Mini Project 2 - Ames Housing Data Analysis

Sarah Adilijiang, Yanqing Gui, Hanyang Peng

Problem 1 - Simpson's Paradox

1. Another example

Like the example given in the assignment. We now try to run a different regression and find a completely different example of the Simpson's paradox within this same data set.

We first regress log(Sale price) on rates of the overall condition of the house and central air conditioning. The coefficient of rates the overall condition of the house is negative. But if we regress also on the size of a garage in car capacity, the coefficient of rates of the overall condition of the house is positive.

$$\text{Model 1: } \log(\text{Sale price}) \sim \text{Overall Cond} + \text{CentralAir}$$

$$\text{Model 2: } \log(\text{Sale price}) \sim \text{Overall Cond} + \text{Garage Cars} + \text{Central Air}$$

```
summary(lm(Y~X[,c(41,4)]))
```

```
##
## Call:
## lm(formula = Y ~ X[, c(41, 4)])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.10635 -0.23797 -0.03033  0.22717  1.48694
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)                11.63125    0.04183 278.052  < 2e-16 ***
## X[, c(41, 4)]CentralAir     0.62196    0.02807  22.157  < 2e-16 ***
## X[, c(41, 4)]Overall Cond -0.03428    0.00632  -5.424  6.3e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3769 on 2920 degrees of freedom
## Multiple R-squared:  0.1459, Adjusted R-squared:  0.1453
## F-statistic: 249.4 on 2 and 2920 DF,  p-value: < 2.2e-16
```

```
summary(lm(Y~X[,c(41,22,4)]))
```

```
##
## Call:
## lm(formula = Y ~ X[, c(41, 22, 4)])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.31907 -0.17009 -0.00823  0.16888  1.07168
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  11.044961   0.034736 317.969  < 2e-16 ***
## X[, c(41, 22, 4)]CentralAir   0.312281   0.022674  13.773  < 2e-16 ***
## X[, c(41, 22, 4)]Garage Cars  0.337740   0.007527  44.873  < 2e-16 ***
## X[, c(41, 22, 4)]Overall Cond 0.015799   0.004989   3.167  0.00156 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.29 on 2919 degrees of freedom
## Multiple R-squared:  0.4946, Adjusted R-squared:  0.494
## F-statistic:   952 on 3 and 2919 DF,  p-value: < 2.2e-16
```

We can explain this phenomenon as follows:

It is obvious that if a house has a large garage, then it will generally have a higher sale price than a house without a garage or with a garage but little car capacity. But we assume that a house with a garage may have a lower rate of the overall condition of the house than those houses without a garage.

Therefore, in the first regression model, we don't consider the variable size of a garage in car capacity or other variables which may make the sale price higher but the rate of the overall condition of the house lower at the same time. Then we can see that the overall condition has negative coefficients, which is against our common sense. However, when we take the garage capacity into consideration and fit a new regression, now we can see that with higher overall condition rates, we will have higher sale price.

2. Issues for selective inference

This type of paradox, i.e. Simpson's paradox, can create some issues. It may get wrong signs of some variables, and then we may interpret the output of a selective inference procedure in the wrong direction.

For example, we run models on a real data set which are discussed before. We can see that in the previous regression, if we regress log(Sale price) on rates the overall condition of the house and central air conditioning. Then the coefficient of the overall condition rates is negative, which means that if a house has a higher overall condition rate, then the house's sale price will be lower. It is obviously wrong and we can say that this interpretation is meaningless. Therefore in the selective inference procedure, we need to avoid Simpson's paradox.

Problem 2 - Selective Inference

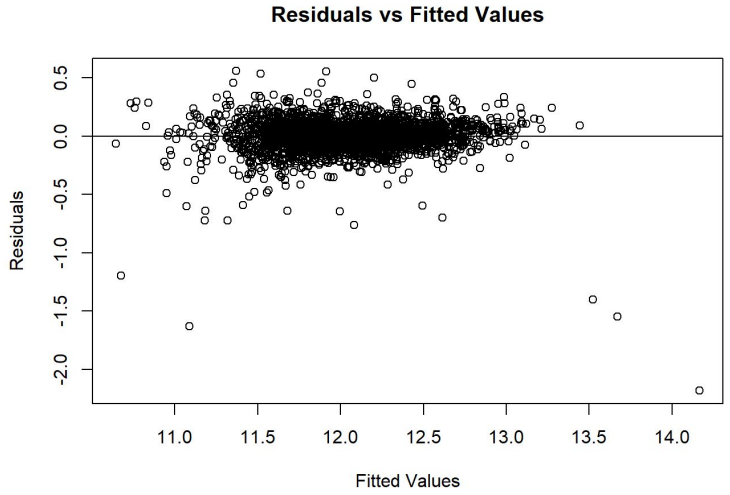
1. Outliers and Other Unusual Observations

To carry out the diagnostics such as outliers, for now, we fit a full linear regression model that includes all the covariates. Actually, if we are going to select a smaller model and report it as a final model, then we should return to check these unusual observations again for the smaller model. Because whether a point is influential and/or appears to be an outlier can change substantially after we make other changes to the model like removing covariates. But here we are just checking the unusual observations generally for the full model to make the data more reasonable so that the linear regression model fits better. Also, note that here we are not checking for the validity of the assumptions of a linear regression model. This will be carried out later in Problem 3.

