HUDK4050: Class Activity 6 Response

Timothy Lee

19/10/2019

Data Management

```
library(tidyverse)
```

```
## -- Attaching packages -------------------------------------------------------------------------------------

## v ggplot2 3.2.1          v purrr   0.3.2
## v tibble  2.1.3          v dplyr   0.8.3.9000
## v tidyr   1.0.0          v stringr 1.4.0
## v readr   1.3.1          v forcats 0.4.0

## -- Conflicts ----------------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
#Load data
DF1 <- read.csv("HUDK405019-clustering.csv", header = TRUE)

#Convert the index numbers of the data frame into the student names.

DF2 <- unite(DF1, col = "Name", 1:2, sep = "_")
rownames(DF2) <- DF2$Name
```

```
#Wrangle data using dplyr to include only the numerical values.
library(varhandle)
library(purrr)

DF3 <- DF2 %>%
        select("months_in_NYC" = How.many.months.have.you.lived.in.New.York.City.,
               "siblings" = How.many.siblings..brothers.sisters..do.you.have.,
               "sport_per_week" = How.many.times.do.you.play.sport.each.week.,
               "miles_from_home" = How.many.miles.do.you.travel.from.home.to.TC.,
               "android_friends" = Estimate.how.many.of.your.friends.own.Android.phones,
               "movies_per_year" = How.many.movies.have.you.seen.in.the.cinema.this.year.,
               "pets" = How.many.pets.have.you.owned.in.your.life.,
               "people_met" = How.many.people.have.you.met.for.the.first.time.this.year.,
               "cook_per_week" = How.many.time.do.you.cook.for.yourself.each.week.,
               "class_load" = How.many.classes.are.you.taking.this.semester.,
               "states_visited" = How.many.states.have.you.visited.in.the.US.,
               "latitude" = What.is.the.latitude.of.the.city.town.you.grew.up.in...Look.up.on.a.map.ser
               "longitude" = What.is.the.longitude.of.the.city.town.you.grew.up.in.
               ) %>%
        mutate("miles_from_home" = as.numeric(unfactor(miles_from_home))) %>%
        mutate("months_in_NYC" = as.numeric(unfactor(months_in_NYC))) %>%
        mutate("android_friends" = as.numeric(unfactor(android_friends))) %>%
        mutate("states_visited" = as.numeric(unfactor(states_visited)))
```

```
## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion
```

```
#Remove spaces
DF3$latitude <- gsub("[[:space:]]", "", DF3$latitude)
DF3$longitude <- gsub("[[:space:]]", "", DF3$longitude)

#Remove non-numbers

DF3$latitude <- gsub("[^[:digit:].]", "", DF3$latitude)
DF3$longitude <- gsub("[^[:digit:].]", "", DF3$longitude)

#Get first 3 characters
DF3$latitude <- substring(DF3$latitude, first = 1, last = 3)
DF3$longitude <- substring(DF3$longitude, first = 1, last = 3)

#convert to numeric

DF3$latitude <- as.numeric(DF3$latitude)
DF3$longitude <- as.numeric(DF3$longitude)

#Scale the data so that no variable has undue influence

DF4 <- DF3 %>% scale() %>% as.data.frame()

#The algorithm can't accept NAs, so we need to either remove them or change them to be the average of v
#I've opted for the latter approach.

col_means <- map_dbl(DF4, mean, na.rm = TRUE) %>% as.list()
DF4 <- replace_na(DF4, replace = col_means)
```

Find lattitudes & longitudes for cities

```
#Unfortunately Google has restricted access to the Googple Maps API so the code below no longer works.

#install.packages("ggmap")
#install.packages("rgdal")
#library(ggmap)
#library(tmaptools)

#Request lattitude and longitude from Google Maps API
#DF2 <- geocode(as.character(DF2$Q1_1), output = "latlon", source = "dsk")
```

Now we will run the K-means clustering algorithm we talked about in class. 1) The algorithm starts by randomly choosing some starting values 2) Associates all observations near to those values with them 3) Calculates the mean of those clusters of values 4) Selects the observation closest to the mean of the cluster 5) Re-associates all observations closest to this observation 6) Continues this process until the clusters are no longer changing

