STAT3215 Final Project: Analysis of Aircraft Flight Delay

Charitarth Chugh (charitarth@uconn.edu (mailto:charitarth@uconn.edu)), Leon Nguyen (leon.nguyen@uconn.edu (mailto:leon.nguyen@uconn.edu))

12/08/23

```
knitr::opts_chunk$set(echo = TRUE, warning = FALSE, cache = TRUE)
library(plyr)
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.4     ✔ readr     2.1.4
## ✔ forcats   1.0.0     ✔ stringr   1.5.1
## ✔ ggplot2   3.4.4     ✔ tibble    3.2.1
## ✔ lubridate 1.9.3     ✔ tidyr     1.3.0
## ✔ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::arrange()   masks plyr::arrange()
## ✖ purrr::compact()   masks plyr::compact()
## ✖ dplyr::count()     masks plyr::count()
## ✖ dplyr::desc()      masks plyr::desc()
## ✖ dplyr::failwith()  masks plyr::failwith()
## ✖ dplyr::filter()    masks stats::filter()
## ✖ dplyr::id()        masks plyr::id()
## ✖ dplyr::lag()       masks stats::lag()
## ✖ dplyr::mutate()    masks plyr::mutate()
## ✖ dplyr::rename()    masks plyr::rename()
## ✖ dplyr::summarise() masks plyr::summarise()
## ✖ dplyr::summarize() masks plyr::summarize()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become
errors
```

```
library(psych)
```

```
##
## Attaching package: 'psych'
##
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
library(car)
```

```
## Loading required package: carData
##
## Attaching package: 'car'
##
## The following object is masked from 'package:psych':
##
##     logit
##
## The following object is masked from 'package:dplyr':
##
##     recode
##
## The following object is masked from 'package:purrr':
##
##     some
```

```
library(alr4)
```

```
## Loading required package: effects
## lattice theme set by effectsTheme()
## See ?effectsTheme for details.
```

```
library(glmnet)
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
## Loaded glmnet 4.1-8
```

```
library(ncvreg)
```

```
##
## Attaching package: 'ncvreg'
##
## The following object is masked from 'package:psych':
##
##     AUC
```

```
library(leaps)

set.seed(3215)
```

Section 1. Introduction

In this report, we will reproduce, analyze, and enhance findings from a research paper written by Kalliguddi and Leboulleuc, which seeks to investigate factors contributing to aircraft flight delays within the airline industry. The paper aims to analyze the on-time performance of domestic flights for the year 2016 and develop a better predictive model to forecast flight delay. This is important because flight delays can incur significant economic losses, damage reputability for airlines, and create inconvenience for passengers.

The remainder of the report is divided into four sections as follows. Section 2 discusses the data description and data source. Section 3 describes the data analysis and predictive modeling techniques used in the study, delving into multiple linear regression, assumption validation, and other miscellaneous techniques. Section 4 describes our additional subanalyses and enhancement to the initial paper. Finally, section 5 provides conclusions and further discussion of the initial study and our enhancement.

Section 2. Description of Data

The data used in both the initial study and this report was obtained from the Bureau of Transportation Statistics (BTS) and analyzed the domestic flight activity from January 2016 to December 2016. BTS provides detailed data for individual flights with more than 23 variables. For the purposes of this research analysis, the study chose quantitative variables that were deemed relevant to measuring flight delay: Departure delay, Taxi in, Taxi out, Carrier delay, Security delay, Weather delay, Late aircraft delay, Distance, and National air system delay.

The study design involved analyzing the on-time performance of domestic flights for the year 2016 and developing a better predictive model to forecast flight delay. The methods of data collection involved obtaining historical data from the BTS and cleaning and processing the data. Note that all missing values were imputed in the raw set; imputations using methods such as additive regression, bootstrapping, and predictive mean matching were done using the “Hmisc” package in R studio. For cross validation, the data was then divided into two parts, the first being the training data and the second being the test/holdout data. Because the authors of the paper observed and collected data on various factors that may contribute to flight delay as opposed to manipulating any variables or treatments, the initial study would be classified an observational study.

To attempt to replicate the data included in this paper, one would need to obtain the same data set from the Bureau of Transportation Statistics and follow the same data cleaning and processing steps described in the paper. The study used R studio and the “Hmisc” package for data imputation, so these tools would also be necessary to replicate the study’s results. However, because the specific procedure of how the dataset was cleaned and how the “Hmisc” package was utilized to impute values were not described in detail, we were limited in reproducibility. After noting this ambiguity and the large size of the raw dataset, for simplicity, we were advised to remove observations which contained any null values to work with a dataset of a manageable size instead of using imputation. We also noted that the Department of Transportation considers a flight to be delayed if it arrives or departs 15 minutes or more later than its scheduled time. In the raw dataset, if a given observation has a `DepDelay` value below 15 minutes, the relevant predictor variables were left as null values.

To summarize, With the limited procedure information we were provided, we downloaded files from the BTS containing domestic flight information from each month of the year 2016, merged them, then cleaned the dataset for observations with null values and any flights that are not considered “delayed” in accordance with the 15 minute tolerance.

```
# Getting the compressed data, reading it, and filtering for missing values.
# its in its own function as a RAM saving measure.
load_data <- function() {
  data_dir <- "./Chugh-Charitarth-Data"
  df_list <- list()
  for (i in list.files(data_dir)) {
    if (i != "readme.html") {
      # Using show_col_types=FALSE to mute extra output
      d <- readr::read_csv(file.path(data_dir, i), show_col_types = FALSE)
      d <- d %>% filter(DepDelay > 15)
      df_list[[i]] <- d
    }
  }
  df <- plyr::ldply(df_list, rbind) # merge all the data frames into one.
  df <- df %>% select(
    DepDelay, CarrierDelay, TaxiIn, TaxiOut,
    Distance, WeatherDelay, NASDelay,
    SecurityDelay, LateAircraftDelay
  )
  df <- filter(df, !dplyr::if_any(dplyr::everything(), is.na))
}
df <- load_data()
```

```
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## • `` -> `...110`
```

```
summary(df)
```

```
##     DepDelay       CarrierDelay         TaxiIn          TaxiOut
##  Min.   :  16.0   Min.   :   0.00   Min.   :  1.000   Min.   :  1.00
##  1st Qu.:  31.0   1st Qu.:   0.00   1st Qu.:  4.000   1st Qu.: 11.00
##  Median :  50.0   Median :   5.00   Median :  6.000   Median : 15.00
##  Mean   :  74.8   Mean   :  25.61   Mean   :  7.785   Mean   : 18.32
##  3rd Qu.:  89.0   3rd Qu.:  25.00   3rd Qu.:  9.000   3rd Qu.: 21.00
##  Max.   :2149.0   Max.   :2142.00   Max.   :250.000   Max.   :186.00
##     Distance       WeatherDelay         NASDelay      SecurityDelay
##  Min.   :  31.0   Min.   :   0.000   Min.   :   0.0   Min.   :  0.0000
##  1st Qu.: 395.0   1st Qu.:   0.000   1st Qu.:   0.0   1st Qu.:  0.0000
##  Median : 692.0   Median :   0.000   Median :   0.0   Median :  0.0000
##  Mean   : 853.8   Mean   :   3.426   Mean   :  12.3   Mean   :  0.1022
##  3rd Qu.:1090.0   3rd Qu.:   0.000   3rd Qu.:   8.0   3rd Qu.:  0.0000
##  Max.   :4983.0   Max.   :1157.000   Max.   :1446.0   Max.   :474.0000
##  LateAircraftDelay
##  Min.   :   0.00
##  1st Qu.:   0.00
##  Median :  15.00
##  Mean   :  31.24
##  3rd Qu.:  40.00
##  Max.   :1484.00
```

```
head(df)
```

	DepDelay <dbl>	CarrierDelay <dbl>	TaxiIn <dbl>	TaxiOut <dbl>	Distance <dbl>	WeatherDelay <dbl>	NASDelay <dbl>	SecurityDelay <dbl>
1	100	0	14	20	986	0	47	0
2	34	0	10	12	986	21	0	0
3	64	63	11	18	986	0	0	0
4	26	26	6	42	986	0	6	0
5	120	0	8	34	986	12	0	0
6	23	7	8	34	986	0	16	0

6 rows | 1-9 of 10 columns

```
nrow(df) # number of filtered observations
```

```
## [1] 741613
```

Section 3. Original Methods, Models, and Analysis

Before beginning any preliminary analysis, we would need to split our clean, processed data into two sets: a training set would be used to construct and validate assumptions about our model, and a test/holdout set would be used to measure the predictive performance of the model. The original paper mentions splitting the data, but does not specify what proportion of data belongs to either set, nor how the sets are sampled. We will assume random sampling was used to generate the sets, and that 70% of the dataset was used for training and 30% for testing. The training set will be used for preliminary analysis.

```
# splitting the dataset
df_train <- dplyr::sample_frac(df, 0.70)
df_test <- dplyr::anti_join(df, df_train)
```

```
## Joining with `by = join_by(DepDelay, CarrierDelay, TaxiIn, TaxiOut, Distance,
## WeatherDelay, NASDelay, SecurityDelay, LateAircraftDelay)`
```

```
df <- NULL # Save RAM
gc()
```

```
##            used  (Mb) gc trigger   (Mb)  max used   (Mb)
## Ncells  2501271 133.6    4620794  246.8   4620794  246.8
## Vcells 11978826  91.4  276852543 2112.3 346056452 2640.3
```

3.1 Preliminary Analysis

```
pairs.panels(df_train,
  ellipses = FALSE, hist.col = "cyan",
  method = "pearson", density = TRUE
)
```

To begin our preliminary analysis, we will begin by creating a predictor plot with Pearson correlation coefficients combined with a scatterplot matrix. This diagram will give us a visual understanding of pairwise relationships between all variables in the study based on our training data. The Pearson correlation coefficient (denoted as r) is used to measure the strength of a linear association between two variables. This value can range from -1 to 1, where magnitude indicates the strength and the signage indicates directionality. For example, in the scatterplot matrix above we can find the r value based on where the row of one variable and the column of the other variable intersect. Based on our training dataset, the variables `TaxiIn` and `NASDelay` have a Pearson's constant of 0.24, indicating a weak positive linear correlation. Utilizing Pearson's constant is a good way of assessing whether there is the issue of multicollinearity, where regressors are highly correlated or are dependent with each other, indicated by large values of r. Multicollinearity can inflate the variance and result in potentially misleading conclusions regarding contributions of the regressors. The original paper uses the value of 0.5 as the threshold for indicating collinearity; the authors observe that since no value of r between any two variables are larger than 0.5, it is assumed that all the variables are independent. The r values of our training data set tend to generally agree with those found in the original paper; the correlation coefficients between pairs of regressors (all variables excluding `DepDelay`, the response variable) are generally weak, so multicollinearity is not likely to be an issue in our analysis.

Scatterplots are utilized to plot two variables against each other to analyze any patterns and see if there is anything that may suggest a relationship between the two. In the scatterplot matrix above, we can analyze the scatterplot of the variables `TaxiIn` and `NASDelay` with a Pearson's constant of 0.24. We can visually observe in the associated scatterplot that the observations tend to be clustered randomly (although observations taper off towards larger values of y) and thus will have a weak linear relationship, which agrees with the Pearson's constant. Randomness in the scatterplot indicates no relationship, while linear patterns may imply a linear relationship. The strength of the linear relationship is indicated by the clarity of the linear pattern; for example, a perfect straight line of observations with no deviations would indicate a perfect linear correlation between two variables. In the original paper, scatterplots comparing regressors (variables not including the response) do not appear to have a distinguishable pattern that would imply a linear relationship, although it is worth noting that most observations tend to adhere close to the axes. When we look at scatterplots with respect to the response variable, we can see linear patterns that emerge from those associated with variables `CarrierDelay`, `WeatherDelay`, `NASDelay`, and `LateAircraftDelay`. Visually, our training data generally agrees with the findings from the original paper with the aforementioned features.

