CS109B Lab 3: Clustering Methods

Example background

This week we will look at seleceted data from Trump's infamous twitter account. In particular, people are curious to know which tweets are authored by the candidate himself, and which are authored by his staff.

Since the Trump campaign did not provide this information, Trump's twitter data is a good example of an *unsupervised* learning task. We don't *know* who authored each tweet, so there is no way to check our predictions to see how well we've done.

Nevertheless, we can make some headway on this question by examining any clusters that appear in the twitter data.

We will limit our field to tweets that occured in 2017.

Example data

Twitter provides a public API that can be used to retrieve tweets and the associated meta-data. Using this API Brendan Brown has been collecting and archiving Trump's twitter feed, and making the data available on github at https://github.com/bpb27/trump_tweet_data_archive (https://github.com/bpb27/trump_tweet_data_archive).

Downloading and extracting

Our first step is to download and extract the data.

```
## download.file("https://github.com/bpb27/trump_tweet_data_archive/raw/master/co
ndensed_2017.json.zip",
##               "trump_tweets_17.zip")
unzip("./trump_tweets_17.zip")
```

The data is provided in JSON format. There are a couple of different JSON readers available in R; I prefer the `jsonlite` package.

```
## install.packages("jsonlite")
library(jsonlite)
tweets <- fromJSON("./condensed_2017.json")
```

Data cleanup

As is typically the case, the twitter data is not in the most convenient format. The next step is to clean up and format the data in a more useful way.

Since not all the tweets originated with Trump's account we'll delete any retweets.

```
## just trump tweets
tweets <- tweets[!tweets$is_retweet, ]
```

Next we'll want to parse the date/time information and store it in meaningful units.

```
## convert timestamp to R date/time
## install.packages("lubridate")
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
tweets$date <- mdy_hms(paste(gsub("^[A-z]+ | \\d+:\\d+:\\d+ \\+0000|", "",
                                  tweets$created_at),
                             gsub("^.* (\\d+:\\d+:\\d+) .*$",
                                  "\\1",
                                  tweets$created_at)))

tweets$day <- yday(tweets$date)
tweets$weekday <- wday(tweets$date)
tweets$time <- hour(tweets$date) + minute(tweets$date)/60
```

We could do content analysis of the words in the tweets themselves, but a simpler approach is to pull out twitter-specific data from the content of the tweets. Specifically, we want to count the number of hashtags, mentions, and links.

```
tweets$hashes <- nchar(gsub("[^#]", "", tweets$text))
tweets$mentions <- nchar(gsub("[^@]", "", tweets$text))
tweets$link <- grepl("https*://", tweets$text)
```

Finally, the source of the tweet is another potentially useful variable. Most of the tweets come from an Android phone or and iPhone, with a handful coming from other sources. For convenience we'll combine devices that were used for a small proportion of the posts.

```
table(tweets$source)
```

```
##
##        Media Studio         Twitter Ads Twitter for Android
##                 116                  33                 281
##    Twitter for iPad  Twitter for iPhone  Twitter Web Client
##                  20                1798                  47
```

```
tweets[!tweets$source %in% c("Twitter for Android",
                             "Twitter for iPhone",
                             "Twitter Web Client"), ]$source <- "Other"

tweets$source <- factor(tweets$source)

summary(tweets)
```

```
##                 source        id_str              text
##  Other              : 169   Length:2295        Length:2295
##  Twitter for Android: 281   Class :character   Class :character
##  Twitter for iPhone :1798   Mode  :character   Mode  :character
##  Twitter Web Client :  47
##
##
##   created_at        retweet_count    in_reply_to_user_id_str
##  Length:2295        Min.   :     2   Length:2295
##  Class :character   1st Qu.: 12333   Class :character
##  Mode  :character   Median : 16658   Mode  :character
##                     Mean   : 19478
##                     3rd Qu.: 23356
##                     Max.   :369530
##  favorite_count   is_retweet           date
##  Min.   :     1   Mode :logical   Min.   :2017-01-01 05:00:10
##  1st Qu.: 58348   FALSE:2295      1st Qu.:2017-04-22 09:51:48
##  Median : 75848                   Median :2017-07-27 18:37:44
##  Mean   : 83865                   Mean   :2017-07-16 04:11:24
##  3rd Qu.:101146                   3rd Qu.:2017-10-13 20:05:28
##  Max.   :616217                   Max.   :2018-01-01 13:37:52
##       day           weekday           time           hashes
##  Min.   :  1.0   Min.   :1.000   Min.   : 0.00   Min.   :0.0000
##  1st Qu.:111.0   1st Qu.:3.000   1st Qu.:11.08   1st Qu.:0.0000
##  Median :208.0   Median :4.000   Median :13.22   Median :0.0000
##  Mean   :196.1   Mean   :4.229   Mean   :13.73   Mean   :0.1847
##  3rd Qu.:286.0   3rd Qu.:6.000   3rd Qu.:19.20   3rd Qu.:0.0000
##  Max.   :365.0   Max.   :7.000   Max.   :23.98   Max.   :4.0000
##     mentions        link
##  Min.   :0.000   Mode :logical
##  1st Qu.:0.000   FALSE:1572
##  Median :0.000   TRUE :723
##  Mean   :0.254
##  3rd Qu.:0.000
##  Max.   :6.000
```

Clustering in R

Base R includes functions useful for cluster analysis, including `dist` for calculating distances, `kmeans` for k-means clustering, and `hclust` for hierarchical clustering.

Many other clustering tools are available in contributed R packages. In fact there is a whole task view devoted to packages for cluster analysis, available at https://cran.rstudio.com/web/views/Cluster.html (https://cran.rstudio.com/web/views/Cluster.html). Of these, the `cluster`, `mclust`, `NbClust`, and `factoextra` packages are of particular interest.

The `cluster` package provides implementations of additional distance measures, as well as additional hierarchical and partitioning cluster algorithms. The `mclust` package provides implementations of model-based clustering algorithms. The `NbClust` package provides an a large number of methods for choosing the number of clusters. Finally the `factoextra` package provides convenient plotting for a wide range of clustering methods.

I will walk you through an example dataset and you will work on tackling the trump data set. The data set I will be using is from the package `cluster.datasets` which is full of useful toy datasets for clustering, that come from real world problems. The data `us.civil.war.battles` contains the Union and Confederate forces and numbers shot. Though there are some battles that are well known to be major and decisive, many others were more ambiuous. Let's see if we can any interesting information from this table of deployments and casualties.

```
#install.packages('cluster.datasets')
library(cluster.datasets)
data(us.civil.war.battles)
?us.civil.war.battles

#Seperate the names for later
civilWar<-us.civil.war.battles[,-1]
rownames(civilWar)<-us.civil.war.battles[,1]
```

Looking for clusters visually

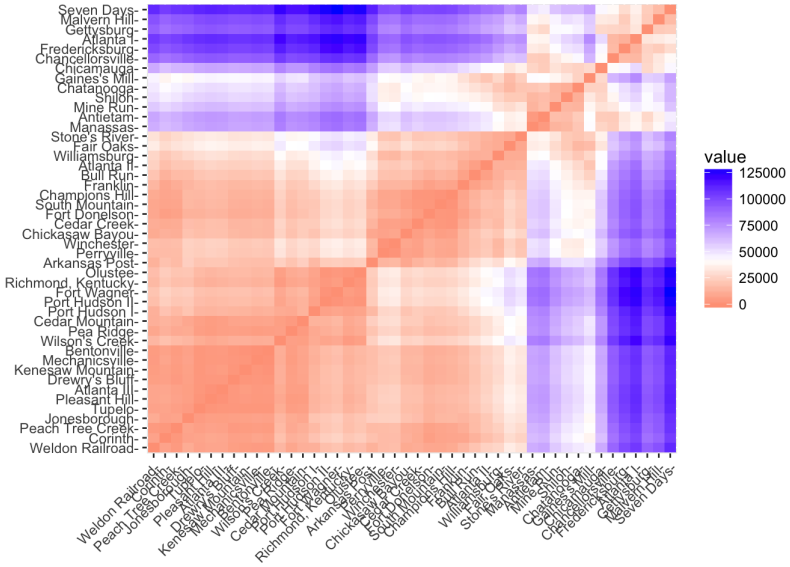
Visualizing Distance Matrix The function `fviz_dist` funciton in the `factoextra` package allows you to visualize a distance matrix. A distance matrix compares two data points through a numerical distance value (e.g. euclidean). The `daisy` function in the `cluster` packages allows you to easily calculate several types of distances. In `fviz_dist` the observations are somewhat sorted so that you are able to dillineate different groups within the observations.

```
library(cluster)
library(factoextra)
```

```
## Loading required package: ggplot2
```

```
## Welcome! Related Books: `Practical Guide To Cluster Analysis in R` at https://
goo.gl/13EFCZ
```

```
cw.dist<-daisy(civilWar,metric='euclidean',stand=F)
#takes about 2 mins....
fviz_dist(cw.dist)
```



