How to choose the number of clusters K?

Rebecca C. Steorts, Duke University

STA 325, Chapter 14.3 ESL (and ISL 10.3)

Agenda

- K-means versus Hierarchical clustering
- How to chooose the number of clusters
- CH index
- Gap statistic

From K-means to Hierarchical clustering

Recall two properties of K-means clustering:

1. It fits exactly K clusters (as specified)
2. Final clustering assignment depends on the chosen initial cluster centers

- Assume pairwise dissimilarites d_{ij} between data points.
- Hierarchical clustering produces a consistent result, without the need to choose initial starting positions (number of clusters).

Catch: choose a way to measure the dissimilarity between groups, called the linkage

- Given the linkage, hierarchical clustering produces a sequence of clustering assignments.
- At one end, all points are in their own cluster, at the other end, all points are in one cluster

How many clusters?

Sometimes, using K-means, K-medoids, or hierarchical clustering, we might have no problem specifying the number of clusters K ahead of time, e.g.,

- Segmenting a client database into K clusters for K salesman
- Compressing an image using vector quantization, where K controls the compression rate

Other times, K is implicitly defined by cutting a hierarchical clustering tree at a given height

But in most exploratory applications, the number of clusters K is unknown. So we are left asking the question: what is the "right" value of K?

This is a hard problem

Determining the number of clusters is a hard problem!

Why is it hard?

- Determining the number of clusters is a hard task for humans toperform (unless the data are low-dimensional). Not only that, it's just as hard to explain what it is we're looking for. Usually, statistical learning is successful when at least one of these is possible

Why is it important?

- E.g., it might mean a big difference scientifically if we were convinced that there were $K = 2$ subtypes of breast cancer vs. $K = 3$ subtypes
- One of the (larger) goals of data mining/statistical learning is automatic inference; choosing K is certainly part of this

Reminder: within-cluster variation

We're going to focus on K-means, but most ideas will carry over to other settings

Recall: given the number of clusters K, the K-means algorithm approximately minimizes the within-cluster variation:

$$W = \sum_{k=1}^{K} \sum_{C(i)=k} \|X_i - \bar{X}_k\|_2^2$$

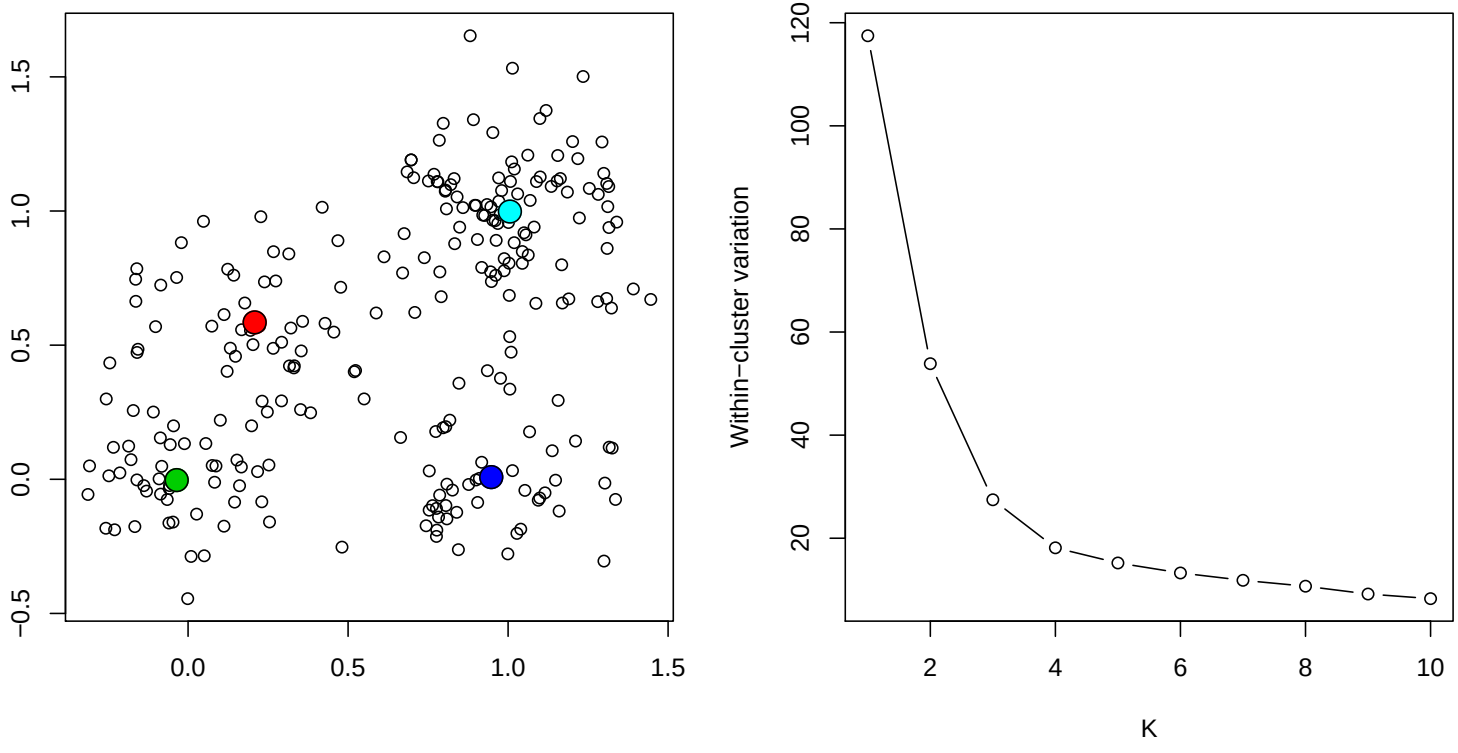
over clustering assignments C, where $\bar{X}_k$ is the average of points in group k, $\bar{X}_k = \frac{1}{n_k} \sum_{C(i)=k} X_i$

