ECS132 Term Project

Harrod Tang (hstang@ucdavis.edu), Rayum Shahed (rshahed@ucdavis.edu)
Joshua Sanchez (jossanchez@ucdavis.edu), Kyle Li (kyjli@ucdavis.edu)

Winter 2021

1 Introduction

2 Part A

2.1 Harrod Tang

“The Association Between Home Chaos and Academic Achievement: The Moderating Role of Sleep” is research conducted in 2017 by Maciel M. Hernandez (currently an assistant professor in the Department of Human Ecology at UC Davis), Rebecca Berger, Anjolii Diaz, Carlos Valiente, Nancy Eisenberg, Tracy Spinard, Leah Doane, Marilyn Thompson, Sarah Johns, and Jody Southworth.

The goal of the study was to observe the effects of sleep on the association between family environment (measured by home chaos levels) and academic achievement (AA) for young children. Sleep was split into two criteria, sleep duration and sleep efficiency. Sleep duration was defined as the actual amount of time the child was asleep (ie. excluding periods of waking in the middle of the night), and sleep efficiency was defined as the percentage of time the child was actually asleep. The study was conducted on 50 kindergarteners and 53 first graders.

The study concluded that there exists a strong, negative correlation between home chaos and AA at high levels of sleep duration, with long sleep duration defined as approximately ten hours of sleep ($p < 0.01$).

A significance test does not appear to be the most accurate representation of the data obtained in this study. It is not unreasonable to reject the null hypothesis that there exists no association between long sleep time and the correlation between home chaos and AA (after all, we already know beforehand that the null hypothesis is false). However, rejecting the null hypothesis at a significance level of $\alpha = 0.01$ misleads the public into believing the accuracy and validity of the study’s claim without accounting for any error. Much of the study was subjective. What one parent might report as a level 3 in home chaos, another might find to be a level 4. The sample size of 103 students (who are solely in kindergarten or first grade) is not reflective of the entire population of elementary schoolchildren due to both the small sample size and the lack of variation within grades. A confidence interval would be a better alternative to significance testing as it would better account for this error. The width of the interval would indicate whether or not a larger sample would need to be taken, and the center of the interval would allow one to make a more accurate claim regarding the relationship between long sleep duration and the correlation between home chaos and AA.

2.2 Rayum Shahed

Sex-specific effects of social defeat stress on miRNA expression in the anterior BNST. Behavioural Brain Research, 401, 113084 by Pei X. Luo, Claire E. Manning, Joe N. Fass, Alexia V. Williams, Rebecca Hao,Katharine L. Campi, Brian C. Trainor

This research paper by Professor Trainor and colleagues investigated how post defeat stress affects miRNA (micro RNA) expression in the BNST (anterior bed nucleus of stria terminalis) in different genders of mice. Since women are at a higher risk of succumbing to stress-related disorders, they looked at how the miRNA of different genders alter from stress and why they differ. They concluded that sex-specific changes occurred in miRNA: stressed female mice had few genes that were unregulated, but male mice didn't express that same characteristic.

In the paper, they predicted gene targets of stress-regulated miRNA and what would happen to them (over regulated, under regulated, etc.) and ranked the results using p-values as long as p was under .05, in order to show some sort of significance. Using a confidence interval would lead to a more insightful analysis because instead of simply having a cut off at p = .05, we could see how confident they were that the interval contained the population mean for the sample genes they chose in the sample of rats. In terms of the study, we could view the the sample mean for each sample of genes and how the intervals for population means were different according to those sample means and that could lead to a more holistic analysis. Additionally, since they are trying to find gene changes that would only occur due to post defeat stress, a confidence interval could say if it is significant or unlikely enough to happen, instead of just capping the p = .05 and leaving out potentially important gene expression changes. Relying solely on the p-value can can also leave out crucial information, such as the margin of error for the prediction. While we are trying to interpret if the gene change is significant we would be not considering how far off that prediction would be, which could be a factor of great importance; a confidence interval allows us to view this information. Although this is not a quantitative concern the p-value makes use of the word significant which can be very misleading, since the interpretation does not mean important.

2.3 Joshua Sanchez

Eastwick, P. W., Luchies, L. B., Finkel, E. J, Hunt, L. L. (2014) The predictive validity of ideal partner preferences: A review and meta-analysis. Psychological Bulletin, 140, 623-665.

The purpose of this meta study was to first determine if the two sexes differed when evaluating romantic partners in the second stage of getting into a relationship also know as "surface contact". This is when two people have spoken to each other briefly. This is different from the first stage which identifies hypothetical mates, and the third stage which is a committed relationship. The two variables shown were physical attractiveness and earning prospects. The second purpose was to create a model for individual evaluations based on this two factors with no regards to sex. We will only speak to the first part as that is the only part that introduces p-values.

The study concludes that there were no significant differences in this stage based off data from the six studies that reported the sexes of their participants. For males physical attractiveness correlated to higher romantic value by an r value of 0.53 and for females it was 0.50. The p-value here was 0.5. Thus the differences were not considered significant. For earnings prospects this was r value was 0.27 for males and 0.28 for females.

Based off this p-value of 0.05 then this difference is insignificance. However, this could be improved by instead using equation 12:21 in the text book to assess the differences. This confidence interval is better suited because the p-value is completely arbitrary. A p value of 0.1 or 0.005 would correlate the evolutionary hypothesis being correct. Instead it best to make a judgement based off the fact the differences are just too small to matter.

2.4 Kyle Li

A Ketogenic Diet Extends Longevity and Healthspan in Adult Mice (2017). Roberts, M. N., Wallace, M. A., Tomilov, A. A., Zhou, Z., Marcotte, G. R., Tran, D., Perez, G., Gutierrez-Casado, E., Koike, S., Knotts, T. A., Imai, D. M., Griffey, S. M., Kim, K., Hagopian, K., McMackin, M. Z., Haj, F. G., Baar, K., Cortopassi, G. A., Ramsey, J. J., Lopez-Dominguez, J. A.

This 2017 research paper by Megan Roberts et al. is about the effects of a ketogenic diet in mice. The study was done on mice of twelve months of age and assigned to a ketogenic diet(KD), low-carbohydrate diet(LCD), or control diet. The KD consisted of 89% cal from fat, the LCD consisted of 70% cal from fat, and the control diet consisted of 65% cal from carbohydrates. Low-carbohydrate diets shift one's metabolism away from carbohydrates and toward fatty acid oxidation and the ketogenic diet is the most extreme low-carbohydrate diet. Each diet consisted of the same number of calories. The researchers observed the health of the mice, including lifespan, memory, strength, speed, body composition, and other measures.

There were many results for all the measurements. All of the following results were stated with $p < 0.05$. The KD group had a 13.6% increase in median lifespan compared to the control group. The novel object recognition test indicated that memory was preserved in old mice fed with a ketogenic diet compared to those fed a low-carbohydrate or control diet. Physical tests indicated that male mice fed a KD for 14 months had greater grip strength, were faster, and more active compared to controls.

These results would have been more insightful if they stated the confidence intervals. With a confidence interval, the researchers could have stated a range of days of the average lifespan of the KD, LCD, and control groups of which they were 95% confident. This would also show the error margins and how precise the observations were. The p value omits this info and only states if their confidence intervals do not intersect. The p value does not show how precise or how far the intervals are from each other.

3 Part B

3.1 Let T denote trip duration. Explore finding a model for f_T from one of our density families.

We began by reading in train.csv and assigning the data set to **trainingData**. We then filtered out this data frame to only include the rows where trainingData$MISSING_DATA was 'False' and assigned it to **trainingData2**. Finally, we performed further data cleaning by removing the rows in trainingData2$POLYLINE that were simply a pair of empty brackets ('[]') and called this data frame **trainingData3**.

We calculated trip times by performing arithmetic operations on the number of commas in each row of trainingData3$POLYLINE.

TRIP_TIME=(floor(str_count(trainingData3$POLYLINE, ',')/2) + 1)

We added these trip times as a column to our already existing data frame and called this new data frame **trainingData4**.

We assigned **timeVect** to trainingData4$TRIP_TIMES. To create the upper and lower fences for our data, we first used the IQR function to find the interquartile range; this created a measurement for the spread of the data. We obtained the upper fence through finding the sum of the third quartile and 1.5 times the IQR. The lower fence was obtained through subtracting 1.5 times the IQR from the first quartile. We removed any outliers by only keeping trip times that were within the range of the upper and lower fences. After removing these outliers, there still remained trips that were 15 seconds long. We found this to be unrealistic and decided to remove these trips as well. Figure 1 is a histogram of the remaining trip times.

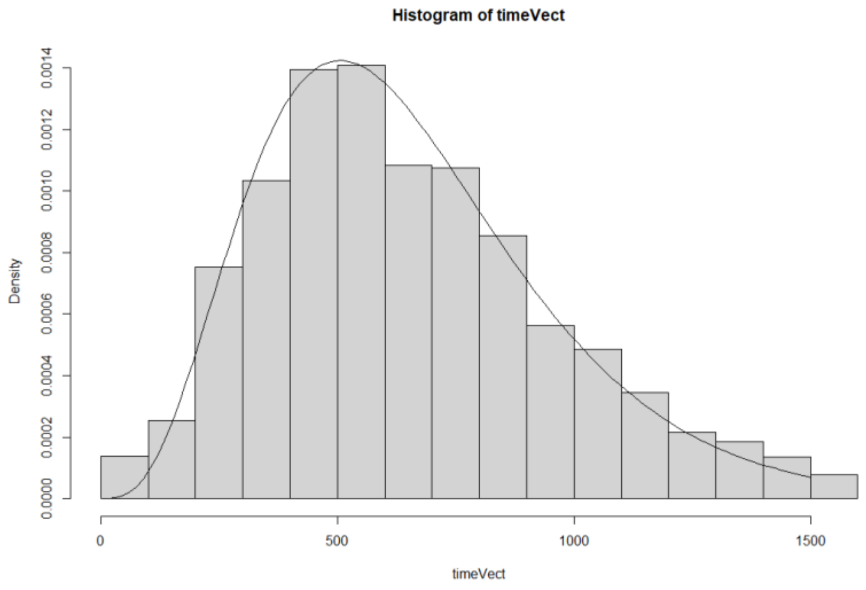


Figure 1: Density of Trip Times

As a group, we determined that the histogram looked gamma distributed. We used the method of moments to find estimates of the variables for a gamma distribution, λ and r. We defined the first moment, **m1**, as the mean of trip times, and the variance, **newS2**, as the mean of the squared difference between trip times and m1. We used these two variables to find our estimates for λ and r, **lambest** and **rest**. We plotted the density of a gamma distribution with **lambest** and **rest** as the parameters and concluded that the gamma distribution was a good fit for the density of trip times. The fit of the gamma density curve on our data can be observed in Figure 1.

3.2 Let B denote the proportion of time a driver is busy, i.e. actually driving rather than waiting for the next fare. Explore finding a model for f_B from one of our density families.

As a group, the first problem we ran into was how "the time a driver is busy" should be defined. Our intuition led us to initially believe that we should find the proportion a driver is driving out of the time of his or her shift for each day worked and to average these proportions. However, we ultimately decided to define the time a driver is busy as the total time spent driving divided by the total time of the driver's shifts for the year. This way, we would better account for the weights of the time spent driving. We also noticed that there were certain trips with missing data that could potentially mislead our analysis. However, we found that out of all 1.7 million trips, only 10 were missing data, leading us to believe that these trips would have little to no effect on our analysis.

