Regression

Aarya Patil and Austin Girouard

How Does Linear Regression Work?

Linear regression works by modelling a predictive function using single or multiple variables from a data set. This lets us see if there is a pattern within the data set and, if there is, gives us the ability to predict data based on test data.

Linear regression is a useful tool because it is simple to understand and implement and can be adjusted to handle shortcomings of other methods, such as over/under fitting. There are also many built in functionalities for performing linear regression in R.

Linear regression does fall short when it comes to confounding variables and hidden variables. Situation where variables correlate with the target and predictor or where variables seem to correlate but it is just by chance can lead to the linear regression model being a poor choice.

Load the Data

```
df <- read.csv ("job_profitability.csv", header = TRUE) # specifies where to load data from
df <- df[,c(3, 4, 5, 10)] # specifies which columns we want
str(df) # print data frame structure
```

```
## 'data.frame':    14479 obs. of  4 variables:
##  $ Jobs_Gross_Margin: num  -4.01 254.13 151.83 -32.15 222.7 ...
##  $ Labor_Pay        : num  0 91 0 0 0 ...
##  $ Labor_Burden     : num  22.2 14.9 133.2 81.2 66.3 ...
##  $ Jobs_Total       : num  79.5 360 289 49 289 ...
```

Sample Training and Testing Data

```
set.seed(1234)
i <- sample(1:nrow(df), nrow(df) * 0.8, replace = FALSE) # split data into 80/20 train/test
train <- df[i, ] # training data
test <- df[-i, ] # testing data
```

Use 5 R Functions for Data Exploration

1. Create a summary of the data

```
summary(df) # creates a summary of the data
```

```
##  Jobs_Gross_Margin    Labor_Pay         Labor_Burden        Jobs_Total
##  Min.   :-11522.96   Min.   :   0.00   Min.   :   0.10   Min.   :  -50.0
##  1st Qu.:    -60.41   1st Qu.:  50.12   1st Qu.:  23.37   1st Qu.:    0.0
##  Median :    100.61   Median :  78.46   Median :  48.17   Median :  251.0
##  Mean   :    297.54   Mean   : 108.24   Mean   :  82.52   Mean   :  599.5
##  3rd Qu.:    431.64   3rd Qu.: 128.19   3rd Qu.:  96.66   3rd Qu.:  714.0
##  Max.   : 19446.88   Max.   :3066.42   Max.   :5940.65   Max.   :34104.4
```

2. Display the number of rows in the data set

```
nrow(df) # generates the number of rows
```

```
## [1] 14479
```

3. Display the number of missing values in each variable

```
df[df == 0.00] <- NA # replaces all values of 0.00 with NA
colSums(is.na(df)) # returns the number of missing values in each variable
```

```
## Jobs_Gross_Margin         Labor_Pay      Labor_Burden        Jobs_Total
##                 0               348                 0              3957
```

4. Output the outliers in the data set

```
df[is.na(df)] <- 0 # replaces all values of NA with 0

# Code for finding the outliers of Jobs_Total
quartiles <- quantile(df$Jobs_Total, probs = c(.25, .75), na.rm = FALSE)
IQR <- IQR(df$Jobs_Total)

Lower <- quartiles[1] - 1.5 * IQR
Upper <- quartiles[2] + 1.5 * IQR

df_outliers <- subset(df, df$Jobs_Total <= Lower | df$Jobs_Total >= Upper) # saves outliers of Jobs_Tot

# Code for finding the outliers of Jobs_Gross_Margin
quartiles <- quantile(df$Jobs_Gross_Margin, probs = c(.25, .75), na.rm = FALSE)
IQR <- IQR(df$Jobs_Gross_Margin)

Lower <- quartiles[1] - 1.5 * IQR
Upper <- quartiles[2] + 1.5 * IQR

df_outliers <- subset(df, df$Jobs_Gross_Margin <= Lower | df$Jobs_Gross_Margin >= Upper) # saves outlie

# Code for finding the outliers of Labor_Pay
quartiles <- quantile(df$Labor_Pay, probs = c(.25, .75), na.rm = FALSE)
IQR <- IQR(df$Labor_Pay)

Lower <- quartiles[1] - 1.5 * IQR
Upper <- quartiles[2] + 1.5 * IQR

df_outliers <- subset(df, df$Labor_Pay <= Lower | df$Labor_Pay >= Upper) # saves outliers of Labor_Pay

# Code for finding the outliers of Labor_Burden
quartiles <- quantile(df$Labor_Burden, probs = c(.25, .75), na.rm = FALSE)
IQR <- IQR(df$Labor_Burden)

Lower <- quartiles[1] - 1.5 * IQR
Upper <- quartiles[2] + 1.5 * IQR

df_outliers <- subset(df, df$Labor_Burden <= Lower | df$Labor_Burden >= Upper) # saves outliers of Labo

# Output the outliers
str(df_outliers) # outputs the outliers
```

```
## 'data.frame':    1160 obs. of  4 variables:
##  $ Jobs_Gross_Margin: num  731 6657 2091 5171 1486 ...
##  $ Labor_Pay        : num  420 695 193 782 243 ...
##  $ Labor_Burden     : num  208 387 208 408 261 ...
##  $ Jobs_Total       : num  1494 10297 3156 7056 2617 ...
```

```
summary(df_outliers) # outputs a summary
```

```
##  Jobs_Gross_Margin    Labor_Pay       Labor_Burden      Jobs_Total
##  Min.    :-11523.0   Min.   :   0.0   Min.   : 206.8   Min.   :  -50.0
##  1st Qu.:    166.2   1st Qu.: 206.6   1st Qu.: 241.5   1st Qu.:  850.5
##  Median :    780.7   Median : 259.7   Median : 293.9   Median : 1701.7
##  Mean   :    934.3   Mean   : 328.8   Mean   : 373.2   Mean   : 2176.9
##  3rd Qu.:   1578.7   3rd Qu.: 361.9   3rd Qu.: 390.9   3rd Qu.: 2928.8
##  Max.   : 19446.9   Max.   :3066.4   Max.   :5940.6   Max.   :34104.4
```

5. Find correlations between columns in the data (Labor_Pay and Labor_Burden, Jobs_Gross_Margin and Jobs_Total)

```
cor.test(df$Labor_Pay, df$Labor_Burden, use = "complete") # finds correlations between Labor_Pay and La
```

