Final Project 704

Rohit Singh

5/14/2018

Automatic Dependent Surveillance Broadcast (ADS-B) DATA ANALYSIS

1. Introduction

In the past two decades, the wireless communication industry has seen an explosive growth in smart devices and data traffic. This is mainly due to the increasing appetite for mobile and web applications services (e.g. YouTube, Facebook, Facetime, etc.) and the new technologies that enable these services (e.g. 4G/LTE and ever-improving versions of WiFi). Data usage over mobile networks are rapidly increasing and will continue to show an upward trend over the next decade [1,2]. The Cisco Visual Networking (CVN) Index forecast report, shows that, by 2019, mobile and WiFi devices will account for 81% of Internet Traffic [3]. However, we are the verge of a spectrum crisis, since most of the spectrum is either too costly to buy or congested to do any operation without going through "the tragedy of the commons." This calls for more spectrum sharing strategies.

To satisfy the large need for spectrum, a number of spectrum bands are being considered for sharing with other applications (eg. Cellular and Public Safety). There are some frequency bands which are being shared based on the separation in the geographic location and/or by limiting the operational power. This is how television and radio stations share the spectrum. Another good example would be sharing between satellites and fixed links.

In this work, we propose a sharing prospective for the 4.2-4.4 GHz band, which is currently being used by the altimeter radar of an aircraft, with LTE [4]. The radar altimeter is used to perform the safety critical task of providing guidance during takeoff and landing of an aircraft. Aircraft flight patterns are carefully controlled and monitored. Thus, interference to a ground-based LTE system from aircraft altimeters and interference to the altimeter from an LTE system will be a function of location and time. Measured power levels are expected to be greater during periods of high airline activity and near airports. A much lower level of power is expected at locations distant from airports and distant from established flight paths.

Sharing the radar altimeter band with LTE comes with challenges, as it is used for the safety-critical task of providing guidance. However, with a deep understanding of the problem, we can easily see opportunities for sharing this band without crossing the safety line.

2. Method

To find a possibility of sharing this band with the LTE in a given space-time domain we need to predict the flight paths. To do so we use the Automatic Dependent Surveillance-Broadcast (ADS-B) a surveillance-based system that determines and broadcasts an aircraft's position, airspeed, and other data. The ADS-B is a reporting technology which allows the Federal Aviation Administration (FAA) to track the aircraft. The data is received by the ground stations for tracking the aircraft and it can also be used by other aircraft to allow them to have self-separation.

ADS-B is an integral part of the US Next Generation Air Transportation System (NextGen) and Airports Authority of India (AAI) in line with ICAO for fixing the shortcomings of the present air-travel. ADS-B will help in managing air traffic, improving visibility, receiving weather information and easily broadcasting

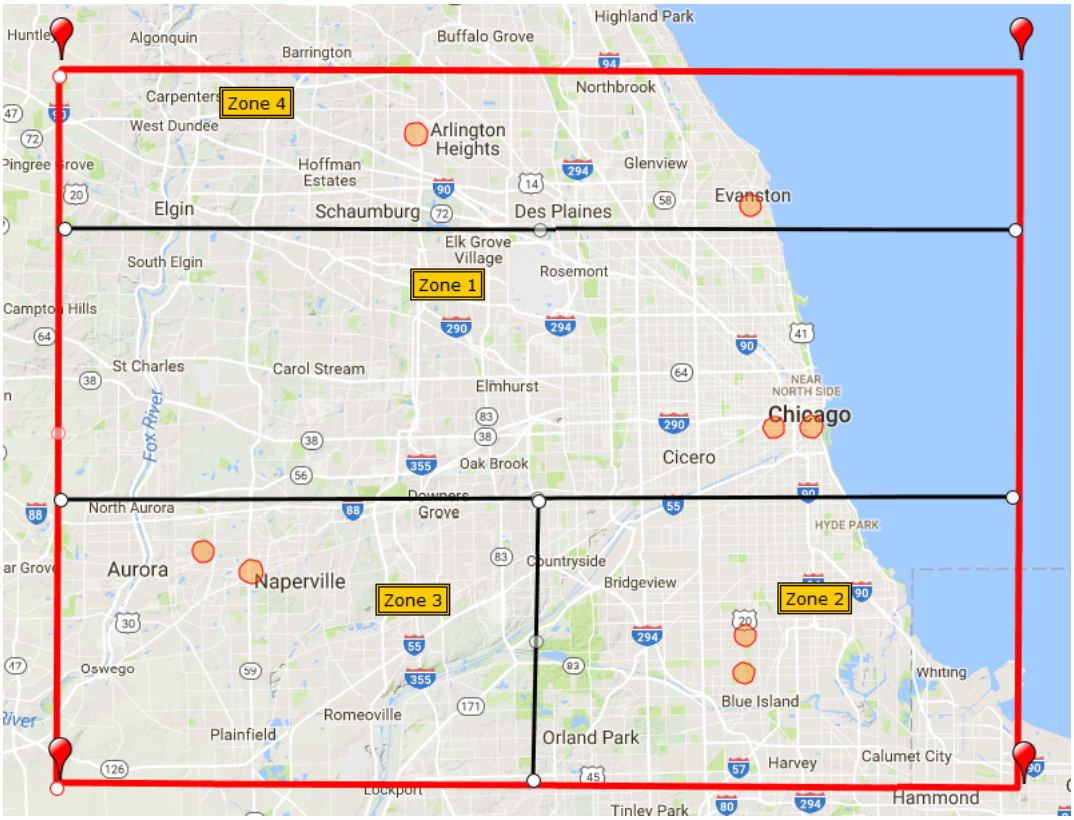


Figure 1: Zone division for Chicagoland Area

flight information. ADS-B equipment is currently mandatory in the Indian and Australian airspace and will become mandatory for all aircraft in the United States by 2020. It is also deployed and used in Europe and Canada. Hence, ADS-B is a technology on which we can depend for real-time aircraft information [4].

Figure 1 shows the Chicaholand area we are interested in, with the region further divided into 4 zones.

Figure 2 illustrates aircraft location data overlay over a 3-month period, with the blue dots signifying airports. It is evident from the plot that there are some distinct paths which are used by the major commercial aircraft while trying to landing or taking from O'Hare and Midway Airports. However, there are flights which have used paths other than the popular ones, but they are not so periodic. Moreover, these flights might be heading out/towards the smaller airports, and because their reduced frequency, they do not seem to look like popular flight paths. From this kind of overlay analysis, we can easily draw some rudimentary conclusions as to which places on the grid are likely to expect more traffic.

3. Data Overview

The data published in ADSBexchange.com. This website provides historical data starting from 20th June 2016. Here we analyze a 5 hour of flight traffic on a Friday evening in the Chicagoland area, as shown in Figure 1. The variables in the data are as follows:

- ICAO Id - The unique identifier of the aircraft. This is a six-digit hexadecimal identifier broadcast by the aircraft over the air in order to identify itself.
- Lat - The aircraft's latitude.
- Long- The aircraft's longitude.

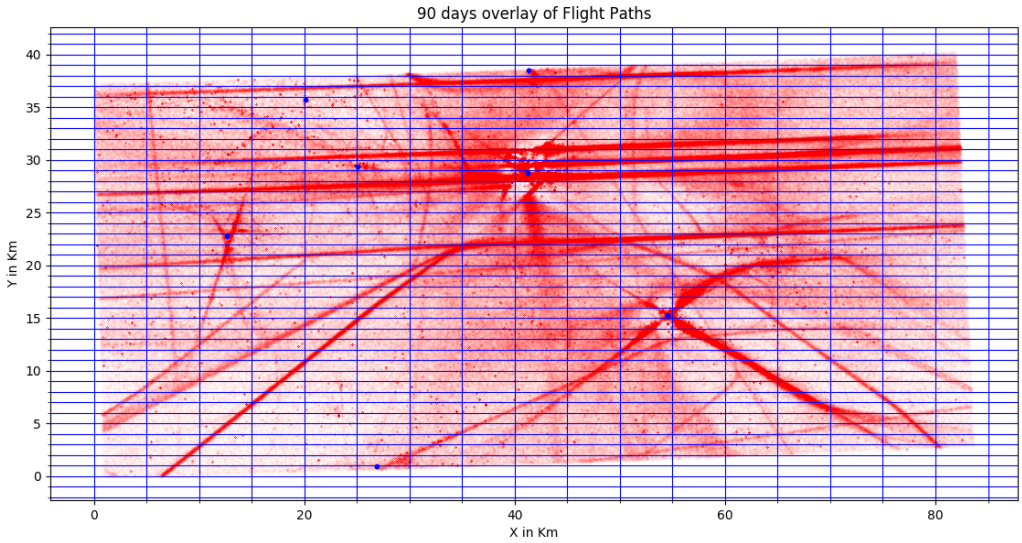


Figure 2: A 90 Day overlay of flight for the locations used by the flights in the given area

- Alt- The altitude in feet at standard pressure. This measurement is above the Mean Sea Level (MSL).
- Speed- The horizontal and vertical speeds of an aircraft at a given time instance. For the vertical speed a negative value suggests a decline in altitude and increment otherwise.
- Time-The time (at UTC in JavaScript ticks) that the position was last reported by the aircraft.

A Detailed extraction of the data is shown in Appendix I. The barplot in Appendix I shows the number of data points available for each of the flights in the data.

3.1. Independent Variable

The goal of this analysis is to predict the flight path based on the variables present in the data. For brevity, we try to predict only the altitude of a given flight. The main reason for choosing the altitude as the independent variable is the application of the radar during landing and taking off. An aircraft movement generally consists of landing (decrease in altitude), taking-off (increase in altitude) and cruising (in the air or taxing at the terminal). Figure 3 shows a snapshot of sample flights and their change in altitude with respect to time. Note the Time axis is a 5hr scale and a sample of the flights have been shown in this figure. The flight in brown is taking off and the flight in gold is landing. The flights in shades of blue are cruising and maybe just passing by the Chicagoland area. In the given time of 5hrs, a flight can perform both tasks of landing and take off, which will depend on the airport, as shown by the flight in green. We consider the two flight paths of landing and taking-off as separate flights and try to form a linear model for a given type of flight path.

Moreover, the radar has a physical constraint which we can use to our advantage to promote sharing with this band. Though the radar is kept all throughout the journey of the flight, it can only measure till $2,500$m. After that altitude, the aircraft relies on GPS and barometric methods to calculate the altitude.

The plot in Figure 3 further highlights that there is some inconsistency in the data with respect to time and may be due to a loss of transmission or errors. We do need to note that there are multiple receivers which receive this ADS-B data from the aircraft and report to a central database in real time. This might often lead to multiple entries for the same flight for the same time, so we had to remove all redundancies, as when in Appendix I.

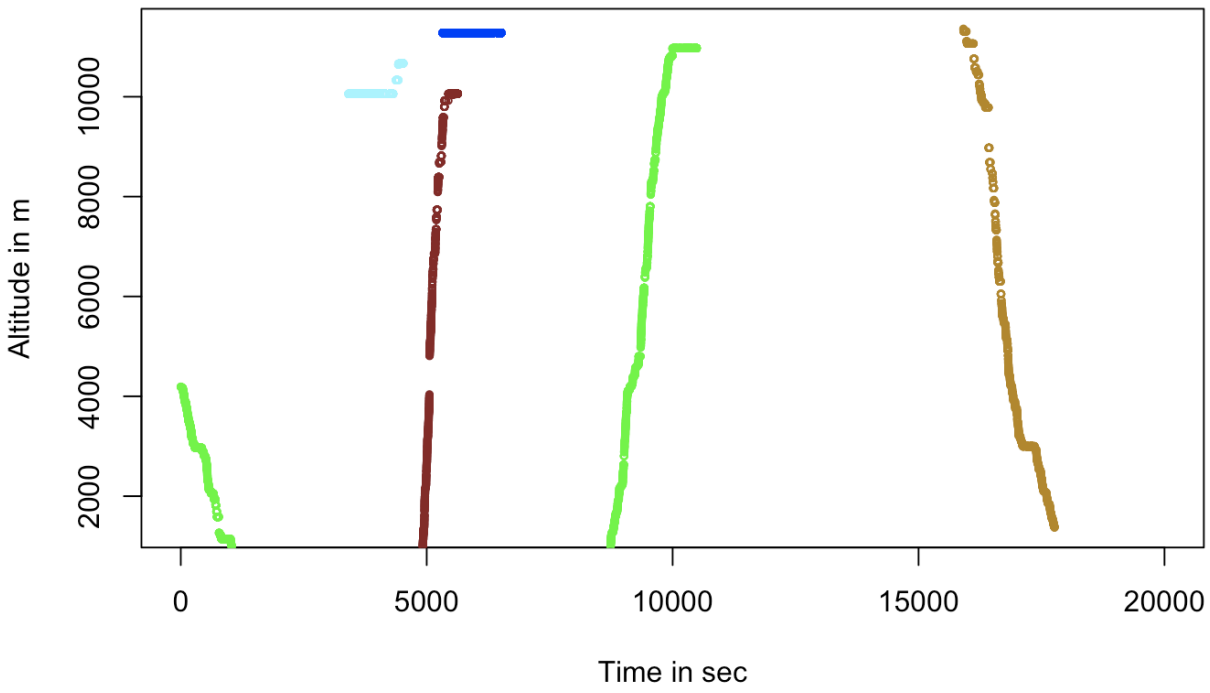


Figure 3: Change in Altitude of Sample Flights Present in the Chigaoland Area

The Q-Q plots for the altitude for the two types of flight paths landing and taking-off are shown in Figure 4 and 5 respectively (Code in Appendix I.D). The altitude distribution for both the figures are in line with Normal distribution at the middle, however, has pretty long tails. It represents the Cauchy Distribution and could be replicated by using its PDF function, however, for simplicity we rely on transformations tools learned (like Box-Cox) to try fit this distribution.

Though flight coordinates (latitude & longitude) are a factor while calculating the euclidean distance, however, the altitude is a physical constraint with the radar. Thus the altitude will help conclude if the flight is trying to land or take off, making the radar application crucial at that point and should be free from harmful interference.

3.2. Dependent Variable

Before we continue with our analysis we need to decide on the independent variables. However, these variables need further extraction and/or processing to be used in our model:

- Extracted Variables: Since the latitude and longitude do consider the curvature of the Earth, it is necessary that they are converted to coordinates by Earth-centered, Earth-fixed (ECEF) method [5]. Variables x and y corresponds to the coordinates of a flight at a given time.
- Multicollinearity: Multicollinearity happens when predictors in a regression are highly correlated with each other and will make it difficult to use transformation tools like GAM and Box-Cox. Often leaving out some variables form the model improves the R^2 value of a model, however, we risk omitted variable biases. So we combine the x and y variables into distance with respect to the origin; $dist = \sqrt{x^2 + y^2}$. We use the Variance Inflation (VIF) as a measure to check for multicollinearity, as shown in Appendix II.B and II.B.
- Lag Variables: We can argue that the altitude of a flight will be dependent on the state of the parameters in the previous time instance. Therefore, we compute the lag terms for the variables x, y, alt, dist,

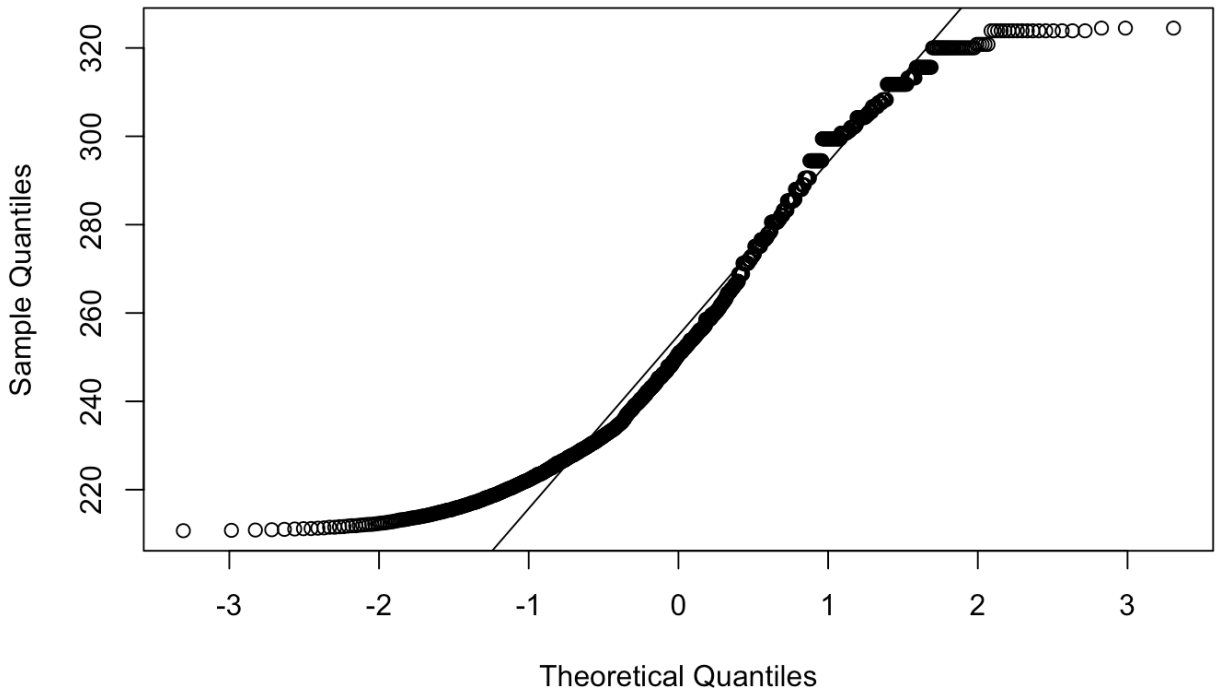


Figure 4: Normal QQ Plot for Altitude of a Flight Landing

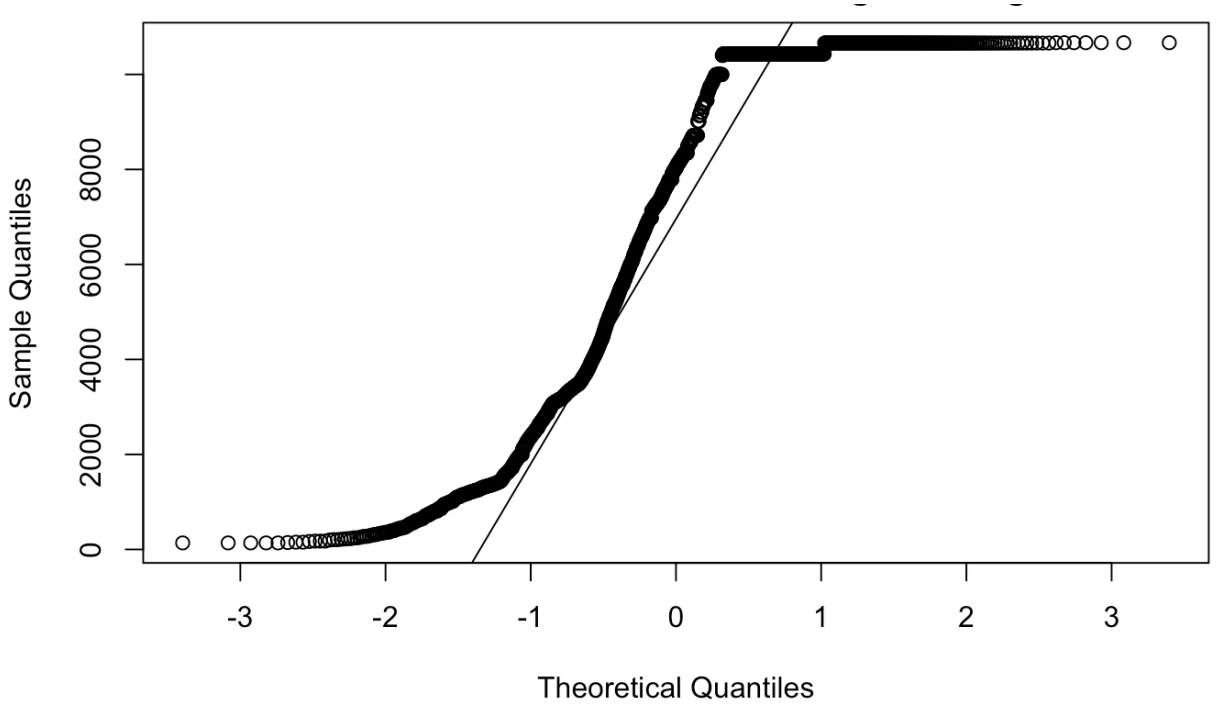


Figure 5: Normal QQ Plot for Altitude of a Flight Taking-off

horizontal speed and vertical speed represented as xlag, ylag, altlag, distlag, speedlag, vspeedlag respectively.

All the above-mentioned manipulations with the variables are shown in Appendix I.C

4. Proposed Models

A flight path can be affected due to geographic location, flight traffic, terminal availability, weather and other external factors. With so many factors the problem can easily get more and more complex. However, to simplify things we try to propose a general model for flights based on the path type (1) Landing or (2) Taking-off. Then use this model to show if it is able to predict for similar flights or not. The reason for selecting these two types is based on the common feature of drastic decrease or increase in altitude during these cases which have common patterns.

The basic parameters like the coordinates, distance from origin and speed are the ones which need to be accounted while trying to propose the models. The horizontal and the vertical speeds cannot be considered individually as the change in speed will also affect the coordinates, hence we consider the interaction terms $distlag * speedla$ and $altlag * vspeedlag$.

We try different combinations for the two types of flights and find the following three models to be promising (Code sown in Appendix II.D and III.D)

$$Model_1 : alt_i = \beta_0 + \beta_1 * x * y$$

$$Model_2 : alt_i = \beta_0 + \beta_1 * x * y + \beta_2 * distlag + \beta_3 * distlag * speedlag$$

$$Model_3 : alt_i = \beta_0 + \beta_1 * x * y + \beta_2 * altlag + \beta_3 distlag * speedlag + \beta_4 altlag * vspeedlag$$

In the next section we move on finding the tranforms for the model and their respective coefficients.

4.1. Individual Flight Model Analysis

In this section we go ahead with the 5-story of the data. Before we move ahead with the story telling of the data, we need to implement the $20 - 60 - 20$ rule and partition the data into 3 parts. The data is divided as FDeval, FDtrain and FDtest which has 20%, 60% and 20% of the data respectively.

4.1.1. Landing

The data summary suggests that all altitude and the coordinates show a Cauchy distribution with long tails. So transforms are required. The conditional story in Appendix II.D shows the Model I and II are pretty decent with few outliers outside the confidence bands, while Model III, which is a complex model, is not able to capture all the points within the confidence bands.

The Box-Cox transforms suggest a lambda of $0.34, 0.14, 0.94$ respectively, as shown in Appendix II.F. However, for the second model, we can approximate the lambda to be zero (since it is within the 95% confidence interval) and consider it a log transform. To get intuition on the independent variable transforms we see the GAM, as shown in Appendix II.G. The GAM fitted vs residual plots suggests that there is some heteroskedasticity with time, however, we are not able to explain the concrete reason for it. The partial residual plots for Model II and III suggests a non-linearity in distlag and altlag respectively. Thus, take a square transform of distlag and log transform for altlag in the respective models.

	Estimate	Std. Error
(Intercept)	2402.3524794	850.8503487
x	-2.9594169	0.3905169
y	0.4992678	0.1710146
x:y	-0.0006188	0.0000662

Figure 6: Coefficient for Model I

	Estimate	Std. Error
(Intercept)	-8644.1135798	2553.3667558
x	0.8622913	0.5672824
y	-0.1625737	0.0470423
I(distlag^2)	-0.0003602	0.0001091
distlag	3.3655843	1.0097265
speedlag	0.8557319	0.0534254
x:y	0.0001572	0.0001203
distlag:speedlag	-0.0001844	0.0000114

Figure 7: Coefficient for Model II

Final transformed Equations are:

$$\frac{alt_i^{0.34} - 1}{0.34} = \beta_0 + \beta_1 * x * y$$

$$log(alt_i) = \beta_0 + \beta_1 * x * y + \beta_2 * distlag^2 + \beta_3 * distlag * speedlag$$

$$\frac{alt_i^{0.9} - 1}{0.9} = \beta_0 + \beta_1 * x * y + \beta_2 * log(altlag) + \beta_3 distlag * speedlag + \beta_4 altlag * vspeedlag$$

The tables in Figures 6-8 show the cooeficients for the 3 models.

4.1.2. Taking-off

Similar to the data summary of landing flight, the altitude and the coordinates show a Cauchy distribution with long tails. So transforms are required. The conditional story in Appendix III.D shows the Model I and II are pretty decent with few outliers outside the confidence bands, while Model III, which is a complex model, is not able to capture all the points within the confidence bands. Model III is worse than it was in the landing flight, yet, we move on with this model.

	Estimate	Std. Error
(Intercept)	10270.3498979	1.894392e+04
x	-10.3715832	3.396211e+01
y	35.0749400	4.514156e+00
log(altlag)	135.2269993	3.105622e+01
distlag	32.8471967	2.729116e+00
speedlag	-27.2944905	3.319268e+01
altlag	0.6260487	5.033900e-03
vspeedlag	1.3760624	7.937240e-01
x:y	-0.0013682	7.691700e-03
distlag:speedlag	0.0059280	7.112600e-03
altlag:vspeedlag	-0.0001328	1.079000e-04

Figure 8: Coefficient for Model III

The Box-Cox transforms suggest a lambda of $0.9, 1.2, 1$ respectively, as shown in Appendix III.F. These transforms are different from the ones we obtained in the landing flight. This gives a hint that the landing and taking-off patterns are generally different. This due to the reason that it is easier to land and flight than take it off. For taking off a substantial amount of thrust is required, coupled with temperature and wind speed. Therefore, the patterns will be different and hence the different transforms.

To get intuition on the independent variable transforms we see the GAM, as shown in Appendix III.G. The GAM fitted vs residual plots suggests that compared to landing flight there isn't much heteroskedasticity with time. The partial residual plots for Model II linear distributions, thus no transforms are recommended. While for the Model III altlag shows a need for a square transform.

Final transformed Equations are:

$$\frac{alt_i^{0.9} - 1}{0.9} = \beta_0 + \beta_1 * x * y$$

$$\frac{alt_i^{1.2} - 1}{1.2} = \beta_0 + \beta_1 * x * y + \beta_2 * distlag + \beta_3 * distlag * speedlag$$

$$alt_i - 1 = \beta_0 + \beta_1 * x * y + \beta_2 * altlag^2 + \beta_3 distlag * speedlag + \beta_4 altlag * vspeedlag$$

The tables in Figures 9-11 show the cooeficients for the 3 models.

4.1.3. Forcasting

The Forecasting Story shows that the MSE for Model II is the least, as shown in Appendix II.I for the landing flight type. Even while testing for a different flight (also of type landing) shows Model II to have the least MSE. Even though Model III is more complex than others, Model I and II proves to be a better predictor

	Estimate	Std. Error
(Intercept)	4.217677e+05	3630.2046224
x	-7.315724e+02	3.3834419
y	8.834004e+01	0.7389676
x:y	-1.508582e-01	0.0007715

Figure 9: Coefficient for Model I

	Estimate	Std. Error
(Intercept)	6.449099e+06	1.488344e+05
x	-1.168684e+04	2.471388e+02
y	7.416612e+02	3.338954e+02
distlag	-6.116891e+02	3.271865e+02
speedlag	-1.942596e+02	3.748917e+02
x:y	-2.434789e+00	4.981760e-02
distlag:speedlag	2.403040e-02	7.879390e-02

Figure 10: Coefficient for Model II

	Estimate	Std. Error
(Intercept)	4577.5157136	9048.1876222
x	-34.5307269	15.1200692
y	43.7034625	6.4036691
I(altlag^2)	-0.0000007	0.0000004
distlag	42.7582891	6.3711553
speedlag	14.8836125	9.7595002
altlag	1.0048751	0.0065452
vspeedlag	1.3990636	0.4845524
x:y	-0.0070168	0.0031276
distlag:speedlag	-0.0031783	0.0020619
altlag:vspeedlag	-0.0001290	0.0000618

Figure 11: Coefficient for Model III

of the data. This is quite evident since the change in altitude is highly correlated with the location of the airports and their airspace. Larger airports have the larger airspace and will have different a particular style of decreasing or increasing the altitude.

On the contrary, the forecasting story for taking-off flight suggests that Model III works best even for different flight (also of type taking off), as shown in Appendix III.I. This suggests that the altitude and the vertical speed is critical in taking off.

4.2 Data Pooling

Now we try to predict the intercepts for the complete, no and partial pooling. The summary of the regressions is shown in Appendix IV. Note that for no pooling the intercepts will be subjected to the icao_addr, so the mean estimate has been shown. All the intercepts are pretty close to the mean of 8500m, which says that the altitudes are centered around the mean. The no-pooling is the closest to 8200m value, however, it is the mean for all the school, so it doesn't make sense. However, the complete pooling compared to the partial pooling shows a higher inclination towards 8500m.

5. Conclusion

From our preliminary analysis of the two flight types namely landing and taking-off, we see that these flight types are unique and can be described by a specific model. Model II which considers the x-y coordinates and horizontal speed proves to work better for the Landing type flights, which is evident as the flights approaching the airport, the rate of decrease in altitude is governed by the airspace allocation for an airport. For detail on an airport and its airspace refer Appendix V. On the contrary for taking off flights the Model III proves to be the best, which considers the x-y coordinates, lagged altitude, and lagged horizontal and vertical speeds. As explained earlier taking off a flight is a complex task and requires a particular amount of thrust which has to

be coupled with good temperature and wind speed. Moreover, due to unavailability of terminals, a flight destined for traveling (say) North might have to take off from a South facing terminal, and cruise in the air for a while to set its correct destination. To account these aspects more data is required like the angles of the flight and weather data.

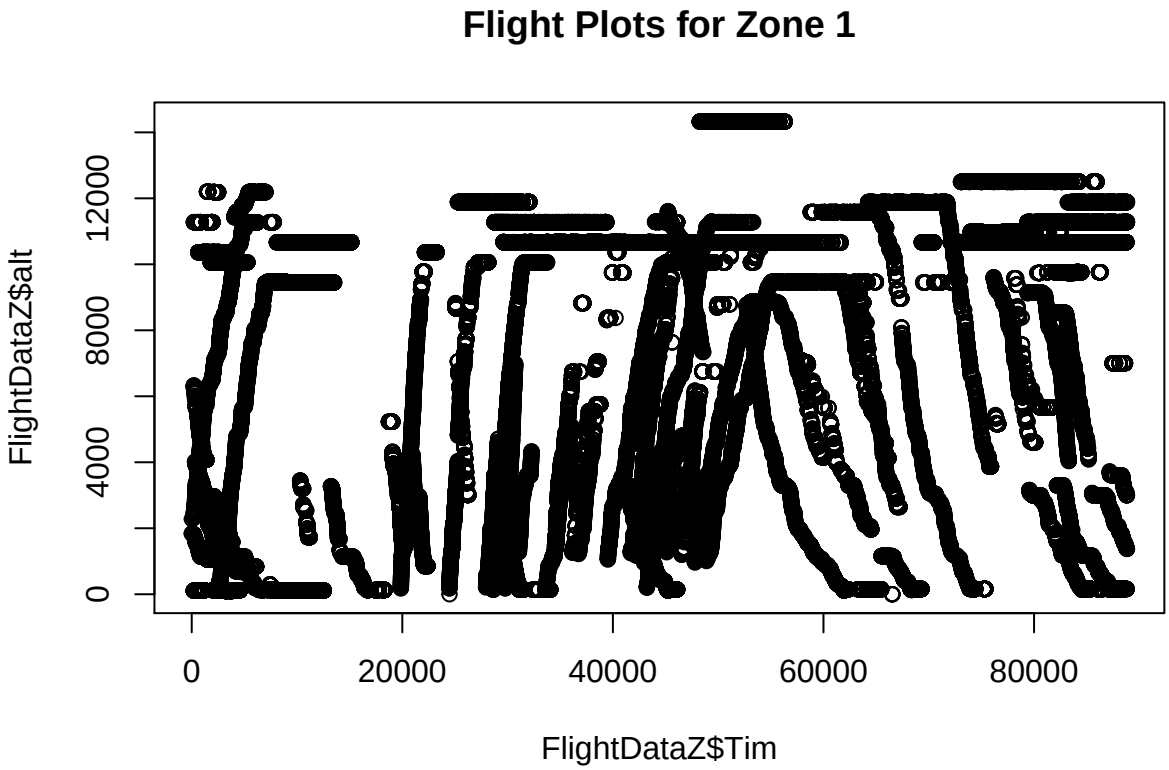
While talking about causality and omitted variable bias, factors like other flight traffic, demand for flights during peak hours/days and weather can play a role while predicting flight patterns. Out of the aforementioned factors, weather can be the critical one, as it controls if a flight will take off or land at an airport at a given time. In case of bad weather, many flights get canceled. Temperature and humidity combined to play an important role in providing thrust, increased climb rate and more lift to the wings. As temperature and altitude change the performance of an aircraft decreases. We feel that weather could have the correlation with the speed of the aircraft, which can alter our model, and maybe make our more precise.

Some limitations of this work, which could not be completed due to lack of data and time, are altitude distribution, dealing with time variant heteroskedasticity and partial pooling the data based on zones/flights. A future work for trying to fit the variable in a Cauchy distribution followed by multilevel regression can be done.