The "unique" function on the TAXI_ID column, returned a vector of individual taxi ids, each of which corresponds to a unique driver; we called this vector **taxiDrivers**. Our function, **driverTimes**, takes in a taxi id as a parameter and returns a vector of trip times for that specific driver. Our function, **getShift**, makes calls to **driverTimes** and returns the proportion of the total time a driver is driving out of the total of his or her shifts. Our final function, **dataAvgBusy**, makes calls to **getShift** and returns a vector of proportions, each of which represents the time each driver is busy.

driverTimesBusy is the vector of proportions of times each driver is busy. To clean the data, we ensured that all proportions were less than 1. Initially, there were two values in **driverTimesBusy** that had the value 2, and there were two more that had the value infinity. It is impossible to spend more time driving than the shift time, so we removed these values. Using the IQR and quantile functions, we removed any remaining outliers using the same method as **above** when cleaning the data for trip times.

The histogram representing the density of the remaining proportions appears normally distributed. We found estimates for μ and σ and plotted the density of a normal distribution with these parameters. The histogram, along with the fit of the curve can be seen in Figure 2. After assessing the fit of the curve, we determined that a normal distribution is a good distribution to represent the density of the proportion of times a driver is busy.

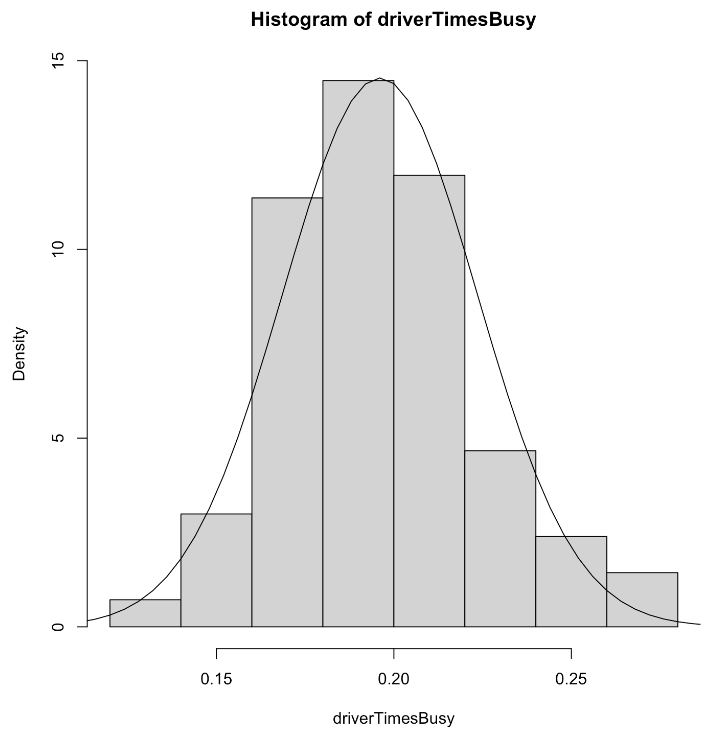


Figure 2: Density of Proportion of Time A Driver is Busy

3.3 Investigate whether the type of call used to summon the taxi makes much difference in mean trip time.

As a group, we determined that confidence intervals would be the best way to gauge whether or not the call type to summon a taxi had a significant impact on mean trip time. We wrote a function, **callTypeTimes**, that returns a vector of trip times for a given call type. With a call to the function for each of the three call types (‘A’, ‘B’, ‘C’), we were able to create the three vectors, **tripTimesA**, **tripTimesB**, and **tripTimesC**. We decided to pass in the cleaned data (removed trips with missing data or empty pair of brackets for trip coordinates) in order to get a more accurate representation of mean trip time for each call type.

Mean of **tripTimesA**: 766.2142
Mean of **tripTimesB**: 681.6612
Mean of **tripTimesC**: 792.7525

We then wrote a function, **findMargError** that takes in two vectors of trip times and returns the radius of the 95 percent confidence interval (margin of error) for the difference of means of the the two vectors. We made calls to findMargError for each pair of call types (A and B, B and C, A and C) and named these calls to the function **margAB**, **margCB**, and **margCA**, respectively. We used the method seen in equation 12.21 in the textbook to construct various confidence intervals for the difference of means for each pair of call types. Below are the margin of errors for a 95% confidence interval.

margAB: 1.958132
margCB: 3.140987
margCA: 2.867397

Below are the respective 95% confidence intervals:

Interval AB: (82.59494, 86.51113)
Interval CB: (108.2239, 113.9586)
Interval CA: (23.39731, 29.67917)

We also generated the 99% confidence intervals:

Interval AB: (81.97966, 87.12641)
Interval CB: (108.2239, 113.9586)
Interval CA: (23.39731, 29.67917)

The AB Interval tells us we are 95% confident that the mean trip time for call type A is about 82-86 seconds greater than the mean trip time for call type B. The Interval BC tells us we are 95% confident that the mean trip time for call type C is typically 108-113 seconds longer than that of trip time B. Interval AC shows we are 95% confident that the mean trip time for call type C is 23-29 seconds greater that the mean trip time for call type A. The small widths of the confidence intervals show the accuracy and validity of the data. Additionally, the margin of error we obtained for each interval was only 1-3 seconds showing extremely limited discrepancies in our confidence intervals. Because the bounds of the confidence intervals are notably far from 0, it is fair to assume that call type has an impact on mean trip time. Using the 99% confidence interval, we were able to verify our claims for the 95% confidence interval.

3.4 Develop models for predicting trip time from other variables, both the explicit ones and also trip distance.

3.4.1 Predicting trip time from trip distance

We found each trip distance by using the function “distm” from the geosphere package to find the distance between the first and last pair of coordinates in each row of POLYLINE. We appended a column of trip distances into **trainingData4**. We extracted the columns for trip times and trip distances and called this new data set **trainingSubset2**. We called qeLin on trainingSubset2, with "TRIP_TIME" as our yName parameter and named the model fit **linOutDist**. Figure 3 is a plot of trip times in relation to trip distances. The linear regression line for predicted trip times has also been included.

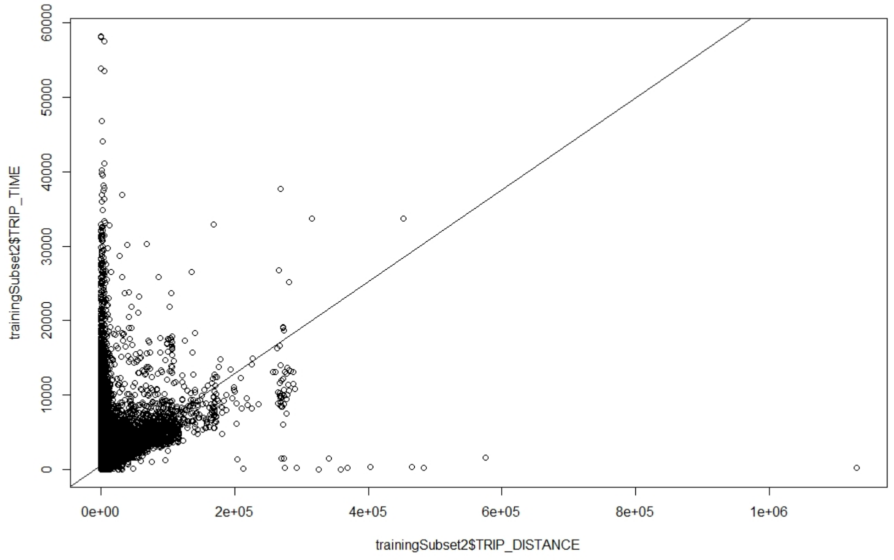


Figure 3: Trip Times vs Trip Distance

linOutDist2 is the same call to qeLin as before, but this time, the holdout parameter is set to NULL. We wanted to observe whether or not the holdout set would affect the predicted trip times. We called "predict" without test distance, **testDist**, and found that linOutDist and linOutDist2 predicted very similar trip times.

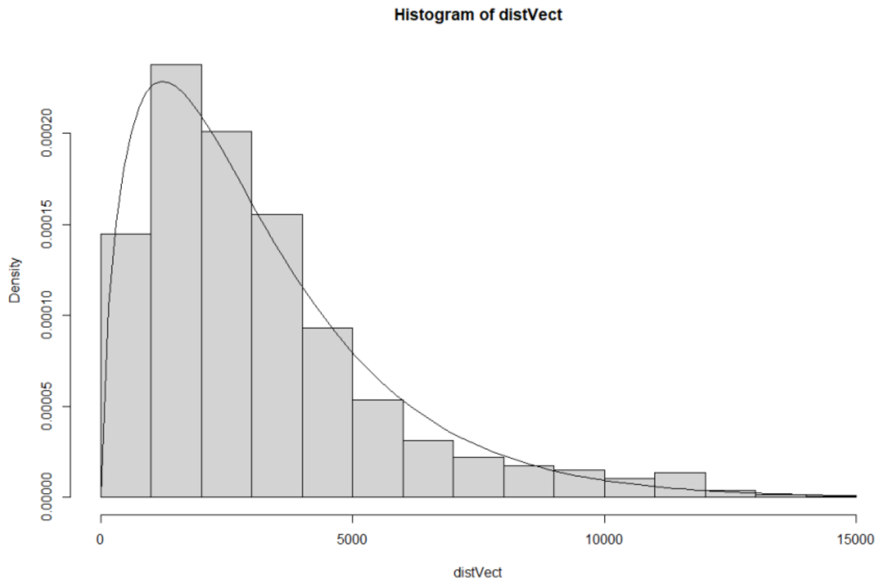
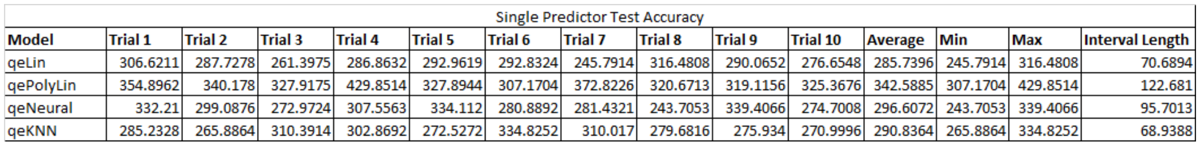


Figure 4: Histogram of Trip Distances

We sought to find a 95% confidence interval for predicted trip times given a set of predictor trip distances. We noted that most of the trip distances are less than 15,000 meters. We created a histogram for the density of trip distances less than 15,000 and were able to find that the trip distances were gamma distributed, as can be seen in Figure 4.

Because we found trip distances to be gamma distributed, we generated 10,000 random values using the rgamma function with our estimates of r and λ from the calculated trip distances vector and called this vector **predictDistances**. We generated a linear model, **linOut**, from **predictDistances** by using qeLin. The predict function was then used to generate a vector of predicted trip times, **predictedTripTimes**. Due to symmetry, we knew that the lower bound for a 95% confidence interval would be at the .025 quantile and the upper bound would be at the .975 quantile. Using qgamma, along with our estimated values for r and λ, we were able to generate the bounds for the 95% interval. We found the lowerbound to be 488 and the upperbound to be 937. To verify our findings, we made a call to **mean(lowerBound $\leq$ predictedTripTimes & predictedTripTimes $\leq$ upperBound)** to find the proportion of predicted trip times that were within our generated confidence interval. The call to the mean function returned a proportion of approximately 0.95.

To compare various prediction models, we also generated predictions using qePolyLin, qeKNN, and qeNeural. The mean absolute prediction errors ($testAcc) of the various prediction models revealed that the neural network model had the smallest prediction error, followed by the linear model and KNN model. The polynomial model had the largest prediction error out of the four methods. This can be seen in Figure 5, where the average prediction error over 10 samples of 100,000 for each method were used to identify the relationship between them.



