Linear Model Prediction

Vincent Wan

- UW ID: 20870719

Librares, Packages, and Configurations

```
install.packages("olsrr")
install.packages("units")
install.packages("sf")
install.packages("caret")
install.packages("varImp")
library(MASS)
library(car)
library(qqtest)
library(leaps)
knitr::opts_chunk$set(fig.width=6, fig.height=4)
```

Summary

Formula

"price~poly(sizeLiving, 2) + poly(latitude, 7) + view + poly(dist_from_downtown, 2) + region + south + poly(sizeLot, 5) + poly(yrb, 6) + poly(saledate, 3) +

east + waterfront + condition_at_least_average + poly(longitude, 2) + poly(bedrm, 2) + bathrm + floors + poly(latitude, 7):close_to_downtown + south:east + region:east +

poly(latitude, 7):poly(longitude, 2) + poly(bedrm, 2):bathrm + waterfront:floors + close_to_downtown:poly(longitude, 2) + poly(latitude, 7):close_to_downtown:poly(longitude,2)"

New Predictors

- south: indicator of whether the house is in the south region of Toronto. The south region of Toronto is indicated by a longitude coordinate less than -79.33759.
- east: indicator of whether the house is in the east region of Toronto. The east region of Toronto is indicated by a latitude coordinate greater than or equal to 44.00227.
- dist_from_downtown: distance from downtown Toronto, using Pythagorean theorem and coordinates. The coordinate of downtown Toronto is (43.6548, -79.3883).
- close_to_downtown: indicator of whether the house is close to downtown Toronto. Being close to downtown Toronto is indicated by a distance of less than 0.475 from downtown.
- region: region in Toronto that the house is located. "Inner" for inner Toronto, and "Outer" for outer Toronto.
- condition_at_least_average: indicator of whether the condition of the house is at least average. This means an encoded score of at least 2.

Very Quick Summary of Model Building Procedure

- I first added a predictor for the distance from downtown Toronto, `dist_from_towntown`, and then transformed it using boxcox transformation so that its distribution is approximately normal. I then defined an indicator predictor for whether the house is close to downtown Toronto or not, `close_to_downtown`. This is because it seemed that the scatterplot of `dist_from_towntown` vs. `price` was a piecewise function, where all distances before -0.5130778 had a quadratic trend, and all distances after -0.5130778 had a different quadratic trend. Hence, the formula that best represented the scatterplot was `price~poly(dist_from_downtown, 2)*close_to_downtown`.
- Next, I added a predictor for inner and outer regions in Toronto, `region`. I also added a predictor for whether the house is in the east region of Toronto (latitude coordinate greater than or equal to 44.00227), as well as a predictor for whether the house is in the south region of Toronto (longitude coordinate less than -79.33759).
- After that, I converted `saledate` into a numerical variate, `waterfront` into a categorical variate, and `view` into a categorical variate. I also removed `sizeBelow` from the dataset since it is linearly dependent with respect to `sizeAbove` and `sizeLiving`.
- I found that the scatterplot of `saledate` vs. `price` followed a cubic trend. Hence, the formula that best represented the scatterplot was `price~poly(saledate, 3)`.
- I transformed `bedrm` using boxcox transformation so that its distribution is approximately normal. I then found that the scatterplot of `bedrm` vs. `price` followed a quadratic trend. Hence, the formula that best represented the scatterplot was `price~poly(bedrm, 2)`.
- I transformed `bathrm` using boxcox transformation so that its distribution is approximately normal. I then found that the scatterplot of `bathrm` vs. `price` followed a linear trend. Hence, the formula that best represented the scatterplot was `price~bathrm`.
- I transformed `sizeLiving` using boxcox transformation so that its distribution is approximately normal. I then found that the scatterplot of `sizeLiving` vs. `price` followed a quadratic trend. Hence, the formula that best represented the scatterplot was `price~poly(sizeLiving, 2)`.
- I transformed `sizeLot` using boxcox transformation so that its distribution is approximately normal. I then found that the scatterplot of `sizeLot` vs. `price` followed a quintic trend. Hence, the formula that best represented the scatterplot was `price~poly(sizeLot, 5)`.
- I transformed `floors` using boxcox transformation so that its distribution is approximately normal. I then found that the scatterplot of `floors` vs. `price` followed a linear trend. Hence, the formula that best represented the scatterplot was `price~floors`.
- I encoded the condition of the house to be a numerical variate from 1 to 5, and then transformed it using boxcox transformation so that its distribution is approximately normal. I then defined an indicator predictor for whether the condition of the house is at least average (encoded score of at least 2), `condition_at_least_average`. This is because it seemed that the scatterplot of `condition` vs. `price` was a piecewise function, where all conditions before 0.8 had a horizontal trend at a certain intercept, and all conditions after 0.8 had a horizontal trend at a different intercept. Hence, the formula that best represented the scatterplot was `price~condition_at_least_average`. I removed `condition` from the dataset since I am not using it.
- I found that the scatterplot of `yrb` vs. `price` followed a 6 degree polynomial trend. Hence, the formula that best represented the scatterplot was `price~poly(yrb, 6)`.
- I found that the scatterplot of `longitude` vs. `price` followed a quadratic trend. Hence, the formula that best represented the scatterplot was `price~poly(longitude, 2)`.
- I found that the scatterplot of `latitude` vs. `price` followed a 7 degree polynomial trend. Hence, the formula that best represented the scatterplot was `price~poly(latitude, 7)`.

- Using the transformations, all the formulas above that best represented their respective scatterplots, alongside the categorical variates `waterfront`, `south`, `east`, `region`, and `view`, I considered a potential model to be the following:

"price ~ poly(saledate, 3) + bathrm + poly(bedrm, 2) + poly(sizeLot, 5) + poly(sizeLiving, 2) +

floors + waterfront + poly(yrb, 6) + condition_at_least_average + south + east + region + view + poly(longitude, 2) +

poly(latitude, 7) + poly(dist_from_downtown, 2) * close_to_downtown"

- Finally, I accounted for interactive effects. I found that the model with interaction between the number of bathrooms and the number of bedrooms resulted in a lower press. In other words, in the model, the term `bathrm*poly(bedrm, 2)` is better than the term `bathrm + poly(bedrm, 2)` when it comes to lowering $PRESS$.
- I found that the model with interaction between the number of house floors and whether the house has a waterfront resulted in a lower press. In other words, in the model, the term `floors*waterfront` is better than the term `floors + waterfront` when it comes to lowering $PRESS$.
- I found that the model with interaction between whether the house is in the south section of Toronto, whether the house is in the east section of Toronto, and whether the house is in the inner region of Toronto, resulted in a lower press. In other words, in the model, the term `south*east*region` is better than the term `south + east + region` when it comes to lowering $PRESS$.
- I found that the model with interaction between the house latitude coordinate, house longitude coordinate, and closeness to downtown, resulted in a lower press. In other words, in the model, the term `poly(longitude, 2)*poly(latitude, 7)*close_to_downtown` is better than the term `poly(longitude, 2)+poly(latitude, 7)+close_to_downtown` when it comes to lowering $PRESS$.
- This resulted in a new potential model being the following:

"price ~ poly(saledate, 3) + bathrm * poly(bedrm, 2) + poly(sizeLot, 5) + poly(sizeLiving, 2) +

floors * waterfront + poly(yrb, 6) + condition_at_least_average + south * east * region + view +

poly(longitude, 2) * poly(latitude, 7) * close_to_downtown + poly(dist_from_downtown, 2) * close_to_downtown"

- Finally, I performed stepwise model selection (forward and backward), and removed `close_to_downtown` to find the "best" model. Here is the best model:

"price~poly(sizeLiving, 2) + poly(latitude, 7) + view + poly(dist_from_downtown, 2) + region + south + poly(sizeLot, 5) + poly(yrb, 6) + poly(saledate, 3) +

east + waterfront + condition_at_least_average + poly(longitude, 2) + poly(bedrm, 2) + bathrm + floors + poly(latitude, 7):close_to_downtown + south:east + region:east +

poly(latitude, 7):poly(longitude, 2) + poly(bedrm, 2):bathrm + waterfront:floors + close_to_downtown:poly(longitude, 2) + poly(latitude, 7):close_to_downtown:poly(longitude,2)"

Model Building Procedure In Detail

Here is the function for the boxcox transformation function that we will use in this document.

```r
boxcox_transform <- function(data, lambda) {
  return(((data ^ lambda) - 1) / lambda)
}
```

Before starting this analysis, I considered fitting all variates into the model (the most basic model), and checking the $PRESS$ statistic. The model formula is `price~.`.

```r
load("linear.Rdata")
print(paste0("PRESS: ", olsrr::ols_press(lm(price~., data=dat))))
```

```
## [1] "PRESS: Inf"
```

From the output, we have that the press is infinity. This means that our next goal is to find a model with press lower than that. I also found that the dataset corresponds to houses in the GTA (Greater Toronto Area), which is very useful context when fitting models.

Distance from Downtown

First, from personal knowledge, I knew that there was some correlation between houses and their distance from downtown. Hence, I decided to add a predictor for that called `dist_from_downtown`.

From a quick Google search, I found that the heart of downtown Toronto is 43.6548 latitude, and -79.3883 longitude. To calculate the distance away from that point (`dist_from_downtown`), I used the variates `latitude` and `longitude`, as well as the default Pythagorean Theorem formula.

```r
# downtown Toronto coordinates
downtown_latitude = 43.6548
downtown_longitude = -79.3883

# calculate distance from downtown for each house, and define new predictor
dat$dist_from_downtown = sqrt((dat$latitude - downtown_latitude)^2
                              + (dat$longitude - downtown_longitude)^2)
```

Here is a histogram for the distance from downtown per house.

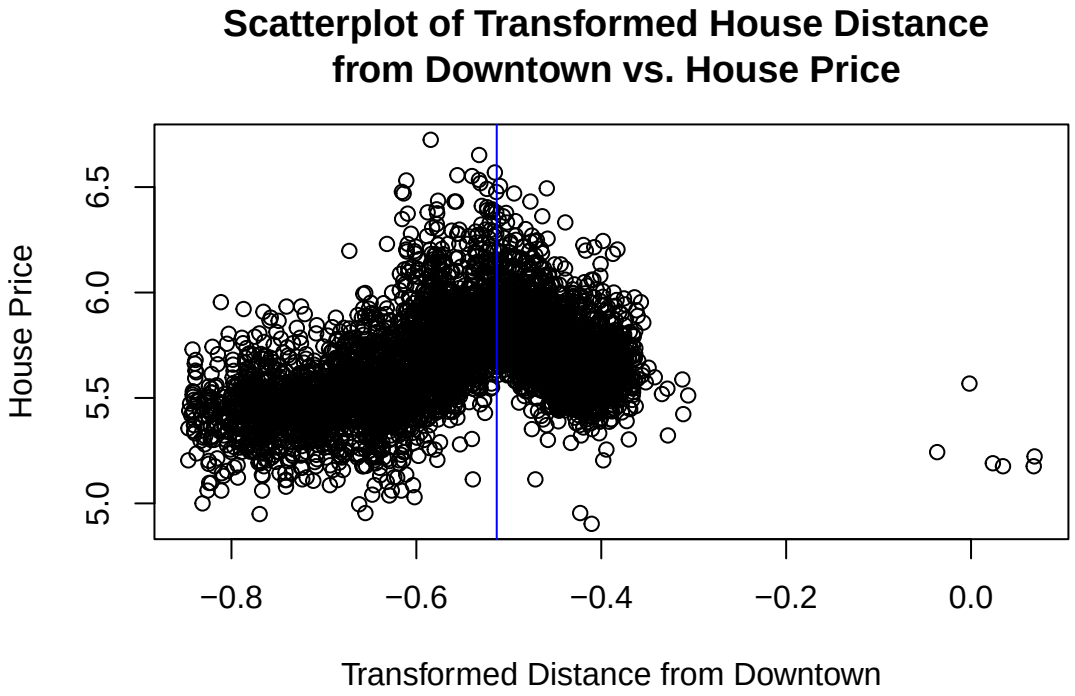
```r
hist(dat$dist_from_downtown, xlab="Distance from Downtown (43.6548, -79.3883)",
     main="Histogram of Distance \n from Downtown per House")
```

Since this histogram does not appear to be normal nor uniform, it follows that a boxcox transformation is needed. To do this, we will use the boxcox library function, find its maximum likelihood, and then use that value to transform the variate using the function defined before.

```
bc_dist <- MASS::boxcox(lm(dat$dist_from_downtown~dat$price), plotit=FALSE)
print(paste0("Maximum Likelihood: ", bc_dist$x[which.max(bc_dist$y)]))
```

```
## [1] "Maximum Likelihood: 1.1"
```

```
dat$dist_from_downtown <- boxcox_transform(dat$dist_from_downtown,
                                           bc_dist$x[which.max(bc_dist$y)])

# plot distance from downtown vs. price of house
plot(dat$dist_from_downtown, dat$price, xlab="Transformed Distance from Downtown",
     ylab="House Price",
     main="Scatterplot of Transformed House Distance \n from Downtown vs. House Price")
abline(v=-0.5130778, col="blue")
```

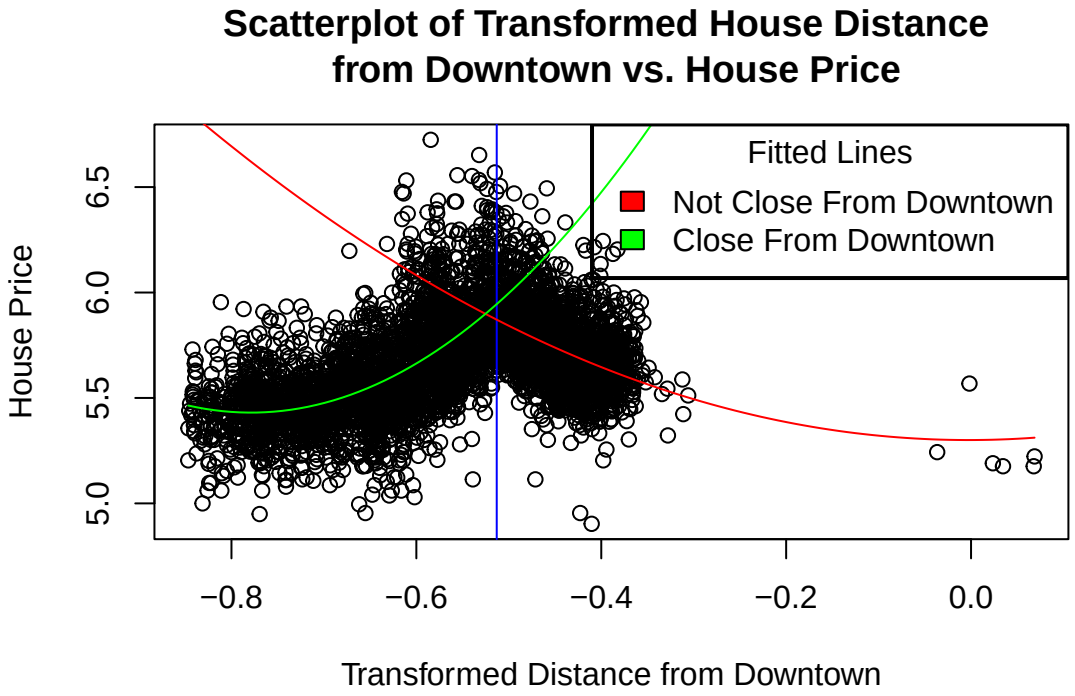


From the plot of the house transformed distance from downtown vs. the price of the house, it seems like there is a cutoff at $x = -0.5130778$, where the data follows an upwards trend when $x < -0.5130778$, and follows a downwards trend when $x > -0.5130778$. Because of this, let's create another predictor `close_to_downtown`. This is an indicator of whether the house is close to downtown Toronto. Being close to downtown Toronto is indicated by a transformed distance of less than -0.5130778.

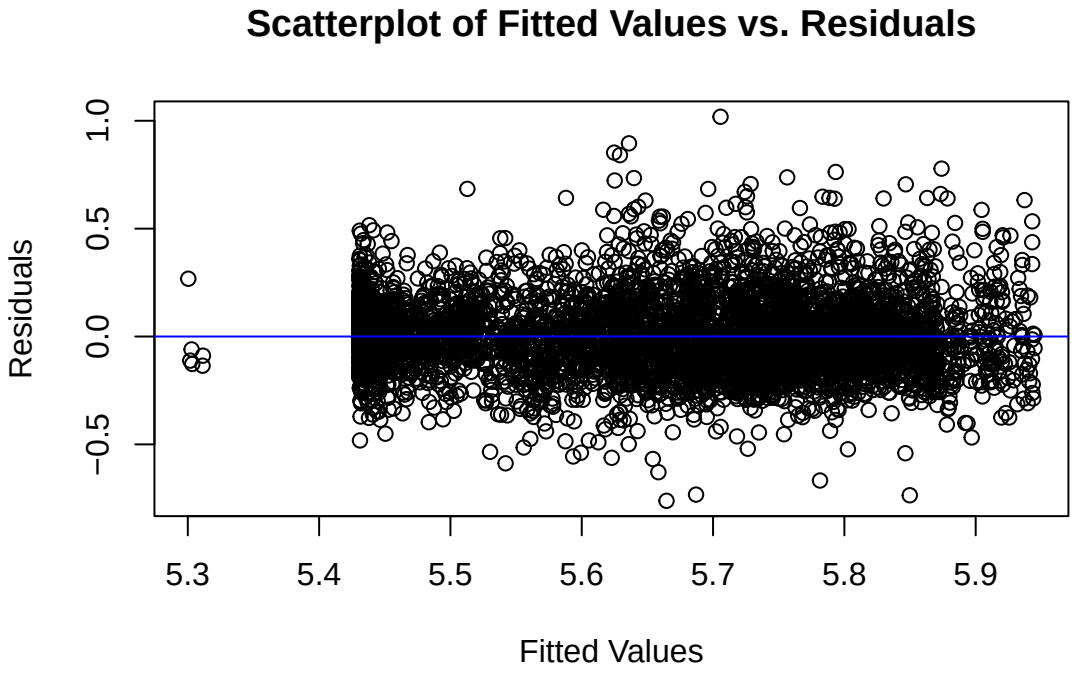
Notice that when the house is close to downtown (i.e. $x < -0.5130778$), the data seems to follow an upwards quadratic trend. When the house is not close to downtown (i.e. $x > -0.5130778$), the data seems to follow a downwards quadratic trend. Let's plot the transformed house distance from downtown vs. the price of the house, alongside the fitted lines. Let's also plot the fitted residuals.

```
# define new predictor for closeness from downtown
dat$close_to_downtown <- factor(as.integer(as.logical(dat$dist_from_downtown < -0.5130778)))

# fit quadratic model
fit_dist <- lm(price~poly(dist_from_downtown, 2)*close_to_downtown, data=dat)

# plot distance from downtown vs. price of house
plot(dat$dist_from_downtown, dat$price, xlab="Transformed Distance from Downtown",
     ylab="House Price",
     main="Scatterplot of Transformed House Distance \n from Downtown vs. House Price")
abline(v=-0.5130778, col="blue")

# predict prices when house is not close from downtown
predicted_prices_far <- predict(
  fit_dist,
  newdata = data.frame(
    dist_from_downtown = seq(min(dat$dist_from_downtown),
                             max(dat$dist_from_downtown),
                             length.out = 100
```

```
    ),
    close_to_downtown = rep(factor(0), 100)
  )
)

# predict prices when house is close from downtown
predicted_prices_close <- predict(
  fit_dist,
  newdata = data.frame(
    dist_from_downtown = seq(min(dat$dist_from_downtown),
                             max(dat$dist_from_downtown),
                             length.out = 100
    ),
    close_to_downtown = rep(factor(1), 100)
  )
)

# add lines to plot
lines(seq(min(dat$dist_from_downtown),
          max(dat$dist_from_downtown),
          length.out = 100),
      predicted_prices_far, col = "red"
)

lines(seq(min(dat$dist_from_downtown),
          max(dat$dist_from_downtown),
          length.out = 100),
      predicted_prices_close, col = "green"
)

legend(x = "topright", box.lwd = 2 , title="Fitted Lines",
       legend=c("Not Close From Downtown", "Close From Downtown"),
       fill = c("red","green"))
```




```
# plot fitted residuals
plot(fit_dist$fitted.values, fit_dist$residuals, xlab="Fitted Values",
     ylab="Residuals", main="Scatterplot of Fitted Values vs. Residuals")
abline(h=0, col="blue")
```



From the scatterplot of the transformed house distance from downtown vs. the house price, we can see that the fitted lines seem to match the trend of the datapoints quite nicely. In fact, even the outliers are somewhat captured by the red fitted line representing the data points not close from downtown. From the scatterplot of fitted values vs. residuals, we can see that the expected value of the residuals is approximately 0, since for each fitted value, there seems to be as many datapoints above the horizontal blue line as there are below. The variance of the residuals does not seem constant, since the spread of the residuals seems to decrease from fitted value 5.425 to fitted value 5.5. This cannot be handled appropriately since we cannot transform the response variate. Finally, there are no noticable outliers.

