Clustering Applied to Tennis Data – Solutions

STA 325: Lab 5, Fall 2018

Today's agenda: Clustering Analysis

Programming partner's: You should have a programming partner for each lab, and you should switch off who is programming, and use each other for help. We will spend about 30–50 minutes per week on lab exercises and you will be expected to bring you laptops to class to work on these exercises in class. Myself and the TA's will be in class to help you.

In this lab, we'll look at tennis data. Let's read in the data and see what it looks like. Specifically, let's read in the file titled Wimbledon.csv into the variable myTennisDataD. In your read.csv() remember to set the stringsAsFactors() parameter to FALSE. We will remove the first two columns of the dataset, paste the names together, and replace the row numberings with these match names. (Now, let's look at the code that puts all these steps together in order to just read in the data and slightly re-shape it.)

```
# read in the data
myTennisDataD <- read.csv('Wimbledon.csv', stringsAsFactors = FALSE)
head(myTennisDataD)
```

```
##     Player1          Player2 Round Result FNL.1 FNL.2 FSP.1 FSW.1 SSP.1
## 1  B.Becker        A.Murray     1      0     0     3    59    29    41
## 2    J.Ward          Y-H.Lu     1      0     1     3    62    77    38
## 3   N.Mahut         J.Hajek     1      1     3     0    72    44    28
## 4 T.Robredo A.Bogomolov Jr.     1      1     3     0    77    40    23
## 5   R.Haase       M.Youzhny     1      0     0     3    68    61    32
## 6 M.Gicquel      V.Pospisil     1      0     0     3    59    41    41
##   SSW.1 ACE.1 DBF.1 WNR.1 UFE.1 BPC.1 BPW.1 NPA.1 NPW.1 TPW.1 ST1.1 ST2.1
## 1    14     5     1    26    18     5     1    28    19    NA     4     3
## 2    35    18     4    60    28    13     1    27    19    NA     7     4
## 3    10    17     3    41    18     8     5    26    17    NA     6     6
## 4    12     6     0    25    11    14     5    14    11    NA     6     6
## 5    15     7     2    32    29     2     0    29    20    NA     4     5
## 6    27     7     6    22    28     6     1    11     6    NA     3     2
##   ST3.1 ST4.1 ST5.1 FSP.2 FSW.2 SSP.2 SSW.2 ACE.2 DBF.2 WNR.2 UFE.2 BPC.2
## 1     2    NA    NA    57    39    43    20    11     2    38    16    10
## 2     6     6    NA    67    85    33    31    12     3    57    32    15
## 3     6    NA    NA    70    34    30    14     4     0    24    13     1
## 4     6    NA    NA    79    35    21     8     1     4    16    27     0
## 5     5    NA    NA    67    53    33    17     9     3    40    26    21
## 6     6    NA    NA    70    56    30    11    25     3    53    30    12
##   BPW.2 NPA.2 NPW.2 TPW.2 ST1.2 ST2.2 ST3.2 ST4.2 ST5.2
## 1     5    23    17    NA     6     6     6    NA    NA
## 2     2    46    39    NA     6     6     7     7    NA
## 3     0    19    12    NA     2     4     3    NA    NA
## 4     0    22    13    NA     2     2     4    NA    NA
## 5     3    44    30    NA     6     7     7    NA    NA
## 6     4    33    26    NA     6     6     7    NA    NA
```

```
dim(myTennisDataD)
```

```
## [1] 114  42
```

```
# remove named columns
myTennisDataC <- myTennisDataD[,-1:-2]
```

```
head(myTennisDataC)
```

```
##   Round Result FNL.1 FNL.2 FSP.1 FSW.1 SSP.1 SSW.1 ACE.1 DBF.1 WNR.1 UFE.1
## 1     1      0     0     3    59    29    41    14     5     1    26    18
## 2     1      0     1     3    62    77    38    35    18     4    60    28
## 3     1      1     3     0    72    44    28    10    17     3    41    18
## 4     1      1     3     0    77    40    23    12     6     0    25    11
## 5     1      0     0     3    68    61    32    15     7     2    32    29
## 6     1      0     0     3    59    41    41    27     7     6    22    28
##   BPC.1 BPW.1 NPA.1 NPW.1 TPW.1 ST1.1 ST2.1 ST3.1 ST4.1 ST5.1 FSP.2 FSW.2
## 1     5     1    28    19    NA     4     3     2    NA    NA    57    39
## 2    13     1    27    19    NA     7     4     6     6    NA    67    85
## 3     8     5    26    17    NA     6     6     6    NA    NA    70    34
## 4    14     5    14    11    NA     6     6     6    NA    NA    79    35
## 5     2     0    29    20    NA     4     5     5    NA    NA    67    53
## 6     6     1    11     6    NA     3     2     6    NA    NA    70    56
##   SSP.2 SSW.2 ACE.2 DBF.2 WNR.2 UFE.2 BPC.2 BPW.2 NPA.2 NPW.2 TPW.2 ST1.2
## 1    43    20    11     2    38    16    10     5    23    17    NA     6
## 2    33    31    12     3    57    32    15     2    46    39    NA     6
## 3    30    14     4     0    24    13     1     0    19    12    NA     2
## 4    21     8     1     4    16    27     0     0    22    13    NA     2
## 5    33    17     9     3    40    26    21     3    44    30    NA     6
## 6    30    11    25     3    53    30    12     4    33    26    NA     6
##   ST2.2 ST3.2 ST4.2 ST5.2
## 1     6     6    NA    NA
## 2     6     7     7    NA
## 3     4     3    NA    NA
## 4     2     4    NA    NA
## 5     7     7    NA    NA
## 6     6     7    NA    NA
```

```
# rename rows with removed columns
row.names(myTennisDataC) <- as.character(apply(myTennisDataD[,1:2], 1,
          paste, collapse = " "))
```

