Lab 5

Hong Xu

September 28, 2015

Task 1 & 2

```
myTennisDataD <- read.csv("Wimbledon.csv", stringsAsFactors = FALSE)
# join the first two columns into one string
playNames <- apply(myTennisDataD[, c(1:2)], 1, paste, collapse = "-")
myTennisDataD <- myTennisDataD[, -c(1,2)]
# put them as row names
rownames(myTennisDataD) <- playNames
# find the columns whose all values are NA
which(apply(myTennisDataD, 2, function(x) all(is.na(x))) == TRUE)
```

```
## TPW.1 TPW.2
##    17    35
```

```
# remove those columns
myTennisDataD <- myTennisDataD[, -c(17, 35)]
# find the columns that contains some NA
naCols <- which(apply(myTennisDataD, 2, function(x) any(is.na(x))) == TRUE)
# fill the missing value with median value
for (n in naCols){
  colMedian <- median(myTennisDataD[, n], na.rm = TRUE)
  myTennisDataD[, n] <- sapply(myTennisDataD[, n],
                               FUN = function(x) ifelse(is.na(x), colMedian, x))
}
iFinalTennisData <- myTennisDataD
```

Task 3

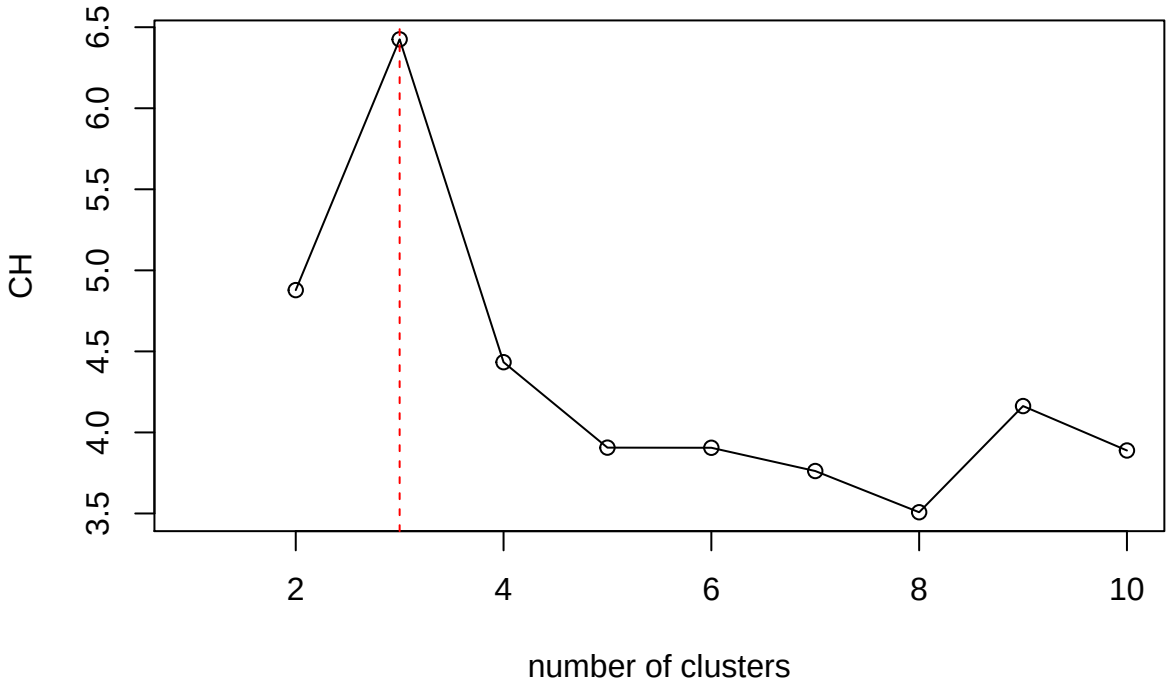
```
distTennisData <- dist(iFinalTennisData, method = "euclidean")
# take the log-euclidean distance
distTennisData <- log(distTennisData)
# start clustering and store the clustered objects
singleLinkage <- hclust(distTennisData, method = "single")
completeLinkage <- hclust(distTennisData, method = "complete")
```