Regions

Next, from personal knowledge, I knew that different regions in Toronto had different house price ranges. Hence, I decided to add a predictor for that called `region`. I defined a bounding box, where all houses located inside the bounding box are considered to be in the "Inner" region of Toronto, and all houses located outside the bounding box are considered to be in the "Outer" region of Toronto.

```
# define bounding box
bb <- data.frame(
  lat_min = 43.5518,
  lat_max = 44.10837,
  lon_min = -79.52619,
  lon_max = -79.42471
)

# define function that checks if the house is in the bounding box
# and updates the dataset
check_in_bounding_box <- function(bounding_box) {
  # add a region column to the dataset
  dat$region <- NA

  # check each point against the bounding box
  for (i in seq_len(nrow(dat))) {
    if (dat$latitude[i] >= bounding_box$lat_min &&
        dat$latitude[i] <= bounding_box$lat_max &&
        dat$longitude[i] >= bounding_box$lon_min &&
        dat$longitude[i] <= bounding_box$lon_max) {
      dat$region[i] <- "Inner"
    } else {
      dat$region[i] <- "Outer"
    }
  }
  return(dat)
}

# check if each house is in bounding box and update dataset
dat <- check_in_bounding_box(bb)

# make region categorical
dat$region <- factor(dat$region)
```

Sections

Next, from personal knowledge, I knew that different geographic sections in Toronto had different house price ranges. Hence, I decided to add two predictors for that called `south` and `east`. `south` is an indicator of whether the house is in the south region of Toronto. The south region of Toronto is indicated by a longitude coordinate less than -79.33759. `east` is an indicator of whether the house is in the east region of Toronto. The east region of Toronto is indicated by a latitude coordinate greater than or equal to 44.00227.

```
# add variates for south and east sections
dat$south <- factor(sapply(dat$longitude, function(x) {
  if (x < -79.33759) 1 else 0
}))
dat$east <- factor(sapply(dat$latitude, function(x) {
  if (x >= 44.00227) 1 else 0
}))
```

Modifying/Removing Old Variates

Summary and N/As

Here is output for the summary of the data.

```
summary(dat)
```

```
##      price          saledate             bedrm           bathrm
##  Min.   :4.903   Length:5000        Min.   :0.000   Min.   :0.000
##  1st Qu.:5.508   Class :character   1st Qu.:3.000   1st Qu.:1.750
##  Median :5.655   Mode  :character   Median :3.000   Median :2.250
##  Mean   :5.667                      Mean   :3.353   Mean   :2.131
##  3rd Qu.:5.811                      3rd Qu.:4.000   3rd Qu.:2.500
##  Max.   :6.724                      Max.   :9.000   Max.   :7.500
##    sizeLiving      sizeLot           floors        waterfront
##  Min.   : 390   Min.   :   520   Min.   :1.000   Min.   :0.0000
##  1st Qu.:1440   1st Qu.:  5024   1st Qu.:1.000   1st Qu.:0.0000
##  Median :1910   Median :  7662   Median :1.500   Median :0.0000
##  Mean   :2089   Mean   : 16192   Mean   :1.509   Mean   :0.0082
##  3rd Qu.:2560   3rd Qu.: 10920   3rd Qu.:2.000   3rd Qu.:0.0000
##  Max.   :7480   Max.   :881654   Max.   :3.500   Max.   :1.0000
##       view            condition      sizeAbove      sizeBelow
##  Min.   :0.0000   poor     :   8   Min.   : 390   Min.   :   0.0
##  1st Qu.:0.0000   fair     :  39   1st Qu.:1220   1st Qu.:   0.0
##  Median :0.0000   average  :3279   Median :1580   Median :   0.0
##  Mean   :0.2364   good     :1286   Mean   :1805   Mean   : 283.7
##  3rd Qu.:0.0000   very_good: 388   3rd Qu.:2240   3rd Qu.: 550.0
##  Max.   :4.0000                    Max.   :6530   Max.   :2850.0
##       yrb          latitude       longitude      dist_from_downtown
##  Min.   :1895   Min.   :43.67   Min.   :-79.67   Min.   :-0.8467
##  1st Qu.:1947   1st Qu.:43.95   1st Qu.:-79.48   1st Qu.:-0.6516
##  Median :1971   Median :44.05   Median :-79.37   Median :-0.5434
##  Mean   :1967   Mean   :44.04   Mean   :-79.36   Mean   :-0.5613
##  3rd Qu.:1992   3rd Qu.:44.16   3rd Qu.:-79.27   3rd Qu.:-0.4625
##  Max.   :2010   Max.   :44.26   Max.   :-78.47   Max.   : 0.0687
```

```
##  close_to_downtown   region     south    east
##  0:2120            Inner: 833   0:2075   0:1750
##  1:2880            Outer:4167   1:2925   1:3250
##
##
##
##
```

Notice that `saledate` is currently considered a categorical variate, where each date is a level. It should instead be a numerical variate, where the ordering is preserved for each date. `waterfront` is currently considered a numerical variate when it shouldn't be, because it represents the binary indicator for a waterfront view. `view` is currently considered a numerical variate when it shouldn't be, because it represents the rating of the quality of the view.

Every other variate is appropriately typed (numerical variates are appropriately numerical, and categorical variates are appropriately categorical).

```
# convert saledate into a numerical variate
dat$saledate <- as.numeric(as.Date(dat$saledate))

# convert numerical variates into categorical ones
dat$waterfront <- factor(dat$waterfront)
dat$view <- factor(dat$view)
```

Next, notice that $sizeBelow + sizeAbove = sizeLiving$. This makes sense since the size of the living space below ground plus the size of the living space above ground equals the size of the living space. However, this means that `sizeBelow` is linearly dependent with respect to `sizeAbove` and `sizeLiving`, hence we do not need it.

```
# remove sizeBelow from the dataset
dat <- subset(dat, select = -c(sizeBelow) )
```

Finally, we check for N/A values in the data.

```
sum(is.na(dat))
```

```
## [1] 0
```

There are no N/A values, so we can proceed.

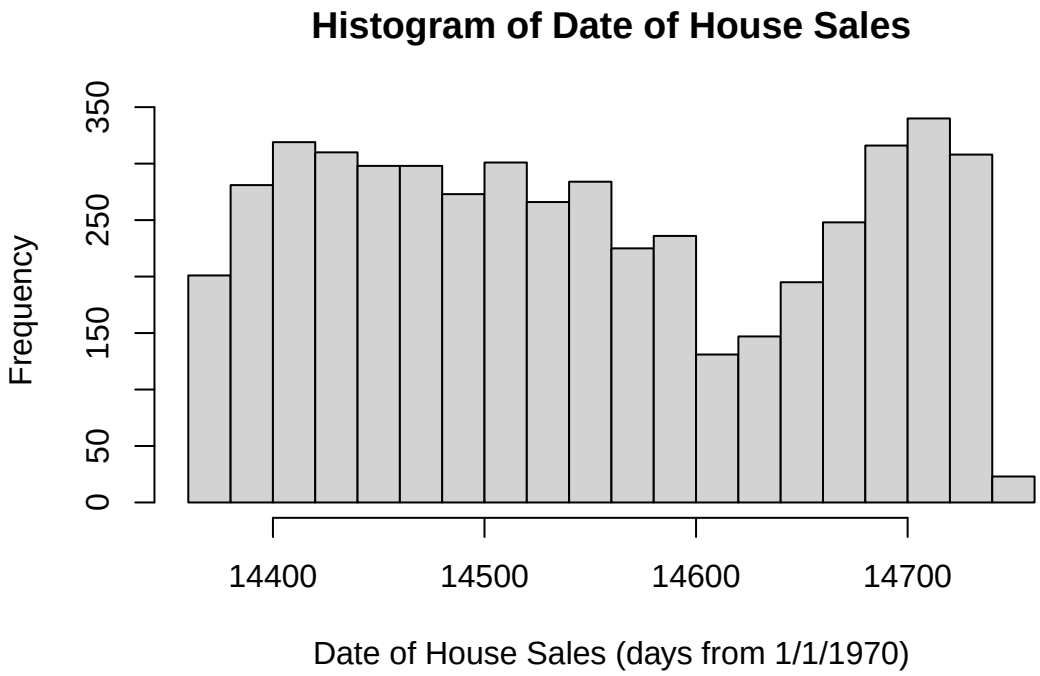
Applying Transformations and Encoding Predictors

Let's normalize the distribution of the predictors and fit appropriate models for each individual predictor. This will most likely result in a lower $PRESS$ statistic.

Date of Sales

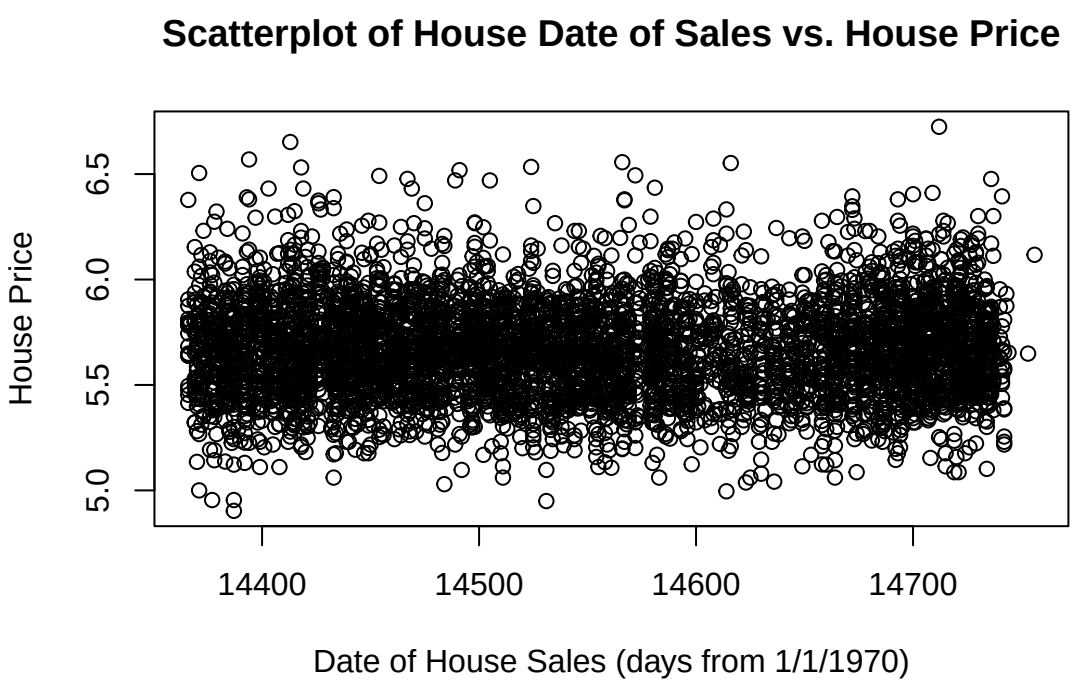
Here is a histogram for the date of the house sales.

```r
hist(dat$saledate, xlab="Date of House Sales (days from 1/1/1970)",
     main="Histogram of Date of House Sales")
```



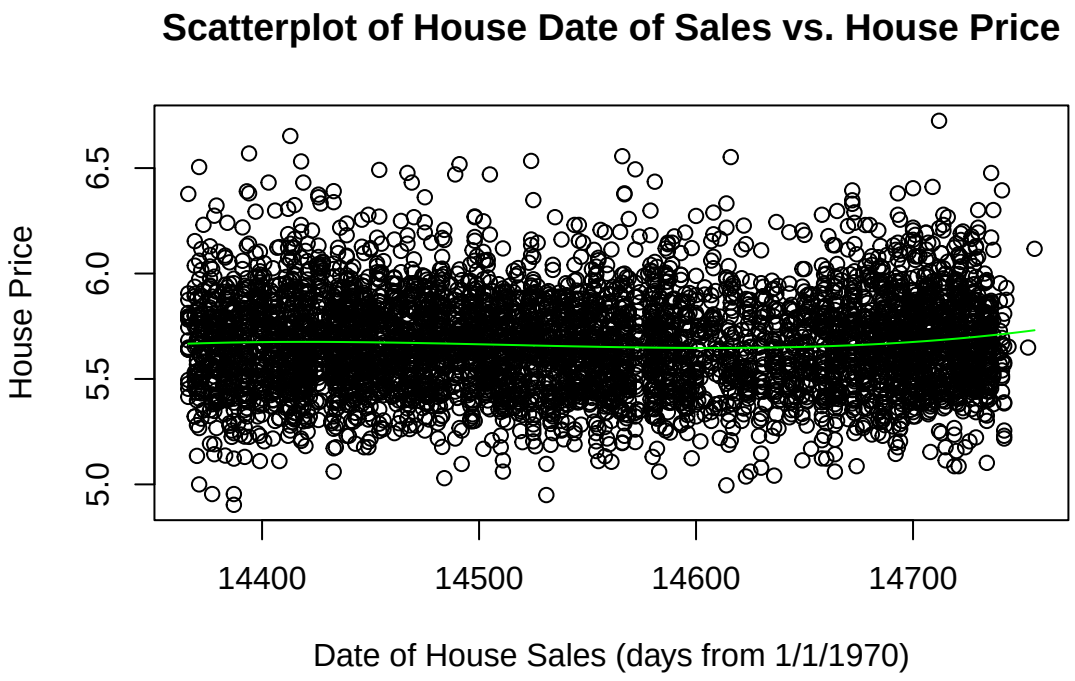
This histogram appears to be approximately uniformly distributed, so there is a possibility that houses are sampled according to their date of sale. Hence, we will not transform the variate date of house sales.

```r
# plot date of sales vs. price of house
plot(dat$saledate, dat$price, xlab="Date of House Sales (days from 1/1/1970)",
     ylab="House Price", main="Scatterplot of House Date of Sales vs. House Price")
```

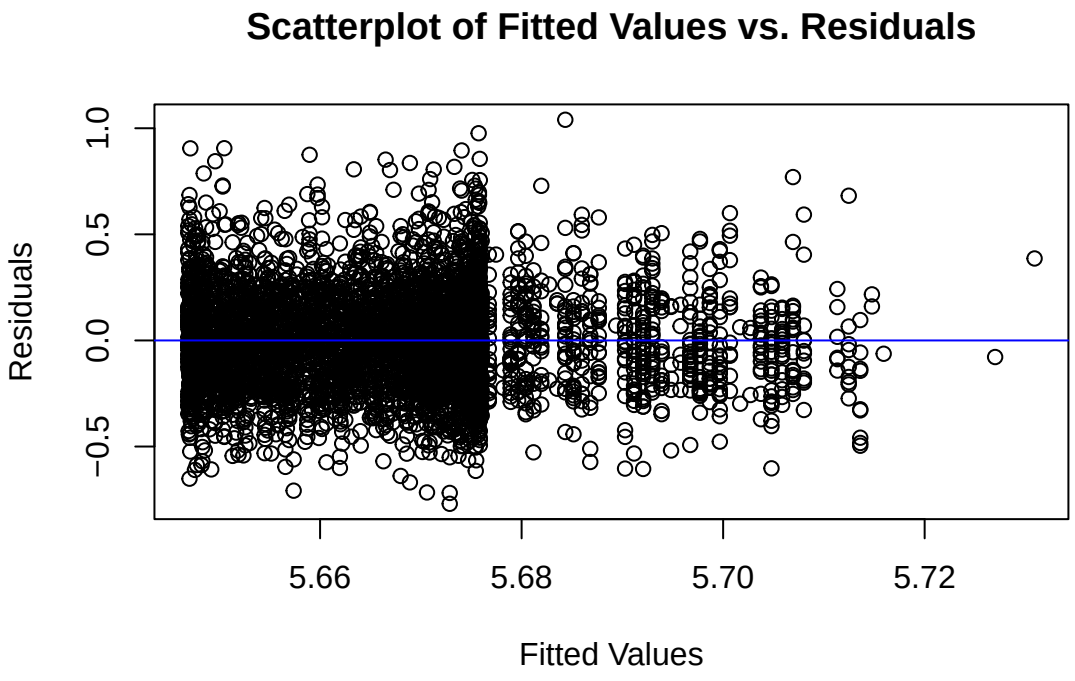


From the plot of the date of the house sale vs. the price of the house, it seems like the data follows a cubic trend. Let's plot the date of the house sale vs. the price of the house, alongside the fitted lines. Let's also plot the fitted residuals.

```
# fit cubic model
fit_saledate <- lm(price~poly(saledate, 3), data=dat)

# plot date of the house sale vs. price of house
plot(dat$saledate, dat$price, xlab="Date of House Sales (days from 1/1/1970)",
     ylab="House Price", main="Scatterplot of House Date of Sales vs. House Price")

# predict prices
predicted_prices_saledate <- predict(
  fit_saledate,
  newdata = data.frame(
    saledate = seq(min(dat$saledate),
                   max(dat$saledate),
                   length.out = 100
    )
  )
)

# add lines to plot
lines(seq(min(dat$saledate),
          max(dat$saledate),
          length.out = 100),
      predicted_prices_saledate, col = "green"
)
```




```
# plot fitted residuals
plot(fit_saledate$fitted.values, fit_saledate$residuals, xlab="Fitted Values",
     ylab="Residuals", main="Scatterplot of Fitted Values vs. Residuals")
abline(h=0, col="blue")
```



From the scatterplot of the house date of sales vs. the house price, we can see that the fitted line seems to match the trend of the datapoints quite nicely. From the scatterplot of fitted values vs. residuals, we can see that the expected value of the residuals is approximately 0, since for each fitted value, there seems to be as many datapoints above the horizontal blue line as there are below. The variance of the residuals does not seem constant, since the spread of the residuals seems to decrease from fitted value 5.64 to fitted value 5.66, and then increase again from 5.66 to 5.68. This cannot be handled appropriately since we cannot transform the response variate. Finally, there are no noticable outliers.

Number of Bedrooms

Here is a histogram for the number of bedrooms per house.

```
hist(dat$bedrm, xlab="Number of Bedrooms",
     main="Histogram of Number of Bedrooms per House")
```

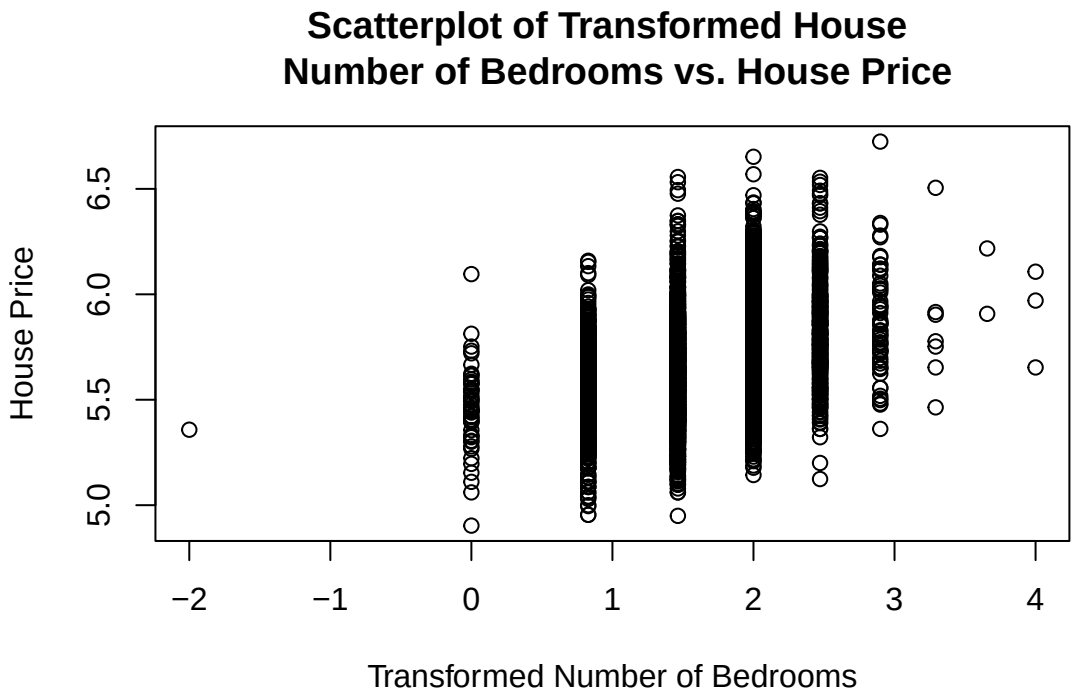
Since this histogram does not appear to be normal nor uniform, it follows that a boxcox transformation is needed. To do this, we will use the boxcox library function, find its maximum likelihood, and then use that value to transform the variate using the function defined before.

```
bedrm_modified <- dat$bedrm + 1
bc_bedrm <- MASS::boxcox(lm(bedrm_modified~dat$price), plotit=FALSE)
print(paste0("Maximum Likelihood: ", bc_bedrm$x[which.max(bc_bedrm$y)]))
```

```
## [1] "Maximum Likelihood: 0.5"
```

```
dat$bedrm <- boxcox_transform(dat$bedrm, bc_bedrm$x[which.max(bc_bedrm$y)])
```

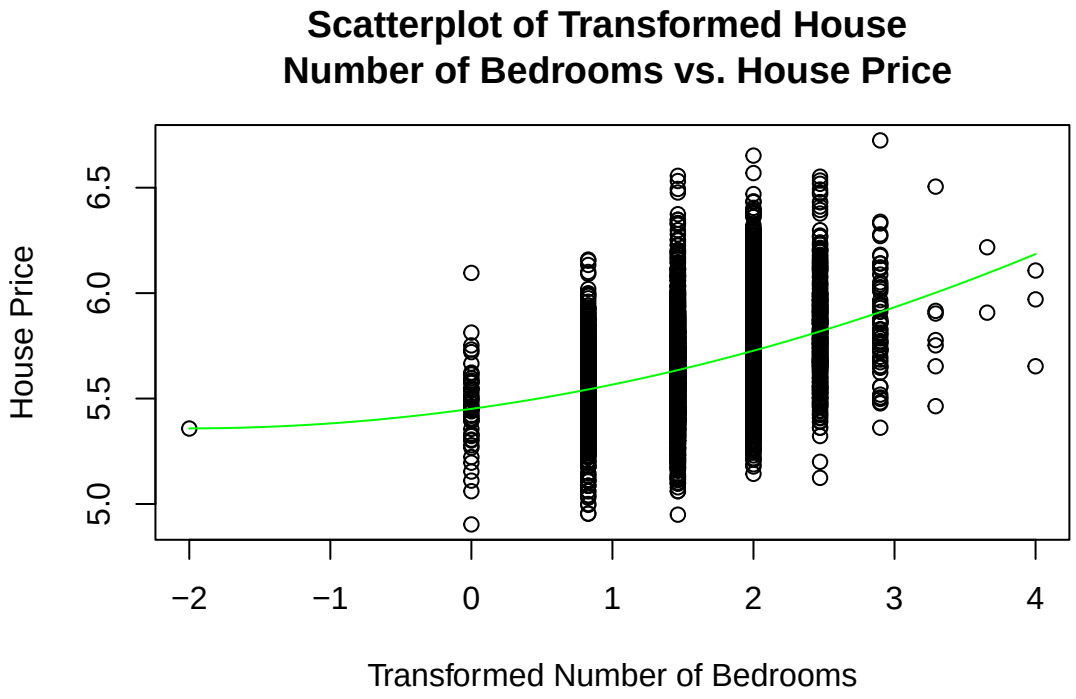
```r
# plot number of bedrooms per house vs. price of house
plot(dat$bedrm, dat$price, xlab="Transformed Number of Bedrooms",
     ylab="House Price",
     main="Scatterplot of Transformed House \n Number of Bedrooms vs. House Price")
```



