Regression

Code

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

Hide

```
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

Hide

```
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
```

Data Exploration

Create a summary of the data

Hide

```
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
```

Display the number of rows in the data set

Hide

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

```
[1] 14479
```

Display the number of missing values in each variable

Hide

```
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
```

Output the outliers in the data set

Hide

```
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 J
obs_Total into df_outliers

# 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
outliers of Jobs_Gross_Margin into df_outliers

# 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 Lab
or_Pay into df_outliers

# 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 Labor_Burden into df_outliers

# 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 ...
```

Hide

```
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  
```

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

Hide

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

```

	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 
```

Hide

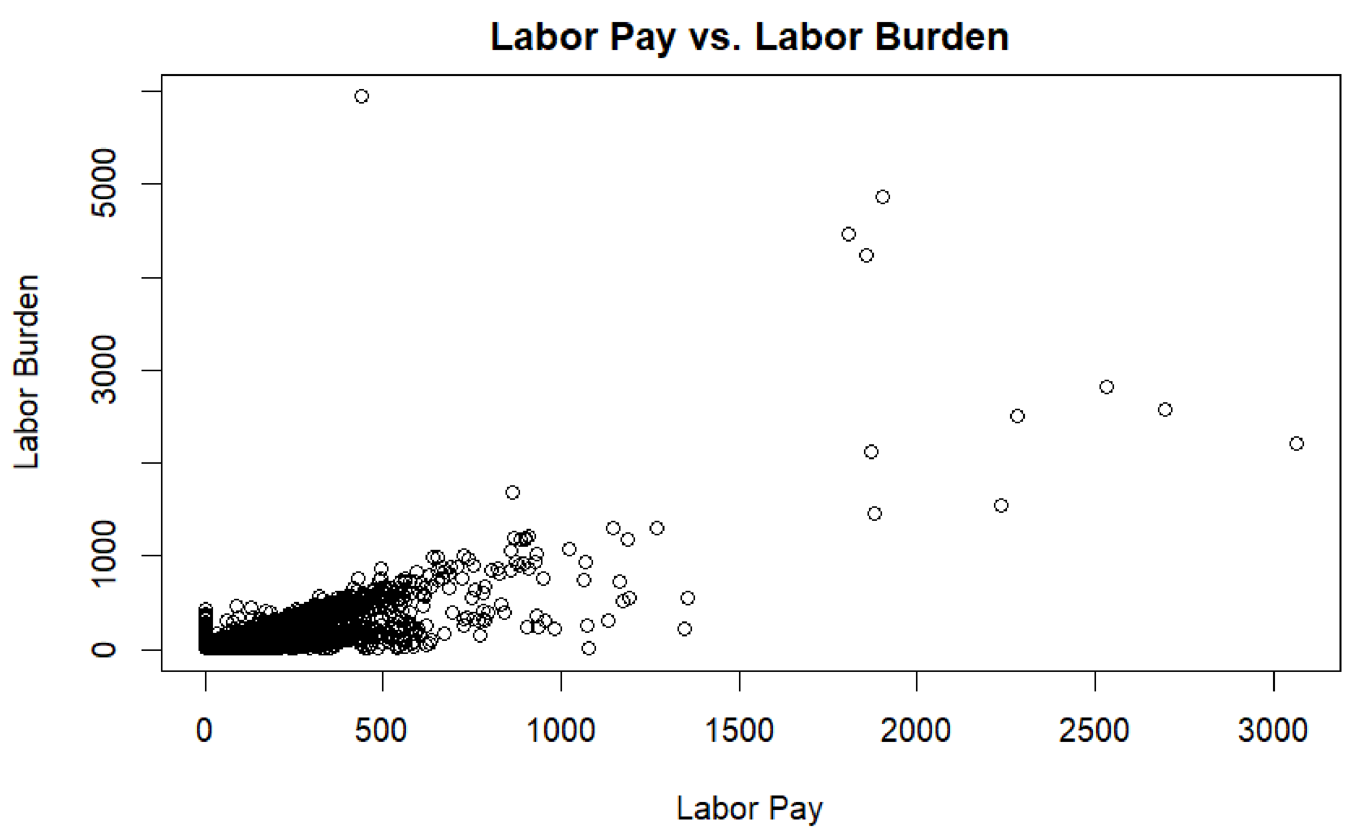
```
cor.test(df$Jobs_Gross_Margin, df$Jobs_Total, use = "complete") # finds correlations between Jobs_Gross_Margin and Jobs_Total
```

