exam_1_clustering_edited

2023-03-23

Exam 1: section 1 - clustering

A run log of this code is provided as a PDF file. You may run this code, explore its actions, and add comments as you wish.

Based on class discussions and homework assignments, You should extend this code as needed in preparation for answering questions about the process presented here.

```
require(tidyverse)
```

```
## Loading required package: tidyverse
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.0     ✔ readr     2.1.4
## ✔ forcats   1.0.0     ✔ stringr   1.5.0
## ✔ ggplot2   3.4.1     ✔ tibble    3.1.8
## ✔ lubridate 1.9.2     ✔ tidyr     1.3.0
## ✔ purrr     1.0.1
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the ]8;;http://conflicted.r-lib.org/ conflicted package]8;; to force all confl
icts to become errors
```

```
require(data.table)
```

```
## Loading required package: data.table
##
## Attaching package: 'data.table'
##
## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
##
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
##
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
require(ggplot2)
require(factoextra)
```

```
## Loading required package: factoextra
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WB
a
```

Adding flags for the code

```
classif    <- c(1, 2, 3)
byCols     <- 2
byRows     <- 1
demo <- T

if(demo) {setwd("/Users/harikrishnadev/Library/Mobile Documents/com~apple~CloudDocs/Scho
ol Work/Sem 2/BUAN 6357/BUAN6357/exam_materials/exam_1")}
```

Reading training data

```
t1         <- fread(file="train1.dat")
t2         <- fread(file="test1.dat")
t1Grp      <- t1$grp; t1$grp     <- NULL
t2Grp      <- t2$grp; t2$grp     <- NULL
train1     <- t1
test1      <- t2
t1 <- as.data.frame(t1)
t2 <- as.data.frame(t2)
if(demo) {
  str(t1)
  str(t2)
  }
```

```
## 'data.frame':    135 obs. of  4 variables:
##  $ V1: num  4.8 4.7 5.2 5.1 4.3 6.3 5.7 6.1 6 6.1 ...
##  $ V2: num  3 3.2 2.7 3.8 3 3.3 2.8 3 2.2 2.8 ...
##  $ V3: num  1.4 1.6 3.9 1.6 1.1 4.7 4.5 4.6 4 4.7 ...
##  $ V4: num  0.3 0.2 1.4 0.2 0.1 1.6 1.3 1.4 1 1.2 ...
## 'data.frame':    15 obs. of  4 variables:
##  $ V1: num  6.4 5 6.4 7.3 4.8 6.1 4.6 5.6 6.5 5.4 ...
##  $ V2: num  3.2 3 2.7 2.9 3.1 3 3.4 3 3 3 ...
##  $ V3: num  5.3 1.6 5.3 6.3 1.6 4.9 1.4 4.5 5.5 4.5 ...
##  $ V4: num  2.3 0.2 1.9 1.8 0.2 1.8 0.3 1.5 1.8 1.5 ...
```

Adding max cluster value and seed value

```
maxGrp     <- 10
starts     <- 10
seed       <- 838216542
set.seed(seed)
```

Creating a functon which returns kmeans totwithss

```
myKmeans  <- function(seed,df,k,ns) {
             set.seed(seed)
             return(kmeans(df,k,ns)$tot.withinss)
             }
```

Getting kmean parameters for all cluster values

```
dt       <- data.table(idx=1:maxGrp, k=1:maxGrp)
kmss     <- dt[, .(wgss=myKmeans(seed,train1,k,starts)), by=.(idx)]
```

Creating a function to calculate the difference and scaled difference

```
dif1      <- function(df) {
             n  <- length(df)
             t1 <- df[1:(n-1)]-df[2:n]
             t2 <- t1/max(t1)
             return(list(d1=t1,d1scaled=t2))
             }
```

Calculate the diff and scaled diff for kmeans and plotting all the graphs to find ideal clusters