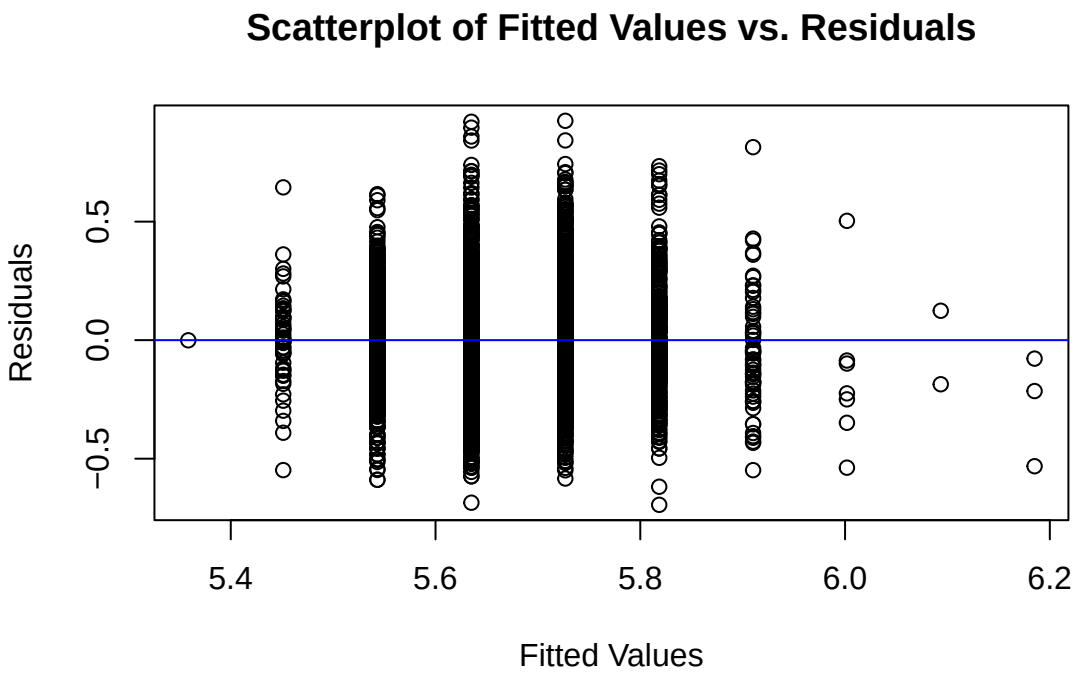
From the plot of the transformed number of bedrooms per house vs. the price of the house, it seems like the data follows a quadratic trend. Let's plot the transformed number of bedrooms per house vs. the price of the house, alongside the fitted lines. Let's also plot the fitted residuals.

```r
# fit quadratic model
fit_bedrm <- lm(price~poly(bedrm, 2), data=dat)

# plot number of bedrooms per house vs. price of house
plot(dat$bedrm, dat$price, xlab="Transformed Number of Bedrooms",
     ylab="House Price",
     main="Scatterplot of Transformed House \n Number of Bedrooms vs. House Price")

# predict prices
predicted_prices_bedrm <- predict(
  fit_bedrm,
  newdata = data.frame(
    bedrm = seq(min(dat$bedrm),
                max(dat$bedrm),
                length.out = 100
    )
  )
)

# add lines to plot
```

```
lines(seq(min(dat$bedrm),
          max(dat$bedrm),
          length.out = 100),
      predicted_prices_bedrm, col = "green"
)
```




```
# plot fitted residuals
plot(fit_bedrm$fitted.values, fit_bedrm$residuals, xlab="Fitted Values",
     ylab="Residuals", main="Scatterplot of Fitted Values vs. Residuals")
abline(h=0, col="blue")
```

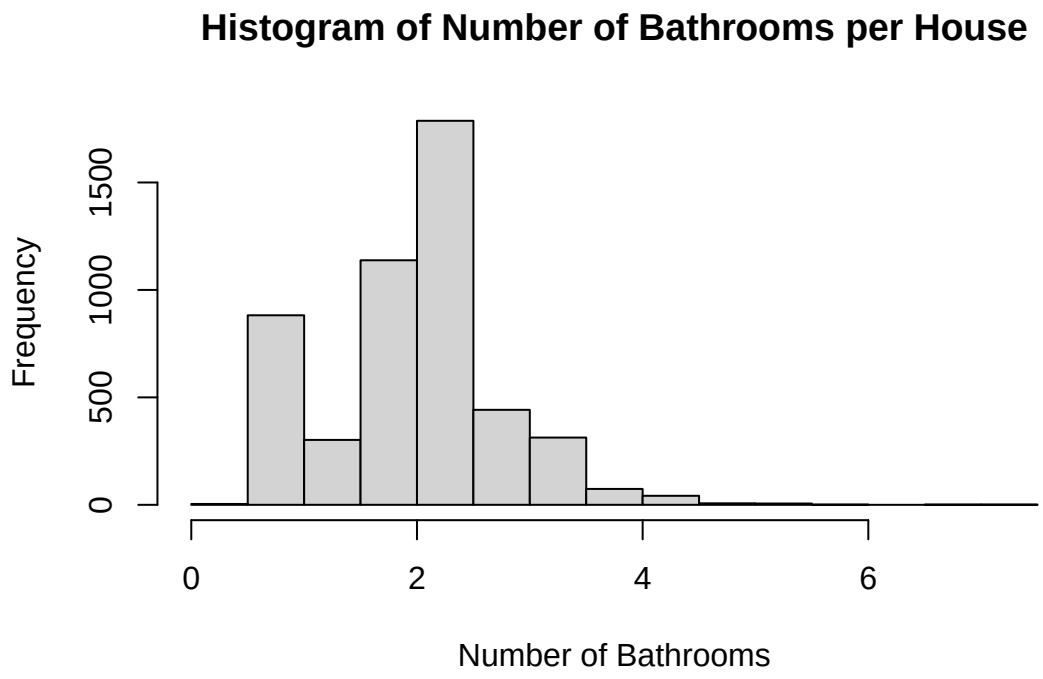


From the scatterplot of the transformed house number of bedrooms vs. the house price, we can see that the fitted line seems to match the trend of the datapoints moderately. From the scatterplot of fitted values vs. residuals, we can see that the expected value of the residuals is approximately 0, since for each fitted value, there seems to be as many datapoints above the horizontal blue line as there are below. The leftmost point and the rightmost points are not concerning since they only make up less than 100 out of the 5000 total datapoints that are captured at fitted value ranges between 5.4 and 5.95, and those data points seem to have expected residual around 0 for each fitted value. The variance of the residuals does not seem constant, since the spread of the residuals seems to increase from fitted value 5.4 to fitted value 5.75, and then decrease again from 5.75 to 6.0. This cannot be handled appropriately since we cannot transform the response variate. Finally, there are no noticable outliers.

Number of Bathrooms

Here is a histogram for the number of bathrooms per house.

```
hist(dat$bathrm, xlab="Number of Bathrooms",
     main="Histogram of Number of Bathrooms per House")
```

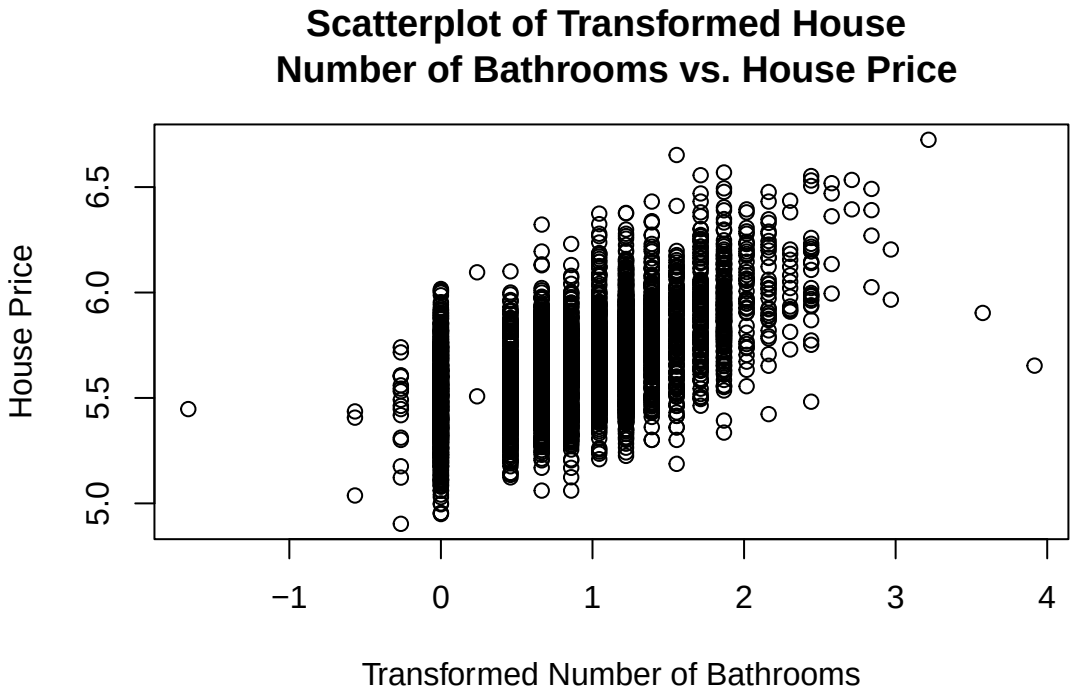


Since this histogram does not appear to be normal nor uniform, it follows that a boxcox transformation is needed. To do this, we will use the boxcox library function, find its maximum likelihood, and then use that value to transform the variate using the function defined before.

```
bathrm_modified <- dat$bathrm + 1
bc_bathrm <- MASS::boxcox(lm(bathrm_modified~dat$price), plotit=FALSE)
print(paste0("Maximum Likelihood: ", bc_bathrm$x[which.max(bc_bathrm$y)]))
```

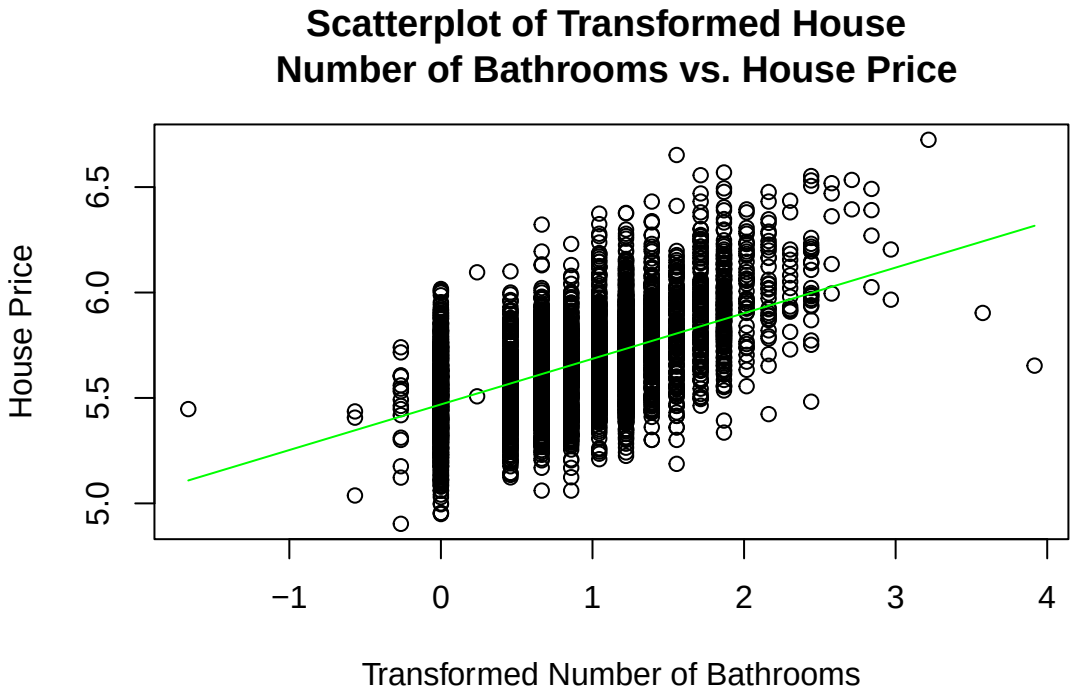
```
## [1] "Maximum Likelihood: 0.6"
```

```
dat$bathrm <- boxcox_transform(dat$bathrm, bc_bathrm$x[which.max(bc_bathrm$y)])

# plot number of bathrooms per house vs. price of house
plot(dat$bathrm, dat$price, xlab="Transformed Number of Bathrooms",
     ylab="House Price",
     main="Scatterplot of Transformed House \n Number of Bathrooms vs. House Price")
```

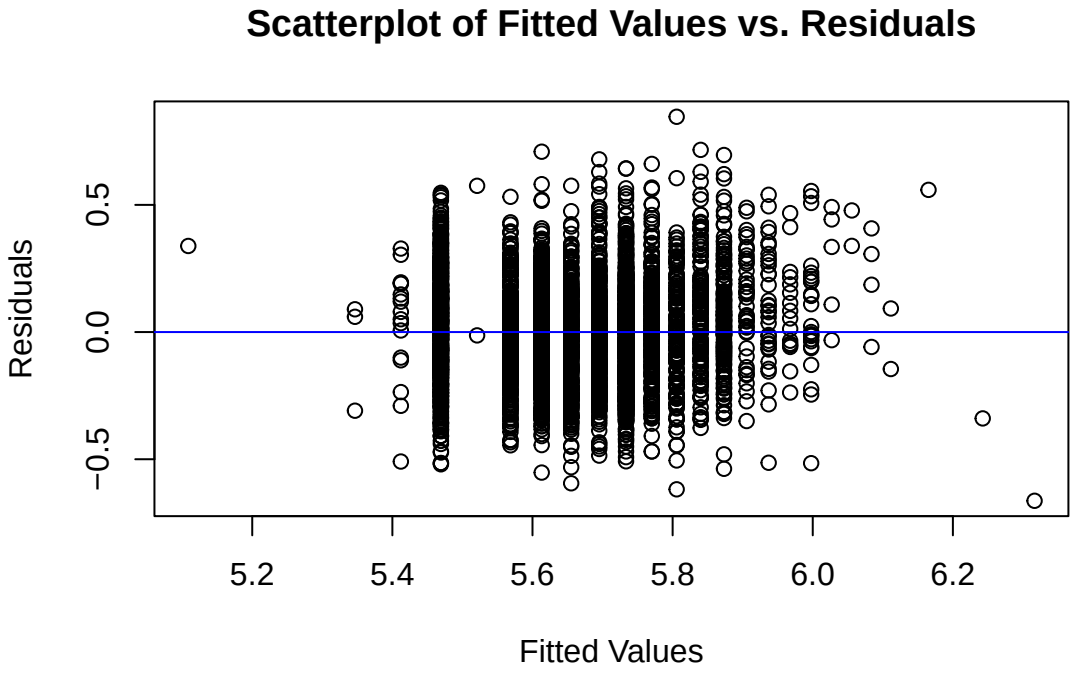


From the plot of the transformed number of bathrooms per house vs. the price of the house, it seems like the data follows a linear trend. Let's plot the transformed number of bathrooms per house vs. the price of the house, alongside the fitted lines. Let's also plot the fitted residuals.

```
# fit linear model
fit_bathrm <- lm(price~bathrm, data=dat)

# plot number of bathrooms per house vs. price of house
plot(dat$bathrm, dat$price, xlab="Transformed Number of Bathrooms",
     ylab="House Price",
     main="Scatterplot of Transformed House \n Number of Bathrooms vs. House Price")

# predict prices
predicted_prices_bathrm <- predict(
  fit_bathrm,
  newdata = data.frame(
    bathrm = seq(min(dat$bathrm),
                 max(dat$bathrm),
                 length.out = 100
    )
  )
)

# add lines to plot
lines(seq(min(dat$bathrm),
          max(dat$bathrm),
          length.out = 100),
      predicted_prices_bathrm, col = "green"
)
```




```r
# plot fitted residuals
plot(fit_bathrm$fitted.values, fit_bathrm$residuals, xlab="Fitted Values",
     ylab="Residuals", main="Scatterplot of Fitted Values vs. Residuals")
abline(h=0, col="blue")
```

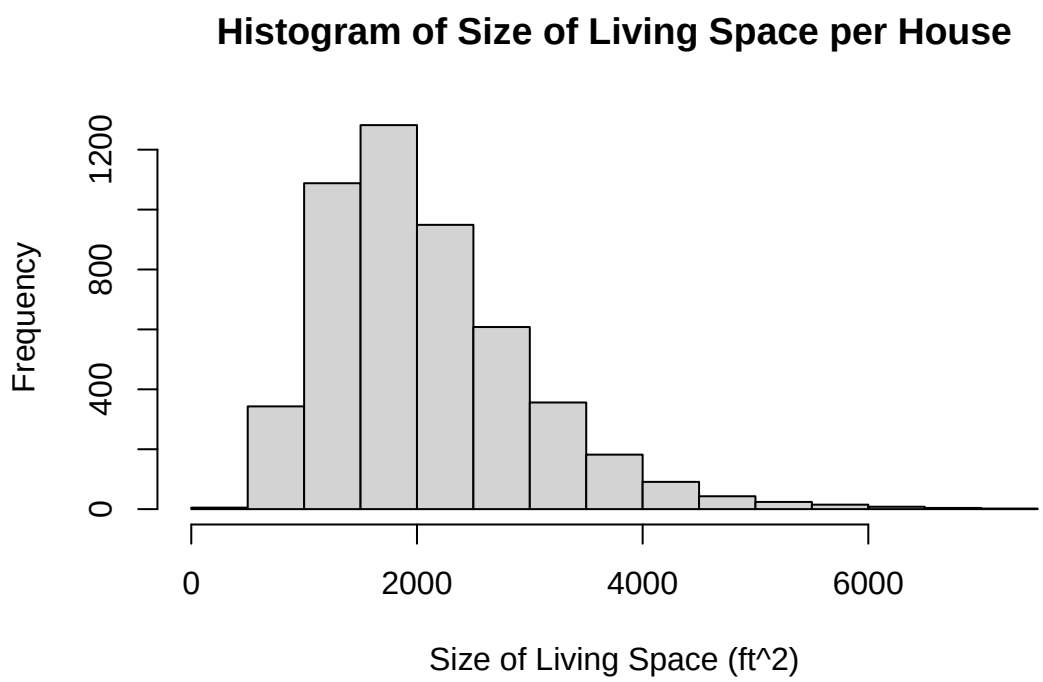


From the scatterplot of the transformed house number of bathrooms vs. the house price, we can see that the fitted line seems to match the trend of the datapoints moderately. From the scatterplot of fitted values vs. residuals, we can see that the expected value of the residuals is approximately 0, since for each fitted value, there seems to be as many datapoints above the horizontal blue line as there are below. The leftmost points and the rightmost points are not concerning since they only account for a very small proportion of the entire dataset; most of the dataset is captured at fitted value ranges between 5.4 and 5.95, and those data points seem to have expected residual around 0 for each fitted value. The variance of the residuals does not seem constant, since the spread of the residuals seems to decrease from fitted value 5.45 to fitted value 5.6, and then increase again from 5.6 to 5.7. This cannot be handled appropriately since we cannot transform the response variate. Finally, there are no noticable outliers besides the one at fitted value less than 5.2, but that point doesn't have much influence on the fitted model since it's not that far off from the rest of the points in terms of trend, and it's only one point.