Single Predictor Test Accuracy													
Model	**Trial 1**	**Trial 2**	**Trial 3**	**Trial 4**	**Trial 5**	**Trial 6**	**Trial 7**	**Trial 8**	**Trial 9**	**Trial 10**	**Average**	**Min**	**Max**	**Interval Length**
qeLin	306.6211	287.7278	261.3975	286.8632	292.9619	292.8324	245.7914	316.4808	290.0652	276.6548	285.7396	245.7914	316.4808	70.6894
qePolyLin	354.8962	340.178	327.9175	429.8514	327.8944	307.1704	372.8226	320.6713	319.1156	325.3676	342.5885	307.1704	429.8514	122.681
qeNeural	332.21	299.0876	272.9724	307.5563	334.112	280.8892	281.4321	243.7053	339.4066	274.7008	296.6072	243.7053	339.4066	95.7013
qeKNN	285.2328	265.8864	310.3914	302.8692	272.5272	334.8252	310.017	279.6816	275.934	270.9996	290.8364	265.8864	334.8252	68.9388

Figure 5: Table of Single Predictor Test Accuracy

Figure 6 is a visual representation of the polynomial regression model and Figure 7 represents the neural network for predicting trip times from trip distances. Figure 8 shows the predictions generated by the KNN model.

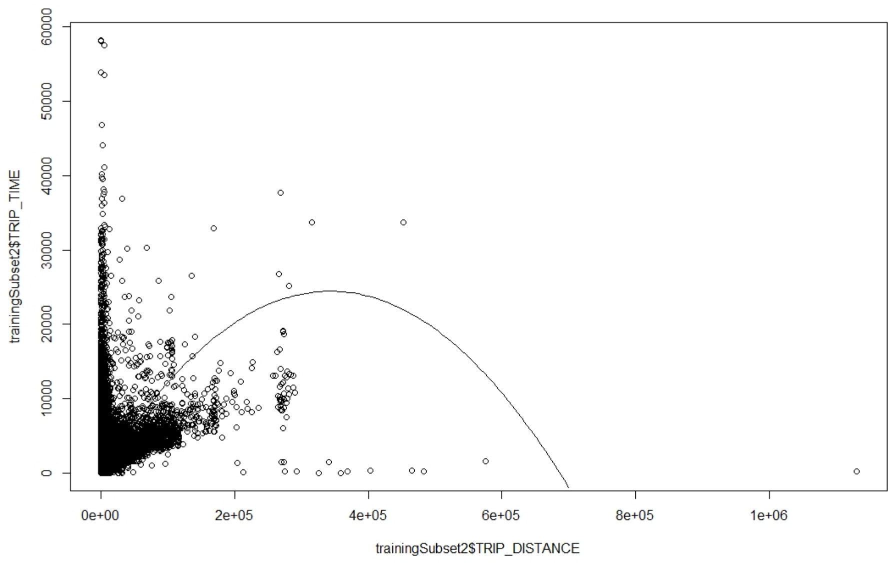


Figure 6: PolyLin with Predictor Trip Distance

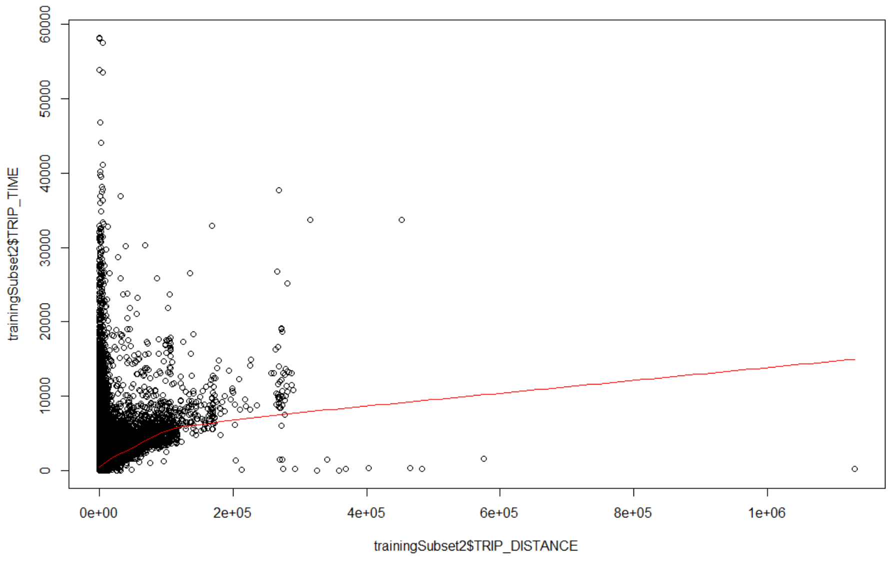


Figure 7: qeNeural Single Predictor Trip Distance

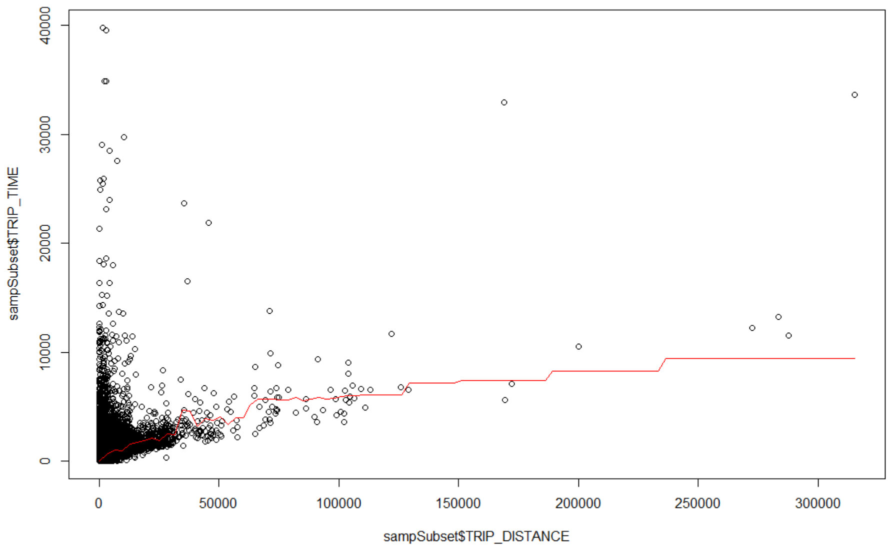


Figure 8: qeKNN Single Predictor Trip Distance

3.4.2 Predicting trip time from origin call

Following a similar process from predicting trip time from trip distance, we instead used origin call. For calculating trip times we used the **method above** to clean the data in POLYLINE. Since origin call contained NA values for call types that were not call type A (trip was dispatched by central), we had to exclude the indices that contained NA from both origin call and trip time. This new data frame called **x** was the training subset we used to predict trip time from origin call. Origin call seemed to have a uniform distribution, other than the first bin, as seen in Figure 9.

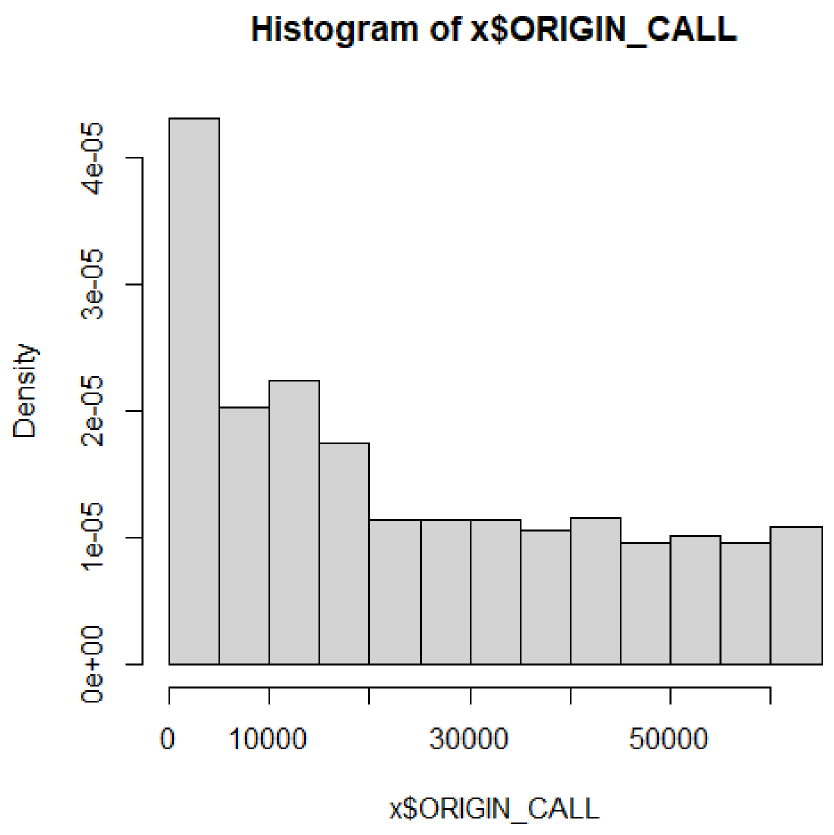


Figure 9: Histogram of Origin Call

We decided not to factor out outliers on origin call, since in terms of context there would be no outliers because each origin call is an identifier for a phone number; thus there is no such thing as an identifier that is ”too large” or ”too small". We are simply trying to predict the trip time given a pseudo phone number. Initially, this resembled a poor idea, since intuitively there seems to be no correlation between a random pseudo phone number and a time of a trip, but Figure 10 suggests a linear plot.

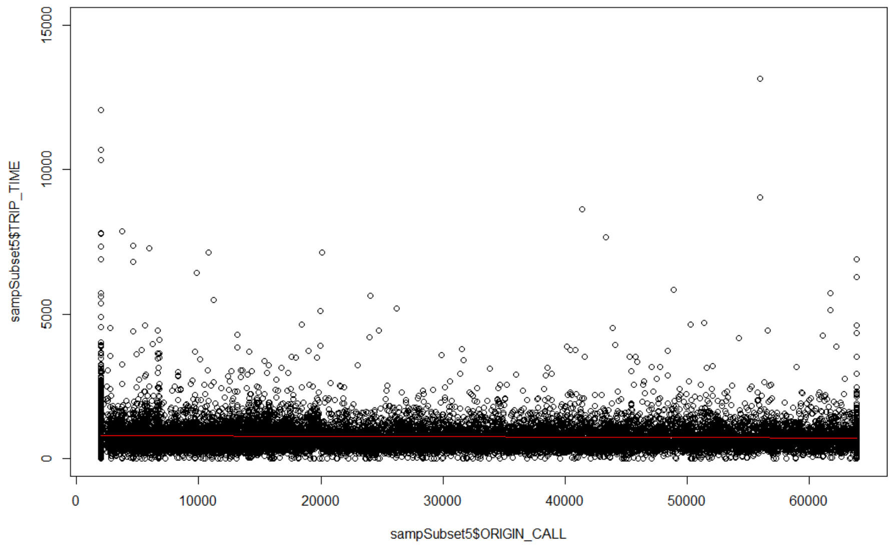


Figure 10: qeLin with Predictor Origin Call

We used a method similar to predicting trip time from distance as **above**, but with an uniform distribution instead. Due to the uniform distribution of origin call that we saw in figure 9, we decided to generate 10,000 random values using runif into a vector called **predictcall** and obtained the min and max from the training subset **x**. Using the training data and the qeLin function we created a linear model called **linOut** to create a vector of predicted trip times called **predictedTripTimes**. Since the **predictcall** is uniformally distributed this would make the **predictedTripTimes** vector uniformally distributed, thus we used qunif to find the middle 95% of the data using the min and max from **predictedTripTimes**. This gave a much smaller confidence interval than above: $716 < \mu < 793$. Again we verified the confidence interval using **mean(lowerBound ≤ predictedTripTimes & predictedTripTimes ≤ upperBound)** and found the proportion to be approximately .95.