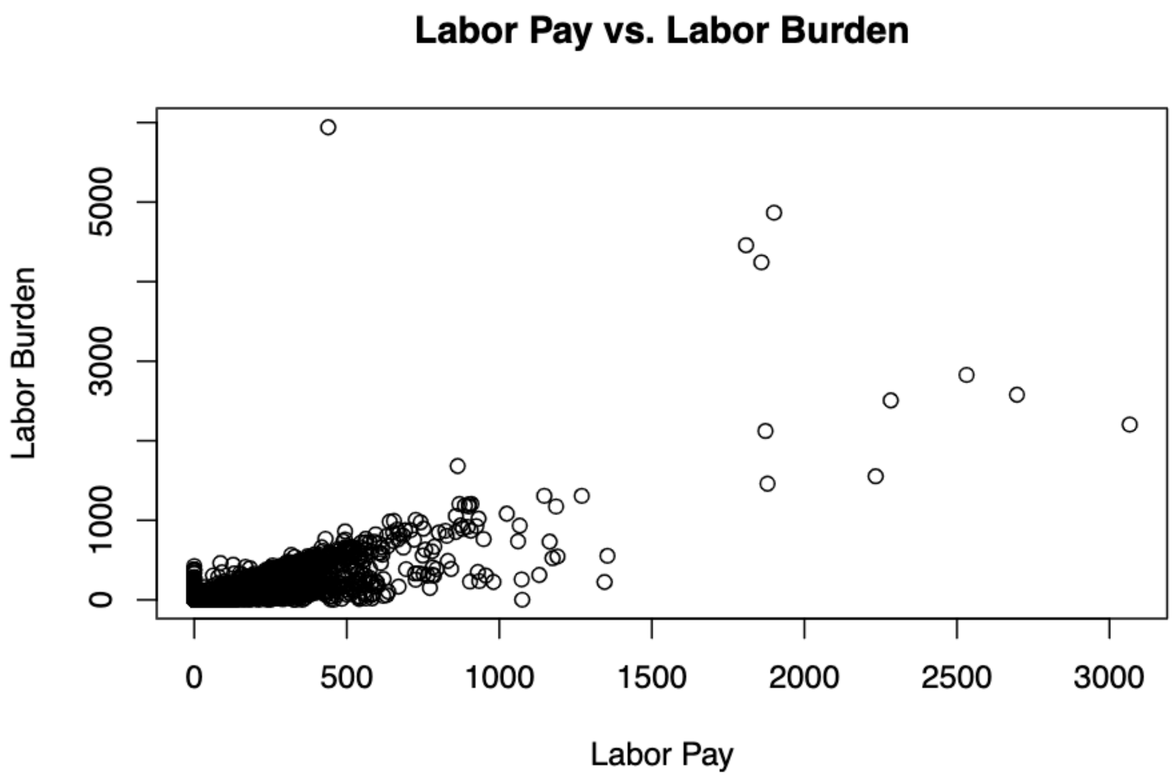
```
##
##  Pearson's product-moment correlation
##
## data:  df$Labor_Pay and df$Labor_Burden
## t = 138.69, df = 14477, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.7482670 0.7622593
## sample estimates:
##       cor
## 0.7553492
```

```
cor.test(df$Jobs_Gross_Margin, df$Jobs_Total, use = "complete") # finds correlations between Jobs_Gross
```

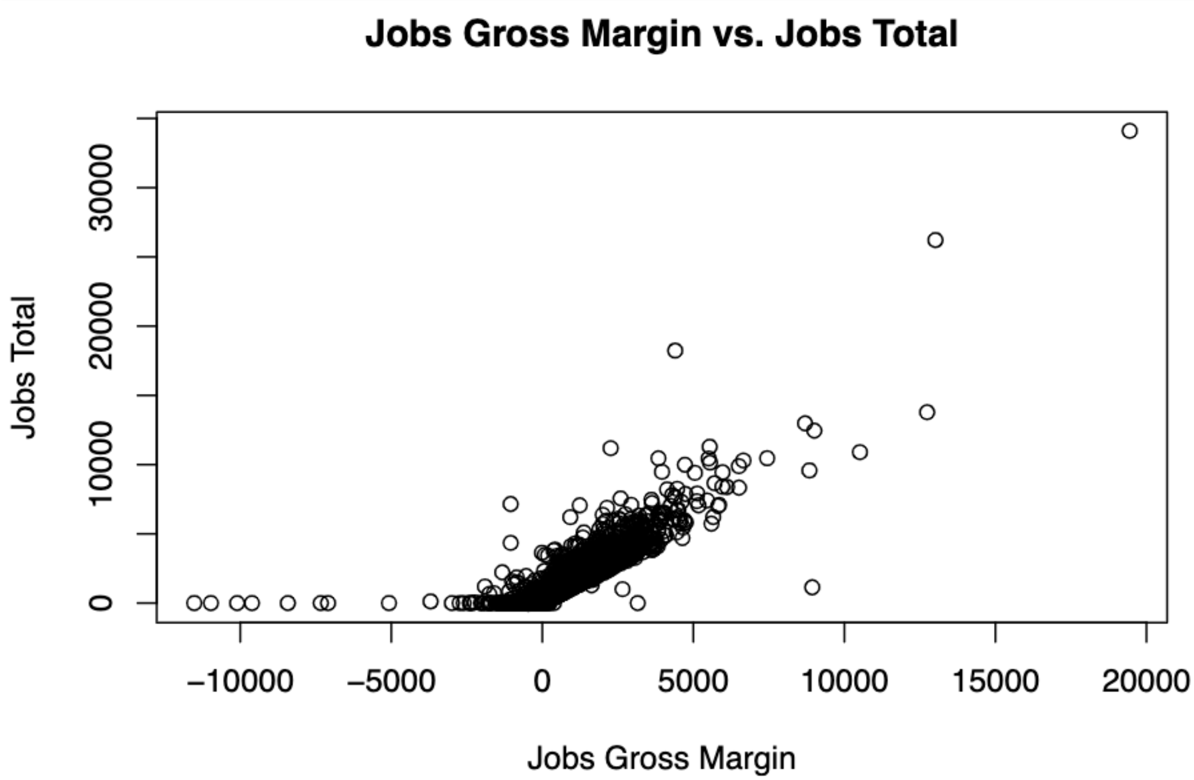
```
##
##  Pearson's product-moment correlation
##
## data:  df$Jobs_Gross_Margin and df$Jobs_Total
## t = 227.62, df = 14477, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.8804737 0.8875900
## sample estimates:
##       cor
## 0.8840831
```

Create 2 Informative Graphs

```
# Graph 1
plot(df$Labor_Pay, df$Labor_Burden, main = "Labor Pay vs. Labor Burden", xlab = "Labor Pay", ylab = "La
```




```r
# Graph 2
plot(df$Jobs_Gross_Margin, df$Jobs_Total, main = "Jobs Gross Margin vs. Jobs Total", xlab = "Jobs Gross
```

