Intro to ML PS 4

Takahiro Minami

3/2/2020

```
knitr::opts_chunk$set(echo = TRUE)
library(skimr)
library(dendextend)
```

```
##
## ---------------------
## Welcome to dendextend version 1.13.3
## Type citation('dendextend') for how to cite the package.
##
## Type browseVignettes(package = 'dendextend') for the package vignette.
## The github page is: https://github.com/talgalili/dendextend/
##
## Suggestions and bug-reports can be submitted at: https://github.com/talgalili/dendextend/issues
## Or contact: <tal.galili@gmail.com>
##
##  To suppress this message use:  suppressPackageStartupMessages(library(dendextend))
## ---------------------
```

```
##
## Attaching package: 'dendextend'
```

```
## The following object is masked from 'package:stats':
##
##     cutree
```

```
library(mixtools)
```

```
## mixtools package, version 1.2.0, Released 2020-02-05
## This package is based upon work supported by the National Science Foundation under Grant No. SES-0518
```

```
library(plotGMM)
library(kableExtra)
library(gridExtra)
library(geosphere)
library(TSdist)
```

```
## Loading required package: proxy
```

```
##
## Attaching package: 'proxy'
```

```
## The following objects are masked from 'package:stats':
##
##     as.dist, dist
```

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

```
##
##     as.matrix
```

```
## Loaded TSdist v3.6. See ?TSdist for help, citation("TSdist") for use in publication.
```

```
library(factoextra)
```

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

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v tibble  2.1.3     v dplyr   0.8.4
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
## v purrr   0.3.3
```

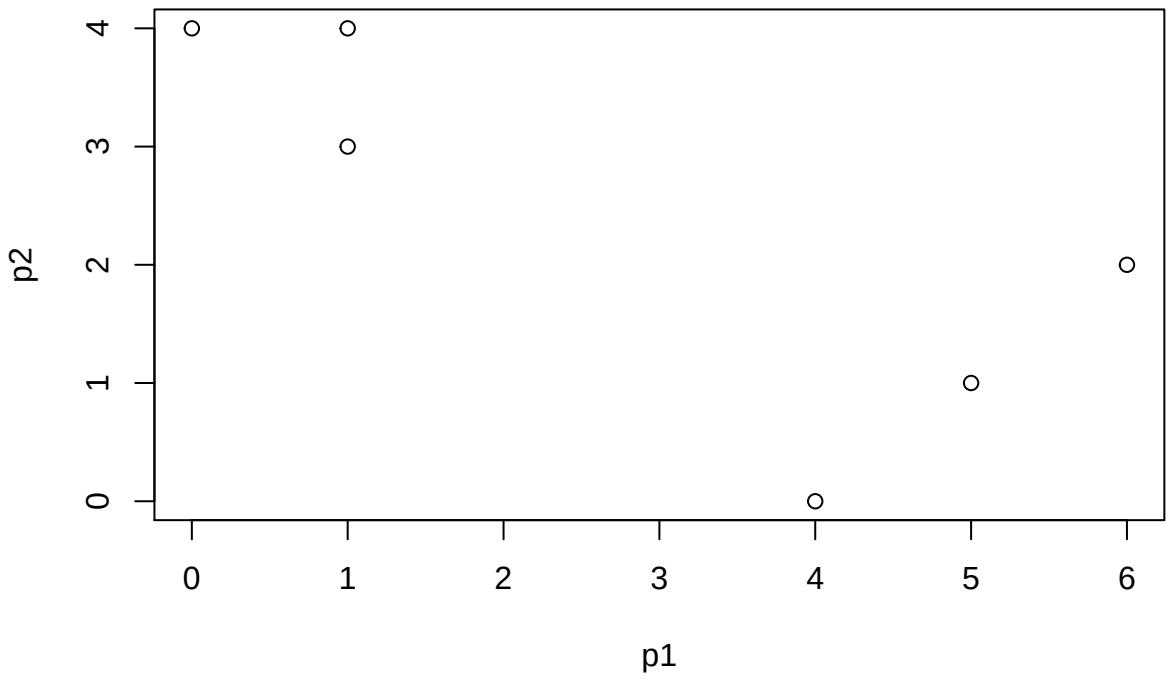
```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::combine()    masks gridExtra::combine()
## x dplyr::filter()     masks stats::filter()
## x dplyr::group_rows() masks kableExtra::group_rows()
## x dplyr::lag()        masks stats::lag()
```

Part 1

```
x <- cbind(c(1, 1, 0, 5, 6, 4), c(4, 3, 4, 1, 2, 0))
```

Question 1

```
# convert x to dataframe for the sake of convenience
x <- data.frame(p1=x[,1], p2=x[,2])
plot(x)
```

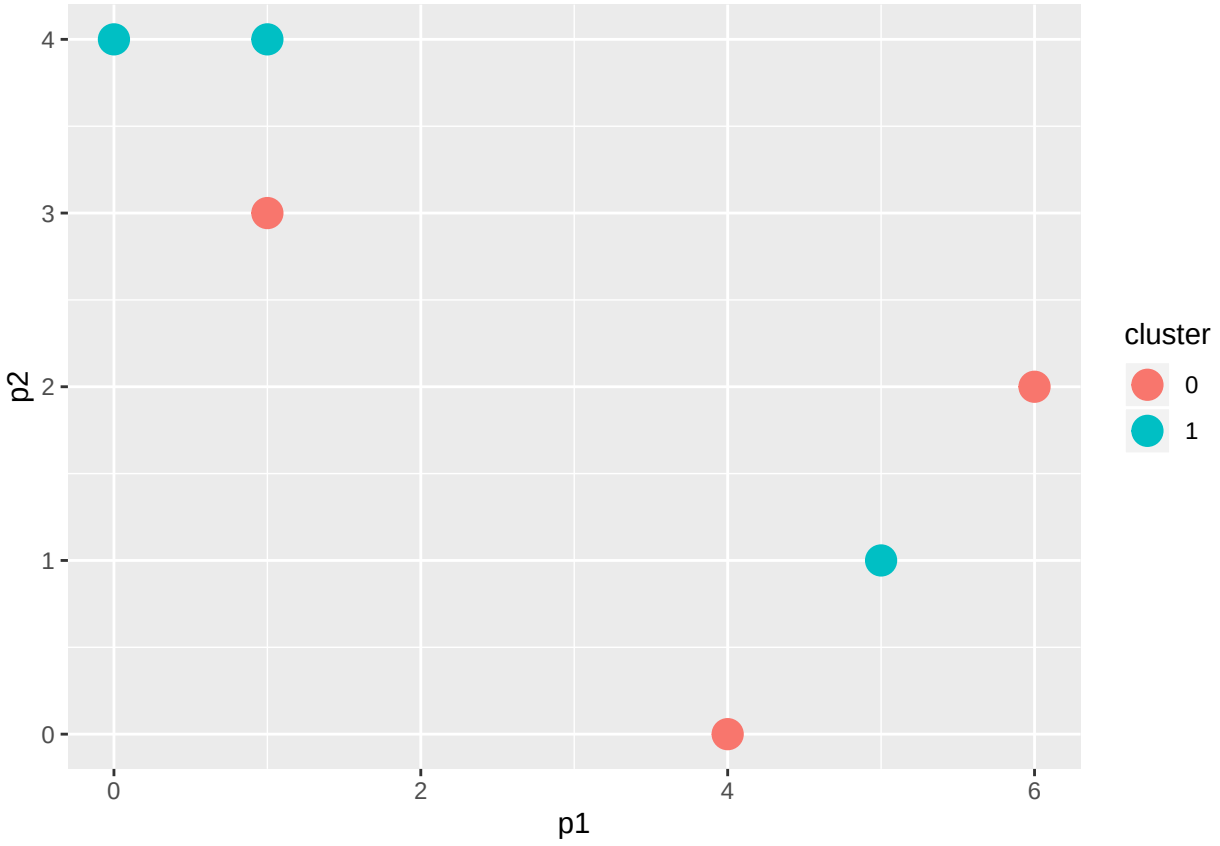


Question 2

