Problem Set 3, Solutions

Statistics 506, Fall 2017

1. Problem3_1.pdf (./Problem3_1.pdf)

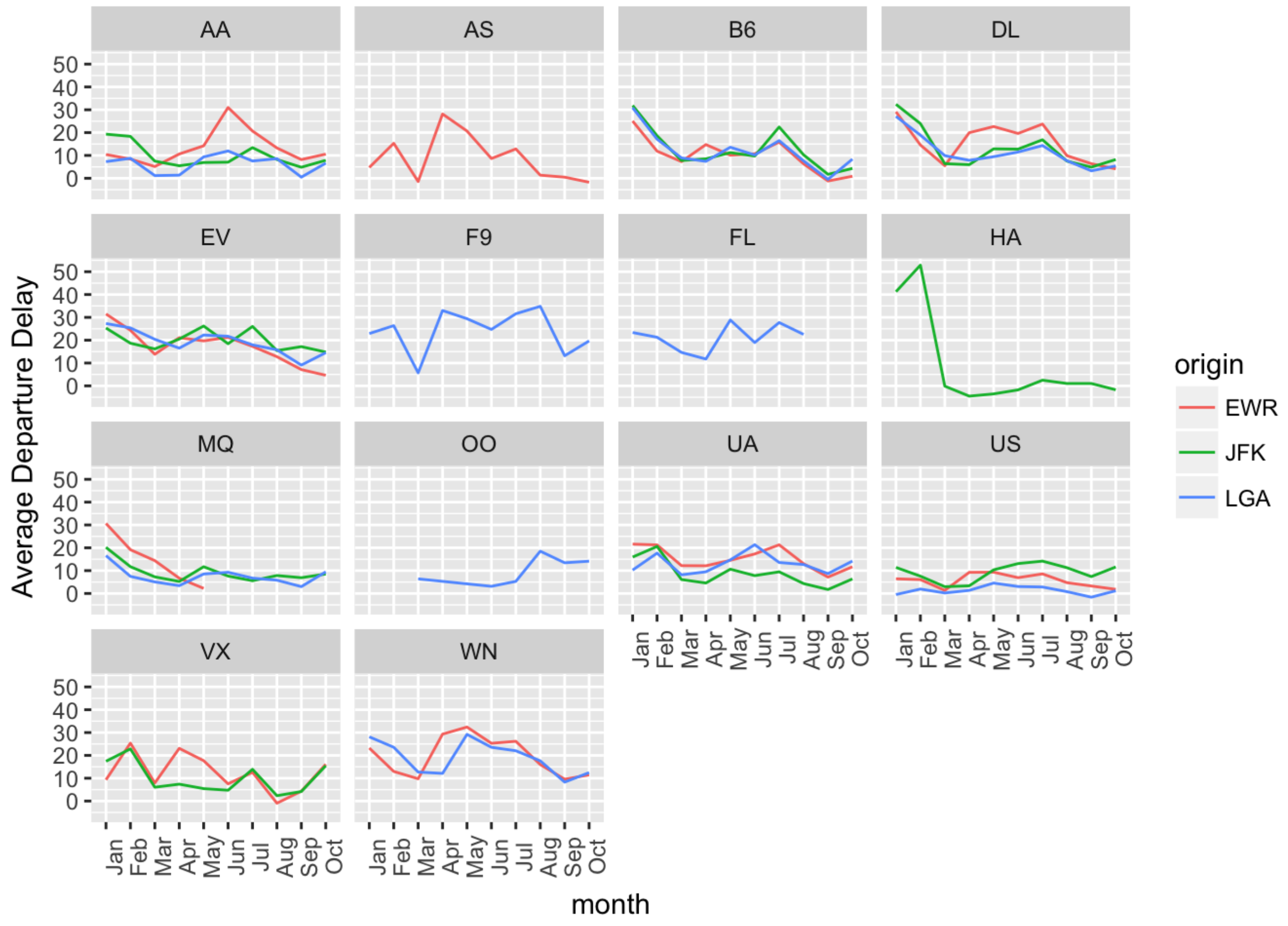
2. *Use the `NYCflights14` data from here (https://github.com/arunsrinivasan/flights/wiki/NYCflights14/flights14.csv) to answer the following questions. Each part can and should be answered using a single (perhaps chained) `data.table` expression prior to any requested plotting.*

First load the `data.table` package and read in the data.

```
## libraries
library(data.table)
library(tidyverse)
library(lubridate)
library(plotly)

## read in data
nyc14 = fread('https://github.com/arunsrinivasan/flights/wiki/NYCflights14/flights14.csv')
```

a. Compute the average departure delay per flight for all carriers by month. Then, graph your results as a spaghetti plot showing the time trends for each carrier.

```
nyc14[,.(avg_dep_delay  = mean(dep_delay)),.(month, origin, carrier)] %>%
  .[,.(month=factor(month.abb[month], month.abb),avg_dep_delay,origin,carrier),] %>%
  ggplot(aes(x=month, y=avg_dep_delay, group=origin, color=origin)) +
  geom_line() +
  facet_wrap(~carrier) +
  ylab("Average Departure Delay") +
  theme(axis.text.x=element_text(angle=90))
```