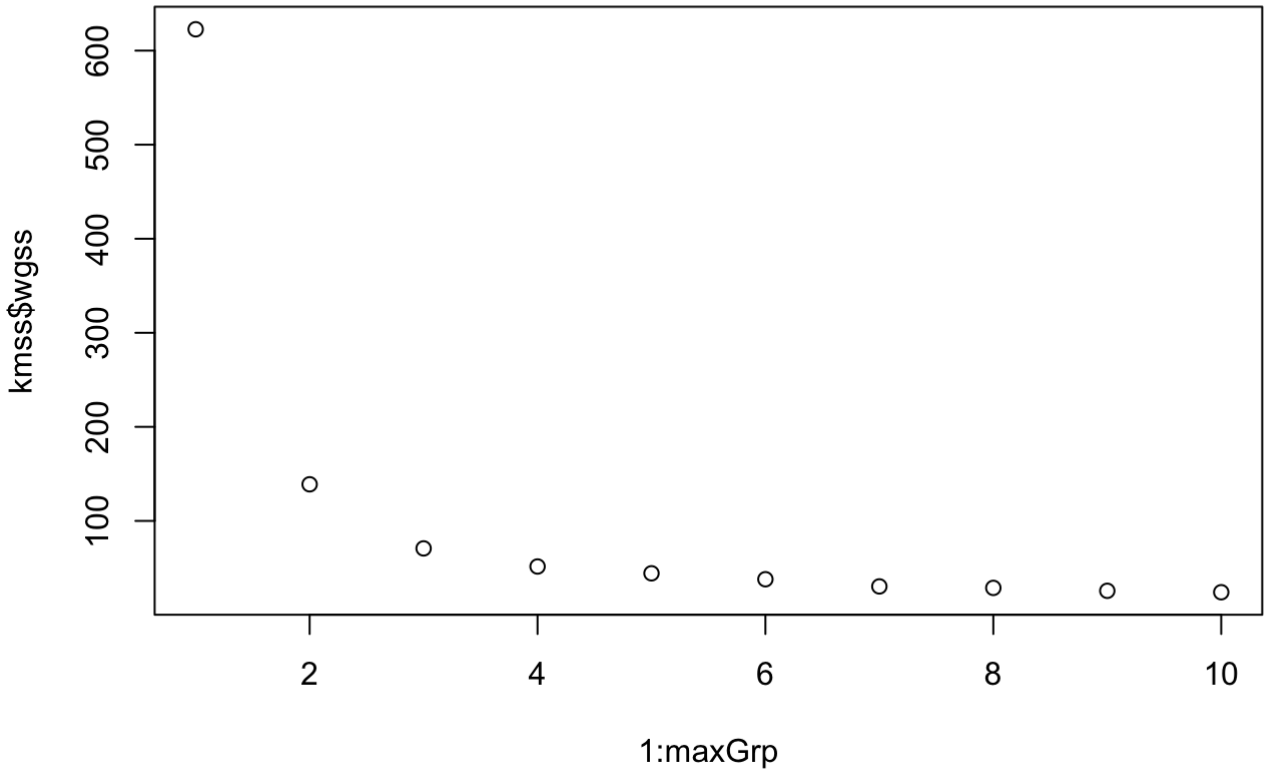
```
kmss$wgss
```

```
##  [1] 622.68519 138.90833  70.77430  51.53153  44.28929  37.98418  30.36137
##  [8]  28.78611  25.69147  24.17375
```

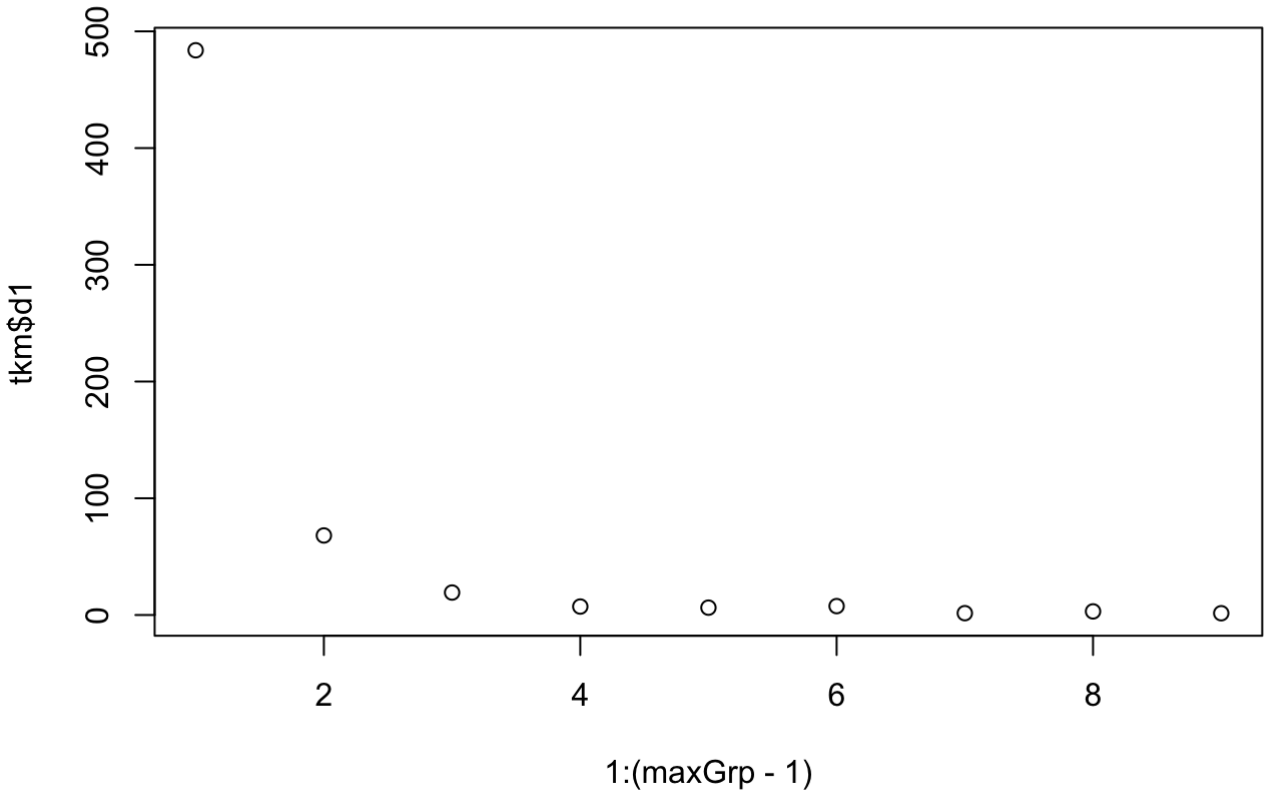
```
(tkm      <- dif1(kmss$wgss)  )
```

```
## $d1
## [1] 483.776852  68.134032  19.242776   7.242233   6.305115   7.622811   1.575254
## [8]   3.094645   1.517715
##
## $d1scaled
## [1] 1.000000000 0.140837726 0.039776140 0.014970192 0.013033107 0.015756874
## [7] 0.003256159 0.006396843 0.003137221
```