3.4.3 Predicting trip time from timestamp

Modeling the processes we did earlier, we used timestamp as a predictor for trip time. Essentially, we sought to find if there is a relationship of when a trip is taken and the duration of the trip. We started by parsing POLYLINE in the same manner to calculate trip times, creating a training subset which included timestamp and trip time, and by creating a histogram of the timestamp to view the distribution.

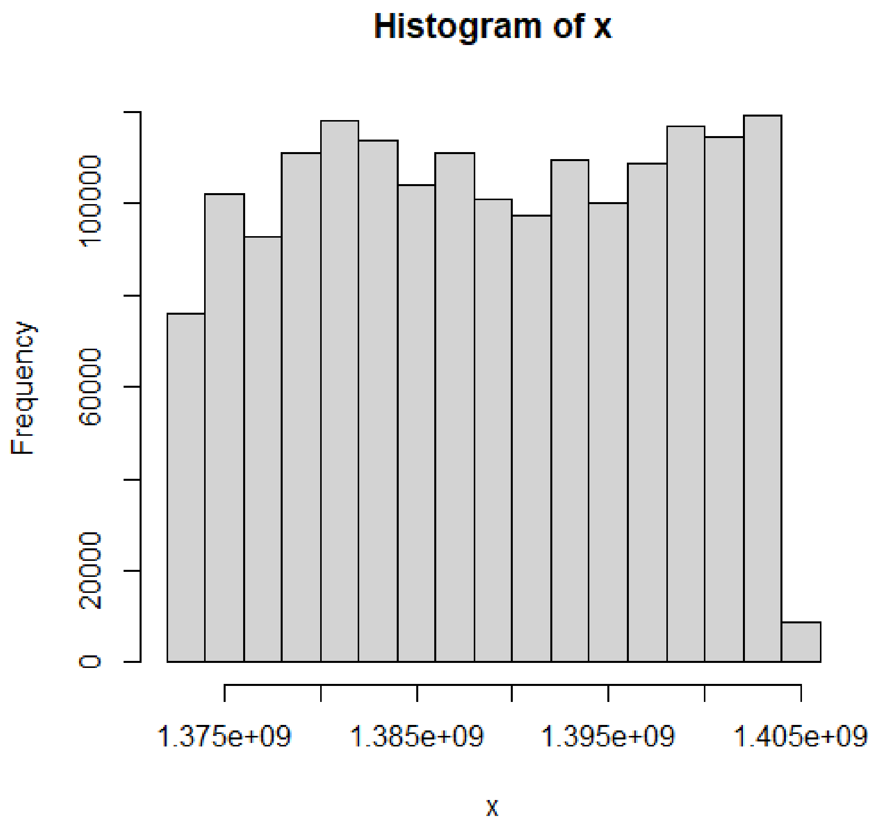


Figure 11: Timestamp Distribution

As seen in Figure 11, the distribution of timestamp appears uniform once again, so we repeat a similar process to origin call from **above**. Again, we decided in the context of timestamp that there is no such thing as an outlier: a time that is too large or too small doesn't make sense. Additionally, we plotted timestamp vs trip time and saw a similar plot to origin call vs trip time(Figure 12), so we inferred a linear plot.

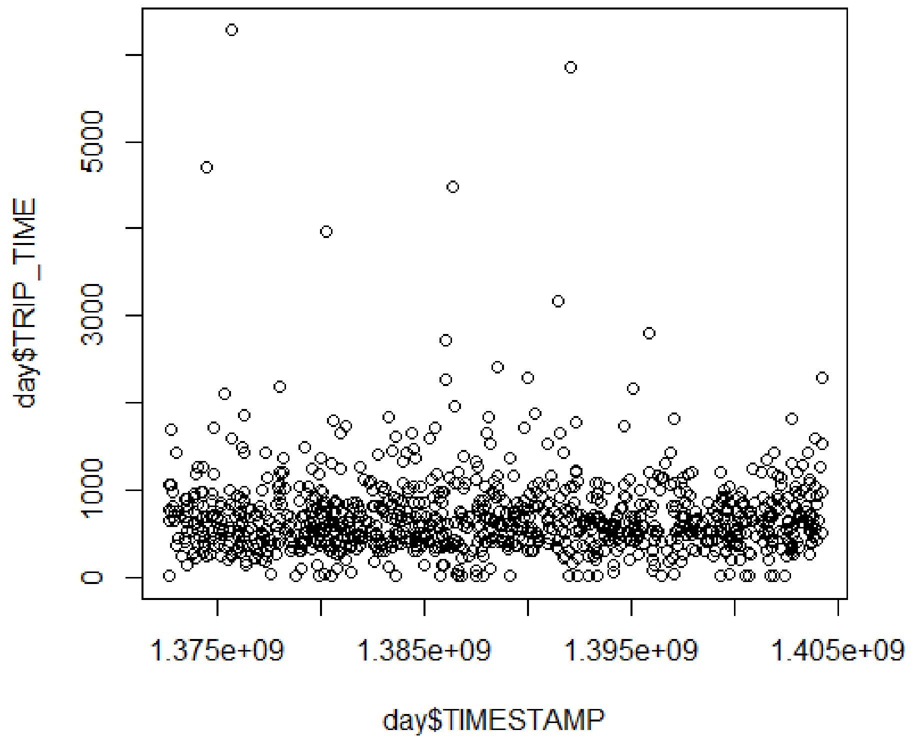


Figure 12: Timestamp Vs. Trip Time

We then followed the same process as **above** in origin call to generate 10,000 random values with runif using the max and min of the training subset dataframe to create **predicttimes** vector, and then passed the training subset into qeLin to create **linOut** which we used in conjunction with **predicttimes** to get a vector of predicted trip times called **predictedTripTimes**. Since, timestamp was uniformally distributed, the **predictedTripTimes** would be as well, thus we used qunif to find the central 95% using the min and max from **predictedTripTimes**. This confidence interval is even smaller, but is included in both of the confidence intervals we have calculated earlier: $727 < \mu < 740$. We ran **mean(lowerBound** $\leq$ **predictedTripTimes & predictedTripTimes** $\leq$ **upperBound)** to verify the interval and found the proportion to be approximately .95, as well.

3.4.4 Predicting Trip Time from time of day

Since the timestamp gave us no insight into whether taxi rides are longer the further into the year the data goes, we decided to see if the time of day matters. We followed the same preliminary methods seen in section **3.4.2**. The calculation for the time of day was simply TIMESTAMP %% 3600*24. The histogram for time of day does not have a clear distribution. As seen in Figure 13.

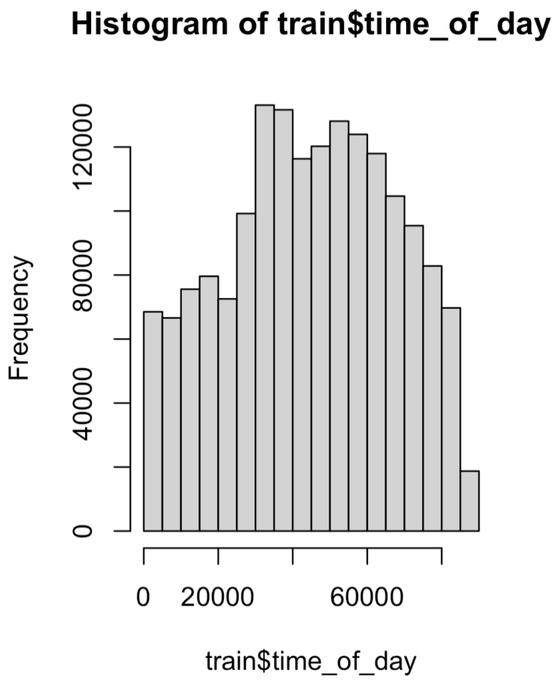


Figure 13: Time of Day Histogram

The histogram was not uniform so we used polylin to determine if it would be worth using as prediction parameter. Our results showed a small correlation between the two in small hump around midday as seen in Figure 14. So we decided to use qeKNN to see if it could predict the results as seen in figure 15. It seemed to have failed, as we could not get an even line with a strong correlation. Instead we got a jagged line with no insight. We concluded that is due to the lack of a significant correlation between the two data sets.

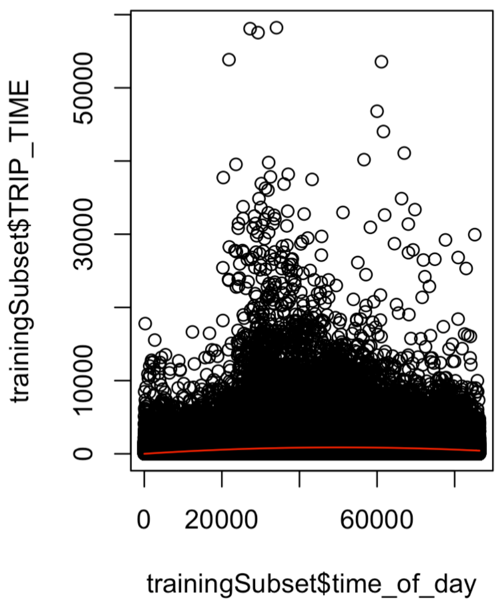


Figure 14: Trip Time Vs. Time Of Day PolyLin

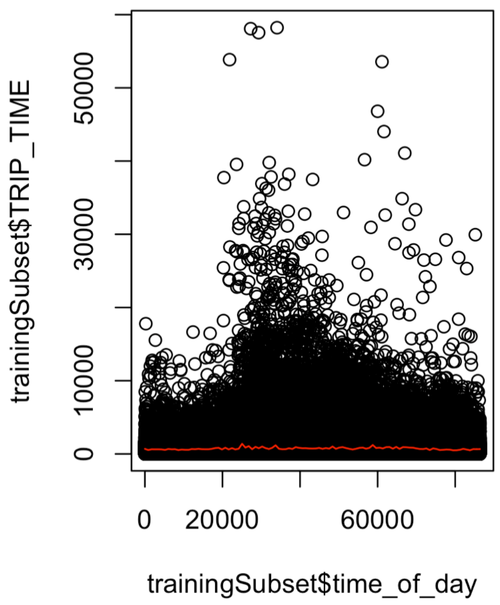


Figure 15: Trip Time Vs. Time of Day KNN prediction

3.4.5 Predicting trip time from multiple predictors

Since we created trip time predictions with multiple predictors separately, we decided to combine the predictors to see the interval and time predicitions we obtain. We started by creating a training subset consisting of origin call, timestamp, trip distance, and trip time. Since, as before origin call has NA values, so we discarded the indices where NA values were present in origin call. We could not graph this training subset vs trip time or do a histogram of it, since it contained multiple predictors. Thus, we generated random values into a vector using the above procedures; rgamma for trip distance and runif for timestamp and origin call. We then created a linear model, **linOut**, using the training subset and qeLin, and passed **linOut** and the vectors of randomly generated values into the predict function resulting in the **predictedTripTimes** vector. We then plotted a histogram of **predictedTripTimes** and saw that it had a gamma distribution similar to that of the distribution trip time of population. This can be seen in Figure 16.