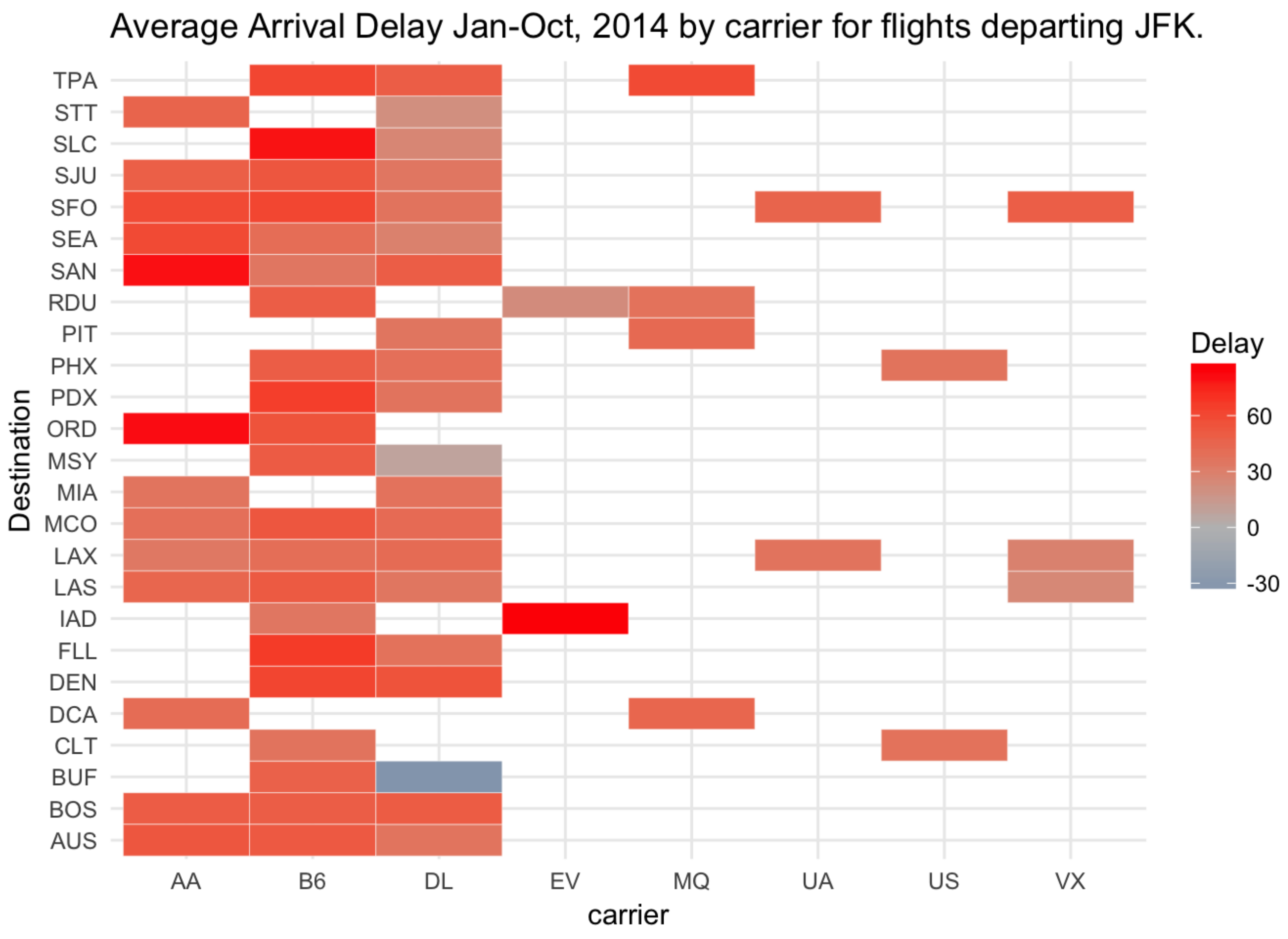
b. Compute the 90^{th} percentile of arrival delays by carrier, origin, and destination. For each origin airport, produce a heat map to display the data after filtering out destinations with only a single carrier.

Here is the computation.

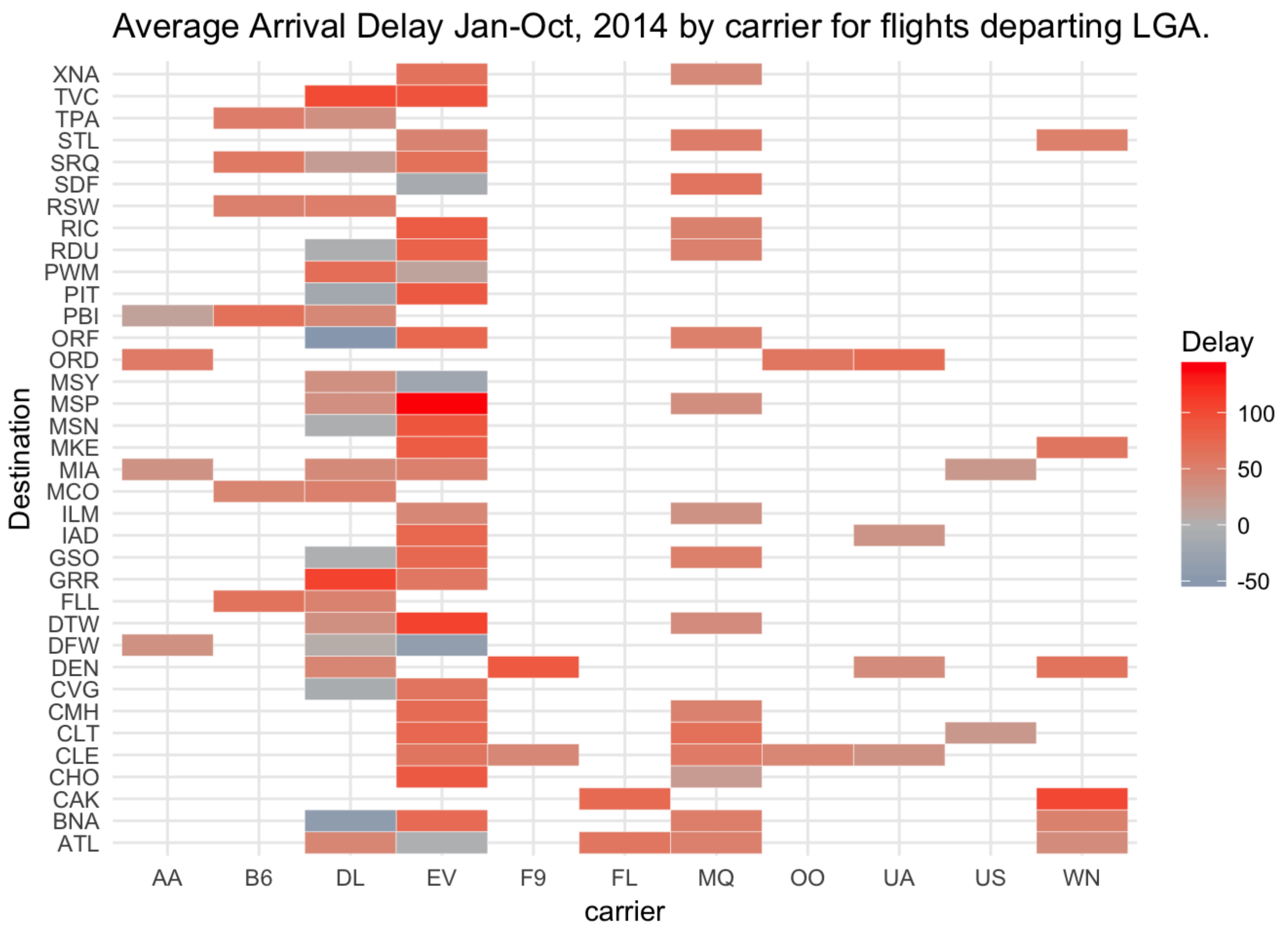
```
arr_delay = nyc14[,.(Delay = quantile(arr_delay, .9)), .(carrier, origin, dest)]
```

Here are the heatmaps.

