NYC TAXI DATA ANALYSIS

SHOULD CARPOOL EXPAND BEYOND MANHATTAN?

Using Public NYC Taxi Data [1]

Tomer Eldor

[1] Data used: “Yellow” taxis data from June 2016, but was cut down into workable sizes. Data was supplemented by querying google using a script for addresses and neighborhoods.

[1] * Discloser: this analysis is smaller and shorter than the full analysis deserved for this project. I show here a partial, simpler version of analyses, but also mention potential better ways I would go about it if I had more time to invest. The data is also very small and not well representative, bring limited in machine power and time. I will lay out my intended analysis along with my actual analysis.

A new carpooling service, for here referred as carpool, is considering expanding its service area from Manhattan to a new neighborhood in Queens, here won't be explicitly named. They are trying to find the optimal way to launch this expansion.[2]

EFFICIENCY OF AGGREGATING RIDES

We inspect the aggregation of rides within various start and end points. [3]
I retrieved address and neighborhood data about 5000 observations, 4900 of which were complete (no NAs). All with pickup origin centered around Queens and Upper East Side, with some around the boundaries of the neighborhoods, by a rough cut of coordinates, which yielded 630 pickups from Queens, and 2014 from Upper East Side. How does the efficiency compare?

Within Queens – *seemingly low efficiency.*

Out of a sample of 4900 pickups from Queens and some and neighboring boroughs (in Queens and Upper East Sides), only 181 were from Queens to Queens – 28%. This shows that there is not a big demand relatively for this kind of journey. **Out of these 181 rides, only 17 of them were within 10 minutes and 800 meters at pickup, which could be aggregated into 6 Carpools of 4 people each.** However, this analysis did not account for the specific

---

[3] **The originally intended method of analysis**:

- Data Pre-Processing: add neighborhoods names, make dates workable, etc.
- One possible method would be **clustering** of space-time; but this had limitations here – see bottom.
- However, in here I intended to analyze using suboptimal simplifying assumptions: "chunking" times into 10 min chunks (by rounding them), and considering observations which are less than ~800m as close enough. I divided the data into time groups and for each of them, checked for each rider if it has a matching close by rider in the same group.
- Intended measures of efficiency:
    - For Carpool: For every 100 rides, how many Carpools would we need to satisfy the demand?
    - For Carpool: What fraction of rides are close enough in time and distance which are "aggregable"?
    - Global: How many CARPOOLs can satisfy the demand, versus how many taxis currently satisfy it?

[2] limitations from performing **clustering** in this case:
1. The data shouldn't be really in clusters; should be uniformly randomly distributed
2. Clustering usually takes the closest clusters, but I prefer to have realistic bounds for maximum allowed distance of time and location which I can interpret and be certain. It's less important for this matter to find the actual best fits; but general groups of possible matches.
3. A distance function would be highly manually sensitive to the function I choose, and since I don't know much about this subject matter, I'm likely to not compose an optimal distance function. It would also be less interpretable; I would prefer limiting to maximum allowed time or distance. Clustering would be less interpretable for this purpose.
4. Mostly, time limit - I didn't have time to learn, implement and run such long clustering procedure (which by comparing every point to another is at least O(n^2)).

direction of the pickups (which we should ensure the next pickups are within the direction of the first ones), and the increased sensitivity to time and location in short *same-neighborhood type* rides. Therefore, if anything, this estimation is an overestimation.
Average and median travel distance within Queens is only 0.8 miles, while trips from Queens to Manhattan are on average 4.3 miles, so is the median (4.35). Thus, trips from Queens to Manhattan are much longer and thus less sensitive to the initial distances between pickups, since they will be sharing much more of the distance together, and will be "saving" more money by sharing since the trips are more expensive: $9.4 in average for within Queens, versus twice as much(!) $19 in average for Queens to Manhattan.
changes when it is *within* Queens, since the travel distance is shorter and therefore the expectations of "closeness" of riders are higher. Meaning, since users are more sensitive to pickups which are a bit further away or later, we would need to have higher demand of close by and timely rides; but it seems to not be the case now.
In sum, Queens is not yet quite as busy as other neighborhoods in Manhattan, so there is not necessarily yet sufficient demand so that there would be many nearby timely trips *within Queens.*

From Queens to Manhattan – *seemingly low efficiency but slightly higher than Within Queens.*

This same sample (4900 pickups from Queens and neighbors near its boundaries) and process shows only 159 rides are from Queens to Manhattan (26%); and from them, only **26 rides** where "aggregable" – within 10 minutes and 800 meters.
However, here people travel much further and more expensively (double as said) to Manhattan and other areas. Therefore, the passengers would seem to have better efficiency, in terms of shared longer routes (length of routes relative to the needed waiting time or initial pickups distance). Thus, there would be higher conversion and satisfaction, which will make this market more relevant.

How does this compare to Carpool's current area of service (e.g. the Upper East Side)?

