# MLPS4_YingZhou

Ying Zhou

2020/3/2

## Performing k-Means By Hand

### 1

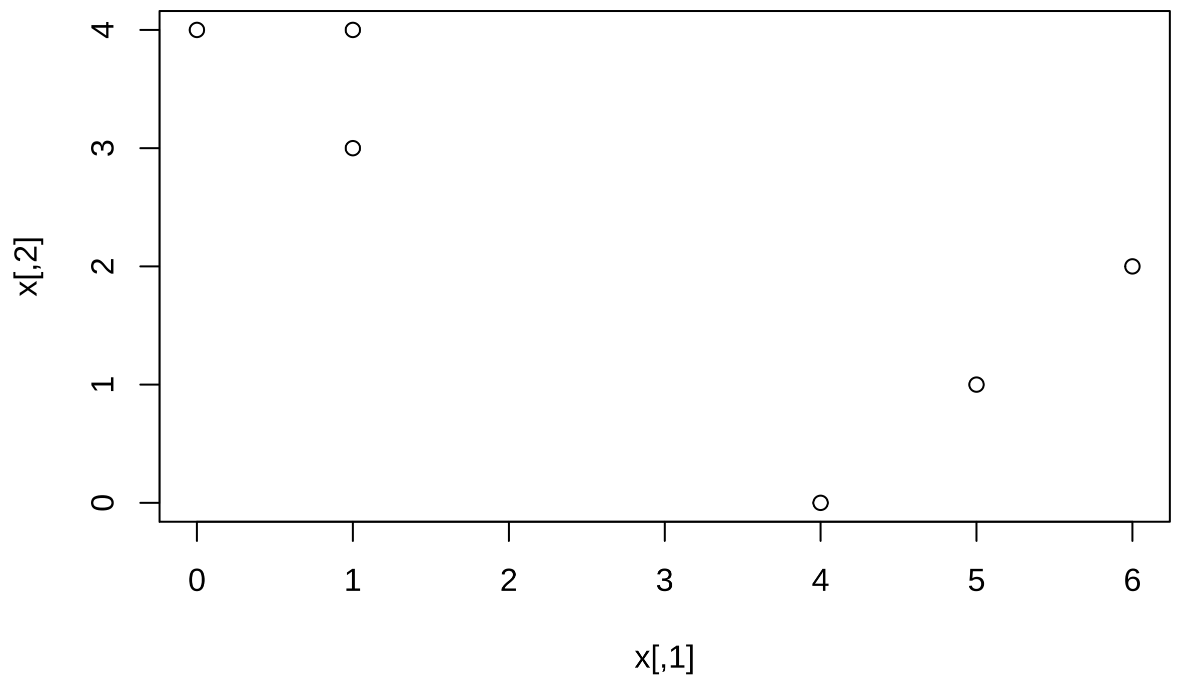
```
library(tidyverse)
```

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

## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

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

```
library(skimr)
library(dendextend)
```

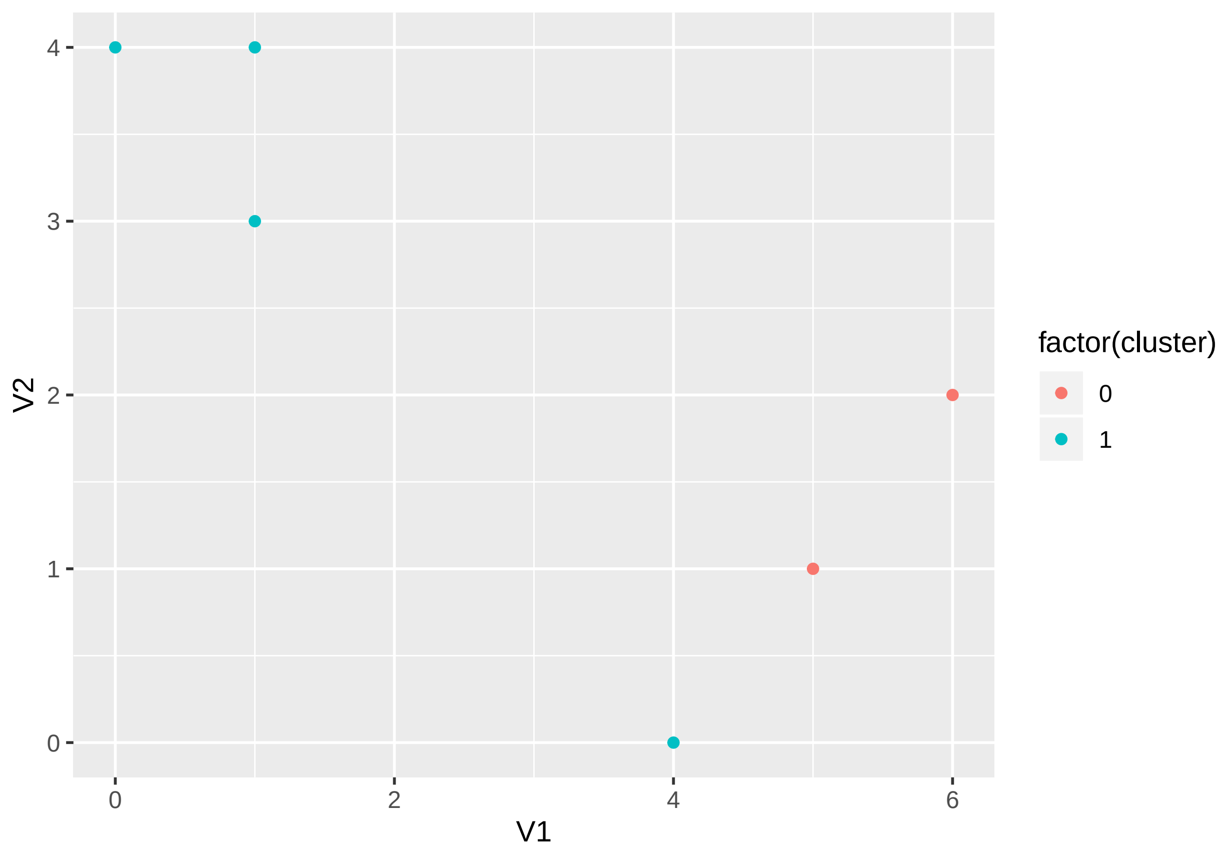
```
##
## ---------------------
## Welcome to dendextend version 1.13.3
## Type citation('dendextend') for how to cite the package.
##
## Type browseVignettes(package = 'dendextend') for the package vignette.
## The github page is: https://github.com/talgalili/dendextend/
##
## Suggestions and bug-reports 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
```

