Andrew Goldstein
April 2012
Intro Data Analysis
Prof. Jay Emerson

New York City Subway Data Analysis

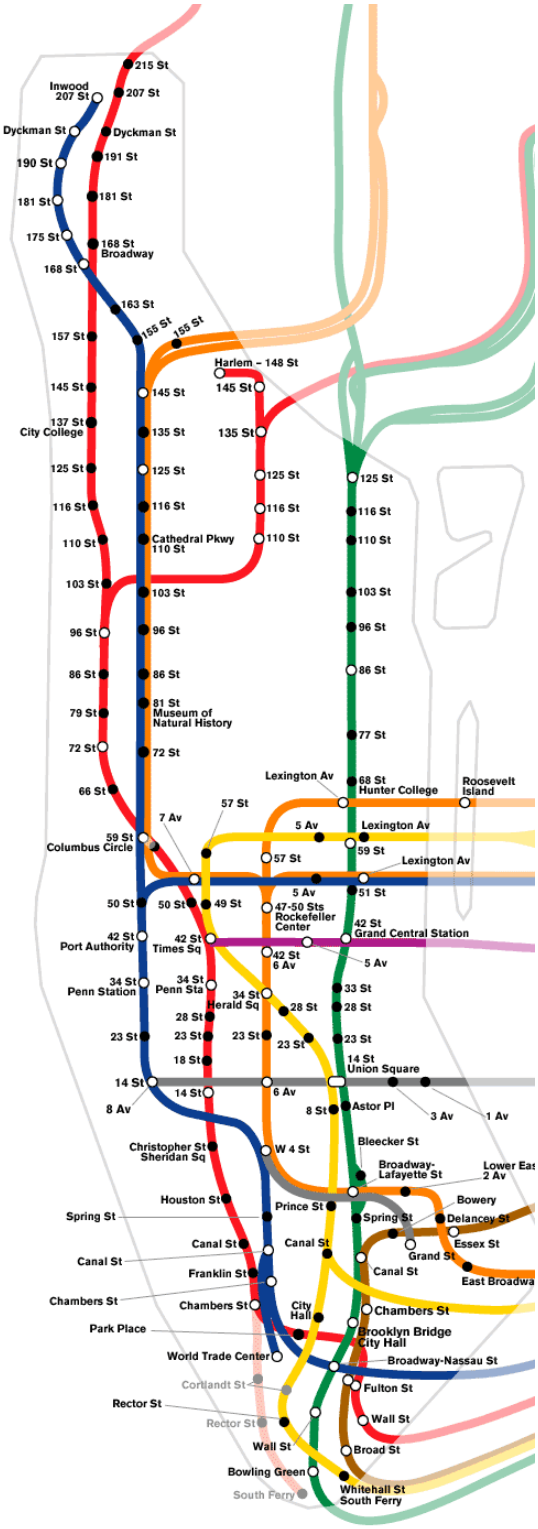
Abstract:

This project is an exploration of a large amount of New York City Subway data from the month of May 2011. I will specifically focus on Lines 1, 2, and 3 – the north/south artery of west Manhattan. The goals of this project are two-fold. First, I will examine the arrival and departure patterns of the subways and make some observations about what time of day tends to have the greatest number of late arrivals, and the extent to which a train arriving late impacts the train behind it. Second, I will create a "moving map" to provide a visual for some of the trends I describe.

Introduction:

The New York City subway is the seventh busiest metro system in the world, averaging over 5.5 million rides per weekday. A large portion of this passenger traffic occurs on Lines 1, 2, and 3 (shown in red on the map at right), one of the central transit arteries of Manhattan. Lines 2 and 3 are express trains, meaning that they only stop at white dots on the map, while Line 1 stops at all stations along its route.

Line 1 extends from the Bronx at 242nd St (off the top of the map) down to South Ferry at the southern tip of Manhattan. I will focus my investigation on the entirety of line 1 as well as the portions of line 2 and 3 which overlap line 1 (between 96th St and Chambers St).



Data Source:

Given that subway schedules differ between weekends and weekdays, I chose to limit myself to an analysis of travel during the 22 weekdays in May 2011. The New York City subway data set is publicly available in 31 separate Comma-Separated-Value (CSV) files at http://www.mta.info/developers/download.html. I also utilized the codebook and a companion file with station information for all stations in the NYC subway system.

Unfortunately, the raw data is not particularly clean and so I developed the following code to process the data from the web files into a usable form. This involved extracting the data from each of the 22 files in an efficient manner and then creating a new matrix that categorized the relevant information from the file (with respect to my project) into a ten column form. The first nine columns in matrix 'mat' are variations on the data in the original CSV file and the tenth column is a placeholder for an eventual indicator of whether a train is on time or late. I have included notations in the code to explain my data cleaning process:

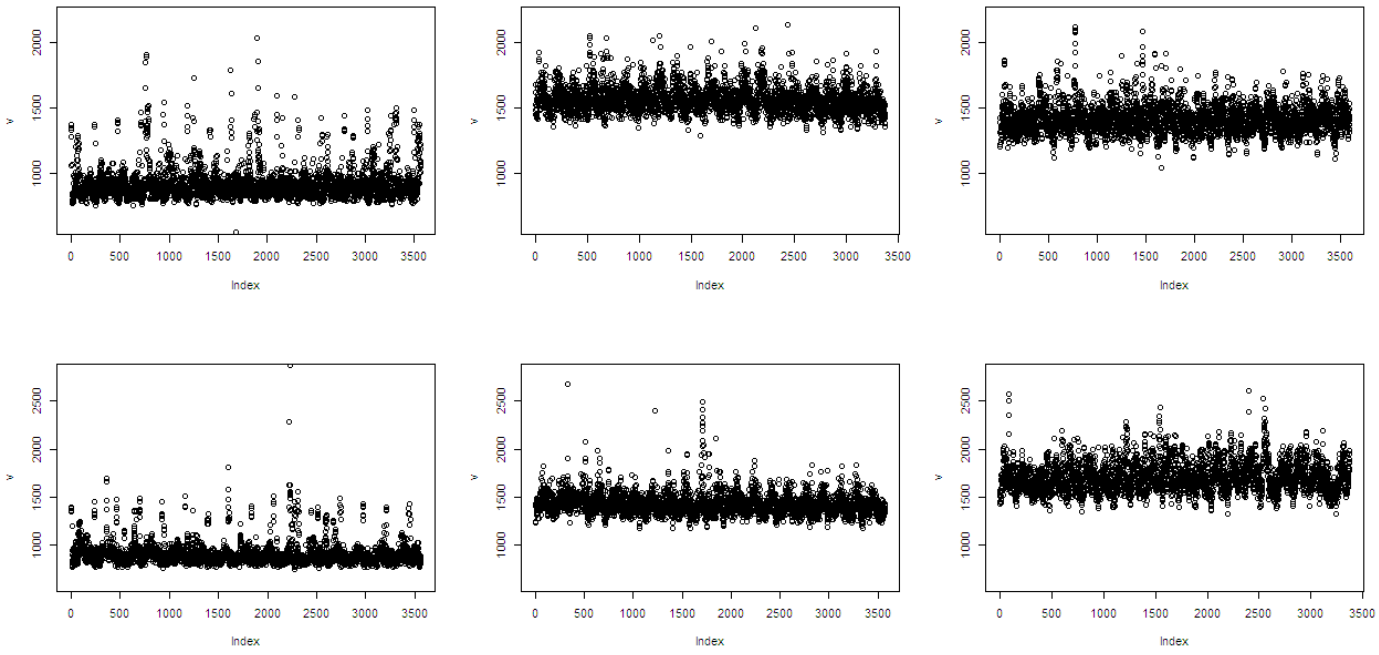
```
      x <- NULL
# Put all files in working directiory
# Scrape Files with a single-digit date
        r <- c(2:6,9)  # weekdays
        basefile <- 'ATS-Data_2011-05-0X.csv'
        for (day in r) {
                thisfile <- gsub("X", day, basefile)
                cat("Scraping", thisfile, "\n")
                z <- read.csv(thisfile, skip=1, header=FALSE, as.is=TRUE)
                x <- rbind(x, z)
        }
# Scrape Files with a double-digit date
        s <- c(10:13,16:20,23:27, 30:31)   # weekdays
        basefilee <- 'ATS-Data_2011-05-XX.csv'
        for (dayy in s) {
                thisfile <- gsub("XX", dayy, basefilee)
                cat("Scraping", thisfile, "\n")
                z <- read.csv(thisfile, skip=1, header=FALSE, as.is=TRUE)
                x <- rbind(x, z)
        }
# Load in Companion File Detailing Station Information
        y <- read.csv('stops.txt', sep=",", as.is=TRUE, skip=1,header=FALSE)
# Create a Matrix with Columns for:
        mat <- matrix(NA, nrow=nrow(x), ncol=10)
      colnames(mat) <-
      c("dayinmay","trainline","stopid","arr/dep","secondsaftermidnight","trainid","stoplo
      ng","stoplat","uptown/downtown","late")
#1: Date in May
       mat1 <- unlist(substr(x[ ,1], 9,10))
  mat[ ,1] <- mat1
#2: Line Number (1,2, or 3)
```

```
  mat[ ,2] <- x[ ,6]

#3: Stop ID#

  mat[ ,3] <- x[ ,7]

#4: Arrival (1) or Departure (2) from a particular station

     mat4 <- unlist(x[,5])  #   1 = 'Arr'   2 = 'Dep'
  mat[ ,4] <- mat4

#5: Seconds after Midnight that Arr or Dep occurs

      x[ ,4] <- substr(x[ ,4], 12,19)
      out1 <- as.numeric(unlist(substr(x[ ,4], 1,2)))
      out2 <- as.numeric(unlist(substr(x[ ,4], 4,5)))
      out3 <- as.numeric(unlist(substr(x[ ,4], 7,8)))
      out4 <- 3600*out1+out2*60+out3
  mat[ ,5] <- out4

#6: Train Number

      mat6 <- unlist(substr(x[ ,2], 4,8))
  mat[ ,6] <- mat6
  mat[,6] <- gsub("+", "0", mat[,6], fixed=TRUE)

#7: Stop Latitude

      index <-match(mat[ ,3],y[ ,1])
  mat[ ,7] <- y[index,5]

#8: Stop Longitude

  mat[ ,8] <- y[index,6]

#9: Uptown or Downtown Train (uptown==0, Downtown==1)

  mat[ ,9] <- x[ ,3]

#10: Placeholder for OnTime/Late Color

  mat[,10] <- 0

################ modify initial matrix ################################################

# Limit scope to trains between 4am and 9pm

      mat <- mat[as.numeric(mat[ ,5]) > 14400 & as.numeric(mat[ ,5]) < 75600, ]

# Limit scope to stops on Line 1 (or where Lines 2 & 3 overlap line 1)

      mat <- mat[as.numeric(mat[,3]) < 150,]

# Specify for only arrivals at a Station

      mat <- mat[mat[,4]==1,]

# Remove all Rows with NA's

      toremove <- apply(is.na(mat),1,any)
      mat <- mat[!toremove,]

# Establish object 'mat' as a matrix

      mat <- matrix(as.numeric(mat), nrow=nrow(mat))
```

I have included the first six rows of the matrix 'mat' on the following page so that the reader can have a better understanding of what is available in this data set:

day	line	stopid	arr/dep	secaftermidnight	trainid	stoplong	stoplat	up/downtown	color
2	2	125	1	14417	3110	40.76825	-73.98193	0	0
2	2	124	1	14497	3110	40.77344	-73.98221	0	0
2	2	123	1	14588	3110	40.77845	-73.98197	0	0
2	2	122	1	14678	3110	40.78393	-73.97992	0	0
2	2	121	1	14760	3110	40.78864	-73.97622	0	0
2	2	120	1	14846	3110	40.79392	-73.97232	0	0

It is important to note that the location of the subway car (latitude and longitude) is known only for the second at which a train car arrives at a new station. In the appendix, I show the code required to "interpolate" the location of each train for each second of each day. Given that each of these cases receives a separate row, this "second-by-second" matrix is extremely large. However, instead of using the rbind() command which would become unwieldy, I chose to pre-define an extremely large matrix of the appropriate number of rows and columns and then use a loop to insert each train's mini-matrix one after another into the structure.



