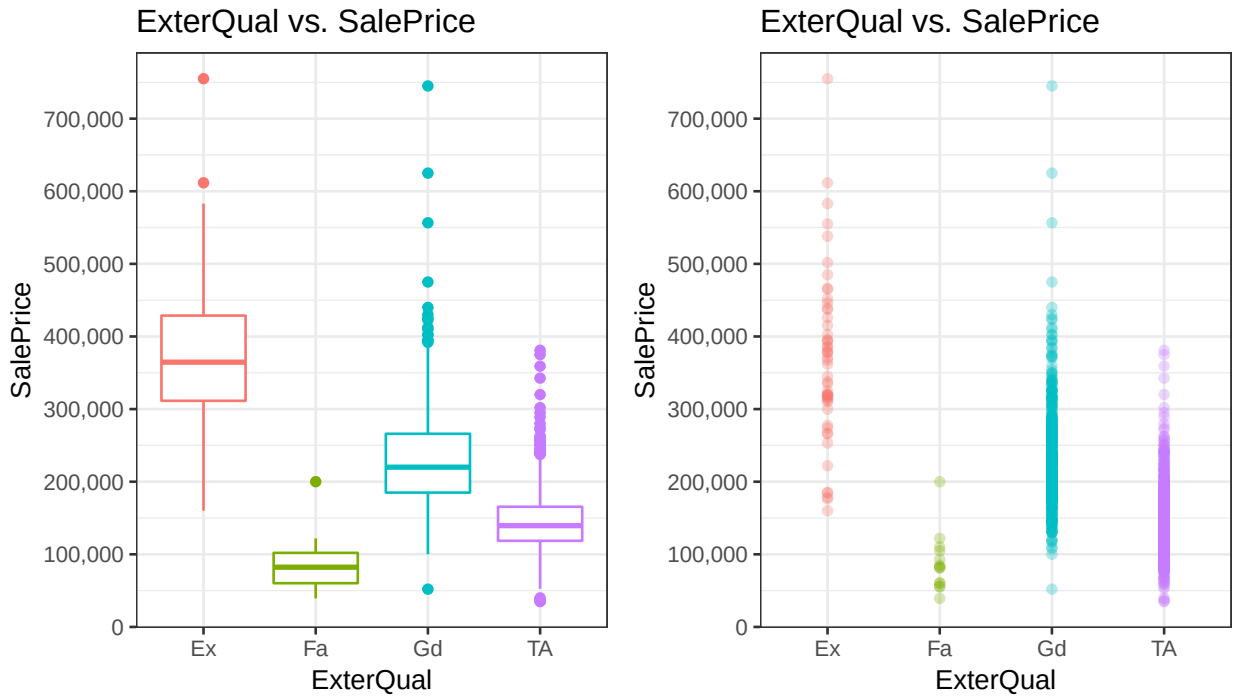
```
# There are houses with non-zero MasVnrArea but "None" as MasVnrType.
# dataset[which(dataset$MasVnrArea != 0 & dataset$MasVnrType == "None"), ]

# We take the index.
ind1 <- dataset[which(dataset$MasVnrArea != 0 & dataset$MasVnrType == "None"), ]$Id


# There are also houses with zero MasVnrArea and anything but "None" as MasVnrType.
# dataset[which(dataset$MasVnrArea == 0 & dataset$MasVnrType != "None"), ]

# We take the index as well.
ind2 <- dataset[which(dataset$MasVnrArea == 0 & dataset$MasVnrType != "None"), ]$Id


# We will set the area to zero and the type to "None" in these cases.
dataset$MasVnrArea[ind1] <- 0
dataset$MasVnrType[ind2] <- "None"
```

2.1.27 ExterQual: Evaluates the quality of the material on the exterior

There are no missing values in "ExterQual".

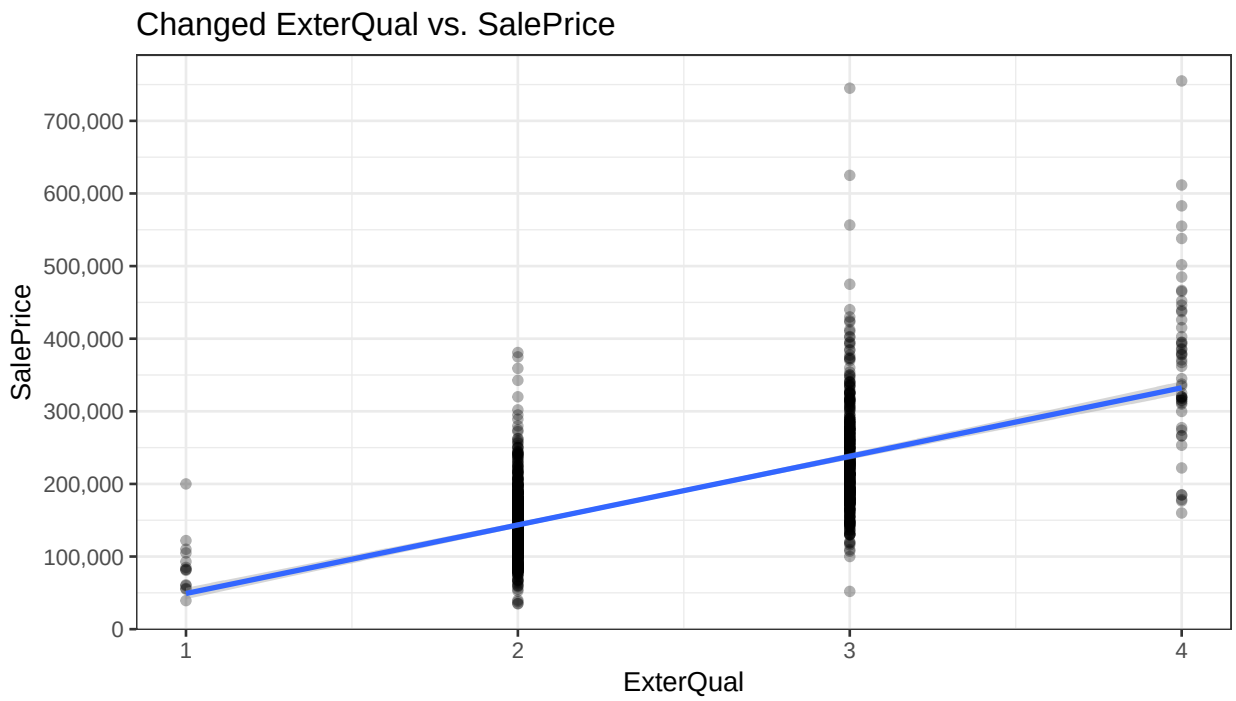
```
summary(dataset$ExterQual)
##   Ex   Fa   Gd   TA
##  107   35  979 1798
```

We plot "ExterQual" against "SalePrice" and observe that "ExterQual" is a relatively good predictor of "SalePrice", although this isn't immediately very clear due to factor level order.



We transform this qualitative ordinal factor into a numeric containing the present levels and plot it again. It now becomes easily apparent that "ExterQual" strongly correlates with "SalePrice".

```
dataset$ExterQual <- as.numeric(factor(dataset$ExterQual, levels=c("Fa",
                                                    "TA", "Gd", "Ex")))
```

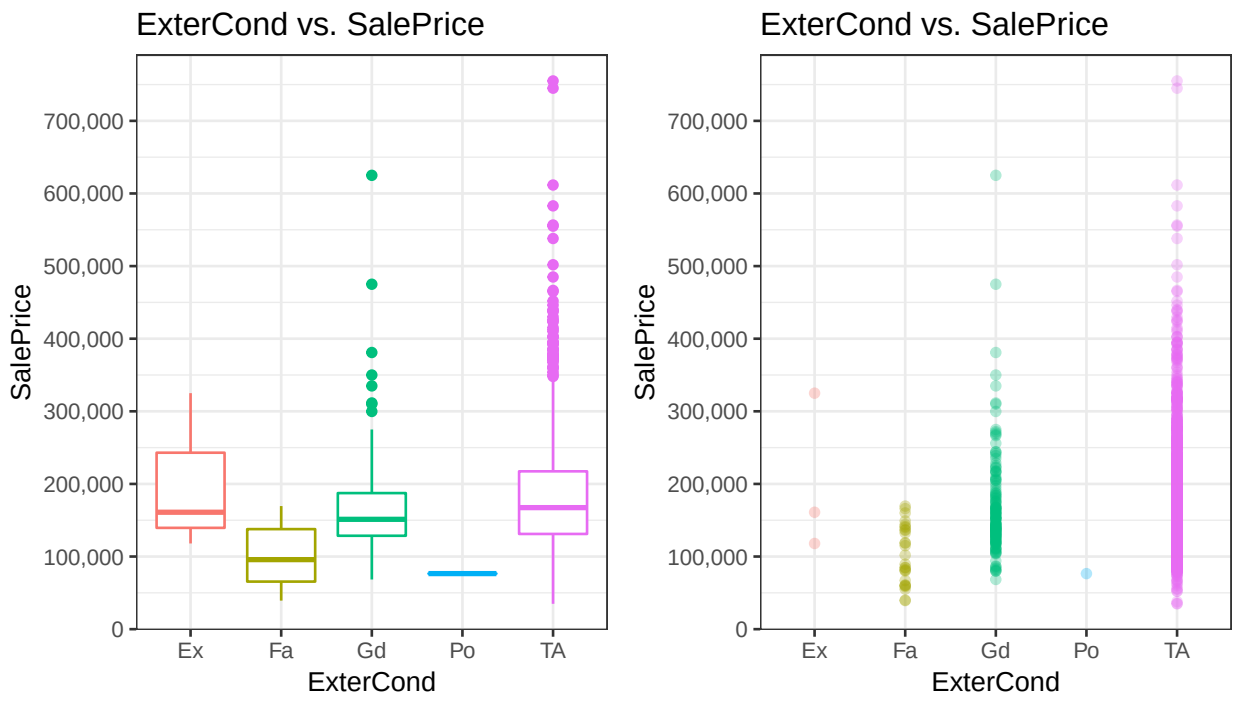


2.1.28 ExterCond: Evaluates the quality of the material on the exterior

There are no missing values in "ExterCond".

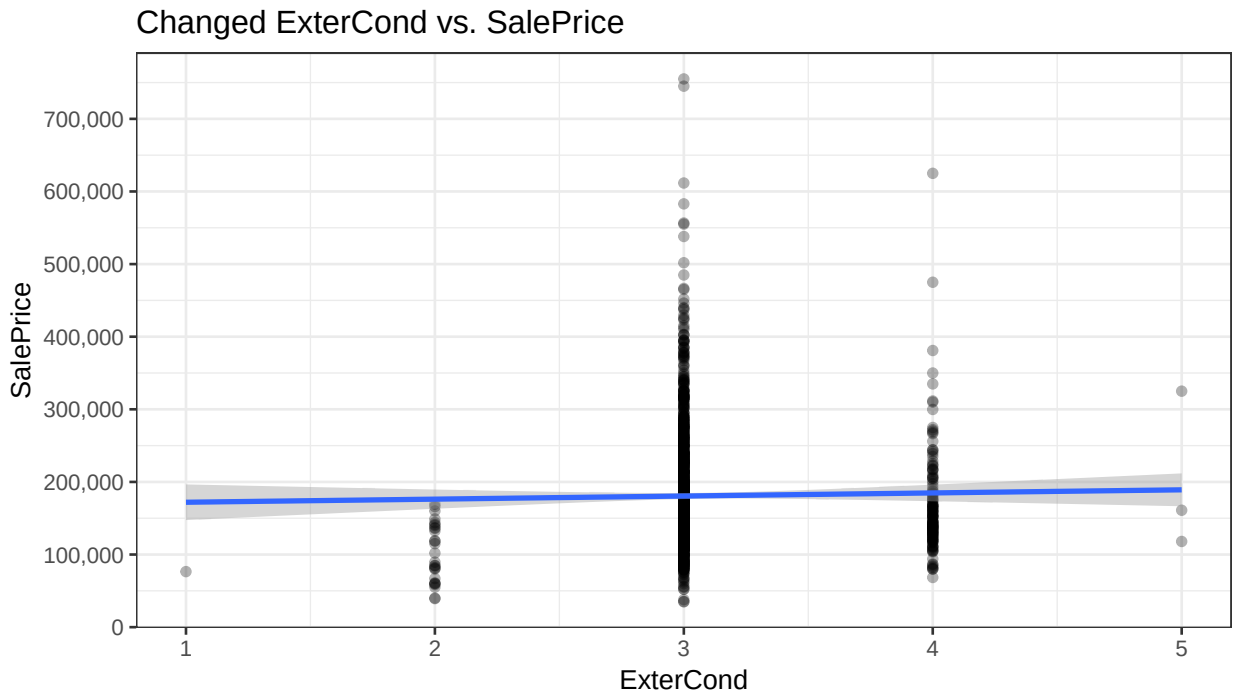
```
summary(dataset$ExterCond)
##   Ex   Fa   Gd   Po   TA
##   12   67  299    3 2538
```

We plot "ExterCond" against "SalePrice" and observe that only a few houses have a "poor" or "excellent" "ExterCond." "ExterCond" is a a weaker predictor of "SalePrice" compared to "ExterQual", similar to "OverallQual" vs. "OverallCond".



We transform this qualitative ordinal factor into a numeric containing the present levels and plot it again. We can see that unlike with "ExterQual", "ExterCond" is not as straightforward a predictor of "SalePrice", as a "typical" value contains most expensive houses. The best and worst category is very sparsely populated. "ExterCond" might be a questionable predictor for "salePrice".

```
dataset$ExterCond <- as.numeric(factor(dataset$ExterCond, levels=c("Po",
                                                    "Fa", "TA", "Gd", "Ex")))
```

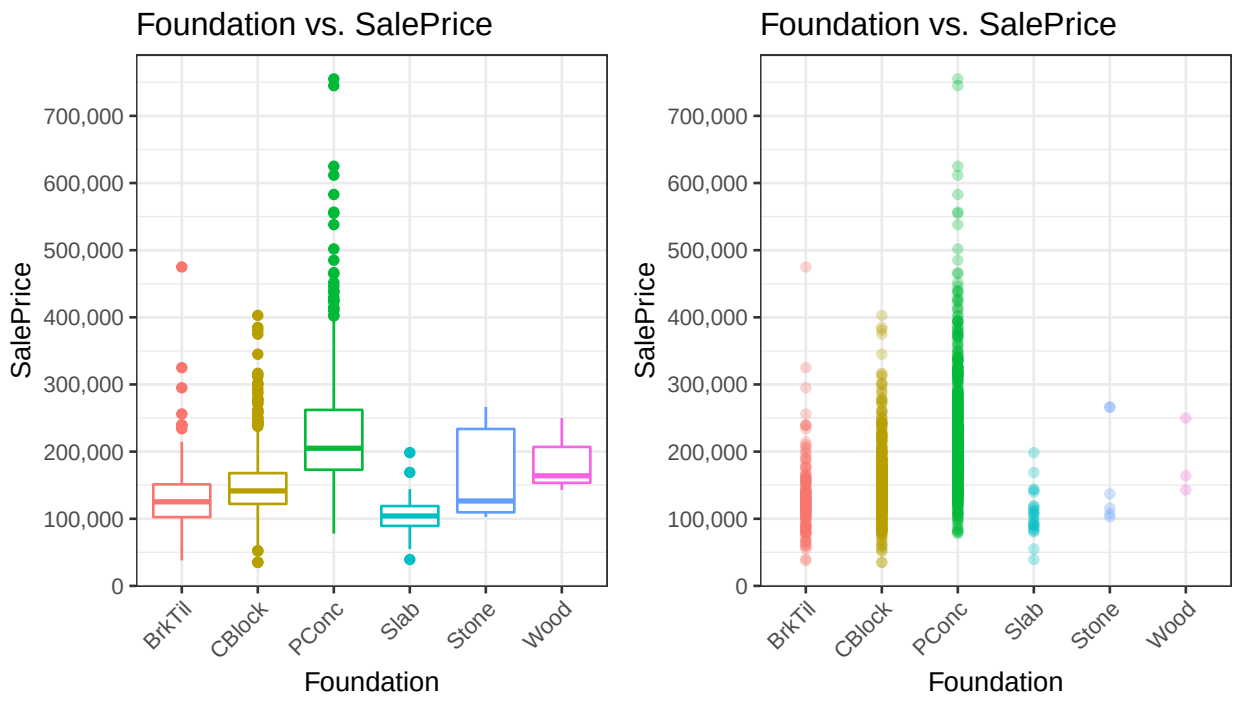


2.1.29 Foundation: Type of foundation

There are no missing values in "Foundation".

```
summary(dataset$Foundation)
## BrkTil CBlock  PConc   Slab  Stone   Wood
##    311   1235   1308     49     11      5
```

When plotting "Foundation" against "SalePrice" we notice that stone and wood "Foundation" values are relatively rare. "Poured concrete"" ("PConc") seems to be the material of choice for houses with higher "SalePrice", while "Slab" seems to be associated with lower values.



2.1.30 BsmtQual: Evaluates the height of the basement

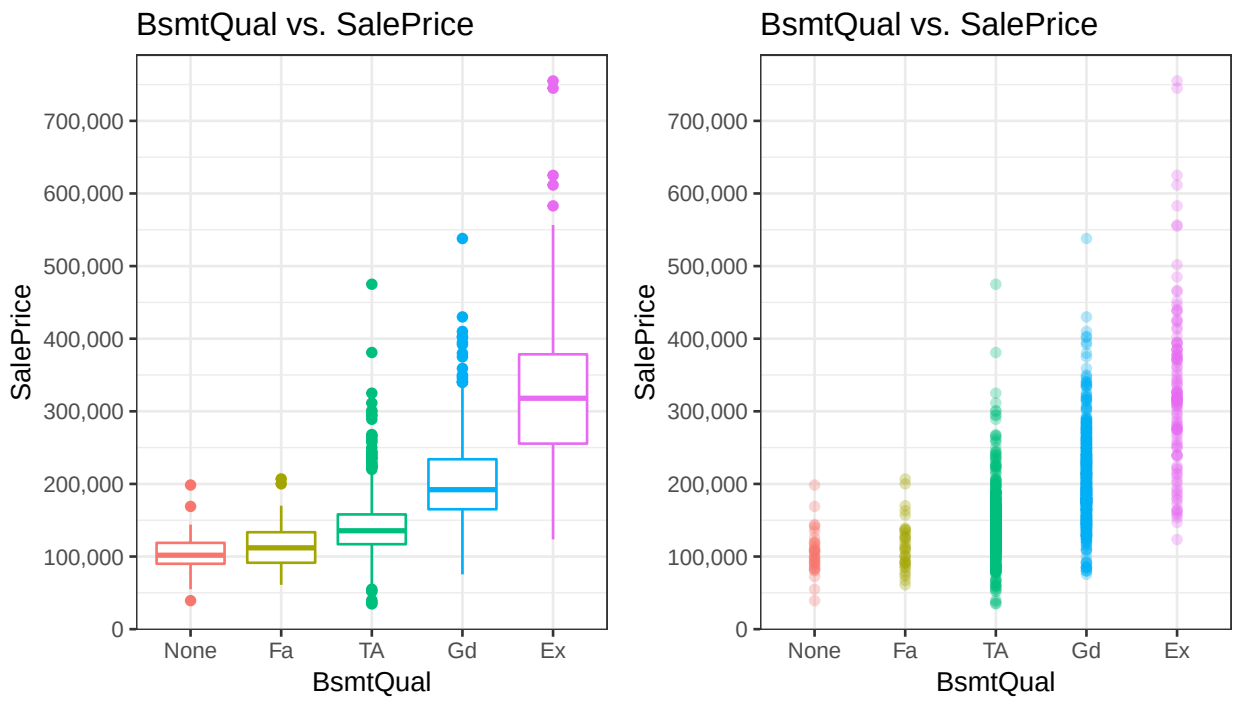
There are missing values in "BsmtQual".

```
summary(dataset$BsmtQual)
##   Ex   Fa   Gd   TA NA's
##  258   88 1209 1283   81
```

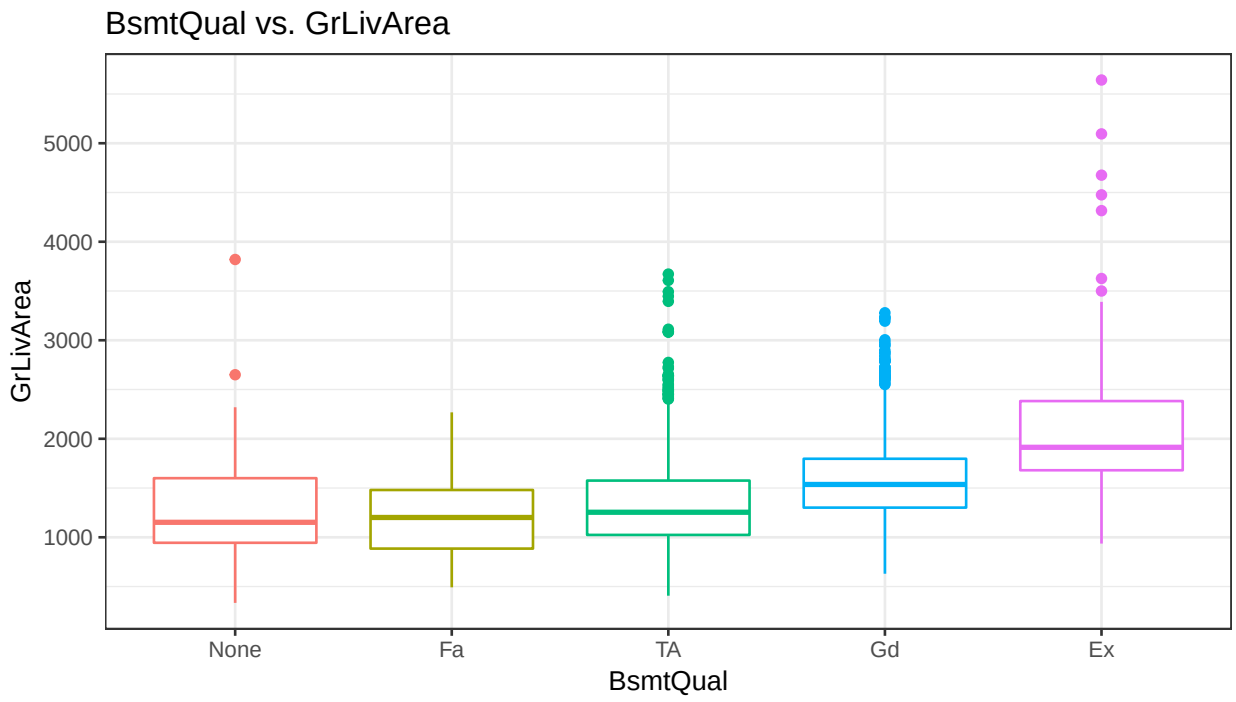
From the data description we know that "No Basement" was encoded as "NA" in this variable. Also, there is no house with a "poor" "BsmtQual". We will therefore replace "NA" with "None" and fix the factor levels accordingly.

```
# We convert to character prior to editting and revert to a factor with appropriate levels.
dataset$BsmtQual <- as.character(dataset$BsmtQual)
dataset$BsmtQual <- str_replace_na(dataset$BsmtQual, replacement = "None")
dataset$BsmtQual <- factor(dataset$BsmtQual, levels = c("None",
                                                        "Fa", "TA", "Gd", "Ex"))
```

From the plots we can see that houses with no basement or a basement with only "fair" quality come with much lower "SalePrice". This might also be because a higher "BsmtQual" indicates a greater height of the basement, whichmight be indicative of a larger house in general.



We plot "BsmtQual" vs. "GrLivArea", as an indication of house size. There seems to be a slight relationship.



2.1.31 BsmtCond: Evaluates the general condition of the basement

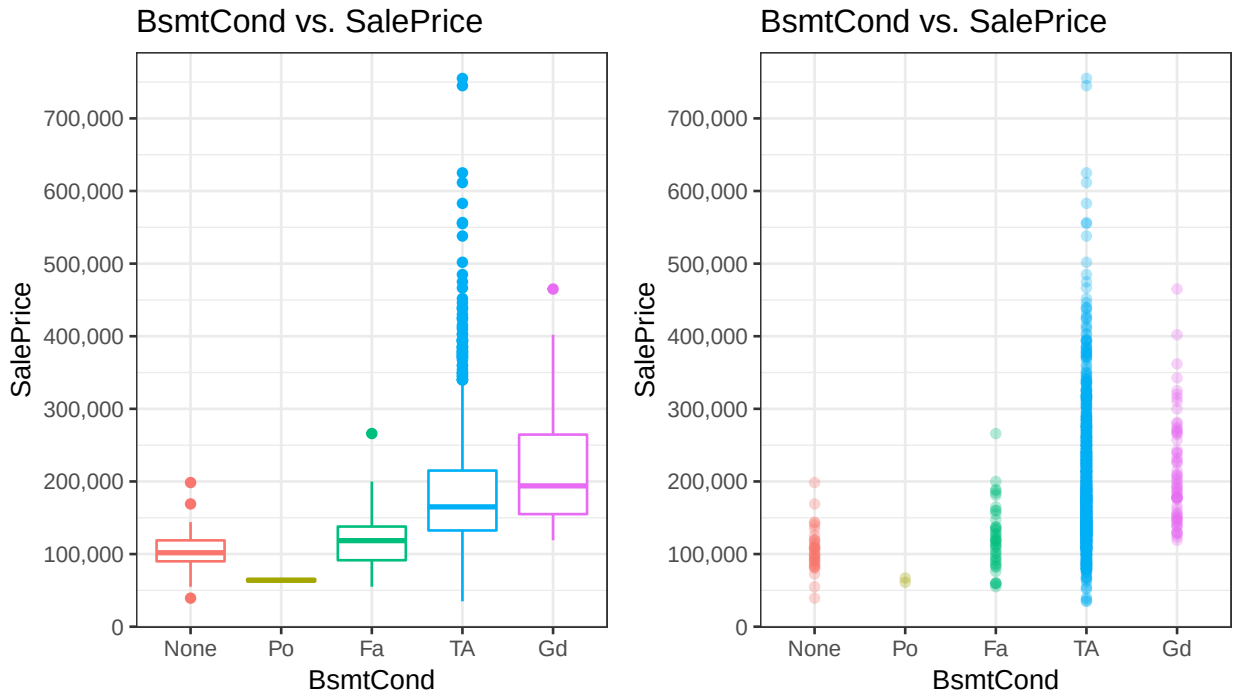
There are missing values in "BsmtCond". From the data description we know that "No Basement" was encoded as "NA" in this variable. Also, not a single house has an "excellent" basement condition.

```
summary(dataset$BsmtCond)
##   Fa   Gd   Po   TA NA's
##  104  122    5 2606   82
```

We will therefore replace "NA" with "None" and fix the factor levels accordingly.

```
dataset$BsmtCond <- as.character(dataset$BsmtCond)
dataset$BsmtCond <- str_replace_na(dataset$BsmtCond, replacement = "None")
dataset$BsmtCond <- factor(dataset$BsmtCond, levels = c("None",
                                                        "Po", "Fa", "TA", "Gd"))
```

From plotting "BsmtCond" against "SalePrice" we can tell that "BsmtCond" seems to be a good indicator of "SalePrice. Although there are only a few houses with a really"poor" basement condition, all of these houses have exceptionally low "SalePrice". Overall, a "typical" or "good" basement condition is important for a houses "SalePrice.



