

Clustering

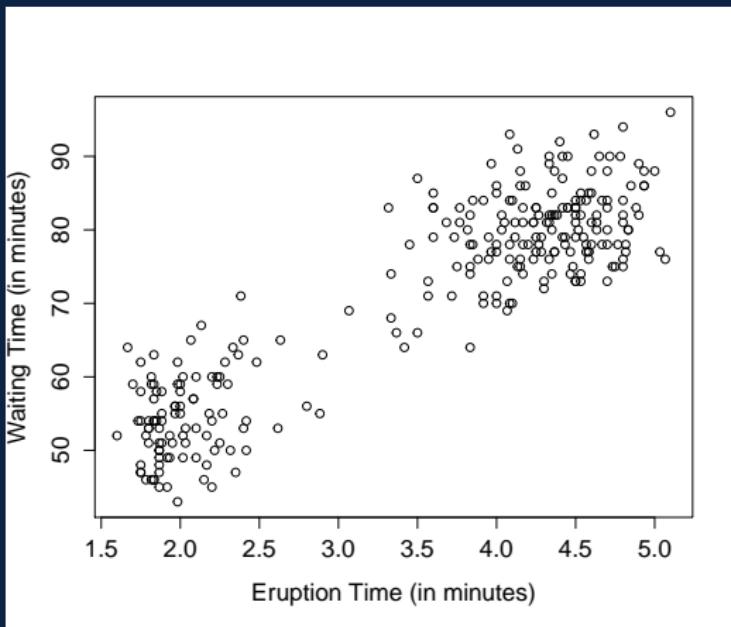
UBCO MDS — DATA 573



Clustering Motivation

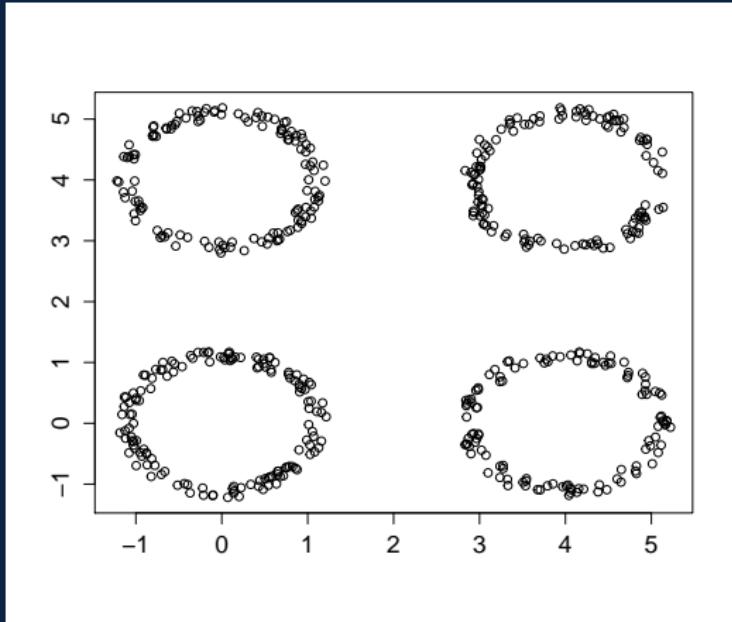
Based on clustering there seem to be 2 groups here

- ▶ Measurements on time to eruption, and length of eruption time of the Old Faithful geyser in Yellowstone National Park.



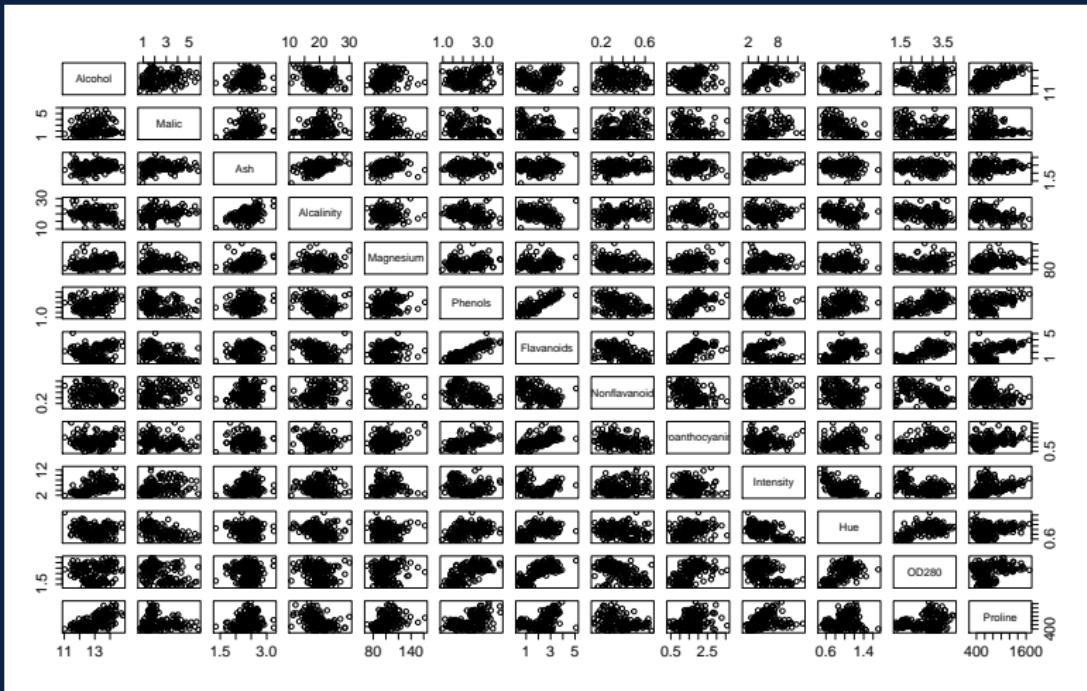
Clustering Motivation

- ▶ Simulated data



Clustering Motivation

- ▶ Scatterplot matrix of wine data (all 13 variables)



Motivation

- ▶ The goal of “clustering” (a form of unsupervised learning) is to find groups such that all observations are
 - ▶ more similar to observations inside their group and
 - ▶ more dissimilar to observations in other groups.
- ▶ Types of clustering algorithms:
 - ▶ Hierarchical
 - ▶ Partitioning
 - ▶ Mixture models

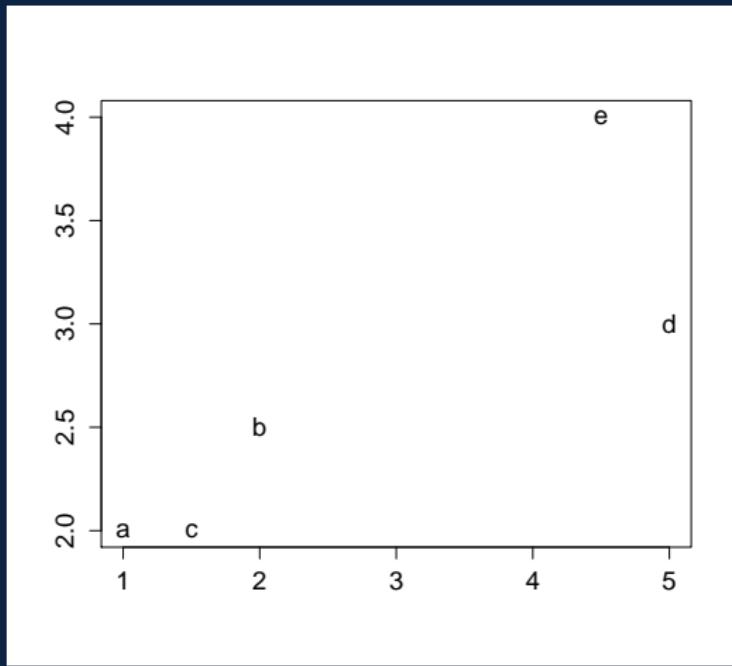


Hierarchical Clustering

- ▶ Once we have information on the distances between all observations in our data set, we're ready to come up with ways to group those observations.
- ▶ Hierarchical clustering, in many ways, is the most straightforward method for finding groups.

- ▶ Hierarchical clustering can be boiled down to the following steps:
 1. Start with all observations in their own groups (n unique groups)
 2. Join the two closest observations (now there are $n - 1$ groups)
 3. Recalculate distances (more on this soon)
 4. Repeat 2) and 3) until you are left with only 1 group.

Simple Example





Example Distance Matrix

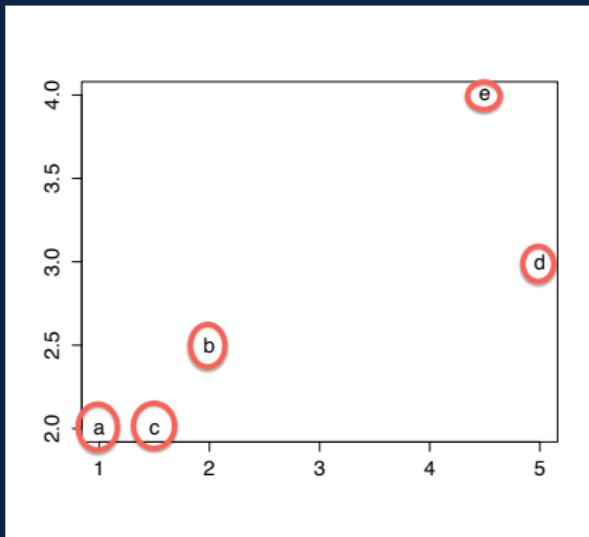
	a	b	c	d
b	1.12			
c	0.50	0.71		
d	4.12	3.04	3.64	
e	4.03	2.92	3.61	1.12

Example Distance Matrix

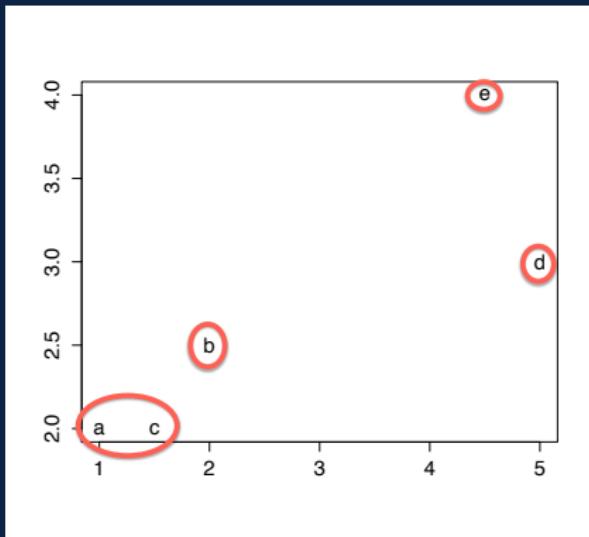
	a	b	c	d
b	1.12			
c	0.50	0.71		
d	4.12	3.04	3.64	
e	4.03	2.92	3.61	1.12

- ▶ So we join observation (a) with observation (c) at the second step of our process.