(1) Missing values

After the data preprocessing carried out in the given R script, now there are n = 2923 data points and p = 49 (not 48 as indicated in the homework) covariates left in the data set. And by checking the missing values, now there are no more NA's in the data set.

(2) Residuals



The "residuals vs fitted values" plot shows that there are some points far away from the other residuals. This may indicate the problem of unusual observations. Therefore, we will check for the unusual observations, i.e. outliers, large leverage points, and influential points, in the following parts. Then, based on these three results, we will decide which observations should be removed from the data set.

(3) Outliers

```
jack[abs(jack)>t][order(abs(jack[abs(jack)>t]),decreasing=TRUE)]
```

```
##       1495       2176        181       2177       1550        372
## -18.453080 -12.791417 -12.114422 -11.540989  -9.100832  -5.501946
##        726       1552       1182        125       1877        709
##  -5.280076  -5.247730  -5.061578  -4.695983   4.693665  -4.637802
##        561       2874
##  -4.607645  -4.352180
```

```
# outlier test
library(car)
outlierTest(model1, n.max=100)
```

```
##        rstudent unadjusted p-value Bonferonni p
## 1495 -18.453080         6.0516e-72   1.7689e-68
## 2176 -12.791417         1.7779e-36   5.1968e-33
## 181  -12.114422         5.5677e-33   1.6274e-29
## 2177 -11.540989         3.7509e-30   1.0964e-26
## 1550  -9.100832         1.6335e-19   4.7746e-16
## 372   -5.501946         4.0869e-08   1.1946e-04
## 726   -5.280076         1.3877e-07   4.0562e-04
## 1552  -5.247730         1.6520e-07   4.8288e-04
## 1182  -5.061578         4.4203e-07   1.2921e-03
## 125   -4.695983         2.7778e-06   8.1196e-03
## 1877   4.693665         2.8093e-06   8.2115e-03
## 709   -4.637802         3.6788e-06   1.0753e-02
## 561   -4.607645         4.2501e-06   1.2423e-02
## 2874  -4.352180         1.3950e-05   4.0775e-02
```

An outlier is a point that does not fit the current model. Here we calculate the jackknife (or externally studentized) residuals, and use a 5% significance level to perform t-tests where a Bonferroni correction is used to correct the significance level. As a result, we find 14 outliers for the full regression model. We also use the outlierTest() function in the car package to confirm our findings. The function does report the same 14 outliers. The indices of these 14 observations are: 1495, 2176, 181, 2177, 1550, 372, 726, 1552, 1182, 125, 1877, 709, 561, 2874. Therefore, we will remove these 14 outliers from the data set.

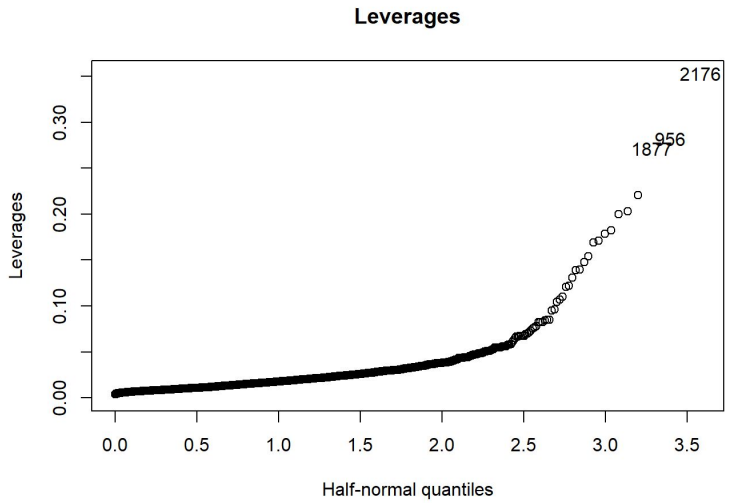
(4) Large leverage points

```
# find large leverage points
diag_H = hatvalues(model1)    # i.e. diag_H is leverages
# or : = influence(model1)$hat
sum(diag_H > 2*mean(diag_H))
```

```
## [1] 184
```

```
large_leverages = diag_H[diag_H>2*mean(diag_H)]
large_leverages = as.numeric(names(large_leverages[order(large_leverages,decreasing=TRUE)]))
large_leverages[1:20]
```

```
##  [1] 2176  956 1877 2493 1495  660 2730    3 1567 2111 2085  364 2273 2565
## [15] 2352 2661 2345 1757 1857 1563
```



A leverage point is unusual in the predictor space, i.e. large leverage values are due to extreme values in X. It has the potential to influence the fit. By calculating the leverages, there are 184 observations that have leverages larger than twice of the mean leverage value. In the above results, only the indices of the first 20 ones are shown. From the above half-norm plot of leverages, we can also see that there are many large leverage data points.

However, a large leverage data point may or may not actually affect the fit. For example, if an extreme X value just lies on the line fitted by other data points, it is not influencing the fitting results. Thus whether a large leverage point is influential or not, we can check it by calculating the cook's distance in the following part.