2.1.32 BsmtExposure: Refers to walkout or garden level walls

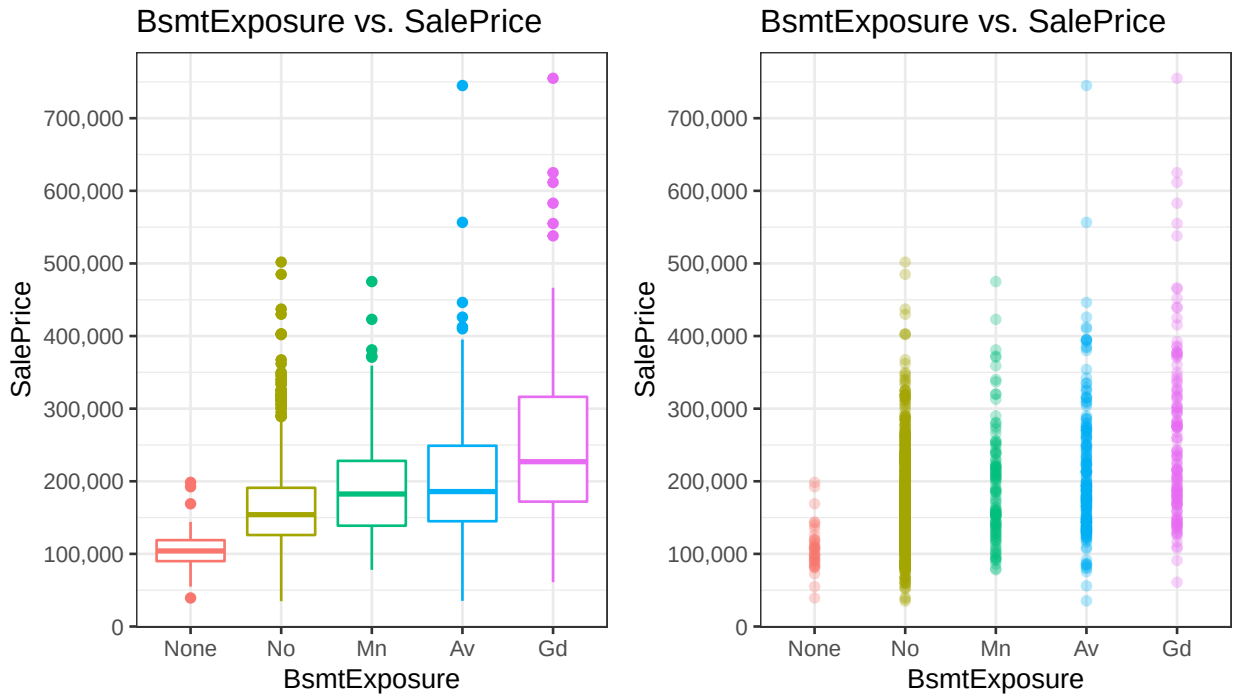
There are missing values in "BsmtExposure". From the data description we know that "No Basement" was encoded as "NA" in this variable. "No exposure" is encoded as "No"

```
summary(dataset$BsmtExposure)
##   Av   Gd   Mn   No NA's
##  418  276  239 1904   82
```

We will therefore replace "NA" with "None" and fix the factor levels accordingly.

```
dataset$BsmtExposure <- as.character(dataset$BsmtExposure)
dataset$BsmtExposure <- str_replace_na(dataset$BsmtExposure, replacement = "None")
dataset$BsmtExposure <- factor(dataset$BsmtExposure, levels = c("None",
                                                                "No", "Mn", "Av", "Gd"))
```

From the plots we can see that greater basement exposure correlates slightly with larger "SalePrice". Houses with no basement ("None") have the lowest "SalePrice".



2.1.33 BsmtFinType1: Rating of basement finished area

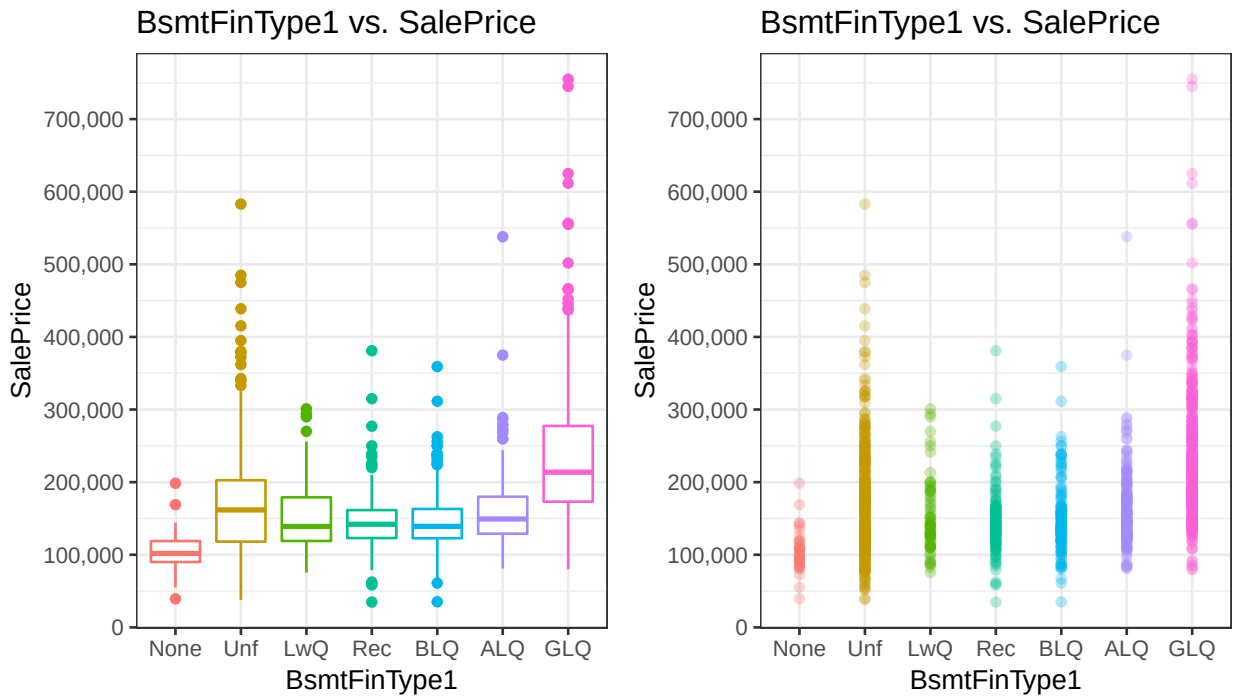
There are missing values in "BsmtExposure". From the data description we know that "No Basement" was encoded as "NA" in this variable.

```
summary(dataset$BsmtFinType1)
##  ALQ  BLQ  GLQ  LwQ  Rec  Unf NA's
##  429  269  849  154  288  851   79
```

We will therefore replace "NA" with "None" and fix the factor levels accordingly.

```
dataset$BsmtFinType1 <- as.character(dataset$BsmtFinType1)
dataset$BsmtFinType1 <- str_replace_na(dataset$BsmtFinType1, replacement = "None")
dataset$BsmtFinType1 <- factor(dataset$BsmtFinType1, levels = c("None", "Unf",
                                                                "LwQ", "Rec", "BLQ", "ALQ", "GLQ"))
```

We plot "BsmtFinType1" against "SalePrice" and observe that houses with unfinished ("Unf") or good quality living quarter ("GLQ") type of finish come with higher "SalePrice".



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

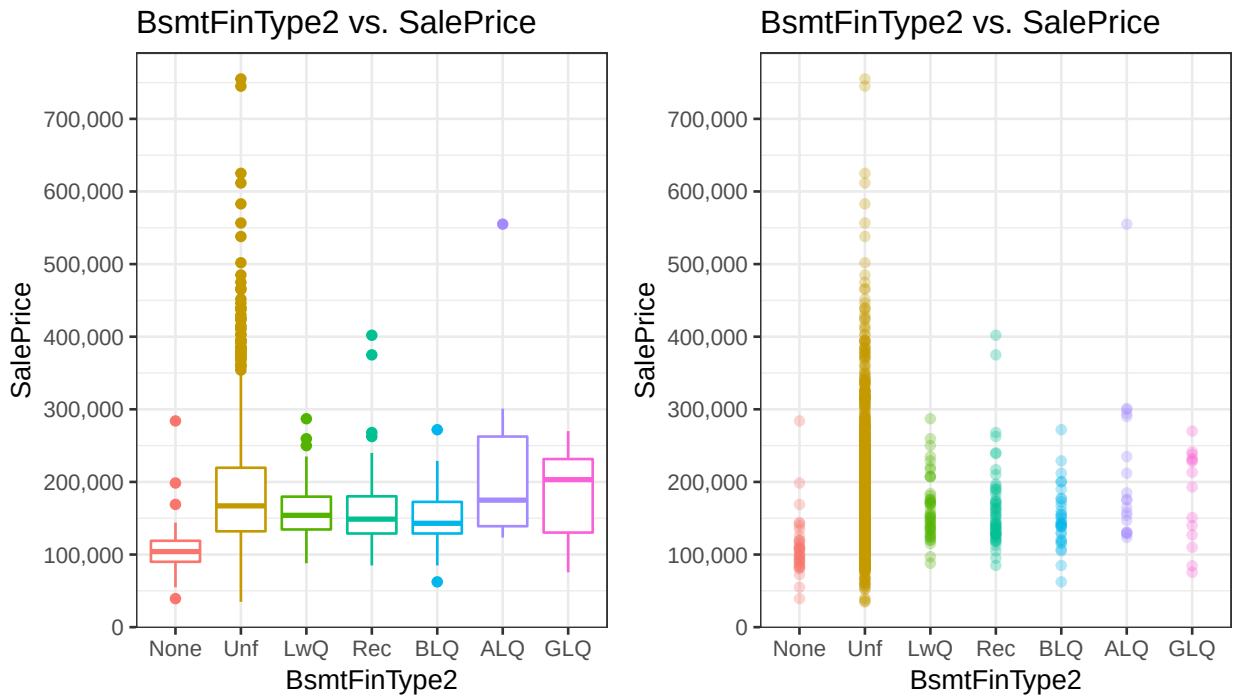
There are missing values in BsmtExposure. From the data description we know that "No Basement" was encoded as "NA" in this variable.

```
summary(dataset$BsmtFinType2)
##  ALQ  BLQ  GLQ  LwQ  Rec  Unf NA's
##   52   68   34   87  105 2493   80
```

We will therefore replace "NA" with "None" and fix the factor levels accordingly.

```
dataset$BsmtFinType2 <- as.character(dataset$BsmtFinType2)
dataset$BsmtFinType2 <- str_replace_na(dataset$BsmtFinType2, replacement = "None")
dataset$BsmtFinType2 <- factor(dataset$BsmtFinType2, levels = c("None",
                                        "Unf", "LwQ", "Rec", "BLQ", "ALQ", "GLQ"))
```

From the plots we can see that houses with unfinished type of finish come with higher "SalePrice".



2.1.35 Basement square feet variables

We will deal with all basement-area related variables at the same time here. We replace the missing Bsmt-values of house 2121 with 0, as this house has no basement.

```
# dataset[2121, ] # This house actually has no basement.
# We impute zeroes for these variables.

dataset$BsmtFinSF1[2121] <- 0
dataset$BsmtFinSF2[2121] <- 0
dataset$BsmtUnfSF[2121] <- 0
dataset$TotalBsmtSF[2121] <- 0
```

We replace the remaining NAs in "BsmtFullBath" and "BsmtHalfBath" with 0 as the respective houses have no basement.

```
# dataset[which(is.na(dataset$BsmtFullBath)), ] # These houses have no basement.

dataset$BsmtFullBath[is.na(dataset$BsmtFullBath)] <- 0
dataset$BsmtHalfBath[is.na(dataset$BsmtHalfBath)] <- 0
```

What is the correlation between "TotalBsmtSF" and the individual measurements of basement square feet? It turns out that the correlation is exactly 1, so "TotalBsmtSF" consists of all the individual values.

```
# Correlation between "TotalBsmtSF" and the individual variables is exactly 1.
cor(dataset$TotalBsmtSF, (dataset$BsmtFinSF1 + dataset$BsmtFinSF2 + dataset$BsmtUnfSF))
## [1] 1
```

Correlation between the individual variables and "SalePrice": They are all mostly weak individually, while "TotalBsmtSF" is highly correlated with "SalePrice".

```r
# Correlation between "TotalBsmtSF" and "SalePrice".
cor(dataset[train$Id, ]$TotalBsmtSF, dataset[train$Id, ]$SalePrice)
## [1] 0.6135806

# Correlation between "BsmtFinSF1" and "SalePrice".
cor(dataset[train$Id, ]$BsmtFinSF1, dataset[train$Id, ]$SalePrice)
## [1] 0.3864198

# Correlation between "BsmtFinSF2" and "SalePrice".
cor(dataset[train$Id, ]$BsmtFinSF2, dataset[train$Id, ]$SalePrice)
## [1] -0.01137812

# Correlation between "BsmtUnfSF" and "SalePrice", which is relatively strong.
cor(dataset[train$Id, ]$BsmtUnfSF, dataset[train$Id, ]$SalePrice)
## [1] 0.2144791
```

The individual basement area variables seem to be redundant, but we observed that having an unfinished basement can be indiciative of a higher "SalePrice" earlier (e.g. relatively large correlation between "BsmtUnfSF" and "SalePrice"). Nevertheless, we will remove the individual measurements.

```r
# Removing the individual basement square feet variables.
dataset <- subset(dataset, select = -c(BsmtFinSF1, BsmtFinSF2, BsmtUnfSF))
```

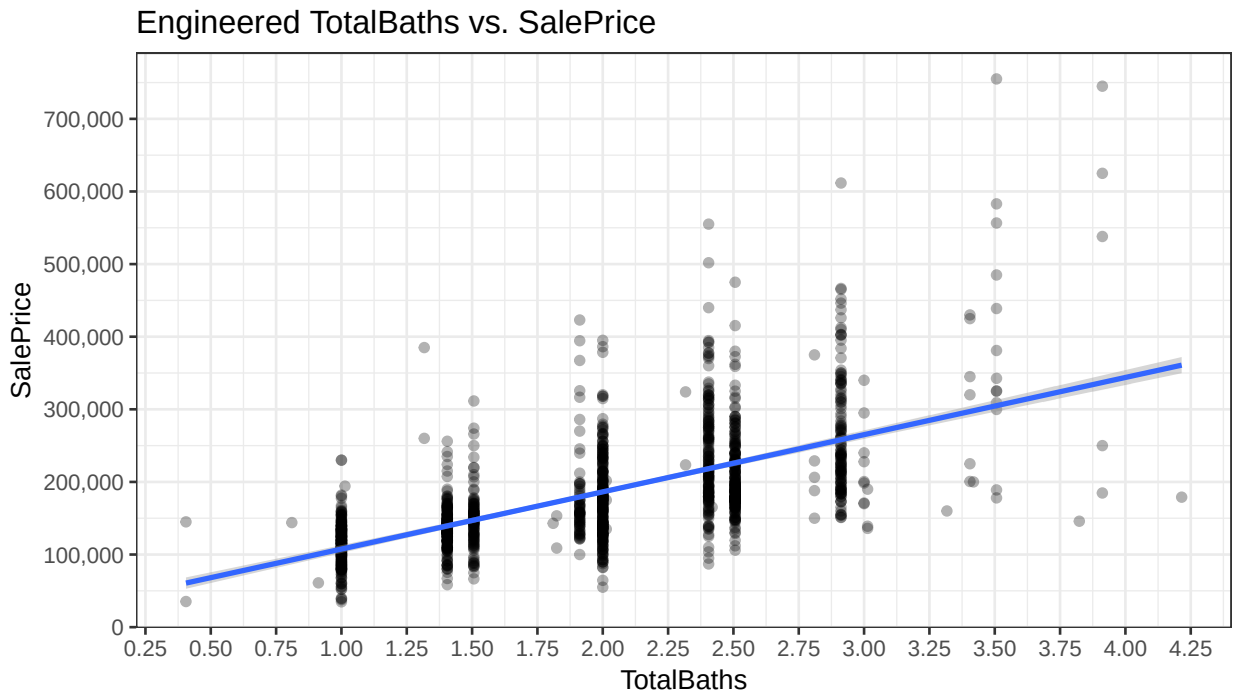
2.1.36 Bathroom variables

The number of total bathrooms can be indicative of a houses size. In the dataset there are 4 different variables describing bathrooms. It could be useful to combine these into a single feature. While each bathroom variable individually has little influence on "SalePrice", combined they might become a stronger predictor. We will value different types of bath according to their "SalePrice" correlation ratio compared to "FullBaths". As shown below, we will create a new variable "TotalBaths" and drop the individual variables afterwards. The contribution of different types of baths (i.e. "FullBath", "HalfBath", "BsmtFullBath", "BsmtHalfBath") are weighted according to their correlation with "SalePrice".

```r
# Full bathrooms have the strongest correlation with "SalePrice"
cor(dataset[train$Id, ]$FullBath, dataset[train$Id, ]$SalePrice)
## [1] 0.5606638

# Half baths are much less valued, about half as much.
cor(dataset[train$Id, ]$HalfBath, dataset[train$Id, ]$SalePrice)
## [1] 0.2841077

# We calculate the ratio of the correlations of Full and Half baths to "SalePrice" to use
# as a weighting factor below.
(FullToHalfBathRatio <- cor(dataset[train$Id, ]$HalfBath,
                            dataset[train$Id, ]$SalePrice) /
                              cor(dataset[train$Id, ]$FullBath,
                                dataset[train$Id, ]$SalePrice))
## [1] 0.5067345

# Basement full and especially half baths have weaker correlation with "SalePrice".
cor(dataset[train$Id, ]$BsmtFullBath, dataset[train$Id, ]$SalePrice)
## [1] 0.2271222
```

```r
cor(dataset[train$Id, ]$BsmtHalfBath, dataset[train$Id, ]$SalePrice)
## [1] -0.01684415

# Weighting factor for "FullBath" to "BsmtFullBath." A basement full bath
# receives a weighting factor of 0.3971685 as calculated below.
(FullToBsmtFullBathRatio <- cor(dataset[train$Id, ]$BsmtFullBath,
                                dataset[train$Id, ]$SalePrice) /
                                  cor(dataset[train$Id, ]$FullBath,
                                    dataset[train$Id, ]$SalePrice))
## [1] 0.4050953

# Weighting factor for "FullBath" to "BsmtHalfBath." This actualy returns
# a negative value! We rather remove basement half baths from the calculations.
(FullToBsmtHalfBathRatio <- cor(dataset[train$Id, ]$BsmtHalfBath,
                                dataset[train$Id, ]$SalePrice) /
                                  cor(dataset[train$Id, ]$FullBath,
                                      dataset[train$Id, ]$SalePrice))
## [1] -0.03004324

# Only very few houses even have "BsmtHalfBaths".
summary(dataset$BsmtHalfBath)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.00000 0.00000 0.06132 0.00000 2.00000

# We will create a variable "TotalBaths", that sums up the various types of baths
# into one variable, taking into consideration the different correlations to "SalePrice"
# for full and half baths. We don't add "BsmtHalfBaths", as their correlation
# with "SalePrice" is very low/negative and only very few houses have any.
dataset$TotalBaths <- dataset$FullBath +
  (dataset$BsmtFullBath * FullToBsmtFullBathRatio) +
  (dataset$HalfBath * FullToHalfBathRatio)

# Correlation of "TotalBaths" with "SalePrice" is higher than of just "FullBaths".
cor(dataset[train$Id, ]$TotalBaths, dataset[train$Id, ]$SalePrice)
## [1] 0.6522255

# The new TotalBaths variable has a correlation of almost 70% with "SalePrice".

# We can remove the previous bath variables.
dataset <- subset(dataset, select = -c(FullBath, HalfBath, BsmtFullBath, BsmtHalfBath))

# Cleanup.
rm(FullToBsmtFullBathRatio, FullToBsmtHalfBathRatio, FullToHalfBathRatio)
```

The combination of 4 different bath variables, “TotalBaths”, has a strong relationship with “SalePrice”, as the total number of bathrooms is in itself indicative of a houses’ size.

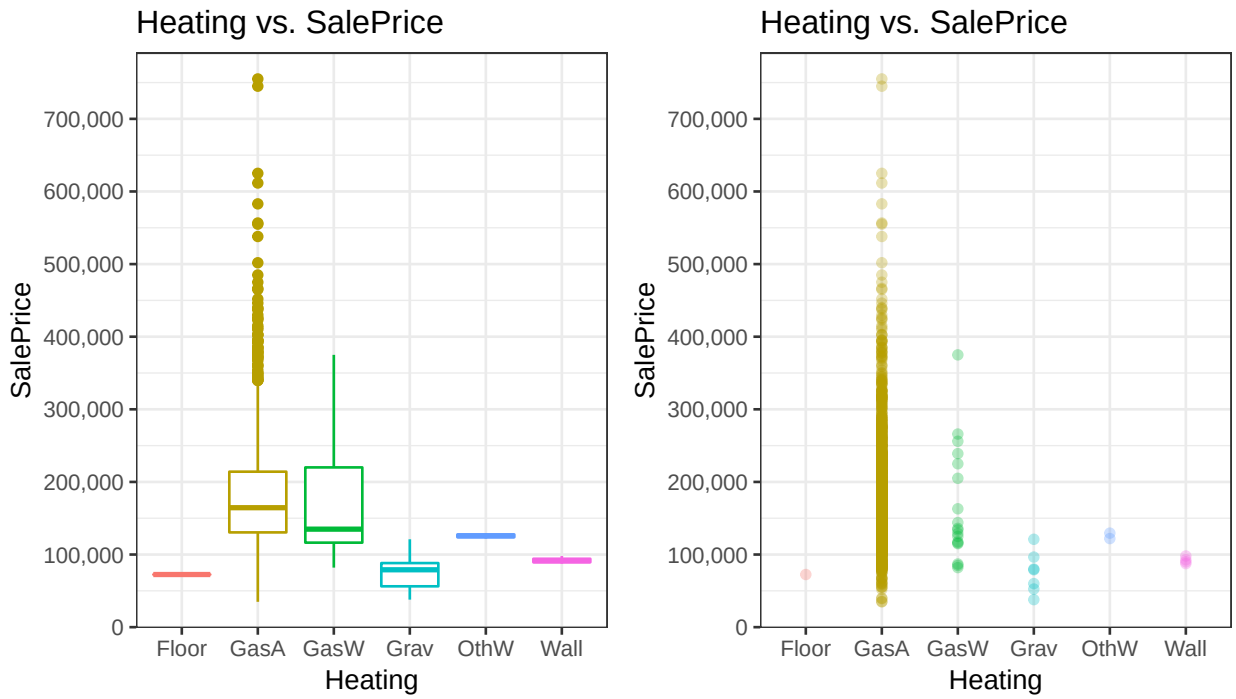


2.1.37 Heating: Type of heating

There are no missing values in Heating, but there is only a single house with "Floor" and 2 houses with "OthW" heating.

```
summary(dataset$Heating)
## Floor  GasA  GasW  Grav  OthW  Wall
##     1  2874    27     9     2     6
```

From the plots we can see that the house with "floor furnace" "Heating" has a low "SalePrice" and that basically any "Heating" other than "GasA" and "GasW" appears to be associated with lower "SalePrice" as well. Most categories are woefully underrepresented, however.

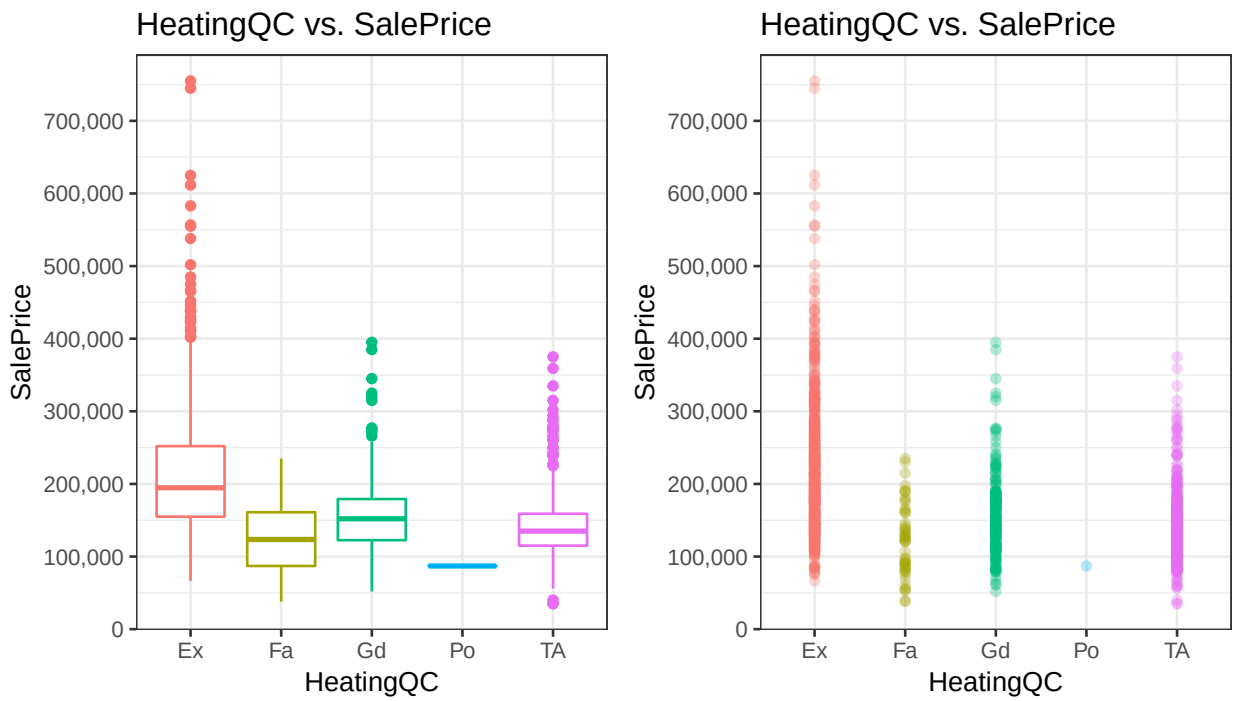


2.1.38 HeatingQC: Heating quality and condition

There are no missing values in "HeatingQC". There are only a few houses with a "poor" "HeatingQC".