The upper left plot displays the distribution of all 3600 downtown express trains (lines 2 & 3) to travel from 96th Street to Chambers Street during the month of May 2011. The upper center plot is of the same distribution for downtown line 1 trains (notice the increase in roughly 500 seconds to travel the same distance due to all the extra stops). The upper right plot is the distribution of all the downtown line 1 trains to travel from 242nd Street (Bronx) to 96th Street during the month of May. The bottom three plots are the analogous plots for uptown trains. The

scale along the x-axes is simply the train number during the month of May (pseudo-chronological order) and the scale along the y-axes is time in seconds.

There are three interesting points to be made about the series of plots on the previous page. First, there is a tighter band of arrival times for the express trains than for the local trains. This is to be expected given that an express train has more start and stop times and therefore, more opportunities to fall off its schedule. Nevertheless, it is good that our common sense is reflected in the plots. Additionally, this lack of grossly tardy express trains will show up in our moving maps.

Secondly, this plot provides motivation for the next few plots. There appears to be 'streaking' in the plots above – multiple trains in a row being late and showing up as a vertical band. One of my initial goals of this project was to learn the extent to which one train arriving late effects the following train. From a cursory graphical approach, it appears that these "bunching" effects might be relevant.

Finally, there is a vague zig-zagging pattern across a few of the plots on the previous page. Could this be a variation depending on the time of day? This is unclear given that there are 22 days of data represented in the plots and there are not 22 "zig-zags." Even so, this also appears to be an interesting question to look into.

From taking a look at the previous plots, I can tell that there is a clear band of typical arrival times for each stop or interval (I have chosen to call an 'interval' the amount of time in seconds between a train's arrival at any stop and then its arrival at the following stop). The outliers can be determined and for each stop, I define a train that is above the 3rd quartile as late. Given that the histogram of values has a disproportionately large tail (there are not a lot of trains that arrive early), the 3rd quartile definition of 'late' actually makes quite a bit of sense both theoretically and empirically. This cutoff occurs for a different number of seconds for each interval given that stops are not evenly spaced up and down Manhattan. I have run a massive loop to gather the third quartile of the travel times between each stop in the uptown and downtown direction for both the express and local trains. I imputed those values into a new reference matrix named 'doug'[1]. The code is shown on the following page:

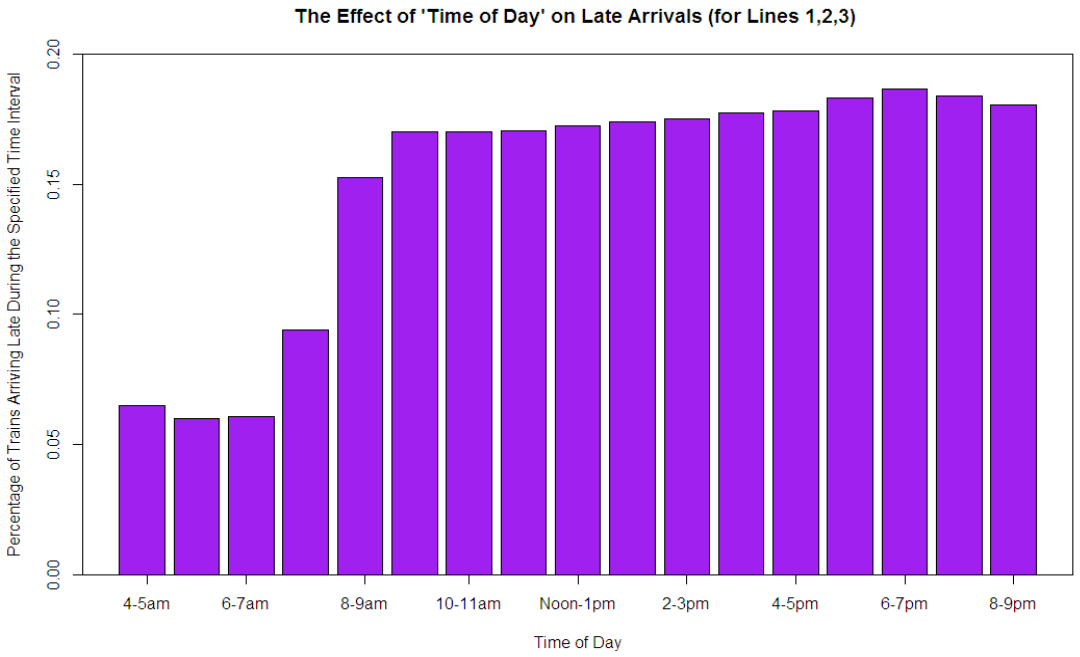
[1] Doug is the name of the Author's younger brother.

```
doug <- matrix(NA, nrow=4, ncol=42)
downtown1 <- c(NA,NA,89 ,83 ,NA,105,80 ,91,91,94,122,102,118, 88,109,113,77
,89,118,115,85,99, 97,91,110,118,122, 81,71,69,69,94,96,71,69,79,106,70,217,NA,NA,NA)
uptown1 <-   c(NA,NA,214,113,NA,90 ,102,98,79,94,96 ,120,106,120,106,115,111,81,86
,129,98,87,100,93,103,108,155,105,78,73,68,77,78,84,83,65, 83,66, 76,NA,NA,NA)
downtown23 <- matrix(NA, nrow=1, ncol=42)
    downtown23[1,20] <- 303
    downtown23[1,23] <- 417
    downtown23[1,27] <- 122
    downtown23[1,28] <- 159
    downtown23[1,32] <- 281
uptown23   <- matrix(NA, nrow=1, ncol=42)
  uptown23[1,23] <- 286
  uptown23[1,27] <- 468
  uptown23[1,28] <- 120
  uptown23[1,32] <- 168
  uptown23[1,37] <- 236
doug[1,] <- downtown1
doug[2,] <- uptown1
doug[3,] <- downtown23
doug[4,] <- uptown23
```

Notice that the top two rows of 'doug' are for line 1 and these rows have very few NA's. The opposite is the case for the bottom two rows of 'doug' (lines 2 &3). This discrepancy is simply due to the number of stops included in the overlap between line 1 and lines 2 & 3. Express trains only stop 5 times on the line 1 route and thus the rest of the values in the last two rows of 'doug' are NA's. I should mention that the NA's in the top two rows are simply due to skipped numbers in the stop identification numbering system.