Size of Living Space in Square Feet

Here is a histogram for the size of living space in square feet per house.

```
hist(dat$sizeLiving, xlab="Size of Living Space (ft^2)",
     main="Histogram of Size of Living Space per House")
```

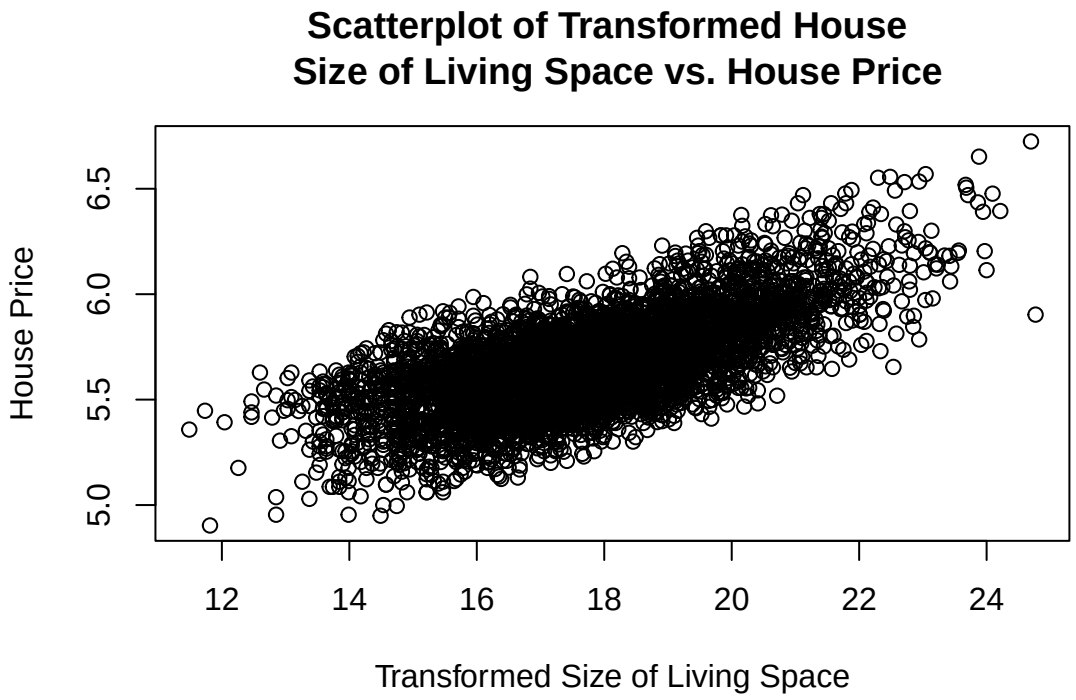


Since this histogram does not appear to be normal nor uniform, it follows that a boxcox transformation is needed. To do this, we will use the boxcox library function, find its maximum likelihood, and then use that value to transform the variate using the function defined before.

```
bc_sizeLiving <- MASS::boxcox(lm(sizeLiving~price, data=dat), plotit=FALSE)
print(paste0("Maximum Likelihood: ", bc_sizeLiving$x[which.max(bc_sizeLiving$y)]))
```

```
## [1] "Maximum Likelihood: 0.2"
```

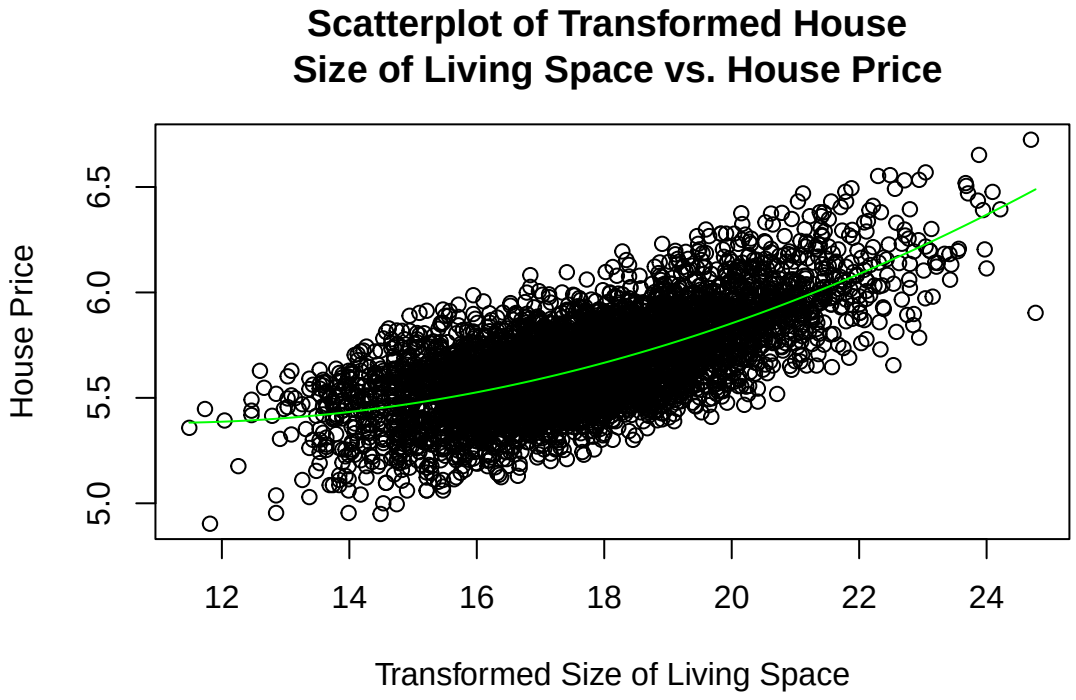
```r
dat$sizeLiving <- boxcox_transform(dat$sizeLiving, bc_sizeLiving$x[which.max(bc_sizeLiving$y)])

# plot size of living space vs. price
plot(dat$sizeLiving, dat$price, xlab="Transformed Size of Living Space",
     ylab="House Price",
     main="Scatterplot of Transformed House \n Size of Living Space vs. House Price")
```



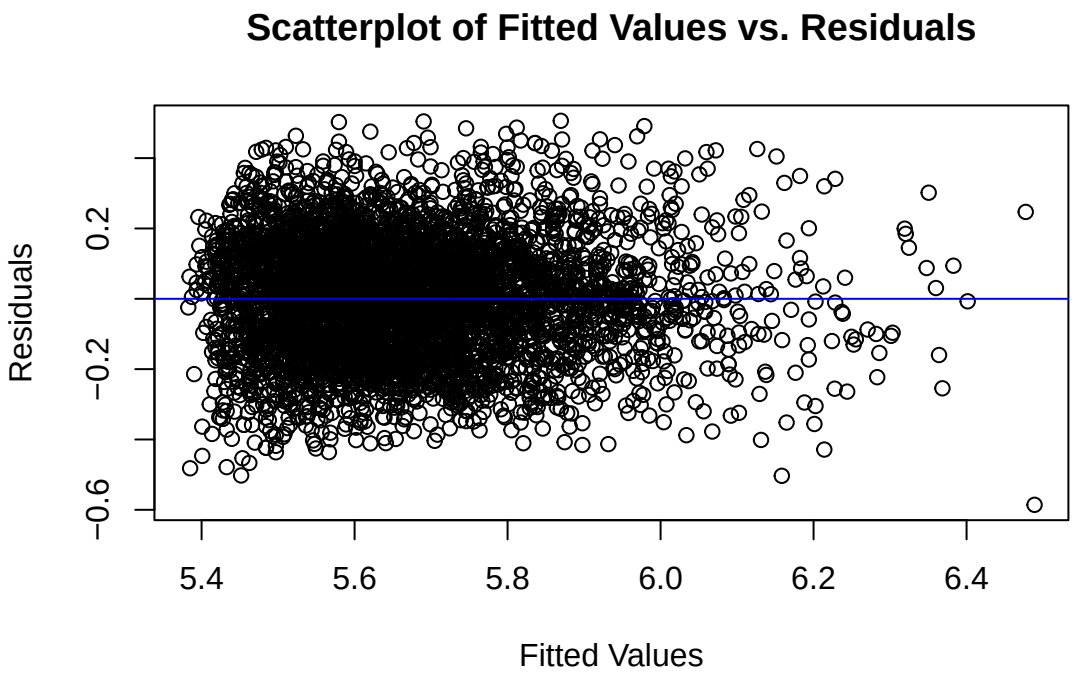
From the plot of the transformed size of living space vs. price, it seems like the data follows a quadratic trend. Let's plot the transformed size of living space vs. price, alongside the fitted lines. Let's also plot the fitted residuals.

```r
# fit quadratic model
fit_sizeLiving <- lm(price~poly(sizeLiving, 2), data=dat)

# plot size of living space vs. price
plot(dat$sizeLiving, dat$price, xlab="Transformed Size of Living Space",
     ylab="House Price",
     main="Scatterplot of Transformed House \n Size of Living Space vs. House Price")

# predict prices
predicted_prices_sizeLiving <- predict(
  fit_sizeLiving,
  newdata = data.frame(
    sizeLiving = seq(min(dat$sizeLiving),
                     max(dat$sizeLiving),
                     length.out = 100
    )
  )
)
```

```r
# add lines to plot
lines(seq(min(dat$sizeLiving),
          max(dat$sizeLiving),
          length.out = 100),
      predicted_prices_sizeLiving, col = "green"
)
```




```r
# plot fitted residuals
plot(fit_sizeLiving$fitted.values, fit_sizeLiving$residuals, xlab="Fitted Values",
     ylab="Residuals", main="Scatterplot of Fitted Values vs. Residuals")
abline(h=0, col="blue")
```

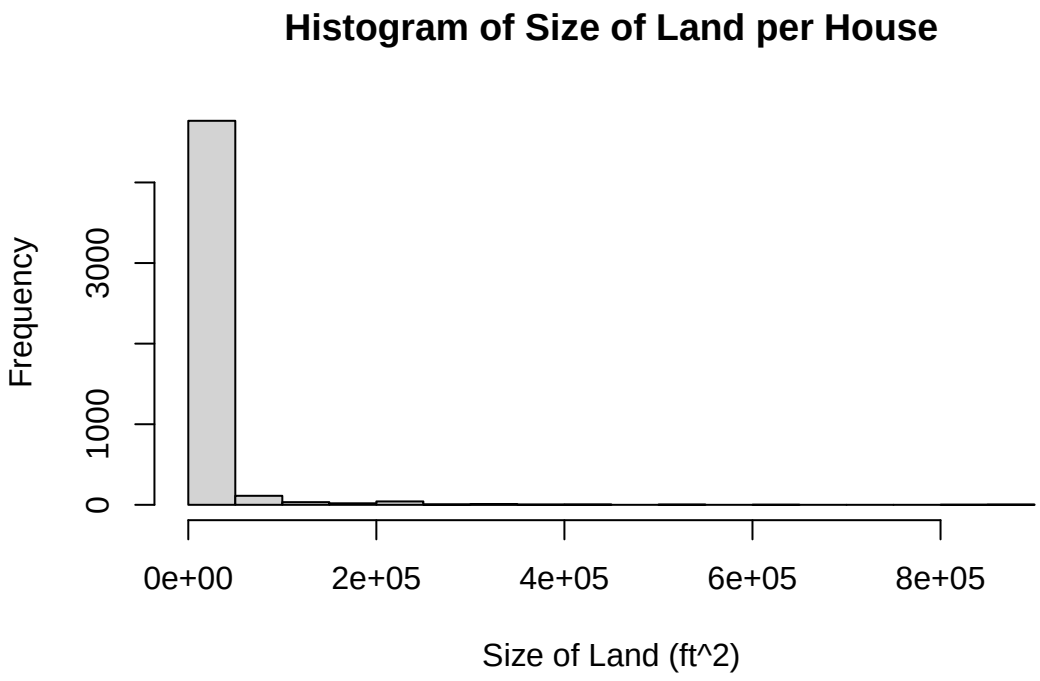


From the scatterplot of the transformed house size of living space vs. the house price, we can see that the fitted line seems to match the trend of the datapoints quite well. From the scatterplot of fitted values vs. residuals, we can see that the expected value of the residuals is approximately 0, since for each fitted value, there seems to be as many datapoints above the horizontal blue line as there are below. The rightmost points are not concerning since they only account for a very small proportion of the entire dataset; most of the dataset is captured at fitted value ranges between 5.4 and 5.95, and those data points seem to have expected residual around 0 for each fitted value. The variance of the residuals here do seem constant, since the spread of the residuals seems to be similar for each fitted value. Finally, there are no noticable outliers besides the rightmost point, but that point doesn't have much influence on the fitted model since it's not that far off from the rest of the points in terms of trend, and it's only one point.

Size of Lot Space in Square Feet

Here is a histogram for the size of land in square feet per house.

```
hist(dat$sizeLot, xlab="Size of Land (ft^2)",
     main="Histogram of Size of Land per House")
```

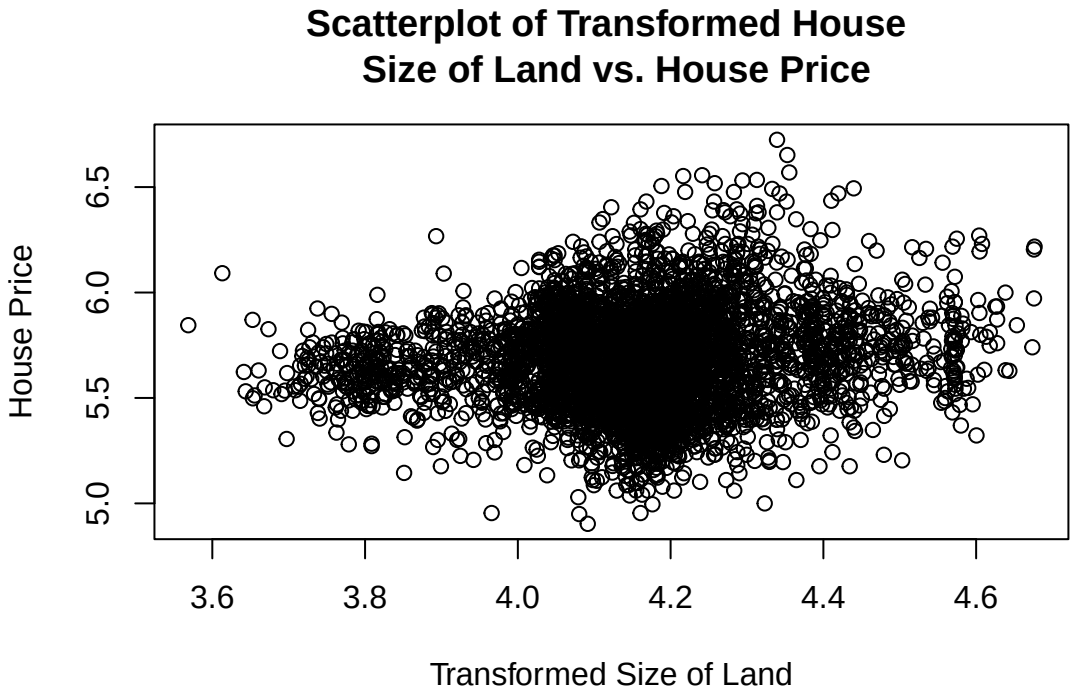


Since this histogram does not appear to be normal nor uniform, it follows that a boxcox transformation is needed. To do this, we will use the boxcox library function, find its maximum likelihood, and then use that value to transform the variate using the function defined before.

```
bc_sizeLot <- MASS::boxcox(lm(sizeLot~price, data=dat), plotit=FALSE)
print(paste0("Maximum Likelihood: ", bc_sizeLot$x[which.max(bc_sizeLot$y)]))
```

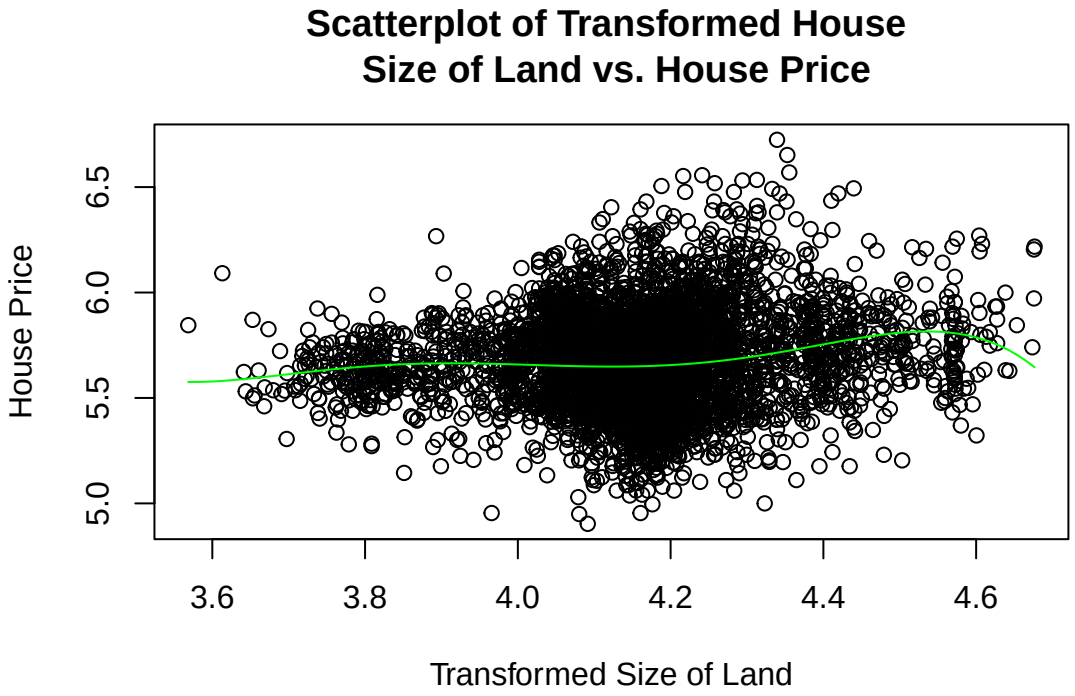
```
## [1] "Maximum Likelihood: -0.2"
```

```
dat$sizeLot <- boxcox_transform(dat$sizeLot, bc_sizeLot$x[which.max(bc_sizeLot$y)])

# plot size of land vs. price
plot(dat$sizeLot, dat$price, xlab="Transformed Size of Land",
     ylab="House Price",
     main="Scatterplot of Transformed House \n Size of Land vs. House Price")
```

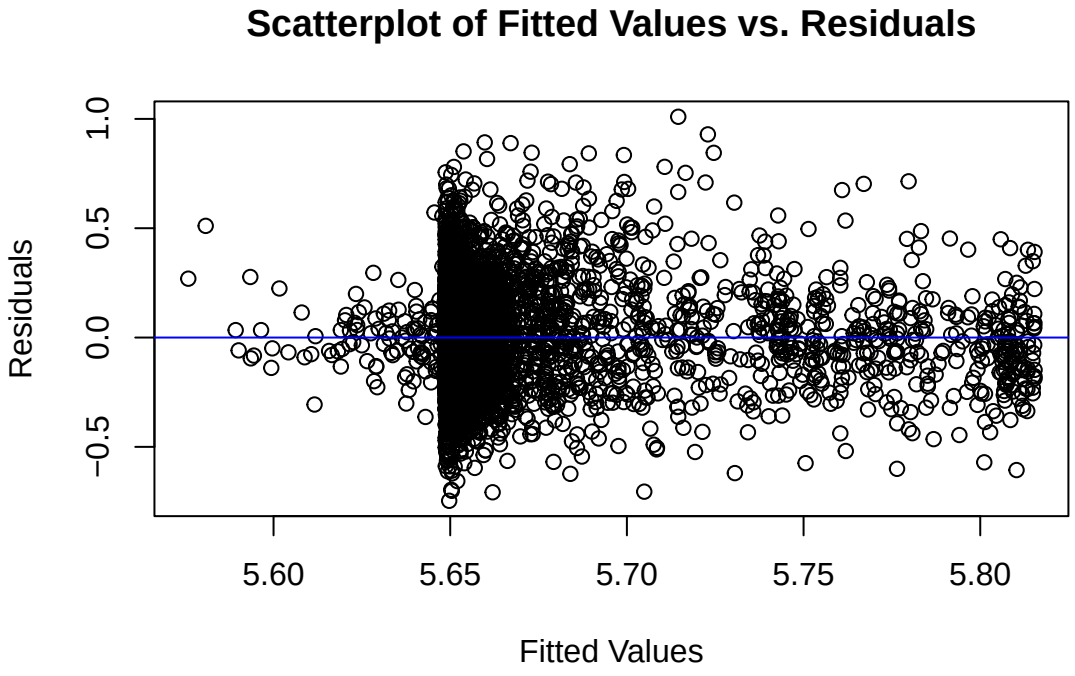


From the plot of the transformed size of land vs. price, it seems like the data follows a quintic trend. Let's plot the transformed size of land vs. price, alongside the fitted lines. Let's also plot the fitted residuals.

```
# fit quintic model
fit_sizeLot <- lm(price~poly(sizeLot, 5), data=dat)

# plot size of land vs. price
plot(dat$sizeLot, dat$price, xlab="Transformed Size of Land",
     ylab="House Price",
     main="Scatterplot of Transformed House \n Size of Land vs. House Price")

# predict prices
predicted_prices_sizeLot <- predict(
  fit_sizeLot,
  newdata = data.frame(
    sizeLot = seq(min(dat$sizeLot),
                  max(dat$sizeLot),
                  length.out = 100
    )
  )
)

# add lines to plot
lines(seq(min(dat$sizeLot),
          max(dat$sizeLot),
          length.out = 100),
      predicted_prices_sizeLot, col = "green"
)
```




```r
# plot fitted residuals
plot(fit_sizeLot$fitted.values, fit_sizeLot$residuals, xlab="Fitted Values",
     ylab="Residuals", main="Scatterplot of Fitted Values vs. Residuals")
abline(h=0, col="blue")
```

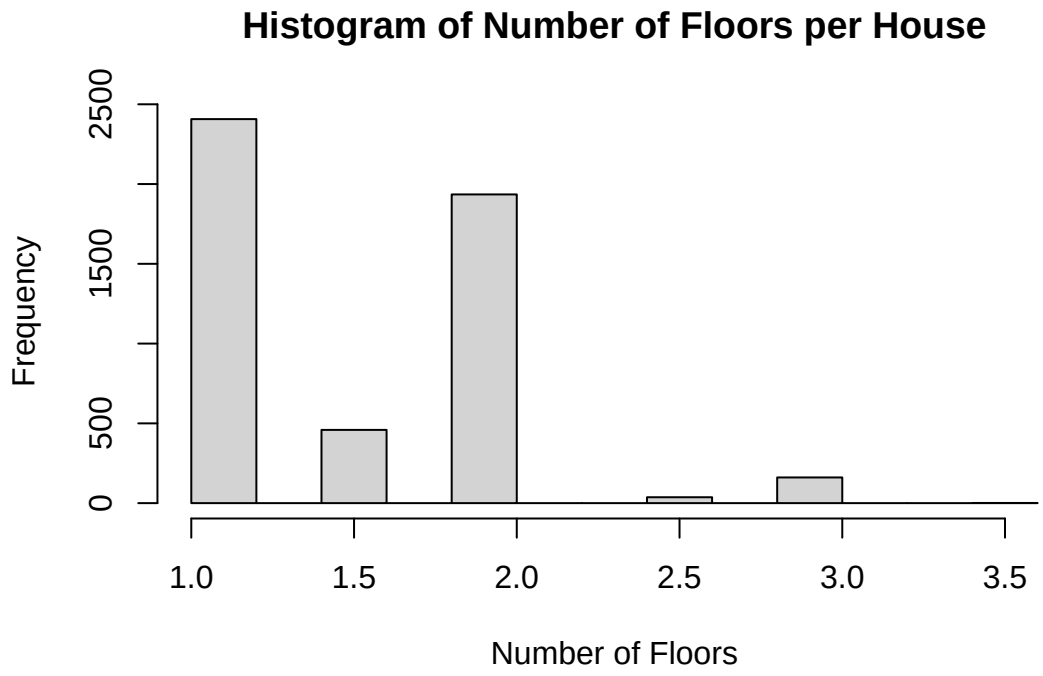


From the scatterplot of the transformed house size of land vs. the house price, we can see that the fitted line seems to match the trend of the datapoints quite well. From the scatterplot of fitted values vs. residuals, we can see that the expected value of the residuals is approximately 0, since for each fitted value, there seems to be as many datapoints above the horizontal blue line as there are below. The leftmost points (fitted values below 5.625) and the rightmost points (fitted values above 5.8) are not concerning since they only account for a very small proportion of the entire dataset; most of the dataset is captured at fitted value ranges between 5.625 and 5.8, and those data points seem to have expected residual around 0 for each fitted value. The variance of the residuals does not seem constant, since the spread of the residuals seems to decrease from fitted value 5.65 to fitted value 5.7. This cannot be handled appropriately since we cannot transform the response variate. Finally, there are no noticable outliers besides the two at fitted values less than 5.55, but those two points don't have much influence on the fitted model since it's not that far off from the rest of the points in terms of trend, and it's only two points.