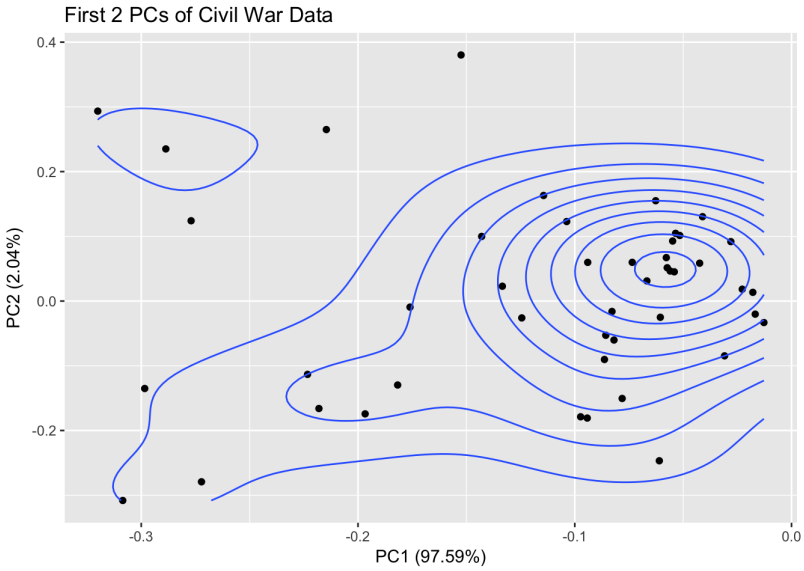
Principle Component Analysis (PCA)

Remember, PCA is useful in finding a low-dimensional representation of a numerical data set.

Visualizing multidimensional data as a whole is extremely difficult to do since a plot can show only two or three dimensions without getting too complicated. PCA gives us an easier way to visualize multidimensional data. The components are a vector representation in which each component holds information about all of the data. Therefore, by plotting the first two components we should be able to see some variability in the data points across all the factors.

This is particularly useful in clustering. We may start to see clusters form by simply looking at the first two components. The function `autoplot` in the `ggfority` package gives us an easy way to plot these components.

```
#install.packages('devtools')
#library(devtools)
#install_github('sinhrks/ggfortify')
library(ggfortify)

cw.pca<-prcomp(civilWar,scale=F,center=F)

autoplot(cw.pca)+
  geom_density2d()+
  ggtitle('First 2 PCs of Civil War Data')
```



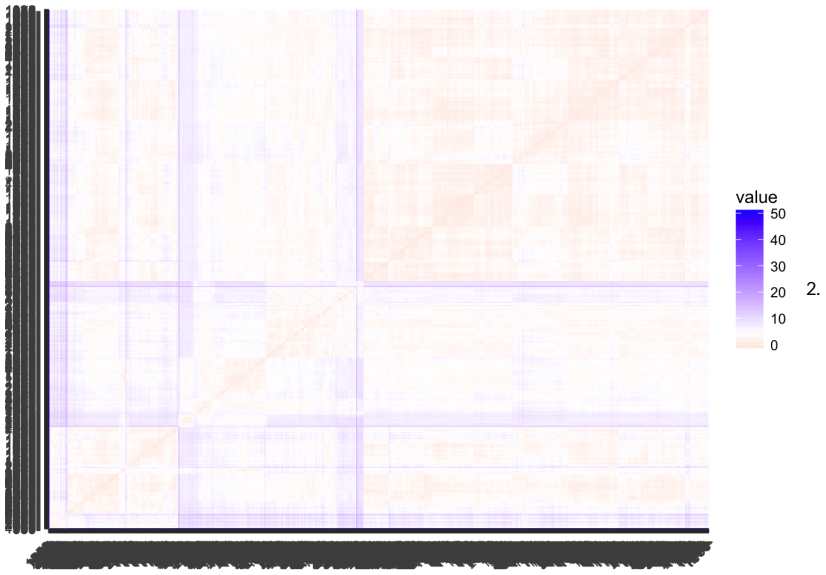
Your turn: Visualize clusters in Trump's numerical Twitter data

Using the numerical data from our tweet data set, let's see if we can visually distinguish any clusters. Later we will learn how to addres including categorical data in our clustering. The code snippet here will help you extrac the numerical data.

```
num.vars <- c("retweet_count", "favorite_count", "day", "weekday",
              "time", "hashes", "mentions")
tweet.num<-tweets[num.vars]
```

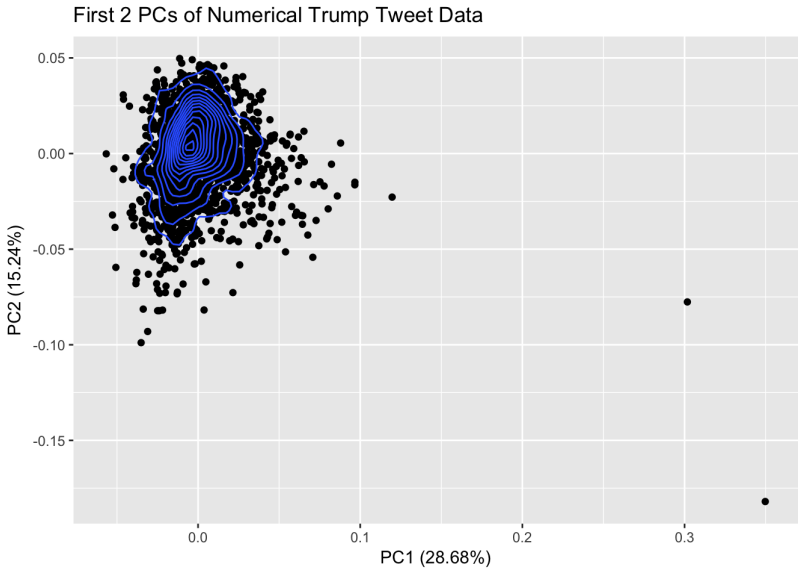
1. Calculate a distance matrix using euclidean distances for Trump's numerical Twitter data. Visualize this using the `fviz_dist` function (this takes a couple mins). How many clusters do you see?

```
tweet.dist<-daisy(tweet.num,metric='euclidean',stand=T)
#takes about 2 mins....
fviz_dist(tweet.dist)
```



2. Implement PCA on the numerical Twitter data. Visualize this using `autoplot`. How many clusters do you see?

```
tweet.pca<-prcomp(tweet.num,scale=T,center=T)

autoplot(tweet.pca)+
  geom_density2d()+
  ggtitle('First 2 PCs of Numerical Trump Tweet Data')
```



3. It is important to realize how much information we are reatining by looking at the first two PCs. Calculate the percent of variation we keep using the first two PCs. What does this mean?

```
pc.vars<-cumsum(tweet.pca$sdev/sum(tweet.pca$sdev))
pc.vars[2]
```

```
## [1] 0.3656822
```

In this case we retain 36%, which is pretty good but we must remain wary that most of the variation is not described by this plot.

Clustering Algorithms

Here we wil be implementing some of the algorithms discussed in class.

