New York City Taxi Analysis

Group 1: Anthony Pagan, Tommy Jenkins, Violeta Stoyanova,
Todd Lisa, Peter Kowalchuk, Eleanor Secoquian
DATA 621
CUNY SPS, MS Data Science

Abstract

"The Partnership for New York City's one-page study asserted that "excess congestion" deprives the five boroughs and the suburbs of Long Island, Westchester and Rockland counties and northern New Jersey $20 billion annually" ("Traffic Congestion", 2018). Moreover, emergency services reports life or death consequences based on NYC traffic conditions, according to multiple sources (FDNY, 2019). We seek to analyze publicly accessible datasets in the service of better understanding the factors that influence traffic as encapsulated in the Kaggle competition: New York City Taxi Trip Duration, originally published in NYC Taxi and Limousine Commission (TLC) ("About TLC", 2019). Analysts have cited the hiring of a technologically savvy commissioner to lead the Taxi and Limousine Commission as an ongoing concern for the New York City government (Skandul, 2019; Sanders, 2019)..We hope that our work with this dataset to predict NYC yellow taxi trip duration will serve to augment the high level of technical analysis that coincides with Kaggle competitions, however we aim for our explanatory statistics to instead offer a better understanding of NYC safety and logistics for everyone. And while the evaluation metric for the Kaggle competition is RMSE, we have instead focused on describing variables clearly, seeking the best resources for tidyverse descriptive statistics online and have kept our model comparisons consistent with linear and multilinear regression analyzed against Box Cox transformation and AIC-BIC evaluations (headortails, 2019).

Introduction and Background

The quality of Kaggle submissions is quite high due to the competitive nature of the platform. We viewed one particular submission as emblematic of a mature yet accessible presentation of the scientific data and thus modeled our exploratory data analysis off of it (headortails, 2019). While many of the competition participants conducted machine learning in order to turn the 1.5 million training observations Including pick-up and drop-off coordinates and times to predict the duration of taxi ride by vendor we instead focused on exploratory statistics and linear model selection for a subset of that data. Further differences remain with regard to our use of the data. Most notably external data regarding weather as well as an advanced algorithm for computing trip trajectories were used. Our analysis accounted for passenger count, vendor ID, day of the week as well as hour, passenger count, storage flag, latitude, longitude, speed, airport travel, and finally trip duration.

Data

With such a large data set our first order of business was to analyze and view the missing values Nearly a third of our data had Na's, which we removed. Viewing the shape and normality of our variables let us to conclude that much of our data including directional data such as latitude and longitude and trip duration was very similar. We reduced our sample to 20% of the nearly 1 million observations and proceeded to factor the data into vendor ID and month, plotting both density and linearity.

While we found that many of our predictor variables may have benefited from transformation or feature engineering to due to lack of constant variance, they seem to approach normality when viewed in the histograms after a log y transformation. The response variable, trip duration, also seemed to lack variance between observations, however it varied significantly with a large number of observations occurring outside of the first and third quartile. As part of our exploratory analysis we were able to partition fast and slow rides by day and chart the flow.

Methodology and Results

We used Box Cox transformation, Cook's Distance, the Breusch-Pagan test, Generalized linear models, Stepwise regression, and Coefficient analysis to evaluate our variables within the following types of models:

- Poisson
- Gaussian
- Negative Binomial

Based on the following tests

H0: Homoscedasticity. The errors have constant variance about the true model.
H1: Heteroscedasticity. The errors have non-constant variance about the true model.

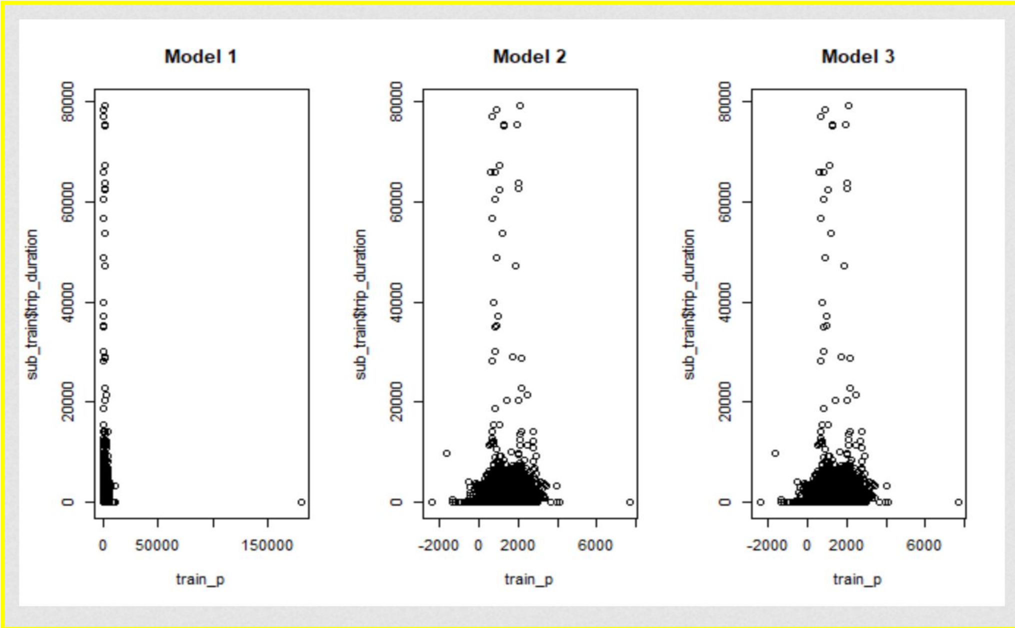
Leverage, Outliers, Influence, coef Change

Selection Model:

	Model 1	Model 2	Model 3
AIC	112978116.99741	4703786.61775627	4703753.17855375
BIC	112978159.30903	4703850.08518705	4703827.22388967

With 3 models computed, we select the model with the lowest combination of AIC and BIC. From the table, we can see the model to pick is model 1.

Model 1 showed the best result. We can observe its performance by plotting the datasets Vendor_ID values against the predicted values.



References

FDNY: Traffic — Not Bike Lanes — is to Blame for Increased Response Times. (2019, September 19). Retrieved December 21, 2019, from Streetsblog New York City website: https://nyc.streetsblog.org/2019/09/19/fdny-traffic-not-bike-lanes-is-to-blame-for-increased-response-times

headsortails. (2019, January 31). NYC Taxi EDA - Update: The fast & the curious. Retrieved December 21, 2019, from Kaggle.com website: https://www.kaggle.com/headsortails/nyc-taxi-eda-update-the-fast-the-curious/report

Li Gong, Xi Liu, Lun Wu & Yu Liu (2016) Inferring trip purposes and uncovering travel patterns from taxi trajectory data, Cartography and Geographic Information Science, 43:2, 103-114, DOI: 10.1080/15230406.2015.1014424

Liu, X., Gong, L., Gong, Y., & Liu, Y. (2015). Revealing travel patterns and city structure with taxi trip data. *Journal of Transport Geography*, *43*, 78–90. https://doi.org/10.1016/j.jtrangeo.2015.01.016

New York City Taxi Trip Duration | Kaggle. (2017). Retrieved December 21, 2019, from Kaggle.com website: https://www.kaggle.com/c/nyc-taxi-trip-duration/data#

