K-means and hierarchical clustering: visualizing results

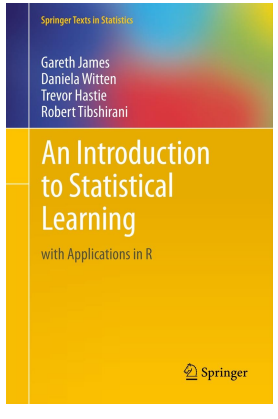
Rei Sanchez-Arias, Ph.D.

Visualizing K-means and hierarchical clustering using `{ggplot2}`, `{factoextra}`, and `{broom}`

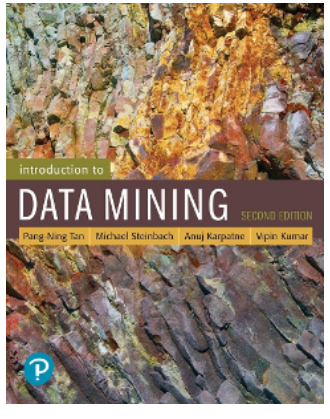
Some Resources

Some resources to check

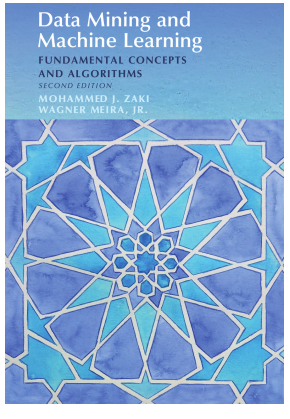
"Introduction to Statistical Learning" by James, Witten, Hastie, Tibshiriani.



"Introduction to Data Mining" by Tan, Steinbach, Karpatne, Kumar



"Data Mining and Machine Learning: Concepts and Algorithms" by Zaki and Meira



> Examples and materials in this set of slides are adapted from the books mentioned above.

Unsupervised Learning

Goals of unsupervised learning

The goal is to discover interesting things about the measurements: is there an informative way to visualize the data? Can we discover subgroups among the variables or among the observations?

Clustering: a broad class of methods for discovering **unknown subgroups** in data.

Techniques for unsupervised learning are of growing importance in a number of fields:

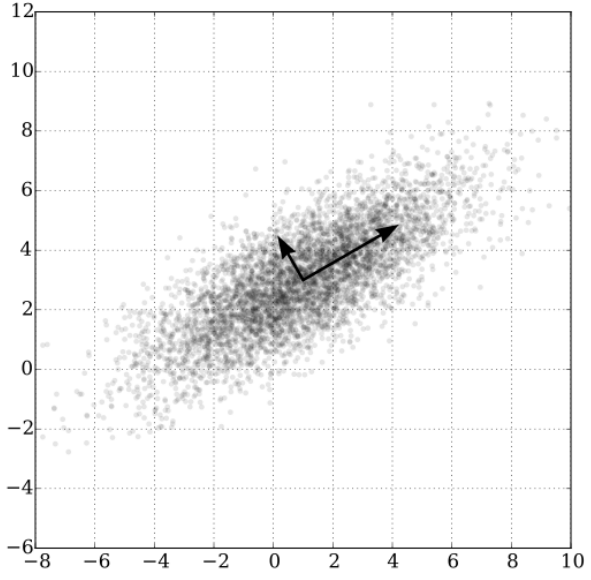
- subgroups of breast cancer patients grouped by their gene expression measurements,
- groups of shoppers characterized by their browsing and purchase histories,
- movies grouped by the ratings assigned by movie viewers.

Clustering

- Clustering refers to a very broad set of techniques for **finding subgroups**, or **clusters**, in a data set.
- We seek a partition of the data into distinct groups so that the observations within each group are quite similar to each other,
- It make this concrete, we must define what it means for two or more observations to be **similar** or **different.**
- Indeed, this is often a domain-specific consideration that must be made based on knowledge of the data being studied.

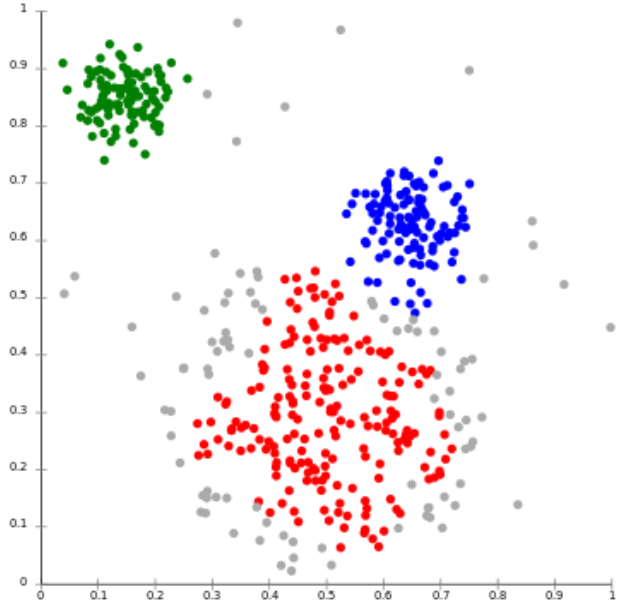
PCA vs Clustering

PCA looks for a **low-dimensional representation** of the observations that explains a good fraction of the variance.



Nicoguaro / CC BY

Clustering looks for **homogeneous subgroups** among the observations.



Chire / CC BY-SA

Two popular clustering methods

- In K**-means clustering**, we seek to partition the observations into a pre-specified number of clusters.
- In **hierarchical clustering**, we do not know in advance how many clusters we want; in fact, we end up with a **tree-like** visual representation of the observations, called a dendrogram, that allows us to view at once the clusterings obtained for each possible number of clusters, from 1 to n.

K-means clustering

Details of K-means clustering

Let $C_1, \ldots, C_K$ denote sets containing the indices of the observations in each cluster. These sets satisfy two properties:

1. $C_1 \cup C_2 \cup \ldots \cup C_K = \{1, \ldots, n\}$. In other words, **each observation belongs to at least one** of the K clusters.
2. $C_i \cap C_j = \emptyset$ for all $i \neq j$. In other words,the clusters are **non-overlapping**: no observation belongs to more than one cluster.