Within Upper East were slightly more popular than "within Queens": 641 out of 2014 : 32% of rides from Upper East, but definitely the rides outside of it were twice the amount. I would conduct the same analysis and compare, but could not finish in time. Intuitively, it seems as though since Manhattan is a much busier and richer city, it should have higher demand for Carpool and therefore would be more efficiently aggregated into rides and benefitted from Carpool.

Should Carpool provide rides only within Queens for this new service or also between Queens and Manhattan?

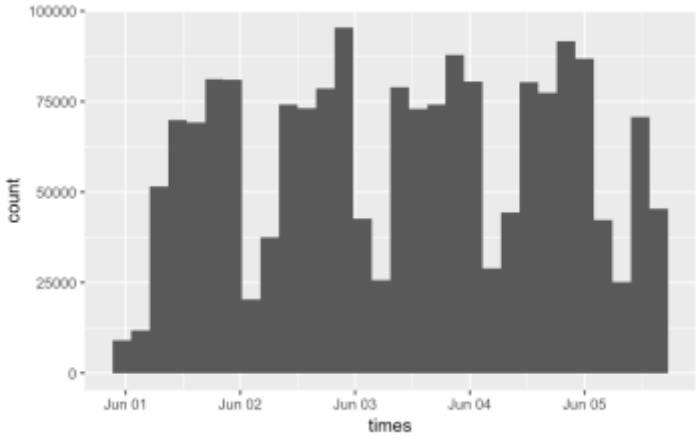
Although this analysis is limited, coupled with the theoretical analysis, **I would not expand to Queens just yet**. It seems from the data that **there is not enough demand t**here for Carpool to be efficient.

2

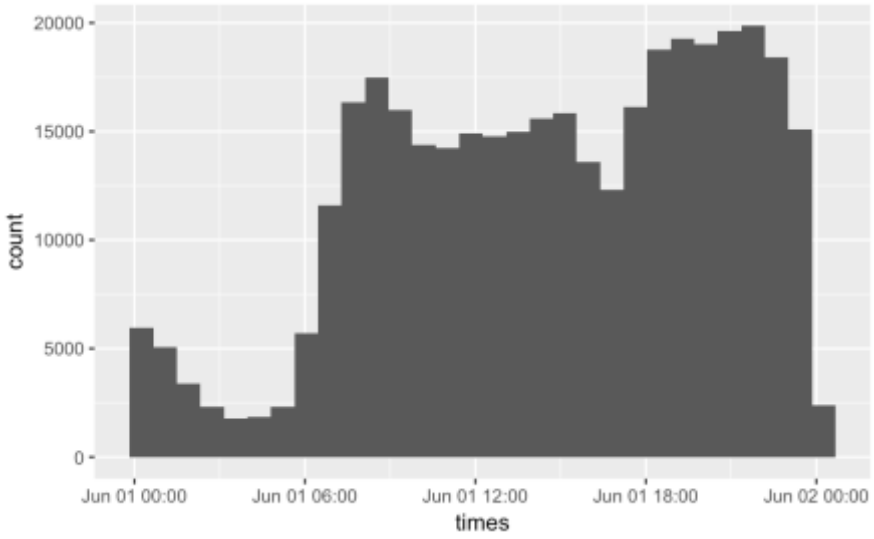
Between these two options, Cross-City would be a better choice. There is probably a lot more **shared** distance traveled between Queens and NYC, therefore it should be more efficient; more passengers would drive in the same direction and would have less relative variance in their route when divided by the total distance of the route; therefore, more passengers are likely to want it.

Should Carpool launch Queens service during all our hours of operation (24/7) or only for certain hours? → *Only at PEAK HOURS.*

Since this analysis does not show high enough demand, we probably should only consider launching this at peak hours**.** **Peak hours** are usually morning, afternoon, and sometimes weekend. We see a pattern of peaks during evening times, and falls during night times.



If we zoon into one day to see at which hours this occurs we see that there is a first jump in taxi trips in the morning around 7-8 (rush time), then a slight fall, and a bigger increase in the evening after 18:00, and stays strong between about 18:00-00:00. And Yes, this *is a (normal) Wednesday.* Additionally, we can assume that morning and evening people go to work. Their work routes are regular. Thus, we can capitalize on that, especially if we can achieve their home/work data from their phone, and send Carpools in the morning (around 7-8) and all evening from 18:00-00:00.



What additional data would you like to see in order to answer questions 1-5 more confidently and how would you incorporate it?