Use holidays as a feature. New York City Taxi Trip Duration | Kaggle. (2017). Retrieved December 21, 2019, from Kaggle.com website: https://www.kaggle.com/c/nyc-taxi-trip-duration/discussion/37192

Qianxi Zheng. (2019). Would Uber Help to Reduce Traffic Congestion? Retrieved December 21, 2019, from Creative Matter website: https://creativematter.skidmore.edu/econ_studt_schol/129/

Sanders, A. (2019, October 3). De Blasio struggles to find new Taxi and Limousine commissioner after first pick's disastrous performance at C. Retrieved December 21, 2019, from nydailynews.com website: https://www.nydailynews.com/news/politics/ny-city-council-de-blasio-official-tlc-commissioner-jeff-lynch-20191003-cfx27yihafcebj4e62wytsprpu-story.html

Skandul, E. (2019, October 17). How New York City taxis can get ahead of Uber and Lyft. Retrieved December 21, 2019, from CSNY website: https://www.cityandstateny.com/articles/opinion/commentary/how-new-york-city-taxis-can-get-ahead-of-uber-and-lyft.html

Traffic congestion costs metro economy $20 billion a year: study. (2018, January 18). Retrieved December 21, 2019, from Crain's New York Business website: https://www.crainsnewyork.com/article/20180118/POLITICS/180119895/traffic-congestion-costs-metro-economy-20-billion-a-year-study

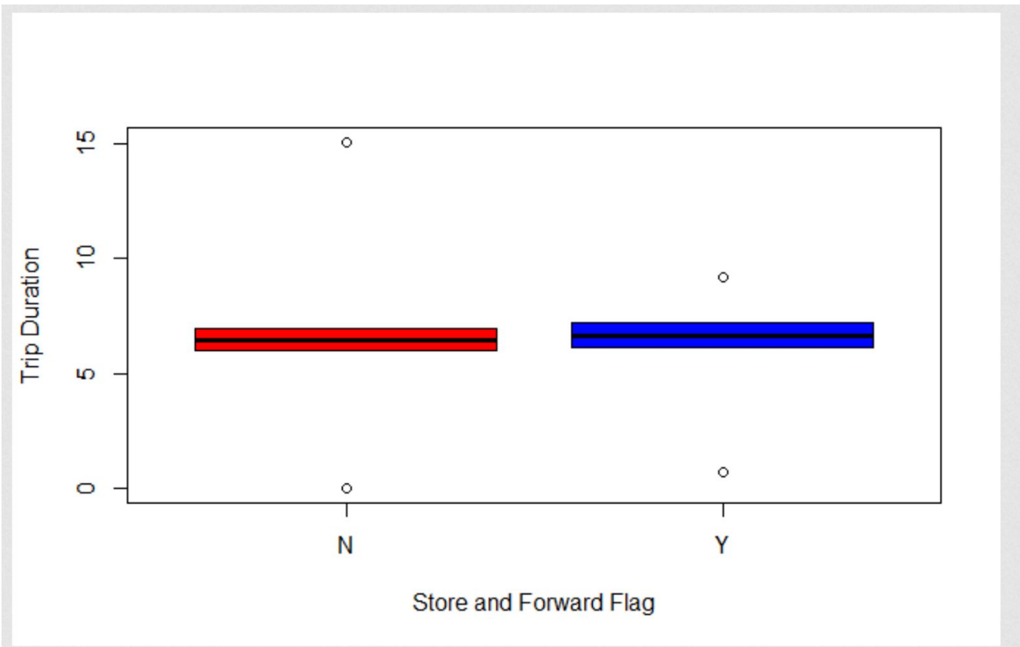
TLC Trip Record Data. (2019). Retrieved December 21, 2019, from Nyc.gov website: https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

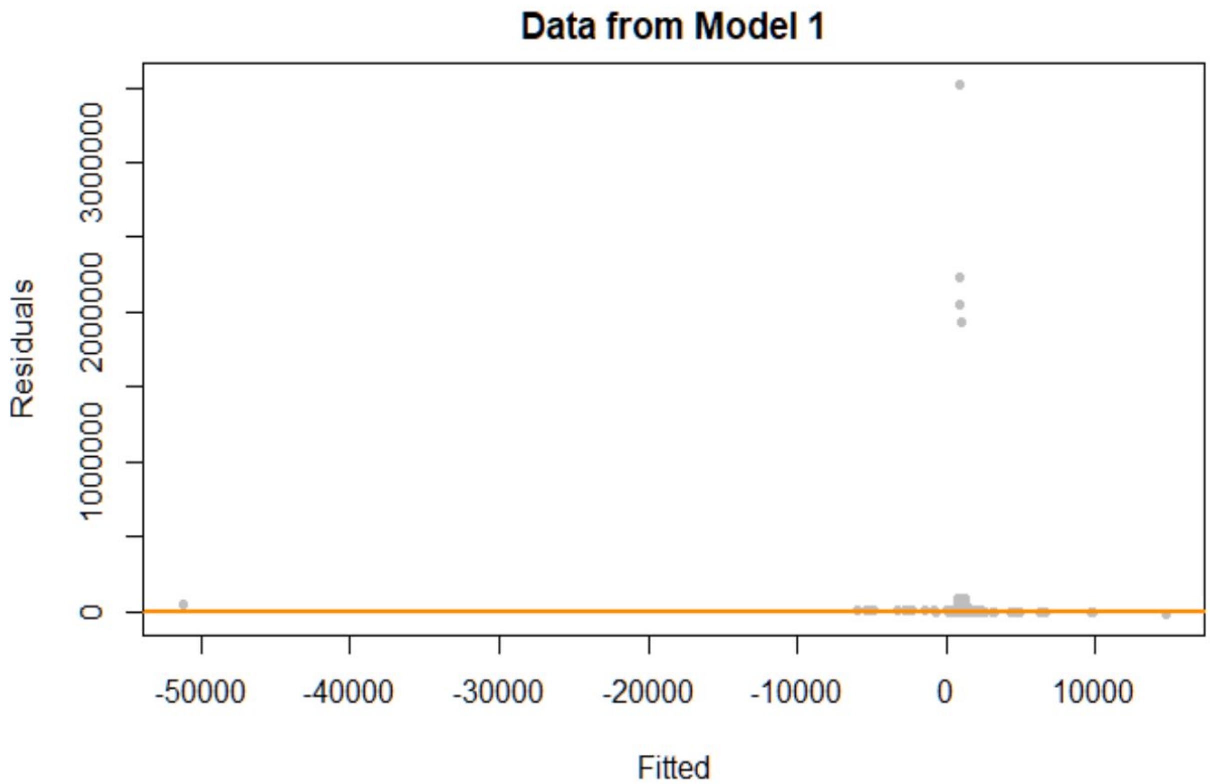
Wang, W., Pan, L., Yuan, N., Zhang, S., & Liu, D. (2015). A comparative analysis of intra-city human mobility by taxi. *Physica A: Statistical Mechanics and Its Applications*, *420*, 134–147. https://doi.org/10.1016/j.physa.2014.10.085

