HarvardX Data Science Professional Certificate PH125.9x Capstone 2: Housing Sale Price Prediction Using a Machine Learning Algorithm

Willie Costa

2020-05-03

Contents

Executive Summary

A machine learning model is presented to estimate the sale prices of homes based on physical characteristics (e.g. age and condition), finish, building material, proximity to potentially desirable and undesirable geographic and logistical features, as well as traditional metrics such as bedroom and bathroom count. The dataset used for this study is the Ames, Iowa housing dataset, itself an effort to expand the level of detail present in the traditional Boston housing dataset. The final prediction algorithm is a regularized LASSO model that incorporates linearization and cross-validation, with a root mean squared error of 0.1257 and an R^2 value of 0.9002. The final model incorporates linear, ordinal, and categorical variables based on the dataset, and incorporates 108 predictor variables out of an expanded set of 290 distinct explanatory variables. Variables were transformed as necessary to minimize skewness and multicollinearity.

Although the present work was performed as part of a capstone course for the Harvard Data Science Professional Certificate, it has the potential for real-world application with minimal rework. The advantage of the present method is that although traditional metrics of home value (historical prices, neighborhood, size, age/condition, nearby features, and market conditions) are incorporated, the application of machine learning also reveals that several factors not typically considered part of the "big six" (including landscaping, lot contour, porch square footage, basement ceiling height, and zoning, among others) exert a statistically significant influence on the sale price of a given home. Note that the present work is meant to complement and enhance the traditional valuation estimates performed by realtors, rather than to replace them: the machine learning algorithm is able to explain approximately 90% of the variation in home sale prices, with a median error of approximately 2%; although median error was not a performance metric for the present work, it is worth nothing that this median prediction error is on par with advertised median errors of software such as Redfin. Regardless, the prediction accuracy is far greater than the 20% margin of error conventionally quoted by existing web- and app-based estimating software or similar "rules of thumb." The presented algorithm therefore serves as an additional tool for a realtor's competitive market analysis when pricing a home, and which can be reconstructed for various markets as necessary.

1 Introduction

This project forms a part of the completion requirements for the HarvardX course PH125.9x, which is the capstone course for the Data Science Professional Certificate curriculum. This project is the second of two projects required for the completion of PH125.9x. The dataset used herein is freely available on Kaggle, and is summarized in the present work where appropriate.

In its simplest form, a machine learning algorithm is one that is capable of making predictions with no *a priori* knowledge of the outcome, with the aim being to process disparate data into an informative and intuitive solution. For the present case, this involves the ability to correctly predict the market value of a house given seventy-nine explanatory variables that describe a multitude of home characteristics, such as square footage, age, and roofing materials. While many of the variables are conventional — such as above-grade living area and total number of bathrooms — some (such as basement ceiling height) are not typically considered driving factors in the decision to buy a home.

The housing dataset used is presented as an expanded alternative to the Boston Housing dataset. The residual mean squared error (RMSE) is the sole metric by which the accuracy of the prediction algorithm is measured, with the squared correlation coefficient (R^2) considered a fallout result of the RMSE minimization. This report presents an overview of the dataset, exploratory data analysis, the creation of the prediction model, and the performance of the model when evaluated against the test set.

1.1 Dataset

The dataset is imported as per the instructions given in the project charter using the following code. The `cache=TRUE` modifier is used so that the code chunk compiles *once*, and subsequent uses of `knit` do not require the import code to execute. A variety of packages and their attendant libraries are installed as follows:

```r
################################################################################
# PART 1: LOAD PACKAGES/LIBRARIES AND INGEST DATA
################################################################################
# Install necessary packages, as well as some 'unnecessary' packages that are
# still useful for data display purposes. Select 'yes' if prompted.
if(!require(car)) install.packages("car")
if(!require(caret)) install.packages("caret")
if(!require(corrplot)) install.packages("corrplot")
if(!require(DataExplorer)) install.packages("DataExplorer")
if(!require(data.table)) install.packages("data.table")
if(!require(e1071)) install.packages("e1071")
if(!require(ggrepel)) install.packages("ggrepel")
if(!require(glmnet)) install.packages("glmnet")
if(!require(gridExtra)) install.packages("gridExtra")
if(!require(Hmisc)) install.packages("Hmisc")
if(!require(magrittr)) install.packages("magrittr")
if(!require(MASS)) install.packages("MASS")
if(!require(Metrics)) install.packages("Metrics")
if(!require(mice)) install.packages("mice")
if(!require(moments)) install.packages("moments")
if(!require(psych)) install.packages("psych")
if(!require(psychometric)) install.packages("psychometric")
if(!require(randomForest)) install.packages("randomForest")
if(!require(repr)) install.packages("repr")
if(!require(Rmisc)) install.packages("Rmisc")
if(!require(tidyverse)) install.packages("tidyverse")
```

```r
# Load libraries
library(car)               # Companion to Applied Regression
library(caret)             # Classification And REgression Training
library(DataExplorer)      # Handy for plotting 'NA' and missing data
library(corrplot)          # Graphically display correlation matrices
library(dplyr)             # Working with dataframe-like objects
library(e1071)             # Latent class analysis, etc.
library(ggrepel)           # Positioning non-overlapping text in figures
library(glmnet)            # For creating generalized linear model
library(gridExtra)         # Misc. functions to work with grid graphics package
library(Hmisc)             # For pretty correlation matrix plots
library(knitr)             # General-purpose programming and tablemaking
library(lattice)           # Trellis graphics
library(magrittr)          # Functions from 'Modern Applied Stats with S', 2004
library(MASS)              # Functions from 'Modern Applied Stats with S', 2004
library(Metrics)           # Evaluation metrics for supervised machine learning
library(mice)              # Diagnostic plots
library(moments)           # Cumulants, skewness, kurtosis, and related tests
library(psych)             # Really just used for graphics capabilities
library(psychometric)      # See above
library(randomForest)      # Classification and regression trees
library(repr)              # String and binary representations of objects
library(Rmisc)             # Functions for data analysis and utility operations
library(tidyverse)         # Pretty much always needed...
```

The training set and test set are given as two separate files from the Kaggle repository, `train.csv` and `test.csv`, respectively. The `file.choose()` command lets the user select the appropriate path to the necessary files regardless of where they are saved; it may be uncommented from the following code chunk as necessary to allow the files to be interactively selected. The training set features 1,460 observations separated into 81 columns, which are not included in this report in the interest of conciseness, hence why `head` and `summary` are commented out in the code chunk.

```r
# Load training and test data. The file.choose() function will allow the user to
# point to wherever the CSV files are saved.
# train_data <- file.choose()
# test_data  <- file.choose()
train_set  <- read.csv('train.csv', stringsAsFactors = FALSE)  # Load training data
test_set   <- read.csv('test.csv', stringsAsFactors = FALSE)   # Load test data
dim(train_set)
```

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

```r
# head(train_set)
# summary(train_set)
```

Sale prices ranged from a minimum of $34,900 to a maximum of $755,000, although the price data are right-skewed. Skewness was found throughout the dataset, and the methods used for treating it are detailed in Sec. 3.

Table 1.1: Summary statistics of sale prices

Minimum	1st Q	Median	Mean	3rd Q	Maximum
$34,900	$129,975	$163,000	$180,921	$214,000	$755,000

2 Exploratory Data Analysis

Prior to the creation of the prediction model, the data were analyzed to determine the behavior of certain variables with respect to sale price, house type, geography, and several other factors. While no one deciding factor determines house prices *per se*, several are likely to influence the market. Note that this exploratory analysis is not intended to cover the likely influence of *all* explanatory variables, since that is the job of the regression algorithm; rather, the exploratory analysis is precisely that: exploratory, specifically examining the likely influences on sale price based on conventional, non-algorithmic methods. For the sake of visualization, the `train_set` data frame is reconstituted as `train_set_graph`, which has an additional column, `sale_k`, added to it. This additional column includes all of the sale prices in thousands, which results in more legible graphics. This data frame is not used elsewhere in the model. Note that not all eighty characteristics are examined; rather, only the variables most likely to influence sale price are considered here. The effects of the others are calculated in the regression model.

2.1 General Overview

A histogram of sale prices is presented in Fig. 2.1, which shows the skewness in sale price first mentioned in Table 1.1. The strong right-skew is inherently logical, as there will be fewer customers in the market for more expensive homes; however, it must be noted that there are several distinct types of homes within the dataset: sixteen different home types, to be exact. The home classification codes are given in Table 2.1 as per the data dictionary[1].

These home groupings are categorical variables, which will have a marked impact on the home's sale price. A more informative view of the sale prices would be boxplots and violin plots of the sale prices segregated by home type. While boxplots show a statistically significant difference in mean and interquartile sale prices between the groups of home types, the actual distributions by home type are also different, as shown by the violin plots. The majority of home sale prices was found to be under \$400,000 regardless of home type. The histogram and violin plots highlight an important limitation of the sale prices given in the dataset. The linear model constructed for the present work, like all linear regression models, rest upon the assumption of normality. As such, high skew and/or high kurtosis on the part of the predictors and response variables is not acceptable. As shown in Figs. 2.4 and 2.5, the home sale prices are highly skewed and are thus unusable for a linear model without transformation. log-transform was used to normalize the sale prices, as shown in Fig. 2.6.

[1]`data_description.txt`

Table 2.1: Home types by type code.

Code	Type
20	Single-story, 1946 & newer
30	Single-story, 1945 & older
40	Single-story with finished attic, all ages
45	1.5-story, unfinished, all ages
50	1.5-story, finished, all ages
60	Two-story, 1946 & newer
70	Two-story, 1945 & older
75	2.5-story, all ages
80	Split- or multilevel
85	Split foyer
90	Duplex, all styles and ages
120	Single-story planned unit development, 1946 & newer
150	1.5-story planned unit development, all ages
160	Two-story planned unit development, 1946 & newer
180	Multilevel planned unit development, including split-level and split-foyer
190	Two-family conversion, all styles and ages

```r
train_set_graph %>% ggplot(aes(sale_k, na.rm = TRUE)) +
     geom_histogram(fill='black', binwidth = 10, na.rm = TRUE) +
     xlab('Sale price, $k') + ylab('Count') +
     theme_bw()
```

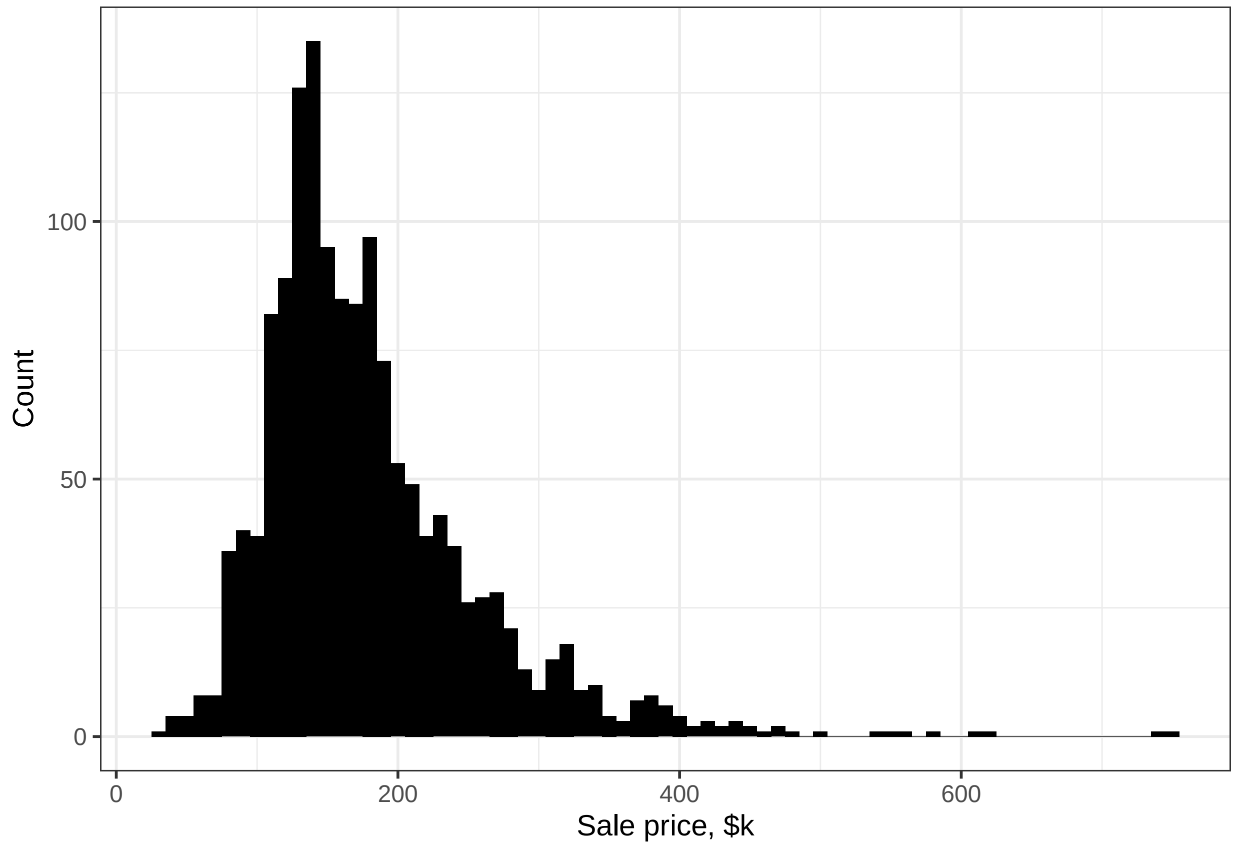

Figure 2.1: Histogram of sale prices.

```r
groups <- train_set_graph %>% group_by(MSSubClass)
groups <- groups$MSSubClass
train_set_graph %>% ggplot(aes(x=factor(groups), y=sale_k)) + geom_boxplot() +
     xlab('House type') + ylab('Sale price, $k') + theme_bw()
```

```r
groups <- train_set_graph %>% group_by(MSSubClass)
groups <- groups$MSSubClass
train_set_graph %>% ggplot(aes(x=factor(groups), y=sale_k)) + geom_violin() +
     scale_y_log10() +
     stat_summary(fun='mean', geom='point', shape=15, size=3, color='blue') +
     stat_summary(fun='median', geom='point', shape=16, size=3, color='red') +
     xlab('House type') + ylab('Sale price, $k') + theme_bw()
```

```r
sale_prices <- rbind(data.frame(version='log(SalePrice+1)',
                x = log(train_set$SalePrice + 1)),
                data.frame(version='Price',x = train_set$SalePrice))

ggplot(data=sale_prices) + facet_wrap(~version, ncol = 2,
                scales = 'free_x') + geom_histogram(aes(x=x), fill = 'black') +
```

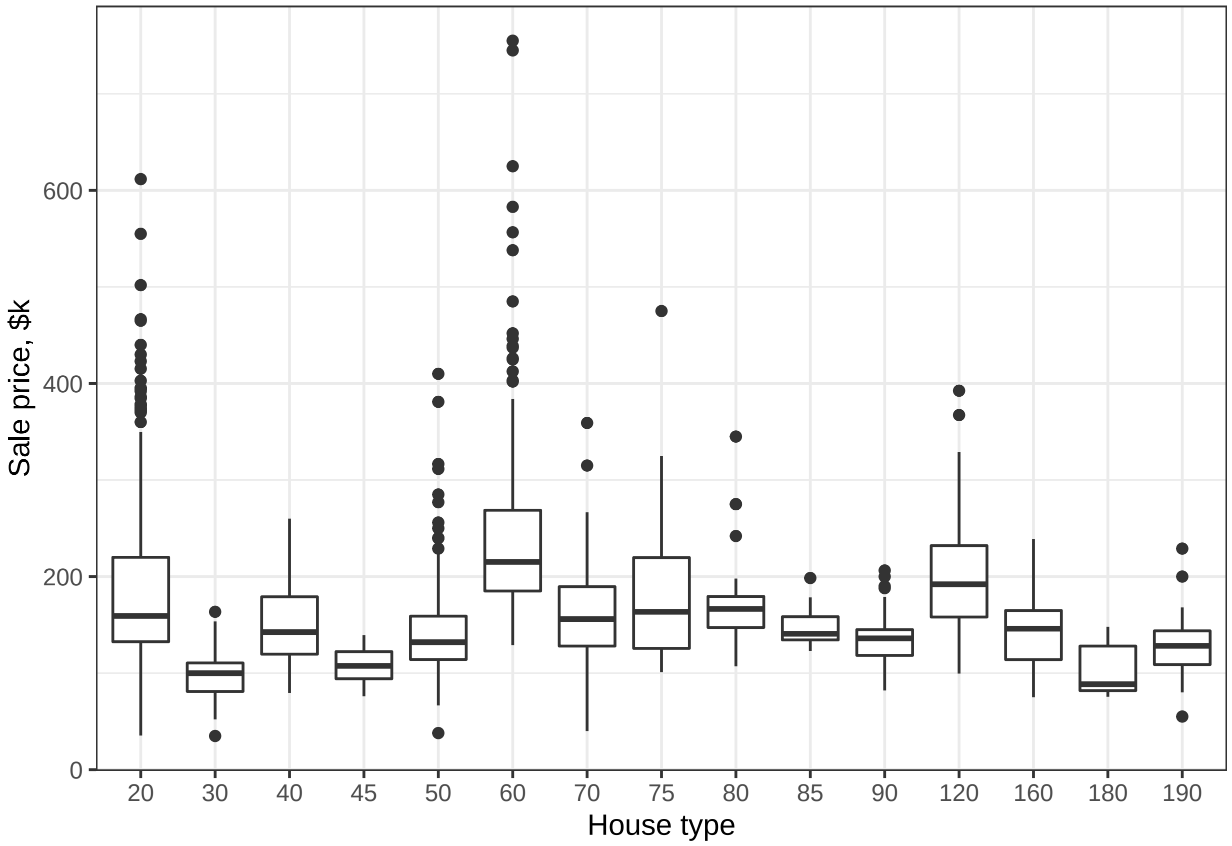

Figure 2.2: Boxplot of home sale prices by house type.

