Linear Regression

Aditi Chaudhari

2022-09-25

Linear Regression

Linear regression is a type of machine learning model which seeks to find a relationship between x (the predictor variables) and y (the target variable) in a data set. The model generally takes the form y = wx + b, with w being the slope in which a change in x corresponds to a change in y and b being the intercept. Linear regression is an important machine learning model because it is relatively simple due to the fact that the coefficients can quantify the effect that the predictor variables have on the target variable, it works well if the data follows a linear pattern, and it has low variance. However, a weakness of linear regression is that it has high bias due to the fact that it assumes that the data is in a linear shape. It is important to consider the strengths and weaknesses of the linear regression model prior to using it.

Data Exploration

Let's delve into exploring the linear regression model.

Here, the data stored in the flights.csv file is read into a data frame. The data set is from https://www.kaggle.com/datasets/lcsldatasets/flights

```
df <- read.csv("flights.csv")
```

Next, the data randomly divided into a training set containing 80% of the original data and a test set containing 20% of the original data.

```
i <- sample(1:nrow(df), nrow(df) * 0.80, replace = FALSE)
train <- df[i,]
test <- df[-i,]
```

After dividing the data into a training set and a test set, we can explore the data using the training set.

Firstly, let's simply see what our data looks like using the head() function, which selects the first n rows of a data frame.

```
head(train, n=10)
```

```
##       Month DayOfMonth DayOfWeek PlaneAge Distance AirTime DepTime ArrTime
## 9835      5         28         3        4     1624     184    1812      32
## 9206      2         16         6        4     1136     146    1957    2354
## 2384      2          5         2        8      386      63    2026    2155
## 665       4         15         2       10      367      59    1414    1527
## 2655      3         18         2        8      371      64    1321    1442
## 11230     1          8         2        9     1728     208    1705    2245
## 1067      6         18         3       10     1166     160    2110    2308
## 9352      3         11         2        4     1670     217     907    1456
## 534       3         27         4       10      361      58    2209    2323
## 1848     11         12         3       10      895     126    1054    1217
##       ArrDelay
```

```
## 9835       -23
## 9206        24
## 2384        15
## 665         -3
## 2655        -3
## 11230       45
## 1067        13
## 9352        21
## 534         13
## 1848       -18
```

Using the summary() function in R provides us with summary statistics for each column.

```
summary(train)
```

```
##      Month          DayOfMonth      DayOfWeek        PlaneAge
##  Min.   : 1.000   Min.   : 1.00   Min.   :1.000   Min.   : 4.00
##  1st Qu.: 3.000   1st Qu.: 8.00   1st Qu.:2.000   1st Qu.: 8.00
##  Median : 6.000   Median :16.00   Median :4.000   Median : 9.00
##  Mean   : 6.426   Mean   :15.75   Mean   :3.884   Mean   :10.18
##  3rd Qu.: 9.000   3rd Qu.:23.00   3rd Qu.:6.000   3rd Qu.:11.00
##  Max.   :12.000   Max.   :31.00   Max.   :7.000   Max.   :23.00
##     Distance         AirTime         DepTime        ArrTime
##  Min.   : 133.0   Min.   : 24.0   Min.   :   1   Min.   :   1
##  1st Qu.: 325.0   1st Qu.: 53.0   1st Qu.: 938   1st Qu.:1106
##  Median : 461.0   Median : 71.0   Median :1339   Median :1513
##  Mean   : 645.3   Mean   : 92.9   Mean   :1354   Mean   :1488
##  3rd Qu.: 862.0   3rd Qu.:121.0   3rd Qu.:1750   3rd Qu.:1918
##  Max.   :2363.0   Max.   :396.0   Max.   :2400   Max.   :2400
##     ArrDelay
##  Min.   :-54.00
##  1st Qu.: -9.00
##  Median : -3.00
##  Mean   :  5.59
##  3rd Qu.:  8.00
##  Max.   :643.00
```

We also want to see how big this data frame is.

Using the nrow() function reveals that there are 19,999 observations.

```
nrow(train)
```

```
## [1] 15999
```

The ncol() function reveals that there are 9 variables.

```
ncol(train)
```

```
## [1] 9
```

Using the str() function, we can see the structure of the data frame. This function reveals that all of the variables are of type integer, which is ideal for linear regression.

```
str(train)
```

```
## 'data.frame':    15999 obs. of  9 variables:
##  $ Month     : int  5 2 2 4 3 1 6 3 3 11 ...
##  $ DayOfMonth: int  28 16 5 15 18 8 18 11 27 12 ...
##  $ DayOfWeek : int  3 6 2 2 2 2 3 2 4 3 ...
```

```
##  $ PlaneAge  : int  4 4 8 10 8 9 10 4 10 10 ...
##  $ Distance  : int  1624 1136 386 367 371 1728 1166 1670 361 895 ...
##  $ AirTime   : int  184 146 63 59 64 208 160 217 58 126 ...
##  $ DepTime   : int  1812 1957 2026 1414 1321 1705 2110 907 2209 1054 ...
##  $ ArrTime   : int  32 2354 2155 1527 1442 2245 2308 1456 2323 1217 ...
##  $ ArrDelay  : int  -23 24 15 -3 -3 45 13 21 13 -18 ...
```

Using the colSums() function, we can see that there are no missing values in any of the columns. It is important to remove missing values prior to performing linear regression.

```
colSums(is.na(train))
```

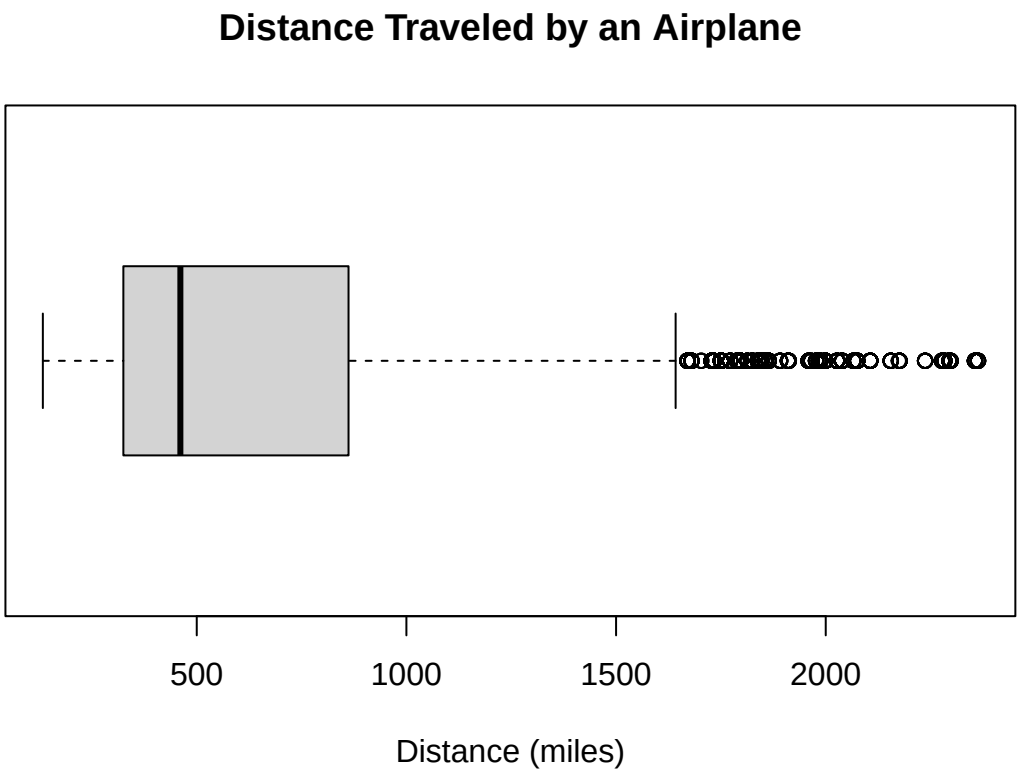
```
##      Month DayOfMonth  DayOfWeek   PlaneAge   Distance    AirTime    DepTime
##          0          0          0          0          0          0          0
##    ArrTime   ArrDelay
##          0          0
```