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



## 2

```r
set.seed(1213)
a=sample(c(0,1), size = nrow(x), replace = TRUE)
#data = matrix(data = x, ncol = 2, nrow = 6)
newdata = cbind(x, a)
library(ggplot2)
dfnewdata=data.frame(newdata)
names(dfnewdata)<- c("V1","V2","cluster")
ggplot(dfnewdata, aes(V1, V2, color=factor(cluster)))+geom_point()
```



## 3

```
list(rep(a, ncol(x)), col(x))
```

```
## [[1]]
##  [1] 1 1 1 0 0 1 1 1 1 0 0 1
##
## [[2]]
##      [,1] [,2]
## [1,]    1    2
## [2,]    1    2
## [3,]    1    2
## [4,]    1    2
## [5,]    1    2
## [6,]    1    2
```

```
centroids=tapply(x, list(rep(a, ncol(x)), col(x)),mean)
centroids
```

```
##     1    2
## 0 5.5 1.50
## 1 1.5 2.75
```

## 4

```
c1=centroids[1]
c2=centroids[3]
c3=centroids[2]
c4=centroids[4]

ecdist<-matrix(1:12,nrow = 6, ncol = 3)
for (i in 1:6){
  ecdist[i,1]=sqrt((x[i,1] - c1)^2 + (x[i,2] - c2)^2)
  ecdist[i,2]=sqrt((x[i,1] - c3)^2 + (x[i,1] - c4)^2)
  if (ecdist[i,1]>ecdist[i,2]) {
    ecdist[i,3]=2
  }else {
    ecdist[i,3]=1
  }
}
newdata2 = cbind(newdata, ecdist)
dfnewdata2=data.frame(newdata2)
names(dfnewdata2)<- c("V1","V2","cluster","dist1","disct2","group")
dfnewdata2
```

```
##   V1 V2 cluster     dist1   disct2 group
## 1  1  4       1 5.1478151 1.820027     2
## 2  1  3       1 4.7434165 1.820027     2
## 3  0  4       1 6.0415230 3.132491     2
## 4  5  1       0 0.7071068 4.160829     1
## 5  6  2       0 0.7071068 5.550901     1
## 6  4  0       1 2.1213203 2.795085     1
```

## 5

```
list(rep(dfnewdata2[,6], ncol(x)), col(x))
```

```
## [[1]]
##  [1] 2 2 2 1 1 1 2 2 2 1 1 1
##
## [[2]]
##      [,1] [,2]
## [1,]    1    2
## [2,]    1    2
## [3,]    1    2
## [4,]    1    2
## [5,]    1    2
## [6,]    1    2
```

```
centroids2=tapply(x, list(rep(dfnewdata2[,6], ncol(x)), col(x)),mean)

ecdist2<-matrix(1:12,nrow = 6, ncol = 3)
for (i in 1:6){
  ecdist2[i,1]=sqrt((x[i,1] - centroids2[1,1])^2 + (x[i,2] - centroids2[1,2])^2)
  ecdist2[i,2]=sqrt((x[i,1] - centroids2[2,1])^2 + (x[i,1] - centroids2[2,2])^2)
  if (ecdist2[i,1]>ecdist2[i,2]) {
    ecdist2[i,3]=2
  }else {
    ecdist2[i,3]=1
  }
}
newdata3 = cbind(newdata2, ecdist2)
dfnewdata3=data.frame(newdata3)
names(dfnewdata3)<- c("V1","V2","cluster","dist1_1","disct2_1","group_1","dist1_2","disct2_2","group_2"
dfnewdata3
```

```
##   V1 V2 cluster   dist1_1 disct2_1 group_1  dist1_2 disct2_2 group_2
## 1  1  4       1 5.1478151 1.820027       2 5.000000 2.687419       2
## 2  1  3       1 4.7434165 1.820027       2 4.472136 2.687419       2
## 3  0  4       1 6.0415230 3.132491       2 5.830952 3.726780       2
## 4  5  1       0 0.7071068 4.160829       1 0.000000 4.533824       1
## 5  6  2       0 0.7071068 5.550901       1 1.414214 5.821416       1
## 6  4  0       1 2.1213203 2.795085       1 1.414214 3.349959       1
```