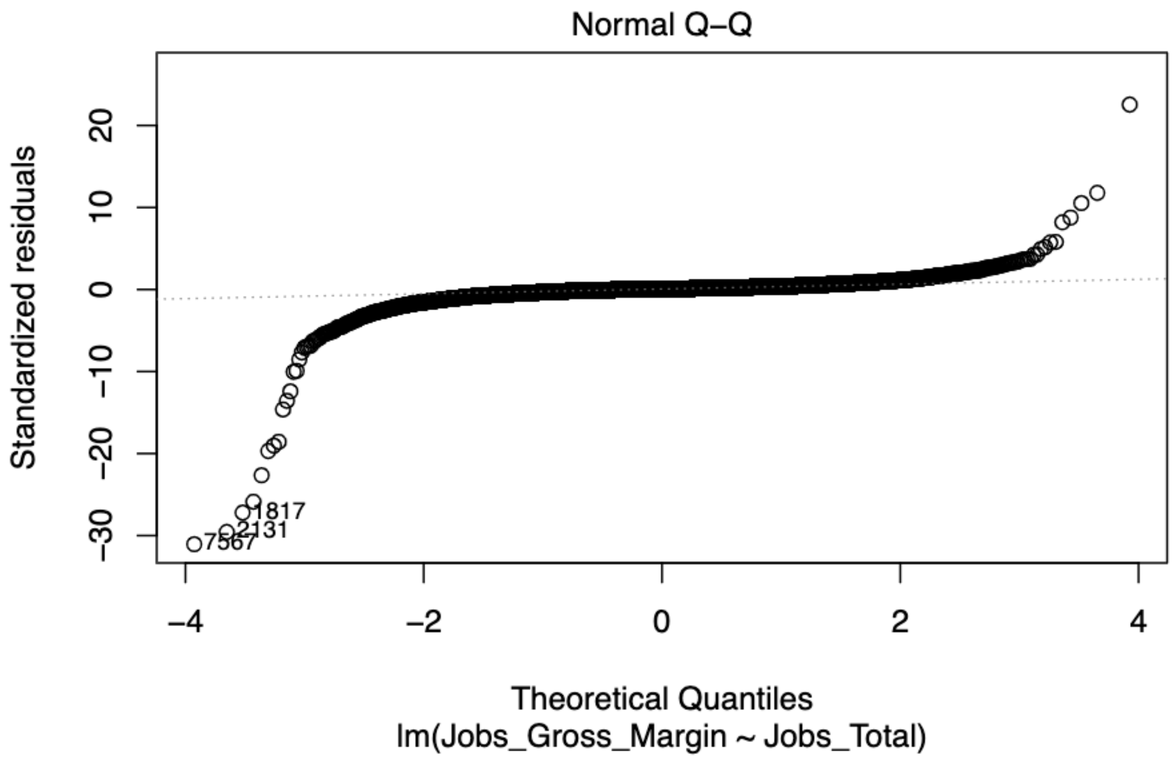


Build a Simple Linear Regression Model

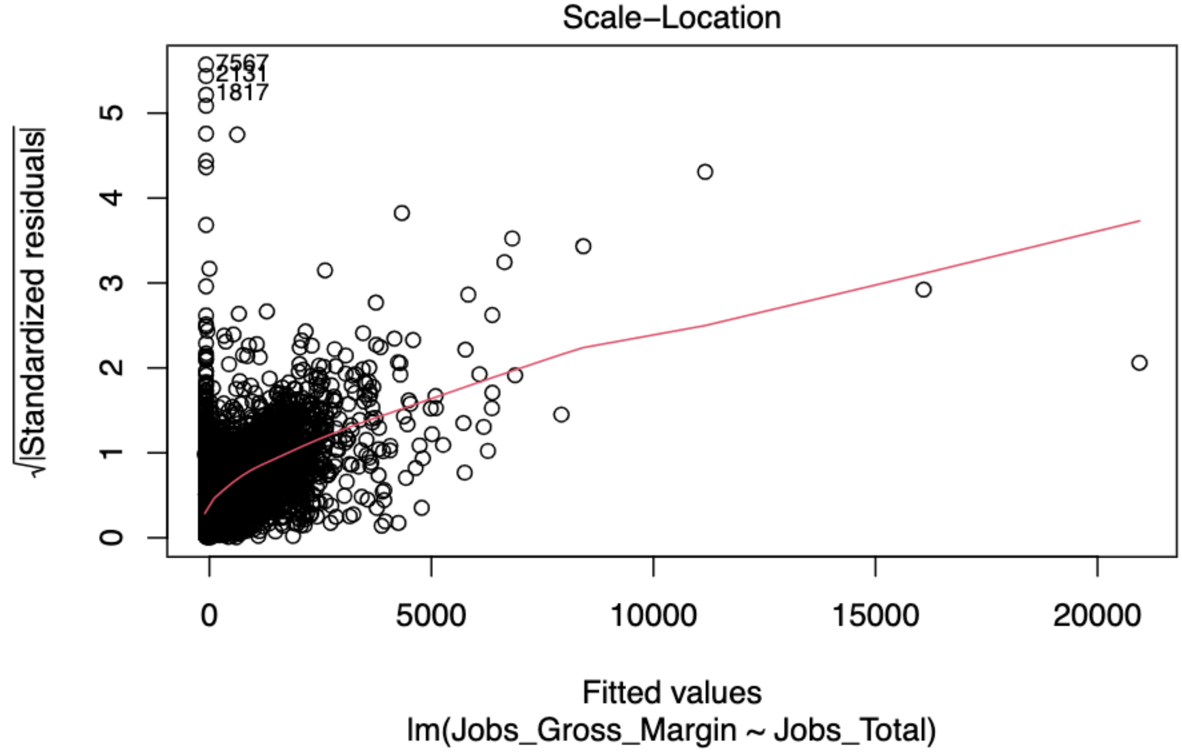
```
lm1 <- lm(Jobs_Gross_Margin ~ Jobs_Total, data = train) # builds linear regression model with single pr
summary(lm1) # shows the linear regression model summary
```

```
##
## Call:
## lm(formula = Jobs_Gross_Margin ~ Jobs_Total, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11449.0    -50.2     25.5     95.6   8311.4
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -73.952300   3.922526  -18.85   <2e-16 ***
## Jobs_Total    0.616336   0.003178  193.94   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 368.7 on 11581 degrees of freedom
## Multiple R-squared:  0.7646, Adjusted R-squared:  0.7646
## F-statistic: 3.761e+04 on 1 and 11581 DF,  p-value: < 2.2e-16
```

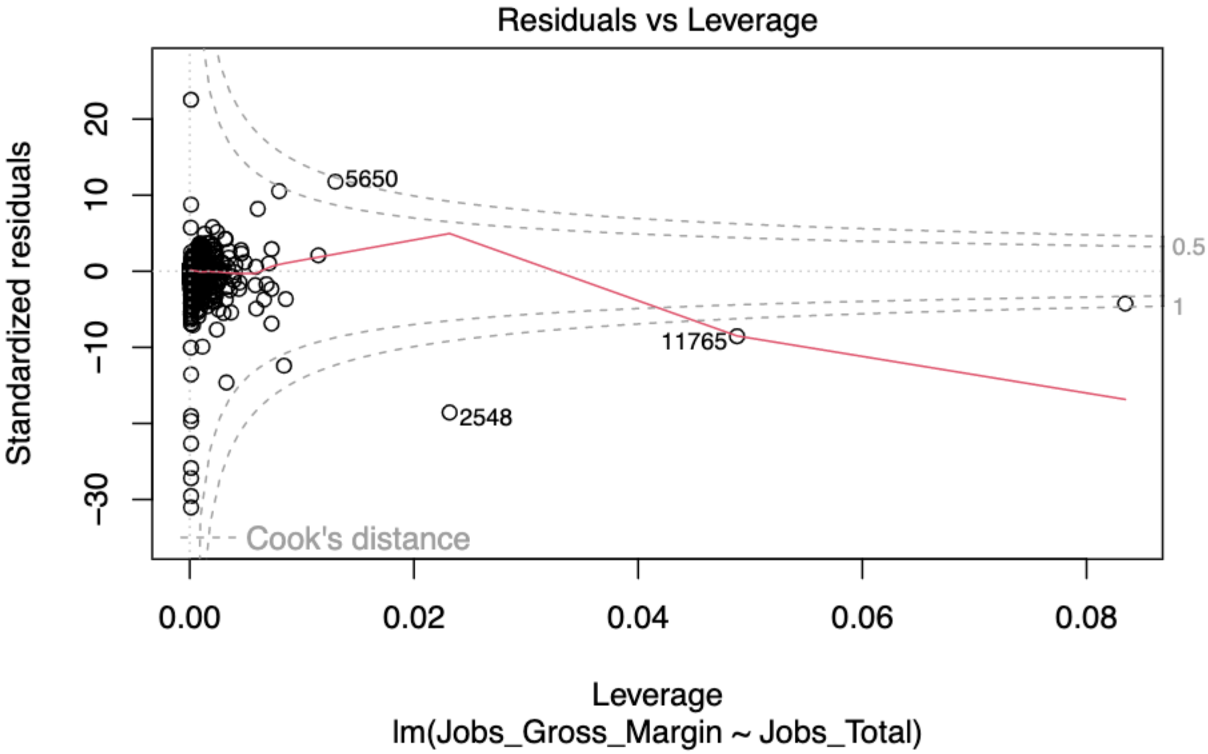
```
plot(lm1) # displays the linear regression model plots
```



Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(Jobs_Gross_Margin ~ Jobs_Total)
1817
2131
7567



Scale−Location
√|Standardized residuals|
Fitted values
lm(Jobs_Gross_Margin ~ Jobs_Total)
7567
2131
1817