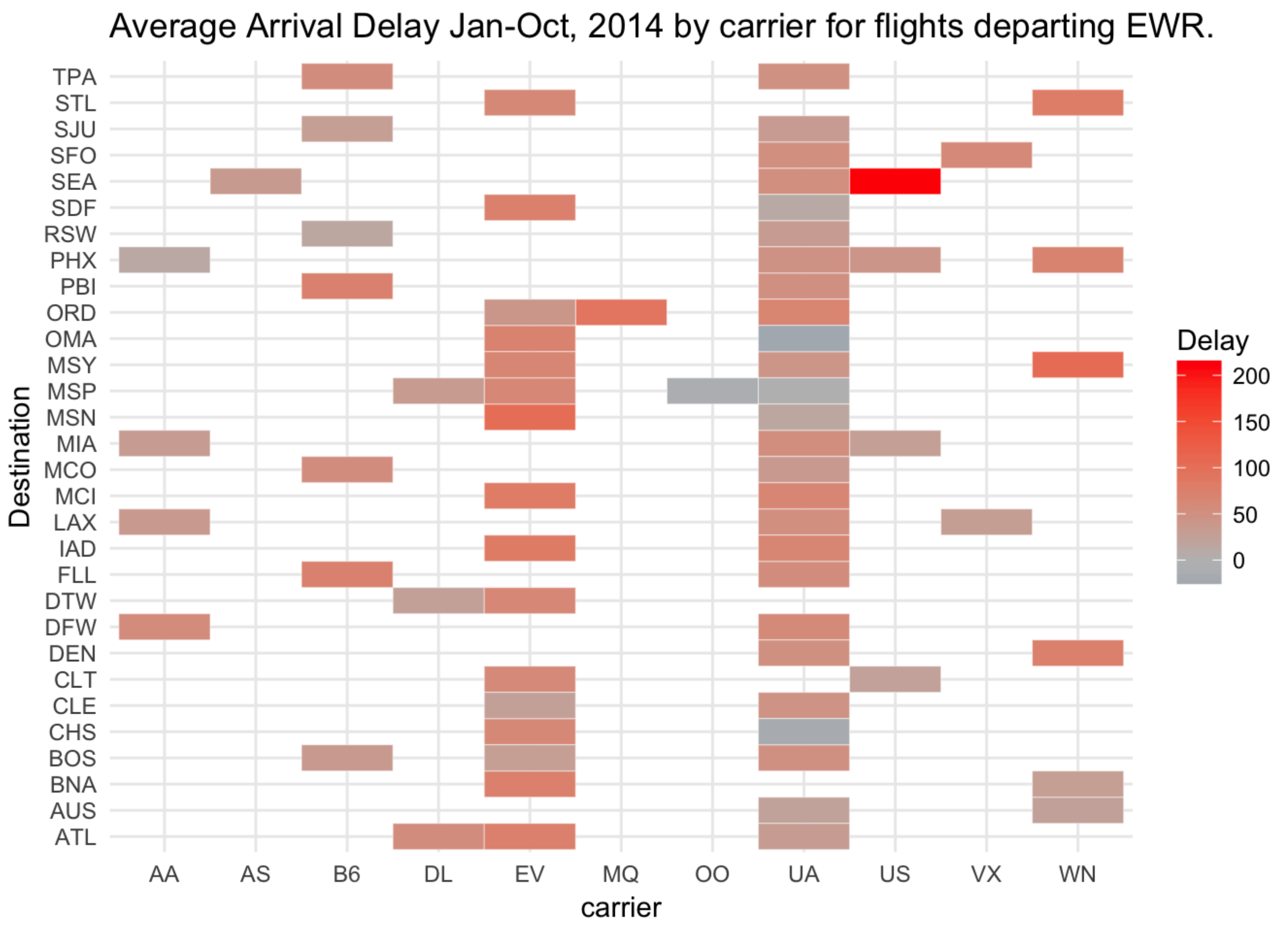
```
jfk_dest =
  arr_delay[origin == 'JFK', .(n_carrier = length(Delay)), .(dest)
           ][n_carrier>1,]$dest

arr_delay[origin == 'JFK' & dest %in% jfk_dest, ] %>%
  ggplot(aes(x=carrier, y=dest)) +
  geom_tile(aes(fill=Delay),color='white') +
  scale_fill_gradient2(low='steelblue',midpoint=0, mid='grey', high='red') +
  theme_minimal() +
  ylab('Destination') +
  ggtitle('Average Arrival Delay Jan-Oct, 2014 by carrier for flights departing JFK.')
```



```
lga_dest =
  arr_delay[origin == 'LGA', .(n_carrier = length(Delay)), .(dest)
          ][n_carrier>1,]$dest

arr_delay[origin == 'LGA' & dest %in% lga_dest, ] %>%
  ggplot(aes(x=carrier, y=dest)) +
  geom_tile(aes(fill=Delay),color='white') +
  scale_fill_gradient2(low='steelblue',midpoint=0, mid='grey', high='red') +
  theme_minimal() +
  ylab('Destination') +
  ggtitle('Average Arrival Delay Jan-Oct, 2014 by carrier for flights departing LGA.')
```



```
ewr_dest =
  arr_delay[origin == 'EWR', .(n_carrier = length(Delay)), .(dest)
           ][n_carrier>1,]$dest

arr_delay[origin == 'EWR' & dest %in% ewr_dest, ] %>%
  ggplot(aes(x=carrier, y=dest)) +
  geom_tile(aes(fill=Delay),color='white') +
  scale_fill_gradient2(low='steelblue',midpoint=0, mid='grey', high='red') +
  theme_minimal() +
  ylab('Destination') +
  ggtitle('Average Arrival Delay Jan-Oct, 2014 by carrier for flights departing EWR.')
```



c. For each origin airport, compute the average departure delay for each of the following time windows:

- 0:00 - 11:59
- 12:00 - 17:59
- 18:00 - 23:59.

```
avg_dep_delay =
  nyc14[,.(origin, dep_delay, window={dep_time > 1159}+{dep_time>1759})
  ][,.(avg_delay=mean(dep_delay)),.(origin, window)]

tab = avg_dep_delay %>% spread(origin,avg_delay) %>%
  mutate(window=factor(window))
levels(tab$window) = c('Morning','12-6 pm','After 6pm')
knitr::kable(tab, digits=1, caption='Average departure delay by time of day.')
```

Average departure delay by time of day.

window	EWR	JFK	LGA
Morning	4.6	4.6	2.0
12-6 pm	13.9	10.2	10.4
After 6pm	35.9	24.5	29.6

d. Within each flight, center and scale the air time by the mean. Next, bin the departure delays into fligths that left early or on time, flights delayed by less than 15 minutes, and flights that left more than 15 minutes late. For each bin of departure delays, compute a 95% confidence interval for the mean relative air time.

```
nyc14[, .(dep_delay_bin = {dep_delay > 0} + {dep_delay > 15},
           air_time = {air_time - mean(air_time)} / mean(air_time)),
         .(flight)          # Bin delays and standardize air time
     ][!is.na(air_time),
         .(avg = mean(air_time), se = sd(air_time)/sqrt(.N)),
         .(dep_delay_bin)  # Summarize standardized air_time by bin
     ][,.(avg=avg*100, lwr = 100*{avg - 1.96*se}, upr = 100*{avg + 1.96*se}),
          keyby=.(dep_delay_bin)
     ] %>%
  mutate(dep_delay_bin = c('On time or early', '15 minutes or less', 'More than 15 minutes'), c
i = sprintf('%4.1f (%3.1f, %3.1f)%%',avg,lwr,upr)) %>%
  select(dep_delay_bin,ci) %>%
  knitr::kable(col.names = c('Delay','Average Relative Air Time'), align = 'lc')
```

Delay	Average Relative Air Time
On time or early	-1.1 (-1.2, -0.9)%
15 minutes or less	2.2 (2.0, 2.5)%
More than 15 minutes	1.2 (0.9, 1.4)%

3. In this question you will use web-scraping to supplement the `NYCflights14` data with distance between all airports. To get you started, I have written a script (./AirportCodeWebScrape.R) using the R package `rvest` to scrape the distances between all destination airports in the `NYCflights14` data from this site (https://www.world-airport-codes.com).

a. Using my script as a model, write a script to scrape the distances between the three origin airports and from each origin airport to each destination airport from the same website. I suggest you test your script using just the three origin airports, then include the destination airports once you have it working. Your script should not find distances between destination airports as these are provided here (./AirportCodeDists.RData).