Alternatively: #interation trying to do by loop j <- 0

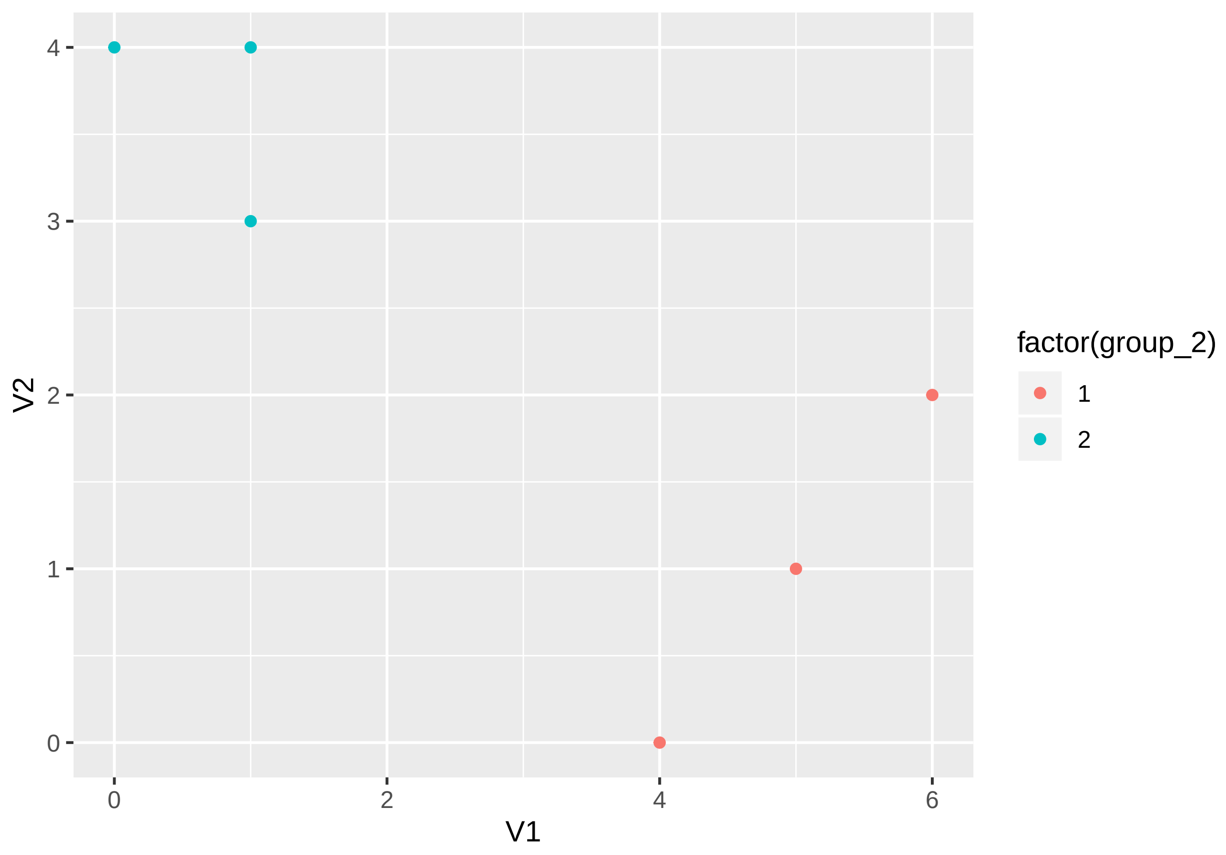
repeat { list(rep(dfnewdata2[,6+3*j], ncol(x)), col(x)) centroids=tapply(x, list(rep(dfnewdata2[,6+3*j], ncol(x)), col(x)),mean) dfnewdata2 <- cbind(dfnewdata2, matrix(0:0,nrow = 6, ncol = 3)) for (i in 1:6){ dfnewdata2[i,7+3*j]=sqrt((x[i,1] - centroids[1,1])^2 + (x[i,2] - centroids[1,2])^2) dfnewdata2[i,8+3*j]=sqrt((x[i,1] - centroids[2,1])^2 + (x[i,1] - centroids[2,2])^2) if (dfnewdata2[i,7+3*j]>dfnewdata2[i,8+3*j]) { dfnewdata2[i,9+3*j]=2 }else { dfnewdata2[i,9+3*j]=1 } } if (dfnewdata2[,9+3*j]==dfnewdata2[,6+3*j]){ break }