Zhang, J. (2012). Smarter outlier detection and deeper understanding of large-scale taxi trip records. *Proceedings of the ACM SIGKDD International Workshop on Urban Computing - UrbComp '12*. https://doi.org/10.1145/2346496.2346521

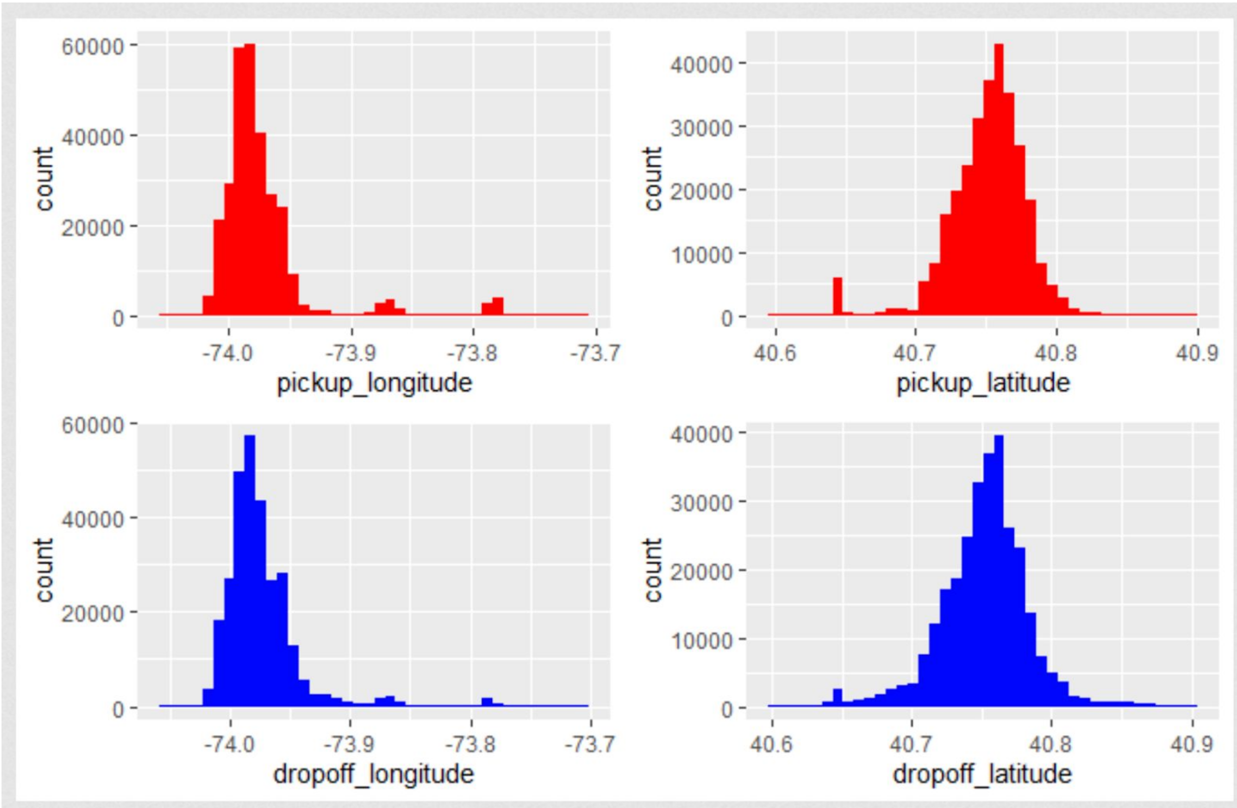