Lab Tasks

1. There are two columns in your dataset that contain only NA values. Remove these. For all other columns that contain missing values, replace the missing values with the median values for that column. Store this cleaned dataset into the variable iFinalTennisData.

```
# remove those columns that are entirely missing
all.miss <- which(apply(myTennisDataC,
  2, function(x){sum(is.na(x)==TRUE)}) ==
  dim(myTennisDataC)[1])

finalTennisData <- myTennisDataC[,-all.miss]

# impute with the median value
iFinalTennisData <- apply(finalTennisData, 2,
    function(x){x[is.na(x)] <- median(x,
    na.rm=TRUE); return(x)})
```

2. Create a log-Euclidean distance matrix for your data and perform single linkage and complete linkage clustering on the data and store these in the variables singeLinkage and completeLinkage respectively.

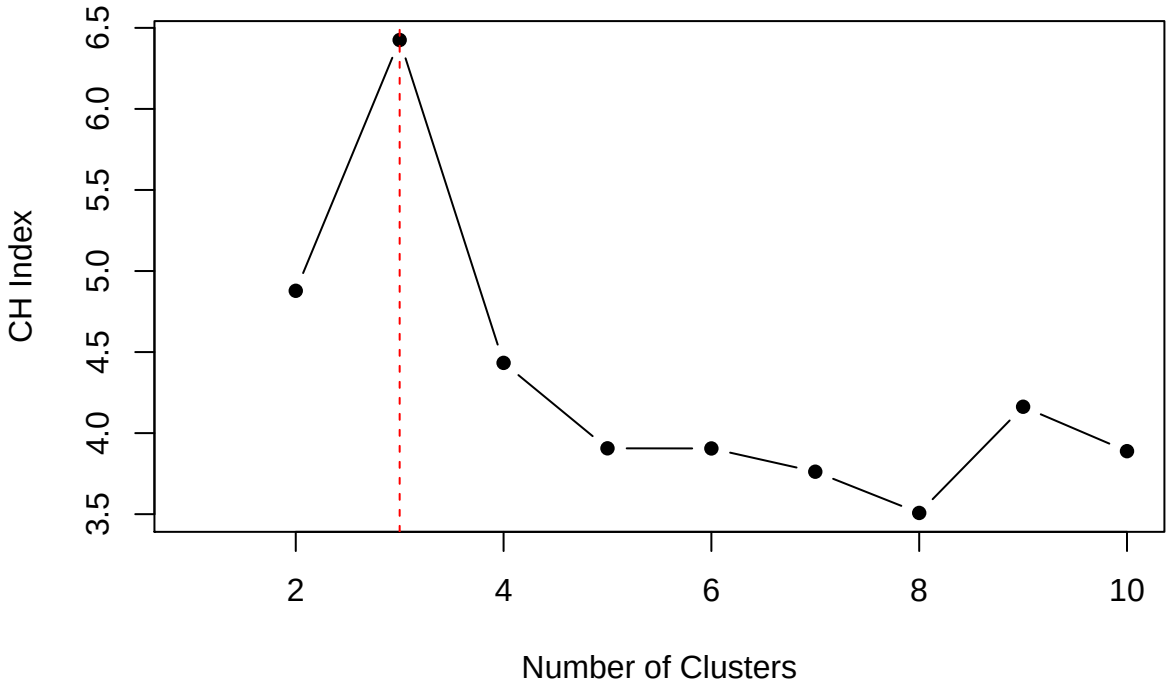
```r
# create a logged distance matrix
distance <- log(dist(iFinalTennisData))
# perform the clustering
singleLinkage <- hclust(distance, method = 'single')
completeLinkage <- hclust(distance, method = 'complete')
```