Data Visualization

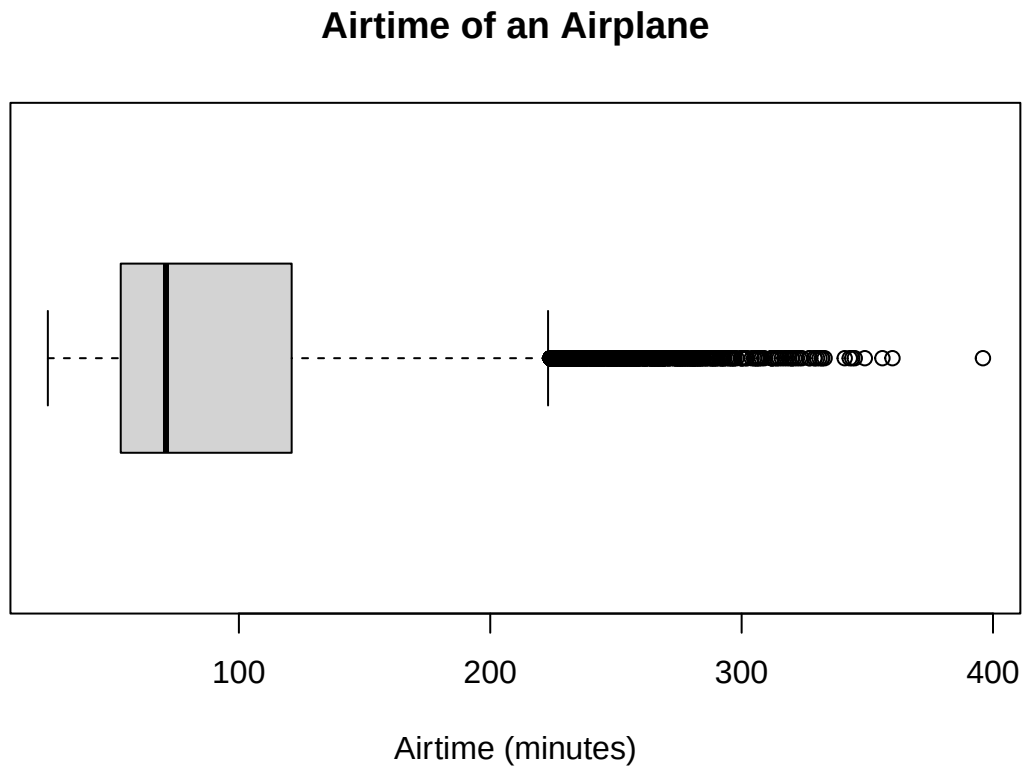
After exploring the data, it is time to visualize it.

First, we can use box-and whisker plots to get a better understanding of certain variables. In this case, we are interested in the correlation between Distance and Airtime, so let's create a box-and-whisker plot for each.

```
boxplot(train$Distance, data=train, horizontal=TRUE,
        main="Distance Traveled by an Airplane",xlab="Distance (miles)")
```



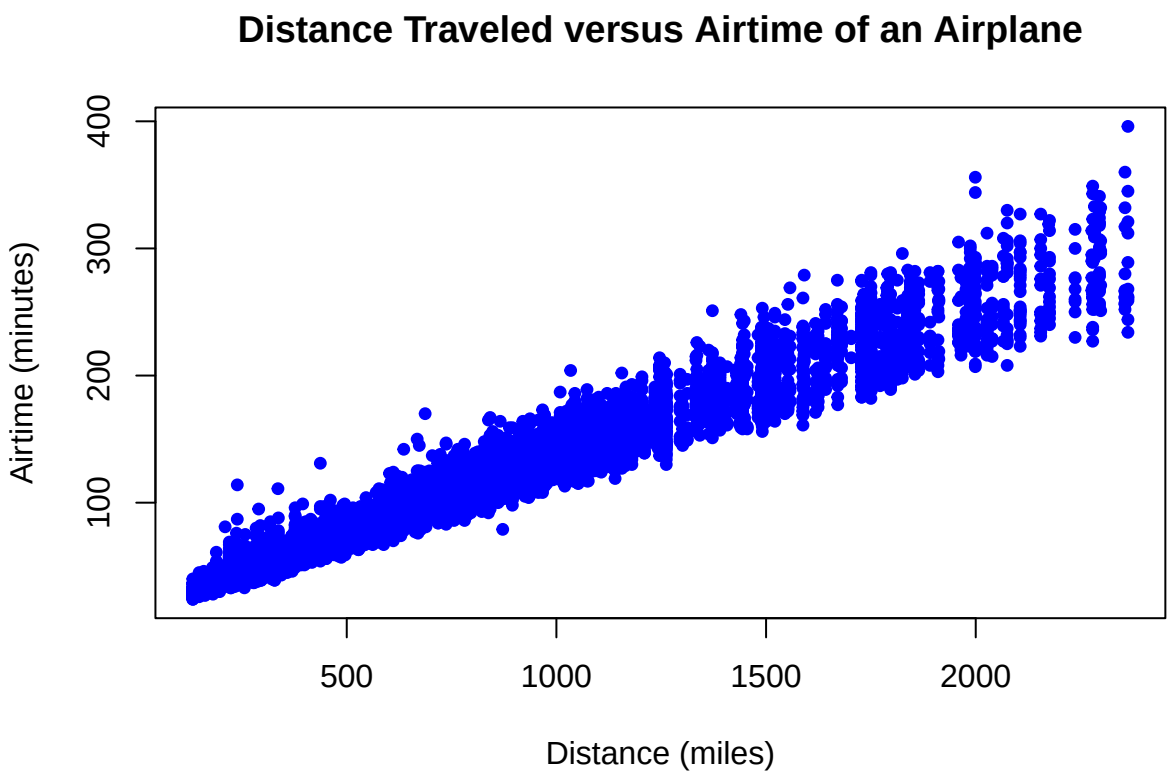
```
boxplot(train$AirTime, data=train, horizontal=TRUE,
        main="Airtime of an Airplane",xlab="Airtime (minutes)")
```



We can gather information on the variation in each variable using these two box-and-whisker plots. It is important to note that the data for each variable has outliers, which will affect the accuracy of the linear regression model that we create.