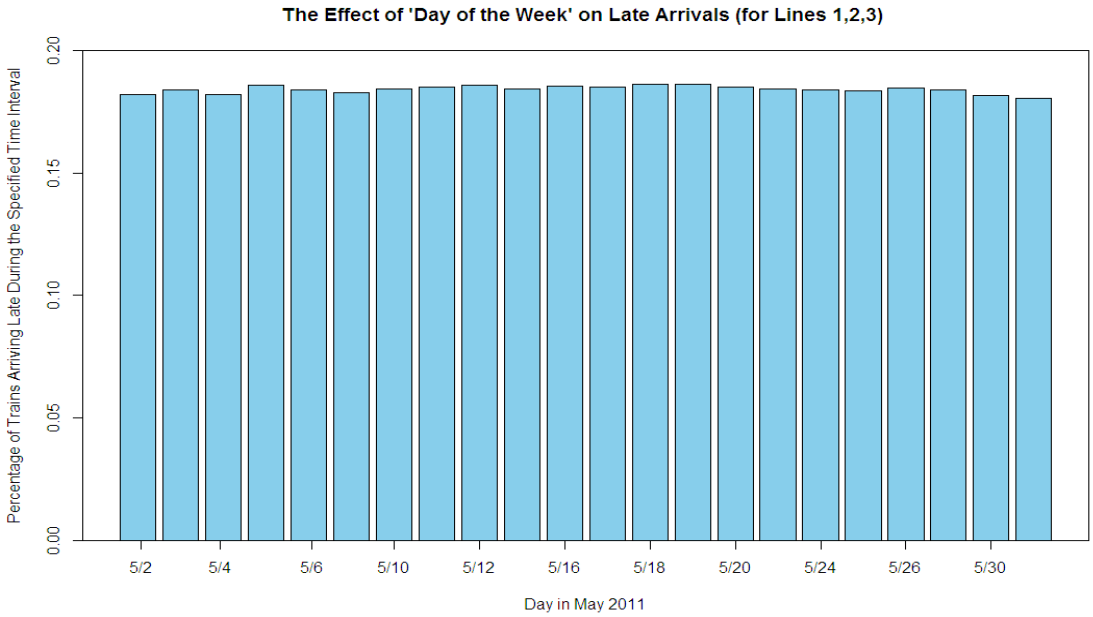
I then compared the time interval for each case to the 3rd quartile value. If it was greater (i.e. the train took more time to travel the distance), the tenth (placeholder) column in 'mat' received a "1" to indicate late arrival between those two stops. I chose to define a train as late only if it was late in that particular one-stop interval. Otherwise, I found that all the subways could be considered late because once they get off schedule, it is very unusual for them to make up time on the rails.

On the following page, I display a graph which splices the data set into 17 smaller sets – each bar corresponds to all the one-stop trips that occurred during an hour range (for example 7am-8am) for the entire month of May:

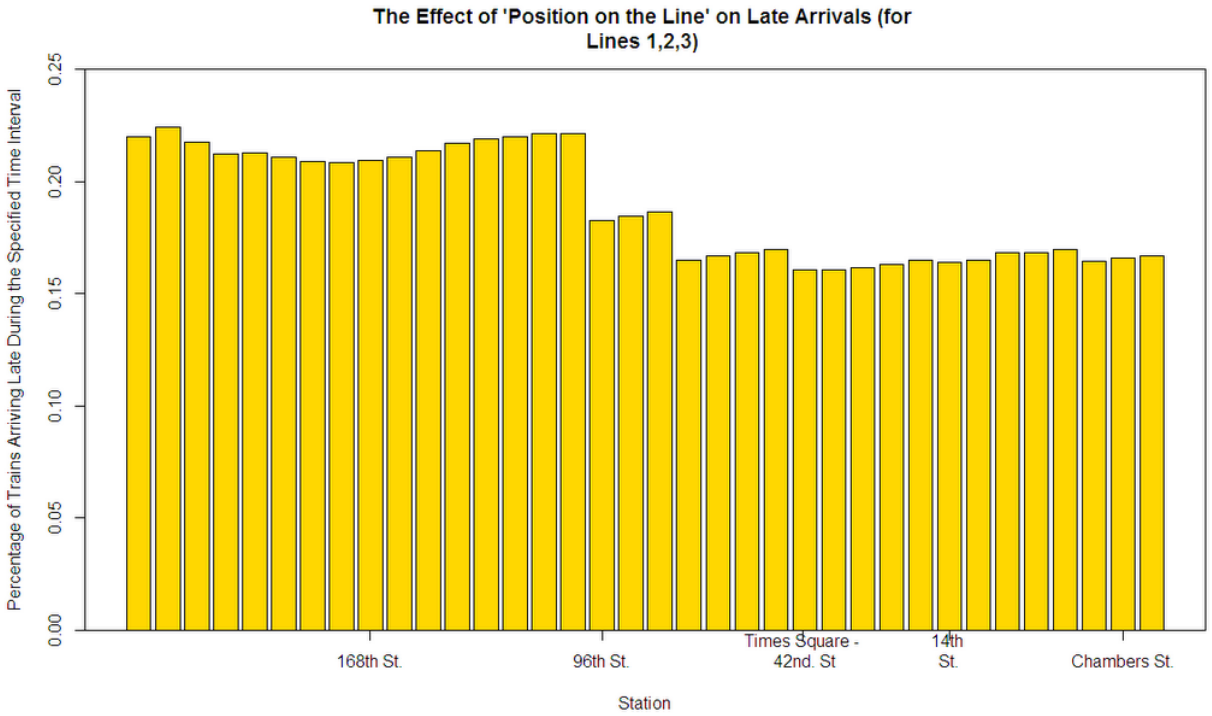


This plot shows that there is a sharp increase in percentage of trains that arrive late beginning during rush hour and then the subway system continues to have a consistent percentage of trains running late throughout the day. This indicates that either the trains have a difficult time getting back on schedule once they are set off *or* evening rush hour is statistically insignificant in terms of delaying *new* trains.