Appendix I

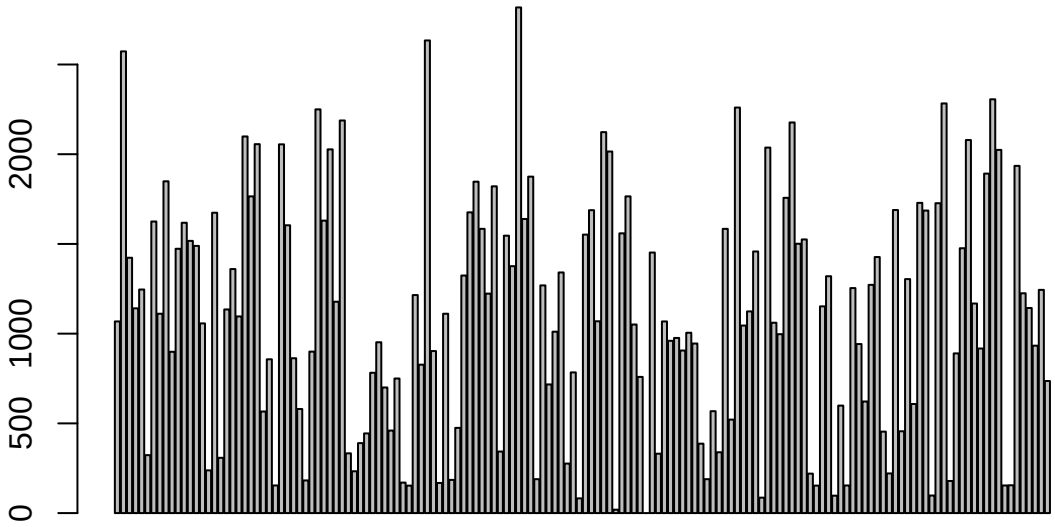
A. General Code for Whole data

```
# load data
load("Flightdata.Rda")

# Remove a outlier flight
f <- as.character("a6ec56")
FlightData <- FlightData[!FlightData$icao_addr==f,]

# Convert Epoch Ticks to timeintervals
FlightData$Tim <- as.numeric(FlightData$time) - as.numeric(FlightData$time[1])

# Flight Plots for Zone 1
FlightDataZ <- FlightData[FlightData$lat>41.5 & FlightData$lat<42.5,]
plot(FlightDataZ$Tim,FlightDataZ$alt, main = "Flight Plots for Zone 1")
```



B. Flight Count

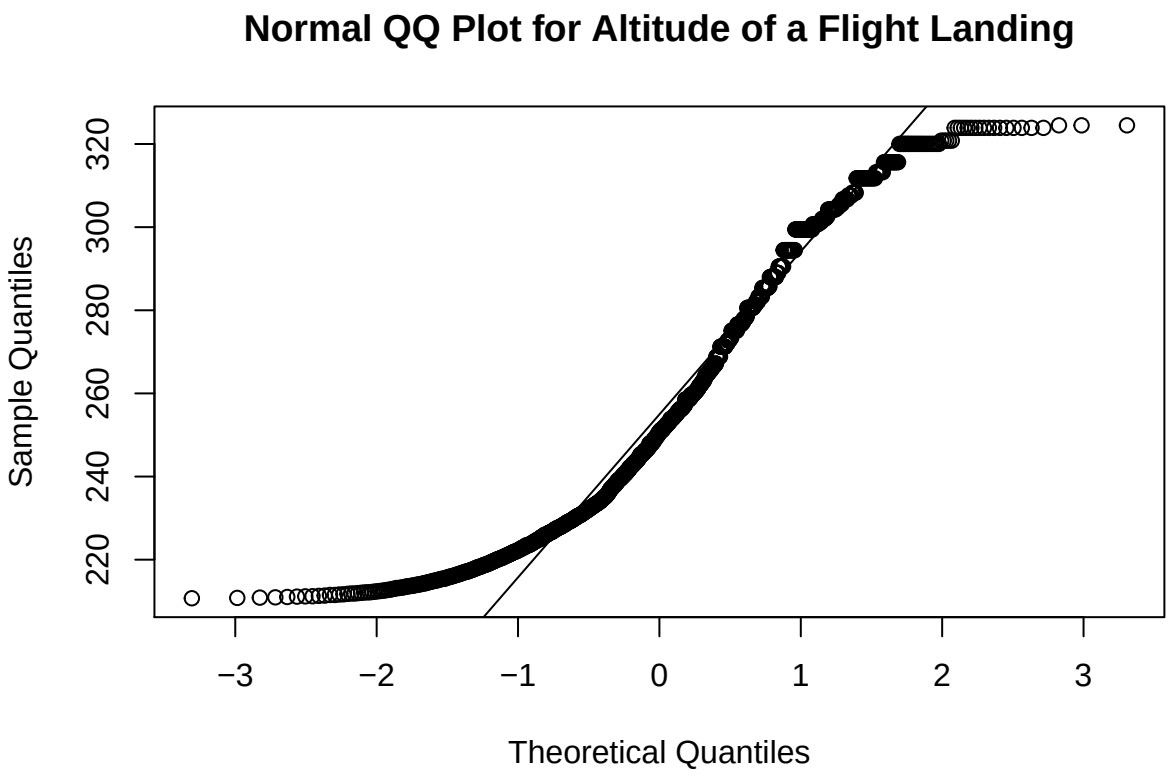
```
FID <- unique(FlightData$icao_addr)
FCount <- data.frame(table(FlightData$icao_addr))
#FCount <- FCount[order(FCount[,2]),]
#f <- as.character(FCount[1,1])

save(FCount,file="Flightdata-count.Rda")

barplot(FCount[,2],
     main   = "Histogram of Flight Count",
     xlab = "Sales (packs per capita)")
```



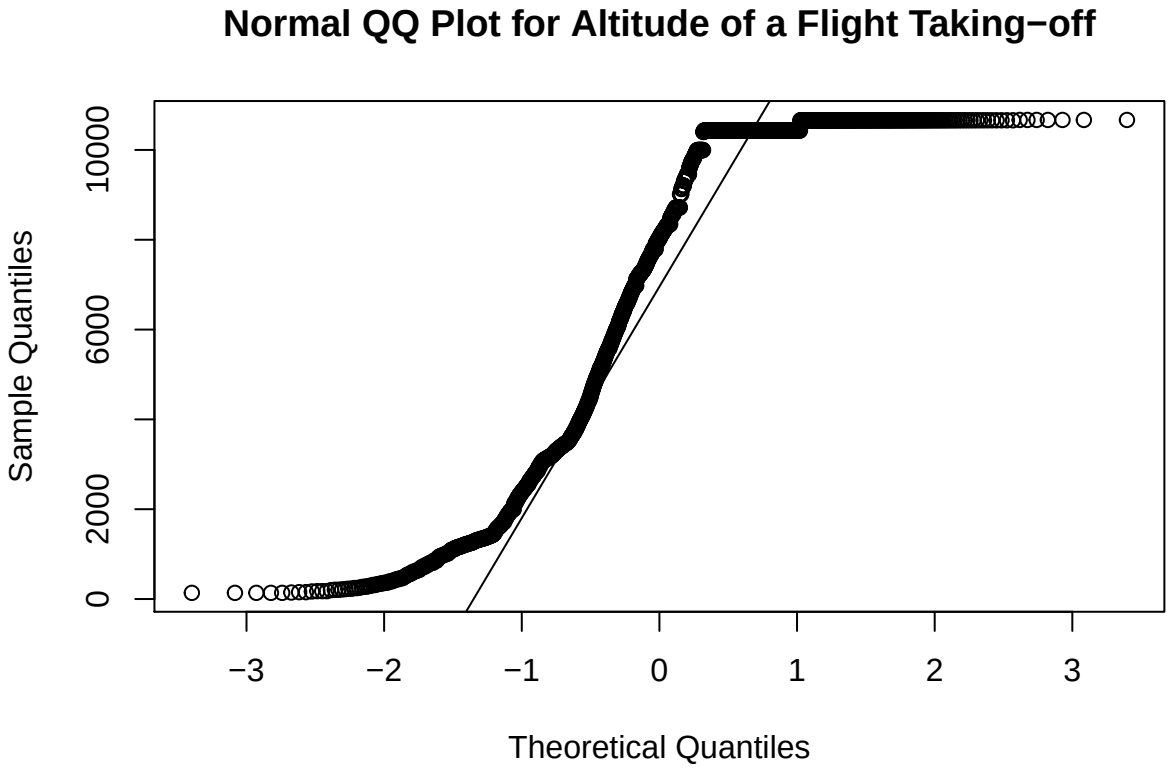
C. Exrtract Individual Flight

D. Exrtract Individual Flight

```
#QQ Plot for altitude
load("Flightdata-flight1.Rda")
qqnorm(fpoints$xlag, main = "Normal QQ Plot for Altitude of a Flight Landing")
qqline(fpoints$xlag)
```




```
load("Flightdata-flight11.Rda")
qqnorm(fpoints$alt, main = "Normal QQ Plot for Altitude of a Flight Taking-off")
qqline(fpoints$alt)
```



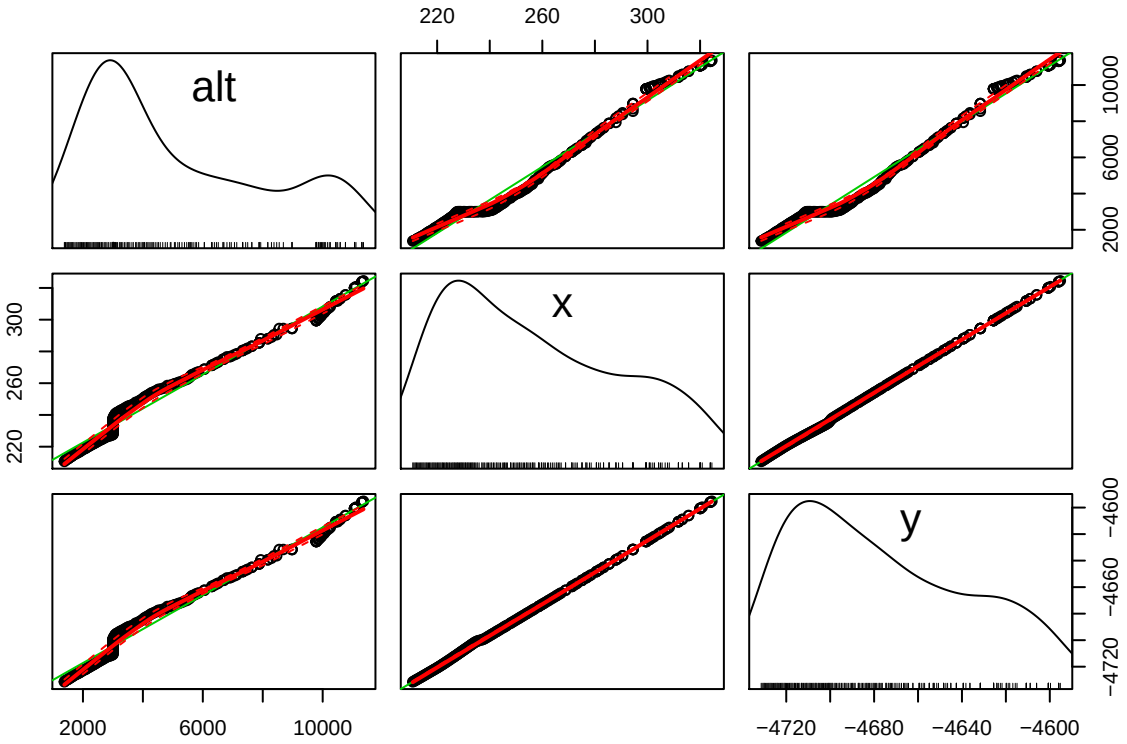
Appendix II

A. Data Partition for a Landing Flight

```
load("Flightdata-flight1.Rda")

row.nos <- rep(1:5, length.out = nrow(fpoints))
#row.nos <- sample(row.nos)

# Create eval and training and test cases
FDeval <- fpoints[row.nos == 1, ]             #20% of Data for Evaluation

FDtrain <- fpoints[row.nos != 1 & row.nos != 5,]  #60% of Data for Training

FDtest <- fpoints[row.nos == 5, ]             #20% of Data for Testing

#############################################################################################

FDtrain.for.Model <- fpoints[row.nos != 5,]  #80% of Data for Model Training.
                                  #This basically a combination on the first 20% and 60%
```

B. Multicolinearity

```
scatterplotMatrix(~alt + x + y ,data = FDeval)
```




```
lreg.f1.s2 <- lm(alt ~ x + y + xlag*speedlag + ylag*speedlag, data = FDeval)
vif(lreg.f1.s2)
```

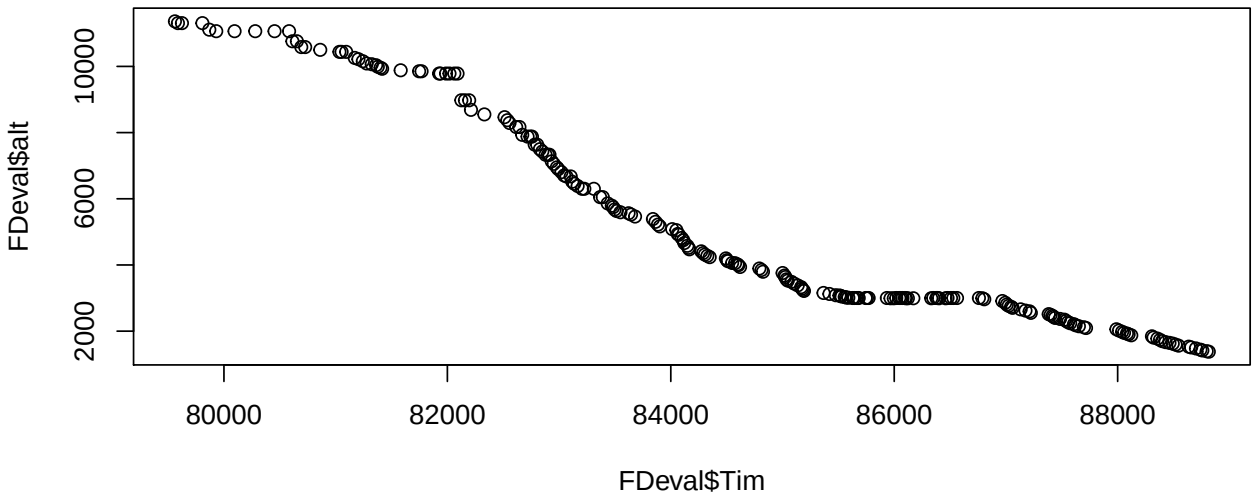
```
##             x             y          xlag      speedlag          ylag
##       4108527       4014062       4907190     725950874       4837885
## xlag:speedlag speedlag:ylag
##       8028122     583165738
```

```
lreg.f1.s2 <- lm(alt ~ x + y + distlag*speedlag, data = FDeval)
vif(lreg.f1.s2)
```

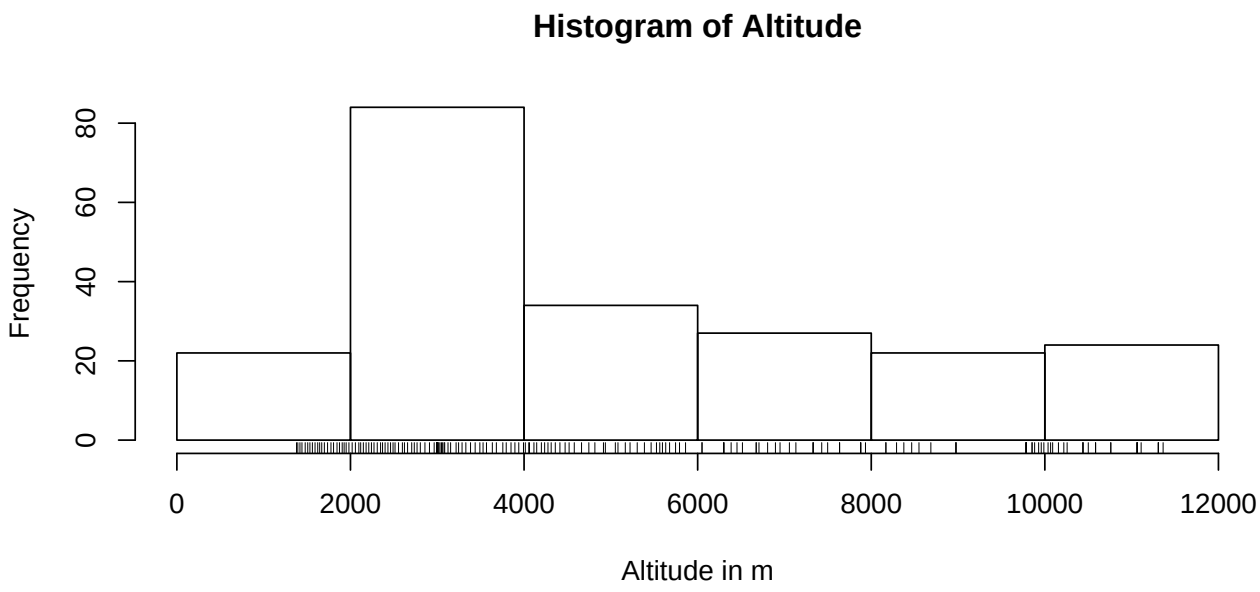
```
##                 x                 y          distlag         speedlag
##          15202.19          27828.17          11119.92          36683.27
## distlag:speedlag
##          32615.09
```

C. Data Summary

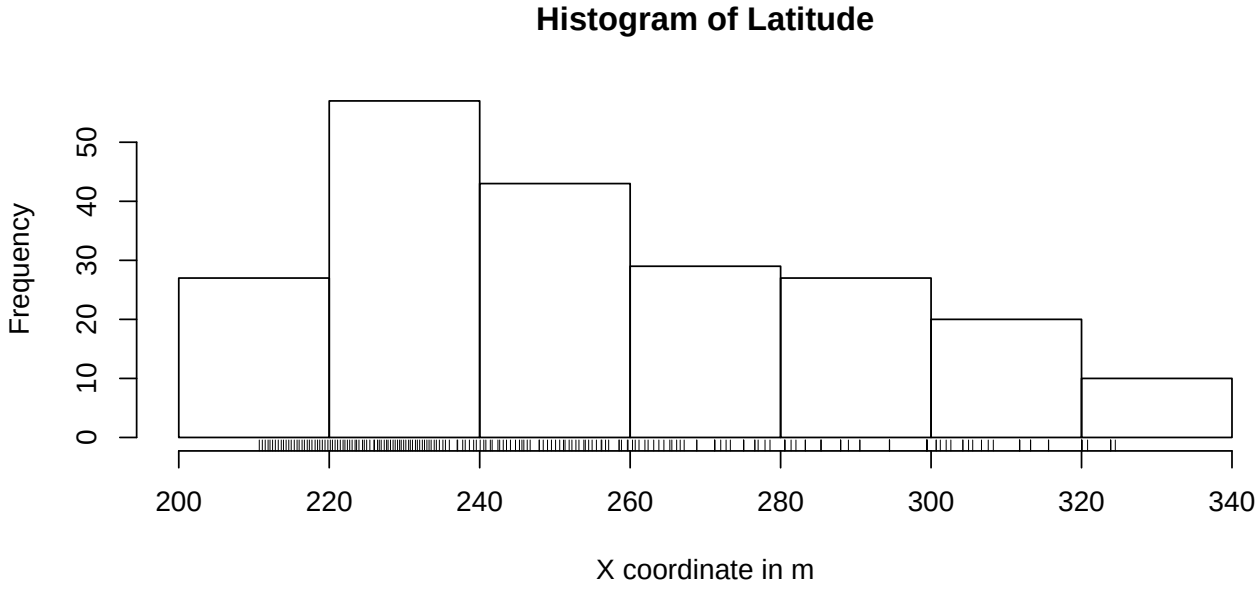
```
plot(FDeval$Tim,FDeval$alt)
```



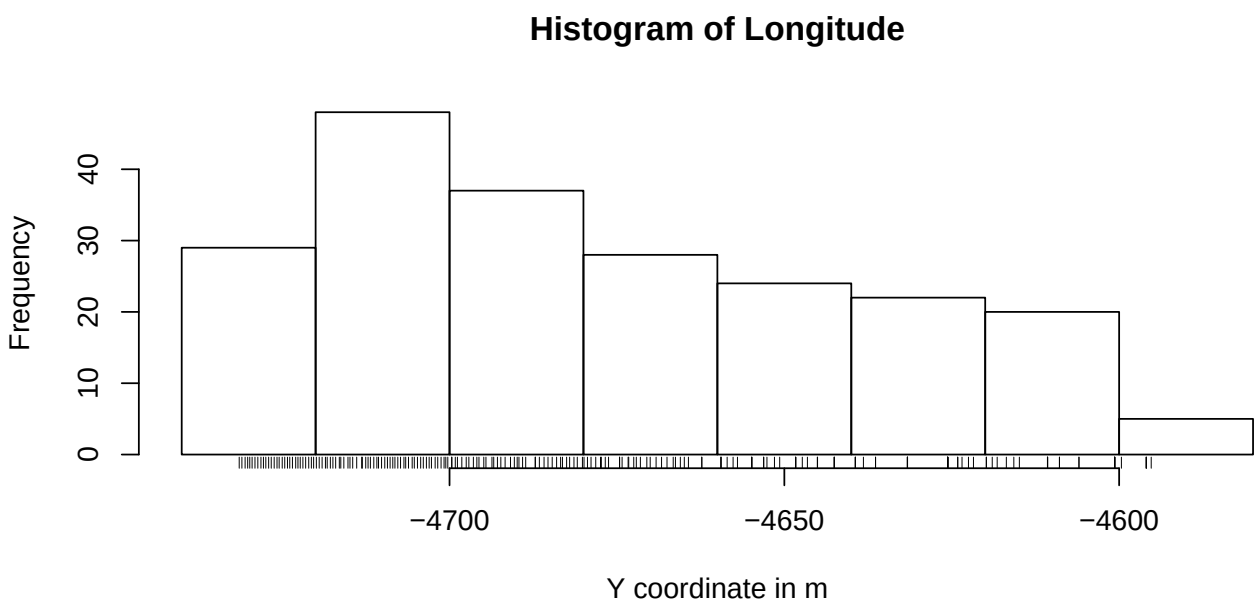

```
hist(FDeval$alt, breaks='FD',
     #xlim   = c(10000,150000),
     #ylim   = c(0,25),
     main   = "Histogram of Altitude",
     xlab = "Altitude in m")
rug(jitter(FDeval$alt))
```



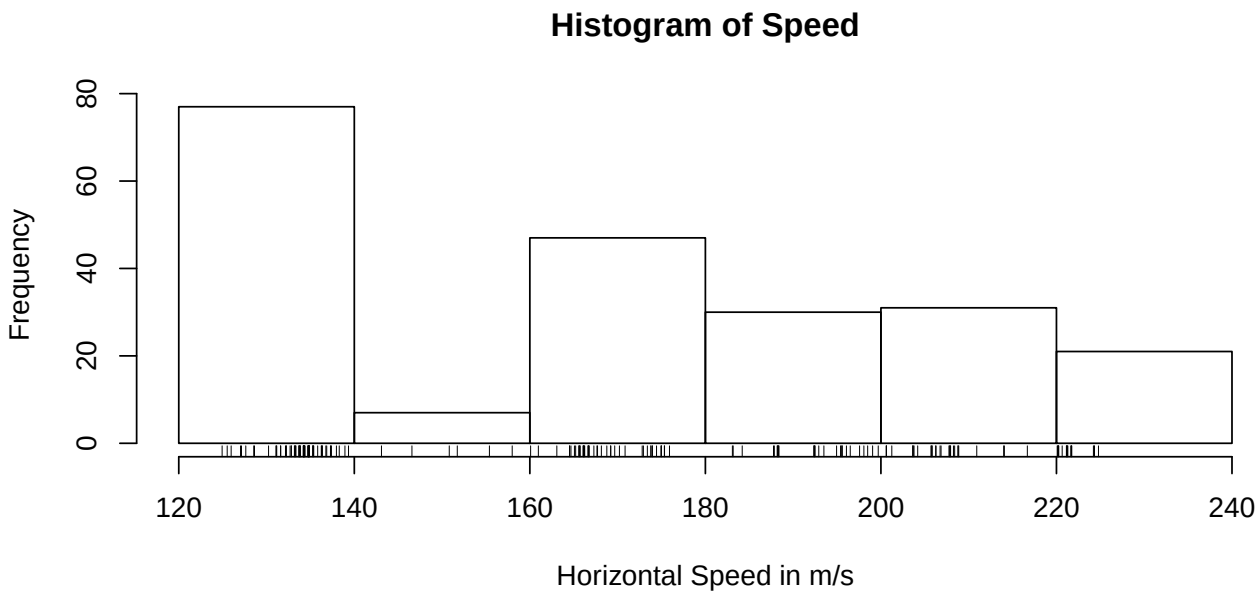

```
hist(FDeval$x, breaks='FD',
     #xlim   = c(10000,150000),
     #ylim   = c(0,25),
     main   = "Histogram of Latitude",
     xlab = "X coordinate in m")
rug(jitter(FDeval$x))
```



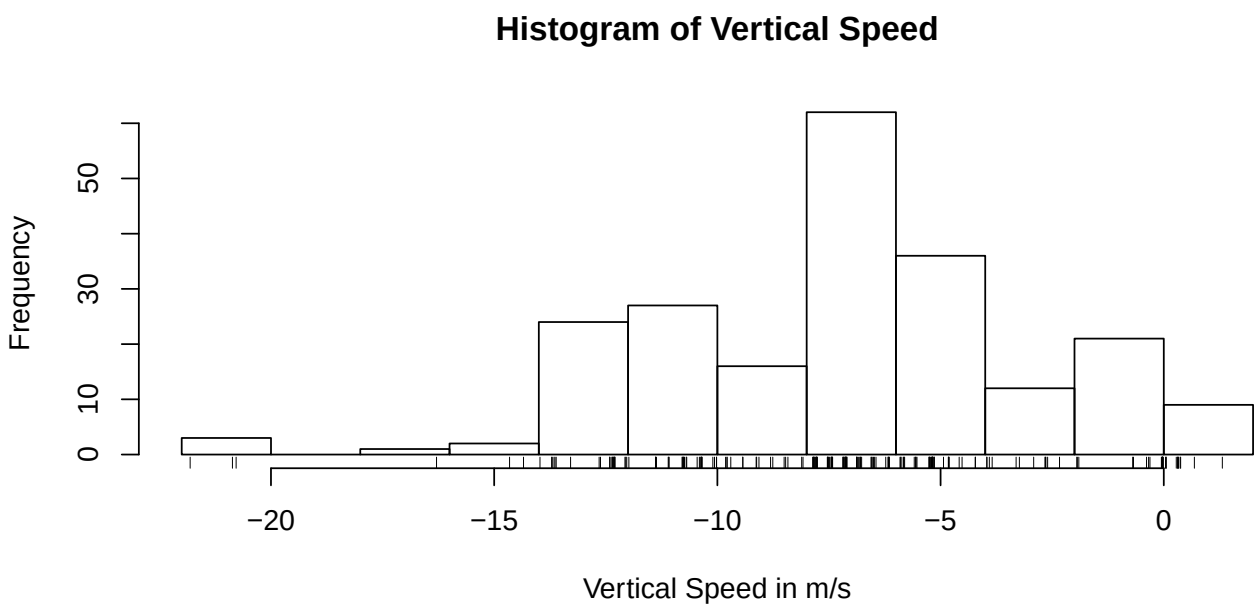

```
#Tranform y points to positive
fpoints$y <- fpoints$y-min(fpoints$y) + 1
fpoints$ylag <- Lag(fpoints$y, shift=1)
hist(FDeval$y, breaks='FD',
     #xlim   = c(10000,150000),
     #ylim   = c(0,25),
     main   = "Histogram of Longitude",
     xlab = "Y coordinate in m")
rug(jitter(FDeval$y))
```




```
hist(FDeval$speed, breaks='FD',
     #xlim   = c(10000,150000),
     #ylim   = c(0,25),
     main   = "Histogram of Speed",
     xlab = "Horizontal Speed in m/s")
rug(jitter(FDeval$speed))
```




```
#fpoints$vert_speed <- fpoints$vert_speed-min(fpoints$vert_speed) + 1
#fpoints$vspeedlag <- Lag(fpoints$vert_speed, shift=1)
hist(FDeval$vert_speed, breaks='FD',
     #xlim   = c(10000,150000),
     #ylim   = c(0,25),
     main   = "Histogram of Vertical Speed",
     xlab = "Vertical Speed in m/s")
rug(jitter(FDeval$vert_speed))
```



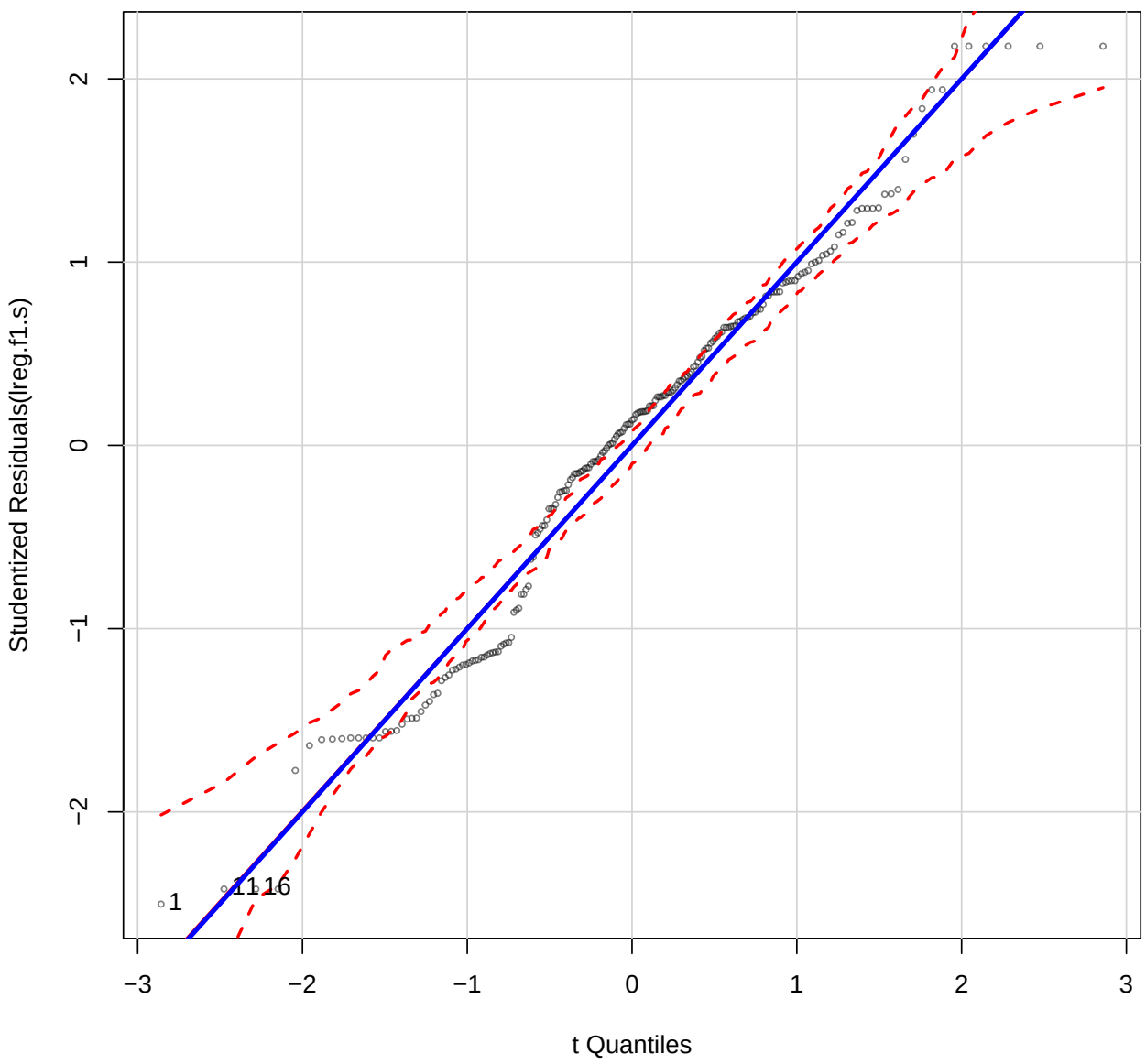
D. Conditional Story

To find the Independent variables and the respective interaction terms

```
#fpoints$TimT <- fpoints$Tim - min(fpoints$Tim)
lreg.f1.s <- lm(alt ~ x*y , data = FDeval)
lreg.f1.s2 <- lm(alt ~  x*y + distlag + distlag*speedlag, data = FDeval)

lreg.f1.c <- lm(alt ~ x*y + altlag, data = FDeval)
lreg.f1.c1 <- lm(alt ~ x*y + altlag*vspeedlag, data = FDeval)
lreg.f1.c2 <- lm(alt ~ x*y + dist*speedlag + altlag*vspeedlag, data = FDeval)
lreg.f1.sc <- lm(alt ~ x*y + altlag + distlag*speedlag + altlag*vspeedlag , data = FDeval)



#summary(lreg.f1.s)
#summary(lreg.f1.s2)
#summary(lreg.f1.sc)

########### QQPLOT
qqPlot(lreg.f1.s,
       main = "Flight regression residuals",
       pch  = 21,
       id.n = 3,
       cex = 0.5,
       col  = rgb(0, 0, 0, .5))
```

```
##  1 11 16
##  1  2  3
```

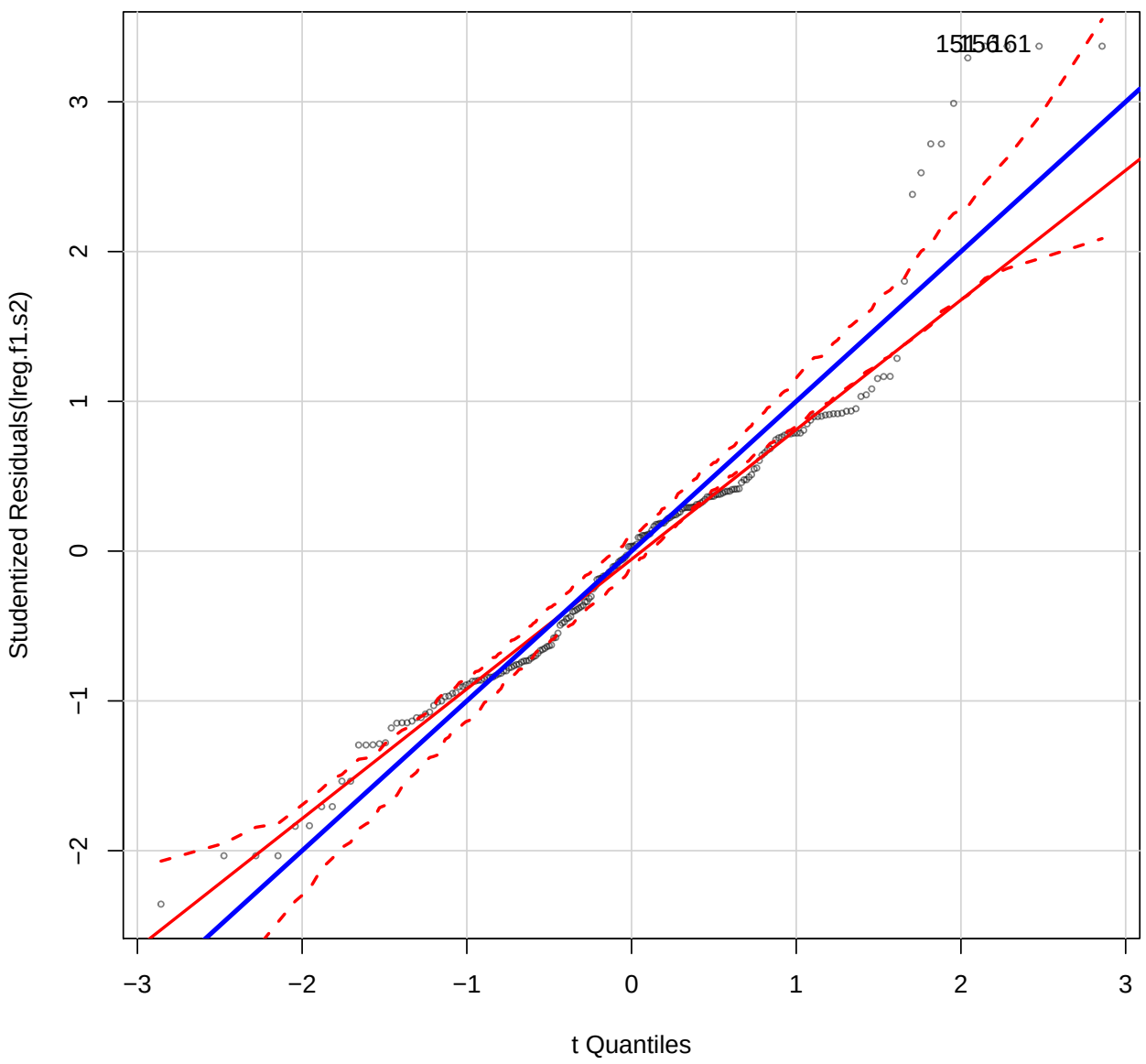
```
abline(a=0, b=1,col="blue",lwd=3)
```




```
qqPlot(lreg.f1.s2,
       main = "Flight regression residuals",
       pch  = 21,
       id.n = 3,
       cex = 0.5,
       col  = rgb(0, 0, 0, .5))
```

```
## 151 156 161
## 209 210 211
```

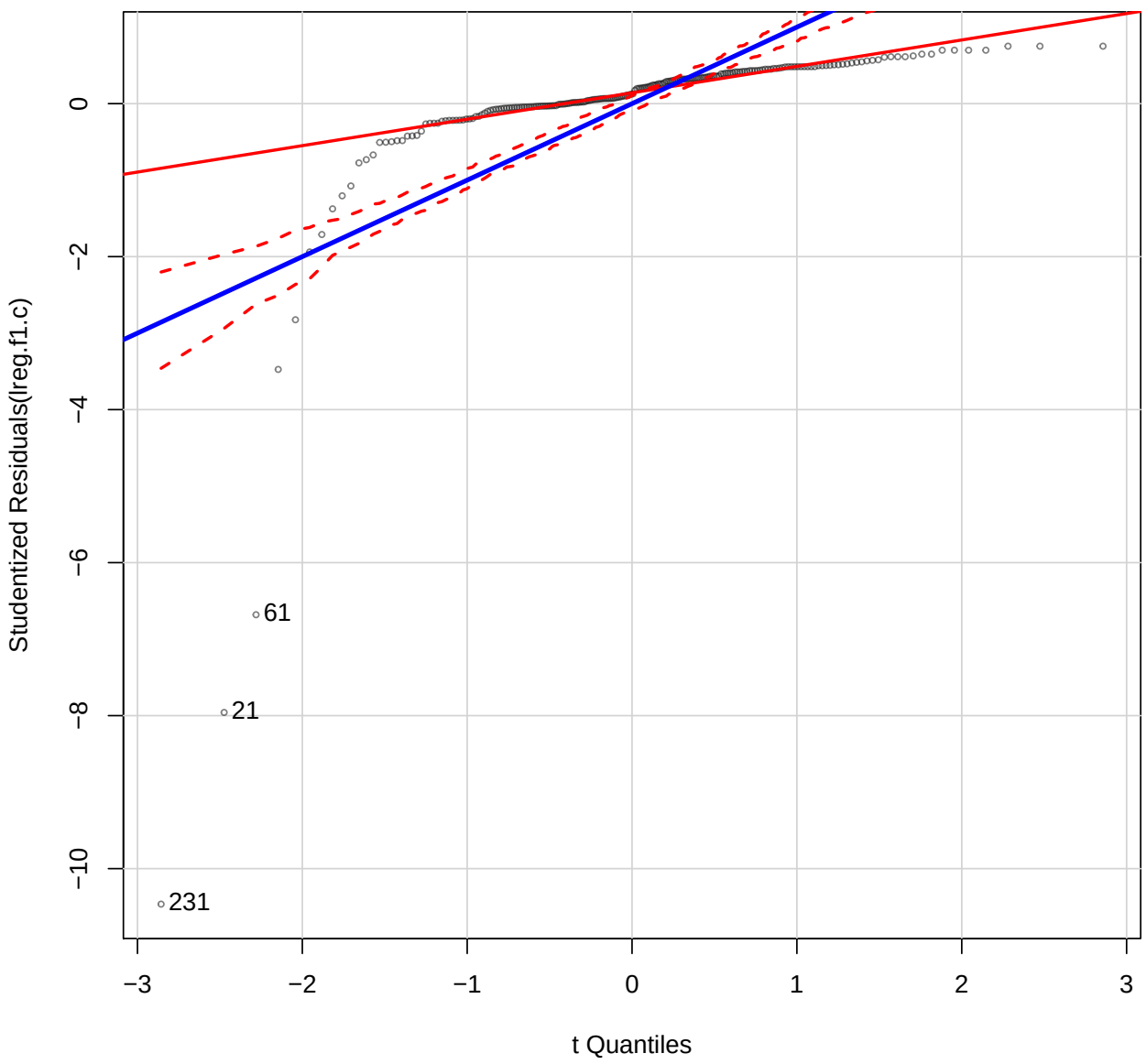
```
abline(a=0, b=1,col="blue",lwd=3)
```




```
qqPlot(lreg.f1.c,
       main = "Flight regression residuals",
       pch  = 21,
       id.n = 3,
       cex = 0.5,
       col  = rgb(0, 0, 0, .5))
```

```
## 231  21  61
##   1   2   3
```

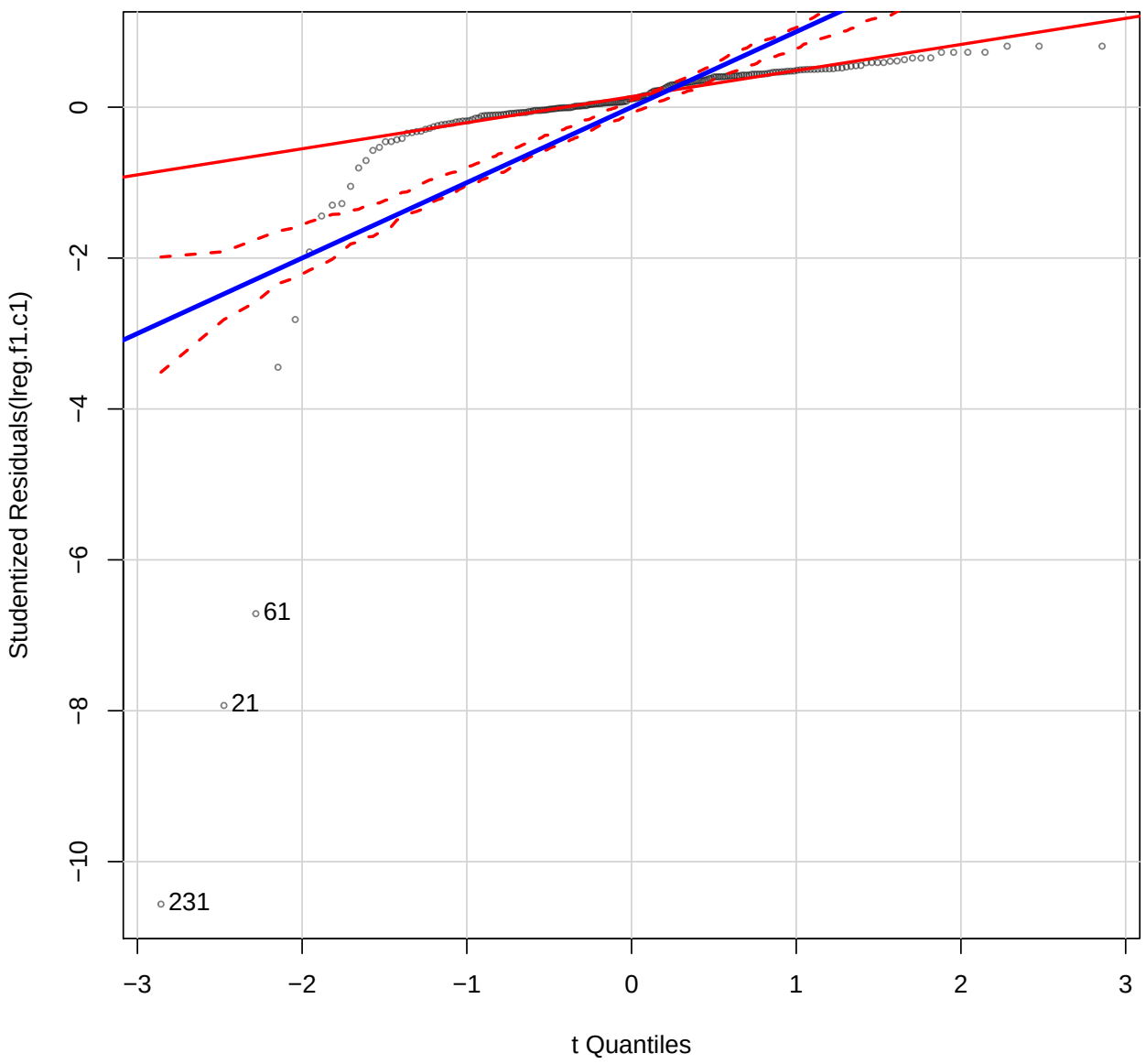
```
abline(a=0, b=1,col="blue",lwd=3)
```




```
qqPlot(lreg.f1.c1,
       main = "Flight regression residuals",
       pch  = 21,
       id.n = 3,
       cex = 0.5,
       col  = rgb(0, 0, 0, .5))
```

```
## 231  21  61
##   1   2   3
```