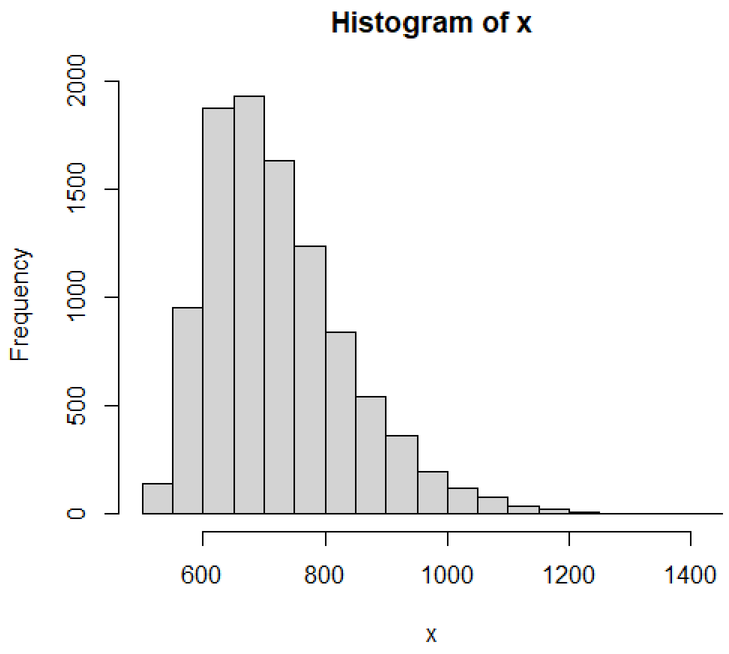


Figure 16: Histogram of trip times from multiple predictors

We then created a 95% confidence interval bounds using qgamma and estimated r and λ using the **predictedTripTimes**. This gave us the central 95% and we verified it using **mean(lowerBound $\leq$ predictedTripTimes & pre-**

dictedTripTimes $\leq$ **upperBound)** to find the proportion of predicted trip times that were within our generated confidence interval and obtained approximately .95. The confidence interval for this prediction was larger than origin call and timestamp, but similar to trip distance, yet it still included origin call and timestamp: $513 < \mu < 965$. The distribution of the **predictedTripTimes** shows that the predictions are accurate, since the distribution of the **predictedTripTimes** models the distribution of population trip time. Additionally, the confidence interval encompass origin call and timestamp, while modeling trip distance, which has a clear correlation with trip time. Figures 17 to 25 show the linear, neural, and random forest models for predicting trip times from trip distance and each respective call type.