One way to accomplish this is to modify the script provided by appending the origin airport codes to the start of `dest_codes` and then running the final loop from 1 to 3. Here (./PS3_Q3.R) is a modified script.

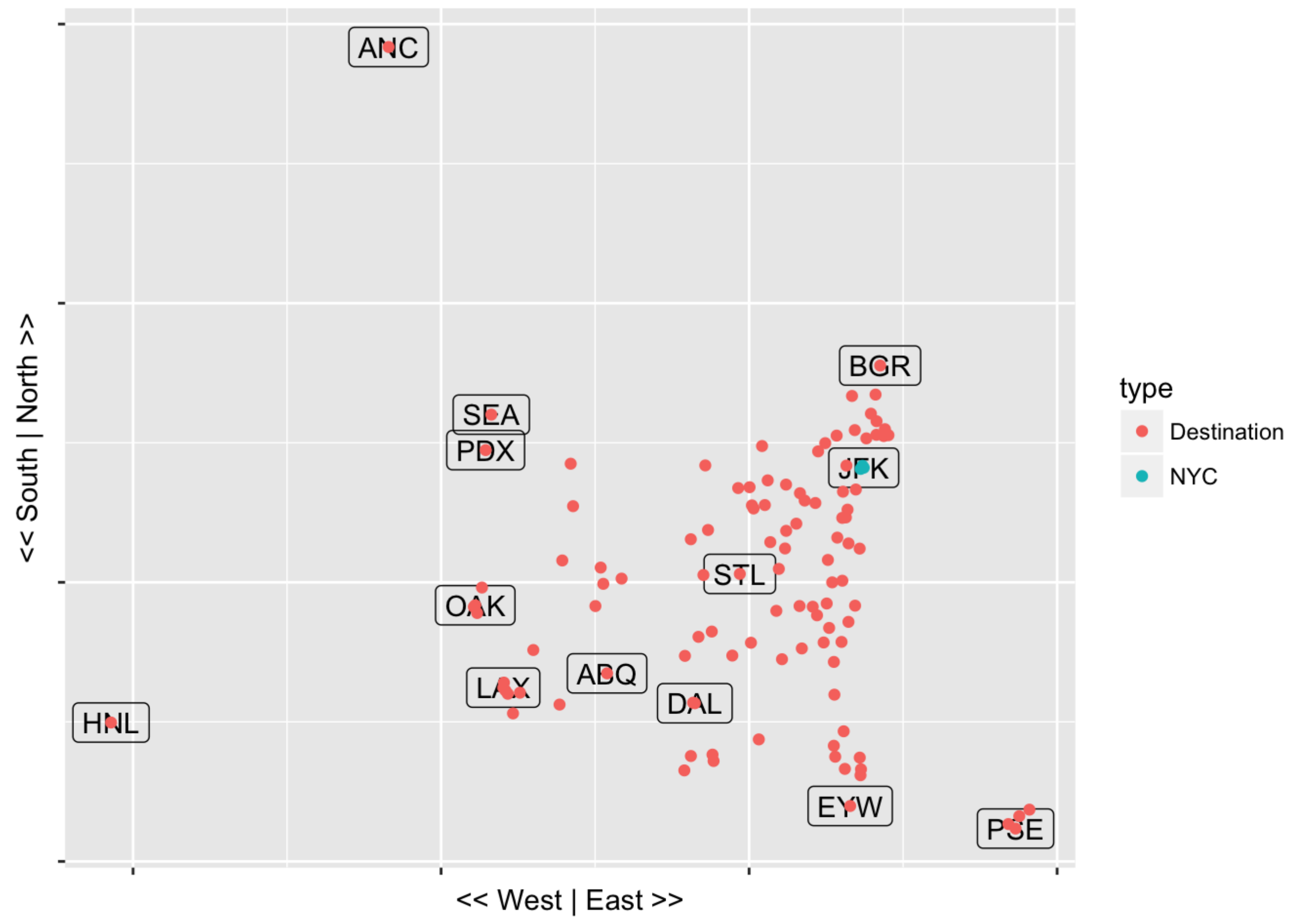
b. Combine the distances from part a, with the distances provided between all destination airports. Reshape these data into a 112 by 112 pairwise distance matrix.

```
# load the modified data
load('./AirportCodeDists_all.RData')

# reshape into a distance matrix
# converting 'from' and 'to' to factors retains order
D_df = df_dist %>%
  mutate(from = factor(from, unique(from)),
         to = factor(to, unique(to))
         ) %>%
  tidyr::spread(to, dist, fill=0)

# LGA was first, DAL last in
D_mat = rbind(
          cbind('LGA'=0,as.matrix(D_df[,-1])),
          'DAL'=0
        )
D_mat = t(D_mat) + D_mat
colnames(D_mat) = rownames(D_mat)
```

c. Use multidimensional scaling to produce a two-dimensional map for these 112 airports.