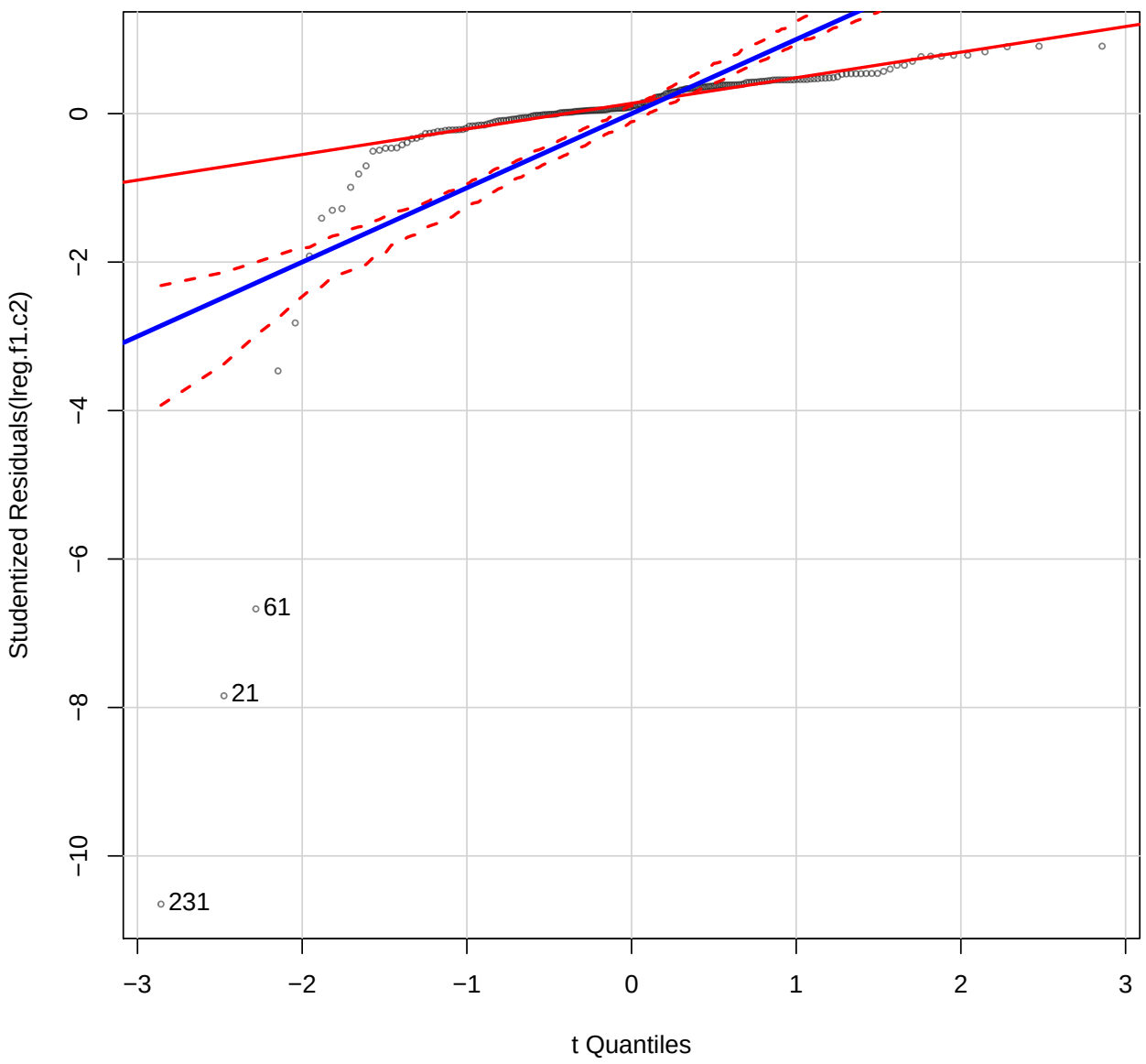
```
abline(a=0, b=1,col="blue",lwd=3)
```




```
qqPlot(lreg.f1.c2,
       main = "Flight regression residuals",
       pch  = 21,
       id.n = 3,
       cex = 0.5,
       col  = rgb(0, 0, 0, .5))
```

```
## 231  21  61
##   1   2   3
```

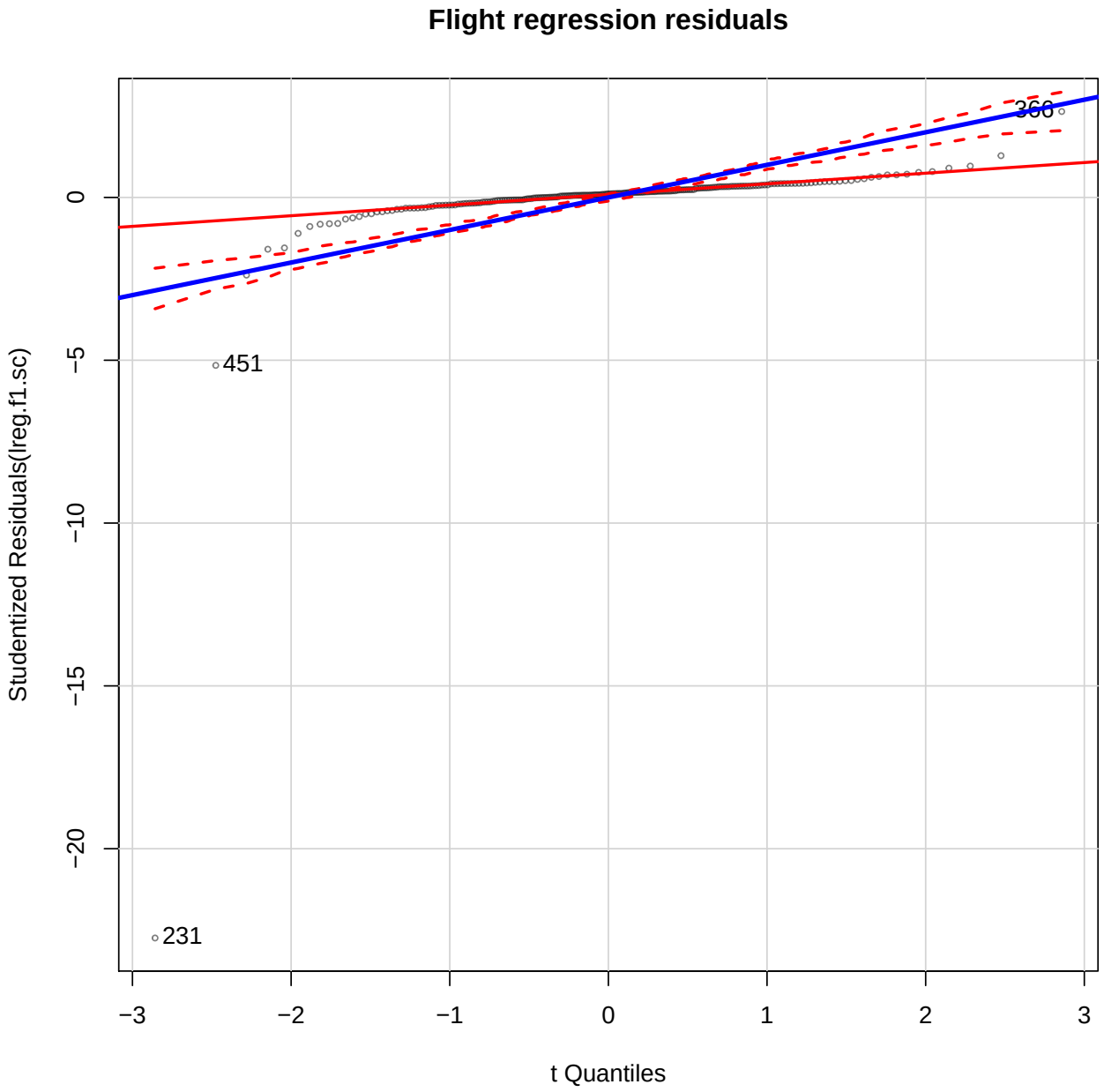
```
abline(a=0, b=1,col="blue",lwd=3)
```




```
qqPlot(lreg.f1.sc,
       main = "Flight regression residuals",
       pch  = 21,
       id.n = 3,
       cex = 0.5,
       col  = rgb(0, 0, 0, .5))
```

```
## 231 451 366
##   1   2 212
```

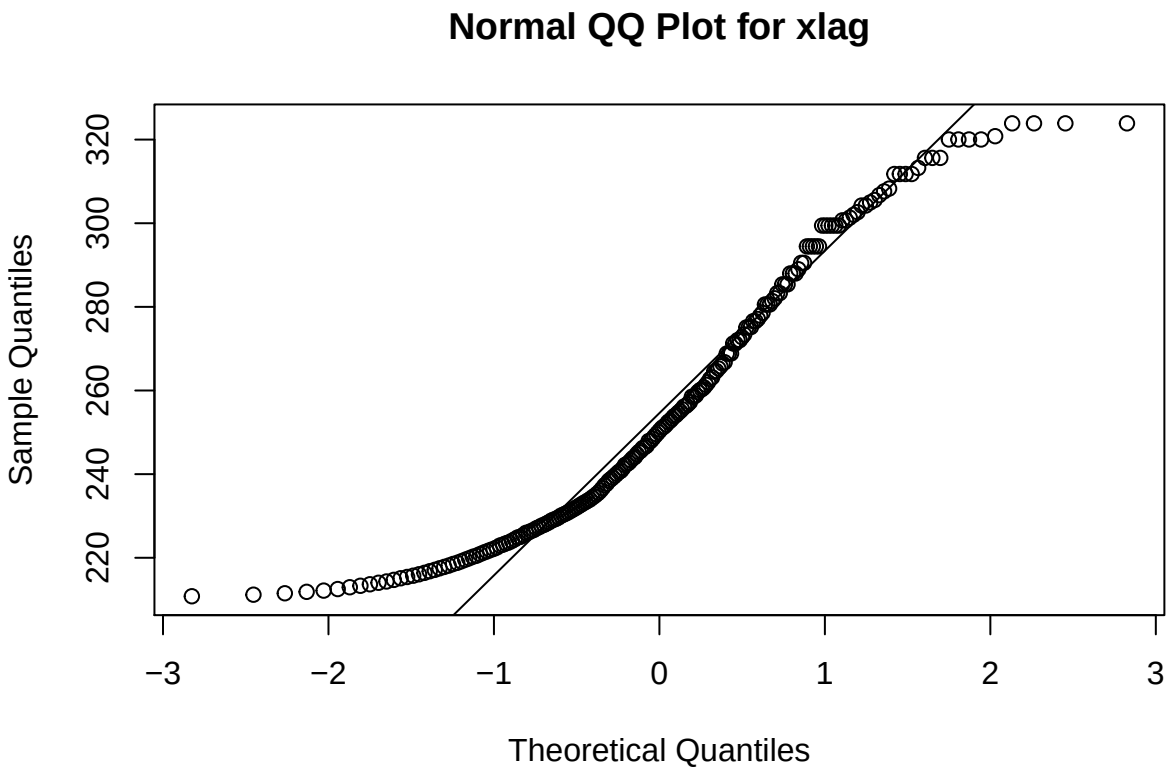
```
abline(a=0, b=1,col="blue",lwd=3)
```



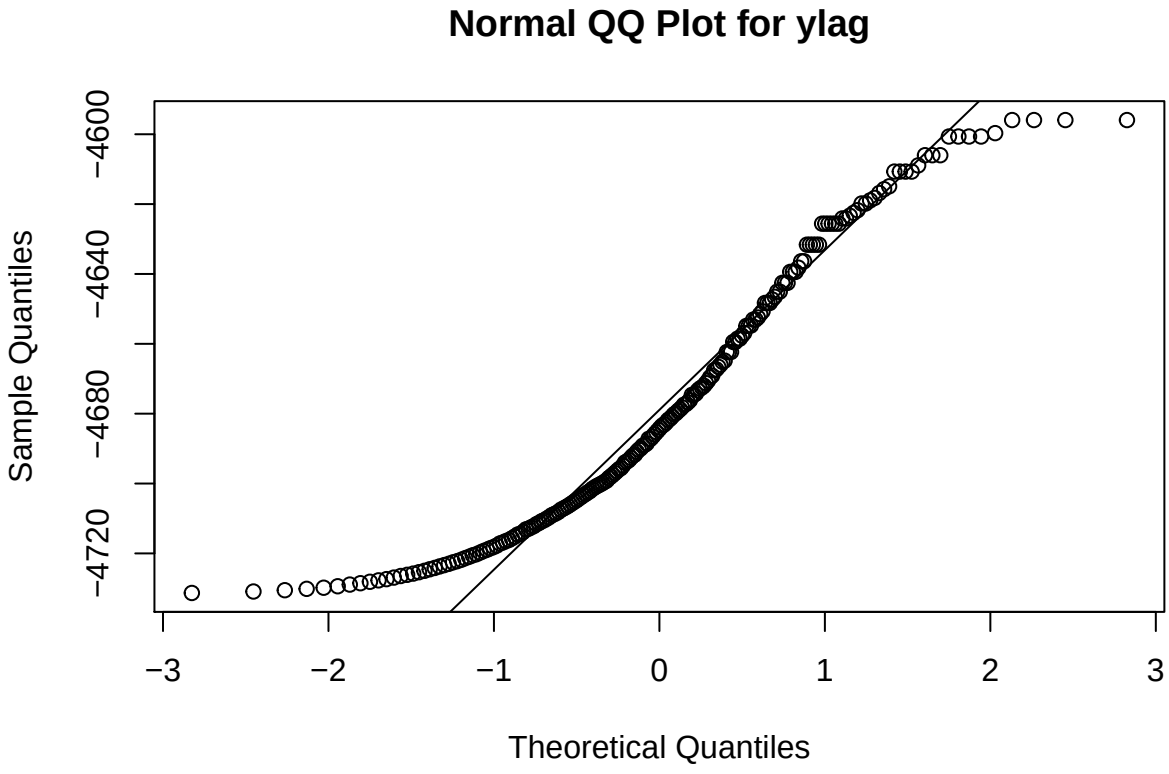

```
############ Influence Plot
#influenceIndexPlot(lreg.f1.s, id.n = 3)
#influenceIndexPlot(lreg.f1.s2, id.n = 3)
#influenceIndexPlot(lreg.f1.c, id.n = 3)
#influenceIndexPlot(lreg.f1.c1, id.n = 3)
#influenceIndexPlot(lreg.f1.c2, id.n = 3)
#influenceIndexPlot(lreg.f1.sc, id.n = 3)
```

E. QQPlot for Lag Terms

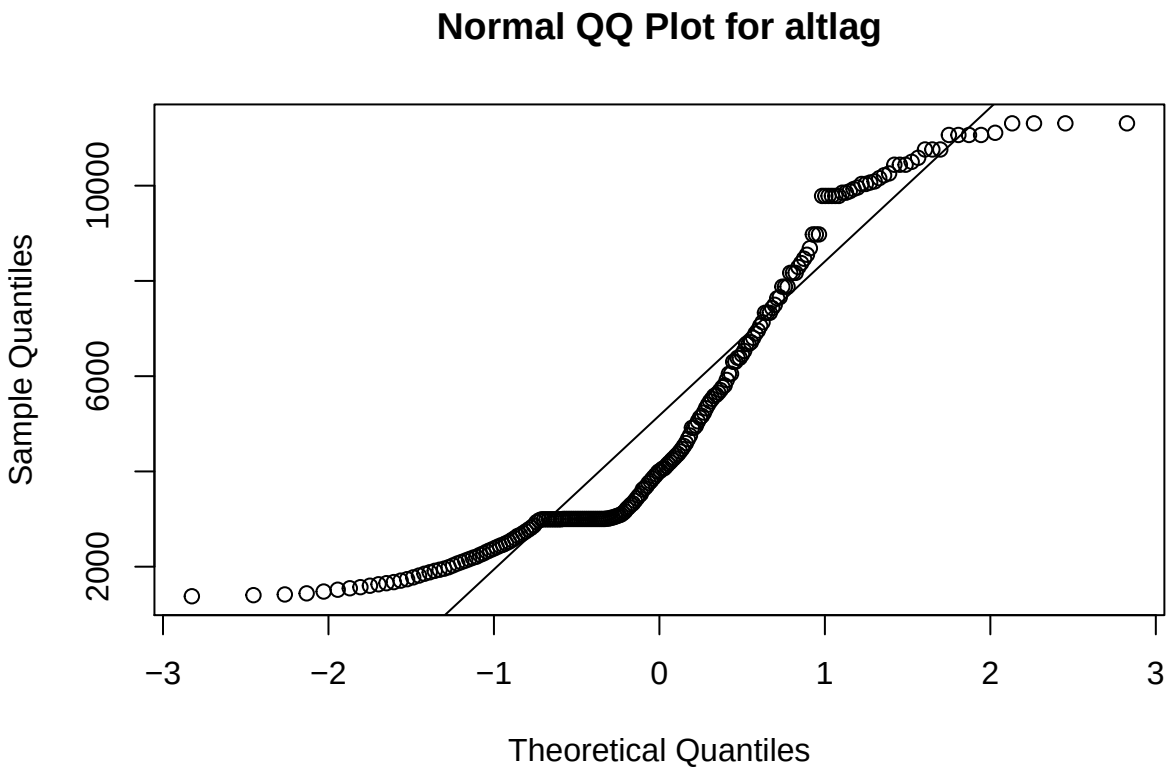
```
qqnorm(FDeval$xlag, main = "Normal QQ Plot for xlag")
qqline(FDeval$xlag)
```



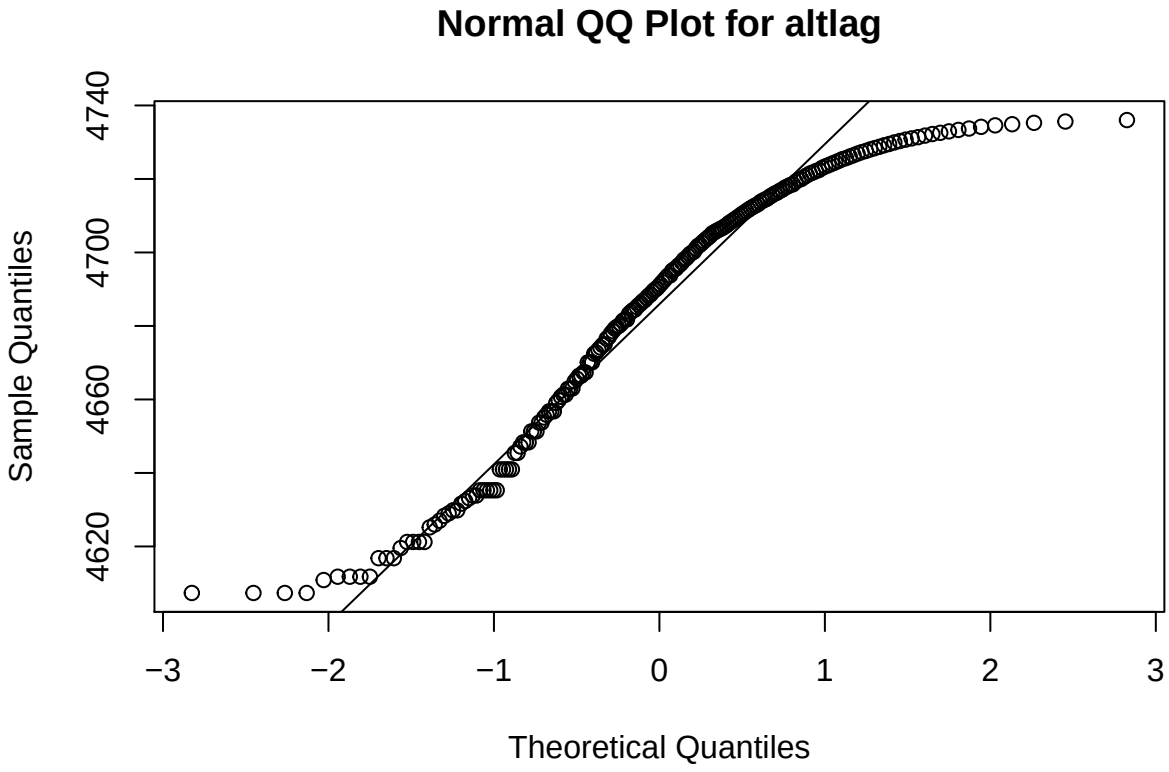

```r
qqnorm(FDeval$ylag, main = "Normal QQ Plot for ylag")
qqline(FDeval$ylag)
```




```r
qqnorm(FDeval$altlag, main = "Normal QQ Plot for altlag")
qqline(FDeval$altlag)
```

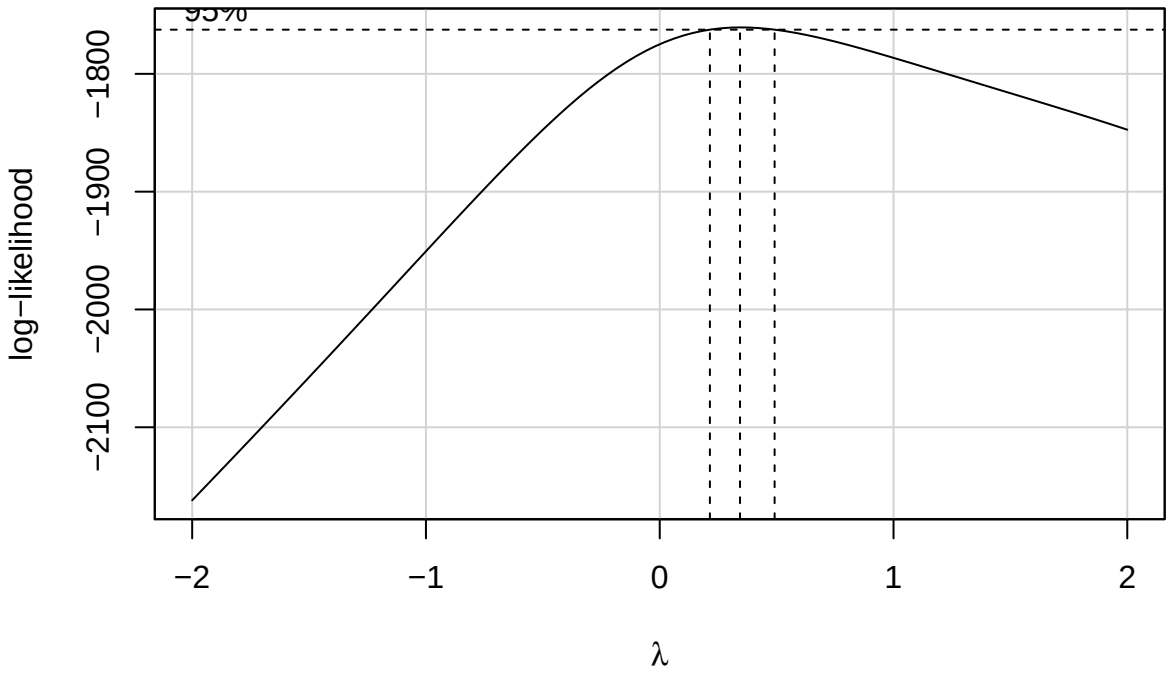



```
qqnorm(FDeval$distlag, main = "Normal QQ Plot for altlag")
qqline(FDeval$distlag)
```

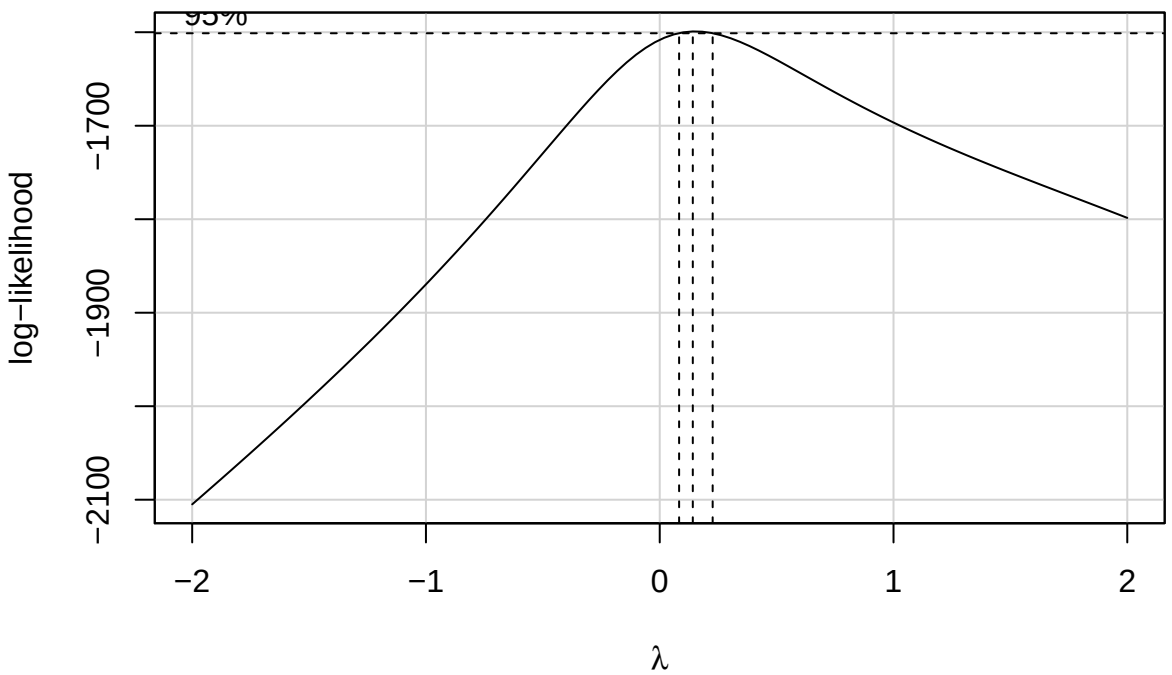


F. Dependengt Variable Transform: Box Cox

```
# check boxCox
#lreg.f1.s <- lm(alt ~ x*y , data = FDeval)
bc1 <- boxCox(FDeval$alt ~ FDeval$x*FDeval$y, family = 'yjPower')
```

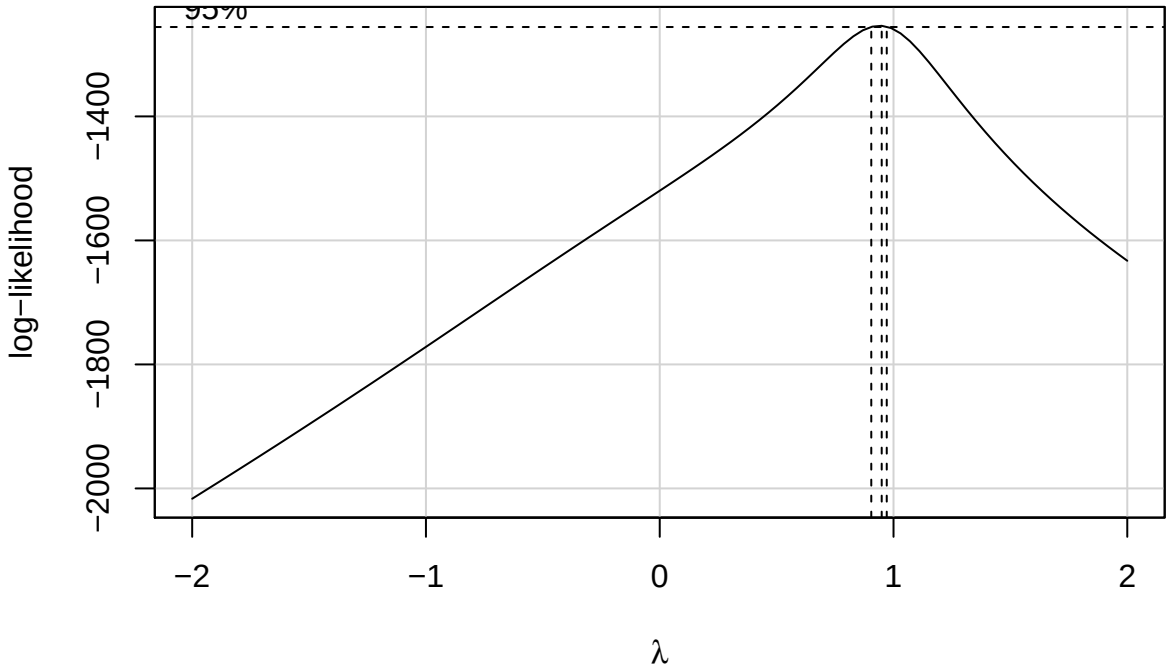



```
#lreg.f1.s2 <- lm(alt ~  x*y + distlag + distlag*speedlag, data = FDeval)
bc2 <- boxCox(FDeval$alt ~ FDeval$x*FDeval$y + FDeval$distlag + FDeval$distlag*FDeval$speedlag, family =
```




```
#lreg.f1.sc <- lm(alt ~ x*y + altlag + distlag*speedlag + altlag*vspeedlag , data = FDeval)
bc3 <- boxCox(FDeval$alt ~ FDeval$x*FDeval$y + FDeval$altlag +
```

```
                        FDeval$dist*FDeval$speedlag +
                        FDeval$altlag*FDeval$vspeedlag, family = 'yjPower')
```




```
lambda1 <- bc1$x[bc1$y == max(bc1$y)]
print(lambda1)
```

```
## [1] 0.3434343
```

```
lambda2 <- bc2$x[bc2$y == max(bc2$y)]
print(lambda2)
```

```
## [1] 0.1414141
```

```
lambda3 <- bc3$x[bc3$y == max(bc3$y)]
print(lambda3)
```

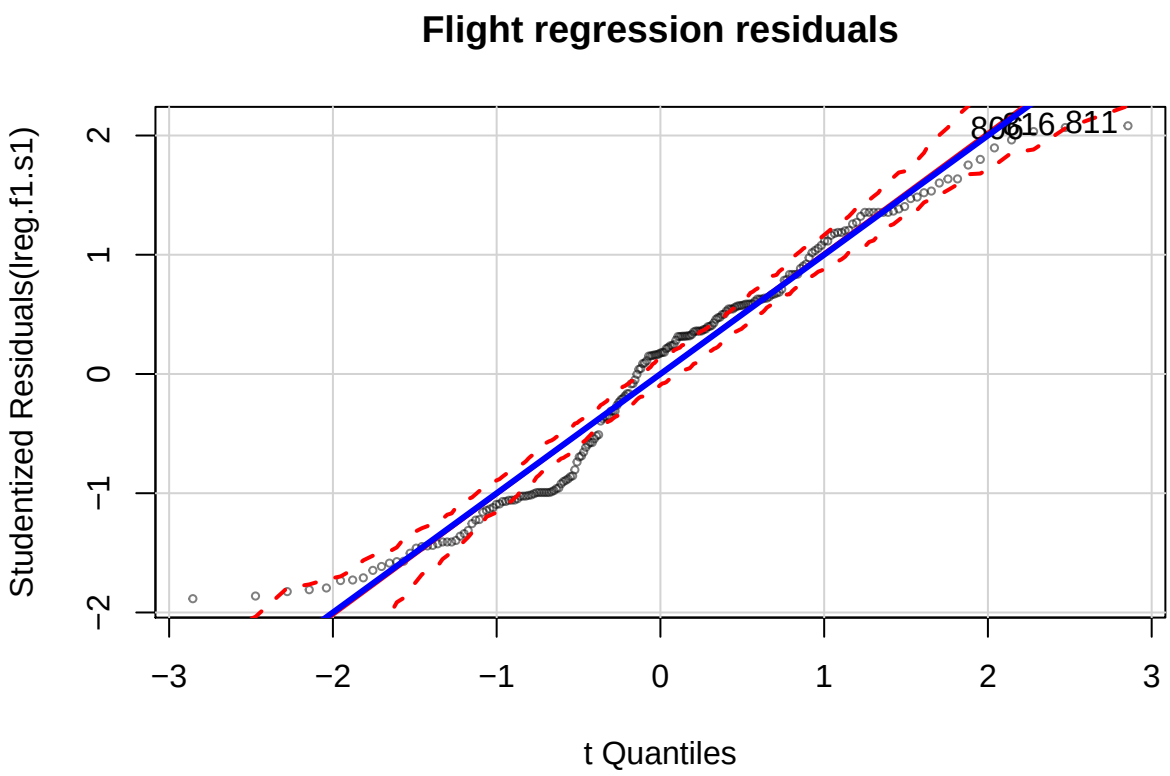
```
## [1] 0.9494949
```

```
#FDeval$Trans.alt1 <- (FDeval$x^(lambda1)-1)/(lambda1)
#FDeval$Trans.alt2 <- (FDeval$x^(lambda2)-1)/(lambda2)
lambda1 <-0.34
lambda3 <-0.9

FDeval <- FDeval[row.names(FDeval) != "231" & row.names(FDeval) != "451",]

lreg.f1.s1 <- lm((alt^(lambda1)-1)/(lambda1) ~ x*y , data = FDeval)
lreg.f1.s2 <- lm(log(alt) ~ x*y + distlag + distlag*speedlag, data = FDeval)
lreg.f1.sc <- lm((alt^(lambda3)-1)/(lambda3) ~ x*y + altlag + distlag*speedlag + altlag*vspeedlag, data


qqPlot(lreg.f1.s1,
       main = "Flight regression residuals",
       pch  = 21,
       id.n = 3,
```

```
       cex = 0.5,
       col  = rgb(0, 0, 0, .5))
```

```
## 806 816 811
## 209 210 211
```

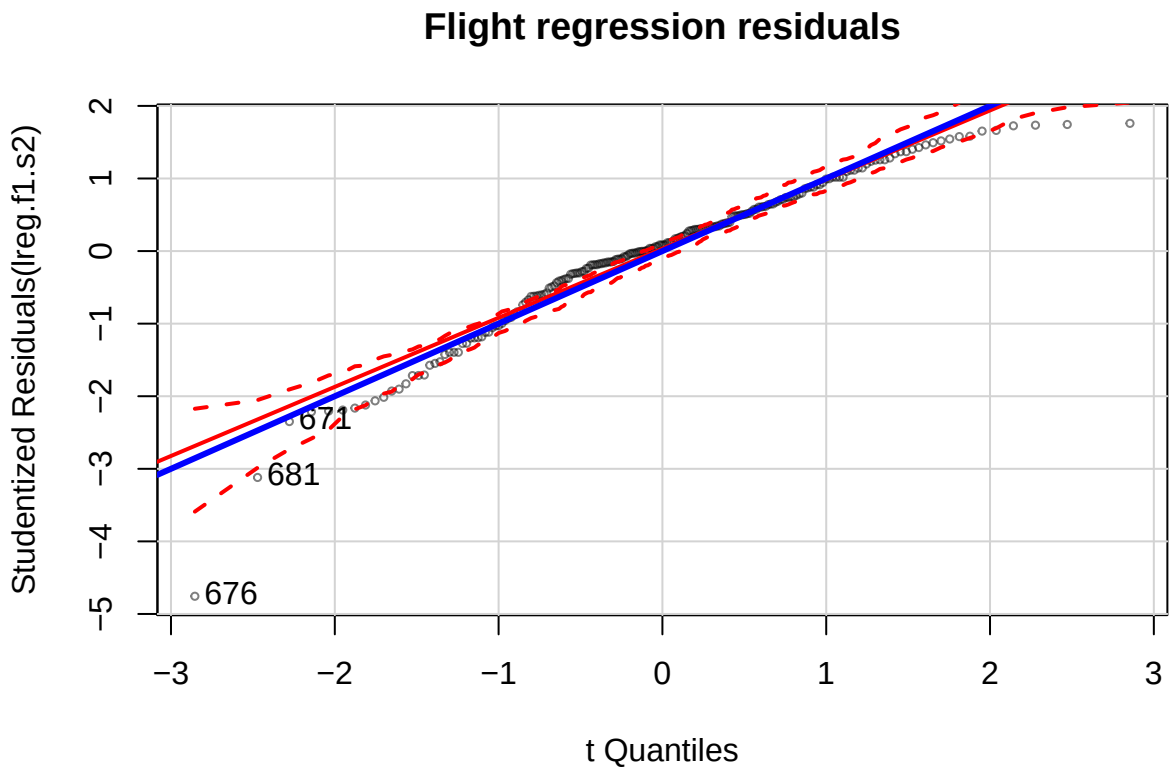
```
abline(a=0, b=1,col="blue",lwd=3)
```




```
qqPlot(lreg.f1.s2,
       main = "Flight regression residuals",
       pch  = 21,
       id.n = 3,
       cex = 0.5,
       col  = rgb(0, 0, 0, .5))
```

```
## 676 681 671
##   1   2   3
```