```
	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
```

Informative Graphs

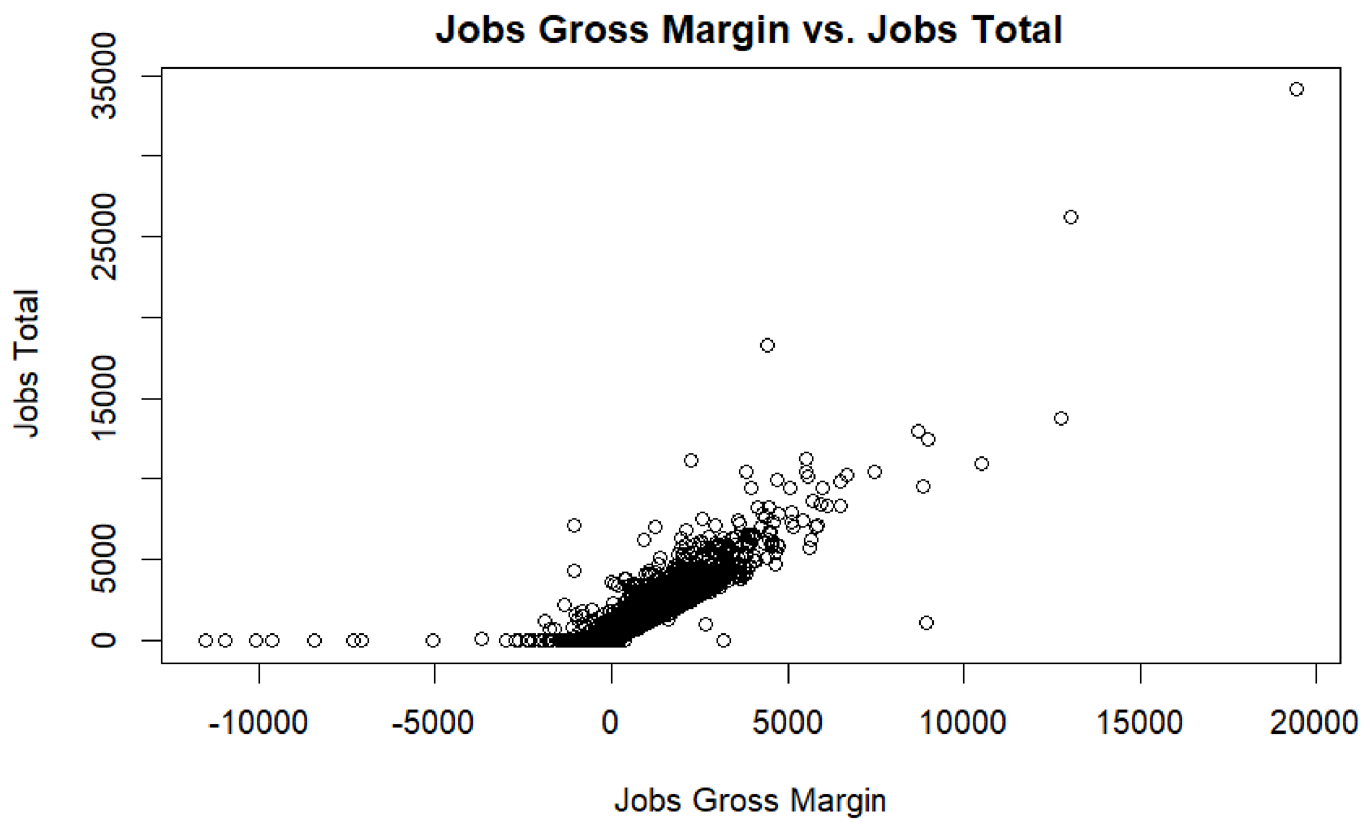
Hide

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