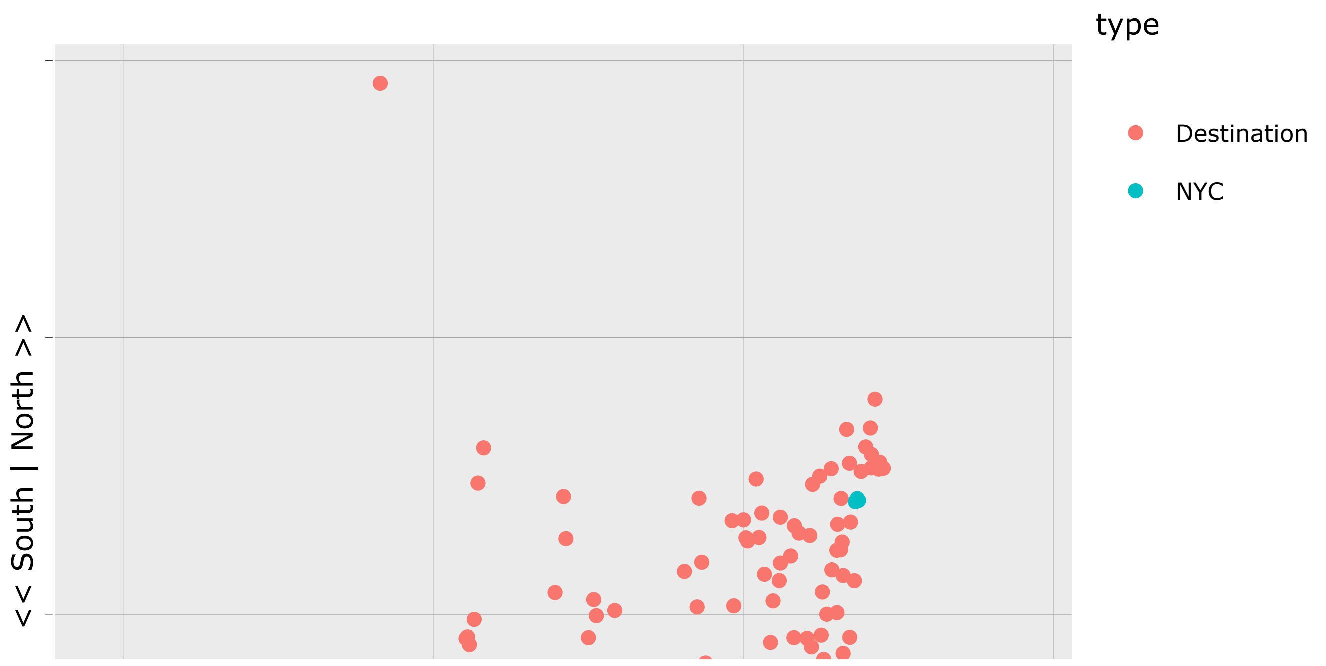
Here I have provided just a subset of labels for reference.

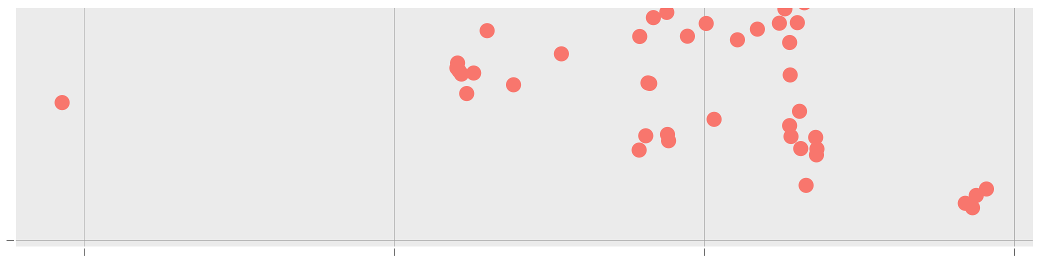
```
## get MDS coordinates
MDS = cmdscale(D_mat)

plot_data =
  tibble(airport = rownames(D_mat),
       x = -MDS[,1], y=-MDS[,2]
       ) %>%
  mutate(type =
   ifelse(airport %in% c('LGA','JFK','EWR'),
          'NYC', 'Destination'),
   label = NA
  )
labels = c('ANC','JFK','HNL','LAX','SEA','PDX','OAK','ABQ','DAL','EYW','PSE','BGR','STL')
plot_data$label[
  match(labels, plot_data$airport)
  ] = labels

plot_data %>%
 ggplot(aes(x=x, y=y)) +
 geom_label(aes(label=label),fill=NA) +
 geom_point(aes(col=type)) +
 xlab('<< West | East >>') +
 ylab('<< South | North >>') +
 theme(axis.text.x=element_blank(),
       axis.text.y=element_blank()
 )
```



You could suppress labels and use a mouse-over if your final plot is displayed in html.

```
p = plot_data %>%
 ggplot(aes(x=x, y=y, Name=airport)) +
 geom_point(aes(col=type)) +
 xlab('<< West | East >>') +
 ylab('<< South | North >>') +
 theme(axis.text.x=element_blank(),
       axis.text.y=element_blank()
  )
ggplotly(p)
```





4. In this question, you will combine the airport distance data from question 3 with the `NYCflights14` data to build visualizations. Where possible, use `data.tables` for aggregation and data manipulation.

a. Determine the number of flights per week from each origin airport to each destination airport among all carriers.

```
# days using lubridate
n_days = as.numeric(
  lubridate::ymd('2014-10-31') - lubridate::ymd('2014-01-01'), "days")

# flights per week
fpw = nyc14[ , .(weekly_flights = .N/n_days), by=.(origin, dest)]
fpw[order(-weekly_flights)]
```

```
##      origin dest weekly_flights
##   1:    JFK  LAX    33.68976898
##   2:    JFK  SFO    24.31683168
##   3:    LGA  ORD    23.27392739
##   4:    LGA  ATL    22.85478548
##   5:    LGA  MIA    16.77887789
##  ---                           
## 217:    LGA  SBN     0.00660066
## 218:    JFK  JAC     0.00330033
## 219:    LGA  DSM     0.00330033
## 220:    EWR  AVP     0.00330033
## 221:    JFK  PHL     0.00330033
```

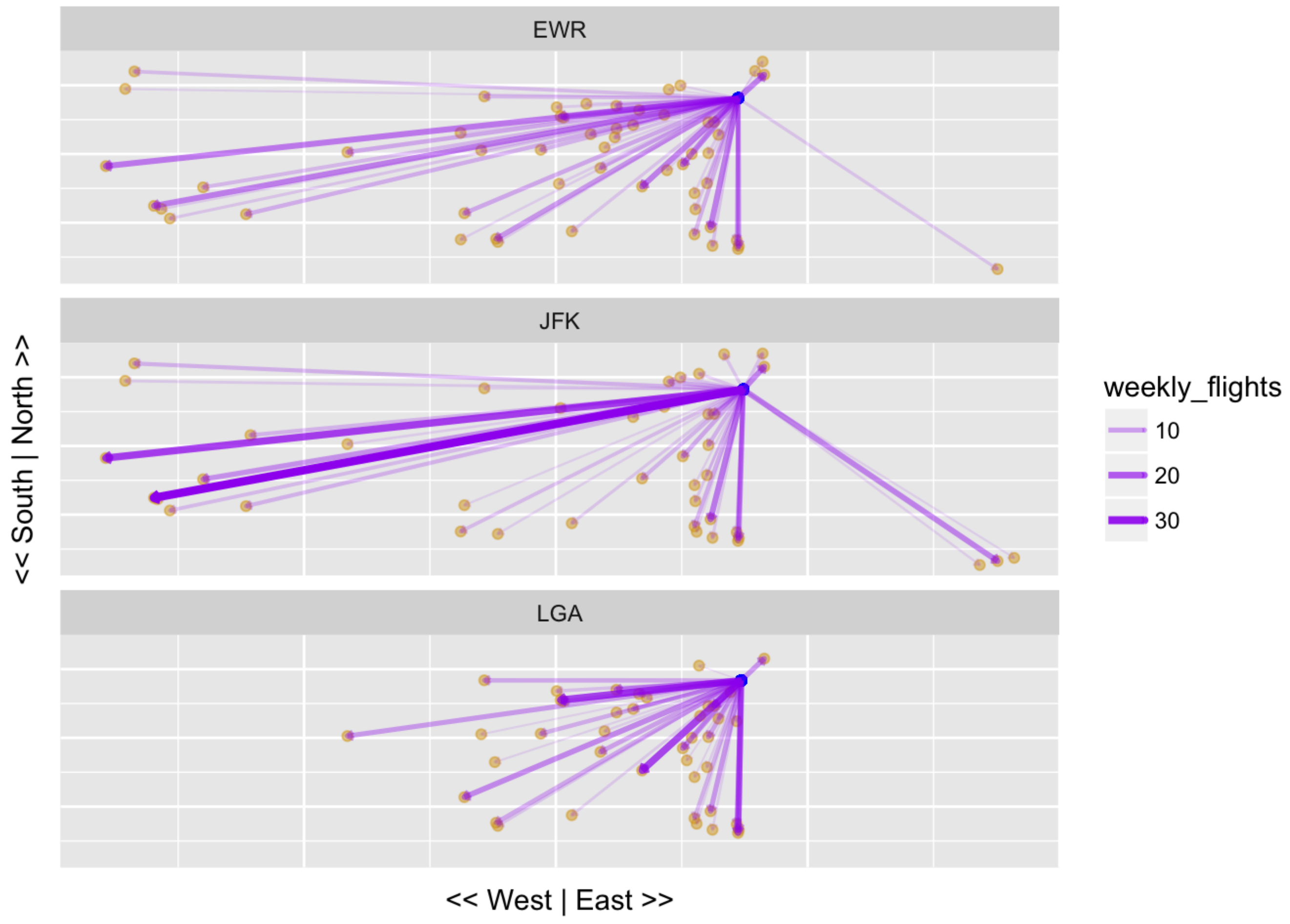
b. Display the data from part “a” as a network graph using the coordinate system from question 3, part “c”. Your display should show airports as circles and have (directed) edges from each origin airport to each destination airport. The thickness of the edges should be proportional to the number of weekly flights found in part “a”.