Partitioning Clustering

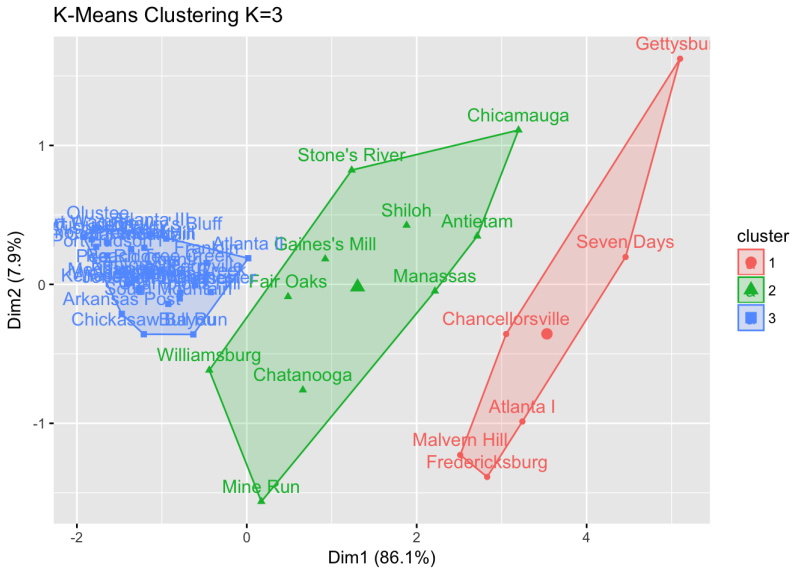
This type of clustering uses an iterative algorithm to find the center of the cluster as well as assign points to their respective cluser.

** K-means** :

- Cluster center is represented by the average of all points.

- Calculated in R using the function `kmeans`
- relatively quick (faster than k-mediods)
- Disadvantages: requires to select cluster in advanced, assignments are not globally optimal

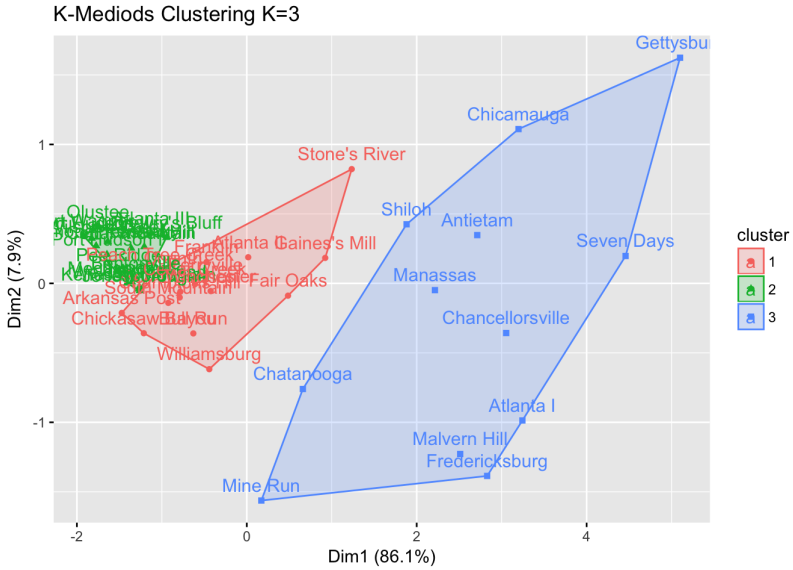
```
cw.kmeans<-kmeans(civilWar,3)
fviz_cluster(cw.kmeans,data=civilWar,main='K-Means Clustering K=3')
```



K-mediods:

- Cluster center is represented by the middle (medoid) observation of all points
- Calculated in R using function `pam`
- Advantages: robust to outliers and irregularly shaped clusters
- Disadvantages: requires to select cluster in advanced, computationally expensive

```
cw.pam<-pam(civilWar,k=3)
fviz_cluster(cw.pam,data=civilWar,main='K-Mediods Clustering K=3')
```



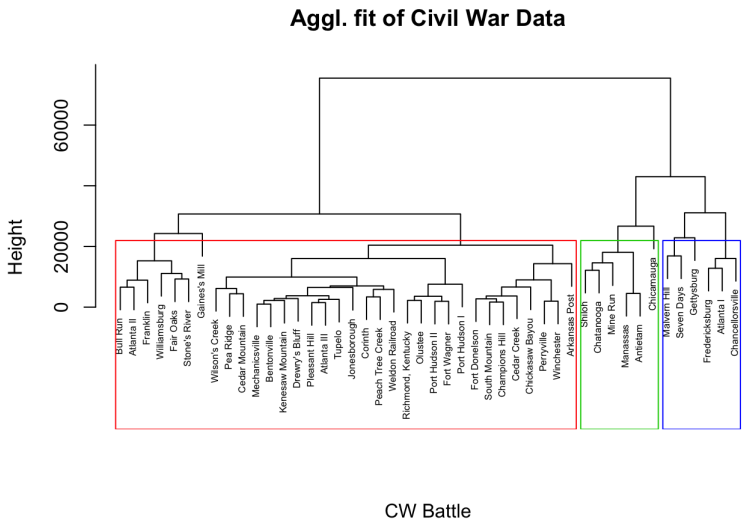
Hierarchical Clustering

Here we create a tree such that branches represent similar observations. To create clusters out of this tree, we find an optimal spot to cut the branches then those on the same branch will be of the same cluster.

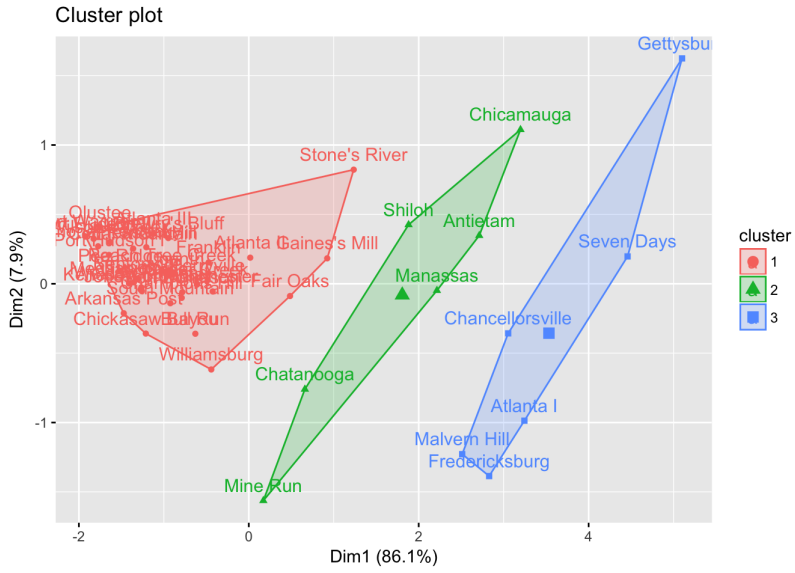
Agglomerative:

- Bottom up approach:Each observation starts as own cluster then gets paired together from there
- Use the `agnes` funciton in R to create tree,ues `cutree` funciton to get clusters from tree.

```
cw.agnes<-agnes(civilWar,method='average')
pltree(cw.agnes, cex=0.5,
       main='Aggl. fit of Civil War Data',
       xlab='CW Battle',sub='')
rect.hclust(cw.agnes,k=3,border=2:5)
```



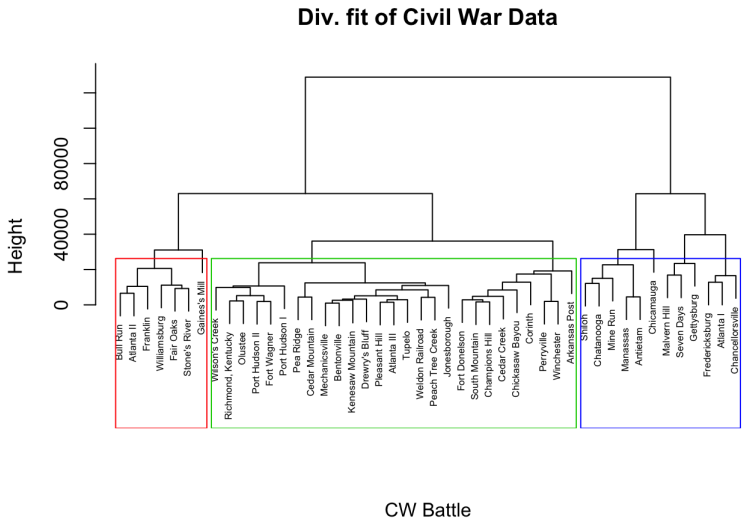

```
cw.grp.agnes<-cutree(cw.agnes,k=3)
fviz_cluster(list(data=civilWar,cluster=cw.grp.agnes))
```