3.2 Multiple Regression Model

An MLR (Multiple Linear Regression) model predicts a quantitative response Y (in this case, the length of Departure Delay measured by `DepDelay`) based on two or more predictor variables, either categorical or quantitative. We can only construct and draw inferences from this model with the following assumptions:

- There exists some linear relationship between each regressors and Y.
- The variance of the errors (difference between predicted and actual response values) should be consistent across all levels of the predictors. (This is the assumption of homoscedasticity.)
- The expected value of the errors should be zero.
- Observations should be independent of each other.
- The errors should follow a normal distribution.

We can express this relationship as a mathematical equation:

$$y = \beta_0 + m \sum i=1 \beta_i x_i + \epsilon$$

In the original report and in this paper, there are no categorical variables to analyze so no numeric encoding is necessary. When regression is run with all eight quantitative regressors, we observed that all of the regressors were significant with an R-squared value of 0.9812, which is somewhat similar to the result from the original paper (with an R-squared value of 0.84). This indicates that the initial main MLR model can explain 98.12% of the variation in the data. Note that since we do not know the exact training dataset used, reproducibility is limited, so we accept approximate results when attempting to replicate the procedure.

```
# run MLR with all 8 regressors
m1 <- lm(DepDelay ~ ., data = df_train)
summary(m1)
```

```
##
## Call:
## lm(formula = DepDelay ~ ., data = df_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -405.14   -6.20   -0.11    6.33  106.75
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        1.941e+01  3.886e-02   499.4   <2e-16 ***
## CarrierDelay       1.007e+00  2.358e-04  4271.4   <2e-16 ***
## TaxiIn            -7.858e-01  2.246e-03  -349.8   <2e-16 ***
## TaxiOut           -7.577e-01  1.330e-03  -569.5   <2e-16 ***
## Distance           3.634e-03  2.484e-05   146.3   <2e-16 ***
## WeatherDelay       9.911e-01  6.193e-04  1600.4   <2e-16 ***
## NASDelay           9.252e-01  4.881e-04  1895.6   <2e-16 ***
## SecurityDelay      9.924e-01  5.405e-03   183.6   <2e-16 ***
## LateAircraftDelay  1.012e+00  3.049e-04  3319.5   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.81 on 519120 degrees of freedom
## Multiple R-squared:  0.9818, Adjusted R-squared:  0.9818
## F-statistic: 3.501e+06 on 8 and 519120 DF,  p-value: < 2.2e-16
```

We can apply a stepwise regression (in both directions) to determine if results will agree with the initial main MLR model.Stepwise regression is an iterative process of adding or removing regressors to find the best fit model based on a specified metric (in this case, we want to find the minimum AIC value). Note that the initial direction of the stepwise regression performed in the initial paper was ambiguous, so both are conducted in the below code.

```
step(m1, direction = "both")
```

```
## Start:  AIC=2471911
## DepDelay ~ CarrierDelay + TaxiIn + TaxiOut + Distance + WeatherDelay +
##     NASDelay + SecurityDelay + LateAircraftDelay
##
##                     Df  Sum of Sq        RSS     AIC
## <none>                               60704173 2471911
## - Distance           1    2503816   63207989 2492891
## - SecurityDelay      1    3942221   64646394 2504572
## - TaxiIn             1   14309720   75013894 2581788
## - TaxiOut            1   37921521   98625694 2723852
## - WeatherDelay       1  299490087  360194261 3396286
## - NASDelay           1  420179602  480883775 3546305
## - LateAircraftDelay  1 1288532836 1349237009 4081874
## - CarrierDelay       1 2133494305 2194198479 4334315
```

```
##
## Call:
## lm(formula = DepDelay ~ CarrierDelay + TaxiIn + TaxiOut + Distance +
##     WeatherDelay + NASDelay + SecurityDelay + LateAircraftDelay,
##     data = df_train)
##
## Coefficients:
##       (Intercept)       CarrierDelay             TaxiIn            TaxiOut
##         19.406632            1.007233           -0.785840           -0.757661
##          Distance       WeatherDelay           NASDelay      SecurityDelay
##          0.003634            0.991098           0.925247            0.992417
## LateAircraftDelay
##          1.012201
```

```
step(lm(DepDelay ~ 1, data = df_train),
  scope = list(upper = m1), direction = "both"
)
```

```
## Start:  AIC=4551826
## DepDelay ~ 1
##
##                     Df  Sum of Sq        RSS     AIC
## + CarrierDelay       1 1520305362 1815906884 4236067
## + LateAircraftDelay  1  713434472 2622777775 4426924
## + WeatherDelay       1  165673367 3170538879 4525386
## + NASDelay           1  118877528 3217334718 4532992
## + TaxiIn             1    3714392 3332497854 4551250
## + Distance           1    1241258 3334970989 4551635
## + TaxiOut            1     658948 3335553298 4551725
## + SecurityDelay      1     376266 3335835981 4551769
## <none>                            3336212246 4551826
##
## Step:  AIC=4236067
## DepDelay ~ CarrierDelay
##
##                     Df  Sum of Sq        RSS     AIC
## + LateAircraftDelay  1 1059026297  756880587 3781761
## + WeatherDelay       1  220883457 1595023427 4168740
## + NASDelay           1  208185641 1607721243 4172856
## + TaxiIn             1    3655786 1812251098 4235023
## + SecurityDelay      1    1312229 1814594654 4235694
## + TaxiOut            1     256176 1815650708 4235996
## + Distance           1      27795 1815879089 4236061
## <none>                            1815906884 4236067
## - CarrierDelay       1 1520305362 3336212246 4551826
##
## Step:  AIC=3781761
## DepDelay ~ CarrierDelay + LateAircraftDelay
##
##                     Df  Sum of Sq        RSS     AIC
## + NASDelay           1  364356332  392524255 3440899
## + WeatherDelay       1  270192173  486688415 3552526
## + SecurityDelay      1    2675765  754204823 3779925
## + TaxiIn             1    2503017  754377570 3780044
## + TaxiOut            1    2383555  754497033 3780126
## + Distance           1    1201610  755678978 3780938
## <none>                            756880587 3781761
## - LateAircraftDelay  1 1059026297 1815906884 4236067
## - CarrierDelay       1 1865897187 2622777775 4426924
##
## Step:  AIC=3440899
## DepDelay ~ CarrierDelay + LateAircraftDelay + NASDelay
##
##                     Df  Sum of Sq        RSS     AIC
## + WeatherDelay       1  276692049  115832206 2807328
## + TaxiOut            1   16101487  376422769 3419158
## + TaxiIn             1    9844263  382679992 3427716
## + SecurityDelay      1    3498369  389025886 3436254
## + Distance           1     770401  391753854 3439882
## <none>                            392524255 3440899
## - NASDelay           1  364356332  756880587 3781761
## - LateAircraftDelay  1 1215196988 1607721243 4172856
## - CarrierDelay       1 2025750299 2418274555 4384784
##
```

```
## Step:  AIC=2807328
## DepDelay ~ CarrierDelay + LateAircraftDelay + NASDelay + WeatherDelay
##
##                     Df  Sum of Sq        RSS     AIC
## + TaxiOut            1   35319141   80513065 2618510
## + TaxiIn             1   11871431  103960775 2751197
## + SecurityDelay      1    3907934  111924272 2789513
## + Distance           1     777944  115054262 2803831
## <none>                             115832206 2807328
## - WeatherDelay       1  276692049  392524255 3440899
## - NASDelay           1  370856208  486688415 3552526
## - LateAircraftDelay  1 1269473360 1385305566 4095561
## - CarrierDelay       1 2106086876 2221919082 4340824
##
## Step:  AIC=2618510
## DepDelay ~ CarrierDelay + LateAircraftDelay + NASDelay + WeatherDelay +
##     TaxiOut
##
##                     Df  Sum of Sq        RSS     AIC
## + TaxiIn             1   13269806   67243259 2525016
## + SecurityDelay      1    4022780   76490285 2591904
## + Distance           1    1548088   78964977 2608433
## <none>                             80513065 2618510
## - TaxiOut            1   35319141  115832206 2807328
## - WeatherDelay       1  295909704  376422769 3419158
## - NASDelay           1  406134214  486647279 3552484
## - LateAircraftDelay  1 1274861115 1355374180 4084224
## - CarrierDelay       1 2125320635 2205833700 4337054
##
## Step:  AIC=2525016
## DepDelay ~ CarrierDelay + LateAircraftDelay + NASDelay + WeatherDelay +
##     TaxiOut + TaxiIn
##
##                     Df  Sum of Sq        RSS     AIC
## + SecurityDelay      1    4035269   63207989 2492891
## + Distance           1    2596865   64646394 2504572
## <none>                             67243259 2525016
## - TaxiIn             1   13269806   80513065 2618510
## - TaxiOut            1   36717517  103960775 2751197
## - WeatherDelay       1  298541200  365784459 3404277
## - NASDelay           1  417615723  484858981 3550575
## - LateAircraftDelay  1 1284143472 1351386731 4082697
## - CarrierDelay       1 2133755120 2200998378 4335917
##
## Step:  AIC=2492891
## DepDelay ~ CarrierDelay + LateAircraftDelay + NASDelay + WeatherDelay +
##     TaxiOut + TaxiIn + SecurityDelay
##
##                     Df  Sum of Sq        RSS     AIC
## + Distance           1    2503816   60704173 2471911
## <none>                             63207989 2492891
## - SecurityDelay      1    4035269   67243259 2525016
## - TaxiIn             1   13282295   76490285 2591904
## - TaxiOut            1   36835390  100043380 2731259
## - WeatherDelay       1  299007756  362215745 3399189
## - NASDelay           1  418615431  481823420 3547316
```

```
## - LateAircraftDelay  1 1286219010 1349427000 4081945
## - CarrierDelay       1 2136341388 2199549378 4335577
##
## Step:  AIC=2471911
## DepDelay ~ CarrierDelay + LateAircraftDelay + NASDelay + WeatherDelay +
##     TaxiOut + TaxiIn + SecurityDelay + Distance
##
##                     Df  Sum of Sq        RSS     AIC
## <none>                              60704173 2471911
## - Distance           1    2503816   63207989 2492891
## - SecurityDelay      1    3942221   64646394 2504572
## - TaxiIn             1   14309720   75013894 2581788
## - TaxiOut            1   37921521   98625694 2723852
## - WeatherDelay       1  299490087  360194261 3396286
## - NASDelay           1  420179602  480883775 3546305
## - LateAircraftDelay  1 1288532836 1349237009 4081874
## - CarrierDelay       1 2133494305 2194198479 4334315
```

```
##
## Call:
## lm(formula = DepDelay ~ CarrierDelay + LateAircraftDelay + NASDelay +
##     WeatherDelay + TaxiOut + TaxiIn + SecurityDelay + Distance,
##     data = df_train)
##
## Coefficients:
##       (Intercept)       CarrierDelay  LateAircraftDelay           NASDelay
##         19.406632            1.007233            1.012201            0.925247
##      WeatherDelay             TaxiOut              TaxiIn      SecurityDelay
##          0.991098           -0.757661           -0.785840            0.992417
##          Distance
##          0.003634
```

From both the forward and backward stepwise regression, we end up with the full model with all eight regressors as expected from the original report. We can write out the multiple linear equation generated from this process, with coefficients rounded to four significant digits like the original report: {DOUBLE CHECK COEFFICIENTS BEFORE SUBMITTING}

Dep Delay = 19.40 + (1.007) CarrierDelay + (1.012) Late Aircraft Delay + (0.9236) NAS Delay + (0.9918) Weather Delay - (0.7595) Taxi Out - (0.7827) TaxiIn + (0.9821) SecurityDelay + (3.631)(10^{-3}) Distance

The coefficients of this MLR model are approximately within 0.2 units of those generated in the original paper. Based on this, we can infer some information about the relationship between the regressors and the response variable. The intercept or β_0 value is predicted to be 19.40 minutes, which means that if all of the regressors are set equal to zero, we predict that on average, we will have a substantial flight delay of 19.40 minutes. The Federal Aviation Administration (FAA) has a standard of 15 minutes as a threshold for a flight to be considered “delayed”. There may be other factors which contribute to departure delay which we are not considering.