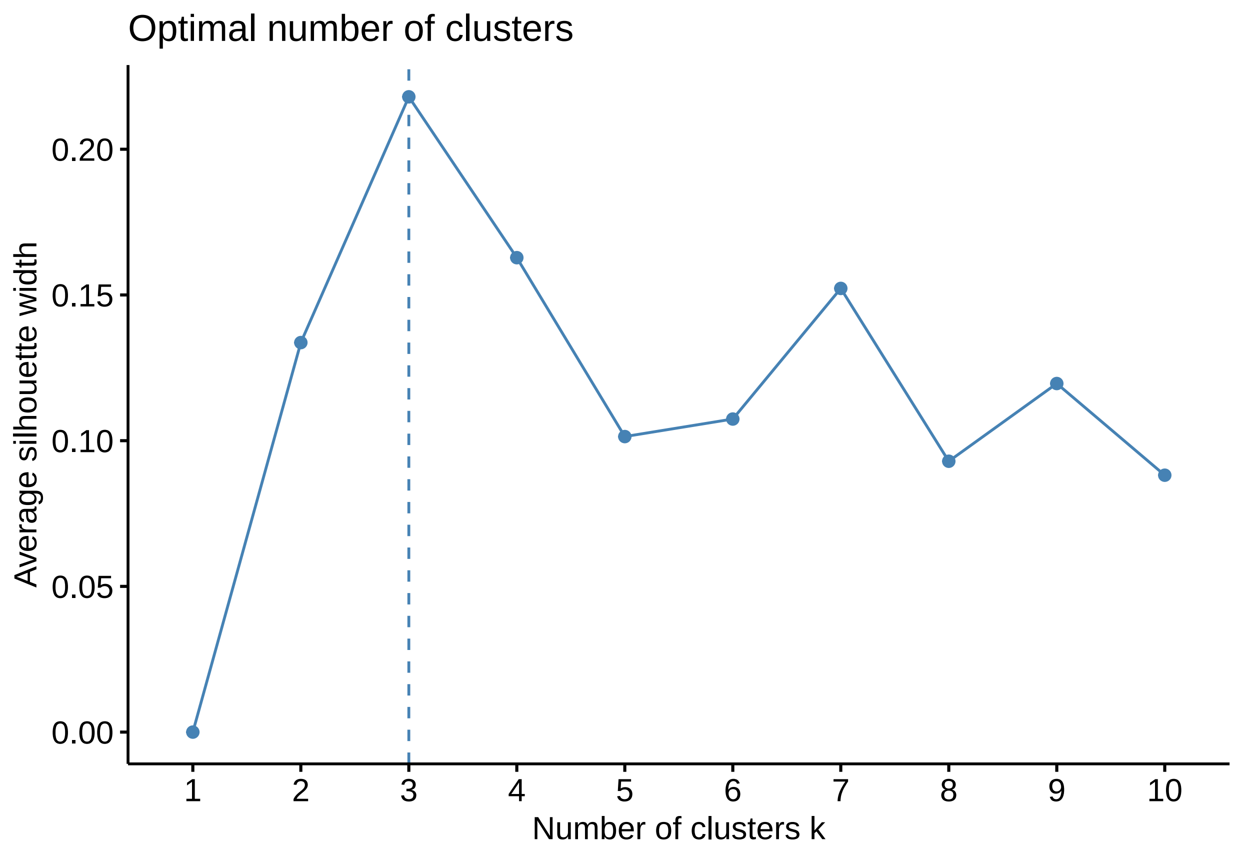
Notice that in this case we have 10 variables and in class we only had 2. It is impossible to vizualise this process with 10 variables.

Also, we need to choose the number of clusters we think are in the data. We will start with 4.

```
#Choosing number of factors using elbow method
library(factoextra)
```

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

```
fviz_nbclust(DF4, kmeans)
```



```
#Computing K means clustering with 5 clusters

fit <- kmeans(DF4, 5)

#We have created an object called "fit" that contains all the details of our clustering including which
#Examining the structure of fit
glimpse(fit)
```

```
## List of 9
##  $ cluster     : int [1:50] 5 4 2 4 4 2 2 4 2 2 ...
##  $ centers     : num [1:5, 1:13] -0.283 -0.314 -0.336 -0.23 2.586 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:5] "1" "2" "3" "4" ...
##   .. ..$ : chr [1:13] "months_in_NYC" "siblings" "sport_per_week" "miles_from_home" ...
##  $ totss       : num 629
##  $ withinss    : num [1:5] 0 189.9 60.4 106.7 37.4
##  $ tot.withinss: num 394
##  $ betweenss   : num 235
##  $ size        : int [1:5] 1 25 4 15 5
##  $ iter        : int 2
##  $ ifault      : int 0
##  - attr(*, "class")= chr "kmeans"
```

```
#We can access the list of clusters by typing "fit$cluster", the top row corresponds to the original or

fit$cluster
```

