Regression Project Code

Sanjana Battula, Farah Beche, Kajal Gupta, and Shreya Ramella

2024-04-21

This dataset provides comprehensive information about each driver's performance throughout the 2023 Formula 1 racing season, including their qualifying times, race positions, lap times, pit stop durations, and final rankings.

year: Formula 1 racing year, specifically for the 2023 season.

code: Driver code, representing the first three letters of each driver's last name.

q1: Lap time recorded by the driver during Qualifying 1 (Q1) session.

q2: Lap time recorded by the driver during Qualifying 2 (Q2) session.

q3: Lap time recorded by the driver during Qualifying 3 (Q3) session.

race_name: Name of the race circuit where the Formula 1 race took place.

position: Grid position of the driver at the start of the race.

rank: Final ranking or finishing position of the driver at the end of the race.

laps: Total number of laps completed by the driver during the race.

fastestLapTime: Fastest lap time recorded by the driver during the race.

tot_pit_time: Total amount of time it took for pit stops during the race, measured in milliseconds.

labels: Description of the status of the driver during the race (e.g., finished, retired, disqualified).

points: Points awarded to each driver based on their finishing position in the race.

statusId: Status of the race for the driver in numbers

fastestlap_ms: Fastest lap time recorded by the driver during the race, measured in milliseconds.

q1_ms: Lap time recorded by the driver during Qualifying 1 (Q1) sessions, measured in milliseconds

q2_ms: Lap time recorded by the driver during Qualifying 2 (Q2) sessions, measured in milliseconds

q3_ms: Lap time recorded by the driver during Qualifying 3 (Q3) sessions, measured in milliseconds

Loading the Dataset

```r
# Loading necessary libraries
library(dplyr)  # for data manipulation
# Clear the environment
rm(list = ls())
# Read data from the CSV file into a dataframe named F1Final
F1Final <- read.csv("F1Data.csv", header = TRUE)
# "F1Data.csv" is the file containing the F1 racing data, with headers present
```

Converting time data (fastestLapTime, q1, q2 & q3) into milliseconds

```r
# Conversions of values into milliseconds
# Removing white spaces
library(lubridate)
F1Final$fastestLapTime <- trimws(F1Final$fastestLapTime)
F1Final$fastestlap_ms = as.numeric(lubridate::ms(as.character(F1Final$fastestLapTime)))*1000
# removing white space for q1
F1Final$q1 <- trimws(F1Final$q1)
F1Final$q1_ms = as.numeric(lubridate::ms(as.character(F1Final$q1)))*1000
# removing white space for q2
F1Final$q2 <- trimws(F1Final$q2)
F1Final$q2_ms = as.numeric(lubridate::ms(as.character(F1Final$q2)))*1000
# removing white space for q3
F1Final$q3 <- trimws(F1Final$q3)
F1Final$q3_ms = as.numeric(lubridate::ms(as.character(F1Final$q3)))*1000
```

Printing the first rows:

```r
# Creating a sub-dataset
#install.packages("dplyr")
library(dplyr)
# Selecting specific columns from the F1Final df and creating a new df(F1_Subdf)
F1_Subdf <- F1Final %>%
  select(statusId, position, rank, fastestlap_ms, tot_pit_time, laps, points,
         q1_ms, q2_ms, q3_ms)
# Displaying the first few rows of the F1_Subdf dataframe
head(F1_Subdf)
```

```
##   statusId position rank fastestlap_ms tot_pit_time laps points q1_ms q2_ms
## 1        1        7    5         96546        49372   57     10 91543 90513
## 2        1        5    3         96156        50669   57     15 91158 90645
## 3       11       10   15         96616        90924   57      0 91204 90809
## 4        1        2    2         96344        49355   57     18 91479 90746
## 5        1       12    8         97379        51042   57      4 91504 91443
## 6       11       17   13         96471        75908   57      0 91892    NA
##   q3_ms
## 1 90384
## 2 90336
## 3    NA
## 4 89846
## 5    NA
## 6    NA
```

Univariable Analysis:

```r
# Display the number of rows in the dataframe
nrow(F1_Subdf)
```

```
## [1] 389
```

```r
# Show descriptive statistics for each variable in the F1_Subdf dataframe
summary(F1_Subdf)
```

```
##     statusId         position          rank        fastestlap_ms
##  Min.   :  1.000   Min.   : 1.00   Min.   : 1.000   Min.   : 67012
##  1st Qu.:  1.000   1st Qu.: 5.00   1st Qu.: 5.000   1st Qu.: 78594
##  Median :  1.000   Median :10.00   Median : 9.000   Median : 87493
##  Mean   :  4.769   Mean   :10.36   Mean   : 9.401   Mean   : 88010
##  3rd Qu.:  1.000   3rd Qu.:15.00   3rd Qu.:14.000   3rd Qu.: 96371
##  Max.   :130.000   Max.   :20.00   Max.   :23.000   Max.   :111682
##
##   tot_pit_time          laps           points            q1_ms
##  Min.   :  20026   Min.   :44.00   Min.   : 0.000   Min.   : 65116
##  1st Qu.:  29698   1st Qu.:53.00   1st Qu.: 0.000   1st Qu.: 78540
##  Median :  48100   Median :57.00   Median : 2.000   Median : 85007
##  Mean   : 393226   Mean   :60.17   Mean   : 5.761   Mean   : 86003
##  3rd Qu.:  80703   3rd Qu.:70.00   3rd Qu.:10.000   3rd Qu.: 91461
##  Max.   :3703013   Max.   :78.00   Max.   :26.000   Max.   :128510
##                                                     NA's   :4
##      q2_ms            q3_ms
##  Min.   : 64951   Min.   : 64391
##  1st Qu.: 77704   1st Qu.: 76818
##  Median : 84704   Median : 84553
##  Mean   : 85266   Mean   : 84060
##  3rd Qu.: 91048   3rd Qu.: 90320
##  Max.   :126688   Max.   :108841
##  NA's   :95       NA's   :198
```

```r
# Extract the mean of the variable of interest
summary_outcome_1 <- summary(F1_Subdf$points)
mean_outcome_2 <- summary_outcome_1["Mean"]
mean_outcome_2
```

```
##     Mean
## 5.760925
```

Simple Linear Regression

```r
# Fit a linear model with all possible predictors (except the outcome variable)
# to predict 'points'
lm_mod1 <- lm(points ~ ., data = F1_Subdf)
# Display summary statistics of the linear model
summary(lm_mod1)
```

```
##
## Call:
## lm(formula = points ~ ., data = F1_Subdf)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.1214 -1.9204 -0.5674  1.4363  8.3019
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    1.692e+01  5.744e+00   2.946  0.00364 **
## statusId       1.004e-01  2.377e-02   4.226 3.77e-05 ***
```

```
## position      -6.313e-01  9.532e-02  -6.623 3.89e-10 ***
## rank          -1.516e+00  6.440e-02 -23.538  < 2e-16 ***
## fastestlap_ms -3.957e-06  1.030e-04  -0.038  0.96938
## tot_pit_time   2.498e-07  3.105e-07   0.804  0.42221
## laps           4.088e-02  4.230e-02   0.966  0.33512
## q1_ms         -4.484e-05  2.535e-04  -0.177  0.85979
## q2_ms          2.441e-05  3.676e-04   0.066  0.94713
## q3_ms          6.155e-05  1.023e-04   0.602  0.54816
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.982 on 181 degrees of freedom
##   (198 observations deleted due to missingness)
## Multiple R-squared:  0.8666, Adjusted R-squared:  0.8599
## F-statistic: 130.6 on 9 and 181 DF,  p-value: < 2.2e-16
```

Rank, Position and statusId are significant predictors.

Interpreting the coeffcients:

statusId: For each unit increase in statusId, the expected points earned by a driver increase by approximately 0.1004, holding all other predictors constant.

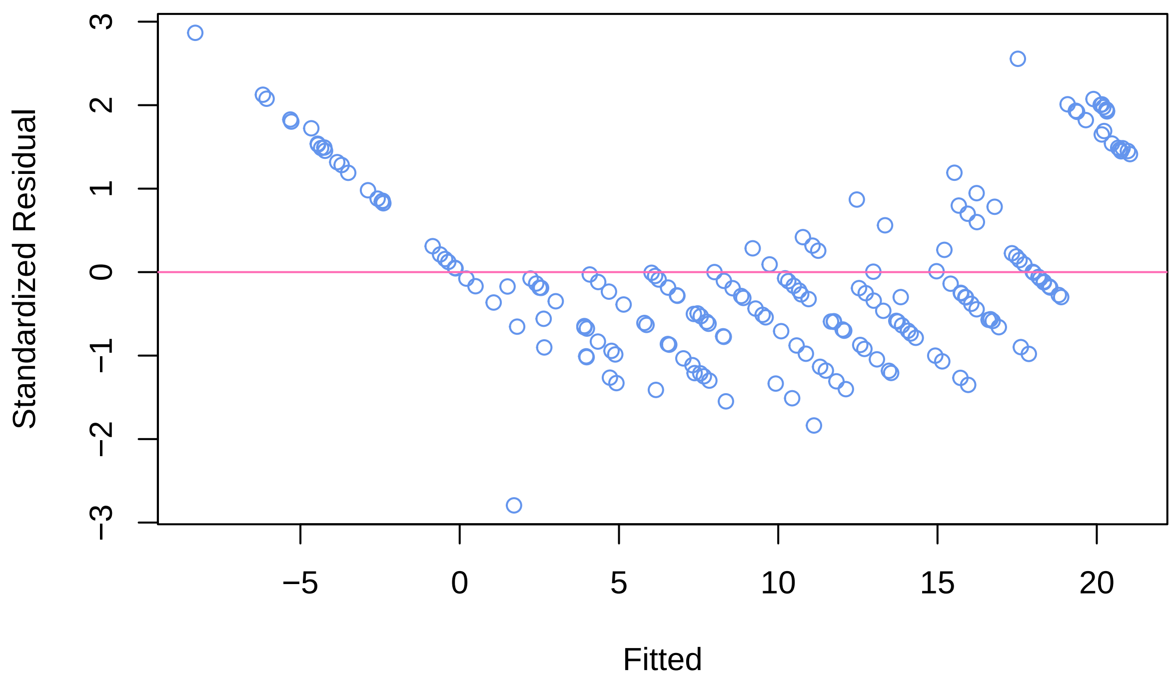
position: For each unit decrease in the driver's position, the expected points earned increase by approximately 0.6313, holding all other predictors constant. This suggests that starting lower in a race leads to more points.