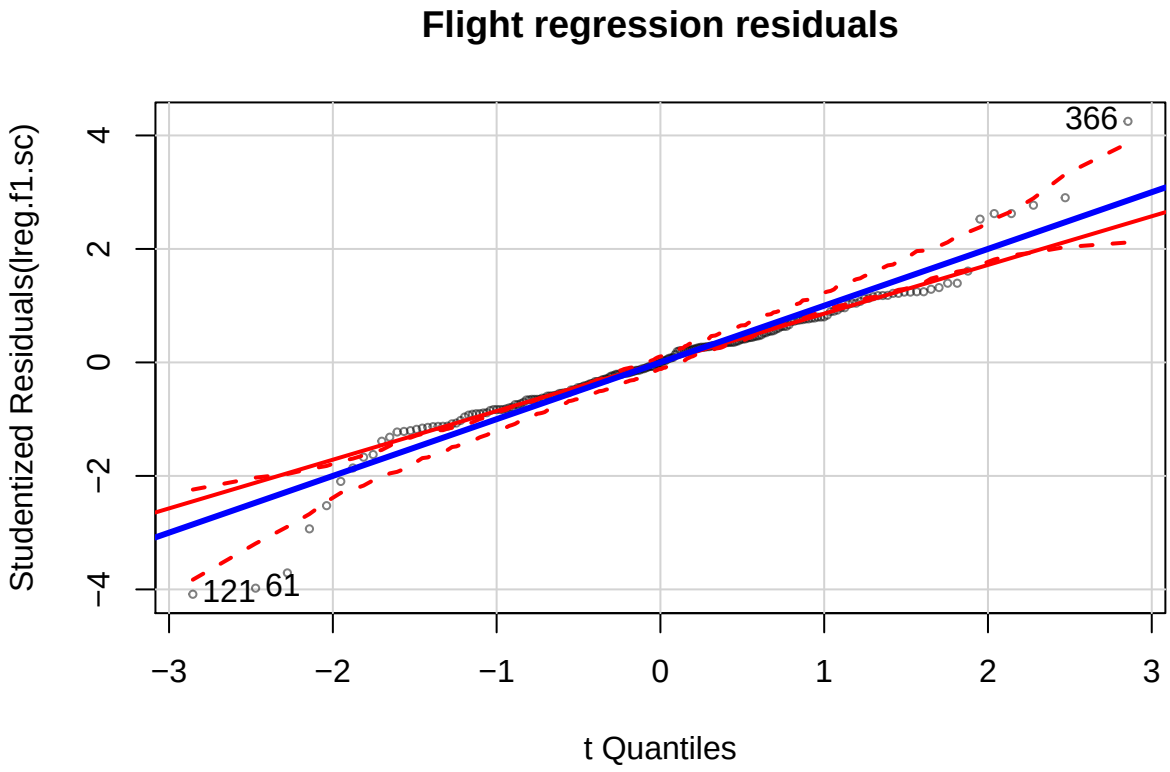
```
abline(a=0, b=1,col="blue",lwd=3)
```




```
qqPlot(lreg.f1.sc,
       main = "Flight regression residuals",
       pch  = 21,
       id.n = 3,
       cex = 0.5,
       col  = rgb(0, 0, 0, .5))
```

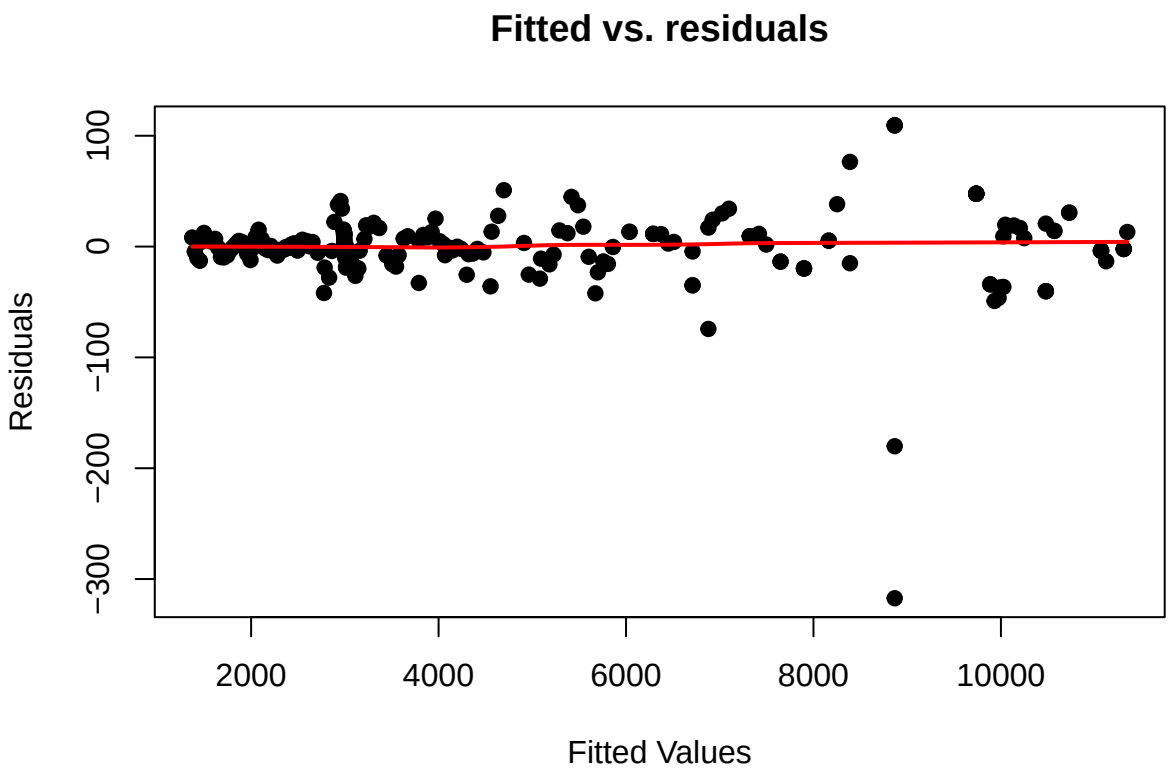
```
## 121  61 366
##   1   2 210
```

```
abline(a=0, b=1,col="blue",lwd=3)
```

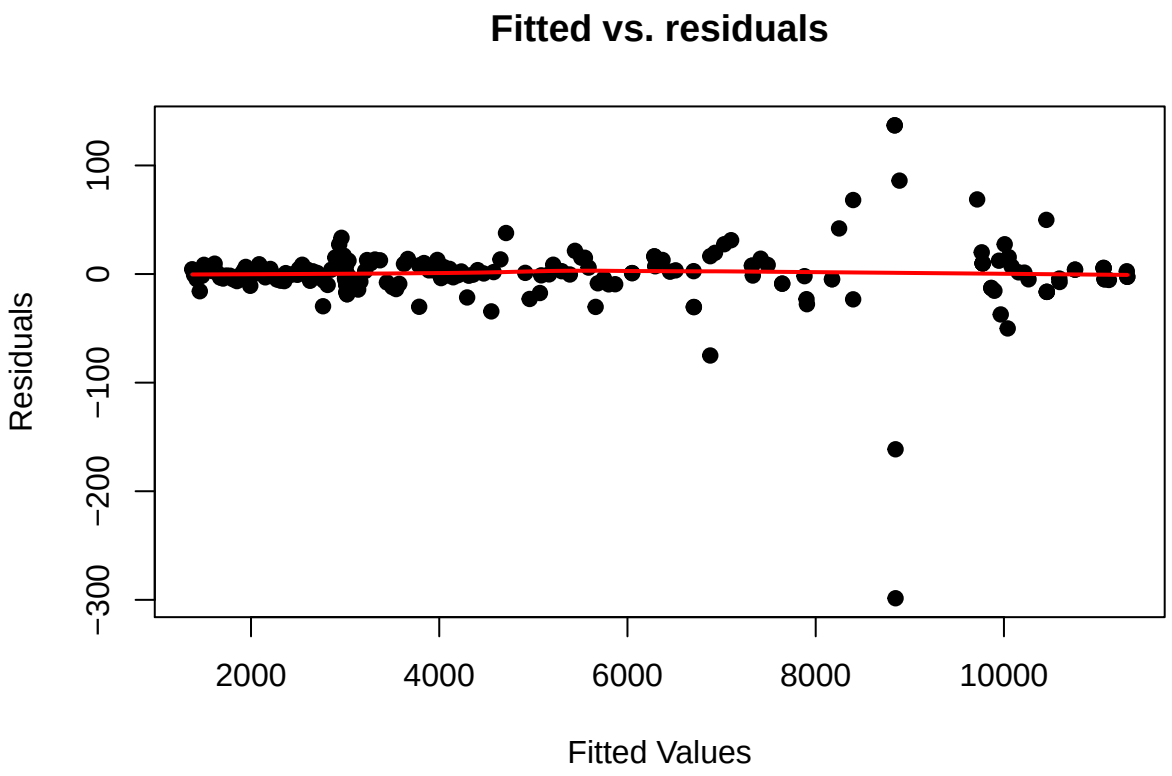


G. GAM

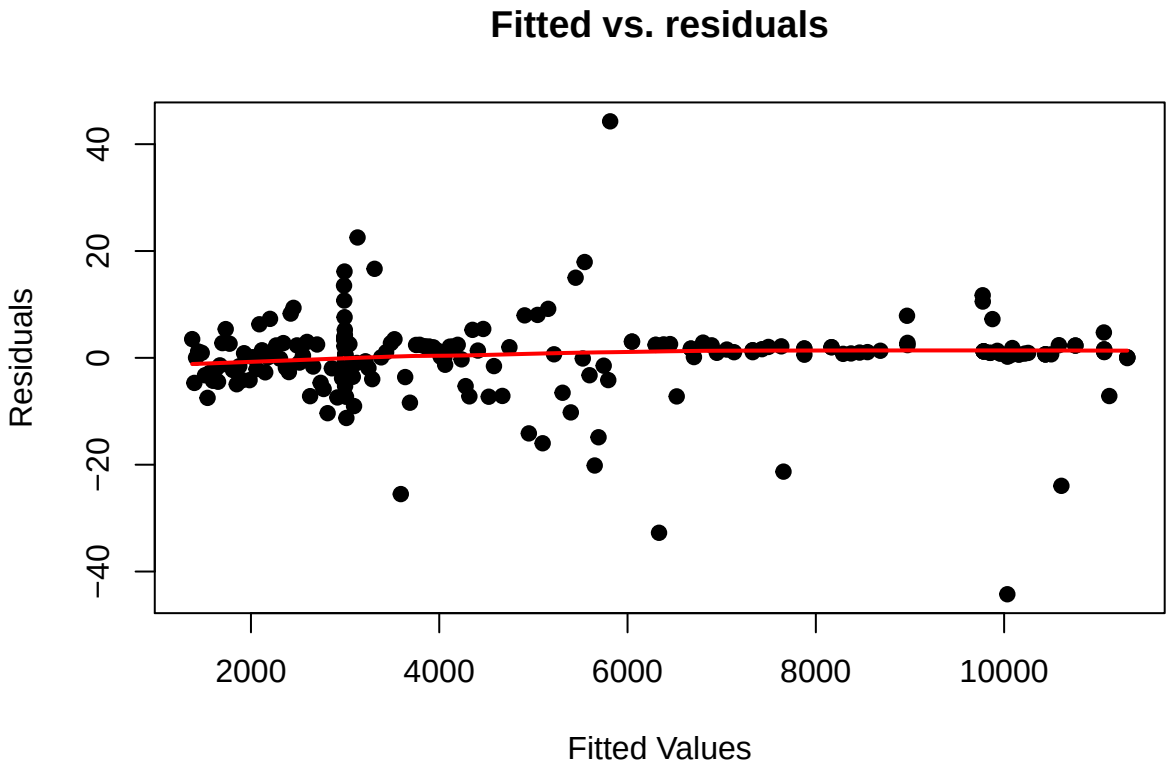
```
# Before Transform
gam.s <- gam(alt ~ s(x,y), data = FDeval)
gam.s2 <- gam(alt ~ s(x,y) + s(distlag) + s(distlag,speedlag), data = FDeval)
gam.sc <- gam(alt ~ s(x,y) + s(altlag) + s(distlag,speedlag) +
               s(altlag,vspeedlag), data = FDeval)

plot(fitted(gam.s), resid(gam.s),
     xlab = "Fitted Values",
     ylab = "Residuals",
     pch = 19,
     col = "black",
     main = "Fitted vs. residuals")
lines(lowess(fitted(gam.s), resid(gam.s)), col = "red", lwd = 2)
```



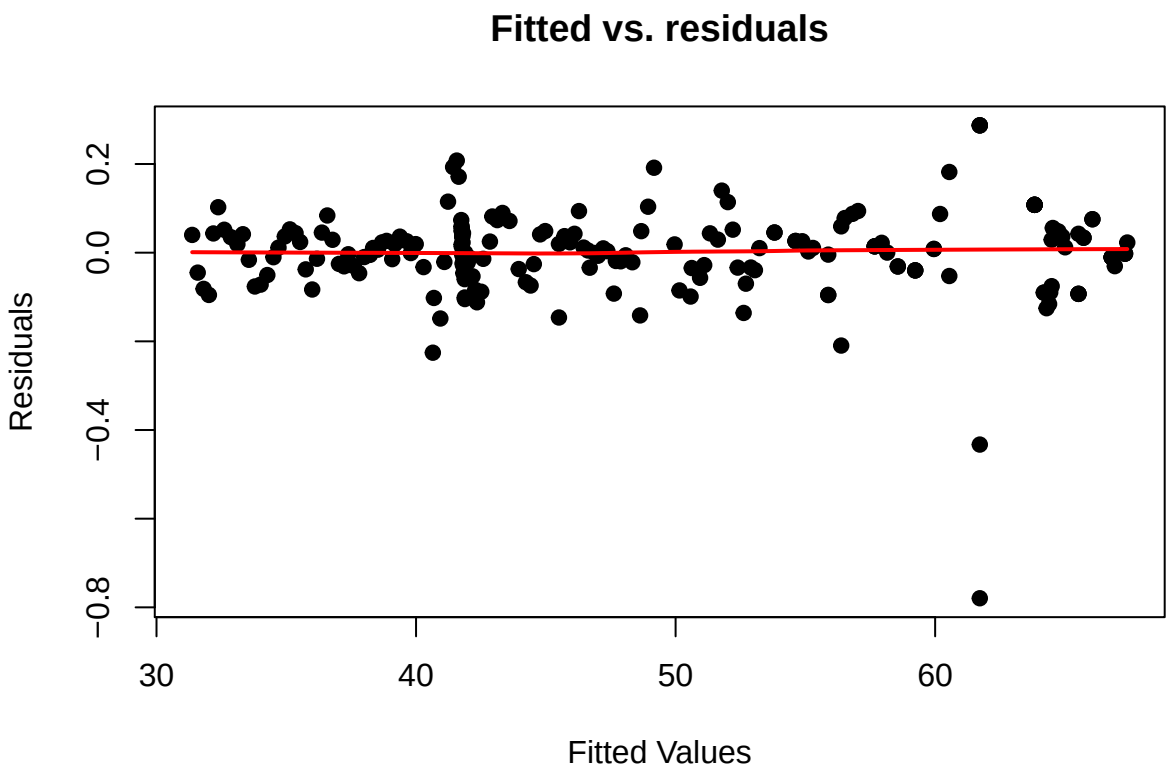

```
plot(fitted(gam.s2), resid(gam.s2),
     xlab = "Fitted Values",
     ylab = "Residuals",
     pch = 19,
     col = "black",
     main = "Fitted vs. residuals")
lines(lowess(fitted(gam.s2), resid(gam.s2)), col = "red", lwd = 2)
```



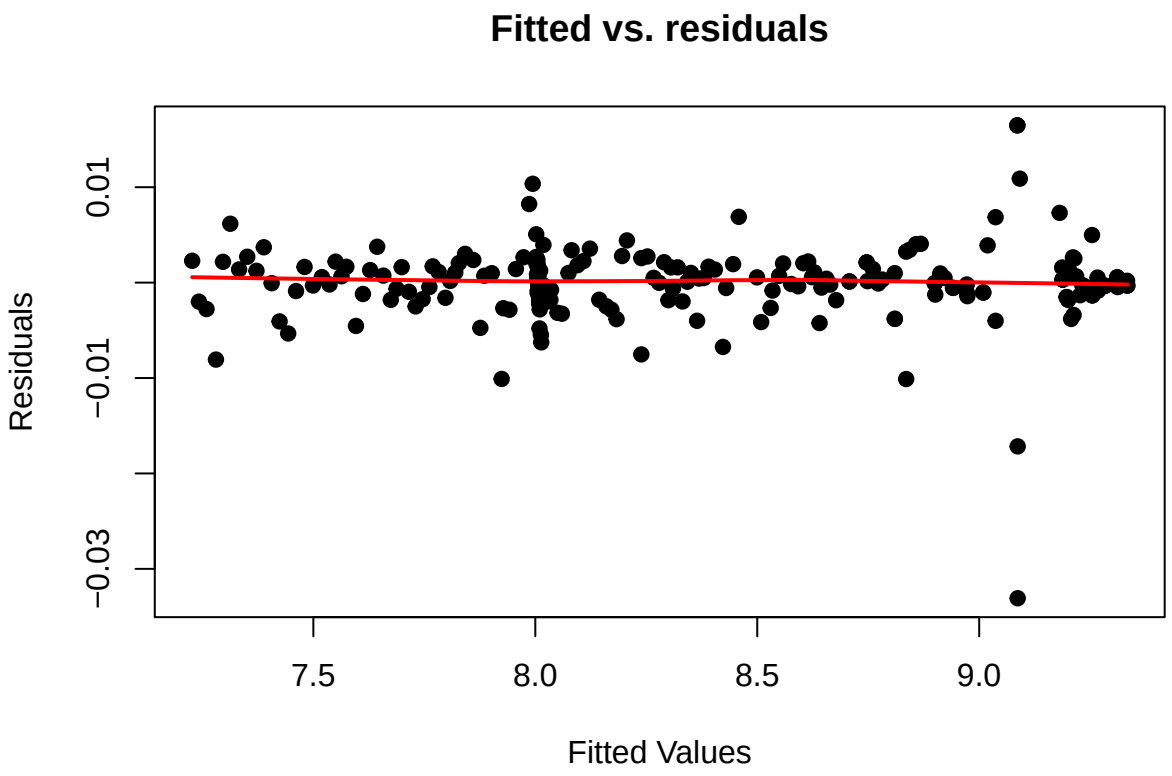

```
plot(fitted(gam.sc), resid(gam.sc),
     xlab = "Fitted Values",
     ylab = "Residuals",
     pch = 19,
     col = "black",
     main = "Fitted vs. residuals")
lines(lowess(fitted(gam.sc), resid(gam.sc)), col = "red", lwd = 2)
```



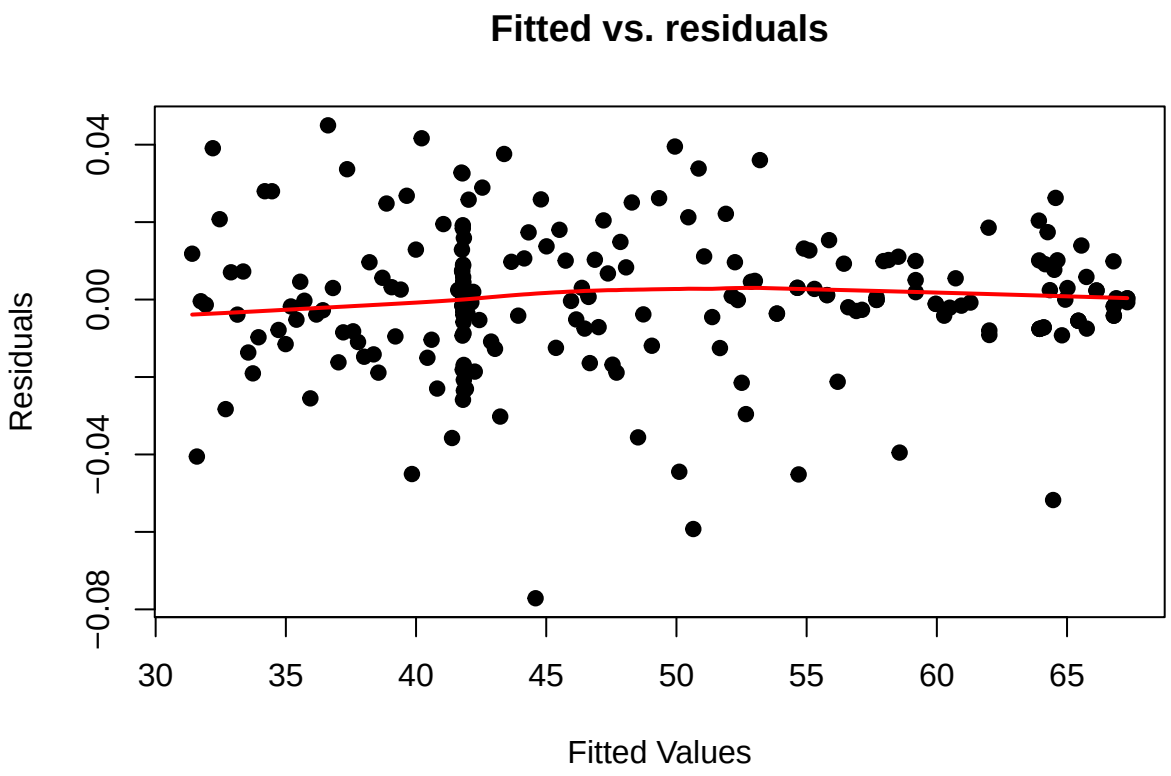

```
# After Transform
gam.s <- gam((alt^(lambda1)-1)/(lambda1) ~ s(x,y), data = FDeval)
gam.s2 <- gam(log(alt) ~ s(x,y) + s(distlag) + s(distlag,speedlag) , data = FDeval)
gam.sc <- gam((alt^(lambda1)-1)/(lambda1) ~ s(x,y) + s(altlag) + s(distlag,speedlag) +
                  s(altlag,vspeedlag), data = FDeval)

plot(fitted(gam.s), resid(gam.s),
     xlab = "Fitted Values",
     ylab = "Residuals",
     pch = 19,
     col = "black",
     main = "Fitted vs. residuals")
lines(lowess(fitted(gam.s), resid(gam.s)), col = "red", lwd = 2)
```




```
plot(fitted(gam.s2), resid(gam.s2),
     xlab = "Fitted Values",
     ylab = "Residuals",
     pch = 19,
     col = "black",
     main = "Fitted vs. residuals")
lines(lowess(fitted(gam.s2), resid(gam.s2)), col = "red", lwd = 2)
```



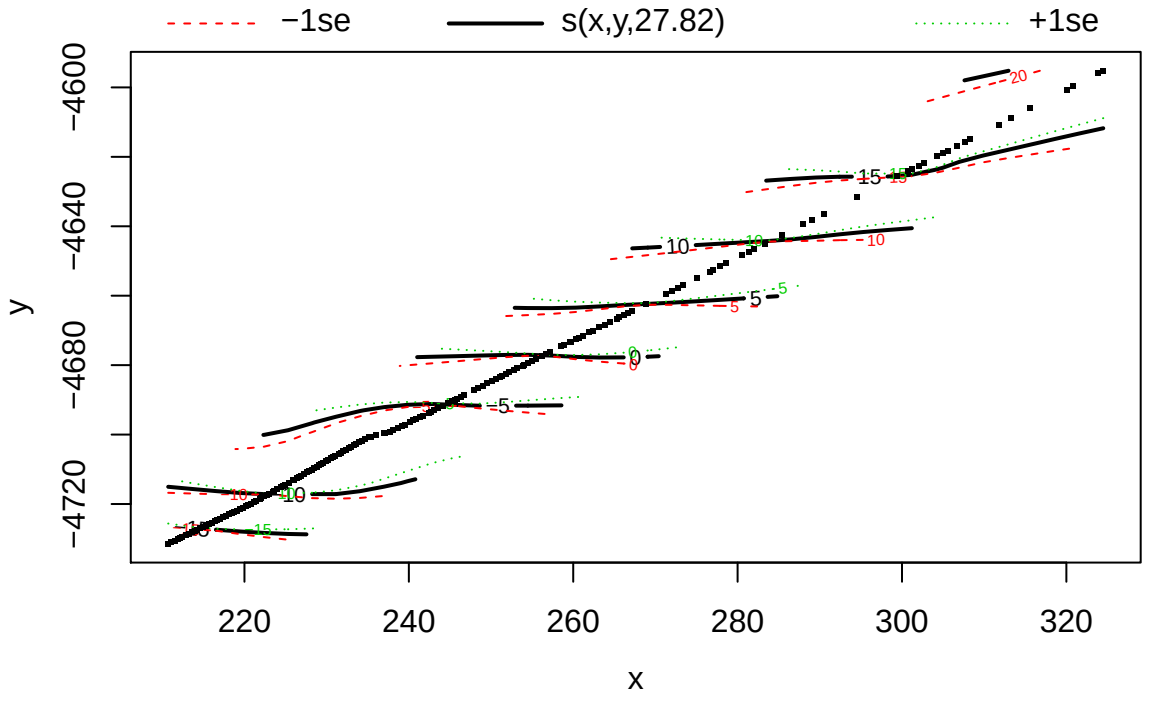

```
plot(fitted(gam.sc), resid(gam.sc),
     xlab = "Fitted Values",
     ylab = "Residuals",
     pch = 19,
     col = "black",
     main = "Fitted vs. residuals")
lines(lowess(fitted(gam.sc), resid(gam.sc)), col = "red", lwd = 2)
```



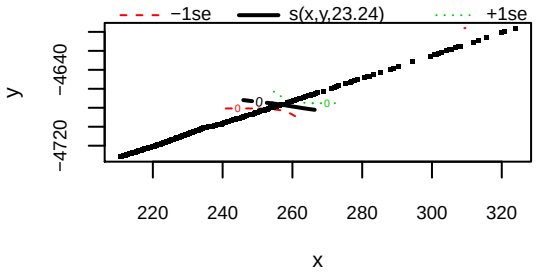
###

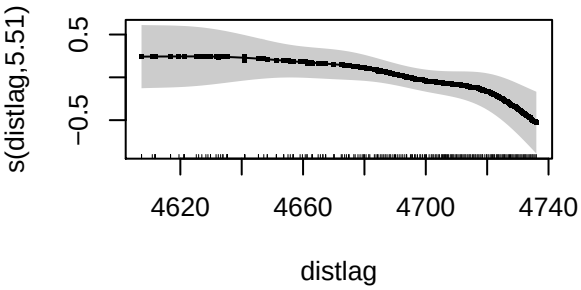
Partial residuals

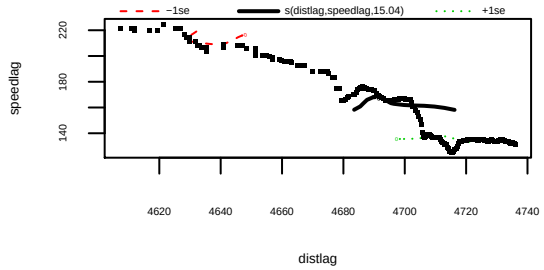
```
plot(gam.s,
     residuals = TRUE,  # Include the partial residuals
     shade= TRUE,  # Include shaded confidence bands
     pages= 1,
     scale= 0,
     cex= 3)
```



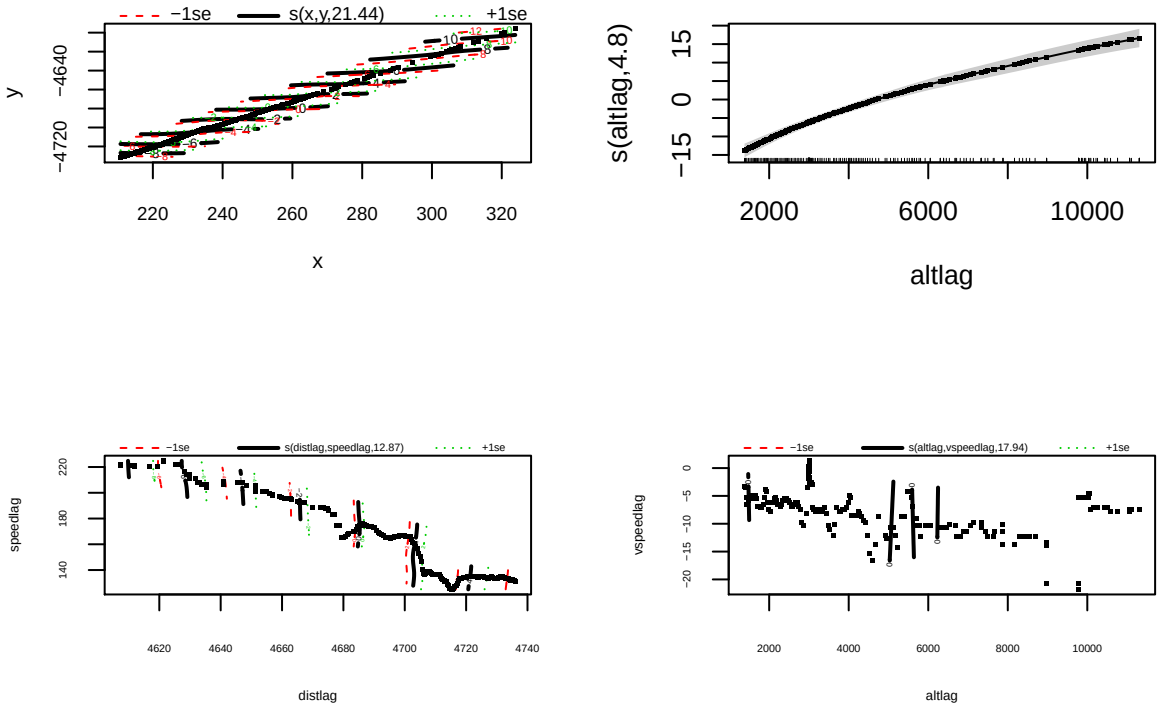
```
plot(gam.s2,
     residuals = TRUE,  # Include the partial residuals
     shade= TRUE,  # Include shaded confidence bands
     pages= 1,
     scale= 0,
     cex= 3)
```








```
plot(gam.sc,
     residuals = TRUE,  # Include the partial residuals
     shade= TRUE,  # Include shaded confidence bands
     pages= 1,
     scale= 0,
     cex= 3)
```

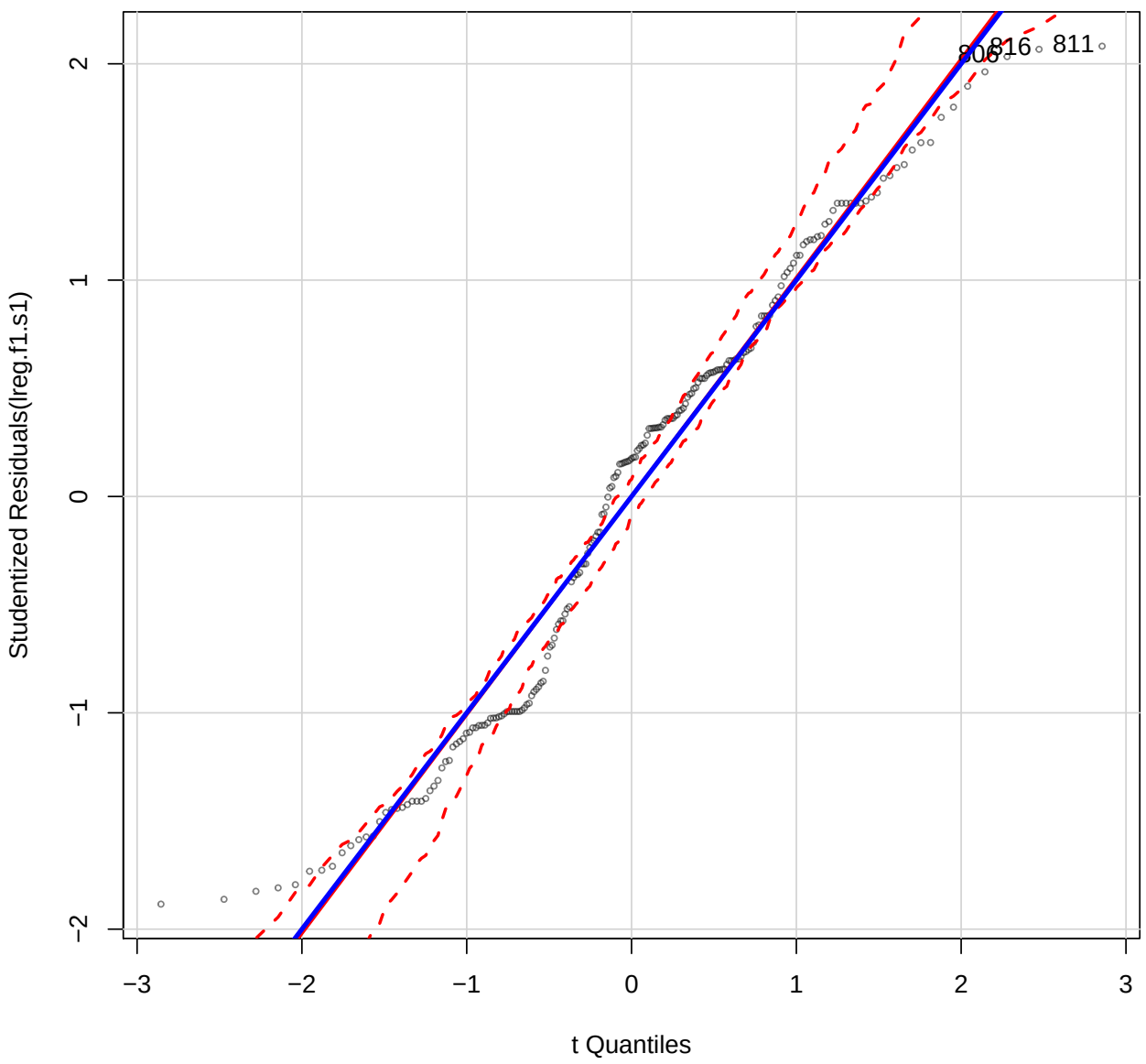


H. Idependent Variable Transforms

```
lreg.f1.s1 <- lm((alt^(lambda1)-1)/(lambda1) ~ x*y , data = FDeval)
lreg.f1.s2 <- lm(log(alt) ~ x*y + I(distlag^2) + distlag*speedlag, data = FDeval)
lreg.f1.sc <- lm((alt^(lambda3)-1)/(lambda3) ~ x*y + log(altlag) + distlag*speedlag + altlag*vspeedlag,

#vary.slope <- lmer(Trans.alt1 ~ log(x) + I(y^2) + dist*speedlag + (1 + dist|icao_addr), data = fpoints

########### QQPLOT
qqPlot(lreg.f1.s1,
       main = "Flight regression residuals",
       pch  = 21,
       id.n = 3,
       cex = 0.5,
       col  = rgb(0, 0, 0, .5))
```

```
## 806 816 811
## 209 210 211
```

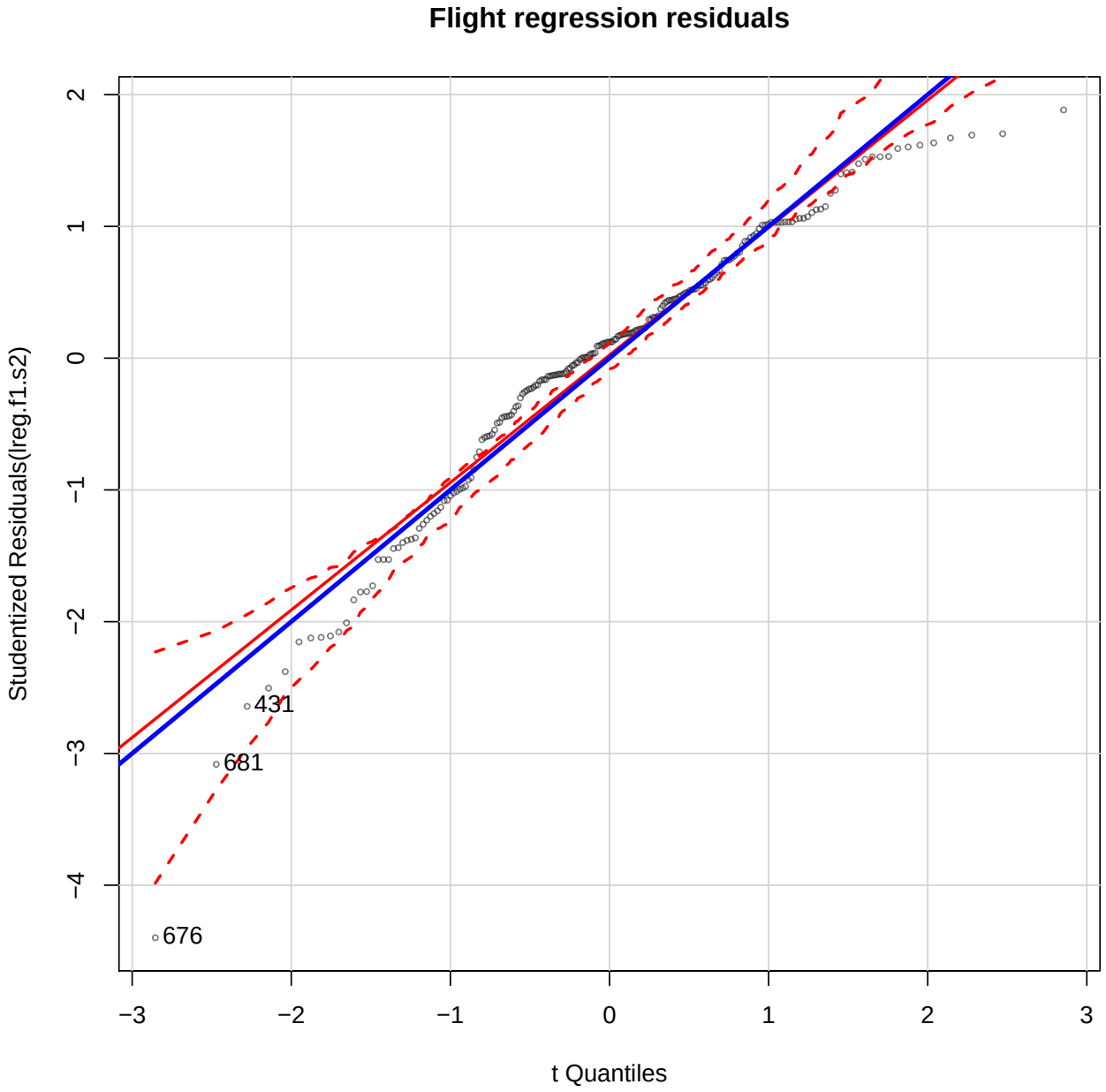
```
abline(a=0, b=1,col="blue",lwd=3)
```




```
qqPlot(lreg.f1.s2,
       main = "Flight regression residuals",
       pch  = 21,
       id.n = 3,
       cex = 0.5,
       col  = rgb(0, 0, 0, .5))
```

```
## 676 681 431
##   1   2   3
```