The idea of a Root mean squared error (RMSE) is also discussed in the original paper. This is calculated by subtracting the observed predictor values from the predicted values, finding the mean of all these values squared (the mean of the squared errors), then taking the square root. For a perfect linear model, we expect an RMSE of 0. We can calculate the RMSE of our model as below:

```
RMSE <- sqrt(mean(m1$residuals^2))
RMSE
```

```
## [1] 10.81363
```

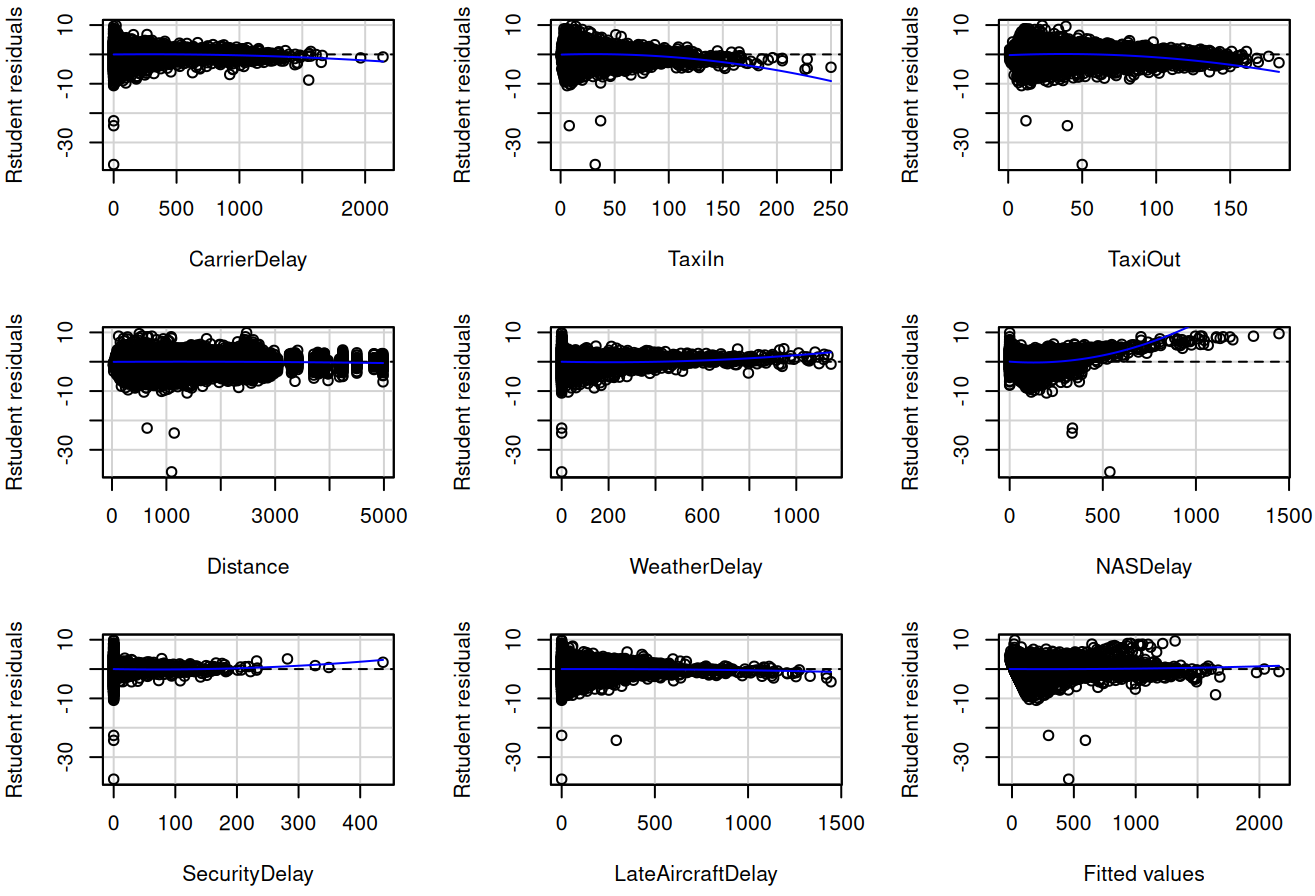
Our RMSE of 10.8 is relatively smaller compared to that of the original report (RMSE = 21.2). This metric serves just to give an idea of how much the errors vary overall in our model based on our training data.

3.2.1 Residual Analysis

Residual plots can be used to check our regression assumptions, specifically if the assumption of linearity is appropriate, if the errors have constant variance, and if we missed important variables that should be included in the model. In an ideal scenario where these assumptions are satisfied, we are expecting to see no distinguishable pattern that emerges from the scatterplots between the residuals and each variable. However, we see a distinct megaphone shape emerge from most of the variables, where there is distinctly more variation with lower values; we will formally test for non-constant variance (heteroscedasticity) in our enhancement. Some of these megaphone shapes seem to curl upward, such as the residual plot for `NASDelay` ; this may indicate some form of non-linearity. Similar to the plots associated with the regressors, a pointed megaphone shape along the x-axis with residuals in the negative emerged from the residual plot for the fitted values. The original paper only provided the residual plot for the fitted values (using direct residuals as opposed to standardized or studentized) but a similar pattern seems to emerge.

In our report, we also included tests for curvature and non-additivity based on the null hypothesis that the regressors have a linear relationship with the response variable. Based on a significance level of 0.05, all regressors except `SecurityDelay` and `LateAircraftDelay` reject the null hypothesis, indicating that there could be some non-linear component in their respective relationships with `DepDelay` ; this needs to be corrected to be used in an MLR model and will be explored further in our enhancements. In the next section, we discuss a remedy to address both the problem of non-linearity and heteroscedasticity.

```
residualPlots(m1, type = "rstudent")
```

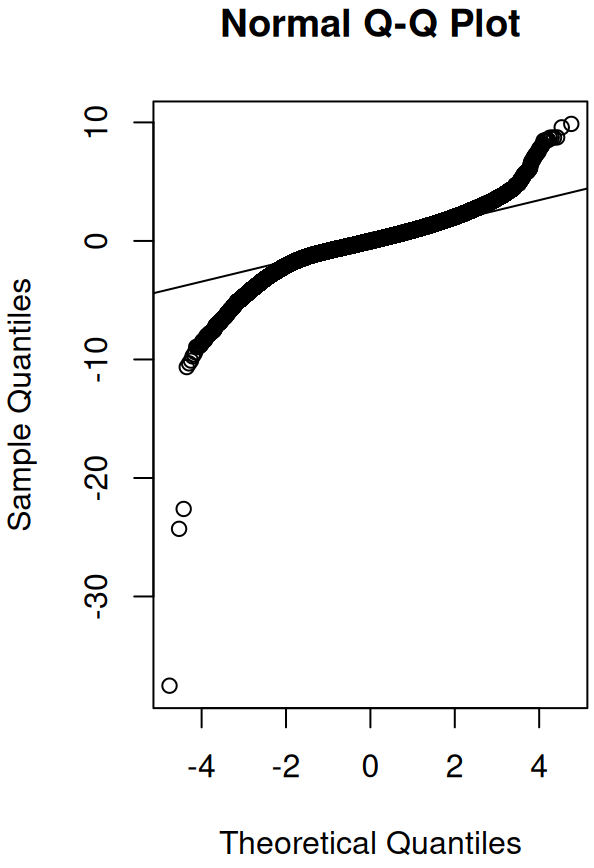


```
##                   Test stat Pr(>|Test stat|)
## CarrierDelay       -13.7715        < 2.2e-16 ***
## TaxiIn             -41.5727        < 2.2e-16 ***
## TaxiOut            -87.0503        < 2.2e-16 ***
## Distance           -12.7803        < 2.2e-16 ***
## WeatherDelay        20.6372        < 2.2e-16 ***
## NASDelay           112.8777        < 2.2e-16 ***
## SecurityDelay        5.0803        3.770e-07 ***
## LateAircraftDelay   -5.0110        5.418e-07 ***
## Tukey test           7.8455        4.314e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3.2.2 Test for Normality

We can test for normality using a normal QQ plot. In the normal QQ plot, the residuals are plotted against the theoretical quantiles of a standard normal distribution. If our assumption of normality is correct, we should see points plotted in a perfect straight line. However based on our training data, we see a heavily tailed distribution, indicating non-normality. This result agrees with the Normal Q-Q plot in the original paper; MLR was still used as a model despite violating the assumption of normality. This issue was overlooked because the original paper also implemented models that do not require normally distributed errors, which are briefly touched upon later in this section. Because such models fall outside of our scope, we will discuss a remedy in the next section to make our data suitable just for a multiple linear regression model.

```
ti <- rstudent(m1)
par(mfrow = c(1, 2))
qqnorm(ti)
qqline(ti)
```



3.2.3 Variance Inflation Factor (VIF)

Multicollinearity can be detected by measuring the Variance Inflation Factor (VIF) for each regressor. All regressors should be independent of each other for a more robust and accurate model; multicollinearity can inflate the value of R^2, overestimating how much variation can be explained by the model. Given a regressor X_j and its coefficient β_j, the VIF can be calculated as such:

$VIF_j = {}^{11}{-}{R_{2j}}$ where R_{2j} is the coefficient of determination for regressing X_j on the remaining regressors. If regressors are correlated with each other, R_{2j} will increase, which also increases VIF. In an ideal model, we should expect to see values of VIF that are very close to 1. If values exceed 5, we should be cautious this would be a strong indication of collinearity and needs to be further investigated. In the `vif()` code below, we see that all the values seem to be relatively close to 1. This suggests that there is little to no multicollinearity between any regressors.