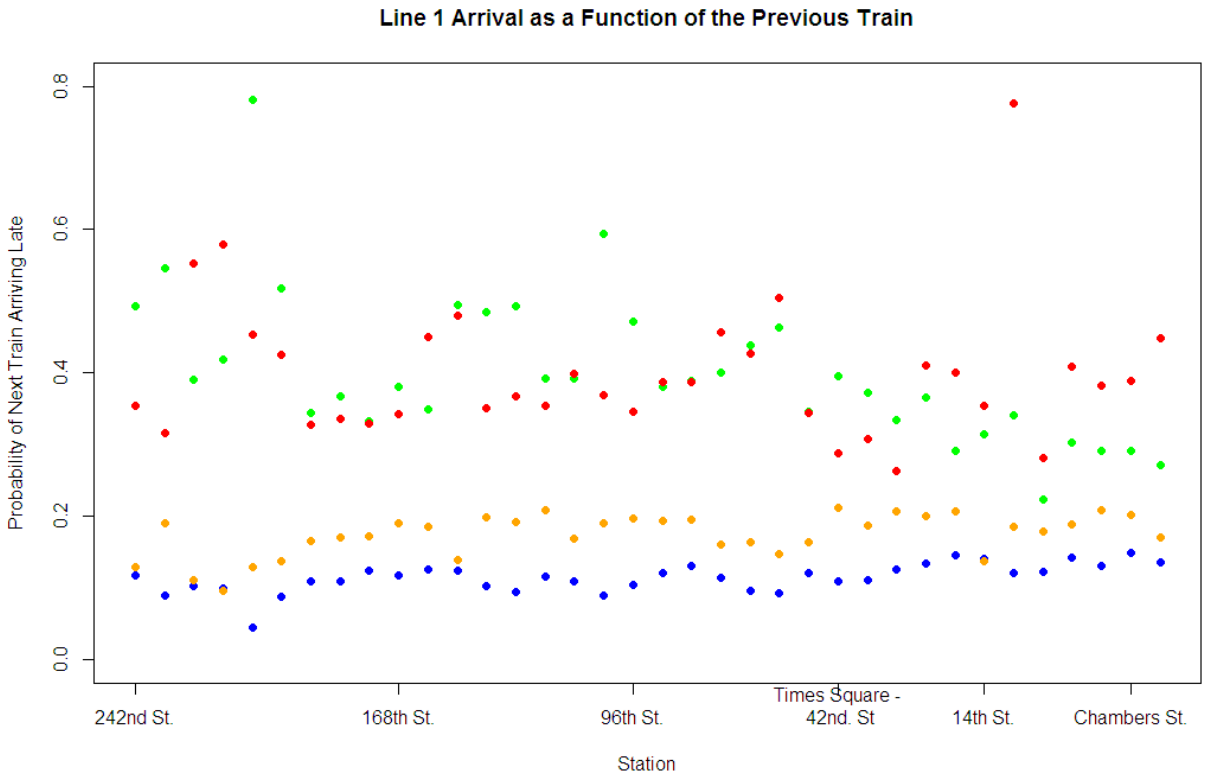
Below, it is clear that there is not a trend based on day of the week; all of the days have a similar percentage of late trains:



Finally, there does not appear to be a trend related to a station or set of stations. I believe that the slight decrease from left to right along the graph is simply buoyed by an influx of express trains entering the line at 96th St. (see graph below). Even if the line was symmetric, the trend is too small to get excited about.



Next, I will focus on the question of "bunching" and the extent to which the late arrival of one train will impact the train behind it. Consider a hypothetical scenario with two subways running along the same tracks. Given that the first train is late, there are two possible cases for the arrival of the second train: either it is late or it is on time. In a world of "zero bunching" and no delays due to train traffic, we would expect a late arrival of the first train to have no effect on the second train. However, the real world dictates otherwise. The plots on the following page show that the uptown and downtown trains that directly follow a late train (red and green, respectively) are significantly more likely to be late than the uptown and downtown trains which follow an on-time train (orange and blue, respectively). I have split up the express trains and the local trains because they run on different sets of tracks and therefore could reasonably be expected to run independently of each other.



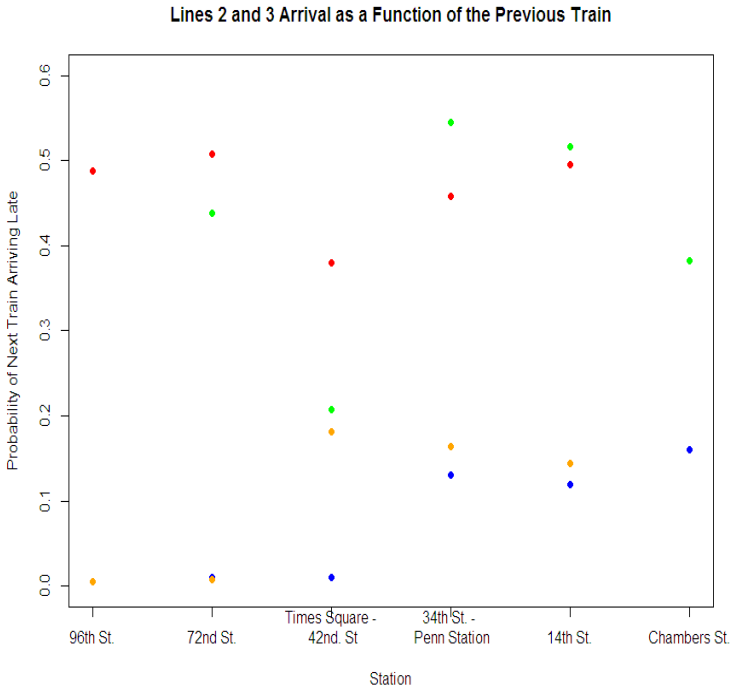
Red: Uptown directly following a late train