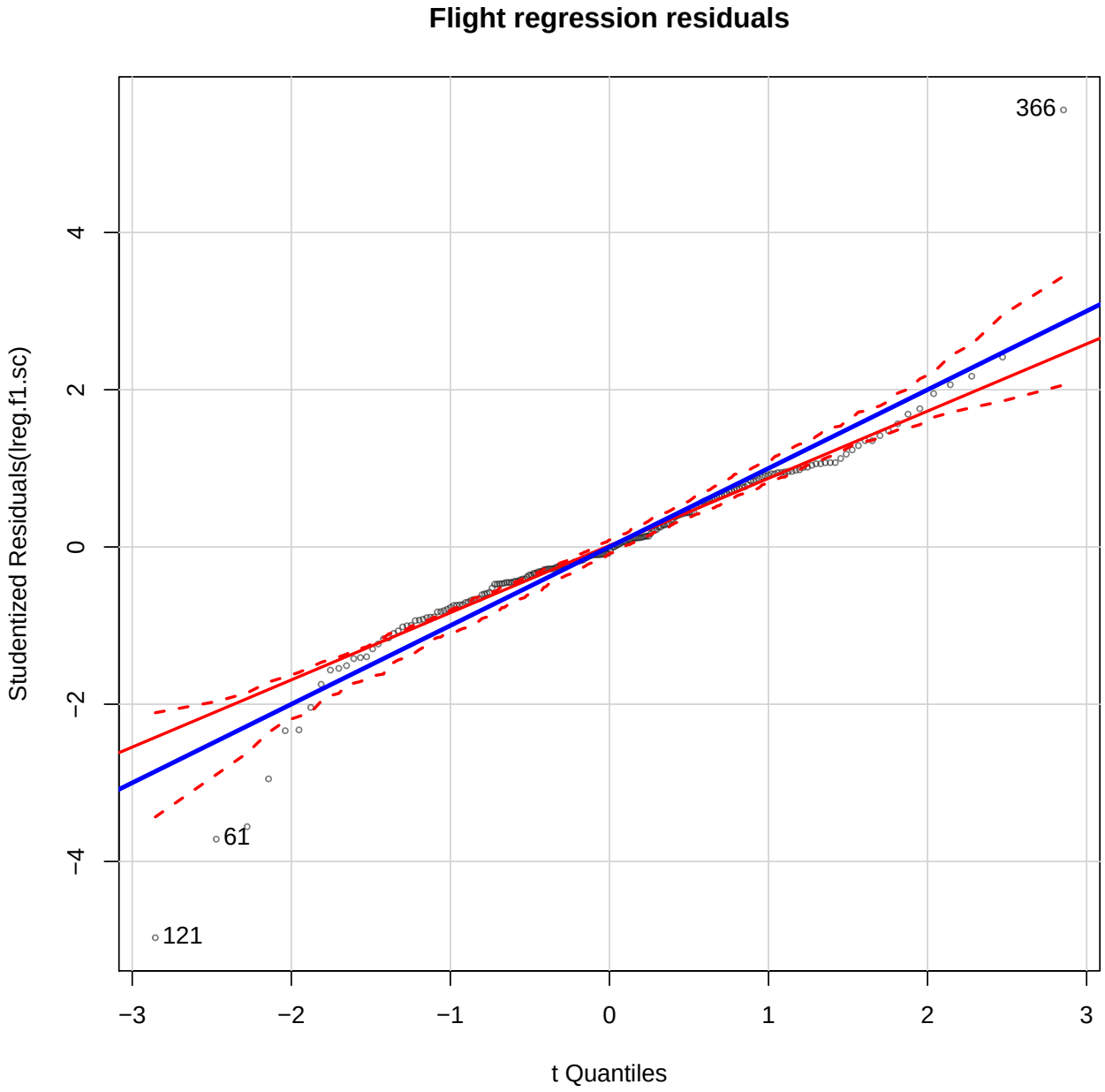
```
abline(a=0, b=1,col="blue",lwd=3)
```




```
qqPlot(lreg.f1.sc,
       main = "Flight regression residuals",
       pch  = 21,
       id.n = 3,
       cex = 0.5,
       col  = rgb(0, 0, 0, .5))
```

```
## 121  61 366
##   1   2 210
```

```
abline(a=0, b=1,col="blue",lwd=3)
```




```
#summary(vary.slope)
```

Final Summary

```
lreg.f1.s1 <- lm((alt^(lambda1)-1)/(lambda1) ~ x*y , data = FDeval)
lreg.f1.s2 <- lm(log(alt) ~ x*y + I(distlag^2) + distlag*speedlag, data = FDeval)
lreg.f1.sc <- lm((alt^(lambda3)-1)/(lambda3) ~ x*y + log(altlag) + distlag*speedlag + altlag*vspeedlag,

kable(summary(lreg.f1.s1)$coeff[,1:2])
```

	Estimate	Std. Error
(Intercept)	2348.5893988	830.4694996
x	-2.9226753	0.3813963

	Estimate	Std. Error
y	0.4880713	0.1669154
x:y	-0.0006113	0.0000648

```
kable(summary(lreg.f1.s2)$coeff[,1:2])
```

	Estimate	Std. Error
(Intercept)	-8387.4847107	2538.5829008
x	0.8045236	0.5640307
y	-0.1586444	0.0467581
I(distlagˆ2)	-0.0003490	0.0001084
distlag	3.2622847	1.0039194
speedlag	0.8541793	0.0534513
x:y	0.0001446	0.0001196
distlag:speedlag	-0.0001840	0.0000114

```
kable(summary(lreg.f1.sc)$coeff[,1:2])
```

	Estimate	Std. Error
(Intercept)	7426.4690971	6367.7244627
x	-52.5437682	11.4165099
y	24.0475859	1.5157216
log(altlag)	141.3122560	10.4484955
distlag	22.2842781	0.9155380
speedlag	9.7156437	11.1480427
altlag	0.3968918	0.0016871
vspeedlag	0.7971281	0.2661884
x:y	-0.0110652	0.0025871
distlag:speedlag	-0.0020168	0.0023889
altlag:vspeedlag	-0.0001235	0.0000362

I. Forcasting

1. Cross Validation

```
crossvalidate <- function(n){
nfolds <- n
case.folds <- rep(1:nfolds, length.out = nrow(FDtrain))

Model1 <- c()
Model2 <- c()
Model3 <- c()


for (fold in 1:nfolds) {

  # Create training and test cases
  train <- FDtrain[case.folds != fold, ]
```

```
  test <- FDtrain[case.folds == fold, ]

  #train Model
  train1 <- lm((alt^(lambda1)-1)/(lambda1) ~ x*y , data = train)

  train2 <- lm(log(alt) ~ x*y + I(distlag^2) + distlag*speedlag, data = train)

  train3 <- lm((alt^(lambda3)-1)/(lambda3) ~ x*y + log(altlag) + distlag*speedlag + altlag*vspeedlag, da


  # Generate test MSEs
  test1 <- ((test$alt^(lambda1)-1)/(lambda1) - predict(train1, test))^2
  rMSEtest1 <- sqrt(sum(test1) / length(test1))

  test2 <- (log(test$alt) - predict(train2, test))^2
  rMSEtest2 <- sqrt(sum(test2) / length(test2))

  test3 <- ((test$alt^(lambda3)-1)/(lambda3) - predict(train3, test))^2
  rMSEtest3 <- sqrt(sum(test3) / length(test3))


  # Append the rMSE from this iteration to vectors
  Model1 <- c(Model1, rMSEtest1)
  Model2 <- c(Model2, rMSEtest2)
  Model3 <- c(Model3, rMSEtest3)
}

# Average the MSEs
Model1.avg <- mean(Model1)
Model2.avg <- mean(Model2)
Model3.avg <- mean(Model3)


return(c(Model1.avg, Model2.avg, Model3.avg))
}


MSE.result.5fold <- crossvalidate(5)
print("      Model 1        |       Model 2     |       Model 3   ")
```

```
## [1] "      Model 1        |       Model 2     |       Model 3   "
```

```
print(paste0(MSE.result.5fold[1]," | ",MSE.result.5fold[2]," | ",MSE.result.5fold[3]))
```

```
## [1] "1.05818258547632 | 0.0302384757916979 | 7.62500306221807"
```

2. Model Testing

Finally, we take the first 80% of the data and train our models and test it with the last 20% of the data.

```
#rMSE for Model 1
Model1.train <- lm((alt^(lambda1)-1)/(lambda1) ~ x*y, data = FDtrain.for.Model)
Model1.test <- ((FDtest$alt^(lambda1)-1)/(lambda1) - predict(Model1.train,FDtest))^2
Model1.rMSEtest <- sqrt(sum(Model1.test) / length(Model1.test))
```

```
#rMSE for Model 2
Model2.train <- lm(log(alt) ~ x*y + I(distlag^2) + distlag*speedlag, data = FDtrain.for.Model)
Model2.test <- (log(FDtest$alt) - predict(Model2.train,FDtest))^2
Model2.rMSEtest <- sqrt(sum(Model2.test) / length(Model2.test))

#rMSE for Model 3
Model3.train <- lm((alt^(lambda3)-1)/(lambda3) ~ x*y + log(altlag) + distlag*speedlag + altlag*vspeedlag
                            data = FDtrain.for.Model)
Model3.test <- ((FDtest$alt^(lambda3)-1)/(lambda3) - predict(Model3.train,FDtest))^2
Model3.rMSEtest <- sqrt(sum(Model3.test) / length(Model3.test))


print(paste0("Model 1 rMSE: ",Model1.rMSEtest))
```

```
## [1] "Model 1 rMSE: 1.06026705390835"
```

```
print(paste0("Model 2 rMSE: ",Model2.rMSEtest))
```

```
## [1] "Model 2 rMSE: 0.0293914827478042"
```

```
print(paste0("Model 3 rMSE: ",Model3.rMSEtest))
```

```
## [1] "Model 3 rMSE: 16.7388816736614"
```

3. Testing for a Different flight

```
load("Flightdata-flight26.Rda")

fpoints$dist <- sqrt((fpoints$xlag)^2 + (fpoints$ylag)^2)

row.nos <- rep(1:5, length.out = nrow(fpoints))
#row.nos <- sample(row.nos)

# Create eval and training and test cases
FDeval <- fpoints[row.nos == 1, ]            #20% of Data for Evaluation

FDtrain <- fpoints[row.nos != 1 & row.nos != 5,]  #60% of Data for Training

FDtest <- fpoints[row.nos == 5, ]            #20% of Data for Testing

#############################################################################################

FDtrain.for.Model <- fpoints[row.nos != 5,]  #80% of Data for Model Training.
                              #This basically a combination on the first 20% and 60%
lreg.f1.s1 <- lm((alt^(lambda1)-1)/(lambda1) ~ x*y , data = FDeval)
lreg.f1.s2 <- lm(log(alt) ~ x*y + I(distlag^2) + distlag*speedlag, data = FDeval)
lreg.f1.sc <- lm((alt^(lambda3)-1)/(lambda3) ~ x*y + log(altlag) + distlag*speedlag + altlag*vspeedlag,


#rMSE for Model 1
Model1.train <- lm((alt^(lambda1)-1)/(lambda1) ~ x*y, data = FDtrain.for.Model)
Model1.test <- ((FDtest$alt^(lambda1)-1)/(lambda1) - predict(Model1.train,FDtest))^2
Model1.rMSEtest <- sqrt(sum(Model1.test) / length(Model1.test))
```

```
#rMSE for Model 2
Model2.train <- lm(log(alt) ~ x*y + I(distlag^2) + distlag*speedlag, data = FDtrain.for.Model)
Model2.test <- (log(FDtest$alt) - predict(Model2.train,FDtest))^2
Model2.rMSEtest <- sqrt(sum(Model2.test) / length(Model2.test))

#rMSE for Model 3
Model3.train <- lm((alt^(lambda3)-1)/(lambda3) ~ x*y + log(altlag) + distlag*speedlag + altlag*vspeedlag
                             data = FDtrain.for.Model)
Model3.test <- ((FDtest$alt^(lambda3)-1)/(lambda3) - predict(Model3.train,FDtest))^2
Model3.rMSEtest <- sqrt(sum(Model3.test) / length(Model3.test))


print(paste0("Model 1 rMSE: ",Model1.rMSEtest))
```

```
## [1] "Model 1 rMSE: 0.634835328398877"
```

```
print(paste0("Model 2 rMSE: ",Model2.rMSEtest))
```

```
## [1] "Model 2 rMSE: 0.0236471731265535"
```

```
print(paste0("Model 3 rMSE: ",Model3.rMSEtest))
```

```
## [1] "Model 3 rMSE: 34.7833624067337"
```

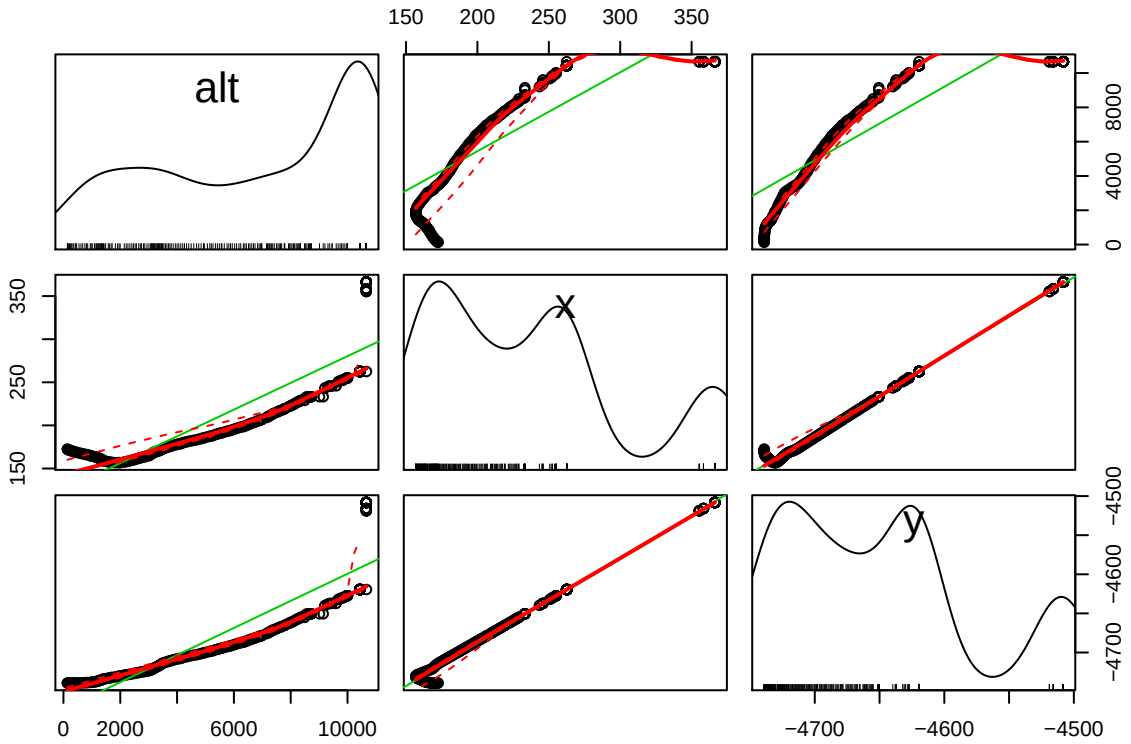
Appendix III

A. Data Partition for a Taking-off Flight

```
load("Flightdata-flight11.Rda")

row.nos <- rep(1:5, length.out = nrow(fpoints))
#row.nos <- sample(row.nos)

# Create eval and training and test cases
FDeval <- fpoints[row.nos == 1, ]            #20% of Data for Evaluation

FDtrain <- fpoints[row.nos != 1 & row.nos != 5,]  #60% of Data for Training

FDtest <- fpoints[row.nos == 5, ]            #20% of Data for Testing

#############################################################################################

FDtrain.for.Model <- fpoints[row.nos != 5,]  #80% of Data for Model Training.
                                #This basically a combination on the first 20% and 60%
```

B. Multicolinearity

```
scatterplotMatrix(~alt + x + y ,data = FDeval)
```




```
lreg.f1.s2 <- lm(alt ~ x + y + xlag*speedlag + ylag*speedlag, data = FDeval)
vif(lreg.f1.s2)
```

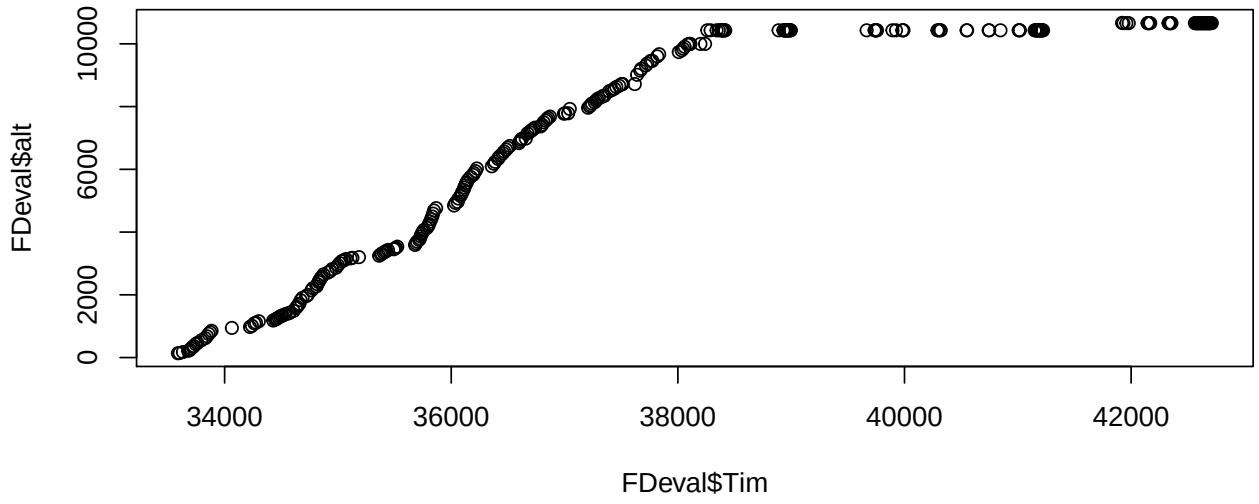
```
##             x             y          xlag      speedlag          ylag
##      513013.2      749601.7      536733.5    31486177.3      773268.2
## xlag:speedlag speedlag:ylag
##      239600.1    26753248.9
```

```
lreg.f1.s2 <- lm(alt ~ x + y + distlag*speedlag, data = FDeval)
vif(lreg.f1.s2)
```

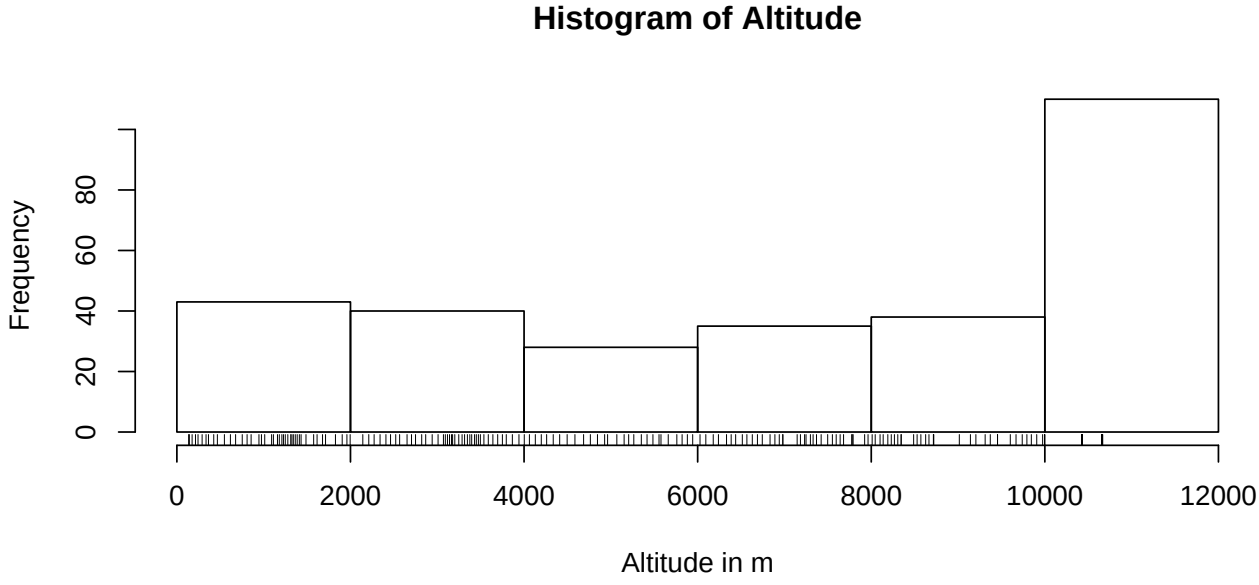
```
##                x                y          distlag        speedlag
##         832.5288      141317.6352      155295.5589      354941.0866
## distlag:speedlag
##      321029.3028
```

C. Data Summary

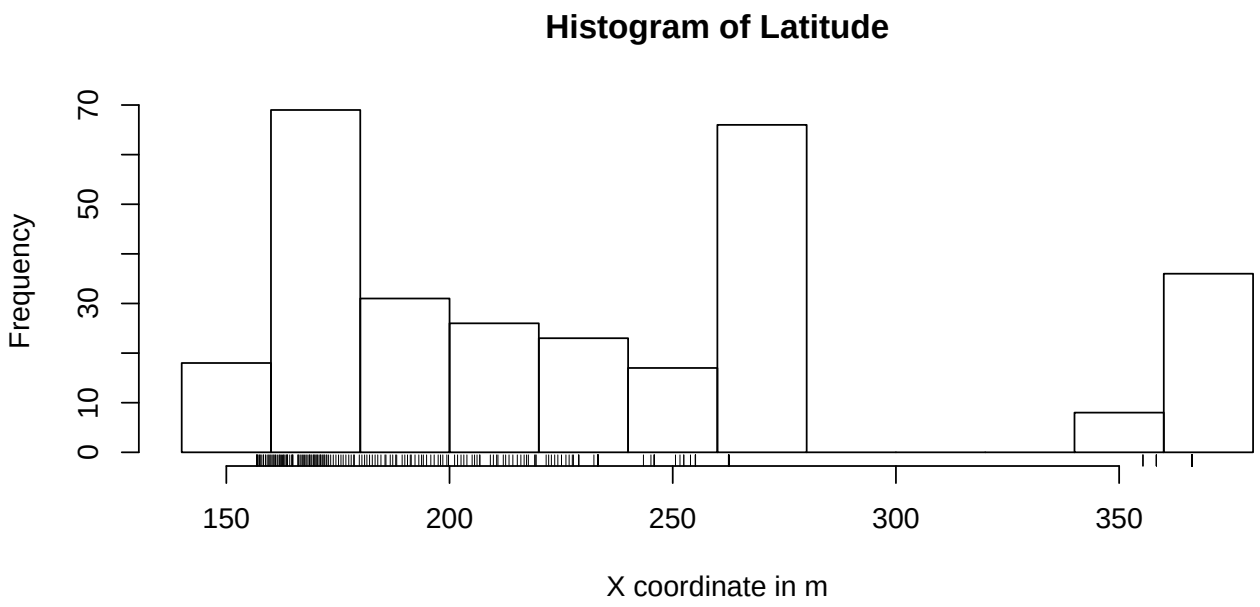
```
plot(FDeval$Tim,FDeval$alt)
```



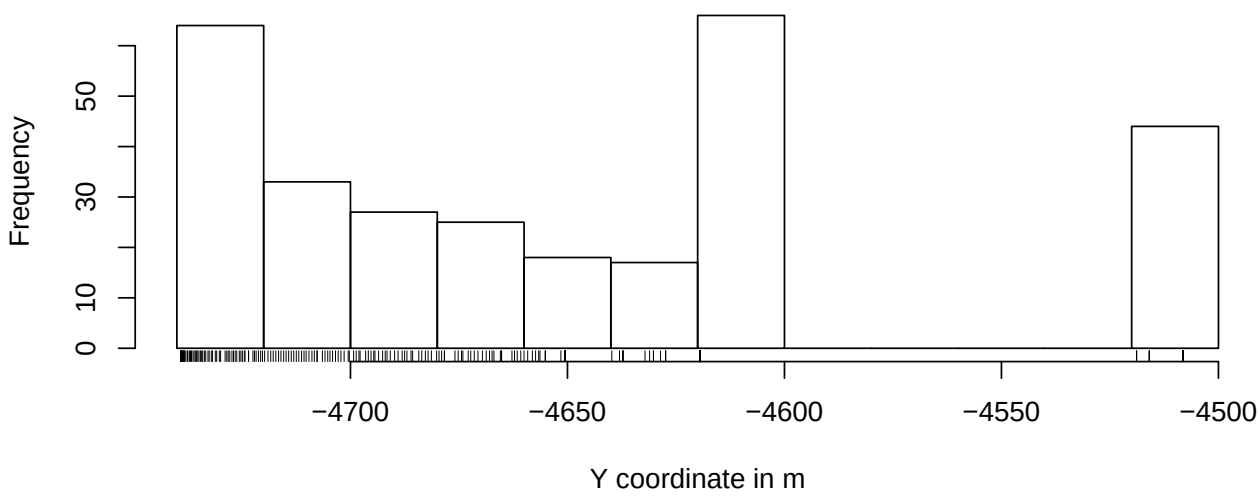

```
hist(FDeval$alt, breaks='FD',
     #xlim   = c(10000,150000),
     #ylim   = c(0,25),
     main   = "Histogram of Altitude",
     xlab = "Altitude in m")
rug(jitter(FDeval$alt))
```




```
hist(FDeval$x, breaks='FD',
     #xlim   = c(10000,150000),
     #ylim   = c(0,25),
     main   = "Histogram of Latitude",
     xlab = "X coordinate in m")
rug(jitter(FDeval$x))
```

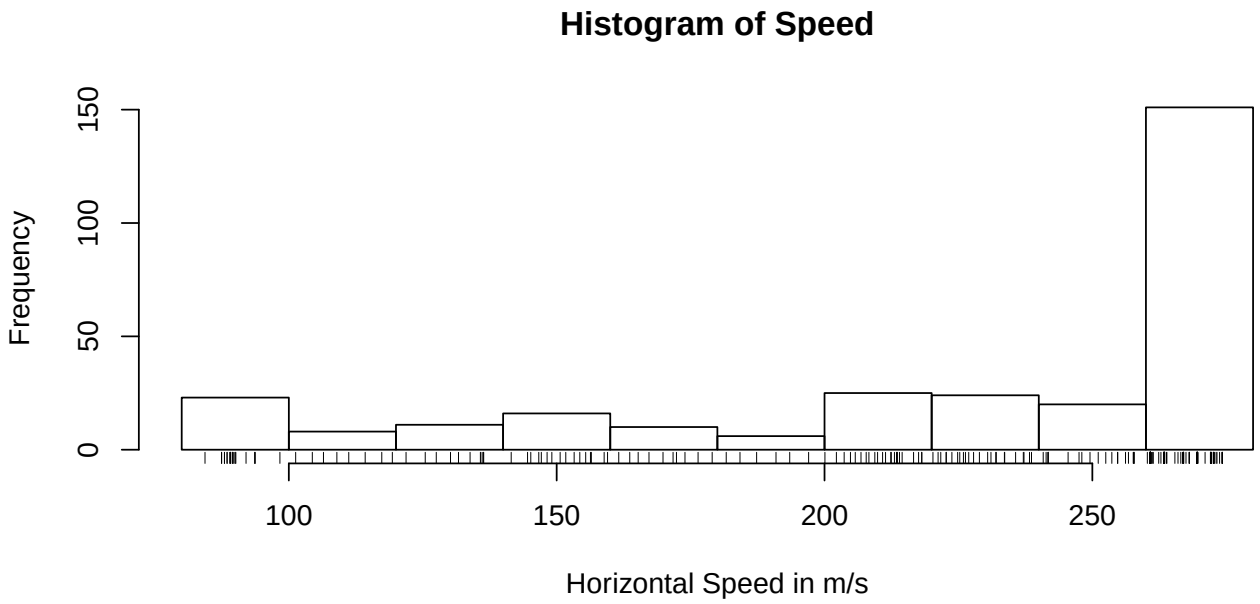



```
#Tranform y points to positive
fpoints$y <- fpoints$y-min(fpoints$y) + 1
fpoints$ylag <- Lag(fpoints$y, shift=1)
hist(FDeval$y, breaks='FD',
     #xlim   = c(10000,150000),
     #ylim   = c(0,25),
     main   = "Histogram of Longitude",
     xlab = "Y coordinate in m")
rug(jitter(FDeval$y))
```

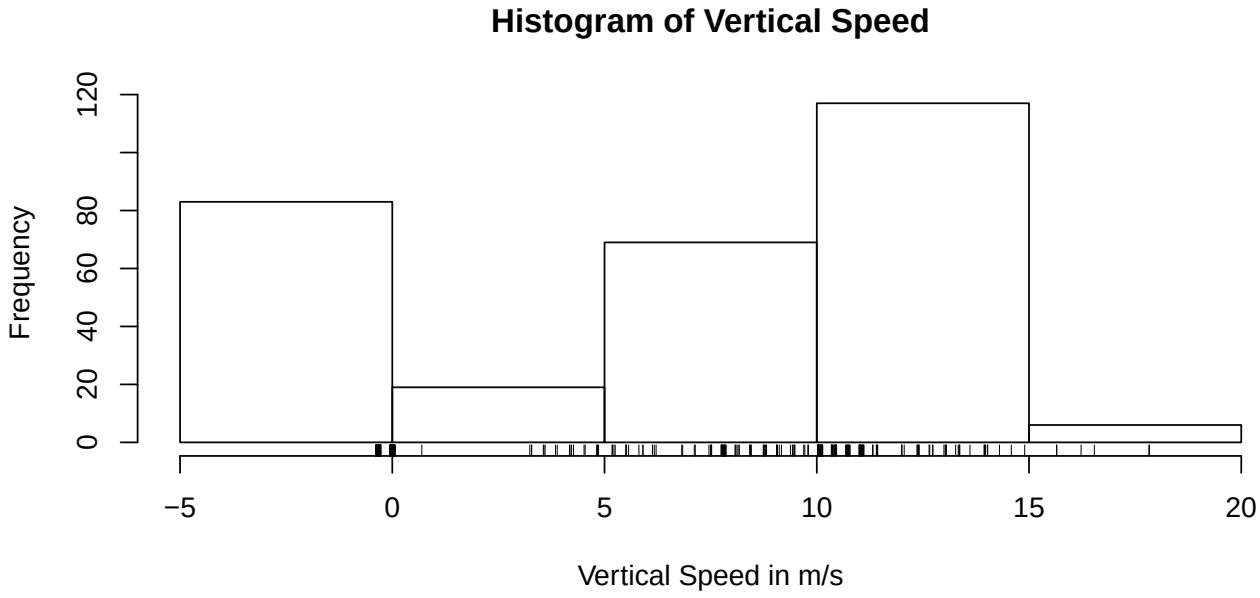



```
hist(FDeval$speed, breaks='FD',
     #xlim   = c(10000,150000),
     #ylim   = c(0,25),
     main   = "Histogram of Speed",
     xlab = "Horizontal Speed in m/s")
```

```
rug(jitter(FDeval$speed))
```




```
#fpoints$vert_speed <- fpoints$vert_speed-min(fpoints$vert_speed) + 1
#fpoints$uspeedlag <- Lag(fpoints$vert_speed, shift=1)
hist(FDeval$vert_speed, breaks='FD',
     #xlim   = c(10000,150000),
     #ylim   = c(0,25),
     main   = "Histogram of Vertical Speed",
     xlab = "Vertical Speed in m/s")
rug(jitter(FDeval$vert_speed))
```



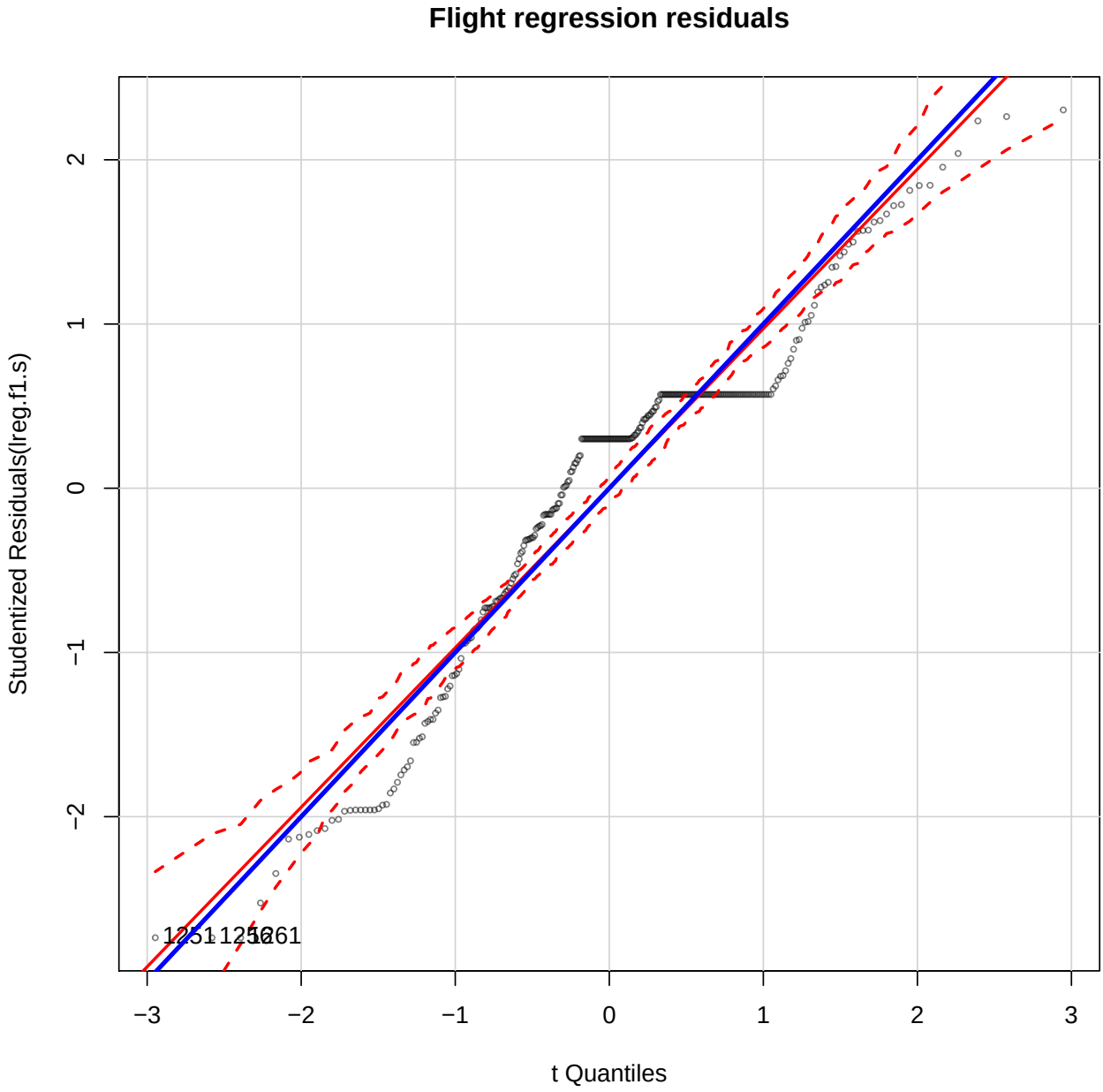
D. Conditional Story

To find the Independent variables and the respective interaction terms

```
FDeval <- FDeval[row.names(FDeval) != "351" & row.names(FDeval) != "6" & row.names(FDeval) != "596" &
                   row.names(FDeval) != "471" & row.names(FDeval) != "496" & row.names(FDeval) != "486"
                    & row.names(FDeval) != "547" & row.names(FDeval) != "574" & row.names(FDeval) != "6

#fpoints$TimT <- fpoints$Tim - min(fpoints$Tim)
lreg.f1.s <- lm(alt ~ x*y , data = FDeval)
lreg.f1.s2 <- lm(alt ~  x*y + distlag + distlag*speedlag, data = FDeval)

lreg.f1.c <- lm(alt ~ x*y + altlag, data = FDeval)
lreg.f1.c1 <- lm(alt ~ x*y + altlag*vspeedlag, data = FDeval)
lreg.f1.c2 <- lm(alt ~ x*y + dist*speedlag + altlag*vspeedlag, data = FDeval)
lreg.f1.sc <- lm(alt ~ x*y + altlag + distlag*speedlag + altlag*vspeedlag , data = FDeval)



#summary(lreg.f1.s)
#summary(lreg.f1.s2)
#summary(lreg.f1.sc)

############ QQPLOT
qqPlot(lreg.f1.s,
       main = "Flight regression residuals",
       pch  = 21,
       id.n = 3,
       cex = 0.5,
       col  = rgb(0, 0, 0, .5))
```

```
## 1251 1256 1261
##    1    2    3
```

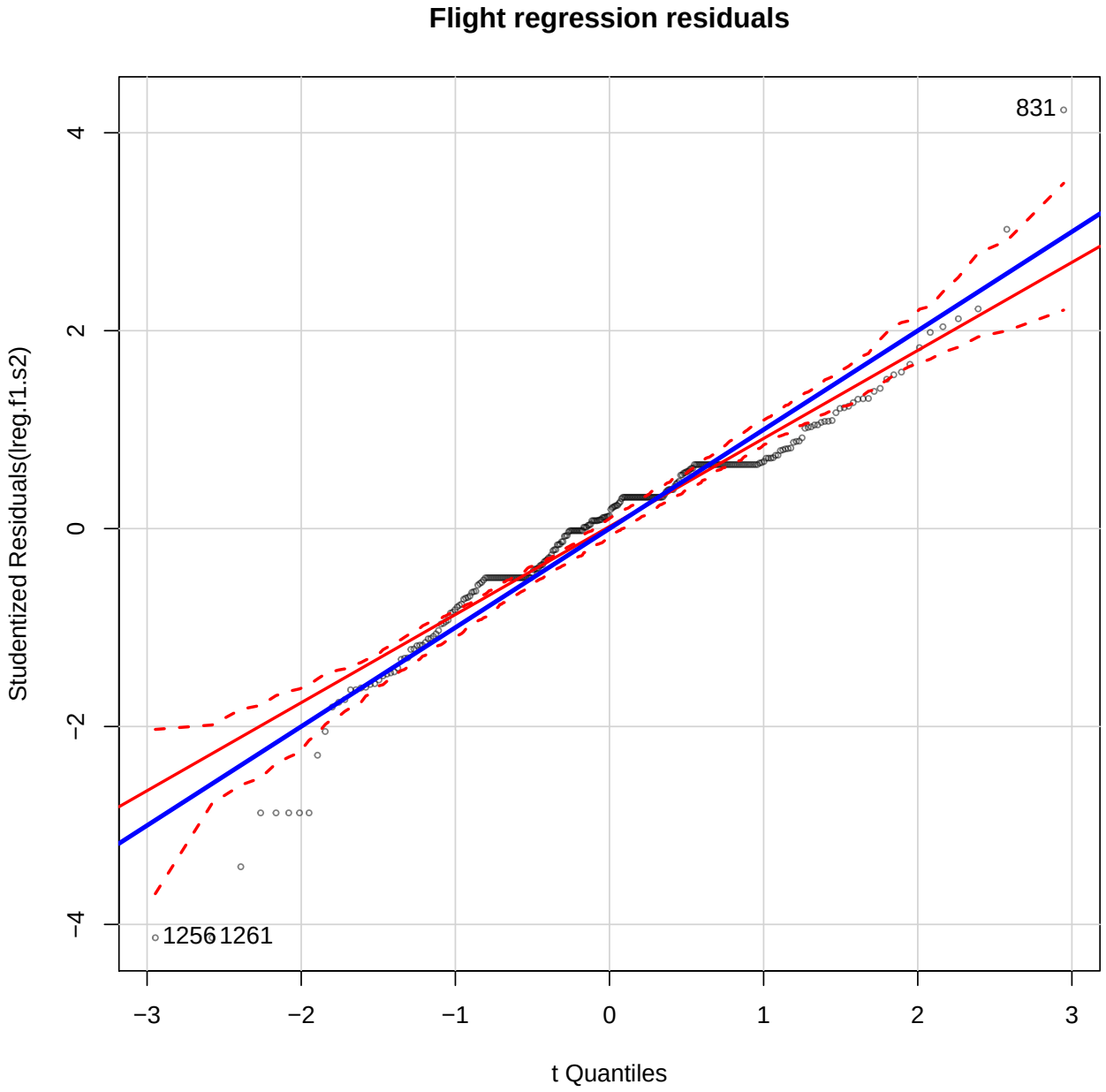
```
abline(a=0, b=1,col="blue",lwd=3)
```




```
qqPlot(lreg.f1.s2,
       main = "Flight regression residuals",
       pch  = 21,
       id.n = 3,
       cex = 0.5,
       col  = rgb(0, 0, 0, .5))
```

```
## 1256 1261  831
##    1    2  287
```