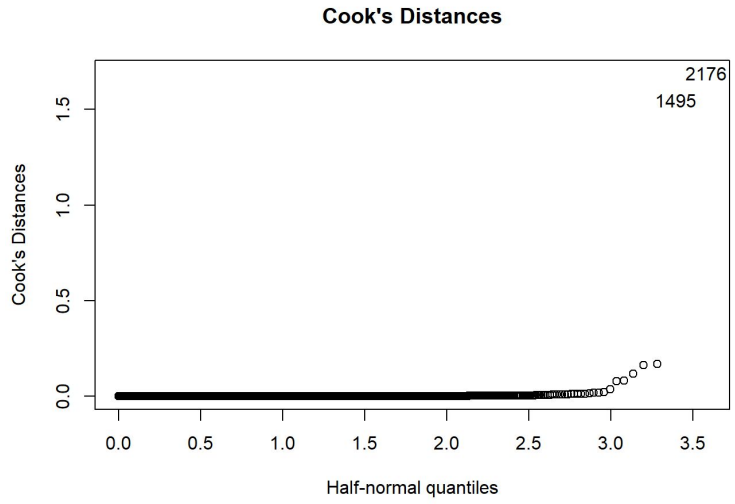
(5) Influential points

```
# find influential points with large Cook's Distance
cook = cooks.distance(model1)
n = nrow(df)
sum(cook > 4/n)
```

```
## [1] 152
```

```
influentials = cook[cook>4/n]
influentials = as.numeric(names(influentials[order(influentials,decreasing=TRUE)]))
influentials[1:18]
```

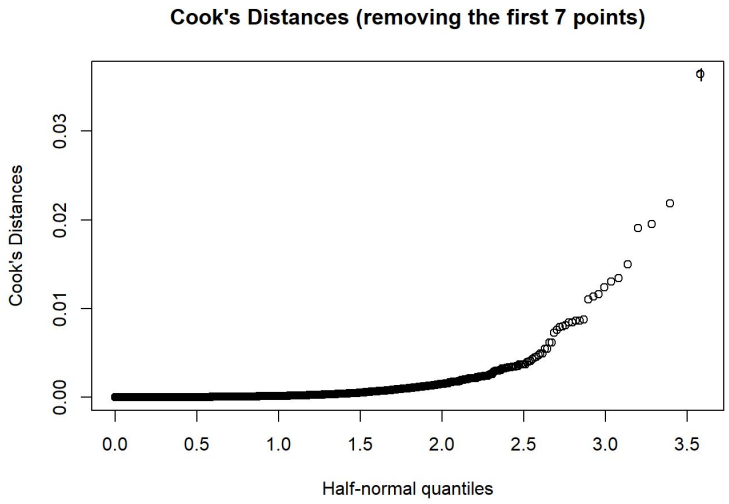
```
##  [1] 2176 1495 1550 1877 2177  181    3 2730  726 1567  956 1182  125 2085
## [15] 1552 2493 1563 2111
```



An influential point is one whose removal from the dataset would cause a large change in the fit. An influential point may or may not be an outlier and may or may not have large leverage but it will tend to have at least one of these two properties.

Generally, a Cook's Distance Di is considered large if Di > 4/n. Here there are 152 observations that have large Cook's Distances thus have large influence on the fitted model. In the above results, only the indices of the first 18 ones are shown. From the above half-norm plot of leverages, we can also see that there are two extremely influential data points: 2176 and 1495. These two observations are also identified as the first two outliers and the two of the first five large leverage data points.

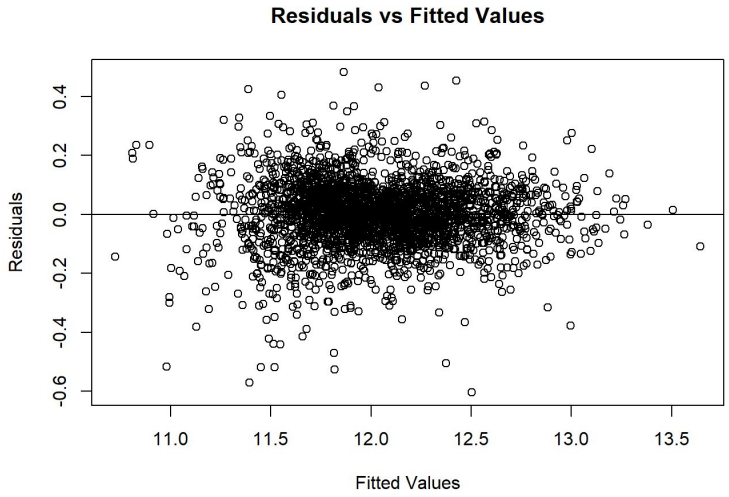
To have a better look at the other influential points and determine which ones to be removed from the data set. Here we also plot the half-norm plot after removing the first 7 influential data points, which are shown to be obviously larger than others in the above figure.



The half-norm plot shows that the 18 most influential data points are good candidates to be removed from the data set. After having a better look at these data points, we can find that they include 10 data points of the 14 outliers and 8 data points of the 20 largest leverage points. Therefore, as a result, we will eventually remove the 18 most influential data points and the other 4 outliers that are not included in these 18 ones.