Green: Downtown directly following a late train

Orange: Uptown directly following an on-time time

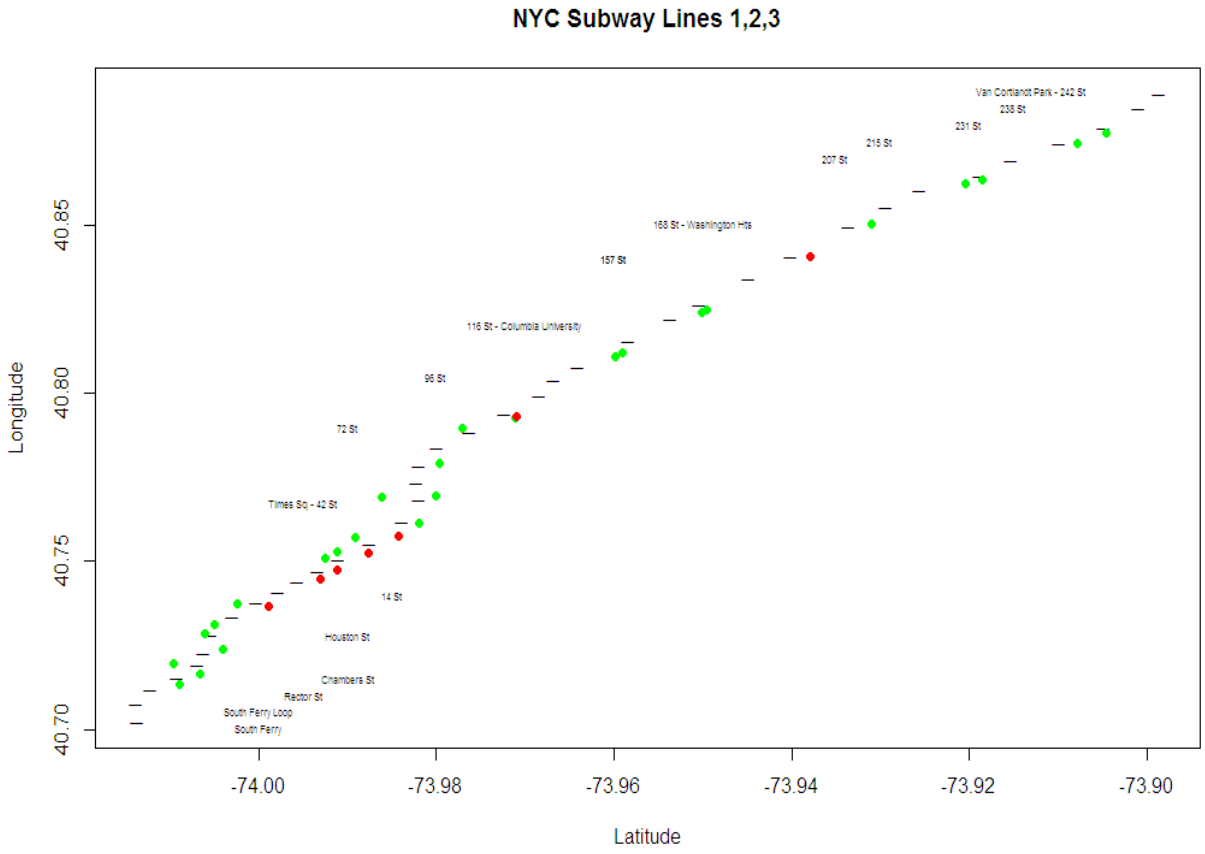
Blue: Downtown directly following an on-time time

Note: Local (line1) and Express (lines 2 & 3) trains run on different sets of tracks. The percent difference between red/green and orange/blue is significantly greater for Express Trains than for Local Trains. However, the percent chance that a local train is late is significantly greater than the percent chance that an express train will be late.



Finally, I will simply present a visual "moving map" of the bunching of trains. The detailed code is included in full in the Appendix. The plot is designed to shuffle through the positions of active trains at a specified range of time. The times can be specified in the fourth line below. Keep in mind that t is a multiple of 5 so t= 9510 corresponds to roughly 5:06 PM: (9510*5+14000)%%3600.

```
b=1:max

plot(mat[b,8], mat[b,7], pch="_", main="NYC Subway Lines 1,2,3", xlab='Latitude',
ylab='Longitude')

for (t in 9500:9550) {
  points(latone[,t]+0.002, longone[,t], col=colorone[,t], pch=16)
  points(lattwo[,t]-0.002, longtwo[,t], col=colortwo[,t], pch=16)
  Sys.sleep(0.03)
  points(latone[,t]+0.002, longone[,t], col='white', pch=16)
  points(lattwo[,t]-0.002, longtwo[,t], col='white', pch=16)
}
```



The figure above is a screen shot of the moving map at t=9510, or 5:06pm. The express trains are shown as dots on the left side of the central spine and the local trains are shown on the right side. It is easy to see that the two lines are independent; while line 1 has bunching of late arrivals between Times Square and 14th St., lines 2 and 3 experience nothing of the sort.

Brief Conclusions:

The Express trains and Local trains are quite different in terms of their late arrival patterns. While a Local train is more likely to be late than an Express train, a previous Express train being late has a greater impact on the next express train than the analogous situation for Local trains. Additionally, Express trains have a narrower band of arrival times than Local trains due to their lack of frequent stops.

Day of the week and specific stop didn't seem to have much of an effect on the probability of late arrival, but the chance of being on a late train does go up as the day progresses. In particular, the morning rush hour is taxing on lines 1, 2, and 3; in a span of 2 hours, the percent chance of being on a late train nearly triples.

From looking at the moving maps and raw data, it appears that Express and Local trains truly are independent. In the future, the "moving map" tool that I developed could be expanded to model other sections of the NYC subway system and capture degree of lateness with a palette of colors instead of just red and green. With so many New Yorkers depending on lines 1, 2, and 3 in Manhattan, I believe that a closer look at the real-time data is warranted. Perhaps a more detailed analysis of these and other subway lines would be able to help us better optimize New Yorkers' travel plans.