Appendices

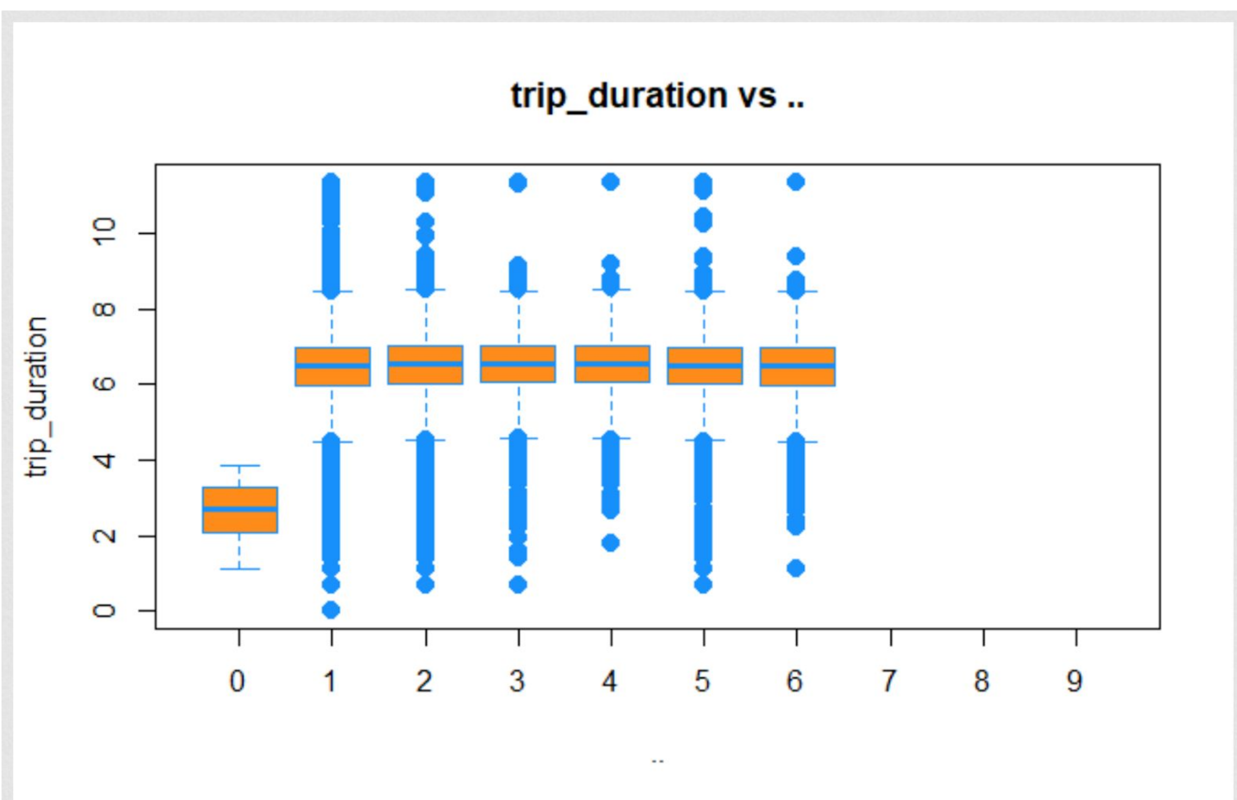
Supplemental tables and/or figures

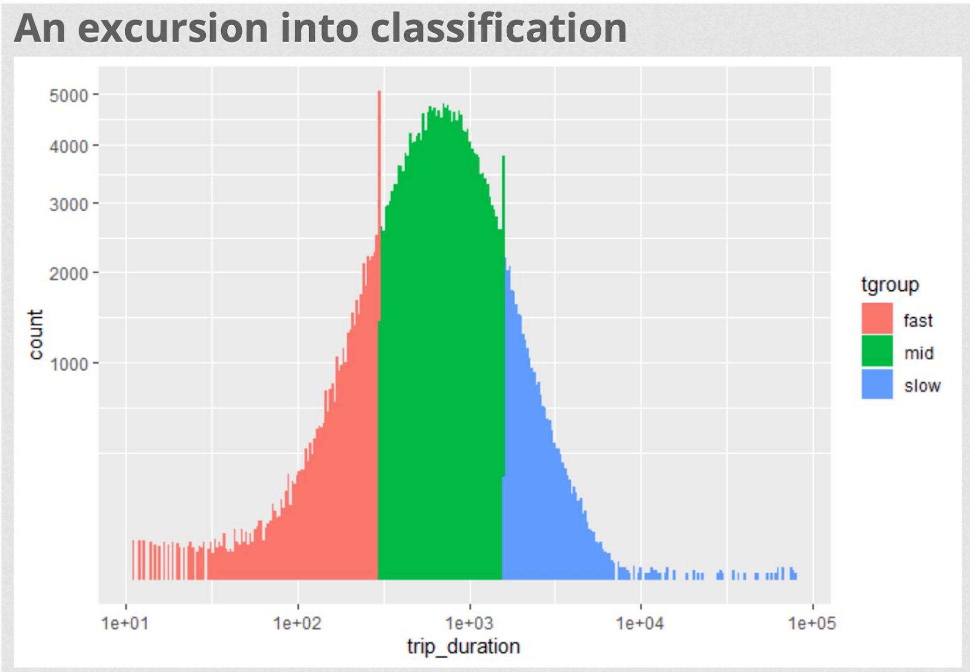




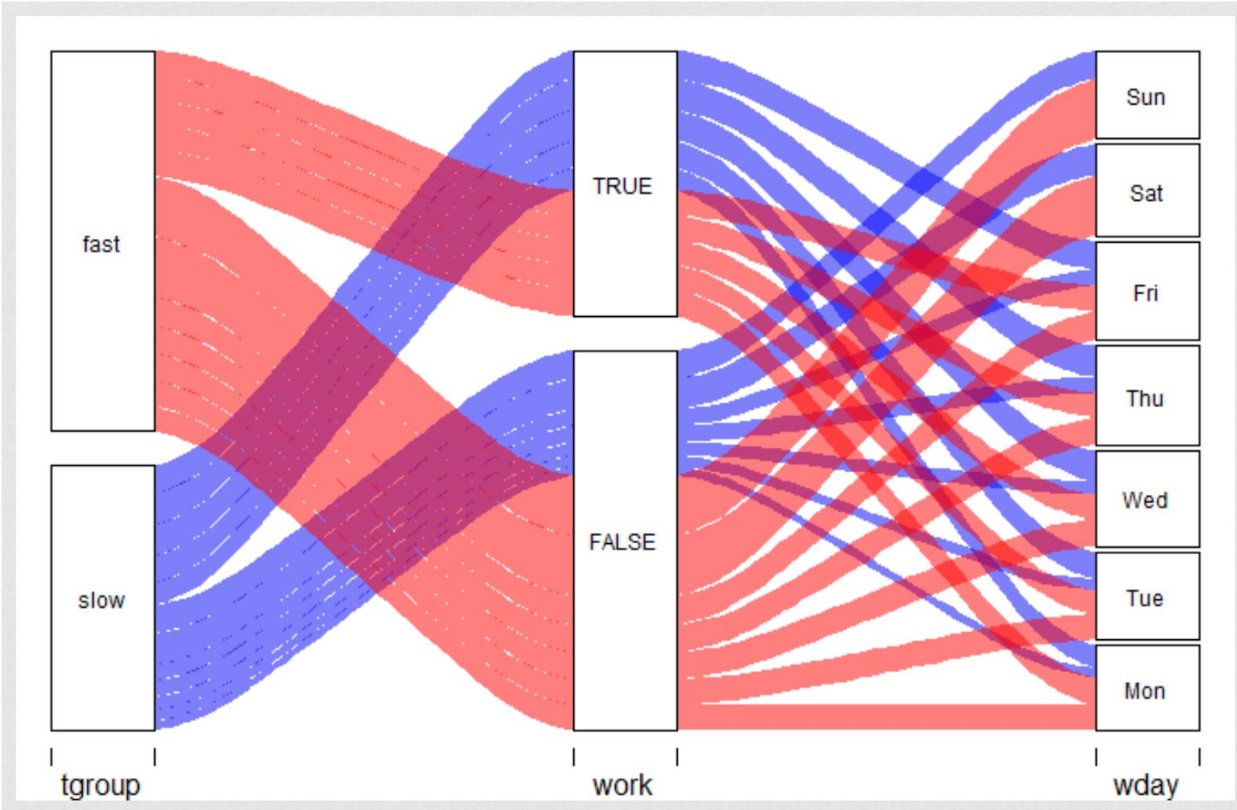
Data from Model 1
Residuals
0
1000000
2000000
3000000
Fitted
-50000
-40000
-30000
-20000
-10000
0
10000