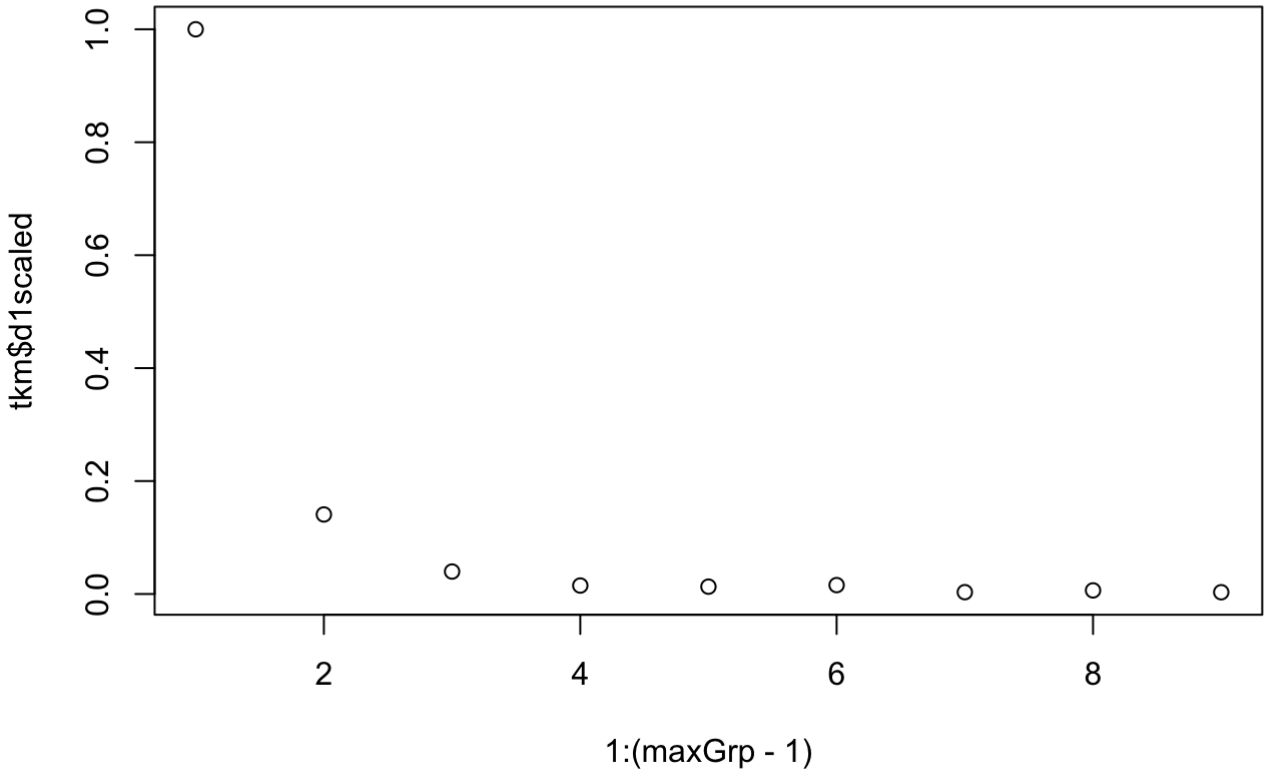
```
plot(1:maxGrp,kmss$wgss)
```




```
plot(1:(maxGrp-1),tkm$d1)
```