Appendix: R Code for the Moving Map on Page 10

```
############### process data into initial matrix ################################

x <- read.csv('ATS-Data_2011-05-03.csv',
                sep=",", as.is=TRUE, skip=1,header=FALSE)
y <- read.csv('stops.txt',
              sep=",", as.is=TRUE, skip=1,header=FALSE)

mat <- matrix(NA, nrow=nrow(x), ncol=10)
colnames(mat) <-
c("dayinmay","trainline","stopid","arr/dep","secondsaftermidnight","trainid","stoplong
","stoplat","uptown/downtown", "color")

      mat1 <- unlist(substr(x[ ,1], 9,10))
  mat[ ,1] <- mat1
  mat[ ,2] <- x[ ,6]
  mat[ ,3] <- x[ ,7]
      mat4 <- unlist(x[,5])  #   1 = 'Arr'   2 = 'Dep'
  mat[ ,4] <- mat4
      x[ ,4] <- substr(x[ ,4], 12,19)
      out1 <- as.numeric(unlist(substr(x[ ,4], 1,2)))
      out2 <- as.numeric(unlist(substr(x[ ,4], 4,5)))
      out3 <- as.numeric(unlist(substr(x[ ,4], 7,8)))
      out4 <- 3600*out1+out2*60+out3
  mat[ ,5] <- out4
      mat6 <- unlist(substr(x[ ,2], 4,8))
  mat[ ,6] <- mat6
  mat[,6] <- gsub("+", "0", mat[,6], fixed=TRUE)
      index <-match(mat[ ,3],y[ ,1])
  mat[ ,7] <- y[index,5]
  mat[ ,8] <- y[index,6]
  mat[ ,9] <- x[ ,3]
  mat[,10] <- 0


################ modify initial matrix
################################################

mat <- mat[as.numeric(mat[ ,5]) > 14400 & as.numeric(mat[ ,5]) < 75600, ]
mat <- mat[as.numeric(mat[,3]) < 150,]
mat <- mat[mat[,4]==1,]
toremove <- apply(is.na(mat),1,any)
mat <- mat[!toremove,]
mat <- matrix(as.numeric(mat), nrow=nrow(mat))
max <- as.numeric(nrow(mat)-1)

################### determine the number of rows in second-by-second matrix
#####################

vn <- 1
for (b in 1:max) {
  if (mat[b,6]==mat[b+1,6] &  mat[b,2]==mat[b+1,2] & mat[b,9]==mat[b+1,9] &
(abs(as.numeric(mat[b,3]) - as.numeric(mat[b+1,3])) < 6) ) {
    v <- as.numeric(mat[b+1,5]) - as.numeric(mat[b,5])
#  if (v>300) cat("Row", b, "time interval", v, "\n")
  } else {
    v <- 1
  }
  vn <- vn + v
}


###### Color Mat[,10]  ##########################
doug <- matrix(NA, nrow=4, ncol=42)
downtown1 <- c(NA,NA,89 ,83 ,NA,105,80 ,91,91,94,122,102,118, 88,109,113,77
,89,118,115,85,99, 97,91,110,118,122, 81,71,69,69,94,96,71,69,79,106,70,217,NA,NA,NA)
uptown1 <-   c(NA,NA,214,113,NA,90 ,102,98,79,94,96 ,120,106,120,106,115,111,81,86
,129,98,87,100,93,103,108,155,105,78,73,68,77,78,84,83,65, 83,66, 76,NA,NA,NA)
downtown23 <- matrix(NA, nrow=1, ncol=42)
    downtown23[1,20] <- 303
```

```
    downtown23[1,23] <- 417
    downtown23[1,27] <- 122
    downtown23[1,28] <- 159
    downtown23[1,32] <- 281
uptown23   <- matrix(NA, nrow=1, ncol=42)
  uptown23[1,23] <- 286
  uptown23[1,27] <- 468
  uptown23[1,28] <- 120
  uptown23[1,32] <- 168
  uptown23[1,37] <- 236
doug[1,] <- downtown1
doug[2,] <- uptown1
doug[3,] <- downtown23
doug[4,] <- uptown23


for (b in 1:max){
  if (mat[b,2]==1 & mat[b,9]==1 & mat[b,3]!=101 & mat[b,3]!=103 & mat[b,3]!=104 &
mat[b,3]!=139 & mat[b,3]!=142 & ((as.numeric(mat[b+1,5])-as.numeric(mat[b,5])) >
doug[1,(as.numeric(mat[b,3])-100)])) {
    mat[b,10] <- 1
  }
  else if (mat[b,2]==1 & mat[b,9]==0 & mat[b,3]!=101 & mat[b,3]!=103 & mat[b,3]!=104
& mat[b,3]!=106 & mat[b,3]!=142 & ((as.numeric(mat[b+1,5])-as.numeric(mat[b,5])) >
doug[2,(as.numeric(mat[b,3])-100)])) {
    mat[b,10] <- 1
  }
}


 e <- mat[mat[,2]!=1,]

k <- 1
r <- NULL
for (b in 3:2993) {
  if ( (abs((e[b,3]) - (e[b+1,3])) == 1) & (abs((e[b,3]) - (e[b-1,3])) == 1) ) {
    r[k] <- (e[b,6])
    k <- k+1
  }
}
r <- unique(r)
p <- NULL
  k <- 1
  for (b in 1:2993){
    for (i in r){
      if (e[b,6]==i) {p[k] <- b
                      k <- k+1}
    }
  }

e <- e[-p,]

for (b in 1:2733) {
  if (e[b,2]==e[b+1,2] & e[b,6]==e[b+1,6] & e[b,9]==1 & e[b+1,9]==1 & (abs(e[b,3]-
e[b+1,3]) < 6) & ((as.numeric(e[b+1,5])-as.numeric(e[b,5])) >
doug[3,(as.numeric(e[b,3])-100)])) {
    e[b,10] <- 1
  }
  else if (e[b,2]==e[b+1,2] & e[b,6]==e[b+1,6] & e[b,9]==0 & e[b+1,9]==0 &
(abs(e[b,3]-e[b+1,3]) < 6) & ((as.numeric(e[b+1,5])-as.numeric(e[b,5])) >