I would like to see also:

1. In-route location, or know how to calculate in-path proximity.
2. Carpool Data, such as pointing towards profitability. In order to make wise business decisions, the external conditions are not the biggest factor; other internal problems and costs might damage the sales/growth/profitability. We probably care not only about the external market; but we would also like to know the costs associated to know how would opening a new branch, diverging drivers from one area to another, and driving longer distances affect the profitability.
3. Other Markets: Why should we assume that taxi users are the only market segment we target? I believe that the following are more useful new market segments
4. Uber & Lyft Data – These are the users which are our users as well; and we should see what is the market for them now.
5. . Not as much taxis
6. Demographic Data – average income, distributions, age, students / workers, where they work, of people in Queens, and how does it compare to different neighborhoods we operate within, and thus compare how this demographic is likely to take Carpool. Of course, we can't truly simulate this herd human emotional behavior.
7. Phone trip data – we can know the home/work bases for people using phone data, and even their accurate trips every day. With that, also Public Transport Data would help to know better the traffic of people in general in the city; and also depending on the demographics (like graduating students) are likelier to start using Carpool than public transit as they get a job and can afford a little more.

How might your answer change over time? What Carpool data would you monitor to ensure the proposed expansion was a good business decision?

It comes down to:

a. **Profitability** of route – as mentioned, important to continuously measure if ti sis still a good business decision.
b. **Growth in profitability and Growth in customers** - see if it was substantial and if it was slightly more than just the first month were everyone came.
c. What is the ratio between demand and supply (how many Carpools are we sending, how full).
d. I would perform a simulation of Carpool after the expansion; although human behavior is unpredictable, in large groups it is slightly more expected. Alternatively, after it has already made, I would do a synthetic control with and without expanding to Queens; thus, measure what is the effect of adding Queens to Carpool's efficiency?

APPENDIX: R CODE

````
---
title: 'Carpool: NYC Taxi Data'
output:
  word_document: default
  html_notebook: default
---

This is the raw working notebook: it has also lines or chunks are code that didn't work for the final analysis,
and some which did but these are where I worked from, changed, and replaced underlying data analyzed, so they are
not always duplicated here for every data used for analysis.
Data downloaded as "Yellow" dataset of June month from NYC TAXI DATA. (Later I was trying to get shapes
information about the neighborhoods as you may see but it didn't work).


Import Data

```{r}
#install.packages("rgeos")
#install.packages("maptools")
library(dplyr)
library(ggmap)
library(rgeos)
library(maptools)
library(sp)
library(rgdal)


setwd("~/Google Drive/Academics (Tomer)/0-2017 Buenos Aires Sem2.2/CS112 DataScience/R_CS112/Carpool")


##### starting with full dataset

#taking some million entries from trip dataset
df <- read.csv("yellow_tripdata_2016-06.csv",nrows = 1000000)
dfull <- read.csv("yellow_tripdata_2016-06.csv",nrows = 2000000)
nrow(df)
#these are too much for my machine to work with efficiently for this draft as a trial.

summary(df$tpep_pickup_datetime)
dfull <- na.omit(dfull) #and omit NAs.
nrow(dfull)

#Visaully explore the data
head(df)
View(df)

#export as shortened CSV
write.csv2(dfull, file="2M NYCtaxi trips yl")

```




Analysis questions:
1.	How would you assess the efficiency of aggregating rides within Queens?
I assume we measure efficiency in terms of:
i.	How many trips would be saved if everyone who take in the same timespan uses Carpool?
  for that, I would measure:
   - How many similar rides within 10 min order. Here I make a computational facilitation assumption: I chunk the
time per hour equally be each 15 minutes, as in 1:00-1:15, 1:15-1:30, etc. This accurately conveys the
"similarity" of time best for the "middle range"
   - How many people there is in each taxi
   - keep a counter of the individual rides and their:
        # of passengers, fare, and distance traveled
        sum up all the similar rides into groups of (6)
   - We'd also like to know


````

```
and potentially later we can also estimate:
ii.     Money – how much they would save by splitting.
iii.    (Gas – how much gas would be saved)

Later, perform the same analysis to
 - From Queens to Manhattan?
3.      From LaGuardia airport, through Queens, to Manhattan
4.      and back?



#group by pickup and dropoff location
### Queens = loc_ids 7,8,179 according to loc_id key from nyctaxi fhv
### Queens is around 40.774444, -73.904167



##Trial Code
```{r}
library(dplyr)
#install.packages("ggmap")
library(ggmap)
#?lubridate

## trying out ggmap library : 1. example
# generate a single example address
lonlat_sample <- as.numeric(geocode("the hollyood bowl"))
  # note the order is longitude, latitiude
res <- revgeocode(lonlat_sample, output="more")
# can then access zip and neighborhood where populated
res$postal_code
res$neighborhood


##trying out ggmap library: 2. ours
names(df)
lonlat1 <- c(df[1:2,]$pickup_longitude, df[1:2,]$pickup_latitude)
lonlat1
typeof(lonlat_sample) ; typeof(lonlat1)
lonlat1adr <- revgeocode(lonlat1, output="more")
lonlat1adr$neighborhood

geocodeQueryCheck()