Hide

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



Build a Simple Linear Regression Model

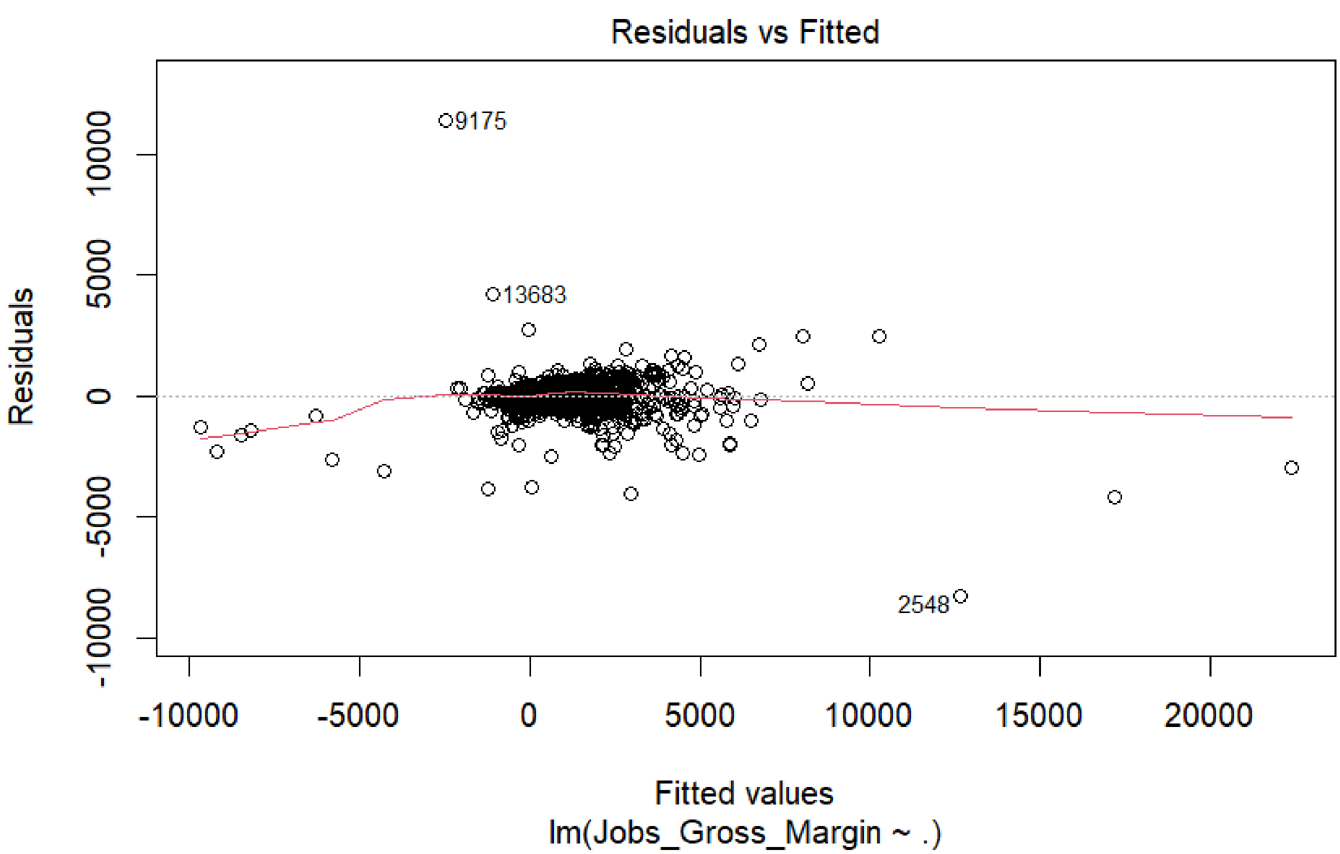