```
set.seed(1345)
x$cluster <- as.factor(sample(c(0,1),nrow(x),replace=TRUE))
kable(x)
```

p1	p2	cluster
1	4	1
1	3	0
0	4	1
5	1	1
6	2	0
4	0	0

```
ggplot(data=x, aes(x=p1,y=p2,col=cluster))+
  geom_point(size=5)
```



Question 3

```
cent0 <- x %>% filter(cluster==0) %>% select(p1,p2) %>%
         centroid()

cent1 <- x %>% filter(cluster==1) %>% select(p1,p2) %>%
         centroid()

cent0
```

```
##           lon      lat
## [1,] 3.666667 1.667024
```

```
cent1
```

```
##      lon      lat
## [1,]   2 3.000813
```

Question 4

```
# distance from centroid
x$dist0 <- c()
x$dist1 <- c()

for (i in 1:nrow(x)){
```

```
  x$dist0[i] <- ((x[i,1]-cent0[1,1])^2+(x[i,2]-cent0[1,2])^2)^0.5
  x$dist1[i] <- ((x[i,1]-cent1[1,1])^2+(x[i,2]-cent1[1,2])^2)^0.5
}

# re-assign cluster
x$cluster_old <- x$cluster
x$cluster_new <- c()

for (i in 1:nrow(x)){
 if (x$dist0[i]<=x$dist1[i]) {
   x$cluster_new[i] <- 0
 } else{
   x$cluster_new[i] <- 1
 }
}

# result
x %>% select(p1,p2,cluster_new)%>%kable()
```

p1	p2	cluster_new
1	4	1
1	3	1
0	4	1
5	1	0
6	2	0
4	0	0

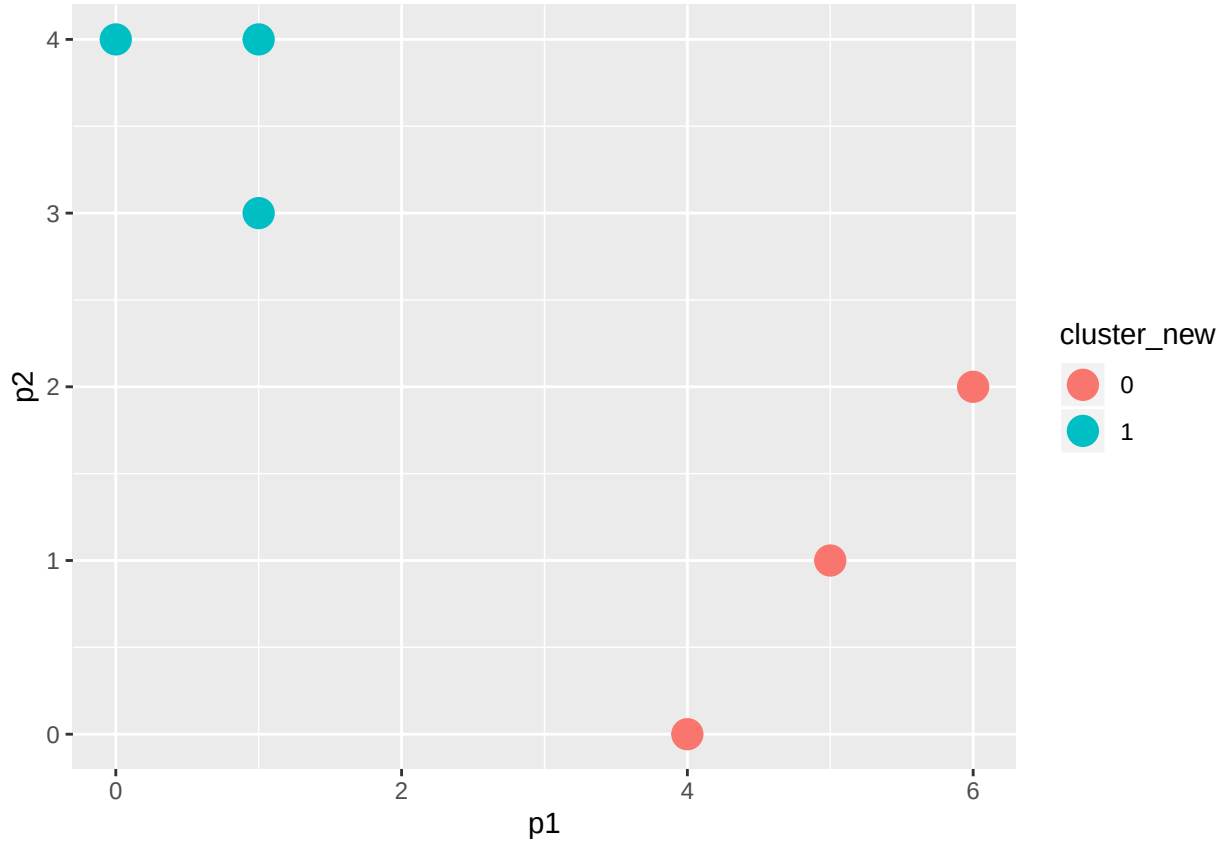
Question 5