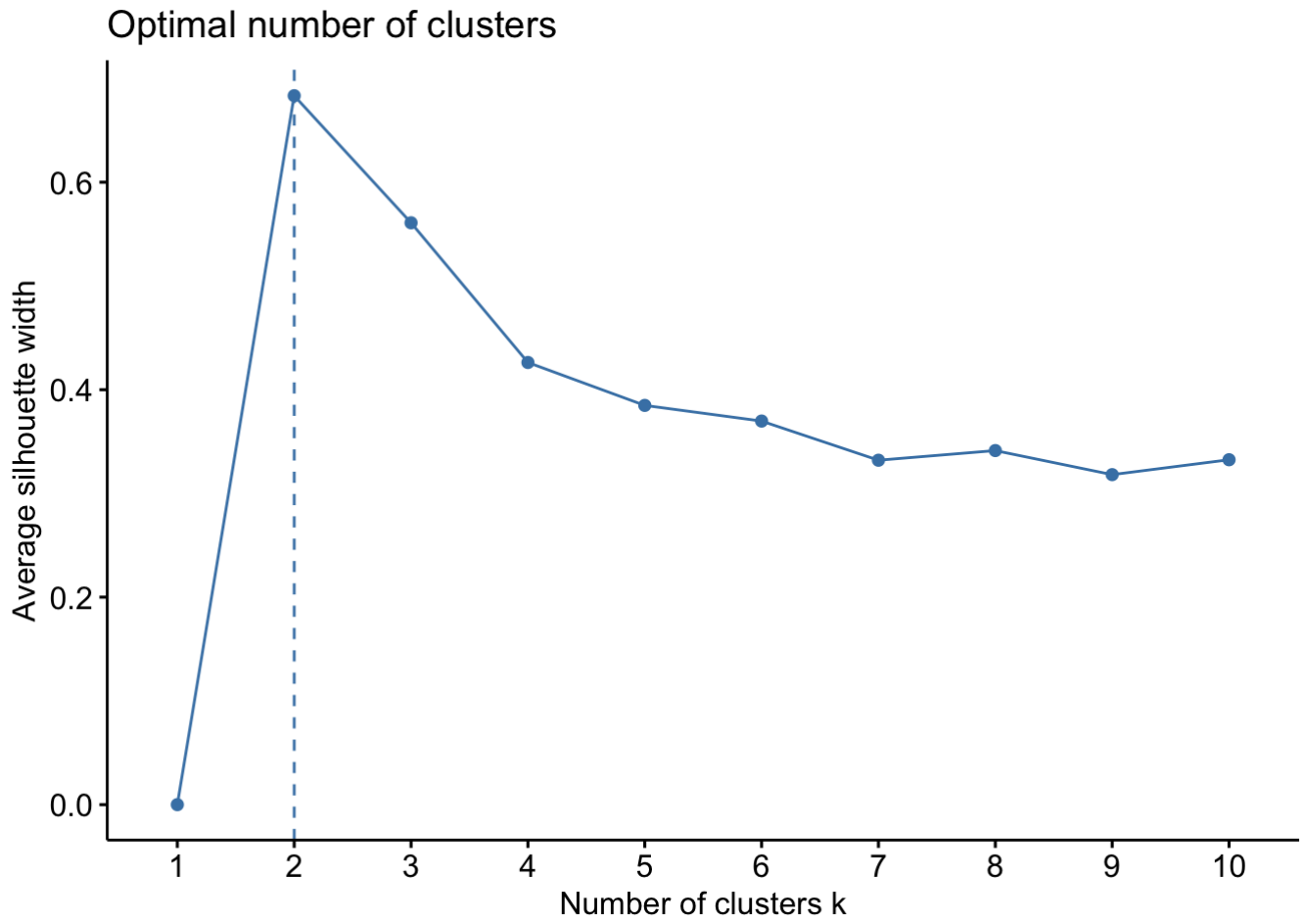



```
plot(1:(maxGrp-1),tkm$d1scaled)
```



Adding Silhouette analysis

```
fviz_nbclust(train1, kmeans, method='silhouette')
```



HClust clustering

adding parameters

```
set.seed(seed)

k        <- 3
km3      <- kmeans(train1,k,nstart=10)
km3clust <- km3$cluster
```