rank: For each unit decrease in the driver's rank, the expected points earned increase by approximately 1.516. This might seem counterintuitive at first, but in Formula 1, a lower rank (closer to 1) actually indicates a better performance and more points.

fastestlap_ms, tot_pit_time, laps, q1_ms, q2_ms, q3_ms: These predictors do not significantly influence the points earned by the driver, based on their coefficients and p-values.

Plots of Linear Model

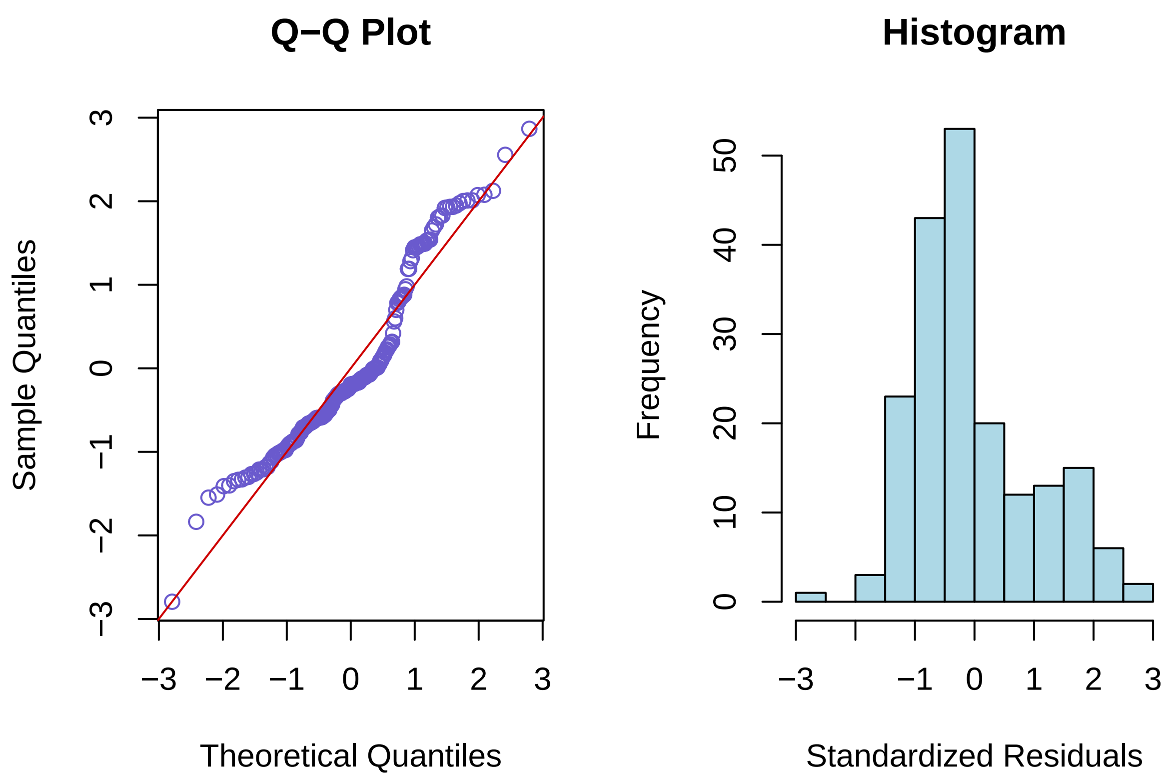
```r
# Plotting Standardized residuals vs. fitted values
plot(fitted(lm_mod1), rstandard(lm_mod1), xlab = "Fitted",
     ylab = "Standardized Residual" , col="cornflowerblue" )
abline(h = 0, col = "hotpink")
```



The residuals are scattered around the horizontal zero line, forming a horizontal band without any clear pattern or trend across the range of fitted values. This suggests that the assumptions of constant variance and linearity in the regression model are likely satisfied. However, there are a few potential outliers or influential points with large positive or negative residuals that may warrant further investigation.

QQ PLOT

```r
# Set up layout for two panels side by side
par(mfrow = c(1, 2))
# Left panel: QQ plot of standardized residuals
qqnorm(rstandard(lm_mod1), main = "Q-Q Plot" , col = "slateblue")
abline(0, 1, col = "red3")
# Right panel: Histogram of standardized residuals
hist(rstandard(lm_mod1), main = "Histogram", xlab = "Standardized Residuals",
     col = "lightblue")
```



The plot on the left is a Q-Q (Quantile-Quantile) plot of standardized residuals against the theoretical quantiles of a normal distribution. In this plot there are points towards the tails that are deviating indicating a violation of the normality assumption for the residuals. The plot on the right is a histogram of the standardized residuals. This plot appears to be slightly skewed to the right, with a peak around 0 and a longer tail on the positive side. This further suggests a deviation from the normality assumption for the residuals. Together, these two plots indicate that the assumptions of normality and homoscedasticity (constant variance of residuals) for the linear regression model may be violated.

Pearsons Correlation

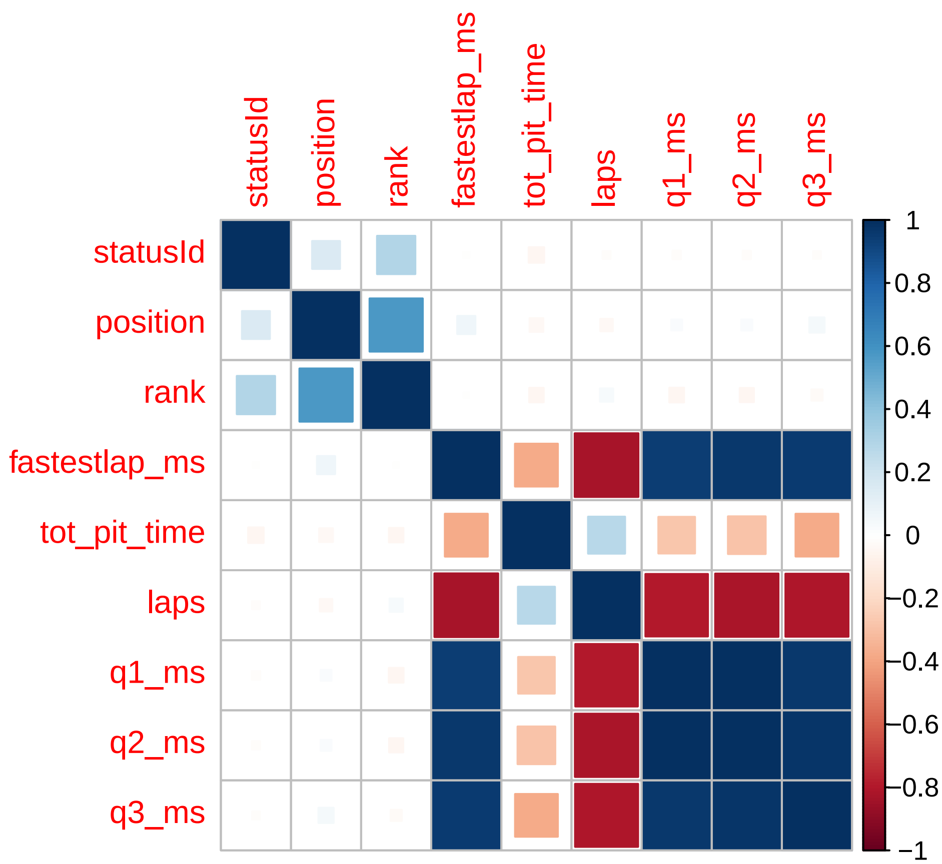
```
# Remove rows with NA values from the F1_Subdf dataframe
F1_df <- na.omit(F1_Subdf)
corr_df <- F1_df %>% select(-points)
# Compute the correlation matrix for the F1_df dataframe
cor_matrix <- cor(corr_df)
# Display the correlation matrix
cor_matrix
```

```
##                   statusId    position         rank fastestlap_ms tot_pit_time
## statusId       1.000000000  0.15804683  0.297646788  -0.007586942  -0.04836672
## position       0.158046835  1.00000000  0.574794721   0.066574575  -0.03810499
## rank           0.297646788  0.57479472  1.000000000  -0.006071924  -0.04215391
## fastestlap_ms -0.007586942  0.06657457 -0.006071924   1.000000000  -0.37355047
## tot_pit_time  -0.048366724 -0.03810499 -0.042153910  -0.373550467   1.00000000
## laps          -0.012438740 -0.03128938  0.035510878  -0.823958204   0.27795288
## q1_ms         -0.014701827  0.02359591 -0.043755882   0.941599250  -0.27268529
## q2_ms         -0.013791289  0.02405340 -0.040400242   0.962428402  -0.28879524
```

```
## q3_ms          -0.012211590  0.04656738 -0.025057451   0.951118445  -0.37126339
##                       laps       q1_ms       q2_ms       q3_ms
## statusId       -0.01243874 -0.01470183 -0.01379129 -0.01221159
## position       -0.03128938  0.02359591  0.02405340  0.04656738
## rank            0.03551088 -0.04375588 -0.04040024 -0.02505745
## fastestlap_ms -0.82395820  0.94159925  0.96242840  0.95111844
## tot_pit_time    0.27795288 -0.27268529 -0.28879524 -0.37126339
## laps            1.00000000 -0.79411707 -0.81363410 -0.80663709
## q1_ms          -0.79411707  1.00000000  0.99509099  0.96139034
## q2_ms          -0.81363410  0.99509099  1.00000000  0.97249422
## q3_ms          -0.80663709  0.96139034  0.97249422  1.00000000
```