```
# repeat re-assignment

if(sum(x$cluster_old==x$cluster_new)!=nrow(x)){
 # if cluster changes in previous process, do re-assignment
  ## revise old cluster
  x$cluster_old <- x$cluster_new

  ## get centroid
  cent0 <- x %>% filter(cluster_old==0) %>% select(p1,p2) %>%
         centroid()

  cent1 <- x %>% filter(cluster_old==1) %>% select(p1,p2) %>%
         centroid()

  ## measure distance
  for (i in 1:nrow(x)){

  x$dist0[i] <- ((x[i,1]-cent0[1,1])^2+(x[i,2]-cent0[1,2])^2)^0.5
  x$dist1[i] <- ((x[i,1]-cent1[1,1])^2+(x[i,2]-cent1[1,2])^2)^0.5
  }

  ## assign cluster
  for (i in 1:nrow(x)){
    if (x$dist0[i]<=x$dist1[i]) {
```

```
      x$cluster_new[i] <- 0
    } else{
      x$cluster_new[i] <- 1
    }
  }
}
```

Question 6

```
x$cluster_new<-as.factor(x$cluster_new)
ggplot(data=x, aes(x=p1,y=p2,col=cluster_new))+
  geom_point(size=5)
```



Part 2

Question 1

```
load("legprof-components.v1.0.RData")
df_ori <- x
head(df_ori)
```

```
##   fips stateabv   state  sessid t_slength slength salary_real   expend year
## 1    1       AL Alabama  1973/4     46.00    36.0    1.768022 125.0973 1974
## 2    1       AL Alabama  1975/6    110.00    74.0    2.933038 203.8466 1976
## 3    1       AL Alabama  1977/8     83.00    60.0    2.082810 184.0115 1978
```

```
## 4    1       AL Alabama 1979/80     65.00    60.0    1.694951 175.9863 1980
## 5    1       AL Alabama  1981/2    218.68   149.1    3.472914 204.1236 1982
## 6    1       AL Alabama  1983/4    188.86   149.1   11.705946 206.6745 1984
##         mds1        mds2
## 1 -1.7061814  0.38482016
## 2 -1.2128817 -0.08150655
## 3 -1.4149657  0.12789978
## 4 -1.5431539  0.27268842
## 5 -0.5013642 -1.00387760
## 6 -0.5840194 -0.74132360
```

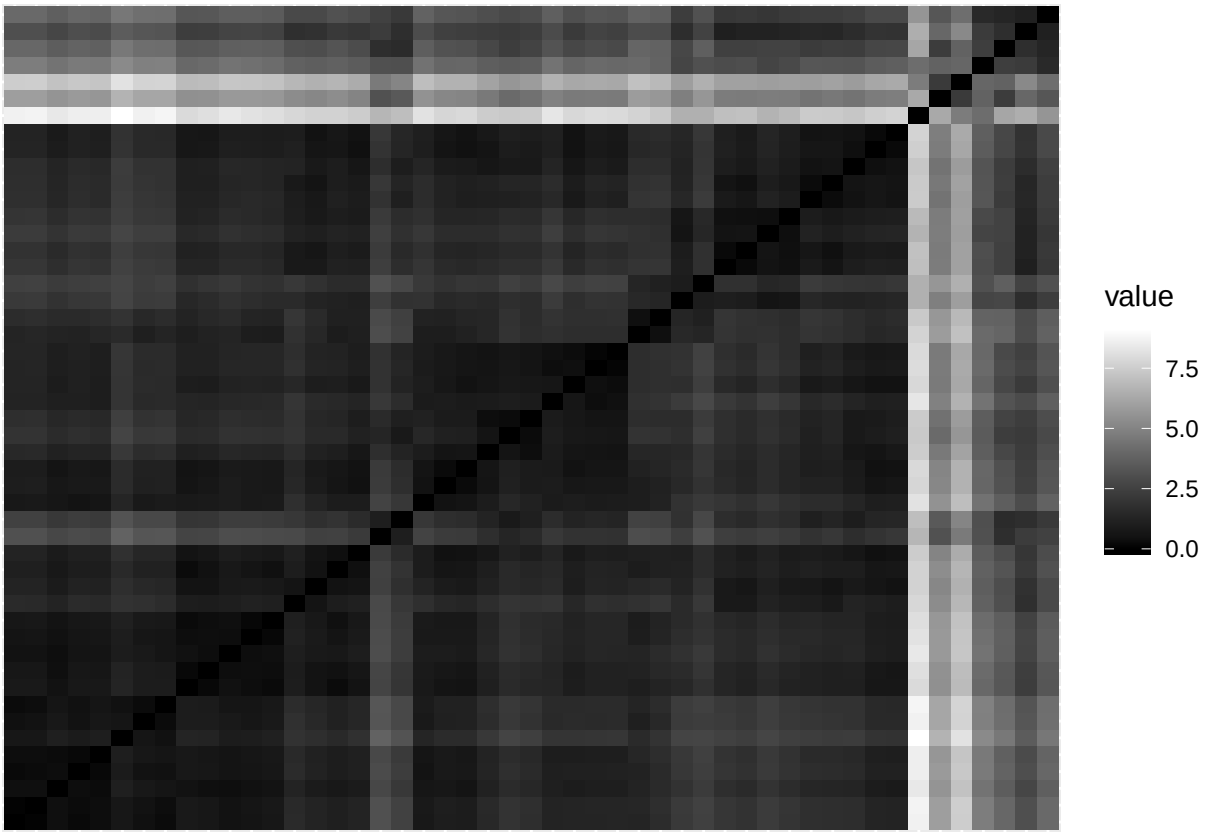
Question 2

```
# select necessary features and omit NA
df <- df_ori %>%
  select(state,sessid,t_slength,slength,salary_real,expend)%>%
  filter(sessid=="2009/10")%>%
  na.omit()