First, we will merge origin and destination coordinates into our data table with flights per week.

```
# get origin and destination coordinates
pd = as.data.table(plot_data)
setkey(pd,'airport')
org  = pd[c('EWR','JFK','LGA'),.(origin=airport,o_x=x,o_y=y)]
dest = pd[!c('EWR','JFK','LGA'),.(dest=airport,d_x=x,d_y=y)]

# merge coordinates into fpw
setkey(fpw,origin)
fpw = fpw[org]

setkey(fpw,dest)
fpw = fpw[dest]
```

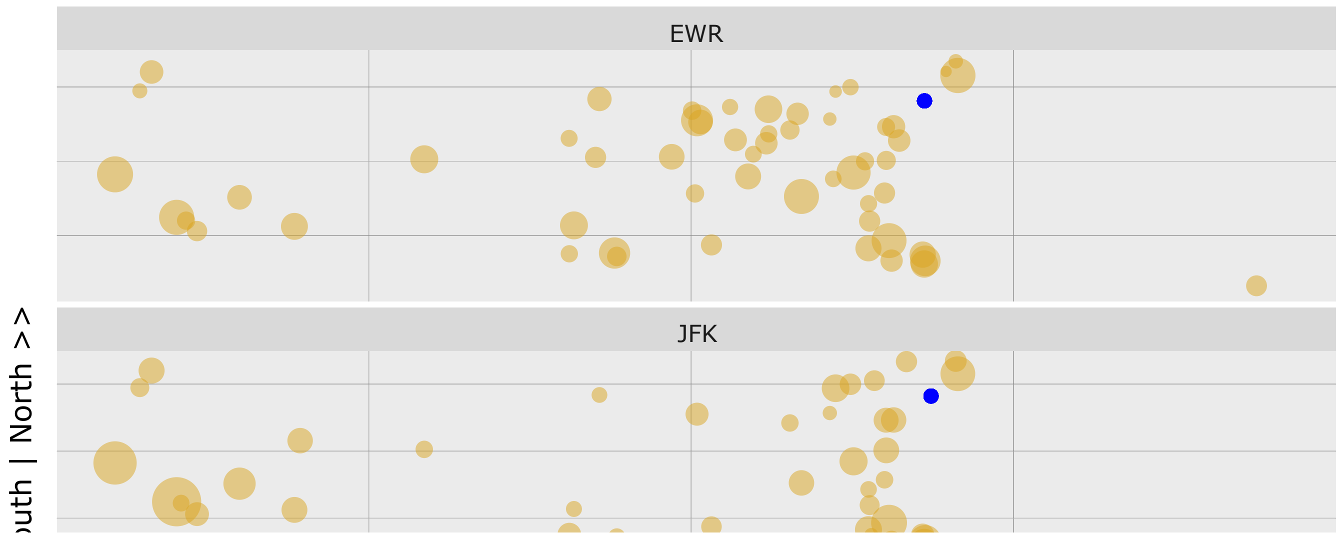
Now we are ready for plotting. To make the display more readable, our approach will be to show only the origin destination pairs averaging more than one flight per week.

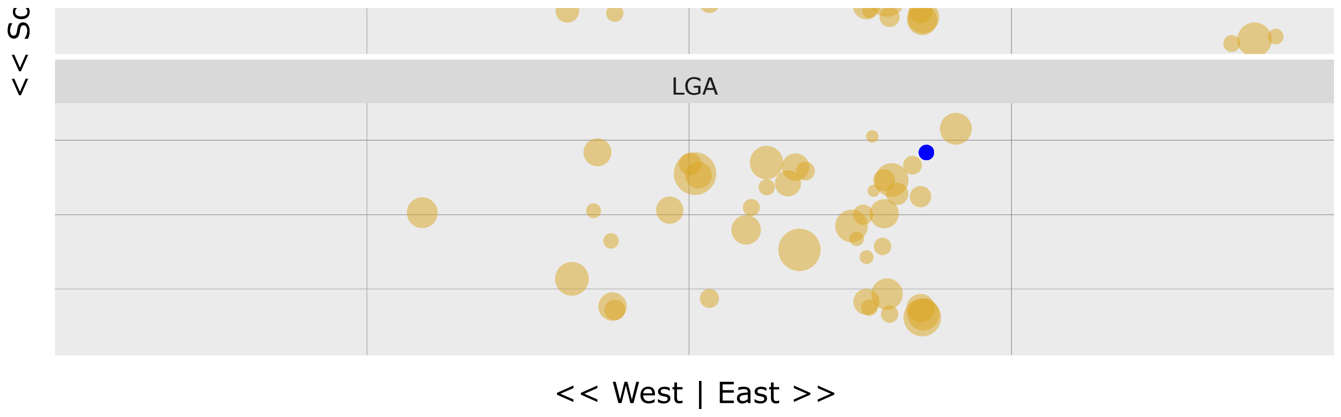
```
pwf0 =
  fpw[weekly_flights>1] %>%
  ggplot(aes(x=d_x, y=d_y, Name=dest)) +
  geom_point(color='goldenrod', alpha=.5) +
  geom_point(aes(x=o_x, y=o_y), color='blue') +
  geom_segment(aes(x=o_x, y=o_y, xend=d_x, yend=d_y, alpha = weekly_flights,
                   size=weekly_flights
                   ),
               color='purple', arrow=arrow(length = unit(0.03, "npc"))
               ) +
  scale_size(range = c(.25, 1.5)) +
  facet_wrap(~origin, dir='v') +
  xlab('<< West | East >>') +
  ylab('<< South | North >>') +
  theme(axis.text.x=element_blank(),
        axis.ticks.x=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank()
  )
pwf0
```