Correlation Matrix plot

```
library(corrplot)
# Plot the correlation matrix using corrplot
corrplot(cor_matrix, method = "square")
```



Outliers

```
# Calculate the standardized residuals from the linear model lm_mod1
std_resid <- rstandard(lm_mod1)
# Identify the indices of observations with standardized residuals > 3 or < -3
outlier_indices <- which(std_resid > 3 | std_resid < -3)
# Print the standardized residuals
print(std_resid)
```

```
##             1             2             4             7             8 
## -0.2219058211  0.0104784894 -0.1138409257  1.5401559638 -0.7022753813 
##             9            12            18            19            21 
## -0.1925773575 -0.7685655303  0.0922366480 -0.3028076666  1.6489932344 
##            25            26            27            28            30 
## -0.7074662581 -0.6091209960 -0.1369096238 -1.5108847413 -0.5918220403 
##            34            36            37            38            42 
##  2.0771875858 -0.0788004506 -0.3797666544 -0.0888070652  1.4505065435 
##            43            45            46            53            54 
## -0.6533323113 -0.4632589062  0.8249331399 -0.8790280537 -0.3417735924 
##            56            59            60            62            64 
##  1.9197179005 -0.1715801949 -0.9202500088 -0.1853669328 -0.8986009615 
##            65            68            69            72            74 
## -0.9430352763 -0.5588913483  0.0509001936 -0.5852197154 -0.2993670340 
##            76            78            79            81            82 
## -1.3292589422 -1.0449398342 -0.6466505582 -1.1145699878 -0.5280846827 
##            84            91            92            97            98 
## -0.2520814239  0.0051809469 -0.1779717587  1.4891406679 -1.2124451858 
##            99           100           101           102           103 
## -0.2555070968 -0.6186301850 -1.3080870927 -0.6609085245 -0.0741848857 
##           105           110           111           112           116 
##  1.5377822659  0.5993745008 -0.0460757621  1.4864017791  1.9760525950 
##           117           118           119           120           122 
## -1.2078864609 -0.6321408014 -0.5120077574 -1.2642286056  1.4923914497 
##           128           130           131           132           136 
##  0.8800250647 -0.4452414313 -0.1221886549 -0.0775517376  1.4135484176 
##           137           138           142           147           148 
## -0.2663426429 -1.2097764824  0.3106646113 -1.0333677351 -0.1080691594 
##           152           153           155           156           157 
##  2.0090474477 -1.1342502915 -0.9861172605 -0.1722626367  0.0030752952 
##           158           160           166           167           171 
## -0.5840600733  0.2115098676  0.5608825515 -0.0080375868  2.0022672461 
##           172           174           175           176           177 
## -1.0085478692 -2.7941565325 -1.4111129035 -0.0068273306 -0.3232078710 
##           178           182           184           185           186 
## -0.3880396665 -0.7866146792 -1.3507546921 -0.6762256400  1.3170270659 
##           187           189           191           194           195 
##  0.8694102972 -0.3641309250  2.0732925724 -0.7749022332  0.0932600431 
##           199           200           202           203           205 
##  1.2830208790 -0.8727569800 -0.2993522019  0.2850227878  0.1881015119 
##           209           211           213           214           215 
##  1.6898424521 -0.1868610781 -0.5711549673 -0.5009722125 -0.1044828522 
##           221           223           226           227           231 
##  0.7829380953 -0.1917107962  1.4840619218 -0.5920958240 -1.8372819823 
##           232           233           235           237           238 
##  1.7227761490 -1.5481406012 -0.8320030228  0.0003505703 -0.0770259096 
##           240           243           244           247           248 
##  0.6996342306  1.9326946318 -0.6597059694 -0.7352261290 -0.1187604996 
##           249           250           252           255           256 
## -0.6875767493 -0.5946604189  0.1568838481  0.2666246969  1.4533571798 
##           257           259           261           263           264 
##  0.8437071024 -1.0187922212  1.4684497319 -1.0007178222  0.2262491195 
##           265           270           271           272           275 
##  0.8543872269 -0.3490777847 -0.1647375471 -0.0282832616  1.9332737671 
```

```
##           276          279          280          281          283
## -0.5417283508 -0.6384767396  0.1441567917 -0.2833700803  0.1222248689
##           285          288          291          293          294
## -0.5645161709 -0.9776240356 -0.2334569058  1.9482005181 -0.2784760774
##           296          297          299          303          306
##  0.0446636316 -0.5954446296 -1.0691232192  0.7971179967  0.3172345285
##           310          311          313          314          315
##  2.5554534749 -0.2481994117 -0.4366486353 -0.1848740815 -0.7019297730
##           320          322          323          324          325
##  0.9459862700 -1.3350353190  1.8278129316  1.8207642656 -1.2670061054
##           329          331          334          335          337
## -0.9799659400 -0.3093442504  1.5275916309 -0.8615101534 -1.2478784282
##           338          340          342          343          345
## -0.2975793673  0.2562902872  1.4472863584 -0.2861696790 -1.1823149417
##           347          352          356          357          358
##  1.1905328683 -0.1890529387  2.8664790131  0.9808455435  2.0103111410
##           359          362          363          364          365
## -1.4017322304 -0.9022218611 -0.2744103064 -1.2998213181 -0.1688828196
##           369          371          372          373          377
##  2.1252167894 -0.4950050209  1.8031541449  0.4188397908  1.9251266328
##           381          382          383          384          386
##  1.1894556008 -0.0578107782 -0.5884425114 -0.1389214812 -0.8692509059
##           388
## -1.1801010900
```

The absence of any output from the line of code indicates that there are no observations in the dataset considered outliers based on their standardized residuals. This implies that none of the data points exhibit unusually large deviations from the predicted values of the outcome variable when compared to the variability of the model. Consequently, the linear regression model appears to adequately capture the relationships between the predictor variables and the outcome variable without any extreme or influential observations. This suggests that the model provides a reasonable fit to the data, and there is no need for further investigation or adjustments to address outlier observations.

Levarage Points

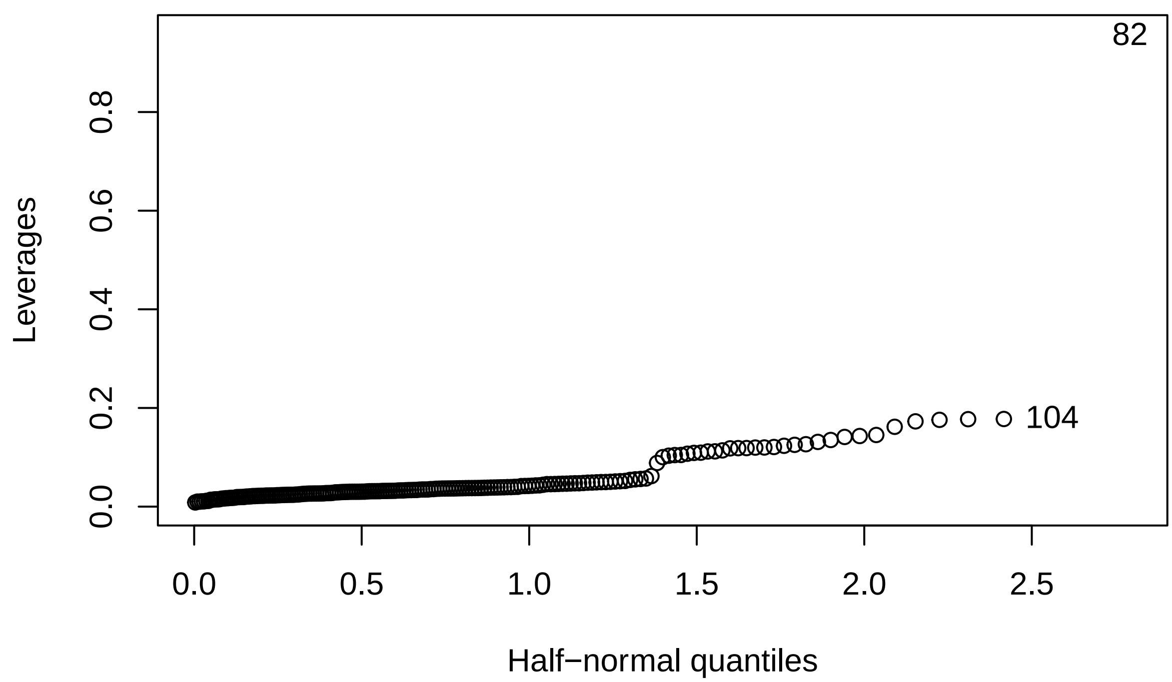
```r
# Calculate the hat values for each observation in the linear model lm_mod1
hatv <- hatvalues(lm_mod1)
# Identify observations with hat values > than twice the mean of all hat values
outlier_hatv <- hatv[hatv > 2 * mean(hatv)]
# Print the identified outlier hat values
outlier_hatv
```

```
##        36        37        38        42        43        45       130       131
## 0.1208626 0.1232711 0.1412422 0.1350758 0.1431305 0.1180867 0.1451899 0.1311677
##       132       136       137       138       142       174       203       205
## 0.1758862 0.1187012 0.1772617 0.1252287 0.1617874 0.9581248 0.1088745 0.1185479
##       211       213       214       215       221       223       227       231
## 0.1728086 0.1139690 0.1815470 0.1118022 0.1072548 0.1196364 0.1198511 0.1264395
##       232       233       235
## 0.1776543 0.1093581 0.1119952
```