Residuals Explanation

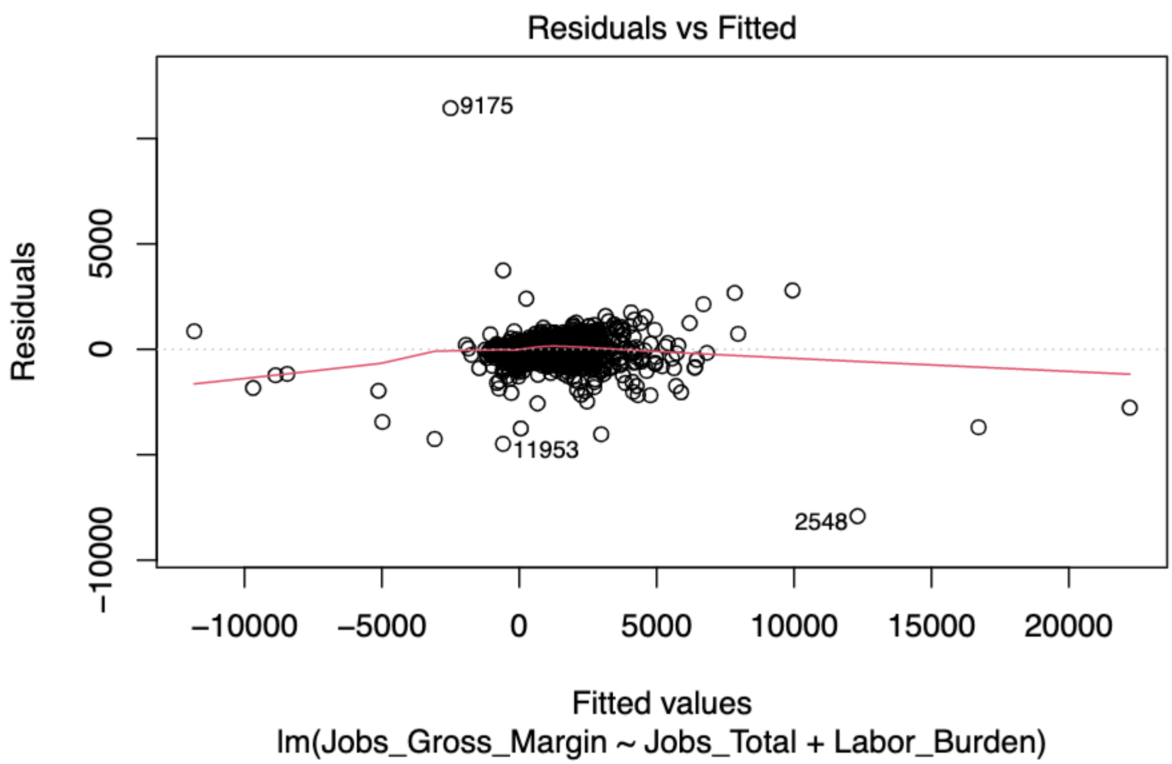
1. Residuals vs. Fitted: A good residual plot for a linear model should show a straight horizontal line. The first Residuals vs Fitted plot shows a fairly straight horizontal line across our data. This means our data is captured effectively by a linear regression model.
2. Normal Q-Q: A Q-Q plot shows if residuals are normally distributed. Again, the goal is to see a straight horizontal line across the plot. Here, we can see that most of the data points fall on a straight line down the middle, meaning the data is roughly normally distributed. We do have some outliers on the left and right sides, which could cause potential problems in our testing.
3. Scale-Location: A Scale-Location plot shows if residuals are spread equally along the ranges of predictors which can be used to check the assumption of equal variance, AKA homoscedasticity. The goal is to see a horizontal line with uniformly random spread points across the whole plot. In our Scale-Location plot, we can see that our data is not very equally spread along the ranges of predictors because the generated line runs steeply from the bottom left to the top right of our graph. This is mostly caused by the existence of outliers seen in our Q-Q plot which are widely increasing the range of our fitted values (from about 5000 to 20000+).
4. Residual vs. Leverage: Finally, the Residuals-Leverage plot helps us find influential outliers within our linear regression analysis. While outliers can exist in data sets, not all majorly impact the results of data analysis. We can see if an outlier is influential if it falls outside of Cook's distance line (a higher Cook's distance means more influence). If there are a couple of data points that are heavily influencing regression results, it may be a good idea to take a closer look into why this is. In our Residuals vs Leverage plot, we can see 2 data points that fall outside the Cook's distance line. These are causing our linear regression model to output skewed results. By removing these 2 points, we could improve the R^2 value of our model.

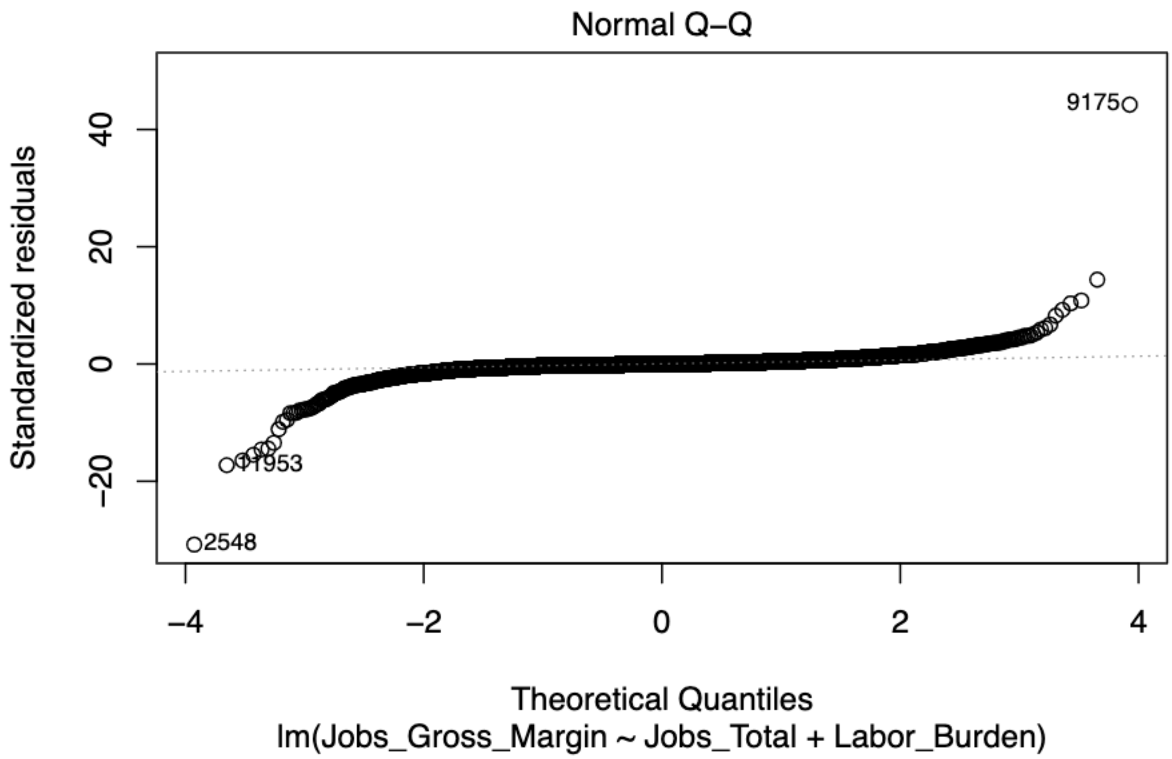
Build a Multiple Linear Regression Model