Creating a function which outputs mean, sd and vcinv

```
prepMHD <- function(df) {
     df$cluster <- NULL
     df$grps    <- NULL
     n          <- nrow(df)
     df2        <- scale(df, center=T, scale=T)
     vcvinv     <- solve(cov(df2))
     return( list(n      = n,
                  avg    = attr(df2, "scaled:center"),
                  sdev   = attr(df2, "scaled:scale"),
                  vcvinv = vcvinv )
           )
     }
```

Getting the covariance matrix for the clusters

```
t          <- train1
t$cluster  <- km3clust
kmMHwk     <- t                    %>%
              group_by(cluster)    %>%
              do(desc=prepMHD(select(.,V1,V2,V3,V4)))
```

```
kmDesc     <- kmMHwk$desc
kmDF       <- 4
nCl        <- 3
```

Getting the mahalanobis distance validations

```
kmTr       <- matrix(NA,nrow=nrow(train1),ncol=nrow(kmMHwk))
for ( i in 1:nrow(kmMHwk) ) {
    tD       <- kmDesc[[i]]
    tdf      <- scale(select(t,V1,V2,V3,V4), center=tD$avg, scale=tD$sdev)
    kmTr[,i] <- mahalanobis(tdf, center=F, cov=tD$vcvinv, inverted=T)
}
```

Getting the frequency distributions of the clusters

```
kmNew          <- apply(kmTr, byRows, which.min)
train1$mhCl    <- kmNew
train1$grp     <- t1Grp
train1$clust   <- km3clust
(tbl446        <-table(train1$grp,   train1$clust, dnn=c("grp"  ,"clust")) )
```

```
##    clust
## grp  1  2  3
##   1 46  0  0
##   2  0  2 44
##   3  0 31 12
```

```
(tbl452        <-table(train1$clust, train1$mhCl,  dnn=c("clust","mhCl" )) )
```

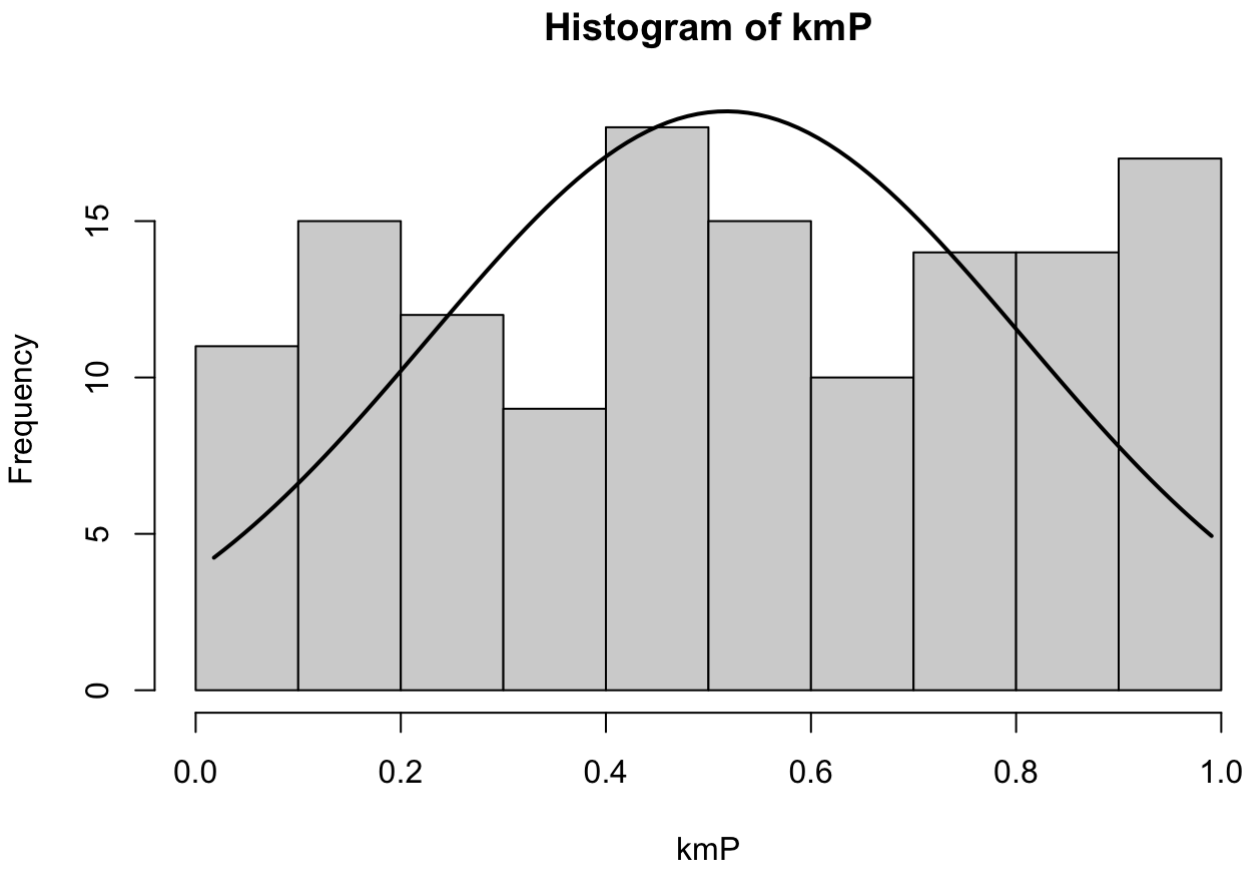
```
##      mhCl
## clust  1  2  3
##     1 46  0  0
##     2  0 31  2
##     3  0  2 54
```

```
kmStat         <- apply(kmTr, byRows, min)
kmP            <- pchisq(kmStat, df=kmDF, lower.tail=F)
summary(kmP)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.01781 0.27053 0.52350 0.51742 0.76194 0.99061
```

```
hist_data <- hist(kmP)
x_values <- seq(min(kmP), max(kmP), length = 100)
y_values <- dnorm(x_values, mean = mean(kmP), sd = sd(kmP))
y_values <- y_values * diff(hist_data$mids[1:2]) * length(kmP)

lines(x_values, y_values, lwd = 2)
```




```
kmTst          <- matrix(NA,nrow=nrow(test1),ncol=nrow(kmMHwk))
for ( i in 1:nrow(kmMHwk) ) {
    tD        <- kmDesc[[i]]
    tdf       <- scale(test1, center=tD$avg, scale=tD$sdev)
    kmTst[,i] <- mahalanobis(tdf, center=F, cov=tD$vcvinv, inverted=T)
    }

tstNew        <- apply(kmTst, byRows, which.min)
test1$mhCl    <- tstNew
test1$grp     <- t2Grp
(tbl472       <- table(test1$grp, test1$mhCl, dnn=c("grp","mhCl")) )
```

```
##    mhCl
## grp 1 2 3
##   1 4 0 0
##   2 0 0 4
##   3 0 5 2
```

