SML 201 – Week 12

John D. Storey

Spring 2016

Contents

High-Dimensional Data

Definition

High-dimensional data (HD data) typically refers to data sets where *many variables* are simultaneously measured on any number of observations.

The number of variables is often represented by p and the number of observations by n.

HD data are collected into a $p \times n$ or $n \times p$ matrix.

Many methods exist for "large p, small n" data sets.

Examples

- Clinical studies
- Genomics (e.g., gene expression)
- Neuroimaging (e.g., fMRI)
- Finance (e.g., time series)
- Environmental studies
- Internet data (e.g., Netflix movie ratings)

Big Data vs HD Data

"Big data" are data sets that cannot fit into a standard computer's memory.

HD data were defined above.

They are not necessarily equivalent.

Cluster Analysis

Definition

Cluster analysis is the process of grouping objects (variables or observations) into groups based on measures of similarity.

Similar objects are placed in the same cluster, and dissimilar objects are placed in different clusters.

Cluster analysis methods are typically described by algorithms (rather than models or formulas).

Types of Clustering

Clustering can be categorized in various ways:

- Hard vs. soft
- Top-down vs bottom-up
- Partitioning vs. hierarchical agglomerative

Top-Down vs Bottom-Up

We will discuss two of the major clustering methods – *hierarchical clustering* and *K-means clustering.*

Hierarchical clustering is an example of *bottom-up* clustering in that the process begings with each object being its own cluster and then objects are joined in a hierarchical manner into larger and larger clusters.

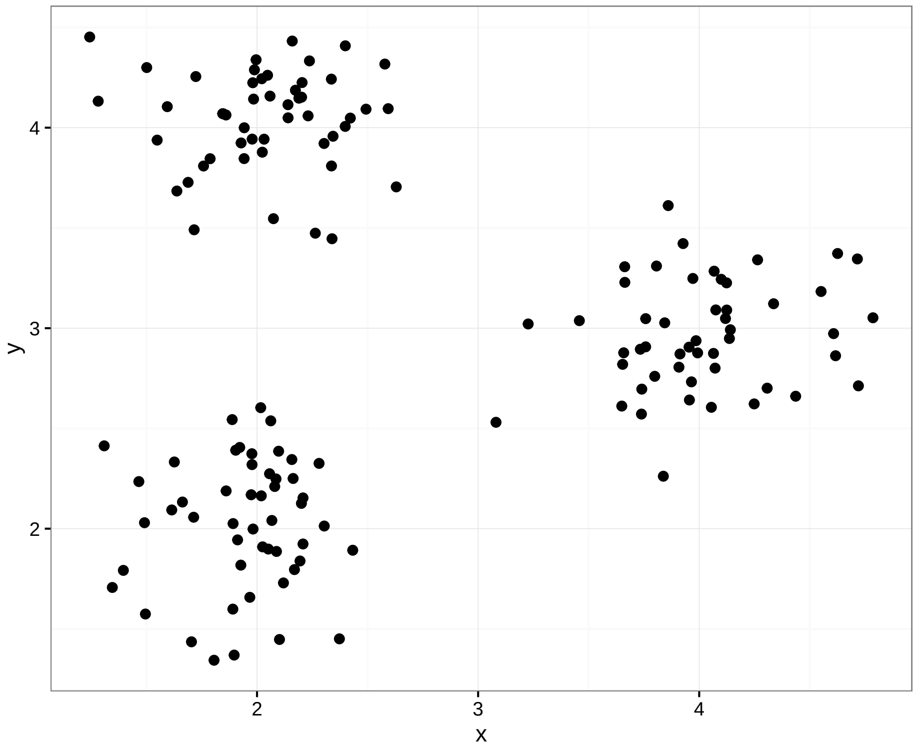
K-means clustering is an example of *top-down* clustering in that the number of clusters is chosen beforehand and then object are assigned to one of the K clusters.

Challenges

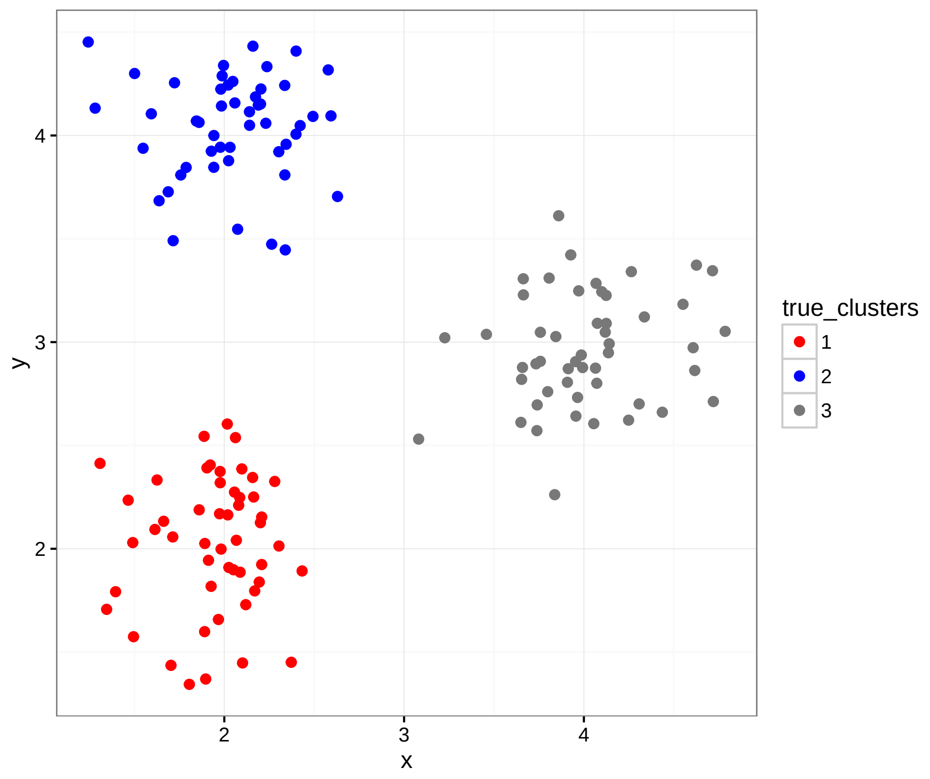
- Cluster analysis method
- Distance measure
- Number of clusters
- Convergence issues

Illustrative Data Sets

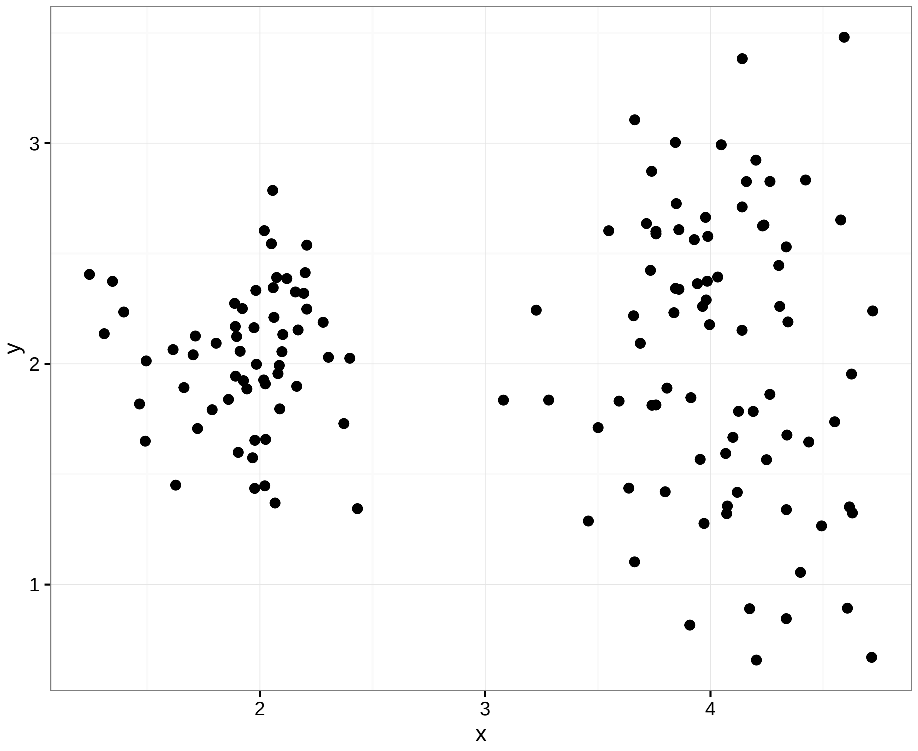
Simulated `data1`



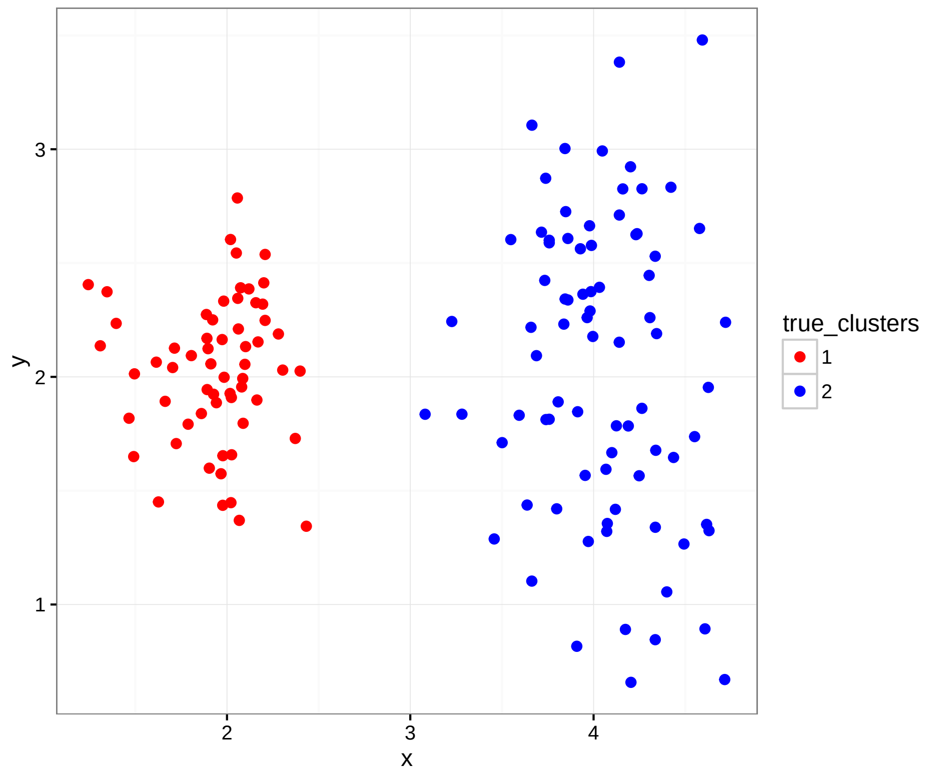
“True” Clusters `data1`



Simulated `data2`



“True” Clusters `data2`



Distance Measures

Objects

Most clustering methods require calculating a "distance" between two objects.

Let $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$ be one object and $\boldsymbol{b} = (b_1, b_2, \ldots, b_n)$ be another object.

We will assume both objects are composed of real numbers.

Euclidean

Euclidean distance is the shortest spatial distance between two objects in Euclidean space.