A bubble plot would have been a better option.

```
pwf =
  fpw[weekly_flights>1] %>%
  ggplot(aes(x=d_x, y=d_y, Name=dest)) +
  geom_point(color='goldenrod',alpha=.5,aes(size=weekly_flights)) +
  geom_point(aes(x=o_x, y=o_y),color='blue') +
  facet_wrap(~origin, dir='v') +
  xlab('<< West | East >>') +
  ylab('<< South | North >>') +
  theme(axis.text.x=element_blank(), axis.ticks.x=element_blank(),
        axis.text.y=element_blank(), axis.ticks.y=element_blank()
  )
ggplotly(pwf)
```





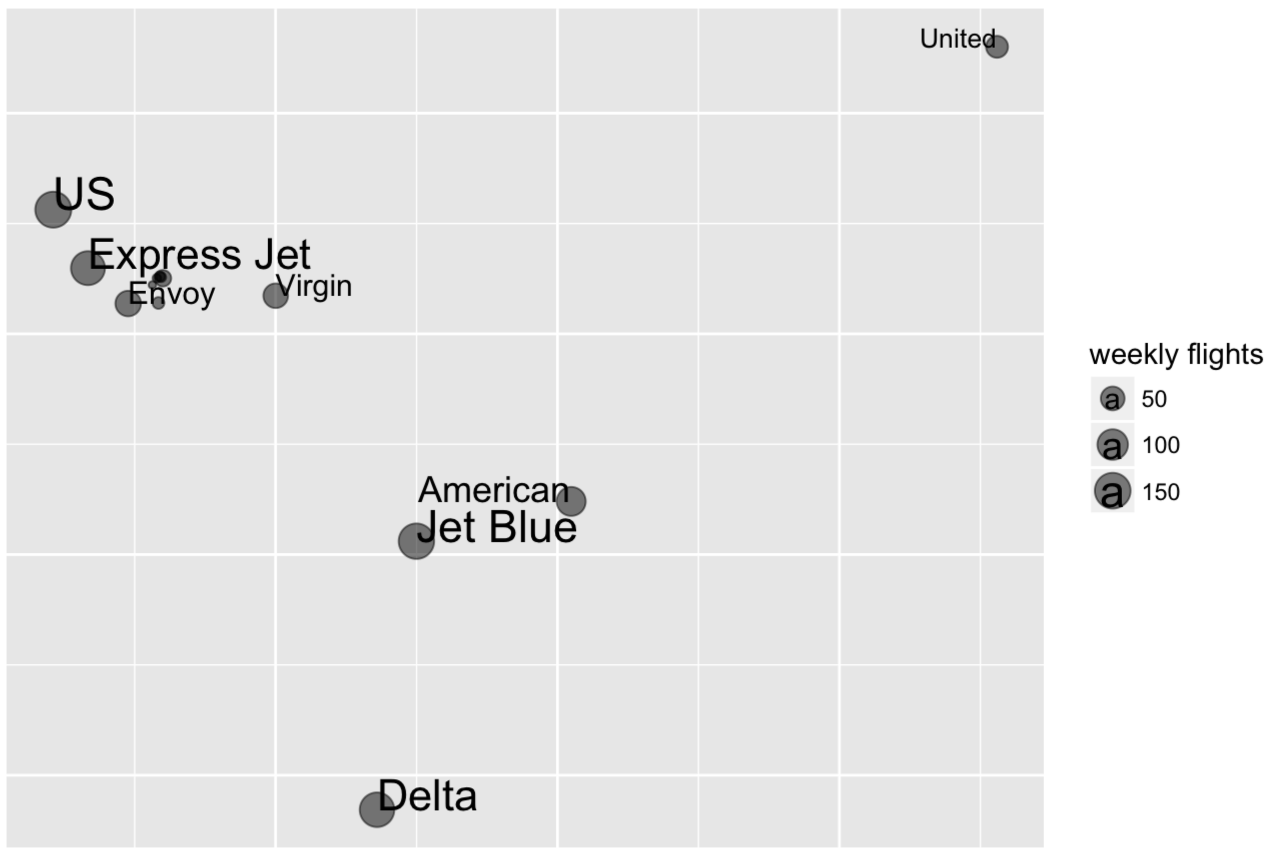
c. Repeat part "a" separately for each carrier. Then compute pairwise distances between carriers based on the frequency of flights between airports. Use MDS to create a 2-dimensional map of the carriers. Briefly discuss your findings.

```
# weekly flights per carrier
wfpc = nyc14[ , .(weekly_flights = .N/n_days), by=.(origin, dest, carrier)]

# reshape so each carrier is a column
wfpc_wide = dcast(wfpc, origin+dest~carrier, fill=0)
```

```
## Using 'weekly_flights' as value column. Use 'value.var' to override
```

```
# distances
wfpc_dist = dist(t(wfpc_wide[,-c("origin","dest")]))

# mds
wfpc_mds = cmdscale(wfpc_dist)

# total flights will make the plot more interesting
N = nyc14[, .(N=.N/n_days), by=.(carrier)]

# data for plotting
wfpc_data =  tibble(carrier = rownames(wfpc_mds), weekly_flights = round(N$N,1),
       x = wfpc_mds[,1], y=wfpc_mds[,2]
       )

# add labels for select airlines
wfpc_data = wfpc_data %>%
  mutate(label=sapply(carrier,function(x)
    switch(x,
           AA='American',
           B6='Jet Blue',
           DL='Delta',
           EV='Express Jet',
           MQ='Envoy',
           UA='United',
           US='US',
           VX='Virgin',
           NA
    ))
  )

# Here is the plotting code
wfpc_plot =
  wfpc_data %>% mutate(`weekly flights`=weekly_flights) %>%
  ggplot(aes(x=x, y=y, size=`weekly flights`, Name=carrier)) +
  geom_point(alpha=.5) +
  geom_text(aes(label=label),hjust='inward',vjust='bottom') +
  theme(axis.text.x=element_blank(),
        axis.ticks.x=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank()
  ) + xlab('') + ylab('') +
  ggtitle('Concept map for airlines serving NYC.')

wfpc_plot
```

Concept map for airlines serving NYC.