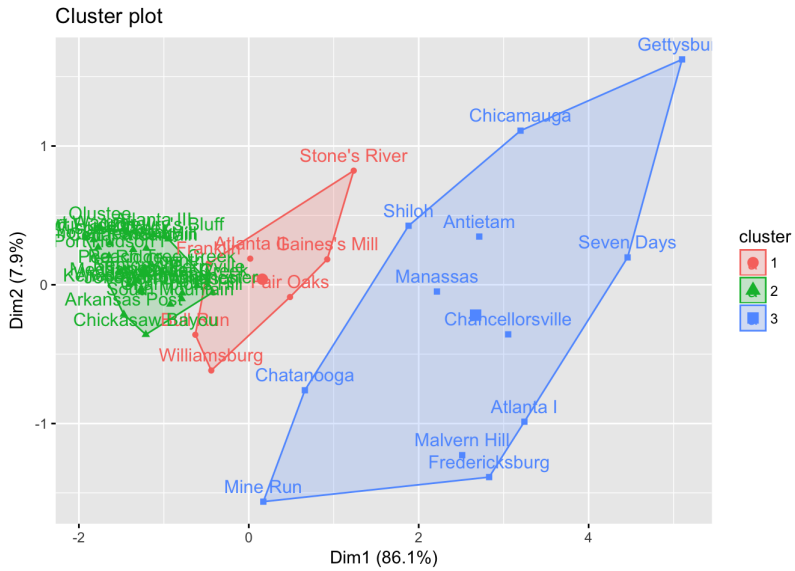
Divisive:

- Top down approach: Observations start as one cluster and seperates from there
- Use the `diana` funciton in R to create tree,ues `cutree` funciton to get clusters from tree.

```
cw.diana<-diana(civilWar)
pltree(cw.diana, cex=0.5,
       main='Div. fit of Civil War Data',
       xlab='CW Battle',sub='')
rect.hclust(cw.diana,k=3,border=2:5)
```




```
cw.grp.diana<-cutree(cw.diana,k=3)
fviz_cluster(list(data=civilWar,cluster=cw.grp.diana))
```



Linkages The linkage is the metric used in deciding wheter to split or merge clusters. Choosing the type of linkage you use can vary your results.

Some downfalls to consider:

- *Single linkage* can have problems with chaining, only need a single pair of points to be close to merge two clusters. Clusters can be too spread out.
- *Complete linkage* can have problems with crowding, point can be closer to points in other clusters than to points in its own cluster. Clusters are not seperated enough
- Cutting an *average linkage* tree has no interpretation
- Average linkage is sensative to transformations of distance.

Other linkages not covered in class include:

- Centroid linkage - measures the distance between group averages
- Minimax linkage - which furthest point has the smallest distance?

Soft Clustering

This is useful if we want to calculate a probability of an observation being in each cluster, rather than a discrete assignment. An easy way to get absolute cluster assignments is to chooose the cluster that has the highest probabity per observation.

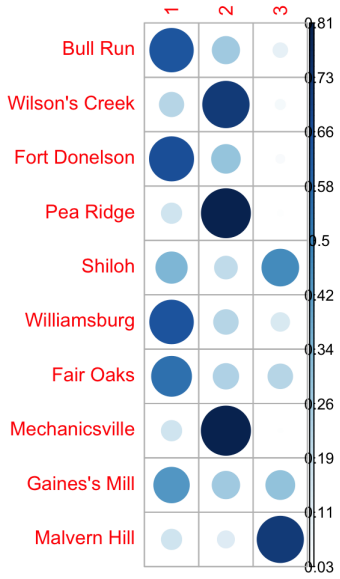
The two methods discussed in class are ** Fuzzy Clustering ** and ** Model Based Clustering **. Fuzzy clustering is implemented in R using the function `fanny` and model based clustering uses `Mclust`.

For fuzzy clustering you must choose a value for `memb.exp` that is strictly greater than one. If `memb.exp` is too low the clusters will be too seperated and membership in a cluster will be of probability 1. If it is too high cluster membership will resemble random assignments and therefore are uninformative.

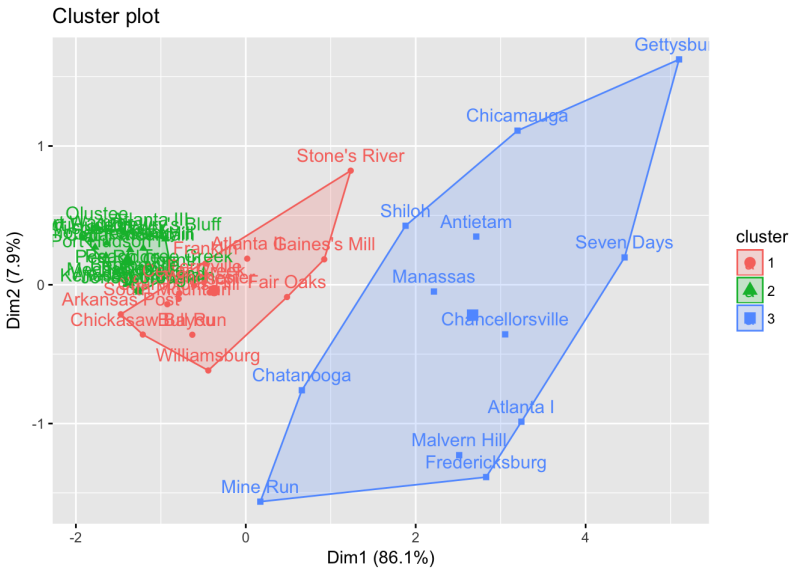
```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
cw.fanny<-fanny(civilWar,k=3,memb.exp=2)
corrplot(head(cw.fanny$membership,10),is.corr=F)
```




```
cw.grp.fanny<-cw.fanny$clustering
fviz_cluster(list(data=civilWar,cluster=cw.grp.fanny))
```

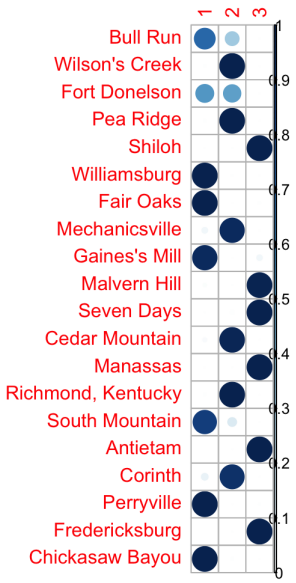


The `Mclust` has many variables that allow you to implemement the EM algorithm in many ways (such as including a prior or model used.) For now you do not have to worry about these. You must specify the variable G if you want a specific number of clusters.

```
library(mclust)
```

```
## Package 'mclust' version 5.4
## Type 'citation("mclust")' for citing this R package in publications.
```

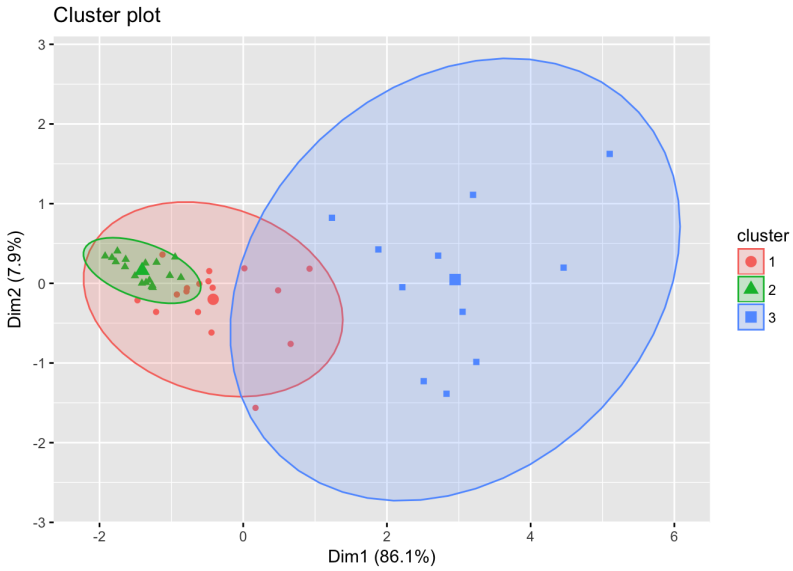
```
cw.mc<-Mclust(civilWar,G=3)
corrplot(head(cw.mc$z,20),is.corr=F)
```




```
fviz_cluster(cw.mc,frame.type='norm',geom='point')
```

```
## Warning: argument frame is deprecated; please use ellipse instead.
```

```
## Warning: argument frame.type is deprecated; please use ellipse.type
## instead.
```