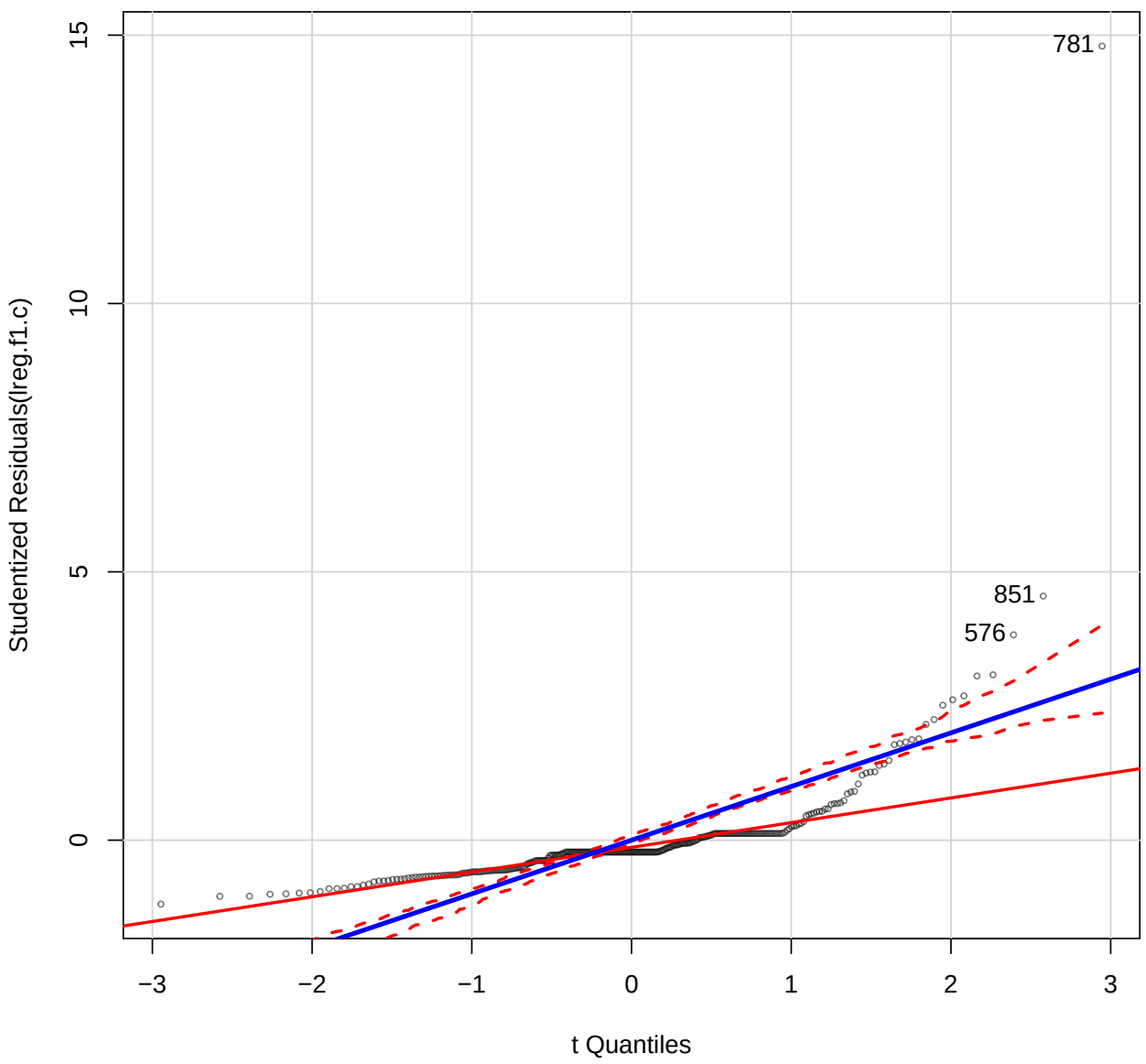
```
abline(a=0, b=1,col="blue",lwd=3)
```




```
qqPlot(lreg.f1.c,
       main = "Flight regression residuals",
       pch  = 21,
       id.n = 3,
       cex = 0.5,
       col  = rgb(0, 0, 0, .5))
```

```
## 576 851 781
## 285 286 287
```

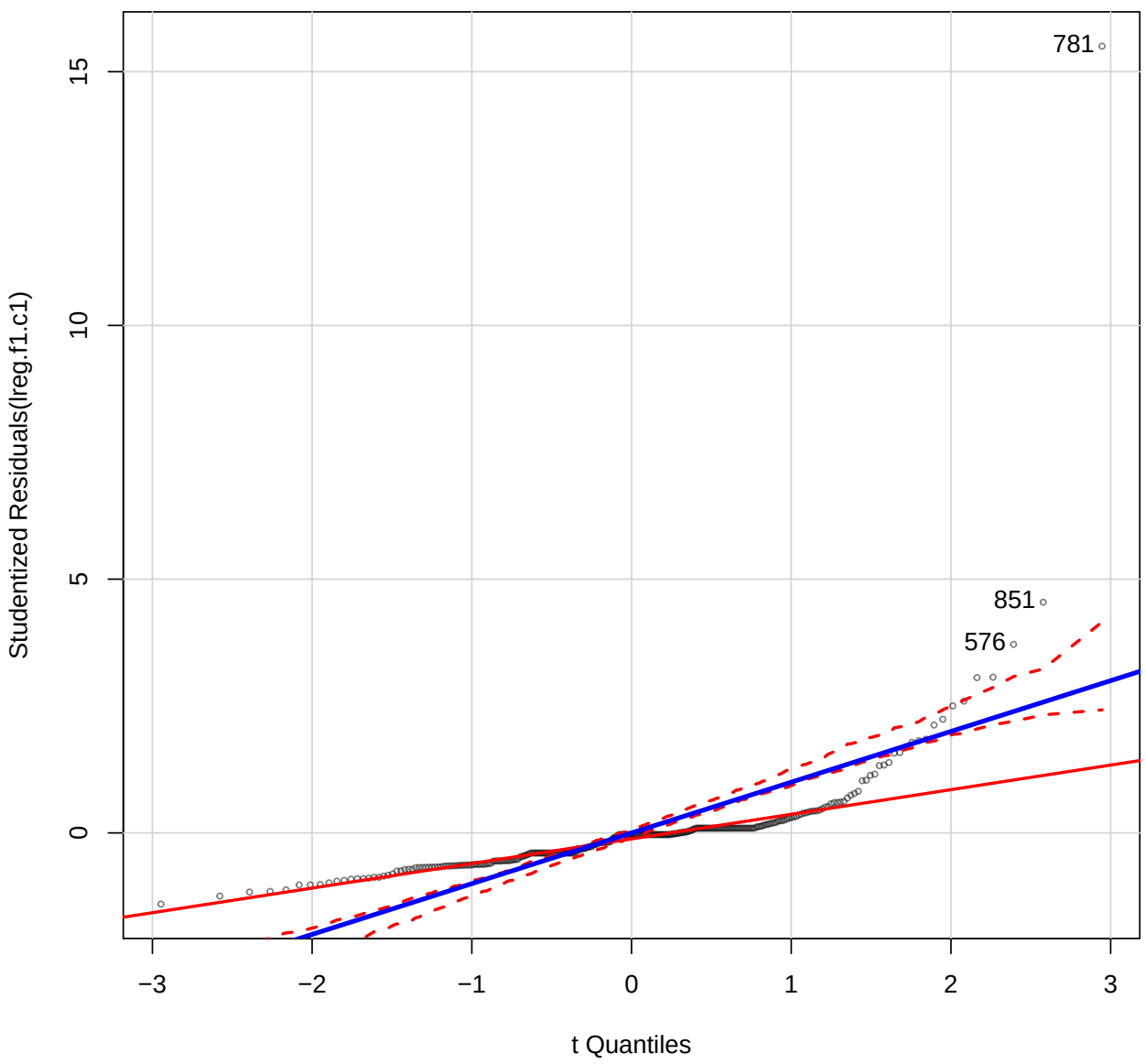
```
abline(a=0, b=1,col="blue",lwd=3)
```




```
qqPlot(lreg.f1.c1,
       main = "Flight regression residuals",
       pch  = 21,
       id.n = 3,
       cex = 0.5,
       col  = rgb(0, 0, 0, .5))
```

```
## 576 851 781
## 285 286 287
```

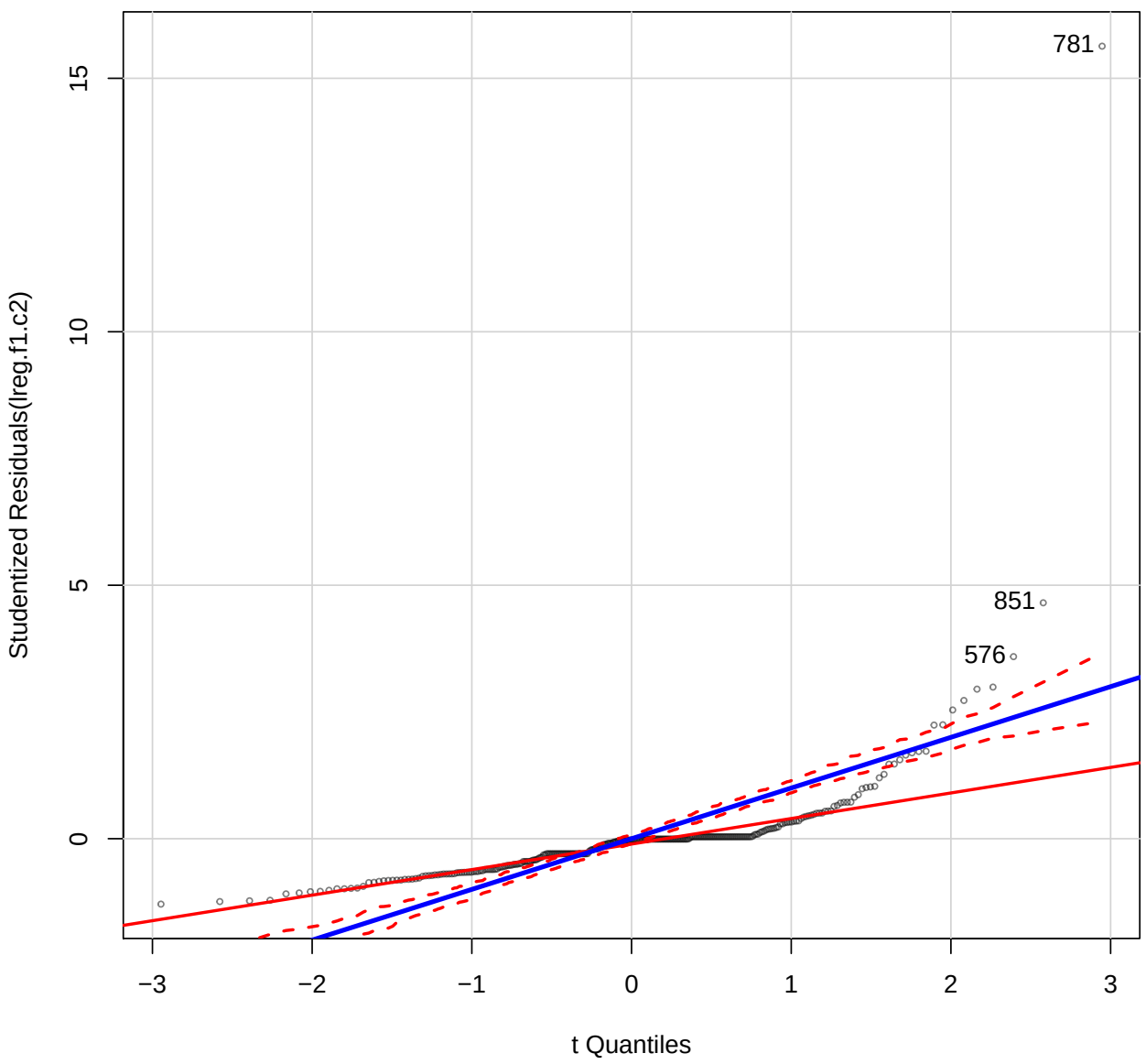
```
abline(a=0, b=1,col="blue",lwd=3)
```




```
qqPlot(lreg.f1.c2,
       main = "Flight regression residuals",
       pch  = 21,
       id.n = 3,
       cex = 0.5,
       col  = rgb(0, 0, 0, .5))
```

```
## 576 851 781
## 285 286 287
```

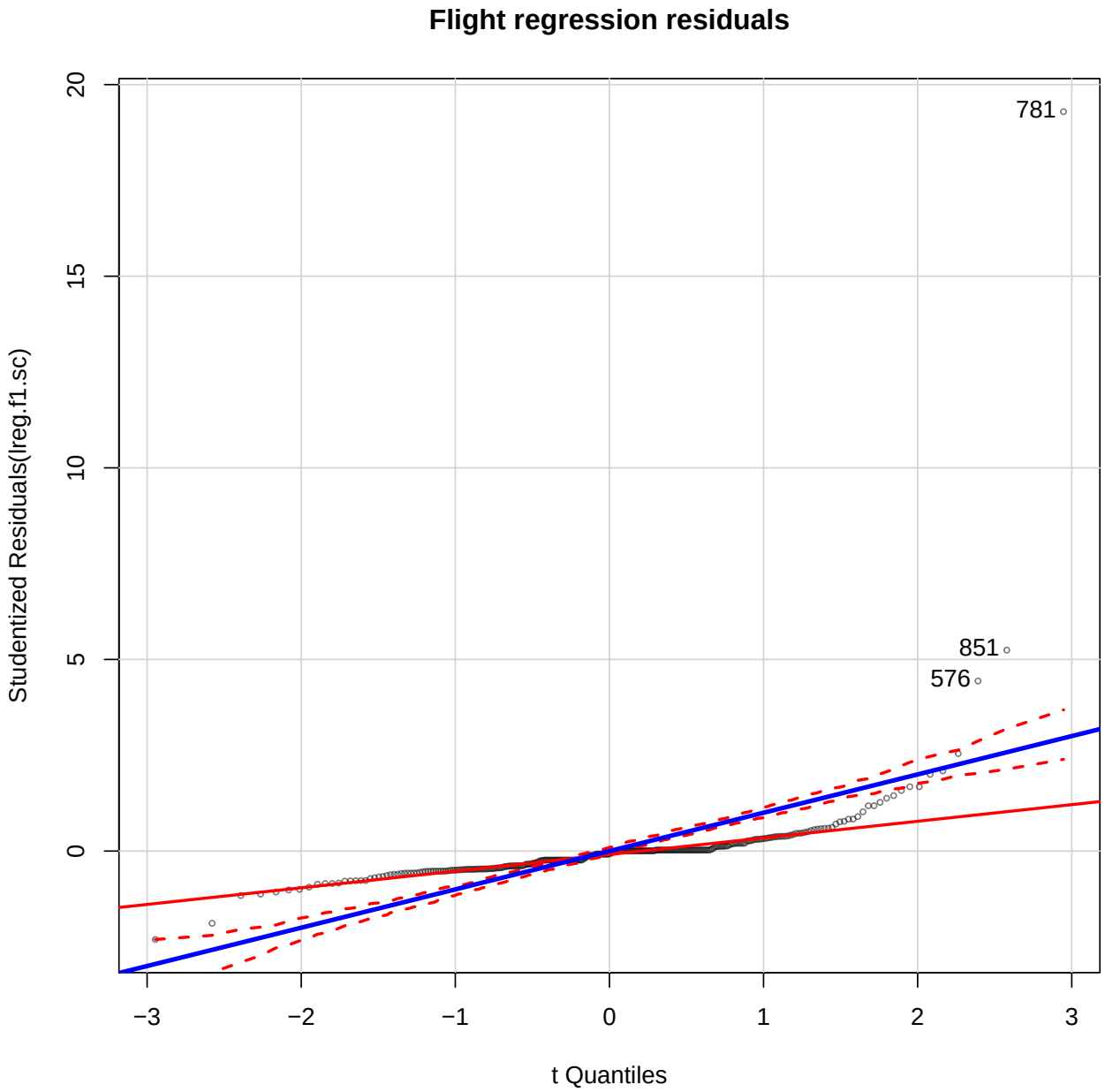
```
abline(a=0, b=1,col="blue",lwd=3)
```




```
qqPlot(lreg.f1.sc,
       main = "Flight regression residuals",
       pch  = 21,
       id.n = 3,
       cex = 0.5,
       col  = rgb(0, 0, 0, .5))
```

```
## 576 851 781
## 285 286 287
```

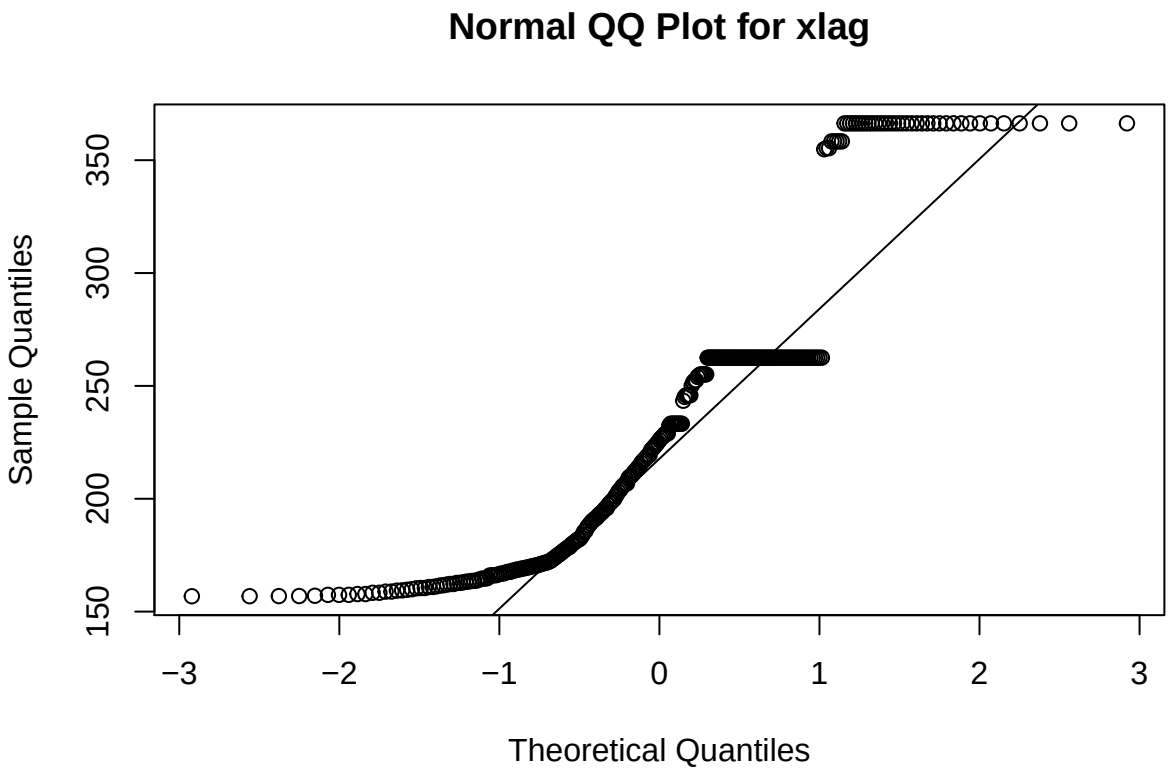
```
abline(a=0, b=1,col="blue",lwd=3)
```



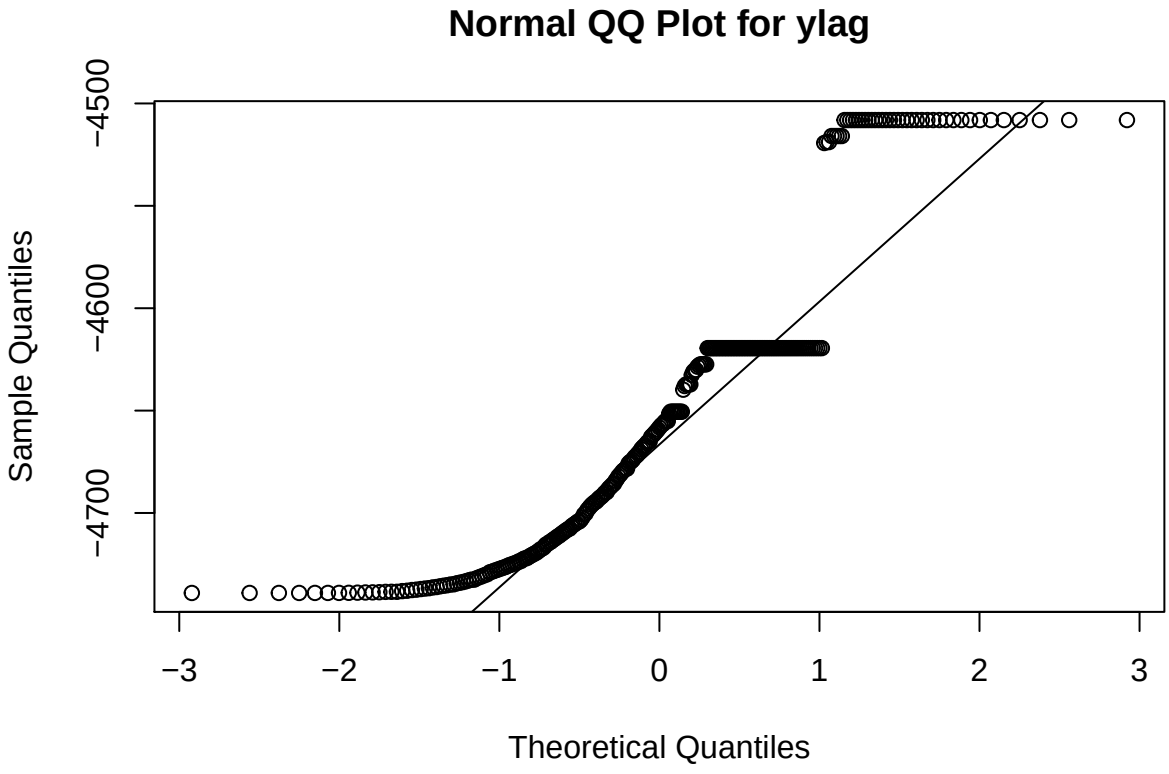

```
############ Influence Plot
#influenceIndexPlot(lreg.f1.s, id.n = 3)
#influenceIndexPlot(lreg.f1.s2, id.n = 3)
#influenceIndexPlot(lreg.f1.c, id.n = 3)
#influenceIndexPlot(lreg.f1.c1, id.n = 3)
#influenceIndexPlot(lreg.f1.c2, id.n = 3)
#influenceIndexPlot(lreg.f1.sc, id.n = 3)
```

E. QQPlot for Lag Terms

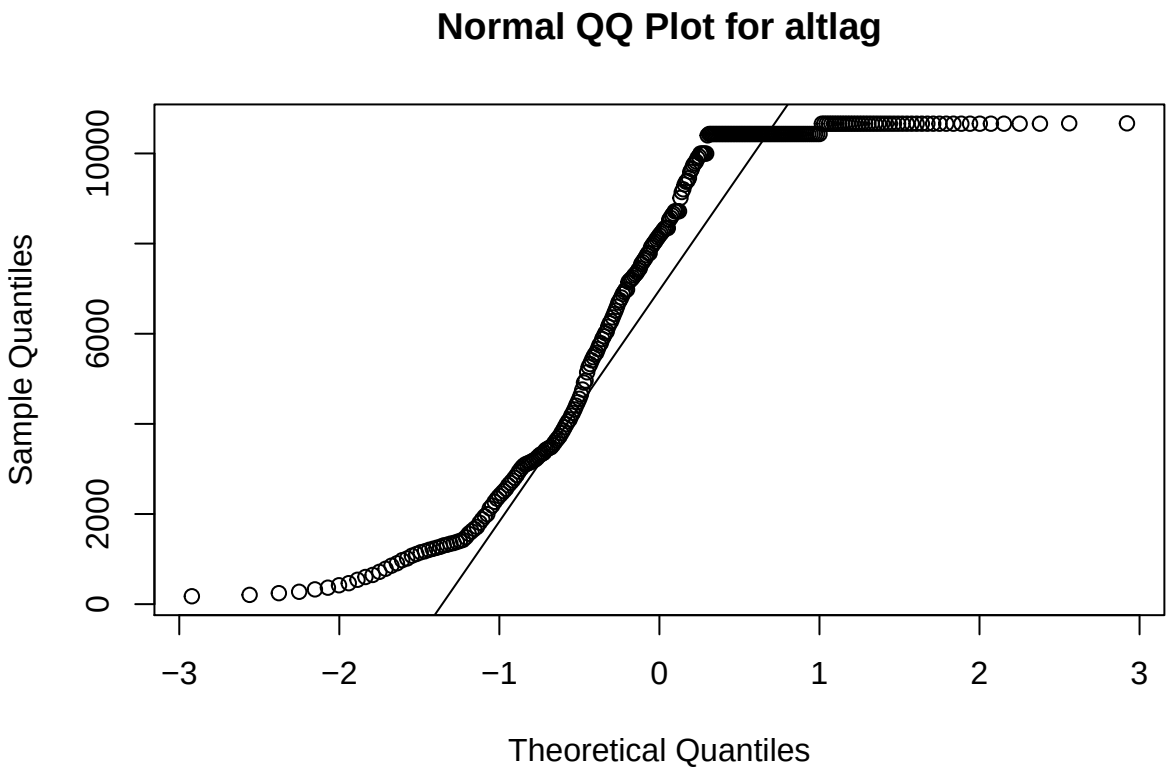
```
qqnorm(FDeval$xlag, main = "Normal QQ Plot for xlag")
qqline(FDeval$xlag)
```



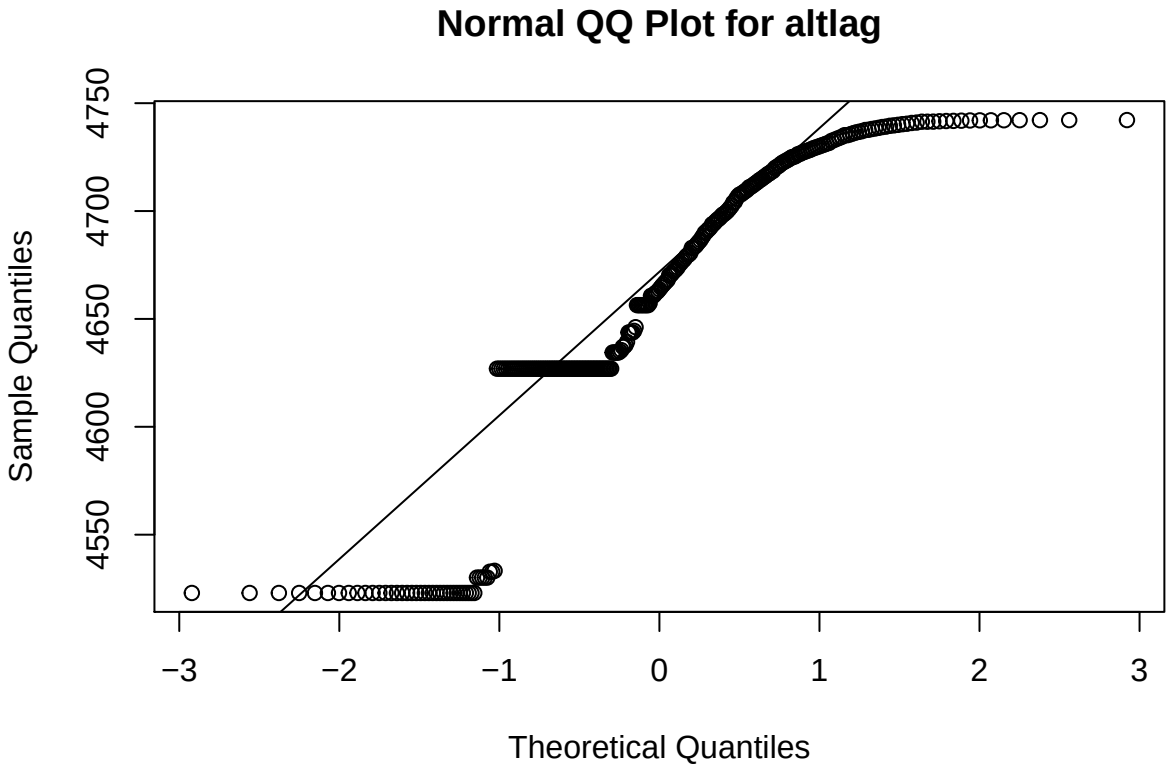

```
qqnorm(FDeval$ylag, main = "Normal QQ Plot for ylag")
qqline(FDeval$ylag)
```




```
qqnorm(FDeval$altlag, main = "Normal QQ Plot for altlag")
qqline(FDeval$altlag)
```

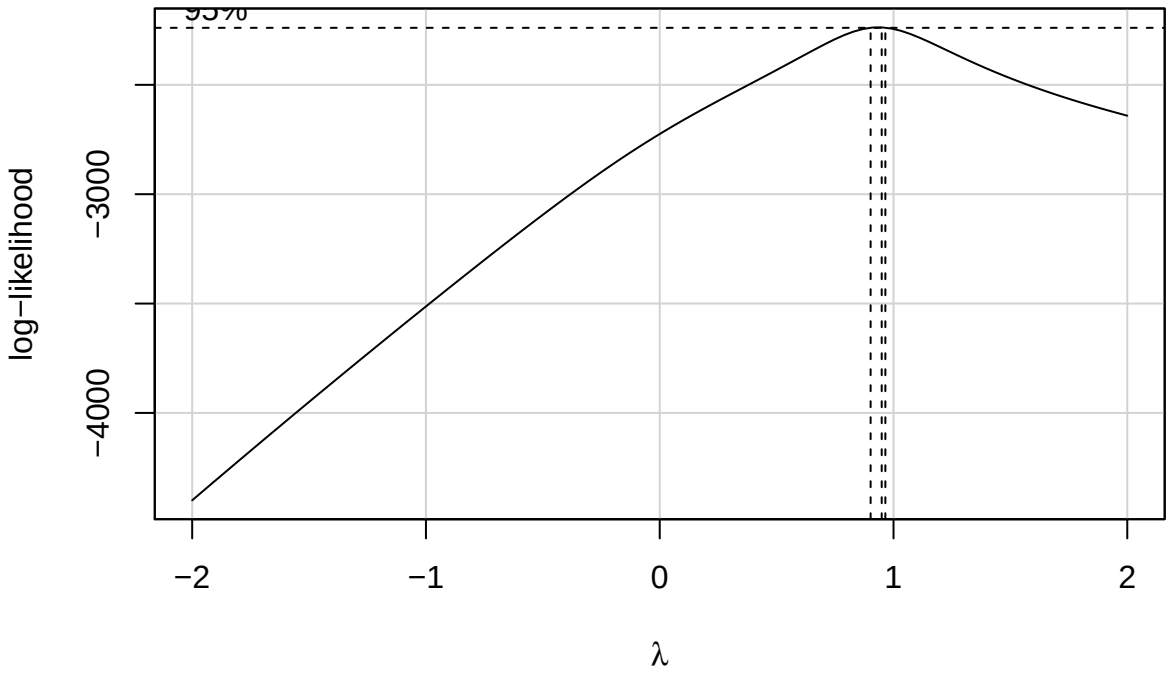



```
qqnorm(FDeval$distlag, main = "Normal QQ Plot for altlag")
qqline(FDeval$distlag)
```

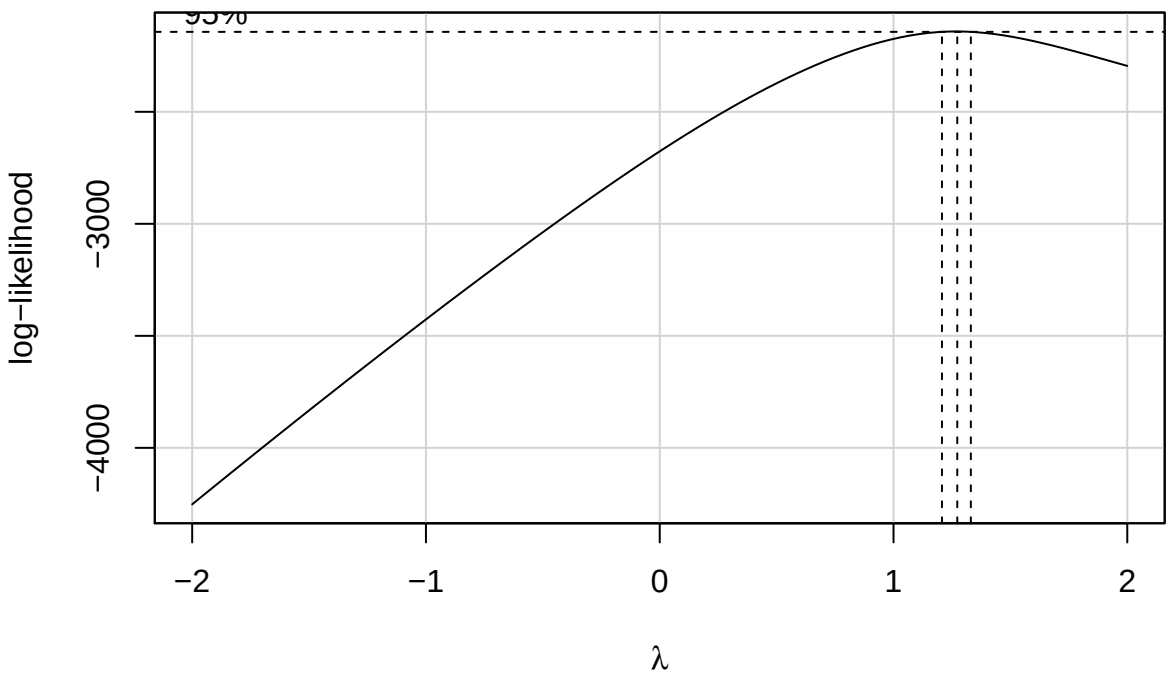


F. Dependent Variable Transform: Box Cox

```
# check boxCox
#lreg.f1.s <- lm(alt ~ x*y , data = FDeval)
bc1 <- boxCox(FDeval$alt ~ FDeval$x*FDeval$y, family = 'yjPower')
```

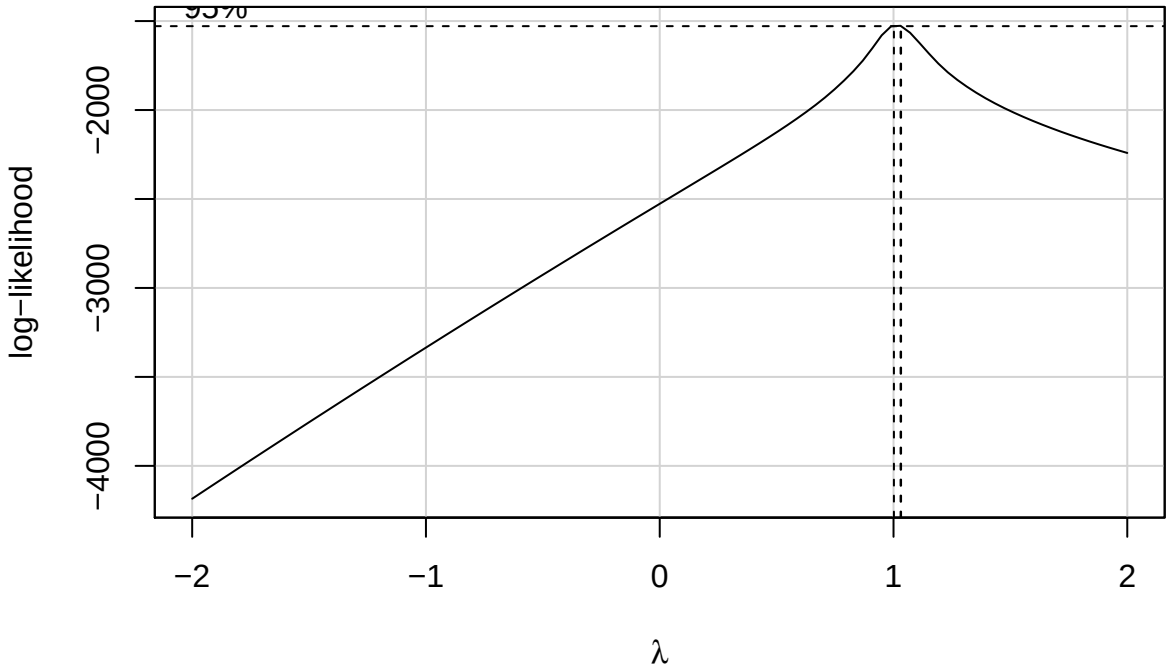



```
#lreg.f1.s2 <- lm(alt ~  x*y + distlag + distlag*speedlag, data = FDeval)
bc2 <- boxCox(FDeval$alt ~ FDeval$x*FDeval$y + FDeval$distlag + FDeval$distlag*FDeval$speedlag, family =
```




```
#lreg.f1.sc <- lm(alt ~ x*y + altlag + distlag*speedlag + altlag*vspeedlag , data = FDeval)
bc3 <- boxCox(FDeval$alt ~ FDeval$x*FDeval$y + FDeval$altlag +
```

```
                        FDeval$dist*FDeval$speedlag +
                        FDeval$altlag*FDeval$vspeedlag, family = 'yjPower')
```




```
lambda1 <- bc1$x[bc1$y == max(bc1$y)]
print(lambda1)
```

```
## [1] 0.9494949
```

```
lambda2 <- bc2$x[bc2$y == max(bc2$y)]
print(lambda2)
```

```
## [1] 1.272727
```

```
lambda3 <- bc3$x[bc3$y == max(bc3$y)]
print(lambda3)
```