j=j+1

}

## 6

```
ggplot(dfnewdata3, aes(V1, V2, color=factor(group_2)))+geom_point()
```



### Clustering State Legislative Professionalism

#### 1

```r
load("C:/Users/zhouy/Desktop/Uchicago/uchicourse/Intro to Machine Learning/PS/PS4/Data and Codebook/leg
```

#### 2

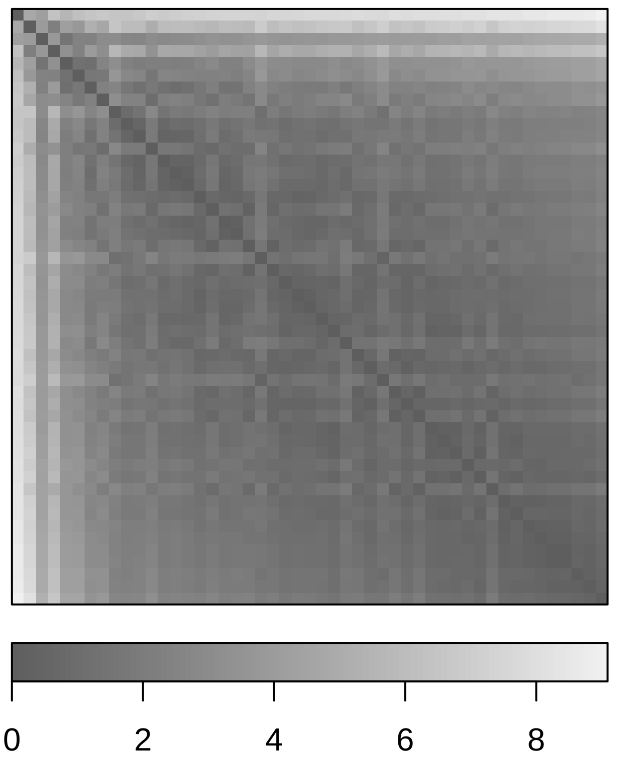
```r
df<-subset(x,x$sessid=="2009/10")
df= subset(df,select = c("state","t_slength","slength","salary_real","expend") )
df<-na.omit(df)
rownames(df) <- df$state
df_sub <- df%>%
  select(t_slength, slength, salary_real, expend) %>%
  scale()
```

#### 3

```r
library("seriation")
```

```
## Registered S3 method overwritten by 'seriation':
##   method         from
##   reorder.hclust gclus
```

```r
df_dist <- dist(df_sub)
dissplot(df_dist)
```



There exist two clusters. The larger cluster is represented by the black part at the right bottom. The larger cluster is represented by the black part at the top left.

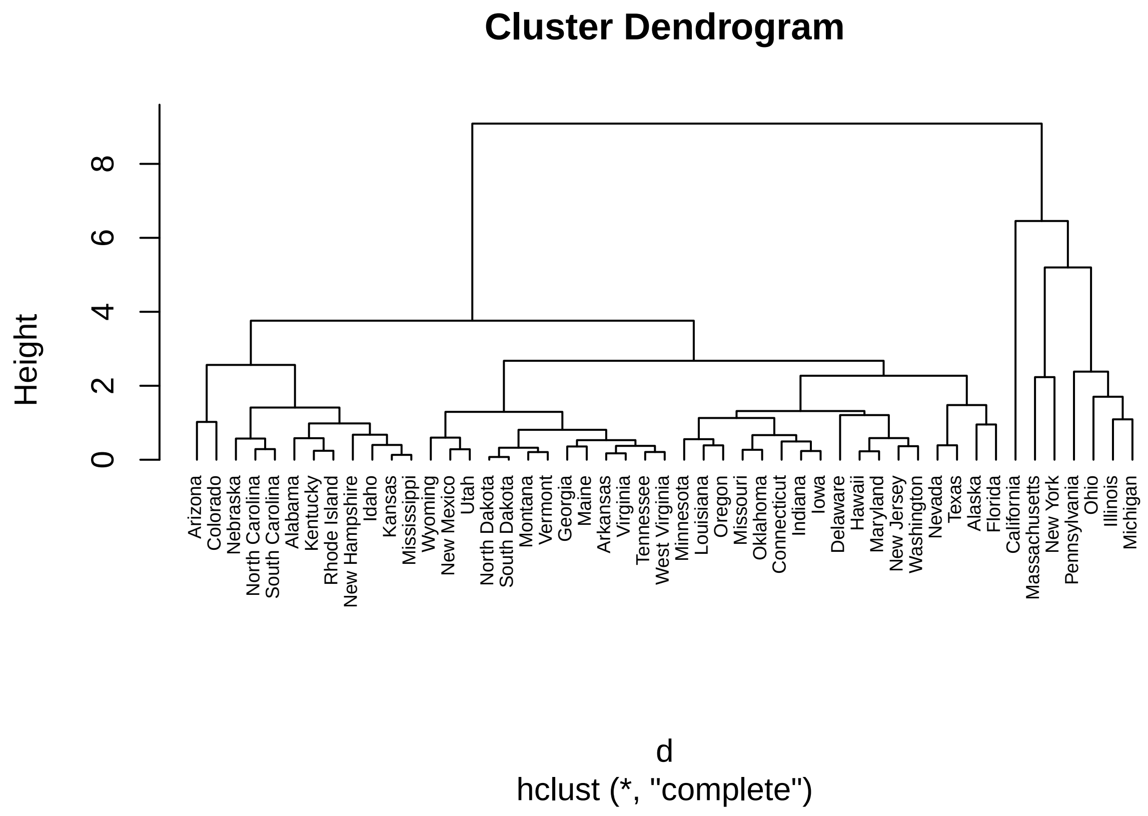
## 4

```r
library(tidyverse)
library(skimr)
library(dendextend)
d <- dist(df_sub, method = "euclidean")
hc1 <- hclust(d, method = "complete" )
hc1
```

```
##
```

```
## Call:
## hclust(d = d, method = "complete")
##
## Cluster method   : complete
## Distance         : euclidean
## Number of objects: 49
```

```
plot(hc1, cex = 0.6, hang = -1)
```