Number of Floors

Here is a histogram for the number of floors per house.

```
hist(dat$floors, xlab="Number of Floors",
     main="Histogram of Number of Floors per House")
```

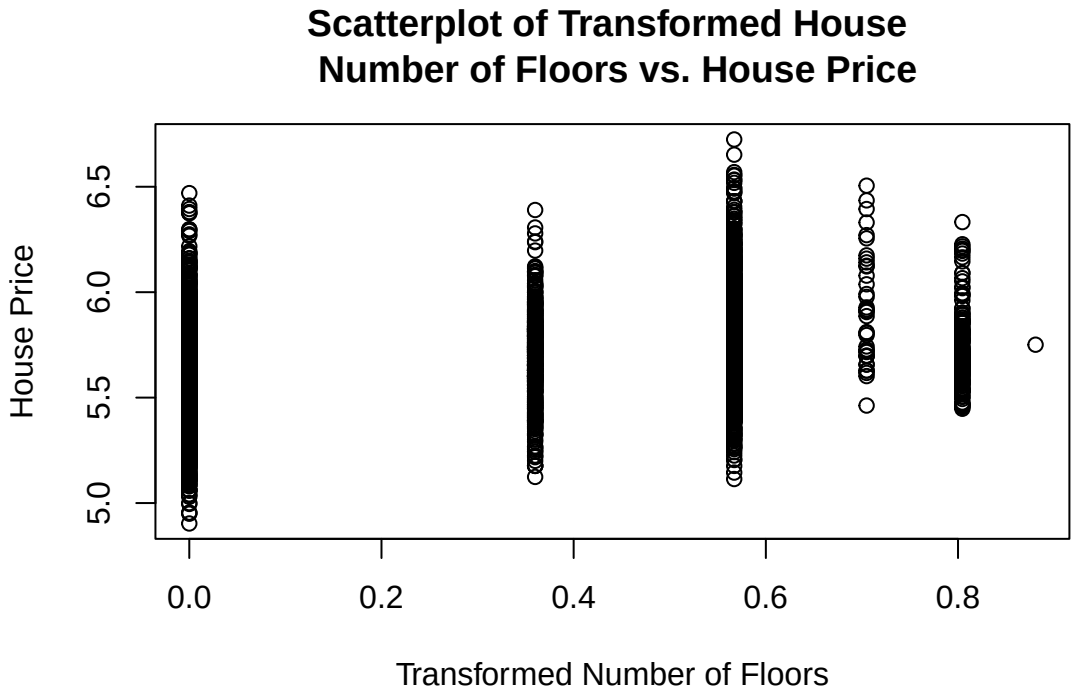


Since this histogram does not appear to be normal nor uniform, it follows that a boxcox transformation is needed. To do this, we will use the boxcox library function, find its maximum likelihood, and then use that value to transform the variate using the function defined before.

```
bc_floors <- MASS::boxcox(lm(floors~price, data=dat), plotit=FALSE)
print(paste0("Maximum Likelihood: ", bc_floors$x[which.max(bc_floors$y)]))
```

```
## [1] "Maximum Likelihood: -0.6"
```

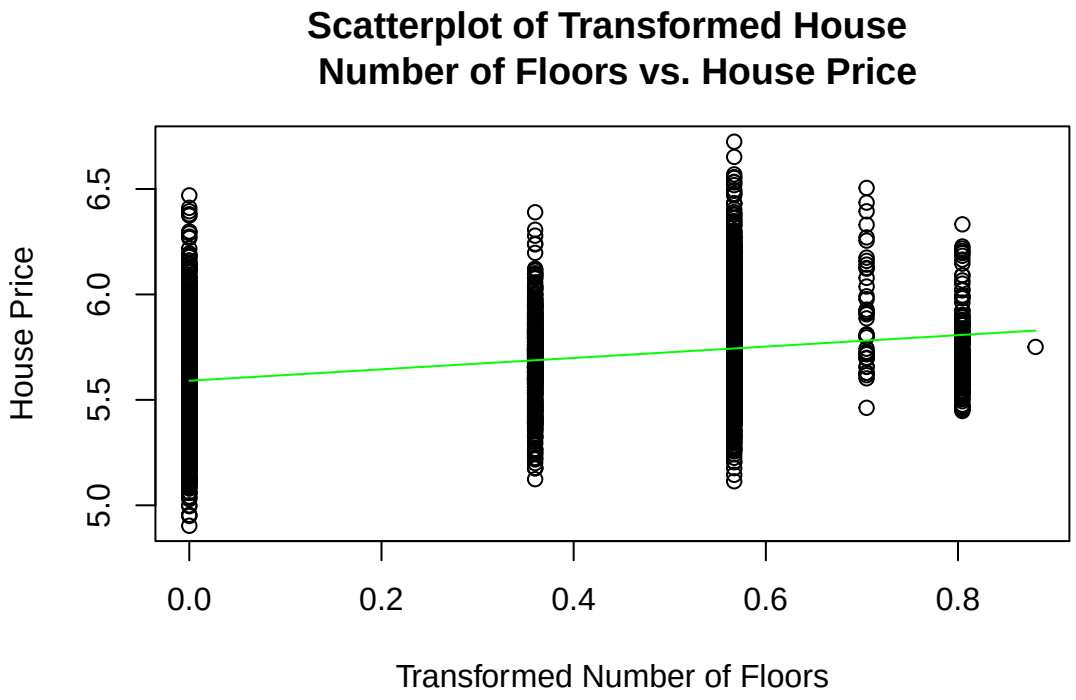
```r
dat$floors <- boxcox_transform(dat$floors, bc_floors$x[which.max(bc_floors$y)])

# plot number of floors vs. price
plot(dat$floors, dat$price, xlab="Transformed Number of Floors",
     ylab="House Price",
     main="Scatterplot of Transformed House \n Number of Floors vs. House Price")
```



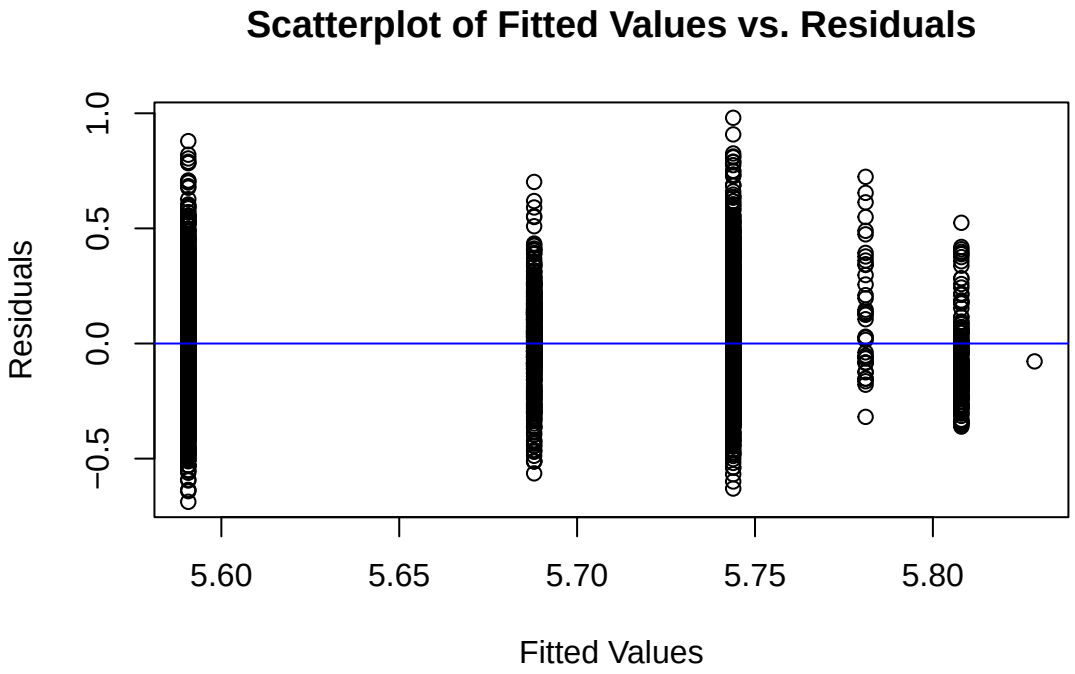
From the plot of the transformed number of floors vs. price, it seems like the data follows a linear trend. Let's plot the transformed number of floors vs. price, alongside the fitted lines. Let's also plot the fitted residuals.

```r
# fit linear model
fit_floors <- lm(price~floors, data=dat)

# plot number of floors vs. price
plot(dat$floors, dat$price, xlab="Transformed Number of Floors",
     ylab="House Price",
     main="Scatterplot of Transformed House \n Number of Floors vs. House Price")

# predict prices
predicted_prices_floors <- predict(
  fit_floors,
  newdata = data.frame(
    floors = seq(min(dat$floors),
                 max(dat$floors),
                 length.out = 100
    )
  )
)
```

```r
# add lines to plot
lines(seq(min(dat$floors),
          max(dat$floors),
          length.out = 100),
      predicted_prices_floors, col = "green"
)
```




```r
# plot fitted residuals
plot(fit_floors$fitted.values, fit_floors$residuals, xlab="Fitted Values",
     ylab="Residuals", main="Scatterplot of Fitted Values vs. Residuals")
abline(h=0, col="blue")
```



From the scatterplot of the transformed house number of floors vs. the house price, we can see that the fitted line seems to match the trend of the datapoints moderately well. From the scatterplot of fitted values vs. residuals, we can see that the expected value of the residuals is approximately 0, since for each fitted value, there seems to be as many datapoints above the horizontal blue line as there are below. The leftmost point and the points between fitted value 5.75 and 5.8 are not concerning since they only account for a very small proportion of the entire dataset; most of the dataset is captured at other fitted values, and those data points seem to have expected residual around 0 for each fitted value. The variance of the residuals does not seem constant, since the spread of the residuals seems to decrease from fitted value below 5.6 to fitted value 5.7, and then increase again from fitted value 5.7 to fitted value 5.75. This cannot be handled appropriately since we cannot transform the response variate. Finally, there are no noticable outliers.

Condition of the House

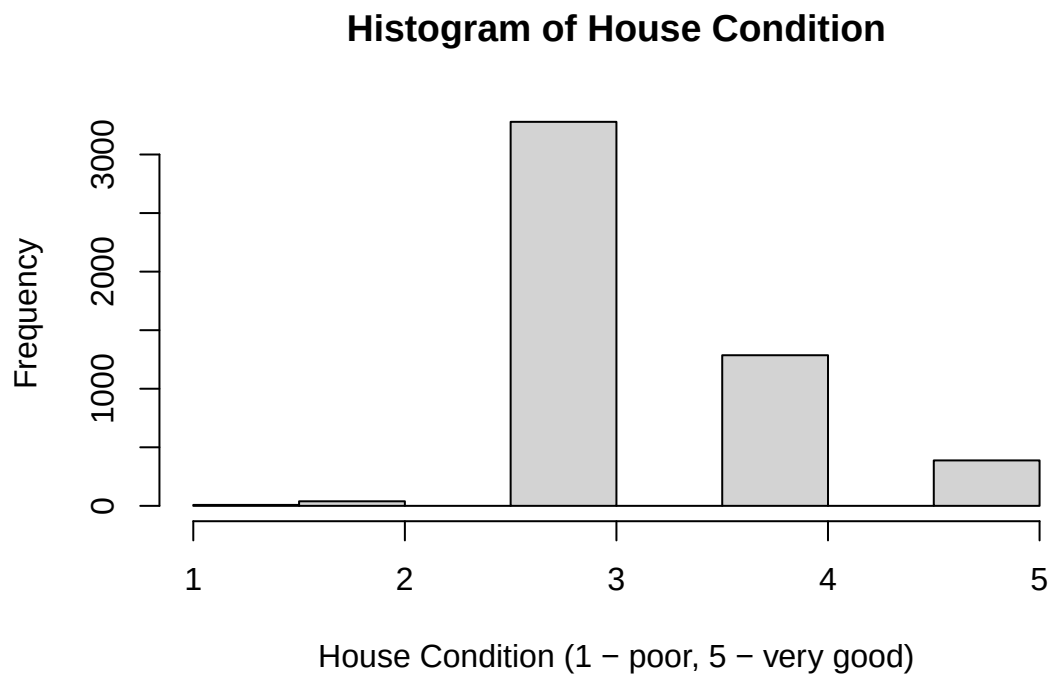
Let's encode the variate for the condition of the house to be a numerical variate from 1 to 5. This is because there might be some linear relationship between the condition of the house, and the price of the house.

```
# encode condition of house
encode_condition <- function(x)
{
  if (x == "poor") {
    return(0);
  } else if (x == "fair") {
    return(1);
  } else if (x == "average") {
    return(2);
  } else if (x == "good") {
    return(3);
  } else if (x == "very_good") {
    return(4);
```

```
  }
}
condition_encoded <- sapply(dat$condition, encode_condition) + 1
```

Here is a histogram for the condition of the houses.

```
hist(condition_encoded, xlab="House Condition (1 - poor, 5 - very good)",
     main="Histogram of House Condition")
```

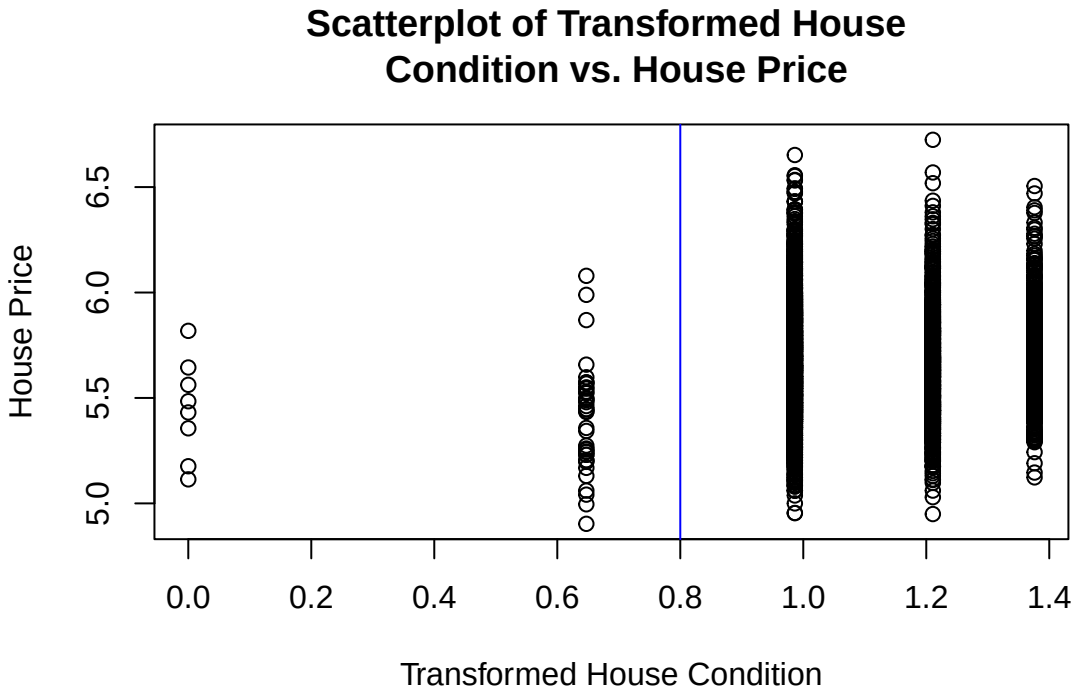


Since this histogram does not appear to be normal nor uniform, it follows that a boxcox transformation is needed. To do this, we will use the boxcox library function, find its maximum likelihood, and then use that value to transform the variate using the function defined before.

```
bc_condition <- MASS::boxcox(lm(condition_encoded~dat$price),
                             lambda = seq(-2, 2, 1/10),
                             plotit=FALSE)
print(paste0("Maximum Likelihood: ", bc_condition$x[which.max(bc_condition$y)]))
```

```
## [1] "Maximum Likelihood: -0.2"
```

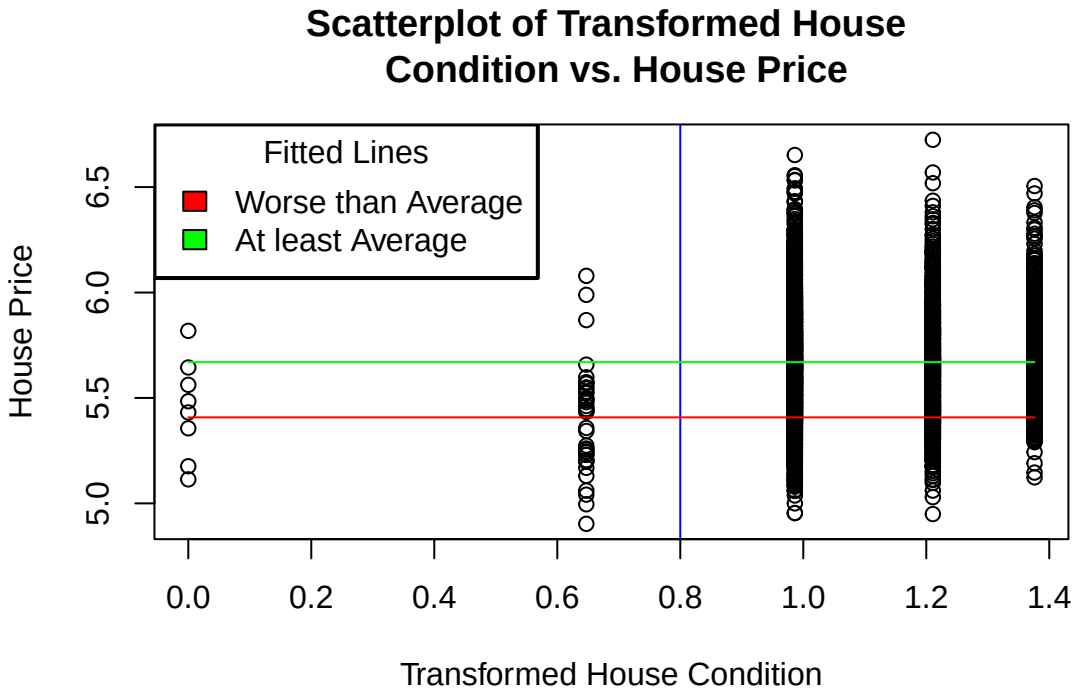
```
dat$condition <- boxcox_transform(condition_encoded, bc_condition$x[which.max(bc_condition$y)])

# plot house condition vs. price
plot(dat$condition, dat$price, xlab="Transformed House Condition",
     ylab="House Price",
     main="Scatterplot of Transformed House \n Condition vs. House Price")
abline(v=0.8, col="blue")
```