Euclidean distance is calculated as:

$$d(\boldsymbol{a}, \boldsymbol{b}) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}$$

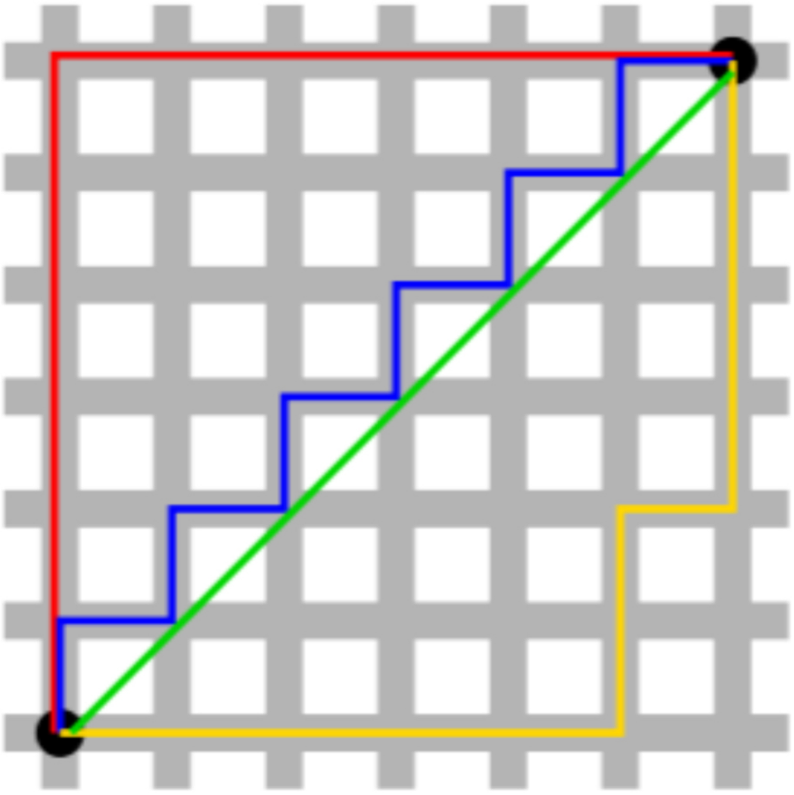
Manhattan

Manhattan distance is sometimes called taxicab distance. If you picture two locations in a city, it is the distance a taxicab must travel to get from one location to the other.

Manhattan distance is calculated as:

$$d(\boldsymbol{a}, \boldsymbol{b}) = \sum_{i=1}^{n} |a_i - b_i|$$

Euclidean vs Manhattan



Green is Euclidean. All others are Manhattan (and equal). Figure from *Exploratory Data Analysis with R*.

`dist()`

A distance matrix – which is the set of values resulting from a distance measure applied to all pairs of objects – can be obtained through the function `dist()`.

Default arguments for `dist()`:

```
> str(dist)
function (x, method = "euclidean", diag = FALSE, upper = FALSE,
    p = 2)
```

The key argument for us is `method=` which can take values `method="euclidean"` and `method="manhattan"` among others. See `?dist`.

Distance Matrix `data1`

```
> sub_data1 <- data1[1:4, c(1,2)]
> sub_data1
         x        y
1 2.085818 2.248086
2 1.896636 1.369547
3 2.097729 2.386383
4 1.491026 2.029814
> mydist <- dist(sub_data1)
> print(mydist)
          1         2         3
2 0.8986772
3 0.1388086 1.0365293
4 0.6335776 0.7749019 0.7037257
```

```
> (sub_data1[1,] - sub_data1[2,])^2 %>% sum() %>% sqrt()
[1] 0.8986772
```

Hierarchical Clustering

Strategy

Hierarchical clustering is a hierarchical agglomerative, bottom-up clustering method that strategically joins objects into larger and larger clusters, until all objects are contained in a single cluster.

Hierarchical clustering results are typically displayed as a dendrogram.

The number of clusters does not necessarily need to be known or chosen by the analyst.

Example: Cancer Subtypes

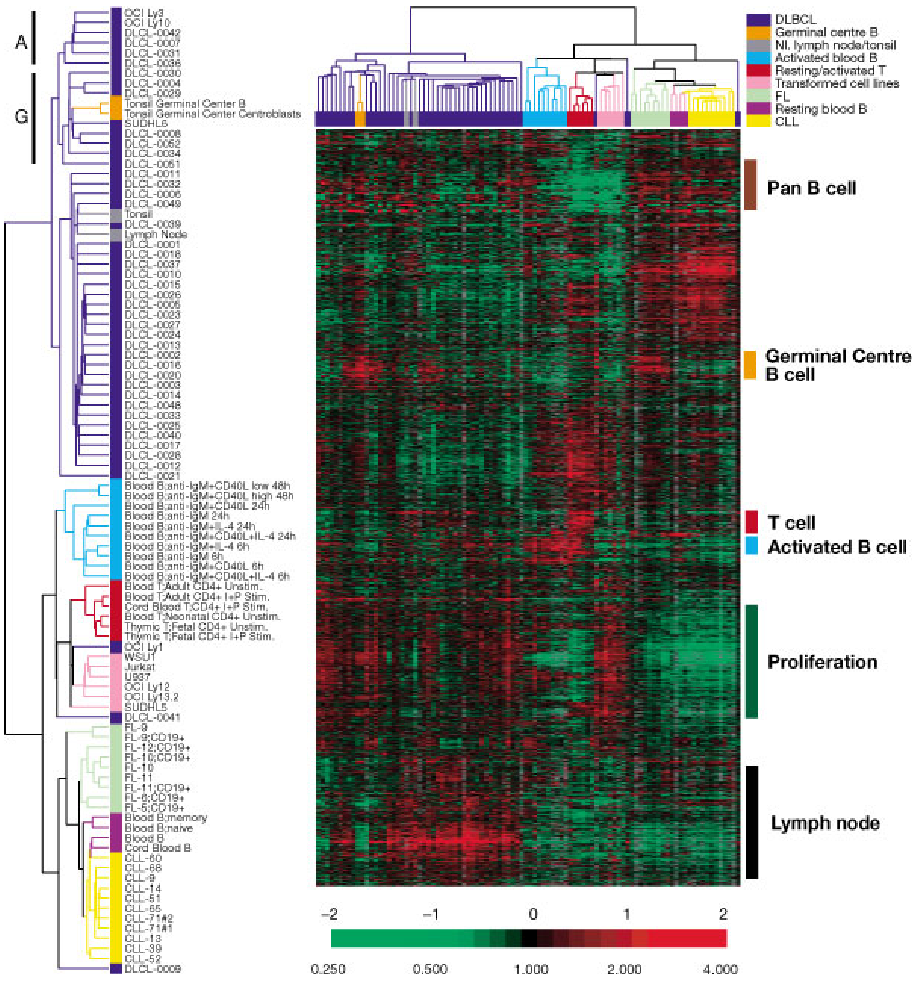


Figure from Alizadeh et al. (2000) *Nature*.

Algorithm

The algorithm for hierarchical clustering works as follows.

1. Start with each object assigned as its own cluster.
2. Calculate a distance between all pairs of clusters.
3. Join the two clusters with the smallest distance.
4. Repeat steps 2–3 until there is only one cluster.

At the very first iteration of the algorithm, all we need is some distance function (e.g., Euclidean or Manhattan) to determine the two objects that are closest.

But once clusters with more than one object are present, how do we calculate the distance between two clusters? This is where a key choice called the *linkage method or criterion* is needed.

Linkage Criteria

Suppose there are two clusters A and B and we have a distance function $d(\boldsymbol{a}, \boldsymbol{b})$ for all objects $\boldsymbol{a} \in A$ and $\boldsymbol{b} \in B$. Here are three ways (among many) to calculate a distance between clusters A and B:

$$\begin{aligned} \text{Complete:} \quad & \max\{d(\boldsymbol{a}, \boldsymbol{b}) : \boldsymbol{a} \in A, \boldsymbol{b} \in B\} && (1) \\ \text{Single:} \quad & \min\{d(\boldsymbol{a}, \boldsymbol{b}) : \boldsymbol{a} \in A, \boldsymbol{b} \in B\} && (2) \\ \text{Average:} \quad & \tfrac{1}{|A||B|} \textstyle\sum_{\boldsymbol{a} \in A} \sum_{\boldsymbol{b} \in B} d(\boldsymbol{a}, \boldsymbol{b}) && (3) \end{aligned}$$

`hclust()`

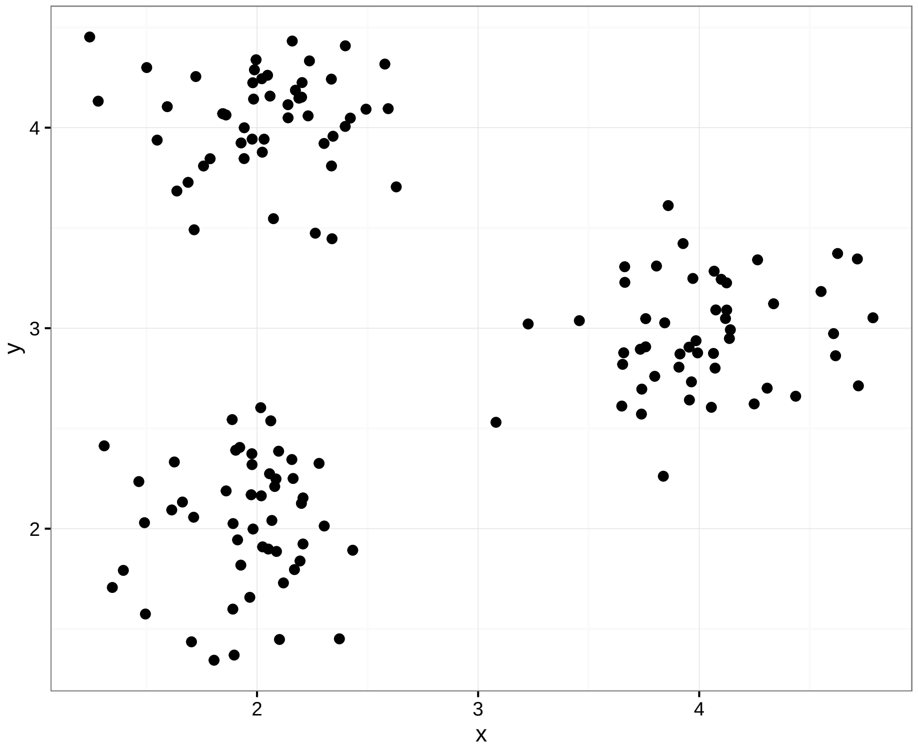
The `hclust()` function produces an R object that contains all of the information needed to create a complete hierarchical clustering.