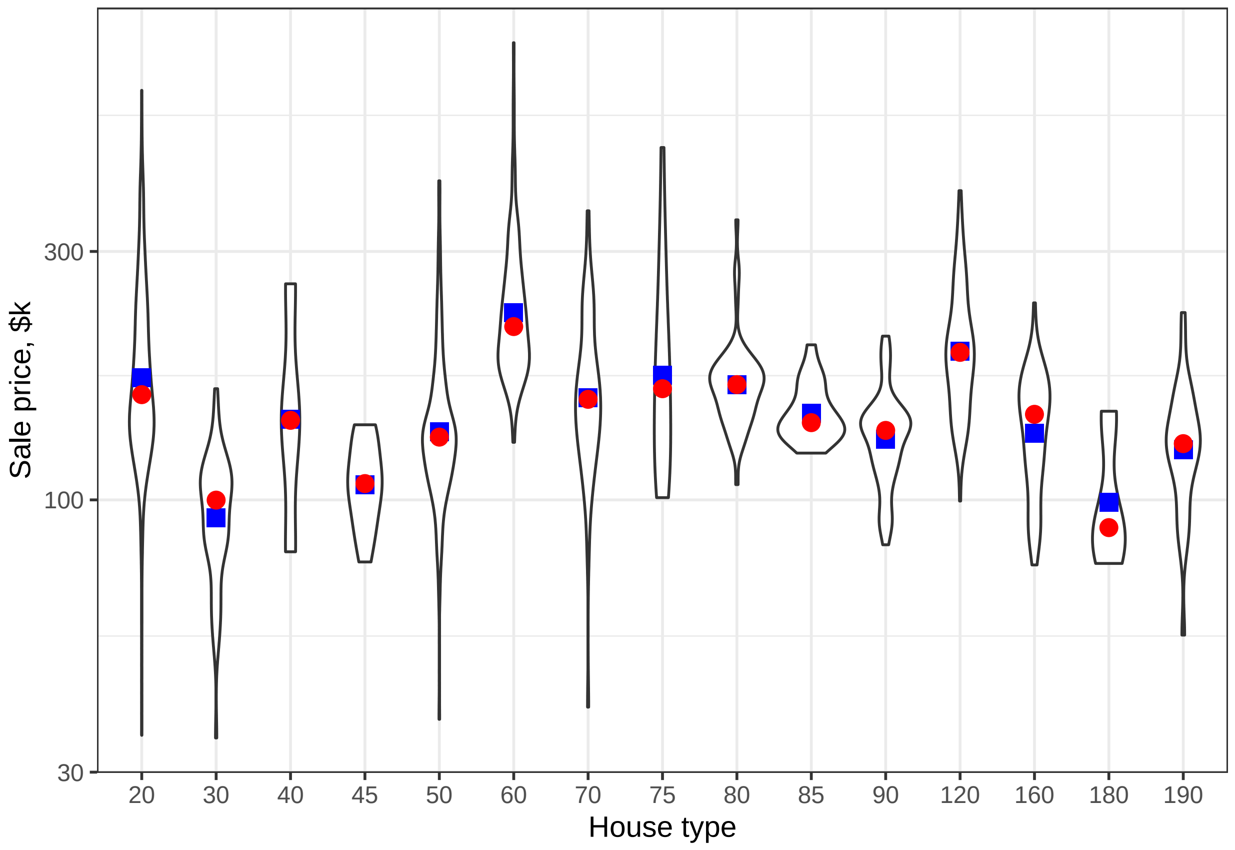

Figure 2.3: Violin plot of home sale prices by house type. Blue = mean, red = median.

```
        xlab('log(SalePrice), left; sale price, right') + ylab('Count') +
        theme_bw()
```

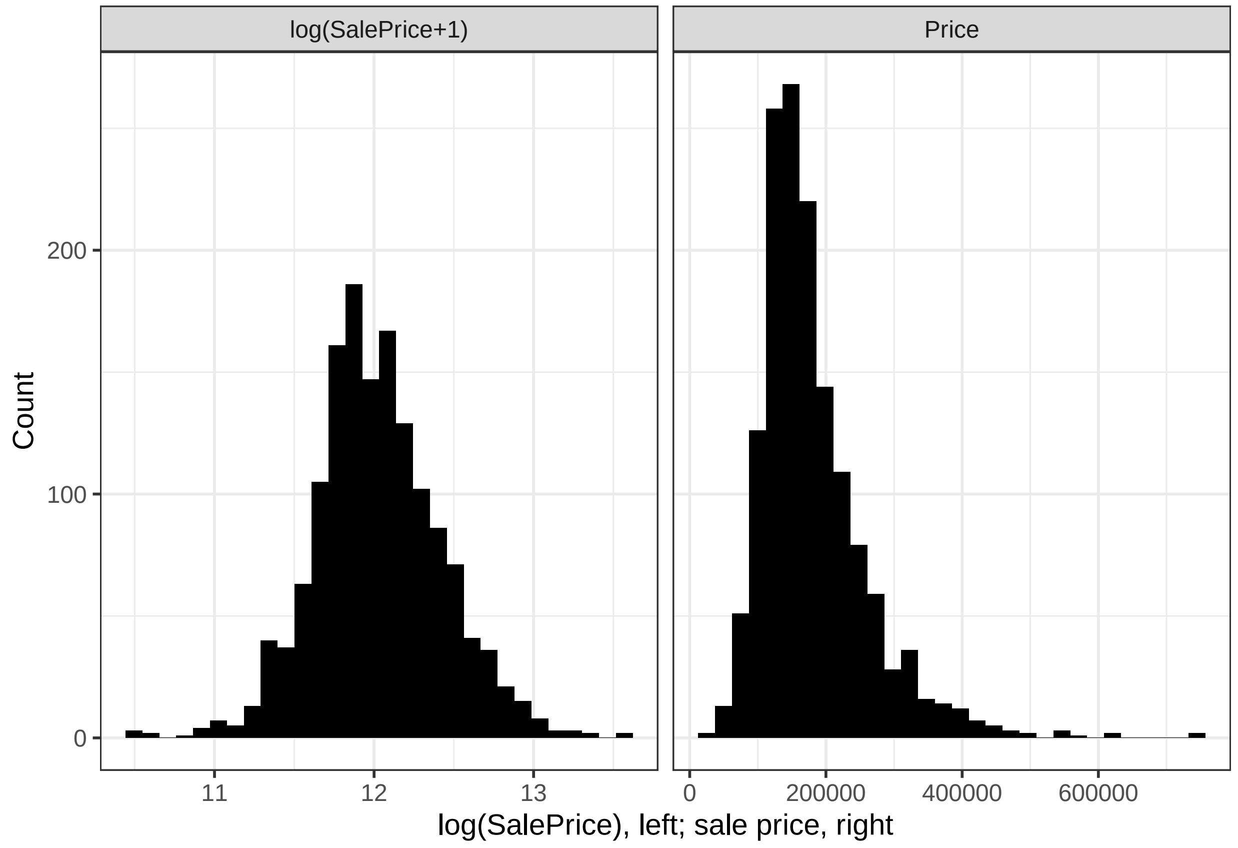

Figure 2.4: Histograms of sale prices and log-transform of sale prices.

```
sale_price_qq <- as.data.frame(train_set$SalePrice)
sale_price_qq %>% ggplot(aes(sample = train_set$SalePrice)) +
        stat_qq() + stat_qq_line() +
        xlab('Theoretical quantiles') + ylab('Sale price') +
        theme_bw()
```

```
sale_price_qq <- mutate(sale_price_qq, logSalePrice = log(train_set$SalePrice))
sale_price_qq %>% ggplot(aes(sample = logSalePrice)) +
        stat_qq() + stat_qq_line() +
        xlab('Theoretical quantiles') + ylab('Sale price') +
        theme_bw()
```

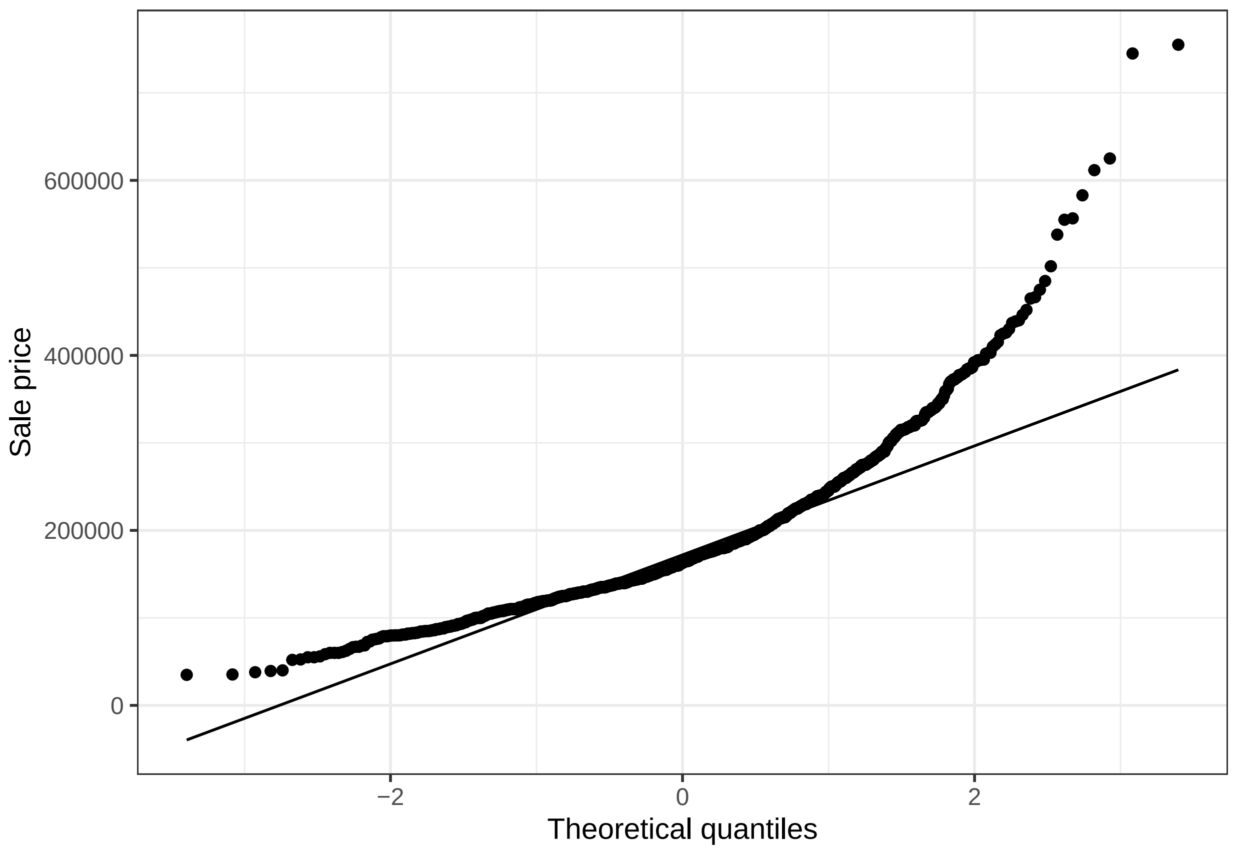

Figure 2.5: Q-Q plot of home sale prices.

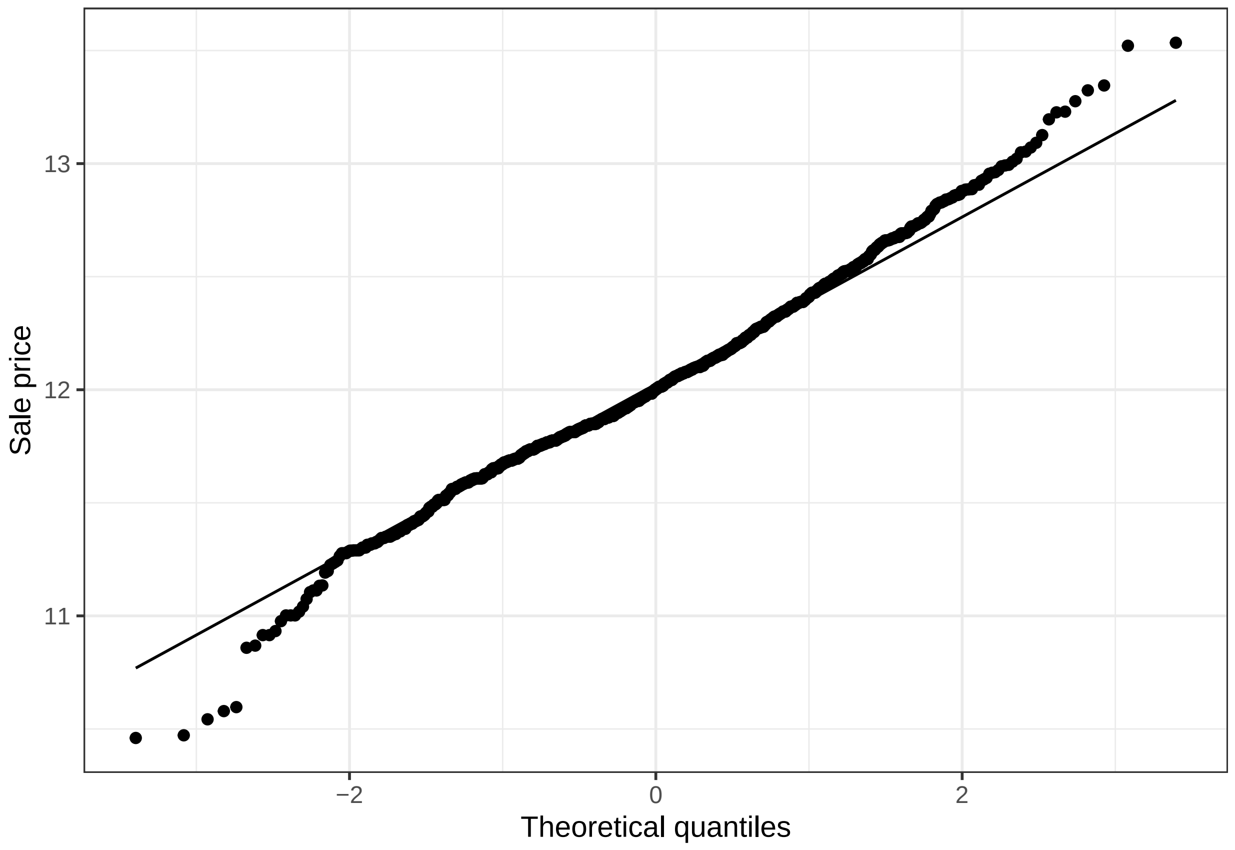

Figure 2.6: Q-Q plot of log-transform home sale prices.

2.2 Effects on Sale Price by Lot Factors

Of course, the house type is not the only potential variable for a home's price. Lot area (in ft.^{2}) and lot frontage (in ft.) are also important factors; however, for the present dataset, lot area apparently has a major effect regardless of overall home quality, as evidenced by the near-vertical distribution of points in Fig. 2.7 (viz. large change in sale price for a small change in lot area). Lot frontage (Fig. 2.8) appears to have a slightly less pronounced effect. Lot area and lot frontage were found to vary greatly by house type, as shown in Figs. 2.9 - 2.12. In particular, all categories of planned developments featured significantly smaller lot sizes and frontages than any other type of home, and the distributions of lot areas and frontages for these planned developments were heavily right-skewed, indicating a preponderance of smaller lots with minimal frontage. This is not surprising — and, indeed, should not scare off potential home buyers — as planned developments will often trade lot size and frontage for shared amenities such as playgrounds, tennis and basketball courts, pools, trails, open spaces, and other assets[2] that may not otherwise be available to the prospective homeowner.

Lot area and frontage are not the only considerations to keep in mind. Oftentimes, a home's value can be influenced (unfairly or not) by the functionality of the lot with respect to the prospective buyer's intended use. For instance, oddly-shaped and hilly lots may adversely affect the ability of the homeowner to install play equipment for their children; similarly, lot contour will impact future landscaping decisions based on factors such as available sunlight, drainage, and maintainability. While the present work is incapable of predicting individual home buyer preferences, if there is inherent bias within the dataset for certain lot configurations or contours these factors should have a measurable effect on sale prices.

As shown in Fig. 2.13, lot area correlates strongly with sale price, but actual lot configuration seems to matter less so. This is evidenced by the tight grouping of points with little apparent variation in slope by configuration type. Aside from some outliers of cul-de-sac and inside lots, it is readily apparent that larger lot areas correspond with higher sale prices regardless of how those lots are configured.

The definitions of land slopes and contours, as specified by the data dictionary, are given in Table 2.2. While slope and contour appear to be interchangeable terms, they are not: contour refers to the general shape of the lot, whereas slope refers to the rate of those shapes. These variables can (and often are) considered in combinations: for instance, a lot with a hillside contour can have a gentle slope and may affect the sale price positively, whereas a depressed (sunken) entrance with a severe slope can give a property a "basement entrance" feel and lower the sale price.

Land slope appears to have very little, if any, effect on sale price, as shown in Figs. 2.14 - 2.17. The amount of overlap in the box plots suggests that there is likely no statistically significant difference in sale price based on land slope or contour. Interestingly, there were more sales in the sub-$30,000 range for level plots (Fig. 2.17) than for any other type, just as there were more low-priced sales for gently-sloping lots (Fig. 2.15) than any other.

[2]https://www.zillow.com/mortgage-learning/what_is_pud/

Table 2.2: Definition of land slope and contour.

Code	Type
Lvl	Near-level
Bnk	Banked: quick & significant rise from street grade to building
HLS	Hillside: significant lateral slope to property
Low	Depression
Gtl	Gentle slope
Mod	Moderate slope
Sev	Severe slope

```r
train_set_graph %>% ggplot(aes(LotArea, sale_k, color = OverallQual)) +
     geom_point(alpha = 0.5) +
     xlab('Lot area, sq.ft.') + ylab('Sale price, $k') + theme_bw()
```

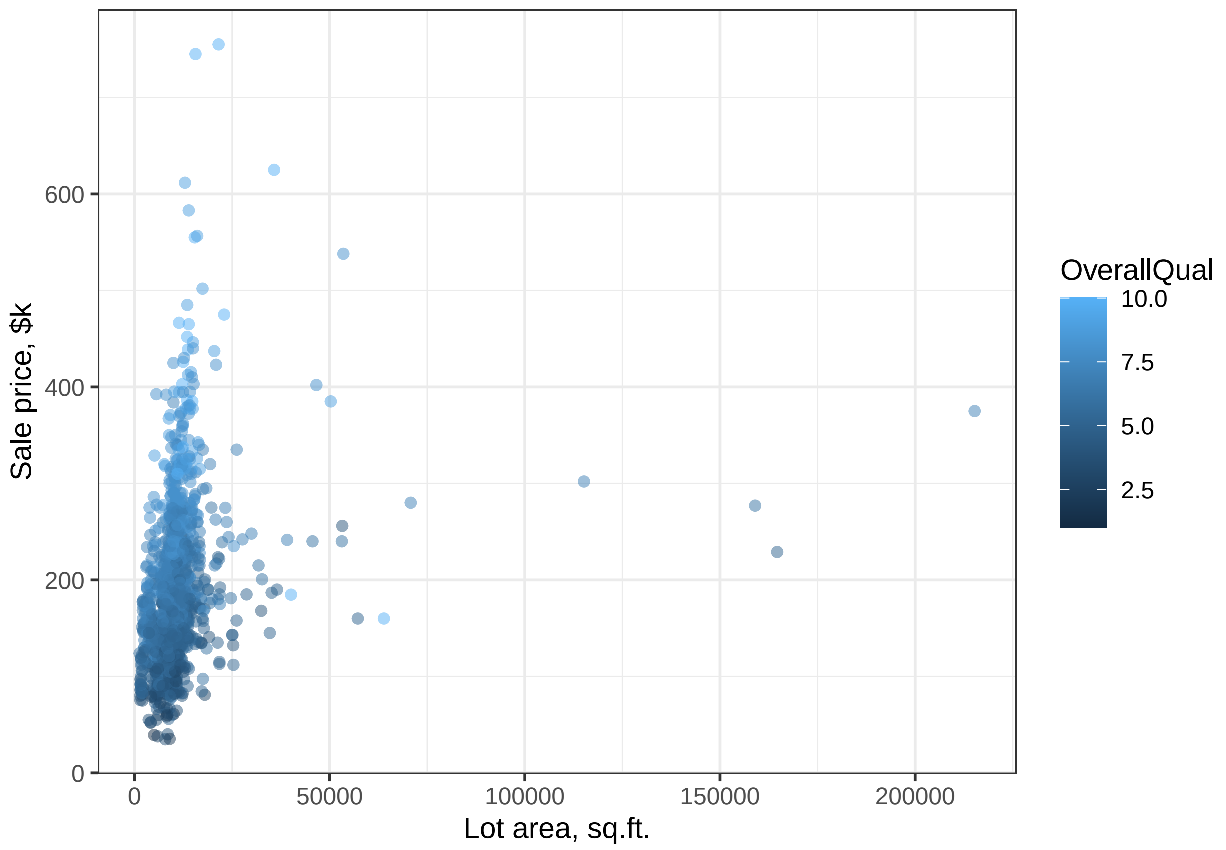

Figure 2.7: Scatter plot of home sale prices vs. lot area by overall quality.

```r
train_set_graph %>% ggplot(aes(LotFrontage, sale_k, color = OverallQual)) +
     geom_point(alpha = 0.5) +
     xlab('Lot frontage, ft.') + ylab('Lot frontage, ft.') + theme_bw()
```

```r
train_set_graph %>% ggplot(aes(x=factor(groups), y=LotArea)) + geom_boxplot() +
     scale_y_log10() +
     xlab('House type') + ylab('Lot area, sq.ft.') + theme_bw()
```

```r
train_set_graph %>% ggplot(aes(x=factor(groups), y=LotArea)) + geom_violin() +
     scale_y_log10() +
     stat_summary(fun='mean', geom='point', shape=15, size=3, color='blue') +
     stat_summary(fun='median', geom='point', shape=16, size=3, color='red') +
     xlab('House type') + ylab('Lot area, sq.ft.') + theme_bw()
```

```r
train_set_graph %>% ggplot(aes(x=factor(groups), y=LotFrontage)) + geom_boxplot() +
     scale_y_log10() +
     xlab('House type') + ylab('Lot frontage, ft.') + theme_bw()
```

```r
train_set_graph %>% ggplot(aes(x=factor(groups), y=LotFrontage)) + geom_violin() +
     scale_y_log10() +
     stat_summary(fun='mean', geom='point', shape=15, size=3, color='blue') +
```