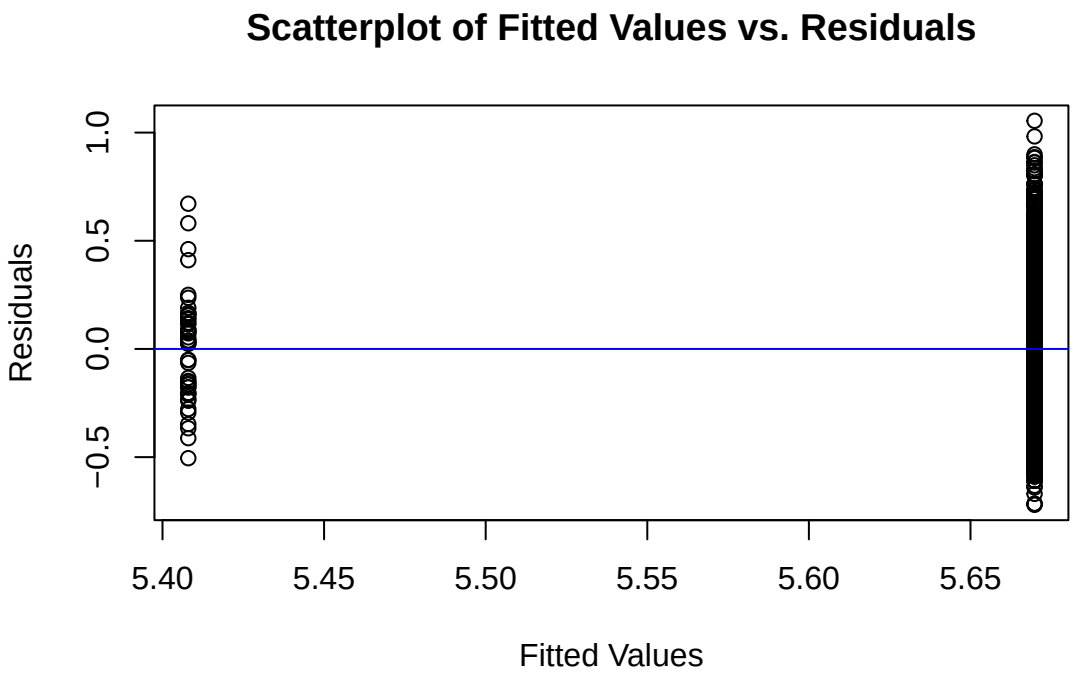
From the plot of the transformed house condition vs. price, it seems like there is a cutoff at $x = 0.8$, where the data follows a straight line with slope 0 with a certain intercept when $x < 0.8$, and follows a straight line with a different intercept when $x > 0.8$. Because of this, let's create another predictor `condition_at_least_average`. This is an indicator of whether the condition of the house is at least average. This means an encoded score of at least 2. Let's plot the transformed house condition vs. price, alongside the fitted lines. Let's also plot the fitted residuals.

```
# define new predictor for the house condition at least average
dat$condition_at_least_average <- factor(sapply(dat$condition, function(x) {
  if (x <= 0.8) 0 else 1
}))

# fit horizontal model
fit_condition <- lm(price~condition_at_least_average, data=dat)

# plot house condition vs. price
plot(dat$condition, dat$price, xlab="Transformed House Condition",
     ylab="House Price",
     main="Scatterplot of Transformed House \n Condition vs. House Price")
abline(v=0.8, col="blue")

# predict prices when house is below average condition
predicted_prices_below <- predict(
  fit_condition,
  newdata = data.frame(
    condition = seq(min(dat$condition),
                    max(dat$condition),
                    length.out = 100
    ),
```

```
    condition_at_least_average = rep(factor(0), 100)
  )
)

# predict prices when house is at least average condition
predicted_prices_above <- predict(
  fit_condition,
  newdata = data.frame(
    condition = seq(min(dat$condition),
                    max(dat$condition),
                    length.out = 100
    ),
    condition_at_least_average = rep(factor(1), 100)
  )
)

# add lines to plot
lines(seq(min(dat$condition),
          max(dat$condition),
          length.out = 100),
      predicted_prices_below, col = "red"
)

lines(seq(min(dat$condition),
          max(dat$condition),
          length.out = 100),
      predicted_prices_above, col = "green"
)

legend(x = "topleft", box.lwd = 2 , title="Fitted Lines",
       legend=c("Worse than Average", "At least Average"),
       fill = c("red","green"))
```




```
# remove condition variate since we are not using it
dat <- subset(dat, select = -c(condition) )

# plot fitted residuals
plot(fit_condition$fitted.values, fit_condition$residuals, xlab="Fitted Values",
     ylab="Residuals", main="Scatterplot of Fitted Values vs. Residuals")
abline(h=0, col="blue")
```

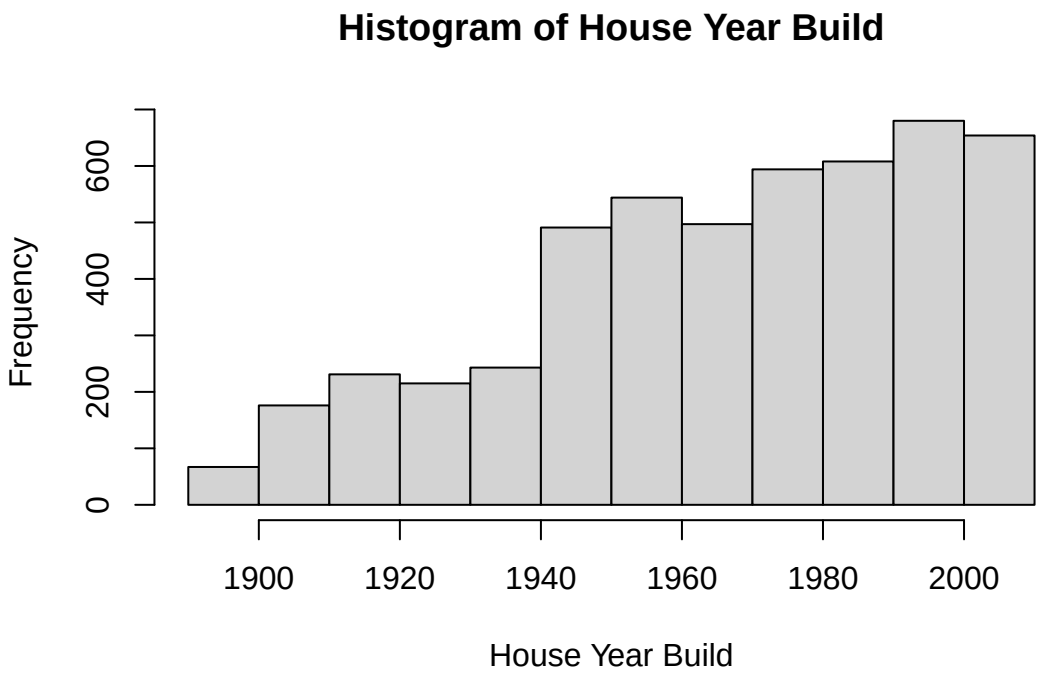


From the scatterplot of the transformed house condition vs. the house price, we can see that the fitted line seems to match the trend of the datapoints moderately well. From the scatterplot of fitted values vs. residuals, we can see that the expected value of the residuals is approximately 0 for both fitted values. For the fitted value at around 5.40, there seems to be as many datapoints above the horizontal blue line as there are below. For the fitted value above 5.65, the range of the datapoints seem to be around -0.75 and 0.75. The variance of the residuals does not seem constant, since the spread of the residuals is much lower for fitted value around 5.4, than the spread of residuals for fitted value above 5.65. This cannot be handled appropriately since we cannot transform the response variate. Finally, there are no noticable outliers.

Year Build

Here is a histogram for the house year build.

```
hist(dat$yrb, xlab="House Year Build",
     main="Histogram of House Year Build")
```



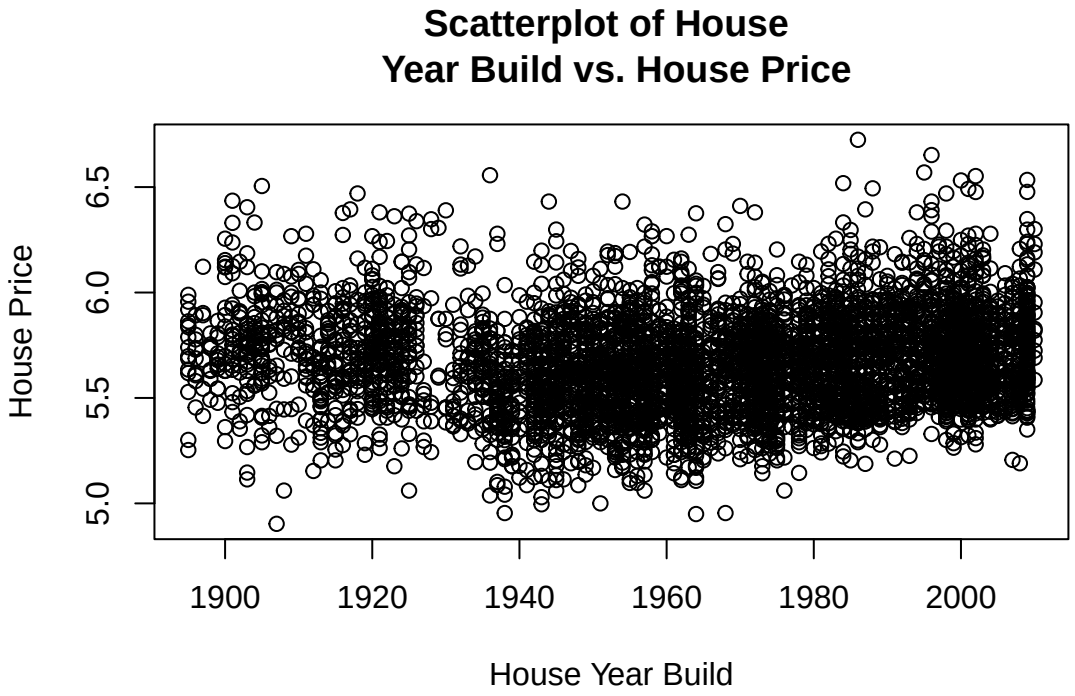
Since this histogram does not appear to be normal nor uniform, it follows that a boxcox transformation is needed. To do this, we will use the boxcox library function, find its maximum likelihood, and then use that value to transform the variate using the function defined before.

```
bc_yrb <- MASS::boxcox(lm(yrb~price, data=dat),
                       lambda = seq(-100, 100, 1/10),
                       plotit=FALSE)
bc_yrb$x[which.max(bc_yrb$y)]
```

```
## [1] 19.2
```

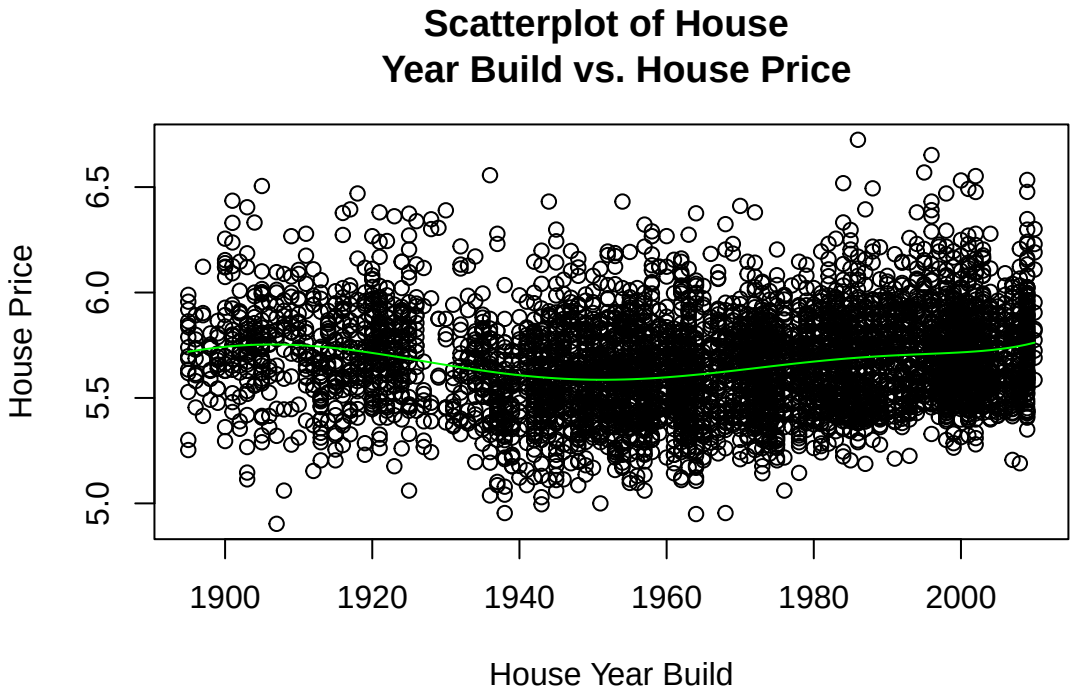
From the output, we have that the boxcox transformation coefficient is 19.2, which is really large. In addition, it doesn't make much sense to consider the house build years to be normally distributed since over time, more and more houses are built due to an increase in population. Hence, we will not transform the variate date of house sales.

```
# plot house year build vs. price
plot(dat$yrb, dat$price, xlab="House Year Build",
     ylab="House Price",
     main="Scatterplot of House \n Year Build vs. House Price")
```

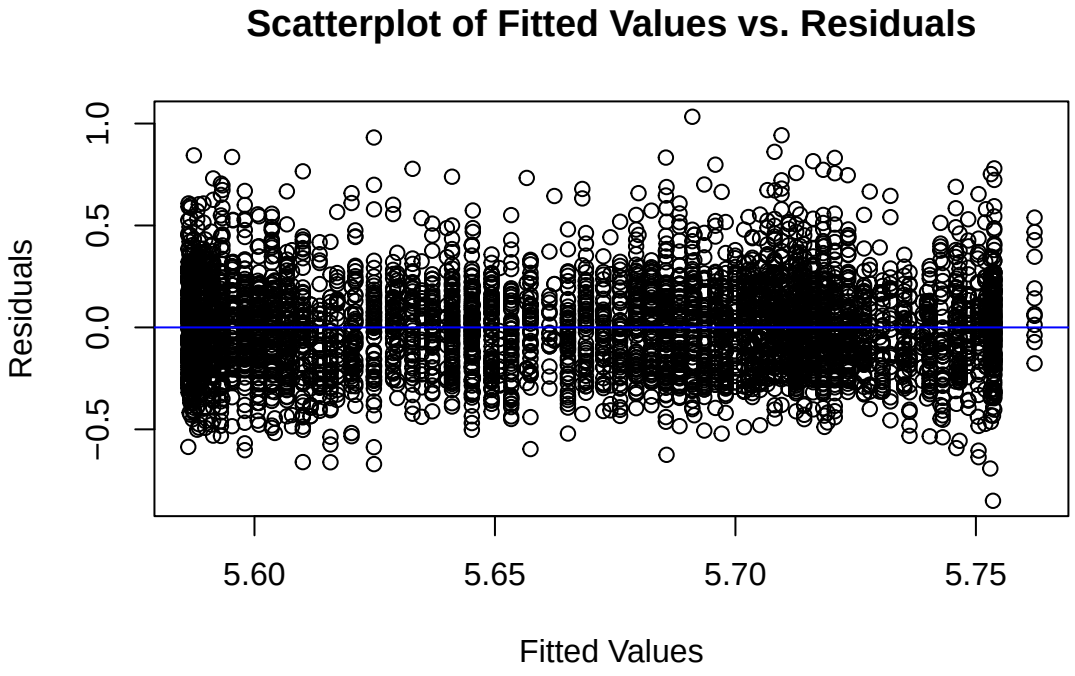


From the plot of the house year build vs. price, it seems like the data follows a 6 degree polynomial trend. Let's plot the house year build vs. price, alongside the fitted lines. Let's also plot the fitted residuals.

```
# fit 6 degree polynomial model
fit_yrb <- lm(price~poly(yrb, 6), data=dat)

# plot house year build vs. price
plot(dat$yrb, dat$price, xlab="House Year Build",
     ylab="House Price",
     main="Scatterplot of House \n Year Build vs. House Price")

# predict prices
predicted_prices_yrb <- predict(
  fit_yrb,
  newdata = data.frame(
    yrb = seq(min(dat$yrb),
              max(dat$yrb),
              length.out = 100
    )
  )
)

# add lines to plot
lines(seq(min(dat$yrb),
          max(dat$yrb),
          length.out = 100),
      predicted_prices_yrb, col = "green"
)
```




```
# plot fitted residuals
plot(fit_yrb$fitted.values, fit_yrb$residuals, xlab="Fitted Values",
     ylab="Residuals", main="Scatterplot of Fitted Values vs. Residuals")
abline(h=0, col="blue")
```

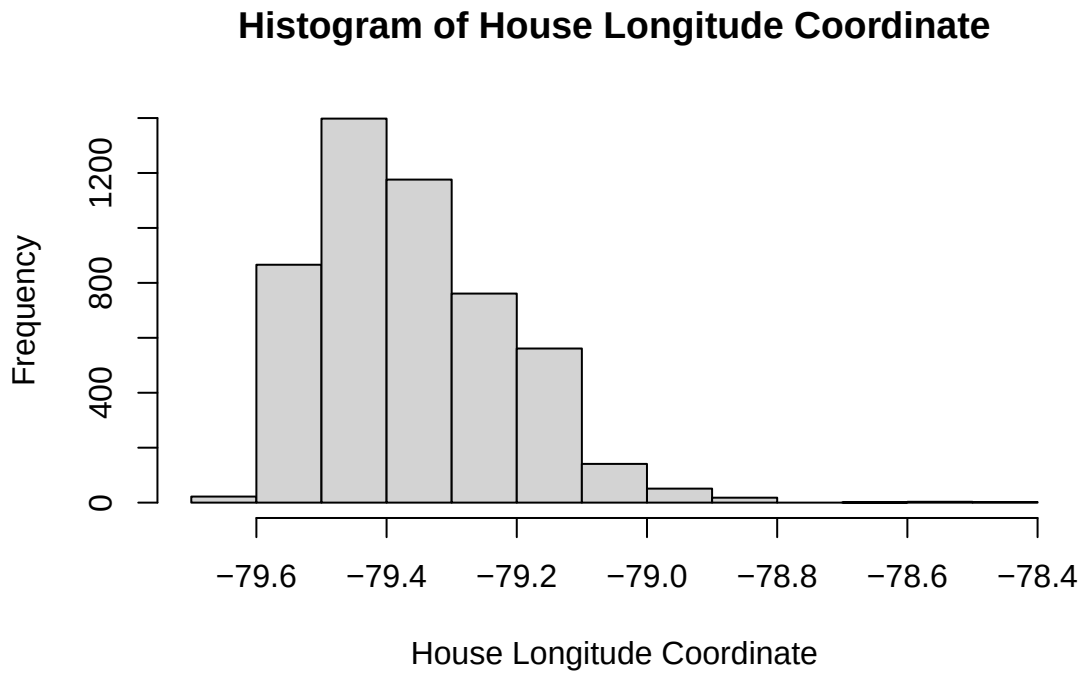


From the scatterplot of the house year build vs. the house price, we can see that the fitted line seems to match the trend of the datapoints quite well. From the scatterplot of fitted values vs. residuals, we can see that the expected value of the residuals is approximately 0, since for each fitted value, there seems to be as many datapoints above the horizontal blue line as there are below. The rightmost points are not concerning since they only account for a very small proportion of the entire dataset; most of the dataset is captured at other fitted values, and those data points seem to have expected residual around 0 for each fitted value. The variance of the residuals does not seem constant, since the spread of the residuals seems to decrease from fitted value below 5.575 to fitted value 5.65. This cannot be handled appropriately since we cannot transform the response variate. Finally, there are no noticable outliers.

Longitude

Here is a histogram for the house longitude coordinate.

```
hist(dat$longitude, xlab="House Longitude Coordinate",
     main="Histogram of House Longitude Coordinate")
```



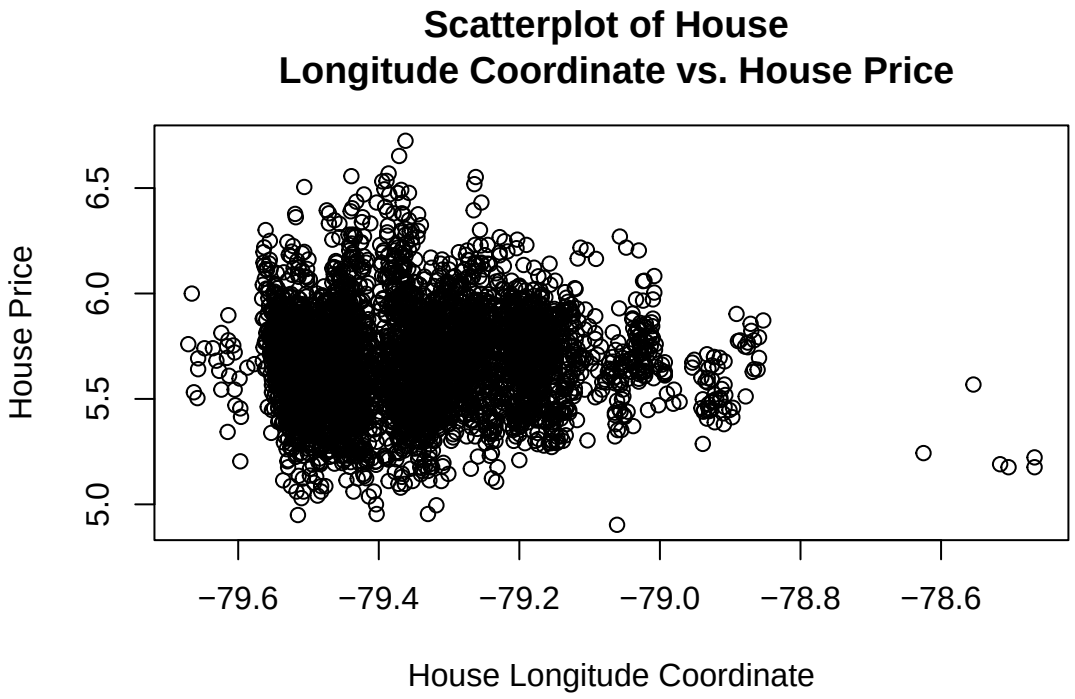
Since this histogram does not appear to be normal nor uniform, it follows that a boxcox transformation is needed. To do this, we will use the boxcox library function, find its maximum likelihood, and then use that value to transform the variate using the function defined before.

```
bc_longitude <- MASS::boxcox(lm(-longitude~price, data=dat),
                             lambda = seq(-1000, 1000, 1/10),
                             plotit=FALSE)
bc_longitude$x[which.max(bc_longitude$y)]
```

```
## [1] 184.2
```

From the output, we have that the boxcox transformation coefficient is 184.2, which is really large. In addition, it doesn't make much sense to consider the house longitude coordinate to be normally distributed since in a real world scenario, houses are not built closely packed together in a single location, and then the farther they are from that location, the more spread out and apart the houses are. Hence, we will not transform the variate house longitude.