# standarization
df[,3:6]<-scale(df[,3:6])

# other things
row.names(df)<-df$state
df <- df[,3:6]
```

Question 3

```
gradient_col = list(low = "black", high = "white")
get_clust_tendency(df, n = 40, gradient = gradient_col)
```

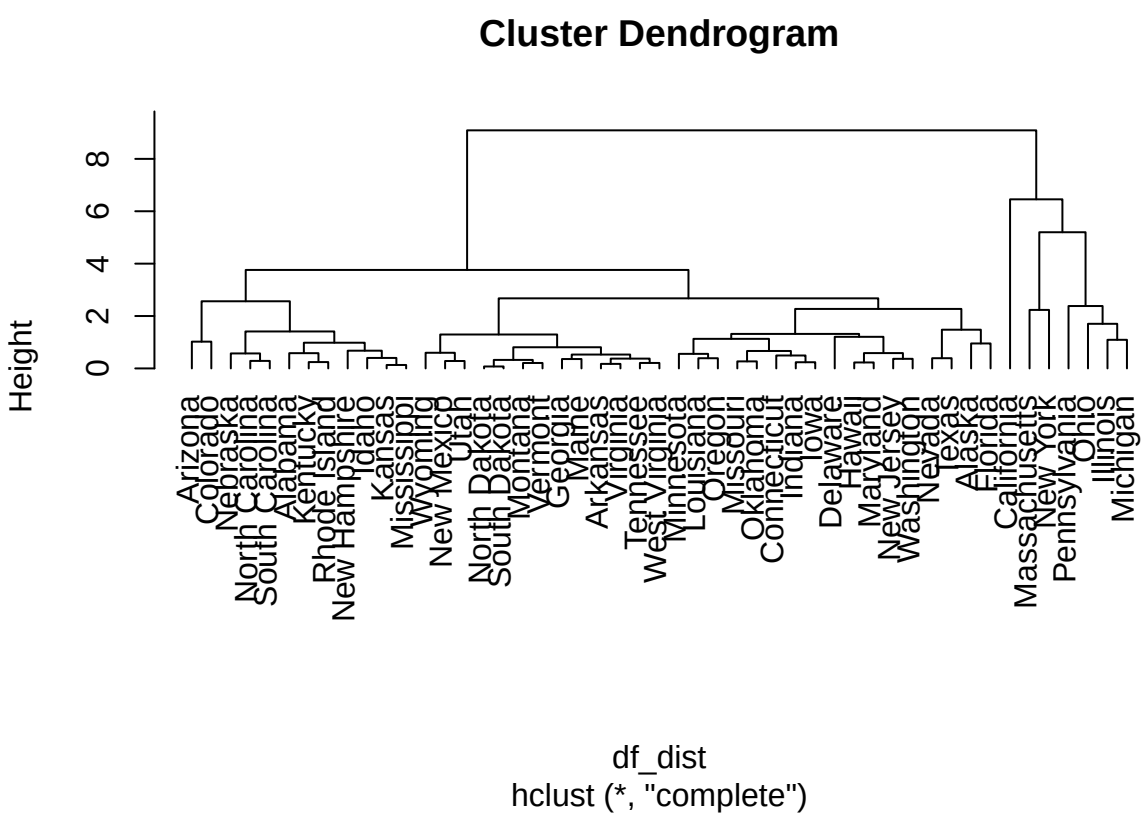
```
## $hopkins_stat
## [1] 0.8406165
##
## $plot
```



The Hopkins statistics is 0.8 and more than 0.5. It implies that the data is non-random and clusterable. This result is supported by the fact that Ordered Dissimilarity Images shows square shaped dark blocks.

Question 4

```
df_dist <- dist(df)

# hierarchical clustering
hc_complete <- hclust(df_dist,
                      method = "complete"); plot(hc_complete, hang = -1)
```



I can find two main cluster in the dendrogram. The largaer one cluster is from Arizona to Florida in the figure, and the other small one is California to Michigan in the figure. Roughly speaking, states geographically close to each other tend to be clustered together (the distance on the space is close). For example, in the smaller main cluster, Michigan and Illinois are clustered at first, then several east coast states (Ohio, Pennsylvania,Massachusetts, and New York) are joined, and California has relatively big distance from the other states in the space.

For the later question, I cut the tree by $k = 2$ and store predicted clustered.

```
# cluster suggested by agglomerative hierarchical clustering
cut <- as.data.frame(cutree(hc_complete, k = 2))

# result data frame
result <- data.frame(hc = as.factor(cut[,1]),
                     state=row.names(df))
row.names(result) <- row.names(df)
```

Question 5

```
set.seed(1234)

kmeans <- kmeans(df,
                 centers = 2,  # k. Number of cluster
                 nstart = 15)
str(kmeans)
```

```
## List of 9
##  $ cluster     : Named int [1:49] 1 1 1 1 2 1 1 1 1 1 ...
##   ..- attr(*, "names")= chr [1:49] "Alabama" "Alaska" "Arizona" "Arkansas" ...
##  $ centers     : num [1:2, 1:4] -0.293 2.1 -0.293 2.101 -0.283 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "1" "2"
##   .. ..$ : chr [1:4] "t_slength" "slength" "salary_real" "expend"
##  $ totss       : num 192
##  $ withinss    : num [1:2] 48.4 40.4
##  $ tot.withinss: num 88.7
##  $ betweenss   : num 103
##  $ size        : int [1:2] 43 6
##  $ iter        : int 1
##  $ ifault      : int 0
##  - attr(*, "class")= chr "kmeans"
```

```
# cluster suggested by k-means
result$kmean <- as.factor(kmeans$cluster)
result %>% select(state,kmean)%>% arrange(kmean)%>%kable()
```

state	kmean
Alabama	1
Alaska	1
Arizona	1
Arkansas	1
Colorado	1
Connecticut	1
Delaware	1
Florida	1
Georgia	1
Hawaii	1
Idaho	1
Illinois	1
Indiana	1
Iowa	1
Kansas	1
Kentucky	1
Louisiana	1
Maine	1
Maryland	1
Minnesota	1
Mississippi	1
Missouri	1
Montana	1
Nebraska	1
Nevada	1
New Hampshire	1
New Jersey	1
New Mexico	1
North Carolina	1
North Dakota	1
Oklahoma	1
Oregon	1
Rhode Island	1
South Carolina	1
South Dakota	1
Tennessee	1
Texas	1
Utah	1
Vermont	1
Virginia	1
Washington	1
West Virginia	1
Wyoming	1
California	2
Massachusetts	2
Michigan	2
New York	2
Ohio	2
Pennsylvania	2

The kmean algorithm divided 49 states into one larger cluster containing 43 states and the other smaller cluster containing 6 states. The smaller cluster includes East coast states and California, which is basically the

same result as hierarchical clustering although Illinois is assigned to the larger cluster in k-means.Interestingly, the within sum of squares for two clusters are not different so much (48.4 and 40.4) while the number of the states in the one cluster is much bigger than the other. This implies that the states in the larger cluster are more homogeneous among the cluster than the states in the smaller cluster.

Question 6