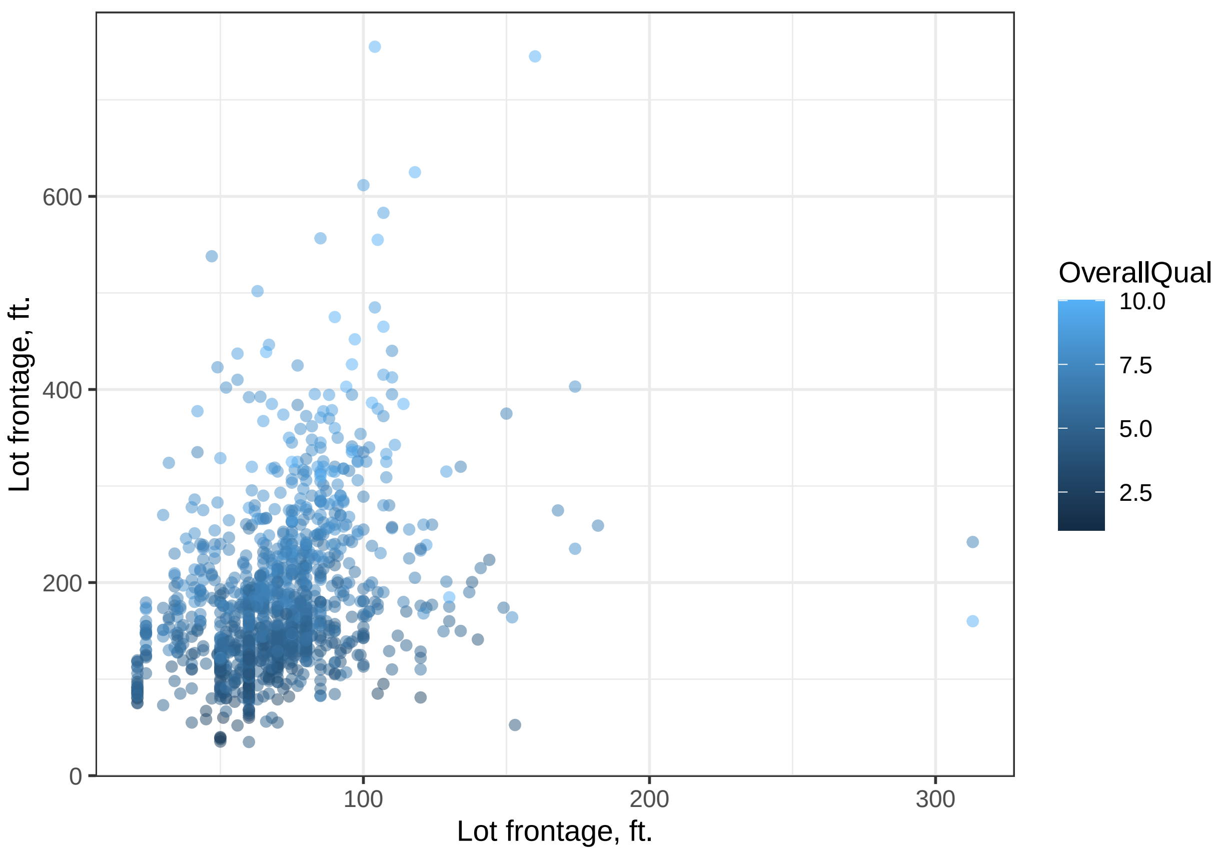

Figure 2.8: Scatter plot of home sale prices vs. lot area by overall quality.

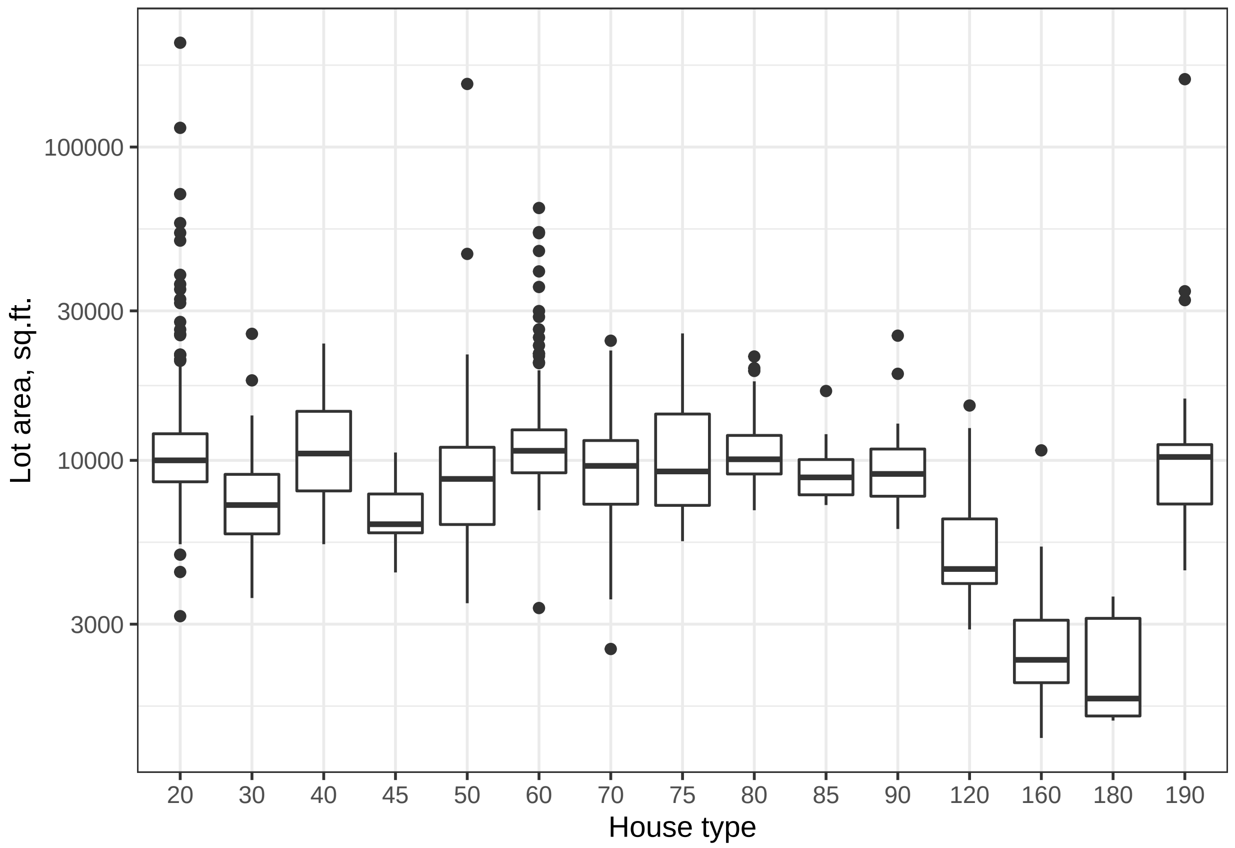

Figure 2.9: Boxplot of lot area by house type.

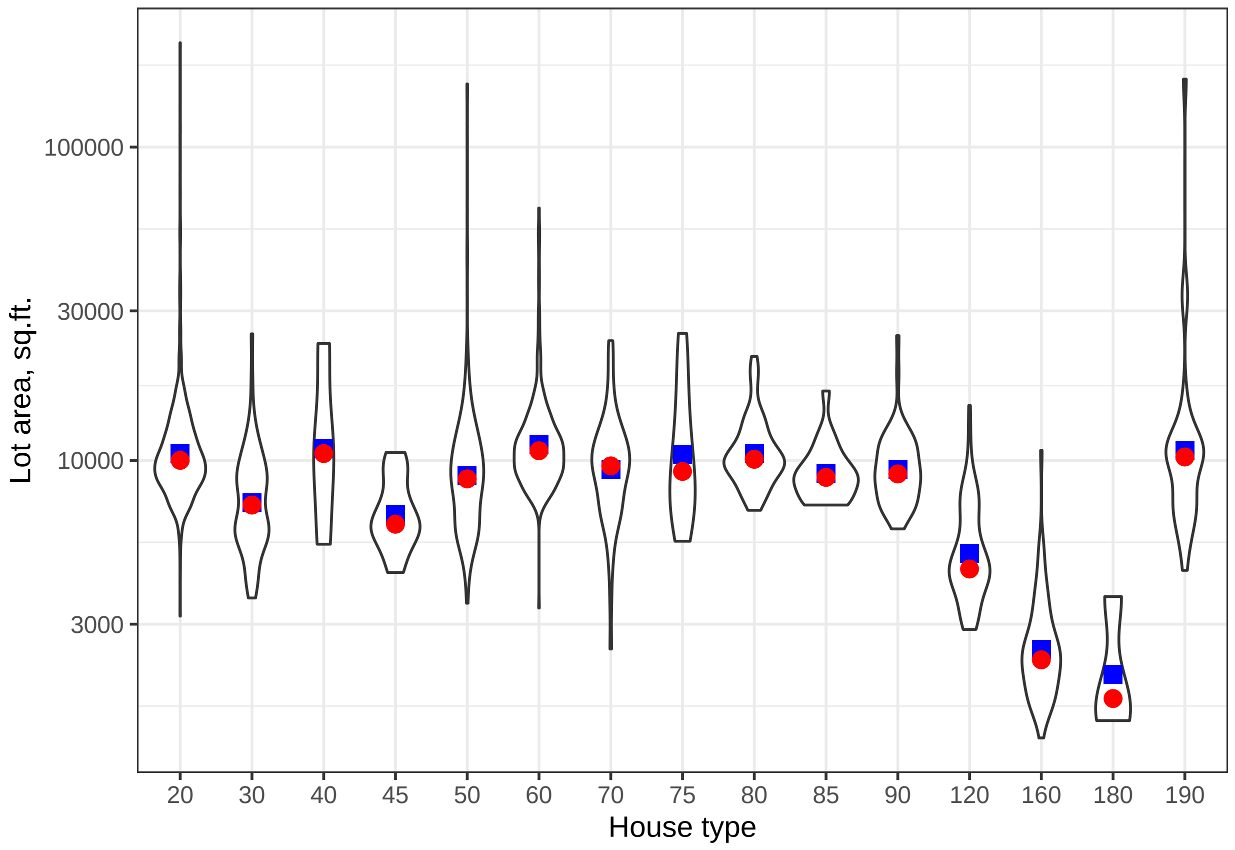

Figure 2.10: Violin plot of lot area by house type. Blue = mean, red = median.

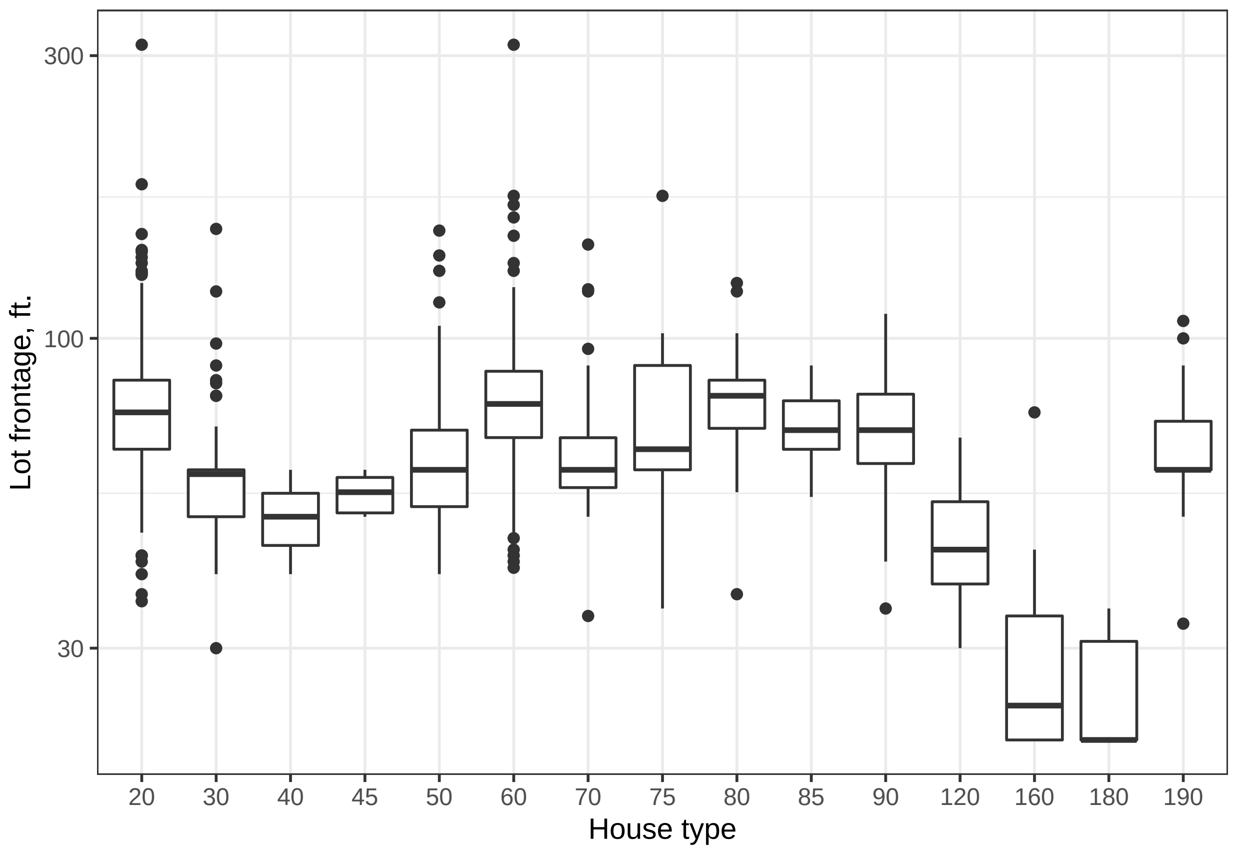

Figure 2.11: Boxplot of lot frontage by house type.

```
    stat_summary(fun='median', geom='point', shape=16, size=3, color='red') +
    xlab('House type') + ylab('Lot frontage, ft.') + theme_bw()
```

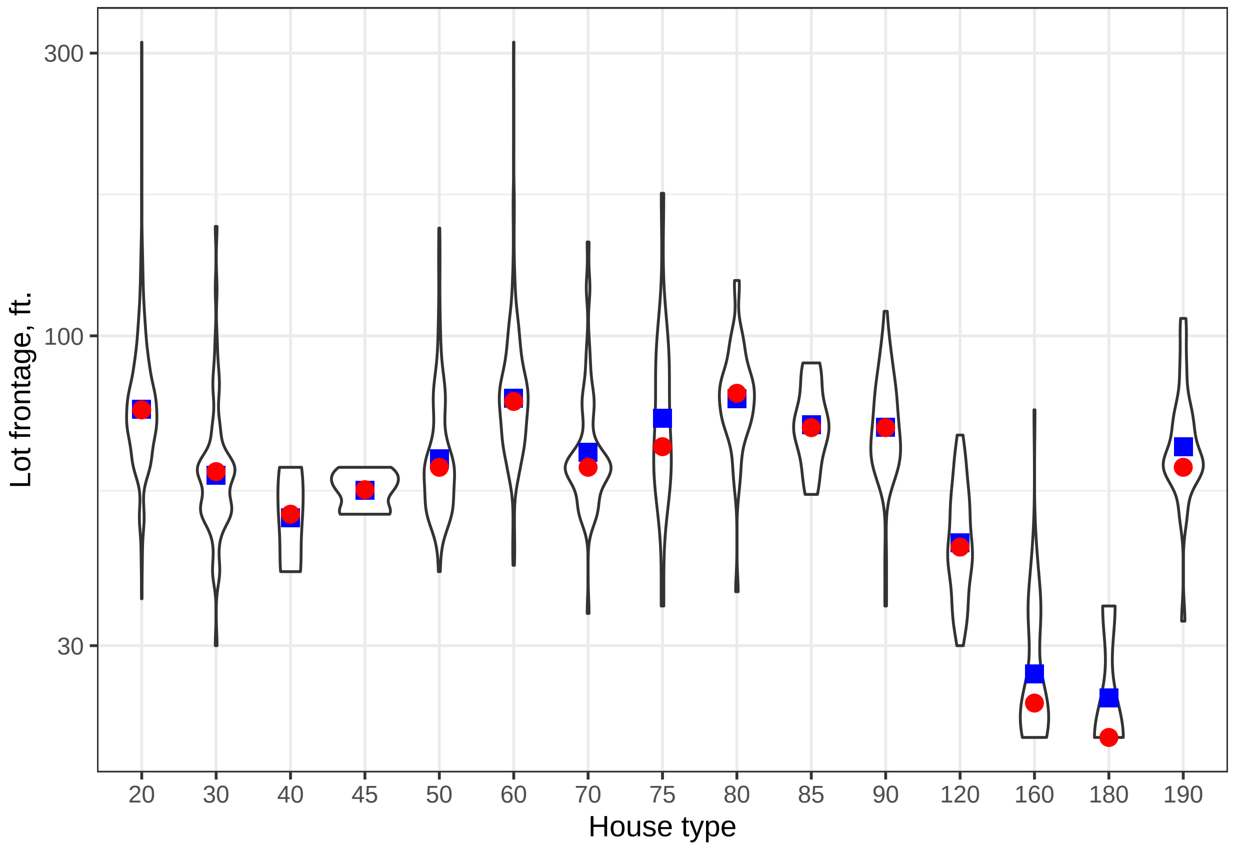

Figure 2.12: Violin plot of lot frontage by house type. Blue = mean, red = median.

```
train_set_graph %>% ggplot(aes(LotArea, sale_k, color = LotConfig)) + geom_point(alpha = 0.5) +
    xlab('Lot area, sq.ft.') + ylab('Sale price, $k') + theme_bw()
```

```
train_set_graph %>% ggplot(aes(x=factor(LandSlope), y=sale_k)) + geom_boxplot() +
    scale_y_log10() +
    xlab('Land slope') + ylab('Sale price, $k') + theme_bw()
```

```
train_set_graph %>% ggplot(aes(x=factor(LandSlope), y=LotFrontage)) + geom_violin() +
    scale_y_log10() +
    stat_summary(fun='mean', geom='point', shape=15, size=3, color='blue') +
    stat_summary(fun='median', geom='point', shape=16, size=3, color='red') +
    xlab('Land slope') + ylab('Sale price, $k') +theme_bw()
```

```
# Sell price vs land contour
train_set_graph %>% ggplot(aes(x=factor(LandContour), y=sale_k)) + geom_boxplot() +
    scale_y_log10() +
    xlab('Land contour') + ylab('Sale price, $k') + theme_bw()
```

```
train_set_graph %>% ggplot(aes(x=factor(LandContour), y=LotFrontage)) + geom_violin() +
    scale_y_log10() +
    stat_summary(fun='mean', geom='point', shape=15, size=3, color='blue') +
    stat_summary(fun='median', geom='point', shape=16, size=3, color='red') +
```

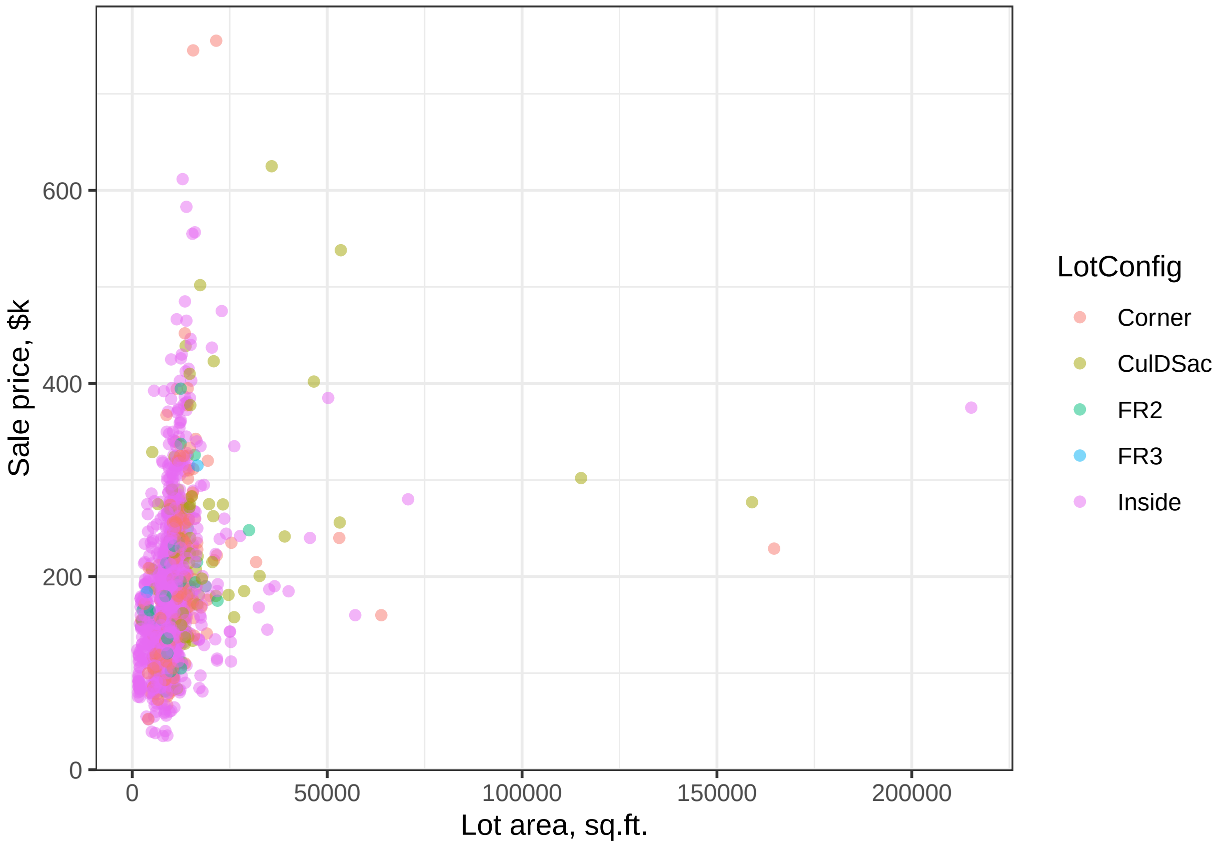

Figure 2.13: Sale price versus lot configuration.