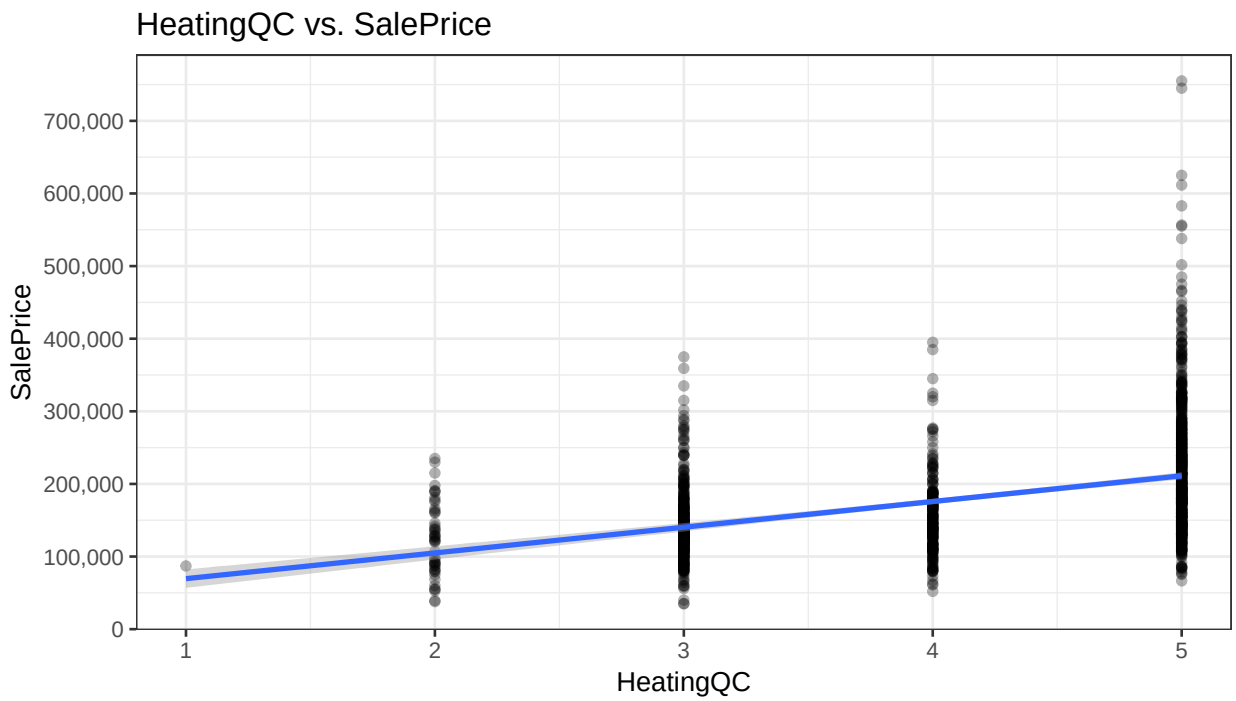
```
summary(dataset$HeatingQC)
##   Ex   Fa   Gd   Po   TA
## 1493   92  474    3  857
```

From the plots we an see that very few houses have poor quality heating and that an excellent quality of heating is associated with a higher "SalePrice". The quality of the heating seems to be more important than its type.



We transform this qualitative ordinal factor into a numeric containing the present levels and plot it again.

```
dataset$HeatingQC <- as.numeric(factor(dataset$HeatingQC,
                                       levels = c("Po", "Fa",
                                        "TA", "Gd", "Ex")))

dataset[train$Id, ] %>%
  ggplot(aes(x = HeatingQC, y = SalePrice)) +
  geom_point(alpha = 0.3) +
  scale_y_continuous(labels = scales::comma, breaks = seq(0, 800000, 100000)) +
  theme_bw() +
  theme(legend.position = "none") +
  ggtitle("HeatingQC vs. SalePrice") +
  geom_smooth(method = "lm")
```

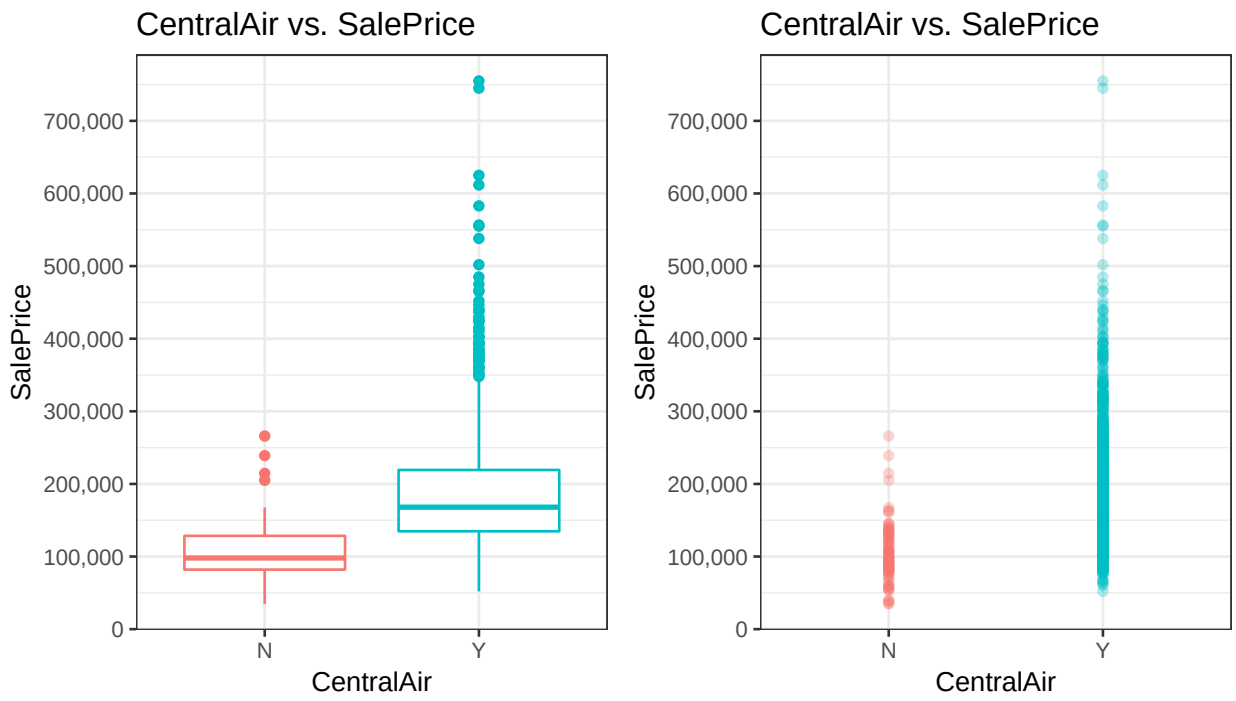


2.1.39 CentralAir: Central air conditioning

There are no missing values in "CentralAir".

```
summary(dataset$CentralAir)
##    N    Y
##  196 2723
```

Clearly, having central air conditioning positively influences sale price.

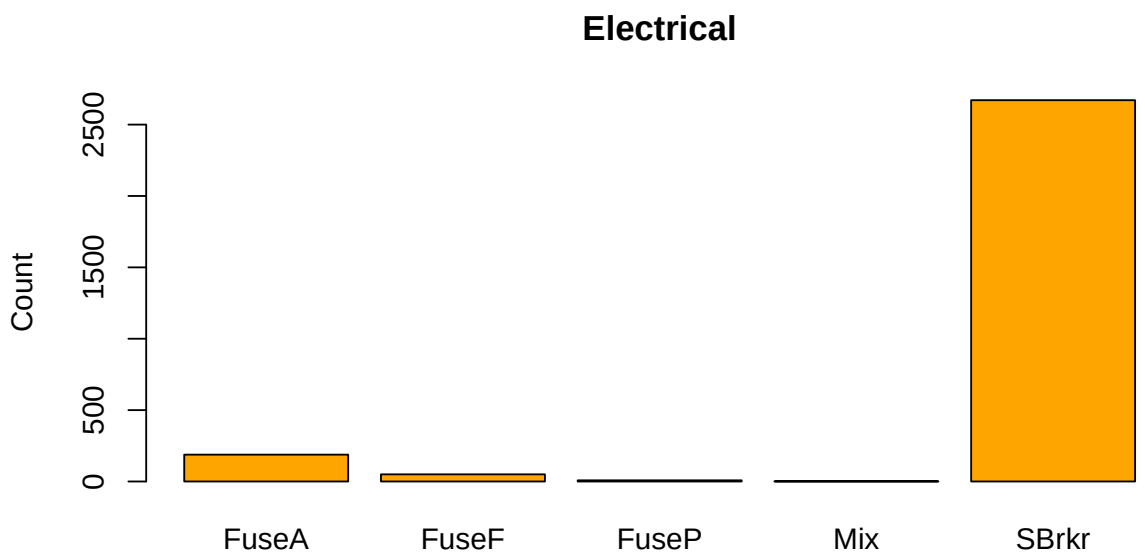


2.1.40 Electrical: Electrical system

There is a missing value in "Electrical".

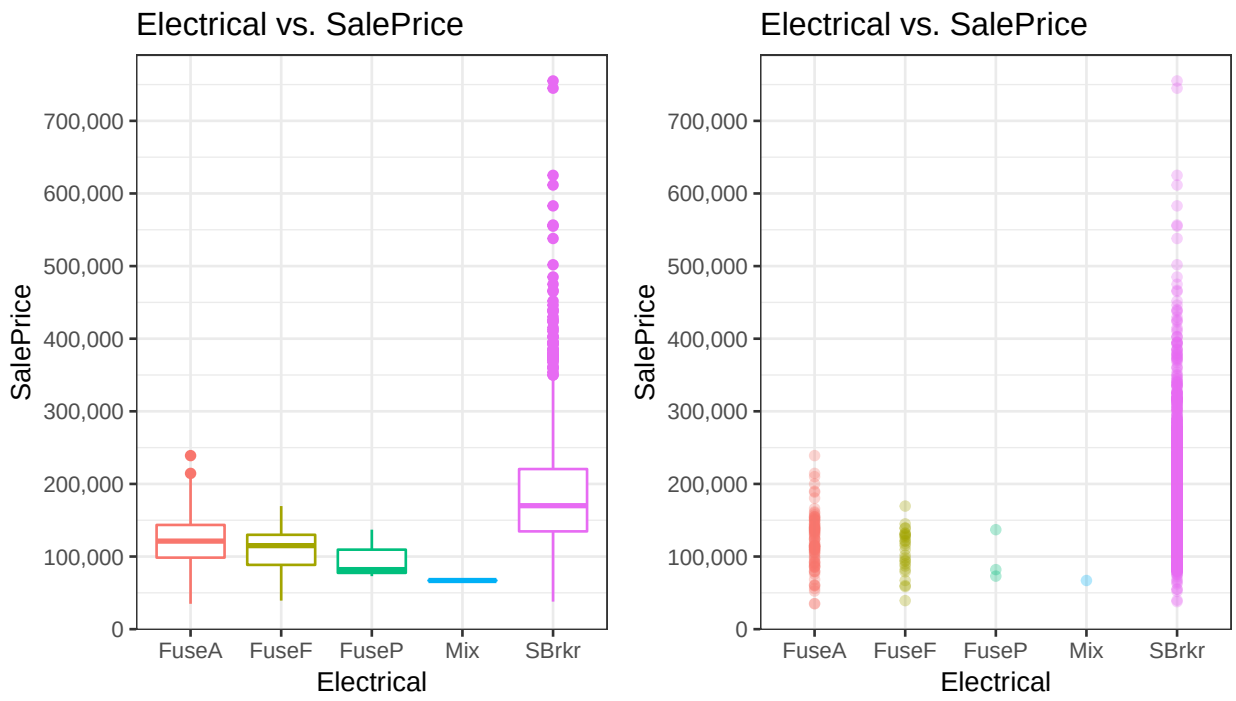
```
summary(dataset$Electrical)
## FuseA FuseF FuseP   Mix SBrkr  NA's
##   188    50     8     1  2671     1
```

```
# There is one missing value in "Electrical".
which(is.na(dataset$Electrical))
## [1] 1380

# The house with the missing "Electrical" value seems pretty normal.
# dataset[1380, ]

# "SBrkr" is the most common value.
plot(dataset[, "Electrical"],
     col = "orange",
     main = "Electrical",
     ylab = "Count"
)
```




```
# We will use kNN-based predictions to impute the missing values.
knn_model <- kNN(dataset, variable = "Electrical", k = 9)

# The kNN-model predicts "SBrkr", the most common type of "Electrical".
knn_model[knn_model$Electrical_imp == TRUE, ]$Electrical
## [1] SBrkr
## Levels: FuseA FuseF FuseP Mix SBrkr

# Missing value imputation of "Electrical".
dataset[which(is.na(dataset$Electrical)), ]$Electrical <-
  knn_model[knn_model$Electrical_imp == TRUE, ]$Electrical
```

From the plots it becomes apparent that “SBrkr” is not only the most common, but also the most valuable kind of “Electrical”.



2.1.41 1stFlrSF: First Floor square feet, 2ndFlrSF: Second floor square feet, LowQualFinSF: Low quality finished square feet (all floors)

There are no missing values in these 3 variables.

```
summary(dataset$X1stFlrSF)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     334     876    1082    1160    1388    5095
summary(dataset$X2ndFlrSF)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0     0.0     0.0   336.5   704.0  2065.0
summary(dataset$LowQualFinSF)
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##    0.000    0.000    0.000    4.694    0.000 1064.000
```

The above ground living area is composed of 1StFlrSF, 2ndFlrSF and LowQualFinSF as evidenced by their combined correlation with "GrLivArea" of exactly 1. We will therefore drop the redundant variables to avoid colinearity.

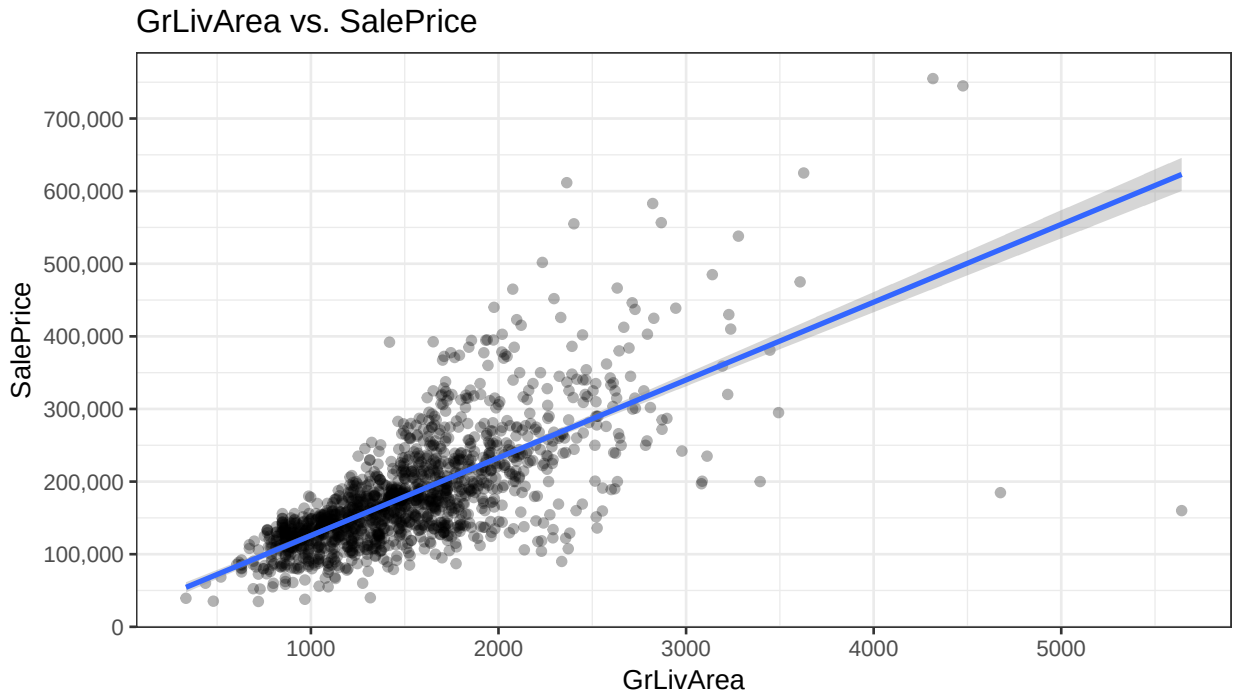
```
# Correlation between "GrLivArea" and the combined individual variables.
cor(dataset$GrLivArea, (dataset$X1stFlrSF + dataset$X2ndFlrSF + dataset$LowQualFinSF))
## [1] 1

# We drop the redundant variables.
dataset <- subset(dataset, select = -c(X1stFlrSF, X2ndFlrSF, LowQualFinSF))
```

2.1.42 GrLivArea: Above grade (ground) living area square feet

There are no missing values in "GrLivArea".

```
summary(dataset$GrLivArea)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     334    1126    1444    1501    1744    5642
```

We plot "GrLivArea" against "SalePrice" and observe its strong, near-linear influence on "SalePrice". Additionally, we change this variables' encoding into numeric. It comes as no big surprise that the above ground living area of a house is extremely impactful on its value.



We also observe two houses with enormous living space, but rather low "SalePrice". These two might be outliers, however upon closer inspection it becomes apparent that both houses have a "SaleType" of "New" and a "SaleCondition" of "Partial". From the data description we know that a "Partial" "SaleCondition" implies that a house was not completed when last assessed. It is therefore possible that these houses' actual value has not yet been updated to reflect their actual worth. Other parameters like "Excellent" "OverallQuality" and "Good" "OverallCond" would also suggest a more expensive "SalePrice" for these houses. While the data description does not explicitly state this, it is conceivable that these houses were actually sold "partially", as in that both houses are "2Story" "1Fam" kind of buildings and that maybe only one of the possibly several appartments within these houses was sold at that price. Whatever might be the case, we will keep these houses in the dataset, as they don't influence the relationship between "GrLivArea" and "SalePrice" too much.

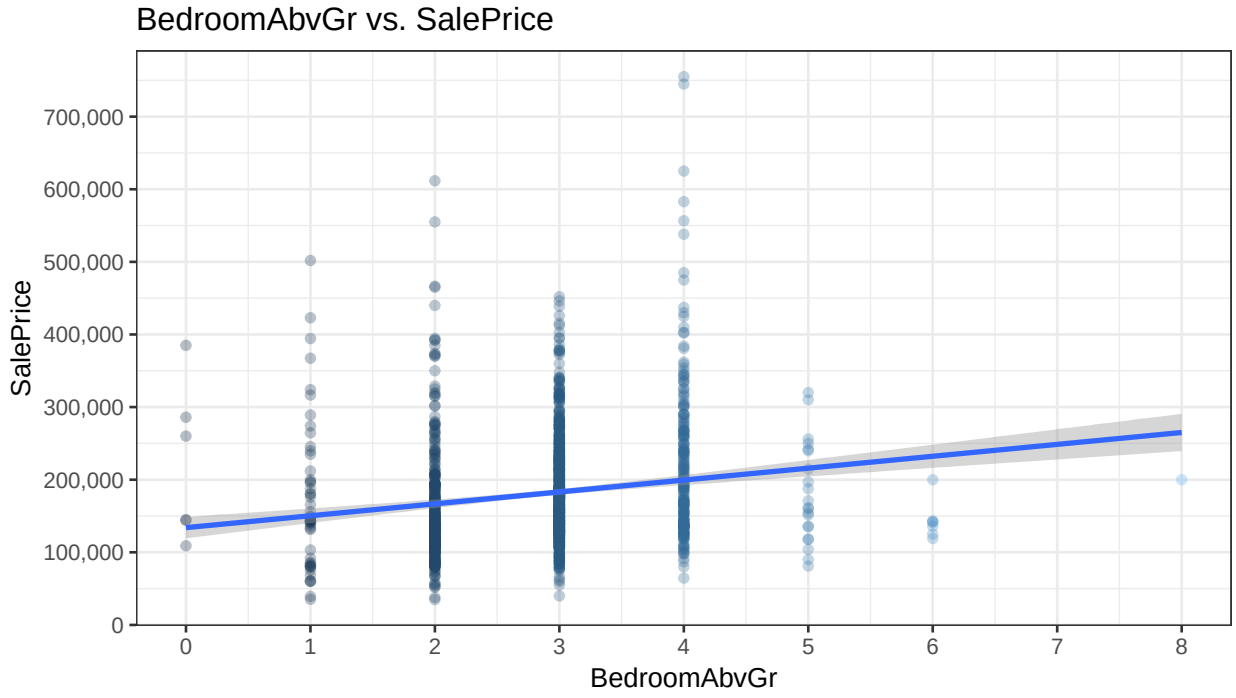
```
# Which are the two "outlier" houses?
which(dataset$GrLivArea > 4500 & dataset$SalePrice < 250000 & dataset$Id %in% train$Id)
## [1]  524 1299

# We take a look at these houses and take note of their "SaleCondition" and "SaleType".
# Both are new homes with partial "SaleType".
# dataset[524, ]
# dataset[1299, ]
```

2.1.43 Bedroom: Bedrooms above grade (does NOT include basement bedrooms)

There are no missing values in "BedroomAbvGr"

```
summary(dataset$BedroomAbvGr)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    2.00    3.00    2.86    3.00    8.00
```

We plot "BedroomAbvGr" against "SalePrice" and notice that While a larger number of bedrooms probably denotes larger and therefore more expensive houses, the absolute number of bedrooms is not necessarily the strongest predictor of "SalePrice".

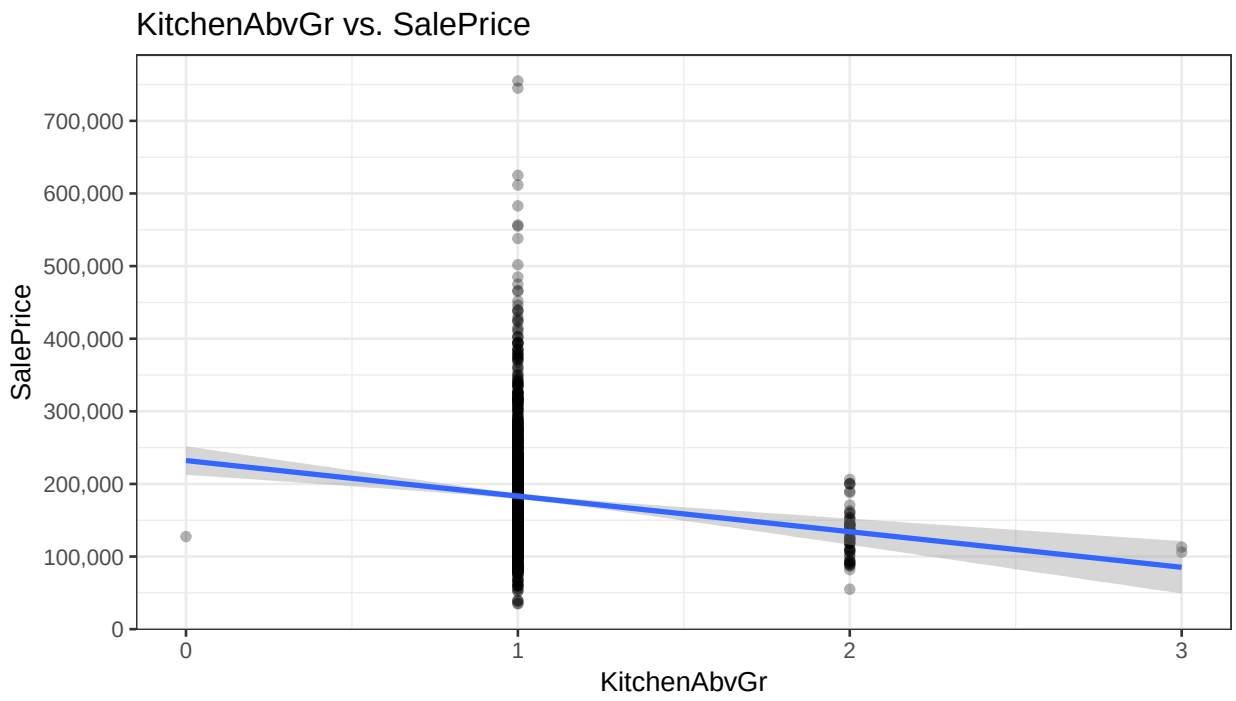


2.1.44 Kitchen: Kitchens above grade

There are no missing values in "KitchenAbvGr".

```
summary(dataset$KitchenAbvGr)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   1.000   1.000   1.045   1.000   3.000
```

From the plot it becomes apparent that the number of kitchens above ground are not a good predictor of "SalePrice". Most houses actually only have one kitchen above ground, while having two or more does not indicate increased value.