Hide

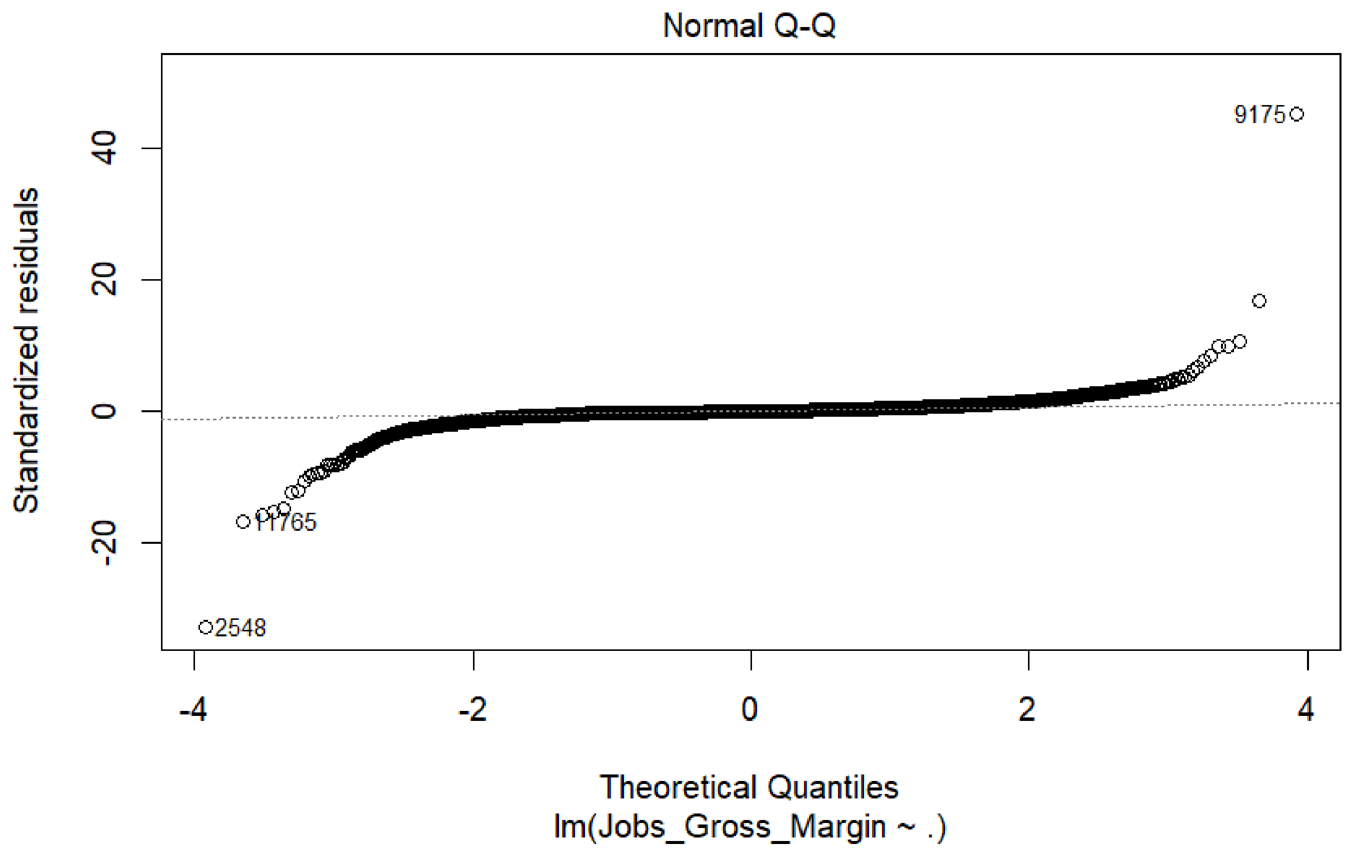
```
lm1 <- lm(Jobs_Gross_Margin ~ ., data = train) # builds linear regression model with all predict
ors on Jobs_Gross_Margin