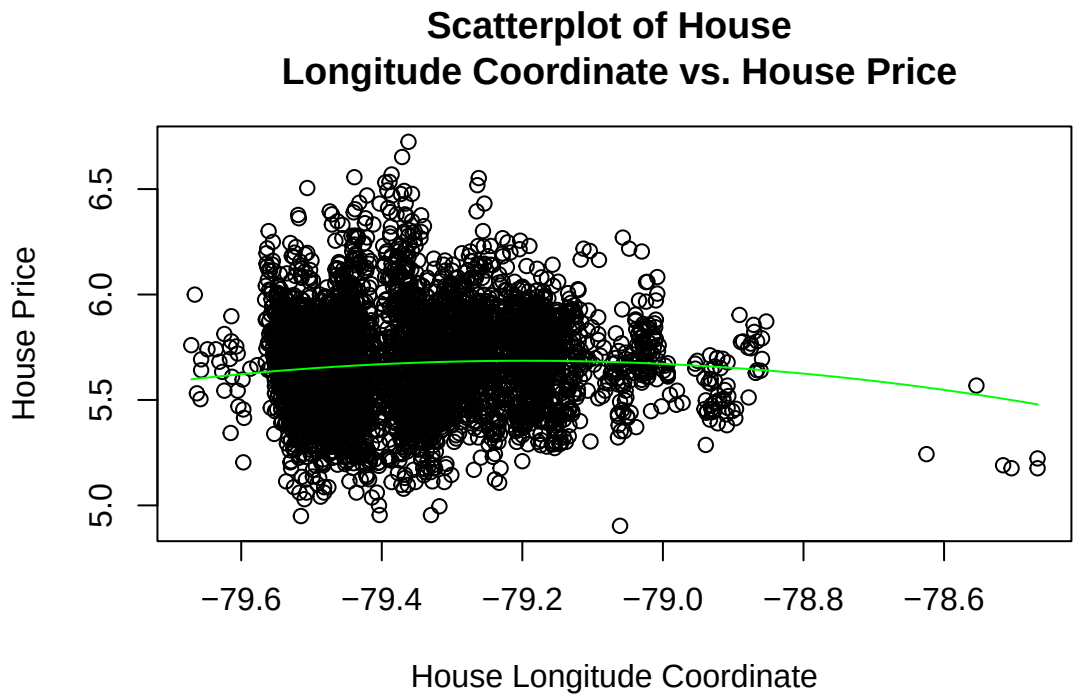
```
# plot house longitude vs. price
plot(dat$longitude, dat$price, xlab="House Longitude Coordinate",
     ylab="House Price",
     main="Scatterplot of House \n Longitude Coordinate vs. House Price")
```



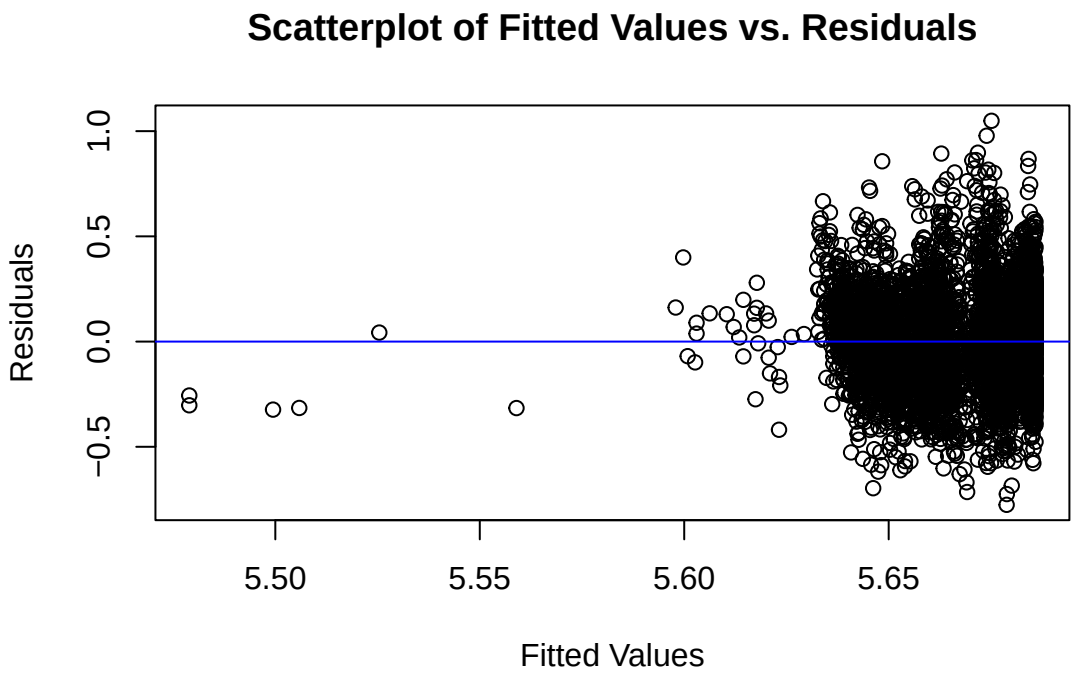
From the plot of the house longitude vs. price, it seems like the data follows a quadratic trend. Let's plot the house longitude vs. price, alongside the fitted lines. Let's also plot the fitted residuals.

```
# fit quadratic model
fit_longitude <- lm(price~poly(longitude, 2), data=dat)

# plot house longitude vs. price
plot(dat$longitude, dat$price, xlab="House Longitude Coordinate",
     ylab="House Price",
     main="Scatterplot of House \n Longitude Coordinate vs. House Price")

# predict prices
predicted_prices_longitude <- predict(
  fit_longitude,
  newdata = data.frame(
    longitude = seq(min(dat$longitude),
                    max(dat$longitude),
                    length.out = 100
```

```
    )
  )
)

# add lines to plot
lines(seq(min(dat$longitude),
          max(dat$longitude),
          length.out = 100),
      predicted_prices_longitude, col = "green"
)
```




```
# plot fitted residuals
plot(fit_longitude$fitted.values, fit_longitude$residuals, xlab="Fitted Values",
     ylab="Residuals", main="Scatterplot of Fitted Values vs. Residuals")
abline(h=0, col="blue")
```

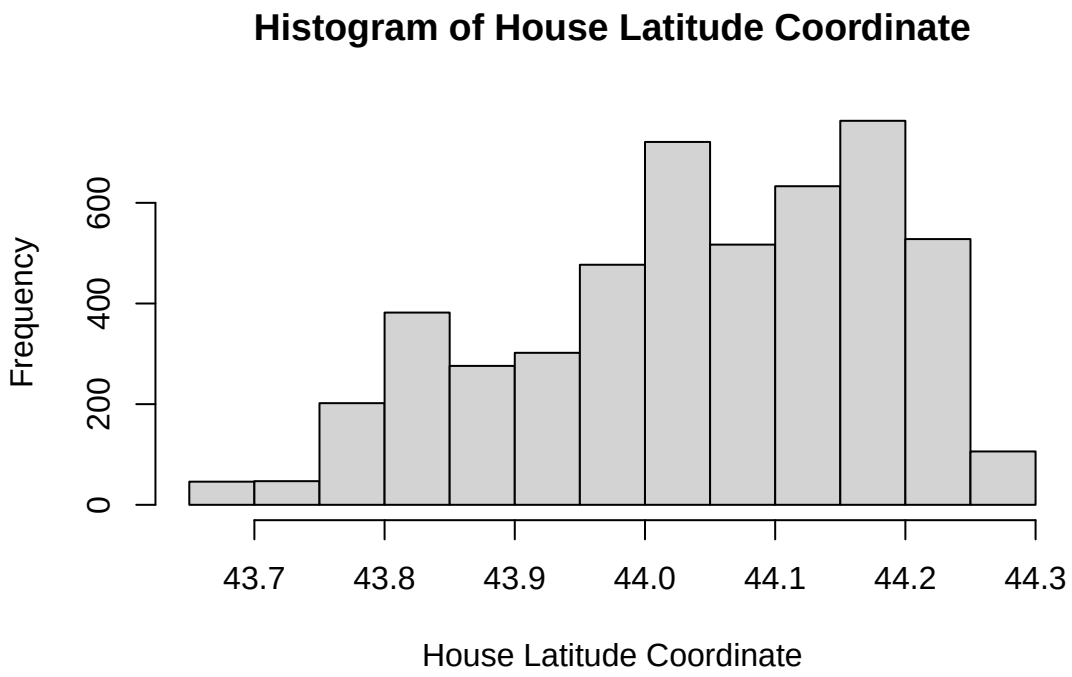


From the scatterplot of the house longitude coordinate vs. the house price, we can see that the fitted line seems to match the trend of the datapoints quite well. From the scatterplot of fitted values vs. residuals, we can see that the expected value of the residuals is approximately 0, since for each fitted value, there seems to be as many datapoints above the horizontal blue line as there are below. The points with fitted value less than 5.625 are not concerning since they only account for a very small proportion of the entire dataset; most of the dataset is captured at fitted values greater than 5.625, and those data points seem to have expected residual around 0 for each fitted value. The variance of the residuals does not seem constant, since the spread of the residuals seems to increase from fitted value 5.65 to fitted value 5.7. This cannot be handled appropriately since we cannot transform the response variate. Finally, there are no noticable outliers, since all the points are close to the fitted line, even though some of them are far way from the others in terms of their fitted value.

Latitude

Here is a histogram for the house latitude coordinate.

```
hist(dat$latitude, xlab="House Latitude Coordinate",
     main="Histogram of House Latitude Coordinate")
```



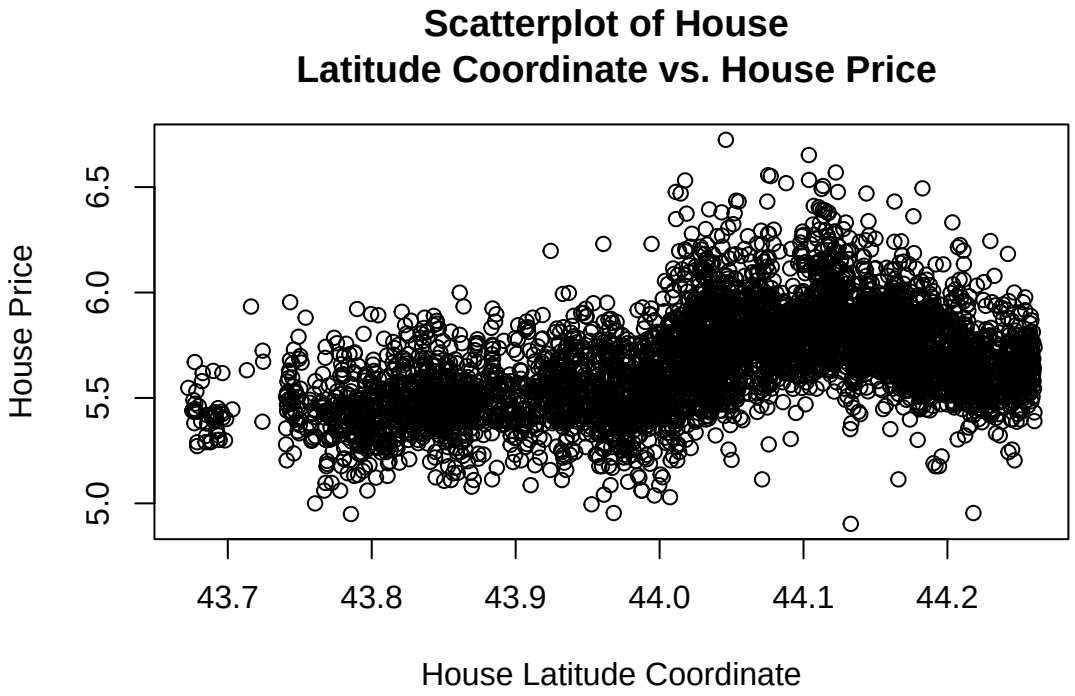
Since this histogram does not appear to be normal nor uniform, it follows that a boxcox transformation is needed. To do this, we will use the boxcox library function, find its maximum likelihood, and then use that value to transform the variate using the function defined before.

```
bc_latitude <- MASS::boxcox(lm(latitude~price, data=dat),
                            lambda = seq(-1000, 1000, 1/10),
                            plotit=FALSE)
bc_latitude$x[which.max(bc_latitude$y)]
```

```
## [1] 54.2
```

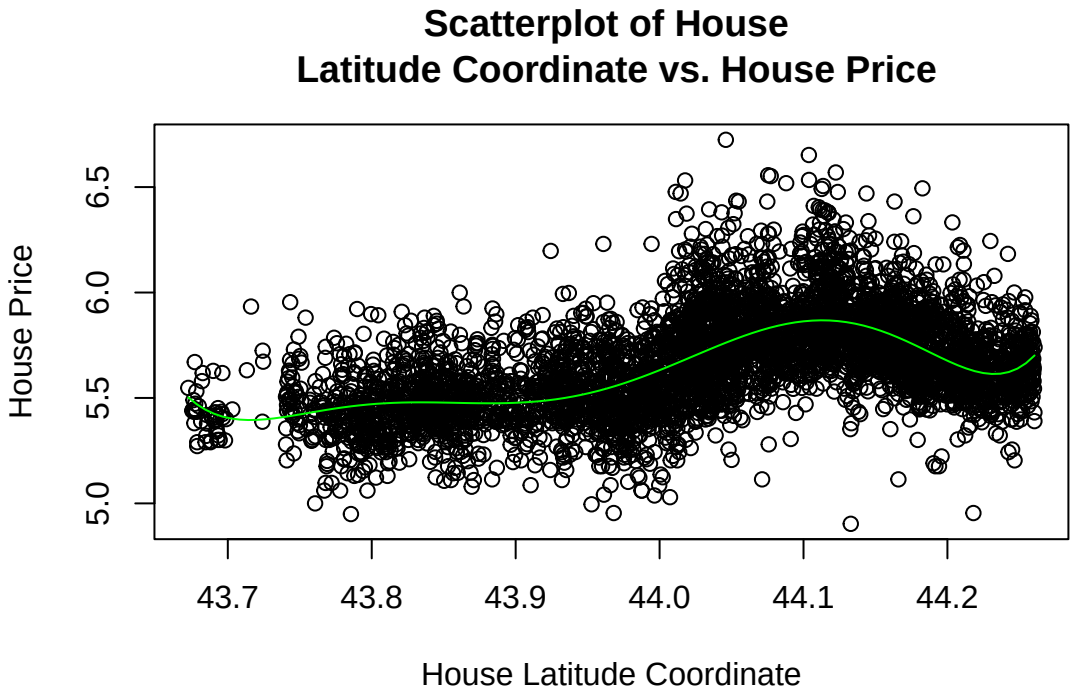
From the output, we have that the boxcox transformation coefficient is 54.2, which is really large. In addition, it doesn't make much sense to consider the house latitude coordinate to be normally distributed since in a real world scenario, houses are not built closely packed together in a single location, and then the farther they are from that location, the more spread out and apart the houses are. Hence, we will not transform the variate house latitude.

```
# plot house latitude vs. price
plot(dat$latitude, dat$price, xlab="House Latitude Coordinate",
     ylab="House Price",
     main="Scatterplot of House \n Latitude Coordinate vs. House Price")
```

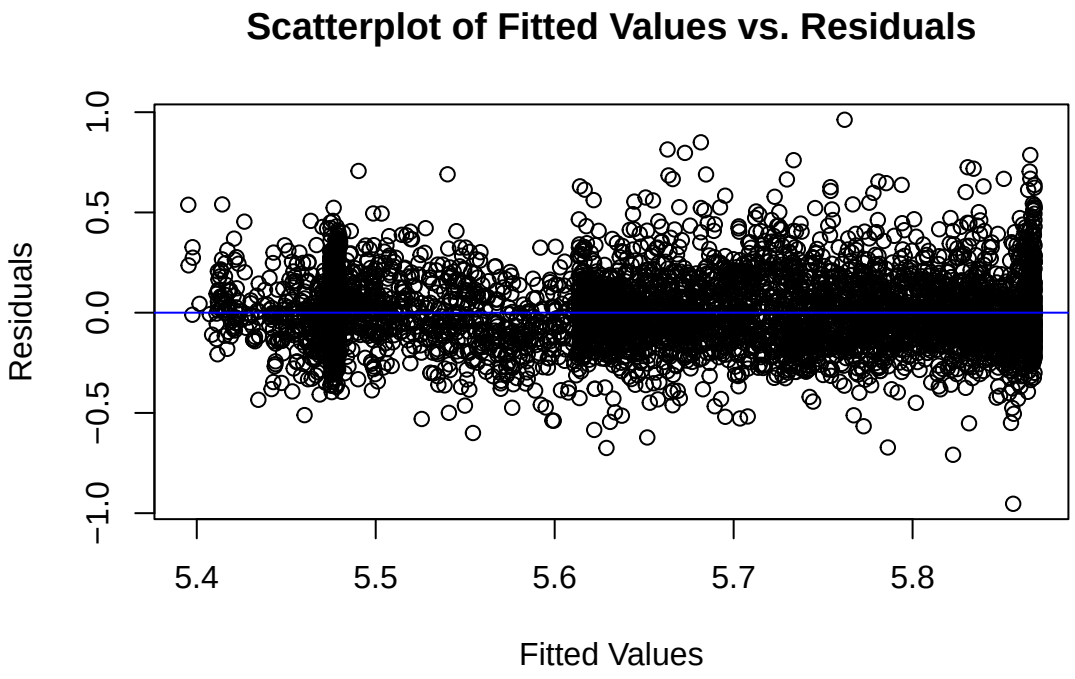


From the plot of the house latitude vs. price, it seems like the data follows a 7 degree polynomial trend. Let's plot the house latitude vs. price, alongside the fitted lines. Let's also plot the fitted residuals.

```
# fit 7 degree polynomial model
fit_latitude <- lm(price~poly(latitude, 7), data=dat)

# plot house latitude vs. price
plot(dat$latitude, dat$price, xlab="House Latitude Coordinate",
     ylab="House Price",
     main="Scatterplot of House \n Latitude Coordinate vs. House Price")

# predict prices
predicted_prices_latitude <- predict(
  fit_latitude,
  newdata = data.frame(
    latitude = seq(min(dat$latitude),
                   max(dat$latitude),
                   length.out = 100
    )
  )
)

# add lines to plot
lines(seq(min(dat$latitude),
          max(dat$latitude),
          length.out = 100),
      predicted_prices_latitude, col = "green"
)
```




```
# plot fitted residuals
plot(fit_latitude$fitted.values, fit_latitude$residuals, xlab="Fitted Values",
     ylab="Residuals", main="Scatterplot of Fitted Values vs. Residuals")
abline(h=0, col="blue")
```



From the scatterplot of the house latitude coordinate vs. the house price, we can see that the fitted line seems to match the trend of the datapoints quite well. From the scatterplot of fitted values vs. residuals, we can see that the expected value of the residuals is approximately 0, since for each fitted value, there seems to be as many datapoints above the horizontal blue line as there are below. The points on the right are not concerning since they only account for a very small proportion of the entire dataset; most of the dataset is to the left of that, and those data points seem to have expected residual around 0 for each fitted value. The variance of the residuals does not seem constant, since the spread of the residuals seems to increase and then decrease from fitted value 5.4 to fitted value 5.5. This cannot be handled appropriately since we cannot transform the response variate. Finally, there are no noticable outliers.

Using the transformed variates, alongisde all the formulas above that best represented their respective scatterplots, and the categorical variates `waterfront`, `south`, `east`, `region`, and `view`, I considered a potential model to be the following:

```
potential_model <- lm(formula = price ~ poly(saledate, 3) + bathrm +
                        poly(bedrm, 2) + poly(sizeLot, 5) + poly(sizeLiving, 2) +
                        floors + waterfront + poly(yrb, 6) + condition_at_least_average +
                        south + east + region + view + poly(longitude, 2) +
                        poly(latitude, 7) + poly(dist_from_downtown, 2) * close_to_downtown,
                      data = dat)
```

Interactive Effects

Now that we transformed and fit our variates individually, we will now consider their interactions and whether utilizing them reduces the $PRESS$ statistic.

Note that the number of bathrooms and the number of bedrooms in a house could have an interactive effect on the price of the house. Indeed, this makes sense because if we consider a house with no bedrooms and no bathrooms, increasing the number of bathrooms wouldn't increase the price of the house that much since there are still no bedrooms. However, increasing both the number of bathrooms and the number of bedrooms would increase the price of the house, since people can now live there.

Comparing $PRESS$ statistics of both models, keeping everything else fixed, we have:

```
model_no_interaction <- lm(formula = price ~ poly(saledate, 3) + bathrm +
                             poly(bedrm, 2) + poly(sizeLot, 5) + poly(sizeLiving, 2) +
                             floors + waterfront + poly(yrb, 6) +
                             condition_at_least_average + south + east + region +
                             view + poly(longitude, 2) + poly(latitude, 7) +
                             poly(dist_from_downtown, 2) * close_to_downtown, data = dat)
print(paste0("Model with no bed and bath interaction: ", olsrr::ols_press(model_no_interaction)))
```

```
## [1] "Model with no bed and bath interaction: 40.0709204310814"
```

```
model_interaction <- lm(formula = price ~ poly(saledate, 3) + bathrm*poly(bedrm, 2) +
                          poly(sizeLot, 5) + poly(sizeLiving, 2) + floors + waterfront +
                          poly(yrb, 6) + condition_at_least_average + south + east +
                          region + view + poly(longitude, 2) + poly(latitude, 7) +
                          poly(dist_from_downtown, 2)*close_to_downtown, data = dat)
print(paste0("Model with bed and bath interaction: ", olsrr::ols_press(model_interaction)))
```

```
## [1] "Model with bed and bath interaction: 40.0181199503189"
```

Note that the press statistic for the model with interaction between the number of bedrooms and the number of bathrooms, has a lower $PRESS$ statistic, compared to the model with no interaction between the number of bedrooms and the number of bathrooms.