#adding a column of neighborhoods

pickup_adrs <- (NULL)
pickup_nbgs <- (NULL)
for(i in 1:nrow(df)) {     #nrow(df)
    row <- df[i,]
    print(row)
    rowcoor <- c(row$pickup_longitude, row$pickup_latitude)
    print(rowcoor)
    pickup_adr <- revgeocode(rowcoor,output="more")
    print(pickup_adr)
    append(pickup_adrs, pickup_adr$address)
    append(pickup_nbgs, pickup_adr$neighborhood)
}

#pickup_adr <- revgeocode(c(df$pickup_longitude, df$pickup_latitude), output="more")

### try
mapping1 <- df[1:2, Address <- mapply(function(long, lat)  {
  print(revgeocode(c(long, lat),output="more"))),
  (df$pickup_longitud, df$pickup_latitude))
  } ]
```
```

```
### try
df$
neighborhood <- lapply(seq(1), function(i) {  #nrow(df)
               revgeocode(
                 as.numeric(c(df[i,6:7])),
               output = "more"
               )$neighborhood
               })


### try
result <- do.call(rbind,
   lapply(1:1, #nrow(data)
    function(i)(revgeocode(as.numeric(df[i,6:7]),output = "more")$neighborhood)))

#can't work for more than 2500 datum; don't know why but not working


#dfQueens =  read.csv("yellow_tripdata_2016-06.csv",nrows = 500000)
plot(dfQueens$pickup_longitude)
summary(dfQueens$pickup_latitude)
lines(dfQueens$pickup_longitude))
dfQueens = df [
         between(df$pickup_longitud,-73.948468,-73.904952) &
         between(df$pickup_latitude,40.760050,40.790000), ]

nrow(dfQueens)


```
approximate radiuses:
Queens:
center:
Latitude:40.766054°
Longitude:-73.923491°

approx boundaries:
around:
Lat: 40.751036, 40.782369
Long: -73.904952, -73.948468

Latitude:40.764884°
Longitude:-73.923148°

Latitude:40.767745°
Longitude:-73.904952°

Latitude:40.751036°
Longitude:-73.937224°

Latitude:40.755588°
Longitude:-73.948468°

Latitude:40.77132°
Longitude:-73.933533°

Latitude:40.782369°
Longitude:-73.919715°

Lat: 40.751036, 40,782369
Long: -73.904952, 73.948468

##FINAL CODE: Add Neighborhood Data
```
```

```{r}

#build df with mainly Queens pickups

#dfQueens =  read.csv("yellow_tripdata_2016-06.csv",nrows = 500000)
Queenstrips = df[
         between(df$pickup_longitud,-73.948468,-73.904952) &
         between(df$pickup_latitude,40.753048,40.790000), ]

coordinates(Queenstrips[,6:7])

#add neighborhood
neighborhood_5000_5100 <- lapply(seq(5000,5100), function(i){revgeocode(as.numeric(c(dfQueens[i,6:7])),output =
"more")$neighborhood})
#neighborhood1_100 <- neighborhood
View(neighborhood1_100) <- matrix(neighborhood1_100)
#cbind.data.frame(dfQueens,neighborhood)
#dfQueens_nb = merge(dfQueens, neighborhood1_100, join='left')
head(dfQueens_nb)
#neighborhood not working anymore because reached max google querying
#typeof(neighborhood[1])


#flatten neighborhood list into a vector
neighborhood_flat_101_2031_2 <- neighborhood_101_2031
neighborhood_flat_101_2031_3<-unlist(neighborhood_flat_101_2031_2)
neighborhood_flat_101_2031[which(c(1,diff(neighborhood_flat_101_2031)) != 0)]




```


#create approximate shapes of boroughs
get a map
draw on top of the map the crude places
create dummy variables - was in either location in pickup/dropoff


```{r}

#create original database csv
summary(dfQueens)
names(dfQueens[,20])
dfQueens_old <- dfQueens
head(dfQueens[,1:19])
nrow(dfQueens)
dfQueens <-  dfQueens[,1:19]
dfQueens_2501_5000 = dfQueens[2501:5000,]
dfQueens_5001_7500 = dfQueens[5001:7500,]
dfQueens_7501_10000 = dfQueens[7501:10000,]


write.csv(dfQueens_7501_10000, file="Queens_7501_10000.csv")
write.csv(dfQueens[12501:15138,], file="Queens_12501_15000.csv")

#read.csv()