For instance, if the ith observation is in the kth cluster, then $i \in C_k$

Details of K-means clustering (cont.)

The idea behind K-means clustering is that a good clustering is one for which the **within-cluster variation** is as small as possible.

The within-cluster variation for cluster C_k is a measure $WCV(C_k)$ of the amount by which the observations within a cluster differ from each other. We want to solve:

$$\min_{C_1,\ldots,C_K} \left\{ \sum_{k=1}^{K} WCV(C_k) \right\}$$

In words, this formula says that we want to partition the observations into K clusters such that the total within-cluster variation, summed over all K clusters, is as small as possible.

Within-cluster variation

Typically we use Euclidean distance

$$WCV(C_k) = \frac{1}{|C_K|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} \left(x_{ij} - x_{i'j}\right)^2$$

where $|C_K|$ denotes the number of observations in the kth cluster

Combining the previous two relationships, gives the optimization problem:

$$\min_{C_1,\ldots,C_K} \left\{ \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} \left(x_{ij} - x_{i'j}\right)^2 \right\}$$

K-means clustering algorithm

1. Randomly assign a number, from 1 to K, to each of the observations. These serve as initial cluster assignments for the observations.

2. Iterate until the cluster assignments stop changing:

   2.1 For each of the K clusters, compute the cluster **centroid.** The kth cluster centroid is the vector of the p feature **means** for the observations in the kth cluster.

   2.2 Assign each observation to the cluster whose centroid is closest (where **closest** is defined using Euclidean distance).