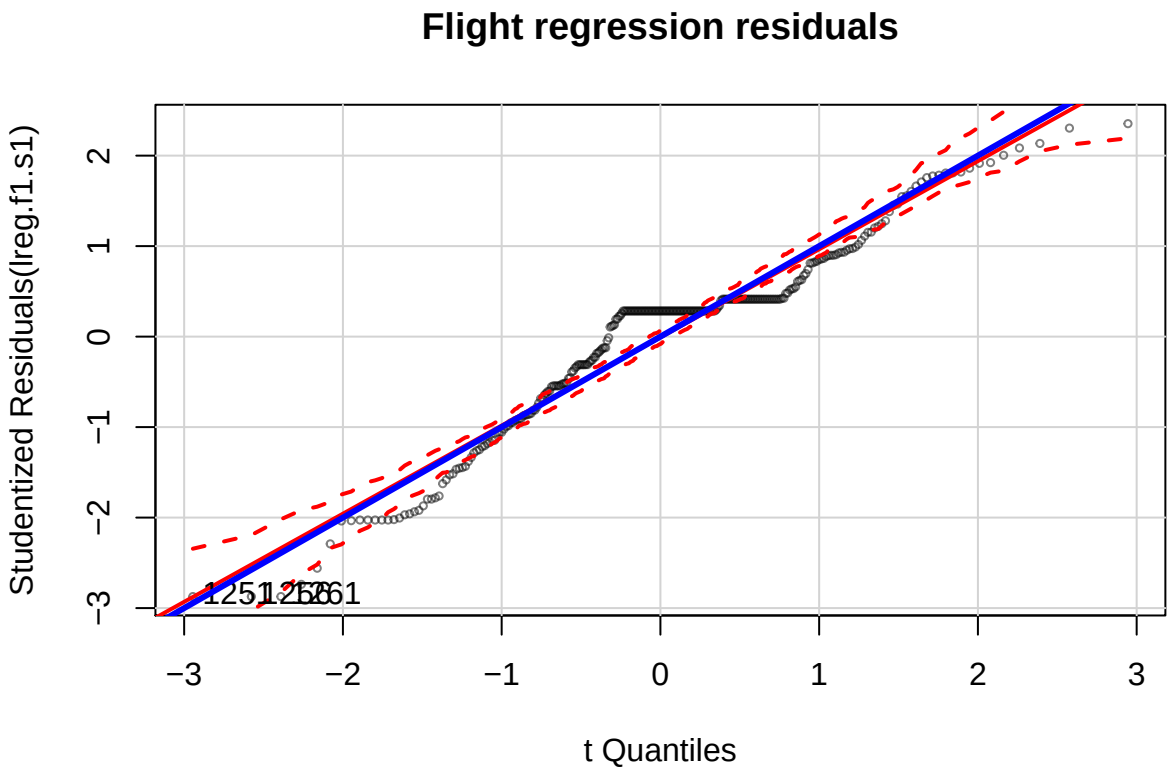
```
## [1] 1.030303
```

```
#FDeval$Trans.alt1 <- (FDeval$x^(lambda1)-1)/(lambda1)
#FDeval$Trans.alt2 <- (FDeval$x^(lambda2)-1)/(lambda2)
lambda1 <-0.9
lambda2 <-1.2
lambda3 <-1

FDeval <- FDeval[row.names(FDeval) != "231" & row.names(FDeval) != "451",]

lreg.f1.s1 <- lm((alt^(lambda1)-1)/(lambda1) ~ x*y , data = FDeval)
lreg.f1.s2 <- lm((alt^(lambda2)-1)/(lambda2) ~ x*y + distlag + distlag*speedlag, data = FDeval)
lreg.f1.sc <- lm((alt^(lambda3)-1)/(lambda3) ~ x*y + altlag + distlag*speedlag + altlag*vspeedlag, data


qqPlot(lreg.f1.s1,
       main = "Flight regression residuals",
       pch  = 21,
```

```
       id.n = 3,
       cex = 0.5,
       col  = rgb(0, 0, 0, .5))
```

```
## 1251 1256 1261
##    1    2    3
```

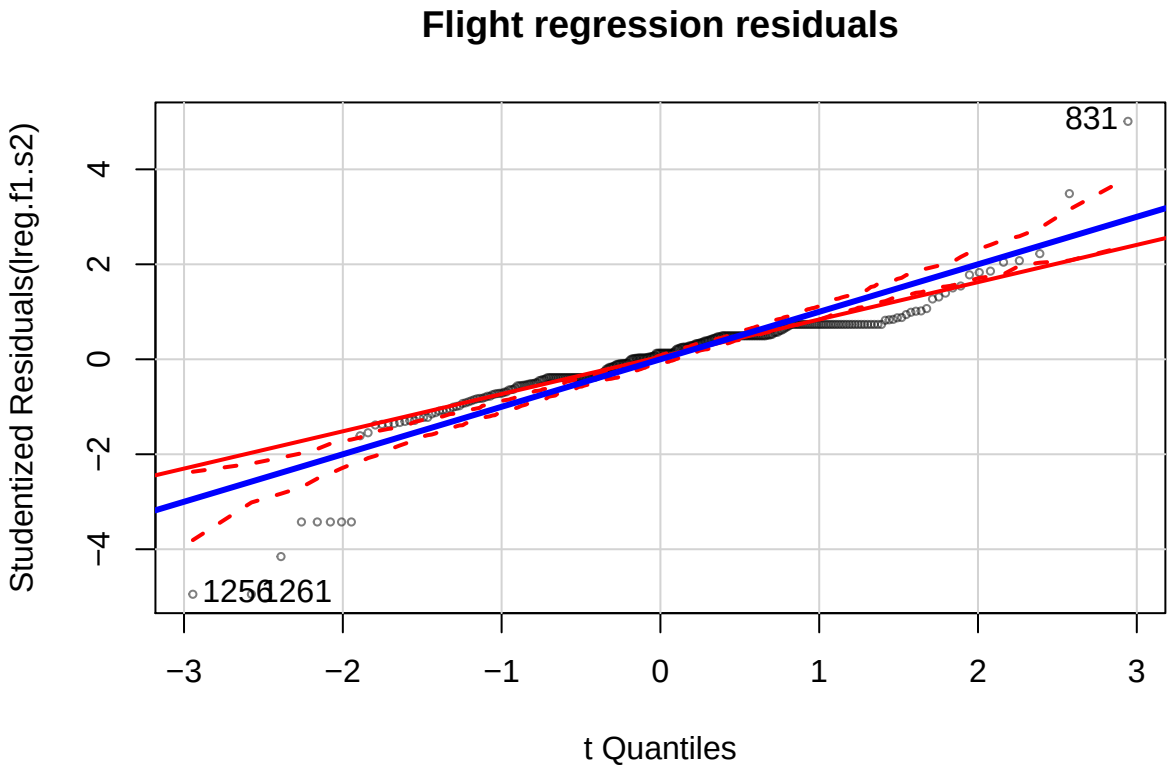
```
abline(a=0, b=1,col="blue",lwd=3)
```




```
qqPlot(lreg.f1.s2,
       main = "Flight regression residuals",
       pch  = 21,
       id.n = 3,
       cex = 0.5,
       col  = rgb(0, 0, 0, .5))
```

```
## 1256 1261  831
##    1    2  285
```

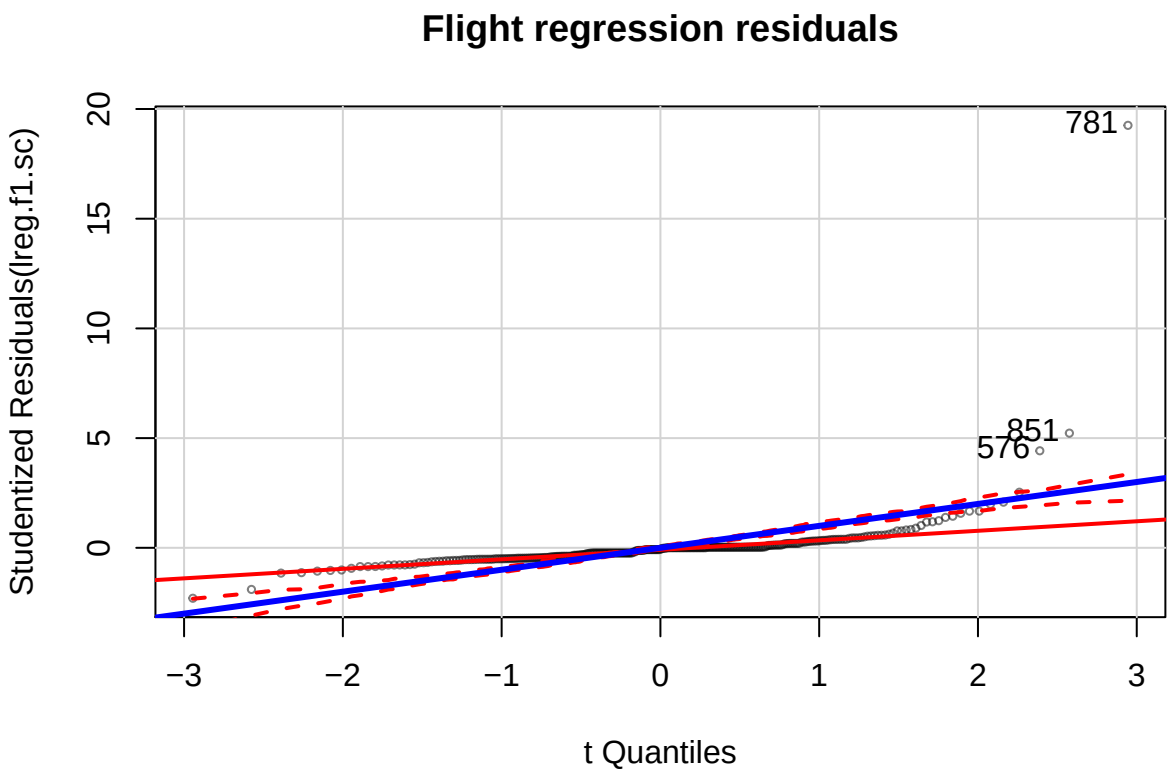
```
abline(a=0, b=1,col="blue",lwd=3)
```




```
qqPlot(lreg.f1.sc,
       main = "Flight regression residuals",
       pch  = 21,
       id.n = 3,
       cex = 0.5,
       col  = rgb(0, 0, 0, .5))
```

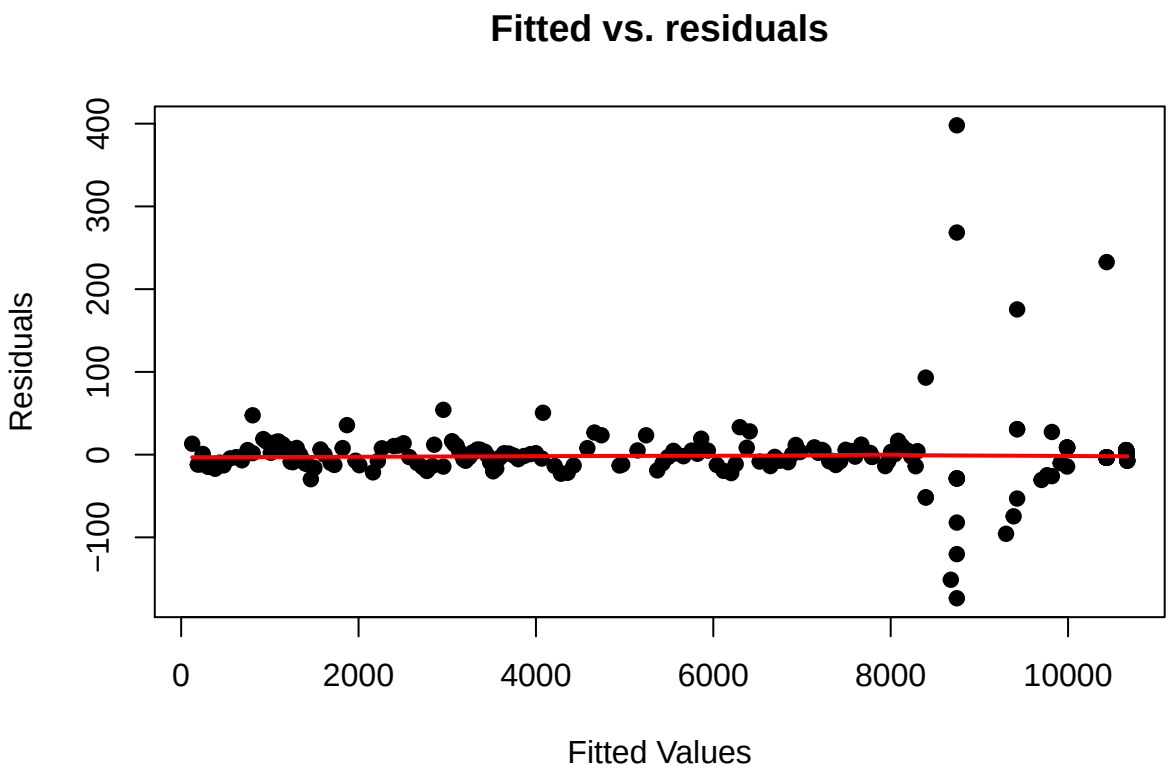
```
## 576 851 781
## 283 284 285
```

```
abline(a=0, b=1,col="blue",lwd=3)
```

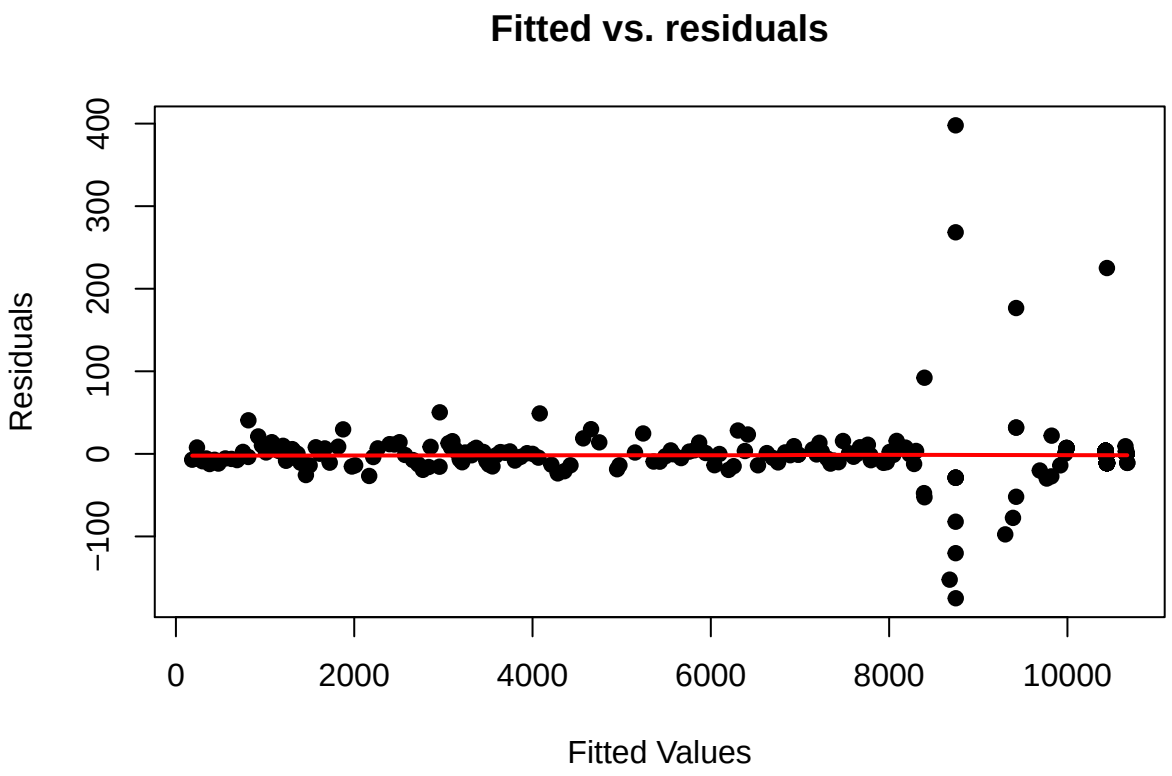


G. GAM

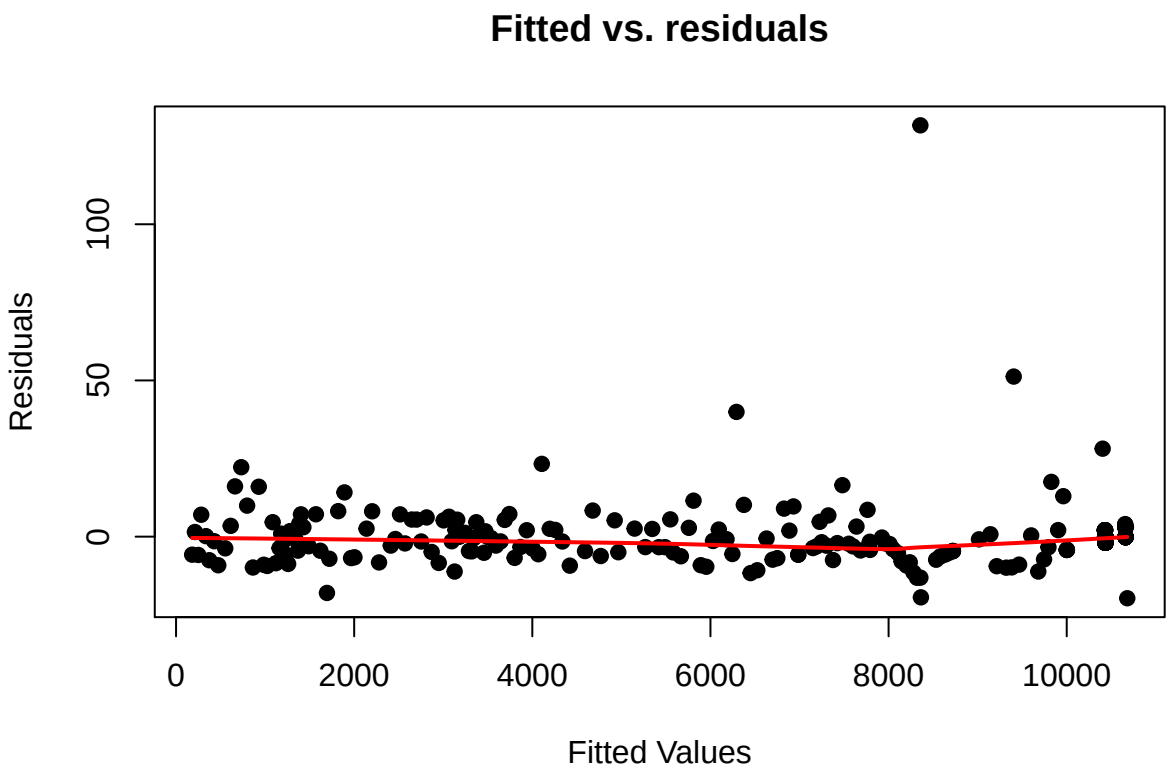
```
# Before Transform
gam.s <- gam(alt ~ s(x,y), data = FDeval)
gam.s2 <- gam(alt ~ s(x,y) + s(distlag) + s(distlag,speedlag), data = FDeval)
gam.sc <- gam(alt ~ s(x,y) + s(altlag) + s(distlag,speedlag) +
               s(altlag,vspeedlag), data = FDeval)

plot(fitted(gam.s), resid(gam.s),
     xlab = "Fitted Values",
     ylab = "Residuals",
     pch = 19,
     col = "black",
     main = "Fitted vs. residuals")
lines(lowess(fitted(gam.s), resid(gam.s)), col = "red", lwd = 2)
```



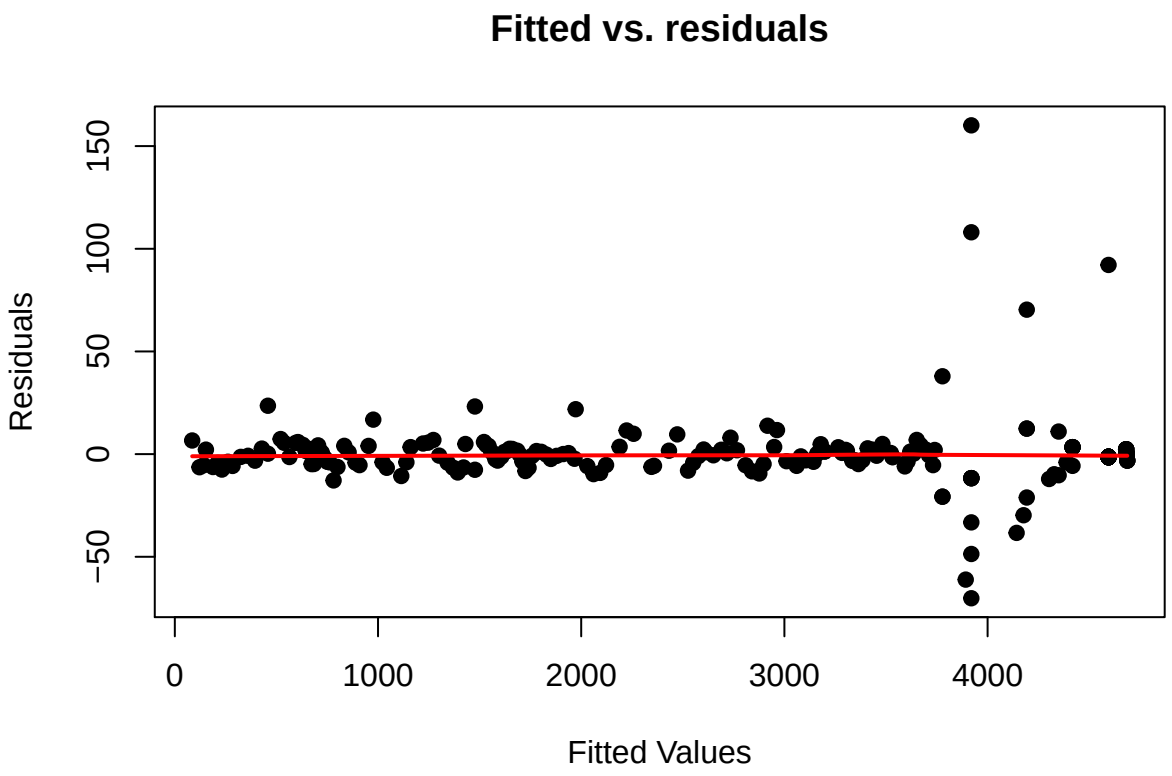

```
plot(fitted(gam.s2), resid(gam.s2),
     xlab = "Fitted Values",
     ylab = "Residuals",
     pch = 19,
     col = "black",
     main = "Fitted vs. residuals")
lines(lowess(fitted(gam.s2), resid(gam.s2)), col = "red", lwd = 2)
```



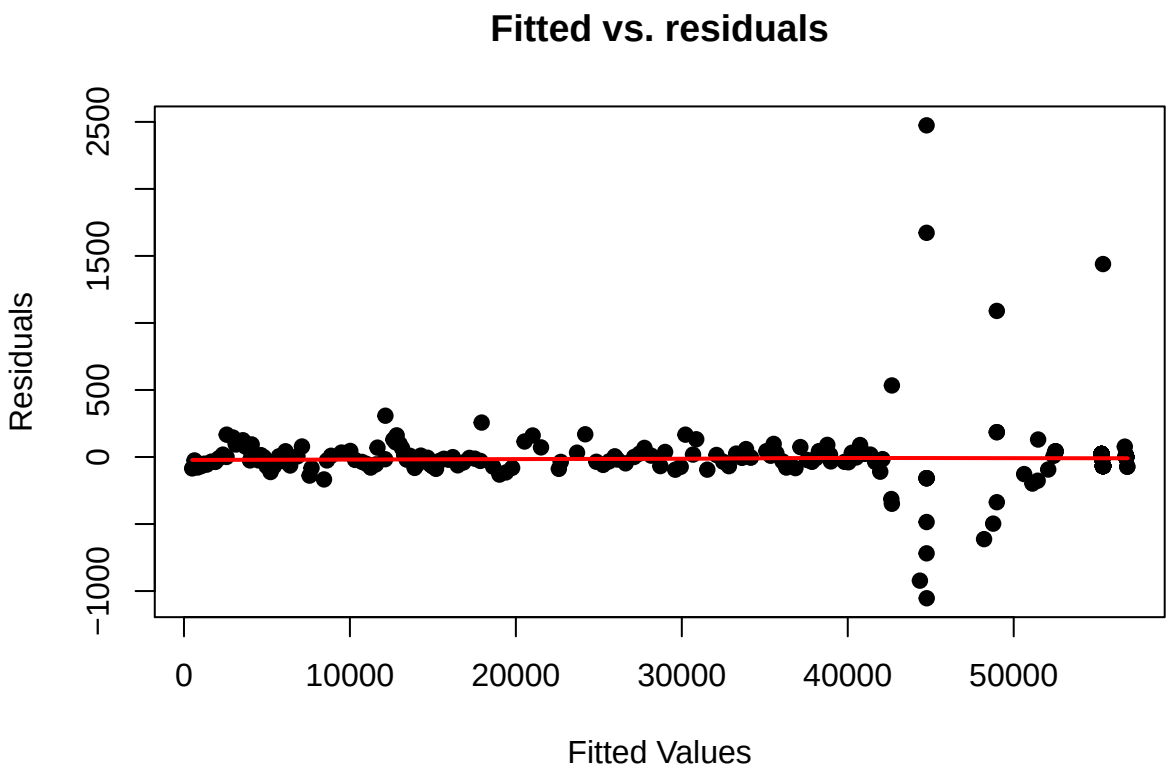

```
plot(fitted(gam.sc), resid(gam.sc),
     xlab = "Fitted Values",
     ylab = "Residuals",
     pch = 19,
     col = "black",
     main = "Fitted vs. residuals")
lines(lowess(fitted(gam.sc), resid(gam.sc)), col = "red", lwd = 2)
```



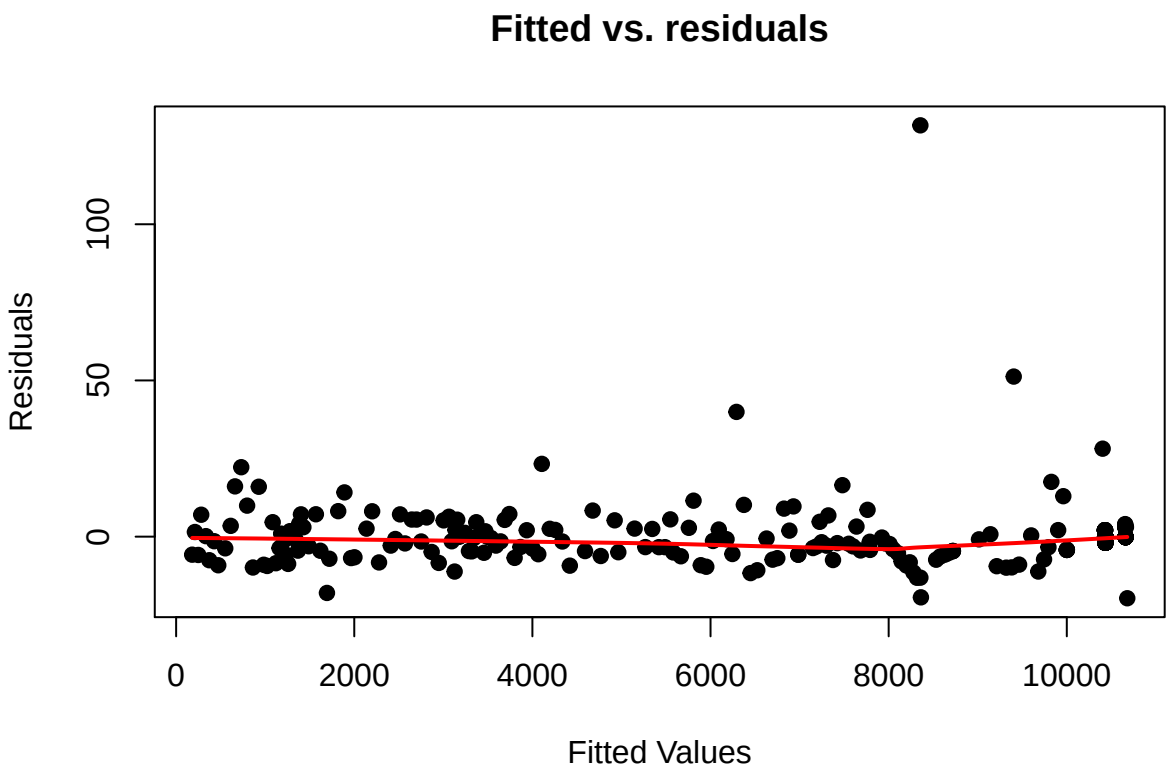

```
# After Transform
gam.s <- gam((alt^(lambda1)-1)/(lambda1) ~ s(x,y), data = FDeval)
gam.s2 <- gam((alt^(lambda2)-1)/(lambda2) ~ s(x,y) + s(distlag) + s(distlag,speedlag) , data = FDeval)
gam.sc <- gam((alt^(lambda3)-1)/(lambda3) ~ s(x,y) + s(altlag) + s(distlag,speedlag) +
                  s(altlag,vspeedlag), data = FDeval)

plot(fitted(gam.s), resid(gam.s),
     xlab = "Fitted Values",
     ylab = "Residuals",
     pch = 19,
     col = "black",
     main = "Fitted vs. residuals")
lines(lowess(fitted(gam.s), resid(gam.s)), col = "red", lwd = 2)
```




```r
plot(fitted(gam.s2), resid(gam.s2),
     xlab = "Fitted Values",
     ylab = "Residuals",
     pch = 19,
     col = "black",
     main = "Fitted vs. residuals")
lines(lowess(fitted(gam.s2), resid(gam.s2)), col = "red", lwd = 2)
```

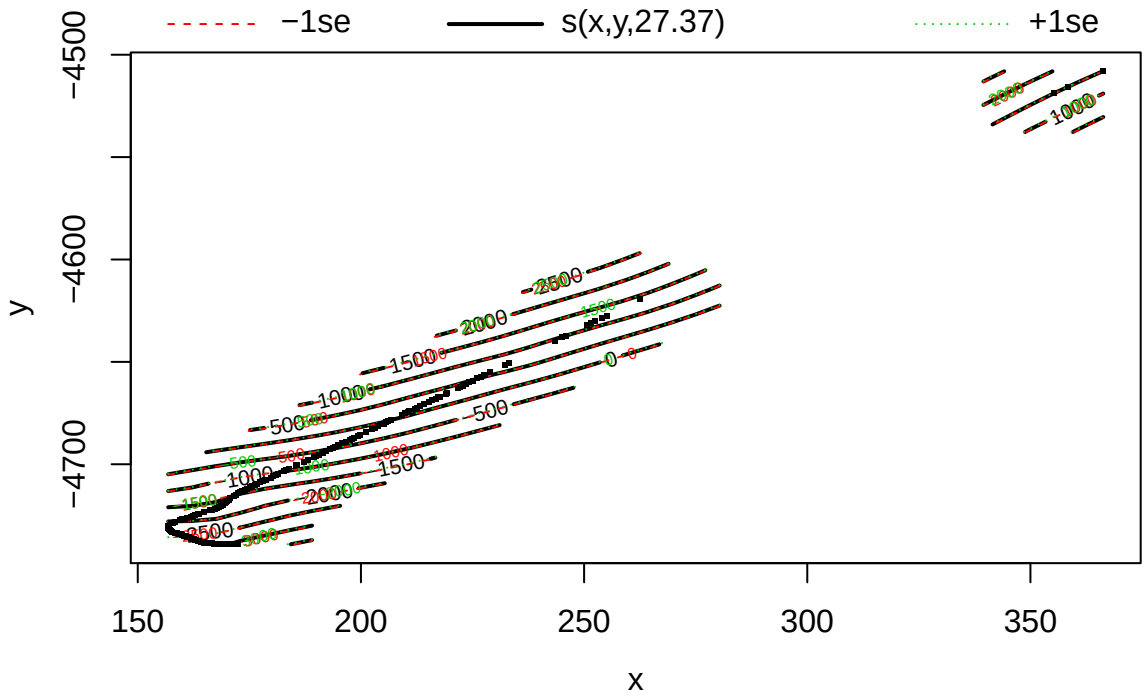



```r
plot(fitted(gam.sc), resid(gam.sc),
     xlab = "Fitted Values",
     ylab = "Residuals",
     pch = 19,
     col = "black",
     main = "Fitted vs. residuals")
lines(lowess(fitted(gam.sc), resid(gam.sc)), col = "red", lwd = 2)
```

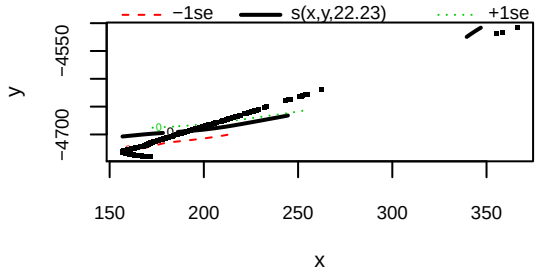


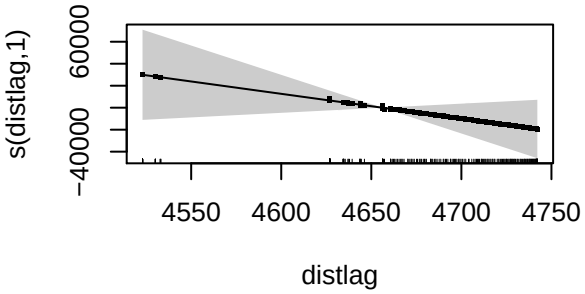
Partial residuals

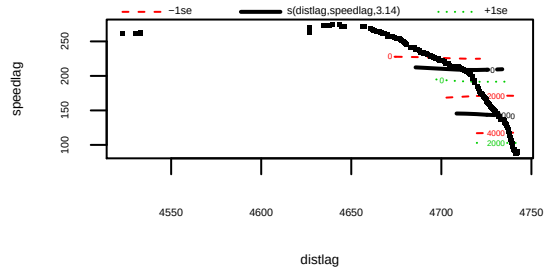
```
plot(gam.s,
     residuals = TRUE,  # Include the partial residuals
     shade= TRUE,  # Include shaded confidence bands
     pages= 1,
     scale= 0,
     cex= 3)
```




```
plot(gam.s2,
     residuals = TRUE,  # Include the partial residuals
     shade= TRUE,  # Include shaded confidence bands
     pages= 1,
     scale= 0,
     cex= 3)
```

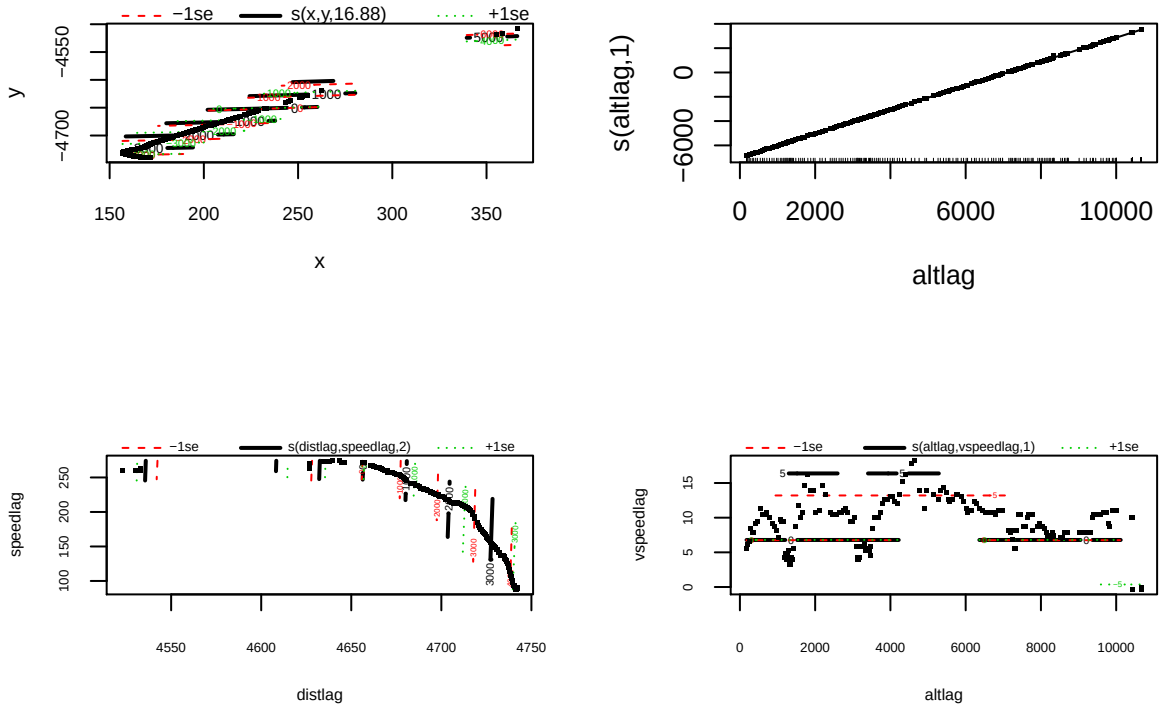







```
plot(gam.sc,
     residuals = TRUE,  # Include the partial residuals
```

```
     shade= TRUE,  # Include shaded confidence bands
     pages= 1,
     scale= 0,
     cex= 3)
```