```


#Trying Carpool ShapeFiles
```{r}
```

```
library(ggmap)
library(rgeos)
library(maptools)
#used tutorial guide from here: https://www.r-bloggers.com/r-and-gis-working-with-shapefiles/

crswgs84 = CRS("+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs")
nbhoods = readShapePoly("./ZillowNeighborhoods-NY/ZillowNeighborhoods-NY.shp",proj4string=crswgs84,verbose=TRUE)

#explore data
class(nbhoods)
str(nbhoods@data)
str(nbhoods@polygons[[1]])

#spatP <- SpatialPoints(gmap("Palo Alto, CA", lonlat = TRUE),
    #proj4string = CRS("+proj=longlat"))

point <- dfQueens[1,6:7]
pointSp <- SpatialPoints(point, proj4string = CRS("+proj=longlat"))

polyHull <- gConvexHull(nbhoods)
gContains(polyHull, nbhoods)

gContains(nbhoods, pointSp, byid = FALSE)


p<-SpatialPoints(list(dfQueens[15,6:7]), proj4string=crswgs84)

gContains(nbhoods,p)

## Error in RGEOSBinPredFunc(spgeom1, spgeom2, byid, func) : TopologyException: side location conflict at -
78.858643000000001 42.958652999999998
```

eventually stuck with this error: Error in RGEOSBinPredFunc(spgeom1, spgeom2, byid, func) : TopologyException:
side location conflict at -78.858643000000001 42.958652999999998
Didn't found proper solution online in time.
Swtiched back to querying google.

```{r}


getinfo.shape("./ZillowNeighborhoods-NY/ZillowNeighborhoods-NY.shp")
#Shapefile type: Polygon, (5), # of Shapes: 579
map <- readShapeSpatial("./ZillowNeighborhoods-NY/ZillowNeighborhoods-NY.shp")
class(map)
head(map@data)
nymap <- map[map$City=="New York",]
#plot(nymap)
#nymap$County
shp <- readShapePoly("./ZillowNeighborhoods-NY/ZillowNeighborhoods-NY.shp")

plot(shp)

dfQueens[1:3,6:7]

#over
```


#try using ShapeFiles : check if points within polygon
```{r}

library(ggmap)
library(rgeos)
library(maptools)
library(rgeos)
```

```
library(sp)
library(rgdal)


#data aquired from here: #http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml
#took the 2016 June      Yellow dataset
#read only first 1 million rows


##### starting with full dataset

#taking first million entries from trip dataset
df <- read.csv("yellow_tripdata_2016-06.csv",nrows = 1000000)

# From this, subsetting the data, taking only datapoints AROUND Queens, then filtering for actually Queens - using
the approximate coordinates ( manually explored on the boundaries of Queens on map)
Queenstrips = df[
         between(df$pickup_longitud,-73.948468,-73.904952) &
         between(df$pickup_latitude,40.753048,40.790000), ]



## getting new york map shapes
ny.map <- readOGR("./ZillowNeighborhoods-NY/ZillowNeighborhoods-NY.shp", layer="ZillowNeighborhoods-NY")
#coordinates(dfQueens[,6:7]) <- c(dfQueens[,6:7])
ny.map$City
#combine is.na() with over() to do the containment test; note that we
# Tell R that Queens coordinates are in the same lat/lon reference system
# as the parks data --  ONLY BECAUSE WE KNOW THIS IS THE CASE!
Queens <- ny.map[ny.map$CITY == "New York"  & ny.map$NAME == "Queens", ]

head(Queenstrips)

# Set the projection of the SpatialPointsDataFrame using the projection of the shapefile

proj4string(Queenstrips) <- proj4string(Queens)

####maybe this???
Queens2 <- spTransform(Queens, CRS("+proj=longlat +datum=WGS84"))
#coordinates(Queenstrips)

in.Queens <- !is.na(over(Queenstrips, as(Queens,"SpatialPolygons")))

# what fraction of sightings were inside a park?
mean(in.Queens)

SP <- SpatialPolygons(list(Polygons(list(Queens), ID="poly")))

Queens.spatial <- as(Queens,"SpatialPolygons")

over(coordinates(dfQueens), as(Queens,"SpatialPolygons"))
over(Queens.spatial, coordinates(dfQueens[,6:7]))



````


##################################
initial tries = unsuccessful
```{r}
```

```
min(mappedtrips$pkp_time)
max(mappedtrips$pkp_time)

### creating time chunks table and seeing the frequency.
pkp_time <- as.Date(pkp_time)
timechunks15 <- (table(cut(pkp_time, breaks = "15 mins")))
summary(timechunks15)
intervals <- as.data.frame(timechunks15)
mappedtrips$timechunks15 <- timechunks15



### Sudocode ####
#intervals <- table(cut(pkp_time, breaks = "15 mins"))
#create a new column call it:
#for i in range(1:end of your dataset:
#                 group_name = intervals[i]
#               mappedtrips[intervals[i]  < mappedtrips$tpep_pickup_datetime < #intervals[i+1]]$newcolumn <-
group_name
#####

mappedtrips$timechunk <- rep(0,nrow(mappedtrips))
#checking
#length(mappedtrips$timechunk)
#nrow((mappedtrips))
#between(mappedtrips$tpep_pickup_datetime, intervals$Var1[1], intervals$Var1[1+1])
#mappedtrips$tpep_pickup_datetime[17]
#intervals$Var1[1]
#intervals$Var1[1+1]

"""
(as.POSIXct(intervals$Var1[i]))
mdy_hms(mappedtrips$tpep_pickup_datetime)
class(mdy_hms(mappedtrips$tpep_pickup_datetime))

for(i in 1:nrow(intervals)) {
  within_i <- between(
    mdy_hms(mappedtrips$tpep_pickup_datetime),
    as.POSIXct(intervals$Var1[i]),
    as.POSIXct(intervals$Var1[i+1]))
  mappedtrips$timechunk[c(within_i)] <-  as.POSIXct(intervals$Var1[i]) }
"""
```