Including categorical features

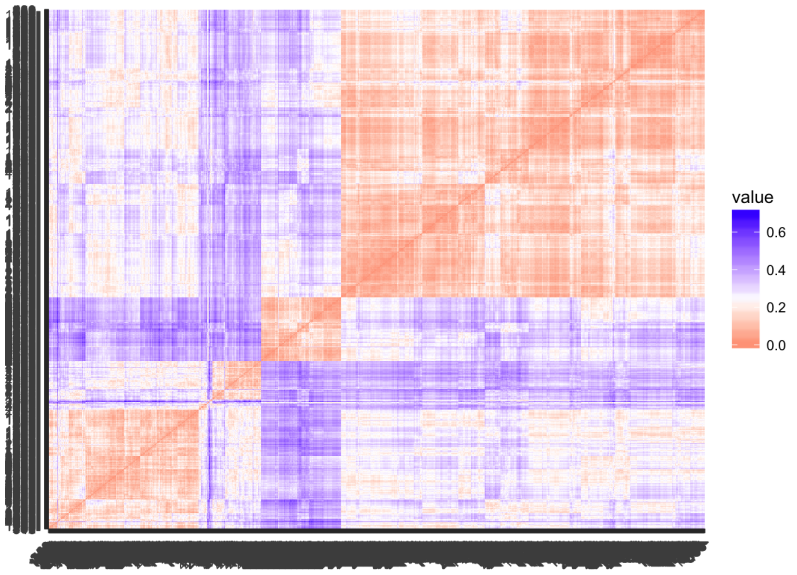
This data set contains both numerical and categorical factors. If we wish to use categorical features we'll need to switch to the `gower` measure of distance, implemented in the `daisy` function (from the `cluster` package).

We are able to `fviz_dist` do visualize the distance matrix.

```
cl.vars <- c("retweet_count","favorite_count", "source", "day", "weekday", "time"
, "hashes", "mentions", "link")
tweet.cat<-tweets[cl.vars]

tweets.dist.cat <- daisy(tweet.cat,
                         metric = "gower",stand=T)
```

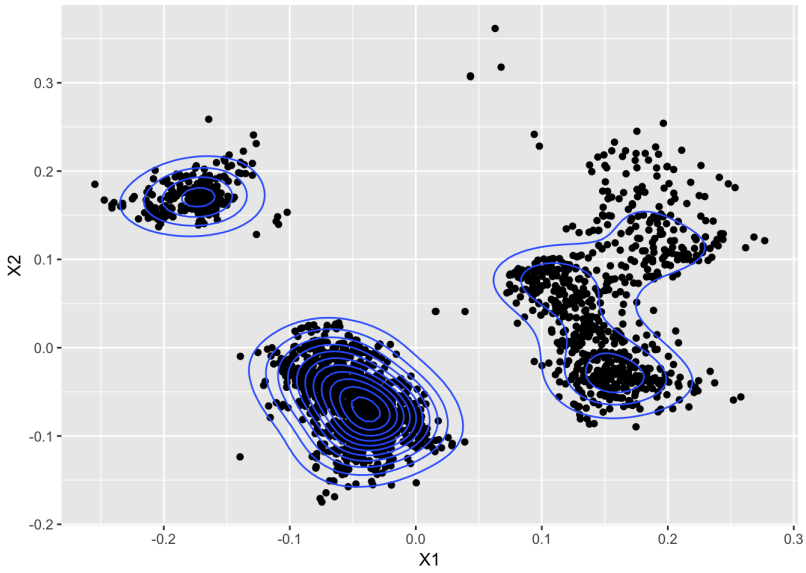
```
## Warning in daisy(tweet.cat, metric = "gower", stand = T): setting 'logical'
## variable 9 to type 'asymm'
```

```
fviz_dist(tweets.dist.cat) #takes ~ 2 mins to run
```



We might wish to use multidimensional scaling as another way of visually inspecting the data for clusters.

```
tweets.mds <- data.frame(tweet.cat,cmdscale(tweets.dist.cat))

ggplot(tweets.mds,
       mapping = aes(x = X1,
                     y = X2)) +
    geom_point()+
    geom_density2d()
```



In almost all of the clustering methods below, the data matrix can be substituted with a distance matrix. The functions for kmeans method and model based clustering are not compatible with categorical data (It's not straightforward to take an "average" of categorical data)

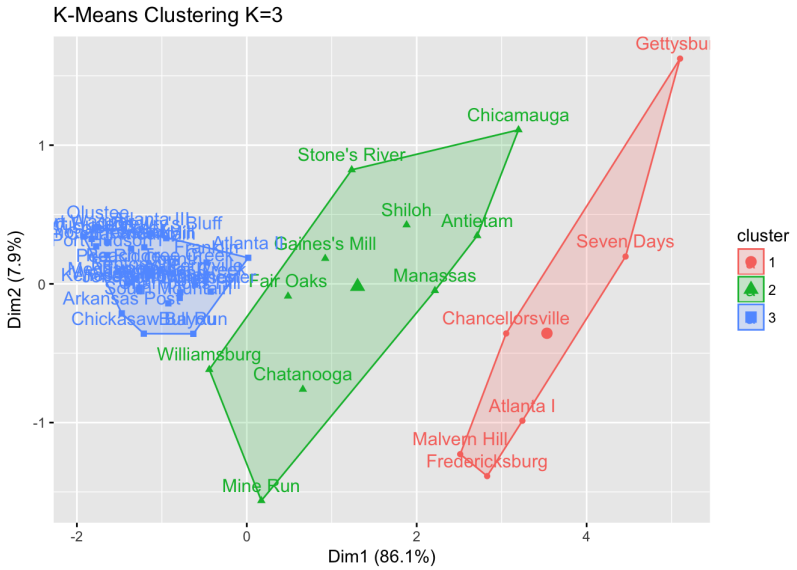
Interpreting the clusters

The outcome of clustering is not as concrete as, for example, classification or regression models. There is no *correct* answer. The first step is to justify the steps you took to build these clusters. The second step is to interperate these clusters. Good questions to ask:

- Are the distributions of any variable different between clusters?
- Is there any intuition gained by your clustering assigments?
- For fuzzy clustering, is there a difference between observations that are strongly in one group versus those between groups?

Let's look at the kmeans model of the Civil War Battles. This wikipedia article will come in handy for the interpretation: https://en.wikipedia.org/wiki/List_of_American_Civil_War_battles (https://en.wikipedia.org/wiki/List_of_American_Civil_War_battles)

```
fviz_cluster(cw.kmeans,data=civilWar,main='K-Means Clustering K=3')
```

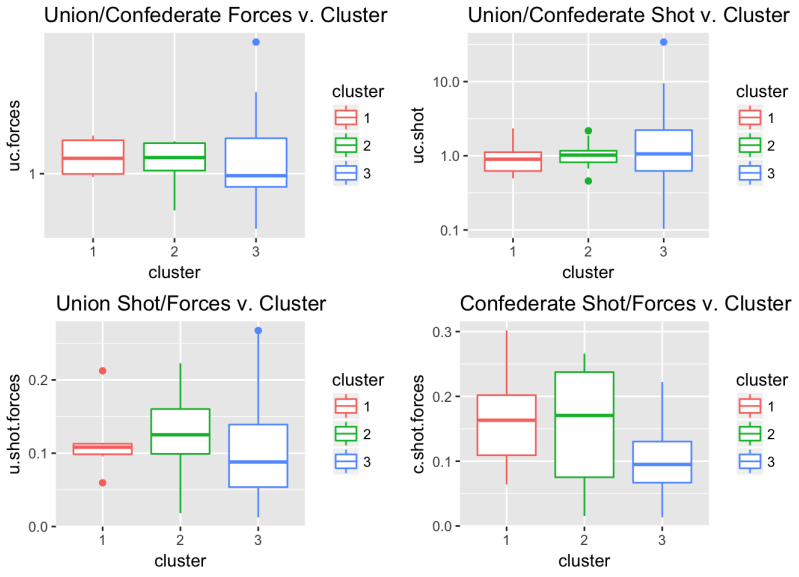


First let's see if there is a difference in the number of forces within the battles of the three groups.

```
#install.packages('gridExtra')
library(gridExtra)
cw.investigate.df<-data.frame(civilWar,cluster=factor(cw.kmeans$cluster))


# Union Forces
p.uf<-ggplot(aes(y=union.forces,x=cluster,col=cluster),data=cw.investigate.df)+
  geom_boxplot()+
  ggtitle('Union Forces v. Cluster')

# Confederate Forces
p.cf<-ggplot(aes(y=confederate.forces,x=cluster,col=cluster),data=cw.investigate.
df)+
  geom_boxplot()+
  ggtitle('Confederate Forces v. Cluster')

grid.arrange(p.uf,p.cf,nrow=1)
```