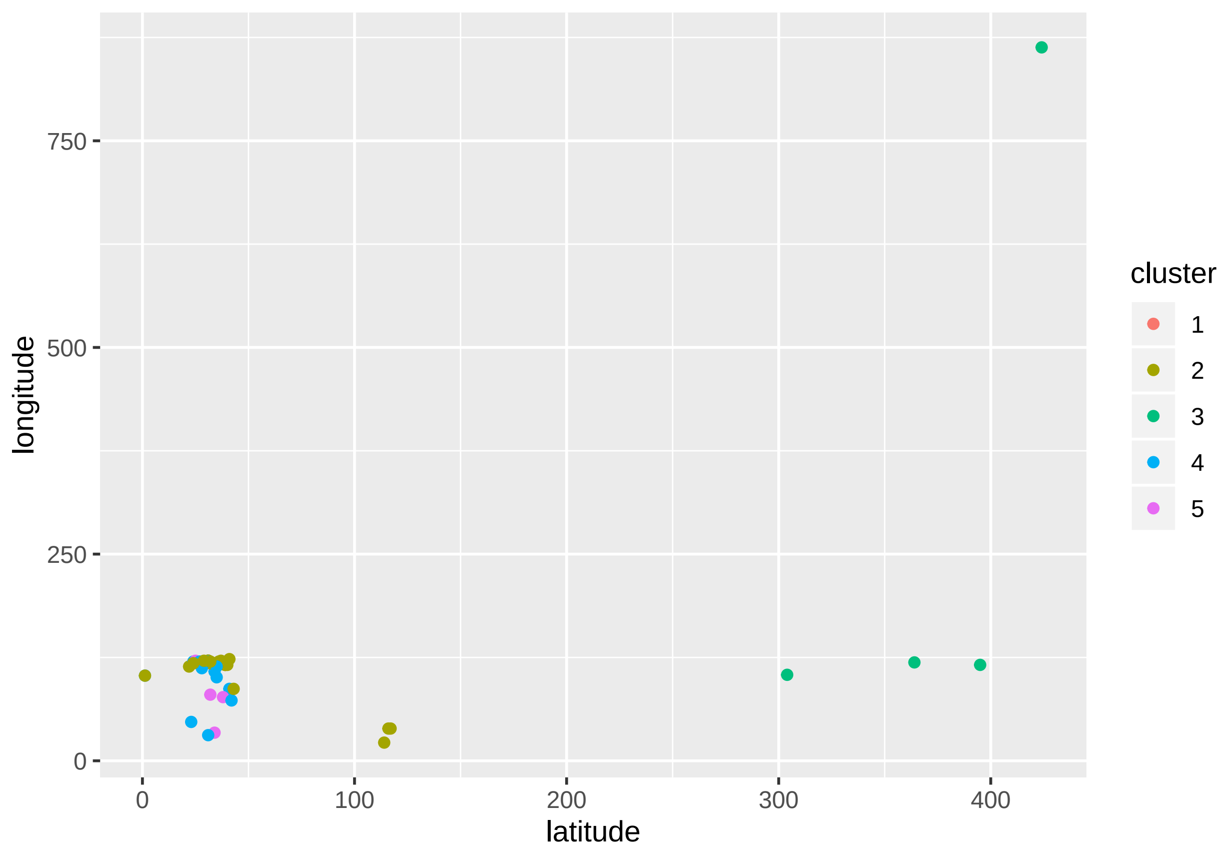
```
##  [1] 5 4 2 4 4 2 2 4 2 2 1 2 4 3 2 5 4 2 2 3 2 4 2 2 2 4 4 2 2 2 2 2 4 2 4
## [36] 5 5 5 4 2 4 3 4 4 2 2 2 2 2 3
```

```
#We can also attach these clusters to the original dataframe by using the "data.frame" command to creat

K4 <- data.frame(DF3, fit$cluster) %>%
        rename(cluster = fit.cluster)
K4$cluster <- as.factor(K4$cluster)

#Have a look at the DF3 dataframe. Lets change the names of the variables to make it more convenient wi

names(DF3) <- c("1", "2", "3", "4", "5", "cluster") #c() stands for concatonate and it creates a vector
```