Default arguments for `hclust()`:

```
> str(hclust)
function (d, method = "complete", members = NULL)
```

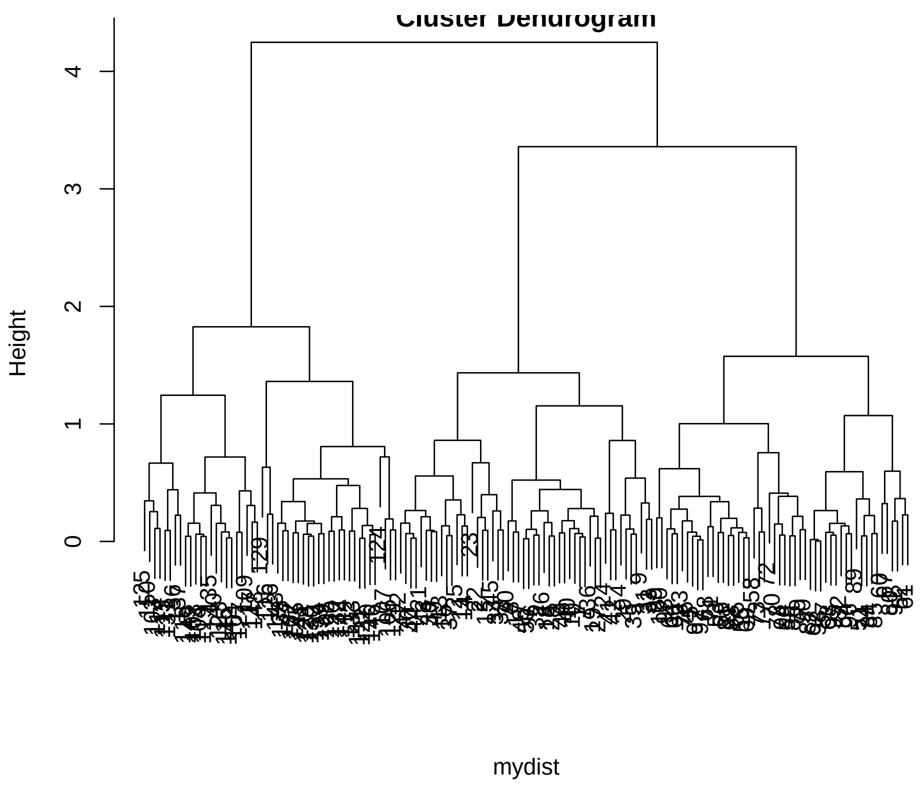
The primary input for `hclust()` is the `d` argument, which is a distance matrix (usually obtained from `dist()`). The `method` argument takes the linkage method, which includes `method="complete"`, `method="single"`, `method="average"`, etc. See `?hclust`.

Hierarchical Clustering of `data1`



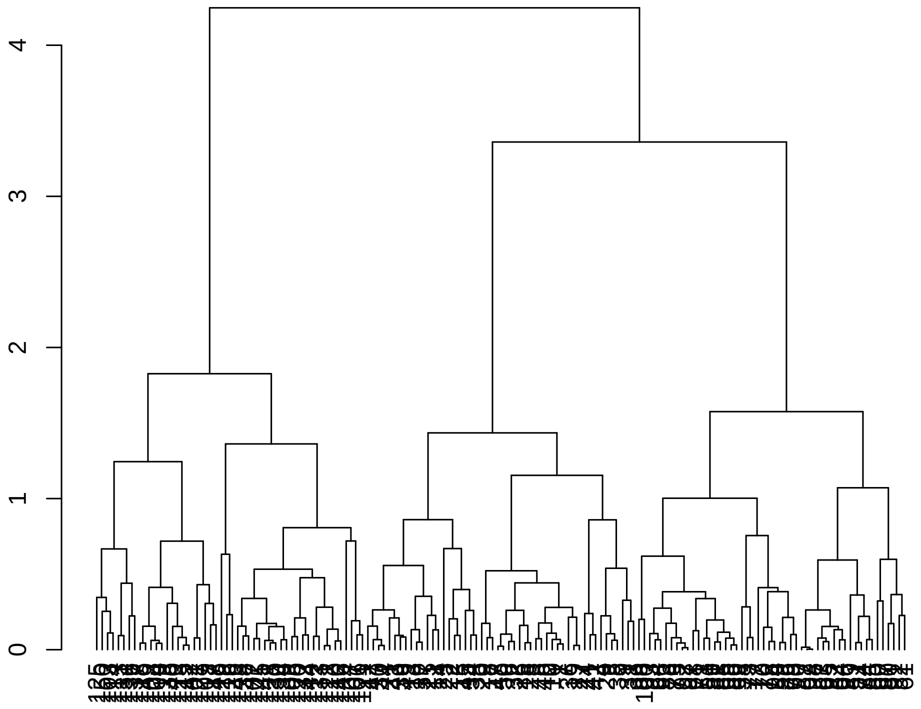
Standard `hclust()` Usage

```
> mydist <- dist(data1, method = "euclidean")
> myhclust <- hclust(mydist, method="complete")
> plot(myhclust)
```



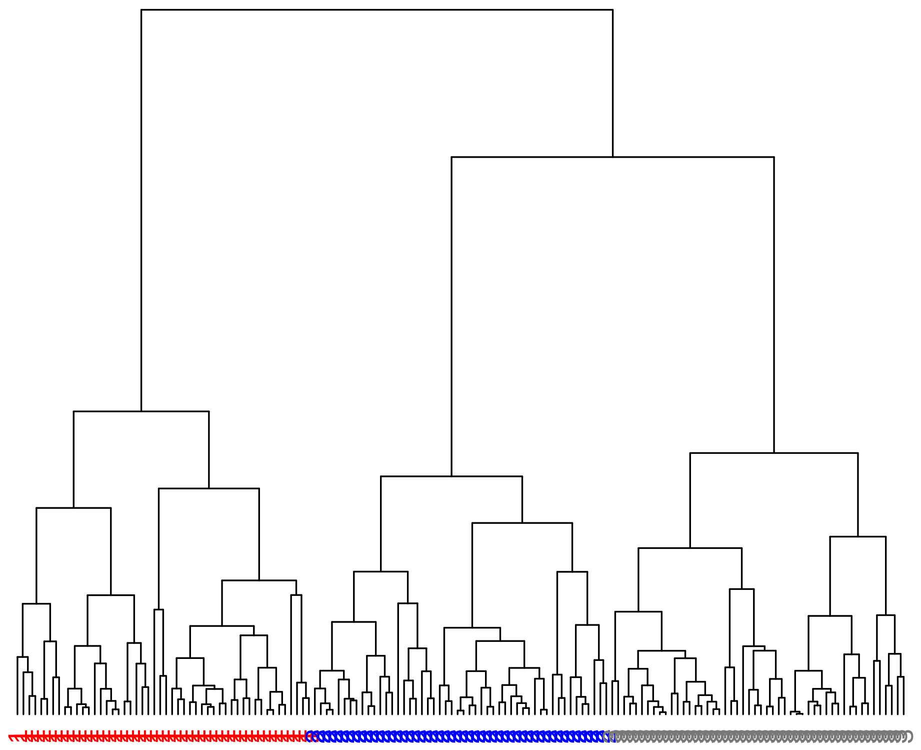
as.dendrogram()

```
> plot(as.dendrogram(myhclust))
```



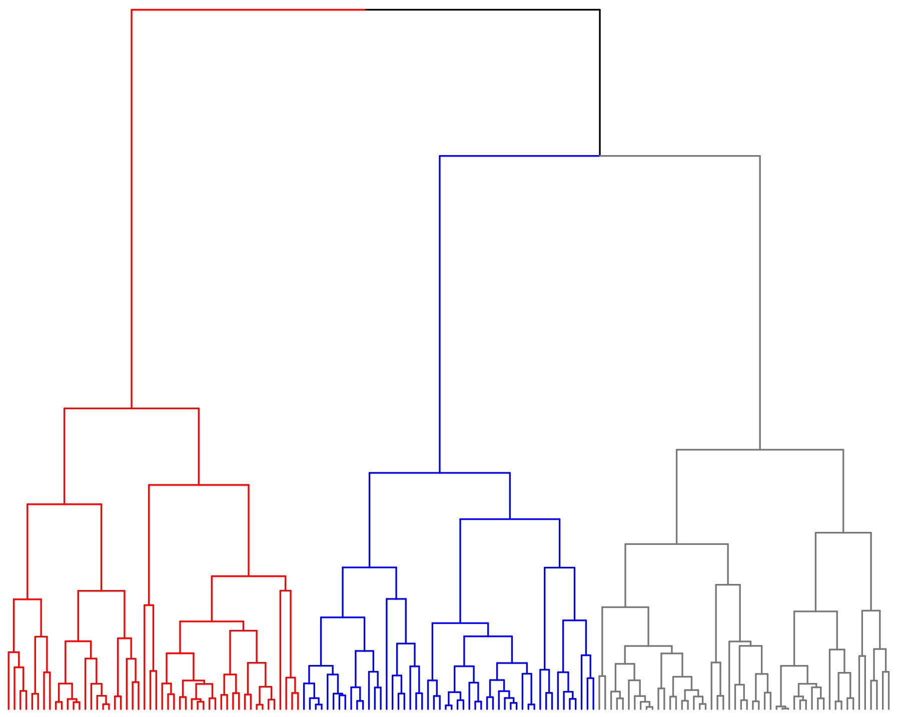
Modify the Labels

```
> library(dendextend)
> dend1 <- as.dendrogram(myhclust)
> labels(dend1) <- data1$true_clusters
> labels_colors(dend1) <-
+   c("red", "blue", "gray47")[as.numeric(data1$true_clusters)]
> plot(dend1, axes=FALSE, main=" ", xlab=" ")
```



Color the Branches

```
> dend2 <- as.dendrogram(myhclust)
> labels(dend2) <- rep(" ", nrow(data1))
> dend2 <- color_branches(dend2, k = 3, col=c("red", "blue", "gray47"))
> plot(dend2, axes=FALSE, main=" ", xlab=" ")
```

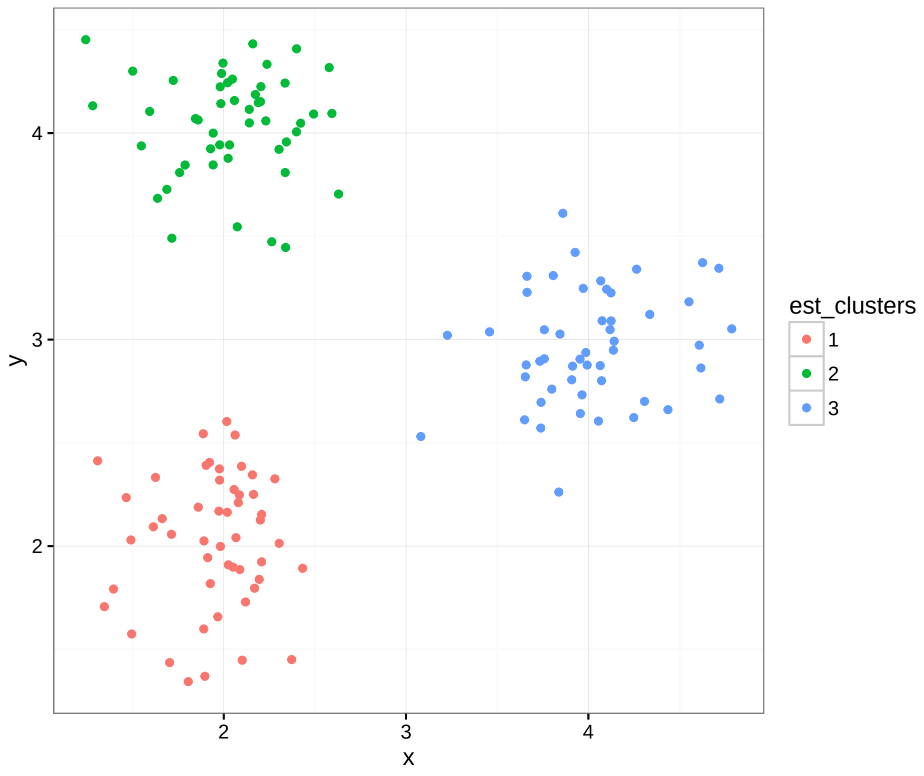


Cluster Assignments $(K = 3)$

```
> est_clusters <- cutree(myhclust, k=3)
> est_clusters
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [30] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
 [59] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [88] 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[117] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[146] 3 3 3 3 3
```