$n = 5$ Groups (our starting point)

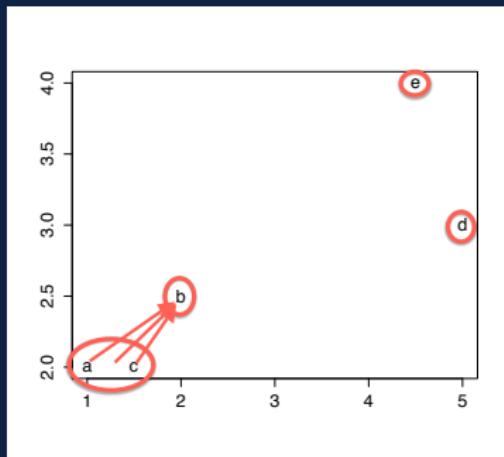


4 Groups (second step)



Recalculating Distances

- ▶ Now that we've joined the two closest observations...how do we determine the distance between that group and the others?



Recalculating Distances

- ▶ The arrows on the plot are three common ways of recalculating distances between groups, also called linkages
 - ▶ Single linkage - define distance as between closest observations.
For this example, $d_{\underline{\{ac\}}\{b\}} = d_{\underline{cb}}$
 - ▶ Complete linkage - define distance as between furthest observations.
For this example, $d_{\underline{\{ac\}}\{b\}} = d_{\underline{ab}}$.
 - ▶ Average linkage - define distance as the average distance between the observations inside group with those outside.
For this example, $d_{\underline{\{ac\}}\{b\}} = \frac{d_{ab} + d_{cb}}{2}$.

Distance Matrix - Single Linkage

	{ac}	b	d
b	0.71		
d	3.64	3.04	
e	3.61	2.92	1.12

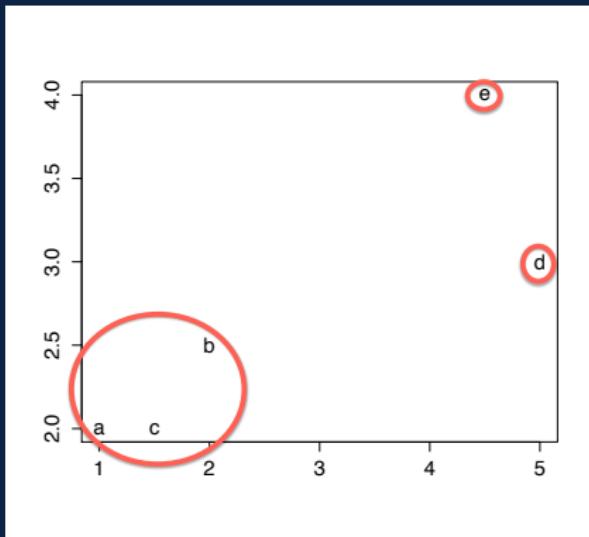
- We've updated the distances between our new group $\{ac\}$ and the other remaining groups.

Distance Matrix - Single Linkage

	{ac}	b	d
b	0.71		
d	3.64	3.04	
e	3.61	2.92	1.12

- ▶ Closest groups are {ac} and {b}. We join those now.

3 Groups

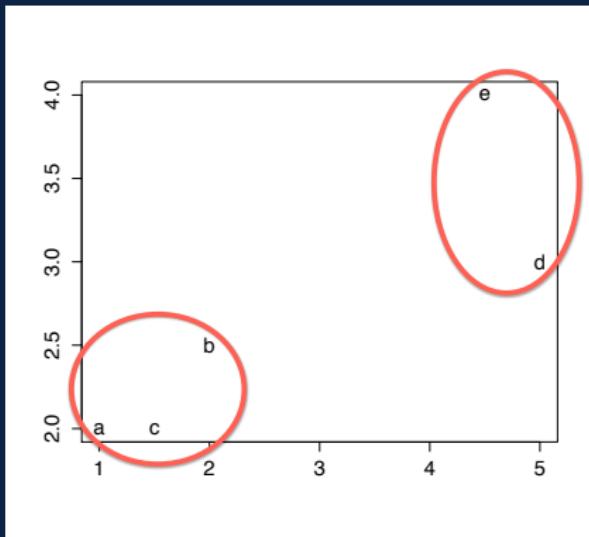


Distance Matrix - Single Linkage

	{abc}	d
d	3.04	
e	2.92	1.12

- ▶ Closest groups are $\{d\}$ and $\{e\}$. We join those now.

2 Groups



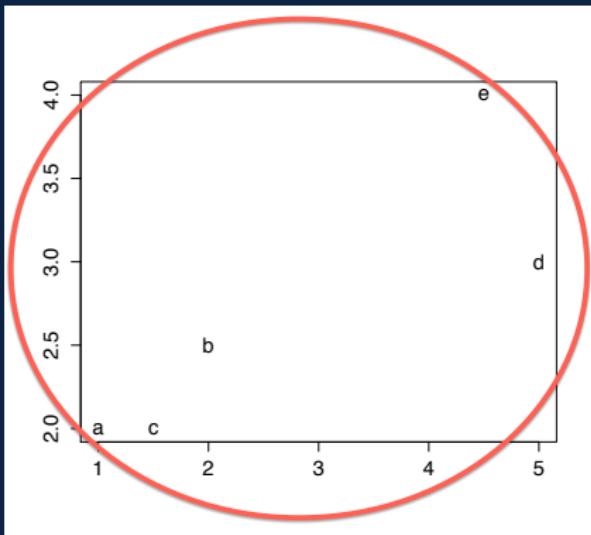
Distance Matrix - Single Linkage



{abc}	
{de}	2.92

- ▶ Closest groups are $\{abc\}$ and $\{de\}$ (only groups left to join). We join those now.

1 Group



Hierarchical Clustering

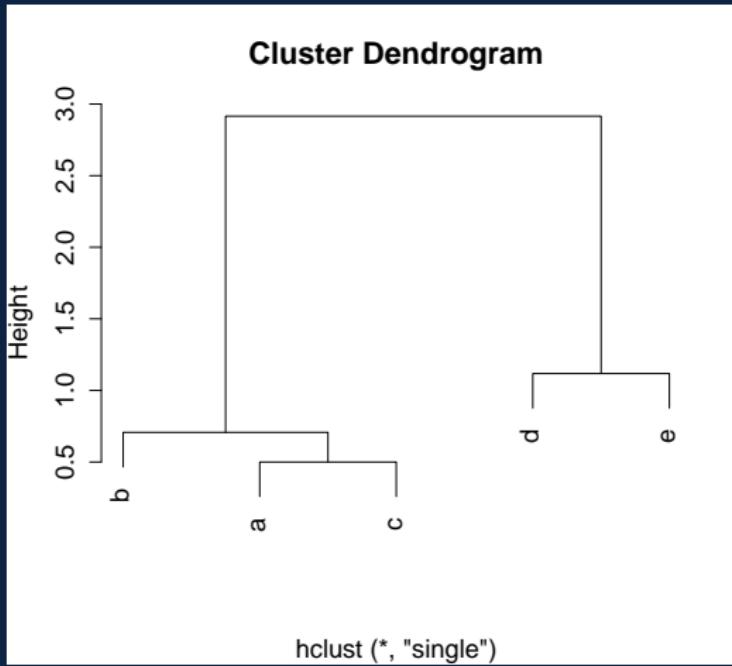
- ▶ What have we accomplished?
- ▶ Technically, we've given potential solutions to the following non-trivial questions:
 - ▶ What would 2 groups look like in this data?
 - ▶ What would 3 groups look like in this data?
 - ▶ What would 4 groups look like in this data?
- ▶ This doesn't answer the more important question: how many groups are in this data and what do they look like?
- ▶ Also, do we really need to go through the algorithm step-by-step to see the process?