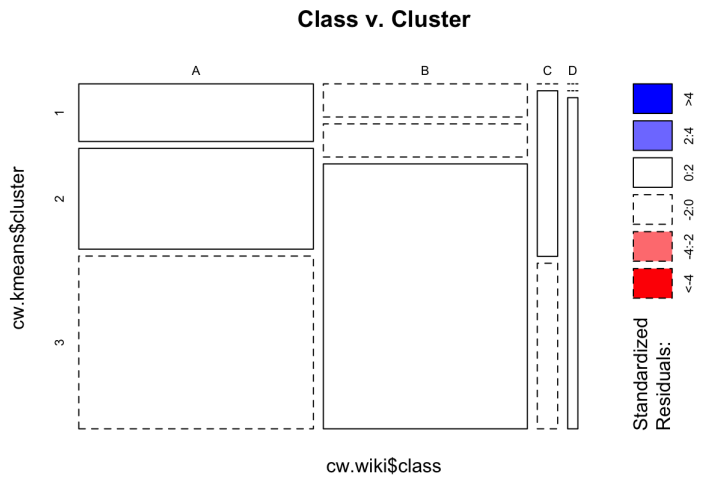
We suspect the number of people shot will correlate heavily with the number of forces with so let's transform the information and see if it tells a different story.

```
# Ratio of Union/Confederate Forces
cw.investigate.df<-data.frame(cw.investigate.df,
                      uc.forces=civilWar$union.forces/civilWar$confederate.force
s)

p.us.forces<-ggplot(aes(y=uc.forces,x=cluster,col=cluster),data=cw.investigate.d
f)+
  geom_boxplot()+
  scale_y_log10() +
  ggtitle('Union/Confederate Forces v. Cluster')

# Ratio of Union/ Confederate Shot
cw.investigate.df<-data.frame(cw.investigate.df,
                      uc.shot=civilWar$union.shot/civilWar$confederate.shot)

p.uc.shot<-ggplot(aes(y=uc.shot,x=cluster,col=cluster),data=cw.investigate.df)+
  geom_boxplot()+
  scale_y_log10() +
  ggtitle('Union/Confederate Shot v. Cluster')

# Ratio of Union shot / Union forces
cw.investigate.df<-data.frame(cw.investigate.df,
                      u.shot.forces=civilWar$union.shot/civilWar$union.forces)

p.u.shot.forces<-ggplot(aes(y=u.shot.forces,x=cluster,col=cluster),data=cw.invest
igate.df)+
  geom_boxplot()+
  ggtitle('Union Shot/Forces v. Cluster')

# Ratio of Confederate shot / Confederate forces
cw.investigate.df<-data.frame(cw.investigate.df,
                      c.shot.forces=civilWar$confederate.shot/civilWar$confederat
e.forces)

p.c.shot.forces<-ggplot(aes(y=c.shot.forces,x=cluster,col=cluster),data=cw.invest
igate.df)+
  geom_boxplot()+
  ggtitle('Confederate Shot/Forces v. Cluster')


grid.arrange(p.us.forces,p.uc.shot,p.u.shot.forces,p.c.shot.forces,nrow=2)
```

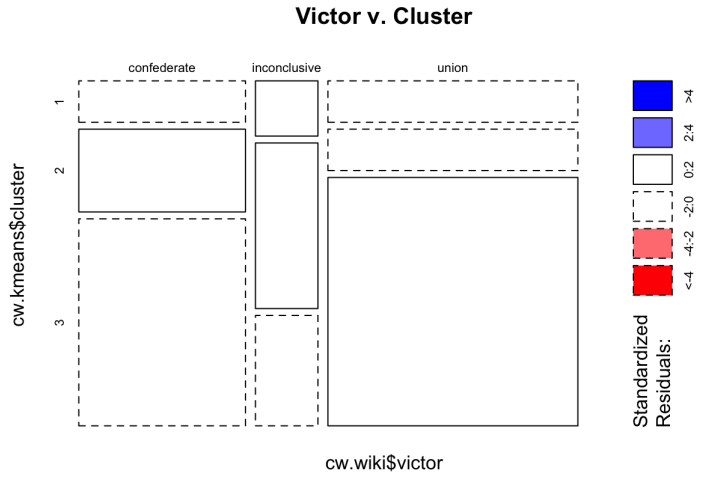


Using the wikipedia article, let's see if our hypothesis about the

```
cw.wiki<-read.csv('civilWar_wiki.csv',header=T)

mosaicplot(cw.wiki$class~cw.kmeans$cluster, shade=T, main='Class v. Cluster')
```




```
mosaicplot(cw.wiki$victor~cw.kmeans$cluster, shade=T,main='Victor v. Cluster')
```



Your turn: Apply clustering methods to Trump's Twitter data

For now let's assume there are 2 clusters since our hypothesis is that there are two sources of tweets. Next lab we will learn how to choose the optimal number of clusters.

REMINDER: Since our variables span across many units of measurment and scales we will want to center and scale our data before clustering.

1. Implement and visualize two different methods of clustering (partition, hierarchical, fuzzy, model based). Apply one to the numerical only data and one to the mixed numerical and categorical data. Explain why you chose each of the particular clustering methods.

```
#Method 1

#Method 2
```

2. Choose one of your methods from part 1. Compare the variables in each cluster (this is best done visually, but you can use `summary` as well.)

Revisit the variable `source`. Do you think that the tweets that come from an andriod phone are different than those that do not? Why are why not? (Hint: Check out `mosaicplot`)

3. Do you think 2 clusters is a good assumption for the number of clusters?

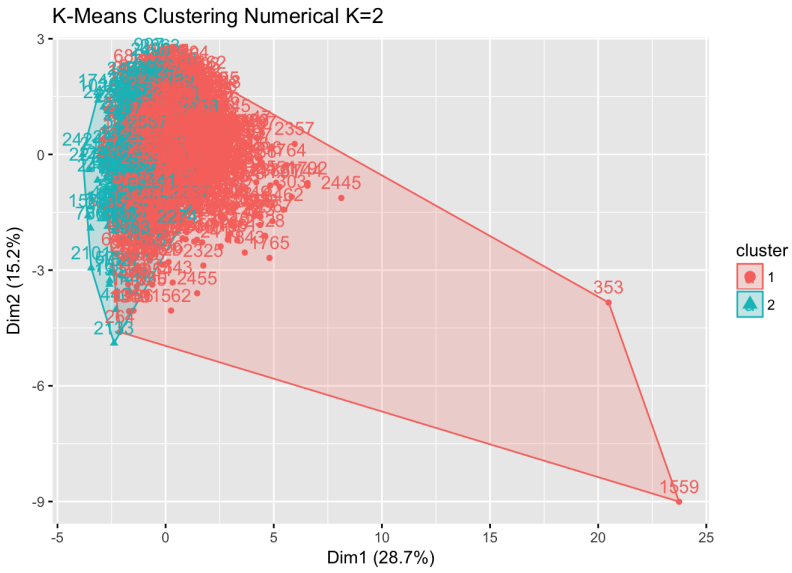
Example Solutions

1. Good explanations (but not limited to):

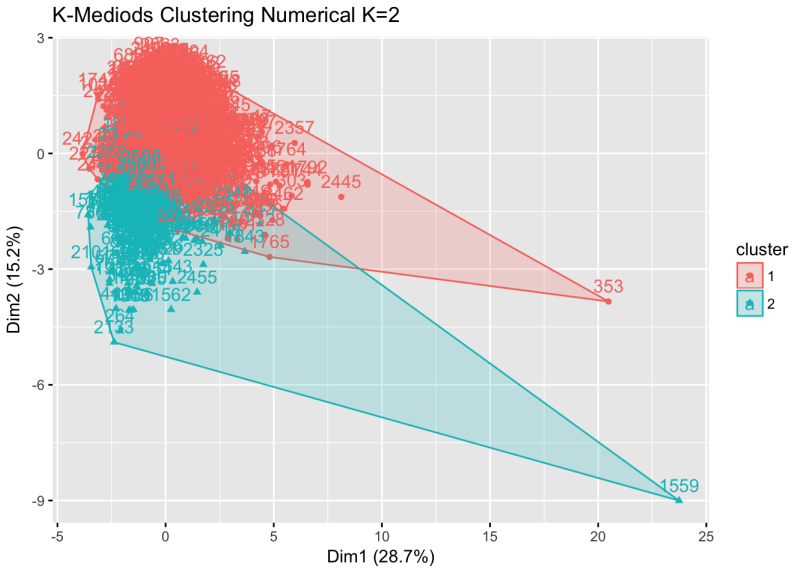
- PAM is robust to outliers, which is clear in the numerical data.
- Fuzzy/model based clustering is good since some tweets may be written by Trump and staff jointly.
- Kmeans cannot be used for numerical data so we use pam
- For Aggl. categorical clustering with single/complete linkage, clusters are not seperated enough (one relatively really small cluster that is not intuitive)

