6.2 Cluster Analysis

Cluster analysis takes a different approach to data exploration. It works by partitioning heterogeneous observations into a set of distinct homogeneous groups, known as *clusters*, where observations share similar characteristics, with the goal of uncovering interesting subgroups in the dataset at hand. For Exam PA, we will learn two commonly used clustering methods, *k-means*[vi] *clustering* and *hierarchical clustering*, and look at their underlying rationale and practical implementations.

Note that although clustering itself is an unsupervised learning technique, the group assignments created as a result of clustering may serve as useful features for constructing a predictive (supervised learning) model. In the data exploration part of a PA exam project, you may be asked to perform clustering and use the output to create a new feature, e.g., Task 3 of the Hospital Readmissions sample project.

Example 6.2.1. (SOA Exam SRM Sample Question 32: Basics about cluster analysis) You are given a set of n observations, each with p features.

Determine which of the following statements is/are true with respect to clustering methods.

I. We can cluster the n observations on the basis of the p features in order to identify subgroups among the observations.

II. We can cluster the p features on the basis of the n observations in order to discover subgroups among the features.

III. Clustering is an unsupervised learning method and is often performed as part of an exploratory data analysis.

(A) None

(B) I and II only

(C) I and III only

(D) II and III only

(E) The correct answer is not given by (A), (B), (C), or (D).

Solution. I. True. This is the usual way we think of clustering, which assigns the n observations into a set of groups within which the observations have similar feature values.

II. True. Although not as common, clustering the p features on the basis of the n observations is technically identical to clustering the n observations on the basis of the p features as soon as we reverse the roles of the rows and columns in the data matrix $\mathbf{X}$. In other words, we apply cluster analysis to the transpose of $\mathbf{X}$.

III. True. Clustering is an unsupervised learning method because there is no target variable. It can be used in exploratory data analysis to learn about relationships between observations or features. **(Answer: (E))**

□

[vi] ISLR uses the term "K-means" (uppercase letter) while the PA modules use "k-means" (lowercase letter).

6.2.1 k-means Clustering

The aim of k-means clustering is to assign each observation in a dataset into one and only one of k clusters. The number of clusters, k, is specified upfront, after which the algorithm automatically searches for the best configuration of the k clusters.

k-means clustering algorithm. How do we define a "good" grouping? Intuitively, the k clusters are chosen such that the variation of the observations *inside each cluster* is as small as possible while the variation *between clusters* should be large. The smaller the *within-cluster variation*, the better the discrimination works. The k-means clustering algorithm formalizes this idea in search of the best similarity grouping. Without inundating you with mathematical formulas that play a minimal role in Exam PA, the k-means clustering algorithm can be described as follows.

- *Initialization:* Randomly select k points in the feature space. These k points will serve as the initial cluster centers.[vii]
- *Iteration:*

Step 1. Assign each observation to the cluster with the closest center in terms of Euclidean distance.[viii]

Step 2. Recalculate the center of each of the k clusters.

Step 3. Repeat Steps 1 and 2 until the cluster assignments no longer change.

Because the algorithm involves iteratively calculating the k means (more precisely, the k centers) of the clusters, such a clustering method is aptly called "k-means" (instead of "k-cluster") clustering.

Example 6.2.2. (Based on CAS Exam MAS-II Spring 2019 Question 42: Implementing k-means clustering algorithm by hand) You have decided to perform k-means clustering with $k = 2$ on the following dataset:

Observation	x_1	x_2
1	5	5
2	4	6
3	3	0
4	5	3
5	5	1
6	3	6
7	2	5

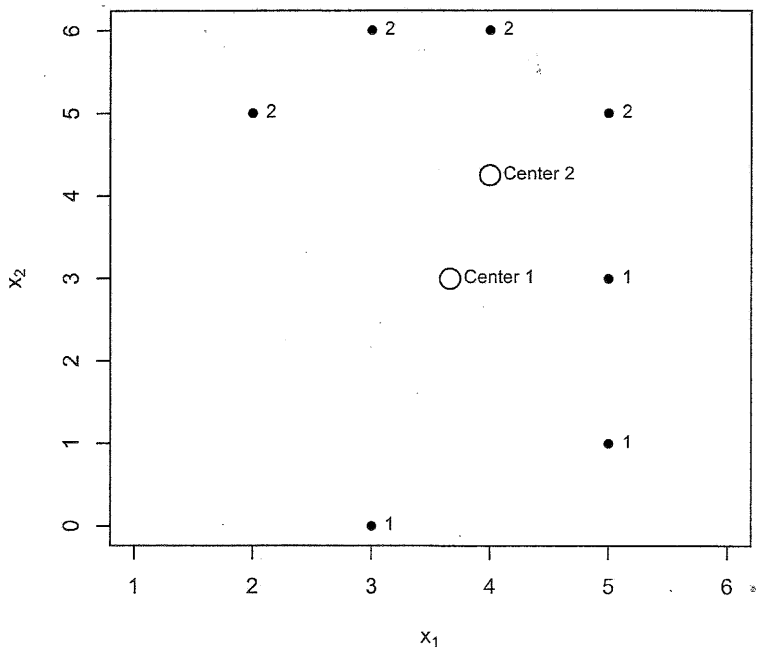
- The center of the initial cluster 1 is $x_1 = 3.667$, $x_2 = 3.000$
- The center of the initial cluster 2 is $x_1 = 4.00$, $x_2 = 4.250$

[vii] ISLR defines the initialization step a bit differently. Instead of randomly selecting k centers, ISLR randomly assigns a number from 1 to k to each observation.