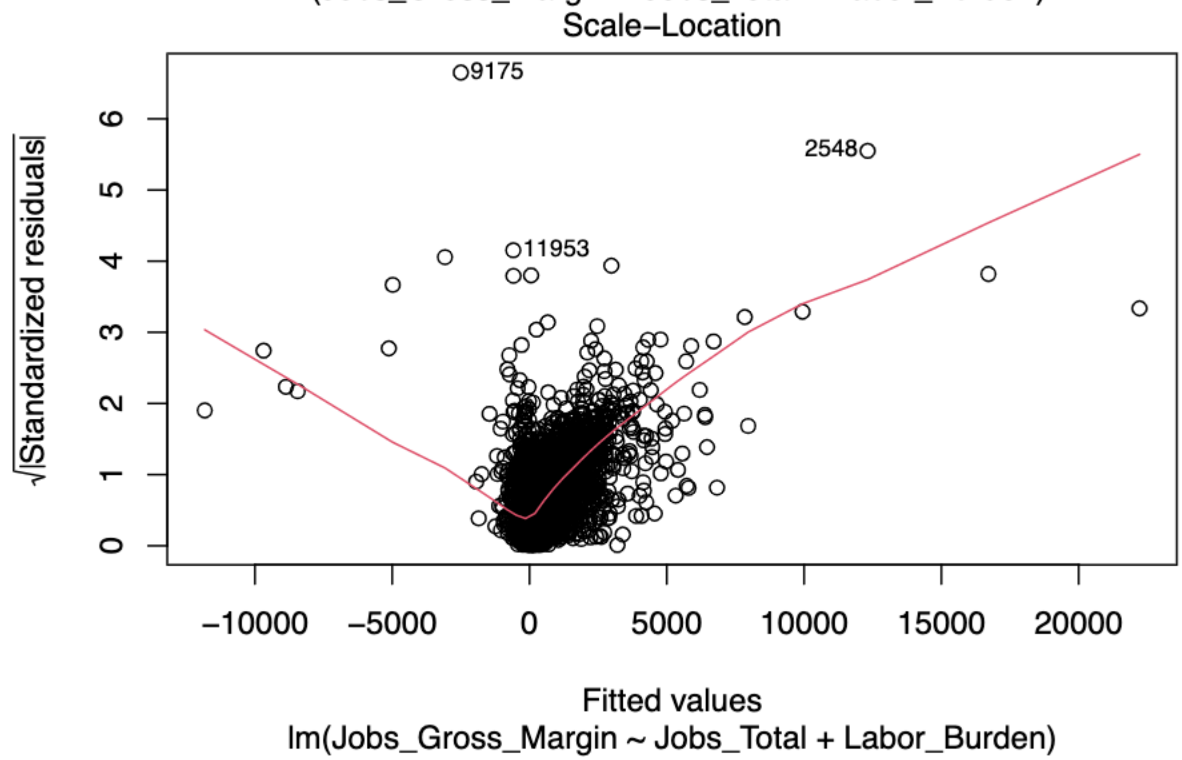
```
lm2 <- lm(Jobs_Gross_Margin ~ Jobs_Total + Labor_Burden, data = train) # builds linear regression model
summary(lm2) # shows the linear regression model summary
```

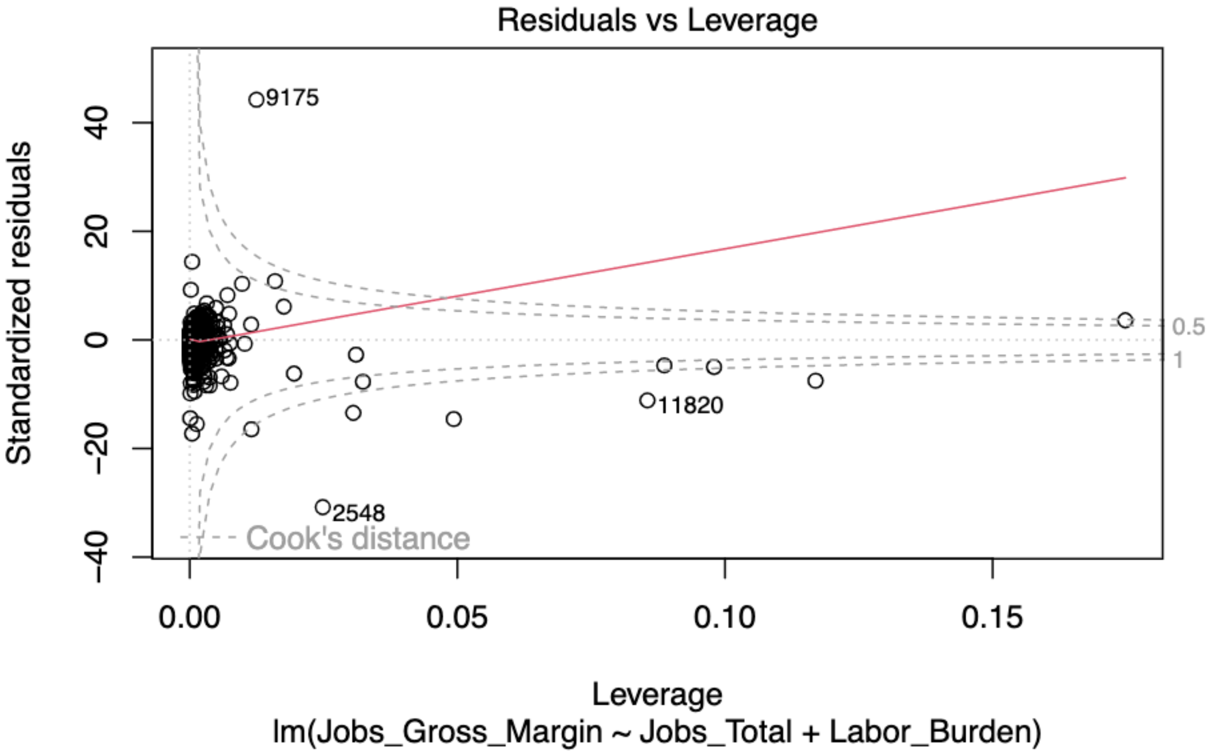
```
##
## Call:
## lm(formula = Jobs_Gross_Margin ~ Jobs_Total + Labor_Burden, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7914.1   -49.8    -6.1    63.3 11436.3
##
## Coefficients:
##               Estimate Std. Error  t value Pr(>|t|)
## (Intercept)  19.976771   2.900934    6.886 6.02e-12 ***
## Jobs_Total    0.736218   0.002502  294.291  < 2e-16 ***
## Labor_Burden -1.994211   0.018451 -108.082  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 260.2 on 11580 degrees of freedom
## Multiple R-squared:  0.8828, Adjusted R-squared:  0.8828
## F-statistic: 4.362e+04 on 2 and 11580 DF,  p-value: < 2.2e-16
```

```
plot(lm2) # displays the linear regression model plots
```




Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(Jobs_Gross_Margin ~ Jobs_Total + Labor_Burden)
9175
11953
2548


Scale−Location
√|Standardized residuals|
Fitted values
lm(Jobs_Gross_Margin ~ Jobs_Total + Labor_Burden)
9175
2548
11953