```
set.seed(2345)
gmm <- mvnormalmixEM(df[,1:4], k = 2)
```

```
## number of iterations= 12
```

```
gmm[-1]
```

```
## $lambda
## [1] 0.8782481 0.1217519
##
## $mu
## $mu[[1]]
## [1] -0.2436660 -0.2547951 -0.1942601 -0.0287773
##
## $mu[[2]]
## [1] 1.7576660 1.8379451 1.4012804 0.2075828
##
##
## $sigma
## $sigma[[1]]
##           [,1]      [,2]      [,3]      [,4]
## [1,] 0.4362442 0.3683113 0.4016391 0.4951863
## [2,] 0.3683113 0.3378781 0.3086969 0.3346515
## [3,] 0.4016391 0.3086969 0.7237510 0.6598573
## [4,] 0.4951863 0.3346515 0.6598573 1.0555764
##
## $sigma[[2]]
##            [,1]      [,2]       [,3]      [,4]
## [1,] 1.38131481 1.4762722 0.06095441 0.3726588
## [2,] 1.47627215 1.7622030 0.32320180 0.4056042
## [3,] 0.06095441 0.3232018 0.58928014 0.2777633
## [4,] 0.37265878 0.4056042 0.27776330 0.3824188
##
##
## $loglik
## [1] -114.1102
##
## $posterior
##             comp.1        comp.2
##  [1,] 1.000000e+00  0.000000e+00
##  [2,] 1.000000e+00  0.000000e+00
##  [3,] 7.724507e-18  1.000000e+00
##  [4,] 1.000000e+00  0.000000e+00
##  [5,] 1.000000e+00 1.838941e-131
##  [6,] 1.000000e+00 6.522743e-215
##  [7,] 1.000000e+00 4.412703e-206
##  [8,] 1.000000e+00  1.023161e-73
##  [9,] 1.000000e+00  0.000000e+00
```

```
## [10,] 1.000000e+00  0.000000e+00
## [11,] 1.000000e+00  6.015365e-87
## [12,] 1.000000e+00  0.000000e+00
## [13,] 3.200042e-02  9.679996e-01
## [14,] 1.000000e+00 1.003994e-280
## [15,] 1.000000e+00 1.381158e-298
## [16,] 1.000000e+00  0.000000e+00
## [17,] 1.000000e+00  0.000000e+00
## [18,] 1.000000e+00  0.000000e+00
## [19,] 1.000000e+00  0.000000e+00
## [20,] 1.000000e+00 4.587811e-156
## [21,] 2.611553e-12  1.000000e+00
## [22,] 6.505592e-04  9.993494e-01
## [23,] 1.000000e+00 1.223810e-247
## [24,] 1.000000e+00  0.000000e+00
## [25,] 1.000000e+00 1.082128e-123
## [26,] 1.000000e+00  0.000000e+00
## [27,] 1.000000e+00  0.000000e+00
## [28,] 1.000000e+00  0.000000e+00
## [29,] 1.000000e+00  0.000000e+00
## [30,] 1.000000e+00 8.763148e-253
## [31,] 1.000000e+00  0.000000e+00
## [32,] 1.057204e-12  1.000000e+00
## [33,] 1.000000e+00  0.000000e+00
## [34,] 1.000000e+00  0.000000e+00
## [35,] 1.505150e-03  9.984949e-01
## [36,] 1.000000e+00 4.870211e-130
## [37,] 1.000000e+00  0.000000e+00
## [38,] 1.000000e+00  4.068518e-73
## [39,] 1.000000e+00  0.000000e+00
## [40,] 1.000000e+00  0.000000e+00
## [41,] 1.000000e+00  0.000000e+00
## [42,] 1.000000e+00  0.000000e+00
## [43,] 1.000000e+00  0.000000e+00
## [44,] 1.000000e+00  0.000000e+00
## [45,] 1.000000e+00  0.000000e+00
## [46,] 1.000000e+00  0.000000e+00
## [47,] 1.000000e+00 8.171374e-144
## [48,] 1.000000e+00  0.000000e+00
## [49,] 1.000000e+00  0.000000e+00
##
## $all.loglik
##  [1] -441.8042 -133.1237 -130.7297 -128.8627 -127.2491 -124.9236 -120.0242
##  [8] -114.1436 -114.1105 -114.1102 -114.1102 -114.1102 -114.1102
##
## $restarts
## [1] 0
##
## $ft
## [1] "mvnormalmixEM"
```

```r
# cluster suggested by k-means
for (i in 1:nrow(df)){
  if (gmm$posterior[i,1]>=gmm$posterior[i,2]){
```

```
    result$gmm[i] <- 1
  } else {
    result$gmm[i] <- 2
  }
}
result$gmm <- as.factor(result$gmm)
result %>% select(state,gmm)%>% arrange(gmm)%>%kable()
```

state	gmm
Alabama	1
Alaska	1
Arkansas	1
California	1
Colorado	1
Connecticut	1
Delaware	1
Florida	1
Georgia	1
Hawaii	1
Idaho	1
Indiana	1
Iowa	1
Kansas	1
Kentucky	1
Louisiana	1
Maine	1
Maryland	1
Minnesota	1
Mississippi	1
Missouri	1
Montana	1
Nebraska	1
Nevada	1
New Hampshire	1
New Jersey	1
New Mexico	1
North Carolina	1
North Dakota	1
Oklahoma	1
Oregon	1
Pennsylvania	1
Rhode Island	1
South Carolina	1
South Dakota	1
Tennessee	1
Texas	1
Utah	1
Vermont	1
Virginia	1
Washington	1
West Virginia	1
Wyoming	1
Arizona	2
Illinois	2
Massachusetts	2
Michigan	2
New York	2
Ohio	2

The gmm algorithm divided 49 states into one larger cluster containing 43 states and the other smaller cluster containing 6 states. Although the big picture (one large cluster and one small cluster) is the same as

hierarchical clustering and gmm, the states included in the smaller cluster are slightly different. In stead of California, Arizona is included in the smaller cluster. Looking at the probability to be assigned to each of two states ("$posterior" in the result output), I find that all states have almost 100% probability to be assigned to their cluster. It seams that the boundary of two cluster is relatively clear in this case.

Question 7

I plot 49 states in two dimensional space and compare how three algorithm assigned them to two clusters. Since I have 4 features, I plot 6 combinations of x and y axis.