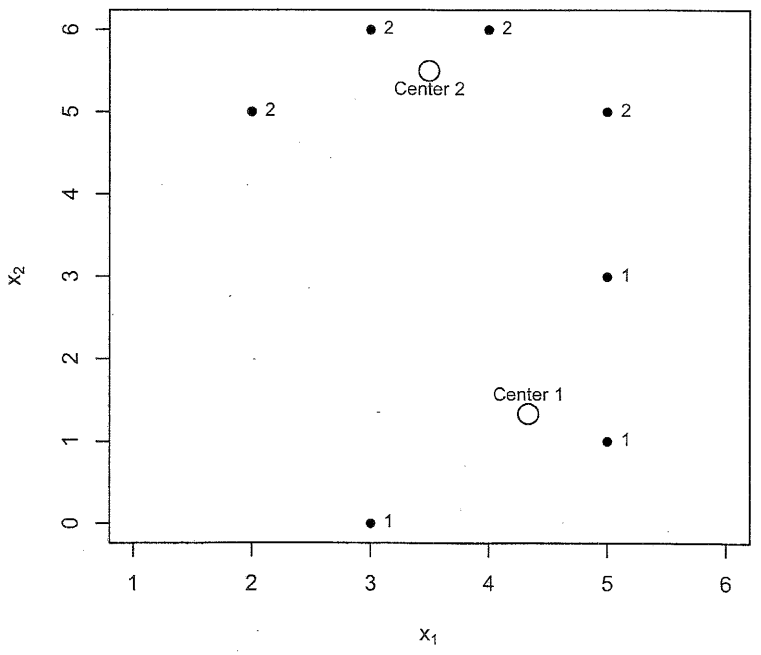
[viii] Note that Euclidean distance requires that the features be numeric and cannot deal with categorical features in their raw form. According to Slide 32 of PA e-learning Module 8, "[i]n this course, we will not be concerned with alternative distance measures and, hence, will only apply clustering to numeric observations."

Determine the centers of the two clusters after the first iteration of the k-means clustering algorithm.

Solution. Because this is an example with $p = 2$ features, we can visualize the dataset by sketching a two-dimensional scatterplot of the 7 observations together with the two initial cluster centers. Each observation is assigned to the cluster with the closest center.



We then recalculate the centers of the two clusters based on the new arrangements: $(\bar{x}_{11}, \bar{x}_{12}) = (13/3, 4/3)$ and $(\bar{x}_{21}, \bar{x}_{22}) = (3.5, 5.5)$.



In fact, at this stage, every observation has been assigned to the closest cluster, so the k-means clustering algorithm stops *(hurrah!)*. The centers of the two final clusters are $(\bar{x}_{11}, \bar{x}_{12}) = (13/3, 4/3)$ and $(\bar{x}_{21}, \bar{x}_{22}) = (3.5, 5.5)$. □

Random initial assignments. In each iteration of the k-means clustering algorithm, the within-cluster variation of the data is automatically reduced. At the completion of the algorithm, we are guaranteed to arrive at a *local* (but not necessarily global) optimum. The algorithm, however, relies on the initial cluster assignments, which are made randomly. A different set of initial cluster centers may give rise to a different final set of clusters and a different local optimum. For some datasets, the presence of outliers may also distort the cluster arrangements. If one of the initial centers is too close to the outlier, then only that outlier will get assigned to that center and form its own cluster well separated from the rest of the data.

To increase the chance of identifying a global optimum and getting a representative cluster grouping, it is always advisable to run the k-means algorithm many times (e.g., 20 to 50) with different initial cluster assignments. Then we select the best solution, i.e., the cluster assignments that result in the lowest within-cluster variation. The R function we will use to run a k-means cluster analysis has a built-in argument to allow for this, as we will see in Subsection 6.2.3.

Importance of standardization. Just like PCA, it is generally recommended that we standardize the features prior to running a cluster analysis. Without standardization, the features on a large scale will dominate the distance calculations and exert a disproportionate impact on the cluster arrangements. In contrast, when the features are standardized, the k-means clustering algorithm attaches an equal weight to all features when performing distance calculations, which is more desirable.

Choosing the number of clusters: The elbow method. Thus far we have performed k-means clustering with a pre-specified value of k. In practice, how should we go about selecting the value of k? The *elbow method* is a possible solution based on the fact that a good grouping is characterized by a small within-cluster variation, but a large between-cluster variation. Following this idea, we can make a plot of the ratio of the between-cluster variation to the total variation in the data against the value of k. When the proportion of variance explained has plateaued out (i.e., the increase that comes with an additional cluster starts to drop off), we are said to have reached an "elbow" and the corresponding value of k provides an appropriate number of clusters to segment the data.

6.2.2 Hierarchical Clustering

The k-means clustering algorithm we used in the preceding subsection requires that the number of clusters k be specified at the start. *Hierarchical clustering*, the second clustering method we will study, operates differently. Not only does it not require the choice of k in advance, the cluster groupings it produces can be displayed using a *dendrogram*,[ix] a tree-based visualization of the resulting "hierarchy" of clusters.

[ix] "Dendrogram" is commonly misspelled as "dendogram."

Inter-cluster dissimilarity. Hierarchical clustering consists of a series of fusions (or mergers) of observations. It starts with each individual observation as its own cluster and successively *fuses* the closest pair of clusters, one pair at a time. The process goes on iteratively until all clusters are eventually fused into a single cluster.

How do we measure the distance or dissimilarity between two clusters? The distance between two observations is straightforward: One can use the Euclidean distance as in k-means clustering. The notion of the distance between two *clusters*, one of which has more than one observation, is somewhat more ambiguous and relies on the *linkage* used. Three commonly used types of linkage functions are:

Linkage	The Inter-cluster Dissimilarity Is...
Complete	The *maximal* pairwise distance between observations in the two clusters (look at the furthest pair of observations)
Single	The *minimal* pairwise distance between observations in the two clusters (look at the closest pair of observations)
Average	The *average* of all pairwise distances between observations in the two clusters

In general, average and complete linkage are preferred over single linkage as they tend to result in more *balanced* (i.e., more evenly sized) clusters.

Example 6.2.3. (SOA Exam SRM Sample Question 1: Calculation of inter-cluster dissimilarity for a given linkage) You are given the following four pairs of observations:

$$x_1 = (-1, 0), \quad x_2 = (1, 1), \quad x_3 = (2, -1), \quad \text{and} \quad x_4 = (5, 10).$$

A hierarchical clustering algorithm is used with complete linkage and Euclidean distance.