Artist-in-residence
@RStudio

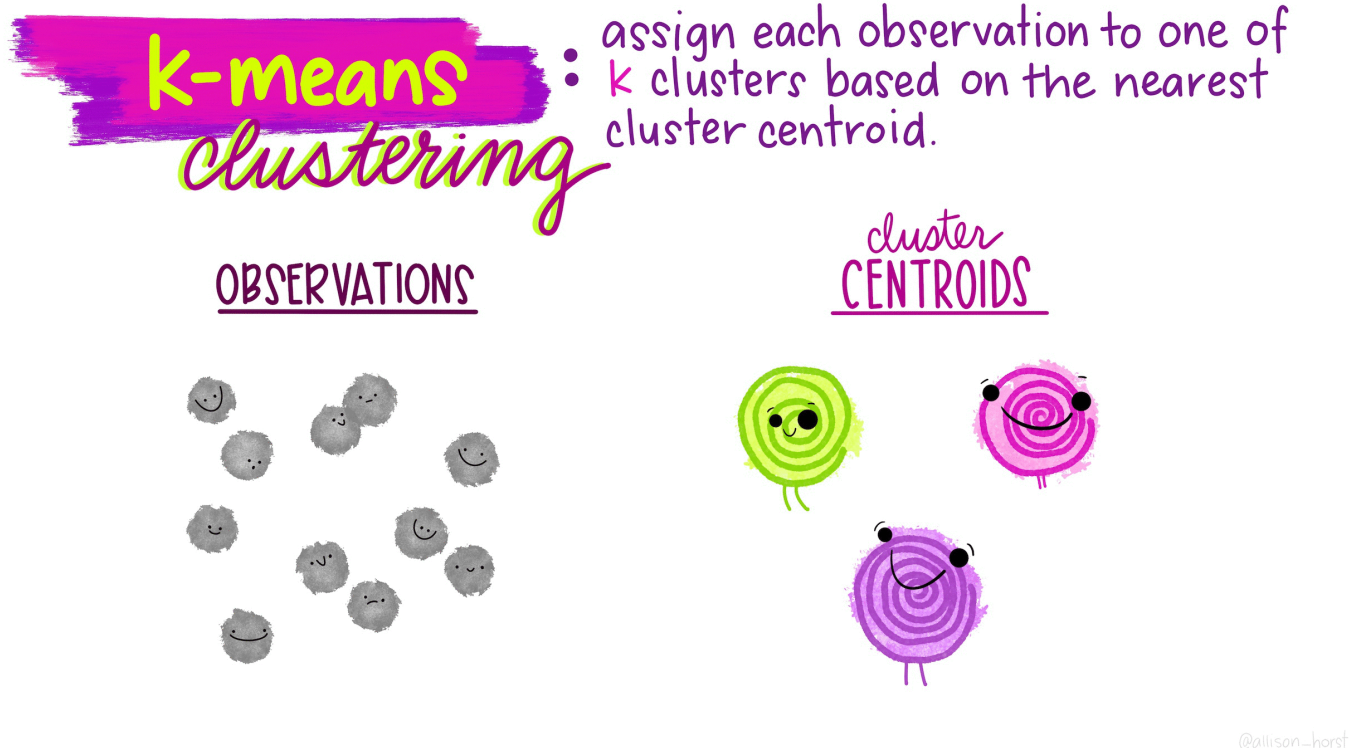
Allison Horst
allisonhorst

Stats | Art | Data Science | SciComm | Teaching

Follow ···

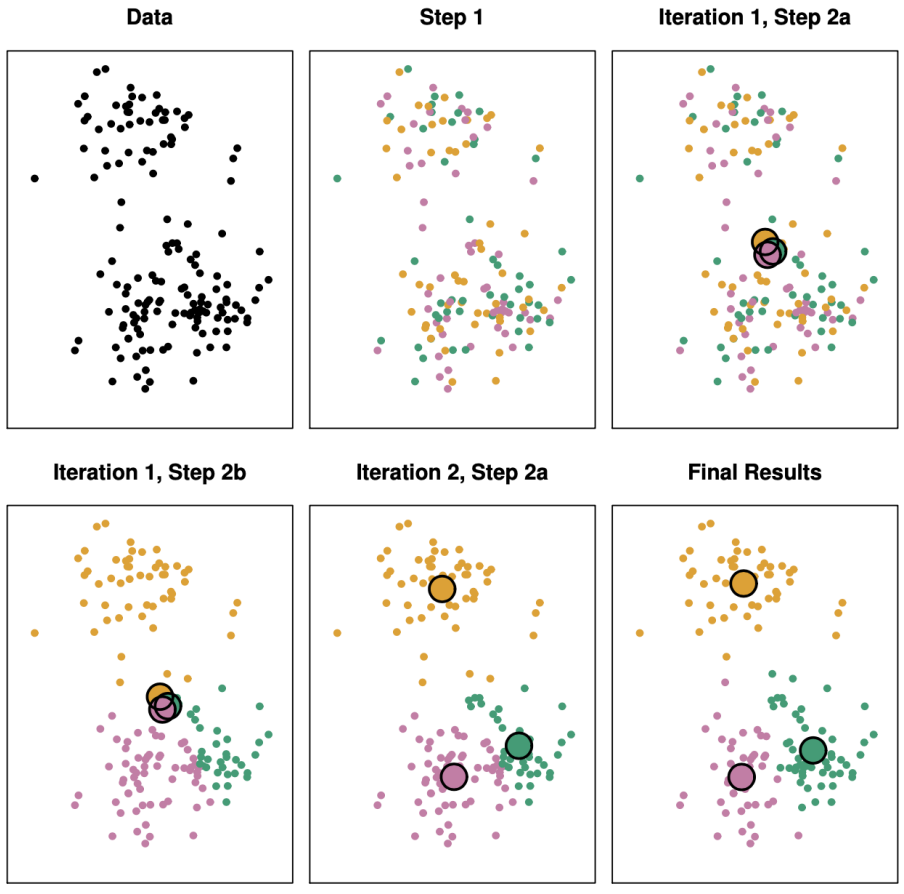
682 followers · **5** following · ☆ **3**

UCSB
https://www.allisonhorst.com/
@allison_horst



Source: Animation using the artwork of Dr. Allison Horst @allison_horst

K-means illustration



From: ISL by James, Witten, Hastie, Tibshiriani.

Top left: The observations are shown.

Top center: In Step 1 of the algorithm, each observation is **randomly assigned** to a cluster.

Top right: In Step 2(a), cluster **centroids** are computed. These are shown as large colored disks. Initial centroids are almost completely overlapping since the initial cluster assignments were chosen randomly.

Bottom left: In Step 2(b), each observation is assigned to the **nearest centroid**.

Bottom center: Step 2(a) is once again performed, leading to **new cluster centroids**.

Bottom right: results after 10 iterations.

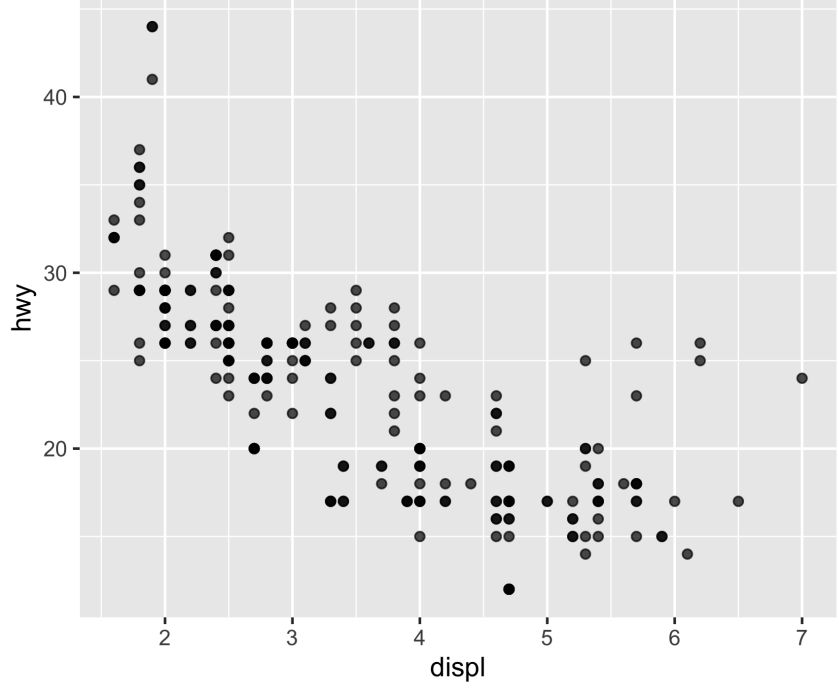
K-means implementation

We can use the `stats::kmeans()` built-in function in R. Below we consider the variables `displ` (engine displacement, in litres) and `hwy` (highway miles per gallon) from the `mpg` dataset for illustration.

```
library(tidyverse)
library(factoextra)
set.seed(5771)    # for randomness
```

The `tidyverse` is used here for data exploration, and `factoextra` as an alternative for easy visualization

```
# select only two columns
mpg_mini <- select(mpg, displ, hwy)
# create a scatterplot with the two attribu
ggplot(data = mpg_mini,
       aes(x = displ, y = hwy)) +
  geom_point(alpha = 0.7)
```



Clustering with `kmeans()`

```
mpg_kmeans <- kmeans(mpg_mini, centers = 3)
```

```
## K-means clustering with 3 clusters of sizes 48, 95, 91
##
## Cluster means:
##      displ      hwy
## 1 2.060417 31.47917
## 2 3.105263 25.34737
## 3 4.598901 17.20879
##
## Clustering vector:
##   [1] 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 2 2
##  [44] 3 2 3 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##  [87] 3 3 3 3 2 2 2 2 3 2 2 2 3 1 1 1 1 1 1 1 1 2 2 1
## [130] 3 3 3 3 3 3 3 3 3 3 3 3 1 2 1 1 2 2 2 2 2 3 3 3 3
## [173] 2 3 3 3 3 3 3 1 2 1 1 2 2 2 2 1 1 1 2 2 2 1 1 1 1
## [216] 1 1 1 1 2 2 1 1 1 2 2 1 1 1 2 1 2 2 2
##
## Within cluster sum of squares by cluster:
## [1] 663.9740 338.5042 443.0029
##  (between_SS / total_SS =  83.3 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "within
## [6] "betweenss"    "size"         "iter"         "ifault
```