Using a scatter plot to plot the distance a plane travels against how much time it spends in the air shows that there is a linear correlation between the two variables.

```
plot(train$Distance, train$AirTime, pch = 19, cex=0.75, col="blue",
     main="Distance Traveled versus Airtime of an Airplane",
     xlab="Distance (miles)", ylab="Airtime (minutes)")
```



Simple Linear Regression

Now, let's create a simple linear regression model to show how the distance a plane travels (in miles) affects the airtime of the plane (in minutes).

```
lm1 <- lm(AirTime~Distance, data=train)
summary(lm1)
```

```
##
## Call:
## lm(formula = AirTime ~ Distance, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -65.792  -5.022  -0.911   4.322 100.049
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.518e+01  1.489e-01   102.0   <2e-16 ***
## Distance    1.204e-01  1.892e-04   636.5   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.77 on 15997 degrees of freedom
## Multiple R-squared:  0.962,  Adjusted R-squared:  0.962
## F-statistic: 4.051e+05 on 1 and 15997 DF,  p-value: < 2.2e-16
```

The summary of the linear regression model reveals that the airtime of a plane can be predicted with the following formula:

airtime (in minutes) = (0.1201 * distance (in miles)) + 15.3041

So for instance, if we wanted to calculate the airtime between two airports that are 879 miles away, we can use the formula to get an airtime of about 121 minutes.

airtime (in minutes) = (0.1201 * 879 miles) + 15.3041 = 120.872

The Residual Standard Error (RSE) is a measure of how off the model is from the data. In this case, the RSE is around 10.81 minutes.

The R-squared value is a measure of the variance in the model explained by the predictor. The closer it is to 1, the more that the variation in airtime can be predicted by the distance the airplane travels. The R-squared value in our model is around 0.9617, which is quite high and ideal.

The F-statistic considers the predictor variable to determine whether it is a significant predictor of the outcome variable. Having a F-statistic greater than 1 and a low p-value indicates that we have confidence in this model.

Next, let's plot the residuals from the linear regression model.

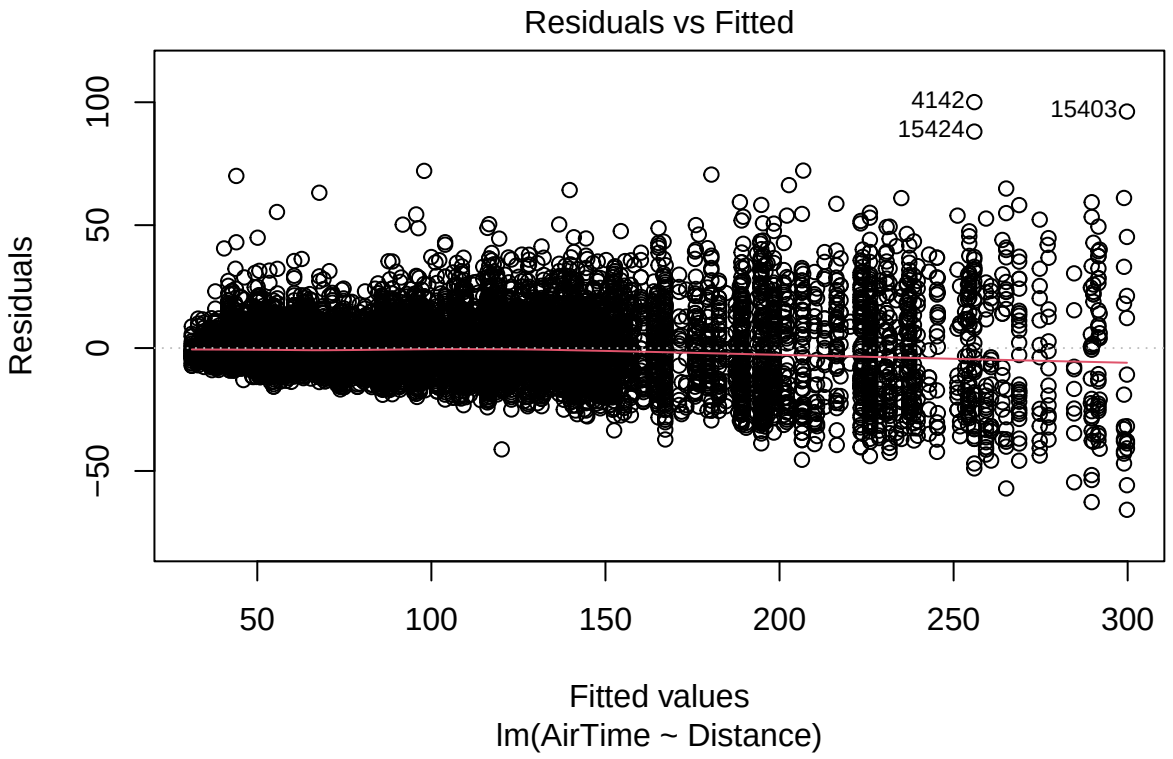
Plot 1 (which plots Residuals vs Fitted) shows that there are equally spread residuals around a horizontal line without any distinct patterns. This is a good indication that there is not a non-linear relationship between the predictor variable and the outcome variable.

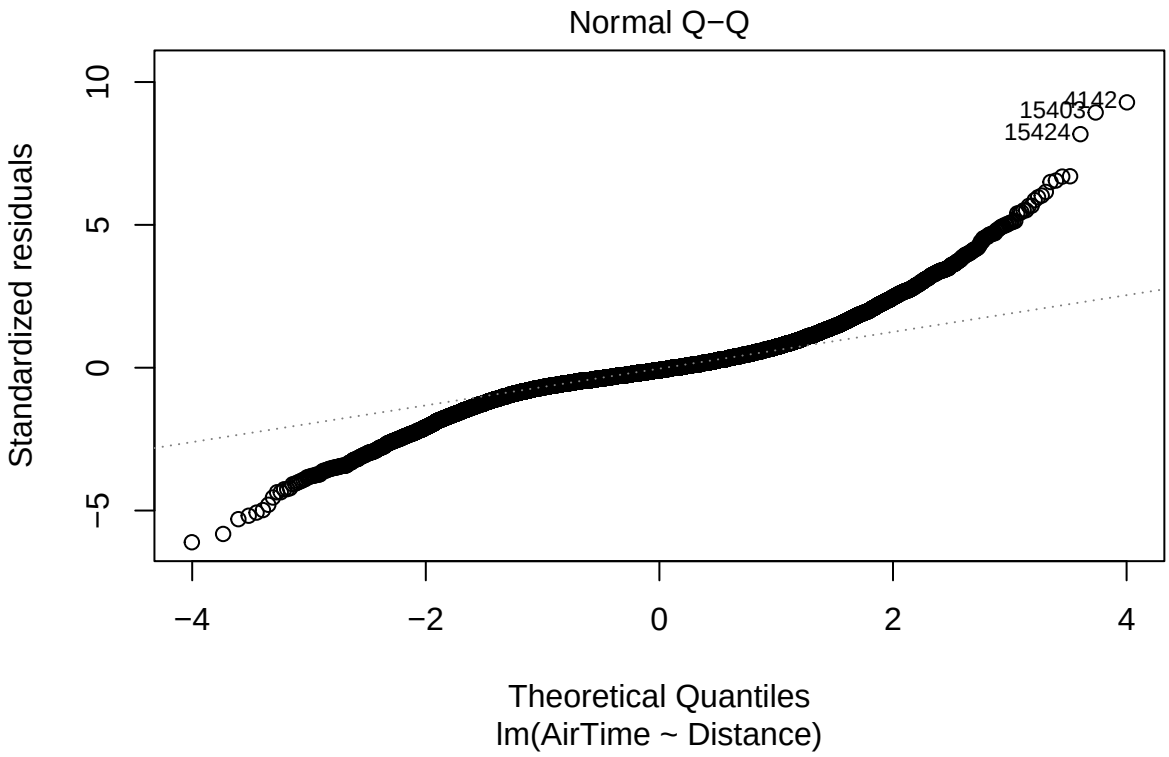
Plot 2 (which plots Normal Q-Q) shows that the residuals are normally-distributed since there is a fairly diagonal line following the dashed line.