The output indicates that certain data points in the dataset have a strong influence on the predictions made by the linear regression model. These points, identified by their respective indices, have leverages (a measure of influence) that are at least twice the average leverage of all data points. This means that these particular *observations disproportionately affect the estimated coefficients and overall fit of the model, potentially due to their extreme values or unique characteristics.*

Leverage Plot

```
# Load the 'faraway' package, which contains the 'halfnorm' function
library(faraway)
# Plot leverage using a half-normal plot with automatic labeling
halfnorm(hatv, nlab = 2, ylab = "Leverages")
```



High leverage points may or may not be influential.

Bonferroni Value

```
# Compute studentized residuals
stud <- rstudent(lm_mod1)
# Calculate the number of observations
n <- nrow(model.matrix(lm_mod1))
# Calculate the number of parameters (including the intercept)
p <- length(coefficients(lm_mod1))
# Calculate the Bonferroni critical value
bonferroni_critical_value <- qt(1 - 0.05 / (n * 2), n - p - 1)
# Print the Bonferroni critical value
print(paste("Bonferroni Value:", bonferroni_critical_value))
```

```
## [1] "Bonferroni Value: 3.72439731939489"
```

```r
# Identify outliers using Bonferroni correction
outliers <- which(abs(stud) > qt(1 - 0.05 / (n * 2), n - p - 1))
```

The absence of any output from the Bonferroni test indicates that no outliers were detected based on this correction method. The Bonferroni correction adjusts the significance level to account for multiple comparisons, reducing the likelihood of false positives when identifying outliers. In this case, none of the observations exhibited studentized residuals that exceeded the Bonferroni critical value, even after considering the adjusted significance level. Therefore, it can be concluded that no outliers were present in the dataset based on the Bonferroni test. This suggests that the linear regression model adequately captures the relationships between the predictor variables and the outcome variable without any extreme or influential observations that would significantly impact the results.

COOK'S distance

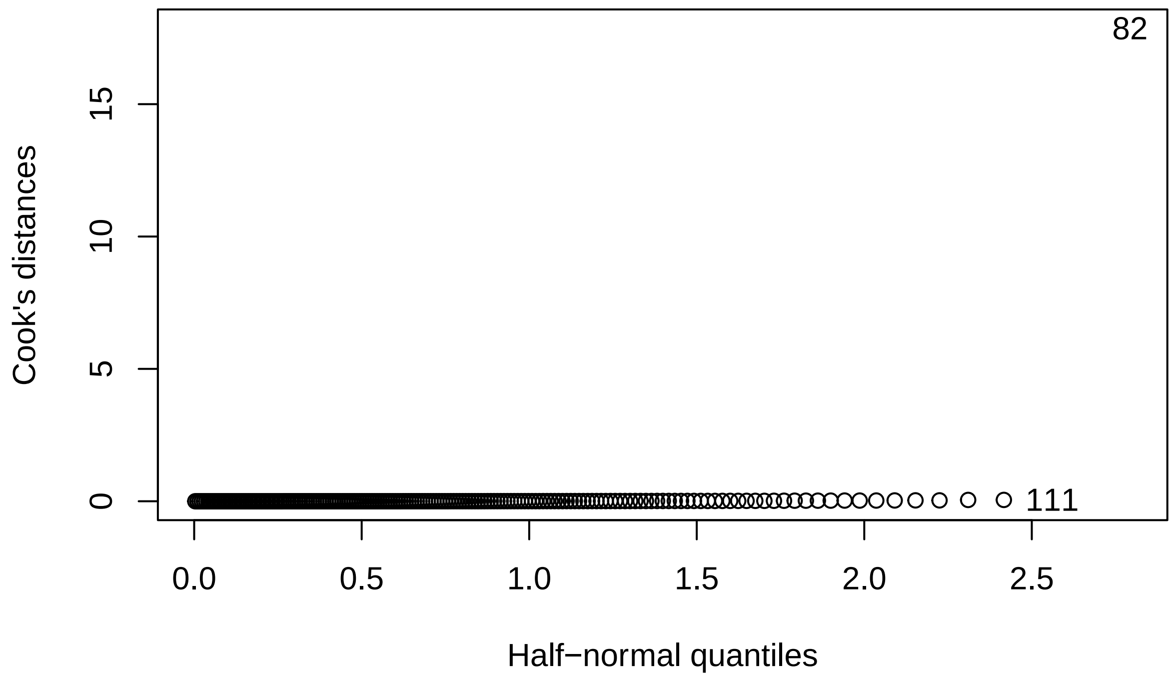
```r
# Calculate Cook's distance for each observation in the linear model lm_mod1
cook <- cooks.distance(lm_mod1)
# Identify points with Cook's distance above 0.5
outlier_cook <- cook[which(cook > 0.5)]
outlier_cook
```

```
##      174
## 17.86349
```

Cook's distance measures the effect of deleting a given observation from a linear regression model. A large Cook's distance for a specific observation suggests that excluding that observation would substantially alter the model's predictions. In this instance, the only observation listed is 174, indicating that removing this particular influential data point would notably affect the model's fitted values. This suggests that observation 174 has characteristics or values that strongly influence the model's predictions, warranting further investigation into its potential impact on the overall analysis.

Cook's Distance Plot

```r
# Load the 'faraway' package, which contains the 'halfnorm' function
library(faraway)
# Plot Cook's distances using a half-normal plot with automatic labeling
halfnorm(cook, nlab = 2, ylab = "Cook's distances")
```



Influential point from the above plot is observation number 82.

Influential Point:

```
F1_Subdf[82,]
```

```
##    statusId position rank fastestlap_ms tot_pit_time laps points q1_ms q2_ms
## 82        1        7    7         91434        22068   57      6 87713 86964
##    q3_ms
## 82 87861
```

Model with removed Influential Observation:

```
F1_Subset <- F1_Subdf[-c(82),]
lmodi <- lm(points ~ ., data = F1_Subset)
summary(lmodi)
```

```
##
## Call:
## lm(formula = points ~ ., data = F1_Subset)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.0989 -1.9299 -0.5798  1.5266  8.2883
##
## Coefficients:
```

```
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     1.694e+01  5.756e+00   2.943  0.00368 **
## statusId        1.003e-01  2.382e-02   4.210 4.03e-05 ***
## position       -6.294e-01  9.558e-02  -6.585 4.84e-10 ***
## rank           -1.516e+00  6.453e-02 -23.496  < 2e-16 ***
## fastestlap_ms -3.353e-06  1.032e-04  -0.032  0.97411
## tot_pit_time    2.489e-07  3.112e-07   0.800  0.42475
## laps            4.061e-02  4.239e-02   0.958  0.33937
## q1_ms          -4.466e-05  2.540e-04  -0.176  0.86062
## q2_ms           2.196e-05  3.684e-04   0.060  0.95253
## q3_ms           6.326e-05  1.026e-04   0.617  0.53810
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.988 on 180 degrees of freedom
##   (198 observations deleted due to missingness)
## Multiple R-squared:  0.8666, Adjusted R-squared:  0.8599
## F-statistic: 129.9 on 9 and 180 DF,  p-value: < 2.2e-16
```

VIF of Full Model

```r
# Extract the model matrix from the linear model excluding the intercept column
VIF <- model.matrix(lm_mod1)[,-1]
# Calculate the Variance Inflation Factors (VIF) for the predictor variables
vif_values <- vif(VIF)
vif_values
```

```
##      statusId      position          rank fastestlap_ms  tot_pit_time
##      1.104508      1.557096      1.612261     25.573835      1.494309
##          laps         q1_ms         q2_ms         q3_ms
##      3.259143    175.041241    324.769888     23.726121
```

VIF measures the severity of multicollinearity, which occurs when predictor variables are highly correlated with each other. Any values over 5 is high while above 10 is severe implying multicollinearity. This suggests high multicollinearity among fastestlap_ms, q1_ms, q2_ms & q3_ms.

VIF of Model removing q2_ms

```r
lm_mod2 <- lm(points ~ position + statusId + rank + fastestlap_ms +
                tot_pit_time + laps + q1_ms + q3_ms, data = F1_Subdf)
# Extract the model matrix from the linear model, excluding the intercept column
VIF2 <- model.matrix(lm_mod2)[,-1]
# Calculate the Variance Inflation Factors (VIF) for the predictor variables
vif_values2 <- vif(VIF2)
vif_values2
```

```
##      position      statusId          rank fastestlap_ms  tot_pit_time
##      1.514253      1.103831      1.611355     13.532164      1.366363
##          laps         q1_ms         q3_ms
##      3.198540     17.319478     19.331708
```

Although reduced the vif values for the predictors (fastestlap_ms, q1_ms, and q3_ms) are still above 10.