CHUNKING by TIME Groups

```{r}
#libraaries
library(lubridate)

#read csvs which have been added the addresses and neighborhoods (to filter neighborhoods; done using google maps
scripts API seperately) , and unite them into one dataframe
mappedtrips <- read.csv("dfQueens_mapped_50-75_10-125.csv")
nrow(mappedtrips)
mappedtrips <- na.omit(mappedtrips)
nrow(mappedtrips)


#parse date column into actual date instead of "factors"
#class(mappedtrips$tpep_pickup_datetime) #checking what type is it
pkp_time <- (mdy_hms(mappedtrips$tpep_pickup_datetime))

#instert into dataframe
mappedtrips$pkp_time <- pkp_time
```

```
########### CHUNKING #############
#Disclaimer: I know this would not be the optimal way to go about this. Here I "chunk" calls by discrete 10 min
intervals on the clock; but this means I might miss matching two rides of 18:04 and 18:06 while matching rides of
18:06 and 18:14. This is not ideal. However, it is still within 10 minutes which is an acceptable waiting time for
users. A better way to do this might be using CLUSTERING, while setting concrete boundaries of maximum allowed
distance between paramteres of location and time. However, I didn't have the time to do this (or probably for the
machine to complete clustering of all the data). Therefore for simplicity I continue here with this simplified
model of chunking into time chunks.



#for each pickup time, round it to the closest round 10 min mark, and save this rounded version into another
column for the same observation, signifying which "10 min" group it belongs to. Later we will match rides within
the same 10min group.

mappedtrips$timechunk15 <- rep(0,nrow(mappedtrips)) #initializing empty column
#function
as.POSIXlt(mappedtrips$tpep_pickup_datetime)
for(i in 1:length(mappedtrips$tpep_pickup_datetime)) {
  curr_row <- as.POSIXlt(mappedtrips$tpep_pickup_datetime)[i]
  rounded_curr_row <- as.POSIXlt( round(as.double(curr_row)/(15*60))*(15*60) , tz="GMT", origin=(as.POSIXlt('1970-
01-01',tz="GMT")))
  mappedtrips$timechunk[i] <- format(rounded_curr_row,"%H:%M")
}

write.csv(mappedtrips,"mappedtrips.csv")


```


Aggregating Rides From Queens to Queens and Manhatten

Subsetting Dataset to pickup neighborhood
```{r}

mappedtrips <- read.csv("mappedtrips.csv")

unique(mappedtrips.complete$Pkp_Neighborhood)
mappedtrips.complete <- mappedtrips[mappedtrips$Pkp_Neighborhood != "??" ,]
length(mappedtrips.complete$Pkp_Neighborhood)
mappedtrips <- mappedtrips.complete

#from Queens at all
from.Queens <- mappedtrips[mappedtrips$Pkp_Neighborhood=="Queens",]
nrow(from.Queens)
#focus on Queens to Queens only
Queens2Queens <- mappedtrips[mappedtrips$Pkp_Neighborhood=="Queens" & mappedtrips$Dpf_Neighborhood=="Queens",]
nrow(Queens2Queens)


# Queens to Manhattan
Manhattan_neighborhoods = c("Upper East Side","Upper West Side","Midtown","Sutton Place","Yorkville","East
Harlem","Midtown East","Lower Manhattan","Central Park","Upper Manhattan","Lincoln Square","Midtown
West","Carnegie Hill","Inwood","Lenox Hill","Hudson Heights","Hell's Kitchen","East Village","Lower East
Side","Gramercy Park")

Queens2mnh <- mappedtrips[mappedtrips$Pkp_Neighborhood=="Queens" &  is.element(mappedtrips$Dpf_Neighborhood,
Manhattan_neighborhoods) , ]
nrow(Queens2mnh)
#View(Queens2mnh)

#uppereast -
```

```
uppereast <- mappedtrips[mappedtrips$Pkp_Neighborhood=="Upper East Side" , ]
nrow(uppereast)


#uppereast2uppereast
uppereast2uppereast <- mappedtrips[mappedtrips$Pkp_Neighborhood=="Upper East Side" &
mappedtrips$Dpf_Neighborhood=="Upper East Side" , ]
nrow(uppereast2uppereast)

mappedtrips[mappedtrips$Pkp_Neighborhood=="Upper East Side" &  mappedtrips$Dpf_Neighborhood=="Upper East Side" , ]


quantile(Queens2Queens$trip_distance)
mean(Queens2Queens$trip_distance)

quantile(Queens2mnh$trip_distance)
mean(Queens2mnh$trip_distance)


mean(Queens2mnh$total_amount)
mean(Queens2Queens$total_amount)


### checking if I have La Guardia info
#unique(mappedtrips$Dpf_Neighborhood)
is.element("%*Guardia%", mappedtrips$Pickup_Address)
which(grepl("Guardia", mappedtrips$Pickup_Address) == TRUE)
#no i don't have within the mapped one.
```

```
Length of our sample is 4988.

Within Queens: 181
mean trip distance:  0.828674
Quantiles trip distance
  0%  25%  50%  75% 100%
0.00 0.46 0.80 1.10 5.60


Queens to Manhattan: 159
mean trip distance: 4.290881
Quantiles trip distance
 0%   25%   50%   75%  100%
 0.38  2.93  4.35  5.40 10.86


uppereast2uppereast: 641

###
1. Check for EACH Pickup,
    1. how many nearby trips where there;
    2. how many _people_ were in these trips in total,
    3. in how many cars
    4. how many Carpools could have taken them (divided by 4? 6?)


### Analyzing Queens to Queens
```{r}
library(geosphere)
library(rgeos)
### chunking by time