Task 4

```
require(clusterCrit)
```

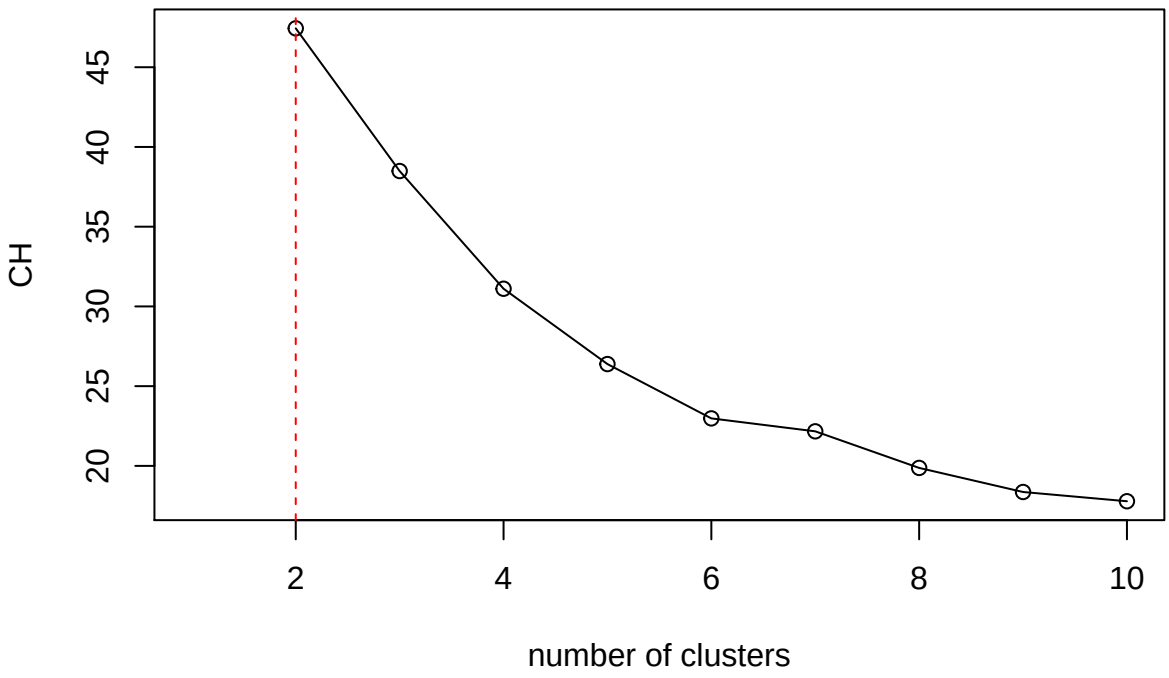
```
## Loading required package: clusterCrit
```

```r
findOptimalClustering <- function(c_object, df) {
  ##
  ## Input:
  ## c_object, the clustered object
  ## df, the original data frame
  ##
  ## Returns:
  ## maxCH, the maximum CH index computed.
  ##
  computeCH <- function(n_cluster, c_object, df) {
    ## a helper function that computes
    ## CH index for a given number of clusters
    part <- cutree(c_object, n_cluster)
    CH <- intCriteria(as.matrix(df), part, "Calinski_Harabasz")
    CH$calinski_harabasz
  }

  CH <- vector(mode="numeric", length=10)
  for (i in 1:10) {
    # for i = 1 the resulted value is NaN
    CH[i] <-  computeCH(i, c_object, df)
  }
  # make the plot
  plot(CH, xlab = "number of clusters")
  lines(CH)
  # find the maximum and add a vertical line
  maxCH <- max(CH, na.rm=T)
  n_max <- which(CH==maxCH)
  abline(v = n_max, col = "red", lty = 2)
  # return the maximum CH index
  return(maxCH)
}

findOptimalClustering(singleLinkage, iFinalTennisData)  # optimal n_cluster: 3
```




```
## [1] 6.425097
```

```
findOptimalClustering(completeLinkage, iFinalTennisData) # optimal n_cluster: 2
```