```
# merge result table and original data
result <- cbind(result,df)
```

```
# hc
p1 <-
ggplot(data=result,
       aes(x=t_slength,y=slength,color=hc))+
  geom_point(size=3)+
  ggtitle("HC")+
  theme(legend.position = c(.1, .95),
    legend.justification = c("right", "top"),
    legend.box.just = "right",
    legend.margin = margin(6, 6, 6, 6))
# kmean
p2 <-
ggplot(data=result,
       aes(x=t_slength,y=slength,color=kmean))+
  geom_point(size=3)+
  ggtitle("K-mean")+
  theme(legend.position = c(.1, .95),
    legend.justification = c("right", "top"),
    legend.box.just = "right",
    legend.margin = margin(6, 6, 6, 6))

# gmm
p3 <-
ggplot(data=result,
       aes(x=t_slength,y=slength,color=gmm))+
  geom_point(size=3)+
  ggtitle("GMM")+
  theme(legend.position = c(.1, .95),
    legend.justification = c("right", "top"),
    legend.box.just = "right",
    legend.margin = margin(6, 6, 6, 6))
```

```
# hc
p4 <-
ggplot(data=result,
       aes(x=t_slength,y=salary_real,color=hc))+
  geom_point(size=3)+
  ggtitle("HC")+
  theme(legend.position = c(.1, .95),
    legend.justification = c("right", "top"),
    legend.box.just = "right",
    legend.margin = margin(6, 6, 6, 6))
```

```r
# kmean
p5 <-
ggplot(data=result,
       aes(x=t_slength,y=salary_real,color=kmean))+
  geom_point(size=3)+
  ggtitle("K-mean")+
  theme(legend.position = c(.1, .95),
    legend.justification = c("right", "top"),
    legend.box.just = "right",
    legend.margin = margin(6, 6, 6, 6))

# gmm
p6 <-
ggplot(data=result,
       aes(x=t_slength,y=salary_real,color=gmm))+
  geom_point(size=3)+
  ggtitle("GMM")+
  theme(legend.position = c(.1, .95),
    legend.justification = c("right", "top"),
    legend.box.just = "right",
    legend.margin = margin(6, 6, 6, 6))
```

```r
# hc
p7 <-
ggplot(data=result,
       aes(x=t_slength,y=expend,color=hc))+
  geom_point(size=3)+
  ggtitle("HC")+
  theme(legend.position = c(.1, .95),
    legend.justification = c("right", "top"),
    legend.box.just = "right",
    legend.margin = margin(6, 6, 6, 6))
# kmean
p8 <-
ggplot(data=result,
       aes(x=t_slength,y=expend,color=kmean))+
  geom_point(size=3)+
  ggtitle("K-mean")+
  theme(legend.position = c(.1, .95),
    legend.justification = c("right", "top"),
    legend.box.just = "right",
    legend.margin = margin(6, 6, 6, 6))

# gmm
p9 <-
ggplot(data=result,
       aes(x=t_slength,y=expend,color=gmm))+
  geom_point(size=3)+
  ggtitle("GMM")+
  theme(legend.position = c(.1, .95),
    legend.justification = c("right", "top"),
    legend.box.just = "right",
    legend.margin = margin(6, 6, 6, 6))
```

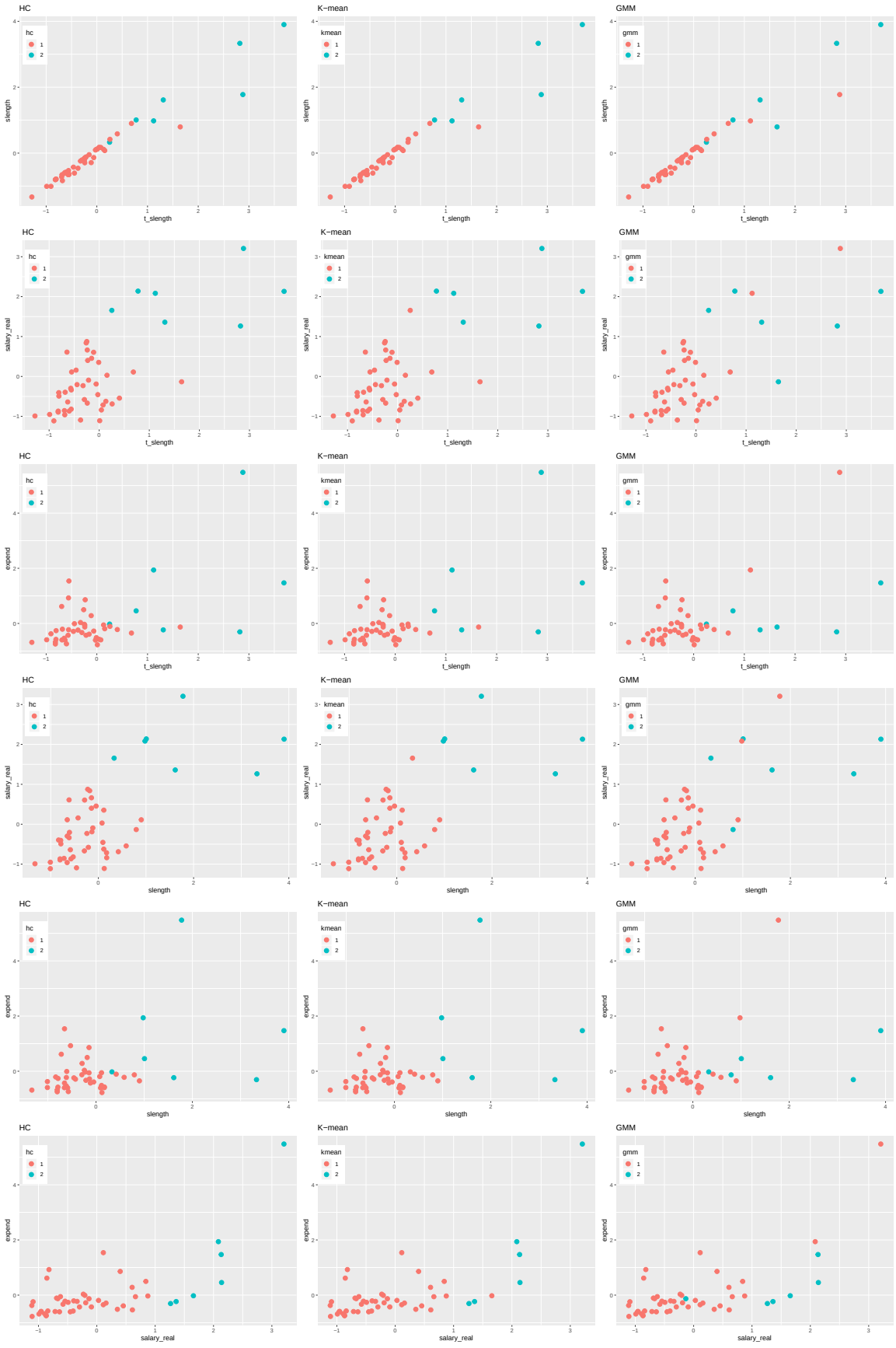
```r
# hc
p10 <-
ggplot(data=result,
       aes(x=slength,y=salary_real,color=hc))+
  geom_point(size=3)+
  ggtitle("HC")+
  theme(legend.position = c(.1, .95),
    legend.justification = c("right", "top"),
    legend.box.just = "right",
    legend.margin = margin(6, 6, 6, 6))
# kmean
p11 <-
ggplot(data=result,
       aes(x=slength,y=salary_real,color=kmean))+
  geom_point(size=3)+
  ggtitle("K-mean")+
  theme(legend.position = c(.1, .95),
    legend.justification = c("right", "top"),
    legend.box.just = "right",
    legend.margin = margin(6, 6, 6, 6))