```
car::vif(m1)
```

```
##      CarrierDelay           TaxiIn           TaxiOut          Distance
##          1.039307          1.074482          1.107934          1.015322
##      WeatherDelay          NASDelay     SecurityDelay LateAircraftDelay
##          1.018771          1.189249          1.000842          1.046284
```

3.2.4 Bonferroni Outliers

Observations with values that deviate significantly from what we would expect can affect our analysis significantly, so we need to take them into account when interpreting results. These observations can be classified as outliers and influential points. The original paper categorizes outliers further, referring to these points as X outliers and Y outliers.

X outliers are determined based on leverage, denoted as h_{ii}. This is a measure of the "extremeness" of observation i with respect to the regressors as a value between 0 and 1. The sum of leverages across all n observations is equal to p + 1, where p is the number of regressors, and the mean leverage is equal to $p+1n$. Larger values of h_{ii} may indicate that observation i is influential; we should be wary if h_{ii} is greater than $2\bar{}h$.

Y outliers are determined based on the magnitude of studentized residuals (denoted t_i) which are computed based on a function that uses leverage to standardize the raw residuals. Observations with larger residuals are more likely to be outliers. We can conduct Bonferroni testing to determine outliers; we can consider an observation to be a Y outlier when $|t_i| > t_{1-\alpha/2n,n-p-1}$. A list of some of the top outliers are generated with the below code with `outlierTest()`, as well as how many outliers there are:

```
outlierTest(m1, n.max = 10) # top 10 outliers
```

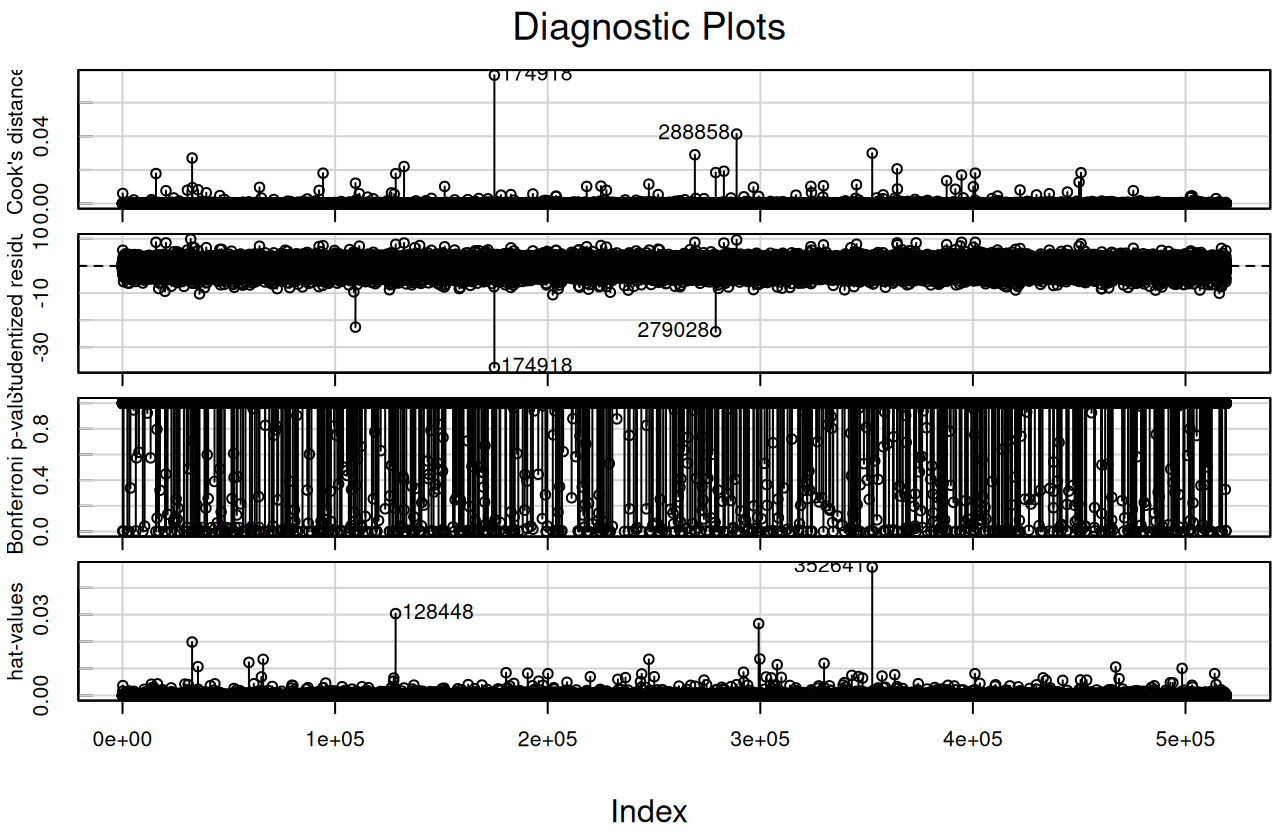
```
##          rstudent unadjusted p-value Bonferroni p
## 174918 -37.525544        9.1709e-308  4.7609e-302
## 279028 -24.292261        2.7999e-130  1.4535e-124
## 109575 -22.613940        3.5859e-113  1.8616e-107
## 202398 -10.644607         1.8593e-26   9.6521e-21
## 36219  -10.330411         5.1628e-25   2.6801e-19
## 515916 -10.096543         5.7517e-24   2.9859e-18
## 32084    9.872273         5.5163e-23   2.8637e-17
## 229456  -9.739105         2.0627e-22   1.0708e-16
## 288858   9.583233         9.4444e-22   4.9029e-16
## 108815  -9.576708         1.0060e-21   5.2226e-16
```

```
length(names(outlierTest(m1, n.max = 99999)$rstudent))
```

```
## [1] 442
```

The figure below with four subplots generated by `influenceIndexPlot` provides a visual representation of outliers and influential points. In the third subplot of this figure, we see Bonferroni p-values observations plotted against observation index. Most of them are large (indicating those observations are not outliers), but there are a substantial number which appear to fall below a significance level of 0.05. The original paper also had a handful of outliers which they made note of in their analysis. In the next section, we will acknowledge outliers and how we chose to handle them in our enhancement process.

```
car::influenceIndexPlot(m1) # first and third subplot
```



3.2.5 Influential observations with Cook's Distance

An observation is considered influential when removing this observation would significantly affect the predicted scores for other observations. This influence can be measured using a metric called Cook's distance (D_i) which is computed for each observation based on the sum of the squared differences between predictions made with all observations (β_j) and predictions made leaving out said observation ($\beta_{j(i)}$): $D_i = {}^{1p} + {}^{1p}\sum j=0(\hat{}\beta_j - \hat{}\beta_{j(i)})^2$

In the first subplot of the figure above, we see Cook's Distance observations plotted against observation index. A common guideline suggests that a Cook's distance exceeding 1 indicates an influential point. In this case, we observe that none of our observations have a distance greater than 1. We can continue assuming our training set doesn't contain any influential observations; this agrees with the findings in the original paper.

3.3 Other Models

Decision Tree and Random Forest

The original paper discusses the use of decision tree and random forest algorithms to determine the most influential regressors without the assumption of normality. A decision tree is a machine learning algorithm used for both classification and regression. It models decisions based on a series of conditions and represents them in a tree-like structure. The decision-making process involves partitioning the data repeatedly based on the features until a stopping condition is met. A random forest builds multiple decision trees and considers their predictions collectively to improve overall performance. It constructs a large number of decision trees during training and outputs the most frequent category (response variable is qualitative) or mean prediction (regression, or response variable is quantitative) of the individual trees. For the purposes of this analysis and enhancement report, we will not discuss these methods as they fall outside of our scope.

Section 4. Enhancement of Original Analysis

4.1 Violation of Assumptions

We noticed that there are numerous outliers in our training dataset. These extreme outliers may affect how we perform our enhancement, but for now we will leave them in our training data.

First we will run a test to formally determine if we can use the assumption of homoscedasticity. The original paper only utilized a visual approach to check this assumption using a residual plot, we will supplement this with a `ncvTest()` from the `car` package prior to applying any enhancements. Our null hypothesis of homoscedasticity is rejected, therefore we cannot assume constant variance of the errors. We will need to remedy this, in conjunction with the issue of non-normality and non-linearity.

```
car::ncvTest(m1)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 23002.29, Df = 1, p = < 2.22e-16
```

4.2 Variable Transformation and Assumption Reassessment

To address multiple violations in our assumptions, we can consider using Box-Cox power transformations on our regressors and response variable. The `powerTransform()` function in the `car` package will determine the best estimates for transformation parameters for each variable. Based on the estimated parameters, we then apply a transformation to each regressor respectively. The goal is to find a transformation of each variable that will result in approximately normal distributions for the residuals, a constant variance of the error, and linear relationships between functions of variables. These transformations are done before we construct the MLR model.

Because Box-Cox transformations require the variables to be strictly positive and there exist observations where the value of the regressor is 0, we have added a very small constant of 0.001 to each observed regressor value to avoid computational issues (avoids taking the logarithm of 0).

When we initially run the `powerTransform()` function, our output provided unexpected estimated parameters, particularly for the `SecurityDelay` regressor. We revisited the summary statistics for this regressor and noted it was extremely skewed due to outliers. We made the decision to perform the Box-Cox transformation without considering observations where outliers have a Bonferroni p-value smaller than a cutoff of 0.05. However, we will keep the outliers in our training data; the purpose of temporarily setting them aside is to find adequate estimates for power transformation parameters.

```
df_train_shifted <- data.frame(lapply(df_train, function(x) x + 0.001))
m11 <- lm(formula = DepDelay ~ ., data = df_train_shifted)
outliers1 <- df_train_shifted[(as.integer(names(abs(outlierTest(m11, n.max = 499)$rstudent)))), ]
df_train_shifted_no_outliers <- dplyr::anti_join(df_train_shifted, outliers1)
```

```
## Joining with `by = join_by(DepDelay, CarrierDelay, TaxiIn, TaxiOut, Distance,
## WeatherDelay, NASDelay, SecurityDelay, LateAircraftDelay)`
```

```
length(df_train_shifted[, 1]) == length(df_train_shifted_no_outliers[, 1]) + 442
```

```
## [1] TRUE
```

```
# verifying outliers were removed
```

```
pT <- powerTransform(
  cbind(
    CarrierDelay, TaxiIn, TaxiOut, Distance, WeatherDelay,
    NASDelay, SecurityDelay, LateAircraftDelay
  ) ~ 1,
  data = df_train_shifted_no_outliers, family = "bcPower"
)
summary(pT)
```

```
## bcPower Transformations to Multinormality
##                   Est Power Rounded Pwr Wald Lwr Bnd Wald Upr Bnd
## CarrierDelay         0.0690        0.07       0.0682       0.0698
## TaxiIn              -0.3551       -0.36      -0.3587      -0.3515
## TaxiOut             -0.3168       -0.32      -0.3210      -0.3126
## Distance             0.1681        0.17       0.1648       0.1713
## WeatherDelay        -1.6254       -1.63      -1.6298      -1.6209
## NASDelay            -0.1231       -0.12      -0.1241      -0.1221
## SecurityDelay      -29.2063      -29.21     -29.2858     -29.1268
## LateAircraftDelay    0.1031        0.10       0.1022       0.1039
##
## Likelihood ratio test that transformation parameters are equal to 0
##  (all log transformations)
##                                          LRT df       pval
## LR test, lambda = (0 0 0 0 0 0 0 0) 6949808  8 < 2.22e-16
##
## Likelihood ratio test that no transformations are needed
##                                           LRT df       pval
## LR test, lambda = (1 1 1 1 1 1 1 1) 38048530  8 < 2.22e-16
```

```
testTransform(pT, lambda = c(0, 0, 0, 0, -2, 0, 1, 0))
```

	LRT	df	pval
	<dbl>	<int>	<chr>
LR test, lambda = (0 0 0 0 -2 0 1 0)	13836976	8	< 2.22e-16

1 row

```
# use integer parameter estimates
# was advised to ignore power of -29 and leave untransformed
```

The criterion for `testTransform()` is if the p-value is greater than 0.05, this means that there is no evidence that we need to change the power transform parameters. Even when removing these outliers, we still get very odd parameter estimates. For simplicity of analysis, we rounded each parameter to the nearest integer (except the -29 power associated with `SecurityDelay`, this is left untransformed), despite the fact that the p-value suggests our chosen transformation is not optimal.