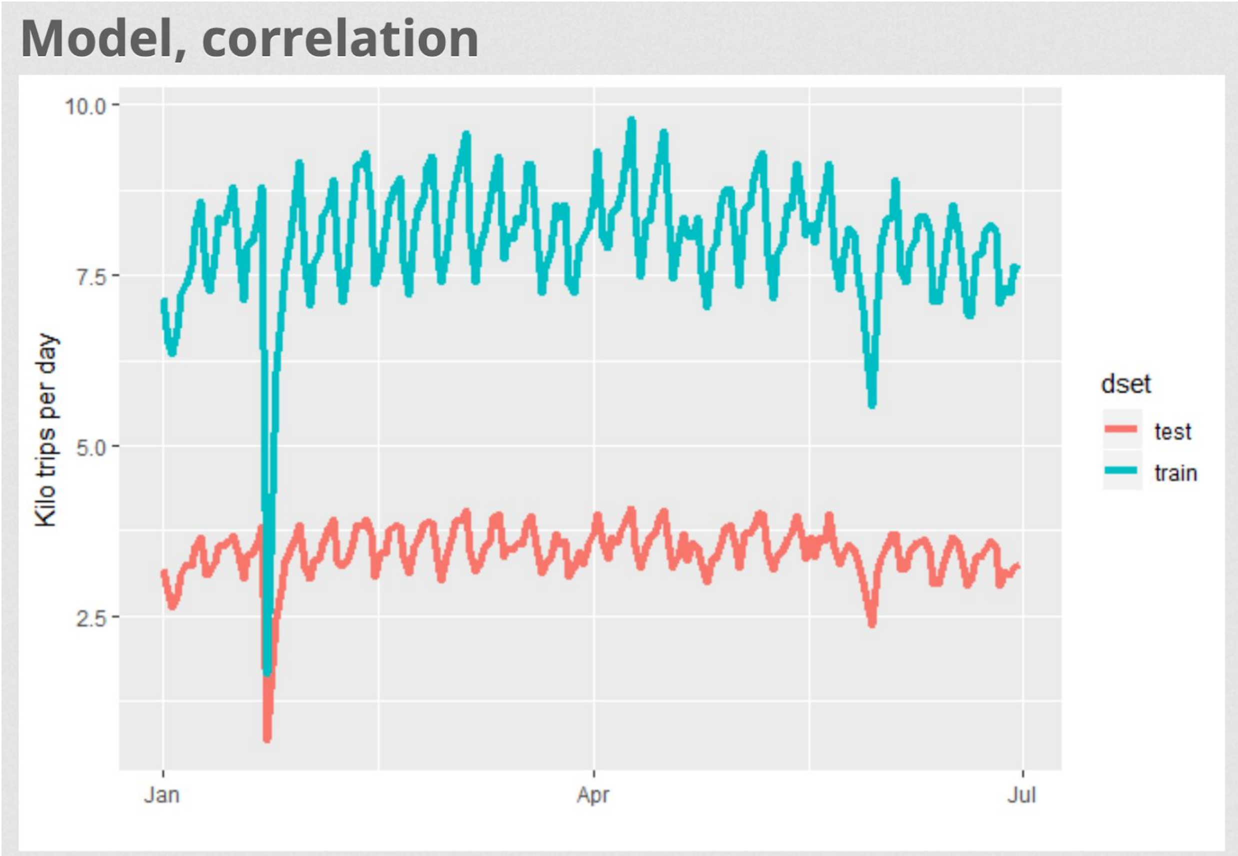

count
pickup_longitude
pickup_latitude
dropoff_longitude
dropoff_latitude
60000
40000
20000
0
30000
10000
-74.0
-73.9
-73.8
-73.7
40.6
40.7
40.8
40.9

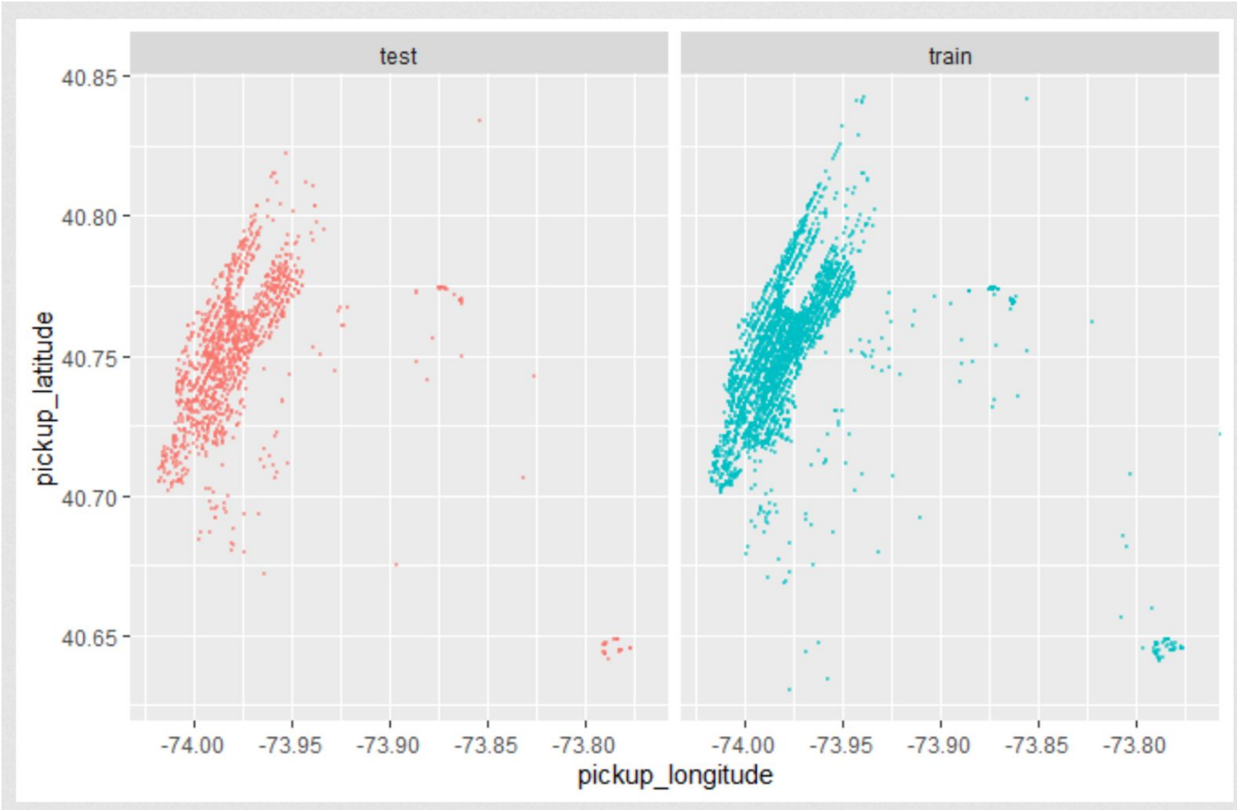

trip_duration vs ..
trip_duration
0
2
4
6
8
10
1
3
5
7
9
..


An excursion into classification
5000
4000
3000
2000
1000
count
1e+01
1e+02
1e+03
1e+04
1e+05
trip_duration
tgroup
fast
mid
slow


1.00
0.75
0.50
0.25
0.00
count
Mon
Tue
Wed
Thu
Fri
Sat
Sun
wday
Jan
Feb
Mar
Apr
May
Jun
month
FALSE
TRUE
work


fast
slow
TRUE
FALSE
Sun
Sat
Fri
Thu
Wed
Tue
Mon
tgroup
work
wday


Model, correlation
10.0
7.5
5.0
2.5
Kilo trips per day
Jan
Apr
Jul
dset
test
train