Clearly a lower value of W is better. So why not just run K-means for a bunch of different values of K, and choose the value of K that gives the smallest $W(K)$?

That's not going to work

Problem: within-cluster variation just keeps decreasing

Example: $n = 250$, $p = 2$, $K = 1, \ldots 10$



Between cluster variation

Within-cluster variation measures how tightly grouped the clusters are. As we increase the number of clusters K, this just keeps going down. What are we missing?

Between-cluster variation measures how spread apart the groups are from each other:

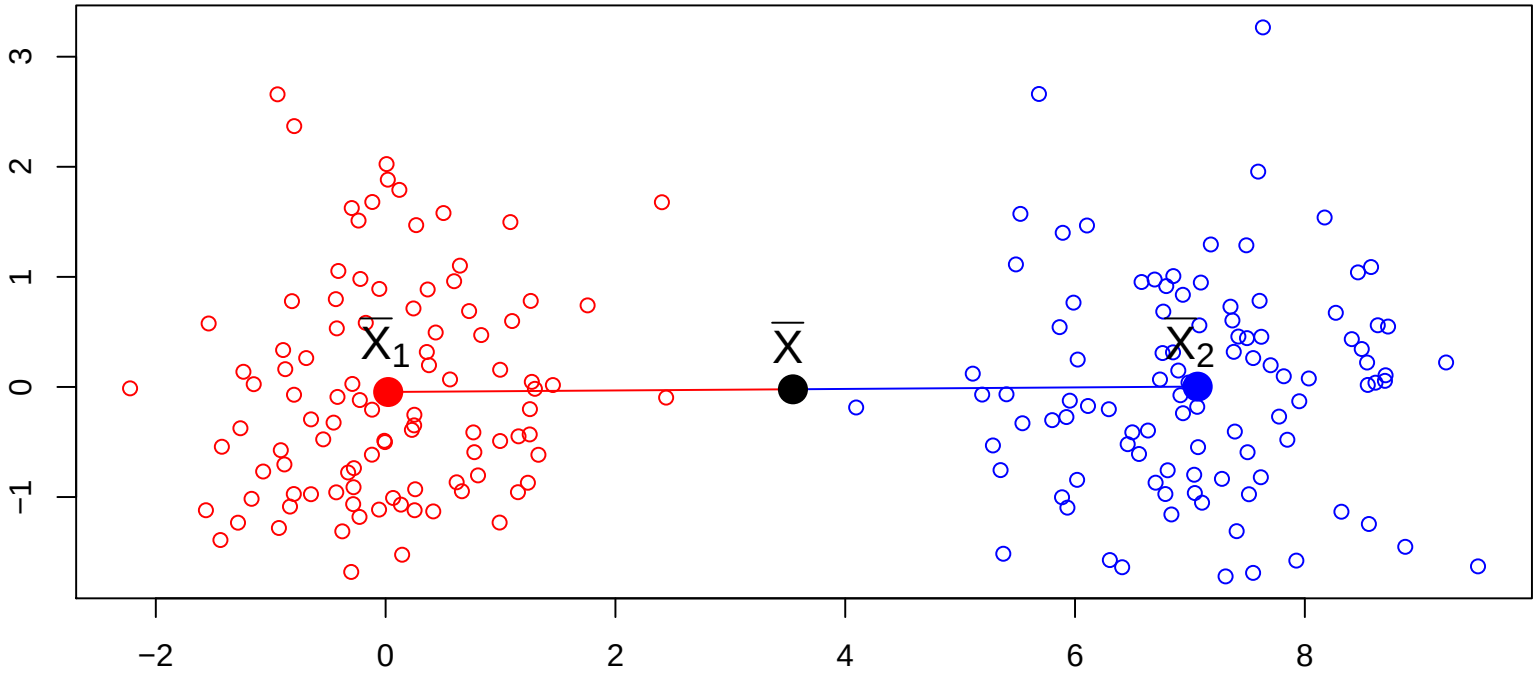
$$B = \sum_{k=1}^{K} n_K \|\bar{X}_k - \bar{X}\|_2^2$$