Creating a function to determine percentage agree

```
pcAgree   <- function(tbl) {
  return ( sum(apply(tbl, byRows, max))/sum(tbl) )
}
```

Getting percent agree for all the matrices

```
#Training
# K-means vs grps
pcAgree(tbl446)
```

```
## [1] 0.8962963
```

```
# Mahanolobis vs K-means
pcAgree(tbl452)
```

```
## [1] 0.9703704
```

```
#Testing
# Mahanolobis vs grps
pcAgree(tbl472)
```

```
## [1] 0.8666667
```

The same exercise for the hclust

```
hcDat         <- train1
hcDat$mhCl    <- NULL
hcDat$grp     <- NULL
hcDat$clust   <- NULL

hc            <- hclust(dist(hcDat)^2,method="complete")

hcwgss        <- function(train,hc,i) {
      t1 <- cutree(hc,i)
      t2 <- data.table(idx=t1,j=1:nrow(train))
      t3 <- t2[, .(ss=sum(scale(train[j,], center=T, scale=F)^2)), by=.(idx)]
      return(sum(t3))
      }

(hcss      <- dt[,.(wgss=hcwgss(hcDat,hc,k)),by=.(idx)] )
```

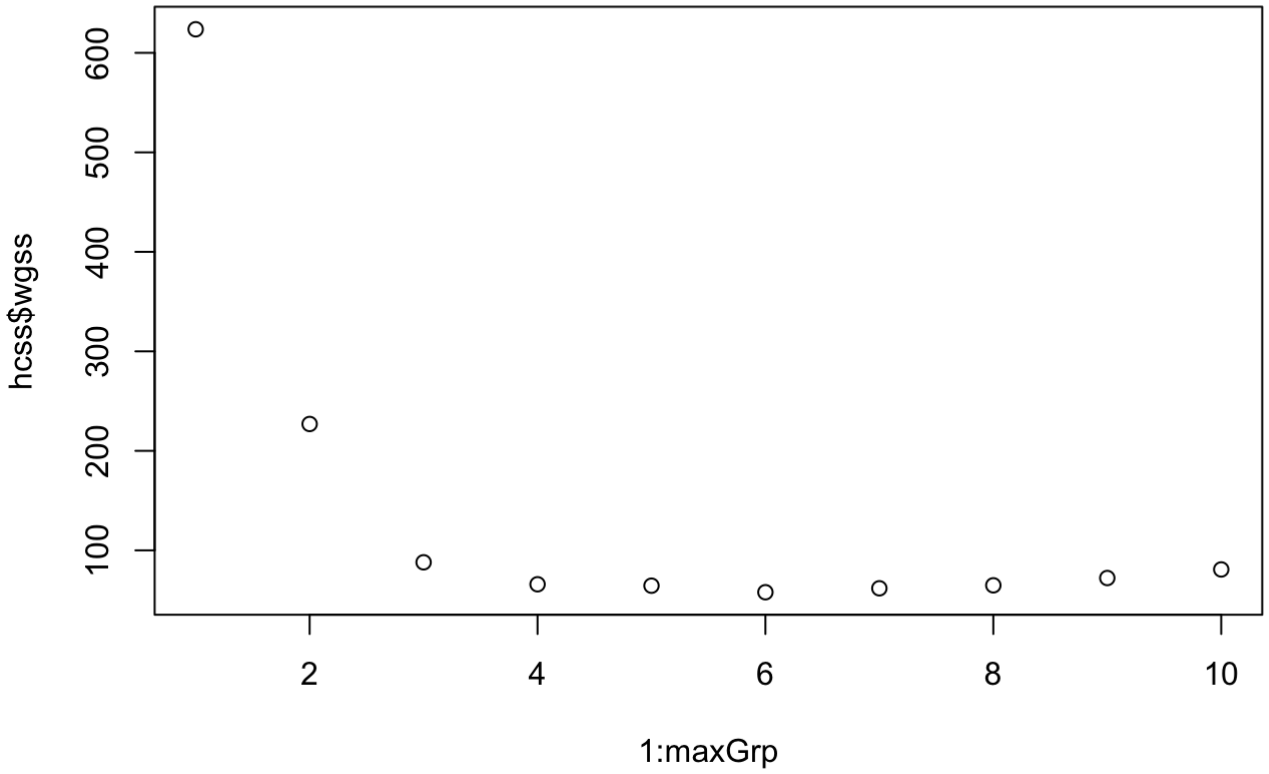
```
##     idx      wgss
##  1:   1 623.68519
##  2:   2 226.98258
##  3:   3  87.97811
##  4:   4  65.89673
##  5:   5  64.48006
##  6:   6  57.89971
##  7:   7  61.85718
##  8:   8  64.84383
##  9:   9  72.18458
## 10:  10  80.79708
```

```
(thc       <- dif1(hcss$wgss) )
```

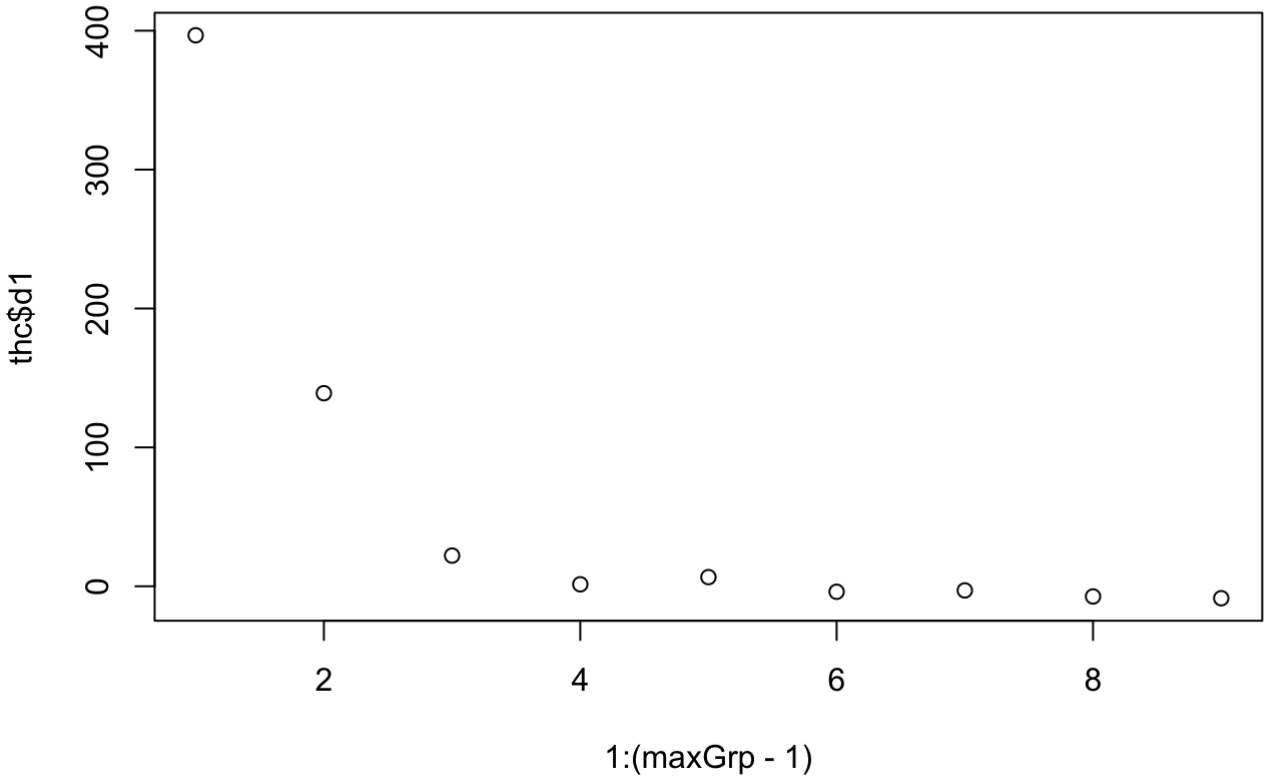
```
## $d1
## [1] 396.702606 139.004468  22.081385   1.416670   6.580346  -3.957465  -2.986657
## [8]  -7.340747  -8.612500
##
## $d1scaled
## [1]  1.000000000  0.350399684  0.055662313  0.003571114  0.016587605
## [6] -0.009975899 -0.007528705 -0.018504408 -0.021710218
```