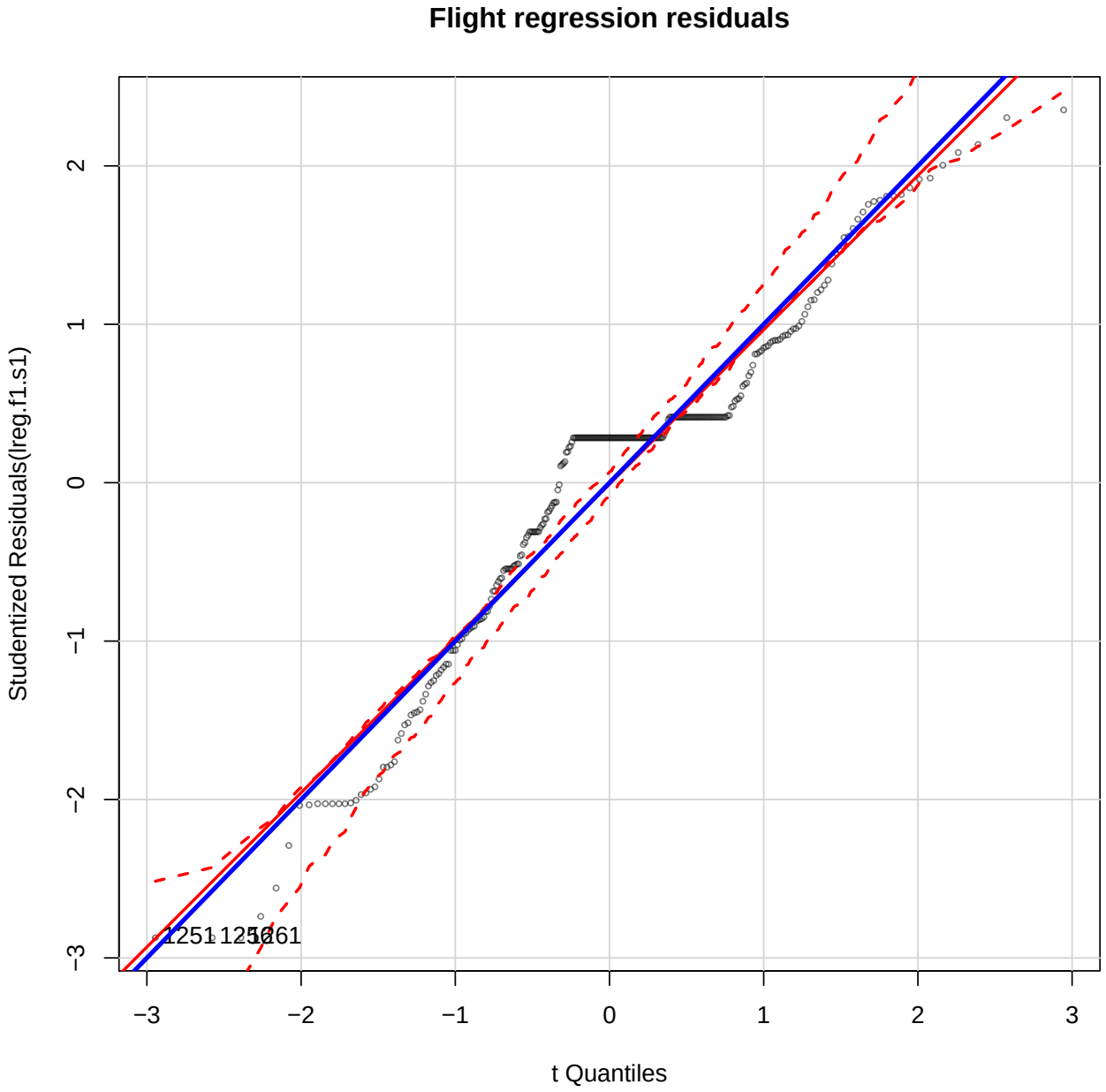


H. Idependent Variable Transforms

```
lreg.f1.s1 <- lm((alt^(lambda1)-1)/(lambda1) ~ x*y , data = FDeval)
lreg.f1.s2 <- lm((alt^(lambda2)-1)/(lambda2) ~ x*y + distlag + distlag*speedlag, data = FDeval)
lreg.f1.sc <- lm((alt^(lambda3)-1)/(lambda3) ~ x*y + I(altlag^2) + distlag*speedlag + altlag*vspeedlag,

#vary.slope <- lmer(Trans.alt1 ~ log(x) + I(y^2) + dist*speedlag + (1 + dist|icao_addr), data = fpoints_

############ QQPLOT
qqPlot(lreg.f1.s1,
       main = "Flight regression residuals",
       pch  = 21,
       id.n = 3,
       cex = 0.5,
       col  = rgb(0, 0, 0, .5))
```

```
## 1251 1256 1261
##    1    2    3
```

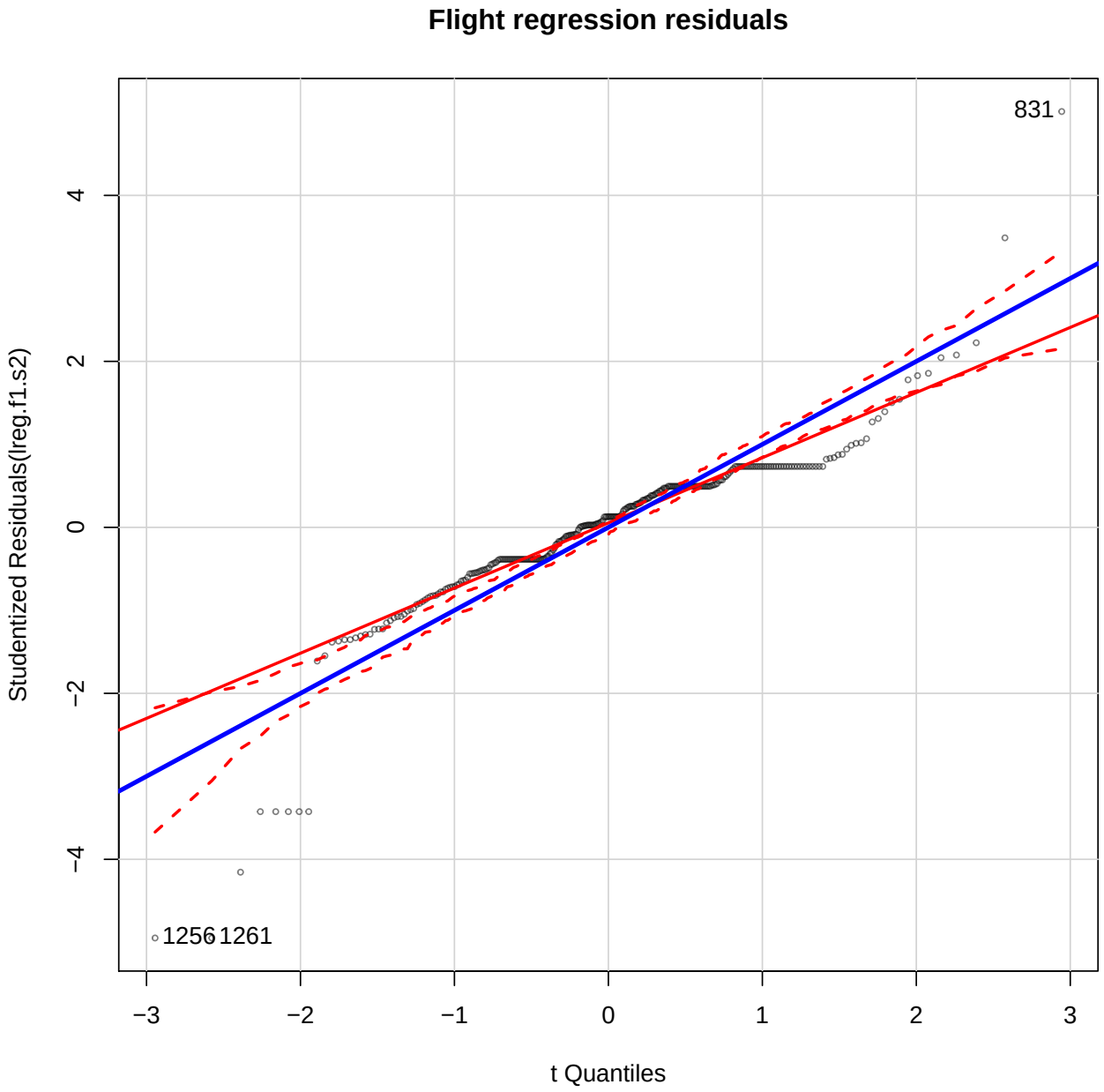
```
abline(a=0, b=1,col="blue",lwd=3)
```




```
qqPlot(lreg.f1.s2,
       main = "Flight regression residuals",
       pch  = 21,
       id.n = 3,
       cex = 0.5,
       col  = rgb(0, 0, 0, .5))
```

```
## 1256 1261  831
##    1    2  285
```

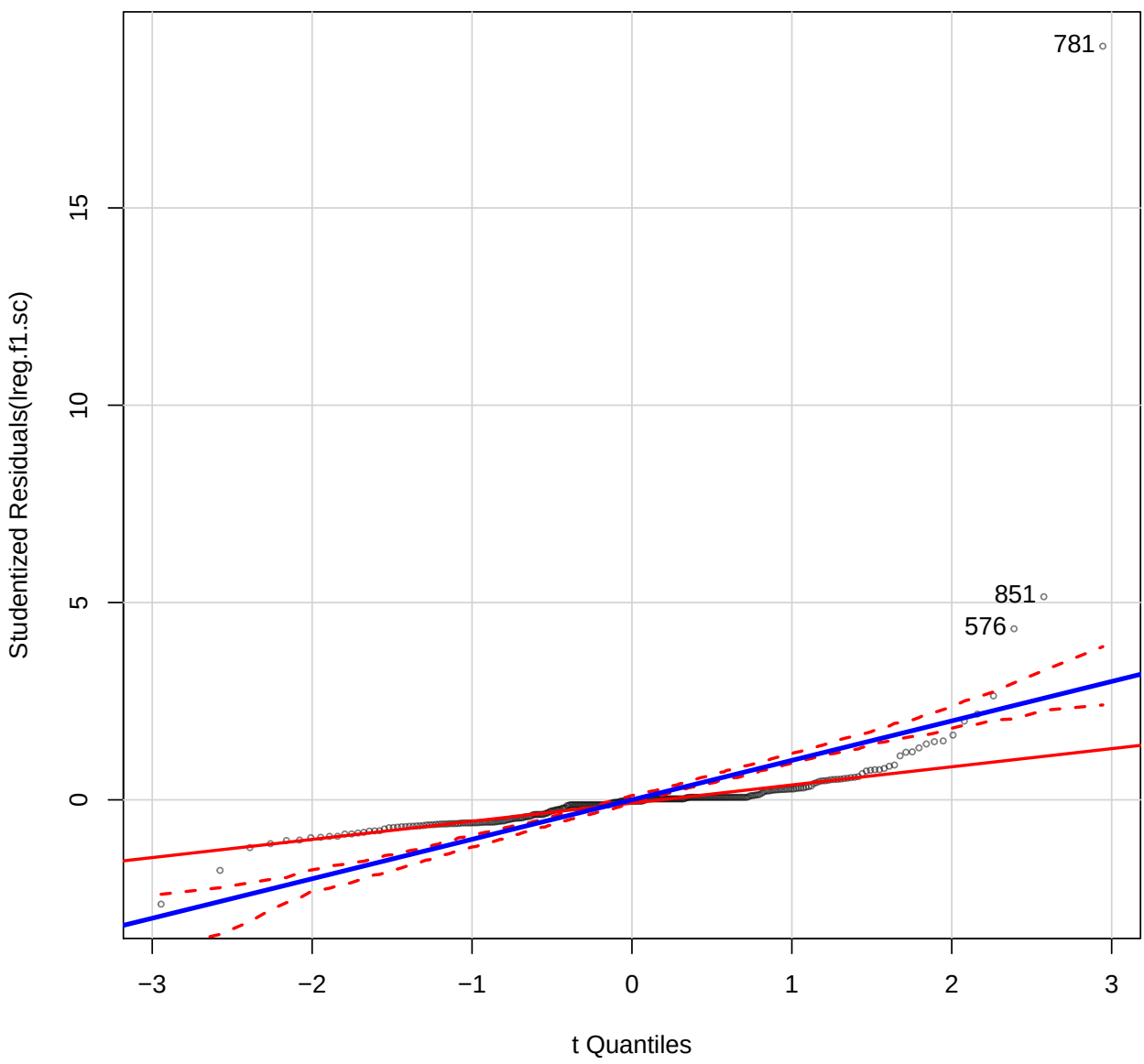
```
abline(a=0, b=1,col="blue",lwd=3)
```




```
qqPlot(lreg.f1.sc,
       main = "Flight regression residuals",
       pch  = 21,
       id.n = 3,
       cex = 0.5,
       col  = rgb(0, 0, 0, .5))
```

```
## 576 851 781
## 283 284 285
```

```
abline(a=0, b=1,col="blue",lwd=3)
```




```
#summary(vary.slope)
```

Final Summary

```
lreg.f1.s1 <- lm((alt^(lambda1)-1)/(lambda1) ~ x*y , data = FDeval)
lreg.f1.s2 <- lm((alt^(lambda2)-1)/(lambda2) ~ x*y + distlag + distlag*speedlag, data = FDeval)
lreg.f1.sc <- lm((alt^(lambda3)-1)/(lambda3) ~ x*y + I(altlag^2) + distlag*speedlag + altlag*vspeedlag,


kable(summary(lreg.f1.s1)$coeff[,1:2])
```

	Estimate	Std. Error
(Intercept)	4.217677e+05	3630.2046224

	Estimate	Std. Error
x	-7.315724e+02	3.3834419
y	8.834004e+01	0.7389676
x:y	-1.508582e-01	0.0007715

```
kable(summary(lreg.f1.s2)$coeff[,1:2])
```

	Estimate	Std. Error
(Intercept)	6.449099e+06	1.488344e+05
x	-1.168684e+04	2.471388e+02
y	7.416612e+02	3.338954e+02
distlag	-6.116891e+02	3.271865e+02
speedlag	-1.942596e+02	3.748917e+02
x:y	-2.434789e+00	4.981760e-02
distlag:speedlag	2.403040e-02	7.879390e-02

```
kable(summary(lreg.f1.sc)$coeff[,1:2])
```

	Estimate	Std. Error
(Intercept)	4577.5157136	9048.1876222
x	-34.5307269	15.1200692
y	43.7034625	6.4036691
I(altlag^2)	-0.0000007	0.0000004
distlag	42.7582891	6.3711553
speedlag	14.8836125	9.7595002
altlag	1.0048751	0.0065452
vspeedlag	1.3990636	0.4845524
x:y	-0.0070168	0.0031276
distlag:speedlag	-0.0031783	0.0020619
altlag:vspeedlag	-0.0001290	0.0000618

I. Forcasting

1. Cross Validation

```
crossvalidate <- function(n){
nfolds <- n
case.folds <- rep(1:nfolds, length.out = nrow(FDtrain))

Model1 <- c()
Model2 <- c()
Model3 <- c()


for (fold in 1:nfolds) {

  # Create training and test cases
  train <- FDtrain[case.folds != fold, ]
```

```
  test <- FDtrain[case.folds == fold, ]

  #train Model
  train1 <- lm((alt^(lambda1)-1)/(lambda1) ~ x*y , data = train)

  train2 <- lm((alt^(lambda2)-1)/(lambda2) ~ x*y + distlag + distlag*speedlag, data = train)

  train3 <- lm((alt^(lambda3)-1)/(lambda3) ~ x*y + I(altlag^2) + distlag*speedlag + altlag*vspeedlag, da


  # Generate test MSEs
  test1 <- ((test$alt^(lambda1)-1)/(lambda1) - predict(train1, test))^2
  rMSEtest1 <- sqrt(sum(test1) / length(test1))

  test2 <- ((test$alt^(lambda2)-1)/(lambda2) - predict(train2, test))^2
  rMSEtest2 <- sqrt(sum(test2) / length(test2))

  test3 <- ((test$alt^(lambda3)-1)/(lambda3) - predict(train3, test))^2
  rMSEtest3 <- sqrt(sum(test3) / length(test3))


  # Append the rMSE from this iteration to vectors
  Model1 <- c(Model1, rMSEtest1)
  Model2 <- c(Model2, rMSEtest2)
  Model3 <- c(Model3, rMSEtest3)
}

# Average the MSEs
Model1.avg <- mean(Model1)
Model2.avg <- mean(Model2)
Model3.avg <- mean(Model3)


return(c(Model1.avg, Model2.avg, Model3.avg))
}


MSE.result.5fold <- crossvalidate(5)
print("      Model 1        |       Model 2     |       Model 3   ")
```

```
## [1] "      Model 1        |       Model 2     |       Model 3   "
```

```
print(paste0(MSE.result.5fold[1]," | ",MSE.result.5fold[2]," | ",MSE.result.5fold[3]))
```

```
## [1] "59.1199869164474 | 572.045684970143 | 55.8464949192663"
```

2. Model Testing

Finally, we take the first 80% of the data and train our models and test it with the last 20% of the data.

```
#rMSE for Model 1
Model1.train <- lm((alt^(lambda1)-1)/(lambda1) ~ x*y, data = FDtrain.for.Model)
Model1.test <- ((FDtest$alt^(lambda1)-1)/(lambda1) - predict(Model1.train,FDtest))^2
Model1.rMSEtest <- sqrt(sum(Model1.test) / length(Model1.test))
```

```
#rMSE for Model 2
Model2.train <- lm((alt^(lambda2)-1)/(lambda2) ~ x*y + distlag + distlag*speedlag, data = FDtrain.for.M
Model2.test <- ((FDtest$alt^(lambda2)-1)/(lambda2) - predict(Model2.train,FDtest))^2
Model2.rMSEtest <- sqrt(sum(Model2.test) / length(Model2.test))

#rMSE for Model 3
Model3.train <- lm((alt^(lambda3)-1)/(lambda3) ~ x*y + I(altlag^2) + distlag*speedlag + altlag*vspeedlag
                              data = FDtrain.for.Model)
Model3.test <- ((FDtest$alt^(lambda3)-1)/(lambda3) - predict(Model3.train,FDtest))^2
Model3.rMSEtest <- sqrt(sum(Model3.test) / length(Model3.test))


print(paste0("Model 1 rMSE: ",Model1.rMSEtest))
```

```
## [1] "Model 1 rMSE: 59.3409963744415"
```

```
print(paste0("Model 2 rMSE: ",Model2.rMSEtest))
```

```
## [1] "Model 2 rMSE: 580.81360577728"
```

```
print(paste0("Model 3 rMSE: ",Model3.rMSEtest))
```

```
## [1] "Model 3 rMSE: 11.7356729011493"
```

3. Testing for a Different flight

```
load("Flightdata-flight14.Rda")

fpoints$dist <- sqrt((fpoints$xlag)^2 + (fpoints$ylag)^2)

row.nos <- rep(1:5, length.out = nrow(fpoints))
#row.nos <- sample(row.nos)

# Create eval and training and test cases
FDeval <- fpoints[row.nos == 1, ]            #20% of Data for Evaluation

FDtrain <- fpoints[row.nos != 1 & row.nos != 5,]  #60% of Data for Training

FDtest <- fpoints[row.nos == 5, ]            #20% of Data for Testing

##############################################################################################

FDtrain.for.Model <- fpoints[row.nos != 5,]  #80% of Data for Model Training.
                              #This basically a combination on the first 20% and 60%


#rMSE for Model 1
Model1.train <- lm((alt^(lambda1)-1)/(lambda1) ~ x*y, data = FDtrain.for.Model)
Model1.test <- ((FDtest$alt^(lambda1)-1)/(lambda1) - predict(Model1.train,FDtest))^2
Model1.rMSEtest <- sqrt(sum(Model1.test) / length(Model1.test))

#rMSE for Model 2
Model2.train <- lm(log(alt) ~ x*y + distlag + distlag*speedlag, data = FDtrain.for.Model)
```

```
Model2.test <- ((FDtest$alt^(lambda2)-1)/(lambda2) - predict(Model2.train,FDtest))^2
Model2.rMSEtest <- sqrt(sum(Model2.test) / length(Model2.test))

#rMSE for Model 3
Model3.train <- lm((alt^(lambda3)-1)/(lambda3) ~ x*y + I(altlag^2) + distlag*speedlag + altlag*vspeedlag
                           data = FDtrain.for.Model)
Model3.test <- ((FDtest$alt^(lambda3)-1)/(lambda3) - predict(Model3.train,FDtest))^2
Model3.rMSEtest <- sqrt(sum(Model3.test) / length(Model3.test))


print(paste0("Model 1 rMSE: ",Model1.rMSEtest))
```

```
## [1] "Model 1 rMSE: 89.7211773840358"
```

```
print(paste0("Model 2 rMSE: ",Model2.rMSEtest))
```

```
## [1] "Model 2 rMSE: 41667.359131364"
```

```
print(paste0("Model 3 rMSE: ",Model3.rMSEtest))
```

```
## [1] "Model 3 rMSE: 8.0582323347435"
```

Appendix IV

A. Complete

```
# complete pooling
com.pool <- lm(alt ~ 1, data = FlightData)
summary(com.pool)$coef
```

```
##             Estimate Std. Error  t value Pr(>|t|)
## (Intercept) 8734.128   8.212327 1063.539        0
```

B. No Pooling

```
# no pooling
no.pool <- lm(alt ~ factor(icao_addr) - 1, data = FlightData)
np.esti <- summary(no.pool)$coef[,1]
summary(no.pool)$coef[,1]
```

```
## factor(icao_addr)06a190 factor(icao_addr)0d039c factor(icao_addr)3949e3
##               5147.6881               4507.3973              10664.4229
## factor(icao_addr)3c70c3 factor(icao_addr)4004c5 factor(icao_addr)4004c8
##               6647.1911              10668.0000              10668.0000
## factor(icao_addr)400a0d factor(icao_addr)405bfe factor(icao_addr)406a05
##              10668.0000              10668.0000              11277.6000
## factor(icao_addr)4245fb factor(icao_addr)4249ea factor(icao_addr)4b1902
##              10057.9253               6995.1962              10144.9798
## factor(icao_addr)4ba94f factor(icao_addr)4d0115 factor(icao_addr)71bc39
##               9448.8000               7186.4072               6869.5417
## factor(icao_addr)780202 factor(icao_addr)846333 factor(icao_addr)8478ab
##              11277.6000               7012.0706              11277.6000
```

```
## factor(icao_addr)a0046f factor(icao_addr)a00d5b factor(icao_addr)a0105b
##              14325.6067              10360.1744               8336.7875
## factor(icao_addr)a01880 factor(icao_addr)a023a5 factor(icao_addr)a0275c
##              10664.1119              11274.2368              11579.1311
## factor(icao_addr)a05939 factor(icao_addr)a09e70 factor(icao_addr)a0c238
##               8838.6749               5643.6636              10058.4000
## factor(icao_addr)a0d724 factor(icao_addr)a0dadb factor(icao_addr)a0f37e
##               4786.8024               4255.7557               6685.0092
## factor(icao_addr)a1025a factor(icao_addr)a10d7f factor(icao_addr)a11afd
##               7805.1003              10010.5866               9748.4184
## factor(icao_addr)a12622 factor(icao_addr)a144bb factor(icao_addr)a18403
##               6059.8541              11279.3110              10668.8233
## factor(icao_addr)a26473 factor(icao_addr)a2c604 factor(icao_addr)a2c9bb
##               6243.5938              10918.0461               9144.8009
## factor(icao_addr)a2e615 factor(icao_addr)a32890 factor(icao_addr)a3361e
##               6938.4515              10344.3258               9254.0781
## factor(icao_addr)a348a1 factor(icao_addr)a34b0a factor(icao_addr)a35278
##               6188.4534               7008.9032               9436.4774
## factor(icao_addr)a35965 factor(icao_addr)a35eeb factor(icao_addr)a362a2
##              10411.4379              10408.2799               9480.4454
## factor(icao_addr)a36659 factor(icao_addr)a37289 factor(icao_addr)a37efc
##               9753.6000               9501.9581               7267.6464
## factor(icao_addr)a3918f factor(icao_addr)a3a67b factor(icao_addr)a3aa32
##              10949.9574              10032.3249              10058.4000
## factor(icao_addr)a3e69d factor(icao_addr)a3ee0b factor(icao_addr)a43f72
##               3425.7421              10363.2000               6974.2732
## factor(icao_addr)a466f1 factor(icao_addr)a4834b factor(icao_addr)a48379
##               2345.1356               5224.6603              12504.1477
## factor(icao_addr)a48ab9 factor(icao_addr)a494c5 factor(icao_addr)a49510
##               3011.5837              10666.9283              10666.5019
## factor(icao_addr)a4afe5 factor(icao_addr)a4e4cc factor(icao_addr)a4eb80
##               6432.4130               7690.5121               9042.7304
## factor(icao_addr)a4f2ee factor(icao_addr)a50126 factor(icao_addr)a50894
##               5276.1659               6326.4597               7730.4148
## factor(icao_addr)a5307f factor(icao_addr)a53781 factor(icao_addr)a5465d
##               5231.5937               4411.4516               6394.3172
## factor(icao_addr)a553db factor(icao_addr)a55b49 factor(icao_addr)a5666e
##               8047.0742              10972.8000               2150.3033
## factor(icao_addr)a577a3 factor(icao_addr)a57b5a factor(icao_addr)a583f5
##               5773.1608               2974.0302              10057.3002
## factor(icao_addr)a58a36 factor(icao_addr)a62778 factor(icao_addr)a64a02
##               8441.8992               8253.0089              10645.6389
## factor(icao_addr)a65527 factor(icao_addr)a662a5 factor(icao_addr)a678ef
##               9632.5687              11269.5789               9362.9516
## factor(icao_addr)a67ca6 factor(icao_addr)a68d27 factor(icao_addr)a695ef
##               9819.9436               8615.3417              11279.1862
## factor(icao_addr)a7215c factor(icao_addr)a724ce factor(icao_addr)a7276c
##               3272.4856              12186.3138               8383.9978
## factor(icao_addr)a75a10 factor(icao_addr)a76df1 factor(icao_addr)a77b6f
##              11282.4974               9755.1459              10363.2000
## factor(icao_addr)a77f26 factor(icao_addr)a7824e factor(icao_addr)a79422
##               9753.6000               9165.5376               8660.6912
## factor(icao_addr)a7bba1 factor(icao_addr)a7de9f factor(icao_addr)a7debf
##              11882.2813               5295.4707               4612.1455
```

```
## factor(icao_addr)a7e579 factor(icao_addr)a7f762 factor(icao_addr)a810af
##               9388.7348                121.4520              10496.0308
## factor(icao_addr)a83174 factor(icao_addr)a841f5 factor(icao_addr)a84abd
##               8337.0019               8174.0228              11278.1017
## factor(icao_addr)a87850 factor(icao_addr)a894f2 factor(icao_addr)a9065e
##              11586.9188              10669.1372               6578.5521
## factor(icao_addr)a92123 factor(icao_addr)a95c9a factor(icao_addr)a98445
##              13716.0764               1086.3899              11887.6725
## factor(icao_addr)a9acb8 factor(icao_addr)a9bd9a factor(icao_addr)a9c1c5
##              10667.4162              10053.7780               7391.8156
## factor(icao_addr)a9c57c factor(icao_addr)a9c5c5 factor(icao_addr)a9cb8c
##               6081.2082              11277.7521               9532.2101
## factor(icao_addr)a9f2c6 factor(icao_addr)a9fe30 factor(icao_addr)aa10c3
##               9445.2649               6893.9501               5372.1495
## factor(icao_addr)aa147a factor(icao_addr)aa568f factor(icao_addr)aa7a83
##               8487.0454               7252.5413                652.5881
## factor(icao_addr)aa7e74 factor(icao_addr)aa83f3 factor(icao_addr)aa9fe3
##              11276.4858               8448.7484              11887.2168
## factor(icao_addr)aaa751 factor(icao_addr)aac3f4 factor(icao_addr)aaced0
##               8031.4800              11278.1639               5551.9888
## factor(icao_addr)aaf4f1 factor(icao_addr)aaf53a factor(icao_addr)ab03cd
##               6985.2610              11272.5492               6534.9499
## factor(icao_addr)ab0a95 factor(icao_addr)ab3214 factor(icao_addr)ab4ba6
##              10427.2604              11506.4333              10770.2324
## factor(icao_addr)ab6cc9 factor(icao_addr)ab7437 factor(icao_addr)ab9091
##              11582.5936              11277.5149              11277.9938
## factor(icao_addr)abaf6e factor(icao_addr)ac6445 factor(icao_addr)ac85f7
##              10217.8676               4478.0676               8653.0515
## factor(icao_addr)acfc74 factor(icao_addr)ad1927 factor(icao_addr)ad2095
##               7092.9652              10668.2618              10660.3602
## factor(icao_addr)ad27aa factor(icao_addr)ad4b72 factor(icao_addr)ad4f29
##               5990.9088               5806.0442               5241.4785
## factor(icao_addr)ad6ec5 factor(icao_addr)add324 factor(icao_addr)ade5b7
##              11765.5438               9796.0543               6726.6200
## factor(icao_addr)c0061a factor(icao_addr)c01cae factor(icao_addr)c021fc
##              10972.8000              11281.1527              11885.9990
```

```
mean(summary(no.pool)$coef[,1])
```

```
## [1] 8616.394
```

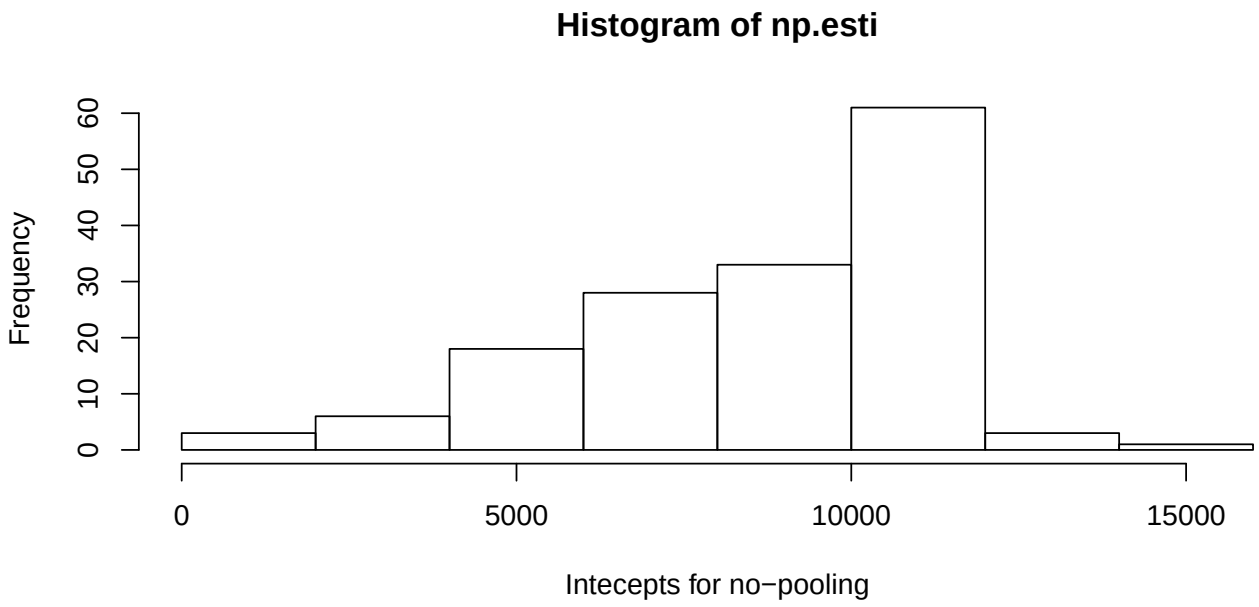
C. Partial Pooling

```
# partial pooling
par.pool <- lmer(alt ~ 1 + (1|icao_addr), data = FlightData)
par.esti <- unlist(coef(par.pool)$icao_addr)
summary(par.pool)$coef
```

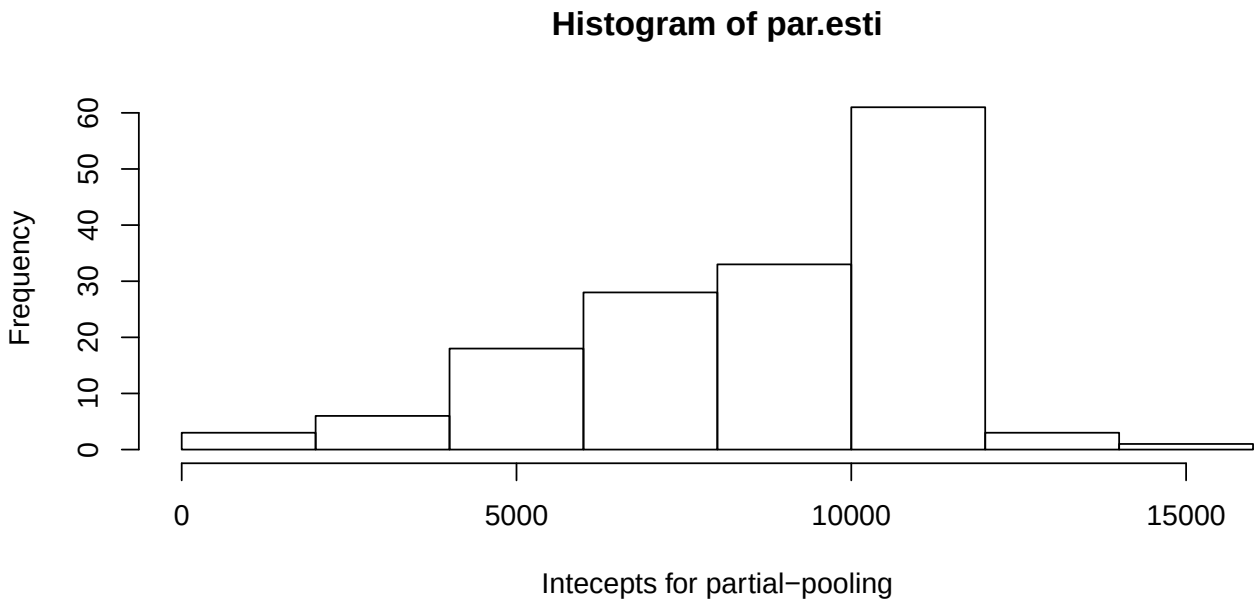
```
##             Estimate Std. Error  t value
## (Intercept) 8616.061   221.4682 38.90427
```

D. Comparison

```
# Distribution of coefficients

hist(np.esti, breaks='FD', xlab = "Intecepts for no-pooling")
```




```
hist(par.esti, breaks='FD', xlab = "Intecepts for partial-pooling")
```



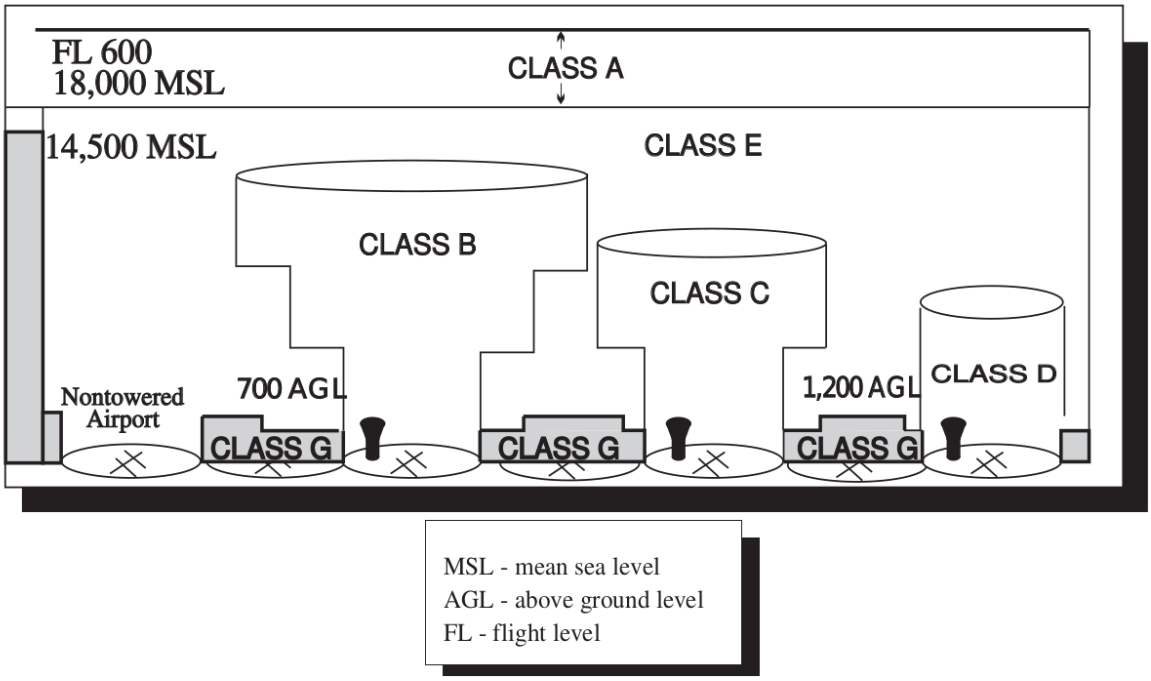


Figure 12: Airspace Division Based on Airport Class

Appendix V

Airspace Division

The ICAO divides airspace into seven classes from A through G (with class F not used in the US). Class A through E are called controlled airspace, which means that within the defined area Air Traffic Control (ATC) services will be provided. Whereas, Class G is uncontrolled airspace, which means that ATC doesn't have any authority and it is out of the domain of the FAA.

In controlled airspace, the level of ATC control on the aircraft depends on the class of the airspace. Controlled airspaces are mainly deployed for areas with a high volume of air traffic, and/or for flights to fly under Instrument Flight Rules(IFR) with ATC guidance and security. Class A is generally 18,000 ft above Mean Sea Level (MSL) or more. In class A only IFR flights are allowed, and flights under Visual Flight Rules (VFR) are not allowed, except for failure of radio communication or emergency. The classes B through D are the airspaces surrounding the nation's busiest airports. The airspace represents an inverted wedding cake as shown in Figure 2.1. An airport is designated a particular class based on the traffic at that airport. Class B is the busiest and Class D is the least busy, and hence the size of the airspace. Generally the elevation of the airspace from the surface for Class B, C and D are 10,000 ft MSL, 4,000 ft MSL and 2,500 ft MSL respectively. However, the area surrounding the airport is tailored based on the airport. Airspaces which is not labeled class A,B,C or D are generally labeled as class E. In United States Class E begins from 14,500 ft MSL to the base of Class A [6].

The uncontrolled airspace (i.e. Class G) extends from the surface till the base of the overlying Class E. ATC doesn't have control in this area, however, there are some VFR minimums which apply to Class G.

The area is between 41.600700 and 42.104991 Latitude, and -88.403735 and -87.408099 Longitude. The reason for considering the area mentioned in Figure 1, is because of its 2 busy airports O'Hare International. (ORD)

and Midway International (MDW) Airports, which are 17 miles and 8 miles away from Downtown Chicago respectively. This close proximity of airports to the downtown provides us with a perfect example where there is high demand for spec- trum and it is quite busy with aircraft. Moreover, O'Hare and Midway Airports are Class B and Class C airports respectively. There are also many Class D airports in this region, like Chicago Executive Airport formerly Palwaukee Municipal Airport (PWK), DuPage Airport (DPA), Schaumburg Airport (06C), Mill Rose Farm RLA Airport (IL68), Lewis University Airport (LOT), Bolingbrook's Clow International Airport (1C5) and Brookeridge Airpark (LL22). Note that the abbreviations associ- ated with each airport name is known as the Location Identifier (LID) for an airport.

Reference

[1] FCC, "Mobile Broadband:The Benefits of Additional Spectrum, Federal Commu- nications Commission,Tech.Rep.,Oct 2010.

[2] T. Beard, G. Ford, R. Saba, and R. S. Jr, "Internet use and job search, Telecomm. Policy, vol. 36, no. 4, pp.260-273, May 2012.

[3] Cisco,"VNI Global Fixed and Mobile Internet Traffic Forecasts," Cisco Vi- sual Networking Index (VNI) Forecast, Tech.Rep., Jun. 2016. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual- networking-index-vni/mobile-white-paper-c11-520862.html

[4] R.Singh, "Spectrum Sharing Opportunity for LTE and Aircraft Radar in the 4.2 - 4.4 GHz Band," Illinois Institute of Technology, ProQuest Dissertations Publishing, 2017. 10606550.

[5] https://en.wikipedia.org/wiki/ECEF

[6] Wikipedia, (2017 Jun.30) Airspace class (United States) [Online]. Available: https://en.wikipedia.org/wiki/Airspace class (United States)

Code Available

https://github.com/roh9singh/19-703-4