3. Using the command intCriteria() in the clusterCrit package, write a function that takes as its inputs the number of clusters to be tested (say 10), a clustered object (e.g SingleLinkage from above) and the original data frame (e.g. iFinalTennisData), and computes the CH Index for each assumed number of clusters and creates a plot of the resulting CH indices for each cluster count with a dotted vertical line indicating the maximum value. Also return the maximum CH Index computed. Test this function for up to 10 clusters with both single and complete linkage on iFinalTennisData, and include the maximum CH Index calculated and the related plots. What should the CH Index value be when the number of clusters is 1 and why?

```r
# Input: max.cuts, clustering object
# DF (dataset)
# Output: plots the max CH index
# Function to calculate the
# optimal number of clusters

myCHIndexPlot <- function(max.cuts = 10,
  clustering, DF = iFinalTennisData){
  # require the package
    require(clusterCrit)
  # for memory management
    CHIndex <- vector('numeric', length = 10)
    # loop through, cut the tree and calculate
    for(i in 1:10){
        CHIndex[i] <- intCriteria(DF,
         cutree(clustering, k = i), 'Calinski_Harabasz')
    }
    # store
    CHIndex <- as.numeric(CHIndex)
    # create the plot
    plot(1:max.cuts, CHIndex, type = "b", pch=16,
    xlab = 'Number of Clusters', ylab = 'CH Index')
    abline(v = which(CHIndex == max(CHIndex, na.rm = TRUE)),
      lty = 2, col = 'red')
    return(max(CHIndex, na.rm = TRUE))
}
# call the function
# single linkage case
res1 <- myCHIndexPlot(clustering = singleLinkage)
```

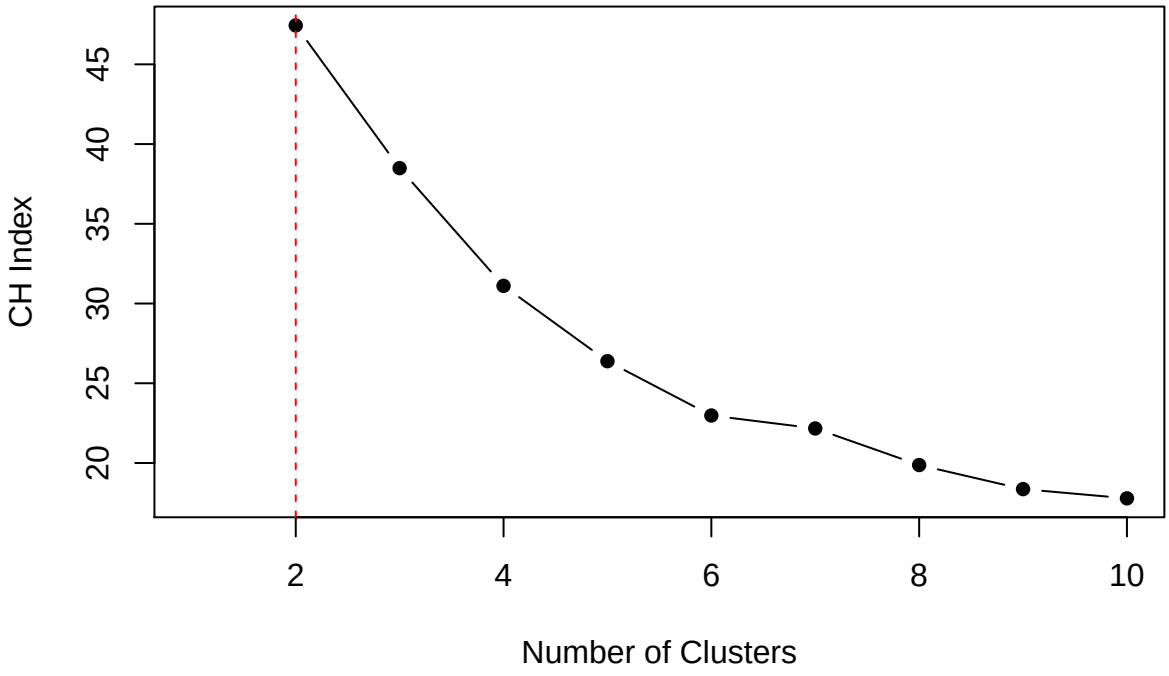
```
## Loading required package: clusterCrit
```




```
print(res1)
```

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

```
# complete linkage case
res2 <- myCHIndexPlot(clusterin  = completeLinkage)
```




```
print(res2)
```

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

There should be no CH index at the first clustering since everything is in the same cluster so the between

cluster variation does not exist!