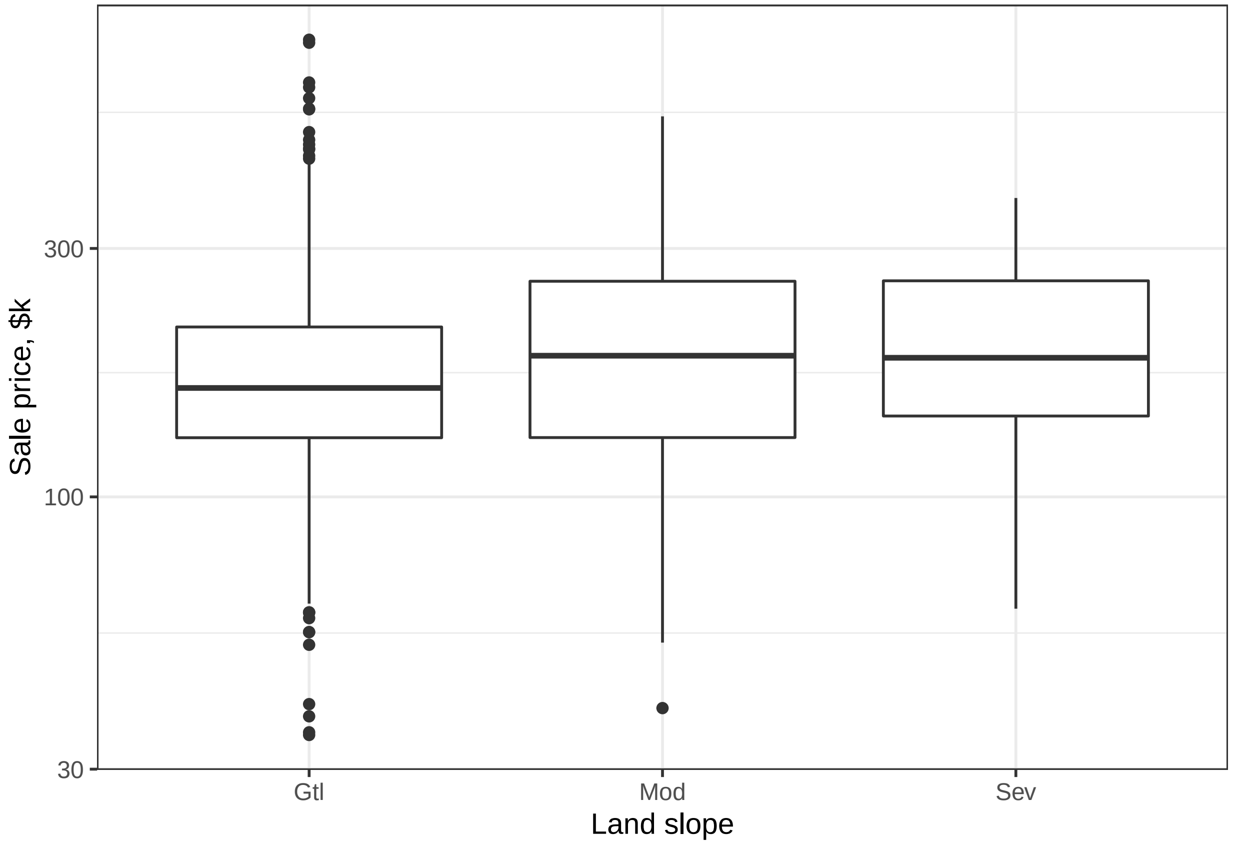

Figure 2.14: Boxplot of sale price vs. land slope.

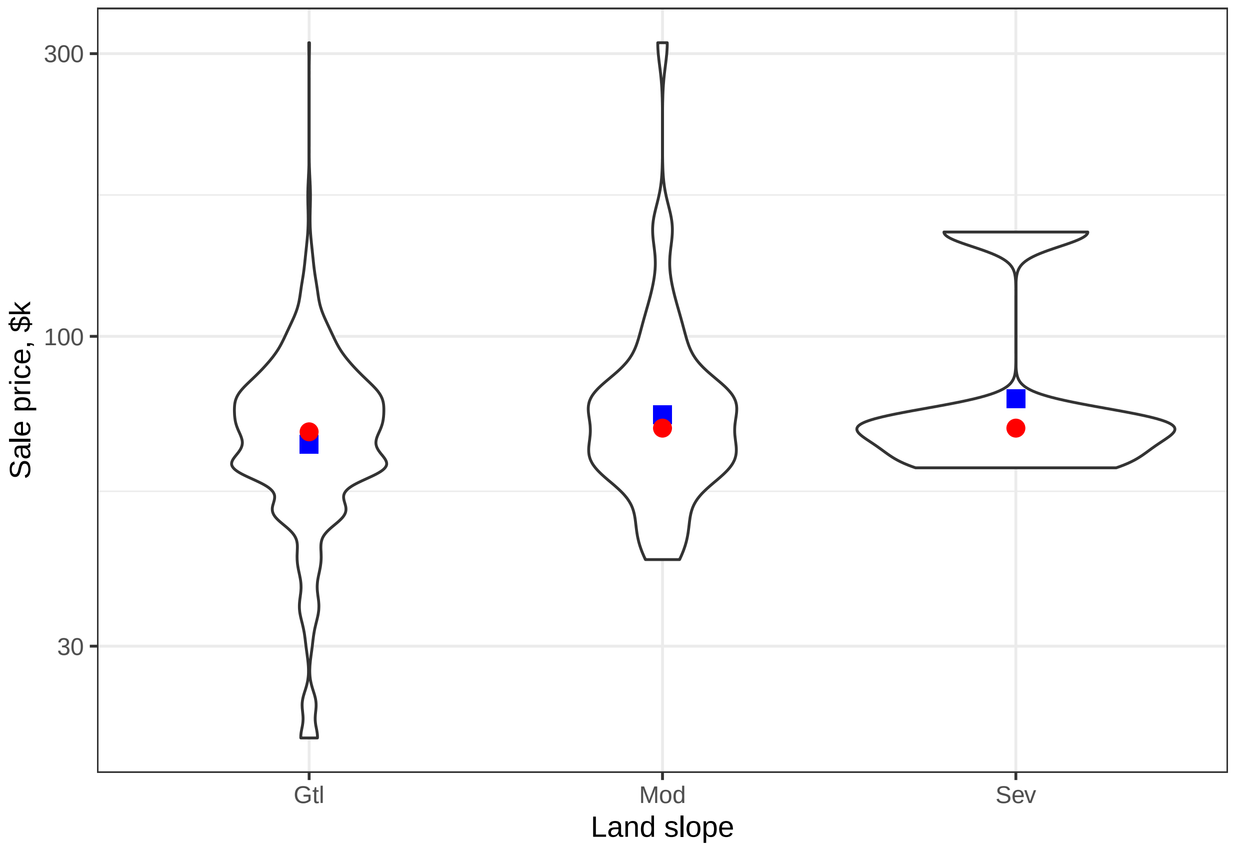

Figure 2.15: Violin plot of sale price vs. land slope. Blue = mean, red = median.

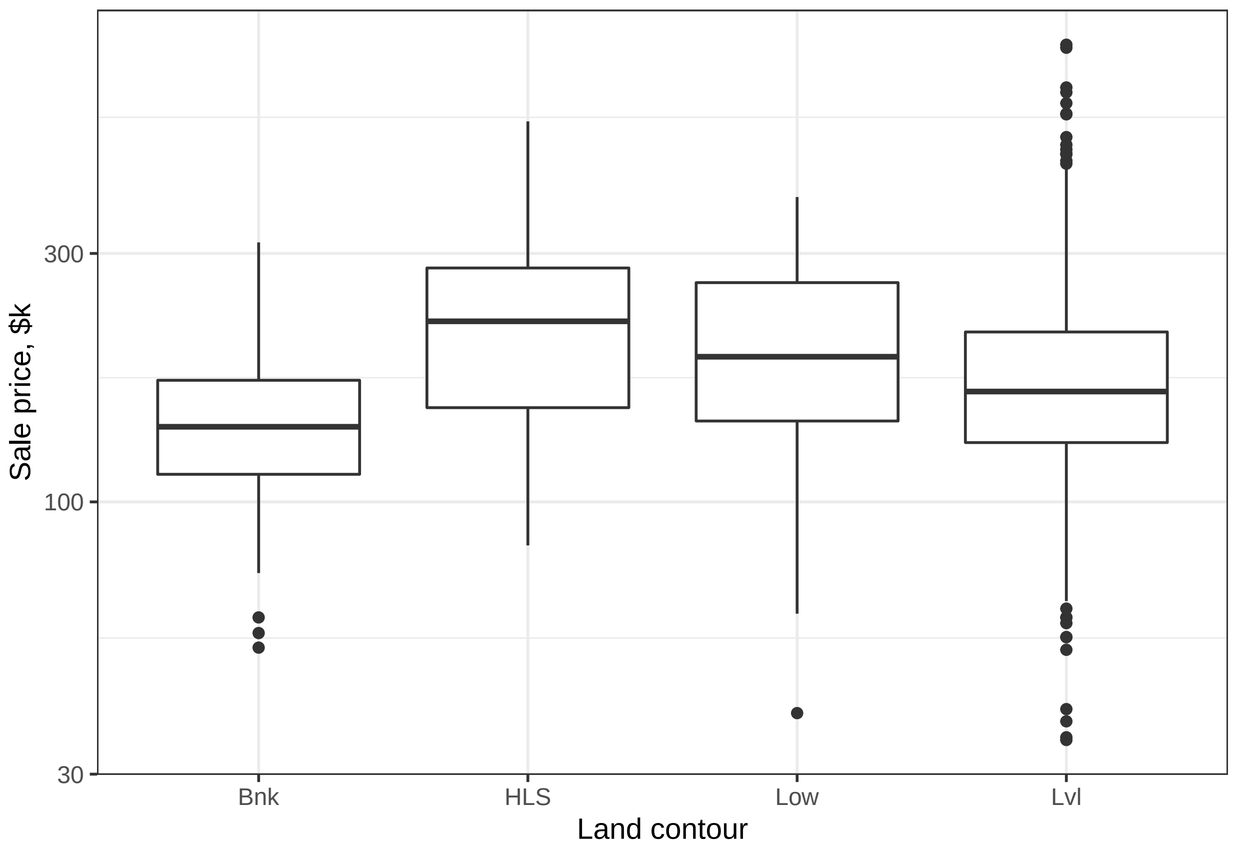

Figure 2.16: Boxplot of sale price vs. land contour.

```
    xlab('Land contour') + ylab('Sale price, $k') +theme_bw()
```

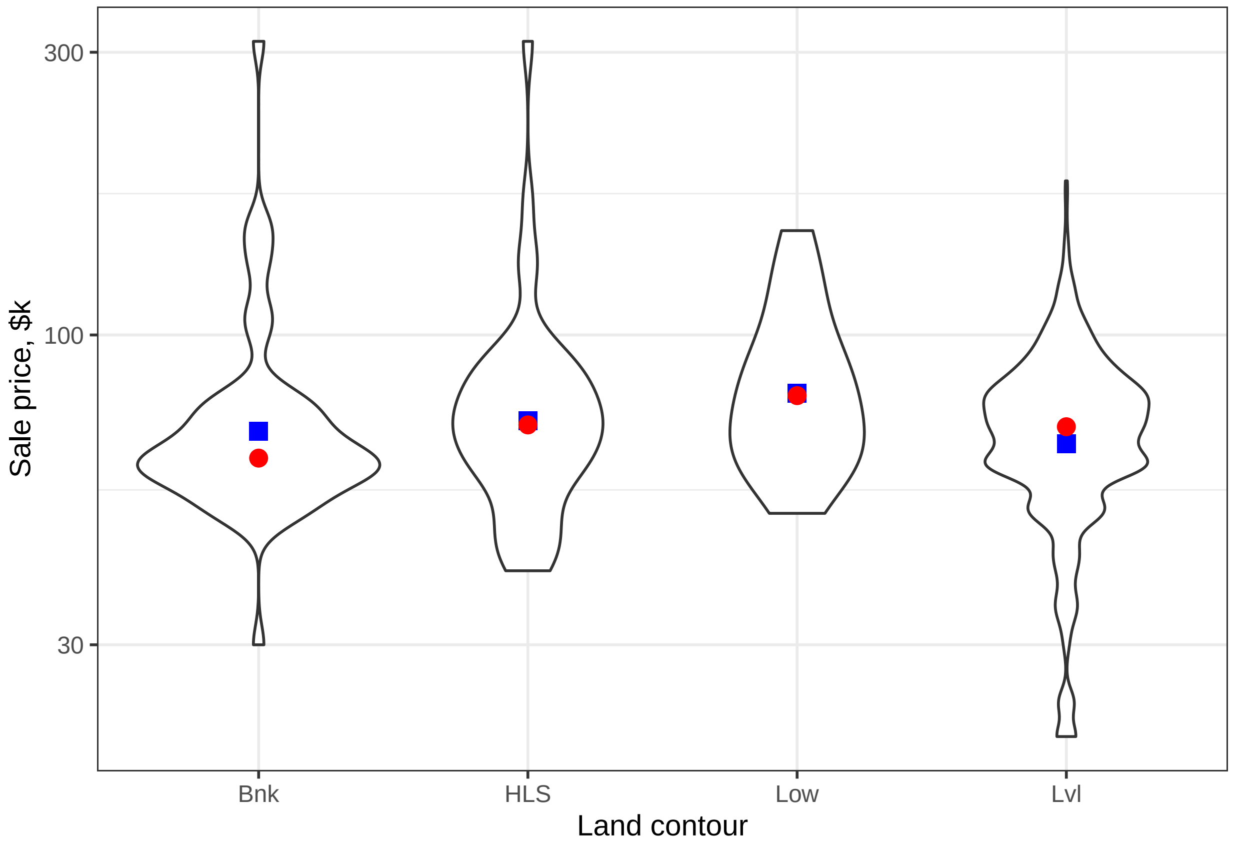

Figure 2.17: Violin plot of sale price vs. land contour. Blue = mean, red = median.

2.3 Effects on Sale Price by House Age

Unsurprisingly, there is a clear and direct correlation between the age of a home's construction and the sale price, as seen in Figs. 2.18 and 2.19. Whether one considers the age of the original construction of the home or the age of the remodel, it is apparent that "newer" will command a premium. The data dictionary does not give any specifics as to the amenities within the homes aside from the basics (e.g. bedrooms, bathrooms, floor area), but it can be safely assumed that newer constructions (original or remodel) will likely incorporate features that are more desirable at the time of purchase, such as varying counter top material, flooring material, or type of floor plan. In particular, there is a clear and direct correlation between sale price and age of remodel construction based on floor space, with newer remodeling efforts commanding a much higher price per square foot (as evidenced by the slope of the points) than older ones. While the extent and type of remodeling is indeterminable from the given dataset, its correlation with sale price makes it apparent that the remodeling work was successful in terms of meeting customer demand within the market.

```
train_set_graph %>% ggplot(aes(YearBuilt, sale_k)) +
     geom_point(alpha=0.5) +
     xlab('Year built') + ylab('Sale price, $k') + theme_bw()
```

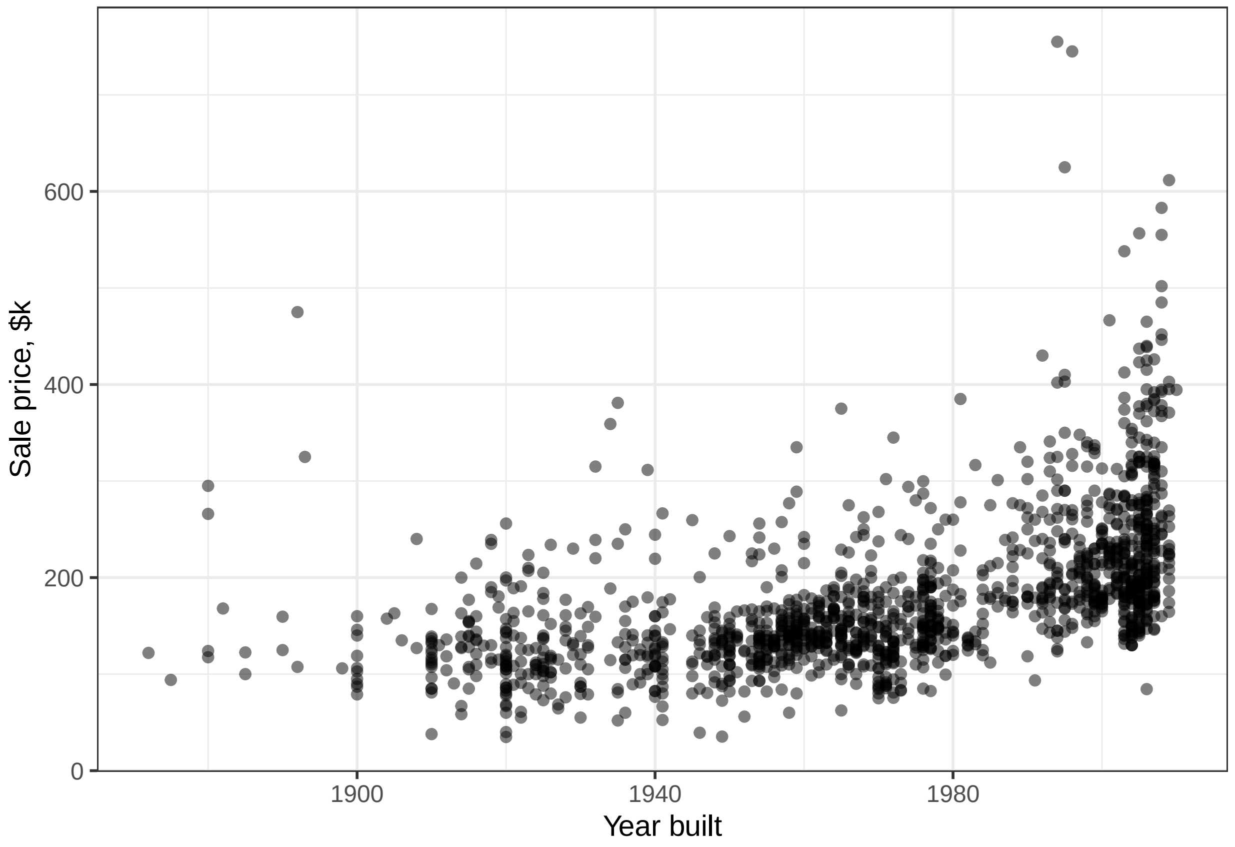

Figure 2.18: Sale price vs. year built.

```
train_set_graph %>% ggplot(aes(GrLivArea, sale_k, color = YearRemodAdd)) +
     geom_point(alpha = 0.5) +
     xlab('Above-grade living area, sq.ft.') + ylab('Sale price, $k') + theme_bw()
```