To check the results, let us add the results to the dataframe of interest, converting the cluster assignments to a *factor* variable:

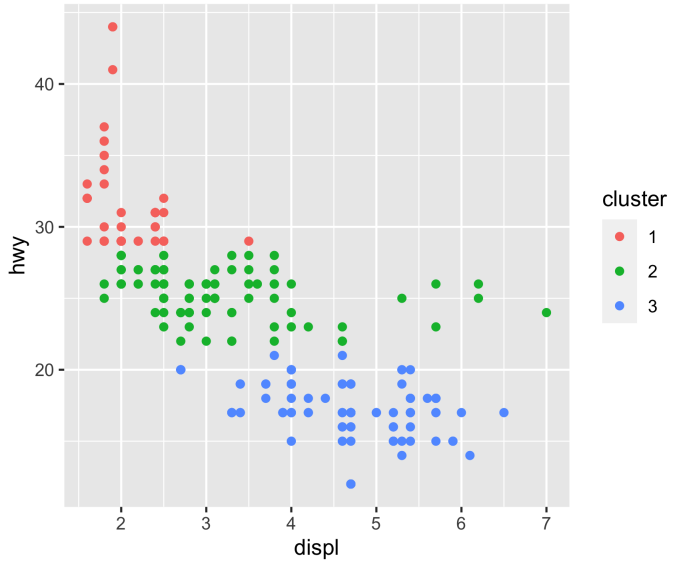
```
mpg_km_results <- mpg_mini %>%
  mutate(cluster = factor(mpg_kmeans$cluste
```

Centroids:

```
##      displ      hwy
## 1 2.060417 31.47917
## 2 3.105263 25.34737
## 3 4.598901 17.20879
```

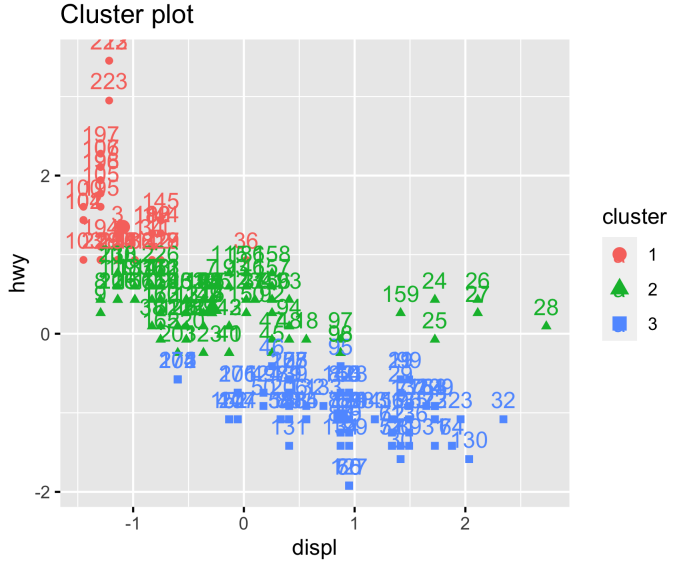
Visualization

```
ggplot(data = mpg_km_results) +
  geom_point(aes(x = displ, y = hwy,
                 color = cluster))
```



Recall that color assignment here is simply a visual aid, that helps us differentiate among the groups (we could also use `shape` as the aesthetic for example)

```
fviz_cluster(mpg_kmeans,
             data = mpg_mini,
             ellipse = FALSE)
```



The `factoextra::fviz_cluster()` can be used to quickly check results from clustering.

Testing other values of K

Next, we use the power of the **broom** package to visualize multiple clustering results. See https://www.tidymodels.org/learn/statistics/k-means/

```
library(broom)
```

We cluster the data 9 times, each using a different value of K, and create columns containing the tidied, glanced and augmented data. Here we use the `map` function from the `purrr` package

```
kclusts <-
  tibble(k = 1:9) %>%
  mutate(
    kclust = map(k, ~kmeans(mpg_mini, .x)),
    tidied = map(kclust, tidy),
    glanced = map(kclust, glance),
    augmented = map(kclust, augment,
                    mpg_mini)
  )
```

We also turn these into 3 separate data sets each representing a different type of data: using `tidy()`, `augment()`, and `glance()`.

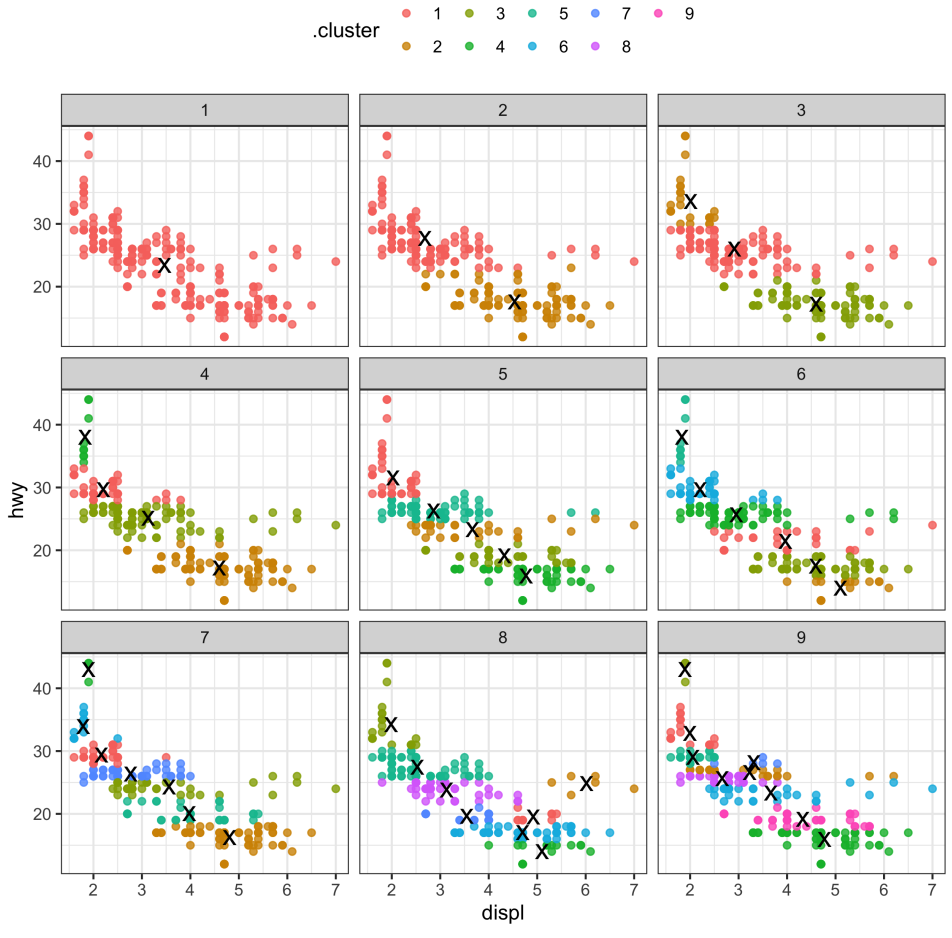
Tidy up and plot

```
clusters <- kclusts %>%
  unnest(cols = c(tidied))

assignments <- kclusts %>%
  unnest(cols = c(augmented))

clusterings <- kclusts %>%
  unnest(cols = c(glanced))
```