summary(lm1) # shows the linear regression model summary
```

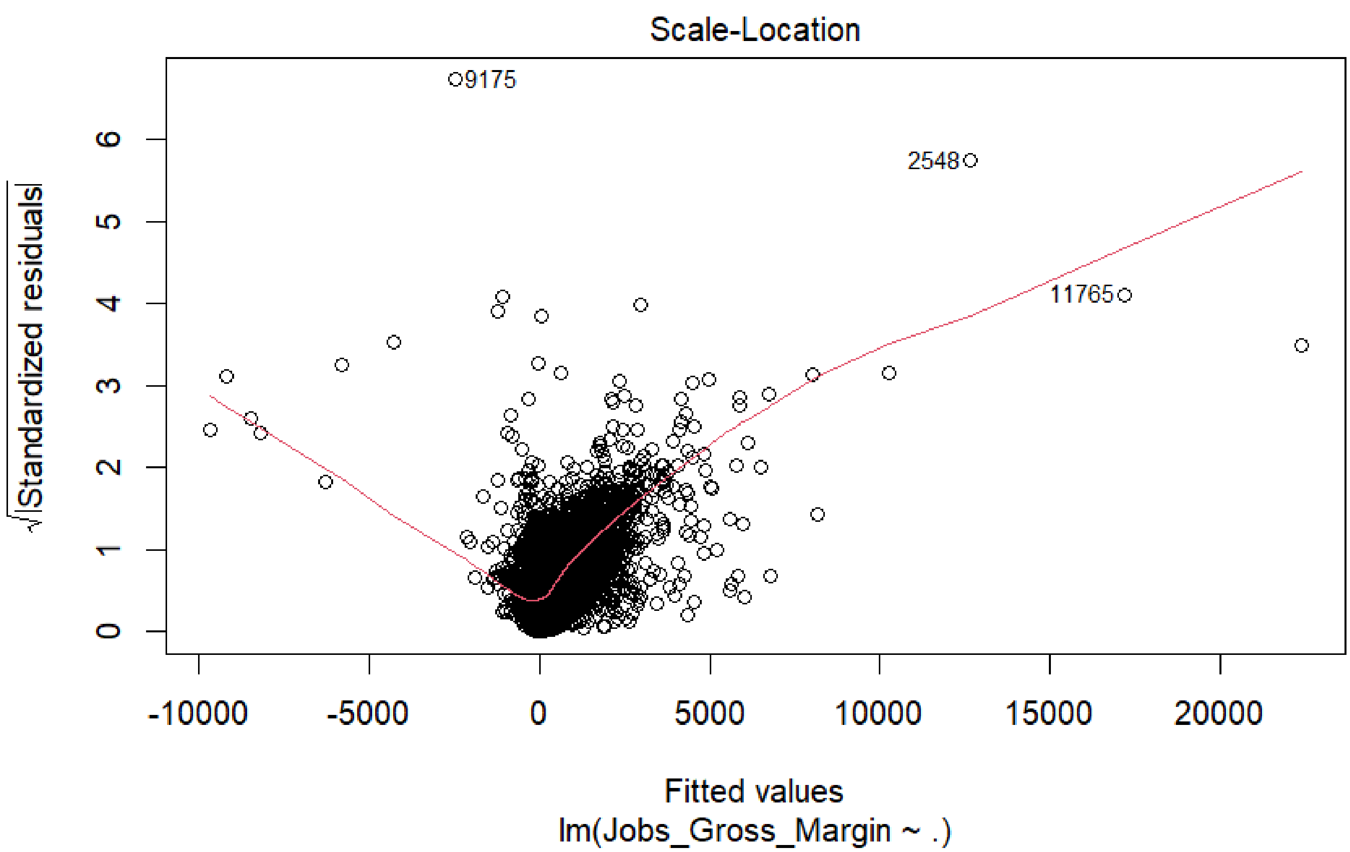
```
Call:
lm(formula = Jobs_Gross_Margin ~ ., data = train)