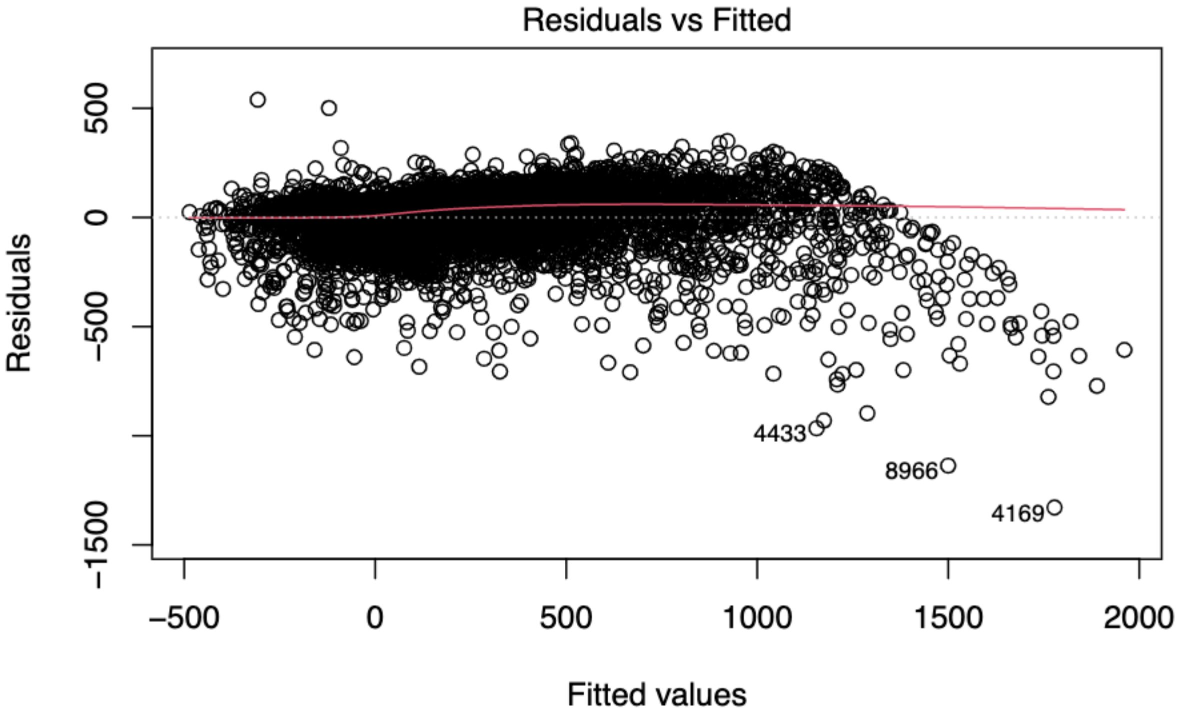
Build a Third Linear Regression Model (Improve the results)

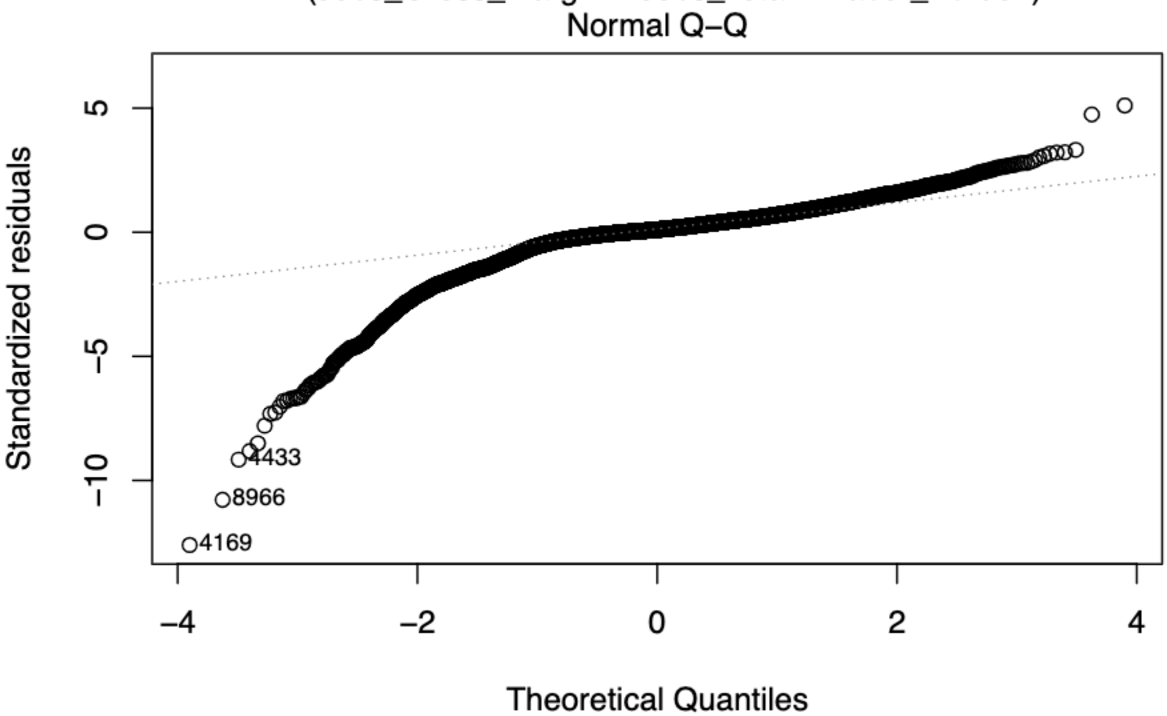
```
# Code to remove the outliers in Jobs_Total
quartiles <- quantile(df$Jobs_Total, probs = c(.10, .90), na.rm = FALSE)
IQR <- IQR(df$Jobs_Total)

Lower <- quartiles[1] - 1.5 * IQR
Upper <- quartiles[2] + 1.5 * IQR

df_no_outlier <- subset(df, df$Jobs_Total > Lower & df$Jobs_Total < Upper)

df <- df_no_outlier # update df to have no outliers

# Code to remove the outliers in Jobs_Gross_Margin
quartiles <- quantile(df$Jobs_Gross_Margin, probs = c(.10, .90), na.rm = FALSE)
IQR <- IQR(df$Jobs_Gross_Margin)

Lower <- quartiles[1] - 1.5 * IQR
Upper <- quartiles[2] + 1.5 * IQR

df_no_outlier <- subset(df, df$Jobs_Gross_Margin> Lower & df$Jobs_Gross_Margin< Upper)

df <- df_no_outlier # update df to have no outliers

# Code to remove the outliers in Jobs_Total
quartiles <- quantile(df$Labor_Burden, probs = c(.10, .90), na.rm = FALSE)
IQR <- IQR(df$Labor_Burden)

Lower <- quartiles[1] - 1.5 * IQR
```