Hierarchical Clustering

- ▶ The solution to both issues is one in the same.
- ▶ We can summarize the algorithm graphically, using what's called a **dendrogram**.
- ▶ On the X-axis, we just have our observations
- ▶ On the Y-axis, we have the distance where the groups are combined

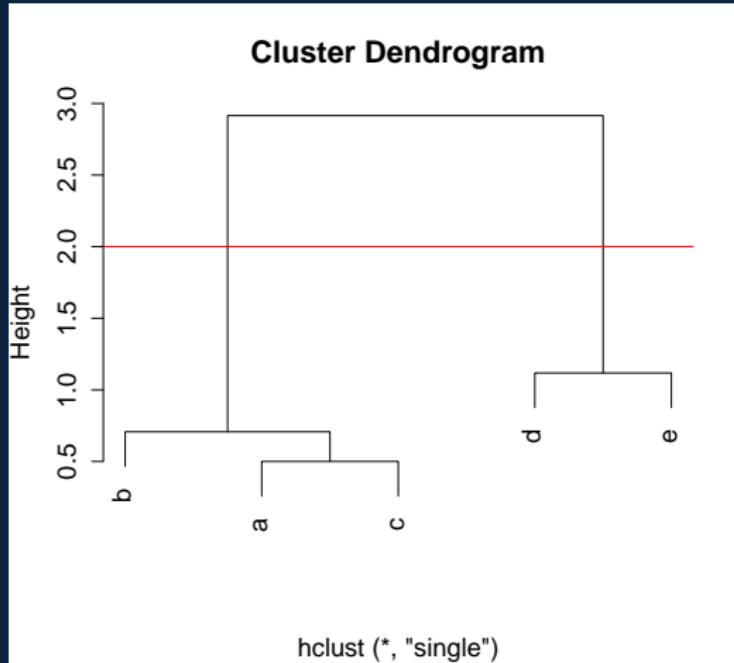
Dendrogram for our Simple Example



Hierarchical Clustering

- ▶ Dendograms show what observations were joined at what distance.
- ▶ They can also be used to determine how many groups are present in the data.
- ▶ Large “jumps” between combining groups signify large distances between groups.
- ▶ So generally, we can “cut” at the largest jump in distance, and that will determine the number of groups, as well as each observation’s group membership.

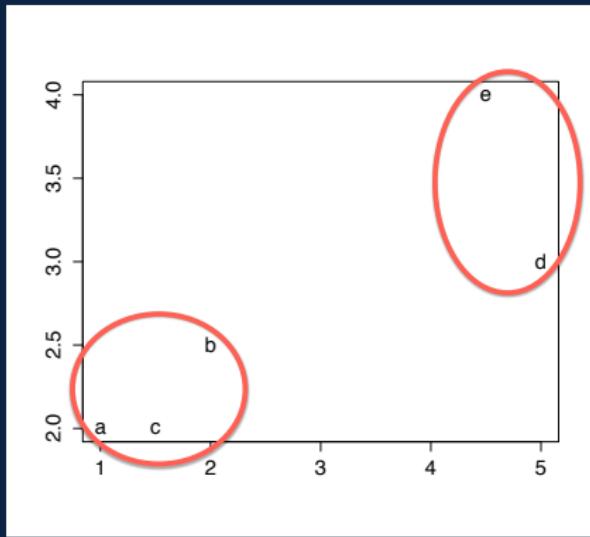
Dendrogram for our Simple Example



- ▶ This suggests a two group solution as “best” ...

2 Groups

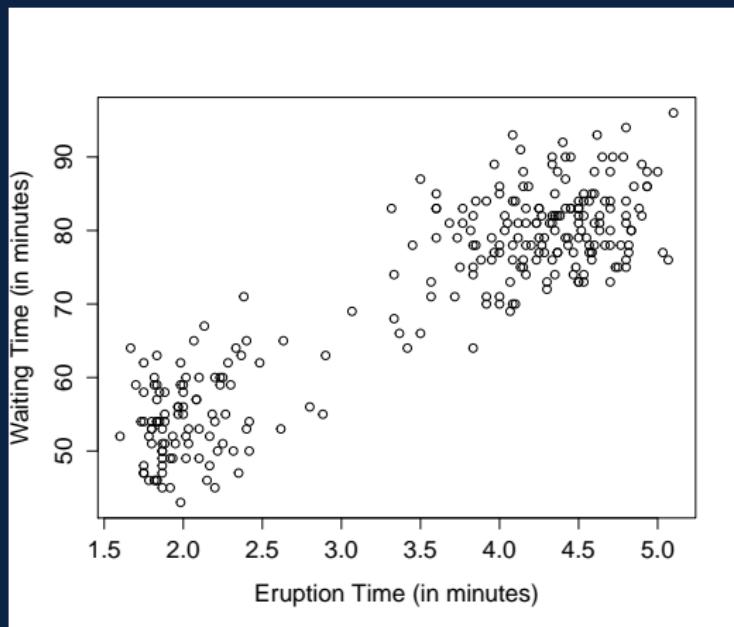
- ▶ Which should match our intuition...



Old Faithful Data Set



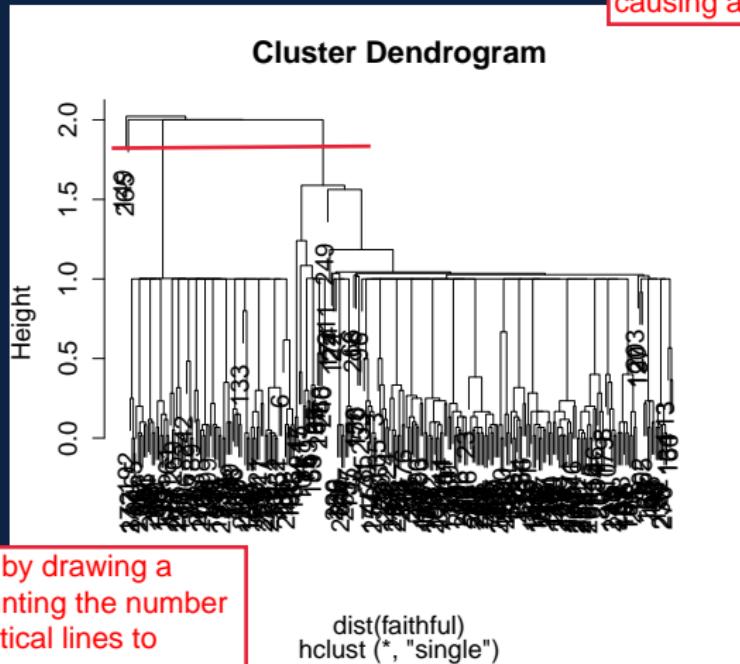
- ▶ Let's apply HC to some more interesting data sets...



Old Faithful Data Set

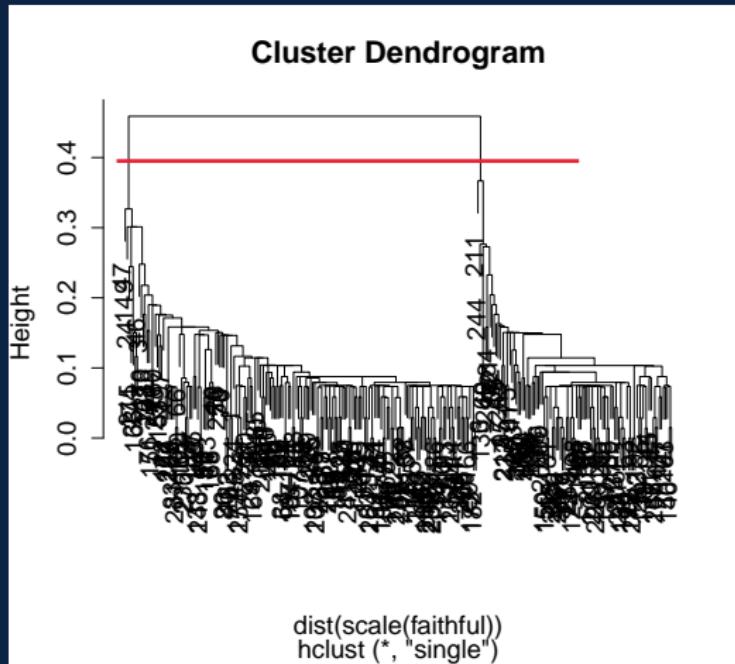
- Here's HC on the raw distance matrix...

did not standardize which is causing a problem



Old Faithful Data Set

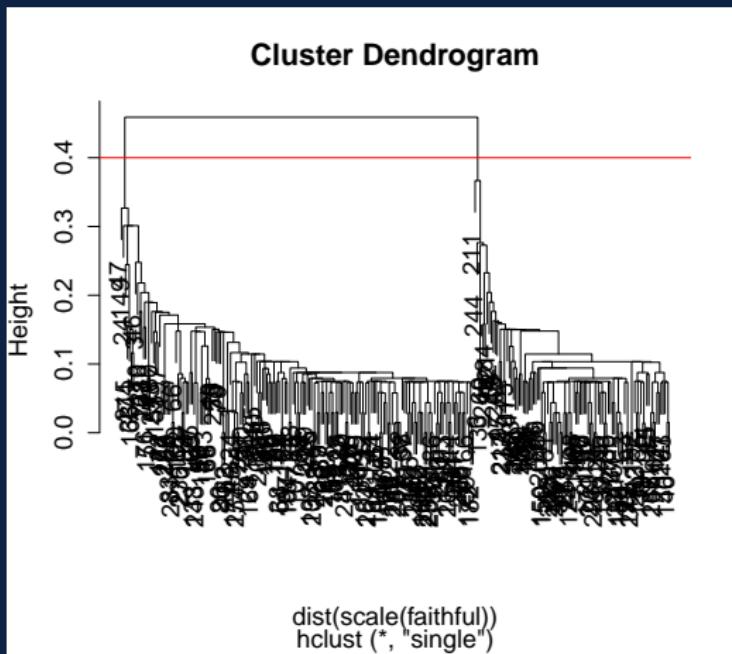
- Here's HC on the standardized distance matrix...