R statistical programming code

Code used in analysis

```
#list.of.packages <-
c("alluvial","caret","caret","corrplot","corrplot","data.table","dplyr","faraway","forcats","geosph#ere","ggplot2","ggplot2","ggplot2","grid","gridExtra","jtools","kableExtra","knitr","leaflet","leaflet.extras","leaps",#"lubridate","maps","MASS","mice","naniar","pander","patchwork","prettydoc","pROC","psych","RColorBrewer","readr","resha#pe2","scales","stringr","tibble","tidyr","tidyverse","xgboost","widgetframe","Rcpp")
#new.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[,"Package"])]
#if(length(new.packages)) install.packages(new.packages)
require(knitr)
knitr::opts_chunk$set(echo = FALSE, warning = FALSE,fig.align='center')

library(faraway)
library(MASS)
library(psych)
library(pROC)
library(corrplot)
library(jtools)
library(mice)
library('corrr')
```

```
library(kableExtra)
library(gridExtra)
library(pander)
library(zoo)
library(lmtest)
library(corrr)
library(broom)

#devtools::install_github("thomasp85/patchwork")
library(patchwork)
library(tidyverse)
library(ggplot2)
library(ggplot2)
library(reshape2)
library(leaps)
library(caret)
library(naniar)
library('ggplot2') # visualisation
library('scales') # visualisation
library('grid') # visualisation
library('RColorBrewer') # visualisation
library('corrplot') # visualisation
library('alluvial') # visualisation
library('dplyr') # data manipulation
library('readr') # input/output
library('data.table') # data manipulation
library('tibble') # data wrangling
library('tidyr') # data wrangling
library('stringr') # string manipulation
library('forcats') # factor manipulation
library('lubridate') # date and time
library('geosphere') # geospatial locations
library('leaflet') # maps
library('leaflet.extras') # maps
library('maps') # maps
library('xgboost') # modelling
library('caret') # modelling
library('widgetframe') #visualizaiton
library('grid')
library('gridExtra')
# Define multiple plot function
#
# ggplot objects can be passed in ..., or to plotlist (as a list of ggplot objects)
# - cols:   Number of columns in layout
# - layout: A matrix specifying the layout. If present, 'cols' is ignored.
#
# If the layout is something like matrix(c(1,2,3,3), nrow=2, byrow=TRUE),
# then plot 1 will go in the upper left, 2 will go in the upper right, and
```

```
# 3 will go all the way across the bottom.
#

multiplot <- function(..., plotlist=NULL, file, cols=1, layout=NULL) {

 # Make a list from the ... arguments and plotlist
 plots <- c(list(...), plotlist)

 numPlots = length(plots)

 # If layout is NULL, then use 'cols' to determine layout
 if (is.null(layout)) {
   # Make the panel
   # ncol: Number of columns of plots
   # nrow: Number of rows needed, calculated from # of cols
   layout <- matrix(seq(1, cols * ceiling(numPlots/cols)),
                    ncol = cols, nrow = ceiling(numPlots/cols))
 }

 if (numPlots==1) {
   print(plots[[1]])
 } else {
   # Set up the page
   grid.newpage()
   pushViewport(viewport(layout = grid.layout(nrow(layout), ncol(layout))))

   # Make each plot, in the correct location
   for (i in 1:numPlots) {
     # Get the i,j matrix positions of the regions that contain this subplot
     matchidx <- as.data.frame(which(layout == i, arr.ind = TRUE))

     print(plots[[i]], vp = viewport(layout.pos.row = matchidx$row,
                                     layout.pos.col = matchidx$col))
   }
 }
}
var_stats<- function(df){
 wt<-data.frame(columns=colnames(df))
 wt$na_count <- sapply(df, function(y) sum(is.na(y)))
 wt$neg_count <- sapply(df, function(y) sum(y<0))
 wt$zero_count <- sapply(df, function(y) sum(as.integer(y)==0))
 wt$unique_count <- sapply(df, function(y) sum(n_distinct(y)))
 print(wt)
 return(wt)
}
rows <-
c("id","vendor_id","pickup_datetime","dropoff_datetime","passenger_count","pickup_longitude","pickup_latitude",
```

```
"dropoff_longitude","dropoff_latitude","store_and_fwd_flag","trip_duration")

def <- c("a unique identifier for each trip",
"a code indicating the provider associated with the trip record",
"date and time when the meter was engaged",
"date and time when the meter was disengaged",
"the number of passengers in the vehicle (driver entered value)",
"the longitude where the meter was engaged",
"the latitude where the meter was engaged",
"the longitude where the meter was disengaged",
"the latitude where the meter was disengaged",
"This flag indicates whether the trip record was held in vehicle memory before sending to the
vendor because the vehicle did not have a connection to the server: Y=store and forward;
N=not a store and forward trip",
"duration of the trip in seconds")

kable(cbind(rows, def), col.names = c("Variable Name", "Definition")) %>% kable_styling()
train <- as_tibble(fread('data/train.csv'))
test <- as_tibble(fread('data/test.csv'))
sample_submit <- as_tibble(fread('data/sample_submission.csv'))
#str(train)
glimpse(train)
#summary(train)
#describe(train)
names(train)
names(test)
#glimpse(test)

#
vars_to_add <- train[!names(train) %in% names(test)]

 #vvvvv
## Combining train and test

combine <- rbind(train %>% mutate(dset = "train"),
                 test %>% mutate(dset = "test",
                              dropoff_datetime = NA,
                              trip_duration = NA))
combine <- combine %>% mutate(dset = factor(dset))
glimpse(combine)
summary(combine)
var_stats(combine)
gg_miss_upset(combine)
summary(complete.cases(combine))
train <- train %>%
  mutate(pickup_datetime = ymd_hms(pickup_datetime),
       dropoff_datetime = ymd_hms(dropoff_datetime),
       vendor_id = factor(vendor_id),
```

```
    passenger_count = factor(passenger_count))
#ggplot(combine, aes(trip_duration)) +
#  geom__histogram(aes(y = ..density..)

attach(train)
boxplot(by(log(train$trip_duration),train$store_and_fwd_flag,summary),col=c("red","blue"),
xlab="Store and Forward Flag", ylab="Trip Duration")
by(log(train$trip_duration),train$store_and_fwd_flag,summary)

#plot(trip_duration ~ dropoff_longitude,pch  = 20,cex  = 2,col  = "grey")

train[sapply(train, function(x) is.numeric(x) && !is.na(x))] %>%
  gather() %>%
  ggplot(aes(value), main="") +
  facet_wrap(~ key, scales = "free") +
  geom_histogram()

sub_train = train%>%sample_frac(.2)
attach(sub_train)
g1<-ggplot(sub_train, aes(x=I(pickup_latitude*pickup_longitude), y=log(trip_duration), color
= store_and_fwd_flag)) +geom_point() +stat_smooth(method="glm", se=TRUE)