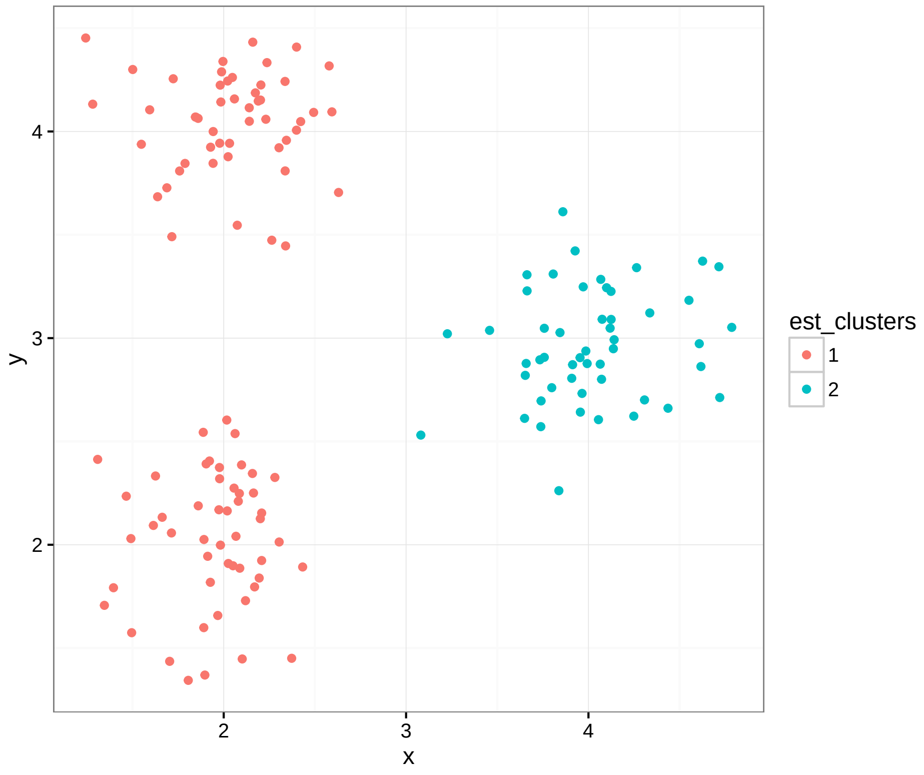
```
> est_clusters <- factor(est_clusters)
> p <- data1 %>%
+   mutate(est_clusters=est_clusters) %>%
+   ggplot()
> p + geom_point(aes(x=x, y=y, color=est_clusters))
```

Cluster Assignments ($K = 3$)



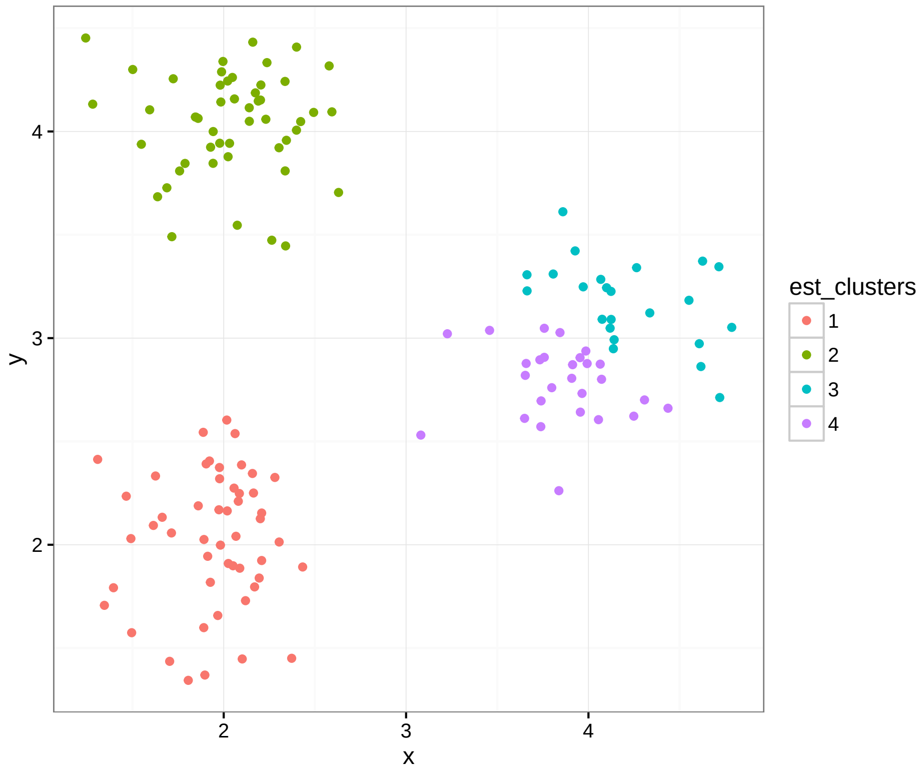
Cluster Assignments ($K = 2$)

```
> (data1 %>%
+     mutate(est_clusters=factor(cutree(myhclust, k=2))) %>%
+     ggplot()) + geom_point(aes(x=x, y=y, color=est_clusters))
```



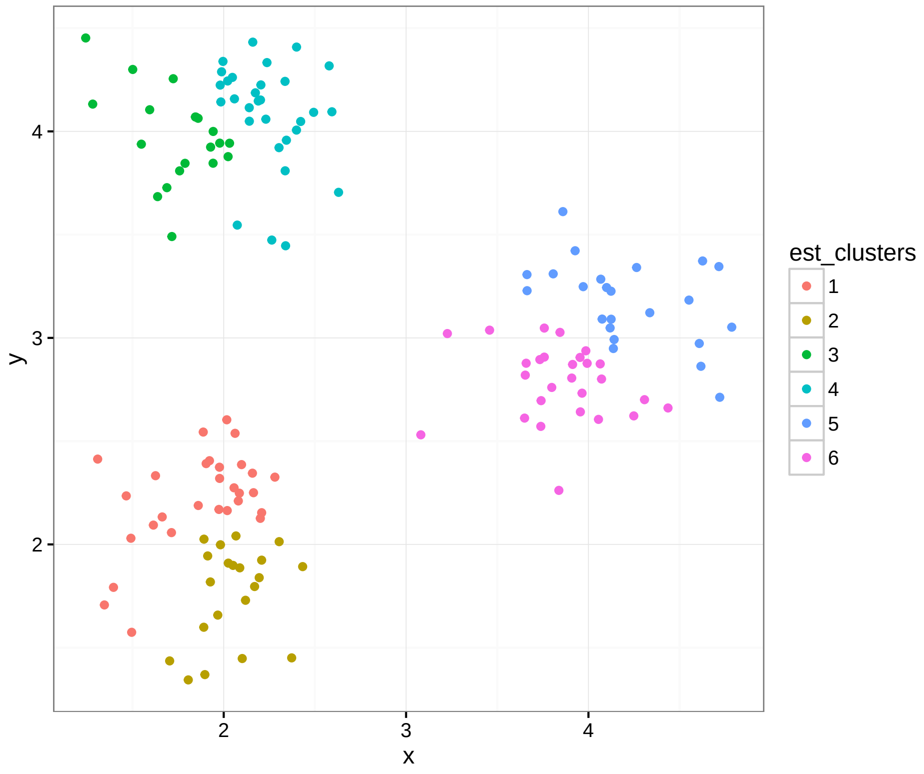
Cluster Assignments ($K = 4$)

```
> (data1 %>%
+    mutate(est_clusters=factor(cutree(myhclust, k=4))) %>%
+    ggplot()) + geom_point(aes(x=x, y=y, color=est_clusters))
```



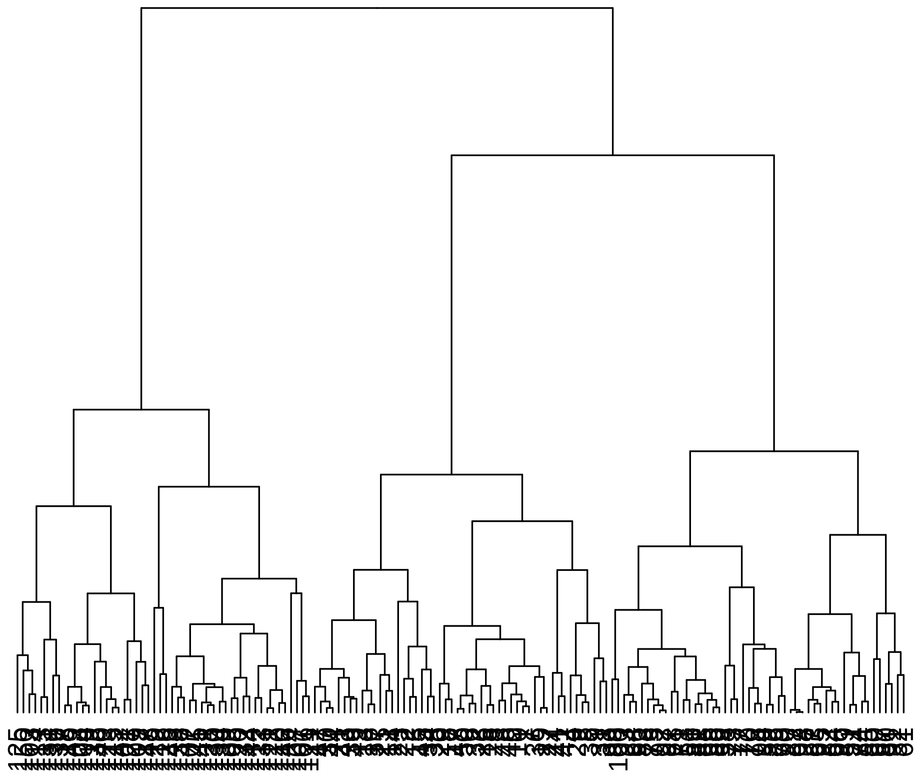
Cluster Assignments $(K = 6)$

```
> (data1 %>%
+    mutate(est_clusters=factor(cutree(myhclust, k=6))) %>%
+    ggplot()) + geom_point(aes(x=x, y=y, color=est_clusters))
```