Visualize your clusters in ggplot

```
#Create a scatterplot that plots location of each student and colors the points according to their clus
ggplot(K4, aes(x = latitude, y = longitude, color = cluster)) + geom_point()
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```

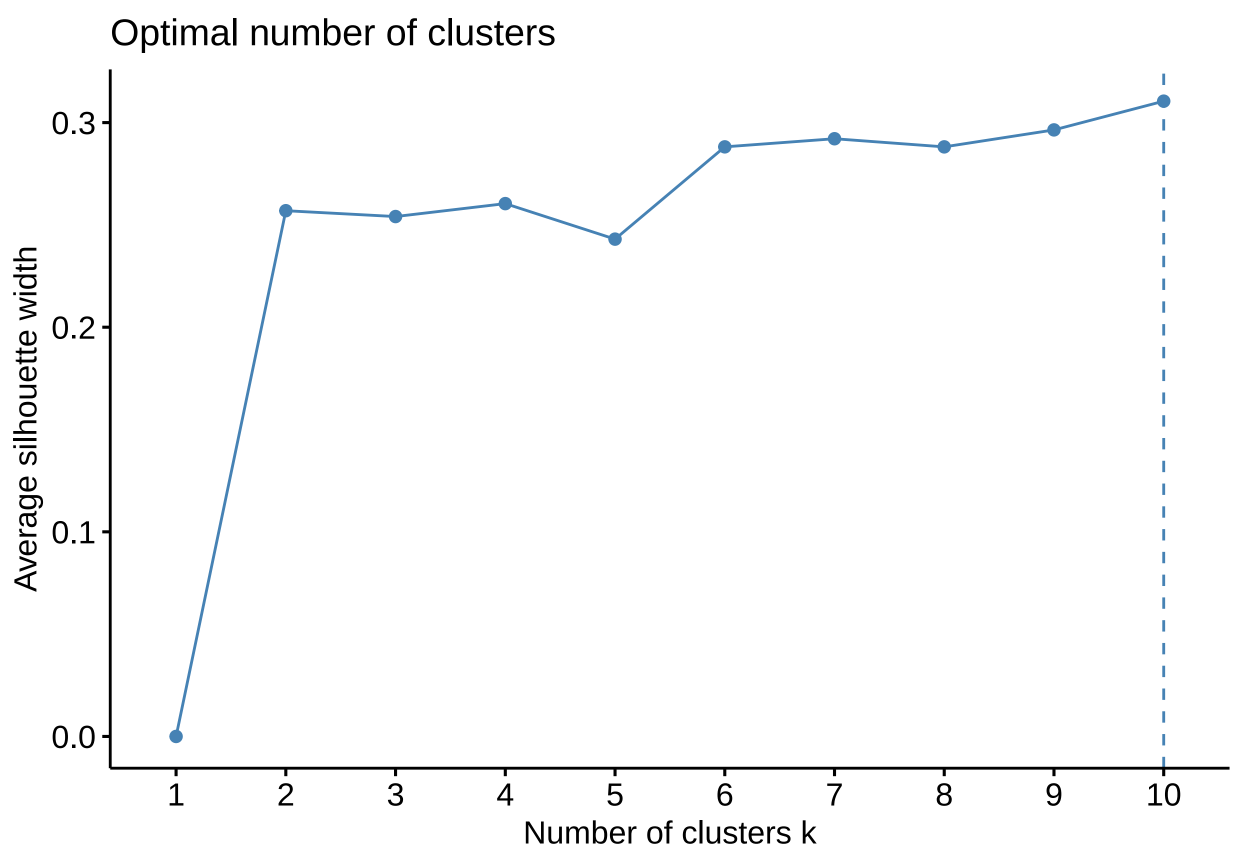


Can you group students from the classes data set in Assignment 2 using K-modes?

```r
#Load the data
load("assn2.RData")

#Assign rownames
rownames(person_class_DF) <- person_class_DF$id
```

```
## Warning: Setting row names on a tibble is deprecated.
```

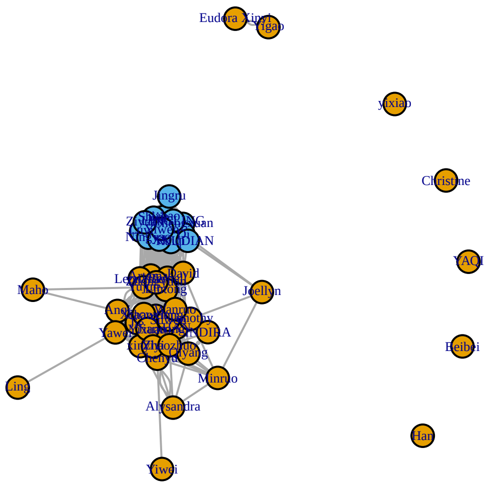
```r
person_class_DF <- person_class_DF %>% select(-id)

#Find number of appropriate clusters
fviz_nbclust(person_class_DF, kmeans)
```



```r
#Perform k means clustering with 2 clusters (any more doesn't seem to help)
class_fit <- kmeans(person_class_DF, 2)

#Import igraph
library(igraph)
```

```
##
## Attaching package: 'igraph'

## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union

## The following objects are masked from 'package:purrr':
##
##     compose, simplify

## The following object is masked from 'package:tidyr':
##
##     crossing

## The following object is masked from 'package:tibble':
##
##     as_data_frame
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

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

```
#Attach cluster info to the original dataframe

plot.igraph(person_person_graph_data,
     layout = layout.fruchterman.reingold,
     vertex.size = 10,
     vertex.color = class_fit$cluster,
     vertex.label.cex = 0.4
     )
```


Eudora
Xinyi
Yigao
yixiao
Christine
Jingru
YAQI
Joellyn
Maho
Beibei
Minruo
Ling
Alysandra
Yiwei
Han