Old Faithful Data Set



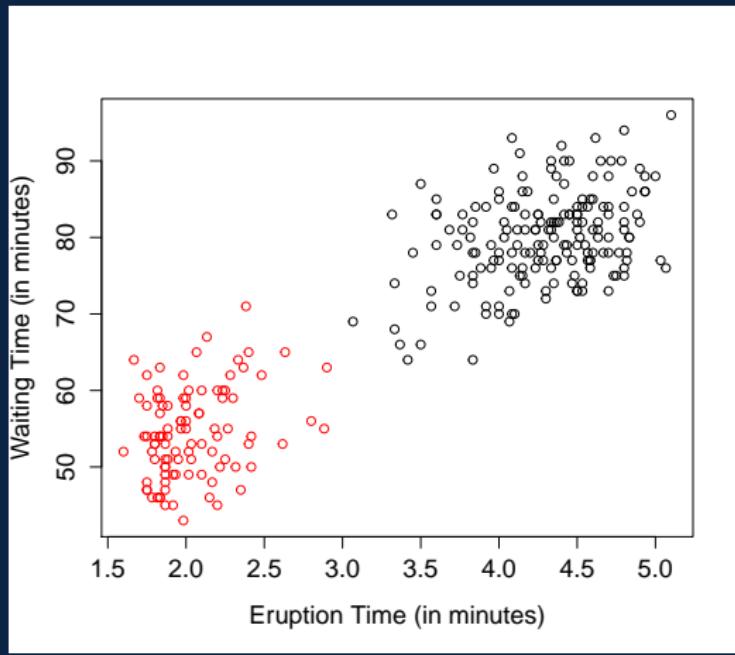
- Here's HC on the standardized distance matrix...



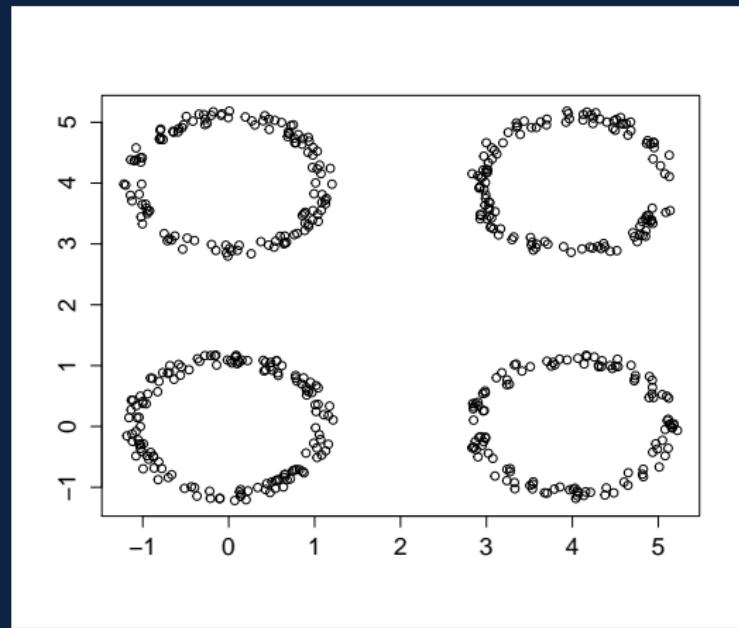
Old Faithful Data Set



- ▶ Which gives the following solution...



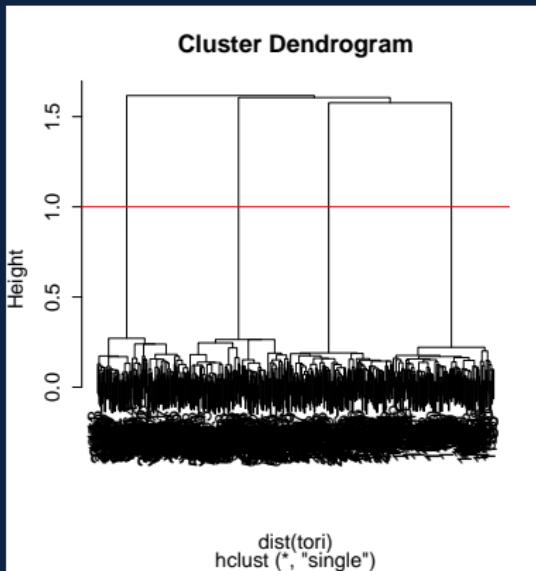
Tori! Data Set



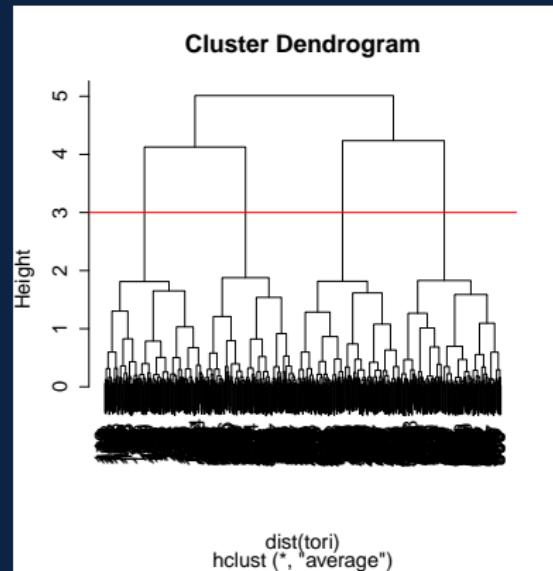
Tori Data Set

- ▶ For “easy” data sets, linkage method doesn’t affect the end result...

Single

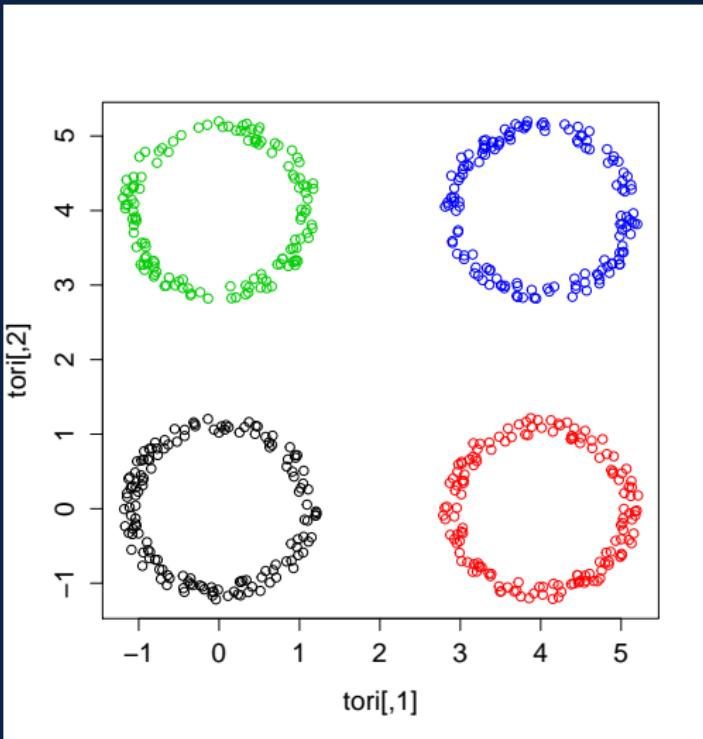


Average



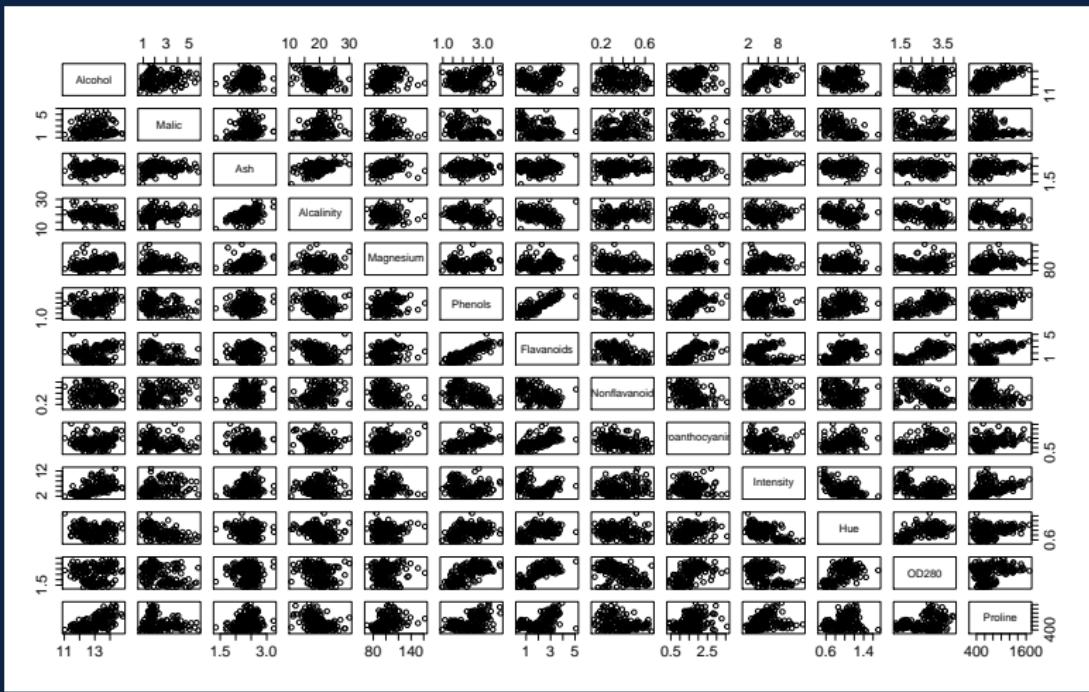
Tori Data Set

- ▶ Which, again, gives the solution we expect...