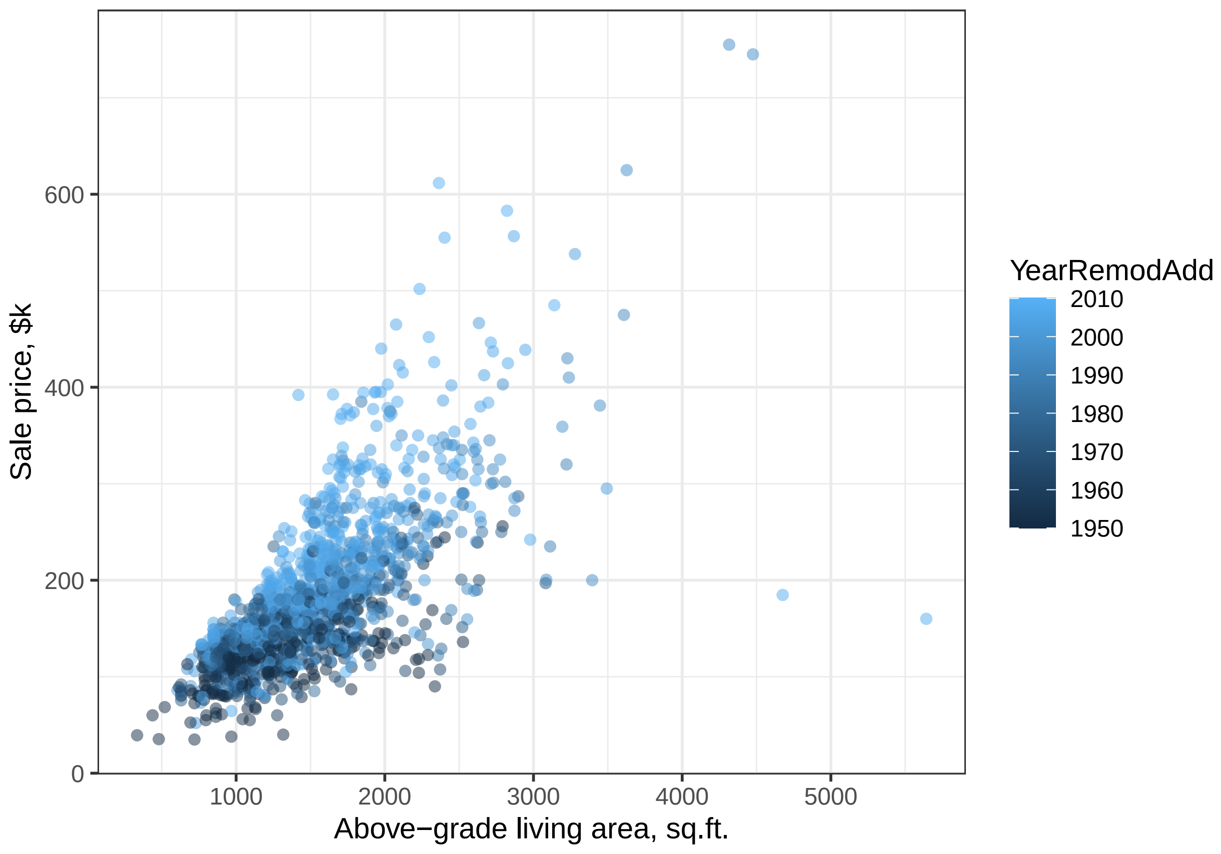

Figure 2.19: Sale price vs. year of above-grade living area remodel.

2.4 Effects on Sale Price by House Characteristics

It is intuitive that the inherent characteristics of a home will directly influence its sale price. More living space, for instance, is usually more desirable (and able to command a higher price) than less living space; of course, this varies with time and cultural taste (as evidenced by the tiny-house fad of recent years), so it cannot be assumed universal. Generally speaking, however, "more is more," and is priced as such. The dataset subdivides homes by dwelling type and dwelling style, which are defined in Tables 2.3 and 2.4.

The effects of these variables on sale price is seen in Figs. 2.20 and 2.21. While a strong positive correlation between sale price and living area was previously established with the age of construction or remodeling effort, there is a clear difference between sale price and above-grade (viz. above-ground) living area by house type and style. Simply put, single-family homes and townhouse end units command a steeper price per square foot of above-grade living area (more vertical slope) than any other type of dwelling; additionally, single-story homes command a higher dollar amount per square foot of above-grade living area than any other house style, although it is closely matched by the dollar/square foot rate of the two-story homes. This is likely easily explained by the fact that these two house styles are the most common, whereas split-levels, split-foyers, and half-story homes may have inherent characteristics that do not appeal to buyers; for instance, a 1.5-story home will usually feature vaulted ceilings and lofts in an attempt to use space efficiently, but the end result is a home that's almost as tall as a two-story (and therefore of comparable cost to heat and cool) but lacks the full functionality and living space afforded by a complete second floor.

Perhaps the most immediate characteristics of a house noticed by potential buyers is quality and condition, both of which are given in the dataset as ordinal variables, with a 1 indicating "very poor" quality or condition and a 10 indicating "very excellent" quality and condition. Overall quality, in this case, refers to the physical materials and finish of the house: granite counter tops, polished hardwood floors, and glass tile are examples of items that would influence the quality variable. Overall condition, on the other hand, refers to the structural and functional characteristics of a house: paint chipping, stained carpeting, misaligned doors, and similar characteristics are the types of items considered when one thinks of a home's overall condition. As seen in Figs. 2.23 - 2.25, there is a clear and statistically significant correlation between the overall quality and condition of a house and its sale price. This should not be surprising: quality matters, and homeowners are likely to be willing to pay a premium to acquire a home with high-quality construction materials (just as home builders can command a higher price for using said materials) and a higher price for a home in better overall condition. The former may be considered a matter of taste, whereas the latter can imply that the prospective home buyer is risk-averse and unwilling to purchase a potentially hazardous structure: as seen in Figs. 2.24 and 2.25, the spread of sale prices for houses with "very poor" (condition rating of 1) or "poor" (condition rating of 2) overall condition is very tight, indicating that these homes are "take it or leave it" affairs with very low demand.

Roof style and material are also important factors to consider when buying a home. While the dataset includes several roof styles and roofing materials, it does not include a separate variable for roof condition. This might be because this is factored in to the overall house condition, although how the items that comprise the Overall Condition value are weighted is unknown. Roof styles are given relatively straightforward descriptors (flat, gable, gambrel, hip, Mansard, or shed), the roofing materials are defined as shown in Table 2.5. The effect of roof style and material on sale price is shown in Figs. 2.26 and 2.27.

The correlation between roof style and sale price does not appear very significant at first glance, although of course the prediction model will calculate the regression coefficients to verify. The predominant types of roofing materials appears to be composite shingles, tar & gravel, and wood shingles, with the remainder of the roofing materials barely registering. In fact, despite the spread in sale prices suggested by Fig. 2.29, composite shingle roofs were overwhelmingly dominant in the dataset, accounting for over 98% of the entries. It is therefore unlikely that either roof type or roof material will significantly contribute to the prediction model after regularization and cross-validation have been performed.

Table 2.3: Types of dwelling

Code	Type
1Fam	Single-family detached
2FmCon	Two-family conversion; originally built as single-family
Duplx	Duplex
TwnhsE	Townhouse end unit
TwnhsI	Townhouse inside unit

Table 2.4: Style of dwelling

Code	Type
1Story	Single-story
1.5Fin	1.5-story; 2nd level finished
1.5Unf	1.5-story; 2nd level unfinished
2Story	Two-story
2.5Fin	2.5-story; 2nd level finished
2.5Unf	2.5-story; 2nd level unfinished
SFoyer	Split foyer
SLvl	Split-level

Table 2.5: Roofing materials.

Code	Type
ClyTile	Clay or tile
CompShg	Standard composite shingle
Membran	Membrane
Metal	Metal
Roll	Roll roof
Tar&Grv	Tar & gravel
WdShake	Wood shakes
WdShngl	Wood shingles

```r
train_set_graph %>% ggplot(aes(GrLivArea, sale_k, color = BldgType)) +
    geom_point(alpha = 0.5) +
    xlab('Above-grade living area, sq.ft.') + ylab('Sale price, $k') + theme_bw()
```

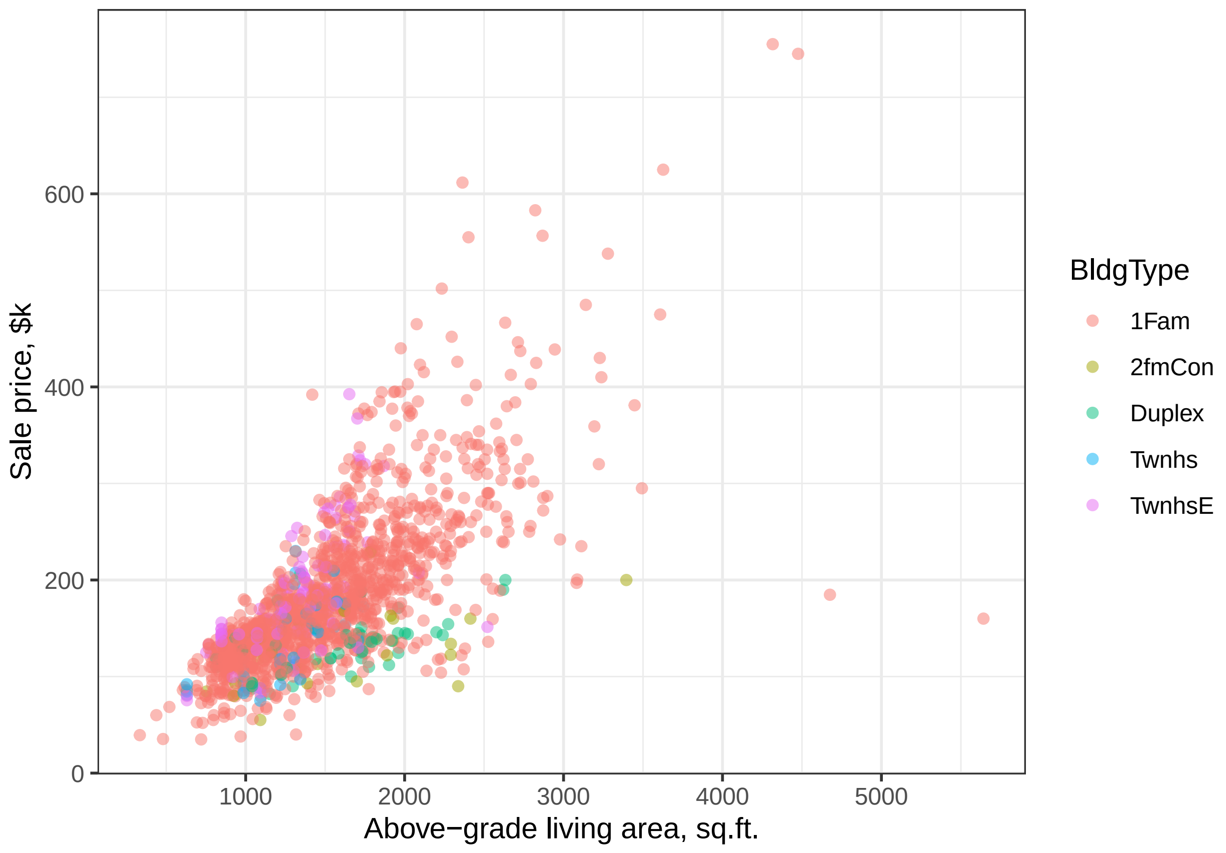

Figure 2.20: Sale price vs. above-grade living area and house type.

```r
train_set_graph %>% ggplot(aes(GrLivArea, sale_k, color = HouseStyle)) +
    geom_point(alpha = 0.5) +
    xlab('Above-grade living area, sq.ft.') + ylab('Sale price, $k') + theme_bw()
```

```r
train_set_graph %>% ggplot(aes(x=factor(OverallQual), y=sale_k)) + geom_boxplot() +
    xlab('Overall quality') + ylab('Sale price, $k') + theme_bw()
```

```r
train_set_graph %>% ggplot(aes(x=factor(OverallQual), y=LotFrontage)) + geom_violin() +
    scale_y_log10() +
    stat_summary(fun='mean', geom='point', shape=15, size=3, color='blue') +
    stat_summary(fun='median', geom='point', shape=16, size=3, color='red') +
    xlab('Overall quality') + ylab('Sale price, $k') +theme_bw()
```

```r
train_set_graph %>% ggplot(aes(x=factor(OverallCond), y=sale_k)) + geom_boxplot() +
    scale_y_log10() +
    xlab('Overall condition') + ylab('Sale price, $k') + theme_bw()
```

```r
train_set_graph %>% ggplot(aes(x=factor(OverallCond), y=LotFrontage)) + geom_violin() +
    scale_y_log10() +
    stat_summary(fun='mean', geom='point', shape=15, size=3, color='blue') +
    stat_summary(fun='median', geom='point', shape=16, size=3, color='red') +
```

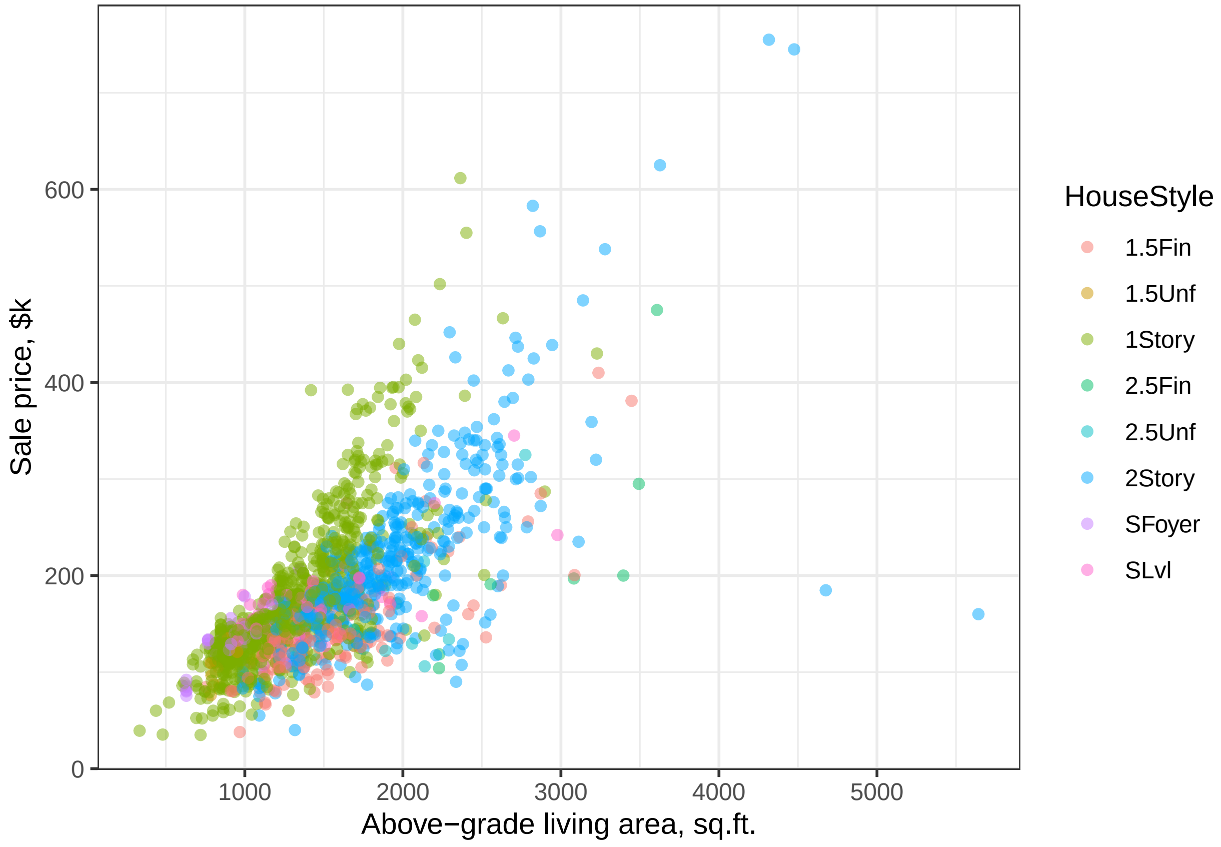

Figure 2.21: Sale price vs. above-grade living area and house style.

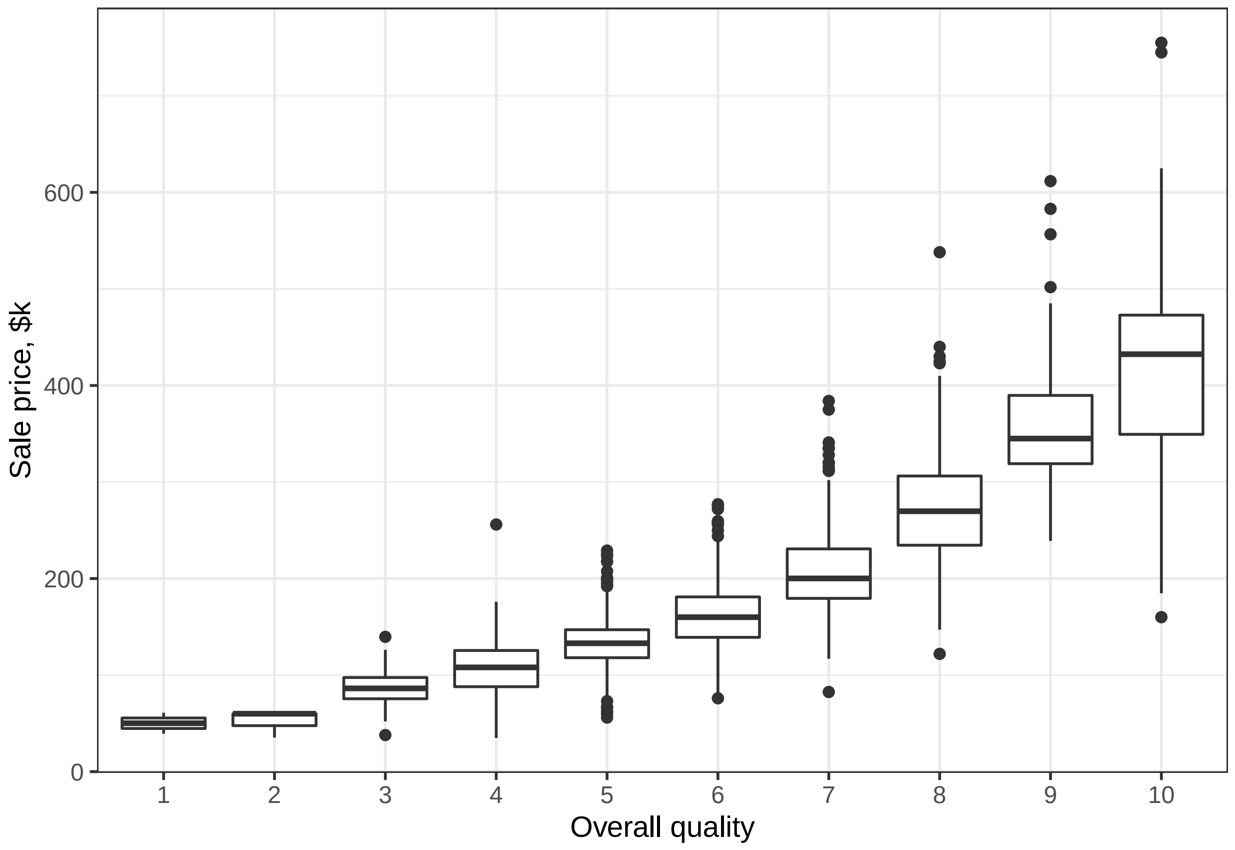

Figure 2.22: Boxplot of sale price vs. overall house quality.