VIF of Model removing fastestlap_ms

```r
lm_mod3 <- lm(points ~ position + statusId + rank + tot_pit_time +
                laps + q1_ms + q2_ms + q3_ms, data = F1_Subdf)
# Extract the model matrix from the linear model, excluding the intercept column
VIF3 <- model.matrix(lm_mod3)[,-1]
# Calculate the Variance Inflation Factors (VIF) for the predictor variables
vif_values3 <- vif(VIF3)
vif_values3
```

```
##     position     statusId         rank tot_pit_time         laps        q1_ms
##     1.514284     1.102663     1.612028     1.315495     3.209408   118.355369
##        q2_ms        q3_ms
##   171.849058    23.620843
```

Although reduced the vif values for the predictors (q1_ms, q2_ms and q3_ms) are still above 10.

VIF of Model removing q1_ms

```r
lm_mod4 <- lm(points ~ position + statusId + rank + tot_pit_time + laps +
                fastestlap_ms + q2_ms + q3_ms, data = F1_Subdf)
# Extract the model matrix from the linear model, excluding the intercept column
VIF4 <- model.matrix(lm_mod4)[,-1]
# Calculate the Variance Inflation Factors (VIF) for the predictor variables
vif_values4 <- vif(VIF4)
vif_values4
```

```
##      position      statusId          rank  tot_pit_time          laps
##      1.528868      1.104298      1.610546      1.453347      3.200740
## fastestlap_ms         q2_ms         q3_ms
##     17.291929     32.134398     22.584112
```

Although reduced the vif values for the predictors (fastestlap_ms, q2_ms and q3_ms) are still above 10.

VIF of Model removing q3_ms

```r
lm_mod5 <- lm(points ~ position + statusId + rank + tot_pit_time + laps +
                fastestlap_ms + q2_ms + q1_ms, data = F1_Subdf)
# Extract the model matrix from the linear model, excluding the intercept column
VIF5 <- model.matrix(lm_mod5)[,-1]
# Calculate the Variance Inflation Factors (VIF) for the predictor variables
vif_values5 <- vif(VIF5)
vif_values5
```

```
##      position      statusId          rank  tot_pit_time          laps
##      1.849435      1.163053      2.026921      1.248401      3.279585
## fastestlap_ms         q2_ms         q1_ms
##     11.653413      8.993870     14.024437
```

If we removed q3_ms the vif value of q2_ms drops significantly to be lower than 10.

AIC

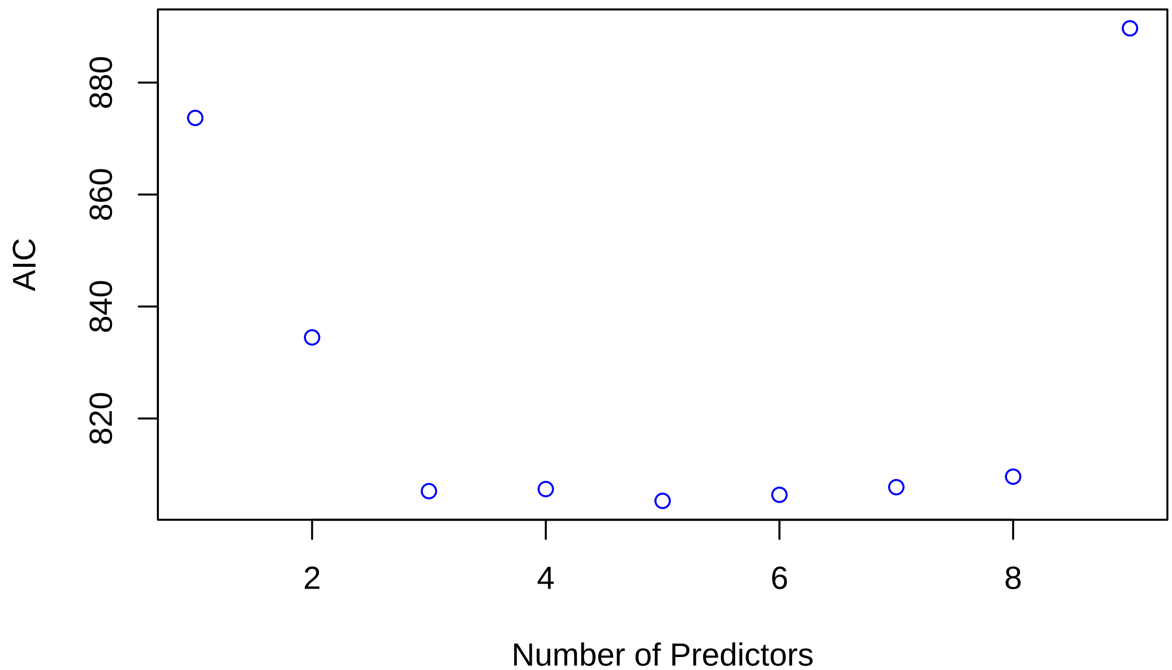
```r
# Load the 'leaps' package for subset selection
library(leaps)
# Create the full model including all predictors
full_model <- lm(points ~ position + statusId + rank + tot_pit_time + laps +
                   fastestlap_ms + q2_ms + q1_ms, data = F1_Subset)
# Subset selection using the 'regsubsets' function from the 'leaps' package
B <- regsubsets(points ~ position + statusId + rank + tot_pit_time + laps +
                  fastestlap_ms + q2_ms + q1_ms, data = F1_Subset)
rs <- summary(B)
# Display the predictors selected by each model size based on the 'regsubsets' output
rs$which
```

```
##   (Intercept) position statusId rank tot_pit_time  laps fastestlap_ms q2_ms
## 1        TRUE    FALSE    FALSE TRUE        FALSE FALSE         FALSE FALSE
## 2        TRUE    FALSE     TRUE TRUE        FALSE FALSE         FALSE FALSE
## 3        TRUE     TRUE     TRUE TRUE        FALSE FALSE         FALSE FALSE
## 4        TRUE     TRUE     TRUE TRUE        FALSE FALSE         FALSE  TRUE
## 5        TRUE     TRUE     TRUE TRUE        FALSE FALSE          TRUE  TRUE
## 6        TRUE     TRUE     TRUE TRUE        FALSE  TRUE          TRUE  TRUE
## 7        TRUE     TRUE     TRUE TRUE        FALSE  TRUE          TRUE  TRUE
## 8        TRUE     TRUE     TRUE TRUE         TRUE  TRUE          TRUE  TRUE
##   q1_ms
## 1 FALSE
## 2 FALSE
## 3 FALSE
## 4 FALSE
## 5 FALSE
## 6 FALSE
## 7  TRUE
## 8  TRUE
```

```r
# Calculate the Akaike Information Criterion (AIC) for each model size
k <- nrow(F1_Subdf) # Number of observations
s <- 2:10 # Model sizes from 2 to 10 predictors
AIC <- k * log(rs$rss / k) + 2 * s
AIC
```

```
## [1] 873.6923 834.4854 807.0182 807.3930 805.2839 806.3642 807.7297 809.6093
## [9] 889.6923
```

```r
# Plot AIC values
plot(AIC ~ I(s - 1), ylab = "AIC", xlab = "Number of Predictors", col = "blue")
```



The AIC (Akaike Information Criterion) test helps identify the best combination of predictors for a model by considering both the goodness of fit and the complexity of the model. In this case, the AIC values were calculated for different model sizes, ranging from 2 to 10 predictors. The model with the lowest AIC value indicates the best trade-off between model fit and complexity. Based on this test we determined that the third model has the lowest AIC value of 807.0182, which includes predictors "statusId" "position" and "rank".

AIC Selected Model

```
# Fit a linear regression model using the predictors selected by AIC
AIC_model <- lm(points ~ statusId + position + rank, data = F1_Subset)
summary(AIC_model)
```

```
##
## Call:
## lm(formula = points ~ statusId + position + rank, data = F1_Subset)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16.2257  -2.6469  -0.6867   2.2717   9.7753
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 18.11218    0.40167  45.092  < 2e-16 ***
## statusId     0.03353    0.01268   2.644  0.00853 **
## position    -0.20145    0.04351  -4.630    5e-06 ***
## rank        -1.10796    0.04880 -22.704  < 2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.589 on 384 degrees of freedom
## Multiple R-squared:  0.7709, Adjusted R-squared:  0.7691
## F-statistic: 430.8 on 3 and 384 DF,  p-value: < 2.2e-16
```