(6) Removing Unusual Observations

As discussed above, we removed the 22 observations, whose indices are: 2176, 1495, 1550, 1877, 2177, 181, 3, 2730, 726, 1567, 956, 1182, 125, 2085, 1552, 2493, 1563, 2111, 372, 709, 561, 2874. After removing, now there are n = 2901 data points and p = 49 covariates left in the data set. We again plot the residuals against the fitted values as shown below. We can see that now the extreme residuals have been removed. However, the residuals still seem to have non-constant variances that would violate the linear regression assumptions. This issue will be discussed later in Problem 3.



2. Variable Selection

After data preprocessing, now all the variables are numerical. We first standardize X, but we don't standardize Y.

```
df3= scale(df2[,1:49])
Y = df2[,50]
```

Then we represent A and b by R code and normalized Ay to [0,1], so we could choose t between 0 and 1 and easily compare b with Ay. We combine grid search of t with cross validation to find the best t value for selective inference.

```
i=1
for (t in seq(0.1,0.9, by = 0.1)) {
```

Then we use Ay<b to conduct marginal screening.

```
a=t(df3)
S = sign(a%*%(Y))
S = -S
S_mat = replicate(ncol(a),S)

S_mat =as.data.frame(S_mat)


A = S_mat*as.matrix(a)
```

```
Y = as.matrix(Y)
#view(as.matrix(A))
maxnum = -min(as.matrix(A)%*% Y)
multiply = as.matrix(A)%*% Y/maxnum
mse=numeric(9)
```

```
  t = 0.3
  b = replicate(49,-t)
  select = which(multiply<b)
```

After marginal screening, we select a set of variables (“select” in R code) to conduct post-selective inference.

3 Post-selective Inference: p-values for selected coefficients

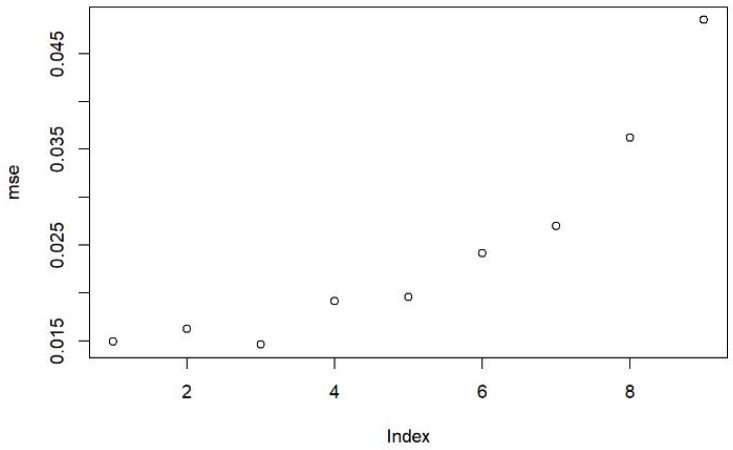
We represent v, prp_v_y in R and use the standard deviation of Y to estimate sigma. Then we plug in these variables to the code to find the P-value of each variable selected from screening.

```
k = sum(multiply<b)
Xs=df3[,select]
c_lower = -Inf; c_upper = Inf
beta=(solve(t(Xs)%*%Xs)%*%t(Xs))%*%Y


p=numeric(k)
for(l in 1:k){
  dim(solve(t(Xs)%*%Xs))
  dim(Xs)
  signL = S[select]
  v=(solve(t(Xs)%*%Xs)%*%t(Xs))[l,]
  v= as.numeric(v)
  prp_v_y = Y-t((t(Y)%*%v)%*%v/sum(v^2)) #double check whether need ^2
  coef1 = sum(A[l,]*v)/sum(v^2)
  coef2 = b[l] - sum(A[l,]*prp_v_y)
  if(coef1>0){c_upper = min(c_upper,coef2/coef1)} # update the upper endpoint
  if(coef1<0){c_lower = max(c_lower,coef2/coef1)} # update the lower endpoint
  beta_l=as.numeric(beta[l,])
  Y_bar = replicate(length(Y), mean(Y))

  beta_l = (beta_l-Y_bar%*%v)/(sqrt(sum(v^2))*sd(Y))
  if (beta_l > 0) {
    p[l] = (pnorm(c_upper) - pnorm(beta_l))/(pnorm(c_upper)-pnorm(c_lower))
    if (p[l] < 0) {
      p[l] = 0
    }
  }
  else {
    p[l] = (pnorm(beta_l) - pnorm(c_lower) )/(pnorm(c_upper)-pnorm(c_lower))
    if (p[l] < 0) {
      p[l] = 0
    }
  }
}
final_selection = select[which(p<0.05)]
print("significant variable for p-value < 0.05")
```

We select the variables whose P-value is less than 0.05 from one side truncated normal test**.**

For each t, we could choose the variables through post-selective inference. We use these variables to fit a linear model and utilize ten-fold cross validation to choose the best t. We plot the MSE for each t from 0.1 to 0.9.

```
  s_data = df2[,c(final_selection, 50)]
  fit = train(Y~., data = s_data,method = "lm", trControl =  trainControl(method = "cv", number = 10, verboseIter = FALSE)
)
  mse[i] <- mean(residuals(fit)^2)
  i = i+1
}
```



From the plot, we could see MSE achieves its minimum when t = 0.3. Thus, t = 0.3 produces the best linear model by selective inference.