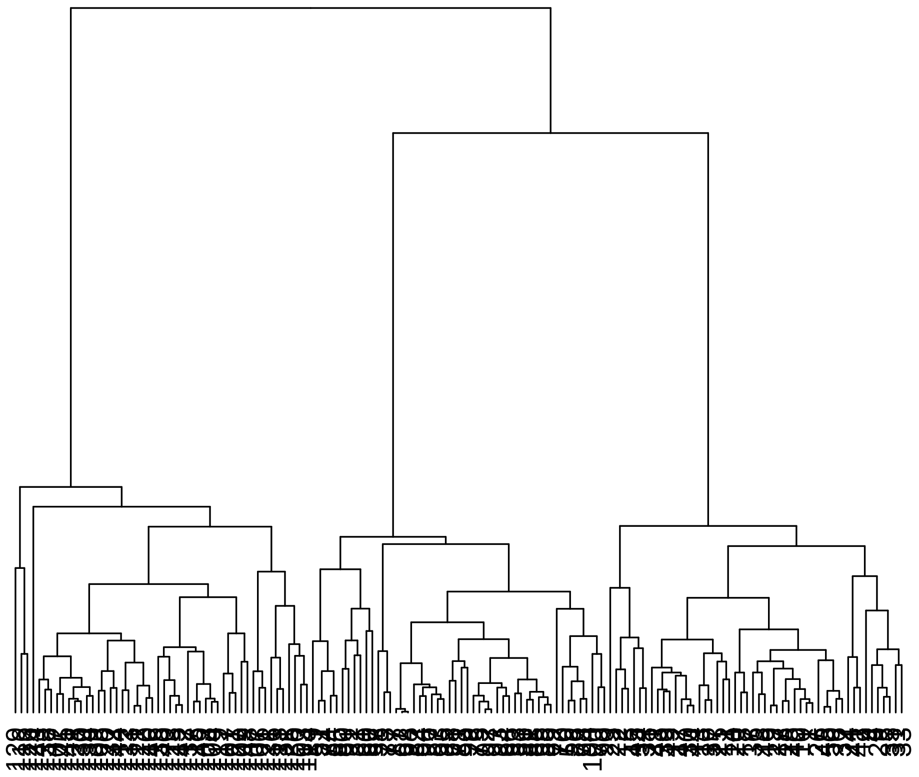
Linkage: Complete (Default)

```
> data1 %>% dist() %>% hclust(method="complete") %>%
+   as.dendrogram() %>% plot(axes=FALSE)
```



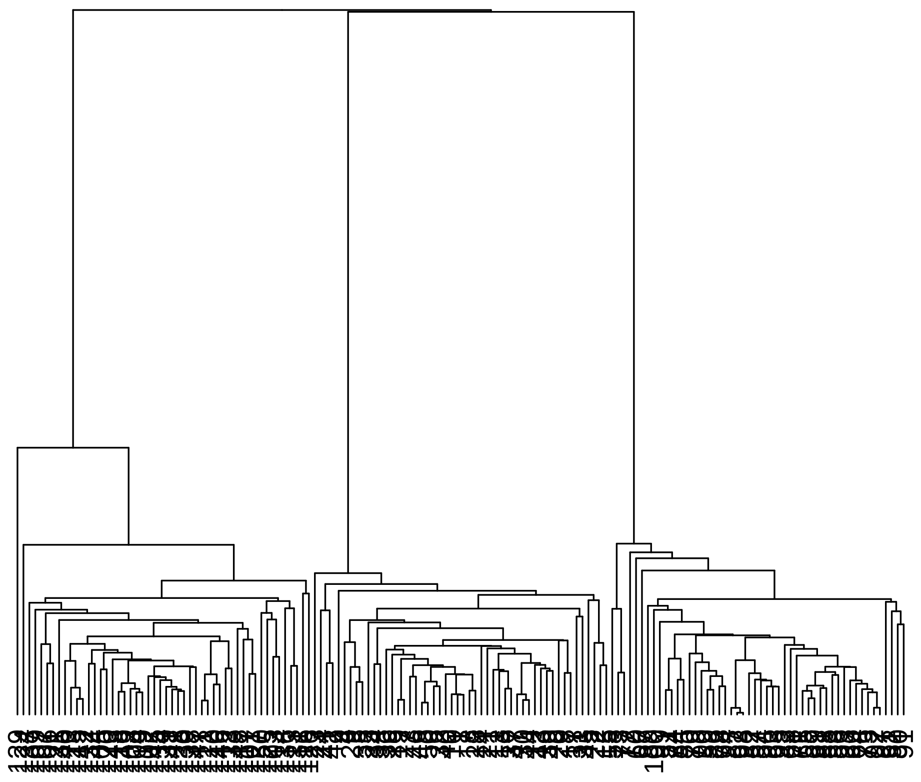
Linkage: Average

```
> data1 %>% dist() %>% hclust(method="average") %>%
+   as.dendrogram() %>% plot(axes=FALSE)
```



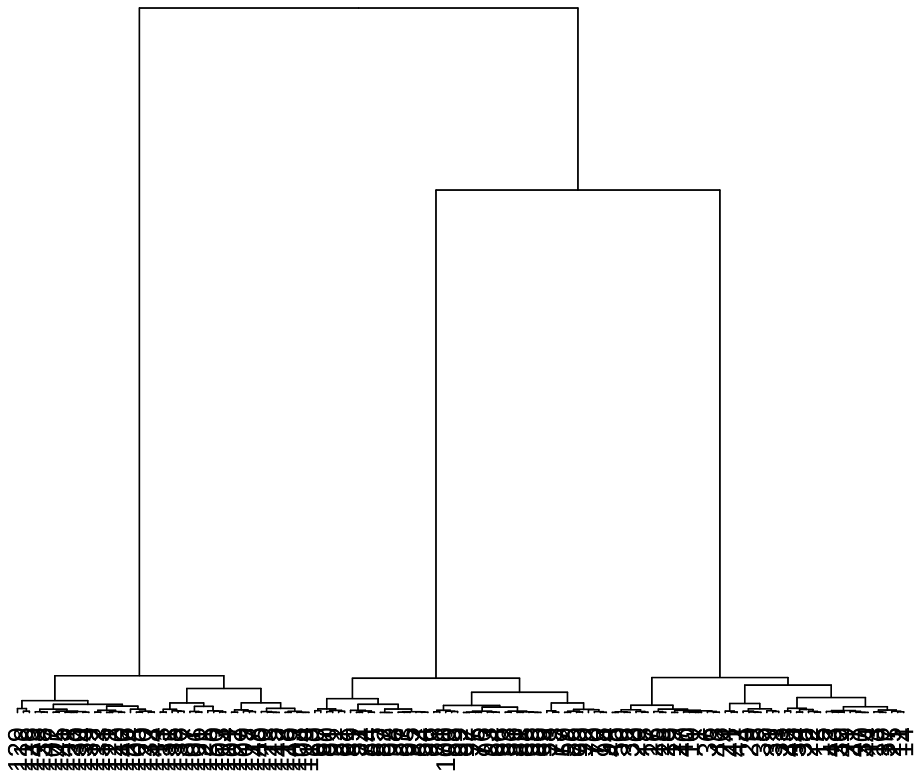
Linkage: Single

```
> data1 %>% dist() %>% hclust(method="single") %>%
+   as.dendrogram() %>% plot(axes=FALSE)
```

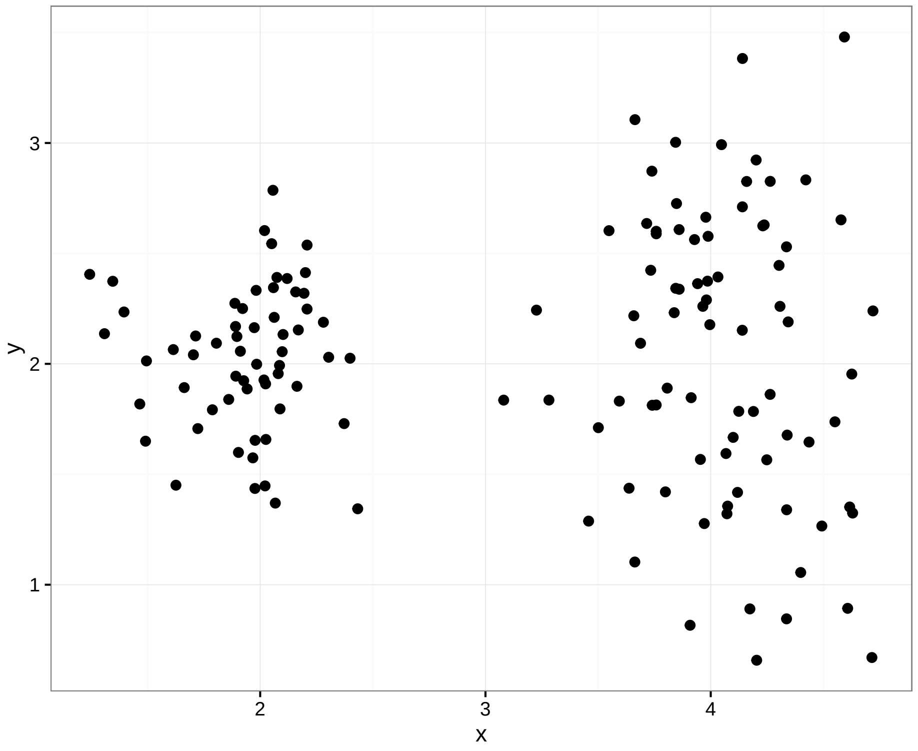


Linkage: Ward

```
> data1 %>% dist() %>% hclust(method="ward.D") %>%
+   as.dendrogram() %>% plot(axes=FALSE)
```