BIC

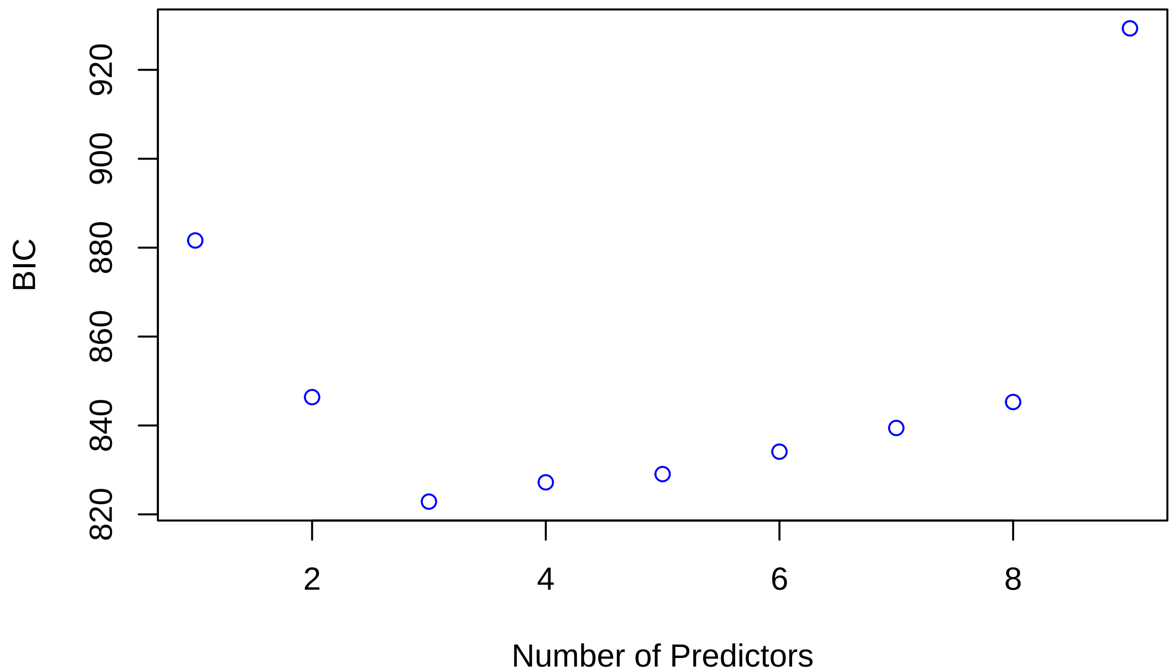
```r
# Perform subset selection using the 'regsubsets' function from the 'leaps' package
B <- regsubsets(points ~ position + statusId + rank + tot_pit_time + laps +
                  fastestlap_ms + q2_ms + q1_ms, data = F1_Subset)
rs <- summary(B)
# Display the predictors selected by each model size based on the 'regsubsets' output
rs$which
```

```
##   (Intercept) position statusId rank tot_pit_time  laps fastestlap_ms q2_ms
## 1        TRUE    FALSE    FALSE TRUE        FALSE FALSE         FALSE FALSE
## 2        TRUE    FALSE     TRUE TRUE        FALSE FALSE         FALSE FALSE
## 3        TRUE     TRUE     TRUE TRUE        FALSE FALSE         FALSE FALSE
## 4        TRUE     TRUE     TRUE TRUE        FALSE FALSE         FALSE  TRUE
## 5        TRUE     TRUE     TRUE TRUE        FALSE FALSE          TRUE  TRUE
## 6        TRUE     TRUE     TRUE TRUE        FALSE  TRUE          TRUE  TRUE
## 7        TRUE     TRUE     TRUE TRUE        FALSE  TRUE          TRUE  TRUE
## 8        TRUE     TRUE     TRUE TRUE         TRUE  TRUE          TRUE  TRUE
##   q1_ms
## 1 FALSE
## 2 FALSE
## 3 FALSE
## 4 FALSE
## 5 FALSE
## 6 FALSE
## 7  TRUE
## 8  TRUE
```

```r
# Calculate the Bayesian Information Criterion (BIC) for each model size
k <- nrow(F1_Subdf) # Number of observations
s <- 2:10 # Model sizes from 2 to 10 predictors
BIC <- k * log(rs$rss / k) + s * log(k)
BIC
```

```
## [1] 881.6194 846.3762 822.8725 827.2108 829.0654 834.1092 839.4384 845.2815
## [9] 929.3281
```

```r
# Plot BIC values
plot(BIC ~ I(s - 1), ylab= "BIC", xlab = "Number of Predictors", col = "blue")
```



The third model has the lowest BIC value of 579.71325. The predictors are statusId, position & rank. The Bayesian Information Criterion (BIC) helps determine the best combination of predictors for a model by balancing goodness of fit with model complexity. In this case, the BIC values were calculated for different model sizes, ranging from 2 to 10 predictors. The model with the lowest BIC value indicates the best trade-off between model fit and complexity. Based on this test we determined that the third model has the lowest BIC value of 822.8725, which includes predictors "statusId" "position" and "rank".

BIC Selected Model

```
# Fit a linear regression model using the predictors selected by BIC
BIC_model <- lm(points ~ statusId + position + rank, data = F1_Subset)
summary(BIC_model)
```

```
##
## Call:
## lm(formula = points ~ statusId + position + rank, data = F1_Subset)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16.2257  -2.6469  -0.6867   2.2717   9.7753
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 18.11218    0.40167  45.092  < 2e-16 ***
## statusId     0.03353    0.01268   2.644  0.00853 **
## position    -0.20145    0.04351  -4.630    5e-06 ***
## rank        -1.10796    0.04880 -22.704  < 2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.589 on 384 degrees of freedom
## Multiple R-squared:  0.7709, Adjusted R-squared:  0.7691
## F-statistic: 430.8 on 3 and 384 DF,  p-value: < 2.2e-16
```

AdjR2

```r
# Perform subset selection using the 'regsubsets' function from the 'leaps' package
B <- regsubsets(points ~ position + statusId + rank + tot_pit_time + laps +
                  fastestlap_ms + q2_ms + q1_ms, data = F1_Subset)
rs <- summary(B)
# Display the predictors selected by each model size based on the 'regsubsets' output
rs$which
```

```
##   (Intercept) position statusId rank tot_pit_time  laps fastestlap_ms q2_ms
## 1        TRUE    FALSE    FALSE TRUE        FALSE FALSE         FALSE FALSE
## 2        TRUE    FALSE     TRUE TRUE        FALSE FALSE         FALSE FALSE
## 3        TRUE     TRUE     TRUE TRUE        FALSE FALSE         FALSE FALSE
## 4        TRUE     TRUE     TRUE TRUE        FALSE FALSE         FALSE  TRUE
## 5        TRUE     TRUE     TRUE TRUE        FALSE FALSE          TRUE  TRUE
## 6        TRUE     TRUE     TRUE TRUE        FALSE  TRUE          TRUE  TRUE
## 7        TRUE     TRUE     TRUE TRUE        FALSE  TRUE          TRUE  TRUE
## 8        TRUE     TRUE     TRUE TRUE         TRUE  TRUE          TRUE  TRUE
##   q1_ms
## 1 FALSE
## 2 FALSE
## 3 FALSE
## 4 FALSE
## 5 FALSE
## 6 FALSE
## 7  TRUE
## 8  TRUE
```

```r
# Calculate Adjusted R-squared values for each model size
adjusted_r_squared <- rs$adjr2
adjusted_r_squared
```

```
## [1] 0.7974300 0.8171625 0.8299146 0.8300356 0.8312355 0.8310454 0.8307289
## [8] 0.8301854
```

The adjusted R-squared value is a measure of how well the predictors in a model explain the variation in the outcome variable, adjusted for the number of predictors in the model. A higher adjusted R-squared value indicates a better fit of the model to the data. Based on this analysis, the fifth model, which includes predictors "statusId," "rank," "fastestlap_ms", "q2_ms", and "position," has the highest adjusted R-squared value of 0.8312355 among all the models tested.

Adjusted R-squared Selected Model

```r
# Fit a linear regression model using the predictors selected by Adjusted R-squared
AdjR2_model <- lm(points ~ statusId + position + rank + fastestlap_ms +
                    q2_ms, data = F1_Subset)
summary(AdjR2_model)
```

```
##
## Call:
## lm(formula = points ~ statusId + position + rank + fastestlap_ms +
##     q2_ms, data = F1_Subset)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.1752 -2.2589 -0.8821  1.7843  8.7819
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    1.948e+01  1.586e+00  12.283  < 2e-16 ***
## statusId       1.012e-01  1.838e-02   5.503 8.29e-08 ***
## position      -3.039e-01  5.905e-02  -5.148 4.90e-07 ***
## rank          -1.342e+00  5.517e-02 -24.329  < 2e-16 ***
## fastestlap_ms -7.235e-05  4.144e-05  -1.746   0.0819 .
## q2_ms          8.301e-05  4.052e-05   2.049   0.0414 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.227 on 287 degrees of freedom
##   (95 observations deleted due to missingness)
## Multiple R-squared:  0.8341, Adjusted R-squared:  0.8312
## F-statistic: 288.6 on 5 and 287 DF,  p-value: < 2.2e-16
```