For t = 0.3, when we conduct the first step of variable selection through marginal screening, there are 29 variables that pass our marginal screening, which is shown below.

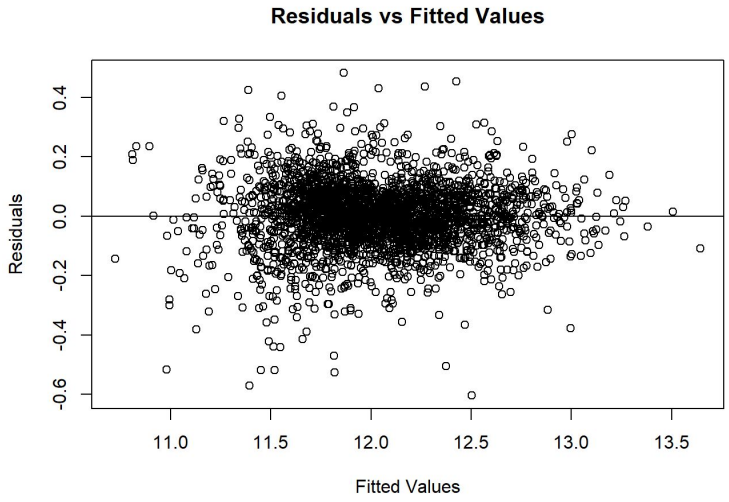
```
##  [1] "Lot.Area"       "Overall.Qual"   "Year.Built"     "Year.Remod.Add"
##  [5] "Mas.Vnr.Area"   "BsmtFin.SF.1"   "1st.Flr.SF"     "2nd.Flr.SF"
##  [9] "Bsmt.Full.Bath" "Full.Bath"      "Half.Bath"      "TotRms.AbvGrd"
## [13] "Fireplaces"     "Garage.Cars"    "Garage.Area"    "Wood.Deck.SF"
## [17] "Open.Porch.SF"  "LotShape"       "ExterQual"      "BsmtQual"
## [21] "BsmtCond"       "BsmtExposure"   "CentralAir"     "KitchenQual"
## [25] "FireplaceQu"    "GarageFinish"   "GarageQual"     "GarageCond"
## [29] "PavedDrive"
```

For post-selective inference, we find there are 27 variables left, so we rule out 2 variables by post-selective inference. The 27 variables are shown below. These are the covariates in our final linear model.

```
##  [1] "Lot.Area"       "Overall.Qual"   "Year.Built"     "Year.Remod.Add"
##  [5] "Mas.Vnr.Area"   "BsmtFin.SF.1"   "1st.Flr.SF"     "2nd.Flr.SF"
##  [9] "Bsmt.Full.Bath" "TotRms.AbvGrd"  "Fireplaces"     "Garage.Cars"
## [13] "Garage.Area"    "Wood.Deck.SF"   "Open.Porch.SF"  "LotShape"
## [17] "ExterQual"      "BsmtQual"       "BsmtCond"       "BsmtExposure"
## [21] "CentralAir"     "KitchenQual"    "FireplaceQu"    "GarageFinish"
## [25] "GarageQual"     "GarageCond"     "PavedDrive"
```

Problem 3 - Checking Assumptions

1. Constant variance



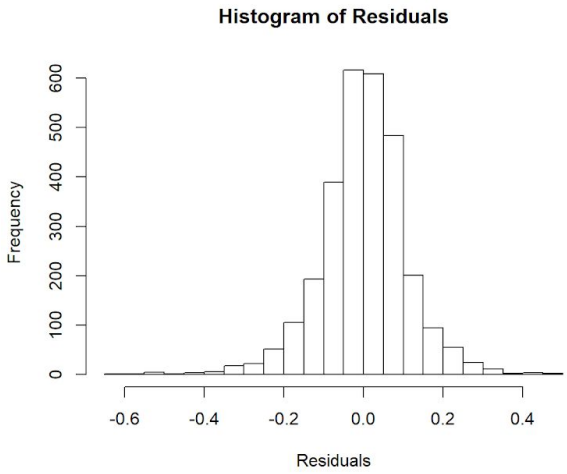
In the “residuals vs fitted values” plot, the residuals seem to have non-constant variances. As the fitted values increases, the variance of the residuals gets smaller. Then we carry out a formal test: Breusch-Pagan test to check heteroscedasticity.