A few houses actually have no kitchen above ground.

```
# Some houses don't have a kitchen above ground.
which(dataset$KitchenAbvGr == 0)
## [1]  955 2588 2860
```

2.1.45 KitchenQual: Kitchen quality

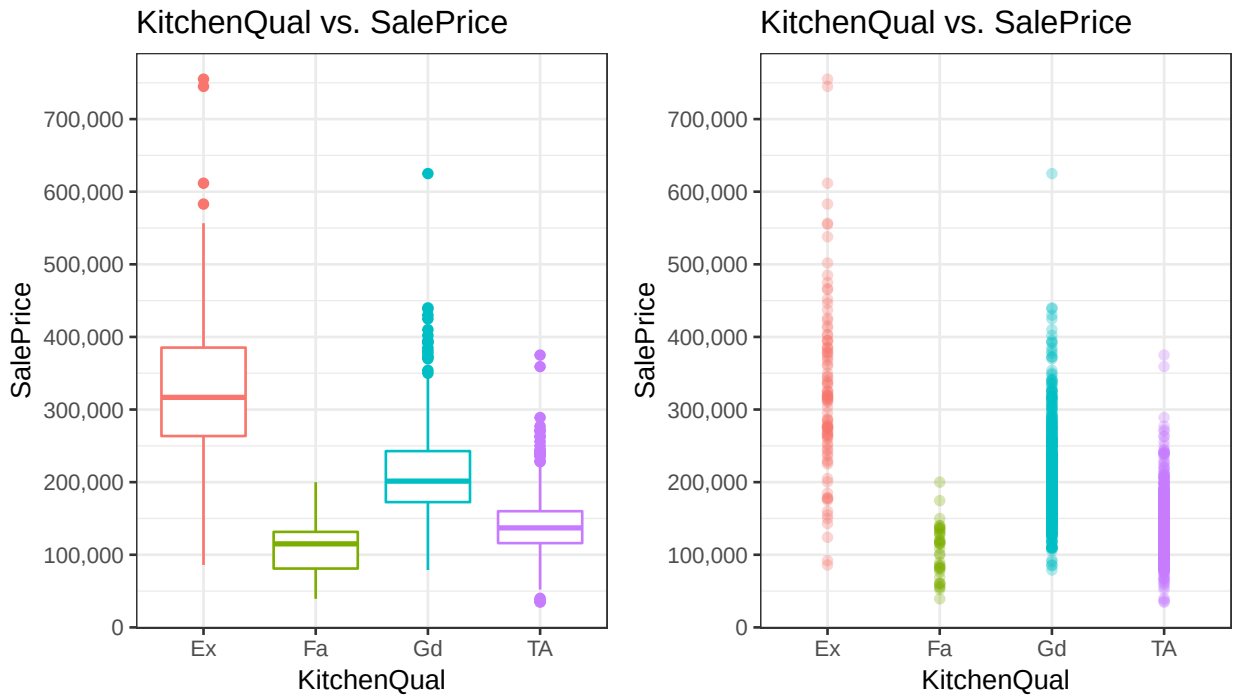
There is one missing value in "KitchenQual".

```
summary(dataset$KitchenQual)
##   Ex   Fa   Gd   TA NA's
##  205   70 1151 1492    1
```

We will use kNN-based predictions to impute the missing value.

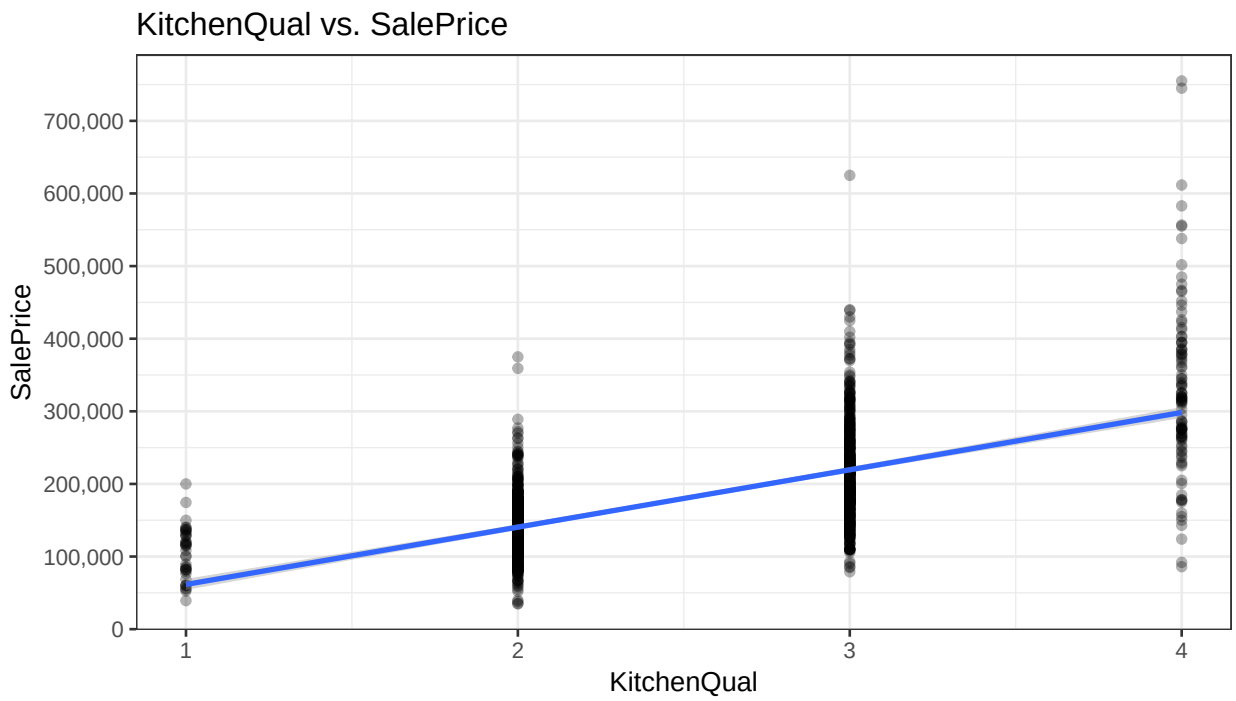
```
# kNN-model.
knn_model <- kNN(dataset, variable = "KitchenQual", k = 9)

# The kNN-model predicts TA, the most common type of "KitchenQual".
knn_model[knn_model$KitchenQual_imp == TRUE, ]$KitchenQual
## [1] TA
## Levels: Ex Fa Gd TA

# Missing value imputation of "KitchenQual".
dataset[which(is.na(dataset$KitchenQual)), ]$KitchenQual <-
  knn_model[knn_model$KitchenQual_imp == TRUE, ]$KitchenQual
```

From the plots we can see that "KitchenQual" strongly correlates with "SalePrice". The quality of a kitchen is much more important than the total number.



We transform this qualitative ordinal factor into a numeric containing the present levels and plot it again.

```
dataset$KitchenQual <- as.numeric(factor(dataset$KitchenQual, levels = c("Fa",
                                                    "TA", "Gd", "Ex")))
# Plotting changed KitchenQual.
dataset[train$Id, ] %>%
  ggplot(aes(x = KitchenQual, y = SalePrice)) +
  geom_point(alpha = 0.3) +
  scale_y_continuous(labels = scales::comma, breaks = seq(0, 800000, 100000)) +
  theme_bw() +
  theme(legend.position = "none") +
  ggtitle("KitchenQual vs. SalePrice") +
  geom_smooth(method = "lm")
```

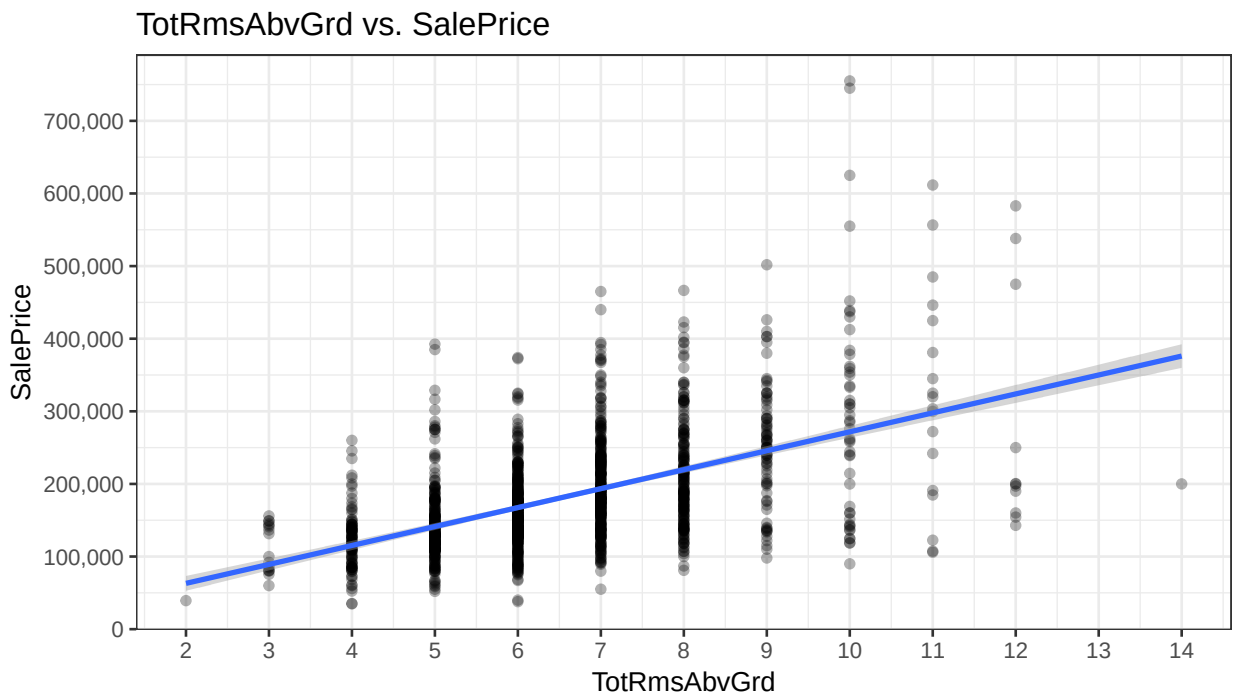


2.1.46 TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)

There are no missing values in "TotRmsAbvGrd".

```
summary(dataset$TotRmsAbvGrd)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   5.000   6.000   6.452   7.000  15.000
```

From the plot it becomes clear that "TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)" is clearly an indicator of a houses size. The more rooms the higher the "SalePrice" in most cases.



2.1.47 Functional: Home functionality (Assume typical unless deductions are warranted)

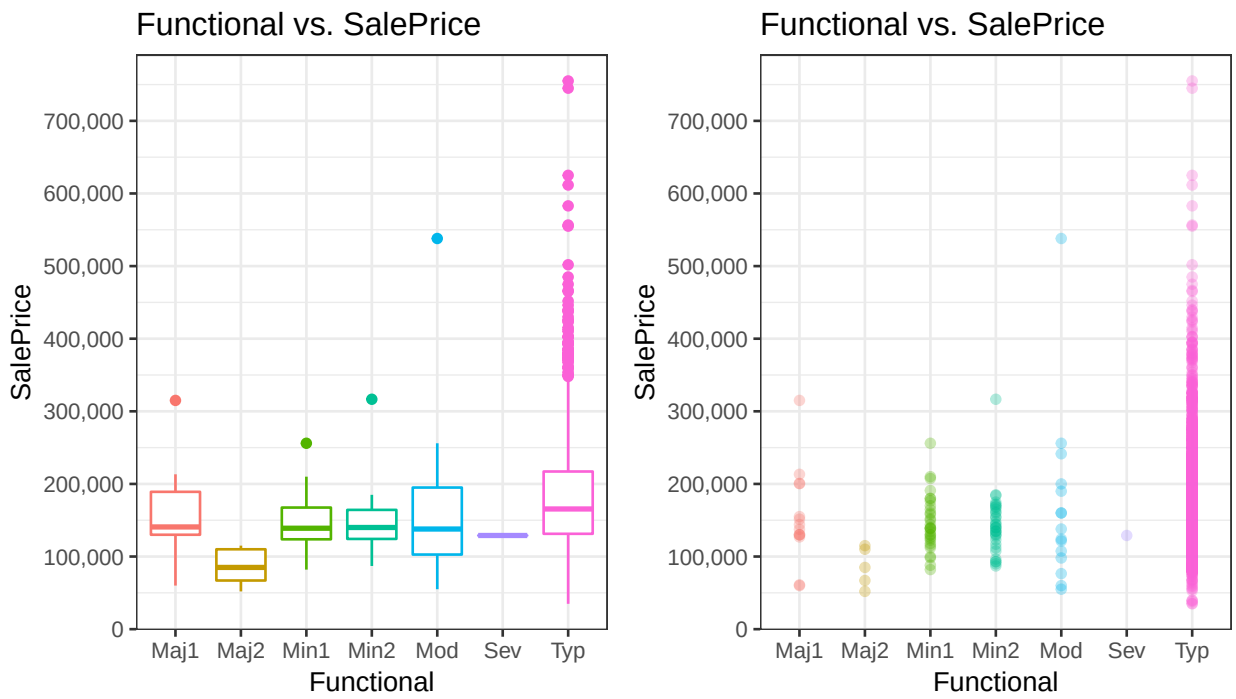
There are a few missing values in "Functional". There is no house with a "Sal" (Salvage only) entry.

```
summary(dataset$Functional)
## Maj1 Maj2 Min1 Min2  Mod  Sev  Typ NA's
##   19    9   65   70   35    2 2717    2
```

We will use kNN-based missing value imputation for the missing values in "Functional".

```
# We will use kNN-based predictions to impute the missing values.
knn_model <- kNN(dataset, variable = "Functional", k = 9)

# The kNN-model predicts "Typ", the most common type of "Functional".
knn_model[knn_model$Functional_imp == TRUE, ]$Functional
## [1] Typ Typ
## Levels: Maj1 Maj2 Min1 Min2 Mod Sev Typ

# Missing value imputation of "Functional".
dataset[which(is.na(dataset$Functional)), ]$Functional <-
  knn_model[knn_model$Functional_imp == TRUE, ]$Functional
```

We plot "Functional" against "SalePrice" and observe that "Maj2" (major deductions 2) seems to influence "SalePrice" negatively. A few houses are "Sev" (severly damaged), but median "SalePrice" remains comparable in these cases. Most houses, also most expensive houses,have "Typ" (typical functionality) values.

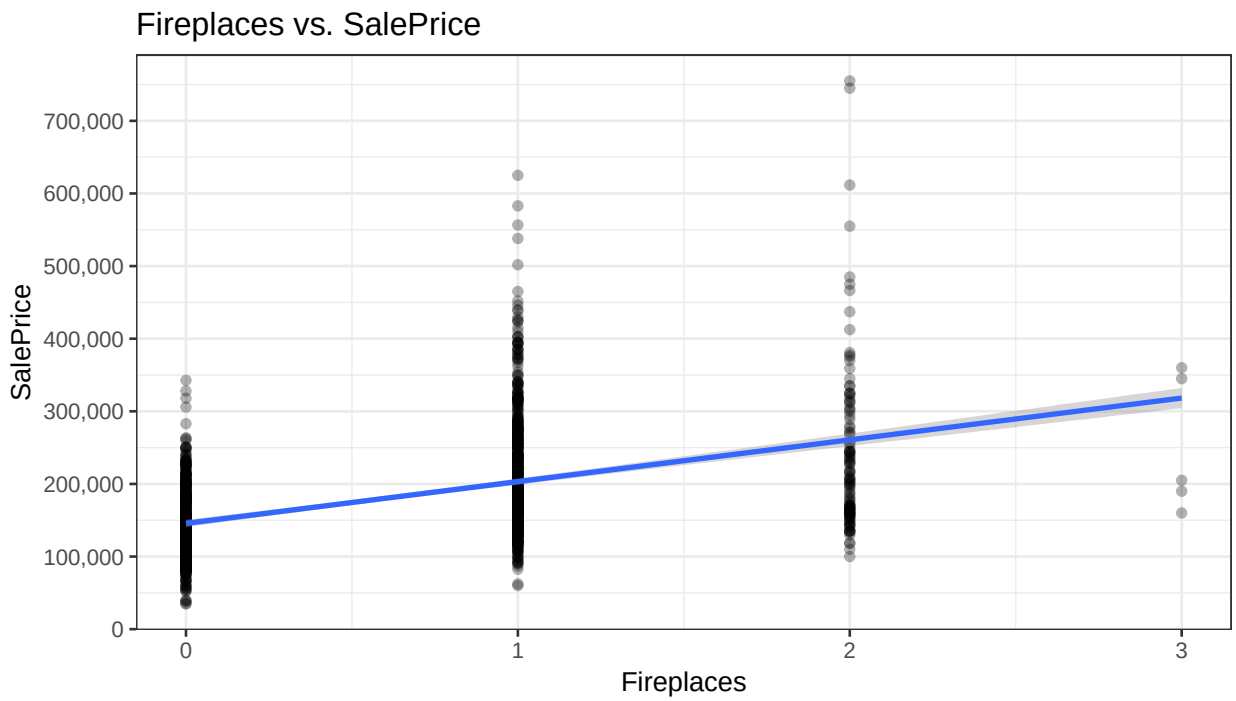


2.1.48 Fireplaces: Number of fireplaces

There are no missing values in "Fireplaces". We change variable encoding to numeric.

```
summary(dataset$Fireplaces)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  1.0000  0.5971  1.0000  4.0000

# Change variable encoding to numeric.
dataset$Fireplaces <- as.numeric(dataset$Fireplaces)
```

"Fireplaces" can be considered another measurement of house size. Having more "Fireplaces" is possitively correlated with "SalePrice". It seems likely that having 1 or more "Fireplaces" of high quality would be strong predictors of "SalePrce. To confirm this, we have to inspect"FireplacesQu" as well.

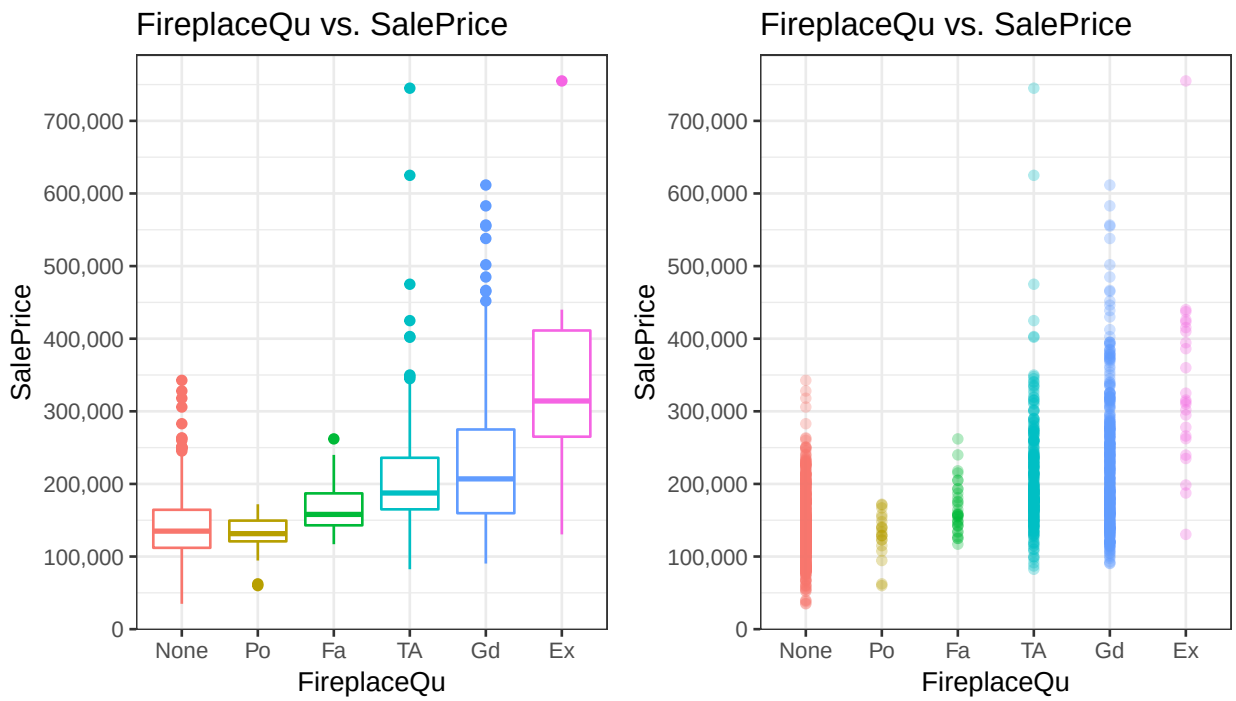


2.1.49 FireplaceQu: Fireplace quality

A large amount of values are missing in "FireplaceQu". From the data description we know that "no fireplace" was originally encoded as "NA" in the data. We can therefore replace all "NAs" with "None" and fix factor levels.

```
summary(dataset$FireplaceQu)
##   Ex   Fa   Gd   Po   TA NA's
##   43   74  744   46  592 1420

# Fixing missing values and factor levels.
dataset$FireplaceQu <- as.character(dataset$FireplaceQu)
dataset$FireplaceQu[which(is.na(dataset$FireplaceQu))] <- "None"
dataset$FireplaceQu <- factor(dataset$FireplaceQu, levels = c("None",
                                        "Po", "Fa", "TA", "Gd", "Ex"))
```

From the plots we can see that higher quality "Fireplaces" are well correlated with higher "SalePrice. However, there are also many houses without any"Fireplaces" that have larger "SalePrice" than some of the houses with "poor", "fair", or "typical" quality "Fireplaces".

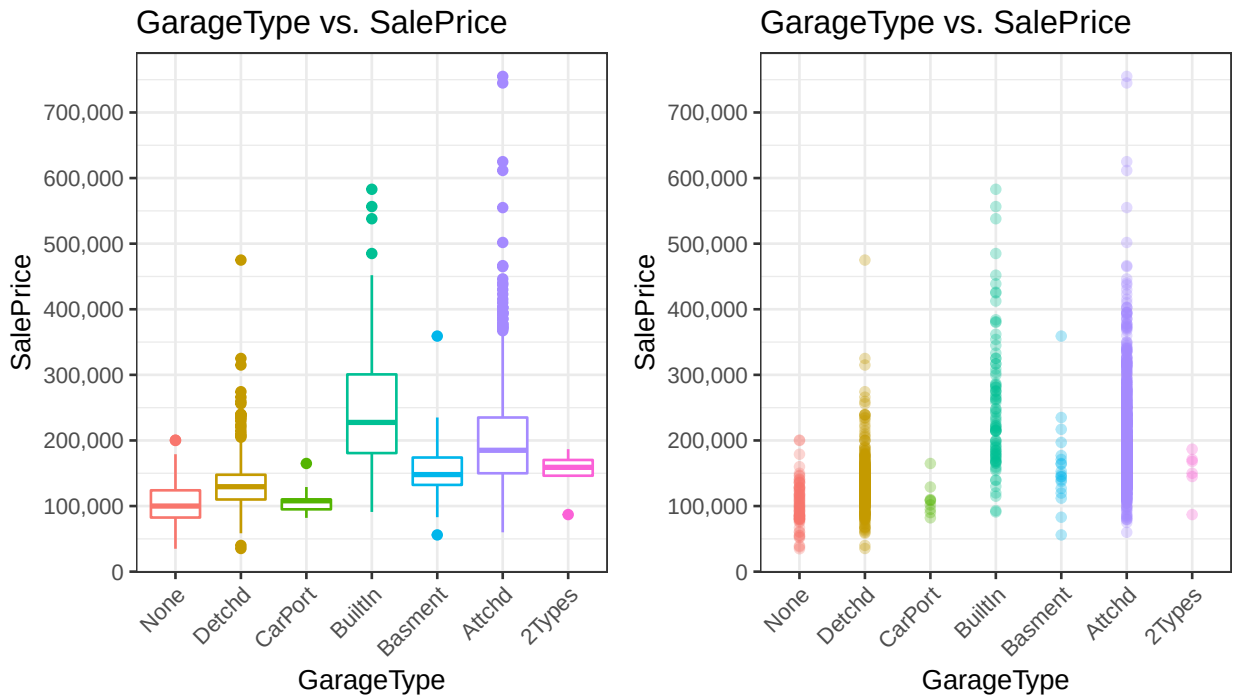


2.1.50 GarageType: Garage location

There are a lot of missing values in "GarageType". From the data description we know that "No Garage" was originally encoded as "NA". We will replace those with "None" and fix factor levels.

```
summary(dataset$GarageType)
##  2Types  Attchd Basment BuiltIn CarPort  Detchd    NA's
##      23    1723      36     186      15     779     157

# Fixing missing values and factor levels.
dataset$GarageType <- as.character(dataset$GarageType)
dataset$GarageType[which(is.na(dataset$GarageType))] <- "None"
dataset$GarageType <- factor(dataset$GarageType, levels = c("None",
        "Detchd", "CarPort", "BuiltIn", "Basment", "Attchd", "2Types"))
```

From the plots we can see that having no garage or just a carport can be predictive of a lower "SalePrice". There is a certain difference between detached and attached garages, too.



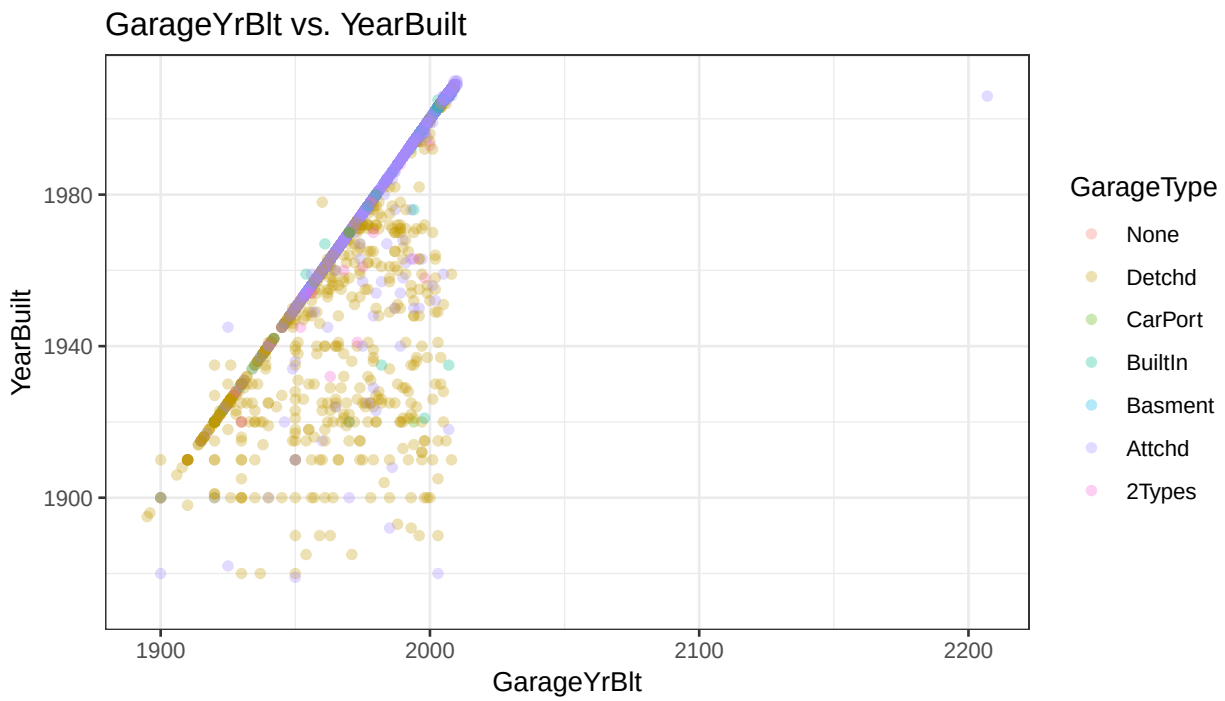
2.1.51 GarageYrBlt: Year garage was built

There are a lot of missing values in "GarageYrBlt", presumably from houses with no garages.

```
summary(dataset$GarageYrBlt)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    1895    1960    1979    1978    2002    2207     159
```

We plot "GarageYrBlt" vs. "YearBuilt" and color by "GarageType".

```
dataset %>%
  ggplot(aes(x = GarageYrBlt, y = YearBuilt, color = GarageType)) +
  geom_point(alpha = 0.3) +
  theme_bw() +
  ggtitle("GarageYrBlt vs. YearBuilt")
```



From the plot we can see that it is mostly detached garages being built after the house itself, otherwise "YearBuilt" and "GarageYrBlt" are mostly identical. Also, there seems to be an outlier, with a garage built sometime in the far future (in 2207 to be exact).