California, Massachusetts, New York, Pennsylvania, Ohio, Illinois, and Michigan constitue one cluster, and remainings states constitute the other cluster. One cluster contains fewer elements, and one contains more.

## 5

```
set.seed(1234)

kmeans<- kmeans(df_sub,
                centers = 2,
                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"
```

## 6

```
library(mixtools)
```

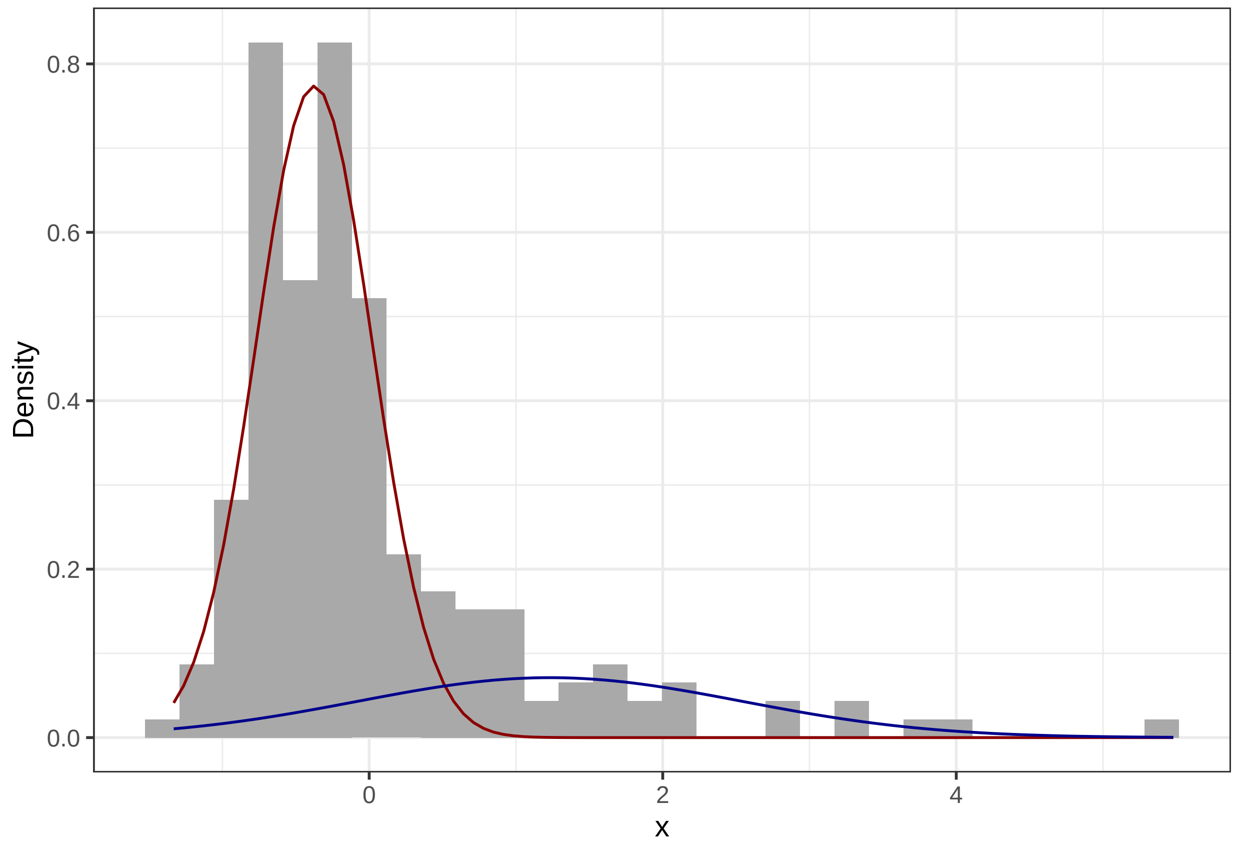
```
## mixtools package, version 1.2.0, Released 2020-02-05
## This package is based upon work supported by the National Science Foundation under Grant No. SES-051
```

```
library(plotGMM)
set.seed(7355)
gmm1 <- normalmixEM(df_sub, k = 2)
```

```
## number of iterations= 40
```

```
ggplot(data.frame(x = gmm1$x)) +
  geom_histogram(aes(x, ..density..), fill = "darkgray") +
  stat_function(geom = "line", fun = plot_mix_comps,
                args = list(gmm1$mu[1], gmm1$sigma[1], lam = gmm1$lambda[1]),
                colour = "darkred") +
  stat_function(geom = "line", fun = plot_mix_comps,
                args = list(gmm1$mu[2], gmm1$sigma[2], lam = gmm1$lambda[2]),
                colour = "darkblue") +
  xlab("x") +
  ylab("Density") +
  theme_bw()
```