uniquetimes <- unique(mappedtrips$timechunk)
length(uniquetimes)
```
```

```
Queens2Queens_uniquetimes <- unique(Queens2Queens$timechunk)
#chunking  time
Queens2Queens_time_2 <- Queens2Queens[Queens2Queens$timechunk == Queens2Queens_uniquetimes[2] , ]

#for i in curr time chunk

#find how many trips nearby pickup

# how many of them have not TOO far dropoff


#### chunking groups into close trips
Queens2Queens_uniquetimes <- unique(Queens2Queens$timechunk)
Queens2Queens$matched <-  rep(0,nrow(Queens2Queens))

for(time in Queens2Queens_uniquetimes[1:5]) {
  #iterating through each time chunk
  print(time)
  #checking within the trips of the same rounded time chunk
  timely_trips <- Queens2Queens[Queens2Queens$timechunk == time , ]
  print(timely_trips)

  #calculate for each unit in time chunk how many nearby pickups with reasonable dropoffs they have
  #for time chunks with exitsing other trips within that time chunk:
  if(length(timely_trips)>1)
  #now do pair-wise comparison within the compatible trips within neighborhood and timechunk (only a handful of
trips mostly, if any!
    for(i in seq(1:length(timely_trips)) ) {
      for(j in seq(2:length(timely_trips)) ) {
        pickup_i <- c(timely_trips$pickup_longitude[i], timely_trips$pickup_latitude[i])
        pickup_j <- c(timely_trips$pickup_longitude[j], timely_trips$pickup_latitude[j])
        distance_pickups = distHaversine(pickup_i, pickup_j)
        if(length(distance_pickups) > 1) append(Queens2Queens$matched[i], c(Queens2Queens$tpep_pickup_datetime,
Queens2Queens$ID) #Queens2Queens$pickup_latitude) #c(Queens2Queens$tpep_pickup_datetime,
Queens2Queens$pickup_longitude, Queens2Queens$pickup_latitude)

                                            )
      }
    }
  }
}




```

Analyzing Queens to Manhattan

```{r}
library(geosphere)
library(rgeos)
### chunking by time



Queens2mnh_uniquetimes <- unique(Queens2mnh$timechunk)
#chunking  time
Queens2mnh_time_2 <- Queens2mnh[Queens2mnh$timechunk == Queens2mnh_uniquetimes[2] , ]

#for i in curr time chunk

#find how many trips nearby pickup

# how many of them have not TOO far dropoff
```

```
#### chunking groups into close trips
Queens2mnh_uniquetimes <- unique(Queens2mnh$timechunk)
Queens2mnh$matched <-  rep(0,nrow(Queens2mnh))

for(time in Queens2mnh_uniquetimes[1:5]) {
  #iterating through each time chunk
  print(time)
  #checking within the trips of the same rounded time chunk
  timely_trips <- Queens2mnh[Queens2mnh$timechunk == time , ]
  print(timely_trips)

  #calculate for each unit in time chunk how many nearby pickups with reasonable dropoffs they have
  #for time chunks with exitsing other trips within that time chunk:
  if(length(timely_trips)>1)
  #now do pair-wise comparison within the compatible trips within neighborhood and timechunk (only a handful of
trips mostly, if any!
    for(i in seq(1:length(timely_trips)) ) {
      for(j in seq(2:length(timely_trips)) ) {
        pickup_i <- c(timely_trips$pickup_longitude[i], timely_trips$pickup_latitude[i])
        pickup_j <- c(timely_trips$pickup_longitude[j], timely_trips$pickup_latitude[j])
        distance_pickups = distHaversine(pickup_i, pickup_j)
        #checking also for dropoff in the same neighborhood
        if((length(distance_pickups) > 1) && (timely_trips$neighborhood[i] == timely_trips$neighborhood[j]))
append(Queens2mnh$matched[i], c(Queens2mnh$tpep_pickup_datetime, Queens2mnh$ID)
                                        )
      }
    }
}

Queens2mnh$matched
```