g2<-ggplot(sub_train, aes(x=I(dropoff_latitude*dropoff_longitude), y=log(trip_duration), color
= store_and_fwd_flag)) +geom_point() +stat_smooth(method="glm", se=TRUE)
g3<-ggplot(sub_train, aes(x=pickup_datetime, y=log(trip_duration), color =
store_and_fwd_flag)) +geom_point() +stat_smooth(method="glm", se=TRUE)
g4<-ggplot(sub_train, aes(x=dropoff_datetime, y=log(trip_duration), color =
store_and_fwd_flag)) +geom_point() +stat_smooth(method="glm", se=TRUE)
grid.arrange(g1, g2, g3, g4, ncol = 2)

g1<-ggplot(sub_train, aes(x=I(pickup_latitude*pickup_longitude), y=log(trip_duration), color
= vendor_id)) +geom_point() +stat_smooth(method="glm", se=TRUE)
g2<-ggplot(sub_train, aes(x=I(dropoff_latitude*dropoff_longitude), y=log(trip_duration), color
= vendor_id)) +geom_point() +stat_smooth(method="glm", se=TRUE)
g3<-ggplot(sub_train, aes(x=pickup_datetime, y=log(trip_duration), color = vendor_id))
+geom_point() +stat_smooth(method="glm", se=TRUE)
g4<-ggplot(sub_train, aes(x=dropoff_datetime, y=log(trip_duration), color = vendor_id))
+geom_point() +stat_smooth(method="glm", se=TRUE)
grid.arrange(g1, g2, g3, g4, ncol = 2)


pairs(sub_train[sapply(sub_train, function(x) is.numeric(x))], col = "dodgerblue")


ssub_train<-sub_train[sapply(sub_train, function(x) is.numeric(x) && !is.na(x))]

ssub_train %>%
  correlate() %>%
  network_plot(min_cor = .2)
```

```
#log(sub_train$trip_duration) %>% as.double() %>% boxplot()
#bins
#scale_x_log10() +
#scale_y_sqrt()
attach(sub_train)
boxplot(log(trip_duration) ~ as.factor(passenger_count),
    xlab   = "..",
    ylab   = "trip_duration",
    main   = "trip_duration vs ..",
    pch    = 20,
    cex    = 2,
    col    = "darkorange",
    border = "dodgerblue")

p1 <- sub_train %>%
  filter(pickup_longitude > -74.05 & pickup_longitude < -73.7) %>%
  ggplot(aes(pickup_longitude)) +
  geom_histogram(fill = "red", bins = 40)

p2 <- sub_train %>%
  filter(dropoff_longitude > -74.05 & dropoff_longitude < -73.7) %>%
  ggplot(aes(dropoff_longitude)) +
  geom_histogram(fill = "blue", bins = 40)

p3 <- sub_train %>%
  filter(pickup_latitude > 40.6 & pickup_latitude < 40.9) %>%
  ggplot(aes(pickup_latitude)) +
  geom_histogram(fill = "red", bins = 40)

p4 <- sub_train %>%
  filter(dropoff_latitude > 40.6 & dropoff_latitude < 40.9) %>%
  ggplot(aes(dropoff_latitude)) +
  geom_histogram(fill = "blue", bins = 40)

layout <- matrix(c(1,2,3,4),2,2,byrow=FALSE)
multiplot(p1, p2, p3, p4, layout=layout)
p1 <- 1; p2 <- 1; p3 <- 1; p4 <- 1
#jfk_coord <- tibble(lon = -73.778889, lat = 40.639722)

#la_guardia_coord <- tibble(lon = -73.872611, lat = 40.77725)


#train$jfk_dist_pick <- distCosine(pick_coord, jfk_coord)

#train$jfk_dist_drop <- distCosine(drop_coord, jfk_coord)

#train$lg_dist_pick <- distCosine(pick_coord, la_guardia_coord)
```

```
#train$lg_dist_drop <- distCosine(drop_coord, la_guardia_coord)


pick_coord <- sub_train %>% select(pickup_longitude, pickup_latitude)
drop_coord <- sub_train %>% select(dropoff_longitude, dropoff_latitude)
sub_train$dist <- distCosine(pick_coord, drop_coord)
#train$bearing = bearing(pick_coord, drop_coord)


sub_train <- sub_train %>%
 mutate(speed = dist/trip_duration*3.6,
        date = date(pickup_datetime),
        month = month(pickup_datetime, label = TRUE),
        wday = wday(pickup_datetime, label = TRUE),
        wday = fct_relevel(wday, c("Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun")),
        hour = hour(pickup_datetime),
        work = (hour %in% seq(8,18)) & (wday %in% c("Mon","Tue","Wed","Thu","Fri")),
#        jfk_trip = (jfk_dist_pick < 2e3) | (jfk_dist_drop < 2e3),
#        lg_trip = (lg_dist_pick < 2e3) | (lg_dist_drop < 2e3),
#        blizzard = !( (date < ymd("2016-01-22") | (date > ymd("2016-01-29"))) )
        )

sub_train <- sub_train %>%
 filter(trip_duration < 22*3600,
        dist > 0 | (near(dist, 0) & trip_duration < 60),
#        jfk_dist_pick < 3e5 & jfk_dist_drop < 3e5,
```

```
        trip_duration > 10,
        speed < 100)

sub_train %>%
  select(-id, -pickup_datetime, -dropoff_datetime, -date) %>% #-jfk_dist_pick,
 #        -jfk_dist_drop, -lg_dist_pick, -lg_dist_drop, -date) %>%
  mutate(passenger_count = as.integer(passenger_count),
        vendor_id = as.integer(vendor_id),
        store_and_fwd_flag = as.integer(as.factor(store_and_fwd_flag)),
 #        jfk_trip = as.integer(jfk_trip),
        wday = as.integer(wday),
        month = as.integer(month),
        work = as.integer(work))%>%
 #        lg_trip = as.integer(lg_trip),
 #        blizzard = as.integer(blizzard),
 #        has_snow = as.integer(has_snow),
  #       has_rain = as.integer(has_rain)) %>%
#
  select(trip_duration, speed, everything()) %>%
  cor(use="complete.obs", method = "spearman") %>%
  corrplot(type="lower", method="circle", diag=FALSE)

train_group <- sub_train %>%
  mutate(tgroup = case_when(trip_duration < 3e2 ~ "fast",
                            trip_duration >= 3e2 & trip_duration <= 1.6e3 ~ "mid",
                            trip_duration > 1.6e3 ~ "slow"))
```

```
train_group %>%
  ggplot(aes(trip_duration, fill = tgroup)) +
  geom_histogram(bins = 300) +
  scale_x_log10() +
  scale_y_sqrt()

train_group <- train_group %>%
  filter(tgroup != "mid")


p1 <- train_group %>%
  ggplot(aes(wday, fill = tgroup)) +
  geom_bar(position = "fill") +
  theme(legend.position = "none")


p2 <- train_group %>%
  ggplot(aes(month, fill = tgroup)) +
  geom_bar(position = "fill") +
  theme(legend.position = "none")


p3 <- train_group %>%
  ggplot(aes(hour, fill = tgroup)) +
  geom_bar(position = "fill")
p7 <- train_group %>%
```

```
  ggplot(aes(work, fill = tgroup)) +
  geom_bar(position = "fill") +
  theme(legend.position = "none")


layout <- matrix(c(1,1,2,2,3,3,3,3,4,5,6,7),3,4,byrow=TRUE)
multiplot(p1, p2,   p7, layout=layout)
p1 <- 1; p2 <- 1;   p7 <- 1

allu_train <- train_group %>%
  group_by(tgroup, work, wday) %>% # jfk_trip
  count() %>%
  ungroup


alluvial(allu_train %>% select(-n),
         freq=allu_train$n, border=NA,
         col=ifelse(allu_train$tgroup == "fast", "red", "blue"),
         cex=0.75,
         hide = allu_train$n < 150,
         ordering = list(
           order(allu_train$tgroup=="fast"),
       #    NULL,
           NULL,
           NULL))

foo <- combine %>%
```

```
  mutate(date = date(ymd_hms(pickup_datetime))) %>%

  group_by(date, dset) %>%

  count() %>%

  ungroup()

foo %>%

  ggplot(aes(date, n/1e3, color = dset)) +