```
pT <- powerTransform(DepDelay ~ log(CarrierDelay) + log(TaxiIn) + log(TaxiOut) +
  log(Distance) + I(WeatherDelay^(-2)) + log(NASDelay) +
  SecurityDelay + log(LateAircraftDelay), df_train_shifted_no_outliers)
summary(pT)
```

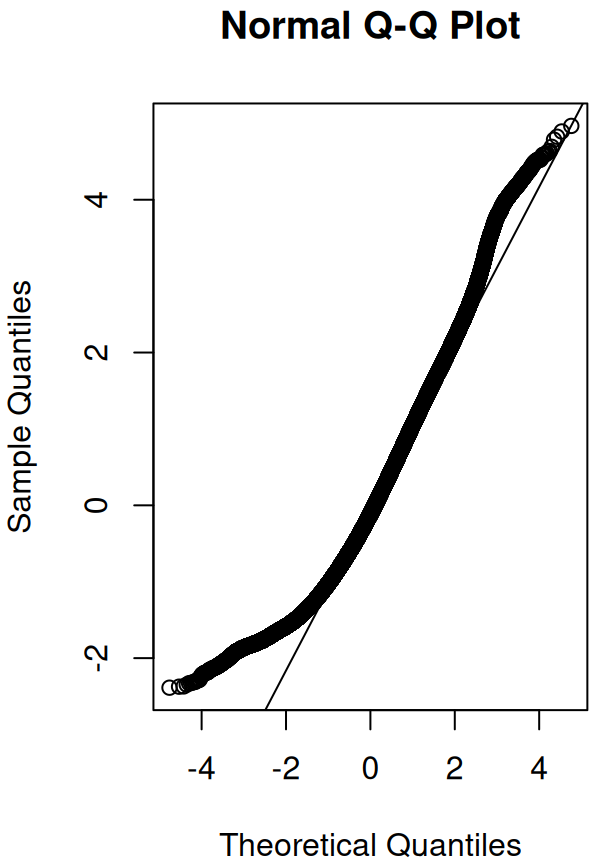
```
## bcPower Transformation to Normality
##    Est Power Rounded Pwr Wald Lwr Bnd Wald Upr Bnd
## Y1   -0.3188       -0.32      -0.3223      -0.3153
##
## Likelihood ratio test that transformation parameter is equal to 0
##  (log transformation)
##                            LRT df       pval
## LR test, lambda = (0) 33814.36  1 < 2.22e-16
##
## Likelihood ratio test that no transformation is needed
##                            LRT df       pval
## LR test, lambda = (1) 740396.1  1 < 2.22e-16
```

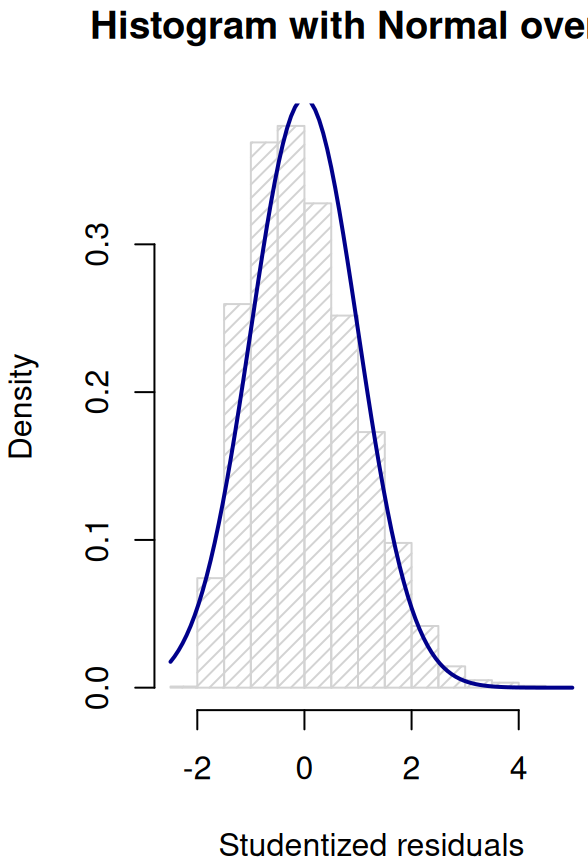
```
# lower LRT is better, log transform DepDelay
```

Once we have determined the appropriate transformations, we can reassess the assumptions of normality, constant variance, and linearity for a new model with the transformed variables:

```
m8 <- lm(log(DepDelay) ~ log(CarrierDelay) + log(TaxiIn) + log(TaxiOut) +
  log(Distance) + I(WeatherDelay^(-2)) + log(NASDelay) +
  SecurityDelay + log(LateAircraftDelay), data = df_train_shifted)
```

```
# Normal QQ Plot: normality
ti <- rstudent(m8)
par(mfrow = c(1, 2))
qqnorm(ti)
qqline(ti)
m <- mean(ti)
std <- sd(ti)
hist(ti,
  density = 20, breaks = 20, prob = TRUE, xlab = "Studentized residuals", # including histogram ov
erlay
  main = "Histogram with Normal overlay"
)
curve(dnorm(x, mean = m, sd = std), col = "darkblue", lwd = 2, add = TRUE, yaxt = "n")
```





In a Normal Q-Q plot, if our assumption of normality is correct, we should see points plotted in a straight line. Based on the normal Q-Q plot above, most of the points seem to adhere to a linear form, so normality assumption may be adequately satisfied. We have also added a density histogram of residuals which appears to follow a normal distribution.

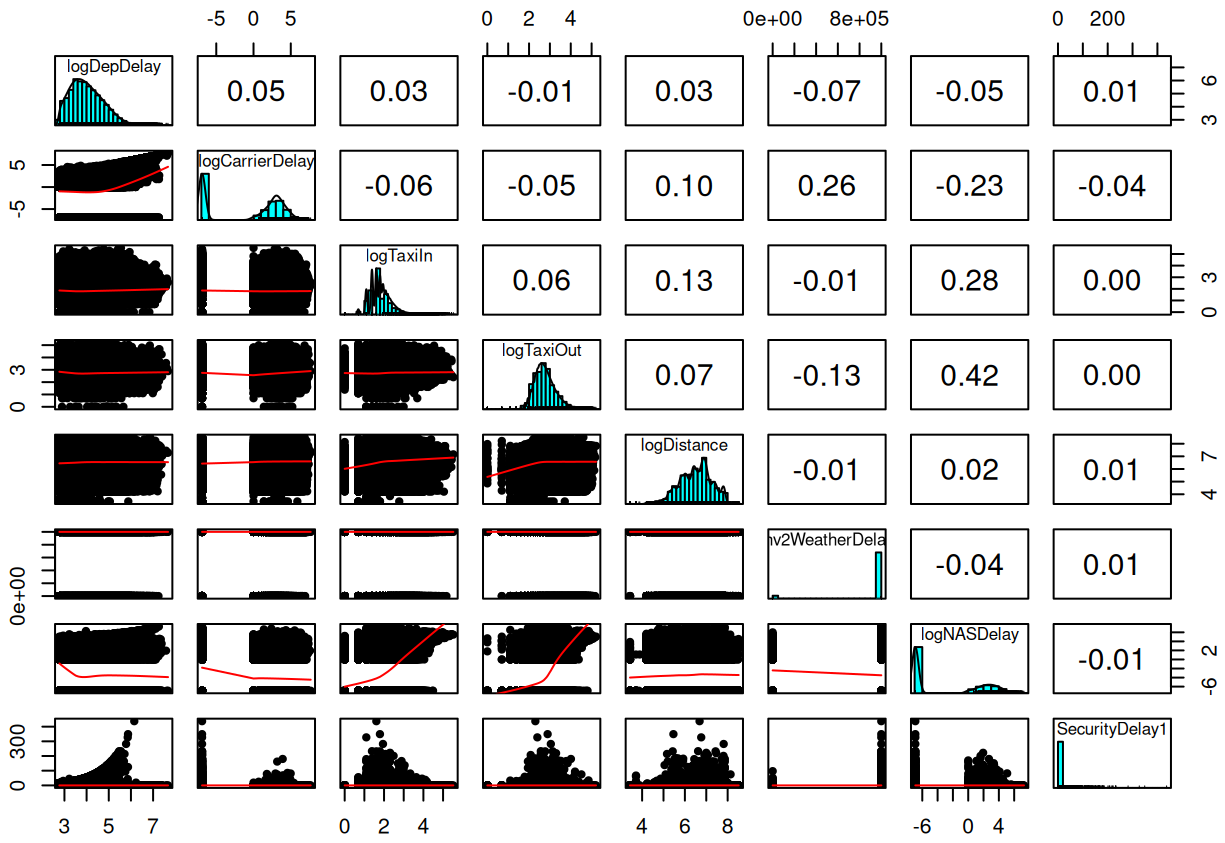
```
# formal non-constant variance test
ncvTest(m8)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 380.7772, Df = 1, p = < 2.22e-16
```

Based on our formal non-constant variance test, we reject the null hypothesis of a constant variance, thus heteroscedasticity is still an issue. The transformations we applied were not adequate enough to remedy completely; we will examine residual plots later to see this in detail.

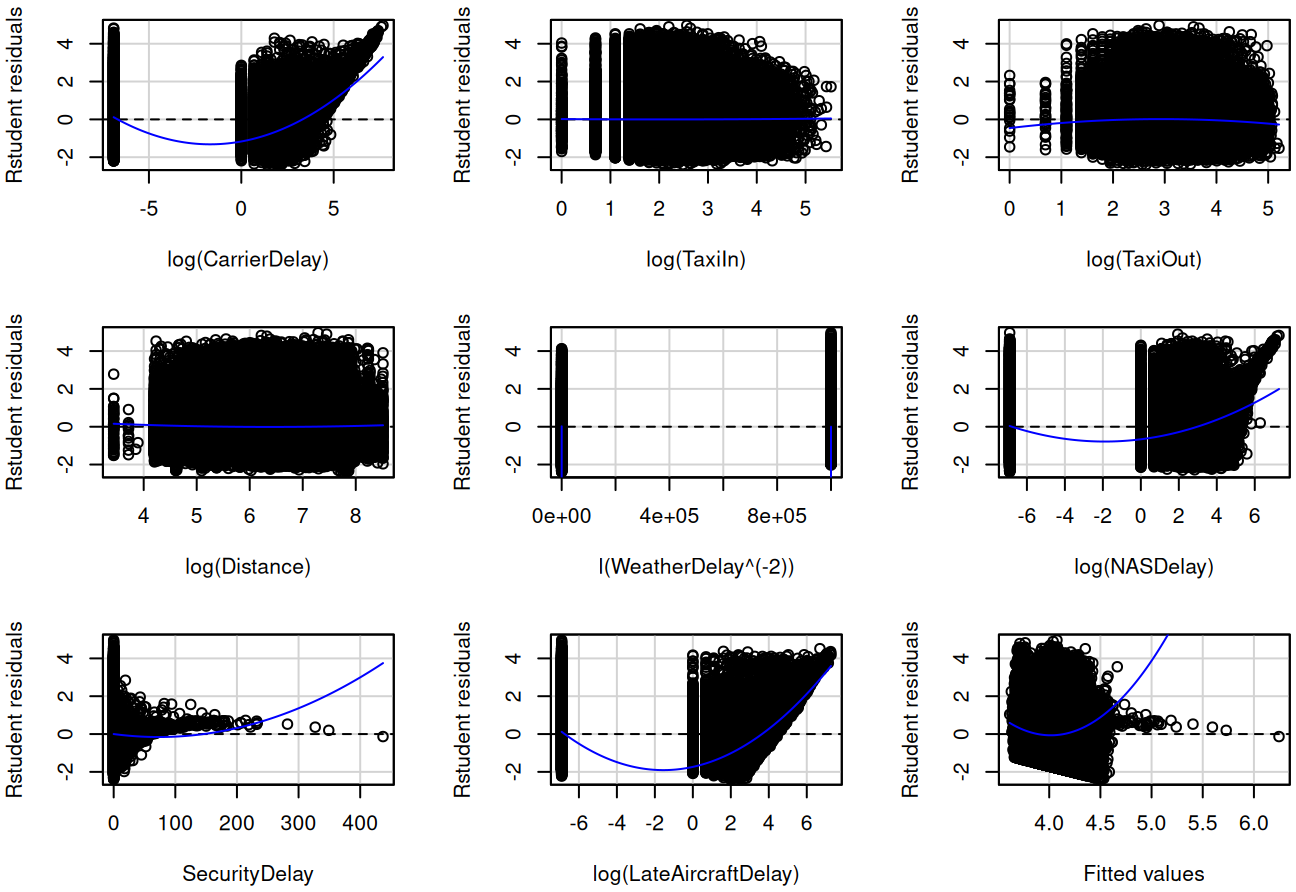
```
df_transformed_vars_train <- transform(df_train_shifted,
  logDepDelay = log(DepDelay), logCarrierDelay = log(CarrierDelay),
  logTaxiIn = log(TaxiIn), logTaxiOut = log(TaxiOut),
  logDistance = log(Distance), inv2WeatherDelay = I(WeatherDelay^(-2)),
  logNASDelay = log(NASDelay), SecurityDelay1 = SecurityDelay,
  logLateAircraftDelay = log(LateAircraftDelay)
)
```

```
# transformed scatterplot matrix: normality, linearity
pairs.panels(df_transformed_vars_train[, 10:17],
  ellipses = FALSE, hist.col = "cyan",
  method = "pearson", density = TRUE
)
```



Based on the scatterplots above, linearity does not appear to be significantly present between `DepDelay` and regressors. Collinearity is not an issue.