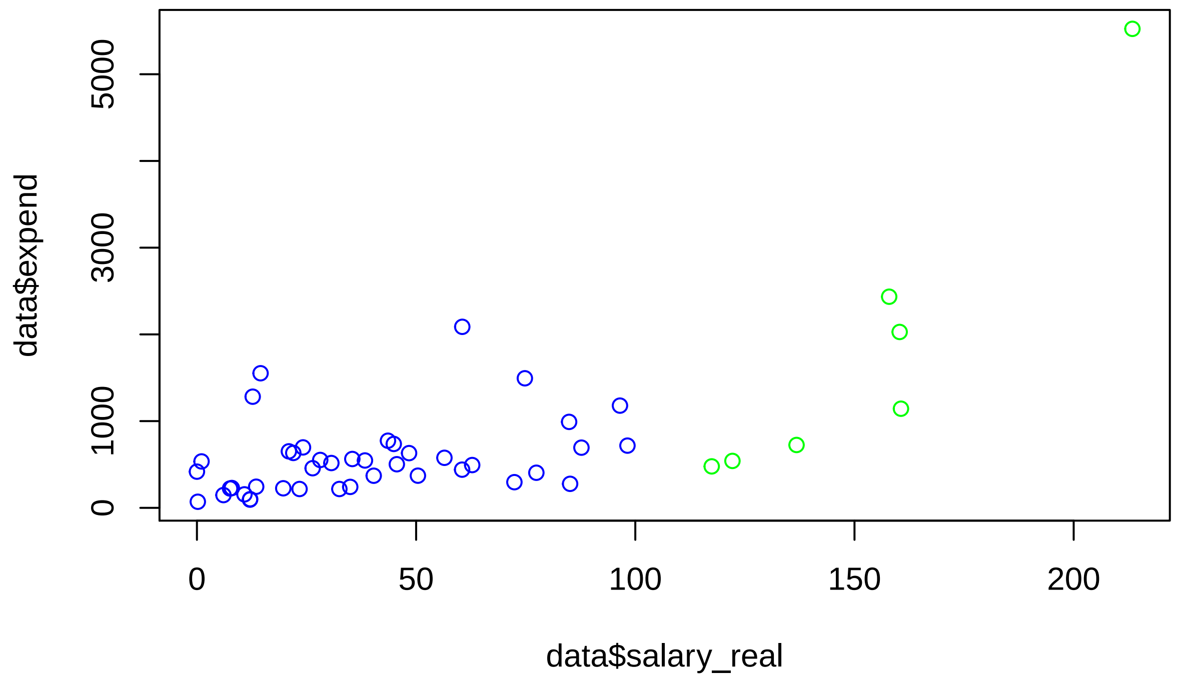
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



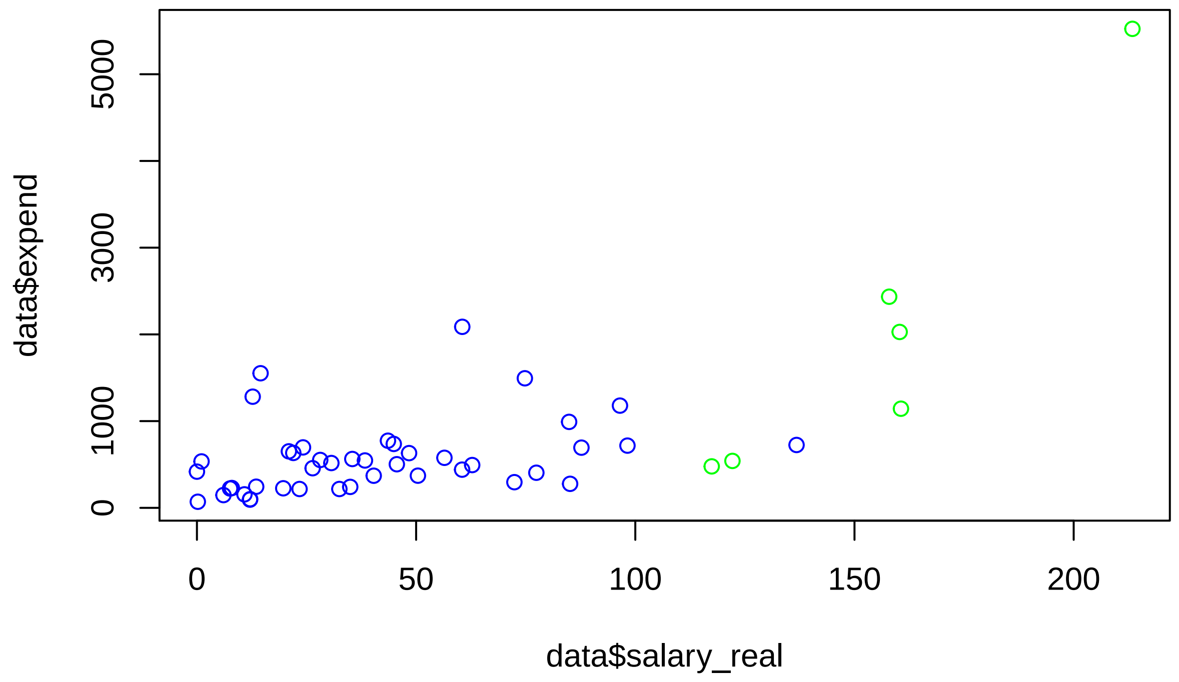
Current GMM doesn't fit well.