Wine Data Set

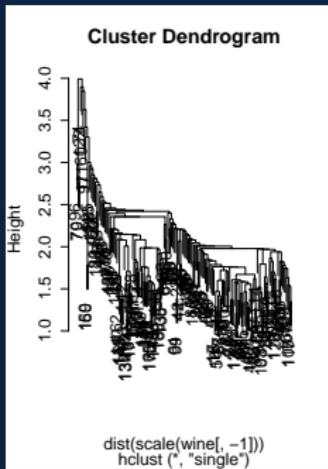
- ▶ Back to the wine!



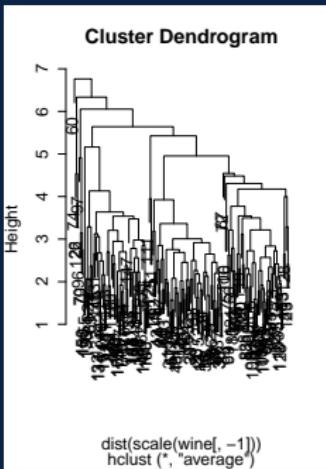
Wine Data Set

- Linkage method **will** affect more difficult clustering problems...

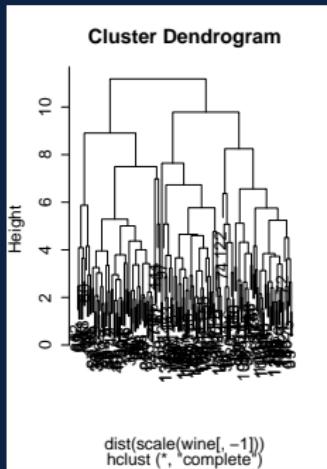
Single



Average



Complete

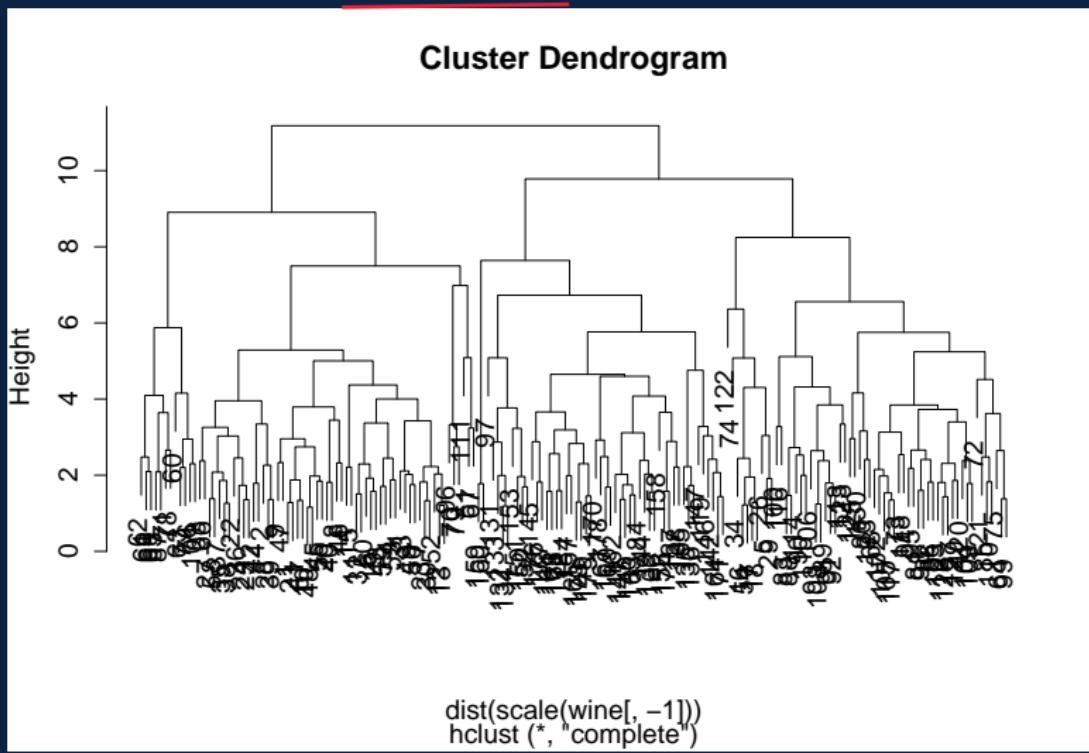


chaining pattern in single linkage occurring - linking observations into the group 1 by 1 - not behavior you want to happen while clustering

Wine Data Set



- ▶ Complete linkage gives the only clear-ish clustering result, suggesting probably 2 or 3 groups. We'll come back to this later.





Hierarchical Clustering: Problems

- ▶ Some cons of HC...
- ▶ Distance matrices (of size $n \times n$, symmetric) must be calculated. For very large samples, this can be time consuming (even for computers).
- ▶ Results are often sensitive to what distance type and what linkage method are used.

k -means Clustering



- ▶ Moving on from hierarchical methods, we'll now consider a method based on partitioning data.
- ▶ k -means clustering is a popular method that requires the user to provide the number of groups they are looking for — though we will discuss one way of seeking evidence for the number of groups.



k -means Algorithm

- ▶ The k -means algorithm has the following steps:
 1. Randomly select k (the number of groups) points in your data. These will serve as the first **centroids**.
 2. Assign all observations to their closest centroid. You now have k groups.
 3. Calculate the means of each group. These are your new centroids.
 4. Repeat 2) and 3) until nothing changes anymore.



k -means on Simulated Data

k -means on Old Faithful (2 groups)



k -means on Old Faithful (3 groups)



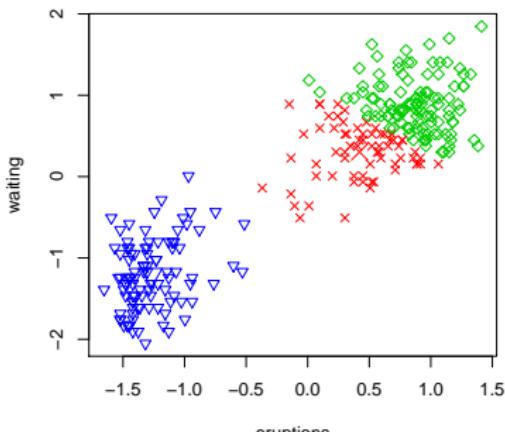
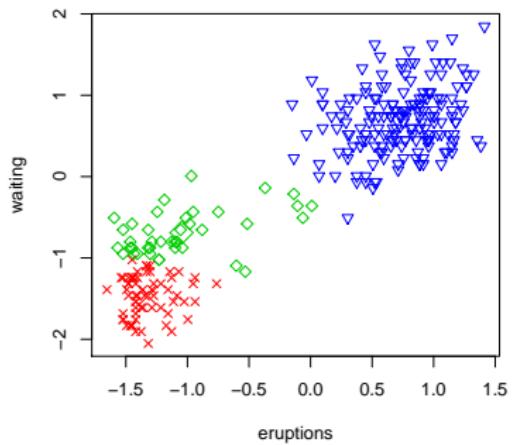
k -means on Old Faithful (3 groups again)



k -means on Old Faithful (3 group comparison)



- This should make the difference in results more clear...





k -means on Tori

k-means Algorithm

- ▶ Why does the *k*-means algorithm work?
 1. Randomly select k (the number of groups) points in your data. These will serve as the first centroids.
 2. Assign all observations to their closest centroid. You now have k groups.
 3. Calculate the means of each group. These are your new centroids.
 4. Repeat 2) and 3) until nothing changes anymore.
- ▶ Both 2) and 3) are recursively finding the minimum within-group sum of squared distances between observations and their centroids.

k -means Algorithm: Pros and Cons

► Pros

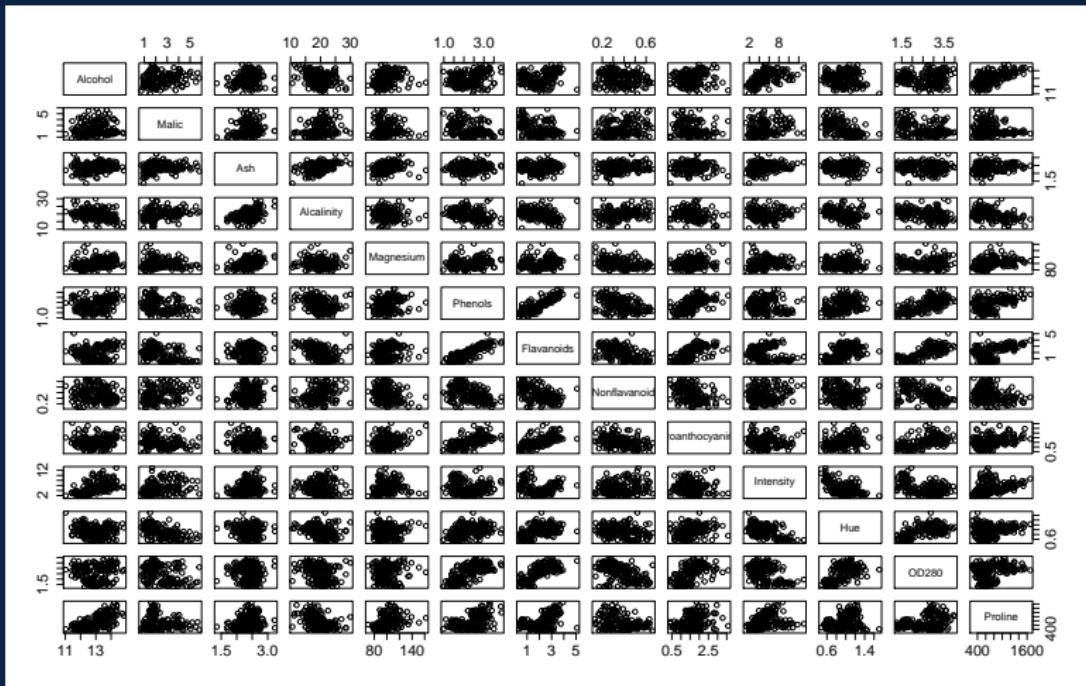
- ▶ Computationally efficient (fast to run), even for large data sets.
- ▶ Only $n \times k$ distance matrices needed (distances between n observations and k centroids).
- ▶ Relatively straightforward concept.
- ▶ Often provides clearer groups than HC.