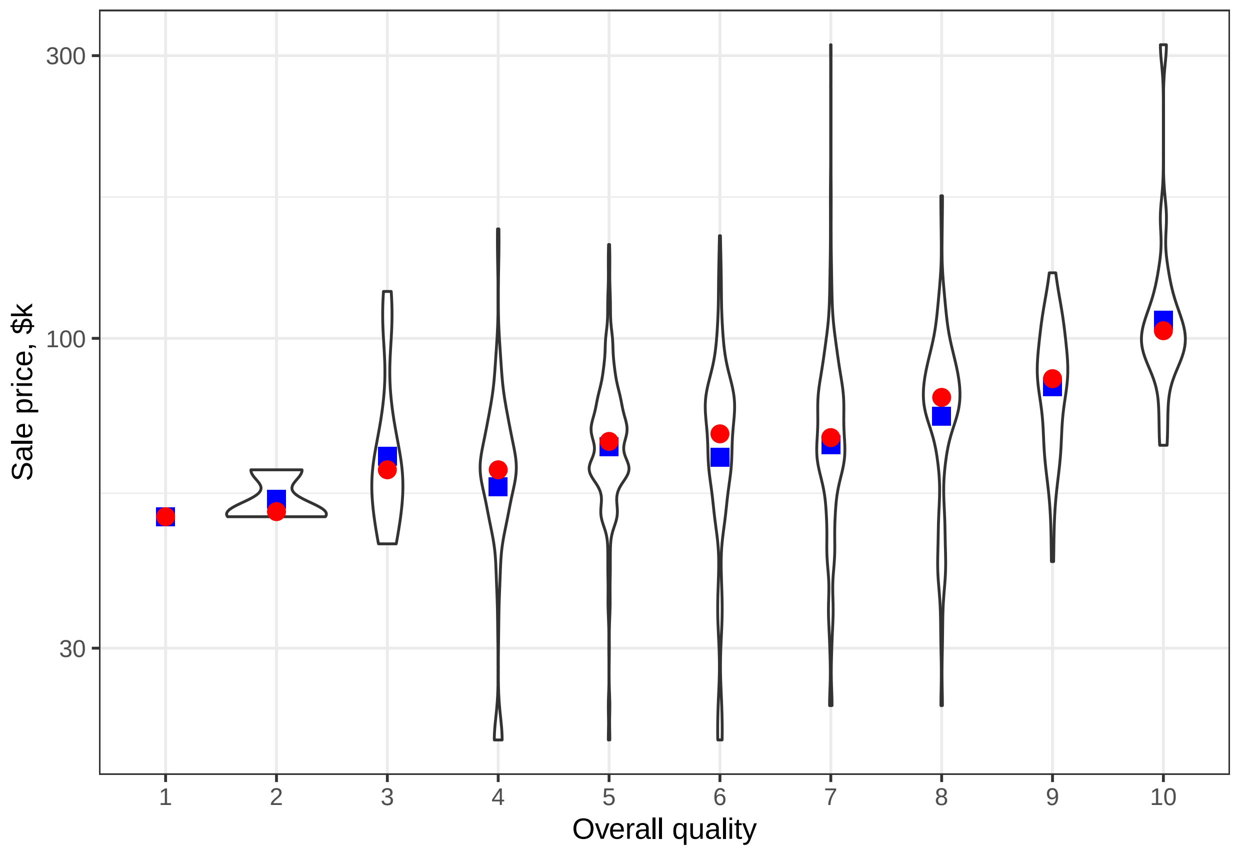

Figure 2.23: Violin plot of lot frontage vs. overall house quality. Blue = mean, red = median.

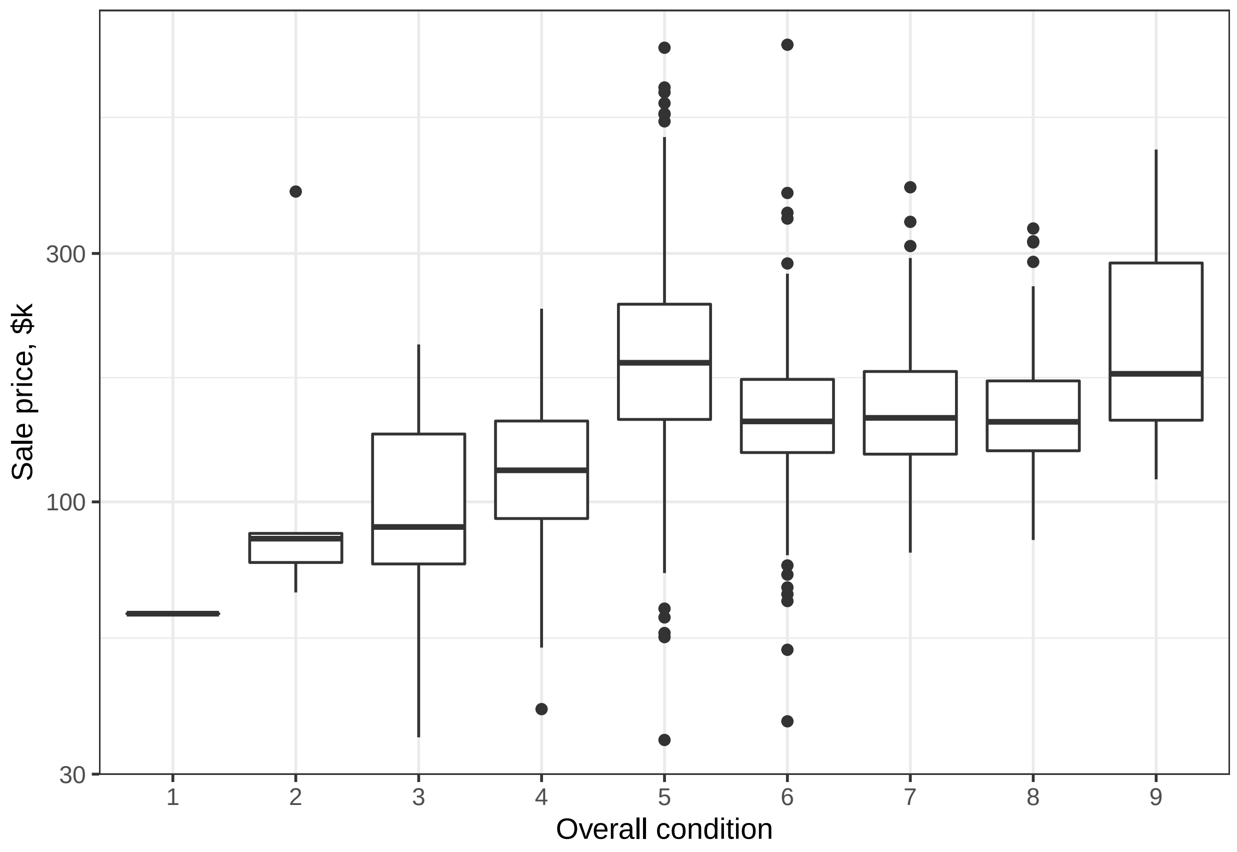

Figure 2.24: Boxplot of sale price vs. overall house condition.

```
    xlab('Overall condition') + ylab('Sale price, $k') + theme_bw()
```

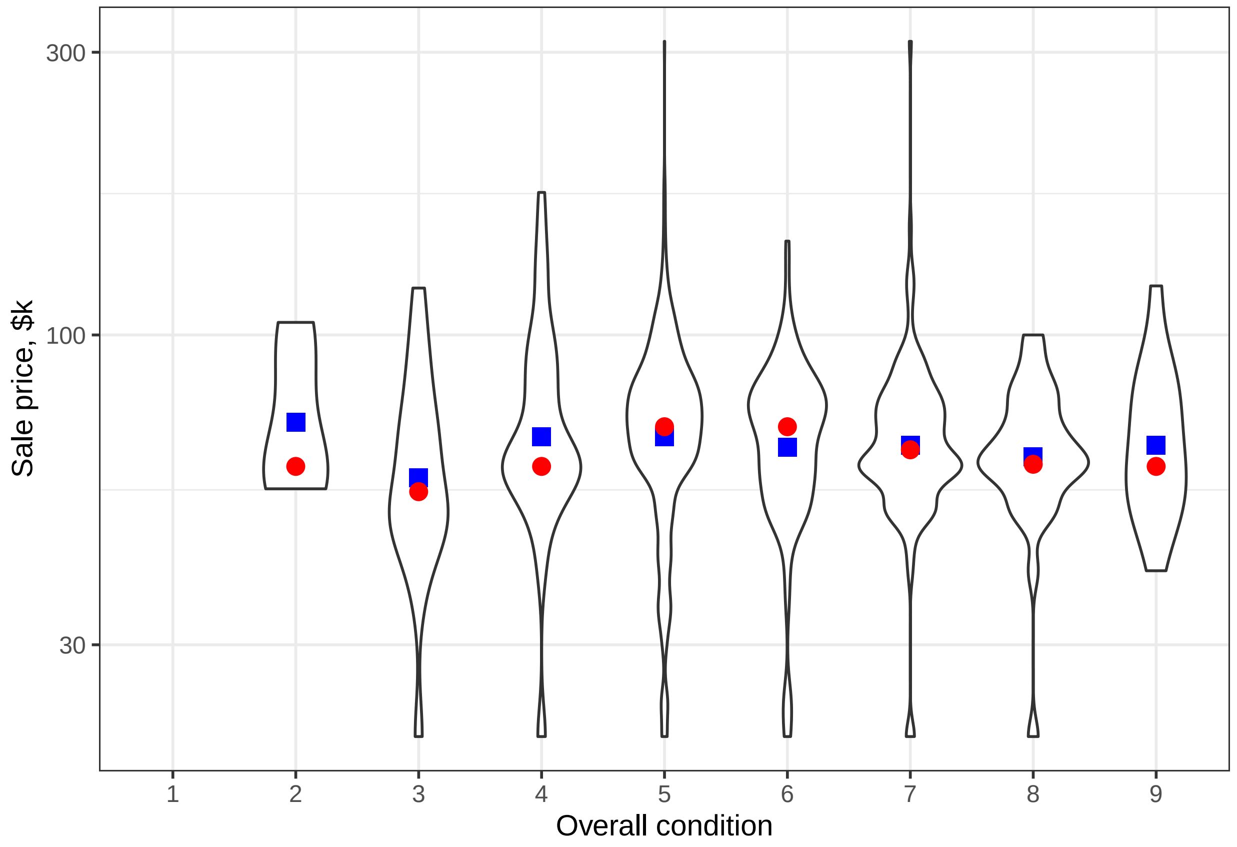

Figure 2.25: Violin plot of sale price by overall house condition.

```
train_set_graph %>% ggplot(aes(x=factor(RoofStyle), y=sale_k)) + geom_boxplot() +
    xlab('Roof style') + ylab('Sale price, $k') + theme_bw()
```

```
train_set_graph %>% ggplot(aes(x=factor(RoofMatl), y=sale_k)) + geom_boxplot() +
    xlab('Roof material') + ylab('Sale price, $k') + theme_bw()
```

```
roof_matl %>% knitr::kable()
```

RoofMatl	n_houses	freq
ClyTile	1	1
CompShg	1434	1
Membran	1	1
Metal	1	1
Roll	1	1
Tar&Grv	11	1
WdShake	5	1
WdShngl	6	1

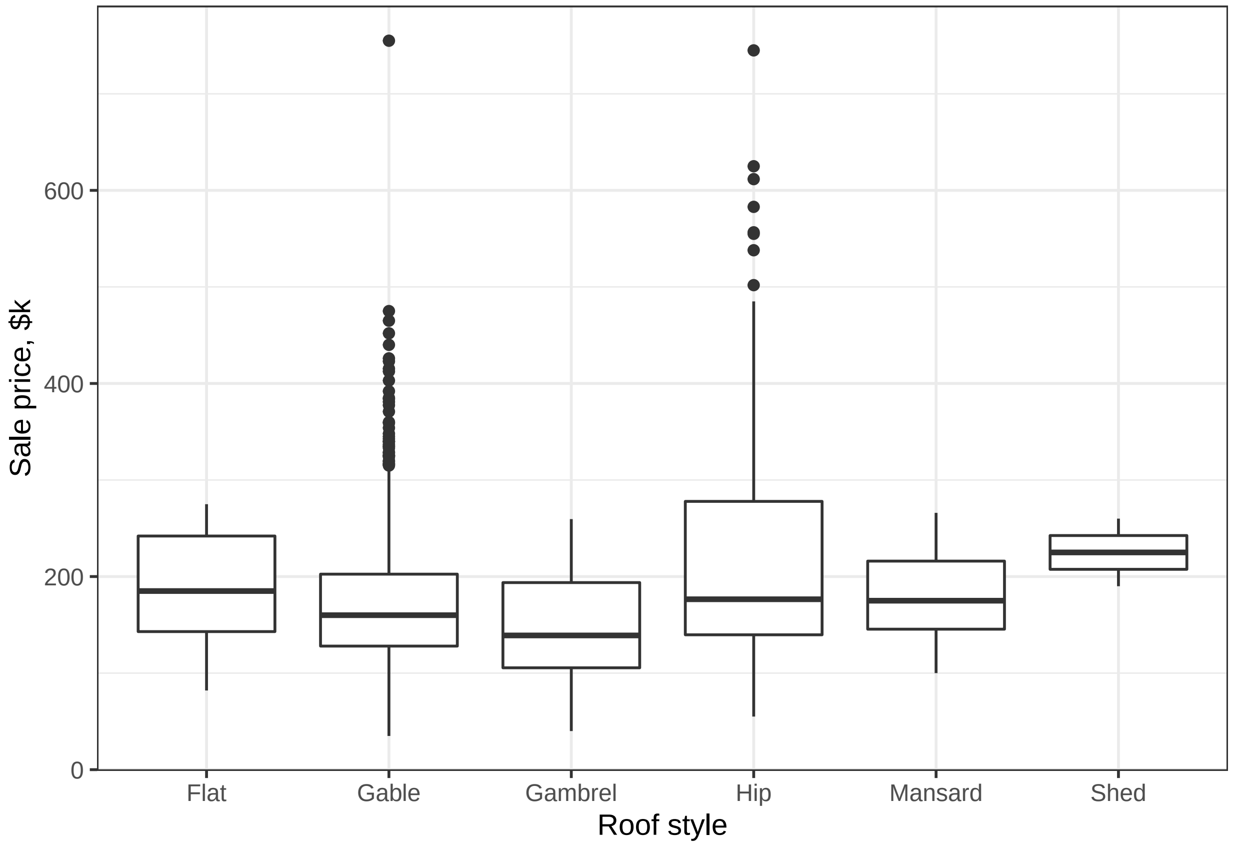

Figure 2.26: Boxplot of sale price vs. roof style.

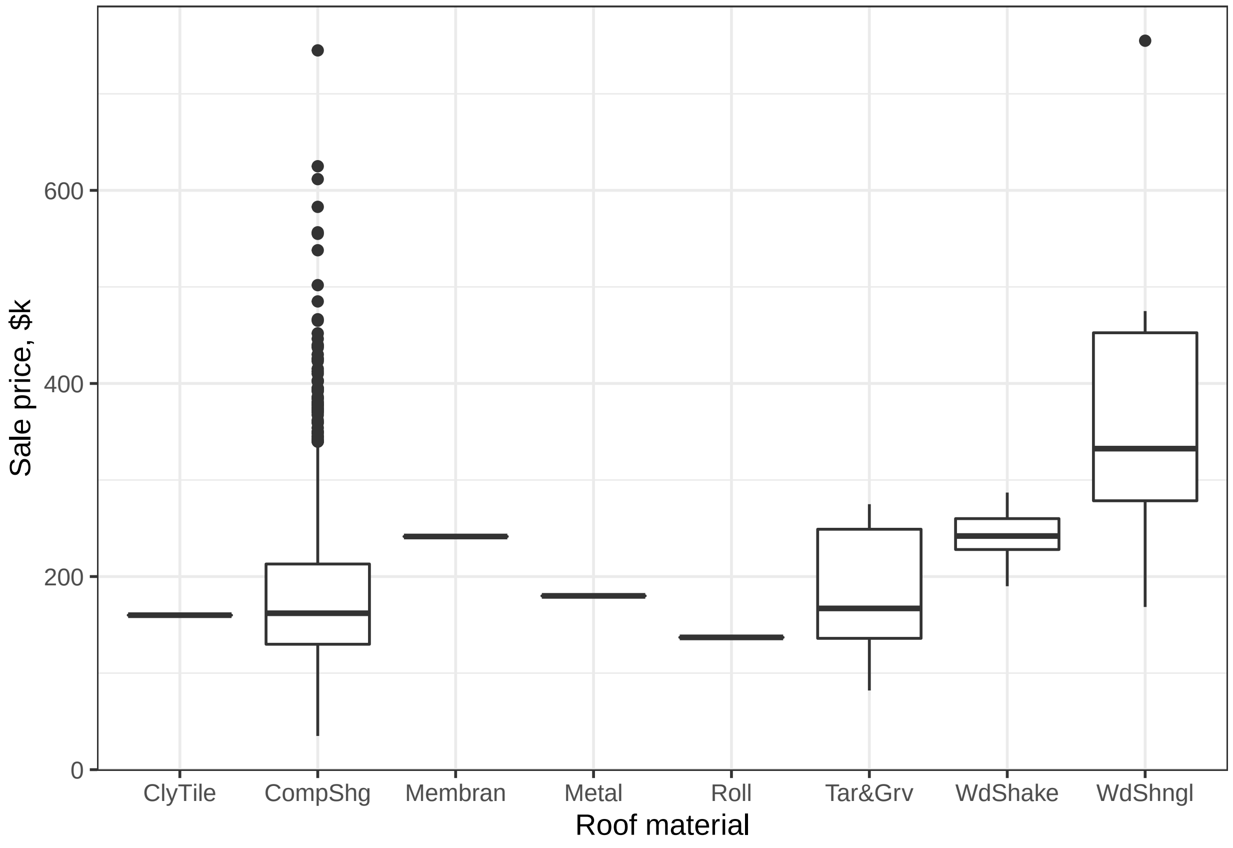

Figure 2.27: Boxplot of sale price vs. roof material.

2.5 Effects on Sale Price by Geography

Geography is another conventional predictor of home value, as evidenced by common perceptions of the "richer" or "poorer" parts of town, or by the perceived value (or lack thereof) in being near main thoroughfares, industrial zones, etc. The dataset provides several discriminators for neighborhood and local factors, both of which are taken to be categorical variables. These are summarized in Tables 2.6 and 2.7. A total of twenty-five neighborhoods and nine independent geographic proximities were considered.