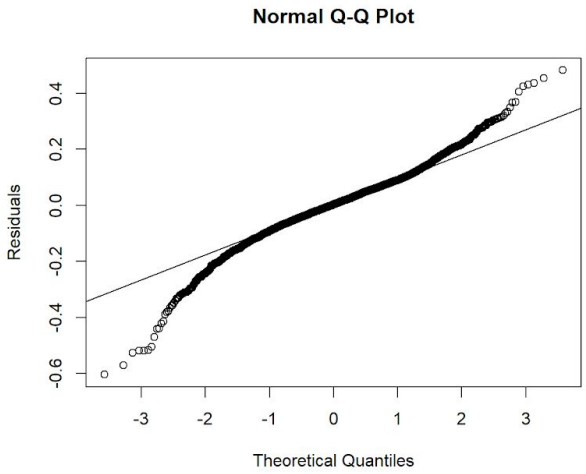
```
# use a formal test: Breusch-Pagan test to check the heteroscedasticity
library(lmtest)
bptest(model2)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  model2
## BP = 253.42, df = 49, p-value < 2.2e-16
```

Breusch-Pagan test’s null hypothesis is the homoscedasticity of the regression model, the alternative being a heteroscedastic model. Here the p-value < 2.2e-16, so we reject the null hypothesis (homoscedasticity) at $\alpha = 0.01$ or even smaller significance level. Therefore, there is very significant evidence against constant variance. As a result, we can conclude that there is a severe heteroscedasticity problem in our full linear regression model. This violates the linear assumption that the errors should have a constant variance. In the last part, we will simulate a data set with a non-constant variance to show the issues that might be caused by it for selective inference.

2. Normality of errors






```
# Shapiro-Wilk test
shapiro.test(model2$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  model2$residuals
## W = 0.97058, p-value < 2.2e-16
```

The main part of the histogram of residuals seems to follow a symmetric, bell-shape. But it is a little right-skewed and the left tail is a little longer than the right tail. So though the main portion of residuals looks normal, the whole distribution seems not to be quite normal.

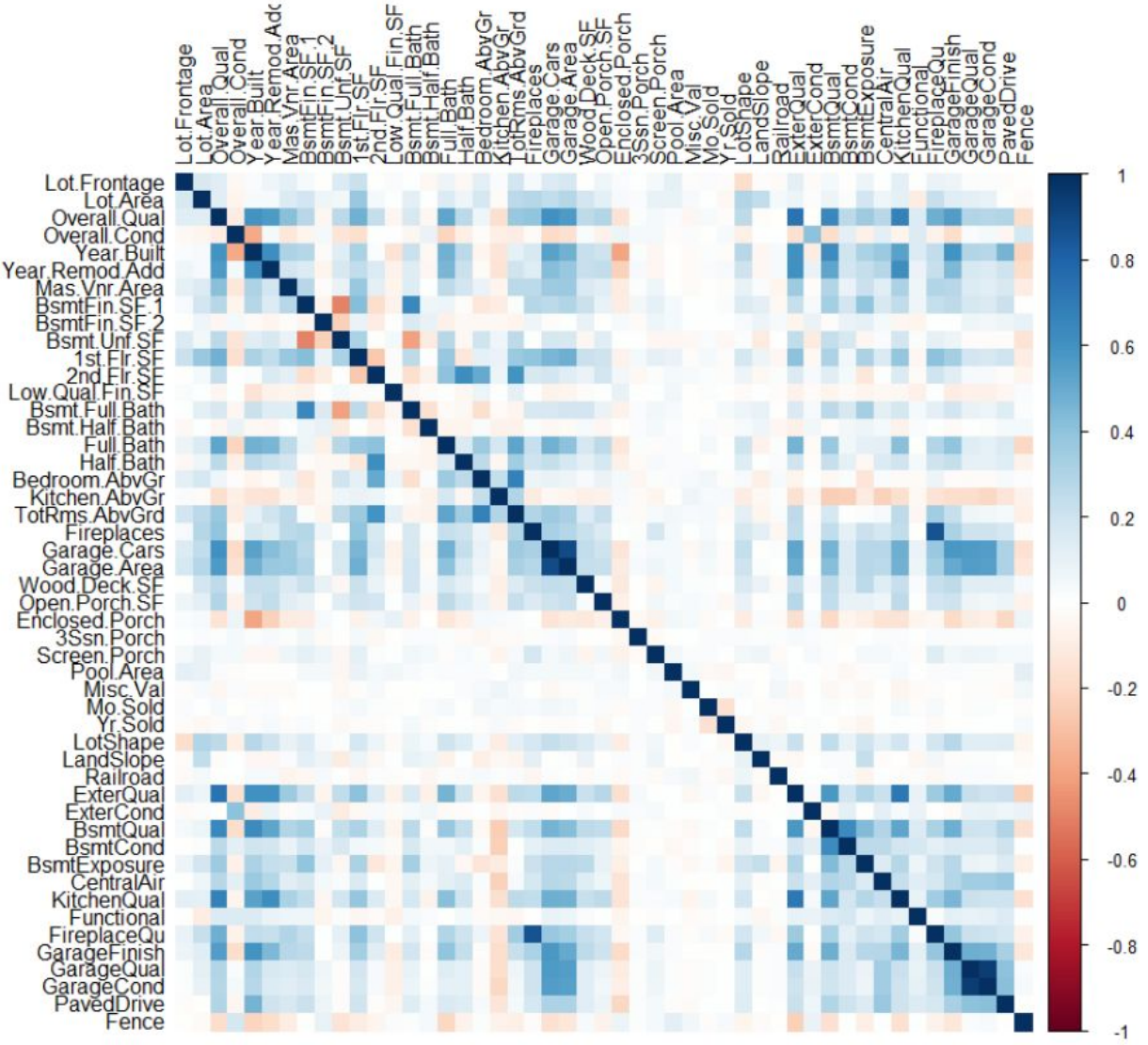
In the Q-Q plot, the middle part of the residuals lies on the line. However, the left tail and right tail do not follow the line, and the left tail is a little longer than the right tail. Since we have already removed the outliers, this looks more like a long-tailed and slightly skewed nonnormality error problem.