## 7

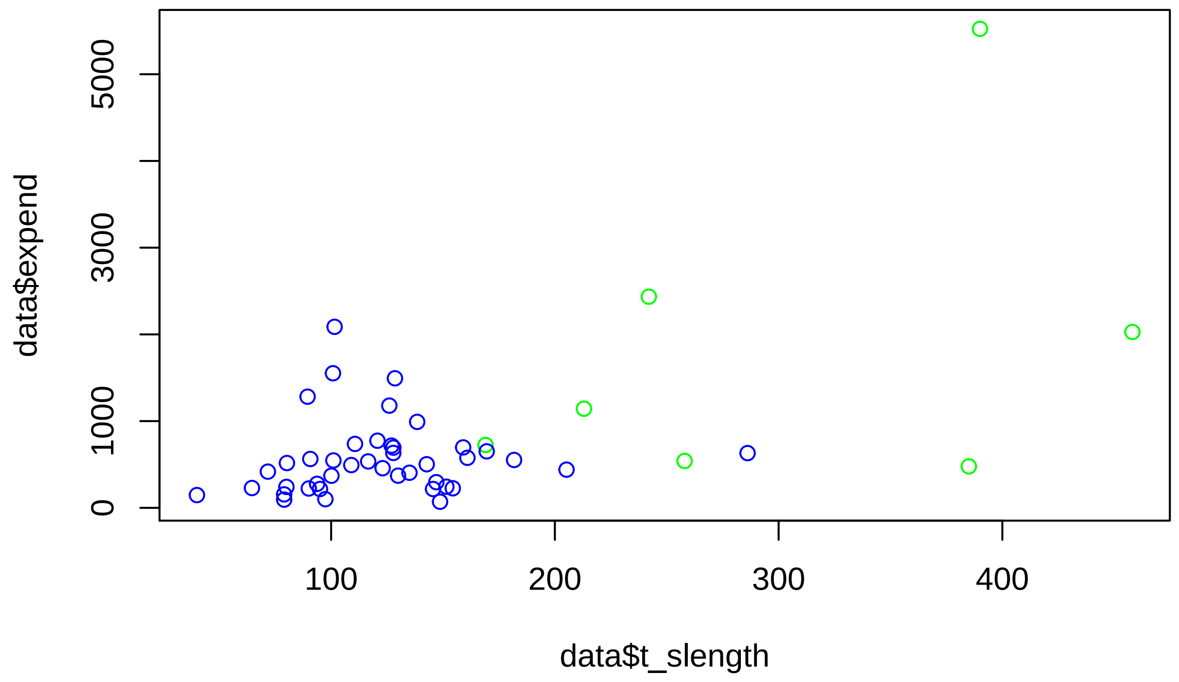
```
data <- add_column(df, hc_assign=1)
cg2 <- c("California", "Massachusetts", "New York", "Pennsylvania", "Ohio", "Illinois", "Michigan")
for (i in 1:nrow(data)) {
  for (s in cg2) {
    if (data$state[i]==s) {
      data$hc_assign[i]=2
    }
  }
}
kc <- as.table(kmeans$cluster)
kc <- data.frame(kc)
colnames(kc)[colnames(kc)=="Freq"] <- "km_assign"
colnames(kc)[colnames(kc)=="Var1"] <- "state"
rownames(kc) <- kc$state
data <- merge(data,kc,by="state")
#agglomerative hierarchical
plot(data$salary_real, data$expend,
     col = ifelse(data$hc_assign == 1,'blue',
           ifelse(data$hc_assign == 2, 'green','red')))
```