Examples of each: Examples for kmeans and pam on numerical data.

```
tweet.kmeans<-kmeans(scale(tweet.num),2)
fviz_cluster(tweet.kmeans,data=scale(tweet.num),main='K-Means Clustering Numerica
l K=2')
```



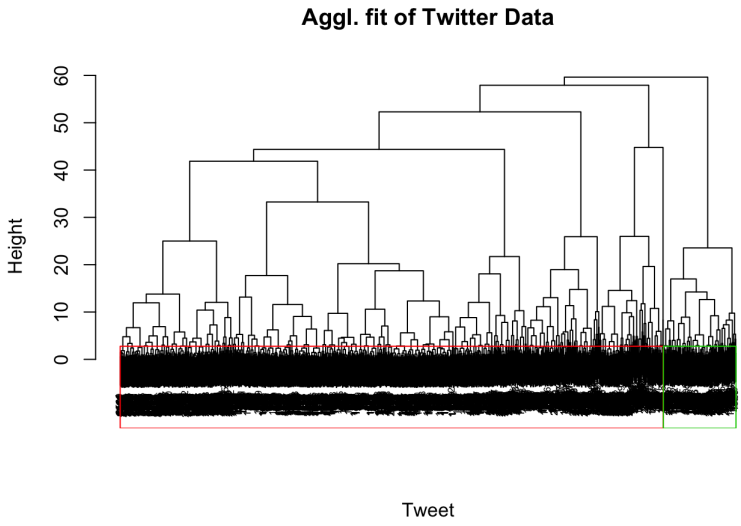

```
tweet.pam<-pam(scale(tweet.num),k=2)
fviz_cluster(tweet.pam,data=scale(tweet.num),main='K-Mediods Clustering Numerical
 K=2')
```



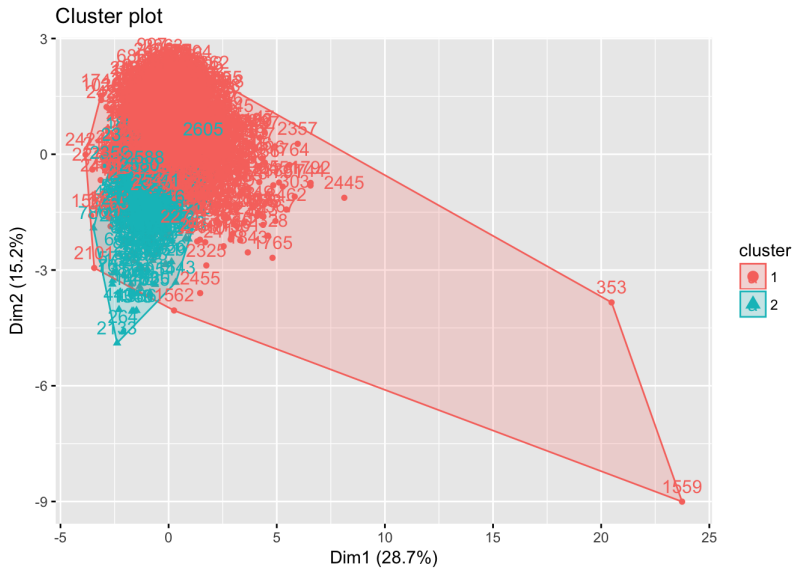
It is easy to see here that pam is more robust to outliers than k means.

Examples for agglomerative on numerical data and divisive on categorical and numerical data.

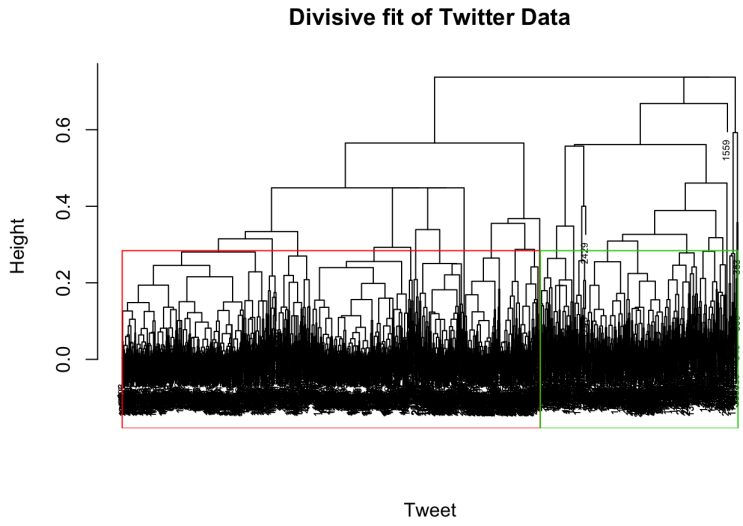
```
tweet.agnes<-agnes(scale(tweet.num),method='ward')
pltree(tweet.agnes, cex=0.5,
       main='Aggl. fit of Twitter Data',
       xlab='Tweet',sub='')
rect.hclust(tweet.agnes,k=2,border=2:5)
```




```
grp.agnes<-cutree(tweet.agnes,k=2)
fviz_cluster(list(data=scale(tweet.num),cluster=grp.agnes))
```



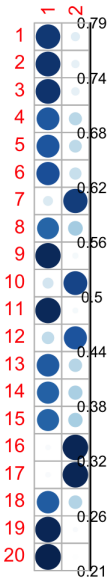

```
tweet.diana<-diana(tweets.dist.cat,diss=T)
pltree(tweet.diana, cex=0.5,
       main='Divisive fit of Twitter Data',
       xlab='Tweet',sub='')
rect.hclust(tweet.diana,k=2,border=2:5)
```




```
grp.diana<-cutree(tweet.diana,k=2)
tweet.df.diana<-data.frame(tweets.mds,cluster=factor(grp.diana))
ggplot(tweet.df.diana,
       mapping = aes(x = X1,
                     y = X2,col=cluster)) +
   geom_point()
```

Fuzzy clustering example on categorical data.

```
library(corrplot)
tweet.fanny<-fanny(tweets.dist.cat,k=2,memb.exp=1.5,diss=T)
corrplot(head(tweet.fanny$membership,20),is.corr=F)
```

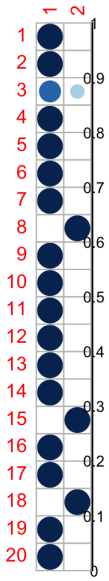



```
grp.fanny<-apply(tweet.fanny$membership,1,which.max)
tweet.df.fanny<-data.frame(tweets.mds,cluster=factor(grp.fanny))
ggplot(tweet.df.fanny,
       mapping = aes(x = X1,
                     y = X2,col=cluster)) +
   geom_point()
```

Here there are many tweets that have some probability of being in a different cluster than assigned. Investigating the differences between these and ones that are strongly clustered could lead to somem interesting insight. We do not puruse this in this lab.

Model based clustering.