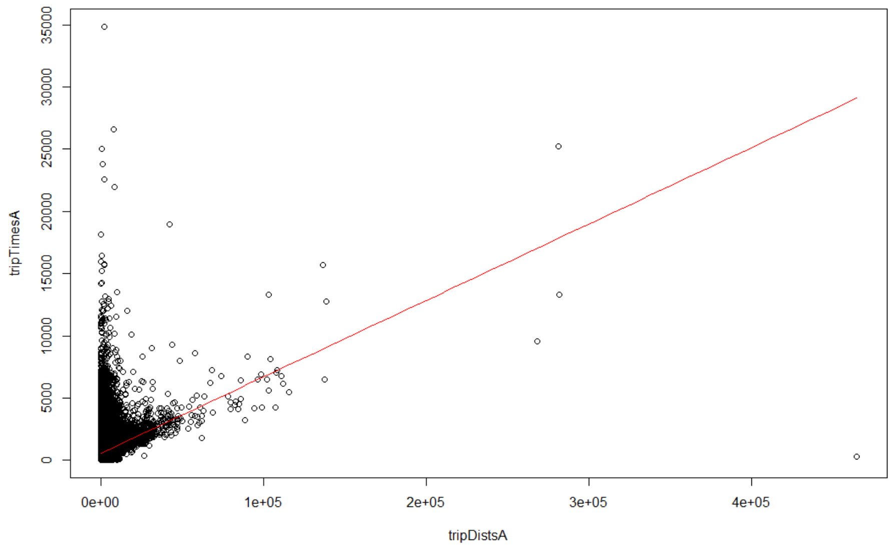


Figure 17: Linear Prediction for Trip Times From Trip Distance and Call Type A

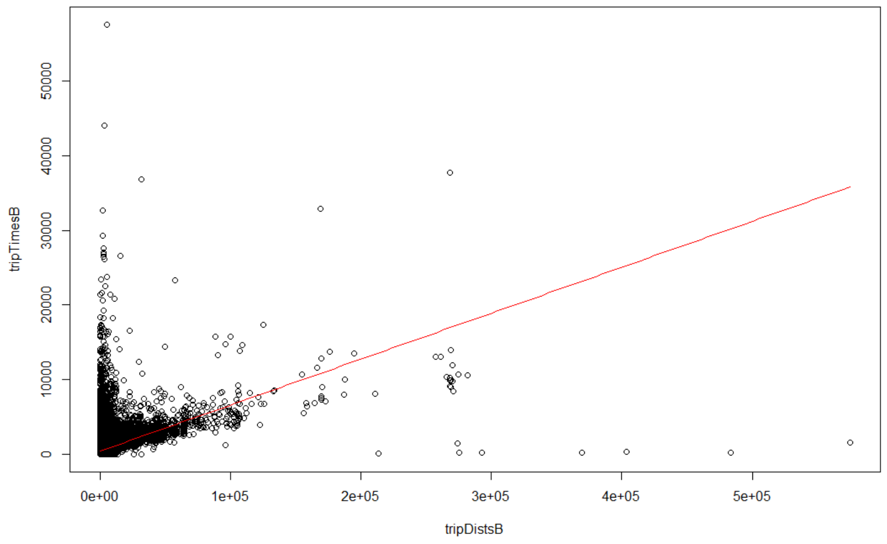


Figure 18: Linear Prediction for Trip Times From Trip Distance and Call Type B

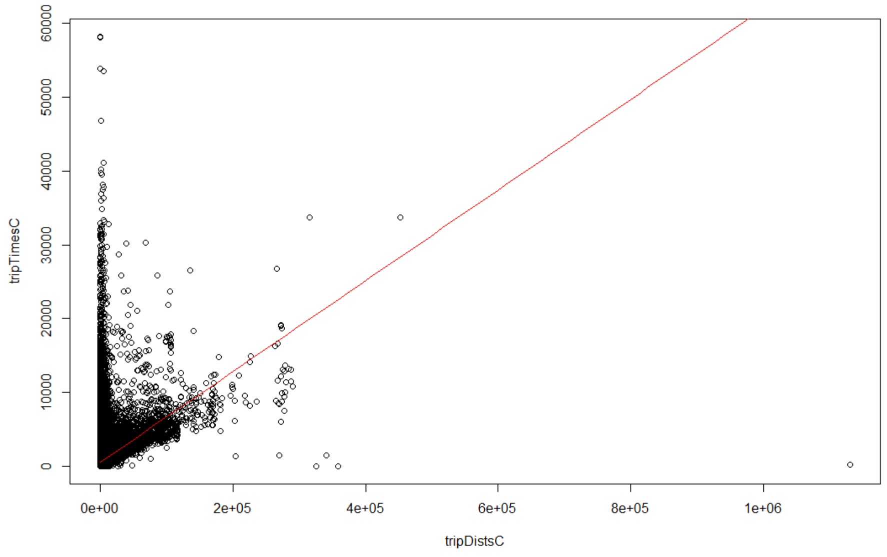


Figure 19: Linear Prediction for Trip Times From Trip Distance and Call Type C

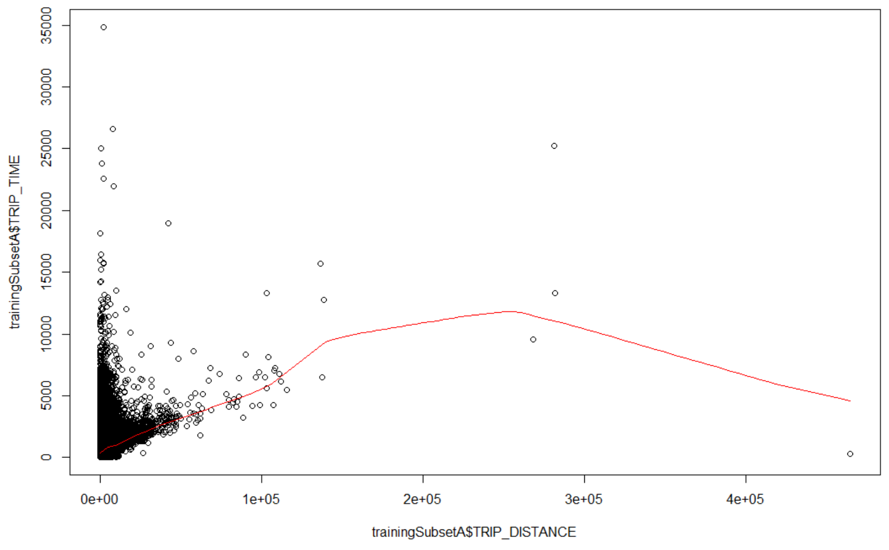


Figure 20: Neural Prediction for Trip Times From Trip Distance and Call Type A

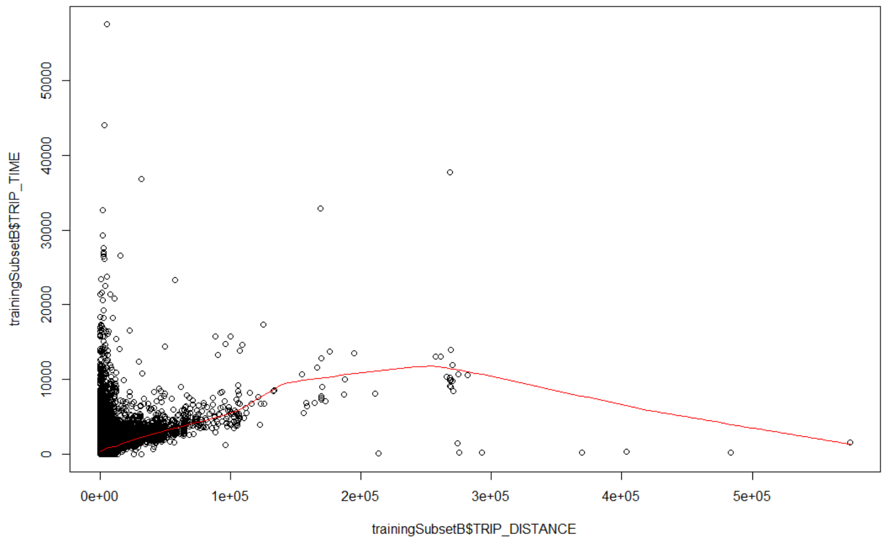


Figure 21: Neural Prediction for Trip Times From Trip Distance and Call Type B

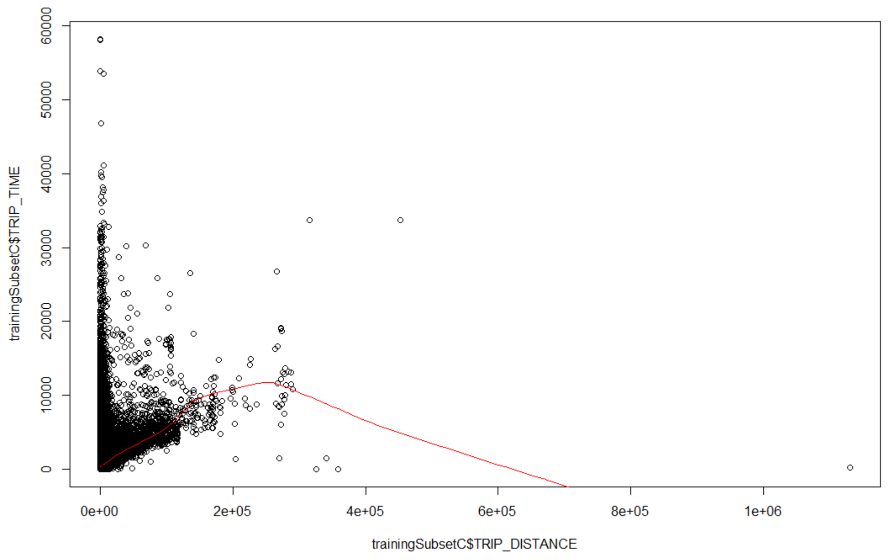


Figure 22: Neural Prediction for Trip Times From Trip Distance and Call Type C

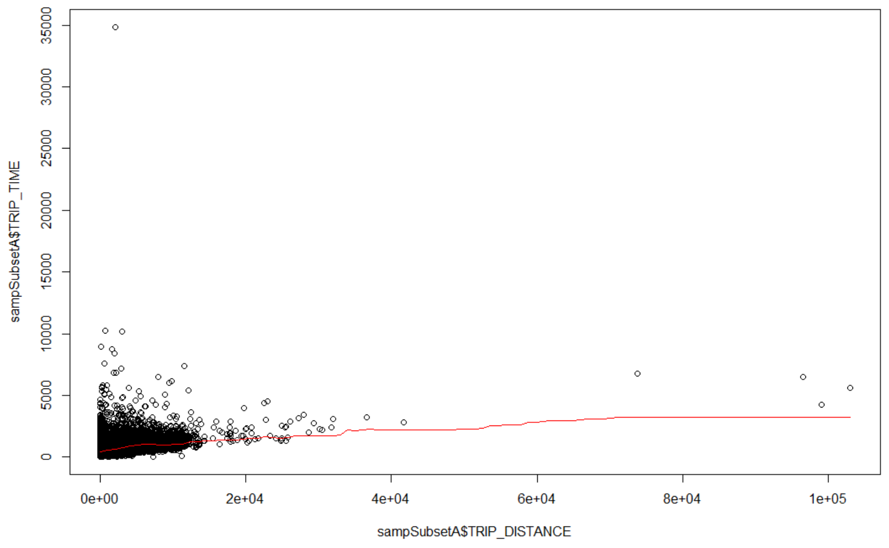


Figure 23: Random Forest Prediction for Trip Times From Trip Distance and Call Type A

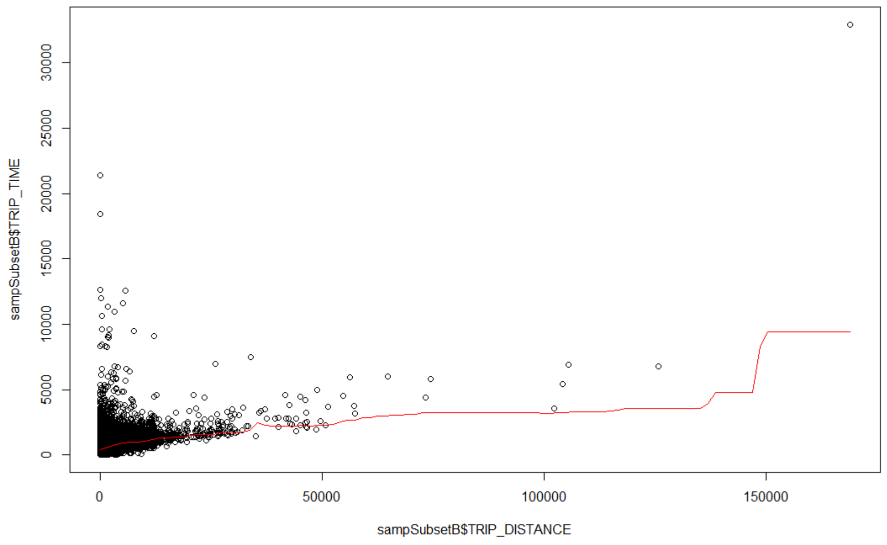


Figure 24: Random Forest Prediction for Trip Times From Trip Distance and Call Type B

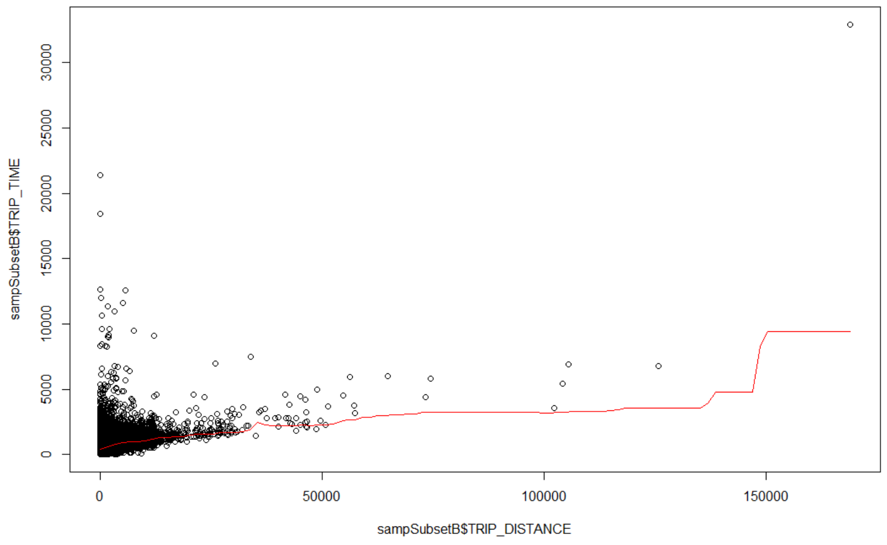


Figure 25: Random Forest Prediction for Trip Times From Trip Distance and Call Type C

3.4.6 Why we didn't choose certain predictors

We decided to not use origin stand or taxi id for a predictor for trip time for two main reasons.

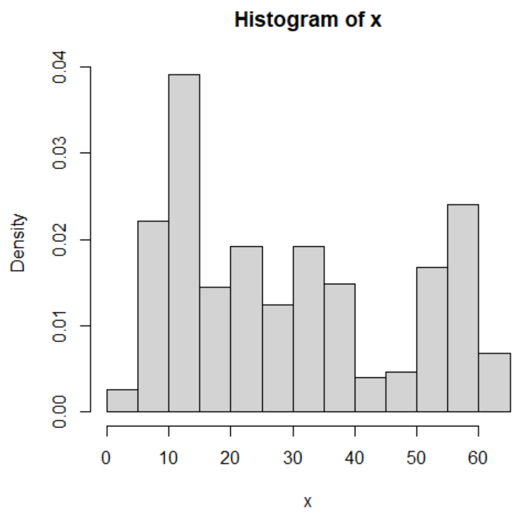


Figure 26: Origin Stand Distribution

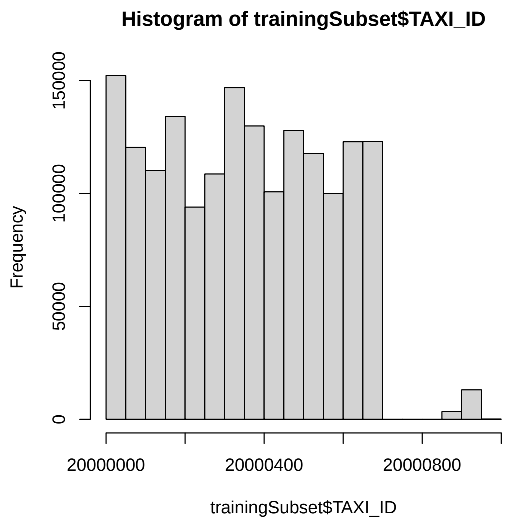


Figure 27: Taxi ID Distribution

Both of the histograms do not have a clear distribution of any sort. Taxi id could be argued as uniform, but we cannot disregard the data on the left tail, since in context there is no such thing as taxi id that is too large or too small. Additionally, even the values on the right tail seem to vary too much to be labeled as uniform distribution. Again, origin stand follows the same logic; the histogram doesn't follow any distribution and seems to vary much more than taxi id. Because they did not have any sort of clear distribution, we could not accurately generate values modeling the distributions of the predictors, which would lead to non accurate predictions of the trip times, making it a bad and unnecessary predictor. Additionally, thinking from an intuitive standpoint there wouldn't seem to be any correlation between the stand a taxi is taken at and trip time or which taxi is taken and trip time. The trip time is not constrained in any sense by those two predictors and trip time would be random for any stand or taxi, making them faulty predictors.

A Part A Research Links

A.1 Harrod Tang

Link: The Association Between Home Chaos and Academic Achievement: The Moderating Role of Sleep

A.2 Rayum Shahed

Link: Sex-specific effects of social defeat stress on miRNA expression in the anterior BNST.

A.3 Joshua Sanchez

Link: Eastwick, P. W., Luchies, L. B., Finkel, E. J, Hunt, L. L. (2014) The predictive validity of ideal partner preferences: A review and meta-analysis. Psychological Bulletin, 140, 623-665.

A.4 Kyle Li

Link:: Roberts, M. N., Wallace, M. A., Tomilov, A. A., Zhou, Z., Marcotte, G. R., Tran, D., Perez, G., Gutierrez-Casado, E., Koike, S., Knotts, T. A., Imai, D. M., Griffey, S. M., Kim, K., Hagopian, K., McMackin, M. Z., Haj, F. G., Baar, K., Cortopassi, G. A., Ramsey, J. J., Lopez-Dominguez, J. A. (2017). A Ketogenic Diet Extends Longevity and Healthspan in Adult Mice. Cell metabolism, 26(3), 539–546.e5.