Step Function

```r
# We are using the F1 dataframe without NA values for running the step function, as it cannot handle NA
# The F1 dataframe includes all predictors (q1, q2, q3) without NA values.
# Create the full model using the F1 dataframe without NA values
new_full_model <- lm(points ~ position + statusId + rank + tot_pit_time + laps +
                       fastestlap_ms + q2_ms + q1_ms, data = F1_df)
# Choose the best model using the stepwise variable selection method
step(new_full_model)
```

```
## Start:  AIC=425.53
## points ~ position + statusId + rank + tot_pit_time + laps + fastestlap_ms +
##     q2_ms + q1_ms
##
##                 Df Sum of Sq    RSS    AIC
## - fastestlap_ms  1       0.1 1613.2 423.54
## - q1_ms          1       0.9 1614.1 423.64
## - q2_ms          1       1.2 1614.3 423.67
## - tot_pit_time   1       3.4 1616.6 423.94
## - laps           1       7.9 1621.1 424.47
## <none>                       1613.2 425.53
## - statusId       1     157.5 1770.6 441.32
## - position       1     387.0 2000.1 464.60
## - rank           1    4932.4 6545.6 691.05
##
## Step:  AIC=423.54
## points ~ position + statusId + rank + tot_pit_time + laps + q2_ms +
##     q1_ms
```

```
##
##                Df Sum of Sq    RSS    AIC
## - q1_ms         1       1.0 1614.2 421.65
## - q2_ms         1       1.7 1615.0 421.75
## - tot_pit_time  1       4.3 1617.5 422.04
## - laps          1       8.2 1621.4 422.51
## <none>                      1613.2 423.54
## - statusId      1     157.9 1771.1 439.38
## - position      1     398.7 2011.9 463.72
## - rank          1    4933.7 6546.9 689.08
##
## Step:  AIC=421.65
## points ~ position + statusId + rank + tot_pit_time + laps + q2_ms
##
##                Df Sum of Sq    RSS    AIC
## - tot_pit_time  1       3.8 1618.0 420.10
## - q2_ms         1       5.0 1619.2 420.24
## - laps          1       7.3 1621.5 420.52
## <none>                      1614.2 421.65
## - statusId      1     157.5 1771.7 437.44
## - position      1     400.2 2014.4 461.96
## - rank          1    4937.7 6551.9 687.23
##
## Step:  AIC=420.1
## points ~ position + statusId + rank + laps + q2_ms
##
##            Df Sum of Sq    RSS    AIC
## - q2_ms     1       4.1 1622.1 418.59
## - laps      1       8.2 1626.2 419.07
## <none>                  1618.0 420.10
## - statusId  1     155.9 1773.9 435.68
## - position  1     400.1 2018.1 460.31
## - rank      1    4954.5 6572.4 685.83
##
## Step:  AIC=418.59
## points ~ position + statusId + rank + laps
##
##            Df Sum of Sq    RSS    AIC
## - laps      1       4.3 1626.4 417.10
## <none>                  1622.1 418.59
## - statusId  1     154.3 1776.4 433.94
## - position  1     399.3 2021.4 458.62
## - rank      1    4958.7 6580.8 684.07
##
## Step:  AIC=417.1
## points ~ position + statusId + rank
##
##            Df Sum of Sq    RSS    AIC
## <none>                  1626.4 417.10
## - statusId  1     153.1 1779.5 432.28
## - position  1     406.3 2032.7 457.69
## - rank      1    4962.4 6588.8 682.30
```

```
##
```

```
## Call:
## lm(formula = points ~ position + statusId + rank, data = F1_df)
##
## Coefficients:
## (Intercept)     position     statusId         rank
##    22.53464     -0.63100      0.09831     -1.51438
```

The step function is a method for variable selection that iteratively adds or removes predictors from a model based on certain criteria, such as the Akaike Information Criterion (AIC). In this analysis, we applied the step function to select the best-fitting model from the full model. The output indicates that the seventh model, with an AIC value of 417.1, is the most optimal model among all the models tested. This model includes the predictors "statusId," "rank," and "position." The lowest AIC value suggests that this model provides the best balance between goodness of fit and model complexity compared to other models considered.

```r
#Summary statistics
step_selected_model <- lm(points ~ statusId + position + rank, data = F1_df)
summary(step_selected_model)
```

```
##
## Call:
## lm(formula = points ~ statusId + position + rank, data = F1_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.7703 -2.0515 -0.6441  1.3990  8.1594
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 22.53464    0.46767  48.185  < 2e-16 ***
## statusId     0.09831    0.02343   4.195 4.20e-05 ***
## position    -0.63100    0.09232  -6.835 1.13e-10 ***
## rank        -1.51438    0.06340 -23.886  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.949 on 187 degrees of freedom
## Multiple R-squared:  0.8652, Adjusted R-squared:  0.863
## F-statistic: 400.1 on 3 and 187 DF,  p-value: < 2.2e-16
```

Based on the results of the lowest AIC, lowest BIC, highest adjusted R-squared, and step function tests, we concluded that the most optimal model was the three predictor model including "statusId," "rank," and "position." These predictors were consistently selected across multiple model selection criteria, indicating their importance in explaining the variation in the outcome variable.

BOX-COX

```r
require(MASS)
# Preprocess data with +1 to handle zeros
F1_Subset$points_1 <- F1_Subset$points + 1
```

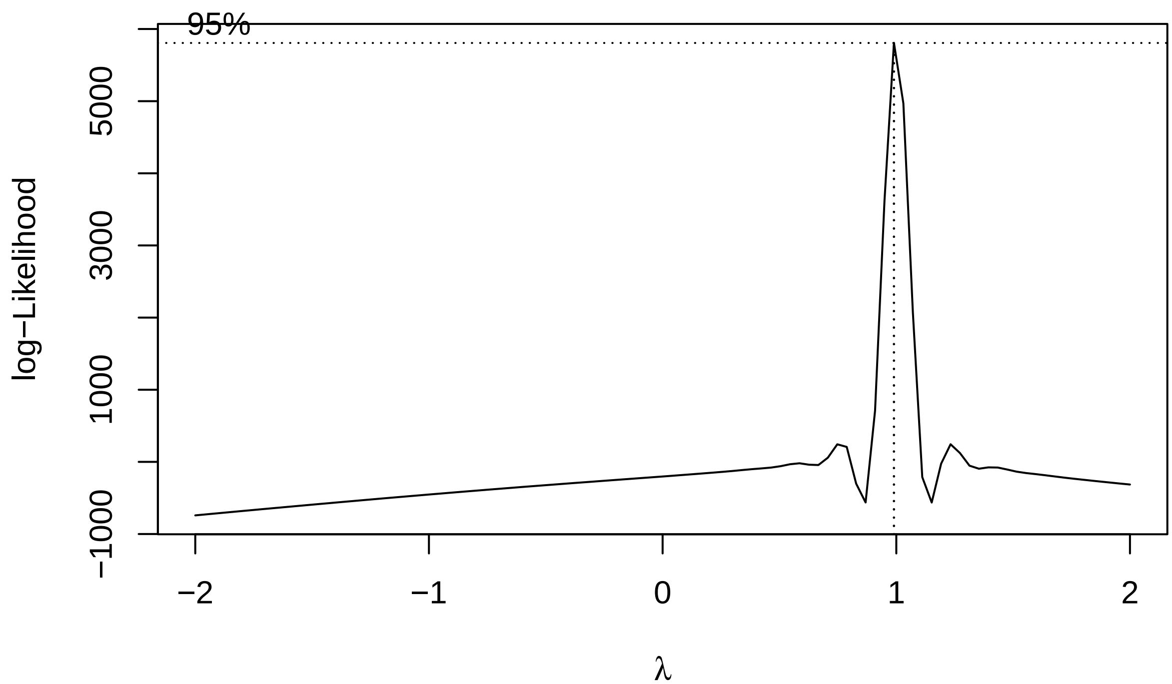
Squaring each predictor of the AIC model - statusId, rank, position
statusId

```r
#squaring statusId
lmod_statid <- (lm(points_1 ~ statusId + I(statusId^2) + position + rank +
                     fastestlap_ms + tot_pit_time + laps + q1_ms + q2_ms +q3_ms
                   + points, F1_Subset))
summary(lmod_statid)
```

```
##
## Call:
## lm(formula = points_1 ~ statusId + I(statusId^2) + position +
##     rank + fastestlap_ms + tot_pit_time + laps + q1_ms + q2_ms +
##     q3_ms + points, data = F1_Subset)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -2.073e-14 -1.493e-15 -7.100e-16  3.080e-16  9.407e-14
##
## Coefficients:
##                 Estimate Std. Error    t value Pr(>|t|)
## (Intercept)    1.000e+00  1.485e-14  6.733e+13  < 2e-16 ***
## statusId      -3.141e-16  3.230e-16 -9.720e-01  0.33214
## I(statusId^2)  2.030e-18  2.442e-18  8.310e-01  0.40700
## position       7.998e-16  2.672e-16  2.993e+00  0.00316 **
## rank          -6.434e-16  3.515e-16 -1.830e+00  0.06888 .
## fastestlap_ms  3.957e-19  2.626e-19  1.507e+00  0.13368
## tot_pit_time  -2.780e-22  7.843e-22 -3.550e-01  0.72338
## laps           8.603e-17  1.076e-16  7.990e-01  0.42522
## q1_ms          6.063e-19  6.466e-19  9.380e-01  0.34965
## q2_ms         -9.410e-19  9.393e-19 -1.002e+00  0.31780
## q3_ms         -7.627e-21  2.577e-19 -3.000e-02  0.97642
## points         1.000e+00  1.912e-16  5.231e+15  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.498e-15 on 178 degrees of freedom
##   (198 observations deleted due to missingness)
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 1.949e+31 on 11 and 178 DF,  p-value: < 2.2e-16
```

finding lambda - statusId

```r
require(MASS)
boxcox_results_statid <- boxcox(lmod_statid, plotit = TRUE)
```




```r
lamda_statid <- boxcox_results_statid$x[which.max(boxcox_results_statid$y)]
print(lamda_statid)
```

```
## [1] 0.989899
```