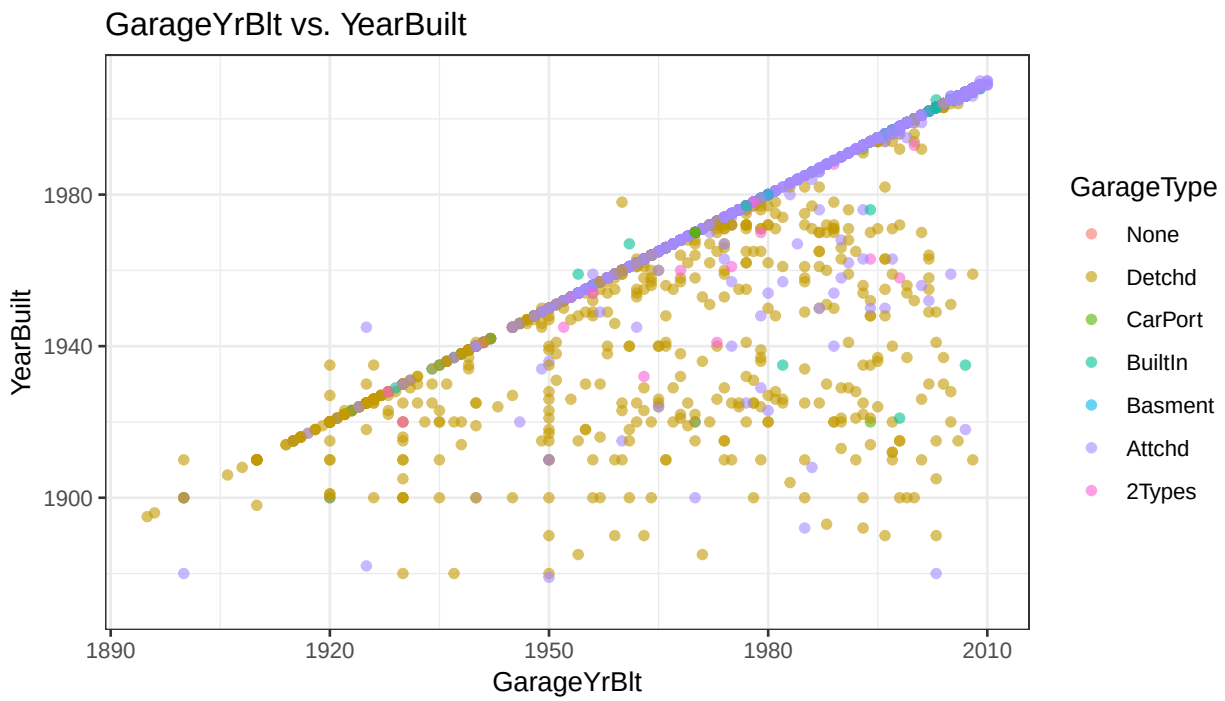
```
# A garage built in the future?
# dataset[which(dataset$GarageYrBlt > 2010), ]
dataset$GarageYrBlt[which(dataset$GarageYrBlt > 2010)]
## [1] 2207
```

From this it seems most likely that the garage was built at the same time as the house and the outlier is based on a typo. We will fix this by changing the year 2207 to 2007 (based on the houses' "YearBuilt").

```
dataset$GarageYrBlt[which(dataset$GarageYrBlt > 2010)] <- 2007
```

We plot "GarageYrBlt" against "YearBuilt" without the outlier.

```
# We plot GarageYrBlt vs. YearBuilt and color by GarageType without the outlier.
dataset %>%
  ggplot(aes(x = GarageYrBlt, y = YearBuilt, color = GarageType)) +
  geom_point(alpha = 0.6) +
  theme_bw() +
  ggtitle("GarageYrBlt vs. YearBuilt")
```



Interestingly, there are some houses with garage building years prior to the house itself. Perhaps building the house took much longer than the garage. Overall, "GarageYrBlt" does not add much in terms of "SalePrice" predictive power. As we can't turn this variable into a factor with a level "None" for not having a garage, we have to either impute the houses' "YearBuilt" or remove "GarageYrBlt" entirely. We will do the latter, as there are more than enough variables about Garages in the dataset and imputing "YearBuilt" might penalize houses that actually acquired their garage more recently.

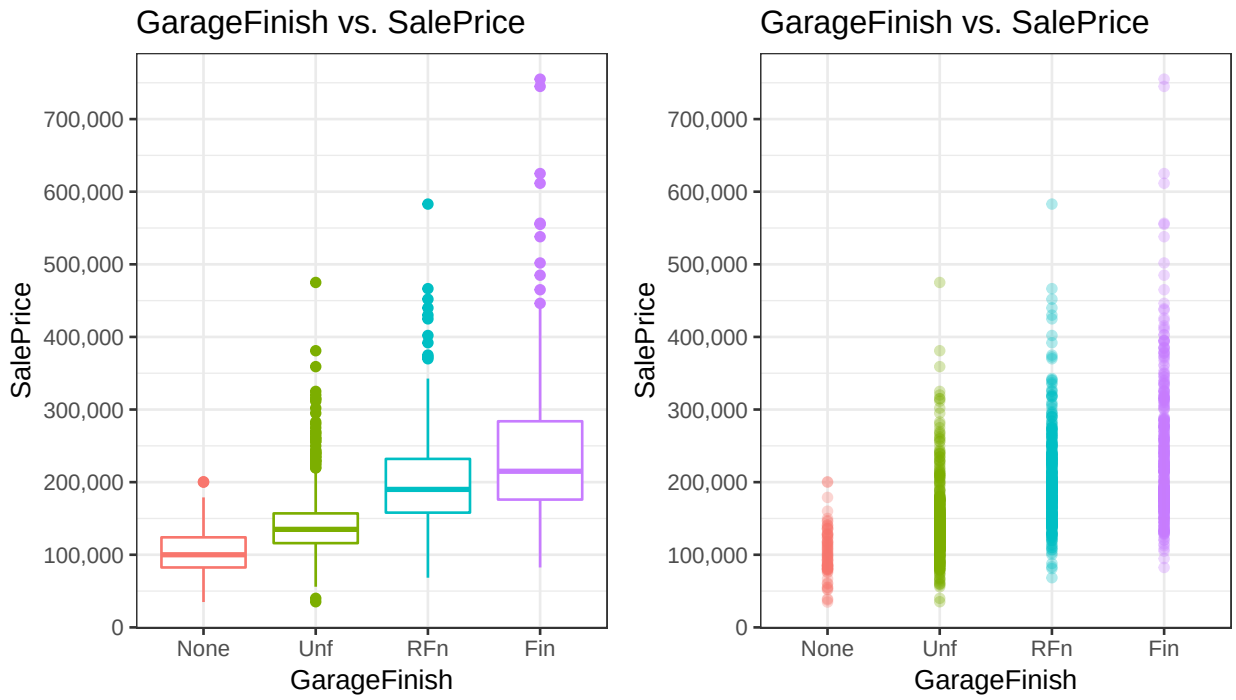
```
# Removal of "GarageYrBlt".
dataset <- subset(dataset, select = - GarageYrBlt)
```

2.1.52 GarageFinish: Interior finish of the garage

There are many missing values in "GarageFinish", presumably from not having a garage in the first place, as seen before. From the data description we know that "No Garage" was originally encoded as "NA". We will replace those with "None" and fix factor levels.

```
summary(dataset$GarageFinish)
##  Fin  RFn  Unf NA's
##  719  811 1230  159

# Fixing missing values and factor levels.
dataset$GarageFinish <- as.character(dataset$GarageFinish)
dataset$GarageFinish[which(is.na(dataset$GarageFinish))] <- "None"
dataset$GarageFinish <- factor(dataset$GarageFinish, levels = c("None",
                                                "Unf", "RFn", "Fin"))
```

We plot "GarageFinish" against "SalePrice" and observe that no or unfinished garages are predictive of lower "SalePrice".



2.1.53 GarageCars: Size of garage in car capacity, GarageArea: Size of garage in square feet

"GarageCars" and "GarageArea" are two variables dealing with garage size. Both contain a missing value.

```
# GarageArea.
summary(dataset$GarageArea)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##     0.0   320.0   480.0   472.9   576.0  1488.0       1

#GarageCars.
summary(dataset$GarageCars)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   0.000   1.000   2.000   1.767   2.000   5.000       1
```

One house is missing a value in both variables. While the house has a "GarageType" of "Detchd", the "NAs" in "GarageQual" and "GarageCond" hint at the house actually not having any garage.

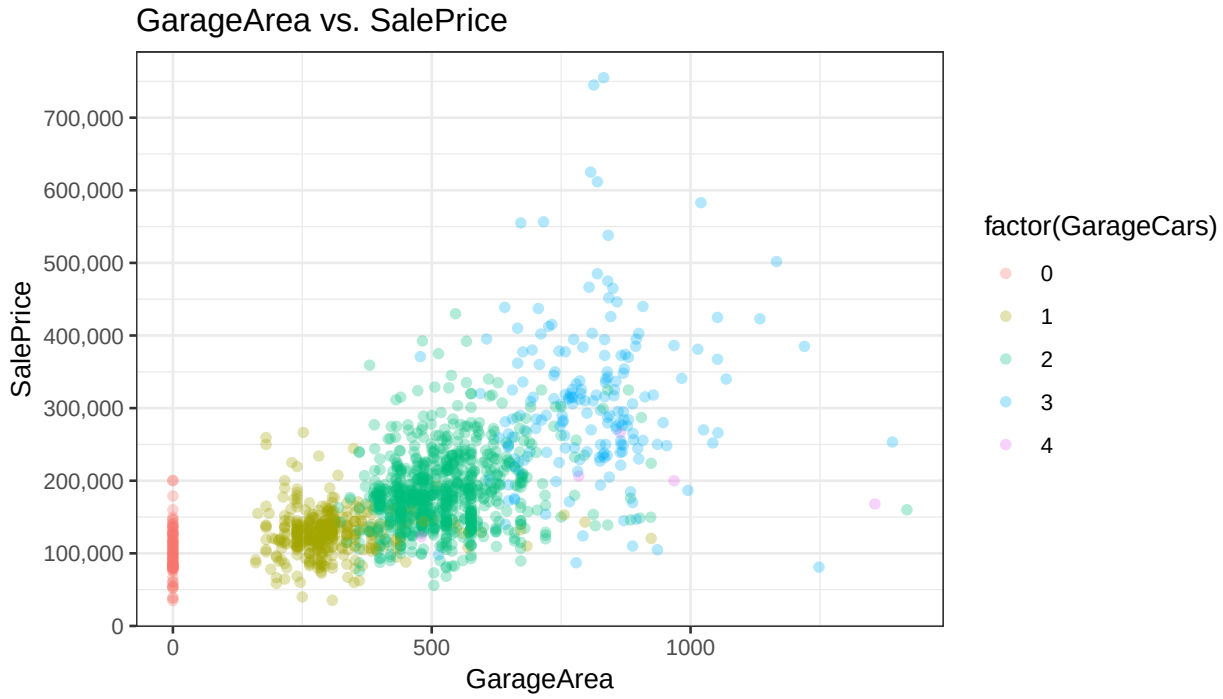
```
# The house with missing values in these variables actually has no garage.
# dataset[which(is.na(dataset$GarageArea)), ]
# dataset[which(is.na(dataset$GarageCars)), ]
```

It seems that this house wrongly has an entry in "GarageType". We will change this to "None" and change the size variables zo zeroes as well.

```
# Chaning GarageType to "None" as this house has no garage.
dataset$GarageType[which(is.na(dataset$GarageArea))] <- "None"

# This house has no garage and therefore no size values.
```

```
dataset$GarageArea[which(is.na(dataset$GarageArea))] <- 0
dataset$GarageCars[which(is.na(dataset$GarageCars))] <- 0
```

We plot "GarageArea" vs. "SalePrice" and color by factorized "GarageCars". From the plot we can clearly see that garage size in terms of area strongly correlates with the number of cars that it is designed to fit.



"GarageArea" and "GarageCars" are highly correlated.

```
# Correlation between "GarageArea" and "GarageCars"."
cor(dataset$GarageArea, dataset$GarageCars)
## [1] 0.8898902
```

GarageCars is slightly stronger correlated with "SalePrice".

```
# Correlation between "GarageArea" and "SalePrice".
cor(dataset[train$Id, ]$SalePrice, dataset[train$Id, ]$GarageArea)
## [1] 0.6234314

# Correlation between "GarageCars" and "SalePrice".
cor(dataset[train$Id, ]$SalePrice, dataset[train$Id, ]$GarageCars)
## [1] 0.6404092
```

In order to avoid having two extremely colinear variables we will drop "GarageArea", the one with slightly lower correlation with "SalePrice".

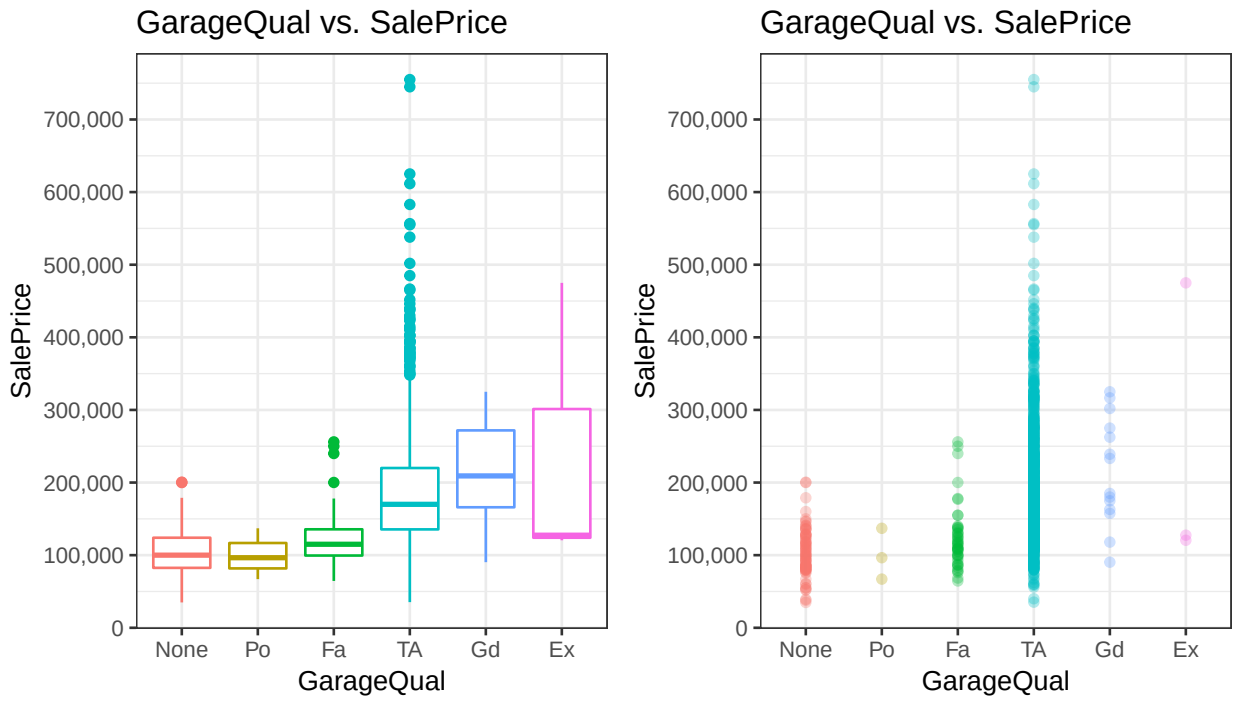
```
dataset <- subset(dataset, select = -GarageArea)
```

2.1.54 GarageQual: Garage quality

There are many missing values in "GarageQual", presumably from these houses having no garage. From the data description we know that "No Garage" was originally encoded as "NA". We will replace those with "None" and fix factor levels.

```
summary(dataset$GarageQual)
##   Ex   Fa   Gd   Po   TA NA's
##    3  124   24    5 2604  159

# Fixing missing values and factor levels.
dataset$GarageQual <- as.character(dataset$GarageQual)
dataset$GarageQual[which(is.na(dataset$GarageQual))] <- "None"
dataset$GarageQual <- factor(dataset$GarageQual, levels = c("None",
                                    "Po", "Fa", "TA", "Gd", "Ex"))
```

We plot "GarageQual" against "SalePrice" and observe that higher "GarageQual" is associated with higher "SalePrice". However, only few houses are in the "Excellent" category, causing a very large range.



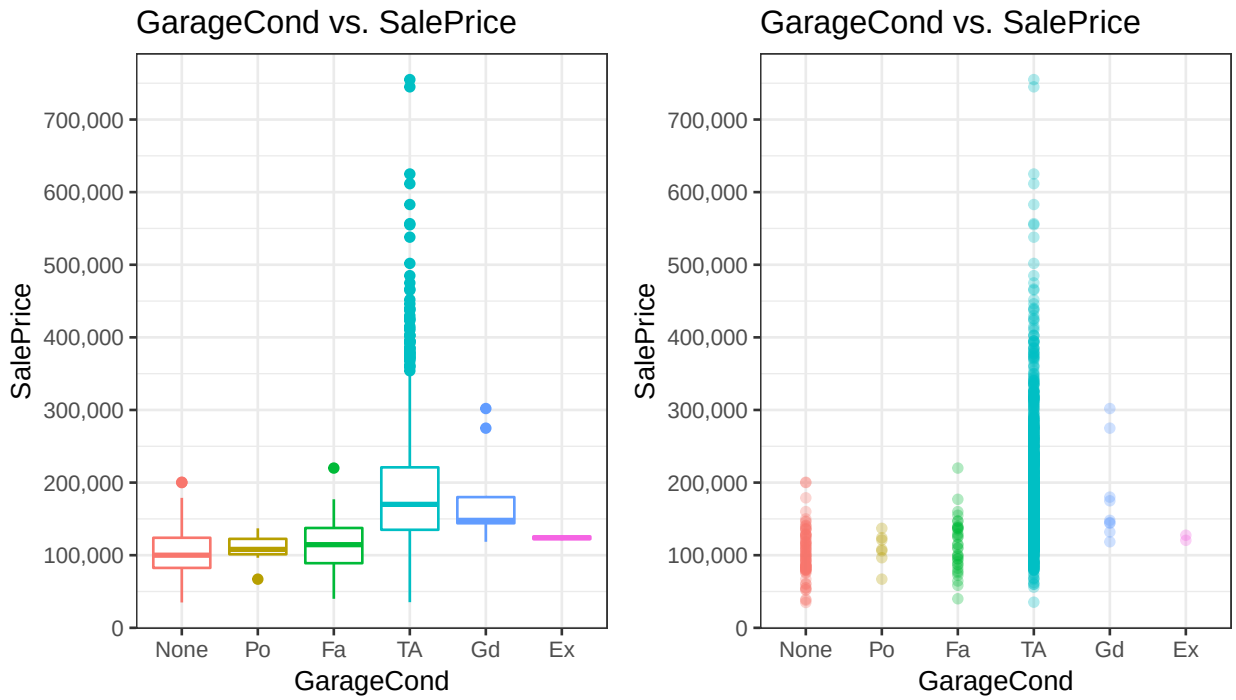
2.1.55 GarageCond: Garage Condition

There are many missing values in "GarageCond", presumably from these houses having no garage. From the data description we know that "No Garage" was originally encoded as "NA". We will replace those with "None" and fix factor levels.

```
summary(dataset$GarageCond)
##   Ex   Fa   Gd   Po   TA NA's
##    3   74   15   14 2654  159
```

```
# Fixing missing values and factor levels.
dataset$GarageCond <- as.character(dataset$GarageCond)
dataset$GarageCond[which(is.na(dataset$GarageCond))] <- "None"
dataset$GarageCond <- factor(dataset$GarageCond, levels = c("None",
                                    "Po", "Fa", "TA", "Gd", "Ex"))
```

From the plots we can see that "GarageCond" is very similar to "GarageQual. In both cases, a"typical" value "TA" seems to be the mode and actually associated with highest "SalePrice".

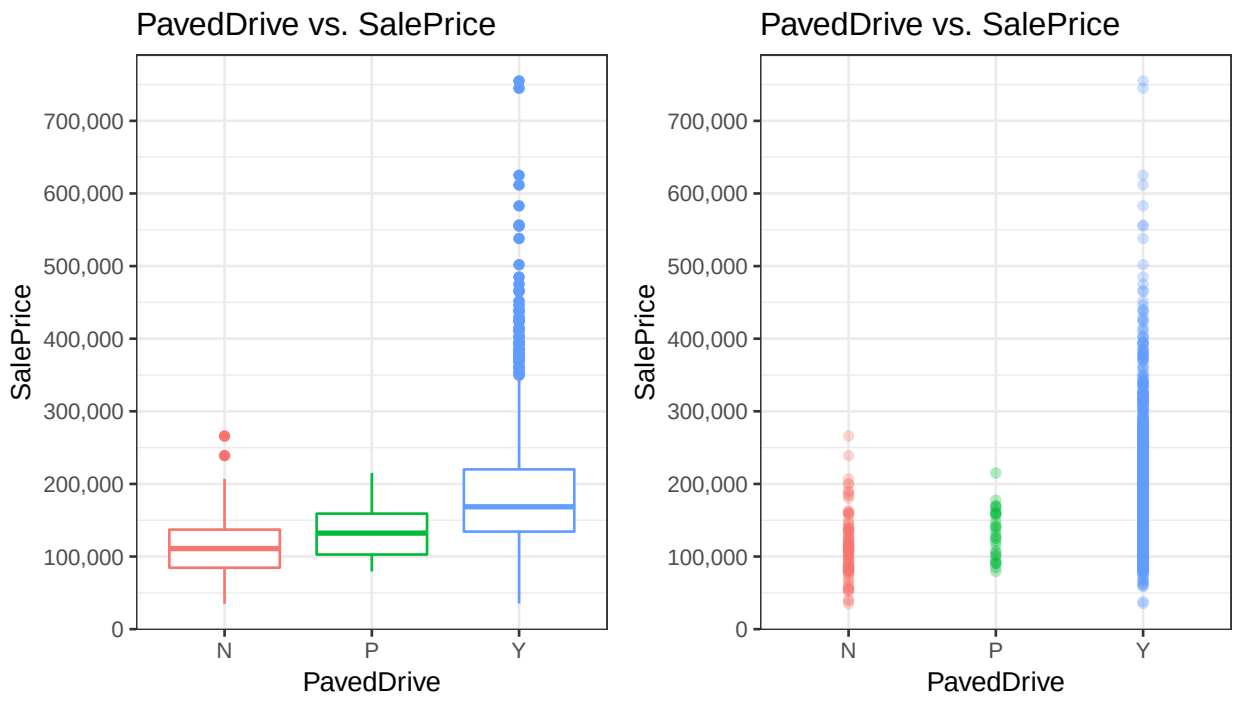


2.1.56 PavedDrive: Paved driveway

There are no missing values in "PavedDrive".

```
summary(dataset$PavedDrive)
##    N    P    Y
##  216   62 2641
```

From the plots we can see that a paved driveway is predictive of a higher "SalePrice".



2.1.57 Variables related to porches

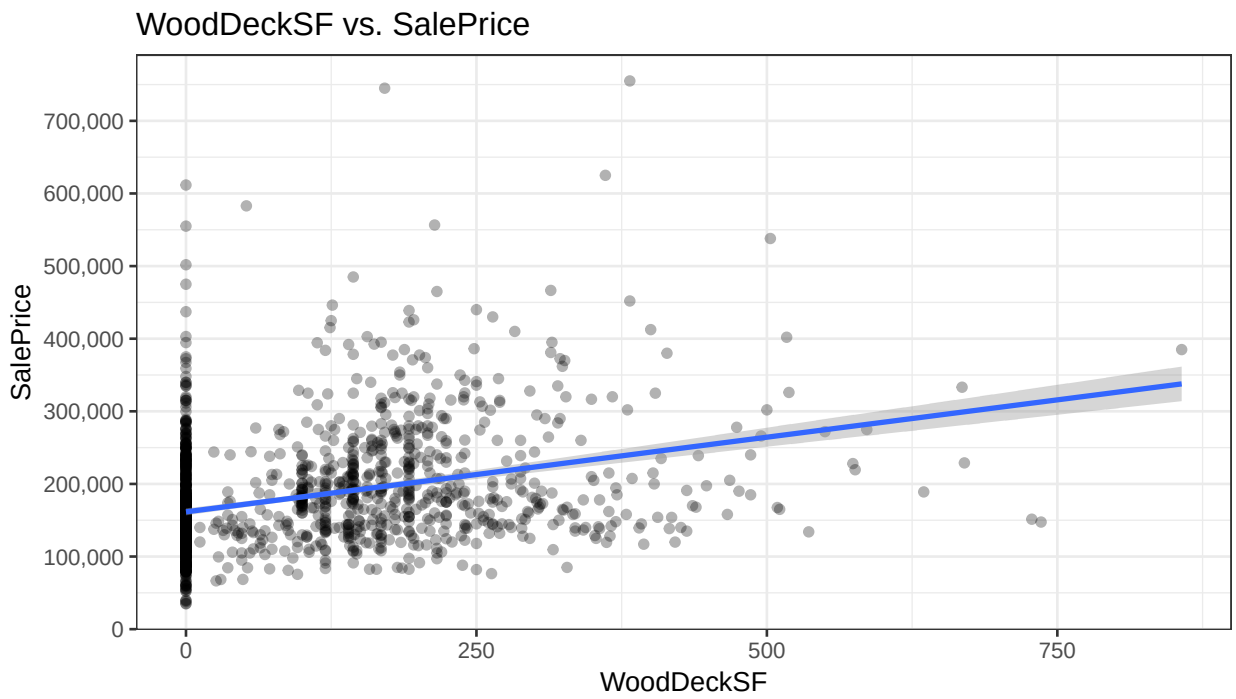
We will first look at all the porch-related variables in isolation.