► Cons

- ▶ As we saw, where the algorithm (randomly) starts can affect the results (local optimization rather than global)
- ▶ Groups will be found no matter what — even if there are no groups present in the data.

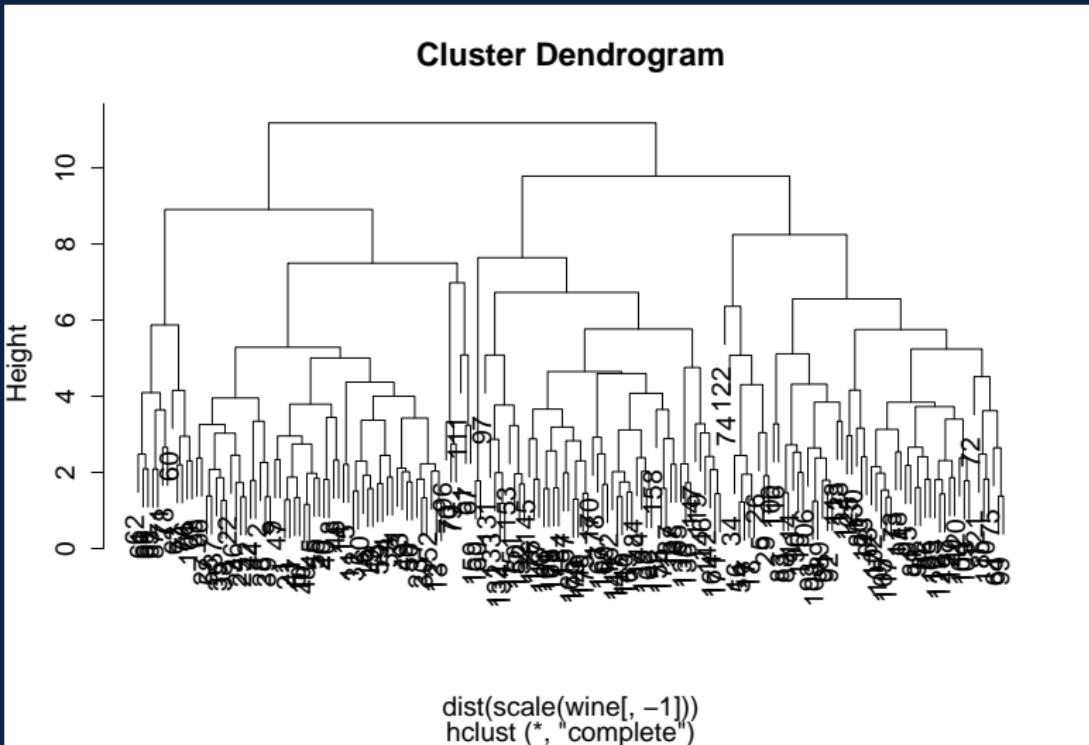
Application to Wine Data Set

- ▶ Back to the wine again. Let's apply HC and k -means and then investigate the results...



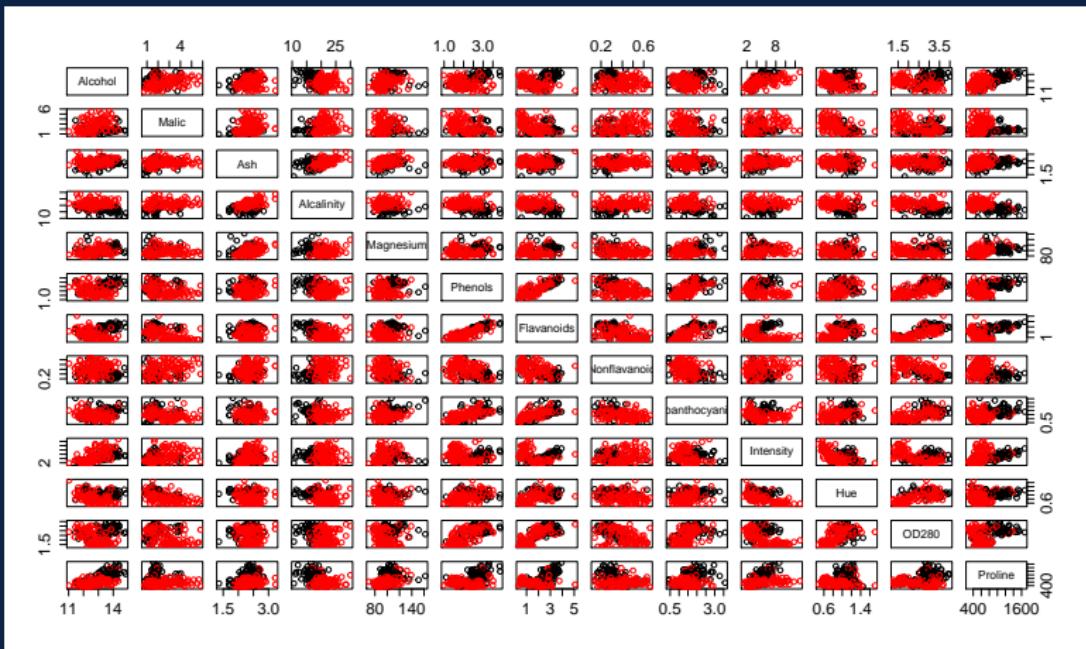
Wine Data Set

- ▶ Complete linkage suggested probably 2 or 3 groups.