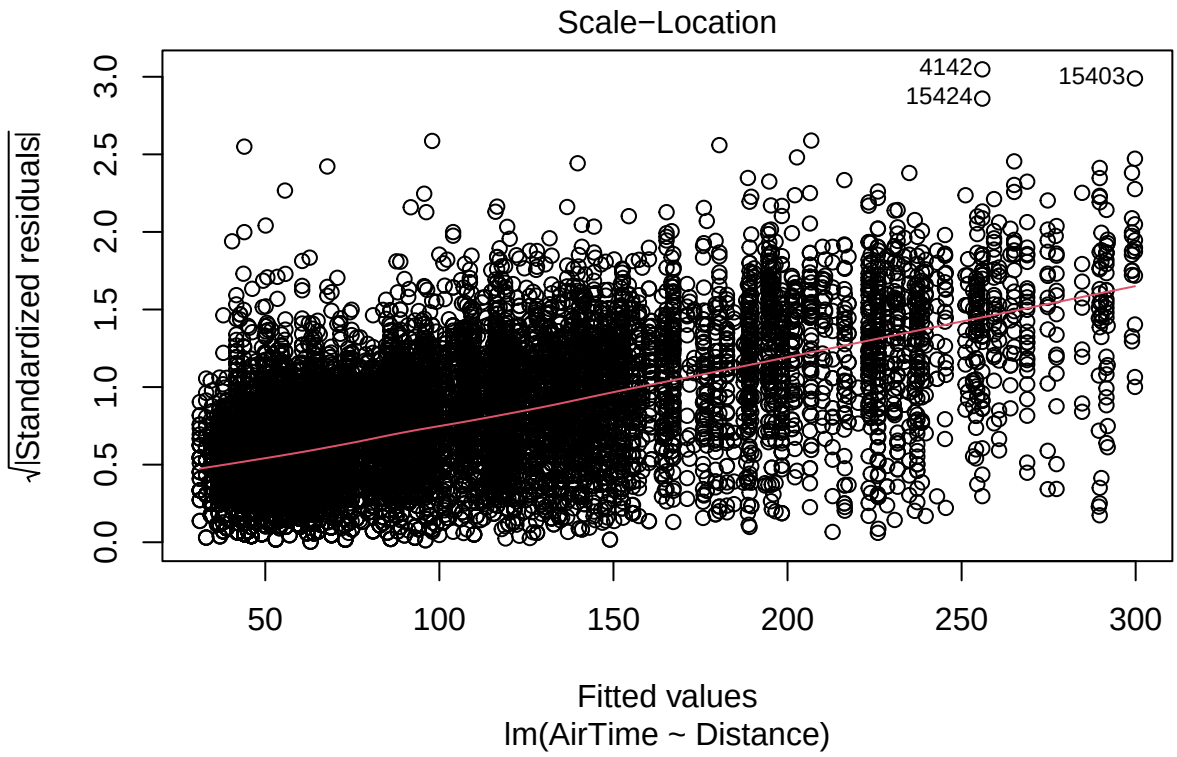
Plot 3 (which plots Scale-Location) shows a fairly horizontal line with with data points equally distributed around the line. This means the data is homoscedastic.

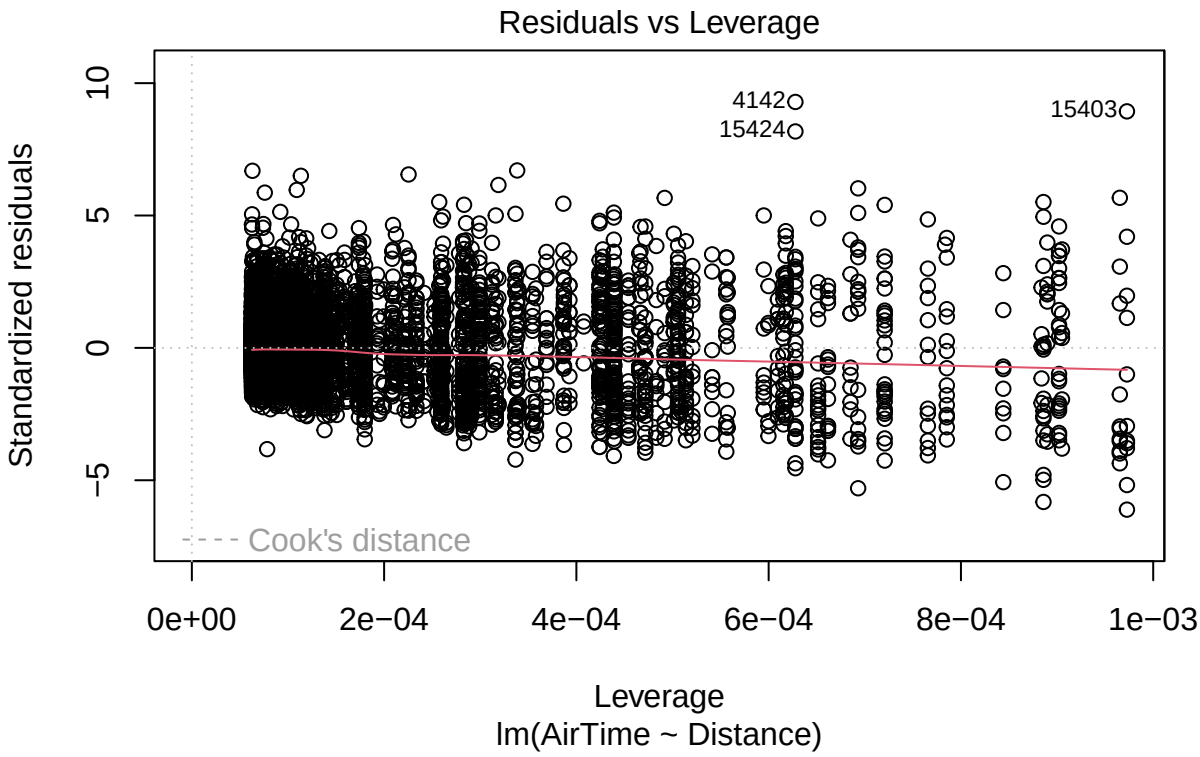
Plot 4 (which plots Residuals vs Leverage) shows which leverage points influence the regression line. It shows outliers (which is a data point with an unusual Y value) and leverage points (which is a data point with an unusual X value).

```
plot(lm1)
```




Normal Q−Q
Standardized residuals
10
5
0
−5
4142
15403
15424
−4
−2
0
2
4
Theoretical Quantiles
lm(AirTime ~ Distance)


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
2.5
3.0
4142
15424
15403
50
100
150
200
250
300
Fitted values
lm(AirTime ~ Distance)


Residuals vs Leverage
Standardized residuals
10
5
0
−5
4142
15424
15403
Cook's distance
0e+00
2e−04
4e−04
6e−04
8e−04
1e−03
Leverage
lm(AirTime ~ Distance)

Multiple Linear Regression

Multiple Linear Regression involves multiple predictor values to predict an outcome value. In this scenario, we want to see how the distance a plane travels (in miles) and how old the plane is (in years) affect the total airtime of the plane (in minutes).

```
lm2 <- lm(AirTime~Distance + PlaneAge, data = train)
summary(lm2)
```