# gmm
p12 <-
ggplot(data=result,
       aes(x=slength,y=salary_real,color=gmm))+
  geom_point(size=3)+
  ggtitle("GMM")+
  theme(legend.position = c(.1, .95),
    legend.justification = c("right", "top"),
    legend.box.just = "right",
    legend.margin = margin(6, 6, 6, 6))
```

```r
# hc
p13 <-
ggplot(data=result,
       aes(x=slength,y=expend,color=hc))+
  geom_point(size=3)+
  ggtitle("HC")+
  theme(legend.position = c(.1, .95),
    legend.justification = c("right", "top"),
    legend.box.just = "right",
    legend.margin = margin(6, 6, 6, 6))
# kmean
p14 <-
ggplot(data=result,
       aes(x=slength,y=expend,color=kmean))+
  geom_point(size=3)+
  ggtitle("K-mean")+
  theme(legend.position = c(.1, .95),
    legend.justification = c("right", "top"),
    legend.box.just = "right",
    legend.margin = margin(6, 6, 6, 6))

# gmm
```

```
p15 <-
ggplot(data=result,
       aes(x=slength,y=expend,color=gmm))+
  geom_point(size=3)+
  ggtitle("GMM")+
  theme(legend.position = c(.1, .95),
    legend.justification = c("right", "top"),
    legend.box.just = "right",
    legend.margin = margin(6, 6, 6, 6))
```

```
# hc
p16 <-
ggplot(data=result,
       aes(x=salary_real,y=expend,color=hc))+
  geom_point(size=3)+
  ggtitle("HC")+
  theme(legend.position = c(.1, .95),
    legend.justification = c("right", "top"),
    legend.box.just = "right",
    legend.margin = margin(6, 6, 6, 6))
# kmean
p17 <-
ggplot(data=result,
       aes(x=salary_real,y=expend,color=kmean))+
  geom_point(size=3)+
  ggtitle("K-mean")+
  theme(legend.position = c(.1, .95),
    legend.justification = c("right", "top"),
    legend.box.just = "right",
    legend.margin = margin(6, 6, 6, 6))

# gmm
p18 <-
ggplot(data=result,
       aes(x=salary_real,y=expend,color=gmm))+
  geom_point(size=3)+
  ggtitle("GMM")+
  theme(legend.position = c(.1, .95),
    legend.justification = c("right", "top"),
    legend.box.just = "right",
    legend.margin = margin(6, 6, 6, 6))
```

```
grid.arrange(p1, p2,p3,p4, p5,p6,p7, p8, p9,
             p10, p11,p12,p13, p14,p15,p16, p17,p18,
             nrow = 6)
```


HC
K−mean
GMM
hc
kmean
gmm
1
2
slength
t_slength
salary_real
expend

All three methods got similar classification results. Especially, I think the k-mean classified states very clearly (I can imagine simple boundary on the plots) based on the plots above. On the other hand, in gmm result, I can find some messy part where state in cluster 1 and 2 are mixed in a small area.

Question 8

I choose WSS as a validation strategy. As discussed in the class, the smaller WSS is better.

First, I developed the original function to calculate WSS.

```
# build function to calculate WSS

wss <- function (result) {

df.wss <-   result %>%
    group_by(cluster) %>%
    mutate(t_slength_bar=mean(t_slength),
           slength_bar=mean(slength),
           salary_real_bar=mean(salary_real),
           expend_bar=mean(expend))

df.wss$t_slength_ws=(df.wss$t_slength-df.wss$t_slength_bar)^2
df.wss$slength_ws=(df.wss$slength-df.wss$slength_bar)^2
df.wss$salary_real_ws=(df.wss$salary_real-df.wss$salary_real_bar)^2
df.wss$expend_bar_ws=(df.wss$expend-df.wss$expend_bar)^2

sum(df.wss[,(ncol(df.wss)-4):ncol(df.wss)])
}
```

Then, I check WSS for three algorithm with $k = 2, 3, 4, 5$.