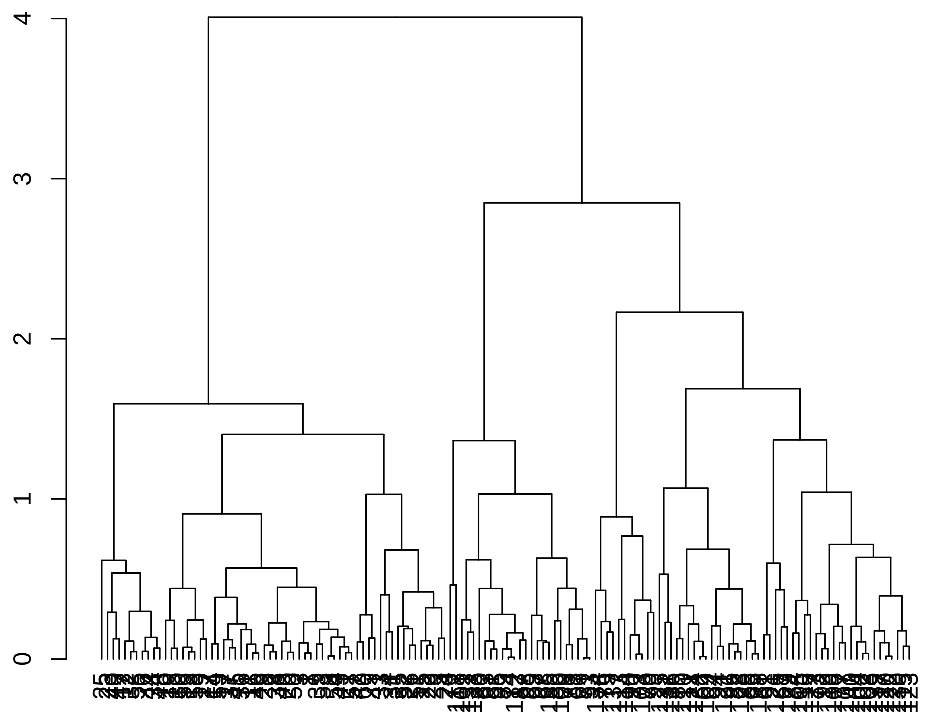


Hierarchical Clustering of `data2`



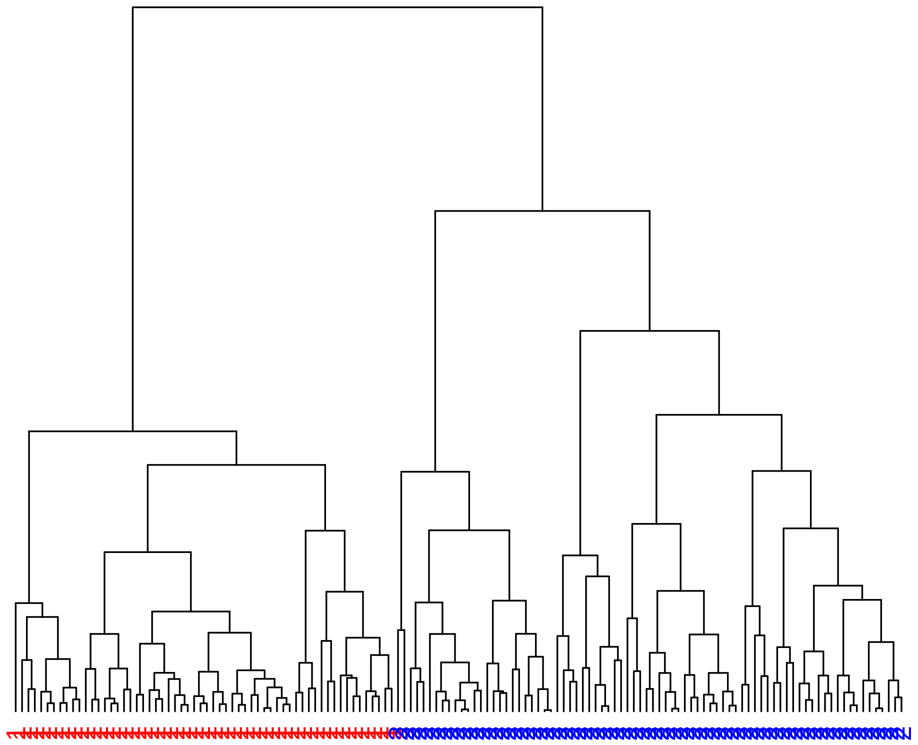
`as.dendrogram()`

```
> mydist <- dist(data2, method = "euclidean")
> myhclust <- hclust(mydist, method="complete")
> plot(as.dendrogram(myhclust))
```



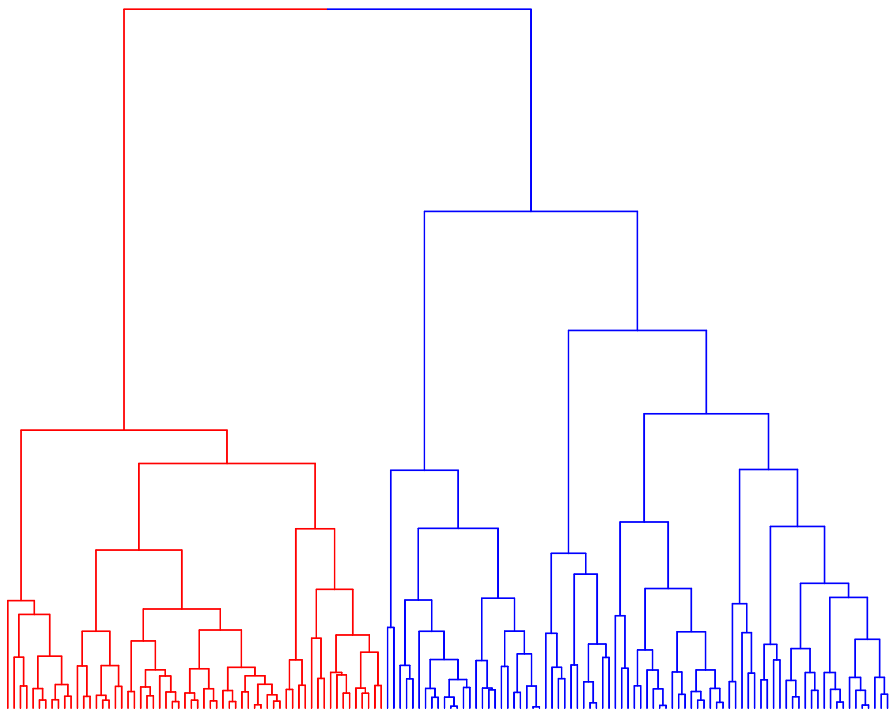
Modify the Labels

```
> library(dendextend)
> dend1 <- as.dendrogram(myhclust)
> labels(dend1) <- data2$true_clusters
> labels_colors(dend1) <-
+   c("red", "blue")[as.numeric(data2$true_clusters)]
> plot(dend1, axes=FALSE, main=" ", xlab=" ")
```



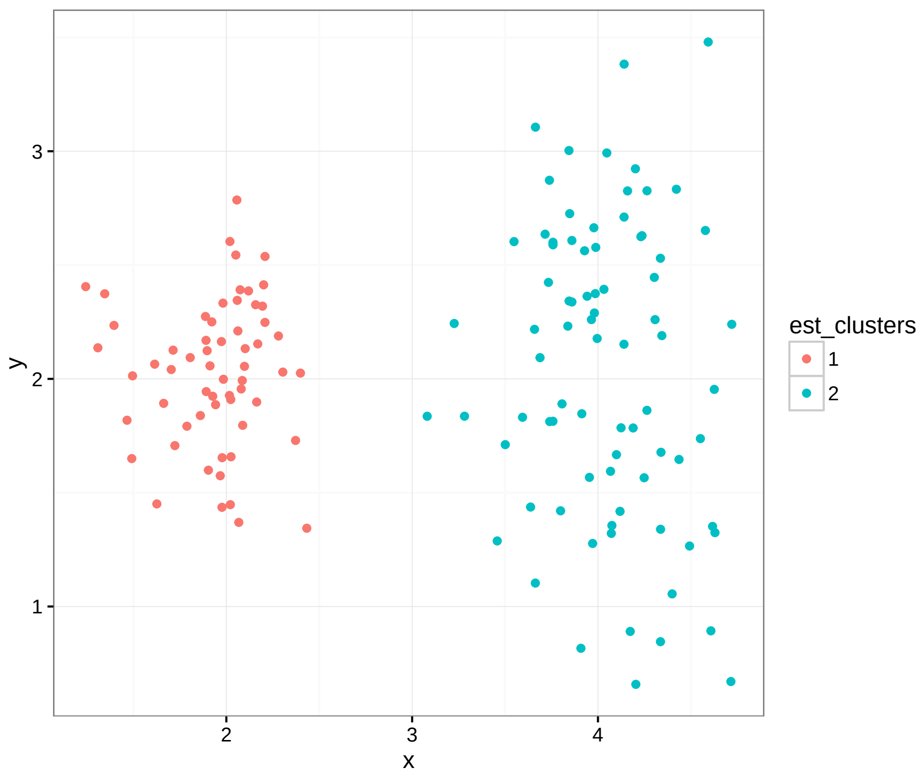
Color the Branches

```
> dend2 <- as.dendrogram(myhclust)
> labels(dend2) <- rep(" ", nrow(data2))
> dend2 <- color_branches(dend2, k = 2, col=c("red", "blue"))
> plot(dend2, axes=FALSE, main=" ", xlab=" ")
```



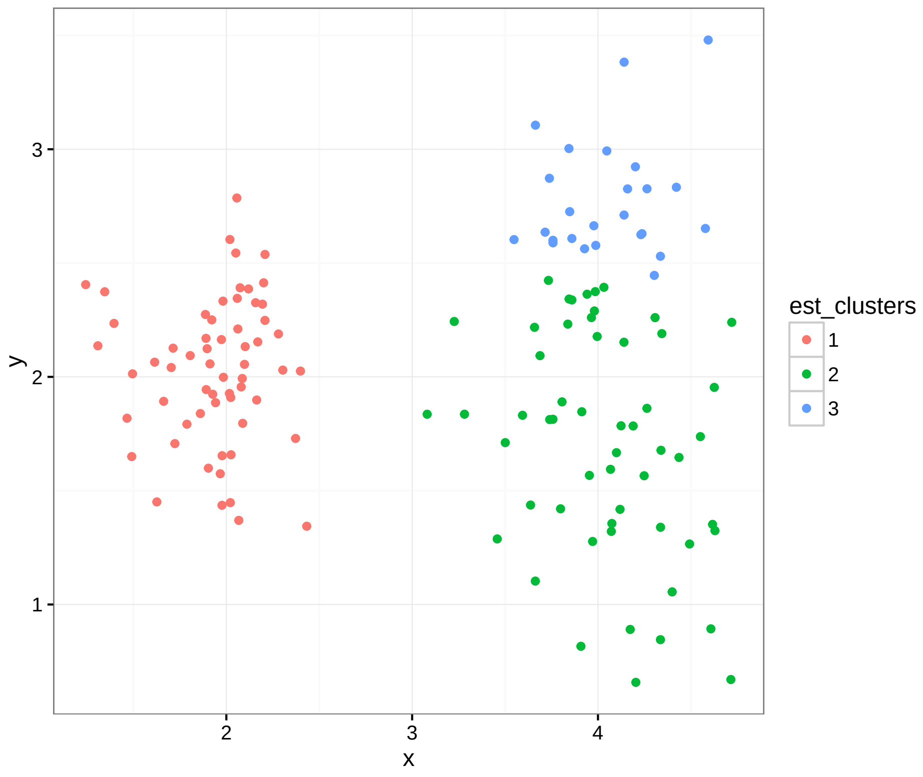
Cluster Assignments $(K = 2)$

```
> (data2 %>%
+    mutate(est_clusters=factor(cutree(myhclust, k=2))) %>%
+    ggplot()) + geom_point(aes(x=x, y=y, color=est_clusters))
```



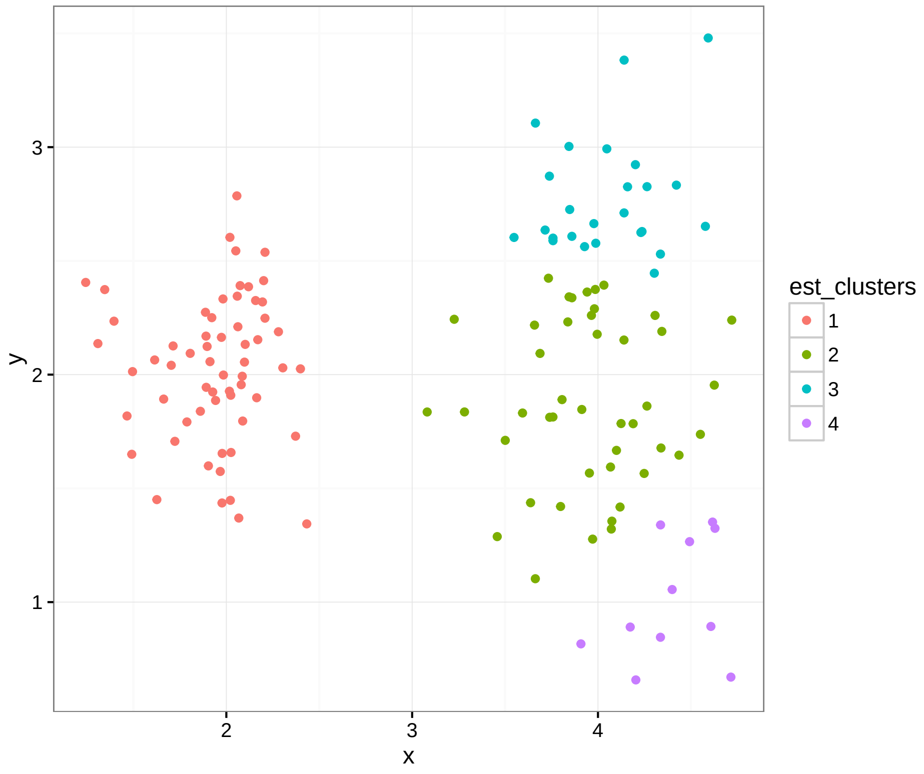
Cluster Assignments ($K = 3$)

```
> (data2 %>%
+    mutate(est_clusters=factor(cutree(myhclust, k=3))) %>%
+    ggplot()) + geom_point(aes(x=x, y=y, color=est_clusters))
```