Calculate the intercluster dissimilarity between $\{x_1, x_2\}$ and x_4.

(A) 2.2

(B) 3.2

(C) 9.9

(D) 10.8

(E) 11.7

Solution. We first calculate the Euclidean distance between all possible pairs of elements from each set. There are two pairs, (x_1, x_4) and (x_2, x_4):

$$\begin{aligned} d_{14} &= \sqrt{(-1-5)^2 + (0-10)^2} = \sqrt{136} = 11.6619, \\ d_{24} &= \sqrt{(1-5)^2 + (1-10)^2} = \sqrt{97} = 9.8489. \end{aligned}$$

With complete linkage, we take the maximum pairwise dissimilarities, which is $\boxed{11.6619}$, as the inter-cluster dissimilarity. **(Answer: (E))** □

Dendrogram. Now that we know how to measure inter-cluster dissimilarity using a given linkage, we are ready to implement the hierarchical clustering algorithm and view the hierarchy of clusters produced via a dendrogram, which not only shows which clusters are in close proximity to one another, but also shows, on the vertical axis, the thresholds of the inter-cluster dissimilarities at which fusions first occur. Clusters which are joined towards the bottom of the dendrogram are rather similar to one another while those which fuse towards the top are rather far apart. A very useful by-product of a dendrogram is that we can view the cluster compositions at once for every desired number of clusters, from 1 to n. In short, *one picture tells it all!*

Example 6.2.4. (CAS Exam MAS-II Fall 2018 Question 42: Performing hierarchical clustering for a small dataset with $p = 1$) You are provided the following data set with a single variable X.

i	X
1	9
2	15
3	4
4	2
5	18

A dendrogram is built from this data set using agglomerative hierarchical clustering with complete linkage and Euclidean distance as the dissimilarity measure.

Calculate the tree height at which observation $i = 1$ fuses.

(A) Less than 6

(B) 6

(C) 7

(D) 8

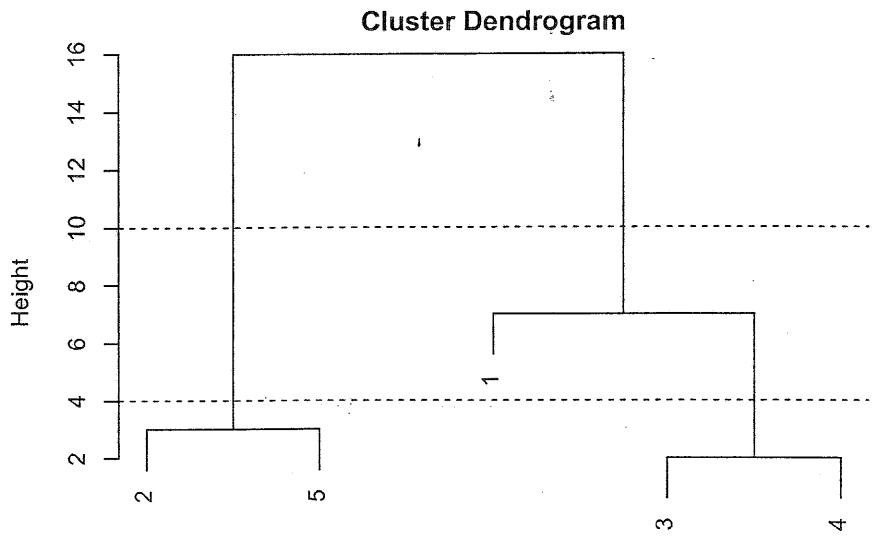
(E) At least 9

Solution. Here we have only $p = 1$ predictor, which allows us to display the dataset on a horizontal line:

- *First fuse:* The two most similar observations are Observations 3 and 4, with a dissimilarity measure of $4 - 2 = 2$. This forms the first merge $\{3, 4\}$.
- *Second fuse:* The next two most similar observations are Observations 2 and 5, with a dissimilarity measure of $18 - 15 = 3$. The second merge is $\{2, 5\}$.
- *Third fuse:* At this stage, $\{1\}$ lies between the two clusters $\{3, 4\}$ and $\{2, 5\}$. To see which cluster $\{1\}$ will be fused with, note that the dissimilarity measure between $\{1\}$ and $\{3, 4\}$ is $\max(9 - 2, 9 - 4) = 7$ and that between $\{1\}$ and $\{2, 5\}$ is $\max(18 - 9, 15 - 9) = 9$, which is higher. Therefore, $\{1\}$ is fused with $\{3, 4\}$, at which point the height of the dendrogram is the distance between $\{1\}$ and $\{3, 4\}$, which is $\boxed{7}$. **(Answer: (C))** □

Remark. (i) In the original version of the exam question, the CAS misspells "dendrogram" as "dendogram."

(ii) The dendrogram for this example is shown below. The last fuse happens at a height of $18 - 2 = 16$.



Once a dendrogram is constructed, we can determine the clusters by making a *horizontal cut* across the dendrogram. The resulting clusters formed are the distinct branches immediately below the cut. The lower the cut, the more clusters we create. In Example 6.2.4, for instance, drawing a horizontal cut at the height of 10 results in two clusters, $\{2,5\}$ and $\{1,3,4\}$, while cutting the dendrogram at a height of 4 produces three clusters, $\{2,5\},\{1\},\{3,4\}$. We can get any desired number of clusters from 1 (when there is no cut) to n (when the height of the cut is 0) by varying the height of the cut.

Note that the height of the cut plays the same role in determining the number of clusters as k in k-means clustering with one important difference: This height need not be specified in advance. Only after the dendrogram is constructed does the number of clusters have to be determined to find the composition of different clusters. If you want to change the number of clusters in the future, all you need to do is vary the height of the horizontal cut across the dendrogram, and no further reconstruction is needed. Compare this with k-means clustering, where changing k requires rerunning the whole algorithm for each value of k.

k-means vs. hierarchical clustering. Table 6.1 compares k-means clustering with hierarchical clustering.