```
# number of k to try
n <- 4 # check k=2...5

# generate df to store the result
df.comp <- data.frame(
  method=c(rep("HC",n),rep("K-mean",n),rep("GMM",n)),
  k=rep(2:(n+1),3),
  WSS=rep(0,3*n))

# HC repeat
for (k in 2:5) {
  # cut tree
  cut <- as.data.frame(cutree(hc_complete, k = k))

  # get cluster
  df_vali <- df
  df_vali$cluster <- as.factor(cut[,1])

  # get WSS
  df.comp$WSS[k-1]<-wss(df_vali)
}

# k-mean
set.seed(20)
```

```r
for (k in 2:5) {
  # kmean
  kmeans <- kmeans(df,
                   centers = k,  # k. Number of cluster
                   nstart = 15)

  # get cluster
  df_vali <- df
  df_vali$cluster <- as.factor(kmeans$cluster)

  # get WSS
  df.comp$WSS[k+n-1]<-wss(df_vali)
}

# GMM
set.seed(25)
for (k in 2:4) {
  # fit GMM
  gmm <- mvnormalmixEM(df, k = k)

  # get cluster
  gmm_class <- c()
  for (i in 1:nrow(df)){
    if (gmm$posterior[i,1]>=gmm$posterior[i,2]){
      gmm_class[i] <- 1
    } else {
      gmm_class[i] <- 2
    }
  }

  df_vali <- df
  df_vali$cluster <- as.factor(gmm_class)

  # get WSS
  df.comp$WSS[k+n+3]<-wss(df_vali)
}
```

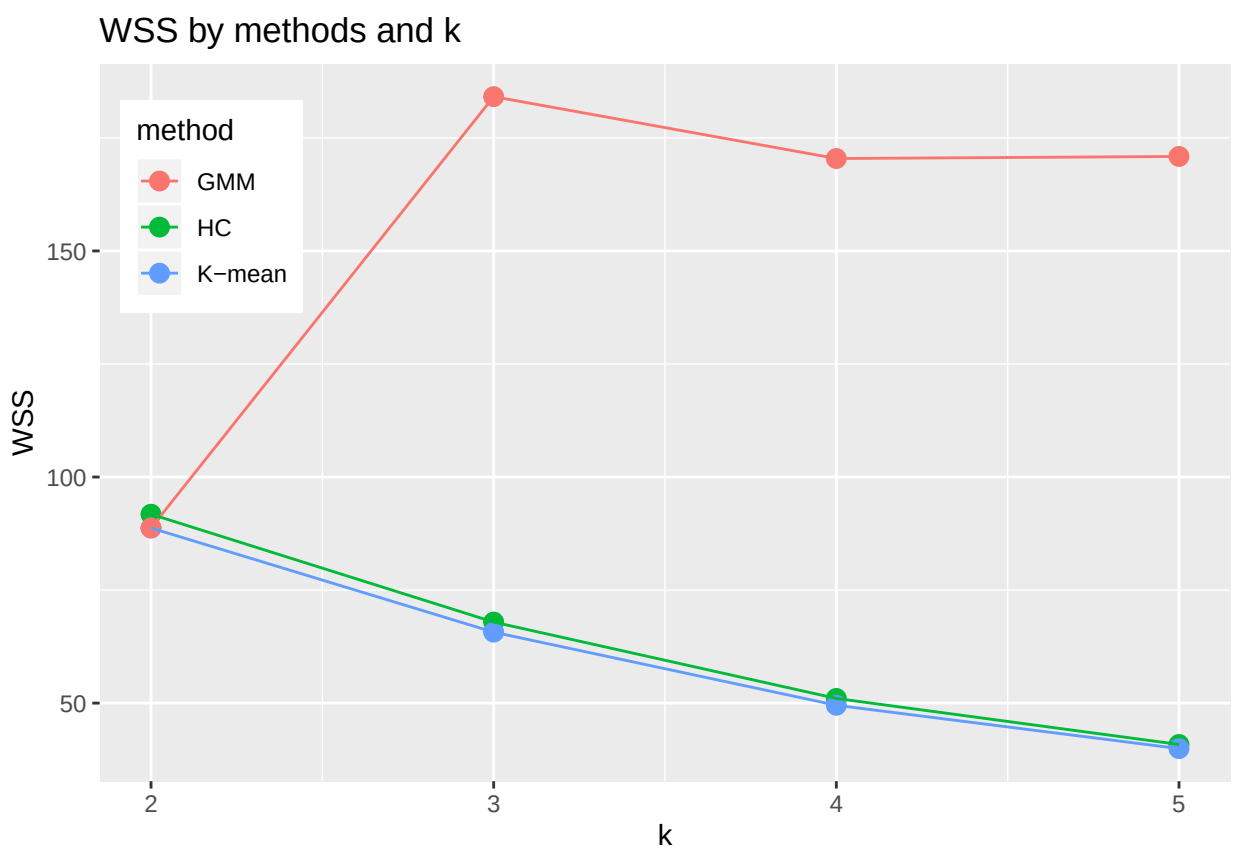
```
## number of iterations= 11
## number of iterations= 23
## number of iterations= 26
```

```r
set.seed(30)
for (k in 5:5) {
  # fit GMM
  gmm <- mvnormalmixEM(df, k = k)

  # get cluster
  gmm_class <- c()
  for (i in 1:nrow(df)){
    if (gmm$posterior[i,1]>=gmm$posterior[i,2]){
      gmm_class[i] <- 1
    } else {
      gmm_class[i] <- 2
    }
```

```
  }

  df_vali <- df
  df_vali$cluster <- as.factor(gmm_class)

  # get WSS
  df.comp$WSS[k+n+3]<-wss(df_vali)
}
```

```
## number of iterations= 18
```

```
# plot


ggplot(df.comp, aes(x=k,y=WSS,color=method))+
  geom_point(size=3)+
  geom_line()+
  ggtitle("WSS by methods and k")+
  theme(legend.position = c(.18, .95),
    legend.justification = c("right", "top"),
    legend.box.just = "right",
    legend.margin = margin(6, 6, 6, 6))
```



Question 9

The validation output shows that hierarchical clustering and k-mean show very similar validity (WSS) for all k while GMM has much higher WSS especially when k is more than 2. I also find that WSS is decreasing as

k increases in hierarchical clustering and k-mean while WSS increases from $k = 2$ to $k = 3$ and it is stable at high level afterwards in GMM.

If I purely follow this result, k-mean is the best method because it has the lowest, and larger k is better (among k I tried in this question, $k = 5$ is optimal).

However, picking optimal k only based on WSS is not necessary reasonable. First of all, WSS only care about intra-cluster homogeneity and don't measure inter-cluster heterogeneity. When k goes up, WSS should basically decrease (WSS is zero when $k = n$), but our ultimate goal is also to maximize inter-cluster heterogeneity. The same argument could stand for choosing method to cluster. It may not be a good idea to choose the method with the lowest WSS because it may allow inter-cluster heterogeneity be relatively small.

I also can imagine that we could choose second best method instead of the method with the lowest WSS when we are interested in other things than maximizing intra-cluster homogeneity and inter-cluster heterogeneity; for example output visualization. In this question, k-mean has the lowest WSS, but hierarchical clustering allows us to draw a insightful tree to capture the classification as I did in the previous question. If we need to show the "easy-to-read" figures instead of the technical result table, choosing hierarchical clustering may be able to be justified.