```
# residual plots: constant variance, linearity
residualPlots(m8, type = "rstudent")
```




```
##                          Test stat Pr(>|Test stat|)
## log(CarrierDelay)         465.2782        < 2.2e-16 ***
## log(TaxiIn)                 1.6447              0.1
## log(TaxiOut)              -15.4013        < 2.2e-16 ***
## log(Distance)              10.0832        < 2.2e-16 ***
## I(WeatherDelay^(-2))       17.1706        < 2.2e-16 ***
## log(NASDelay)             215.3539        < 2.2e-16 ***
## SecurityDelay               6.1598        7.291e-10 ***
## log(LateAircraftDelay)    508.7107        < 2.2e-16 ***
## Tukey test                 99.0036        < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Some regressors have residuals with constant variance while others have more questionable distributions. Assumption of constant variance is not satisfied. Some of the residuals are dispersed randomly with no distinct pattern which help support the assumption of linearity, but we should still proceed with caution.

4.3 Model Selection and Evaluation

In addition to transforming our variables so that they are better suited for a multiple linear regression model, our goal is to find an MLR model with fewer regressors with an equivalent if not greater predictive ability. We are selecting a model based on the principle of parsimony, or the idea that a “simpler” model with fewer regressors but is equally effective is preferred. We will utilize multiple methods of variable discovery and model selection, including subset selection and regularized methods.

Subset selection is the process of iteratively going through models of different sizes, then selecting the best one based on AIC/BIC. The regularized methods we will use in the below code are LASSO, Adaptive LASSO, Elastic Net, and SCAD, each with slightly different methods of minimizing RSS (residual sum of squares) based on the number of regressors. This allows us to have an assortment of models to choose from.

When comparing models fit on the same dataset with the same response variable and sample size, a model with larger R_{2a} is preferred, but smaller AIC. R_{2a} is an adjusted R^2 value which accounts for the size of the model. $R_{2a} = 1 - (^{n-1}_{n-(p_c+1)})\frac{RSS_{p_C}}{SYY}$ where p_C is the number of regressors in the model, RSS_{p_C} is the residual sum of squares, and SYY is total variation.

When we develop a regression model based on a given dataset, this model is more likely to understate variability in making future predictions via residual mean squared error (and thus its square root RMSE). We can also evaluate the predictive ability of a given regression model using cross validation procedures, which compares test data to predictions from provided regressor values. We use MSE, MAE, and VEcv as metrics to measure predictive capability.

MSE (mean squared prediction error) is computed as: $\frac{\sum_{i=1}^{n_H}(y_{H_i} - \hat{y}_{*H_i})^2}{n_H}$ where y_{H_i} is the value of the response variable in validation case i, $\hat{y}_{*H_i}$ is the predicted value for validation case i based on the training dataset, and n_H is the number of cases in the validation dataset. If this metric is relatively close to the square RMSE (denoted $\hat{\sigma}^2$), then $\hat{\sigma}^2$ is an adequate indication of this model's predictive ability. Otherwise if MSE is significantly larger than $\hat{\sigma}^2$, then MSE should be used instead.

MAE (mean absolute prediction error) is the average absolute prediction error, and is less sensitive to outliers. MAE is computed as: $\frac{\sum_{i=1}^{n_H}|y_{H_i} - \hat{y}_{*H_i}|}{n_H}$ Our goal is to find a model that minimizes both MSE and MAE, so that the model is more robust in predicting capability.

VEcv is the amount of variance explained based on cross-validation, computed as:
$1 - \frac{\sum_{i=1}^{n_H}(y_{H_i} - \hat{y}_{*H_i})^2}{\sum_{i=1}^{n_H}(y_{H_i} - \bar{y}_{H_i})^2}$ Where $\bar{y}_H$ is sample mean of test responses y_{H_i} of all test observations.

These three metrics as well as the selected regressors for the candidate model are displayed after each method's output.

4.3.1 Subset Selection

```
df_test_shifted <- data.frame(lapply(df_test, function(x) x + 0.001))

df_transformed_vars_test <- transform(df_test_shifted,
  logDepDelay = log(DepDelay), logCarrierDelay = log(CarrierDelay),
  logTaxiIn = log(TaxiIn), logTaxiOut = log(TaxiOut),
  logDistance = log(Distance), inv2WeatherDelay = I(WeatherDelay^(-2)),
  logNASDelay = log(NASDelay), SecurityDelay1 = SecurityDelay,
  logLateAircraftDelay = log(LateAircraftDelay)
)
```

```
f <- log(DepDelay) ~ log(CarrierDelay) + log(TaxiIn) + log(TaxiOut) +
  log(Distance) + I(WeatherDelay^(-2)) + log(NASDelay) +
  SecurityDelay + log(LateAircraftDelay)
out <- regsubsets(f, nvmax = 8, data = df_train_shifted)
cbind(summary(out)$which, adjr2 = summary(out)$adjr2, BIC = summary(out)$bic)
```

```
##   (Intercept) log(CarrierDelay) log(TaxiIn) log(TaxiOut) log(Distance)
## 1           1                 0          0            0             0
## 2           1                 1          0            0             0
## 3           1                 1          0            0             0
## 4           1                 1          1            0             0
## 5           1                 1          1            0             0
## 6           1                 1          1            0             1
## 7           1                 1          1            0             1
## 8           1                 1          1            1             1
##   I(WeatherDelay^(-2)) log(NASDelay) SecurityDelay log(LateAircraftDelay)
## 1                    0             0             0                     1
## 2                    0             0             0                     1
## 3                    1             0             0                     1
## 4                    1             0             0                     1
## 5                    1             0             1                     1
## 6                    1             0             1                     1
## 7                    1             1             1                     1
## 8                    1             1             1                     1
##        adjr2        BIC
## 1 0.00894354  -4638.417
## 2 0.01598990  -8330.420
## 3 0.02752651 -14440.528
## 4 0.02887033 -15146.228
## 5 0.02936145 -15396.669
## 6 0.02979502 -15616.449
## 7 0.02998951 -15708.366
## 8 0.02999112 -15697.066
```

```
x.test <- as.matrix(df_transformed_vars_test[, 11:18])
y.test <- df_transformed_vars_test$logDepDelay
x.train <- as.matrix(df_transformed_vars_train[, 11:18])
y.train <- df_transformed_vars_train$logDepDelay
n.test <- dim(df_transformed_vars_test)[1]
syy.test <- (n.test - 1) * var(y.test)
```

```
m7 <- lm(logDepDelay ~ logCarrierDelay + logTaxiIn +
  logDistance + inv2WeatherDelay + logNASDelay +
  SecurityDelay1 + logLateAircraftDelay, data = df_transformed_vars_train)
m8 <- lm(logDepDelay ~ logCarrierDelay + logTaxiIn + logTaxiOut +
  logDistance + inv2WeatherDelay + logNASDelay +
  SecurityDelay1 + logLateAircraftDelay, data = df_transformed_vars_train)
# full model is a candidate
```

```
m7.predicted <- predict(m7, newdata = as.data.frame(x.test))
sse <- sum((y.test - m7.predicted)^2)
m7.out <- c(
  mse = sse / n.test,
  mae = mean(abs(y.test - m7.predicted)),
  VEcv = (1 - sse / syy.test) * 100
)
m7.out
```

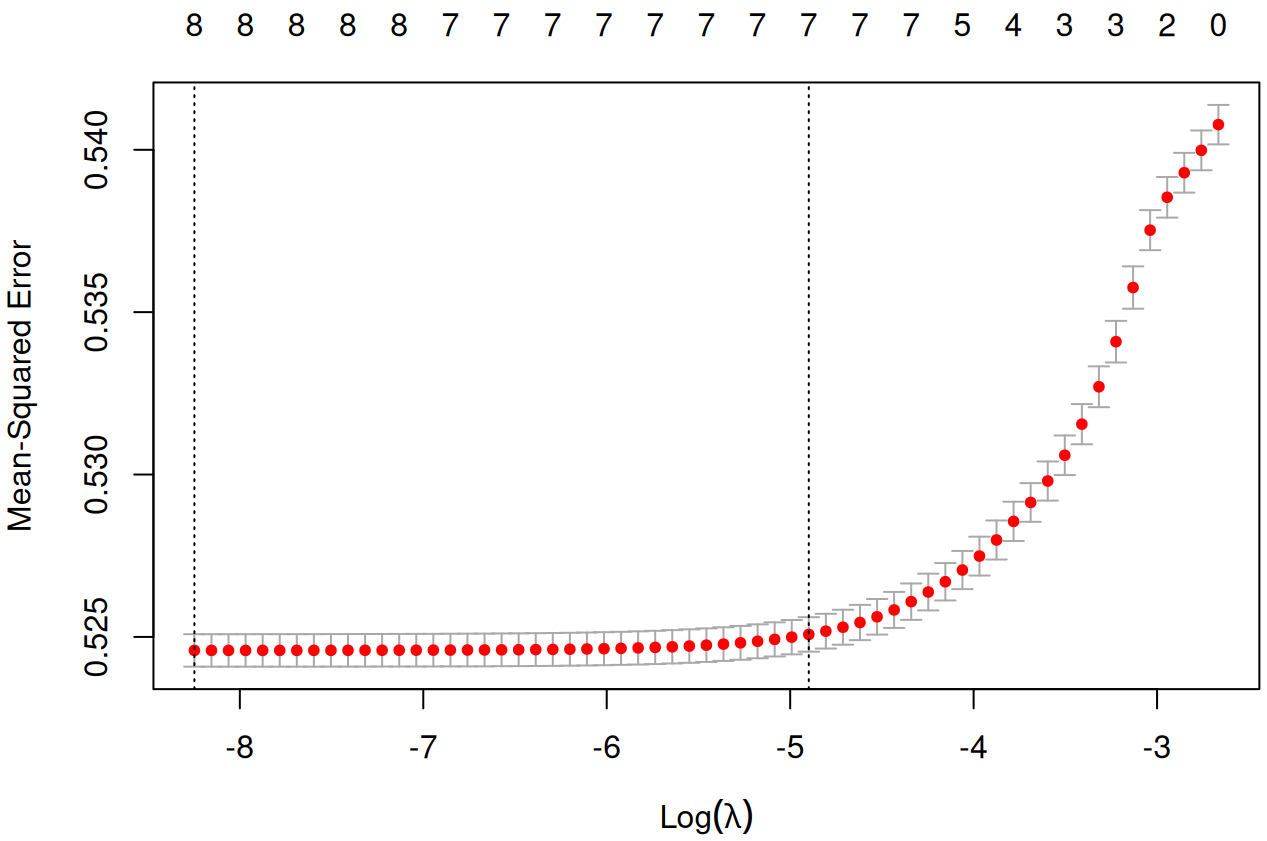
```
##       mse       mae      VEcv
## 0.5215175 0.5868347 2.9423162
```

```
m8.predicted <- predict(m8, newdata = as.data.frame(x.test))
sse <- sum((y.test - m8.predicted)^2)
m8.out <- c(
  mse = sse / n.test,
  mae = mean(abs(y.test - m8.predicted)),
  VEcv = (1 - sse / syy.test) * 100
)
m8.out
```

```
##       mse       mae      VEcv
## 0.5215070 0.5868242 2.9442623
```