Example 6.2.5. (SOA Exam SRM Sample Question 34: Basic facts about clustering methods – I) Determine which of the following statements is/are true about clustering methods:

Item	k-Means Clustering	Hierarchical Clustering
Is randomization needed?	Yes (Needed for determining initial cluster centers)	No
Is the number of clusters pre-specified?	Yes (k needs to be specified)	No
Are the clusters nested?	No	Yes (a hierarchy of clusters)

Table 6.1: Key differences between k-means clustering and hierarchical clustering.

I. If k is held constant, k-means clustering will always produce the same cluster assignments.

II. Given a linkage and a dissimilarity measure, hierarchical clustering will always produce the same cluster assignments for a specific number of clusters.

III. Given identical data sets, cutting a dendrogram to obtain five clusters produces the same cluster assignments as k-means clustering with $k = 5$.

(A) I only

(B) II only

(C) III only

(D) I, II, and III

(E) The correct answer is not given by (A), (B), (C), or (D).

Solution. I. False. k-means clustering requires random initial assignment of clusters. We may get different cluster assignments every time we run the algorithm even if the value of k stays the same.

II. True. Hierarchical clustering is deterministic, not relying on a random initial assignment.

III. False. The two methods differ in their approaches and hence may not yield the same cluster assignments. **(Answer: (B))**

□

Example 6.2.6. (SOA Exam SRM Sample Question 40: Basic facts about clustering methods – II) Determine which of the following statements about clustering is/are true.

I. Cutting a dendrogram at a lower height will not decrease the number of clusters.

II. k-means clustering requires plotting the data before determining the number of clusters.

III. For a given number of clusters, hierarchical clustering can sometimes yield less accurate results than K-means clustering.

(A) None

(B) I and II only

(C) I and III only

(D) II and III only

(E) The correct answer is not given by (A), (B), (C), or (D).

Solution. I. True. The number of clusters increases (resp. decreases) as the height of the cut decreases (resp. increases).

II. False. There is no need to plot the data to perform k-means clustering.

III. True. k-means does a fresh analysis for each value of k while for hierarchical clustering, the clusters obtained by cutting the dendrogram at a given height are always nested within the clusters obtained by cutting the dendrogram at any greater height. If such a nested structure does not exist in the population underlying the data, then hierarchical clustering may yield less accurate results than k-means clustering. **(Answer: (C))**

□

6.2.3 Simple Case Study

In this case study we will perform k-means clustering and hierarchical clustering on the classical `iris` flower dataset that comes with base R. After completing this case study, you should be able to:

- Run a k-means cluster analysis in R using the `kmeans()` function.
- Run a hierarchical cluster analysis in R using the `hclust()` function.
- Use the output of a cluster analysis to create useful features for prediction.

Data description. For 50 flowers from each of three species of iris, the `iris` dataset carries the measurements, in centimeters, of four variables: Sepal length, sepal width, petal length, and petal width. The three species are *Iris setosa*, *versicolor*, and *virginica*. For the purposes of this case study, we will only take sepal length and petal length as available features. Such a bivariate setting allows us to visualize the results of cluster analysis using two-dimensional scatterplots and gain insights into how cluster analysis works. (If you are interested, you may perform a PCA to reduce the four variables to two PCs, then do a cluster analysis.)

In CHUNK 1, we load the `iris` data, retain only `Sepal.Length`, `Petal.Length`, and `Species`, and print a summary.

```
# CHUNK 1
data(iris)
iris <- iris[, c(1, 3, 5)]
summary(iris)
```

```
##   Sepal.Length    Petal.Length        Species
##  Min.   :4.300   Min.   :1.000   setosa    :50
##  1st Qu.:5.100   1st Qu.:1.600   versicolor:50
##  Median :5.800   Median :4.350   virginica :50
##  Mean   :5.843   Mean   :3.758
##  3rd Qu.:6.400   3rd Qu.:5.100
##  Max.   :7.900   Max.   :6.900
```

To decide if scaling is necessary, we look at the standard deviation of `Sepal.Length` and `Petal.Length`, in the first part of CHUNK 2.

```
# CHUNK 2
apply(iris[, -3], 2, sd)

## Sepal.Length Petal.Length
##    0.8280661    1.7652982
```

The two standard deviations are not quite the same in comparison to the range of values of the two variables, so let's perform scaling before doing cluster analysis.

```
# CHUNK 2 (Cont.)
iris$Sepal.Length <- scale(iris$Sepal.Length)
iris$Petal.Length <- scale(iris$Petal.Length)

apply(iris[, -3], 2, sd)

## Sepal.Length Petal.Length
##            1            1
```

The two variables have unit standard deviation following standardization, as desired.

TASK 1: Use observations from a k-means cluster analysis to generate a new feature

Run a k-means cluster analysis on `Sepal.Length` and `Petal.Length`. Select the best number of clusters. Create a factor variable that could be used in place of these two variables.

Implementing a k-means cluster analysis. In R, k-means clustering is implemented by the `kmeans()` function, which takes the data matrix $\mathbf{X}$ on which cluster analysis is performed as the first argument, followed by the `centers` argument, which specifies k, the number of clusters used. The function also has an `nstart` argument, which controls the number of random selection of initial cluster centers and only the round with the best result will be reported. The default value of `nstart` is 1, but in general a larger value between 20 and 50 is recommended to improve the chance of identifying a global optimum.