doug[4,(as.numeric(e[b,3])-100)])) {
    e[b,10] <- 1
  }
}

mat <- rbind(e, mat)
################## create second-by-second matrix
########################################

sbs <- matrix(NA, nrow=2101168, ncol=7)
vn <- 1
for (b in 1:18965) {
  if (mat[b,6]==mat[b+1,6] & mat[b,2]==mat[b+1,2] & mat[b,9]==mat[b+1,9] &
abs(as.numeric(mat[b,3]) - as.numeric(mat[b+1,3])) < 6 ) {
```

```
    v <- as.numeric(mat[b+1,5]) - as.numeric(mat[b,5])
    foo <- matrix(as.numeric(unlist(mat[b,5:10])), nrow=v, ncol=6, byrow=TRUE)
    foo[,1] <- as.numeric(mat[b,5]) + 0:(v-1)   # Time column all at once.
    foo[v,3:4] <- as.numeric(unlist(mat[b+1,7:8])) # Get ending position
    foo[2:(v-1),3] <- foo[1,3] + (1:(v-2))/(v-1) * (foo[v,3]-foo[1,3]) # Interpolate
longitude
    foo[2:(v-1),4] <- foo[1,4] + (1:(v-2))/(v-1) * (foo[v,4]-foo[1,4]) # Interpolate
latitude
    sbs[vn:(vn+v-1),1:6] <- foo[1:v,]
    sbs[vn:(vn+v-1),7]<- mat[b,2]
  } else {
    v <- 1
    foo <- matrix(as.numeric(unlist(mat[b,5:10])), nrow=v, ncol=6, byrow=TRUE)
    sbs[vn,1:6] <- foo[1,]
    sbs[vn,7]<- mat[b,2]
  }
  vn <- vn + v
}


################## print 'new' matrix (trims down sbs to multiples of 5 seconds)
#################################

j<- 1
k <-1
one <- matrix(NA, nrow=241128, ncol=7)
two <- matrix(NA, nrow=179156, ncol=7)
for (i in 1:2101168){
  if ((sbs[i,1]%%5)==0) {
    if (sbs[i,7]==1) {
      one[j,] <- sbs[i,]
      j <- j+1
    }else {
      two[k,] <- sbs[i,]
      k <- k+1
    }
  }
}


################## CREATE long, lat, color matrices ################################


longone <- matrix(NA, nrow=369, ncol=13000)
latone <- matrix(NA, nrow=369, ncol=13000)
colorone <- matrix(NA, nrow=369, ncol=13000)
longtwo <- matrix(NA, nrow=876, ncol=13000)
lattwo <- matrix(NA, nrow=876, ncol=13000)
colortwo <- matrix(NA, nrow=876, ncol=13000)

g <- 0
i <- 1
n<- 1
for (n in 1:241127){
  if (one[n,2]==one[n+1,2]){
    j <- (one[n,1]-14000)/5
    longone[i,j] <- one[n,3]
    latone[i,j] <- one[n,4]
      if (one[n,6]==1) {colorone[i,j] <- "red"}
      else if (one[n,6]==0) {colorone[i,j] <- "green"}
  } else {i <- i+1}

}

i <- 1
n<- 1
for (n in 1:179155){
  if (two[n,2]==two[n+1,2]){
    j <- (two[n,1]-14000)/5
    longtwo[i,j] <- two[n,3]
    lattwo[i,j] <- two[n,4]
    if (two[n,6]==0) {colortwo[i,j] <- "green"}
    else  {colortwo[i,j] <- "red"}
```

```
    } else {i <- i+1}

}

################ MOVING MAP ######################

b=1:max

plot(mat[b,8], mat[b,7], pch="_", main="NYC Subway Lines 1,2,3", xlab='Latitude',
ylab='Longitude')
text(-73.913, 40.89,cex=0.5, paste ("Van Cortlandt Park - 242 St"), col='black')
text(-73.95, 40.85,cex=0.5, paste ("168 St - Washington Hts"), col='black')
text(-73.915, 40.885,cex=0.5, paste ("238 St"), col='black')
text(-73.995, 40.767,cex=0.5, paste ("Times Sq - 42 St"), col='black')
text(-74.00, 40.70,cex=0.5, paste ("South Ferry"), col='black')
text(-73.96, 40.84,cex=0.5, paste ("157 St"), col='black')
text(-73.92, 40.88,cex=0.5, paste ("231 St"), col='black')
text(-73.93, 40.875,cex=0.5, paste ("215 St"), col='black')
text(-73.935, 40.87,cex=0.5, paste ("207 St"), col='black')
text(-73.97, 40.82,cex=0.5, paste ("116 St - Columbia University"), col='black')
text(-73.995, 40.71,cex=0.5, paste ("Rector St"), col='black')
text(-73.96, 40.84,cex=0.5, paste ("157 St"), col='black')
text(-73.98, 40.805,cex=0.5, paste ("96 St"), col='black')
text(-74.00, 40.705,cex=0.5, paste ("South Ferry Loop"), col='black')
text(-73.99, 40.715,cex=0.5, paste ("Chambers St"), col='black')
text(-73.99, 40.79,cex=0.5, paste ("72 St"), col='black')
text(-73.985, 40.74,cex=0.5, paste ("14 St"), col='black')
text(-73.99, 40.728,cex=0.5, paste ("Houston St"), col='black')

t <- 9510
for (t in 9500:9550) {
  points(latone[,t]+0.002, longone[,t], col=colorone[,t], pch=16)
  points(lattwo[,t]-0.002, longtwo[,t], col=colortwo[,t], pch=16)
  Sys.sleep(0.03)
  points(latone[,t]+0.002, longone[,t], col='white', pch=16)
  points(lattwo[,t]-0.002, longtwo[,t], col='white', pch=16)
}
```