Analyzing Upper East Side to Upper East Side

```{r}
library(geosphere)
library(rgeos)
### chunking by time

uppereast2uppereast_uniquetimes <- unique(uppereast2uppereast$timechunk)
#chunking  time
uppereast2uppereast_time_2 <- uppereast2uppereast[uppereast2uppereast$timechunk ==
uppereast2uppereast_uniquetimes[2] , ]

#for i in curr time chunk

#find how many trips nearby pickup

# how many of them have not TOO far dropoff


#### chunking groups into close trips
uppereast2uppereast_uniquetimes <- unique(uppereast2uppereast$timechunk)
uppereast2uppereast$matched <-  rep(0,nrow(uppereast2uppereast))

for(time in uppereast2uppereast_uniquetimes[1:5]) {
  #iterating through each time chunk
  print(time)
  #checking within the trips of the same rounded time chunk
  timely_trips <- uppereast2uppereast[uppereast2uppereast$timechunk == time , ]
  print(timely_trips)

  #calculate for each unit in time chunk how many nearby pickups with reasonable dropoffs they have
  #for time chunks with exitsing other trips within that time chunk:
```

```
  if(length(timely_trips)>1)
  #now do pair-wise comparison within the compatible trips within neighborhood and timechunk (only a handful of
trips mostly, if any!
    for(i in seq(1:length(timely_trips)) ) {
      for(j in seq(2:length(timely_trips)) ) {
        pickup_i <- c(timely_trips$pickup_longitude[i], timely_trips$pickup_latitude[i])
        pickup_j <- c(timely_trips$pickup_longitude[j], timely_trips$pickup_latitude[j])
        distance_pickups = distHaversine(pickup_i, pickup_j)
        if(length(distance_pickups) > 1) append(uppereast2uppereast$matched[i],
c(uppereast2uppereast$tpep_pickup_datetime, uppereast2uppereast$ID) #uppereast2uppereast$pickup_latitude)
#c(uppereast2uppereast$tpep_pickup_datetime, uppereast2uppereast$pickup_longitude,
uppereast2uppereast$pickup_latitude)

                                          )
      }
    }
  }
}

uppereast2uppereast$matched

#checking out clusters just for fun?
library(cluster)
cluster <- cluster::agnes((uppereast2uppereast[6:7,]),diss=FALSE,metric = "manhattan",method="complete")
cluster
plot(cluster)
```

Generating distance matrix

```{r}
#generating distance matrix
dist_Queens2Queens <- distm(as.matrix(cbind(Queens2Queens$pickup_longitude, Queens2Queens$pickup_latitude)),
fun=distHaversine)

dist_Queens2mnh <- distm(as.matrix(cbind(Queens2mnh$pickup_longitude, Queens2mnh$pickup_latitude)),
fun=distHaversine)

distm(x,fun=distHaversine)
```

Time Graph

```{r}
library(ggplot2)
fulldf <- read.csv2("2M NYCtaxi trips yl.csv")

library(ggplot2)

df1m <- fulldf[1:1000000,]
times = as.POSIXct(fulldf$tpep_pickup_datetime[as.POSIXct( fulldf$tpep_pickup_datetime) < as.POSIXct("2016-06-
02")],tz="UTC")
qplot(times, geom="histogram")



```

Reverse Geocoding script used to get address/meighborhood out of Lang/Lan Carpool google.maps:

```
var geocoder = Maps.newGeocoder(),
    cache = CacheService.getScriptCache();
}

function reverseGeocodeLatLng(latitude, longitude) {
var cacheKey = latitude + "+" + longitude,
    cached = cache.get(cacheKey);
  if(cached !== null) {
    return JSON.parse(cached);
  }
  Utilities.sleep(2000);

  var result = geocoder.reverseGeocode(latitude, longitude),
  //First response is the most specific
      response = result.results[0];
  if(response === undefined) {
    return "??";
  }
  function g(fieldName) {
    for (var i = 0; i < response.address_components.length; i++) {
      if(response.address_components[i].types.indexOf(fieldName) != -1) {
        return response.address_components[i];
      }
    }
    return {"long_name": "??"};
  }
  rValue = [[g("route").long_name, g("street_number").long_name, g("postal_code").long_name,
g("neighborhood").long_name, g("locality").long_name, g('country').long_name]];
  cache.put(cacheKey, JSON.stringify(rValue));
  return rValue;
}
```