B TermProject.R Code

```
library(regtools)
library(dplyr)
library(stringr)
library(geosphere)

# data cleaning and setting up additional data sets
trainingData <- read.csv("train.csv")
trainingData2<- trainingData[! trainingData$MISSING_DATA %in% "
    ↪ True", ]
trainingData3<- trainingData2[! trainingData2$POLYLINE %in% "[]",
    ↪  ]

len <- length(trainingData3$POLYLINE)
dist <- vector()

for(i in 1:len) {
  temp1 <- gsub("\\[|\\]", "", trainingData3$POLYLINE[i])
  temp <- str_split(temp1, ",")
  tempLen <- length(temp[[1]])
  # finding distance between first and last pair of coordinates
  dist[i] <- distm(c(as.double(temp[[1]][1]), as.double(temp
      ↪ [[1]][2])), c(as.double(temp[[1]][tempLen-1]), as.double(
      ↪ temp[[1]][tempLen])), fun = distHaversine)
}

# adding a column of trip times and trip distances to data set
trainingData4 <- cbind(trainingData3[1:9], TRIP_TIME=(floor(str_
    ↪ count(trainingData3$POLYLINE, ',')/2) + 1) * 15, TRIP_
    ↪ DISTANCE=dist)


timeVect <- trainingData4$TRIP_TIME

# Part B Bullet 1
# removing outliers
quantileTime <- quantile(timeVect)
timeIQR <- IQR(timeVect)
timeVect <- timeVect[timeVect < quantileTime[4] + timeIQR * 1.5]
timeVect <- timeVect[timeVect > quantileTime[2] - timeIQR * 1.5]
timeVect <- timeVect[timeVect > 15]
hist(timeVect)
quantileTime[2] - timeIQR * 1.5
```

```
# generating parameters for gamma distribution
m1 <- mean(timeVect)
newS2 <- mean((timeVect - m1)^2)
rest <- m1^2 / newS2
lambest <- m1 / newS2
hist(timeVect, freq = FALSE)
curve(dgamma(x, rest, lambest), add = TRUE)

# Part B Bullet 2
taxiDrivers <- unique(trainingData$TAXI_ID) # vector of the taxi
    ↪ drivers

# returns vector of trip times for a specific driverID
driverTimes <- function(driverID, dataframe) {
  polylineCol <- dataframe$POLYLINE
  vectPoly <- polylineCol[dataframe$TAXI_ID == driverID]
  retVect <- vector()
  i <- 0
  for(trip in vectPoly){
    i <- i + 1
    retVect[i] <- ((floor(str_count(vectPoly[i], ',')/2) + 1) *
        ↪ 15)
  }
  retVect
}

# returns proportion driverID is busy
getShift <- function(driverID, dataframe) {
  times <- dataframe$timestamp
  totalTime <- 0
  shiftVect <- vector()
  busyVect <- vector()
  i <- 0
  j <- 0
  for(driver in driverID) {
    vectTimes <- times[dataframe$TAXI_ID == driver] # extract the
        ↪ trip start times that correspond to a specific driver
    tripDurations <- driverTimes(driver, dataframe) # get the
        ↪ durations of the driver's trips
    differenceVect <- diff(vectTimes) # calculate the difference
        ↪ between start times of trips
    for(difference in differenceVect) {
      j <- j + 1
      if(difference > 14400) { # if the difference between start
          ↪ times > 4 hours, assume new shift
        i <- i + 1
```

```r
        shiftVect[i] <- totalTime + tripDurations[j]
        totalTime <- 0
      }
      else {
        totalTime <- totalTime + difference
      }
    }
    busyTime <- sum(tripDurations) # add up times of all the trips
    shiftTime <- sum(shiftVect) # find the total shift time for
        ↪ the driver
  }
  return(busyTime/shiftTime)
}

# returns vector of proportion each driver is busy
dataAvgBusy <- function(taxiDrivers, dataFrame) {
  retVect <- vector()
  index <- 0
  for(driver in taxiDrivers) {
    index <- index + 1
    avg <- getShift(driver, dataFrame)
    retVect[index] <- avg
  }
  return(retVect)
}

driverTimesBusy <- dataAvgBusy(taxiDrivers, trainingData)

# Removing outliers
driverTimesBusy <- driverTimesBusy[driverTimesBusy < 1]
busyIQR <- IQR(driverTimesBusy)
busyQuantile <- quantile(driverTimesBusy)
driverTimesBusy <- driverTimesBusy[driverTimesBusy < busyQuantile
    ↪ [4] + busyIQR * 1.5]
driverTimesBusy <- driverTimesBusy[driverTimesBusy > busyQuantile
    ↪ [2] - busyIQR * 1.5]

# modeling normal distribution on data
hist(driverTimesBusy, freq = FALSE)
mu <- mean(driverTimesBusy)
sdev <- sqrt(mean((driverTimesBusy - mu) ^ 2))
curve(dnorm(x, mu, sdev), 0, 0.4, add = TRUE)

# Part B Bullet 3
# returns vector of trip times for each call type
callTypeTimes <- function(callType, dataframe) {
  vectTime <- dataframe$TRIP_TIME
  vectTime[dataframe$CALL_TYPE == callType]
}

tripTimesA <- callTypeTimes('A', trainingData4)
tripTimesB <- callTypeTimes('B', trainingData4)
tripTimesC <- callTypeTimes('C', trainingData4)

# finds radius of confidence interval for difference of means
    ↪ between vectors of two call types
findMargError <- function(callType1Times, callType2Times) {
  mean1 <- mean(callType1Times)
  mean2 <- mean(callType2Times)
  sdev1 <- sqrt(mean((callType1Times - mean1) ^ 2))
  sdev2 <- sqrt(mean((callType2Times - mean2) ^ 2))
  n1 <- length(callType1Times)
  n2 <- length(callType2Times)
  retVal <- sqrt((sdev1^2 / n1) + (sdev2^2 / n2))
  return(retVal * qnorm(0.975)) # substitute in qnorm(0.995) for
      ↪ 99%, qnorm(0.9995) for 99.9% confidence intervals
}

margAB <- findMargError(tripTimesA, tripTimesB)
margCB <- findMargError(tripTimesC, tripTimesB)
margCA <- findMargError(tripTimesC, tripTimesA)

diffAB <- mean(tripTimesA) - mean(tripTimesB)
diffCA <- mean(tripTimesC) - mean(tripTimesA)
diffCB <- mean(tripTimesC) - mean(tripTimesB)

# generating confidence intervals
AB_CI <- c(diffAB - margAB, diffAB + margAB)
CB_CI <- c(diffCB - margBC, diffCB + margBC)
CA_CI <- c(diffCA - margAC, diffCA + margAC)
AB_CI
CB_CI
CA_CI
```

B.1 Confidence Interval Calculation Code

```
#for prediciting trip time from origin call
trainingSubset <- trainingData4[,c(3,10)] # training subset for
    ↪ origin call and trip time
trainingSubset <- trainingSubset[!is.na(trainingSubset$ORIGIN_
    ↪ CALL),] #excludes incices that have NA
```

```
x <- trainingData4$ORIGIN_CALL
x <- na.exclude(x)
maxx <- max(x)
minx <- min(x)
predicttimes <- runif(10000, minx, maxx) #generates random values
    ↪  based on origin call distribution
predictTimes_df <- data.frame(ORIGIN_CALL = predicttimes)
linOut <- qeLin(trainingSubset, 'TRIP_TIME')
x <- predict(linOut, predictTimes_df)
hist(x) #histogram of predicted times
maxx <- max(x)
minx <- min(x)
lowerB <- qunif(0.025, minx, maxx)
upperB <- qunif(.975, minx, maxx)
CI <- c(lowerB, upperB) #creates 95% confidence interval
prop <- mean(lowerB <= x & x<= upperB) #find the proportion of
    ↪ predicted times in confidence interval

#for prediciting trip time from timestamp
trainingSubset <- trainingData4[,c(6,10)] # training subset for
    ↪ timestamp and trip time
x <- trainingData4$TIMESTAMP
maxx <- max(x)
minx <- min(x)
predicttimes <- runif(10000, minx, maxx) #generates random values
    ↪  based on timestamp distribution
predictTimes_df <- data.frame(TIMESTAMP = predicttimes)
linOut <- qeLin(trainingSubset, 'TRIP_TIME')
x <- predict(linOut, predictTimes_df)
hist(x) #histogram of predicted times
maxx <- max(x)
minx <- min(x)
lowerB <- qunif(0.025, minx, maxx)
upperB <- qunif(.975, minx, maxx)
CI <- c(lowerB, upperB) #creates 95% confidence interval
prop <- mean(lowerB <= x & x<= upperB) #find the proportion of
    ↪ predicted times in confidence interval

#predicting trip time from distance, call origin, and timestamp
trainingSubset <- trainingData4[,c(3,6,10,11)] # training subset
    ↪ for distance, call origin, timestamp, and trip time
trainingSubset <- trainingSubset[!is.na(trainingSubset$ORIGIN_
    ↪ CALL),] #excludes incices that have NA

x <- trainingSubset$TRIP_DISTANCE
xiqr <- IQR(x)
```

```
q <- quantile(x)
x <- x[x < q[4] + xiqr * 1.5]
x <- x[x > q[2] - xiqr * 1.5]
m <- mean(x)
s2 <- mean((x-m)^2)
rest <- m^2/s2
lest <- m/s2
predictDistances <- rgamma(10000, rest, lest) #generates random
    ↪ values based on distance distribution
predictDist_df <- data.frame(TRIP_DISTANCE = predictDistances)

x <- trainingSubset$TIMESTAMP
xiqr <- IQR(x)
maxx <- max(x)
minx <- min(x)
predicttimes <- runif(10000, minx, maxx) #generates random values
    ↪  based on timestamp distribution
predictTimes_df <- data.frame(TIMESTAMP = predicttimes)

x <- trainingData4$ORIGIN_CALL
x <- na.exclude(x)
maxx <- max(x)
minx <- min(x)
predictcall <- runif(10000, minx, maxx) #generates random values
    ↪ based on origin call distribution
predictcall_df <- data.frame(ORIGIN_CALL = predictcall)

linOut <- qeLin(trainingSubset, 'TRIP_TIME') #creates 95%
    ↪ confidence interval
x <- predict(linOut, c(predictcall_df,predictTimes_df,predictDist
    ↪ _df))
hist(x) #histogram of predicted times

m <- mean(x)
s2 <- mean((x-m)^2)
rest <- m^2/s2
lest <- m/s2
lowerB <- qgamma(0.025, rest, lest)
upperB <- qgamma(.975, rest, lest)
CI <- c(lowerB, upperB) #creates 95% confidence interval based on
    ↪  gamma
prop <- mean(lowerB <= x & x<= upperB) #find the proportion of
    ↪ predicted times in confidence interval

#creating histograms of taxi id and origin stand
```

```
trainingSubset <- trainingData4[,c(5,10)] #5 is taxi id and 4 is
    ↪ origin stand
#trainingSubset <- trainingSubset[!is.na(trainingSubset$ORIGIN_
    ↪ STAND),] #keep commented for taxi id, otherwise uncomment
x <- trainingData4$TAXI_ID # $TAXI_ID is for taxi id and $ORIGIN_
    ↪ STAND is for origin stand
#x <- na.exclude(x) #keep commented for taxi id, otherwise
    ↪ uncomment
hist(x) #see distribution of taxi id or origin stand
```