where as before $\bar{X}_k$ is the average of points in group k, and $\bar{X}$ is the overall average, i.e.

$$\bar{X}_k = \frac{1}{n_k} \sum_{C(i)=k} X_i \quad \text{and} \quad \bar{X} = \frac{1}{n} \sum_{i=1}^{n} X_i$$

Example: between cluster variation

Example: $n = 100,\ p = 2,\ K = 2$



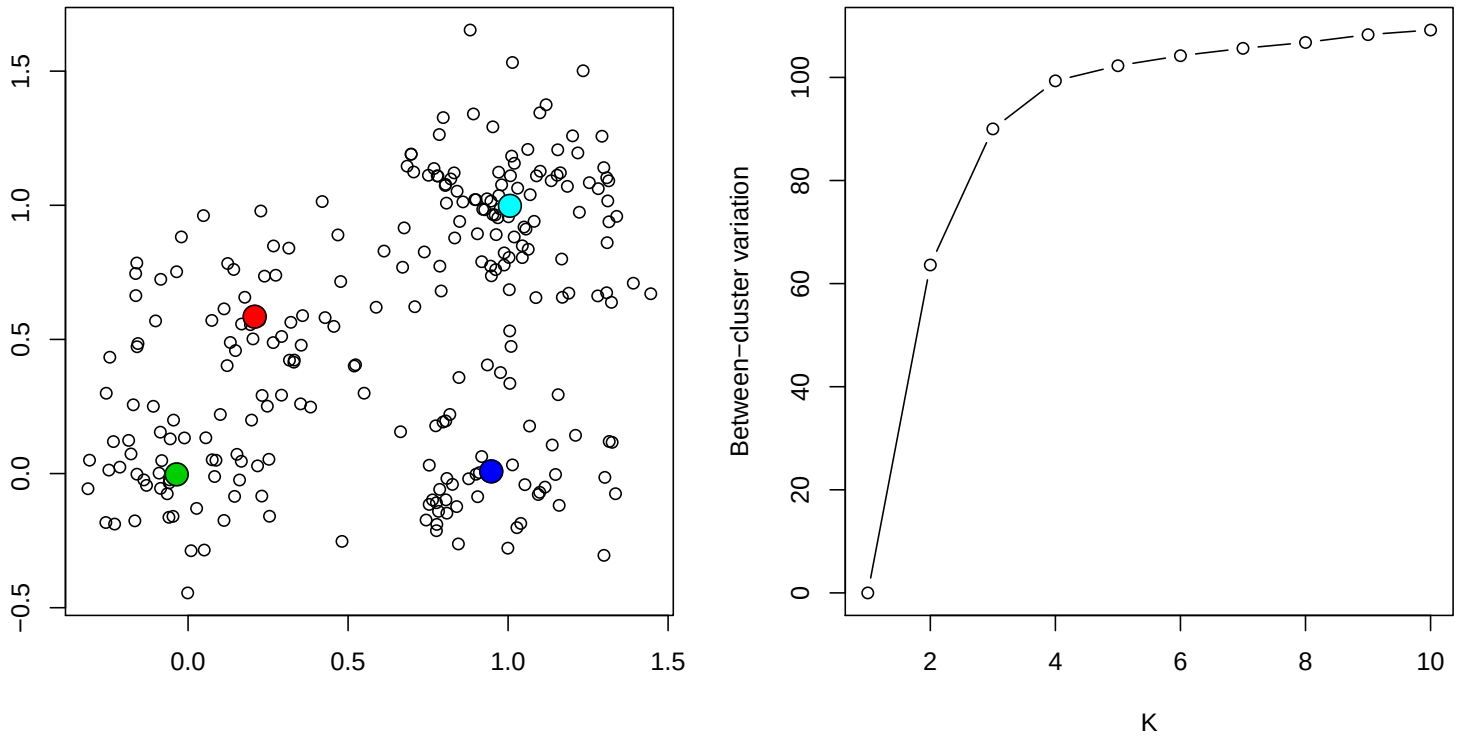
$$B = n_1 \|\bar{X}_1 - \bar{X}\|_2^2 + n_2 \|\bar{X}_2 - \bar{X}\|_2^2$$