WoodDeckSF: Wood deck area in square feet

There are no missing values in "WoodDeckSF". We change variable encoding to numerical.

```
summary(dataset$WoodDeckSF)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    0.00    0.00   93.71  168.00 1424.00

# We change encoding to numerical.
dataset$WoodDeckSF <- as.numeric(dataset$WoodDeckSF)
```

From the plot we an see that "WoodDeckSF" positively correelates with "SalePrice".

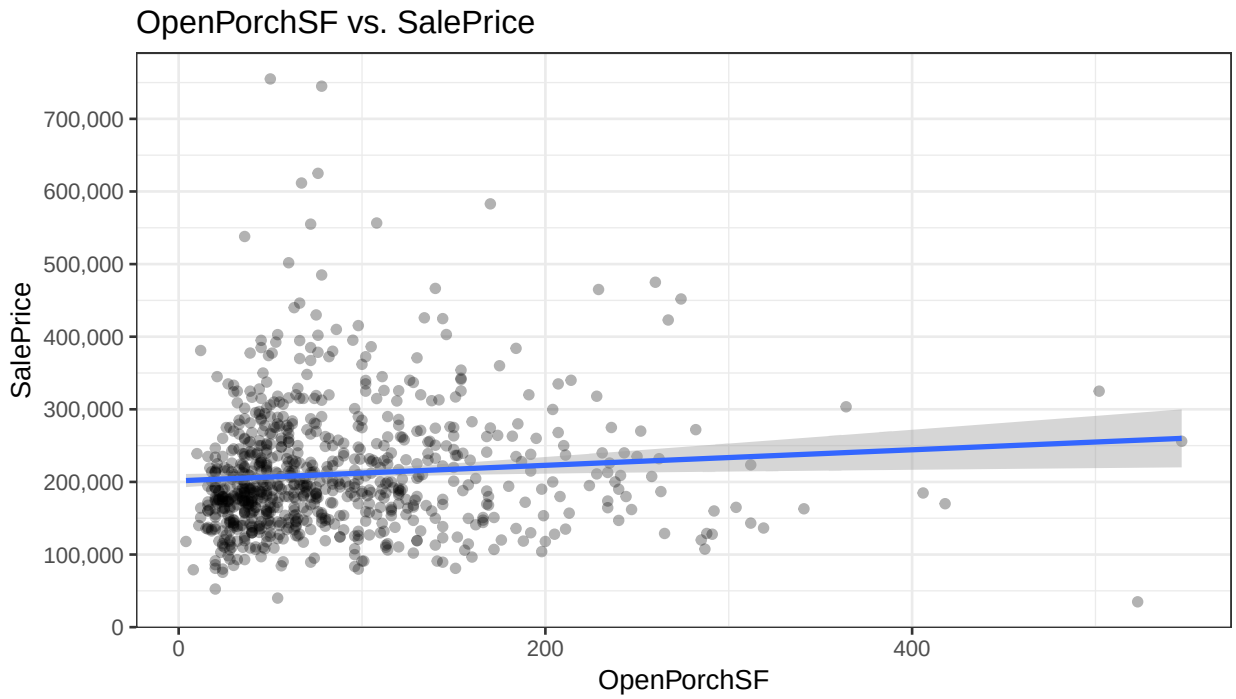


OpenPorchSF: Open porch area in square feet

There are no missing values in "OpenPorchSF". We change encoding to numerical

```
summary(dataset$OpenPorchSF)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    0.00   26.00   47.49   70.00  742.00

# We change encoding to numerical.
dataset$OpenPorchSF <- as.numeric(dataset$OpenPorchSF)
```

From the plot we can see that "OpenPorchSF" weakly positively correlates with "SalePrice".



EnclosedPorch: Enclosed porch area in square feet

There are no missing values in EnclosedPorch. We change encoding to numerical.

```
summary(dataset$EnclosedPorch)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0     0.0     0.0    23.1     0.0  1012.0

# We change encoding to numerical.
dataset$EnclosedPorch <- as.numeric(dataset$EnclosedPorch)
```

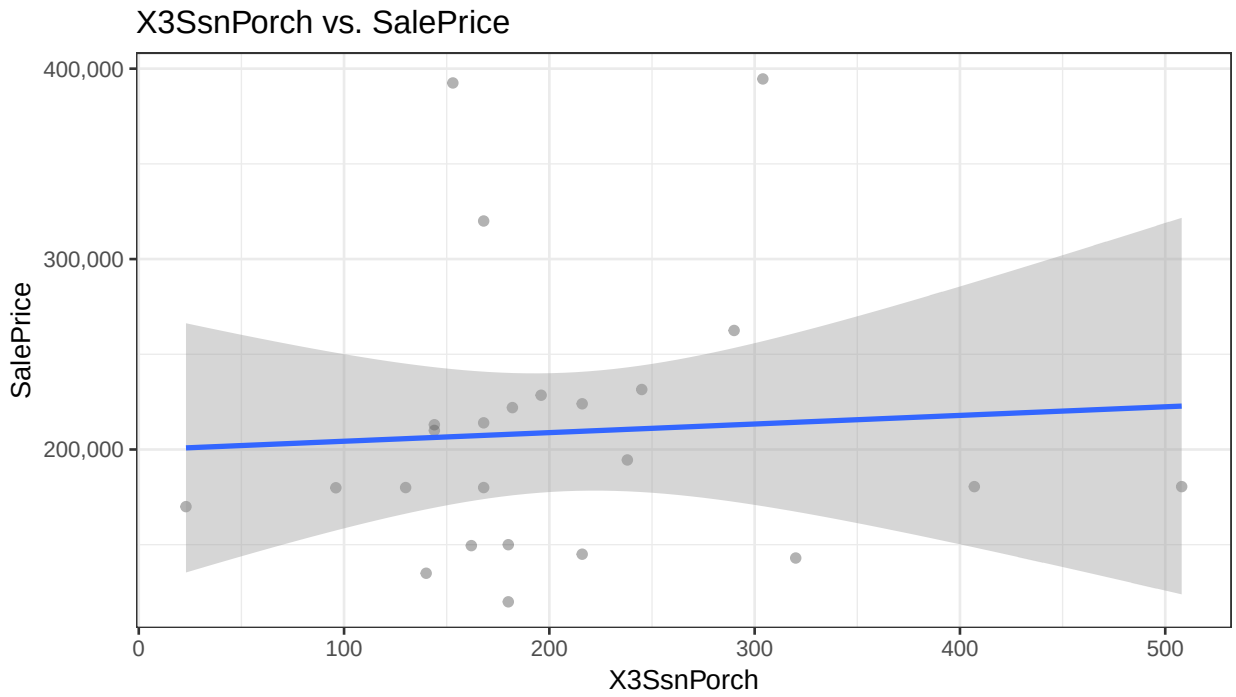
From the plot we can see that "EnclosedPorchSF" weakly positively correlates with "SalePrice".

X3SsnPorch: Enclosed porch area in square feet

There are no missing values in "X3SsnPorch". We change encoding to numerical.

```
summary(dataset$X3SsnPorch)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   0.000   2.602   0.000 508.000

# We change encoding to numerical.
dataset$X3SsnPorch <- as.numeric(dataset$X3SsnPorch)
```

From the plot we can see that "X3SsnPorch" does not seem to correlate much with "SalePrice".

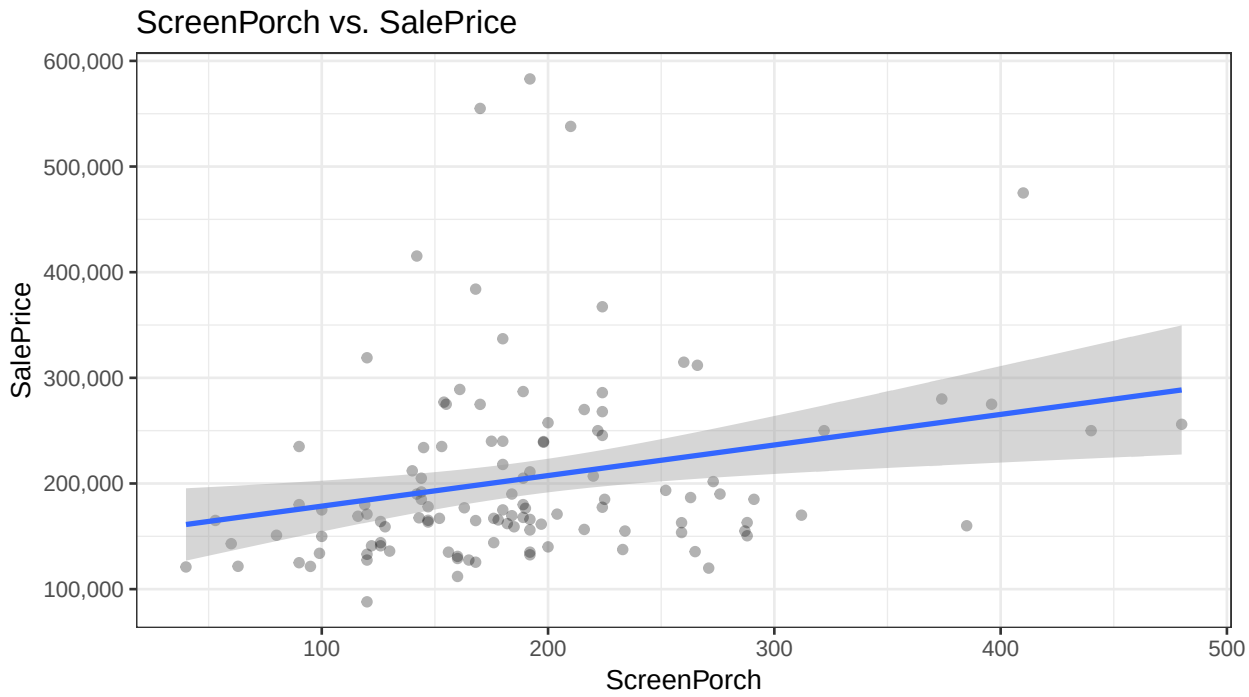


ScreenPorch: Enclosed porch area in square feet

There are no missing values in "ScreenPorch". We change encoding to numerical.

```
summary(dataset$ScreenPorch)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    0.00    0.00   16.06    0.00  576.00

# We change encoding to numerical.
dataset$ScreenPorch <- as.numeric(dataset$ScreenPorch)
```

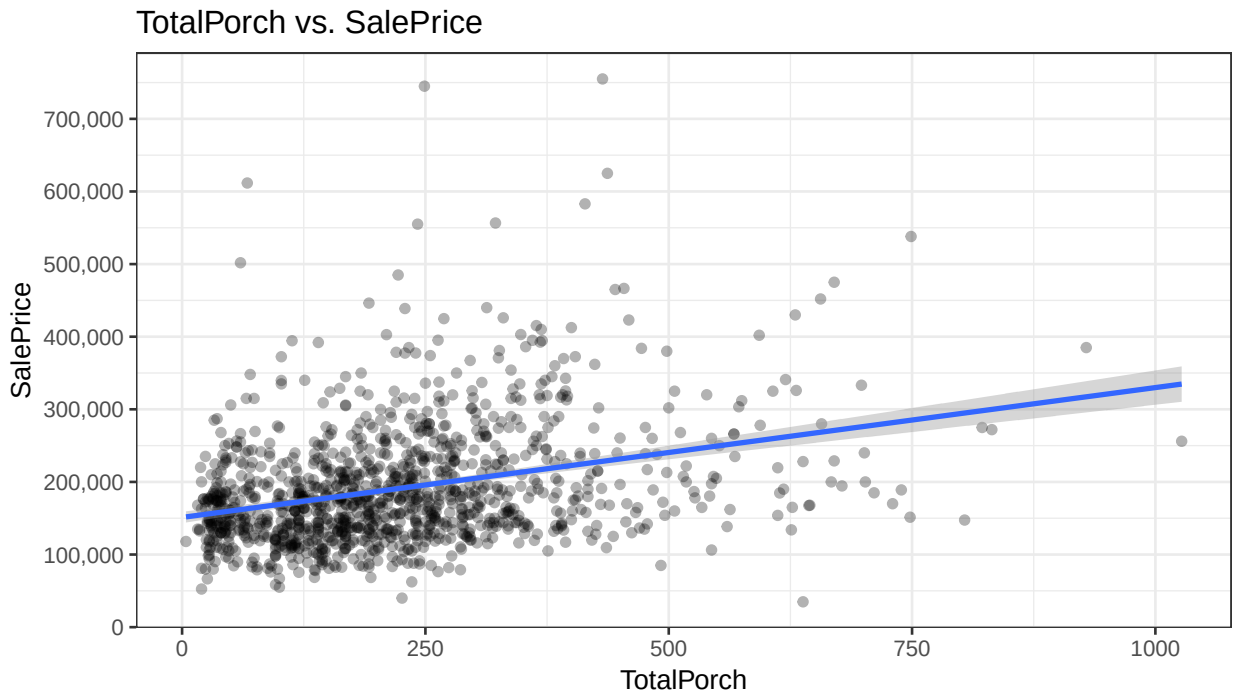
From the plot we can see that "ScreenPorch" weakly positively correlates with "SalePrice".



It seems that all the porch-related variables are individually not very predictive of "SalePrice". We will try and combine them into one new variable, "TotalPorch".

```
# Combining the individual porch-related variables into one, "TotalPorch".
dataset$TotalPorch <- dataset$WoodDeckSF + dataset$OpenPorchSF +
  dataset$EnclosedPorch + dataset$X3SsnPorch + dataset$ScreenPorch
```

We then plot "TotalPorch" against "SalePrice" for all "TotalPorch" > 0. The combined variable seems to correlate rather well with "SalePrice".



"TotalPorch" as a decent correlation with "SalePrice".

```
# Correlation between "TotalPorch" and "SalePrice".
cor(dataset[train$Id, ]$SalePrice, dataset[train$Id, ]$TotalPorch)
## [1] 0.390993
```

We will remove the individual porch-related variables from the dataset. The combined variable "TotalPorch" disregards any differenes in value between the individual variables, however.

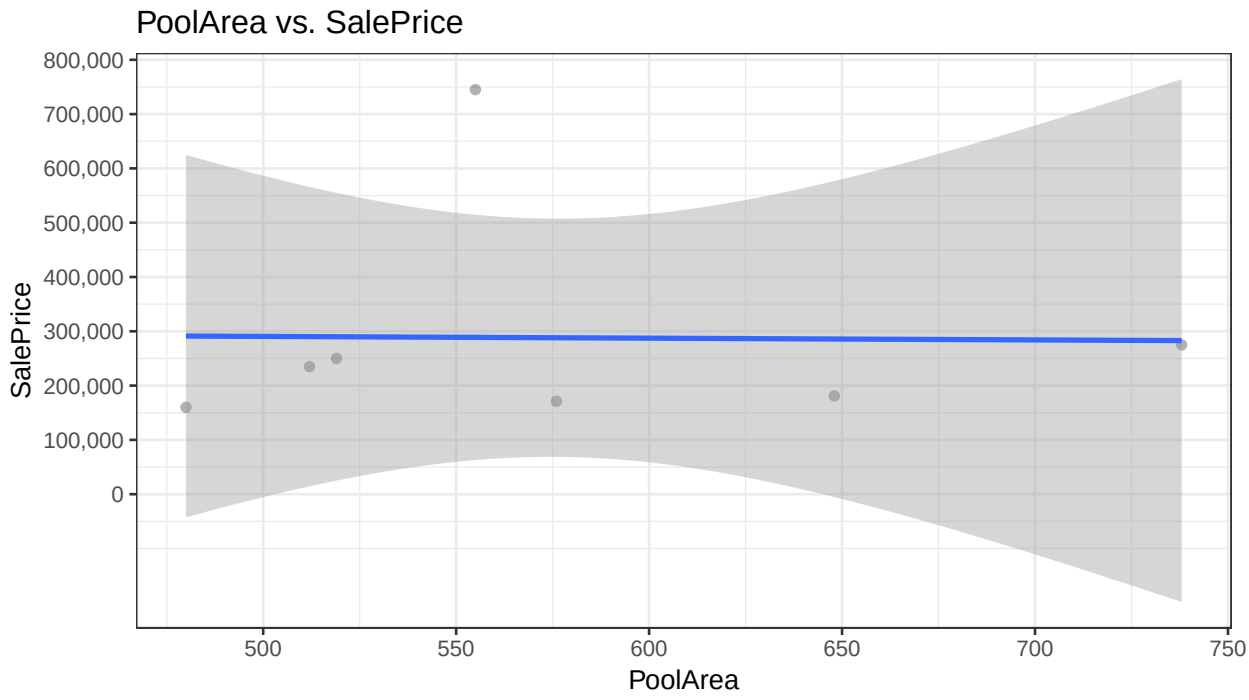
```
dataset <- subset(dataset, select = -c(OpenPorchSF,
      EnclosedPorch, X3SsnPorch, ScreenPorch, WoodDeckSF))
```

2.1.58 PoolArea: Pool area in square feet

There are no missing values in "PoolArea". We convert it to be numerical.

```
# We convert "PoolArea" to numerical.
dataset$PoolArea <- as.numeric(dataset$PoolArea)

# Summary.
summary(dataset$PoolArea)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   0.000   2.252   0.000 800.000
```

We plot "PoolArea" against "SalePrice" for all "PoolArea" > 0. "PoolArea" seems to have little predictive power in terms of "SalePrice". Also, only a few houses actually have a pool. Is having a pool regardless of size in general more important? We will explore "PoolQC".

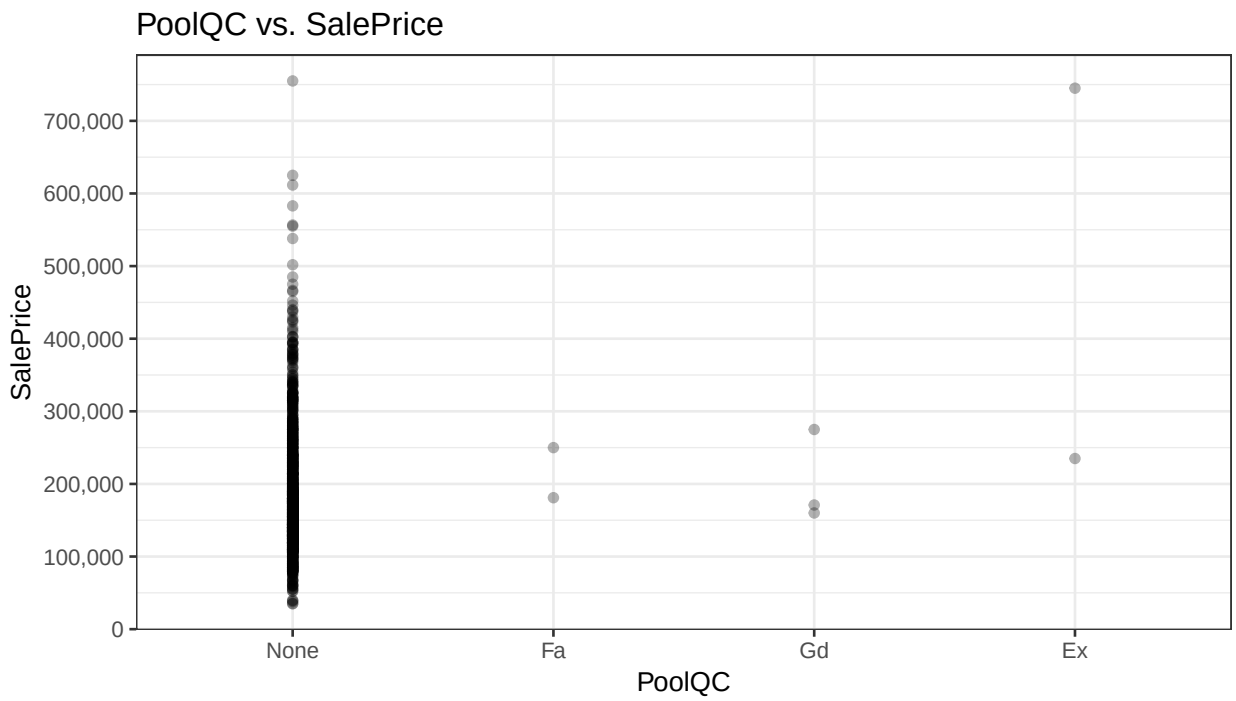


2.1.59 PoolQC: Pool quality

Almost all values are missing in "PoolQC". We know from the data description that not having a pool is wrongly encoded as "NA". We fill fix this problem and correct factor levels.

```
summary(dataset$PoolQC)
##   Ex   Fa   Gd NA's
##    4    2    4 2909

# Fixing missing values and factor levels.
dataset$PoolQC <- as.character(dataset$PoolQC)
dataset$PoolQC[which(is.na(dataset$PoolQC))] <- "None"
dataset$PoolQC <- factor(dataset$PoolQC, levels = c("None",
                                        "Fa", "Gd", "Ex"))
```

As described above, few houses actually have a pool. Both pool-related variables seem to encode very little information.

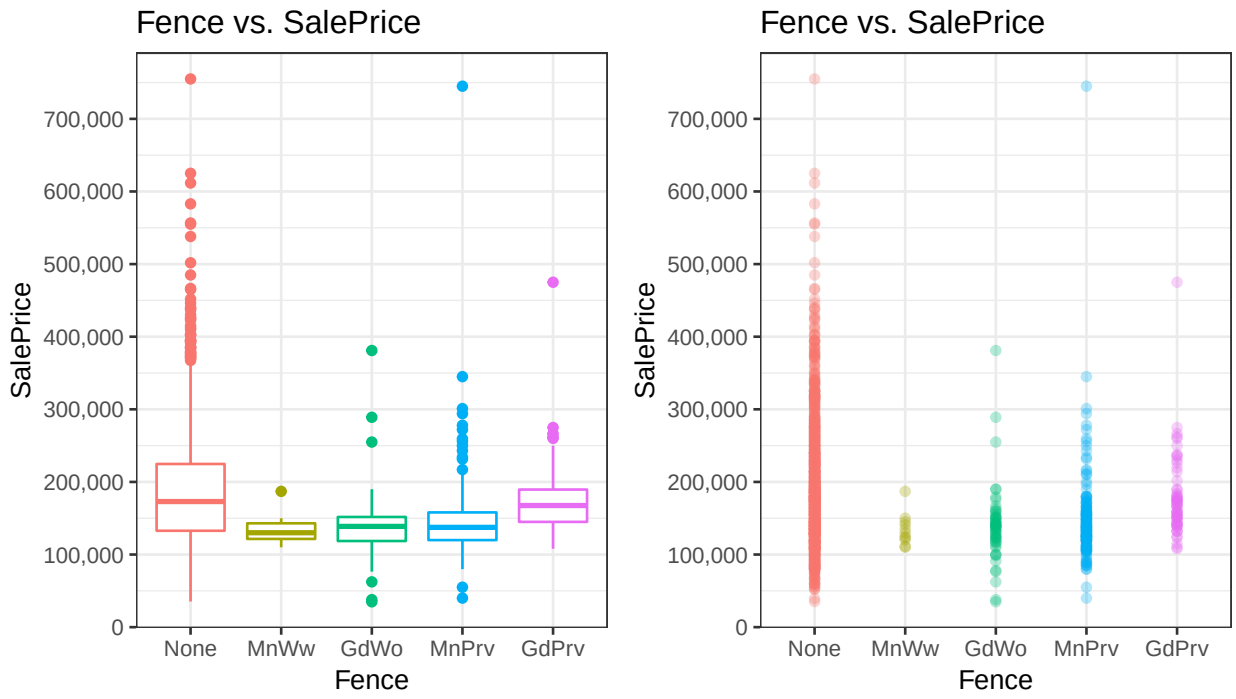


2.1.60 Fence: Fence quality

Most values are missing from "Fence". We know from the data description that not having a fence is wrongly encoded as "NA". We fill fix this problem and correct factor levels.

```
summary(dataset$Fence)
## GdPrv  GdWo MnPrv  MnWw  NA's
##   118   112   329    12  2348

# Fixing missing values nad factor levels.
dataset$Fence <- as.character(dataset$Fence)
dataset$Fence[which(is.na(dataset$Fence))] <- "None"
dataset$Fence <- factor(dataset$Fence, levels = c("None",
                      "MnWw", "GdWo", "MnPrv", "GdPrv"))
```

From the plots we can see that "Fence" is not giving much information at all. "Fence" seems to have little importance in terms of "SalePrice".

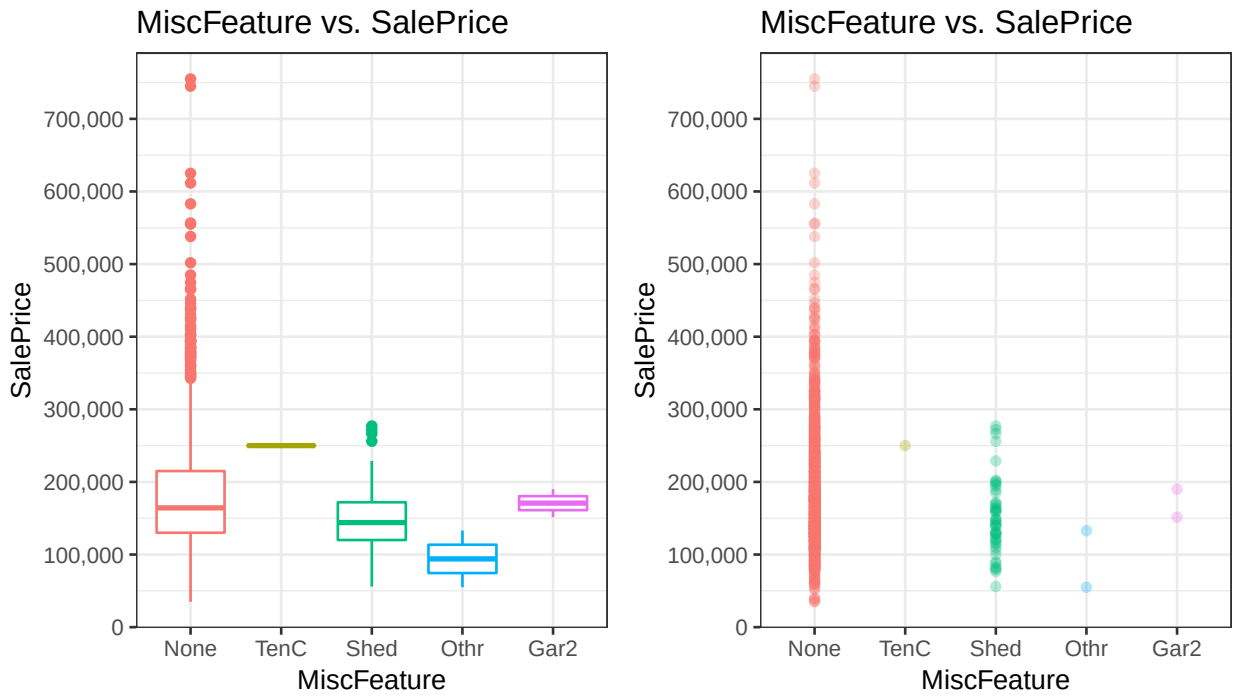


2.1.61 MiscFeature: Miscellaneous feature not covered in other categories

Most values are missing from "MiscFeature". We know from the data description that not having a "MiscFeature" is wrongly encoded as "NA".

```
# We fill fix this problem and correct factor levels.
summary(dataset$MiscFeature)
## Gar2 Othr Shed TenC NA's
##    5    4   95    1 2814

# Fixing missing values and factor levels.
dataset$MiscFeature <- as.character(dataset$MiscFeature)
dataset$MiscFeature[which(is.na(dataset$MiscFeature))] <- "None"
dataset$MiscFeature <- factor(dataset$MiscFeature, levels = c("None",
                              "TenC", "Shed", "Othr", "Gar2", "Elev"))
```

From the plots we can see that while having a tennis court might increase "SalePrice", there are simply not enough data points for the "MiscFeature" variable to be very useful.