Next, note that the number of house floors and whether the house has a waterfront could have an interactive effect on the price of the house. Indeed, this makes sense because houses with less floors are cheaper houses, hence whether they have a waterfront or not wouldn't affect the price as much since the house is cheaper. In other words, houses with more floors are more expensive, hence adding a waterfront to it would affect the price more since it is actually worth it given the expensive house.

Comparing $PRESS$ statistics of both models, keeping everything else fixed, we have:

```
model_no_interaction <- lm(formula = price ~ poly(saledate, 3) + bathrm*poly(bedrm, 2) +
                             poly(sizeLot, 5) + poly(sizeLiving, 2) + floors +
                             waterfront + poly(yrb, 6) + condition_at_least_average +
                             south + east + region + view + poly(longitude, 2) +
                             poly(latitude, 7) + poly(dist_from_downtown, 2)*
                             close_to_downtown, data = dat)
print(paste0("Model with no floors and waterfront interaction: ",
             olsrr::ols_press(model_no_interaction)))
```

```
## [1] "Model with no floors and waterfront interaction: 40.0181199503189"
```

```
model_interaction <- lm(formula = price ~ poly(saledate, 3) + bathrm*poly(bedrm, 2) +
                          poly(sizeLot, 5) + poly(sizeLiving, 2) + floors*waterfront +
                          poly(yrb, 6) + condition_at_least_average + south + east +
                          region + view + poly(longitude, 2) + poly(latitude, 7) +
                          poly(dist_from_downtown, 2)*close_to_downtown, data = dat)
print(paste0("Model with floors and waterfront interaction: ",
             olsrr::ols_press(model_interaction)))
```

```
## [1] "Model with floors and waterfront interaction: 39.877898331502"
```

Note that the press statistic for the model with interaction between the number of floors and whether the house has a waterfront, has a lower $PRESS$ statistic, compared to the model with no interaction between the number of floors and whether the house has a waterfront.

Next, note that whether the house is in the south section of Toronto, whether the house is in the east section of Toronto, and whether the house is in the inner region of Toronto, could all have an interactive effect on the price of the house. Indeed, this makes sense because houses in certain areas can be worth a lot more than houses in other areas.

Comparing $PRESS$ statistics of both models, keeping everything else fixed, we have:

```
model_no_interaction <- lm(formula = price ~ poly(saledate, 3) + bathrm*poly(bedrm, 2) +
                             poly(sizeLot, 5) + poly(sizeLiving, 2) + floors*waterfront +
                             poly(yrb, 6) + condition_at_least_average + south + east +
                             region + view + poly(longitude, 2) + poly(latitude, 7) +
                             poly(dist_from_downtown, 2)*close_to_downtown, data = dat)
print(paste0("Model with no south, east, and region interaction: ",
             olsrr::ols_press(model_no_interaction)))
```

```
## [1] "Model with no south, east, and region interaction: 39.877898331502"
```

```
model_interaction <- lm(formula = price ~ poly(saledate, 3) + bathrm*poly(bedrm, 2) +
                          poly(sizeLot, 5) + poly(sizeLiving, 2) + floors*waterfront +
                          poly(yrb, 6) + condition_at_least_average + south*east*region +
                          view + poly(longitude, 2) + poly(latitude, 7) +
                          poly(dist_from_downtown, 2)*close_to_downtown, data = dat)
print(paste0("Model with south, east, and region interaction: ",
             olsrr::ols_press(model_interaction)))
```

```
## [1] "Model with south, east, and region interaction: 38.1342038849154"
```

Note that the press statistic for the model with that interaction has a lower $PRESS$ statistic, compared to the model without that interaction.

Finally, note that the house latitude coordinate, house longitude coordinate, and closeness to downtown indicator could have an interactive effect on the price of the house. Indeed, this makes sense because again, houses in certain areas (i.e. both south and east sections) can be worth a lot more than houses in other areas.

Comparing $PRESS$ statistics of both models, keeping everything else fixed, we have:

```
model_no_interaction <- lm(formula = price ~ poly(saledate, 3) + bathrm*poly(bedrm, 2) +
                             poly(sizeLot, 5) + poly(sizeLiving, 2) + floors*waterfront +
                             poly(yrb, 6) + condition_at_least_average + south*east*region +
                             view + poly(longitude, 2) + poly(latitude, 7) +
                             poly(dist_from_downtown, 2)*close_to_downtown, data = dat)
print(paste0("Model with no latitude, longitude, and close_to_downtown interaction: ",
             olsrr::ols_press(model_no_interaction)))
```

```
## [1] "Model with no latitude, longitude, and close_to_downtown interaction: 38.1342038849154"
```

```
model_interaction <- lm(formula = price ~ poly(saledate, 3) + bathrm*poly(bedrm, 2) +
                          poly(sizeLot, 5) + poly(sizeLiving, 2) + floors*waterfront +
                          poly(yrb, 6) + condition_at_least_average + south*east*region +
                          view + poly(longitude, 2)*poly(latitude, 7) * close_to_downtown +
                          poly(dist_from_downtown, 2)*close_to_downtown, data = dat)
print(paste0("Model with latitude, longitude, and close_to_downtown interaction: ",
             olsrr::ols_press(model_interaction)))
```

```
## [1] "Model with latitude, longitude, and close_to_downtown interaction: 33.6978141529747"
```

Note that the press statistic for the model with that interaction has a lower $PRESS$ statistic, compared to the model without that interaction.

Hence, from all of this, it follows that a model we can consider is the following.

```
potential_model <- lm(formula = price ~ poly(saledate, 3) + bathrm*poly(bedrm, 2) +
                        poly(sizeLot, 5) + poly(sizeLiving,  2) + floors*waterfront +
                        poly(yrb, 6) + condition_at_least_average + south*east*region +
                        view + poly(longitude, 2)*poly(latitude, 7) * close_to_downtown +
                        poly(dist_from_downtown, 2) * close_to_downtown, data = dat)
```

Stepwise Model Selection

Let's perform stepwise model selection (forward and backward) to find the "best" model.

```
step_model <- step(lm(price~1, data=dat),
     scope = list(
       upper = ~ poly(saledate, 3) + bathrm*poly(bedrm, 2) +
         poly(sizeLot, 5) + poly(sizeLiving,  2) + floors*waterfront +
         poly(yrb, 6) + condition_at_least_average + south*east*region +
         view + poly(longitude, 2)*poly(latitude, 7) * close_to_downtown +
         poly(dist_from_downtown, 2) * close_to_downtown,
       lower = ~1),
     trace = 0,
     direction = "both")
step_model$call$formula
```

```
## price ~ poly(sizeLiving, 2) + poly(latitude, 7) + view + poly(dist_from_downtown,
##     2) + close_to_downtown + region + south + poly(sizeLot, 5) +
##     poly(yrb, 6) + poly(saledate, 3) + east + waterfront + condition_at_least_average +
##     poly(longitude, 2) + poly(bedrm, 2) + bathrm + floors + poly(latitude,
##     7):close_to_downtown + south:east + region:east + poly(latitude,
##     7):poly(longitude, 2) + poly(bedrm, 2):bathrm + waterfront:floors +
##     close_to_downtown:poly(longitude, 2) + poly(latitude, 7):close_to_downtown:poly(longitude,
##     2)
```

Conclusion

Computing the *PRESS* of our model using the potential "best" model formula from the output in the last section, we have the following.

```
optimal_model <- lm(formula = price~poly(sizeLiving, 2) + poly(latitude, 7) +
  view + poly(dist_from_downtown, 2) + close_to_downtown + region + south +
  poly(sizeLot, 5) + poly(yrb, 6) + poly(saledate, 3) + east + waterfront +
  condition_at_least_average + poly(longitude, 2) + poly(bedrm, 2) + bathrm +
  floors + poly(latitude, 7):close_to_downtown + south:east + region:east +
  poly(latitude, 7):poly(longitude, 2) + poly(bedrm, 2):bathrm + waterfront:floors +
  close_to_downtown:poly(longitude, 2) + poly(latitude, 7):close_to_downtown:poly(longitude,
  2), data = dat)
print(paste0("PRESS: ", olsrr::ols_press(optimal_model)))
```

```
## [1] "PRESS: 33.6302390927961"
```

I also found that removing `close_to_downtown` from the formula resulted in a lower *PRESS* score, hence I removed it.

```
optimal_model <- lm(formula = price~poly(sizeLiving, 2) + poly(latitude, 7) +
  view + poly(dist_from_downtown, 2) + region + south +
  poly(sizeLot, 5) + poly(yrb, 6) + poly(saledate, 3) + east + waterfront +
  condition_at_least_average + poly(longitude, 2) + poly(bedrm, 2) + bathrm +
  floors + poly(latitude, 7):close_to_downtown + south:east + region:east +
  poly(latitude, 7):poly(longitude, 2) + poly(bedrm, 2):bathrm + waterfront:floors +
```

```
  close_to_downtown:poly(longitude, 2) + poly(latitude, 7):close_to_downtown:poly(longitude,
  2), data = dat)
print(paste0("PRESS: ", olsrr::ols_press(optimal_model)))
```

```
## [1] "PRESS: 33.6040439233797"
```

From the output, we have that this model $PRESS$ is around 33.60. This is the lowest $PRESS$ that I found given the dataset and predictors, which concludes the summary of this document.

Side Note: Multicollinearity

As a side note, notice that there is moderate/strong evidence of multicollinearity in the optimal model.

```
car::vif(optimal_model)
```

```
## there are higher-order terms (interactions) in this model
## consider setting type = 'predictor'; see ?vif
```

```
##                                                          GVIF Df
## poly(sizeLiving, 2)                                8.151832e+00  2
## poly(latitude, 7)                                  1.001412e+29  7
## view                                               2.403691e+00  4
## poly(dist_from_downtown, 2)                        3.415391e+07  2
## region                                             3.860158e+00  1
## south                                              1.019440e+01  1
## poly(sizeLot, 5)                                   7.283798e+00  5
## poly(yrb, 6)                                       1.022482e+01  6
## poly(saledate, 3)                                  1.048294e+00  3
## east                                               3.387066e+01  1
## waterfront                                         3.703109e+00  1
## condition_at_least_average                         1.054431e+00  1
## poly(longitude, 2)                                 7.225323e+13  2
## poly(bedrm, 2)                                     1.270580e+01  2
## bathrm                                             3.529085e+00  1
## floors                                             2.425132e+00  1
## poly(latitude, 7):close_to_downtown                2.263726e+26  7
## south:east                                         1.635927e+01  1
## region:east                                        1.416701e+01  1
## poly(latitude, 7):poly(longitude, 2)               4.565413e+40 14
## poly(bedrm, 2):bathrm                              9.827586e+00  2
## waterfront:floors                                  2.906685e+00  1
## poly(longitude, 2):close_to_downtown               4.671807e+12  2
## poly(latitude, 7):poly(longitude, 2):close_to_downtown 2.555974e+47 14
##                                                    GVIF^(1/(2*Df))
## poly(sizeLiving, 2)                                       1.689716
## poly(latitude, 7)                                       117.888749
## view                                                      1.115860
## poly(dist_from_downtown, 2)                              76.446937
## region                                                    1.964728
## south                                                     3.192867
## poly(sizeLot, 5)                                          1.219652
```

```
## poly(yrb, 6)                                                1.213774
## poly(saledate, 3)                                           1.007892
## east                                                        5.819851
## waterfront                                                  1.924346
## condition_at_least_average                                  1.026855
## poly(longitude, 2)                                       2915.508488
## poly(bedrm, 2)                                              1.887993
## bathrm                                                      1.878586
## floors                                                      1.557283
## poly(latitude, 7):close_to_downtown                        76.293478
## south:east                                                  4.044659
## region:east                                                 3.763909
## poly(latitude, 7):poly(longitude, 2)                       28.322024
## poly(bedrm, 2):bathrm                                       1.770564
## waterfront:floors                                           1.704900
## poly(longitude, 2):close_to_downtown                     1470.182450
## poly(latitude, 7):poly(longitude, 2):close_to_downtown     49.331806
```

Removing terms with moderate/strong evidence of multicollinearity (scaled VIF above 2.2) results in the following model.

```
model_no_mc <- lm(formula = price~poly(sizeLiving, 2) +
  view + region + south +
  poly(sizeLot, 5) + poly(yrb, 6) + poly(saledate, 3) + waterfront +
  condition_at_least_average + poly(bedrm, 2) + bathrm +
  floors + poly(latitude, 7):close_to_downtown +
  poly(latitude, 7):poly(longitude, 2) + poly(bedrm, 2):bathrm + waterfront:floors, data = dat)
car::vif(model_no_mc)
```

```
## there are higher-order terms (interactions) in this model
## consider setting type = 'predictor'; see ?vif
```

```
##                                          GVIF Df GVIF^(1/(2*Df))
## poly(sizeLiving, 2)                  7.932334  2        1.678225
## view                                 2.307248  4        1.110163
## region                               1.885628  1        1.373182
## south                                2.585225  1        1.607863
## poly(sizeLot, 5)                     5.523900  5        1.186382
## poly(yrb, 6)                         8.490675  6        1.195121
## poly(saledate, 3)                    1.034685  3        1.005699
## waterfront                           3.649804  1        1.910446
## condition_at_least_average           1.044335  1        1.021927
## poly(bedrm, 2)                      12.403291  2        1.876654
## bathrm                               3.506594  1        1.872590
## floors                               2.405960  1        1.551116
## poly(latitude, 7):close_to_downtown 53661.761165 14        1.475423
## poly(latitude, 7):poly(longitude, 2) 41768.752868 14        1.462280
## poly(bedrm, 2):bathrm                9.566169  2        1.758671
## waterfront:floors                    2.883979  1        1.698228
```

```
print(paste0("PRESS: ", olsrr::ols_press(model_no_mc)))
```

```
## [1] "PRESS: 36.6614552404478"
```

This model has $PRESS$ statistic 36.66, which is higher than that of `optimal_model`. I used the `optimal_model` in the `LinearModel` function, not the model with no evidence of multicollinearity. Users could use this model if multicollinearity is a big concern (with the slight tradeoff of higher $PRESS$).

1.Preprocessing

1.1 Loading data

```
load("linear.Rdata")
```

2. Model building

```
# May the magic model appear!
source("20870719.R")
LinearModel(dat)
```

```
##
## Call:
## lm(formula = price ~ poly(sizeLiving, 2) + poly(latitude, 7) +
##     view + poly(dist_from_downtown, 2) + region + south + poly(sizeLot,
##     5) + poly(yrb, 6) + poly(saledate, 3) + east + waterfront +
##     condition_at_least_average + poly(longitude, 2) + poly(bedrm,
##     2) + bathrm + floors + poly(latitude, 7):close_to_downtown +
##     south:east + region:east + poly(latitude, 7):poly(longitude,
##     2) + poly(bedrm, 2):bathrm + waterfront:floors + close_to_downtown:poly(longitude,
##     2) + poly(latitude, 7):close_to_downtown:poly(longitude,
##     2), data = dat)
##
## Coefficients:
##                                             (Intercept)
##                                               4.269e+00
##                                    poly(sizeLiving, 2)1
##                                               7.381e+00
##                                    poly(sizeLiving, 2)2
##                                               6.897e-01
##                                      poly(latitude, 7)1
##                                               2.338e+02
##                                      poly(latitude, 7)2
##                                              -5.751e+02
##                                      poly(latitude, 7)3
##                                               4.988e+02
##                                      poly(latitude, 7)4
##                                              -1.664e+02
##                                      poly(latitude, 7)5
##                                              -7.904e+01
##                                      poly(latitude, 7)6
##                                               8.280e+01
##                                      poly(latitude, 7)7
```

```
##                                        -2.126e+01
##                                             view1
##                                         7.232e-02
##                                             view2
##                                         5.260e-02
##                                             view3
##                                         8.669e-02
##                                             view4
##                                         1.363e-01
##                          poly(dist_from_downtown, 2)1
##                                         1.078e+02
##                          poly(dist_from_downtown, 2)2
##                                         1.865e+01
##                                       regionOuter
##                                         1.433e-02
##                                            south1
##                                        -1.992e-02
##                                 poly(sizeLot, 5)1
##                                         3.241e+00
##                                 poly(sizeLot, 5)2
##                                        -3.331e-01
##                                 poly(sizeLot, 5)3
##                                        -8.714e-02
##                                 poly(sizeLot, 5)4
##                                         3.763e-01
##                                 poly(sizeLot, 5)5
##                                        -2.183e-01
##                                     poly(yrb, 6)1
##                                         6.213e-01
##                                     poly(yrb, 6)2
##                                         1.126e+00
##                                     poly(yrb, 6)3
##                                         5.839e-01
##                                     poly(yrb, 6)4
##                                        -2.042e-01
##                                     poly(yrb, 6)5
##                                         2.094e-01
##                                     poly(yrb, 6)6
##                                         8.525e-02
##                               poly(saledate, 3)1
##                                         8.211e-01
##                               poly(saledate, 3)2
##                                         5.032e-01
##                               poly(saledate, 3)3
##                                         3.328e-01
##                                             east1
##                                        -3.600e-02
##                                       waterfront1
##                                         1.104e-01
##                        condition_at_least_average1
##                                         1.141e-01
##                                poly(longitude, 2)1
##                                        -1.621e+03
##                                poly(longitude, 2)2
```

```
##                                         6.272e+02
##                                    poly(bedrm, 2)1
##                                        -2.886e-01
##                                    poly(bedrm, 2)2
##                                        -1.275e-01
##                                            bathrm
##                                         2.725e-02
##                                            floors
##                                         3.099e-02
##                     poly(latitude, 7)1:close_to_downtown1
##                                        -4.388e+02
##                     poly(latitude, 7)2:close_to_downtown1
##                                         4.467e+02
##                     poly(latitude, 7)3:close_to_downtown1
##                                        -5.838e+02
##                     poly(latitude, 7)4:close_to_downtown1
##                                         1.106e+02
##                     poly(latitude, 7)5:close_to_downtown1
##                                         5.403e+01
##                     poly(latitude, 7)6:close_to_downtown1
##                                        -9.040e+01
##                     poly(latitude, 7)7:close_to_downtown1
##                                         2.029e+01
##                                      south1:east1
##                                         7.770e-02
##                                  regionOuter:east1
##                                         9.302e-02
##                   poly(latitude, 7)1:poly(longitude, 2)1
##                                         2.405e+05
##                   poly(latitude, 7)2:poly(longitude, 2)1
##                                        -3.026e+05
##                   poly(latitude, 7)3:poly(longitude, 2)1
##                                         3.134e+05
##                   poly(latitude, 7)4:poly(longitude, 2)1
##                                        -2.169e+05
##                   poly(latitude, 7)5:poly(longitude, 2)1
##                                         1.006e+05
##                   poly(latitude, 7)6:poly(longitude, 2)1
##                                        -3.312e+04
##                   poly(latitude, 7)7:poly(longitude, 2)1
##                                         5.830e+03
##                   poly(latitude, 7)1:poly(longitude, 2)2
##                                        -9.866e+04
##                   poly(latitude, 7)2:poly(longitude, 2)2
##                                         1.204e+05
##                   poly(latitude, 7)3:poly(longitude, 2)2
##                                        -1.188e+05
##                   poly(latitude, 7)4:poly(longitude, 2)2
##                                         7.768e+04
##                   poly(latitude, 7)5:poly(longitude, 2)2
##                                        -3.233e+04
##                   poly(latitude, 7)6:poly(longitude, 2)2
##                                         8.696e+03
##                   poly(latitude, 7)7:poly(longitude, 2)2
```

```
##                                                 -9.605e+02
##                                     poly(bedrm, 2)1:bathrm
##                                                 -1.361e-01
##                                     poly(bedrm, 2)2:bathrm
##                                                 -3.725e-01
##                                         waterfront1:floors
##                                                  2.051e-01
##                     poly(longitude, 2)1:close_to_downtown1
##                                                  1.477e+03
##                     poly(longitude, 2)2:close_to_downtown1
##                                                 -7.990e+02
## poly(latitude, 7)1:poly(longitude, 2)1:close_to_downtown1
##                                                 -2.508e+05
## poly(latitude, 7)2:poly(longitude, 2)1:close_to_downtown1
##                                                  2.928e+05
## poly(latitude, 7)3:poly(longitude, 2)1:close_to_downtown1
##                                                 -3.204e+05
## poly(latitude, 7)4:poly(longitude, 2)1:close_to_downtown1
##                                                  2.124e+05
## poly(latitude, 7)5:poly(longitude, 2)1:close_to_downtown1
##                                                 -1.026e+05
## poly(latitude, 7)6:poly(longitude, 2)1:close_to_downtown1
##                                                  3.236e+04
## poly(latitude, 7)7:poly(longitude, 2)1:close_to_downtown1
##                                                 -5.997e+03
## poly(latitude, 7)1:poly(longitude, 2)2:close_to_downtown1
##                                                  8.622e+04
## poly(latitude, 7)2:poly(longitude, 2)2:close_to_downtown1
##                                                 -1.324e+05
## poly(latitude, 7)3:poly(longitude, 2)2:close_to_downtown1
##                                                  1.099e+05
## poly(latitude, 7)4:poly(longitude, 2)2:close_to_downtown1
##                                                 -8.354e+04
## poly(latitude, 7)5:poly(longitude, 2)2:close_to_downtown1
##                                                  2.953e+04
## poly(latitude, 7)6:poly(longitude, 2)2:close_to_downtown1
##                                                 -9.733e+03
## poly(latitude, 7)7:poly(longitude, 2)2:close_to_downtown1
##                                                  7.563e+02
```