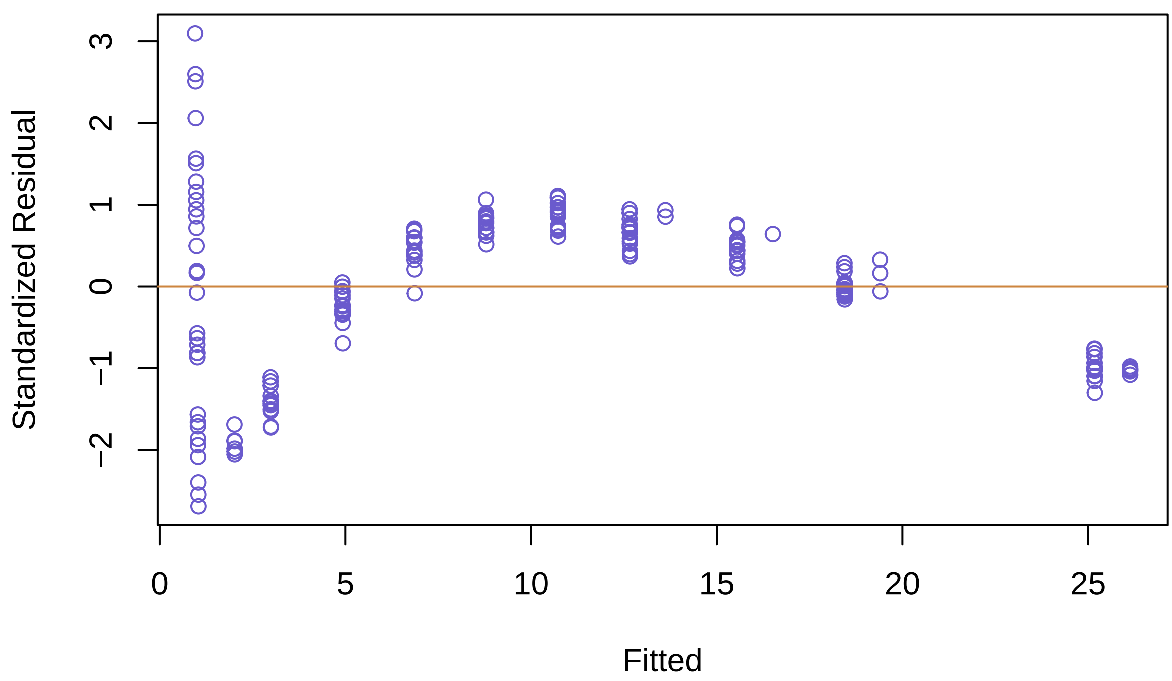
Transformed Model: statusId(squared)

```r
require(MASS)
# add code to transform model based on lambda
trans_val_statid <- (F1_Subset$points_1)^(lamda_statid)
# Fit a new linear model with the transformed variable
lmodTrans_statid <- lm(trans_val_statid ~ ., data = F1_Subset)
summary(lmodTrans_statid)
```

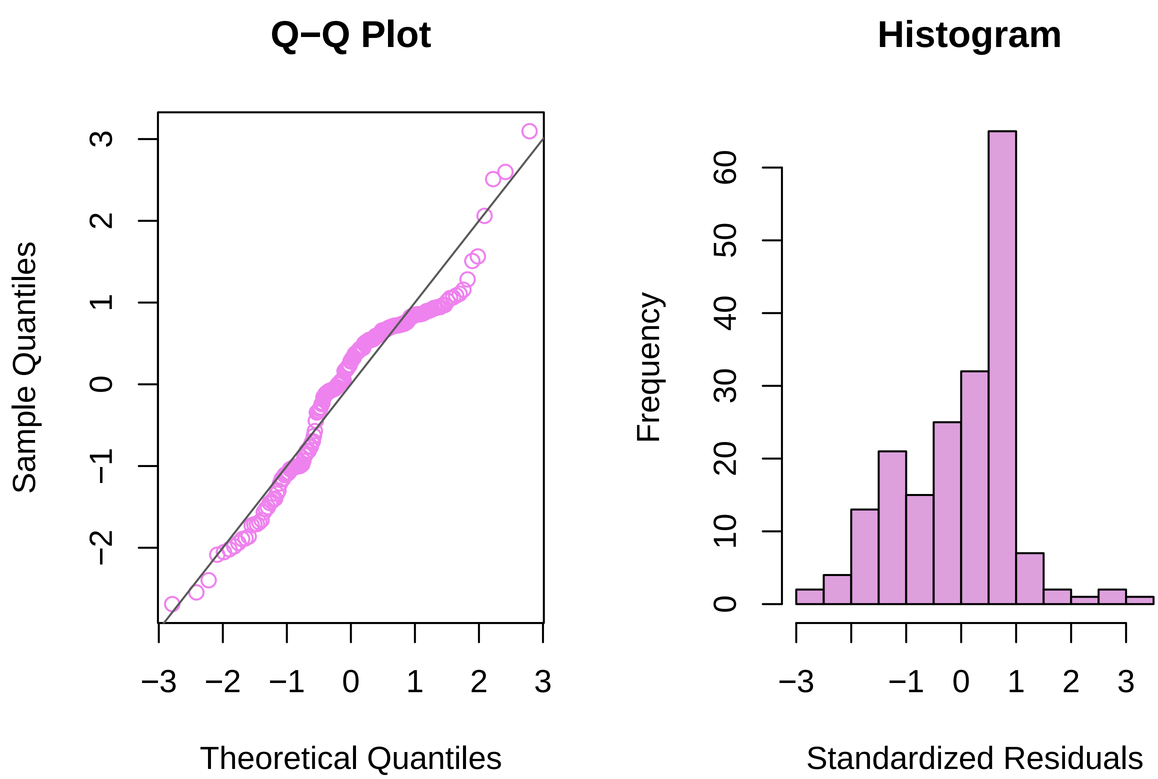
```
##
## Call:
## lm(formula = trans_val_statid ~ ., data = F1_Subset)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.042966 -0.012629  0.004493  0.011240  0.047682
##
## Coefficients: (1 not defined because of singularities)
##                  Estimate Std. Error  t value Pr(>|t|)
```

```
## (Intercept)     1.177e+00  3.200e-02   36.769  < 2e-16 ***
## statusId        4.961e-04  1.356e-04    3.659 0.000333 ***
## position       -8.900e-04  5.783e-04   -1.539 0.125543
## rank           -1.468e-02  7.068e-04  -20.776  < 2e-16 ***
## fastestlap_ms   3.459e-09  5.604e-07    0.006 0.995082
## tot_pit_time    5.951e-10  1.693e-09    0.352 0.725620
## laps            2.494e-04  2.308e-04    1.081 0.281315
## points          9.592e-01  4.048e-04 2369.370  < 2e-16 ***
## q1_ms          -1.212e-07  1.380e-06   -0.088 0.930067
## q2_ms           5.005e-07  2.001e-06    0.250 0.802753
## q3_ms          -1.768e-07  5.576e-07   -0.317 0.751510
## points_1               NA         NA       NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01623 on 179 degrees of freedom
##   (198 observations deleted due to missingness)
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 4.272e+06 on 10 and 179 DF,  p-value: < 2.2e-16
```

```r
plot(fitted(lmodTrans_statid), rstandard(lmodTrans_statid), xlab = "Fitted",
     ylab = "Standardized Residual" , col="slateblue")
abline(h = 0, col = "peru")
```



```
# Set up layout for two panels side by side
par(mfrow = c(1, 2))
# Left panel: QQ plot of standardized residuals
qqnorm(rstandard(lmodTrans_statid), main = "Q-Q Plot" , col = "violet")
abline(0, 1, col = "grey35")
# Right panel: Histogram of standardized residuals
hist(rstandard(lmodTrans_statid), main = "Histogram ",
     xlab = "Standardized Residuals" ,col = "plum")
```




```
par(mfrow = c(1, 1))
```

We performed this transformation on the model for each predictor and got the same output. The lambda value was very close to 1 at 0.989899. Despite using a lambda value to transform the model we were unable to make the model fit better.

Prediction

```
# Fetch the last row of the F1 dataframe to get an example of data for prediction
noOfRows <- nrow(F1_df)
F1_df[noOfRows,]
```

```
##     statusId position rank fastestlap_ms tot_pit_time laps points q1_ms q2_ms
## 388        1        3    6         88138        42853   58      8 84487 84278
##     q3_ms
## 388 83782
```

```r
# Create a new dataframe with predictor variables for which predictions will be made
newdata <- data.frame(position = 3, rank = 6, statusId = 1)
# Use the function to generate a prediction interval using the AIC_model
predict_interval <- predict(AIC_model, newdata, interval = "predict")
# Print the prediction interval
print("Prediction Interval:")
```

```
## [1] "Prediction Interval:"
```

```r
predict_interval
```

```
##       fit      lwr      upr
## 1 10.8936 3.812087 17.97511
```

```r
# Use the function again to generate a confidence interval using the AIC_model
confidence_interval <- predict(AIC_model, newdata, interval = "confidence")
# Print the confidence interval
print("Confidence Interval:")
```

```
## [1] "Confidence Interval:"
```

```r
confidence_interval
```

```
##       fit      lwr      upr
## 1 10.8936 10.30123 11.48597
```

The prediction interval [3.80, 17.96], signifies a 95% probability that a future observation of points will be contained in the interval, given the values of position = 3, rank = 6, and statusId = 1. At the same time, there is a 5% probability that the next observation of points will not land between the interval, given these predictor values. Additionally, the confidence interval for the average points of the driver [10.29, 11.47], indicates that we are 95% confident that the average points for this driver across multiple races will lie within this interval. This interval accounts for the uncertainty associated with estimating the average points based on the available data. Therefore, it serves as a measure of the precision of our estimate of the driver's average performance, allowing us to assess the range within which the true average points are likely to be situated.