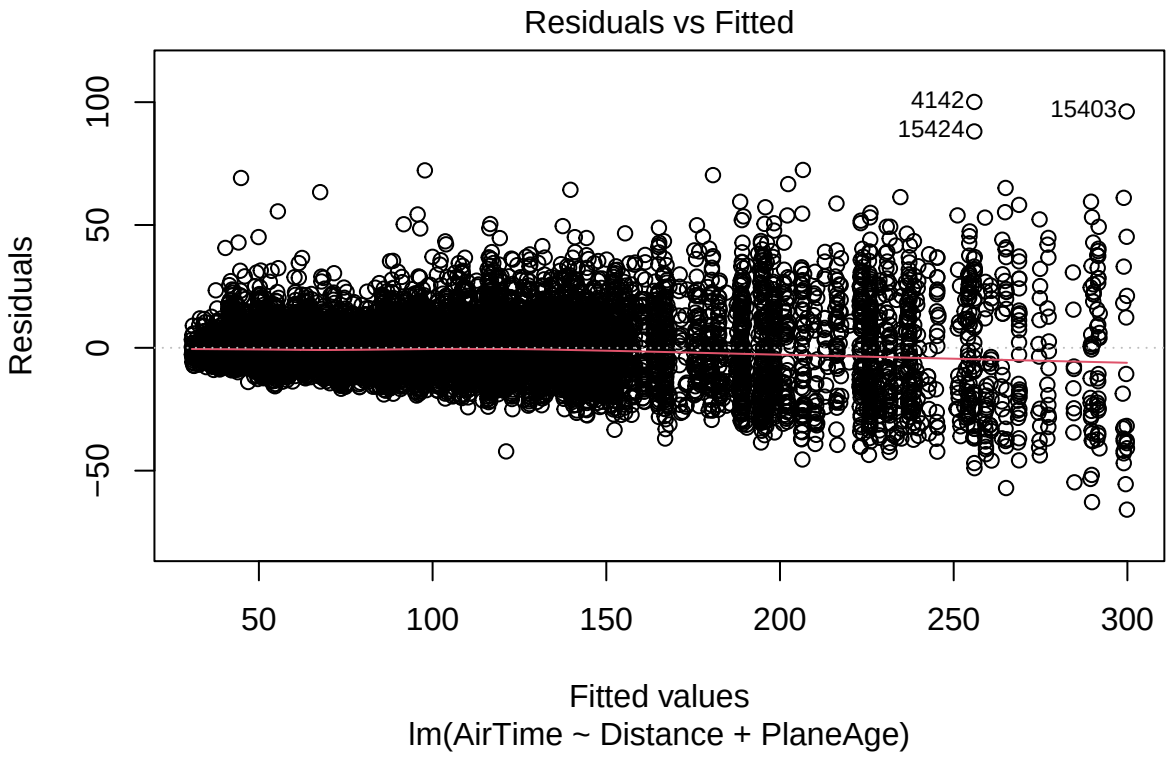
```
##
## Call:
## lm(formula = AirTime ~ Distance + PlaneAge, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -65.871  -5.098  -0.915   4.339 100.079
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.438e+01  2.413e-01  59.588  < 2e-16 ***
## Distance    1.205e-01  1.905e-04 632.855  < 2e-16 ***
## PlaneAge    7.317e-02  1.723e-02   4.248 2.17e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.77 on 15996 degrees of freedom
## Multiple R-squared:  0.9621, Adjusted R-squared:  0.9621
## F-statistic: 2.028e+05 on 2 and 15996 DF,  p-value: < 2.2e-16
```

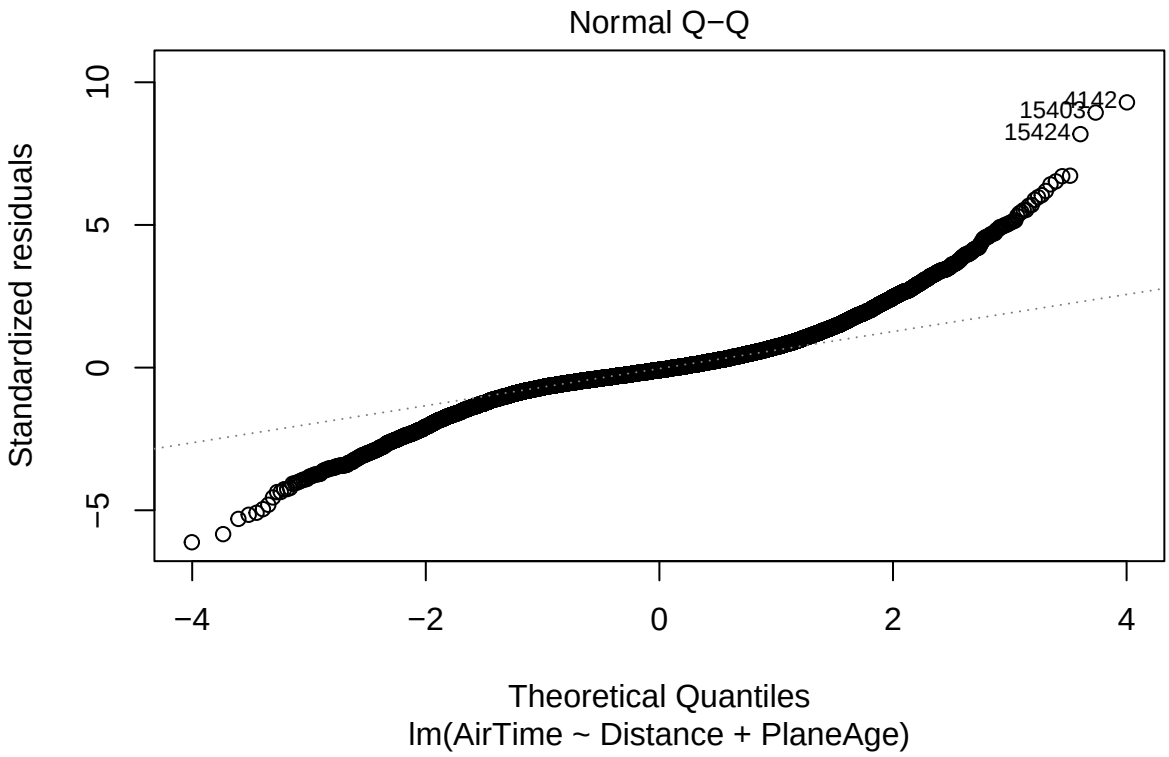
The multiple linear regression model reveals that the equation to predict the airtime of a plane based off the distance the plane travels and the age of the plane is:

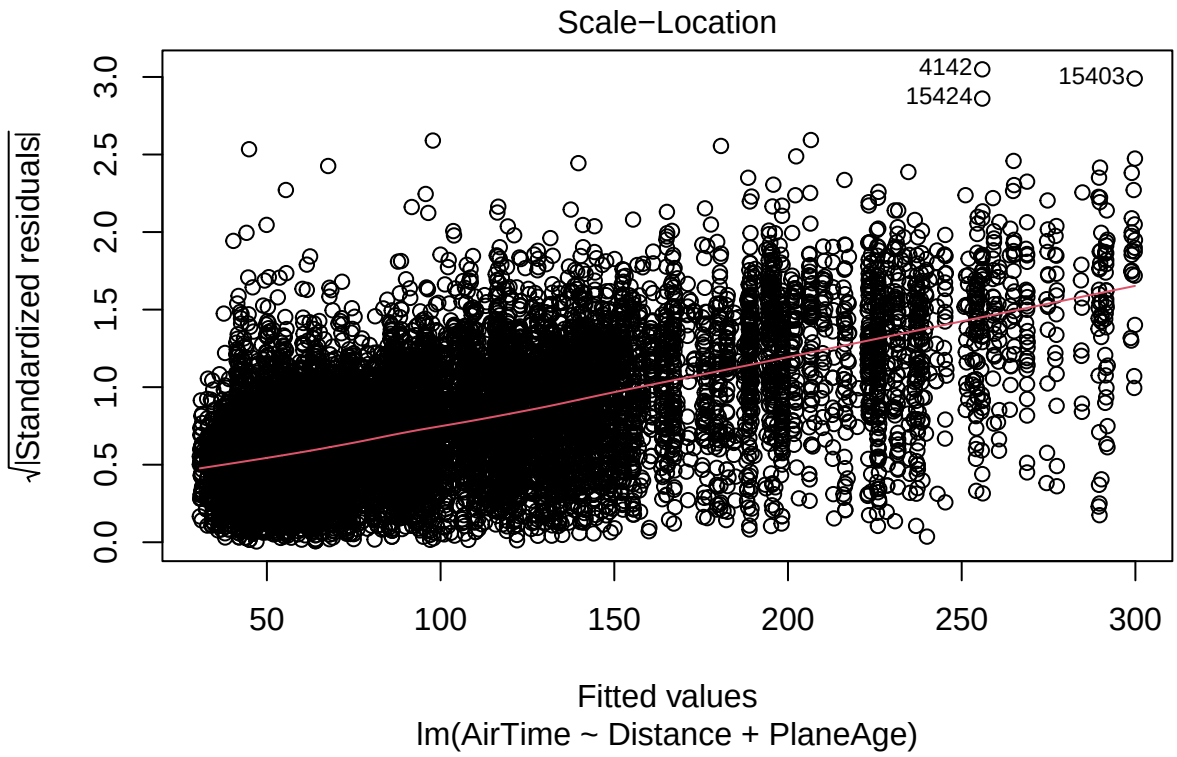
airtime (in minutes) = (0.12023 * distance (in miles)) + (0.07494 * plane_age (in years)) + 14.47610

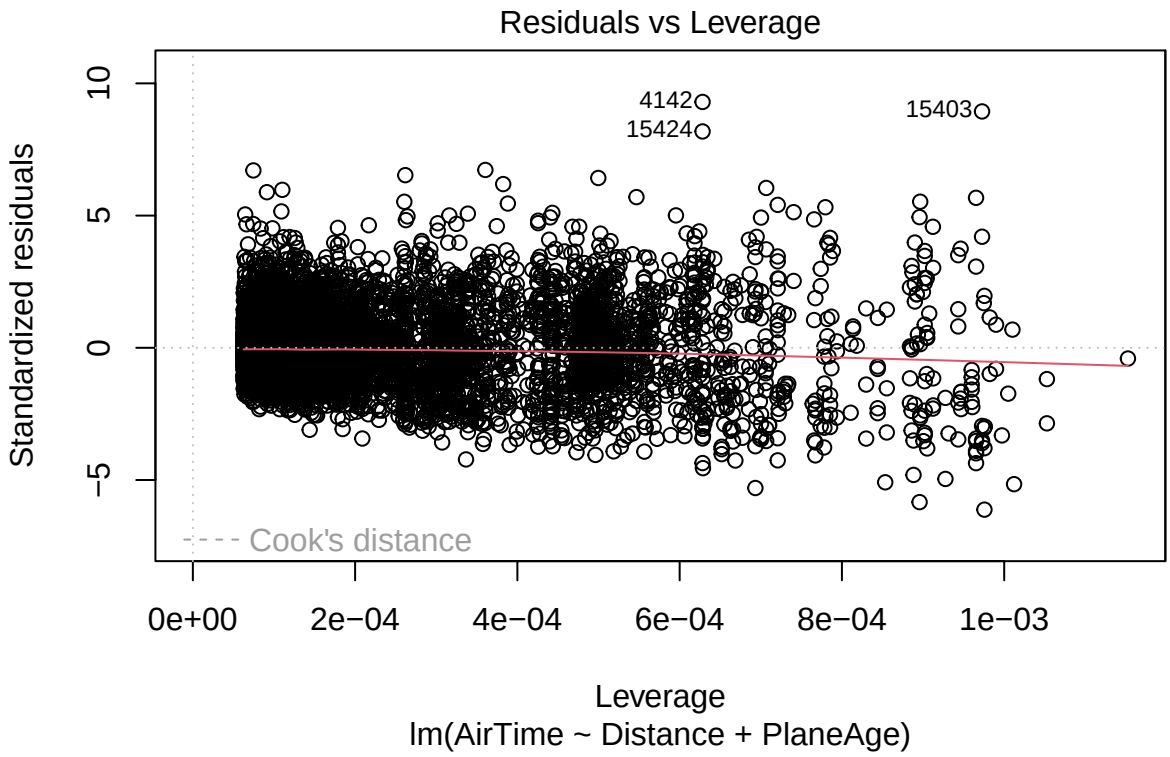
Plotting the residuals show that this linear regression model fits the data well.

```
plot(lm2)
```


Residuals vs Fitted
Residuals
100
50
0
−50
4142
15424
15403
50
100
150
200
250
300
Fitted values
lm(AirTime ~ Distance + PlaneAge)


Normal Q−Q
Standardized residuals
10
5
0
−5
4142
15403
15424
−4
−2
0
2
4
Theoretical Quantiles
lm(AirTime ~ Distance + PlaneAge)


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
2.5
3.0
4142
15424
15403
50
100
150
200
250
300
Fitted values
lm(AirTime ~ Distance + PlaneAge)



Let us also see how the airtime of an airplane (in minutes) can be affected by the distance an airplane travels (in miles) along with the age of the plane (in years) along with the arrival delay of the plane (in minutes).

```
lm3 <- lm(AirTime~Distance + PlaneAge + ArrDelay, data=train)
summary(lm3)
```