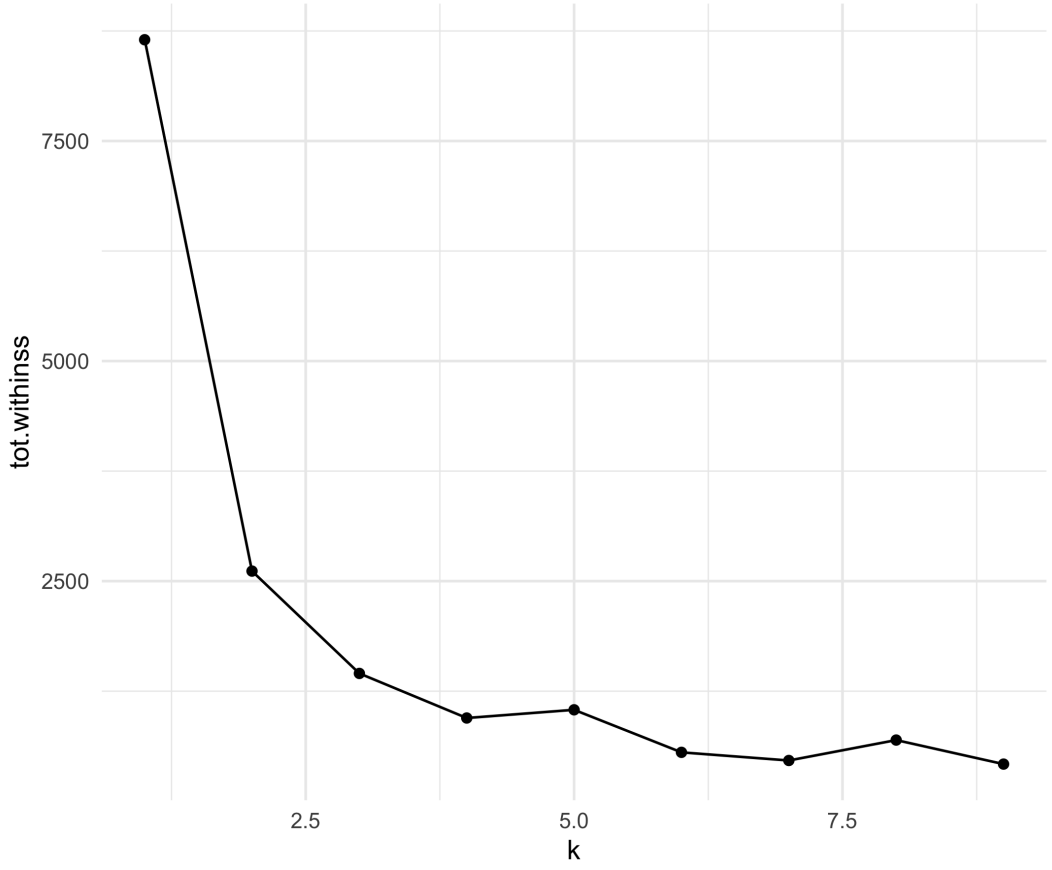
```
ggplot(assignments,
       aes(x = displ, y = hwy)) +
  geom_point(aes(color = .cluster),
             alpha = 0.8) +
  facet_wrap(~ k) +
  geom_point(data = clusters, size = 5,
             shape = "x") +
  theme_bw() +
  theme(legend.position = "top")
```



Tidy up and plot (cont.)

The data from `glance()` fills a different but equally important purpose; it lets us view trends of some summary statistics across values of K. Of particular interest is the **total within sum of squares**, saved in the `tot.withinss` column.

```
ggplot(clusterings,
       aes(x = k, y = tot.withinss)) +
  geom_line() +
  geom_point() +
  theme_minimal()
```