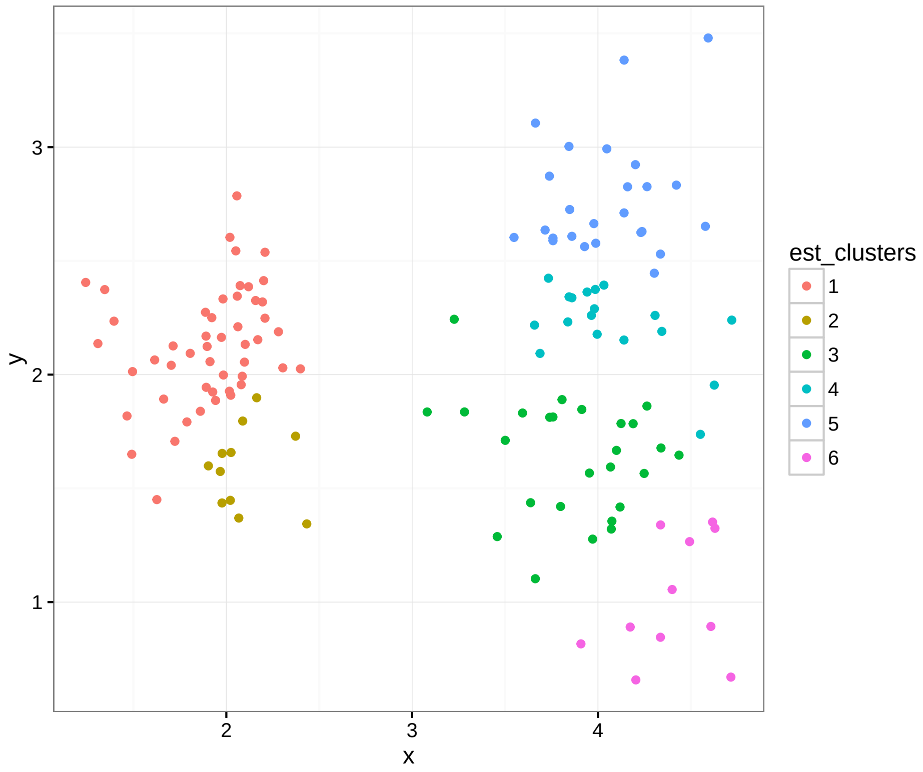
Cluster Assignments ($K = 4$)

```
> (data2 %>%
+     mutate(est_clusters=factor(cutree(myhclust, k=4))) %>%
+     ggplot()) + geom_point(aes(x=x, y=y, color=est_clusters))
```



Cluster Assignments $(K = 5)$

```
> (data2 %>%
+    mutate(est_clusters=factor(cutree(myhclust, k=6))) %>%
+    ggplot()) + geom_point(aes(x=x, y=y, color=est_clusters))
```



K-Means Clustering

Strategy

K-means clustering is a top-down, partitioning cluster analysis method that assigns each object to one of K clusters based on the distance between each object and the cluster centers, called *centroids*.

This is an iterative algorithm with potential random initial values.

The value of K is typically unknown and must be determined by the analyst.

Centroid

A centroid is the coordinate-wise average of all objects in a cluster.

Let A be a given cluster with objects $\boldsymbol{a} \in A$. Its centroid is:

$$\bar{\boldsymbol{a}} = \frac{1}{|A|} \sum_{\boldsymbol{a} \in A} \boldsymbol{a}$$

Algorithm

The number of clusters K must be chosen beforehand.

1. Initialize K cluster centroids.
2. Assign each object to a cluster by choosing the cluster with the smalllest distance (e.g., Euclidean) between the object and the cluster centroid.
3. Calculate new centroids based on the cluster assignments from Step 2.
4. Repeat Steps 2–3 until convergence.

Notes

The initialization of the centroids is typically random, so often the algorithm is run several times with new, random initial centroids.

Convergence is usually defined in terms of neglible changes in the centroids or no changes in the cluster assignments.

`kmeans()`

K-means clustering can be accomplished through the following function:

```
> str(kmeans)
function (x, centers, iter.max = 10L, nstart = 1L, algorithm = c("Hartigan-Wong",
    "Lloyd", "Forgy", "MacQueen"), trace = FALSE)
```

- `x`: the data to clusters, objects along rows
- `centers`: either the number of clusters K or a matrix giving initial centroids
- `iter.max`: the maximum number of iterations allowed
- `nstart`: how many random intial K centroids, where the best one is returned

`fitted()`

The cluster centroids or assigments can be extracted through the function `fitted()`, which is applied to the output of `kmeans()`.

The input of `fitted()` is the object returned by `kmeans()`. The key additional argument is called `method`.

When `method="centers"` it returns the centroids. When `method="classes"` it returns the cluster assignments.

K-Means Clustering of `data1`

```
> km1 <- kmeans(x=data1[,-3], centers=3, iter.max=100, nstart=5)
> est_clusters <- fitted(km1, method="classes")
> est_clusters
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [30] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3
 [59] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [88] 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[117] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[146] 2 2 2 2 2
```

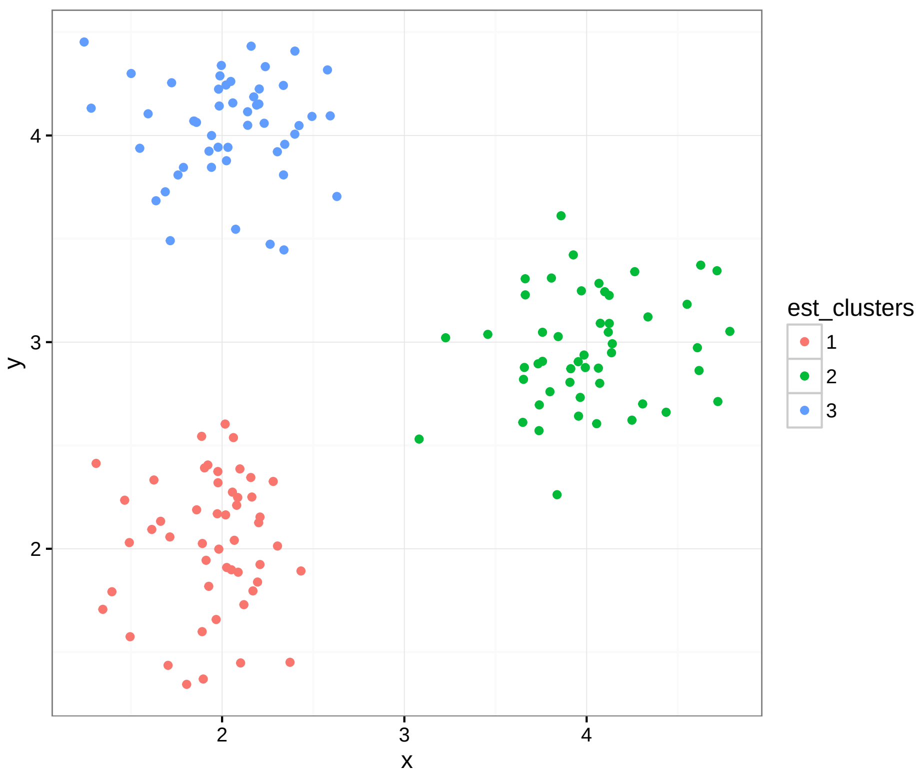
Centroids of `data1`

```
> centroids1 <- fitted(km1, method="centers") %>% unique()
> centroids1
         x        y
1 1.943184 2.028062
3 2.042872 4.037987
2 4.015934 2.962279
```

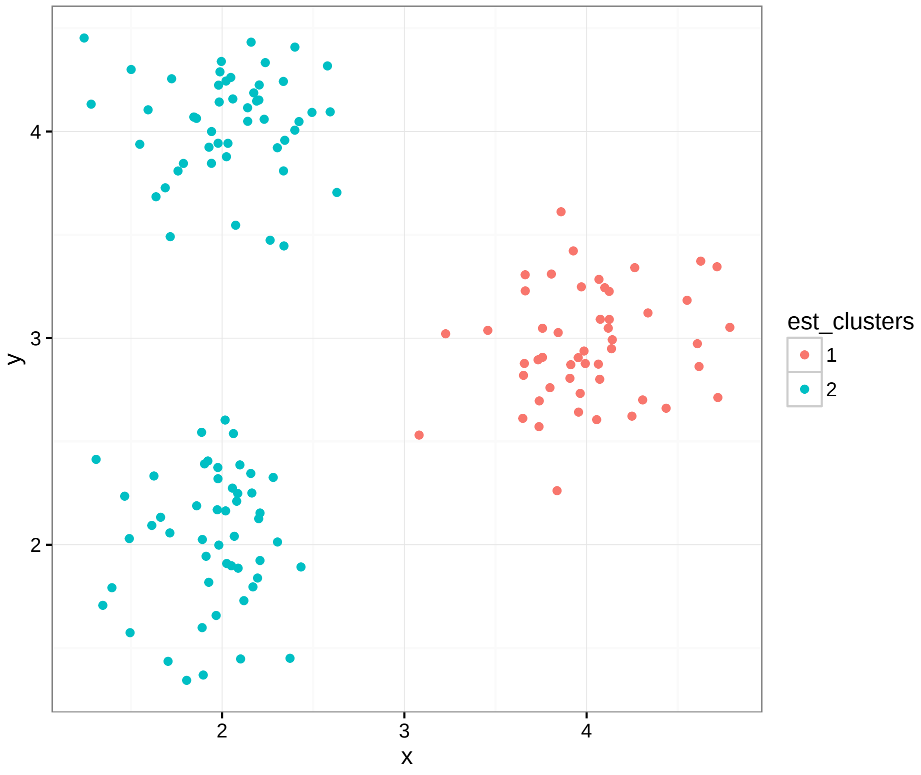
```
> est_clusters <- fitted(km1, method="classes")
> data1 %>% mutate(est_clusters = factor(est_clusters)) %>%
+   group_by(est_clusters) %>% summarize(mean(x), mean(y))
Source: local data frame [3 x 3]

  est_clusters  mean(x)  mean(y)
        (fctr)    (dbl)    (dbl)
1            1 1.943184 2.028062
2            2 4.015934 2.962279
3            3 2.042872 4.037987
```

Cluster Assignments ($K = 3$)