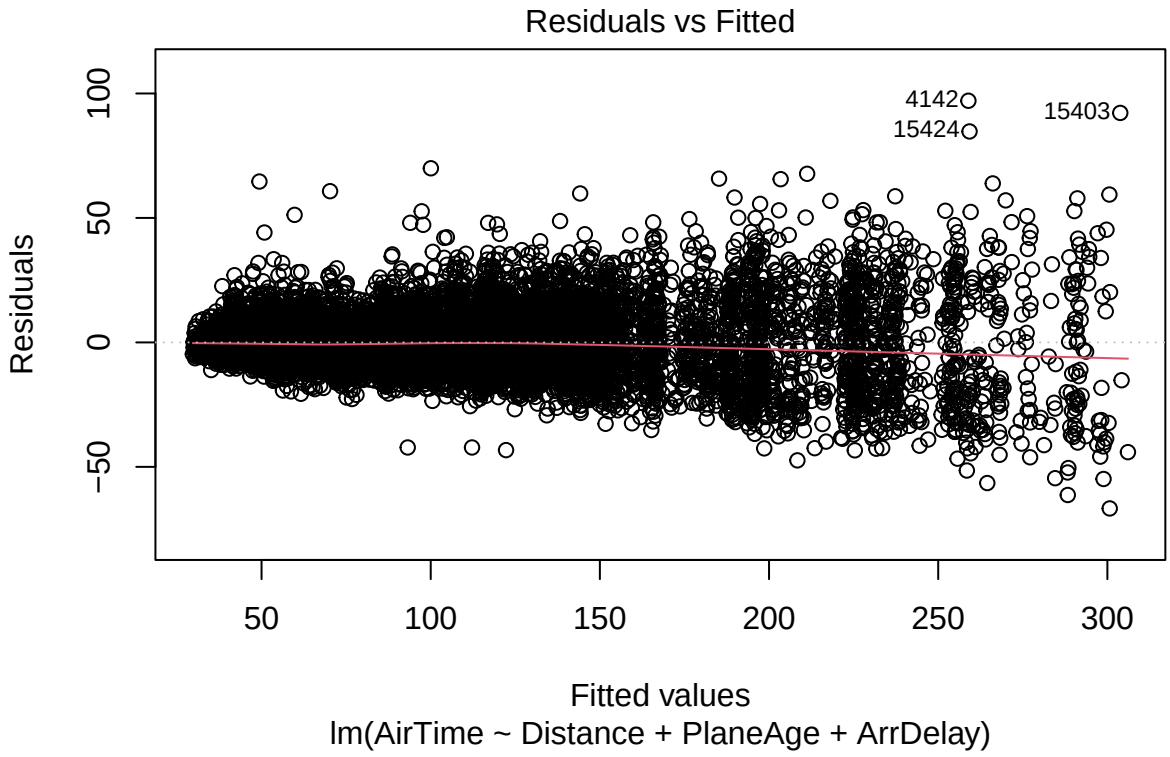
```
##
## Call:
## lm(formula = AirTime ~ Distance + PlaneAge + ArrDelay, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -66.697  -5.052  -0.781   4.402  97.073
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.412e+01  2.392e-01  59.034  < 2e-16 ***
## Distance    1.207e-01  1.887e-04 639.581  < 2e-16 ***
## PlaneAge    6.220e-02  1.706e-02   3.646 0.000267 ***
## ArrDelay    4.737e-02  2.590e-03  18.288  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.66 on 15995 degrees of freedom
## Multiple R-squared:  0.9628, Adjusted R-squared:  0.9628
## F-statistic: 1.381e+05 on 3 and 15995 DF,  p-value: < 2.2e-16
```

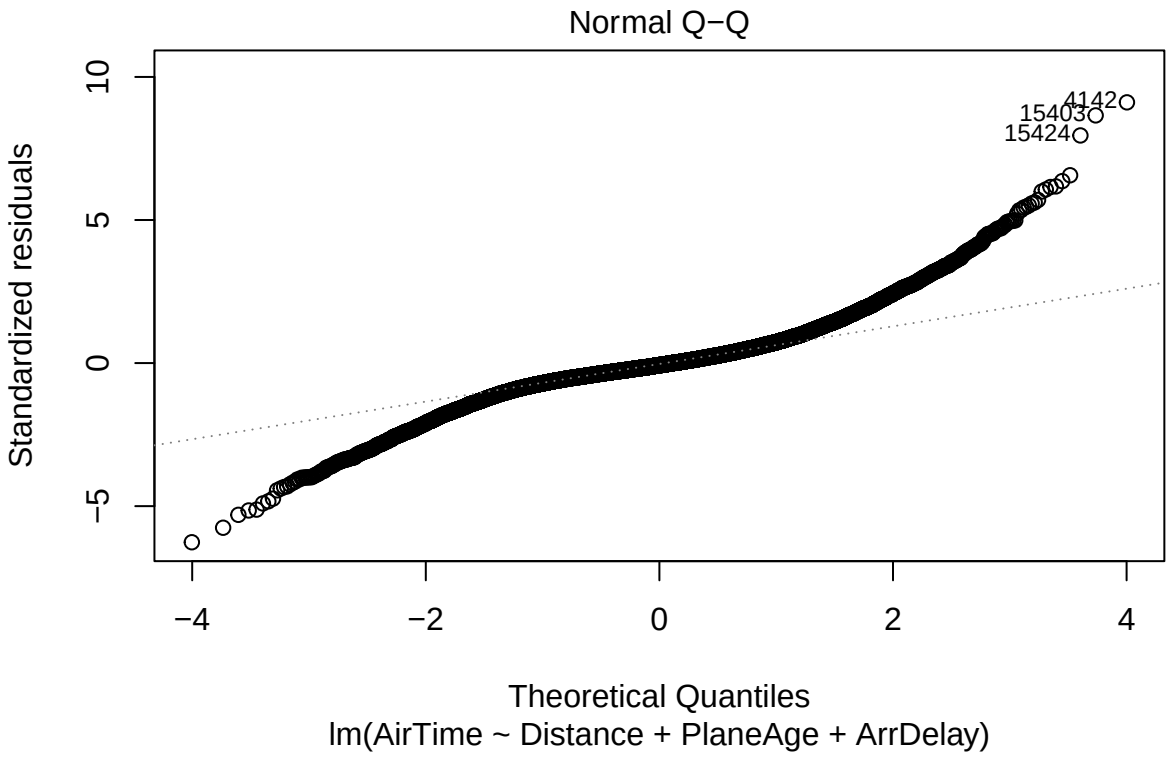
The multiple linear regression model reveals that the equation to predict the airtime of a plane based off the distance the plane travel, the age of the plane, and the arrival delay of the airplane is:

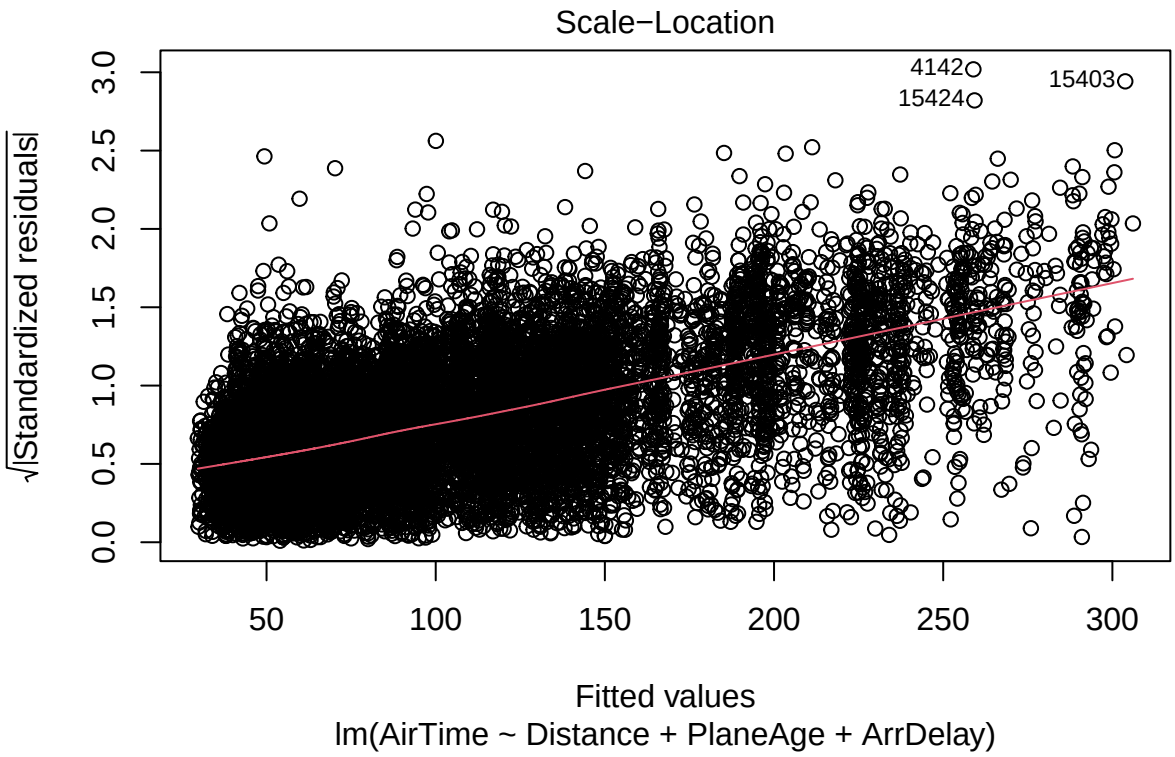
airtime (in minutes) = (0.12041 * distance(in miles)) + (0.06701 * plane_age(in years)) + (0.04851 * arrival_delay(in minutes)) + 14.16348