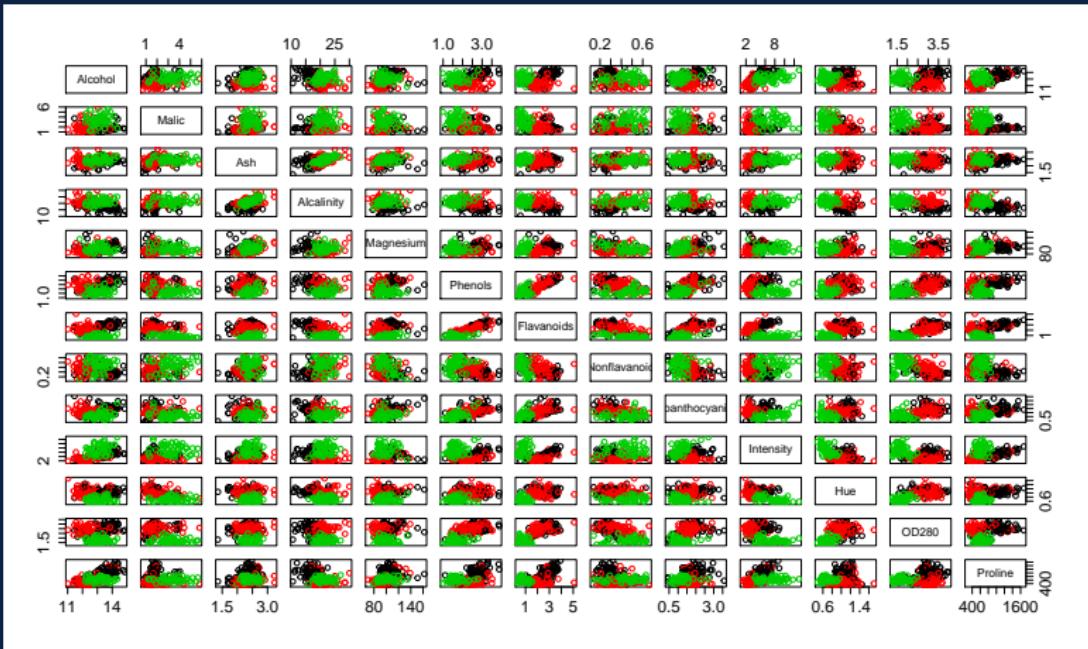
Wine Data Set

- HC 2 groups



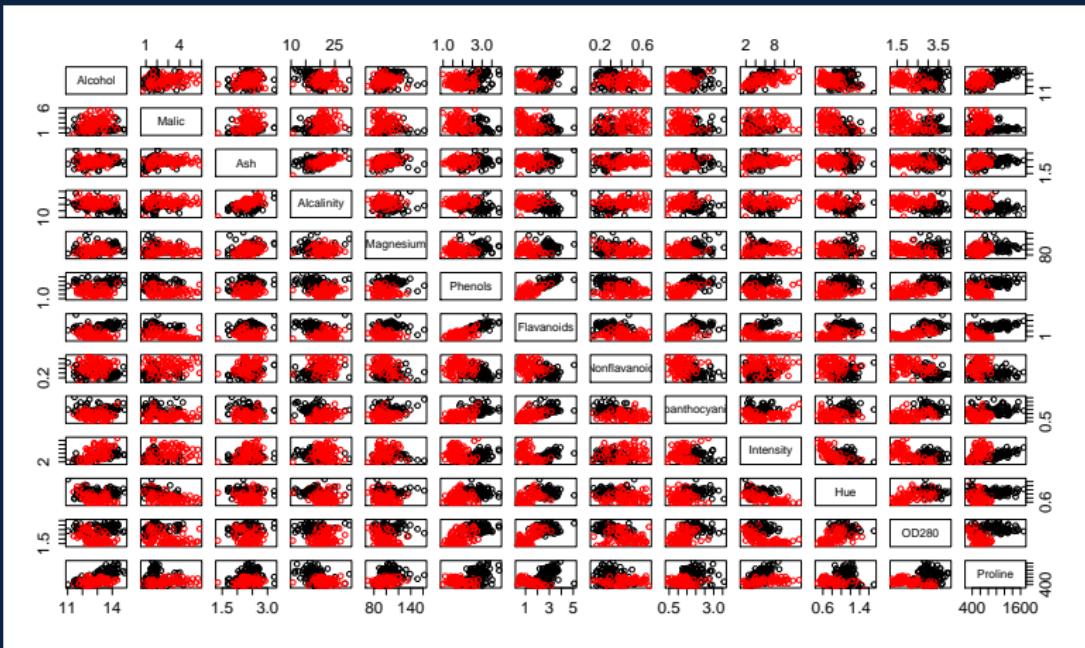
Wine Data Set

- HC 3 groups



Wine Data Set

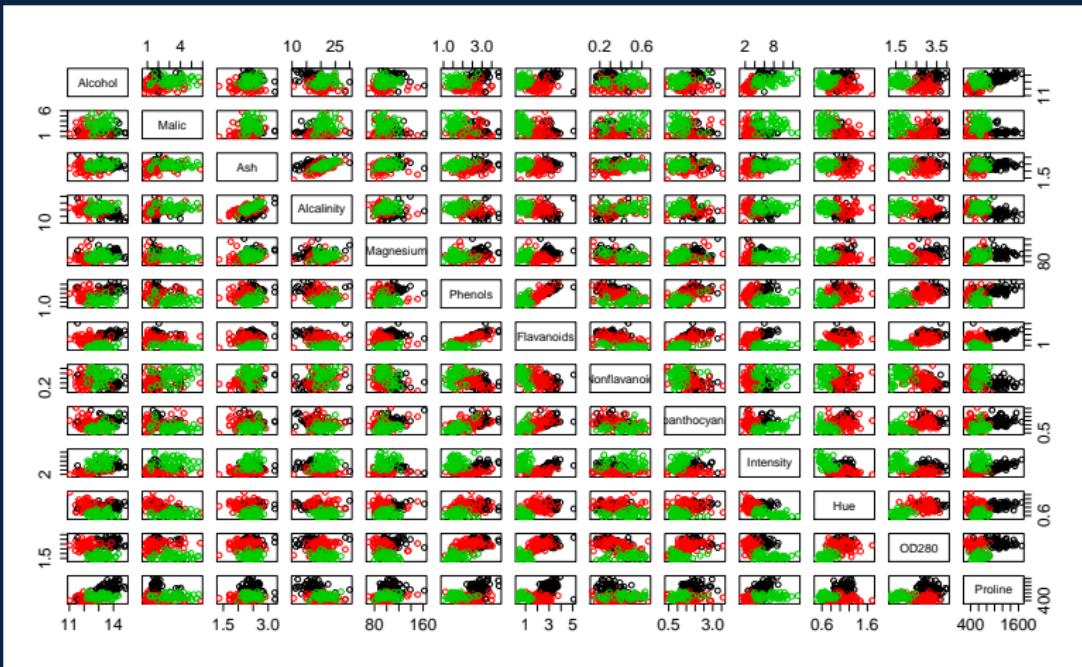
- ▶ k -means 2 groups



Wine Data Set



- ## ► k-means 3 groups





Comparing Results

- ▶ By looking at the plots, it seems as though the results are pretty similar between k -means and HC.
- ▶ The resolution makes it tough to tell, but the solutions are not identical.

Comparing Results



- ▶ Classification tables

	k-M			HC		
	1	2	3	1	2	3
Barolo	59	0	0	51	8	0
Grignolino	3	65	3	18	50	3
Barbera	0	0	48	0	0	48

- ▶ Both methods are finding most of the group structure, but...
 - ▶ *k*-means only misclassifies 6 of the 178 wines
 - ▶ HC misclassifies 29 of them



Dealing with Random Starts

- ▶ As mentioned, k -means minimizes the within group sum of squares (WSS).
- ▶ But it does so as a local minimizer, not global.
- ▶ Further, since it's initialized randomly, results change from run to run.

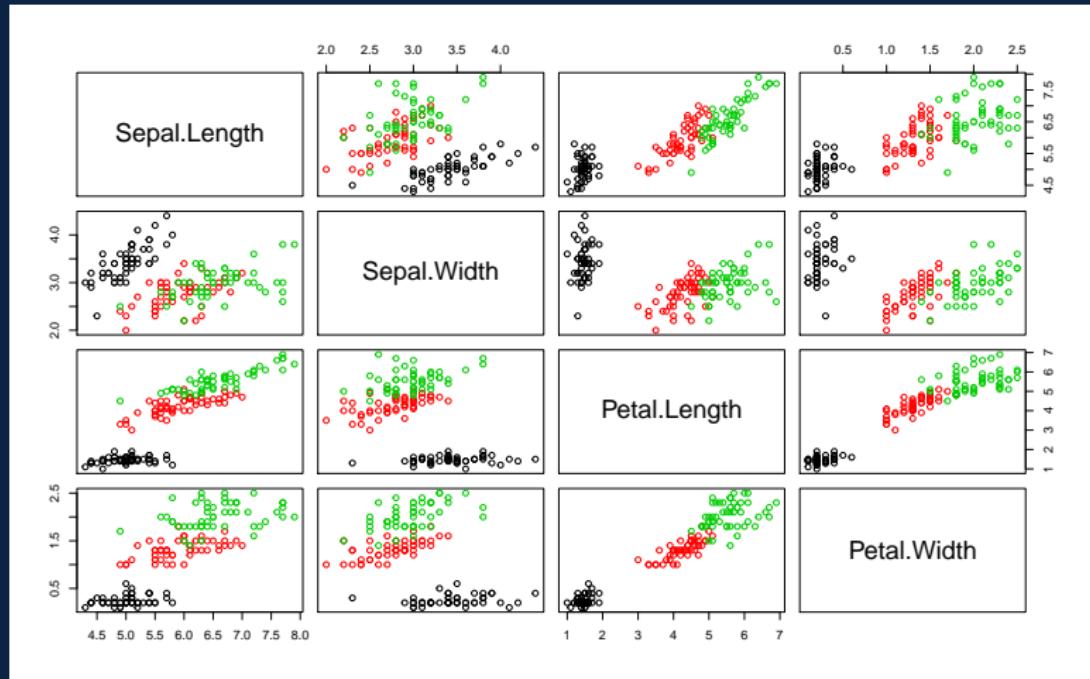
Dealing with Random Starts



- ▶ Importantly: it's fast!
- ▶ So we can run it many times, and choose the solution with the smallest within-group sum of squares for those runs.
- ▶ In fact, there's a built in argument in R...

Example

- Famous Iris data set!



Example

```
kiris <- kmeans(iris[,-5], 3)  
table(iris[,5], kiris$cluster)
```

	1	2	3
setosa	0	50	0
versicolor	48	0	2
virginica	14	0	36

be careful with labelling as it can be changed - what matters is points that were grouped together in the previous run are still grouped together in current run

```
kiris <- kmeans(iris[,-5], 3)  
table(iris[,5], kiris$cluster)
```

	1	2	3
setosa	0	50	0
versicolor	2	0	48
virginica	36	0	14

Example

```
kiris <- kmeans(iris[,-5], 3)
table(iris[,5], kiris$cluster)

      1   2   3
setosa    33 17  0
versicolor 0   4 46
virginica  0   0 50
```

```
kiris <- kmeans(iris[,-5], 3)
table(iris[,5], kiris$cluster)

      1   2   3
setosa    0   0 50
versicolor 48  2  0
virginica 14 36  0
```

Example

```
kiris <- kmeans(iris[,-5], 3, nstart=25)
```

```
table(iris[,5], kiris$cluster)
```

	1	2	3
setosa	0	50	0
versicolor	2	0	48
virginica	36	0	14

tells R to run it many times
and output lowest WSS
solution

this does not take into
account looking at stability of
solution between run - this
would have to be
investigated on our own

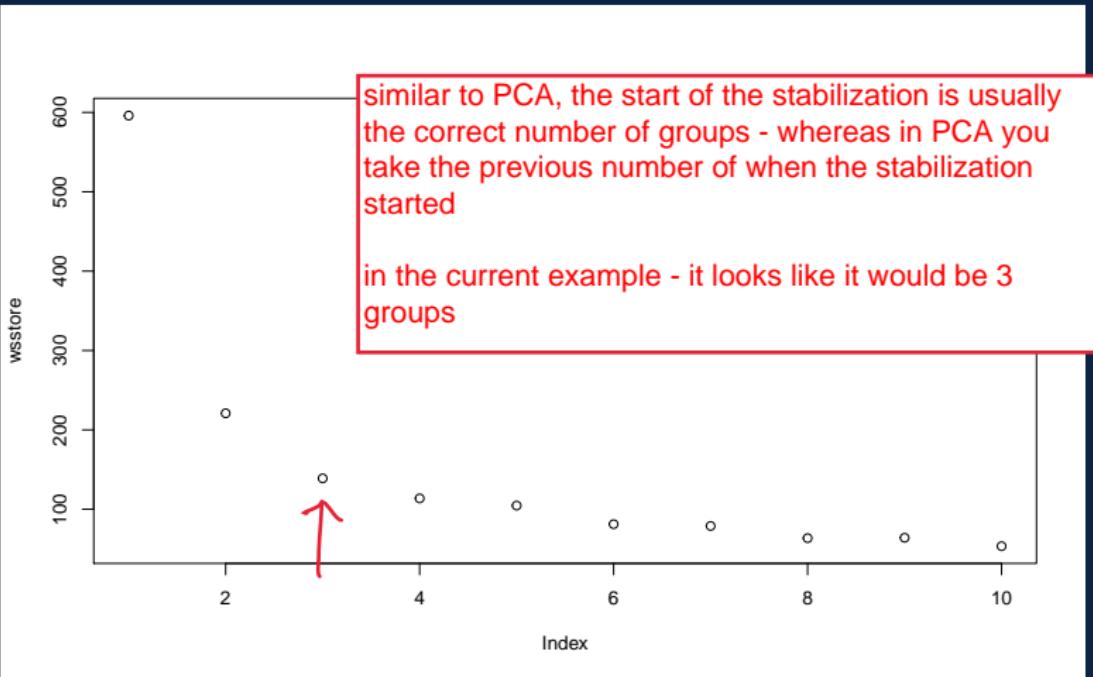
Determining Number of Groups

- ▶ For hierarchical, we have an argument for the number of groups present in the data (largest jump in dendrogram).
- ▶ For k -means, we need to specify k but can still provide some guidance
- ▶ We can plot the WSS for differing numbers of groups