B.2 Part B Prediction Code

```
sampData <- sample_n(trainingData4, 100000)

## Single Predictor: Trip Distance
sampSubset <- data.frame(sampData[10:11])
trainingSubset <- data.frame(trainingData4[10:11])

testAcc10 <- vector()
# qeLin with single predictor: Trip Distance
for (i in 1:10) {
  linout <- qeLin(sampSubset, 'TRIP_TIME')
  testAcc10[i] <- linout$testAcc
}
testAcc10 # 10 values of testAcc for qeLin

linout <- qeLin(trainingSubset, 'TRIP_TIME')
predictx <- function(x) { # calls linout.predict(x)
  x1 <- data.frame('TRIP_DISTANCE' = x)
  predict(linout, x1)
}

plot(trainingSubset$TRIP_DISTANCE, trainingSubset$TRIP_TIME)
curve(predictx(x),add=TRUE, col="RED")

# qePolyLin with single predictor: Trip Distance
for (i in 1:10) {
  polylinout <- qePolyLin(sampSubset, 'TRIP_TIME')
  testAcc10[i] <- polylinout$testAcc
}
testAcc10 # 10 values of testAcc for qePolyLin

polylinout <- qePolyLin(trainingSubset, 'TRIP_TIME')
predictx <- function(x) { # calls polylinout.predict(x)
  x1 <- data.frame('TRIP_DISTANCE' = x)
  predict(polylinout, x1)
}

plot(trainingSubset$TRIP_DISTANCE, trainingSubset$TRIP_TIME)
curve(predictx(x),add=TRUE, col="RED")

# qeKNN with single predictor: Trip Distance
for (i in 1:10) {
  knnout <- qeKNN(sampSubset, 'TRIP_TIME')
  testAcc10[i] <- knnout$testAcc
}
testAcc10 # 10 values of testAcc for qeKNN

knnout <- qeKNN(trainingSubset, 'TRIP_TIME')
predictx <- function(x) { # calls knnout.predict(x)
  x1 <- data.frame('TRIP_DISTANCE' = x)
  predict(knnout, x1)
}

plot(trainingSubset$TRIP_DISTANCE, trainingSubset$TRIP_TIME)
curve(predictx(x),add=TRUE, col="RED")

# qeNeural with single predictor: Trip Distance
for (i in 1:10) {
  neuralout <- qeNeural(sampSubset, 'TRIP_TIME')
  testAcc10[i] <- neuralout$testAcc
}
testAcc10 # 10 values of testAcc for qeNeural

neuralout <- qeNeural(trainingSubset, 'TRIP_TIME')
predictx <- function(x) { # calls neuralout.predict(x)
  x1 <- data.frame('TRIP_DISTANCE' = x)
  predict(neuralout, x1)
}

plot(trainingSubset$TRIP_DISTANCE, trainingSubset$TRIP_TIME)
curve(predictx(x),add=TRUE, col="RED")

## Single Predictor: Origin Call
origcallSubset <- trainingData4[! trainingData4$ORIGIN_CALL %in%
    ↪ NA]
origcallSubset2 <- data.frame(c(origcallSubset[3], origcallSubset
    ↪ [10]))

# qeLin with single predictor: Origin Call
linout <- qeLin(origcallSubset2, 'TRIP_TIME')
```

```
linout$testAcc

predictx <- function(x) { # calls linout.predict(x)
  x1 <- data.frame('ORIGIN_CALL' = x)
  predict(linout, x1)
}

plot(origcallSubset2$ORIGIN_CALL, origcallSubset2$TRIP_TIME)
curve(predictx(x),add=TRUE, col="RED")

# qePolyLin with single predictor: Origin Call
polylinout <- qePolyLin(origcallSubset2, 'TRIP_TIME')
polylinout$testAcc

predictx <- function(x) { # calls polylinout.predict(x)
  x1 <- data.frame('ORIGIN_CALL' = x)
  predict(polylinout, x1)
}

plot(origcallSubset2$ORIGIN_CALL, origcallSubset2$TRIP_TIME)
curve(predictx(x),add=TRUE, col="RED")

# qeKNN with single predictor: Origin Call
knnout <- qeKNN(origcallSubset2, 'TRIP_TIME')
knnout$testAcc

predictx <- function(x) { # calls knnout.predict(x)
  x1 <- data.frame('ORIGIN_CALL' = x)
  predict(knnout, x1)
}

plot(origcallSubset2$ORIGIN_CALL, origcallSubset2$TRIP_TIME)
curve(predictx(x),add=TRUE, col="RED")

# qeNeural with single predictor: Origin Call
neuralout <- qeNeural(origcallSubset2, 'TRIP_TIME')
neuralout$testAcc

predictx <- function(x) { # calls neuralout.predict(x)
  x1 <- data.frame('ORIGIN_CALL' = x)
  predict(neuralout, x1)
}

plot(origcallSubset2$ORIGIN_CALL, origcallSubset2$TRIP_TIME)
curve(predictx(x),add=TRUE, col="RED")
```

```
## Single Predictor: Call Type

trainingSubset2 <- data.frame(c(trainingData4[2], trainingData4
    ↪ [10]))
#substitute A,B,C with 1,2,3 to plot
temp2 <- gsub("A", 1, trainingData4$CALL_TYPE)
temp3 <- gsub("B", 2, temp2)
x1 <- gsub("C", 3, temp3)

# spread out data 0.8 units to better illustrate density
x1 <- as.numeric(x1) + runif(1704759, min=0, max=0.8)

# qeLin with single predictor: Call Type
linout <- qeLin(trainingSubset2, 'TRIP_TIME')
predictx <- function(x) { # calls knnout.predict(x)
  x1 <- data.frame('CALL_TYPE' = x)
  predict(linout, x1)
}
plot(x1,trainingData4$TRIP_TIME)
curve(predictx("A")+0*x,1,1.8,add=TRUE, col='RED')
curve(predictx("B")+0*x,2,2.8,add=TRUE, col='GREEN')
curve(predictx("C")+0*x,3,3.8,add=TRUE, col='BLUE')

# zoom in to 0-5000 Trip Time to show difference in values of A,
    ↪ B, C
plot(x1,trainingData4$TRIP_TIME,ylim=c(0,5000))
curve(predictx("A")+0*x,1,1.8,add=TRUE, col='RED')
curve(predictx("B")+0*x,1,2.8,add=TRUE, col='GREEN')
curve(predictx("C")+0*x,1,3.8,add=TRUE, col='BLUE')

## Two Predictors: Trip Distance and Call Type
trainingSubset3 <- data.frame(c(trainingData4[2], trainingData4
    ↪ [10:11]))

callTypeTimes <- function(callType, dataframe) {
  vectTime <- dataframe$TRIP_TIME
  vectTime[dataframe$CALL_TYPE == callType]
}

tripTimesA <- callTypeTimes('A', trainingData4)
tripTimesB <- callTypeTimes('B', trainingData4)
tripTimesC <- callTypeTimes('C', trainingData4)

callTypeDists <- function(callType, dataframe) {
  vectTime <- dataframe$TRIP_DISTANCE
  vectTime[dataframe$CALL_TYPE == callType]
```

```
}

tripDistsA <- callTypeDists('A', trainingData4)
tripDistsB <- callTypeDists('B', trainingData4)
tripDistsC <- callTypeDists('C', trainingData4)

# qeLin with two predictors: Trip Distance and Call Type
linout <- qeLin(trainingSubset3, 'TRIP_TIME')
linout$testAcc

predicta <- function(x){
  x1 <- data.frame(CALL_TYPE='A', TRIP_DISTANCE=x)
  predict(linout, x1)
}
predictb <- function(x){
  x1 <- data.frame(CALL_TYPE='B', TRIP_DISTANCE=x)
  predict(linout, x1)
}
predictc <- function(x){
  x1 <- data.frame(CALL_TYPE='C', TRIP_DISTANCE=x)
  predict(linout, x1)
}

plot(tripDistsA, tripTimesA)
curve(predicta(x),add=TRUE, col="RED")

plot(tripDistsB, tripTimesB)
curve(predictb(x),add=TRUE, col="RED")

plot(tripDistsC, tripTimesC)
curve(predictc(x),add=TRUE, col="RED")

# qeNeural with two predictors: Trip Distance and Call Type
neuralout <- qeNeural(trainingSubset3, 'TRIP_TIME')
neuralout$testAcc

predicta <- function(x){
  x1 <- data.frame(CALL_TYPE='A', TRIP_DISTANCE=x)
  predict(neuralout, x1)
}
predictb <- function(x){
  x1 <- data.frame(CALL_TYPE='B', TRIP_DISTANCE=x)
  predict(neuralout, x1)
}
predictc <- function(x){
  x1 <- data.frame(CALL_TYPE='C', TRIP_DISTANCE=x)
  predict(neuralout, x1)
}

plot(tripDistsA, tripTimesA)
curve(predicta(x),add=TRUE, col="RED")

plot(tripDistsB, tripTimesB)
curve(predictb(x),add=TRUE, col="RED")

plot(tripDistsC, tripTimesC)
curve(predictc(x),add=TRUE, col="RED")

# qeRF with two predictors: Trip Distance and Call Type
sampSubset2 <- data.frame(c(sampData[2], sampData[10:11]))
rfout <- qeRF(sampSubset2, 'TRIP_TIME')
rfout$testAcc

predicta <- function(x){
  x1 <- data.frame(CALL_TYPE='A', TRIP_DISTANCE=x)
  predict(rfout, x1)
}
predictb <- function(x){
  x1 <- data.frame(CALL_TYPE='B', TRIP_DISTANCE=x)
  predict(rfout, x1)
}
predictc <- function(x){
  x1 <- data.frame(CALL_TYPE='C', TRIP_DISTANCE=x)
  predict(rfout, x1)
}

plot(sampSubsetA$TRIP_DISTANCE, sampSubsetA$TRIP_TIME)
curve(predicta(x),add=TRUE, col="RED")

plot(sampSubsetB$TRIP_DISTANCE, sampSubsetB$TRIP_TIME)
curve(predictb(x),add=TRUE, col="RED")

plot(sampSubsetC$TRIP_DISTANCE, sampSubsetC$TRIP_TIME)
curve(predictc(x),add=TRUE, col="RED")

## Testing Different Hyperparameter Accuracies
testAcc10 <- vector()
for (i in 1:10) {
  xout <- qeNeural(sampSubset2, 'TRIP_TIME', nEpoch = 40) # Can
      ↪ substitute qeNeural with other models such as qeRF, and
      ↪ nEpoch to other parameter or values
  testAcc10[i] <- xout$testAcc
```

```
}
testAcc10
    #3.4.4
    traininital <- read.csv("train.csv")
    trainingData2<- traininital[! traininital$MISSING_DATA %in% "
        ↪ True", ]
    trainingData3<- trainingData2[! trainingData2$POLYLINE %in%
        ↪ "[]", ]
    train <- trainingData3[! trainingData3$POLYLINE %in% NA, ]

    tripDuration <- nrow(train)
    library(stringr)
    for(i in 1:nrow(train))
        tripDuration[i] <- (floor(str_count(train[i,"POLYLINE
            ↪ "],",")+1)/2)*15
    train$TRIP_TIME <- tripDuration

    time_of_day <- nrow(train)
    for(i in 1:nrow(train))
        time_of_day[i] <- train[i,"TIMESTAMP"] %% (3600*24)
    train$time_of_day <- time_of_day
    trainingSubset <- data.frame(train[10:11])
    library(regtools)
    polylinout <- qePolyLin(trainingSubset, 'TRIP_TIME')
    predictx <- function(x) { # calls polylinout.predict(x)
        x1 <- data.frame('time_of_day' = x)
        predict(polylinout, x1)
    }
    plot(trainingSubset$time_of_day, trainingSubset$TRIP_TIME)
    curve(predictx(x),add=TRUE, col="RED")
```

C Group Contributions

C.1 Harrod Tang

laTeX write up (3.1, 3.2, 3.3, 3.4.1), initial code (3.1, 3.2, 3.3, 3.4.1), graph generation (3.1, 3.2, 3.3, 3.4.1). Developed confidence interval solution in 3.4.1 that formed the basis for generating later confidence intervals. Edited rough draft of laTeX document.

C.2 Rayum Shahed

latex write up (3.4.2, 3.4.3, 3.4.5, 3.4.6), devised a way to generate random values for predictors and make accurate predictions and test the predicted trip times and coded it . Created graphs for predictors vs trip time and histogram for predictors.

C.3 Joshua Sanchez

laTex Write up(3.3, 3.4.4), Intial code (3.2, 3.3, 3.4.4), graph generation(3.4.5). I came up with an alternative solution for B.2 to verify that our results we correct. I introduced simply making it the total time driving over the total shift time. In section b.4 I came up with the code that would extract the numbers from the string in Polyline. I also came up with the code for finding the difference between two GPS coordinates from the library geosphere. I had an alternative implementation that would track the all distances between Polyline instead of just the last and first. Did section 3.4.4 by myself with some help from Kyle.

C.4 Kyle Li

Code (3.2,3.3,3.4.1, 3.4.2, 3.4.3, 3.4.5), graphs (3.4.1, 3.4.2, 3.4.3, 3.4.5). Organized LaTeX code snippet and figures. Generated tables of test accuracies for the single predictor models and for the different hyperparameters for qeRF and qeNeural. Generated all prediction graphs except for 3.4.4.