We also carried out a formal test to check the normality of errors. The Shapiro-Wilk test's null hypothesis is that data follows a normal distribution. Here the Shapiro-Wilk test has a p-value < 2.2e-16, so we reject the null hypothesis (normal) at $\alpha = 0.01$ or even smaller significance level. Therefore, there is very significant evidence against the normality of the residuals. As a result, we can conclude that the errors are not normally distributed in our full linear regression model, which also violates the linear assumption and may cause problems in selecting models and making inferences. Specifically, here in the selective inference, the nonnormality of errors will make the truncated normal distribution not hold.

When nonnormality of errors is found, the resolution depends on the type of problem found. For the skewness problem, it might be solved by a transformation of the response. Here we have already performed a log-transformation of the response, while the response before the

transformation has a highly skewed distribution. Now the skewness is much better and probably can be ignored. If we want to further remove the skewness, we may need to consider other transformations or removing more influential points. However, we should be careful to remove more influential points, because it may cause overfitting problem or selecting wrong models by subjectively making the data fit the model better. On the other hand, for the long-tailed distribution problem, it's better to use other more robust estimators than the least-squares estimators since now the least-squares estimators may not be optimal and will cause inaccuracies. Or we should explore more into the model structure, consider higher-order linear regressions, or even use models other than linear models.

3. Non-independent data (correlations between covariates)



```
##       [,1]           [,2]             [,3]
##  [1,] "GarageQual"   "GarageCond"     "0.946"
##  [2,] "Garage.Cars"  "Garage.Area"    "0.892"
##  [3,] "Fireplaces"   "FireplaceQu"    "0.864"
##  [4,] "Overall.Qual" "ExterQual"      "0.734"
##  [5,] "Bedroom.AbvGr" "TotRms.AbvGrd" "0.679"
##  [6,] "Overall.Qual" "KitchenQual"    "0.676"
##  [7,] "BsmtFin.SF.1" "Bsmt.Full.Bath" "0.648"
##  [8,] "Overall.Qual" "BsmtQual"       "0.64"
##  [9,] "BsmtCond"     "BsmtQual"       "0.625"
## [10,] "Year.Built"   "BsmtQual"       "0.622"
## [11,] "2nd.Flr.SF"   "Half.Bath"      "0.614"
## [12,] "Year.Built"   "Year.Remod.Add" "0.612"
## [13,] "Year.Built"   "ExterQual"      "0.605"
```

```r
# check the gvif values of each variables
library(car)
gvifs = car::vif(model2)
gvifs[gvifs>5][order(gvifs[gvifs>5],decreasing=TRUE)]
```

```
##   GarageCond   GarageQual BsmtFin.SF.1 `1st.Flr.SF`  Garage.Cars
##    10.229752     9.923546     7.120374     6.630625     6.369878
##  Bsmt.Unf.SF `2nd.Flr.SF`  Garage.Area   Year.Built
##     6.148091     5.784448     5.708352     5.240060
```

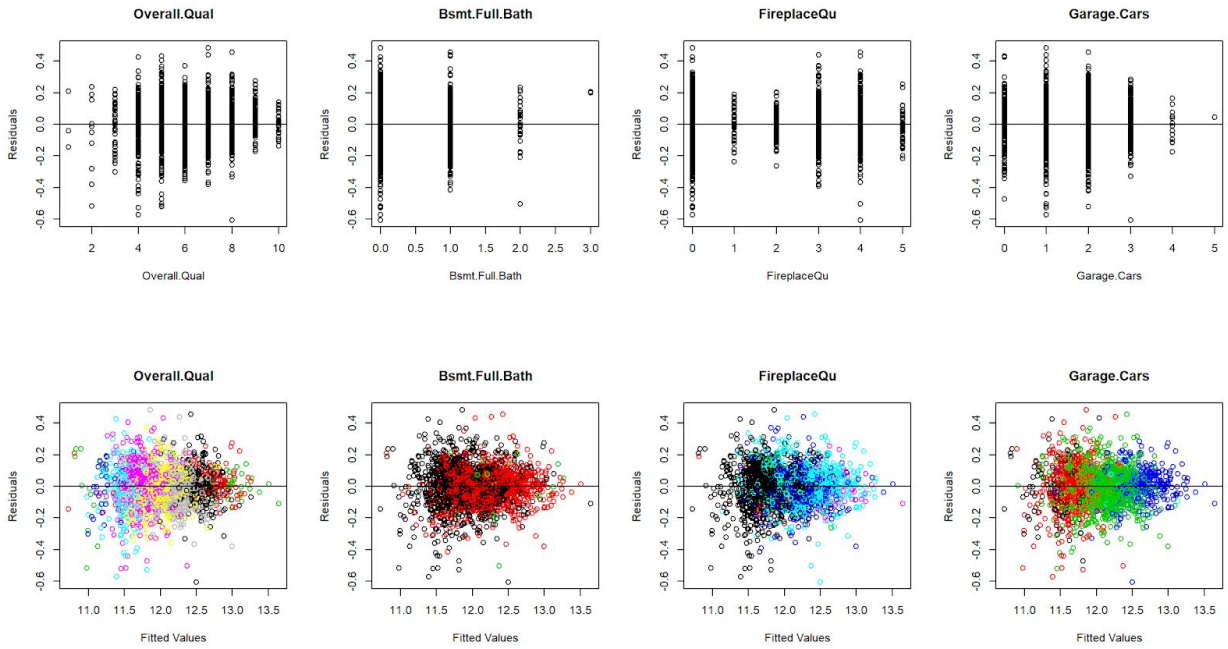
After the data preprocessing carried out in the given R script, now all the covariates are numerical, so we can calculate the pairwise correlations between the numerical covariates. We can see that there are 13 large pairwise correlations (larger than 0.6), which reveals their pairwise collinearities. To formally check if there is any collinearity problem, we also checked the variance inflation factors (VIF). It is suggested that the straightforward VIF can't be used if there are variables with more than one degree of 10 freedom (e.g. categorical variables with more than two levels) and instead we should use the GVIF (generalized variance inflation factor) function in the car package. For continuous variables, the GVIF values are the same as VIF values, however, for categorical variables, GVIF values are the VIFs corrected by the number of degrees of freedom (df) of the categorical variables. In practice, it's common to say that VIF greater than 5 is problematic. Here we see that there are 19 GVIF values greater than 5, which indicates significant variance inflation as well as collinearity problems in this data set.