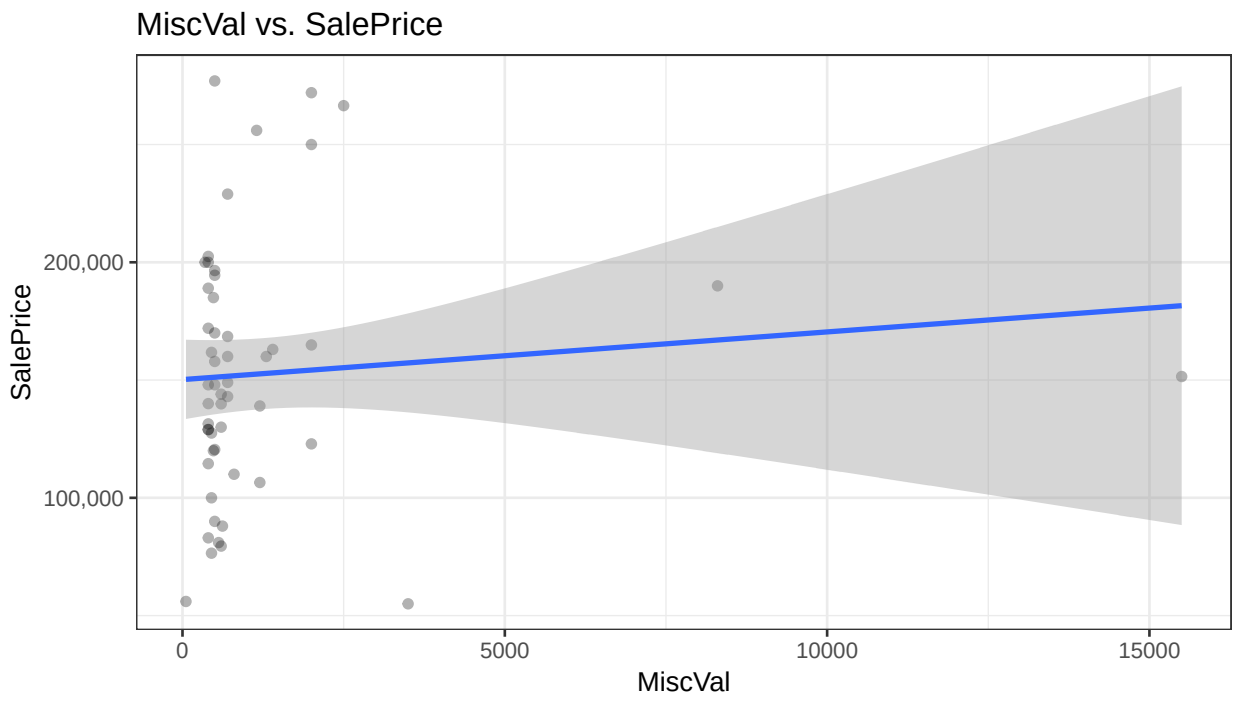


2.1.62 MiscVal: $Value of miscellaneous feature

There are no missing values in "MiscVal". We convert it to be numerical.

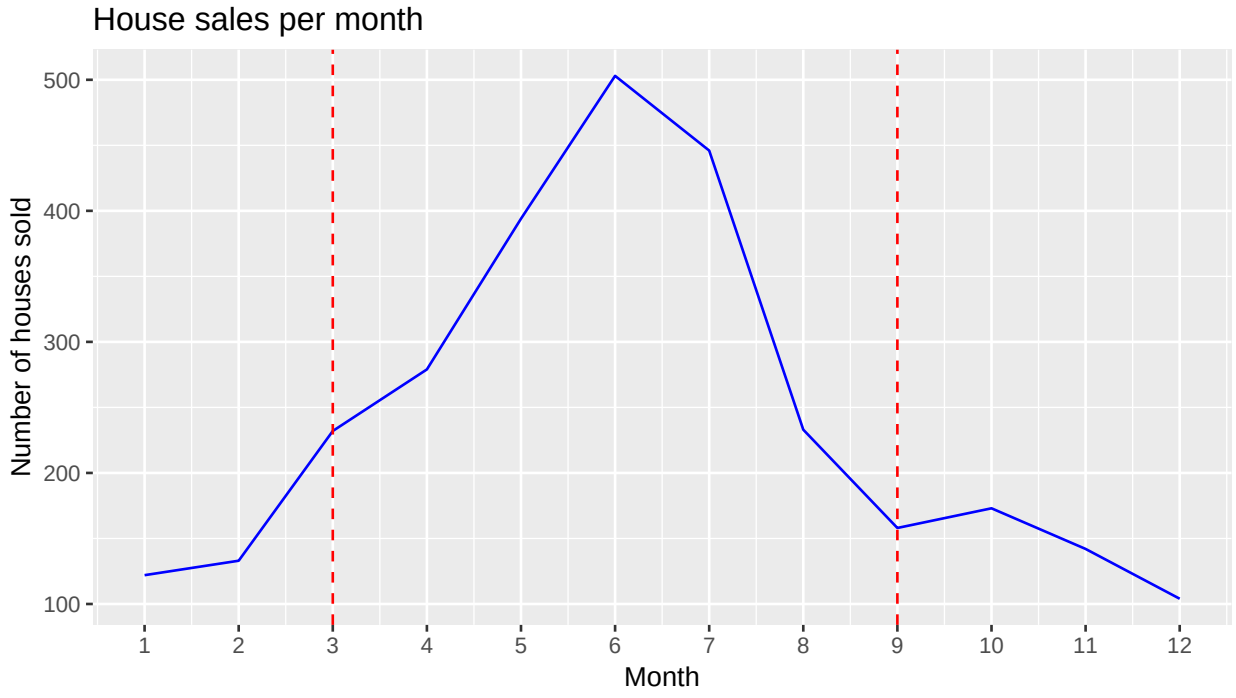
```
# We convert "MiscVal"" to numerical.
dataset$MiscVal <- as.numeric(dataset$MiscVal)

# Summary.
summary(dataset$MiscVal)
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##     0.00     0.00     0.00    50.83     0.00 17000.00
```

From the plots we can see that the Dollar-value of "MiscFeatures" is not necessarily the best indicator for "SalePrice" increases.

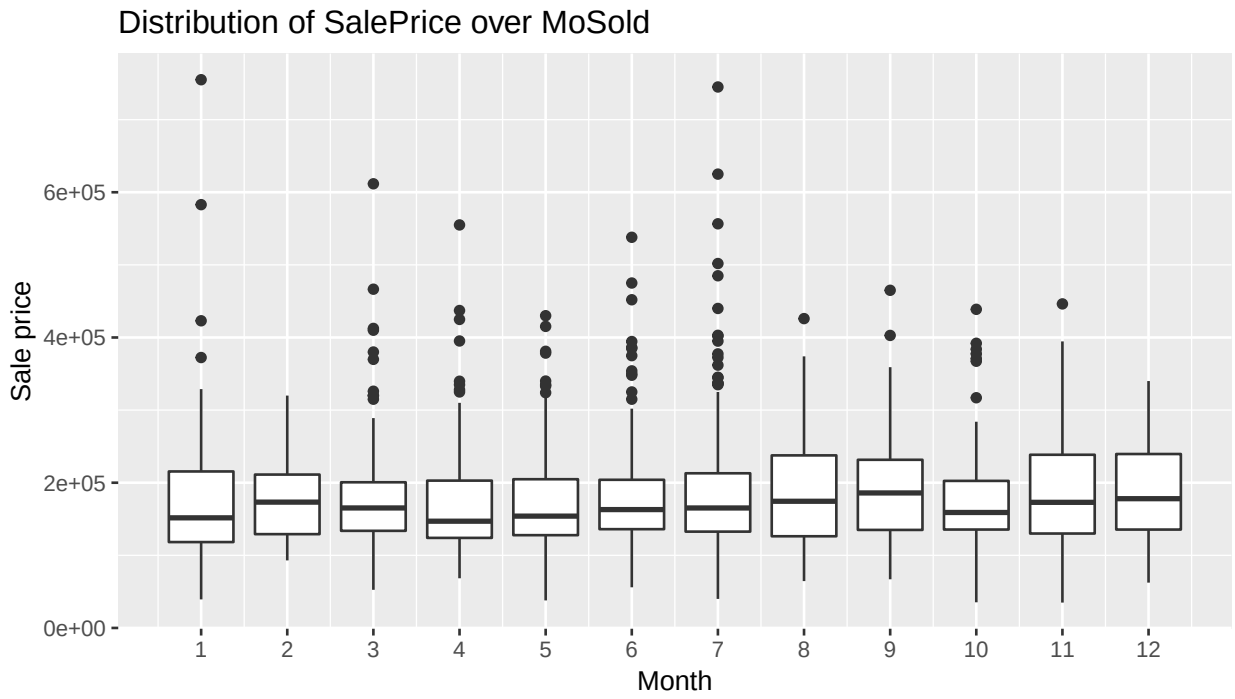


2.1.63 MoSold: Month Sold (MM), YrSold: Year Sold (YYYY)

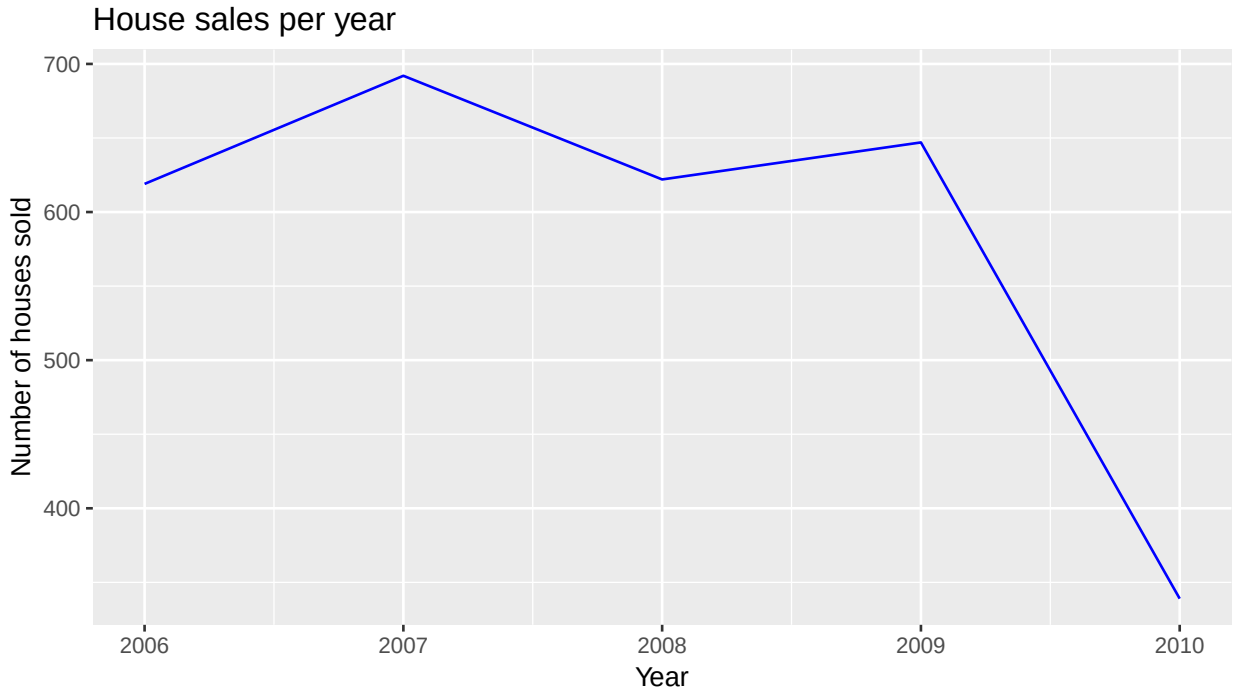
We plot "MonthSold" and "YearSold" - do these features influence "SalePrice"? We plot the amount of sold houses per month. Most houses are sold during the Spring/Summer months of the year, as indicated by the red dashed lines.



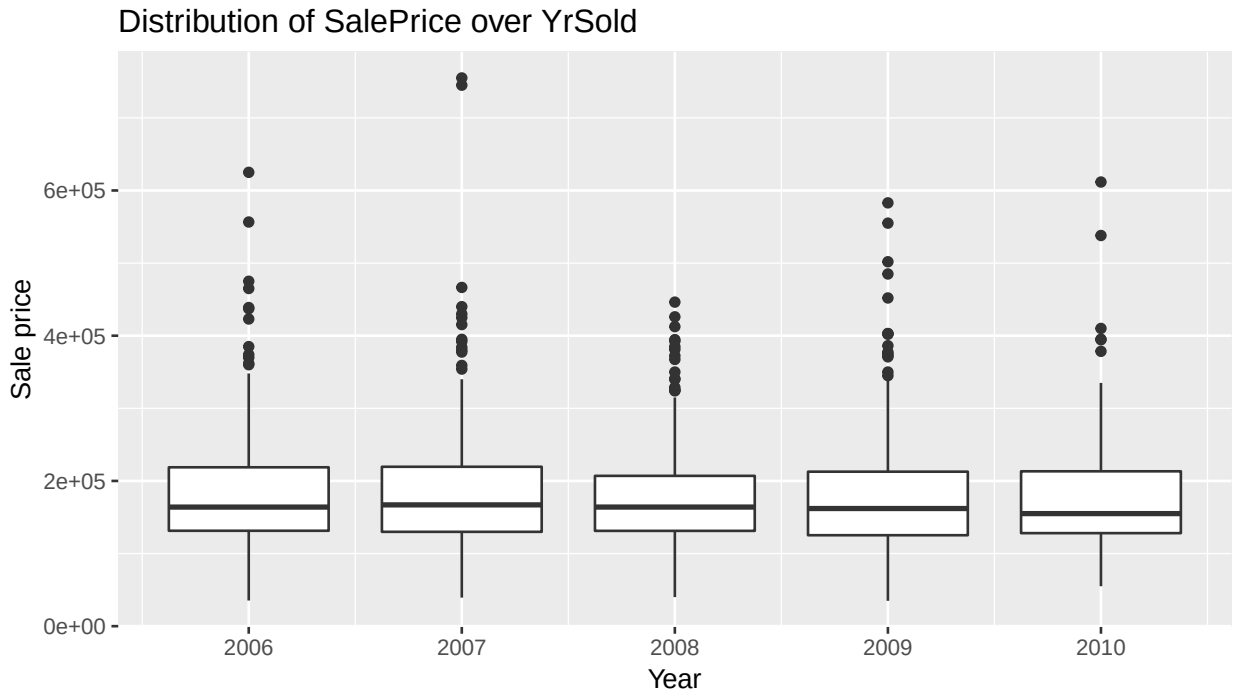
Does this influence “SalePrice”? “MoSold” doesn’t seem to influence “SalePrice” at all. There are simply more houses being sold in the summer months.



We plot the amount of sold houses per year. The amount is relatively similar, but much less entries are in 2010, the most recent year in the dataset.



"YrSold" doesn't seem to influence "SalePrice" at all, even though we know that the financial crisis of 2008 might have impacted prices.



2.1.64 SaleType: Type of sale

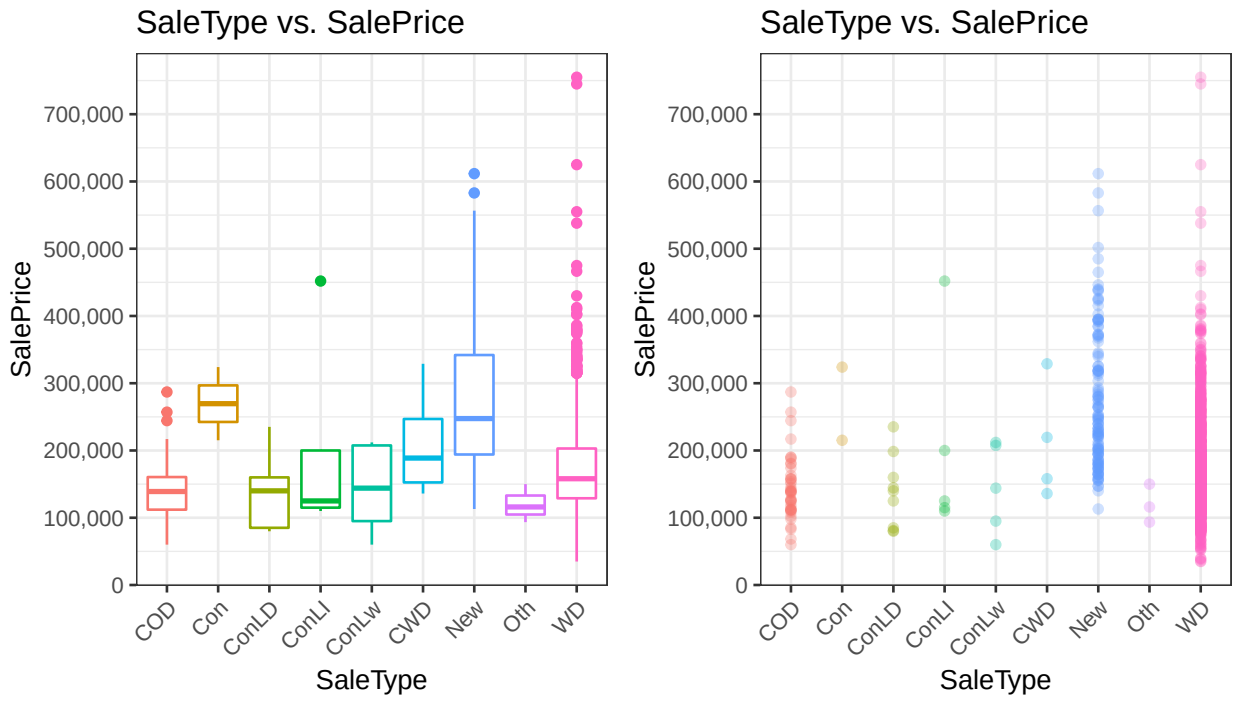
There is a missing value in "SaleType".

```
summary(dataset$SaleType)
##   COD   Con ConLD ConLI ConLw   CWD   New   Oth    WD  NA's
##    87     5    26     9     8    12   239     7  2525     1
```

We use kNN-based missing value imputation. The model predicts "WD", the mode, as the value to impute for the missing "SaleType" entry.

```
#kNN-model.
knn_model <- kNN(dataset, variable = "SaleType", k = 9)

# Predicted "SaleType" value. "WD" is the most common value for "SaleType".
knn_model[knn_model$SaleType_imp == TRUE, ]$SaleType
## [1] WD
## Levels: COD Con ConLD ConLI ConLw CWD New Oth WD

# We impute the value.
dataset$SaleType[which(is.na(dataset$SaleType))] <-
  knn_model[knn_model$SaleType_imp == TRUE, ]$SaleType
```

From the plots we can see that some "SaleTypes" occur more seldom than others and that sales of new houses and warranty deed conventional ("WD") are indicative of higher "SalePrice". Some categories have simply too few entries.

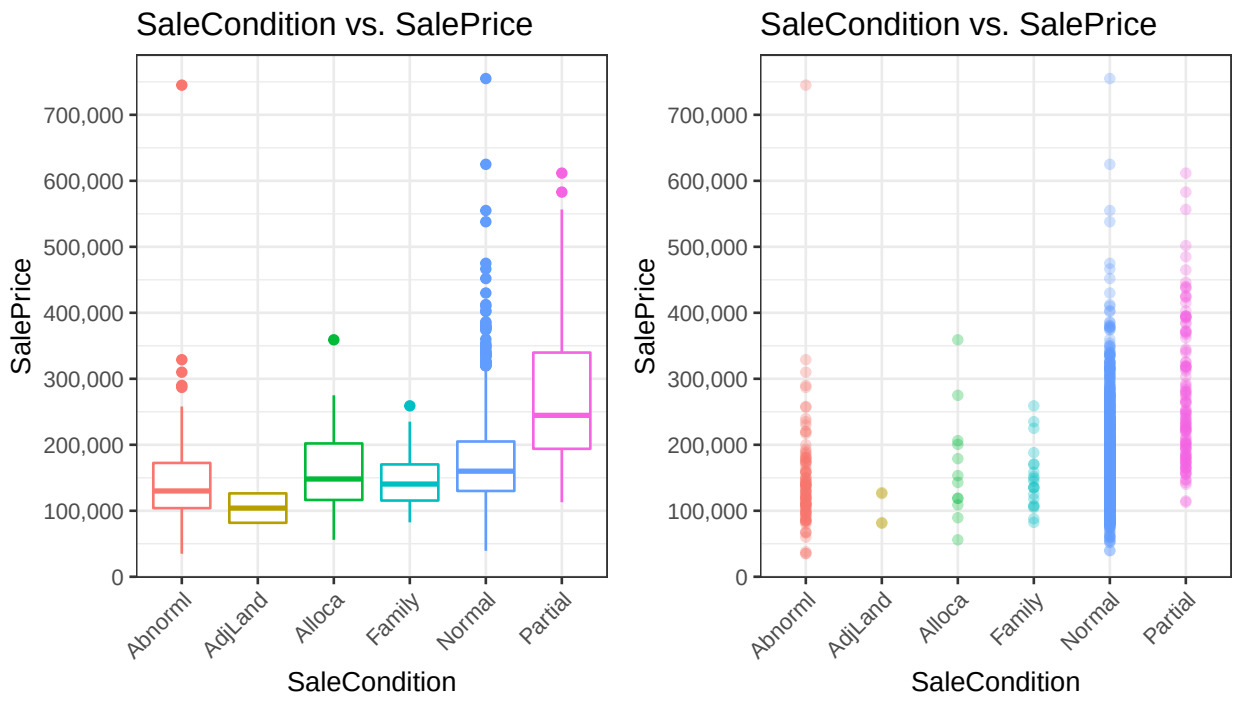


2.1.65 SaleCondition: Condition of sale

There are no missing values in "SaleCondition".

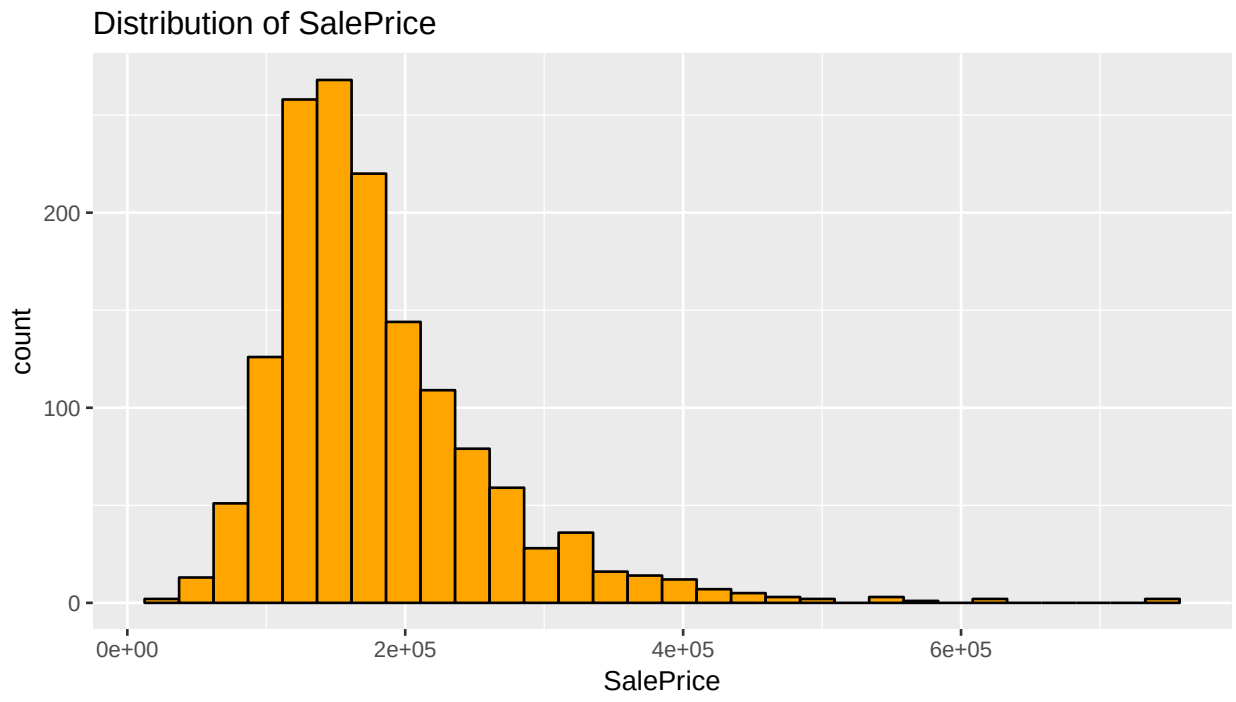
```
summary(dataset$SaleCondition)
## Abnorml AdjLand  Alloca  Family  Normal Partial
##     190      12      24      46    2402     245
```

From the plots we can see that "partial", as a measure of "newness" seems to indicate higher "SalePrice". Most houses are sold under normal conditions.