4.3.2 LASSO

```
mod.mat <- model.matrix(f, data = df_transformed_vars_train)
out.lasso <- cv.glmnet(
  x = mod.mat[, -1],
  y = log(df_transformed_vars_train$DepDelay),
  standardize = TRUE, intercept = TRUE, alpha = 1, nfolds = 3
)
# nfolds is 3 to reduce computational power needed for a very large dataset
plot(out.lasso)
```




```
coef(out.lasso, s = out.lasso$lambda.1se)
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                                  s1
## (Intercept)             4.160900e+00
## log(CarrierDelay)       1.507578e-02
## log(TaxiIn)             3.318312e-02
## log(TaxiOut)            .
## log(Distance)           1.317045e-02
## I(WeatherDelay^(-2))   -2.977816e-07
## log(NASDelay)          -1.239103e-03
## SecurityDelay           2.731701e-03
## log(LateAircraftDelay)  1.747368e-02
```

```
idx.1se <- which(abs(coef(out.lasso, s = out.lasso$lambda.1se)) > 0)
rownames(coef(out.lasso, s = out.lasso$lambda.1se))[idx.1se]
```

```
## [1] "(Intercept)"            "log(CarrierDelay)"      "log(TaxiIn)"
## [4] "log(Distance)"          "I(WeatherDelay^(-2))"   "log(NASDelay)"
## [7] "SecurityDelay"          "log(LateAircraftDelay)"
```

```
m_lasso <- lm(
  log(DepDelay) ~ log(CarrierDelay) + log(TaxiIn) + log(Distance) +
    I(WeatherDelay^(-2)) + log(NASDelay) + SecurityDelay + log(LateAircraftDelay),
  data = df_transformed_vars_train
)
# Calculate Metrics
lasso.predicted <- predict(out.lasso, out.lasso$lambda.1se, new = x.test)
sse <- sum((y.test - lasso.predicted)**2)
lasso_metrics <- c(
  mse = sse / n.test,
  mae = mean(abs(y.test - lasso.predicted)), VEcv = (1 - sse / syy.test) * 100
)
lasso_metrics
```

```
##       mse       mae      VEcv
## 0.5219354 0.5872180 2.8645283
```

4.3.3 Adaptive LASSO

```
ols.beta <- coef(lm.fit(x = cbind(1, x.train), y = y.train))
alasso <- cv.glmnet(
  x = mod.mat[, -1], y = log(df_transformed_vars_train$DepDelay),
  type.measure = "mse", alpha = 1, nlambda = 200, penalty = 1 / abs(ols.beta[-1])
)
alasso.predicted <- predict(alasso, alasso$lambda.1se, new = x.test)
sse <- sum((y.test - alasso.predicted)**2)
alasso.out <- c(
  mse = sse / n.test,
  mae = mean(abs(y.test - alasso.predicted)), VEcv = (1 - sse / syy.test) * 100
)
coef(alasso, s = alasso$lambda.1se)
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                                s1
## (Intercept)            3.86750031
## log(CarrierDelay)      0.01030786
## log(TaxiIn)            0.03535374
## log(TaxiOut)           .
## log(Distance)          0.01427535
## I(WeatherDelay^(-2))   .
## log(NASDelay)          .
## SecurityDelay          .
## log(LateAircraftDelay) 0.01516983
```

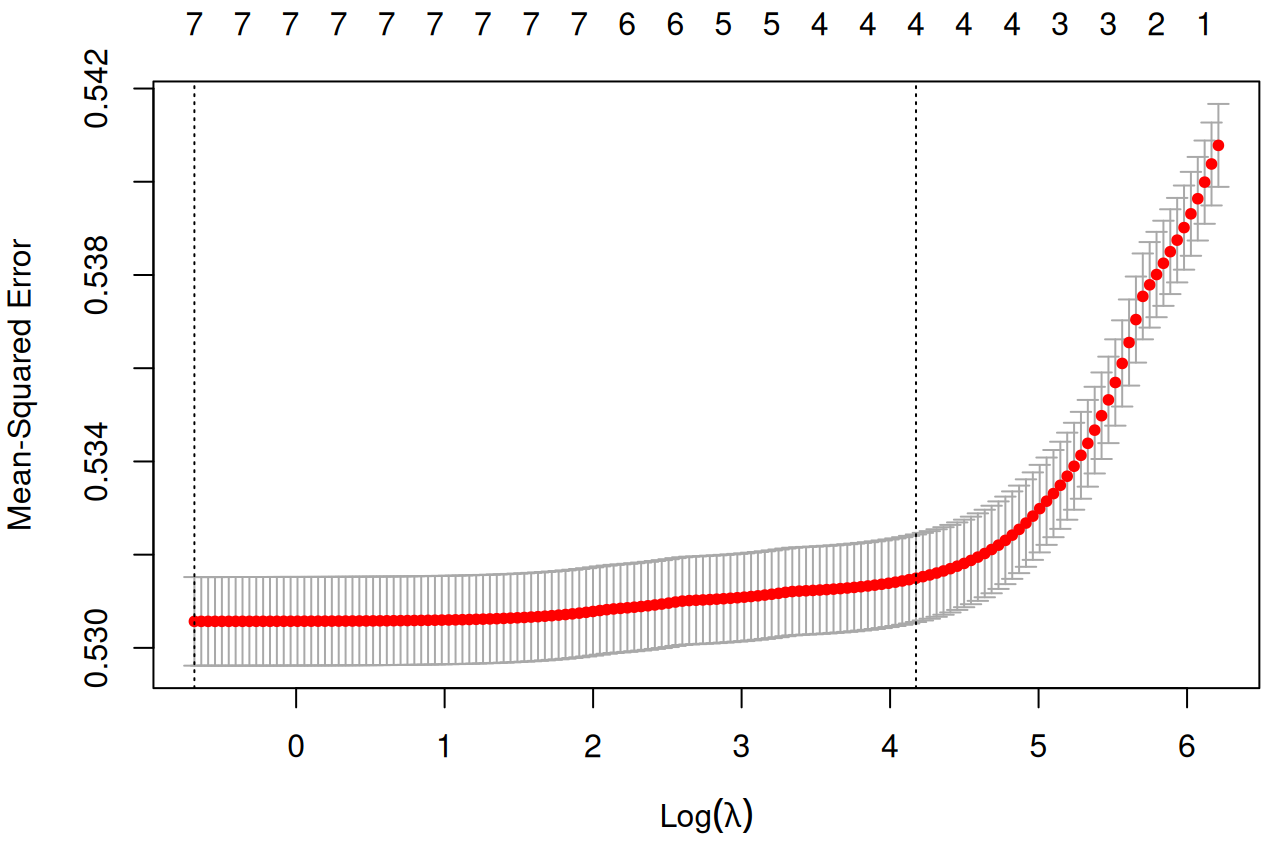
```
idx_1se <- which(abs(coef(alasso, s = alasso$lambda.1se)) > 0)
rownames(coef(alasso, s = alasso$lambda.1se))[idx_1se]
```

```
## [1] "(Intercept)"            "log(CarrierDelay)"      "log(TaxiIn)"
## [4] "log(Distance)"          "log(LateAircraftDelay)"
```

```
m_adaptive_lasso <- lm(
  log(DepDelay) ~ log(CarrierDelay) +
    log(TaxiIn) + log(Distance) + log(LateAircraftDelay),
  data = df_transformed_vars_train
)
alasso.out
```

```
##       mse       mae      VEcv
## 0.5279747 0.5904351 1.7405767
```

```
plot(alasso)
```



4.3.4 Elastic Net

```
models <- list()
results_elastic_net <- data.frame(name = character(), alpha = numeric(), mse = numeric(), mae = nu
meric(), Vecv = numeric())
colnames(results_elastic_net) <- c("name", "alpha", "mse", "mae", "Vecv")
for (i in 0:20) {
  name <- paste0("alpha", i / 20, sep = "")
  models[[name]] <- cv.glmnet(x.train, y.train,
    type.measure = "mse",
    alpha = i / 20, family = "gaussian"
  )
  predicted <- predict(models[[name]],
    s = models[[name]]$lambda.1se, newx = x.test
  )
  sse <- sum((y.test - predicted)**2)
  mse <- sse / n.test
  mae <- mean(abs(y.test - predicted))
  vecv <- (1 - sse / syy.test) * 100
  results_elastic_net <- rbind(results_elastic_net, data.frame(
    name = name, alpha = i / 20, mse = mse, mae = mae, VEcv = vecv
  ))
}
results_elastic_net[7, ]
```

	name	alpha	mse	mae	VEcv
	<chr>	<dbl>	<dbl>	<dbl>	<dbl>
7	alpha0.3	0.3	0.5225139	0.5876208	2.756868

1 row

```
coef(models$alpha0.3, s = models$alpha0.3$lambda.1se)
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                                s1
## (Intercept)          4.157967e+00
## logCarrierDelay      1.329890e-02
## logTaxiIn            2.696694e-02
## logTaxiOut           .
## logDistance          1.067529e-02
## inv2WeatherDelay    -2.675954e-07
## logNASDelay         -1.150071e-03
## SecurityDelay1       1.612694e-03
## logLateAircraftDelay 1.585926e-02
```

```
models$alpha0.3
```

```
##
## Call:  cv.glmnet(x = x.train, y = y.train, type.measure = "mse", alpha = i/20,      family = "g
aussian")
##
## Measure: Mean-Squared Error
##
##      Lambda Index Measure       SE Nonzero
## min 0.00072    63  0.5246 0.001138       8
## 1se 0.03286    22  0.5257 0.001161       7
```

```
idx.1se <- which(abs(coef(models$alpha0.3, s = models$alpha0.3$lambda.1se)) > 0)
rownames(coef(models$alpha0.3, s = models$alpha0.3$lambda.1se))[idx.1se]
```

```
## [1] "(Intercept)"          "logCarrierDelay"      "logTaxiIn"
## [4] "logDistance"          "inv2WeatherDelay"     "logNASDelay"
## [7] "SecurityDelay1"       "logLateAircraftDelay"
```

```
m_elastic_net <- lm(log(DepDelay) ~ log(CarrierDelay) + log(TaxiIn) + I(WeatherDelay**(-2)) + log
(Distance) + log(LateAircraftDelay), data = df_transformed_vars_train)
```