The distributions of sale prices by neighborhood and geographic feature are shown in Fig. 2.28 and 2.29. There is clearly a significant difference in neighborhood location for some, but not all, neighborhoods: for instance, there's a significant difference in pricing between Bloomington Heights and Bluestem or Brookside, but pricing for the Clear Creek, College Creek, Crawford, and Sawyer West neighborhoods share heavy overlap and are probably not different to a statistically significant degree. Similarly, some pronounced differences were noted between certain geographic proximities — such as for positive off-site features — and sale price. Interestingly, the pricing spread for houses adjacent to arterial streets and east-west railroads is similar, as are the mean sale prices for houses located near or adjacent to the north-south railroad(s). While explanations for this are not given in the dataset, perhaps these differences arise partially due to the volume and frequency of rail traffic and its attendant noise footprint. All houses within the dataset can be flagged with up to two geographic proximities (e.g. adjacent to arterial street and adjacent to east-west railroad), and this interaction is considered in the model; however, for exploratory analysis it is sufficient to identify factors that are likely to influence the sale price of a home.

Table 2.7: Neighborhoods by identifier

Code	Type
Blmngtn	Bloomington Heights
Blueste	Bluestem
BrDale	Briardale
BrkSide	Brookside
ClearCr	Clear Creek
CollgCr	College Creek
Crawfor	Crawford
Edwards	Edwards
Gilbert	Gilbert
IDOTRR	Iowa DOT and Rail Road
MeadowV	Meadow Village
Mitchel	Mitchell
Names	North Ames
NoRidge	Northridge
NPkVill	Northpark Villa
NridgHt	Northridge Heights
NWAmes	Northwest Ames
OldTown	Old Town
SWISU	South & West of Iowa State University
Sawyer	Sawyer
SawyerW	Sawyer West
Somerst	Somerset
StoneBr	Stone Brook
Timber	Timberland
Veenker	Veenker

Table 2.8: Geographic proximities by identifier

Code	Type
Artery	Adjacent to arterial street
Feedr	Adjacent to feeder street
Norm	Normal
RRNn	Within 200' of North-South Railroad
RRAn	Adjacent to North-South Railroad
PosN	Near positive off-site feature: park, greenbelt, etc.
PosA	Adjacent to postive off-site feature
RRNe	Within 200' of East-West Railroad
RRAe	Adjacent to East-West Railroad

```
train_set_graph %>% ggplot(aes(x=factor(Neighborhood), y=sale_k)) + geom_boxplot() +
     xlab('Neighborhood') + ylab('Sale price, $k') + theme_bw() +
     theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

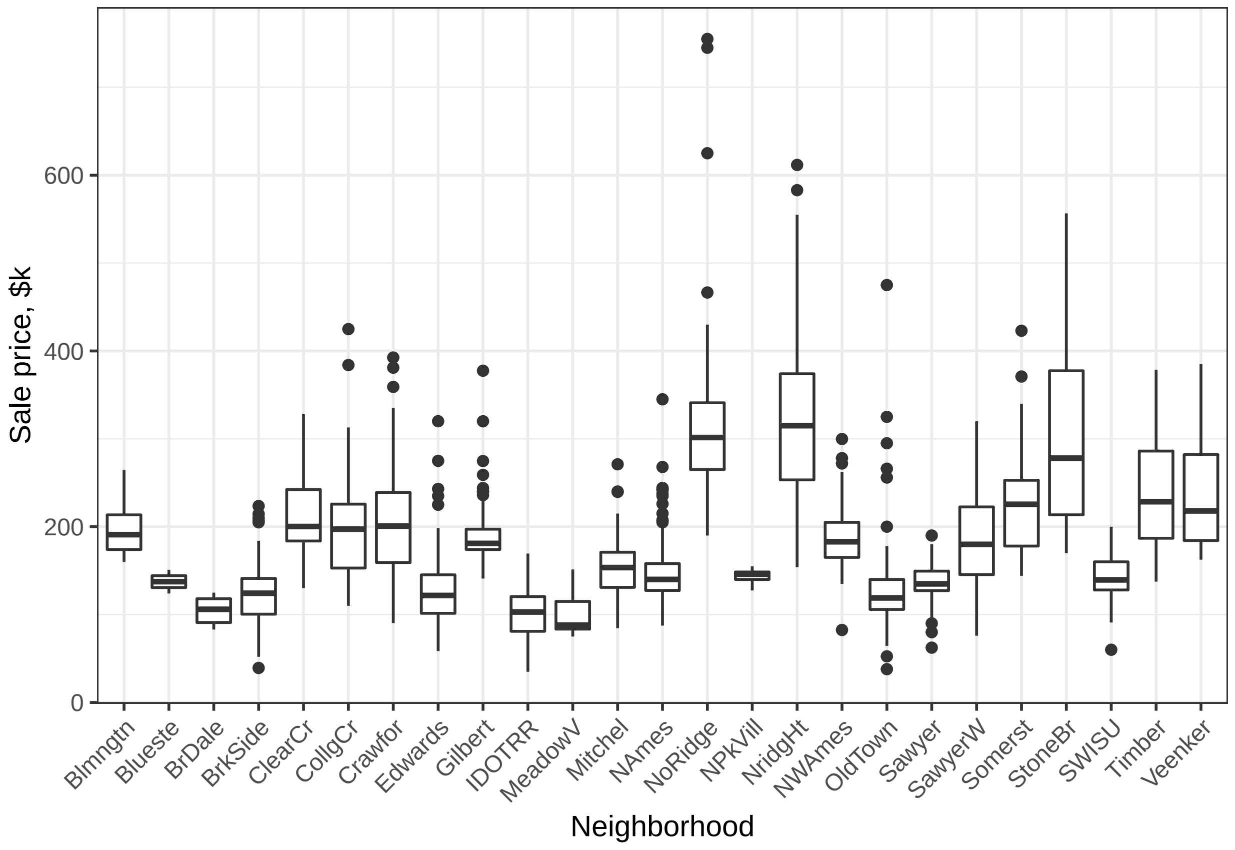

Figure 2.28: Boxplot of sale price vs. neighborhood.

```
train_set_graph %>% ggplot(aes(x=factor(Condition1), y=sale_k)) + geom_boxplot() +
     xlab('Conditions in close proximity') + ylab('Sale price, $k') + theme_bw()
```

3 Data Preparation & Model Creation

The first step in model creation is data preprocessing, which will eliminate undesirable NAs from the dataset and recategorize non-continuous variables as categorical or ordinal variables, as necessary. Examination of the complete dataset showed several NAs that required cleaning. Furthermore, skewed variables must be log-transformed as previously discussed. Both the training and test datasets were combined into one large dataset, `all_data`, for preprocessing; this variable was then split into the final training and test sets for model training and testing.

```
train_set2 <- within(train_set,rm(SalePrice)) # Remove last column
all_data <- rbind(train_set2, test_set)
set.seed(1, sample.kind = 'Rounding')
train_set$SalePrice <- log(train_set$SalePrice + 1)
```

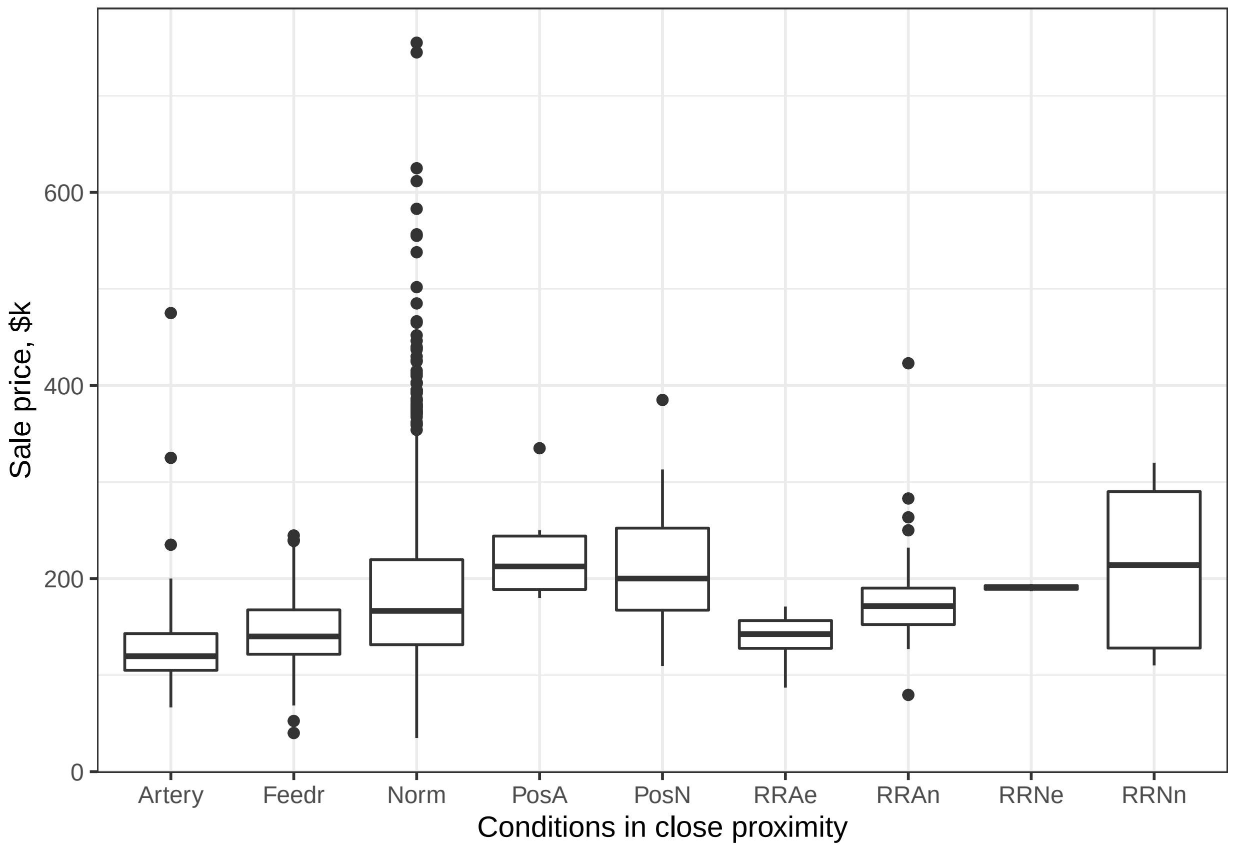

Figure 2.29: Boxplot of sale price vs. proximity to conditions.

3.1 Data Preparation

Variable transforms were performed by first separating predictors into `class` or `numeric` types, and then calculating the skew for each feature. Skewness is particularly damaging to a small sample size, which the present dataset can certainly be considered. The method of calculating skewness for n observations is given as

$$G_1 = g_1 \sqrt{\frac{n(n-1)}{n-2}}$$

by the `e1071` CRAN documentation based on the methods outlined by Joanes and Gill[3]. For the present work, only variables with a skewness greater than 0.7 were transformed; this value was held constant and serves as a potential tuning parameter for future work.Categorical variables must be transformed into numerical values to be useful in regression; however, due to their nature there's no inherent way to order them, so they must be transformed into a set of dummy variables with a simple binary value and any NA levels set to zero.

For numeric (continuous) variables, missing values for a given feature are replaced by the median value of that feature. For instance, a missing value of first-floor square footage would be replaced by the median value of first-floor square footage for the entire dataset. This is not ideal, and rests on the assumption of "most common = most likely." Furthermore, using the median value, rather than the mean value, will avoid potentially skewing the data.

The `all_data` data frame is then reconstructed with the processed data, and the final training and test sets are created. Some variables are combined to avoid effects of multicollinearity: for instance, all of the bathrooms in the basement and above-grade areas are combined into a `total_baths` object. This potentially reduces some of the resolution of the final model, but in exchange allows greater separation between predictor variables (lower p-values), likely contributing to a more powerful predictive model.

[3]Joanes, D.N.; Gill, C.A., "Comparing Measures of Sample Skewness and Kurtosis," *The Statistician*, 47, pp. 183–189, 1988.

```r
class_vars <- sapply(names(all_data),function(x){class(all_data[[x]])})
num_vars <-names(class_vars[class_vars != "character"])
skew_numeric <- sapply(num_vars,function(x){e1071::skewness(all_data[[x]],
                na.rm=TRUE, type = 2)})
skew_numeric <- skew_numeric[skew_numeric > 0.7]
for(x in names(skew_numeric)) {
        all_data[[x]] <- log(all_data[[x]] + 1)}
```

```r
cat_vars <- names(class_vars[class_vars == 'character'])
dummy_vars <- dummyVars(~.,all_data[cat_vars])
cat_vars2 <- predict(dummy_vars,all_data[cat_vars])
cat_vars2[is.na(cat_vars2)] <- 0
```

```r
# For missing values in the numeric variables vector, substitute the median of
# that feature. Avoid using the mean to reduce the effect of outliers.
num_vars2 <- all_data[num_vars]
for (x in num_vars) {
        median_value <- median(train_set[[x]],na.rm = TRUE)
        all_data[[x]][is.na(all_data[[x]])] <- median_value}
```

```r
all_data <- cbind(all_data[num_vars],cat_vars2)
train_set2 <- all_data[1:nrow(train_set),]
train_set2 <- mutate(train_set2, total_sf = TotalBsmtSF + X1stFlrSF + X2ndFlrSF + GrLivArea)
train_set2 <- mutate(train_set2, total_baths = BsmtFullBath + (0.5*BsmtHalfBath) + FullBath + (0.5*Half
train_set2 <- mutate(train_set2, porch_sf = OpenPorchSF + EnclosedPorch + X3SsnPorch + ScreenPorch)
test_set2  <- all_data[(nrow(train_set)+1):nrow(all_data),]
test_set2 <- mutate(test_set2, total_sf = TotalBsmtSF + X1stFlrSF + X2ndFlrSF + GrLivArea)
test_set2 <- mutate(test_set2, total_baths = BsmtFullBath + (0.5*BsmtHalfBath) + FullBath + (0.5*HalfBa
test_set2 <- mutate(test_set2, porch_sf = OpenPorchSF + EnclosedPorch + X3SsnPorch + ScreenPorch)
y <- train_set$SalePrice
```

3.2 Model Creation

The model used for the present work is a Gaussian-family generalized linear regression model fitted with a penalized maximum likelihood. Both a ridge and LASSO (least absolute shrinkage and selection operator) model were incorporated, since there are minuscule differences between the two. The regularization is computed at a variety of regularization parameters λ_i, and this particular method is robust enough to handle multiple shapes of data, including very large and very sparse matrices. The standard formulation of this model is as follows[4].

For x observations in $x_i \in \mathbf{R}^p$ and the responses $y_i \in \mathbf{R}$, where $i = 1, \ldots, N$, the objective function is

$$\min_{(\beta_0,\beta)\in\mathbf{R}^{p+1}} \frac{1}{2N}\sum_{i=1}^{N}\left(y_i - \beta_0 - x_i^T\beta\right)^2 + \lambda\left[(1-\alpha)\left|\frac{|\beta|_2^2}{2} + \alpha\right| |\beta|_1\right]$$

where α ranges between zero (ridge model) and one (LASSO model). For predictors $\tilde{\beta_0}, \tilde{\beta_l} \forall j \in 1,] \ldots, p$ the gradient becomes

$$\tilde{\beta_j} \leftarrow \frac{S\left(\frac{1}{N}\sum_{i=1}^{N} x_{ij}\left(y_i - \tilde{y_i}^j\right), \lambda\alpha\right)}{1 + \lambda(1-\alpha)}$$

where $\tilde{y_i}^j = \tilde{\beta_0} + \sum_{l \neq j}$ and $S(z, \gamma)$ is a thresholding operator. Note that this only applies when the $\mathbf{x}$ variables are standardized to have unit variance or near-unit variance, hence why the aforementioned transforms are necessary. Although the `glmnet` function can automatically calculate values of λ_i for use in the model, these were manually specified for the present work. Note that the set of regularization parameters λ must begin at their maximum value and decrement to zero for the model to function properly. For the present work, five-fold cross-validation was used with twenty-five repetitions, although these parameters can obviously be tuned based on user preference, data size, and computing power. As previously stated, the sole metric for assessing model performance is minimization of RMSE, which is defined as

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}$$

for N observations of predictor $\hat{y_i}$ and actual value y_i. In this case, this refers to the response variable, which is the sale price.

[4]From https://web.stanford.edu/~hastie/glmnet/glmnet_alpha.html#lin

```
train_control <- trainControl(method = 'repeatedcv', number = 5,
                              repeats = 25, verboseIter = FALSE)

# Ridge model (alpha = 0)
lambdas <- seq(10,0,-0.001)
ridge_model <- train(x = train_set2, y = y, method = 'glmnet', metric = 'RMSE',
                     maximize = FALSE, trControl = train_control,
                     tuneGrid=expand.grid(alpha = 0,  lambda = lambdas))

# LASSO model (alpha = 1) with narrowed lambda range
lasso_model <- train(x = train_set2, y = y, method='glmnet', metric='RMSE',
                maximize = FALSE, trControl = train_control,
                tuneGrid = expand.grid(alpha = 1,
                lambda = c(1,0.1,0.05,0.01,seq(0.005,0.001,-0.001),
                           0.0005,0.0005,0.0001)))
```

4 Results and Discussion

4.1 Model Results

Plotting of the complexity parameter λ versus the RMSE of the ridge model shows a minimum RMSE for $\lambda < 0.2$, as seen in Fig. 4.1. This narrows the range of λ_i that must be evaluated by the LASSO model. An R^2 value in excess of 0.90 is considered acceptable, and no further optimization is deemed necessary.

```r
# View results - lamdba can certainly be less than 0.2
ggplot(data = filter(ridge_model$result, lambda<0.2),aes(x = lambda, y = RMSE)) +
        geom_line() +
        xlab('Lambda') + ylab('RMSE') + theme_bw()
```

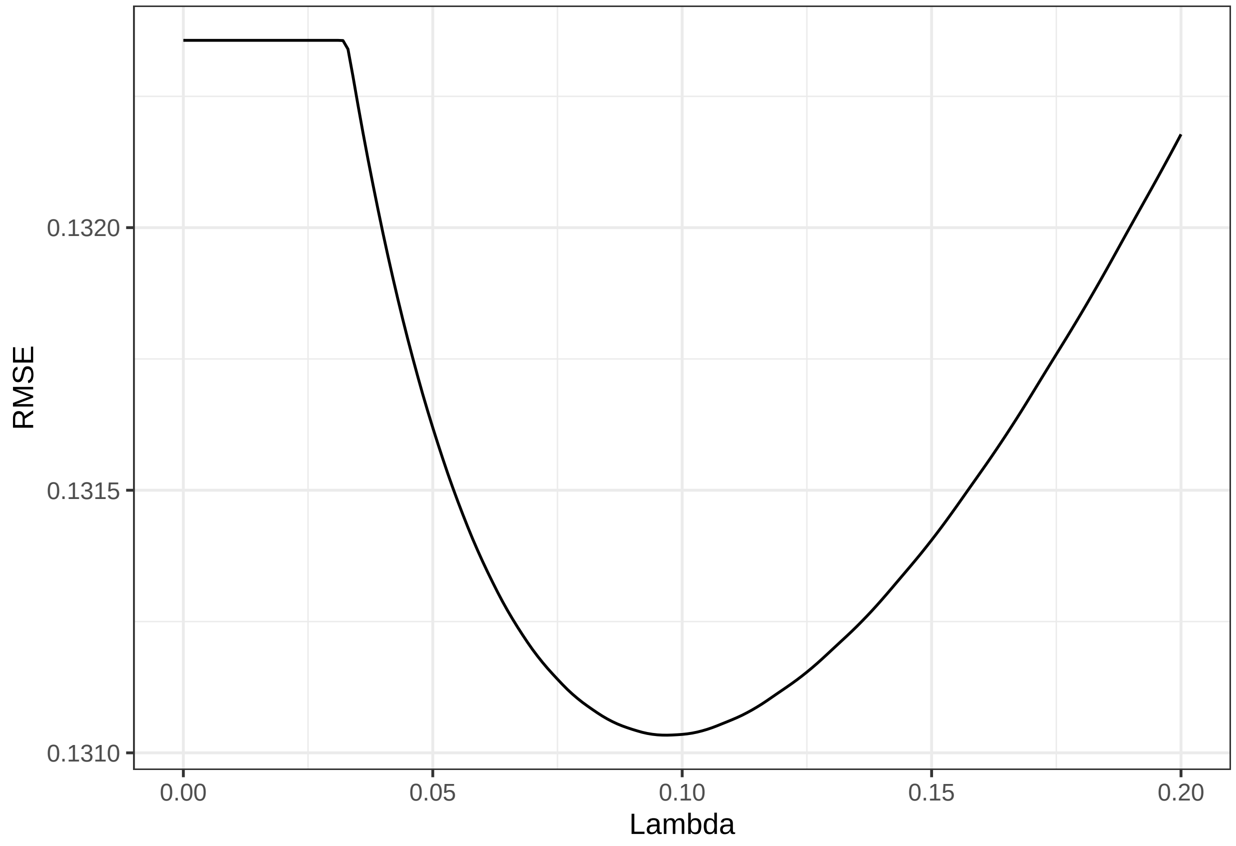

Figure 4.1: Results of complexity parameter for ridge model.