Example

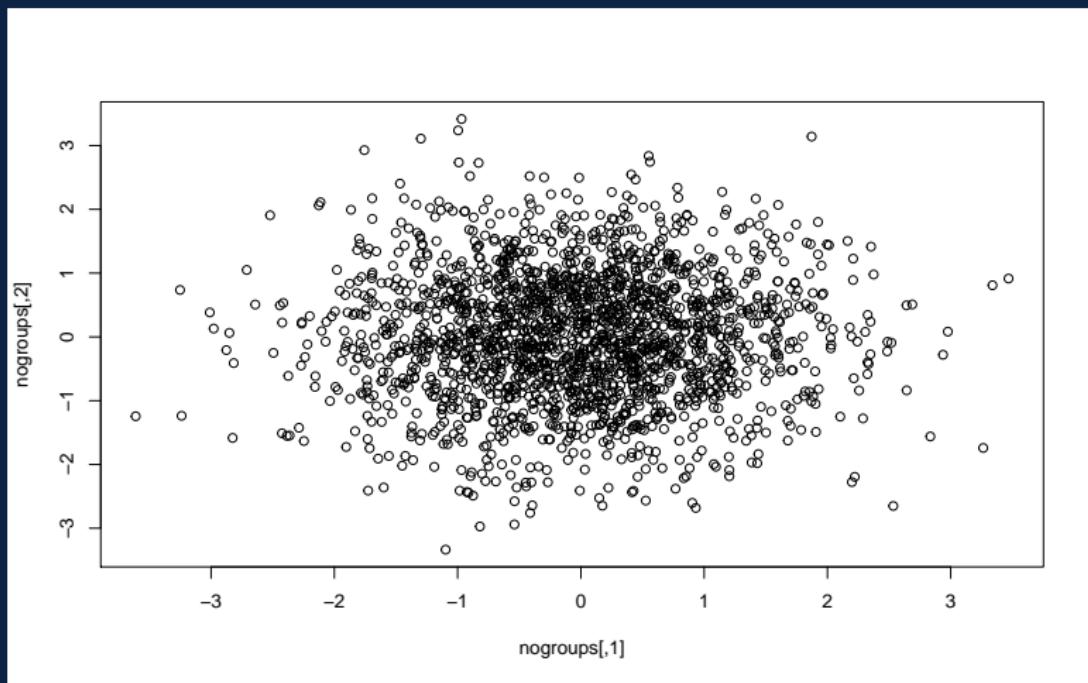
```
clustore <- matrix(0, nrow=150, ncol=10)
wsstore <- NULL
for(i in 1:10){
  dum <- kmeans(scale(iris[,-5]), i, nstart=25)
  clustore[, i] <- dum$cluster
  wsstore[i] <- dum$tot.withinss
}
plot(wsstore)
```

Example



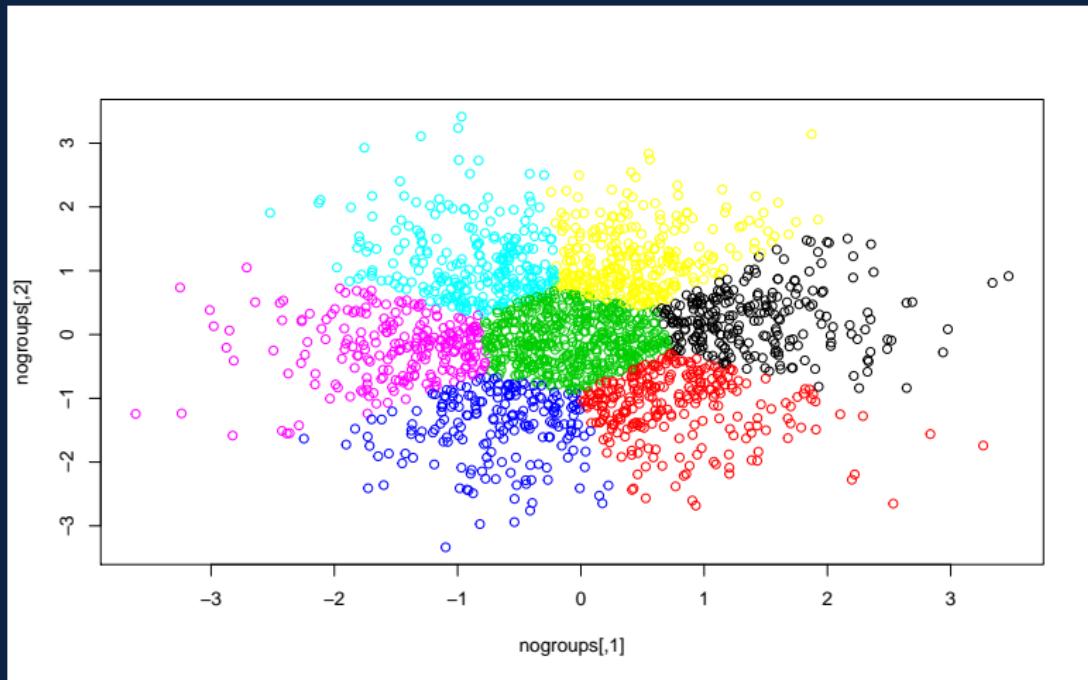
No groups?

- ▶ Also as previously noted, k-means can seem like it finds groups, even when they may not exist.



No groups?

- ▶ See...

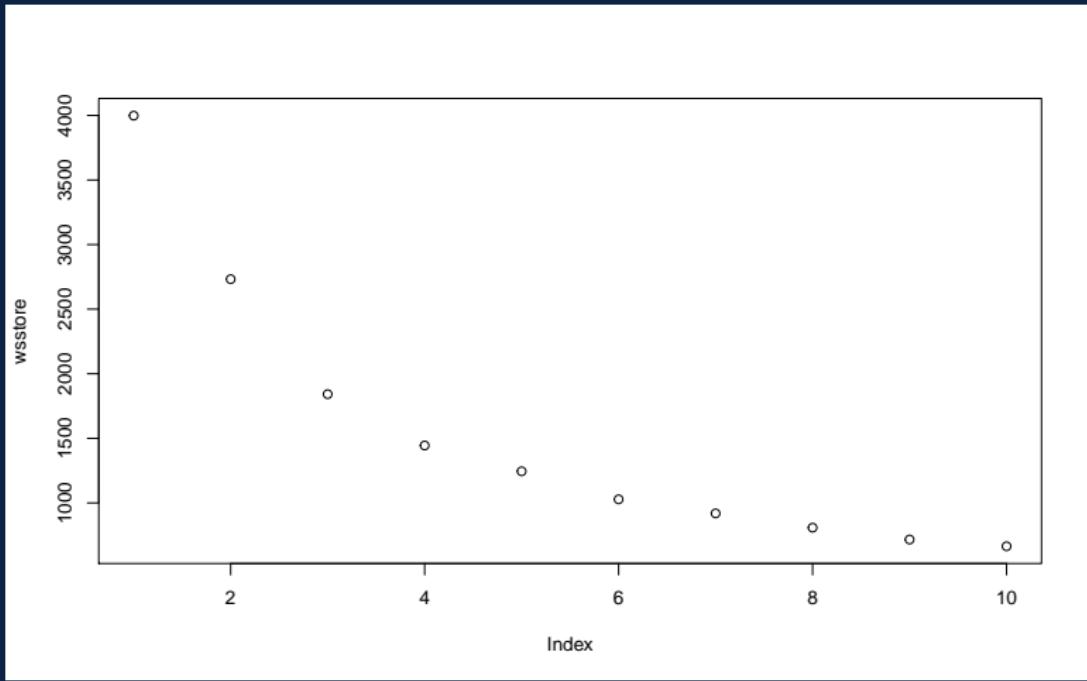




Example

```
clustore <- matrix(0, nrow=2000, ncol=10)
wsstore <- NULL
for(i in 1:10){
  dum <- kmeans(scale(nogroups), i)
  clustore[, i] <- dum$cluster
  wsstore[i] <- dum$tot.withinss
}
plot(wsstore)
```

No groups?



Partitioning and Hierarchical

- ▶ Neither HC or k -means can really be considered “statistical,” in the sense of:
 - ▶ What model are we assuming?
 - ▶ How can we test whether the model is accurate?
- ▶ They are basically *ad hoc* methods that mostly follow from intuition and tend to perform alright.
- ▶ Next, we’ll scrape the surface of clustering via mixture models —
the unsupervised equivalent of discriminant analysis.



THE UNIVERSITY OF BRITISH COLUMBIA