  geom_line(size = 1.5) +

  labs(x = "", y = "Kilo trips per day")


pick_good <- combine %>%

  filter(pickup_longitude > -75 & pickup_longitude < -73) %>%

  filter(pickup_latitude > 40 & pickup_latitude < 42)

pick_good <- sample_n(pick_good, 5e3)



pick_good %>%

  ggplot(aes(pickup_longitude, pickup_latitude, color = dset)) +

  geom_point(size=0.1, alpha = 0.5) +

  coord_cartesian(xlim = c(-74.02,-73.77), ylim = c(40.63,40.84)) +

  facet_wrap(~ dset) +

  #guides(color = guide_legend(override.aes = list(alpha = 1, size = 4))) +

  theme(legend.position = "none")

attach(sub_train)
sample_model = glm(trip_duration ~
pickup_datetime:dropoff_datetime+pickup_latitude:pickup_longitude+dropoff_latitude:dropo
ff_longitude, family = poisson)

par(mfrow = c(2,3))
plot(sample_model,
```

```
    pch  = 20,
    cex  = 2,
    col  = "grey")
abline(sample_model, lwd = 3, col = "darkorange")
halfnorm(hatvalues(sample_model))

summary(sample_model)
#confint(sample_model, level = 0.99)
tidy(sample_model)
#augment(sample_model)
glance(sample_model)
cooks.distance(sample_model)[11] > 4 / length(cooks.distance(sample_model))
bptest(sample_model)
par(mfrow = c(2,3))
hist(resid(sample_model))

qqnorm(resid(sample_model), main = "Normal Q-Q Plot, sample_model", col = "darkgrey")
qqline(resid(sample_model), col = "dodgerblue", lwd = 2)

plot(log(fitted(sample_model)), log(resid(sample_model)), col = "grey", pch = 20,
    xlab = "Fitted", ylab = "Residuals", main = "Data from Model 1")
abline(h = 0, col = "darkorange", lwd = 2)

plot(which(hatvalues(sample_model) > 2 * mean(hatvalues(sample_model)), TRUE))

plot(rstandard(sample_model)[abs(rstandard(sample_model)) > 2])
cd_sample_model_add = cooks.distance(sample_model)
sum(cd_sample_model_add > 4 / length(cd_sample_model_add))


large_cd_train = cd_sample_model_add > 4 / length(cd_sample_model_add)
plot(cd_sample_model_add[large_cd_train])



coef(sample_model)
sample_model_add_fix = lm(trip_duration ~ dropoff_longitude,
                data = train,
                subset = cd_sample_model_add <= 4 / length(cd_sample_model_add))
coef(sample_model_add_fix)

#set.seed(42)
#shapiro.test(resid(sample_model))

boxcox(sample_model, plotit = TRUE)
attach(sub_train)
```

```r
sample_model2 = glm(trip_duration ~
pickup_datetime:dropoff_datetime+pickup_latitude:pickup_longitude+dropoff_latitude:dropo
ff_longitude+store_and_fwd_flag, family = gaussian)

par(mfrow = c(2,3))
plot(sample_model2,
    pch  = 20,
    cex  = 2,
    col  = "grey")
abline(sample_model2, lwd = 3, col = "darkorange")
halfnorm(hatvalues(sample_model2))

summary(sample_model2)
#confint(sample_model2, level = 0.99)
tidy(sample_model2)
#augment(sample_model)
glance(sample_model2)
cooks.distance(sample_model2)[11] > 4 / length(cooks.distance(sample_model2))
bptest(sample_model2)
par(mfrow = c(2,3))
hist(resid(sample_model2))

qqnorm(resid(sample_model2), main = "Normal Q-Q Plot, sample_model", col = "darkgrey")
qqline(resid(sample_model2), col = "dodgerblue", lwd = 2)

plot(log(fitted(sample_model2)), log(resid(sample_model2)), col = "grey", pch = 20,
    xlab = "Fitted", ylab = "Residuals", main = "Data from Model 2")
abline(h = 0, col = "darkorange", lwd = 2)

plot(which(hatvalues(sample_model2) > 2 * mean(hatvalues(sample_model2)), TRUE))

plot(rstandard(sample_model2)[abs(rstandard(sample_model2)) > 2])
cd_sample_model2_add = cooks.distance(sample_model2)
sum(cd_sample_model2_add > 4 / length(cd_sample_model2_add))


large_cd_train = cd_sample_model2_add > 4 / length(cd_sample_model2_add)
plot(cd_sample_model2_add[large_cd_train])



coef(sample_model2)
sample_model2_add_fix = lm(trip_duration ~ dropoff_longitude,
                data = train,
                subset = cd_sample_model2_add <= 4 / length(cd_sample_model2_add))
coef(sample_model2_add_fix)

#set.seed(42)
```

```
#shapiro.test(resid(sample_model))

boxcox(sample_model2, plotit = TRUE)
attach(sub_train)
sample_model3 = lm(trip_duration ~
pickup_datetime:dropoff_datetime+pickup_latitude:pickup_longitude+dropoff_latitude:dropo
ff_longitude+store_and_fwd_flag+vendor_id, family = negative.binomial(1))

par(mfrow = c(2,3))
plot(sample_model3,
    pch  = 20,
    cex  = 2,
    col  = "grey")
abline(sample_model3, lwd = 3, col = "darkorange")
halfnorm(hatvalues(sample_model3))

summary(sample_model3)
#confint(sample_model3, level = 0.99)
tidy(sample_model3)
#augment(sample_model)
glance(sample_model3)
cooks.distance(sample_model3)[11] > 4 / length(cooks.distance(sample_model3))
bptest(sample_model3)
par(mfrow = c(2,3))
hist(resid(sample_model3))

qqnorm(resid(sample_model3), main = "Normal Q-Q Plot, sample_model", col = "darkgrey")
qqline(resid(sample_model3), col = "dodgerblue", lwd = 2)

plot(log(fitted(sample_model3)), log(resid(sample_model3)), col = "grey", pch = 20,
    xlab = "Fitted", ylab = "Residuals", main = "Data from Model 3")
abline(h = 0, col = "darkorange", lwd = 2)

plot(which(hatvalues(sample_model3) > 2 * mean(hatvalues(sample_model3)), TRUE))

plot(rstandard(sample_model3)[abs(rstandard(sample_model3)) > 2])
cd_sample_model3_add = cooks.distance(sample_model3)
sum(cd_sample_model3_add > 4 / length(cd_sample_model3_add))


large_cd_train = cd_sample_model3_add > 4 / length(cd_sample_model3_add)
plot(cd_sample_model3_add[large_cd_train])



coef(sample_model3)
sample_model3_add_fix = lm(trip_duration ~ dropoff_longitude,
                data = train,
```

```
                subset = cd_sample_model3_add <= 4 / length(cd_sample_model3_add))
coef(sample_model3_add_fix)

#set.seed(42)
#shapiro.test(resid(sample_model))

boxcox(sample_model3, plotit = TRUE)
m1AIC <- AIC(sample_model)
m1BIC <- BIC(sample_model)
m2AIC <- AIC(sample_model2)
m2BIC <- BIC(sample_model2)
m3AIC <- AIC(sample_model3)
m3BIC <- BIC(sample_model3)


AIC <- list(m1AIC, m2AIC, m3AIC)
BIC <- list(m1BIC, m2BIC, m3BIC)
kable(rbind(AIC, BIC), col.names = c("Model 1", "Model 2", "Model 3"))  %>%
  kable_styling(full_width = T)
eval_p<-predict(sample_model3,sub_train, type = "response")
write.csv(eval_p,"predicted_eval_values.csv")
par(mfrow = c(1,3))
train_p<-predict(sample_model,sub_train, type = "response")
plot(train_p,sub_train$trip_duration,main = "Model 1")
train_p<-predict(sample_model2,sub_train, type = "response")
plot(train_p,sub_train$trip_duration,main = "Model 2")
train_p<-predict(sample_model3,sub_train, type = "response")

plot(train_p,sub_train$trip_duration,main = "Model 3")
```