```r
model_results = tibble(Method = 'Ridge', RMSE = mean(ridge_model$resample$RMSE),
                       R_sq. = mean(ridge_model$resample$Rsquared))
model_results <- rbind(model_results, tibble(Method = 'LASSO',
                    RMSE = mean(lasso_model$resample$RMSE), R_sq. = mean(lasso_model$resample$Rsquared)
model_results %>% knitr::kable()
```

Method	RMSE	R_sq.
Ridge	0.1310337	0.8915881
LASSO	0.1250579	0.9014647

4.2 Final Model

Once categorical and ordinal variables were considered and coded appropriately, a total of 290 explanatory variables were used in the regression model. The aforementioned soft thresholding and formulation of the LASSO method selects only a subset of explanatory variables. LASSO is also capable of performing covariant selection (ridge does not), easing some of the data cleaning requirements that might otherwise be present. The LASSO regression coefficients are first extracted and separated into a `beta` vector and an `unpicked` vector, the former of which contains all of the regression coefficients chosen by the LASSO model. Of the 290 total variables used, only 108 were selected for the final model, and are sorted in terms of increasing value. The values for these 108 variables is given in Fig. 4.2.

Upon first glance, Fig. 4.2 is not usable, nor is every single variable useful despite their (sometimes small) nonzero values. The β_i values are split by positive and negative values, and the top ten of each were selected for inclusion into the vector `beta_final`. This can obviously be a point for further study: fewer predictor variables will result in a higher RMSE and lower R^2 value, but it must be noted that the performance of the *full* LASSO model has already been described and found to be acceptable. Selecting the ten most influential positive and negative predictors and showing their values is done purely for the sake of human readability: the LASSO model uses all 108 selected variables regardless.

The prediction model is of the form

$$Y_i = 2.249 + \{\beta\}^T \{\mathbf{x}\} + \epsilon$$

where $\{\beta\}$ is a vector of prediction coefficients for predictor variables $\{\mathbf{x}\}$ and ϵ is an error term. The final selected values for all coefficients are shown as follows.

```r
regression_coefs <- data.frame(coefficient_name = dimnames(coef(lasso_model$finalModel,
                                        s = lasso_model$bestTune$lambda))[[1]],
                               coefficient_value = matrix(coef(lasso_model$finalModel,
                                        s = lasso_model$bestTune$lambda)))
betas    <- filter(regression_coefs,coefficient_value != 0)
unpicked <- nrow(filter(regression_coefs,coefficient_value == 0))

# Reorder regression coefficients in ascending value
betas <- arrange(betas, -coefficient_value)
cat('Number of variables selected by LASSO model:',nrow(betas),
    '\nNumber of variables not selected by LASSO model:',unpicked)
```

```
## Number of variables selected by LASSO model: 106
## Number of variables not selected by LASSO model: 187
```

```r
ggplot(data = betas, aes(x = coefficient_name, y = coefficient_value)) +
        geom_bar(stat = 'identity') +
        coord_flip() +
        xlab('Regression coefficient') + ylab('Coefficient value') + theme_bw()
```

```r
beta_positives <- betas %>% filter(coefficient_value > 0)
intercept      <- beta_positives[1,]
beta_positives <- beta_positives[-1,]
beta_negatives <- betas %>% filter(coefficient_value < 0)
beta_positives <- rbind(head(beta_positives,10))
beta_negatives <- rbind(tail(beta_negatives,10))
beta_final     <- rbind(beta_positives, beta_negatives)

# Reorder regression coefficients in ascending value
betas <- arrange(betas, -coefficient_value)
```

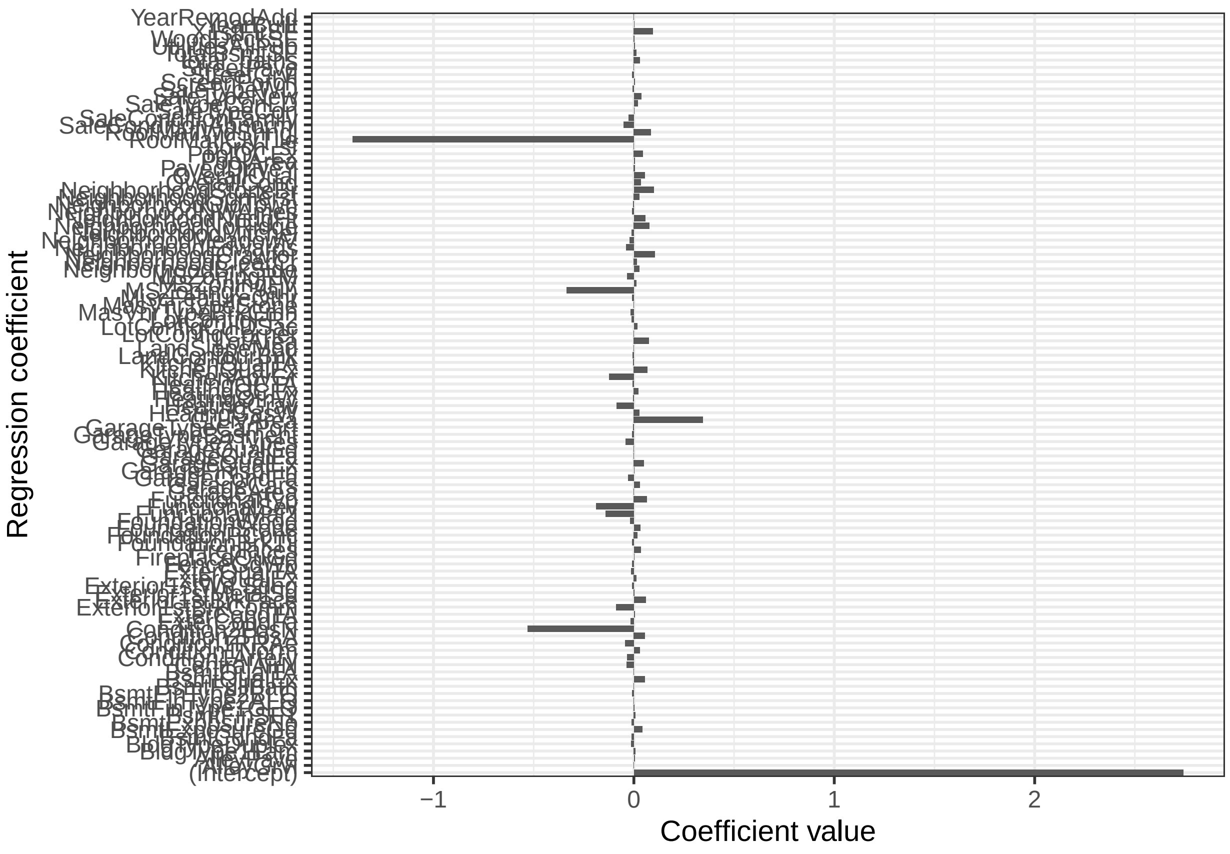

Figure 4.2: LASSO model coefficient values, all selected variables.

```r
ggplot(data = beta_final, aes(x = reorder(coefficient_name, coefficient_value),
                              y = coefficient_value)) +
        geom_bar(stat = 'identity') +
        #ylim(-1.5,0.6) +
        coord_flip() +
        xlab('Regression coefficients') + ylab('Coefficient value') + theme_bw()
```

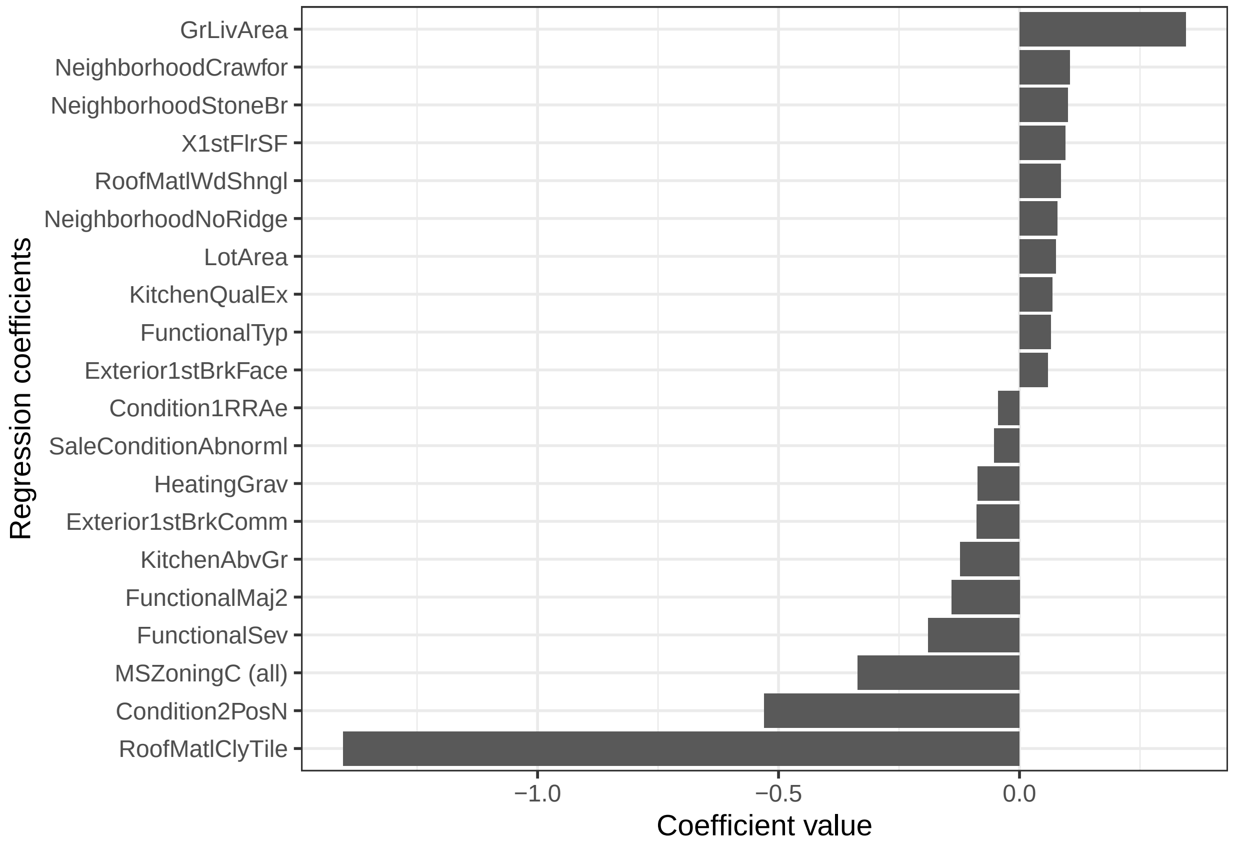

Figure 4.3: LASSO model coefficient values, top 20 most influential.

```r
all_betas = tibble(Coefficient = betas$coefficient_name, Value = betas$coefficient_value)
all_betas %>% knitr::kable()
```

Coefficient	Value
(Intercept)	2.7419809
GrLivArea	0.3449970
NeighborhoodCrawfor	0.1038931
NeighborhoodStoneBr	0.0998012
X1stFlrSF	0.0949580
RoofMatlWdShngl	0.0853574
NeighborhoodNoRidge	0.0778578
LotArea	0.0752392
KitchenQualEx	0.0676095
FunctionalTyp	0.0650296
Exterior1stBrkFace	0.0587644
NeighborhoodNridgHt	0.0565112
Condition2PosA	0.0553181

Coefficient	Value
BsmtQualEx	0.0537801
OverallQual	0.0537509
GarageQualEx	0.0501260
PoolQCEx	0.0453422
BsmtExposureGd	0.0422910
SaleTypeNew	0.0368809
OverallCond	0.0341123
Fireplaces	0.0337995
FoundationStone	0.0321841
total_baths	0.0310364
GarageCars	0.0295013
Condition1Norm	0.0288750
HeatingGasW	0.0279465
NeighborhoodSomerst	0.0277303
NeighborhoodBrkSide	0.0262449
HeatingQCEx	0.0233283
SaleTypeConLD	0.0195375
FoundationPConc	0.0183604
LotConfigCulDSac	0.0172196
NeighborhoodClearCr	0.0153116
ExterQualEx	0.0135501
TotalBsmtSF	0.0133604
MSZoningFV	0.0114908
BldgType1Fam	0.0079877
BsmtFinSF1	0.0079480
PavedDriveY	0.0061018
ExterCondTA	0.0049393
UtilitiesAllPub	0.0047150
PoolArea	0.0046003
AlleyPave	0.0043981
ScreenPorch	0.0041085
BsmtFinType1GLQ	0.0030636
WoodDeckSF	0.0026287
Exterior1stMetalSd	0.0025381
SaleTypeCon	0.0019002
GarageFinishFin	0.0018744
FireplaceQuGd	0.0016123
YearBuilt	0.0015698
porch_sf	0.0012033
YearRemodAdd	0.0007412
LandSlopeMod	0.0006831
GarageQualGd	0.0004555
BsmtFinType2ALQ	0.0003437
GarageArea	0.0000940
BsmtFullBath	0.0000894
LotConfigCorner	0.0000628
MasVnrTypeStone	0.0000097
StreetPave	0.0000000
GarageQualFa	-0.0002561
AlleyGrvl	-0.0017137
BsmtQualTA	-0.0018311
HeatingOthW	-0.0027480

Coefficient	Value
KitchenQualTA	-0.0027742
GarageTypeCarPort	-0.0039551
NeighborhoodOldTown	-0.0047386
HeatingQCTA	-0.0053256
SaleTypeWD	-0.0060832
LandContourBnk	-0.0074628
GarageTypeBasment	-0.0075660
BsmtFinType2BLQ	-0.0078204
FoundationBrkTil	-0.0082660
Exterior1stWd Sdng	-0.0085324
NeighborhoodNWAmes	-0.0090711
MiscFeatureOthr	-0.0092368
StreetGrvl	-0.0092614
FenceGdWo	-0.0093674
BsmtExposureNo	-0.0106255
LotConfigFR2	-0.0107908
BsmtCondFa	-0.0115888
NeighborhoodMitchel	-0.0119460
ExterQualTA	-0.0126558
BldgTypeDuplex	-0.0131135
ExterCondFa	-0.0163419
MasVnrTypeBrkCmn	-0.0163695
FoundationWood	-0.0200244
NeighborhoodMeadowV	-0.0202935
SaleConditionFamily	-0.0268732
GarageCondFa	-0.0275782
Condition1Artery	-0.0330437
MSZoningRM	-0.0339884
CentralAirN	-0.0366652
NeighborhoodEdwards	-0.0394825
GarageType2Types	-0.0419296
Condition1RRAe	-0.0445334
SaleConditionAbnorml	-0.0523614
HeatingGrav	-0.0862290
Exterior1stBrkComm	-0.0891217
KitchenAbvGr	-0.1225372
FunctionalMaj2	-0.1410863
FunctionalSev	-0.1895500
MSZoningC (all)	-0.3354000
Condition2PosN	-0.5300609
RoofMatlClyTile	-1.4030529

4.3 Discussion

For a majority of the selected variables, the effect on predicted sale price was intuitive and matched the interpretations presented in Sec. 2. As seen in Fig. 4.3, house prices were driven primarily by above-grade living area, first-floor square footage, excellent kitchen quality, and location in the Crawford, Edwards, or Stone Brook neighborhoods. Major-deduction and severe functional damage (e.g. `FunctionalMaj2` and `FunctionalSev`) were found to adversely affect the sale price, as would be expected, along with commercial zoning, gravity furnaces ("octopus ducting" that can fill a room with duct work [5]) and abnormal sale conditions (short-sales, foreclosures, etc.). Less expected was that an above-grade kitchen or common-brick face would adversely affect the sale price, or that adjacency to an east-west railroad would adversely affect sale price more than adjacency to a north-south railroad (perhaps the assumption of rail line-specific variances of traffic volume and frequency was valid). The adverse effect of clay tile roofing material on sale price is likely overweight, as only one home in the dataset had clay tile roofing (see the `RoofMatl kable` in Sec. 2.4); it may be possible that this home had other factors that more strongly influenced its sale price, and the selection of the roofing material variable was in error. Proximity to a second positive feature (`Condition2PosN`) as a negative factor was also unexpected, especially when one considers that proximity or adjacency to a first positive feature (`Condition1PosN` or `Condition1PosA`, respectively) was not even a top-ten positive factor. It may be possible that the region penalizes "too much of a good thing," and proximity to multiple positive features increases traffic and/or noise near the homes; however, this is purely conjecture, and is not knowable from the dataset.

Despite these apparent inconsistencies, the LASSO model is preferred due to the higher R^2 and lower RMSE. The difference between the two models is not significant, but accuracy is critical in real estate, and a lost sale for want of a prediction with 1-2% more accuracy is a real concern.

[5]https://structuretech1.com/gravity-furnaces/

5 Conclusions and Recommendations for Future Work

A machine learning algorithm using a LASSO regression model was developed from the Ames, Iowa housing price dataset. Through regularization and five-fold cross-validation the final prediction model is given as

$$Y_i = 2.249 + \{\beta\}^T \{\mathbf{x}\} + \epsilon$$

where $\{\beta\}$ is a vector of prediction coefficients for the 108 predictor variables $\{\mathbf{x}\}$ that were selected by the algorithm. The final model was found to have an RMSE of 0.1251 and R^2 of 0.9015. The majority of predictor variables behaved in a fashion consistent with intuition and real estate intuition, although outliers may have adversely affected the performance of the algorithm and the training set should be reconstructed without these outliers present; however, the performance of the model was deemed adequate and these adjustments would be fine-tuning at best, and were not performed for the present work.

Several future work items remain, none of which are possible with the given dataset. Firstly, the effect of square footage on sale price was strong and positive, as expected; however, real estate is an industry subject to personal and cultural preferences, both of which change over time. A longitudinal study of the influence of certain factors on sale price would be of practical and research interest, as it can form the basis for an AI-driven model that can attempt to predict sale price at a future date based on historical social and macro trends. Similarly, buyer qualities were not accounted for in the dataset, but it stands to reason that demographics and profession will shape a buyer's wants in a home, and thus also influence their willingness to pay a premium for certain features or to invest in a home only if other features are discounted. The proximity of certain types of homes to certain industries or job opportunities is likewise lacking. Roof type and roofing material are critical variables when home-shopping, but roof condition was not present in the dataset and would likely play a major role in a purchase decision (viz. re-roofing a home is expensive and time-consuming); it is likely that roof condition is one of the factors that comprise the overall quality and/or overall condition variables, but this is unknown and it is not known what influence roof condition would have on those variables at any rate. Finally, no rigorous skewness testing was performed, and the value of the skewness parameter used for data transformation can be considered general-purpose at best; however, real data are "messy" and are rarely well-composed, and it's possible that a more rigorous treatment of skewness might result in the transformation of more variables than strictly necessary.