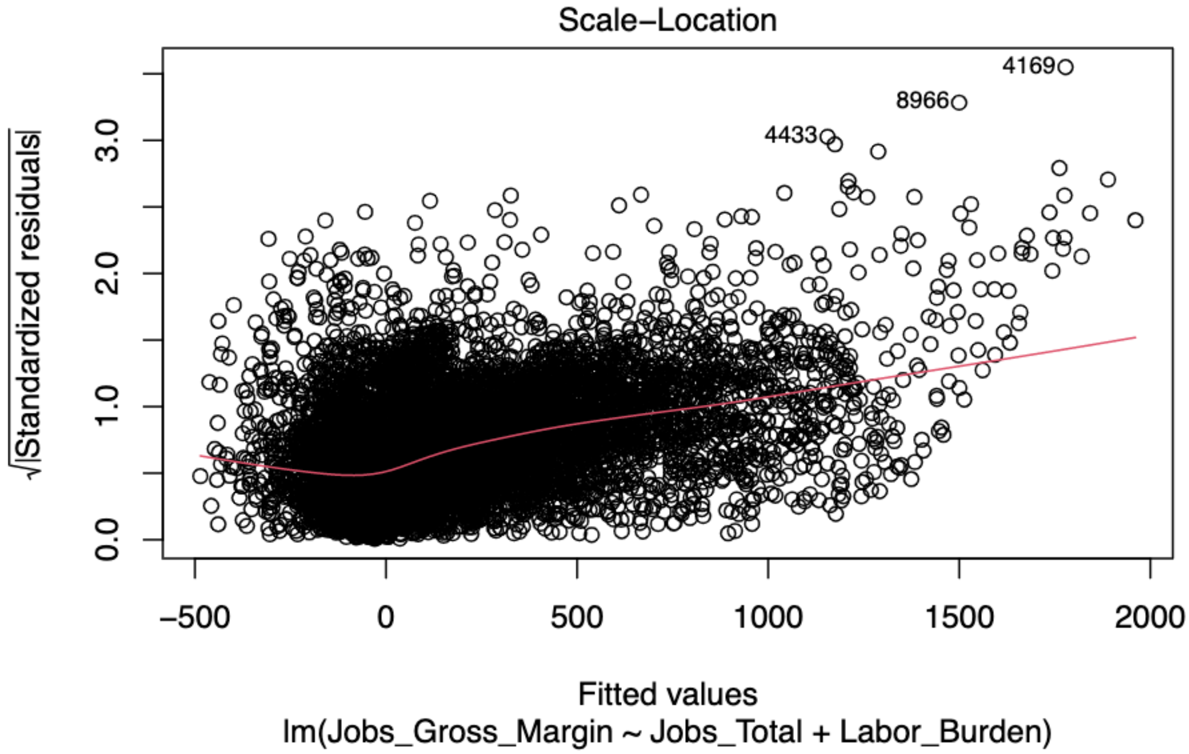
```r
Upper <- quartiles[2] + 1.5 * IQR

df_no_outlier <- subset(df, df$Labor_Burden > Lower & df$Labor_Burden < Upper)

df <- df_no_outlier # update df to have no outliers

# Resampling test and training data with cleaned data set
set.seed(1234)
i <- sample(1:nrow(df), nrow(df) * 0.8, replace = FALSE) # split data into 80/20 train/test
train <- df[i, ] # training data
test <- df[-i, ] # testing data

# Plot the model
lm3 <- lm(Jobs_Gross_Margin ~ Jobs_Total + Labor_Burden, data = train) # builds linear regression model
summary(lm3) # shows the linear regression model summary
```

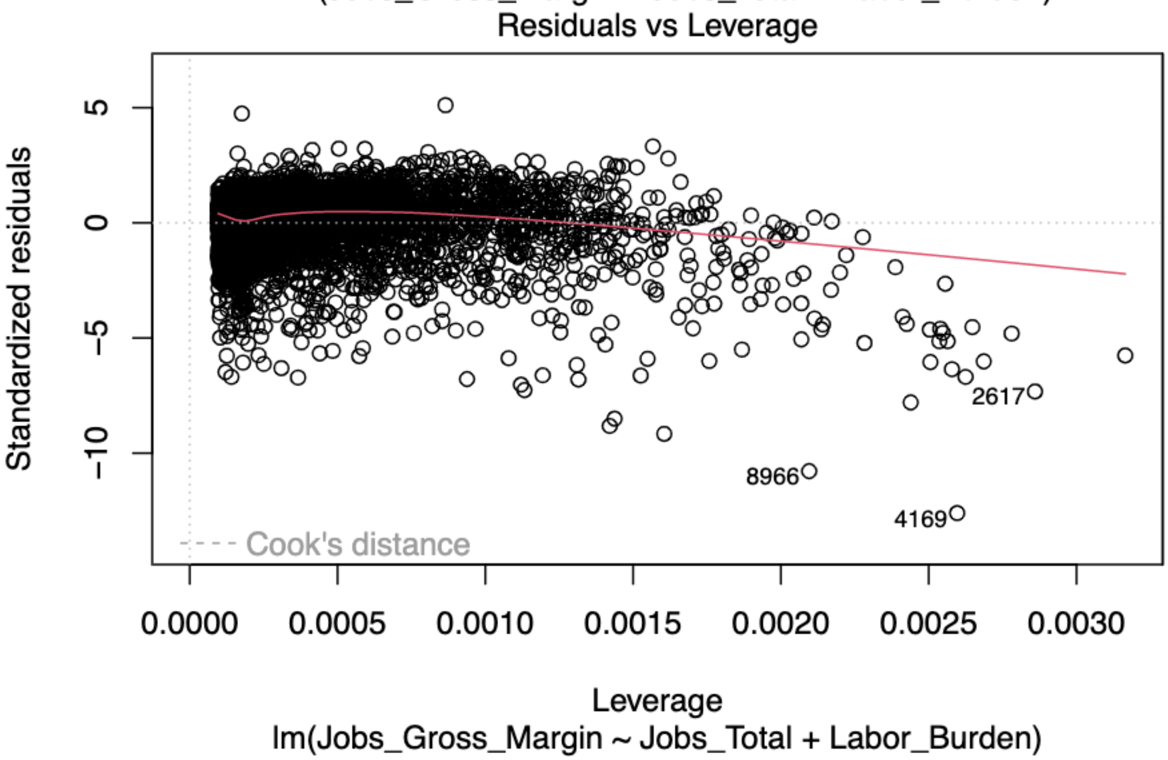
```
##
## Call:
## lm(formula = Jobs_Gross_Margin ~ Jobs_Total + Labor_Burden, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1328.62   -23.52    11.09    51.85   538.98
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -19.162399   1.622456  -11.81   <2e-16 ***
## Jobs_Total     0.808323   0.002662  303.63   <2e-16 ***
## Labor_Burden  -1.814713   0.023358  -77.69   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 105.6 on 10389 degrees of freedom
## Multiple R-squared:  0.9035, Adjusted R-squared:  0.9035
## F-statistic: 4.865e+04 on 2 and 10389 DF,  p-value: < 2.2e-16
```

```r
plot(lm3) # displays the linear regression model plots
```


Residuals vs Fitted
Residuals
500
0
−500
−1500
−500
0
500
1000
1500
2000
4433
8966
4169
Fitted values
lm(Jobs_Gross_Margin ~ Jobs_Total + Labor_Burden)


Normal Q−Q
Standardized residuals
5
0
−5
−10
−4
−2
0
2
4
4433
8966
4169
Theoretical Quantiles
lm(Jobs_Gross_Margin ~ Jobs_Total + Labor_Burden)





Result Comparison

Using a multiple linear regression model, we were able to improve the R^2 value from 0.7646 to 0.8828. This is because adding an additional variable to our algorithm helped tighten the distribution of our data (as seen in the Normal Q-Q plot). However, using this model did result in our residuals being spread less evenly throughout the plot due to an increase of outliers. Also, the Residuals vs. Leverage graph saw a slight decrease in accuracy.

Our third model utilized the same multiple linear regression model but with the removal of outliers beyond the 20% and 80% percentiles. This helped solve all of the issues presented in the second model and improved the R^2 value up to 0.9035.