Hierarchical clustering

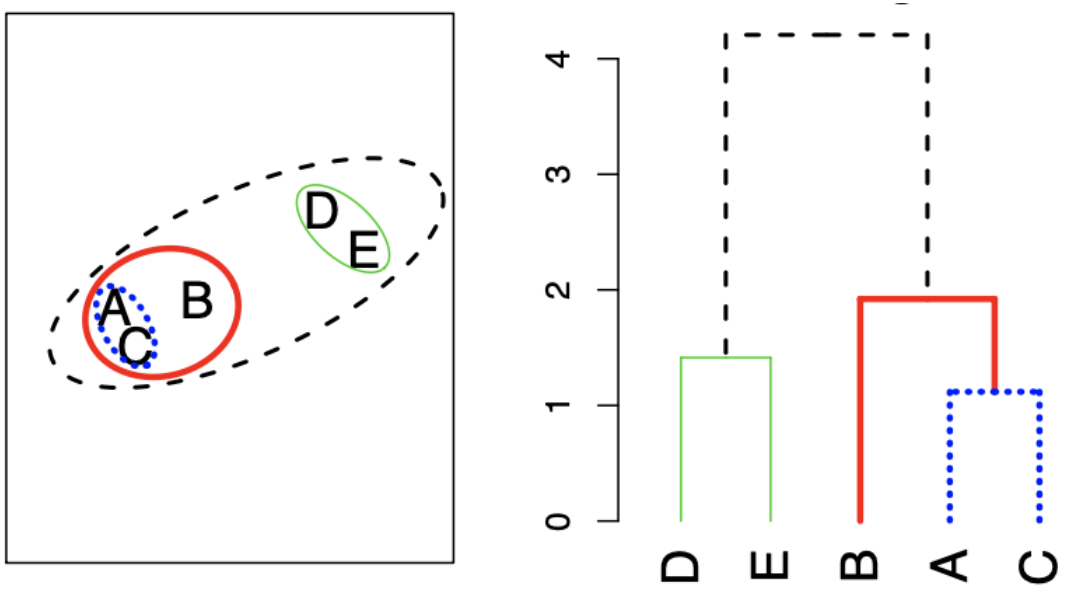
Comparison

K-means clustering requires us to pre-specify the number of clusters K. Sometimes this can be a disadvantage (later we discuss strategies for choosing K)

Hierarchical clustering is an alternative approach which does not require that we commit to a particular choice of K.

We describe bottom-up or **agglomerative** clustering. This is the most common type of hierarchical clustering, and refers to the fact that a *dendrogram* is built starting from the leaves and combining clusters up to the trunk.

Illustration 1



From: ISL by James, Witten, Hastie, and Tibshiriani.

The approach in words:

- Start with each point in its own cluster.
- Identify the **closest** two clusters and merge them.
- Repeat.
- Ends when all points are in a single cluster.

Illustration 2

- Observations 5 and 7 are quite similar to each other, as are observations 1 and 6.
- Observation 9 is no more similar to observation 2 than it is to observations 8, 5, and 7, even though observations 9 and 2 are close together in terms of horizontal distance.
- This is because observations 2, 8, 5, and 7 all fuse with observation 9 at the same height, approximately 1.8.

Figure from "Introduction to Statistical Learning"_ by James, Witten, Hastie, Tibshiriani.

The raw data on the right was used to generate the dendrogram on the left.

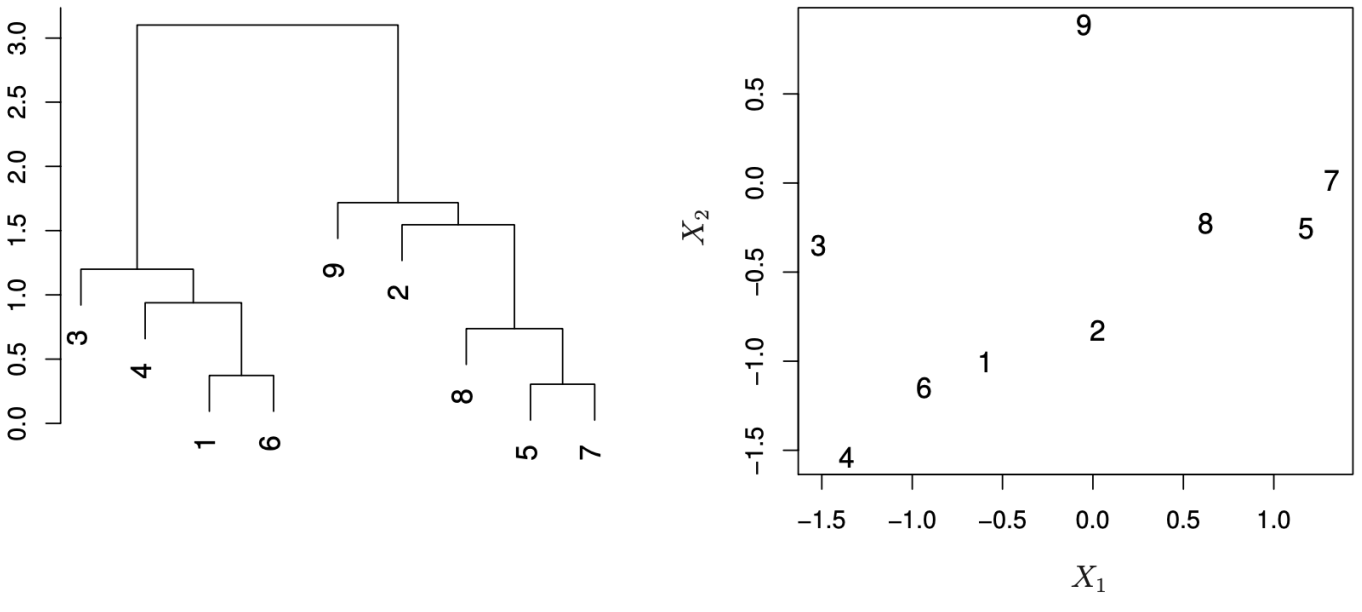
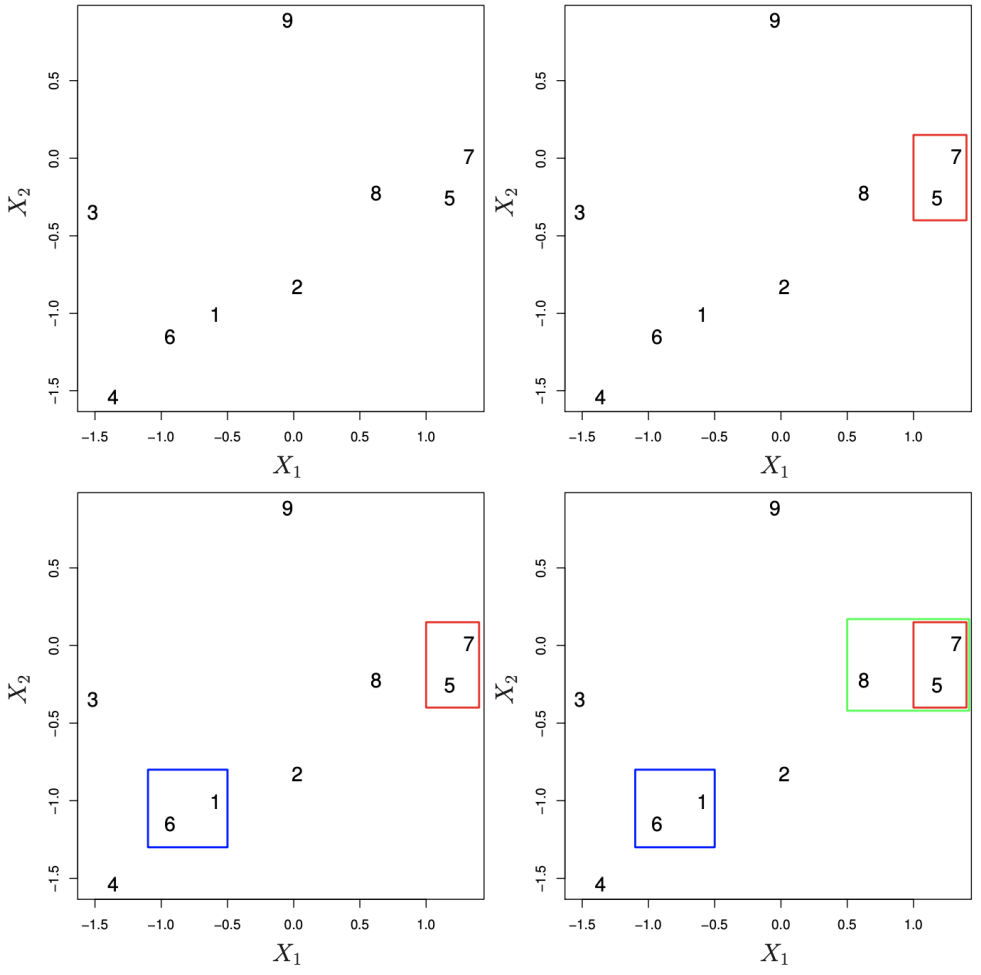