Creating the graph of the total within ss and diffs

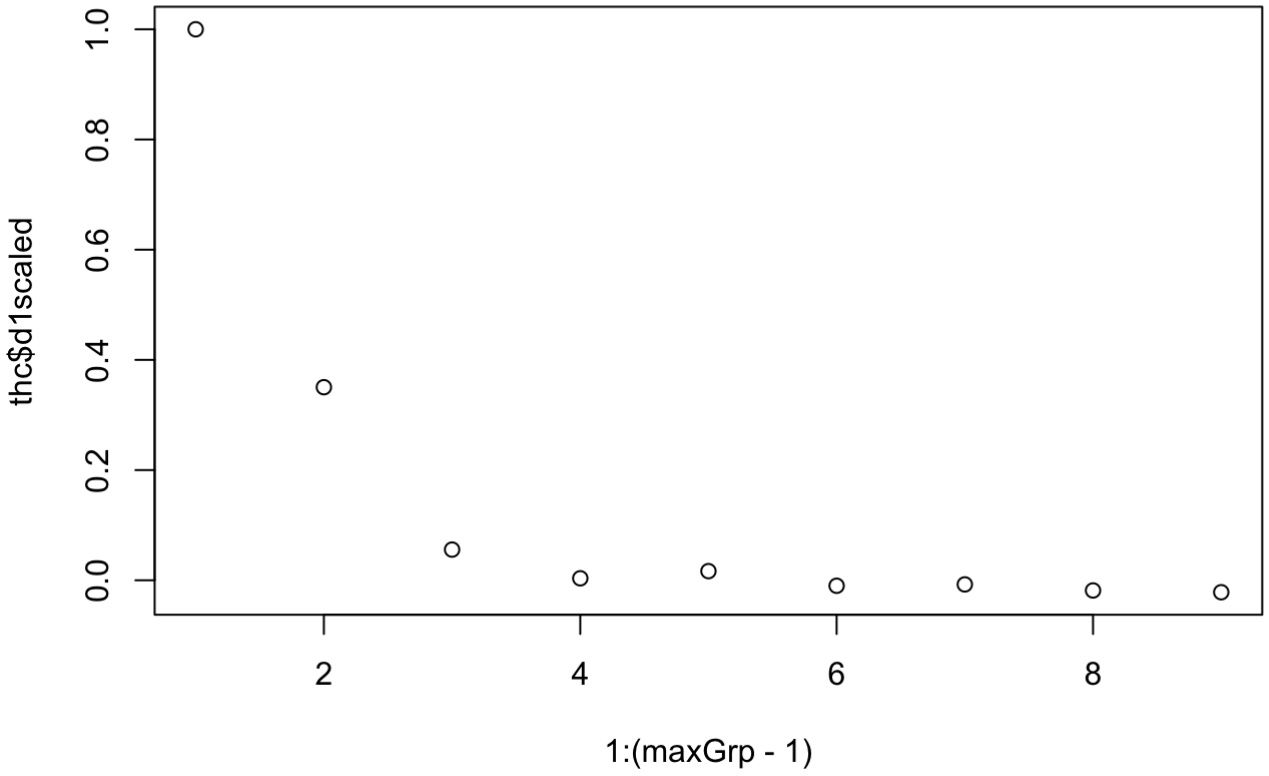
```
plot(1:maxGrp,hcss$wgss)
```




```
plot(1:(maxGrp-1),thc$d1)
```




```
plot(1:(maxGrp-1),thc$d1scaled)
```


thc$d1scaled
0.0
0.2
0.4
0.6
0.8
1.0
2
4
6
8
1:(maxGrp - 1)