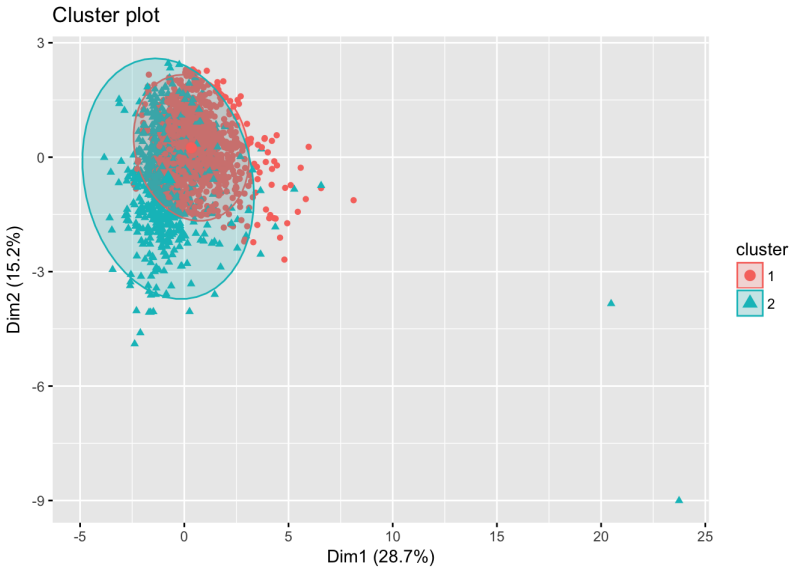
```
library(mclust)
tweet.mc<-Mclust(scale(tweet.num),G=2)
corrplot(head(tweet.mc$z,20),is.corr=F)
```




```
fviz_cluster(tweet.mc,frame.type='norm',geom='point')
```

```
## Warning: argument frame is deprecated; please use ellipse instead.
```

```
## Warning: argument frame.type is deprecated; please use ellipse.type
## instead.
```

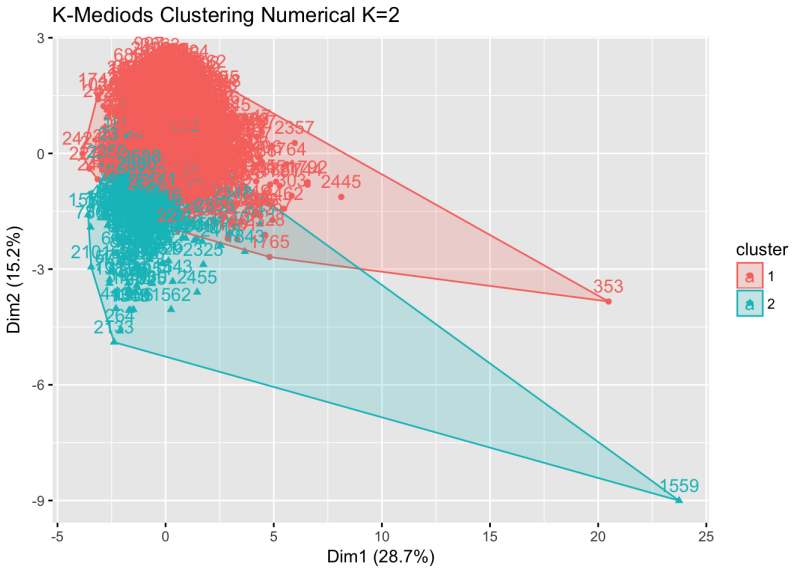


According to the corrplot, many of the tweets are strongly associated withe one cluster over another, though the PC plot shows a lot of over lap. This could be due to the fact that our PC plot only shows 36% of the variation in the data (see part 3 in Your Turn Part 1.)

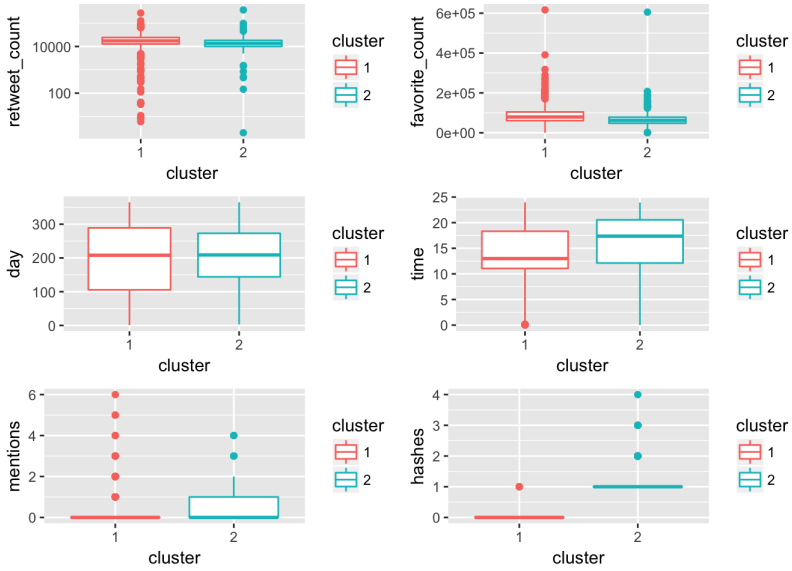
2. Example conclusion

We will used the results from PAM using numerical data to find evidence for our hypothesis. The blue color in the mosaic plot let's us know which assignments have less than expected (equal in both groups) and the red signifies which assignments have more than expected. Group 1 definitely appears to have more tweets sent from an android.

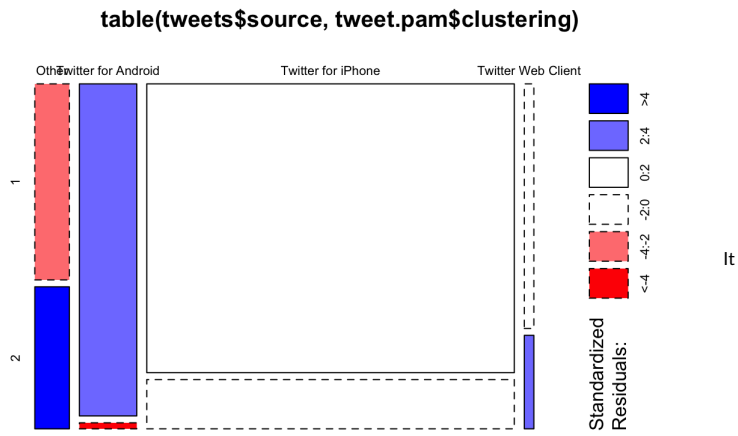
```
library(gridExtra)

fviz_cluster(tweet.pam,data=scale(tweet.num),main='K-Mediods Clustering Numerical
 K=2')
```




```
tweet.df.sum<-data.frame(tweet.num,cluster=as.factor(tweet.pam$clustering))
p.rt<-ggplot(aes(y=retweet_count,x=cluster,col=cluster),data=tweet.df.sum)+
  geom_boxplot()+
  scale_y_log10()

p.fc<-ggplot(aes(y=favorite_count,x=cluster,col=cluster),data=tweet.df.sum)+
  geom_boxplot()

p.d<-ggplot(aes(y=day,x=cluster,col=cluster),data=tweet.df.sum)+
  geom_boxplot()

p.t<-ggplot(aes(y=time,x=cluster,col=cluster),data=tweet.df.sum)+
  geom_boxplot()

p.h<-ggplot(aes(y=hashes,x=cluster,col=cluster),data=tweet.df.sum)+
  geom_boxplot()

p.m<-ggplot(aes(y=mentions,x=cluster,col=cluster),data=tweet.df.sum)+
  geom_boxplot()

grid.arrange(p.rt,p.fc,p.d,p.t,p.m,p.h,
             nrow=3)
```




```
mosaicplot(table(tweets$source,tweet.pam$clustering),shade=T)
```



It appears that cluster 1 has a slightly larger tweet count and favorite count. Cluster 2 tends to be later in the day as well as have slightly more hases and mentions. According to our mosaic plot, clster one has significantly more tweets from an Andriod phone while cluster 2 has significantly more from another source.

3. Based on our hypothesis and the PCA image of data, it appears reasonable to assume that there are only two clusters. Looking at the distance matrix it appears there may be up to 4 clusters.

Lab Survey

Please fill this out at the end of lab each week. This will be anonymous and will NOT be used for attendance. At the end you will have the opporutnity to leave any unanswered questions. If enough people ask the same question I will post an answer in supplementary material.

https://goo.gl/forms/NzuMN3IuAOHNPLSm2 (https://goo.gl/forms/NzuMN3IuAOHNPLSm2)