Illustration 2 (cont.)



From: ISL by by James, Witten, Hastie, and Tibshiriani.

Artist-in-residence
@RStudio

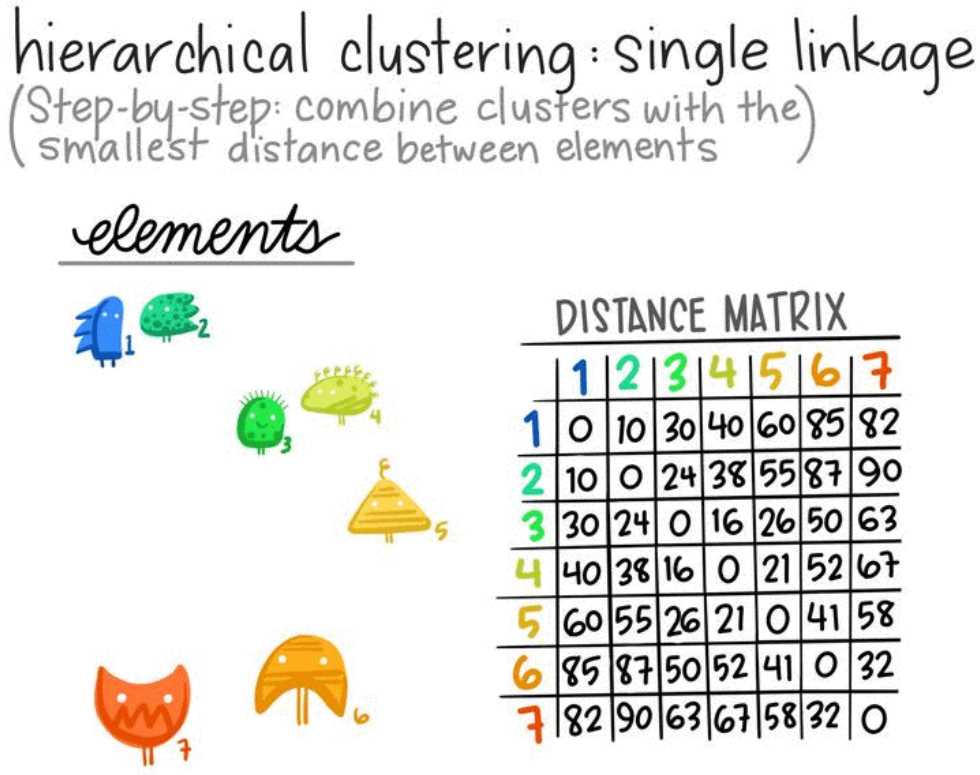
Allison Horst
allisonhorst

Stats | Art | Data Science | SciComm | Teaching

Follow ···

682 followers · **5** following · ☆ **3**

UCSB
https://www.allisonhorst.com/
@allison_horst



	1	2	3	4	5	6	7
1	0	10	30	40	60	85	82
2	10	0	24	38	55	87	90
3	30	24	0	16	26	50	63
4	40	38	16	0	21	52	67
5	60	55	26	21	0	41	58
6	85	87	50	52	41	0	32
7	82	90	63	67	58	32	0

1/7

Source: Animation using the artwork of Dr. Allison Horst @allison_horst

Types of linkage

Linkage	Description
Complete	Maximal inter-cluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the **largest** of these dissimilarities.
Single	Minimal inter-cluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the **smallest** of these dissimilarities.
Average	Mean inter-cluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the **average** of these dissimilarities.
Centroid	Dissimilarity between the centroid for cluster A (a mean vector of length p) and the centroid for cluster B. Centroid linkage can result in undesirable inversions.