In linear regression model, collinearity will lead to imprecise estimates of coefficients. The signs of the coefficients can be the opposite of what intuition about the effect of the predictor might suggest. The standard errors will be inflated so that t-tests may fail to reveal significant factors. The fit will become very sensitive to measurement errors where small changes in the response can lead to large change in the estimated coefficients. Also, in the selection inference procedure, it may make the inequalities become not independent from each other.

Therefore, it's better to keep only one predictor within a highly correlated group of predictors in the model, so that we can keep the model simple and reduce the inflated variance. And after

removing the highly correlated covariates in the full model, from where we can perform the following selection procedures.

4. Clusters/different distributions within different subpopulations



After exploring for the distributions of residuals for all the originally categorical variables, here we choose some of the interesting ones to show. We can see that the residuals don't have the same variance and also don't have the same distribution for different levels within a categorical variable. This may result in different distributions for different subpopulations. Therefore, it's better to further explore into the subpopulations and check whether making different models for different subpopulations would make the models fit better, which may also solve the problems discussed above such as having non-constant variance in the fitted model.

5. Simulation: non-constant variance problem for selective inference

As we find that we have non-constant variance. We may meet some issues and select some variables which are not the real factor of the response. Non-constant variance happens may because there exists some correlation between the response and the predictors, or there are correlations between some of the predictors and the noise. And non-constant variance may get wrong results from our regression and selection. We will simulate some data to show why non-constant variance will have some problems.

In our simulation, we set n=10000, the full model size p=50, and true model size s=15. So we generate a 10000*50 random matrix as X. Then we standardize the data and create a vector beta, which has dimension d=50. And we set the first 15 entries of beta randomly from

Unif(-8,8), and other entries are still 0. And we set epsilon as non-constant variance, the first 2500 points have sd 20, next 2500 points have sd 1, next 2500 points have sd 30, and final 2500 points have sd 15. We define Y as below:

$$Y = X^T * \beta + \varepsilon$$

```
set.seed(100)
n=10000
p=50
s=15 #true model size
X=matrix(rnorm(n*p),n,p)
X=scale(X,center=FALSE,scale=sqrt(colSums(X^2)))
beta=rep(0,p)
beta[1:s]=runif(s,-8,8)
epsilon=c(rnorm(2500,0,20),rnorm(2500,0,1),rnorm(2500,0,30),rnorm(2500,0,15))
Y=X%*%beta+epsilon
```

So we can easily see that the true predictors are the first 15 columns of matrix X out of 50 columns. So in the selection procedure, we will set t=0.3. However, if we use the marginal screening selection procedure coded in problem 2 with t=0.3. Then we will select predictors as below.

```
##  [1]  1  2  3  4  6  7 11 13 15 18 23 25 31 33 34 36 37 38 39 40 41 42 43
## [24] 44 45 46 49
```

We can see that if the model is non-constant variance, we will select many predictors which are not true predictors. And we may think these variables are meaningless. However, if we set epsilon as constant variance, and the data are defined the same way.

```
set.seed(100)
n=10000
p=50
s=15 #true model size
X=matrix(rnorm(n*p),n,p)
X=scale(X,center=FALSE,scale=sqrt(colSums(X^2)))
beta=rep(0,p)
beta[1:s]=runif(s,-8,8)
epsilon=rnorm(10000)
Y=X%*%beta+epsilon
```

However, this time by using the marginal screen selection procedure with t=0.5, we can see that this time, we select predictors which are all true predictors. Therefore, we can tell that non-constant variance problems do have issues for the selection procedure. And if we have non-constant variance, we may select many predictors which are meaningless and can be considered as wrong selection for our interpretation.

```
## [1]  1  2  6  7  8 10 11 12 13
```