In CHUNK 3, we run a k-means cluster analysis on `Sepal.Length` and `Petal.Length` with $k = 3$, which is arbitrarily selected for the time being. To save time, we will set `nstart` to 20 (feel free to change it to a larger value).

```
# CHUNK 3
# kmeans() uses random initial cluster centers
set.seed(1)

# Select the variables on which a k-means cluster analysis is run
cluster_vars <- iris[, c("Sepal.Length", "Petal.Length")]
km3 <- kmeans(cluster_vars, centers = 3, nstart = 20)
km3

## K-means clustering with 3 clusters of sizes 43, 53, 54
##
## Cluster means:
##   Sepal.Length Petal.Length
## 1    1.2255135    1.0250063
## 2    0.0365328    0.4160503
## 3   -1.0117281   -1.2245544
##
## Clustering vector:
##   [1] 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 3 3 3 3 3 3 3
##  [36] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 2 1 2 1 2 2 3 1 2 3 2 2 2 2 1 2 2 2 2
##  [71] 2 2 2 2 2 1 1 1 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 3 2 2 2 2 3 2 1 2 1 1 1
## [106] 1 2 1 1 1 1 1 1 2 2 1 1 1 1 2 1 2 1 2 1 1 2 2 1 1 1 1 1 2 2 1 1 1 2 1
## [141] 1 1 2 1 1 1 2 1 2 2
##
## Within cluster sum of squares by cluster:
## [1] 17.54315 11.87468 13.32996
##  (between_SS / total_SS =  85.7 %)
##
## Available components:
```

```
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

As the output indicates, we have partitioned the 150 observations in the `iris` data into 3 groups of sizes 43, 53, and 54, with widely different cluster centers. The "`Clustering vector`" contains the labels of the groups (`"1"`, `"2"`, or `"3"`) each observation is classified to. To access the results of the k-means cluster analysis for further use, note that a `kmeans` object is a list with various components listed in the last part of the output of CHUNK 3. In Exam PA, the most useful components are `cluster`, `betweenss`, and `totss`, which store the group assignments (same as "`Clustering vector`"), the between-cluster sum of squares, and the total sum of squares, respectively.

In CHUNK 4, we first extract the (numeric) vector of group assignments and convert it to a factor, then make two scatterplots for `Sepal.Length` and `Petal.Length`, one colored by the group assignments and one colored by `Species` (which can be viewed as the true group assignments), to visualize the results of the k-means cluster analysis; see Figure 6.2.1. It is clear that the group assignments returned by the cluster analysis match the true assignments rather closely, with the observations in the bottom left corner, middle part, and top right corner being predominantly `setosa`, `versicolor`, and `virginica`, respectively. For the purposes of data exploration and prediction, it seems advisable to use the group assignments as a new feature to summarize the `iris` data and to predict which species a flower belongs to.

Choosing the best value of k. In CHUNK 4 above, we "cheated" (☻!) and set $k = 3$ because behind the scenes there are three groups in the `iris` data indicated by the `Species` variable. In practice, we have to select the value of k before running a k-means cluster analysis, one way being the elbow method described in Subsection 6.2.1. In CHUNK 5, we manually perform a k-means cluster analysis with one to ten clusters, compute the ratio of between-cluster sum of squares and total sum of squares for each round, and create an elbow plot; see Figure 6.2.2. (There are more efficient ways requiring more advanced R programming to produce this graph, but here we follow the code in the PA e-learning modules and the Hospital Readmissions sample project.)

```
# CHUNK 4
iris$group <- as.factor(km3$cluster)

library(ggplot2)
library(gridExtra)
p1 <- ggplot(iris, aes(x = Sepal.Length, y = Petal.Length, col = group)) +
  geom_point(size = 2)

p2 <- ggplot(iris, aes(x = Sepal.Length, y = Petal.Length, col = Species)) +
  geom_point(size = 2)

grid.arrange(p1, p2, ncol = 2)
```

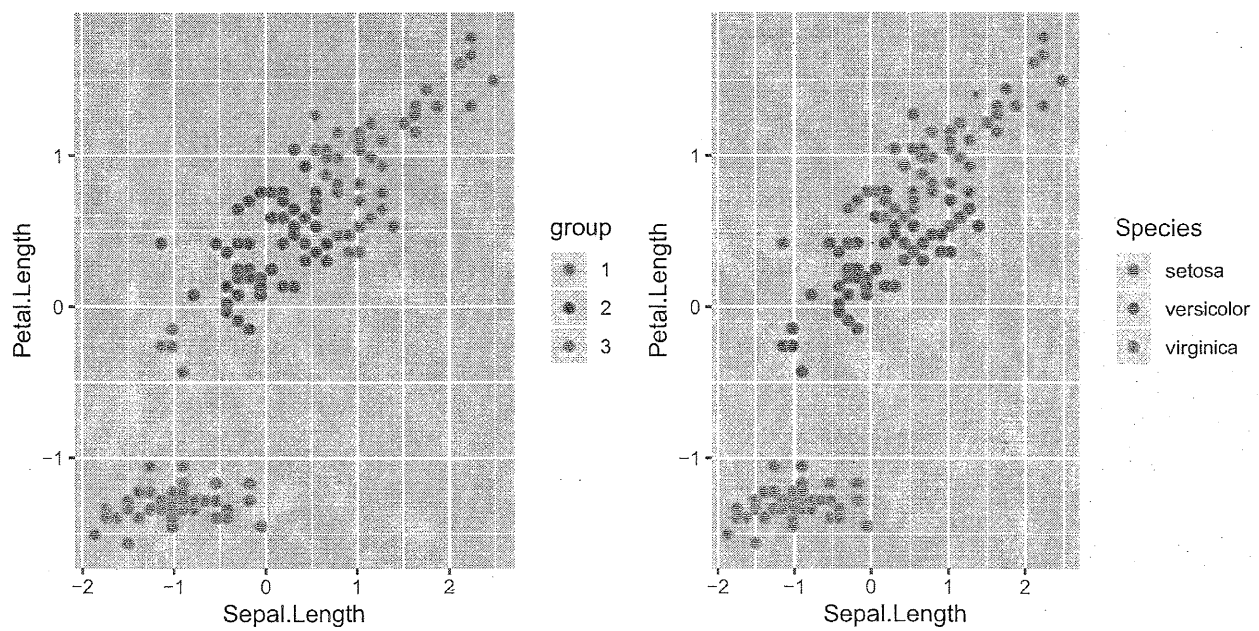


Figure 6.2.1: Visualizing the results of k-means cluster analysis with $k = 3$ for the `iris` data.