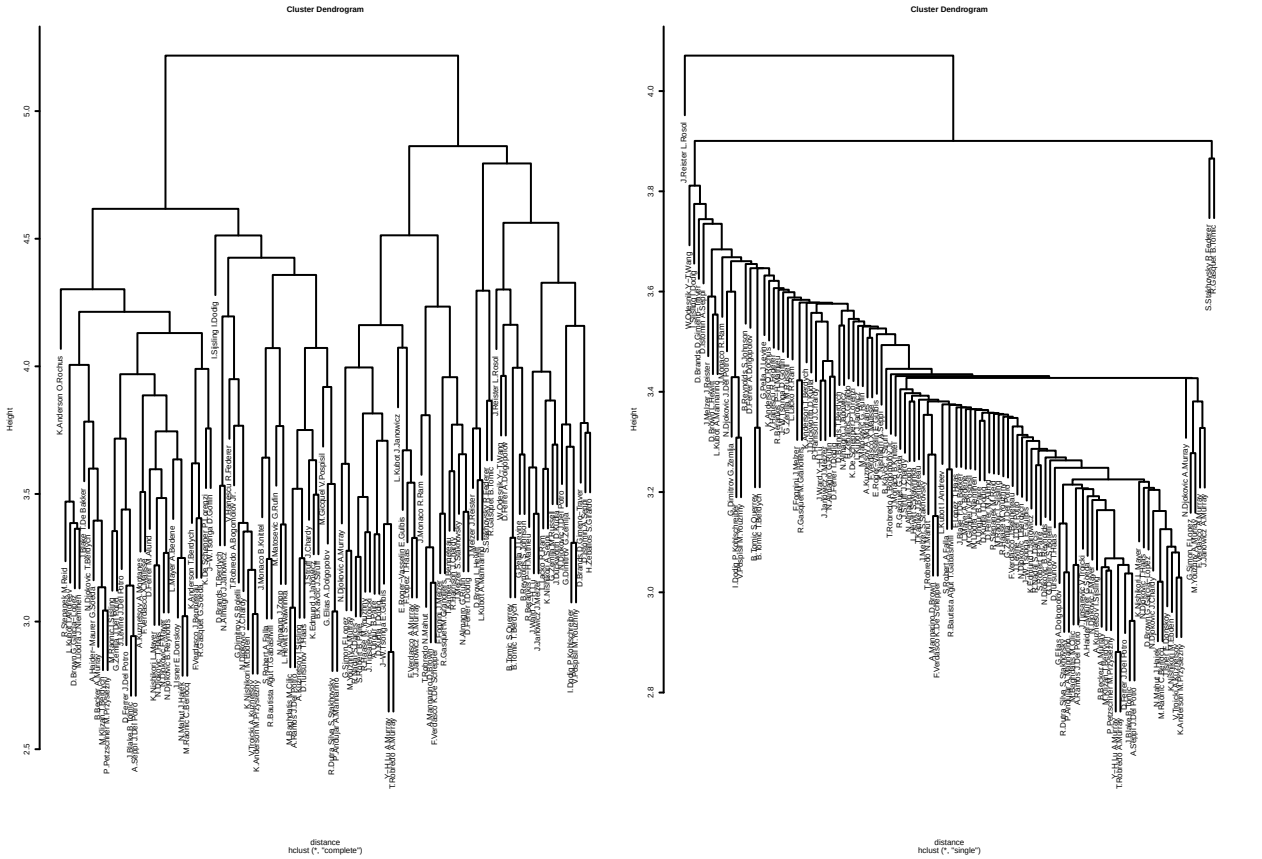
4. For each clustering (based on linkage) what is the optimal number of clusters?

For single and complete linkage respectively, the optimal number of clusters based upon CH index plots (from part 3) is 3 and 2.

5. Create a dendogram for each of your two clustering assignments. Based on the data type, what is the most appropriate type of clustering and why?

We can create the dendograms using the following code:

```
# create the matrix
par(mfrow=c(1,2))
# set font size and type
par(cex = 0.22, font = 1)
# create the dendograms
plot(completeLinkage)
plot(singleLinkage)
```



In this setup, complete linkage clustering appears to be optimal, which creates groups of matches that are more expansive and clearly defined than with single linkage clustering.