Predict and Evaluate on the Test Data Using Metrics Correlation and MSE

```
# Results for LM1

# Jobs_Total
jt_pred1 <- predict(lm1, newdata = test)
jt_cor1 <- cor(jt_pred1, test$Jobs_Total)
jt_mse1 <- mean((jt_pred1 - test$Jobs_Total)^2)
jt_rmse1 <- sqrt(jt_mse1)

# Jobs_Gross_Margin
jgm_pred1 <- predict(lm1, newdata = test)
jgm_cor1 <- cor(jgm_pred1, test$Jobs_Gross_Margin)
jgm_mse1 <- mean((jgm_pred1 - test$Jobs_Gross_Margin)^2)
jgm_rmse1 <- sqrt(jgm_mse1)

# Labor_Pay
lp_pred1 <- predict(lm1, newdata = test)
lp_cor1 <- cor(lp_pred1, test$Labor_Pay)
lp_mse1 <- mean((lp_pred1 - test$Labor_Pay)^2)
lp_rmse1 <- sqrt(lp_mse1)

# Labor_Burden
lb_pred1 <- predict(lm1, newdata = test)
lb_cor1 <- cor(lb_pred1, test$Labor_Burden)
lb_mse1 <- mean((lb_pred1 - test$Labor_Burden)^2)
lb_rmse1 <- sqrt(lb_mse1)

# Results for LM2

# Jobs_Total
jt_pred2 <- predict(lm2, newdata = test)
jt_cor2 <- cor(jt_pred2, test$Jobs_Total)
jt_mse2 <- mean((jt_pred2 - test$Jobs_Total)^2)
jt_rmse2 <- sqrt(jt_mse2)

# Jobs_Gross_Margin
jgm_pred2 <- predict(lm2, newdata = test)
jgm_cor2 <- cor(jgm_pred2, test$Jobs_Gross_Margin)
jgm_mse2 <- mean((jgm_pred2 - test$Jobs_Gross_Margin)^2)
jgm_rmse2 <- sqrt(jgm_mse2)

# Labor_Pay
lp_pred2 <- predict(lm2, newdata = test)
lp_cor2 <- cor(lp_pred2, test$Labor_Pay)
lp_mse2 <- mean((lp_pred2 - test$Labor_Pay)^2)
lp_rmse2 <- sqrt(lp_mse2)
```

```
# Labor_Burden
lb_pred2 <- predict(lm2, newdata = test)
lb_cor2 <- cor(lb_pred2, test$Labor_Burden)
lb_mse2 <- mean((lb_pred2 - test$Labor_Burden)^2)
lb_rmse2 <- sqrt(lb_mse2)

# Results for LM3

# Jobs_Total
jt_pred3 <- predict(lm3, newdata = test)
jt_cor3 <- cor(jt_pred3, test$Jobs_Total)
jt_mse3 <- mean((jt_pred3 - test$Jobs_Total)^2)
jt_rmse3 <- sqrt(jt_mse3)

# Jobs_Gross_Margin
jgm_pred3 <- predict(lm3, newdata = test)
jgm_cor3 <- cor(jgm_pred3, test$Jobs_Gross_Margin)
jgm_mse3 <- mean((jgm_pred3 - test$Jobs_Gross_Margin)^2)
jgm_rmse3 <- sqrt(jgm_mse3)

# Labor_Pay
lp_pred3 <- predict(lm3, newdata = test)
lp_cor3 <- cor(lp_pred3, test$Labor_Pay)
lp_mse3 <- mean((lp_pred3 - test$Labor_Pay)^2)
lp_rmse3 <- sqrt(lp_mse3)

# Labor_Burden
lb_pred3 <- predict(lm3, newdata = test)
lb_cor3 <- cor(lb_pred3, test$Labor_Burden)
lb_mse3 <- mean((lb_pred3 - test$Labor_Burden)^2)
lb_rmse3 <- sqrt(lb_mse3)

# Output all results
cat("-----LM1-----\n") #lm1 results
```

```
## -----LM1-----
```

```
cat(cat("LM1 Jobs_Total\n"), cat("\t", paste('Correlation:', jt_cor1), "\n"), cat("\t", paste('MSE:', j
```

```
## LM1 Jobs_Total
##   Correlation: 0.999999999999988
##   MSE: 73348.07643763
##   rMSE: 270.828500046856
```

```
cat(cat("LM1 Jobs_Gross_Margin\n"), cat("\t", paste('Correlation:', jgm_cor1), "\n"), cat("\t", paste('
```

```
## LM1 Jobs_Gross_Margin
##   Correlation: 0.921300435123491
##   MSE: 19514.1601812032
##   rMSE: 139.693092818519
```

```
cat(cat("LM1 Labor_Pay\n"), cat("\t", paste('Correlation:', lp_cor1), "\n"), cat("\t", paste('MSE:', lp
```

```
## LM1 Labor_Pay
##   Correlation: 0.526665574510713
##   MSE: 63584.5440707287
```

```
##   rMSE: 252.159759023379
```

```
cat(cat("LM1 Labor_Burden\n"), cat("\t", paste('Correlation:', lb_cor1), "\n"), cat("\t", paste('MSE:',
```

```
## LM1 Labor_Burden
##   Correlation: 0.429108074591634
##   MSE: 72206.5549855196
##   rMSE: 268.712774139079
```

```
cat("\n-----LM2-----\n") # lm2 results
```

```
##
## -----LM2-----
```

```
cat(cat("LM2 Jobs_Total\n"), cat("\t", paste('Correlation:', jt_cor2), "\n"), cat("\t", paste('MSE:', j
```

```
## LM2 Jobs_Total
##   Correlation: 0.951365847456146
##   MSE: 70122.1650576283
##   rMSE: 264.805900722828
```

```
cat(cat("LM2 Jobs_Gross_Margin\n"), cat("\t", paste('Correlation:', jgm_cor2), "\n"), cat("\t", paste('
```

```
## LM2 Jobs_Gross_Margin
##   Correlation: 0.949143169446278
##   MSE: 12261.8594569348
##   rMSE: 110.73328071061
```

```
cat(cat("LM2 Labor_Pay\n"), cat("\t", paste('Correlation:', lp_cor2), "\n"), cat("\t", paste('MSE:', lp
```

```
## LM2 Labor_Pay
##   Correlation: 0.383087334812998
##   MSE: 83601.3372783318
##   rMSE: 289.138958423682
```

```
cat(cat("LM2 Labor_Burden\n"), cat("\t", paste('Correlation:', lb_cor2), "\n"), cat("\t", paste('MSE:',
```

```
## LM2 Labor_Burden
##   Correlation: 0.129979579362159
##   MSE: 97710.3093770102
##   rMSE: 312.586483036311
```

```
cat("\n-----LM3-----\n") # lm3 results
```

```
##
## -----LM3-----
```

```
cat(cat("LM3 Jobs_Total\n"), cat("\t", paste('Correlation:', jt_cor3), "\n"), cat("\t", paste('MSE:', j
```

```
## LM3 Jobs_Total
##   Correlation: 0.967469445957732
##   MSE: 59703.5633830609
##   rMSE: 244.343126326608
```

```
cat(cat("LM3 Jobs_Gross_Margin\n"), cat("\t", paste('Correlation:', jgm_cor3), "\n"), cat("\t", paste('
```

```
## LM3 Jobs_Gross_Margin
##   Correlation: 0.950991187325645
##   MSE: 11070.003612071
##   rMSE: 105.214084665842
```

```
cat(cat("LM3 Labor_Pay\n"), cat("\t", paste('Correlation:', lp_cor3), "\n"), cat("\t", paste('MSE:', lp
```

```
## LM3 Labor_Pay
##   Correlation: 0.412658163208319
##   MSE: 100353.1271735
##   rMSE: 316.78561705592
```

```
cat(cat("LM3 Labor_Burden\n"), cat("\t", paste('Correlation:', lb_cor3), "\n"), cat("\t", paste('MSE:',
```

```
## LM3 Labor_Burden
##   Correlation: 0.186636844006684
##   MSE: 114638.557602642
##   rMSE: 338.58316201879
```

The predictions for Jobs_Total had a high correlation to the actual testing data across all 3 models. We predict this is because of the strong correlation in Labor_Pay to Jobs_Total columns seen in the data set. Jobs_Gross_Margin also had high correlation values due to its correlation to Labor_Pay (and this Jobs_Total).

Minimal success was found in the predictions for Labor_Pay and Labor_Burden in any of our model cases. This goes to show that correlation of x to y does NOT imply correlation of y to x when it comes to linear regression models.