Types of linkage (cont.)

Linkage	Description
Complete	$\max\{d(a,b) : a \in A, b \in B\}$
Single	$\min\{d(a,b) : a \in A, b \in B\}$
Average	$\frac{1}{\lvert A\rvert \cdot \lvert B\rvert} \sum_{a \in A} \sum_{b \in B} d(a,b)$
Centroid	$\lVert c_A - c_B \rVert$ where c_A and c_B are the centroids of clusters A and B respectively

Clustering with `stats::hclust()`

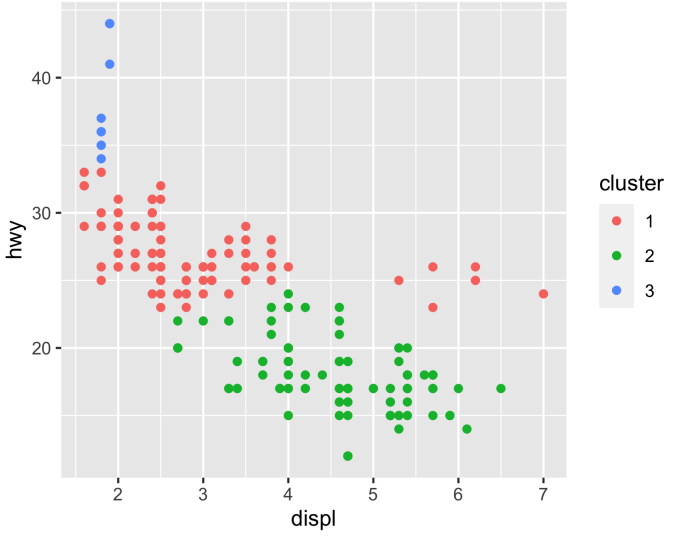
```
# get distance matrix
mpg_dist <- dist(mpg_mini)
# hierarchical clustering with average linka
mpg_hclust <- hclust(mpg_dist,
                     method = "average")
```

```
##
## Call:
## hclust(d = mpg_dist, method = "average")
##
## Cluster method   : average
## Distance         : euclidean
## Number of objects: 234
```

To find the cluster assignments, let us *cut* the dendrogram tree to the proper number of clusters using the `cutree()` (in our case 3)

```
mpg_cut_3 <- cutree(mpg_hclust, k = 3)
mpg_hc_results <- mpg_mini %>%
  mutate(cluster = factor(mpg_cut_3))
```

```
ggplot(data = mpg_hc_results) +
  geom_point(aes(x = displ, y = hwy,
                 color = cluster))
```



Using `factoextra::hcut()`

```
library(factoextra)
mpg_hcut <- hcut(mpg_mini, k = 3,
                 hc_method = "average",
                 hc_func = "hclust")
```

```
##
## Call:
## stats::hclust(d = x, method = hc_method)
##
## Cluster method   : average
## Distance         : euclidean
## Number of objects: 234
```

Computes hierarchical clustering (`hclust`, `agnes`, `diana`) and cut the tree into K clusters. It also accepts correlation based distance measure methods such as `"pearson"`, `"spearman"` and `"kendall"`.

```
fviz_dend(mpg_hcut, rect = TRUE)
```

Multiple groups using average linkage

```
hclusts <-
  tibble( # define number of groups to test
    clusters = 2:4)%>%
  mutate( # map method to hcut()
    myhclust = map(clusters,
                   ~factoextra::hcut(
                     mpg_mini,
                     hc_method = "average",
                     hc_func = "hclust",
                     k = .x ))
         )
```

We create individual plots and then "add" them using the `patchwork` package

```
q1 <- factoextra::fviz_dend(
  hclusts$myhclust[[1]],
  rect = TRUE)+
  labs(title = paste0("Creating ",
                      hclusts$clusters[1],
                      " groups"))
q2 <- factoextra::fviz_dend(
  hclusts$myhclust[[2]],
  rect = TRUE)+
  labs(title = paste0("Creating ",
                      hclusts$clusters[2],
                      " groups"))
q3 <- factoextra::fviz_dend(
  hclusts$myhclust[[3]],
  rect = TRUE)+
  labs(title = paste0("Creating ",
                      hclusts$clusters[3],
                      " groups"))
```

```r
library(patchwork)
(q1 + q2 + q3) +
  plot_layout(nrow = 1, widths = c(0.31, 0.31, 0.31))
```

Testing multiple linkage functions

```
hclusts <-
  tibble( # define the methods to try
    method = c("average",
               "single",
               "complete"))%>%
  mutate( # map method to hcut()
    myhclust = map(method,
                   ~factoextra::hcut(
                     mpg_mini,
                     hc_method = .x,
                     hc_func = "hclust",
                     k = 3 ))
         )
```

We create individual plots and then "add" them using the `patchwork` package

```
p1 <- factoextra::fviz_dend(
  hclusts$myhclust[[1]],
  rect = TRUE)+
  labs(title = paste0("Using ",
                      hclusts$method[1],
                      " linkage"))
p2 <- factoextra::fviz_dend(
  hclusts$myhclust[[2]],
  rect = TRUE)+
  labs(title = paste0("Using ",
                      hclusts$method[2],
                      " linkage"))
p3 <- factoextra::fviz_dend(
  hclusts$myhclust[[3]],
  rect = TRUE)+
  labs(title = paste0("Using ",
                      hclusts$method[3],
                      " linkage"))
```

```r
library(patchwork)
(p1 + p2 + p3) +
  plot_layout(nrow = 1, widths = c(0.31, 0.31, 0.31))
```

Some practical issues

- Should the observations or features first be **standardized** in some way? For instance, maybe the variables should be centered to have mean zero and scaled to have standard deviation one.
- In the case of hierarchical clustering:
  - What **dissimilarity** measure should be used?
  - What type of **linkage** should be used?
- **How many** clusters to choose? (in both K-means or hierarchical clustering). Difficult problem. No agreed-upon method. See Elements of Statistical Learning, chapter 13 for more details.

- Also check the dendextend package for creating dendrograms!