```
# CHUNK 5
km1 <- kmeans(cluster_vars, centers = 1, nstart = 20)
km2 <- kmeans(cluster_vars, centers = 2, nstart = 20)
km3 <- kmeans(cluster_vars, centers = 3, nstart = 20)
km4 <- kmeans(cluster_vars, centers = 4, nstart = 20)
km5 <- kmeans(cluster_vars, centers = 5, nstart = 20)
km6 <- kmeans(cluster_vars, centers = 6, nstart = 20)
km7 <- kmeans(cluster_vars, centers = 7, nstart = 20)
km8 <- kmeans(cluster_vars, centers = 8, nstart = 20)
km9 <- kmeans(cluster_vars, centers = 9, nstart = 20)
km10 <- kmeans(cluster_vars, centers = 10, nstart = 20)

var.exp <- data.frame(k = 1:10,
                      bss_tss = c(km1$betweenss/km1$totss,
                                  km2$betweenss/km2$totss,
                                  km3$betweenss/km3$totss,
                                  km4$betweenss/km4$totss,
                                  km5$betweenss/km5$totss,
                                  km6$betweenss/km6$totss,
                                  km7$betweenss/km7$totss,
                                  km8$betweenss/km8$totss,
                                  km9$betweenss/km9$totss,
                                  km10$betweenss/km10$totss))
```

Based on Figure 6.2.2, the elbow is clearly at $k = 3$, beyond which the gains in the between-cluster sum of squares appear to be minimal, so a reasonable choice for k is 3. This is not surprising given that there really are three groups in the `iris` data. The group assignments based on a k-means cluster analysis with $k = 3$ were created in CHUNK 4 and they will serve as useful replacements for `Sepal.Length` and `Petal.Length`, which should be dropped (just like PCA!). The removal is done in CHUNK 6.

```
# CHUNK 5 (Cont.)
ggplot(var.exp, aes(x = k, y = bss_tss)) +
  geom_point() +
  geom_line() +
  ggtitle("Elbow plot")
```

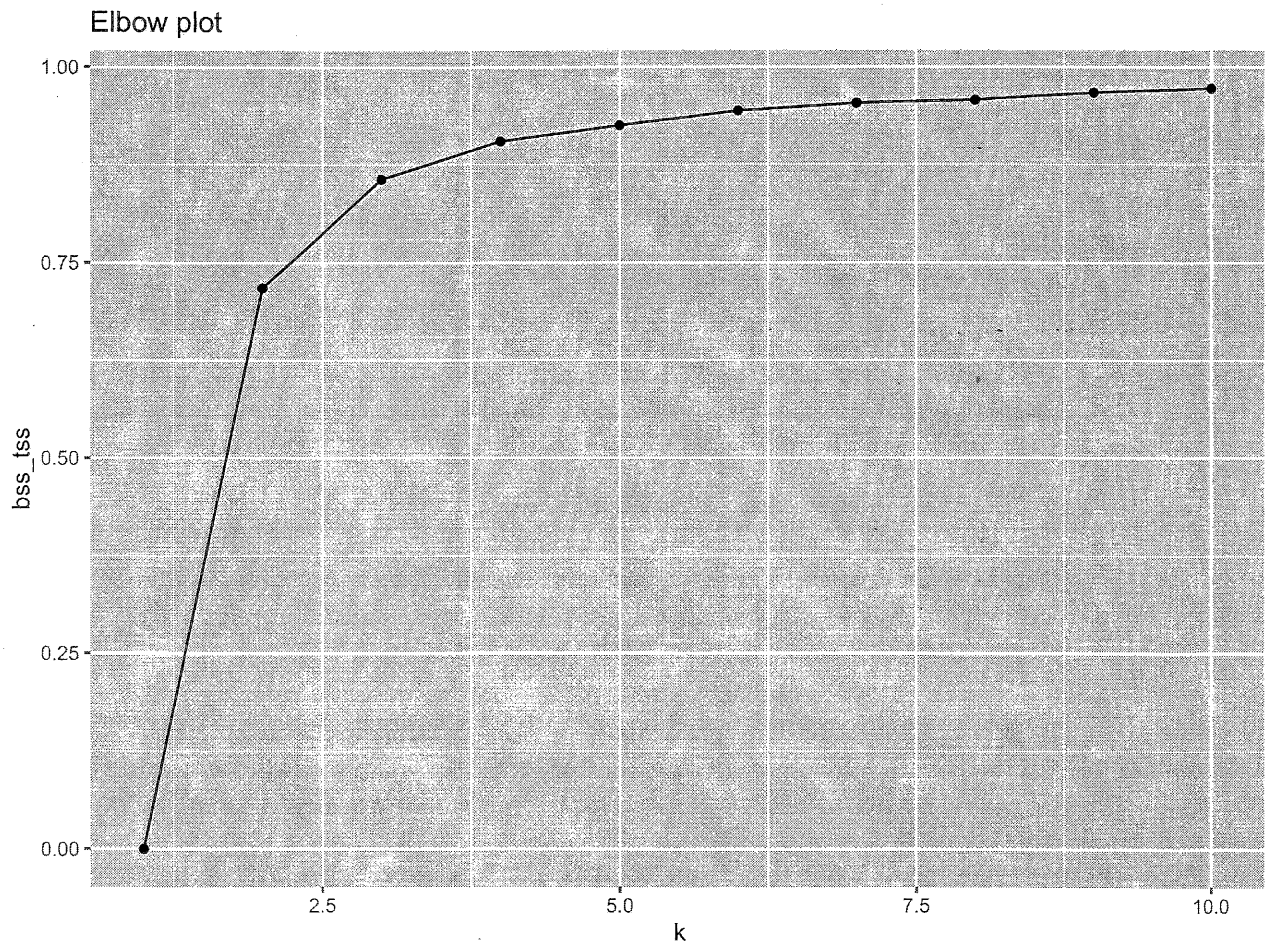


Figure 6.2.2: An elbow plot for the k-means cluster analysis run on the `iris` data.