Any resaonble comments are fine. Some things you may notice are that: airlines with small volume are clustered together and similar to US, Envoy, Express Jet and Virgin; American and Jet Blue have a similar profile; and, Delta and United are the most unique.

d. Compute the average weekly number of flights for each carrier between all origin and destination airports. Repeat the visualization from part "c" after normalizing the frequency data to control for the average weekly number of flights by each carrier. Briefly dicsuss your findings and contrast with what you found in part "c".

The main point here is to control for differences in overall volume between carriers.

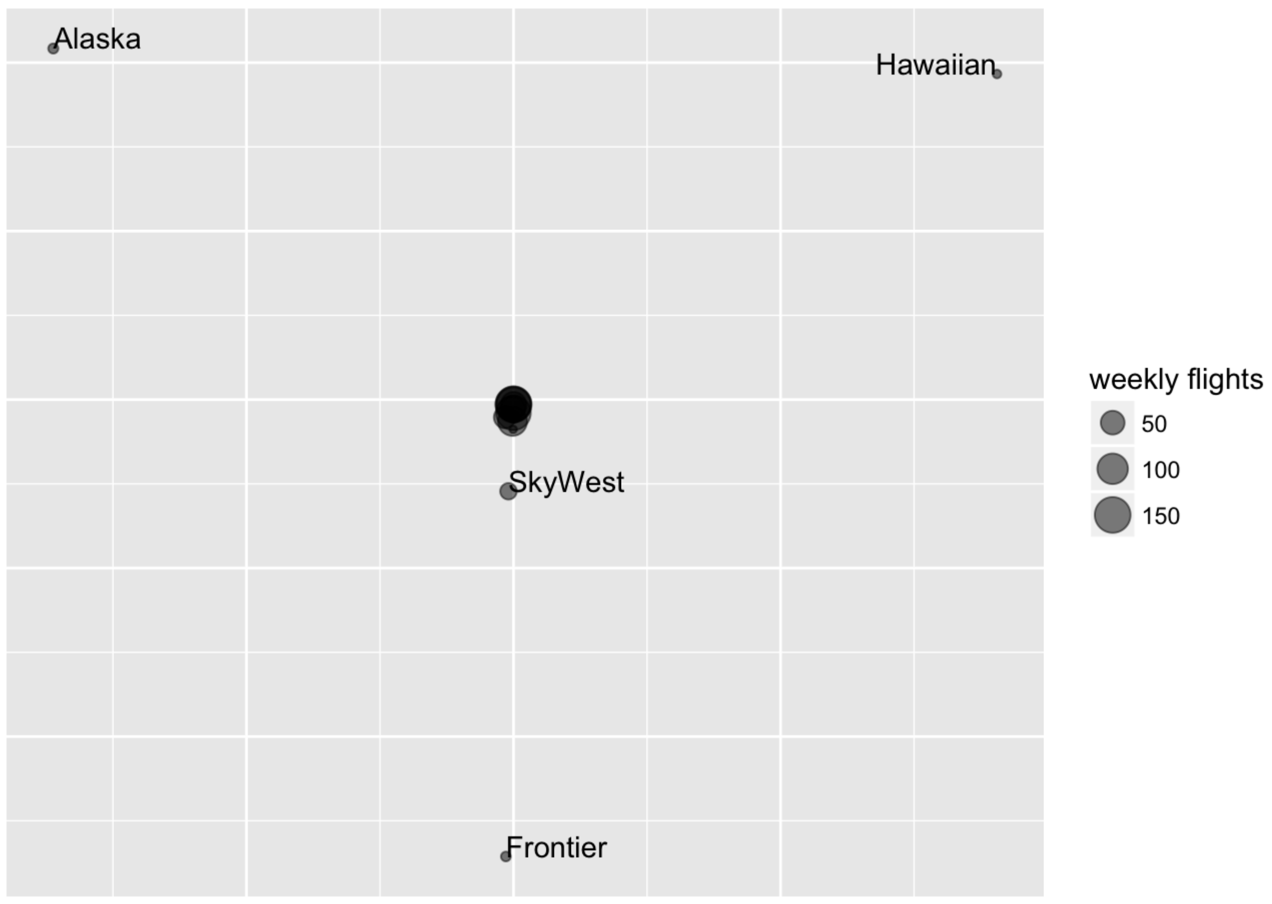
```
# Compute carrier volumes
N_carrier = nyc14[,.(N=.N/n_days), by=.(carrier)]
setkey(N_carrier,carrier)
N_carrier
```

```
##     carrier           N
##  1:      AA  86.8052805
##  2:      AS   1.8943894
##  3:      B6 146.7953795
##  4:      DL 137.5676568
##  5:      EV 131.4158416
##  6:      F9   1.5610561
##  7:      FL   4.1287129
##  8:      HA   0.8580858
##  9:      MQ  61.2508251
## 10:      OO   0.6600660
## 11:      UA 152.6963696
## 12:      US  55.2805281
## 13:      VX  15.8316832
## 14:      WN  39.2805281
```

```
# Merge into wfpc and compute proportion of flights by carrier
setkey(wfpc,carrier)
wfpc = wfpc[N_carrier]
pfpc = wfpc[,.(prop_flights = weekly_flights/N),by=.(origin,dest,carrier)]

# distance, mds, and plotting data
pfpc_wide = dcast(pfpc, origin+dest~carrier, fill=0)
```

```
## Using 'prop_flights' as value column. Use 'value.var' to override
```

```
pfpc_dist = dist(t(pfpc_wide[,-c("origin","dest")]))
pfpc_mds = cmdscale(pfpc_dist)

pfpc_data = wfpc_data %>% mutate(x=pfpc_mds[,1], y=pfpc_mds[,2])

pfpc_plot =
  pfpc_data %>%
  mutate(`weekly flights`=weekly_flights,
         label=sapply(carrier,function(x)
    switch(x,
           AS='Alaska',
           HA='Hawaiian',
           F9='Frontier',
           OO='SkyWest',
           NA
    ))
  ) %>%
  ggplot(aes(x=x, y=y, Name=carrier)) +
  geom_point(alpha=.5, aes(size=`weekly flights`)) +
  geom_text(aes(label=label),hjust='inward',vjust='bottom') +
  theme(axis.text.x=element_blank(),
        axis.ticks.x=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank()
  ) + xlab('') + ylab('')

pfpc_plot
```



From the plot you can see that most of the airlines are similar, with four smaller arlines standing out: Alaskan, Hawaiian, SkyWest, and Frontier. A quick look at the normalized data will reveal why:

```
head(pfpc[order(-prop_flights)])
```

```
##    origin dest carrier prop_flights
## 1:    EWR  SEA      AS    1.0000000
## 2:    JFK  HNL      HA    1.0000000
## 3:    LGA  DEN      F9    0.9873150
## 4:    LGA  ORD      OO    0.9450000
## 5:    LGA  ATL      FL    0.7258193
## 6:    JFK  LAX      VX    0.3281217
```

These airlines have all or nearly all of their flights connecting a single origin and destination pair (likely to their respective ‘hub’ cities.)

A reasonable next step would be to repeat the MDS exercise after filtering these carriers and presenting the two plots together.