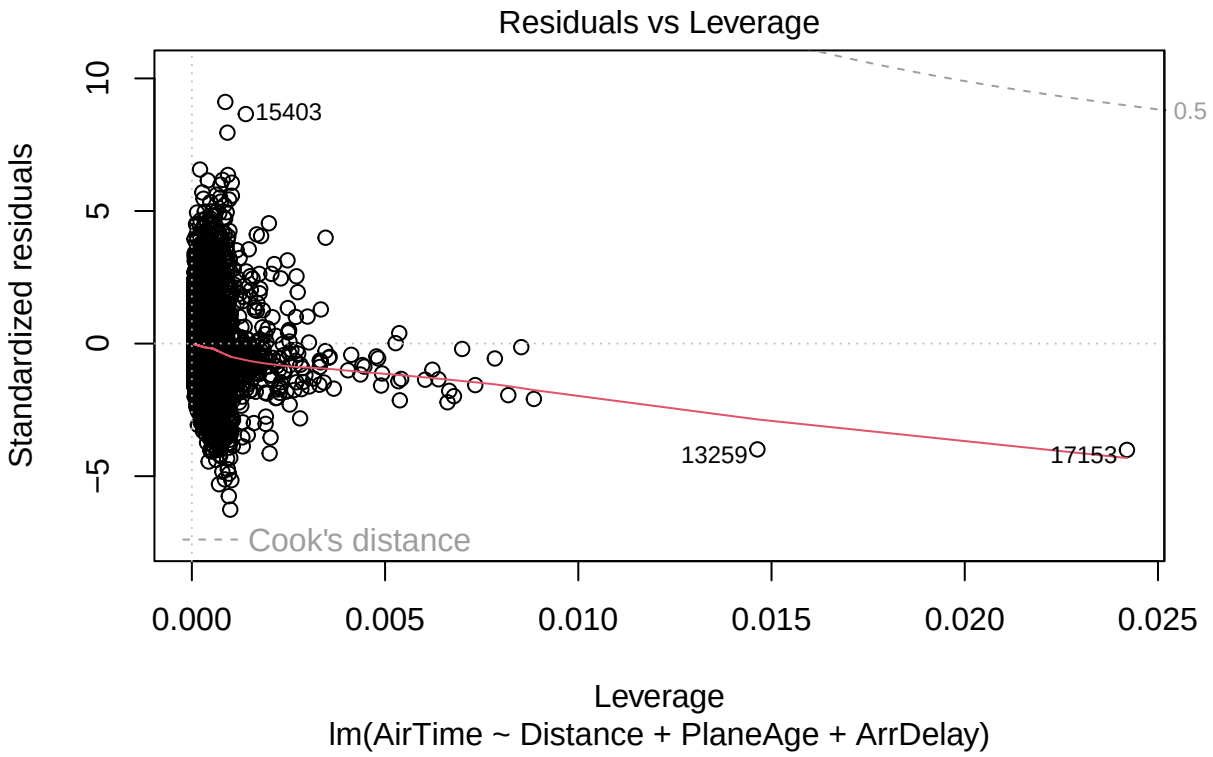
Plotting the residuals show that this linear regression model fits the data well.

```
plot(lm3)
```




Normal Q−Q
Standardized residuals
10
5
0
−5
4142
15403
15424
−4
−2
0
2
4
Theoretical Quantiles
lm(AirTime ~ Distance + PlaneAge + ArrDelay)


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
2.5
3.0
4142
15424
15403
50
100
150
200
250
300
Fitted values
lm(AirTime ~ Distance + PlaneAge + ArrDelay)



Evaluating Results

In this exercise, we created three linear regression models. lm1 sought to understand how the distance a plane travels (in miles) affects the airtime of the plane (in minutes). lm2 sought to understand how the distance a plane travels (in miles) and how old the plane is (in years) affect the total airtime of the plane (in minutes). lm3 sought to understand how the airtime of an airplane (in minutes) can be affected by the distance an airplane travels (in miles) along with the age of the plane (in years) along with the arrival delay of the plane (in minutes). Out of all of the linear regression models created, lm3 is the best model. It has the lowest RSE, the highest R-squared value, and a F-statistic greater than one combined with a low p-value. All of these statistics make it the best model to predict the airtime of an airplane with.

Prediction and Evaluation with Test Data

We can use the testing data set to evaluate our linear regression models. Calculating metrics such as correlation and mean square error (MSE) provide valuable insight into evaluation. A correlation closer to +1 would show that changes in the predictor variables lead to changes in the outcome variable. The data shows that lm3 has the highest correlation. The MSE averages the squared difference between the actual values versus the predicted values in a data set. Once again, lm3 has the lowest MSE. The reason why lm3 may have higher correlation between predictor variables and the outcome variable and also a lower MSE could be because it uses the most predictor variables to predict the outcome variable. By using more predictor variables, a linear regression model may be able to obtain a more accurate outcome variable.

For lm1:

```
pred1 <- predict(lm1, newdata=test)
```

```
correlation1 <- cor(pred1, test$AirTime)
print(paste("correlation: ", correlation1))
```

```
## [1] "correlation:  0.981348878047933"
```

```
mse1 <- mean((pred1 - test$AirTime)^2)
print(paste("mse: ", mse1))
```

```
## [1] "mse:  113.447827925782"
```

```
rmse1 <- sqrt(mse1)
print(paste("rmse: ", rmse1))
```

```
## [1] "rmse:  10.6511890381207"
```

For lm2:

```
pred2 <- predict(lm2, newdata=test)
```

```
correlation2 <- cor(pred2, test$AirTime)
print(paste("correlation: ", correlation2))
```

```
## [1] "correlation:  0.981377415872705"
```

```
mse2 <- mean((pred2 - test$AirTime)^2)
print(paste("mse: ", mse2))
```

```
## [1] "mse:  113.276202224534"
```

```
rmse2 <- sqrt(mse2)
print(paste("rmse: ", rmse2))
```

```
## [1] "rmse:  10.6431293435969"
```

For lm3:

```
pred3 <- predict(lm3, newdata=test)
```

```
correlation3 <- cor(pred3, test$AirTime)
print(paste("correlation: ", correlation3))
```

```
## [1] "correlation:  0.981747033020673"
```

```
mse3 <- mean((pred3 - test$AirTime)^2)
print(paste("mse: ", mse3))
```

```
## [1] "mse:  111.075987033033"
```

```
rmse3 <- sqrt(mse3)
print(paste("rmse: ", rmse3))
```

```
## [1] "rmse:  10.5392593208931"
```