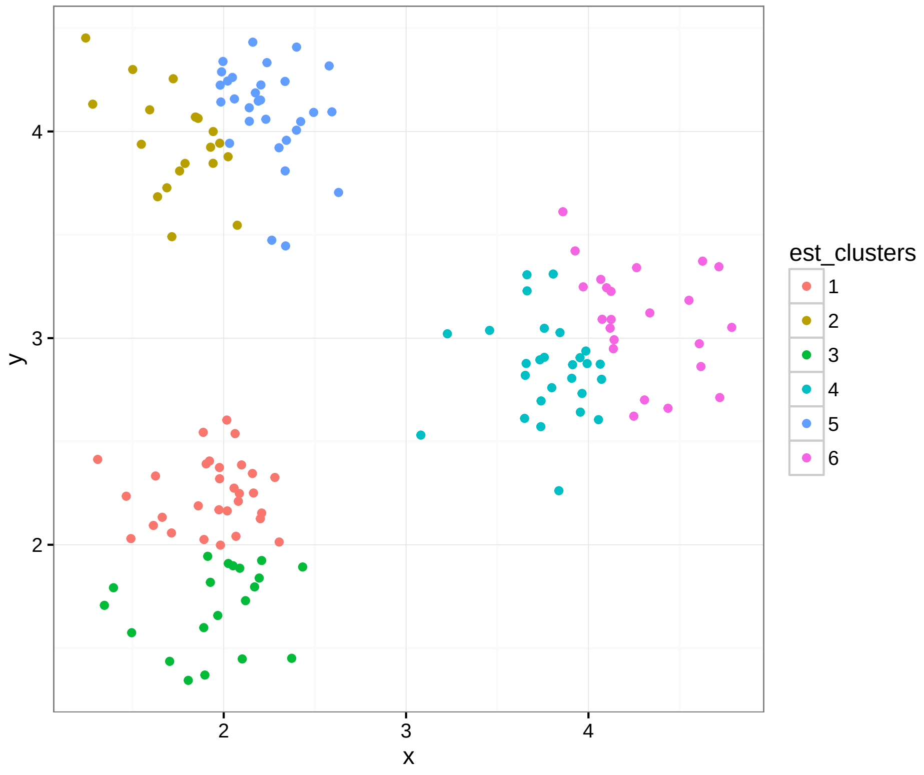
```
> est_clusters <- factor(est_clusters)
> ggplot(data1) + geom_point(aes(x=x, y=y, color=est_clusters))
```



Cluster Assignments ($K = 2$)



Cluster Assignments ($K = 6$)

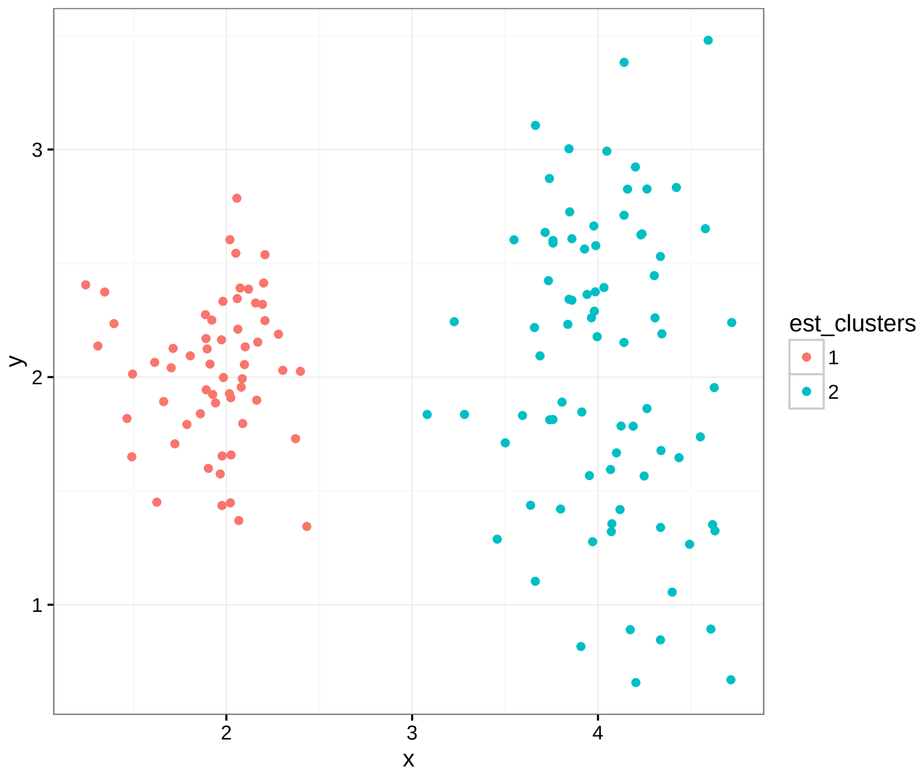


K-Means Clustering of `data2`

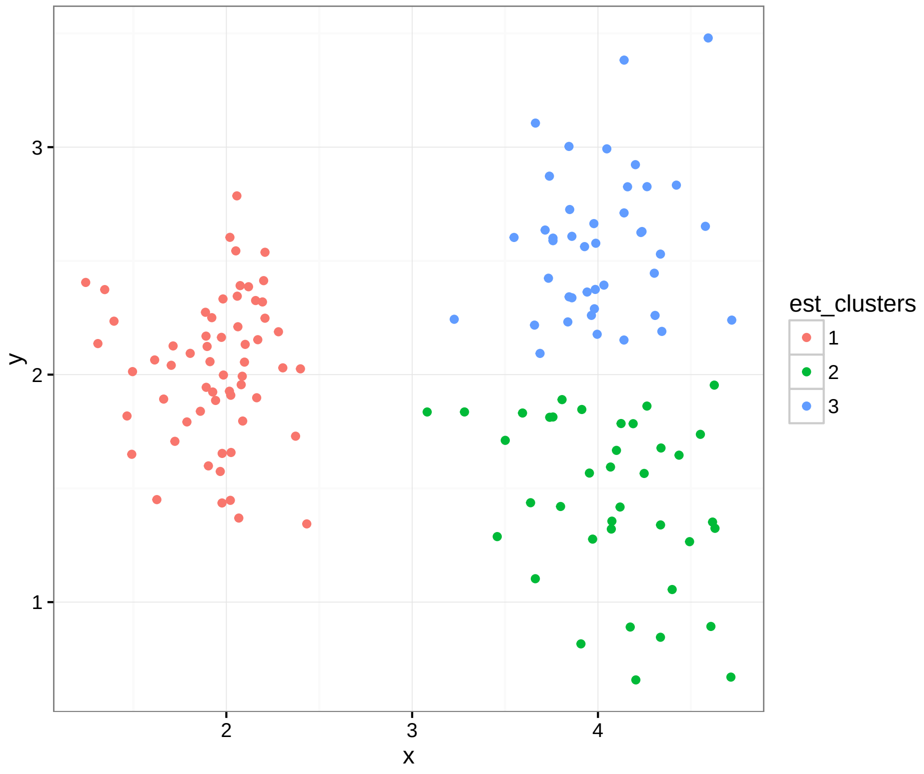
```
> km2 <- kmeans(x=data2[,-3], centers=2, iter.max=100, nstart=5)
> est_clusters <- fitted(km2, method="classes")
> est_clusters
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [30] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [59] 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [88] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[117] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

Cluster Assignments ($K = 2$)

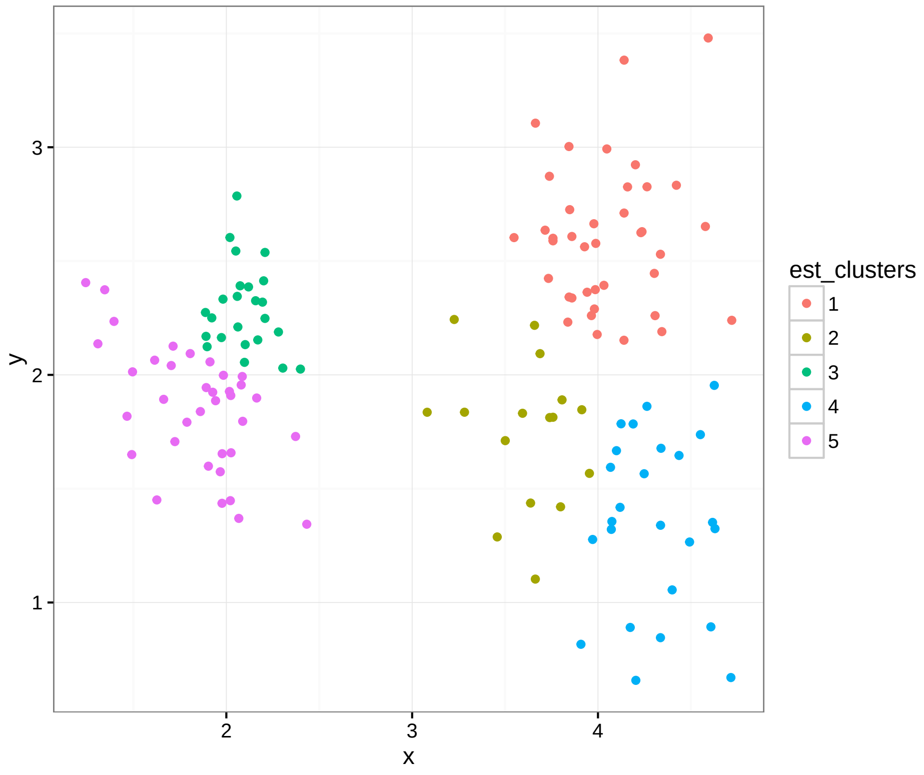
```
> est_clusters <- factor(est_clusters)
> ggplot(data2) + geom_point(aes(x=x, y=y, color=est_clusters))
```



Cluster Assignments ($K = 3$)



Cluster Assignments $(K = 5)$



Dimension Reduction

Principal Components Analysis

Summary of SML 201

Extras

License

https://github.com/SML201/lectures/blob/master/LICENSE.md

Source Code

https://github.com/SML201/lectures/tree/master/week12

Session Information

```
> sessionInfo()
R version 3.2.3 (2015-12-10)
Platform: x86_64-apple-darwin13.4.0 (64-bit)
Running under: OS X 10.11.4 (El Capitan)

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods
[7] base

other attached packages:
[1] dendextend_1.1.8 broom_0.4.0      dplyr_0.4.3
[4] ggplot2_2.1.0    knitr_1.12.3     magrittr_1.5
[7] devtools_1.11.1

loaded via a namespace (and not attached):
 [1] Rcpp_0.12.4       whisker_0.3-2     mnormt_1.5-4
 [4] munsell_0.4.3     lattice_0.20-33   colorspace_1.2-6
 [7] R6_2.1.2          highr_0.5.1       stringr_1.0.0
[10] plyr_1.8.3        tools_3.2.3       parallel_3.2.3
[13] grid_3.2.3        nlme_3.1-127      gtable_0.2.0
[16] psych_1.5.8       DBI_0.3.1         withr_1.0.1
[19] htmltools_0.3.5   lazyeval_0.1.10   yaml_2.1.13
[22] digest_0.6.9      assertthat_0.1    tidyr_0.4.1
[25] reshape2_1.4.1    formatR_1.3       memoise_1.0.0
[28] evaluate_0.8.3    rmarkdown_0.9.5.9 labeling_0.3
[31] stringi_1.0-1     scales_0.4.0
```