4.3.5 SCAD

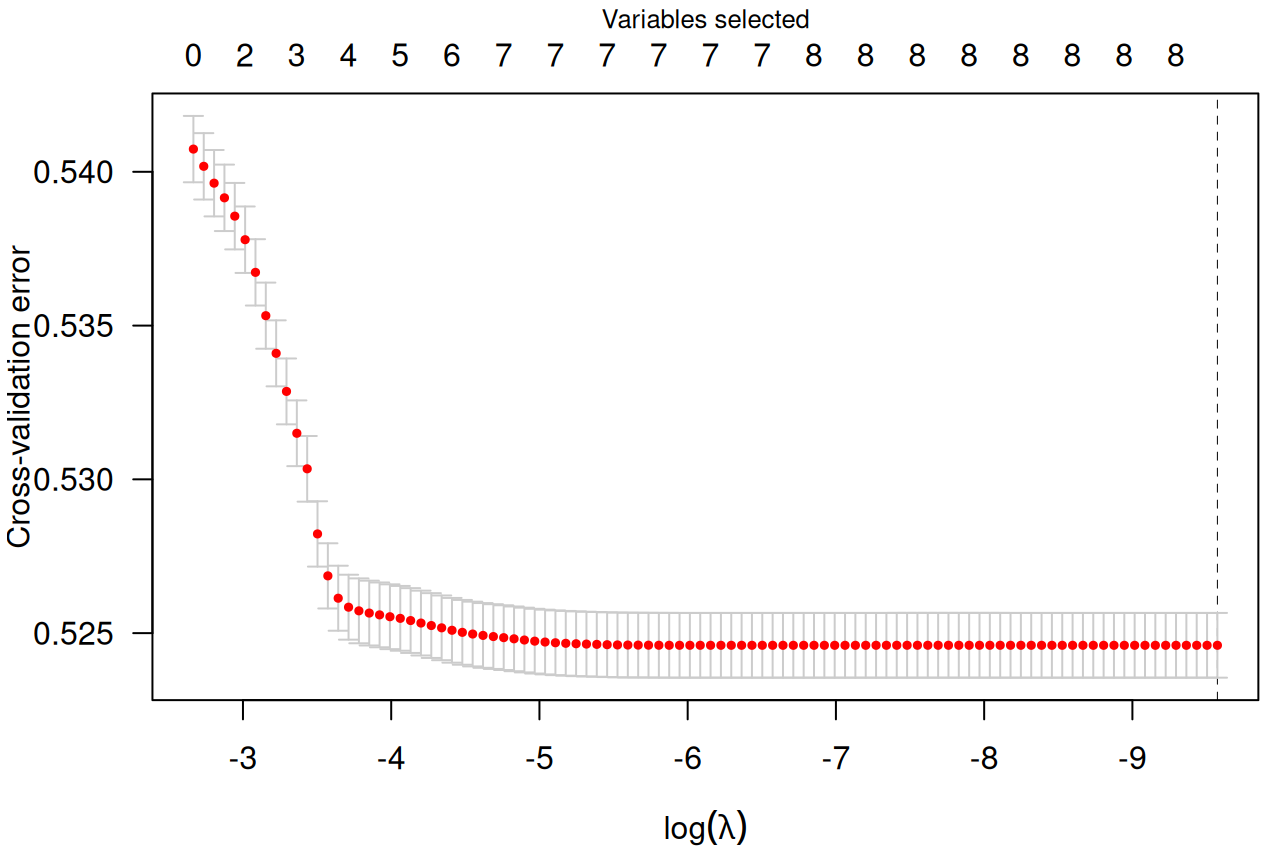
```
scad <- cv.ncvreg(X = mod.mat[, -1], y = log(df_transformed_vars_train$DepDelay), standardize = TR
UE, intercept = TRUE, nfolds = 3, penalty = "SCAD")
scad.predicted <- predict(scad, which = scad$min, X = cbind(x.test), type = "response")
sse <- sum((y.test - scad.predicted)**2)
scad.out <- c(mse = sse / n.test, mae = mean(abs(y.test - scad.predicted)), VEcv = (1 - sse / syy.
test) * 100)

coef(scad, s = scad$lambda.1se)
```

```
##           (Intercept)      log(CarrierDelay)            log(TaxiIn)
##          4.116769e+00           1.769203e-02           5.006444e-02
##           log(TaxiOut)          log(Distance)   I(WeatherDelay^(-2))
##          3.090100e-03           2.076845e-02          -3.463094e-07
##          log(NASDelay)          SecurityDelay log(LateAircraftDelay)
##          -2.563422e-03           5.696945e-03           1.983695e-02
```

```
beta <- scad$fit$beta[, scad$min]
idx <- which(abs(beta) > 0)
names(beta)[idx]
```

```
## [1] "(Intercept)"            "log(CarrierDelay)"      "log(TaxiIn)"
## [4] "log(TaxiOut)"           "log(Distance)"          "I(WeatherDelay^(-2))"
## [7] "log(NASDelay)"          "SecurityDelay"          "log(LateAircraftDelay)"
```

```
plot(scad)
```




```
m_elastic_net <- lm(log(DepDelay) ~ log(CarrierDelay) + log(TaxiIn) + I(WeatherDelay**(-2)) + log
(Distance) + log(LateAircraftDelay), data = df_transformed_vars_train)
scad.out
```

```
##       mse       mae      VEcv
## 0.5215070 0.5868242 2.9442623
```

4.3.6 Control Evaluation

```
m1 <- lm(exp(logDepDelay) ~ exp(logCarrierDelay)
  + exp(logTaxiIn) + exp(logTaxiOut) +
  exp(logDistance) + I(inv2WeatherDelay^(-1 / 2))
  + exp(logNASDelay) + SecurityDelay1 +
  exp(logLateAircraftDelay), data = df_transformed_vars_train)
m1.predicted <- predict(m1, newdata = as.data.frame(x.test))
sse <- sum((y.test - m1.predicted)^2)
m1.out <- c(
  mse = sse / n.test, mae = mean(abs(y.test - m7.predicted)),
  VEcv = (1 - sse / syy.test) * 100
)
m1.out
```

```
##           mse           mae          VEcv
##  1.095271e+04  5.868347e-01 -2.038267e+06
```

4.4 Model Summary

The following are candidate models with their regressors, and predictive performance metrics.

```
# mod.summary <- data.frame(
# method=c('Subset Selection 1', 'Subset Selection 2', 'LASSO', 'Adaptive LASSO', 'Elastic Net',
'SCAD'),
#                  regressors=c(
# 'log(Carrier Delay), log(Taxi In),  log(Distance), (WeatherDelay)^(-2), log(NAS Delay), Security
Delay, log(Late Aircraft Delay)',
# 'log(Carrier Delay), log(Taxi In), log(Taxi Out), log(Distance), (WeatherDelay)^(-2), log(NAS De
lay), SecurityDelay, log(Late Aircraft Delay)',
#  'log(Carrier Delay), log(Taxi In),  log(Distance), (WeatherDelay)^(-2), log(NAS Delay), Securit
yDelay, log(Late Aircraft Delay)',
# 'log(Carrier Delay), log(Taxi In), log(Distance),  log(Late Aircraft Delay)',
# 'log(Carrier Delay), log(Taxi In),  log(Distance), (WeatherDelay)^(-2), log(NAS Delay), Security
Delay, log(Late Aircraft Delay)',
# mod_summary = data.frame(c(m7.out,m8.out,lasso_metrics, alasso.out,results_elastic_net[7,][3:5],
scad.out))
methods <- c(
  "Subset Selection1", "Subset Selection2",
  "Lasso", "AdaptiveLasso", "ElasticNet", "SCAD"
)
elastic_net <- results_elastic_net[7, ][3:5]
mod_summary <- rbind(m7.out, m8.out)
mod_summary <- rbind(mod_summary, lasso_metrics)
mod_summary <- rbind(mod_summary, alasso.out)
mod_summary <- rbind(mod_summary, elastic_net)
mod_summary <- rbind(mod_summary, scad.out)
mod_summary <- cbind(methods, mod_summary)
mod_summary <- as.data.frame(mod_summary)

# Add Regressors used in each method using names(coef(model))
regressors <- rbind(
  toString(names(m7$coefficients)[-1]), toString(names(m8$coefficients)[-1]), toString(names(out.l
asso$glmnet.fit$beta[, 1])[-1]), toString(names(alasso$glmnet.fit$beta[, 1])[-1]),
  toString(names(models$alpha0.3$glmnet.fit$beta[, 1])[-1]), toString(names(scad$fit$beta[, 1])[-
1])
)
mod_summary <- cbind(mod_summary, regressors)

mod_summary
```

	methods <chr>	**mse** <dbl>	**mae** <dbl>	**VEcv** <dbl>
m7.out	Subset Selection1	0.5215175	0.5868347	2.942316
m8.out	Subset Selection2	0.5215070	0.5868242	2.944262
lasso_metrics	Lasso	0.5219354	0.5872180	2.864528
alasso.out	AdaptiveLasso	0.5279747	0.5904351	1.740577
7	ElasticNet	0.5225139	0.5876208	2.756868
6	SCAD	0.5215070	0.5868242	2.944262

6 rows | 1-5 of 6 columns

To determine the best model, we aim to have the lowest mean standard prediction error (MSE), mean absolute prediction error (MAE), and largest proportion of variance explained by cross validation. We found that the model with all eight transformed regressors has the lowest MSE and MSE, and highest VEcv. This model was selected by both the subset selection method and SCAD method. We noted that in other candidate models, the most likely regressor to be excluded first was `log(TaxiOut)`.

In comparison to the original, untransformed model, we have significantly reduced the mean standard prediction error and mean absolute prediction error, but have in turn significantly reduced variance explained by cross validation. By attempting to mold the erratic variance from the data to fit our MLR assumptions, we sacrifice some interpretability.

Section 5. Discussion and Conclusion

There are some discrepancies between the processes in which the original authors procured their analysis and our attempted replication due to limited information provided. The specific procedure of how the original dataset was cleaned, and how the "Hmisc" package was utilized to impute values were not described in detail. We were advised to remove observations which contained any null values (instead of using imputation) to work with a dataset of a manageable size.

In addition to pre-processing, to determine appropriate Box-Cox transformations, we added a small constant of 0.001 to each observation's regressors and response variable to avoid computational errors, and removed outliers with the largest residuals. The purpose of using the `powerTransform()` function was to search for suitable parameters that indicated how to transform the variables appropriately. We then extended those transformations on the outliers, acknowledging that these may not be as suitable for these observations.

Despite the differences in our processes, we were able to verify which MLR assumptions were violated, and find an approximation to the original MLR model based on the procedure provided. Both the original paper and our replication violated assumptions of normality, linearity, and homoscedasticity. The original report relied on visual plots to determine features that validate (or invalidate) assumptions, while our enhancement added formal tests. Independence of observations is also questionable because flights may interfere with each other when airports have finite capacities. There were a substantial number of outliers, but no influential points. Both analyses also determine that collinearity is not likely to be an issue.

Our enhancement attempts to address these violations by applying the aforementioned power transformations, which was able to contribute to normality (according to Q-Q plot) and some linearity (random patterns in residual plots). We conclude that we can create a better multiple linear regression model based on additional metrics considered in this enhancement that were not included in the original paper. Our goal was to identify a model with better or similar predictive capability with less regressors. Since MLR assumptions are not fully satisfied, what we had determined to be the best model as well as its other candidates should be used with caution.

References:

Original Paper:

```
@article{kalliguddi_leboulluec_2017,
title={Predictive Modeling of Aircraft Flight Delay},
volume={5}, DOI={https://doi.org/10.13189/ujm.2017.051003},
number={10}, journal={Universal Journal of Management},
author={Kalliguddi, Anish M. and Leboulluec, Aera K.},
year={2017}, month={Oct}, pages={485–491} }
```

MLA citation: - Kalliguddi, Anish M., and Aera K. Leboulluec. "Predictive Modeling of Aircraft Flight Delay." Universal Journal of Management, vol. 5, no. 10, Oct. 2017, pp. 485–91, https://doi.org/10.13189/ujm.2017.051003 (https://doi.org/10.13189/ujm.2017.051003).

(We also referenced notes from Dr. Schifano on Blackboard frequently.)