$$W = \sum_{C(i)=1} \|X_i - \bar{X}_1\|_2^2 + \sum_{C(i)=2} \|X_i - \bar{X}_2\|_2^2$$

Still not going to work

Bigger B is better, can we use it to choose K? Problem: between-cluster variation just keeps increasing

Running example: $n = 250$, $p = 2$, $K = 1, \ldots 10$



CH index

Ideally we'd like our clustering assignments C to simultaneously have a small W and a large B

This is the idea behind the CH index.[1] For clustering assignments coming from K clusters, we record CH score:

$$\text{CH}(K) = \frac{B(K)/(K-1)}{W(K)/(n-K)}$$

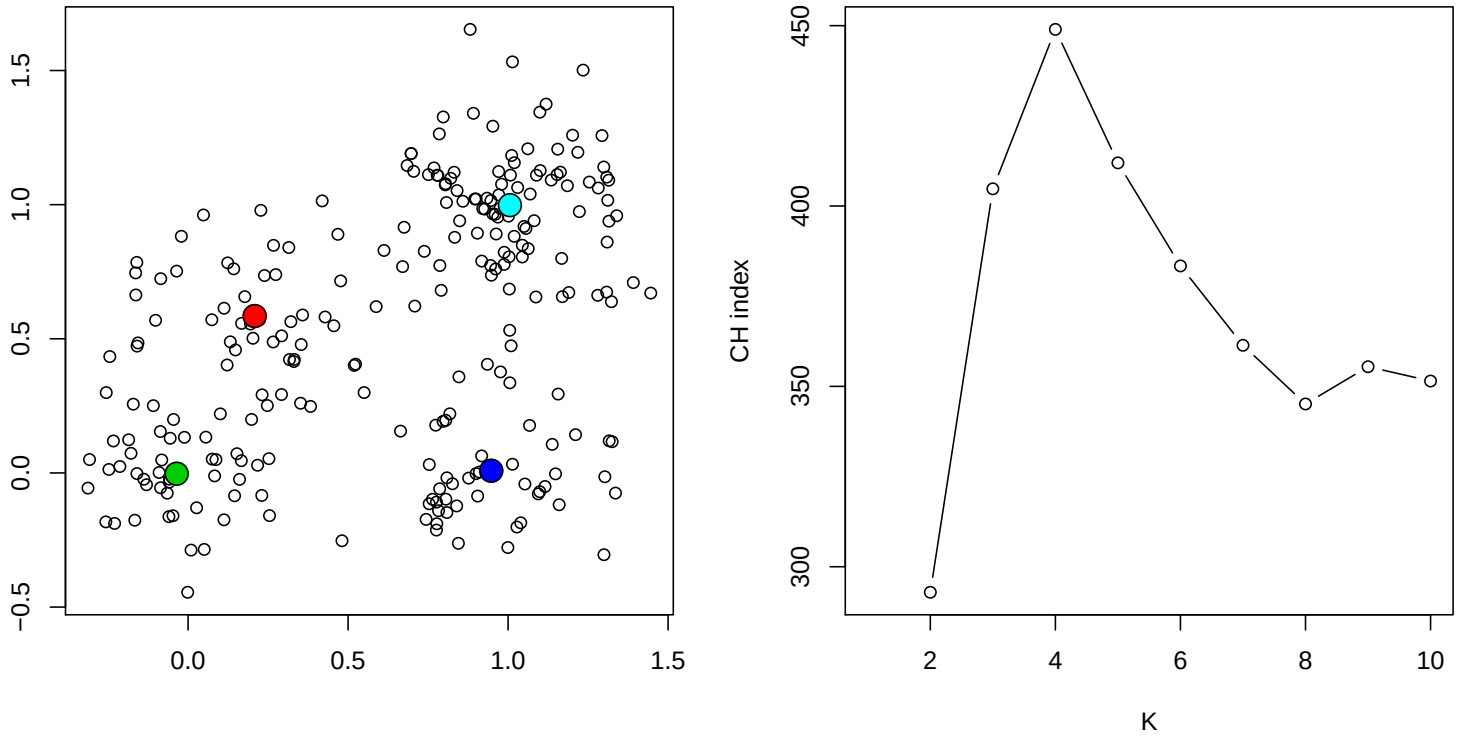
To choose K, just pick some maximum number of clusters to be considered K_{max} (e.g., $K = 20$), and choose the value of K with the largest score $\text{CH}(K)$, i.e.,