```
# CHUNK 6
iris$Sepal.Length <- NULL
iris$Petal.Length <- NULL
head(iris)

##   Species group
## 1  setosa     3
## 2  setosa     3
## 3  setosa     3
## 4  setosa     3
## 5  setosa     3
## 6  setosa     3

tail(iris)

##        Species group
## 145 virginica     1
## 146 virginica     1
## 147 virginica     2
## 148 virginica     1
## 149 virginica     2
## 150 virginica     2
```

TASK 2: Implementing a hierarchical cluster analysis

Run a hierarchical cluster analysis on `Sepal.Length` and `Petal.Length`.

The PA e-learning modules have a very light discussion on hierarchical cluster analysis and they do not cover any quantitative tools for determining the number of clusters. For completeness, we will still run a hierarchical cluster analysis, also on the `iris` data.

Implementing a hierarchical cluster analysis. In R, a hierarchical cluster analysis can be implemented by the `hclust()` function. This function does not take a data frame carrying the variables of interest as an argument; instead, it requires an $n \times n$ numeric matrix carrying the inter-observation Euclidean distances as an input.

In CHUNK 7, we first reload the `iris` data, scale `Sepal.Length` and `Petal.Length`, and use the `dist()` function to compute the inter-observation Euclidean distance matrix. Of dimension 150×150, this matrix captures the distance (with respect to `Sepal.Length` and `Petal.Length`) for each of the $\binom{150}{2}$ pairs of observations in the `iris` data.

```
# CHUNK 7
# Reload the iris data
data(iris)
iris <- iris[, c(1, 3, 5)]
iris$Sepal.Length <- scale(iris$Sepal.Length)
iris$Petal.Length <- scale(iris$Petal.Length)
```

```
# Calculate the distance matrix
dist_matrix <- dist(iris[, c("Sepal.Length", "Petal.Length")])
```

In CHUNK 8, we use the distance matrix to run three rounds of hierarchical cluster analysis, with complete, single, and average linkage. By default, the `hclust()` function uses complete linkage, although the type of linkage can be changed by the `method` argument. Then the `plot()` function is applied to plot the three dendrograms; see Figure 6.2.3. We can see that the complete and average linkage yield more balanced and aesthetically attractive clusters in the sense that a given horizontal cut in the dendrogram will tend to produce clusters of a similar number of observations. In what follows, we will use the complete linkage.

Visualizing the results of hierarchical clustering. To determine the cluster labels for each observation, we can use the `cutree()` (meaning to cut a tree) function, which takes as arguments an `hclust` object and an integer specifying the number of clusters desired.[x] In CHUNK 9, we first define a function called `plot_cluster_slice` (also used in the PA e-learning modules) to create a scatterplot for `Sepal.Length` and `Petal.Length` colored by the cluster groups, taking an `hclust` object and the number of clusters as arguments, then apply the function to visualize the cluster groups when two, three, four, and five clusters are desired; see Figure 6.2.4. Comparing Figures 6.2.1 and 6.2.4, we can see that the three clusters are rather similarly distributed.

Unlike k-means cluster analysis, hierarchical cluster analysis has no systematic methods (at least not covered in the PA e-learning modules) for determining the number of clusters used, or equivalently, for determining where we should cut the dendrogram, and there is not much we can do without further information about the dataset at hand.

[x] Alternatively, you can specify the height at which the dendrogram should be cut.

```
# CHUNK 8
hc.complete <- hclust(dist_matrix)
hc.single <- hclust(dist_matrix, method = "single")
hc.average <- hclust(dist_matrix, method = "average")

plot(hc.complete, cex = 0.3)
plot(hc.single, cex = 0.3)
plot(hc.average, cex = 0.3)
```

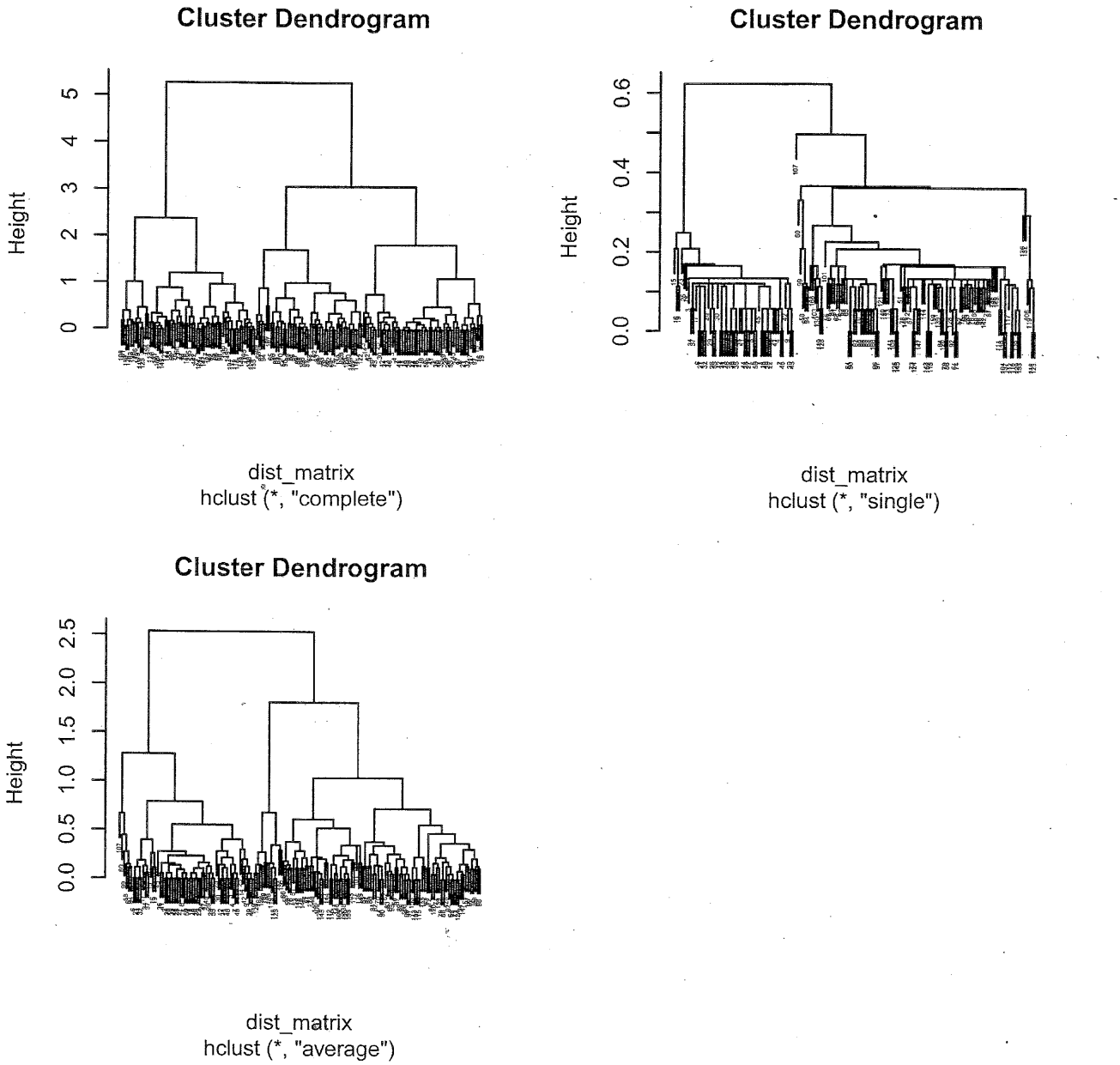


Figure 6.2.3: The dendrograms for hierarchical clustering with complete, single, and average linkage applied to the `iris` data.