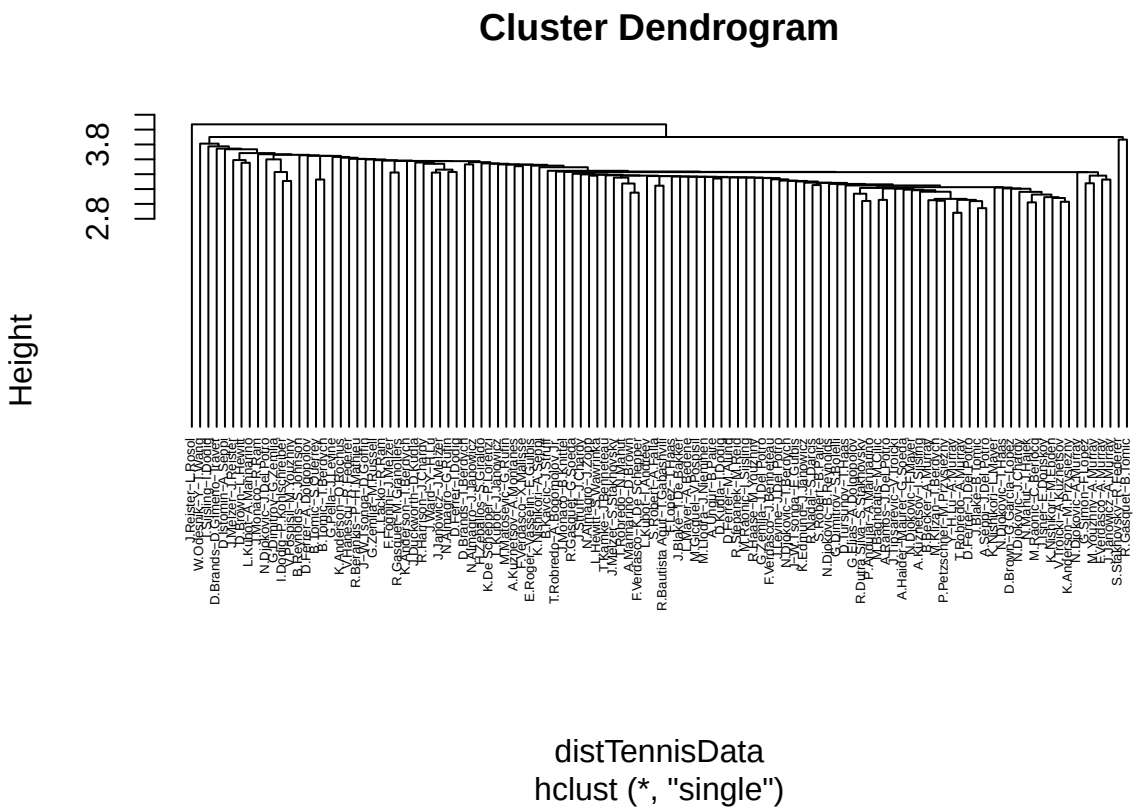



```
## [1] 47.4376
```

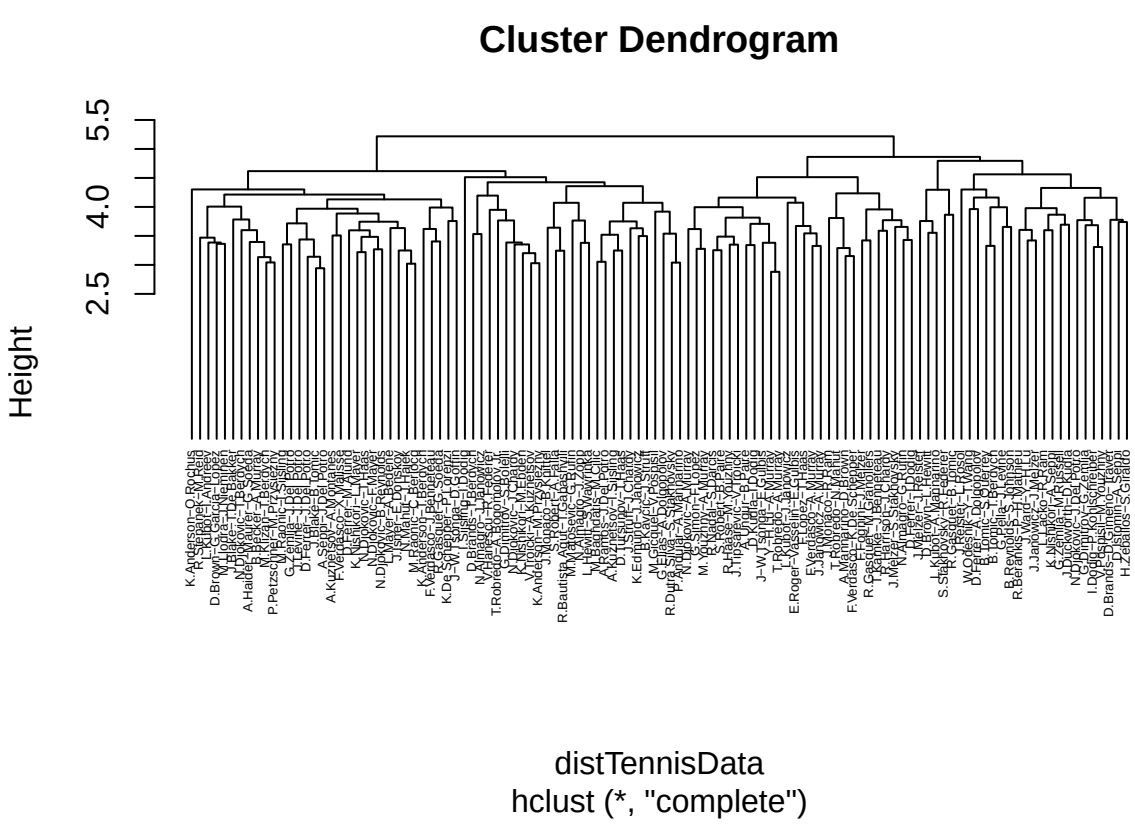
```
## CH Index value for 1 cluster is NaN, i.e. undefined.
```

Task 5 & 6

```
## The optimal number of clustering for
## single linkage is 3
## complete linkage is 2

plot(singleLinkage, cex = 0.4, hang = - 1)
```




```
plot(completeLinkage, cex = 0.4, hang = -1)
```




```
## Judging from the shape of dendrogram I tend to believe single linkage
## is a better option since the matches are incrementally grouped together
## at a small step, which reflect the fact that most games are similar.
## The results are more interesting as well:
## One cluster, comprised of two games, represents
## "the underdog beats top-seed" category:
## S.Stakhovsky-R.Federer and R.Gasquet-B.Tomic
## Another cluster of only one game, J.Reister-L.Rosol, is also kind of unique.
```