$$\hat{K} = \underset{K \in \{2,\ldots K_{\text{max}}\}}{\arg\max} \text{CH}(K)$$

[1] Calinski and Harabasz (1974), "A dendrite method for cluster analysis"

Example: CH index

Running example: $n = 250$, $p = 2$, $K = 2, \ldots 10$.



We would choose $K = 4$ clusters, which seems reasonable

General problem: the CH index is not defined for $K = 1$. We could never choose just one cluster (the null model)!

Gap statistic

It's true that $W(K)$ keeps dropping, but how much it drops at any one K should be informative

The gap statistic[2] is based on this idea.

We compare the observed within-cluster variation $W(K)$ to $W_{\text{unif}}(K)$, the within-cluster variation we'd see if we instead had points distributed uniformly (over an encapsulating box).

[2] Tibshirani et al. (2001), "Estimating the number of clusters in a data set via the gap statistic"

Gap statistic

The gap for K clusters is defined as

$$\text{Gap}(K) = \log W_{\text{unif}}(K) - \log W(K)$$

The quantity $\log W_{\text{unif}}(K)$ is computed by simulation: we average the log within-cluster variation over, say, 20 simulated uniform data sets.

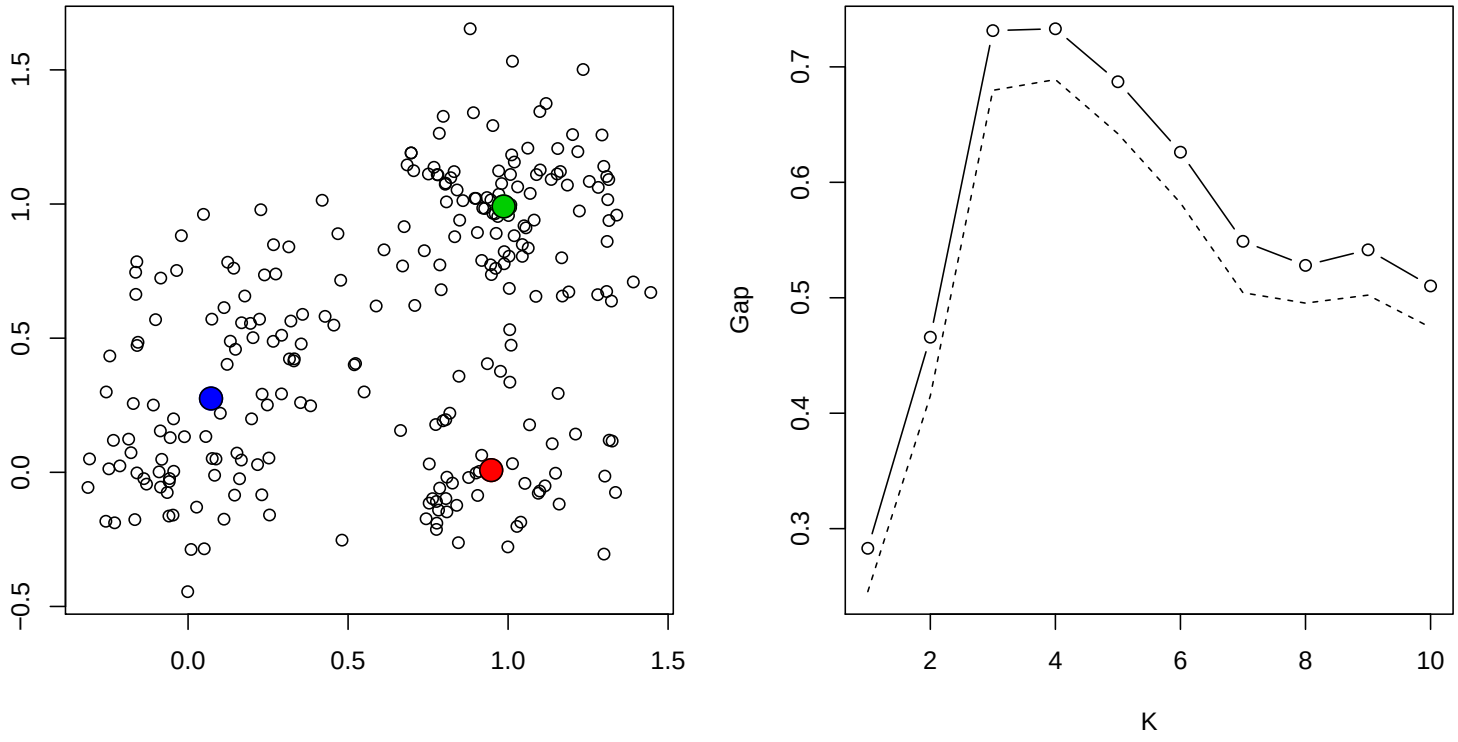
We also compute the standard error of $s(K)$ of $\log W_{\text{unif}}(K)$ over the simulations.

Then we choose K by

$$\hat{K} = \min \Big\{ K \in \{1, \ldots K_{\text{max}}\} : \text{Gap}(K) \geq \text{Gap}(K+1) - s(K+1) \Big\}$$

Example: Gap statistic

Running example: $n = 250$, $p = 2$, $K = 1, \ldots 10$



We would choose $K = 3$ clusters, which is also reasonable

The gap statistic does especially well when the data fall into one cluster. (Why? Hint: think about the null distribution that it uses)

CH index and Gap statistic in R

The CH index can be computed using the kmeans function in the base distribution, which returns both the within-cluster variation and the between-cluster varation

The gap statistic is implemented by the function gap in the package lga, and by the function gap in the package SAGx. (Beware: these functions are poorly documented and it's unclear what clustering method they're using)

Summary

Determining the number of clusters is both a hard and important problem. We can't simply try to find K that gives the smallest achieved within-class variation. We defined between-cluster variation, and saw we also can't choose K to just maximize this

Two methods for choosing K: the CH index, which looks at a ratio of between to within, and the gap statistic, which is based on the difference between within-class variation for our data and what we'd see from uniform data

Application to Tennis data set

We will partially walk through an application that you will finish on your own.

Task 1

Read in the file titled Wimbledon.csv into the variable myTennisDataD.

In your read.csv() remember to set the stringsAsFactors() parameter to FALSE.

Remove the first two columns of your dataset, paste the names together, and replace the row numberings with these match names.

Task 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

Solution to Task 1

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

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

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

# 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)})
```

Task 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.

Task 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?

Task 4

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

Task 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?

Full exercise

The full exercise can be found here:

https://github.com/resteorts/data-clean/tree/main/exercises/exericise-clustering-tennis