```
# CHUNK 9
# Simple function to create a plot given a data frame, an hclust ob-
ject, and number of clusters
plot_cluster_slice <- function(df, hc, numclusters) {
  df$clusters <- as.factor(cutree(hc, numclusters))
  ggplot(data = df, aes(x = Sepal.Length, y = Petal.Length, col = clusters)) +
    geom_point()
}

plot_cluster_slice(iris, hc.complete, 2)
plot_cluster_slice(iris, hc.complete, 3)
plot_cluster_slice(iris, hc.complete, 4)
plot_cluster_slice(iris, hc.complete, 5)
```

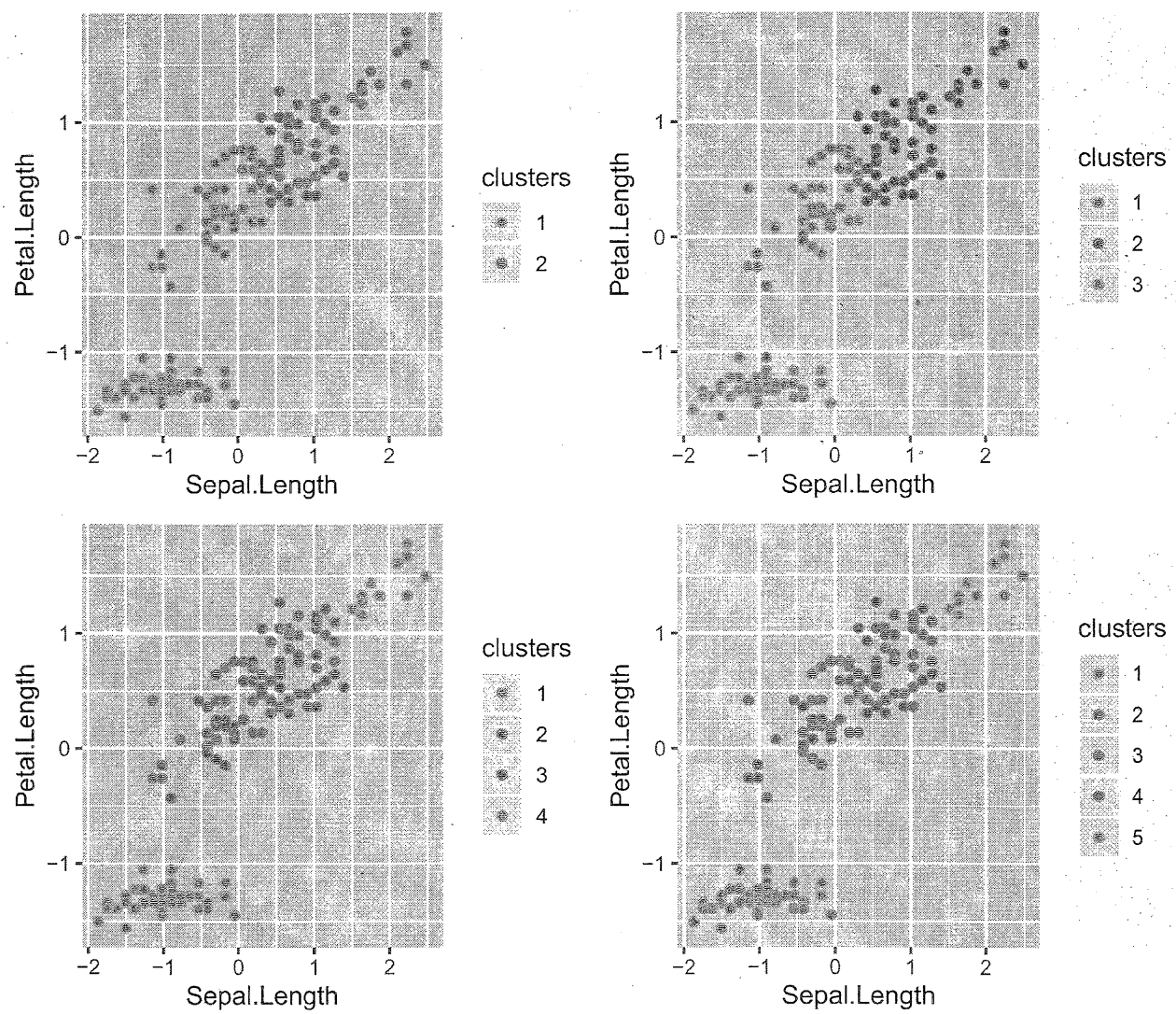


Figure 6.2.4: Results for hierarchical cluster analysis with complete linkage for two, three, four, and five clusters for the `iris` data.