Residuals:
    Min      1Q  Median      3Q     Max
-8255.9   -50.5    -9.3    56.1 11383.4

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  57.291767   3.195990   17.93   <2e-16 ***
Labor_Pay    -0.838964   0.033567  -24.99   <2e-16 ***
Labor_Burden -1.575908   0.024559  -64.17   <2e-16 ***
Jobs_Total    0.768291   0.002754  278.96   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 253.5 on 11579 degrees of freedom
Multiple R-squared:  0.8888,	Adjusted R-squared:  0.8888
F-statistic: 3.085e+04 on 3 and 11579 DF,  p-value: < 2.2e-16
```

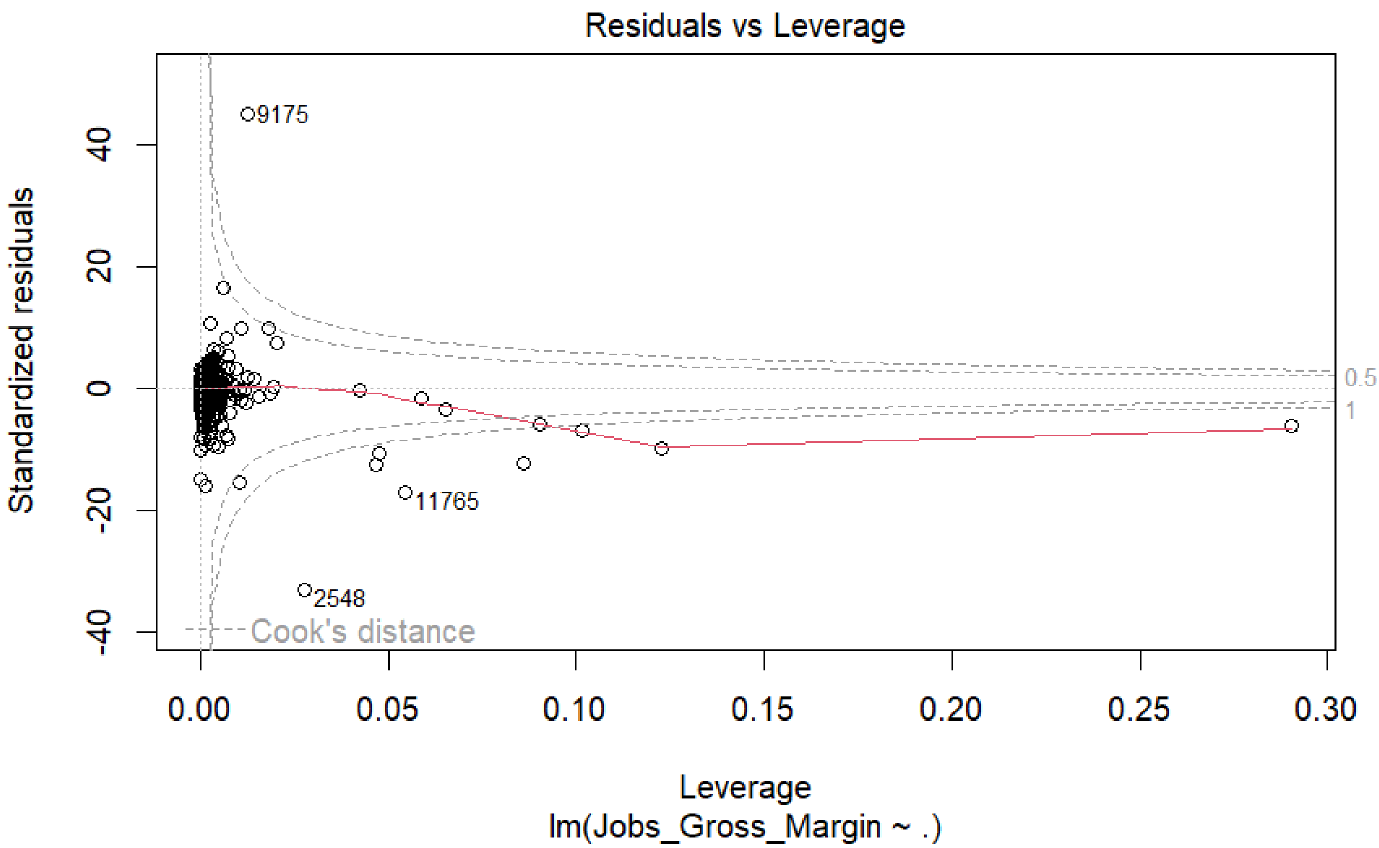
Hide

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




Normal Q-Q
Standardized residuals
Theoretical Quantiles
lm(Jobs_Gross_Margin ~ .)
9175
11765
2548
40
20
0
-20
-4
-2
0
2
4


Scale-Location
√|Standardized residuals|
Fitted values
lm(Jobs_Gross_Margin ~ .)
9175
2548
11765
6
5
4
3
2
1
0
-10000
-5000
0
5000
10000
15000
20000



Hide

```
pred <- predict(lm1, newdata=test)
cor_lm <- cor(pred, test$Jobs_Gross_Margin)
mse_lm <- mean((pred - test$Jobs_Gross_Margin)^2)
print(paste("cor=", cor_lm))
```

```
[1] "cor= 0.959206463271685"
```

Hide

```
print(paste("mse=", mse_lm))
```

```
[1] "mse= 36101.9618087527"
```

Hide

```
print(paste("rmse=",  sqrt(mse_lm)))
```

```
[1] "rmse= 190.005162584475"
```