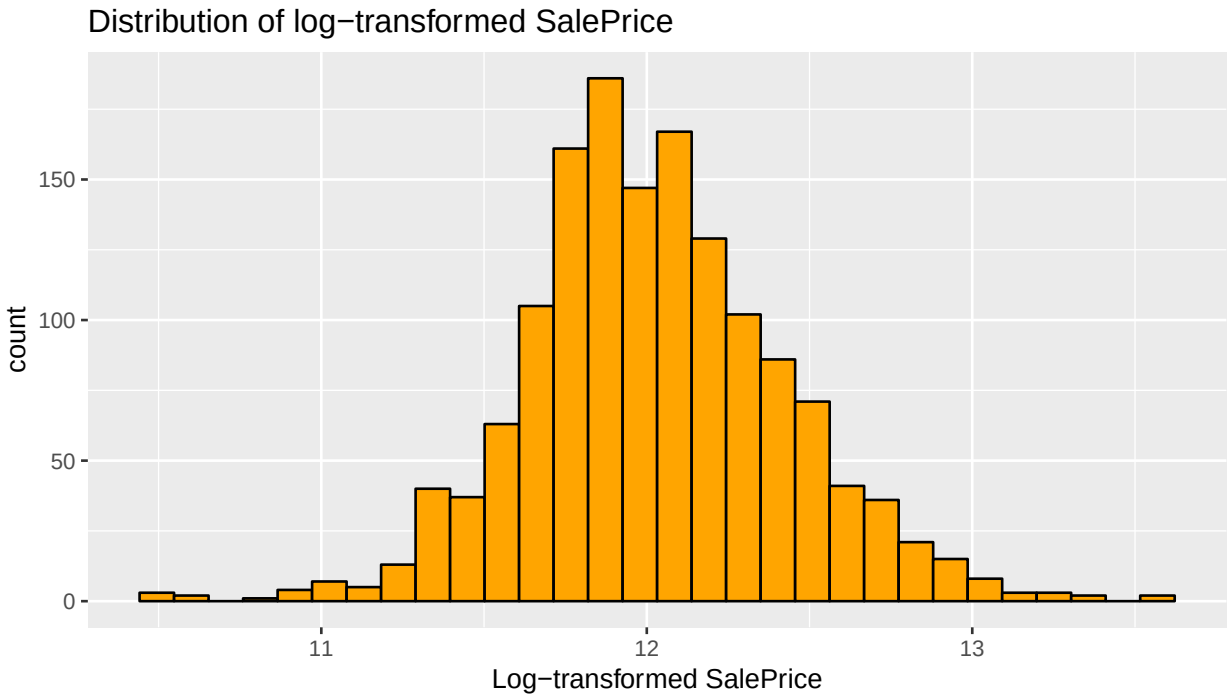
2.1.66 Skewedness of the outcome variable SalePrice

Monetary values are often log-normally distributed. How does "SalePrice" look like? The histogram below indicates a certain skewedness.



We can check for skewedness with a function. The skew is considerably higher than 0.8. A log-transformation could potentially lead to "SalePrice" being more normal. It is plotted below.

```
# Determine skew of "SalePrice".
e1071::skewness(dataset[train$Id, ]$SalePrice)
## [1] 1.879009

# Plotting the log-transformation.
dataset[train$Id, ] %>%
  ggplot(aes(log1p(SalePrice))) +
  geom_histogram(bins = 30, color = "black", fill = "orange") +
  xlab("Log-transformed SalePrice") +
  ggtitle("Distribution of log-transformed SalePrice")
```



We log-transform "SalePrice".

```
dataset$SalePrice <- log1p(dataset$SalePrice)
```

2.2 Summary of exploratory data analysis and dataset manipulations

The purpose of the somewhat exhaustive exploratory data analysis was to develop a certain intuition about all the different features and their level of potential influence over the outcome variable "SalePrice". While box- and scatter-plots certainly don't reveal more intricate relationships between certain variables, they do however help to get a good idea about the general direction and distribution of categories within a feature.

Some light feature engineering was conducted in an attempt to improve the value of some of the features. It would also have been possible to exclude some more of the variables with little influence on "SalePrice" or those which might do more harm than good (e.g. due to having underreprsented catagories).

All missing values have been dealt with and we can once again separate `dataset` into `train` and `test`. The temporary "SalePrice" column in `test` is removed.

```
train <- dataset[train$Id, ]
test <- subset(dataset[test$Id, ], select = -SalePrice)
```

Missing values were mostly produced by erroneous encoding in the dataset, where the absence of a garage for example was entered as "NA". Actual missing values were mostly dealt with via kNN-based imputation. The kNN algorithm mostly predicted the most common value for a given feature/category. The value for the k-nearest neighbours was set to k = 9 arbitrarily and could have been optimized for each individual case. The kNN algorithm assumes that a value can be estimated/approximated by the values that are closest to it, based on other variables (i.e. houses with very similar other characteristics).

3 Modelling approaches and results

First we write a loss-function to determine the residual mean squared error, or RMSE of the model. The function calculates the residuals/error and then takes the root mean square of them.

Next, we split `train` into a separate `train_set` and `test_set` for algorithm evaluation purposes (we won't use the real `test` subset for this and treat it as completely new data (the "hold-out" set) for final evaluations only).

3.1 Simple linear regression as a baseline model

As our first, very simple model we predict house "SalePrice" via simple linear regression with just "GrLivArea". We also generate a table to keep track of the RMSEs our various models generate. Simple linear regression with a single predictor, "GrLivArea", achieved the RMSE printed below.

Model	RMSE
Simple_lm	0.2876433

3.2 Multivariate linear regression utilizing several predictors

Multivariate linear regression utilizing several predictors achieved a much lower RMSE, which is printed below.

Model	RMSE
Simple_lm	0.2876433
Multi_lm_several	0.1901242

3.3 Multivariate linear regression utilizing all available predictors

Multivariate linear regression utilizing all available predictors achieves an even lower RMSE, which is printed below.

Model	RMSE
Simple_lm	0.2876433
Multi_lm_several	0.1901242

Model	RMSE
Multi_lm_all	0.1393704

3.4 Gradient boosting machine model: XGBoost

We will utilize the XGBoost algorithm to build models to predict "SalePrice". XGBoost requires the dataset to be entirely in numerical format, which can be achieved via so-called "one-hot-encoding" of the variables. One way to do this is to use the "vtreat" package and its function "designTreatmentsZ", which devises a "treatment plan" to "one-hot-encode" all relevant variables at once. This "treatment plan" is then used via the `prepare()` function to do the "one-hot-encoding" on the `train_set` and on the `test_set`. We thus generate `train_set_treated` and `test_set_treated` for use with XGBoost.

Next we will use the `xgb.cv()` function to determine the total number of rounds `nrounds` that improve RMSE until only the training RMSE reduces further, while the cross-validated RMSE already reached a minimum. While the training RMSE may continue to decrease on more and more rounds of boosting iterations ("overfitting"), the test RMSE usually does not after some point. After running `xgb.cv()` we can access its event log to find the optimal number of iterations. As the treated `train_set` no longer contains the outcome variable "SalePrice", we have to use the untreated `train_set` to provide it as a `label`. For our baseline XGBoost model, we will use mostly default parameters. We print the determined RMSE value below.

Model	RMSE
Simple_lm	0.2876433
Multi_lm_several	0.1901242
Multi_lm_all	0.1393704
XGBoost_baseline	0.1455617

The RMSE has worsened compared to the simple linear regression approach. But can we do better? We try and tune our XGBoost-model with the `caret` package.

3.5 Hyperparameter tuning of the XGBoost algorithm "xgbTree" with caret

We will evaluate different values for the hyperparameters of the "xgbTree" algorithm in the `caret` package. As expansive grid searches become computationally very expensive the more parameters are evaluated at the same time (easily into the thousands of models), we will instead evaluate no more than 2 parameters per tuning round and try to visually inspect their influence on model performance. In the first round of tunings, we will determine `max_depth`, the maximal depth of trees to use. Larger trees allow faster learning, but make the model more and more complex and prone to overfitting. The increased model complexity means that the model learns variable relations that might be very specific to the training data and not generalize well to new data. In addition, we will examine different values of `min_child_weight`, which defines the minimum sum of weights of all observations required in a child. Higher values here will counteract overfitting by preventing the model from learning variable relations that might be very specific to the particular sample which was selected for a particular tree. Large values of `min_child_weight` might lead to underfitting the training data. We will select some fairly standard values for the other hyperparameters to avoid exploding the number of models we will train in this first tuning step. In addition, we make use of the `caret::preProcess` option to remove near-zero variance predictors and to center and scale our data prior to model fitting. Please note that the range of selected hyperparameters has been reduced empirically to reduce computational time as much as possible. Additionally, the learning rate `eta` has been set at a relatively low value of 0.05 to get a fine-grained step size.

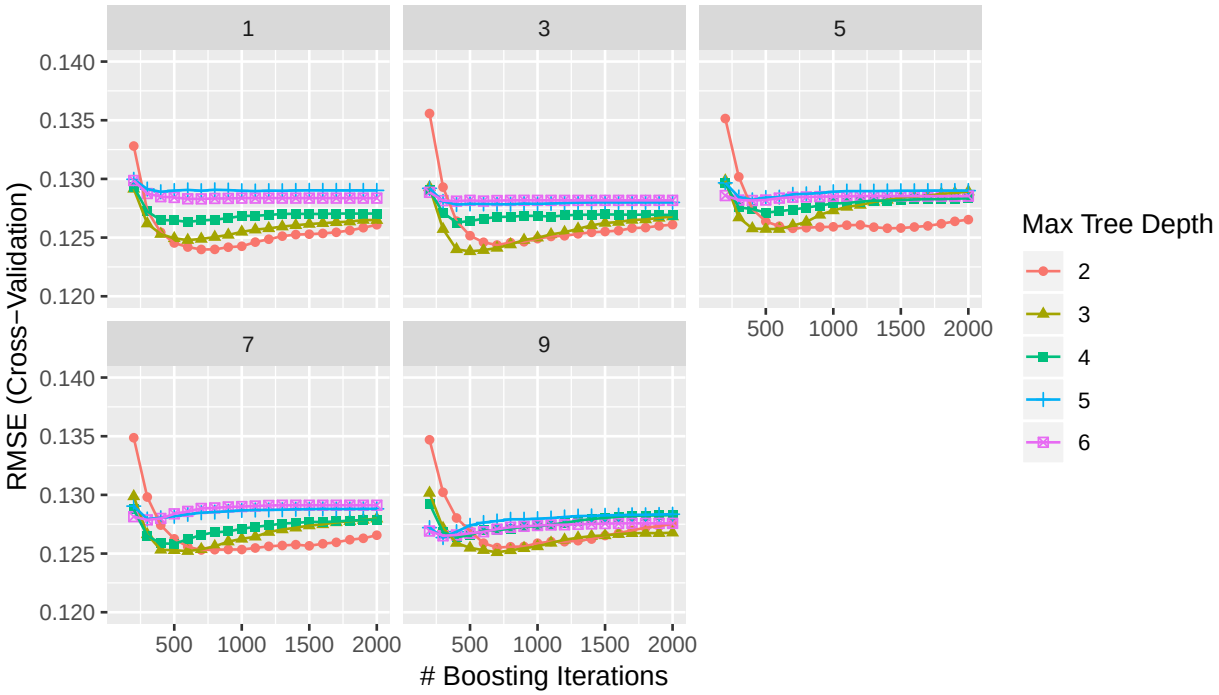
After training and fitting the model, we will visually inspect which set of hyperparameters achieved the lowest

RMSE value as determined by 20-fold cross-validation. Afterwards we will predict against our `test_set` ("hold-out-set") and determine "out-of-bag" RMSE.

3.5.1 First round of hyperparameter tuning: max_depth and min_child_weight

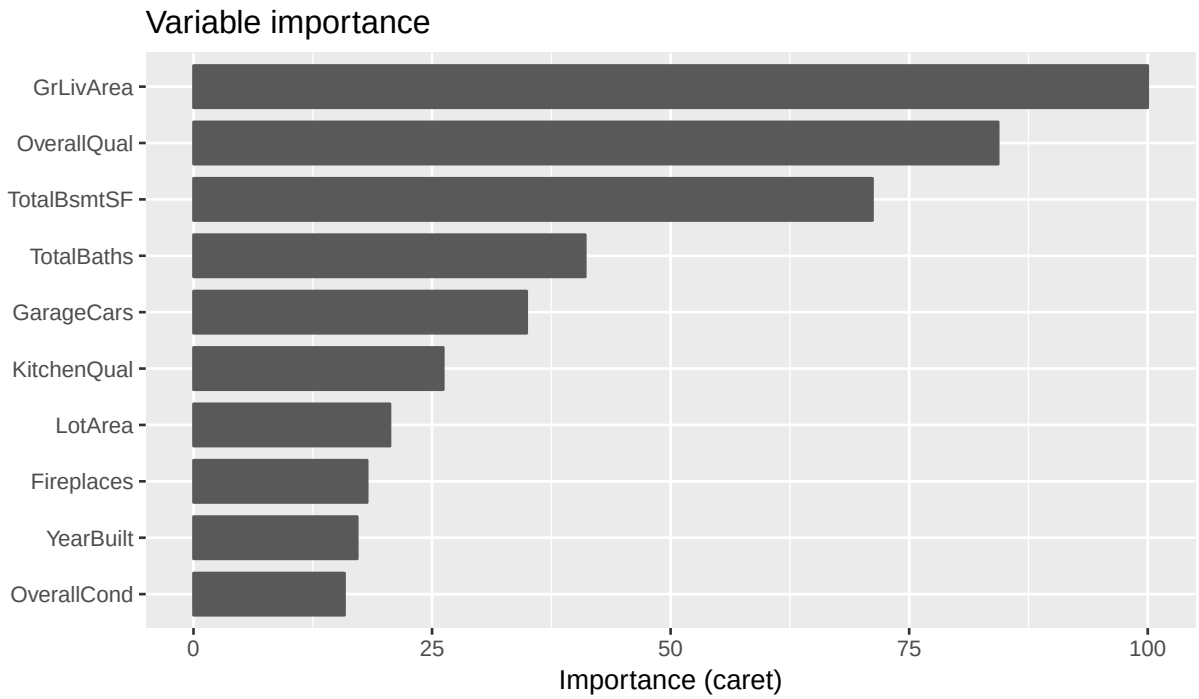
In this first round of hyperparameter tuning we select the optimal values for `max_depth` and `min_child_weight`. Standard values were set for the remaining hyperparameters `colsample_bytree` and `subsample`, which will be optimized in the following tuning steps. The learning rate `eta` is fixed to 0.05 and different values will be investigated during the final tuning round. We print the best determined tuning parameters below and visualize the process.

```
##     nrounds max_depth  eta gamma colsample_bytree min_child_weight
## 125     500         3 0.05     0                1                3
##     subsample
## 125         1
```



From the tuning plots and the selected best parameters we can see that trees of `max_depth` 3 were chosen, along with a `min_child_weight` of 3. Higher values of `max_depth` require less iterations to converge, but don't reach as low RMSE values as more shallow trees do.

We can also visualize which variables were most important in the modelling process. Unsurprisingly, "GrLivArea" and "TotalBsmtSF" as measures of living space are extremely important in determining "SalePrice". "OverallQual" is the most important variable. "GarageCars" and "TotalBaths" can also be viewed as measurements of house size and living space. The "YearBuilt" is also very important, as is the "KitchenQual" and "OverallCond".



We can now predict on `test_set` and record "out-of-bag" RMSE.

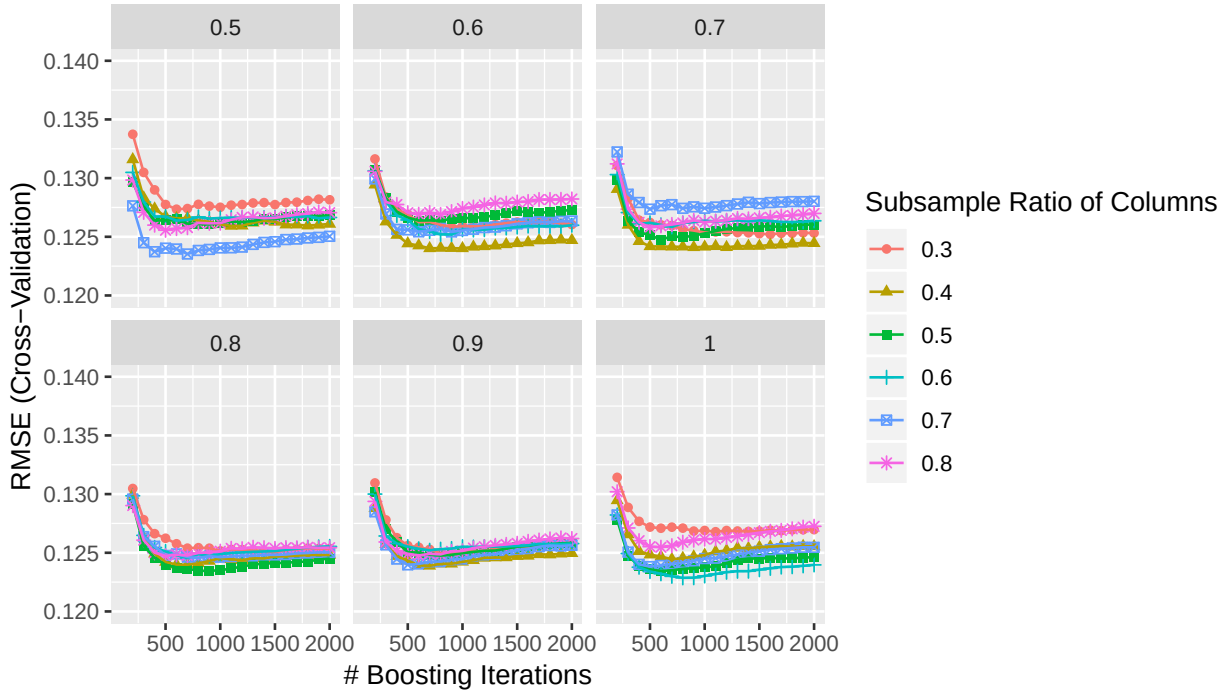
Model	RMSE
Simple_lm	0.2876433
Multi_lm_several	0.1901242
Multi_lm_all	0.1393704
XGBoost_baseline	0.1455617
caret_xgbTree_1st_tune	0.1263893

The RMSE has improved significantly with these modelling settings. Next, we will tune the remaining hyperparameters to see whether we can improve things even further.

3.5.2 Second round of hyperparameter tuning: colsample_bytree and subsample

After we determined good values for `max_depth` and `min_child_weight` in the first tuning round, we will now test several different values for `colsample_bytree` and `subsample`. The latter denotes the fraction of observations to be randomly samples for each tree, thereby preventing overfitting if values are lower. For example, a setting of 0.7 would mean that "xgbTree" would randomly sample 70% of the training data prior to growing trees in each iteration. `colsample_bytree` acts in similar fashion and denotes the subsample ratio of columns when constructing each tree. We print the best determined tuning parameters below and visualize the process.

```
##     nrounds max_depth  eta gamma colsample_bytree min_child_weight
## 468     800         3 0.05     0              0.6                3
##     subsample
## 468         1
```



A combination of 0.6 `colsample_bytree` and 1 `subsample` was chosen. We can now predict on `test_set` and record "out-of-bag" RMSE. We note that RMSE was substantially improved with these tuning settings.

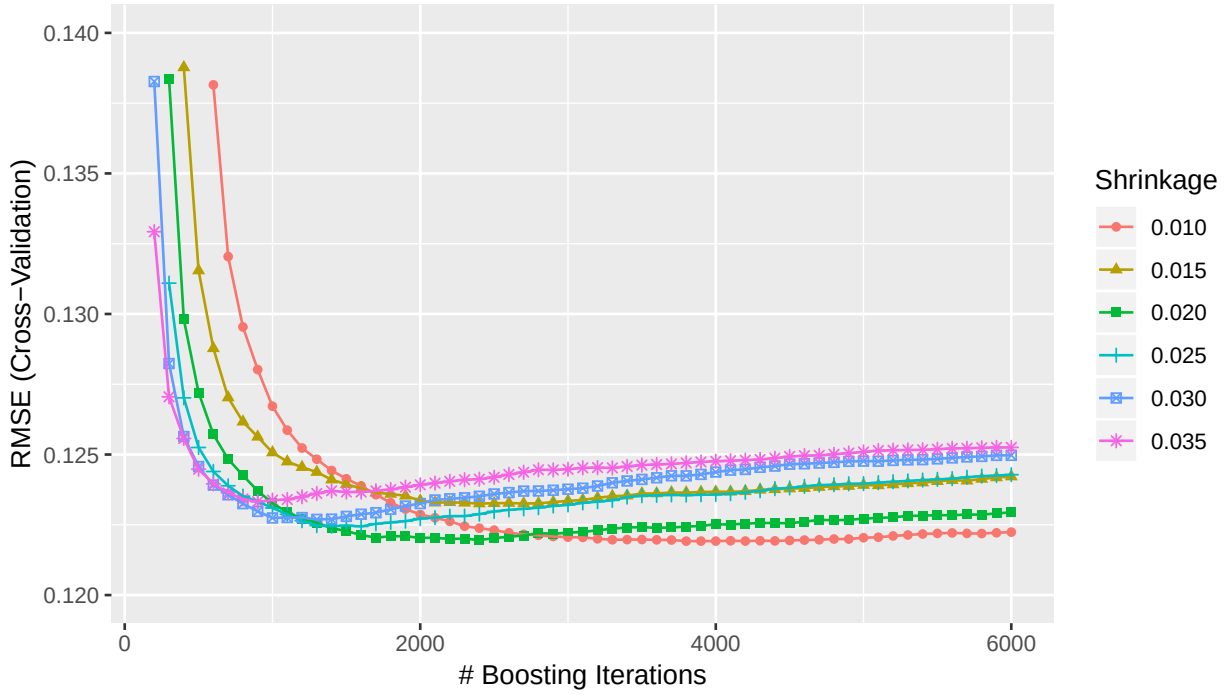
Model	RMSE
Simple_lm	0.2876433
Multi_lm_several	0.1901242
Multi_lm_all	0.1393704
XGBoost_baseline	0.1455617
caret_xgbTree_1st_tune	0.1263893
caret_xgbTree_2nd_tune	0.1212019

3.5.3 Fitting the model with optimized hyperparameters on the entire training subset

As a final step we fit the model with the above determined parameters on the entire training subset `train`. Accordingly, `train` has to be prepared via "one-hot-encoding". We can then fit the final "xgbTree" model with the hyperparameters from the 2nd tuning round. The optimal number of iterations and the learning rate `eta` is chosen at this point. We can print the best tuning parameters below.

```
# Print the best tuning parameters.
xgb_final_model$bestTune
##    nrounds max_depth  eta gamma colsample_bytree min_child_weight
## 37    3800         3 0.01     0              0.6                3
##    subsample
## 37         1
```

We can then visualize the final model and take a look at the lowest cross-validation RMSE value printed below, prior to predicting against `test` (which doesn't contain a "SalePrice" column).




```
## [1] 0.1219193
```

Finally, we can predict on `test` and prepare the submission file for entry into the *Kaggle* competition, where the submission is evaluated and scored on a public leaderboard. At the time of this writing, the generated submission file scored an RMSE of 0.12243 and achieved a rank of 1267 on the public leaderboard of the respective competition. The public leaderboard is calculated with approximately 50% of the test data and the final results will be based on the other 50%, so final scores are subject to change. Curiously, the actual public score at *Kaggle* is quite close to the RMSE determined by cross-validation during the final model fitting above (RMSE of 0.1219193). As the cross-validation folds are not exactly the same between different runs of the script, the exact results may vary.

4 Concluding remarks

For this "Choose your own" capstone project I've conducted exploratory data analysis, wrangled the data in various ways, and built a machine learning algorithm with the `caret` package to predict house prices in a dataset from *Kaggle*. The dataset came with a relatively large number of variables containg all kinds of information about houses. I've shown that certain variables have a much larger influence on a houses' sale price than others and how some variables could potentially do more harm than good if, for example, they have many categories with very few entries. I've imputed missing values via kNN-based predictions wherever feasible and noted that mostly the variables' mode was predicted. Most missing values were generated by erroneous variable encoding in the dataset itself, which was easily fixed by simple replacement with appropriate terms. I've conducted some slight feature engineering, for example by combining several individual variables into one.

I developed my algorithm in several steps by tuning different sets of hyperparameters on a training subset and then validating its performance on a test subset of data. The thusly determined hyperparameters were then used in a final model fitting process on the entire `train` dataset. The predictions of the resulting model on `test` were then used to generate a submission file, which was submitted to *Kaggle* and achieved an public leaderboard rank of 1267 with an RMSE of 0.12243 at the time of writing (some of the best submitted models score RMSEs of around 0.105). I've also compared the performance of my developed algorithm with the performance of a simple multivariate linear regression, which performed quite well considering its minimal computational cost in comparison. The exact results may vary between different executions of the code, as the cross-validation folds differ at each run.

Further improvements in model performance could be achieved by further feature engineering and by employing an ensemble of well-trained machine learning algorithms ("stacked regressions"). However, the goal of my project was not to score the lowest possible RMSE, but to practice data exploration, wrangling and building a relatively simple and interpretable algorithm. Due to the sheer number of features, I ended up mostly not going into greater depth in my data exploration. By systematically exploring each and every feature of the dataset, I acquired a certain intuition about their possible contribution to the outcome variable "SalePrice". It also helped in dealing with missing values and finding opportunities for some slight feature engineering. Optimally, I would have tested the influence on model performance of each and every modification I have applied to the dataset to test and see whether it was actually beneficial. However, this would have required very frequent re-running of the modelling process and probably would lead to a large increase in the size of the report. I ended up completely re-doing the entire data exploration part three times and ultimately decided not to test the influence of my dataset modifications in isolation. I also spent large amounts of time selecting hyperparameters for my algorithm and have probably re-run every part over a hundred times (throughout, I was able to iteratively improve my algorithms' score on *Kaggle*), even though the improvements in terms of RMSE were relatively minor. Through this I learned that simply "turning the big knobs" (as in, `nrounds`, `max_depth`, and `eta` for "xgbTree" in `caret`) already leads to decent performance improvements.

5 Acknowledgements

I would like to thank the entire edx course staff for providing a very stimulating learning environment throughout all of the courses and for making these learning materials available and exciting. I thank all the active edx forum users for providing help and ample discussion. I thank all the people who helped in providing the dataset. And finally, a big thank you to my peers for spending their time and effort to review my capstone projects.