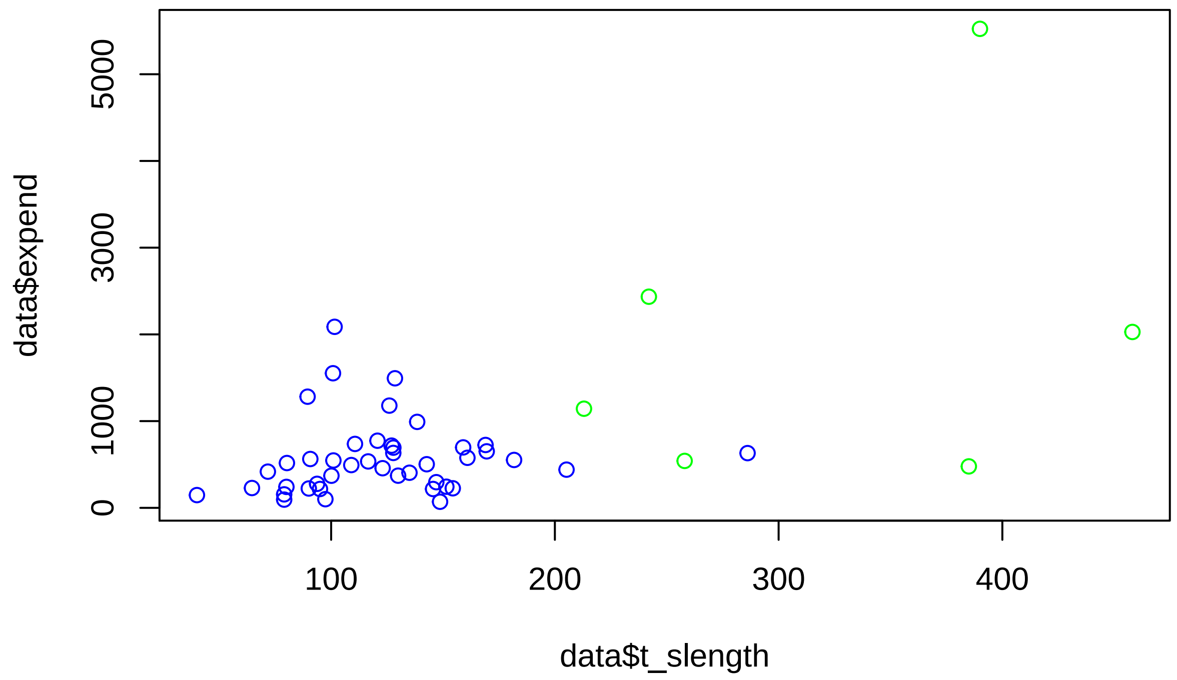
```
#k-means
plot(data$salary_real, data$expend,
     col = ifelse(data$km_assign == 1,'blue',
                  ifelse(data$km_assign == 2, 'green','red')))
```



```
#agglomerative hierarchical
plot(data$t_slength, data$expend,
     col = ifelse(data$hc_assign == 1,'blue',
           ifelse(data$hc_assign == 2, 'green','red')))
```



```
#k-means
plot(data$t_slength, data$expend,
     col = ifelse(data$km_assign == 1,'blue',
                  ifelse(data$km_assign == 2, 'green','red')))
```



The dendrogram and density plot are shown previously which indicate a larger cluster and a smaller cluster. Two plots of HAC and kmeans across the same two features are quite the same. HAC and kmeans give very similar results. The only difference is whether Illinois belongs to the larger or smaller cluster.

## 8

```
library(clValid)
library(mclust)
```

```
## Package 'mclust' version 5.4.5
## Type 'citation("mclust")' for citing this R package in publications.
```

```
##
## Attaching package: 'mclust'
```

```
## The following object is masked from 'package:mixtools':
##
##     dmvnorm
```

```
## The following object is masked from 'package:purrr':
##
##     map
```

```r
val <- clValid(df_sub, 2:5, validation = "internal", clMethods = c("hierarchical", "kmeans", "model"))
summary(val)
```

```
##
## Clustering Methods:
##  hierarchical kmeans model
##
## Cluster sizes:
##  2 3 4 5
##
## Validation Measures:
##                                  2       3       4       5
##
## hierarchical Connectivity   6.0869  6.9536 16.1885 18.6774
##              Dunn           0.3637  0.4371  0.2562  0.2836
##              Silhouette     0.6994  0.6711  0.4932  0.4440
## kmeans       Connectivity   8.4460 10.8960 16.1885 28.7437
##              Dunn           0.1735  0.2581  0.2562  0.1090
##              Silhouette     0.6458  0.6131  0.4932  0.3042
## model        Connectivity  10.7393 28.6119 39.0687 67.8401
##              Dunn           0.1522  0.0633  0.0225  0.0258
##              Silhouette     0.6314  0.2588  0.1861  0.0085
##
## Optimal Scores:
##
##              Score  Method       Clusters
## Connectivity 6.0869 hierarchical 2
## Dunn         0.4371 hierarchical 3
## Silhouette   0.6994 hierarchical 2
```

## 9

(1) What can you take away from the fit? From validation statistics, HAC gives overall largest Dunn and Silhouette, but lowest for Connectivity. GMM gives highest connectivity. kmeans is at middle.

(2) Which approach is optimal? And optimal at what value of k? HAC is optimal from the result of optimal scores in #8. The optimal value of k is 2 for Connectivity and Silhouette, is 3 for Dunn.

(3) What are reasons you could imagine selecting a technically "sub-optimal" clustering method, regardless of the validation statistics? Other reasons can be that the algorithms of the mothod has more meaningful interpretation or is suitable for division needs. For example, k-means is friendly to explain how to cluster according to the spacial distance. GMM is friendly to divide softly.