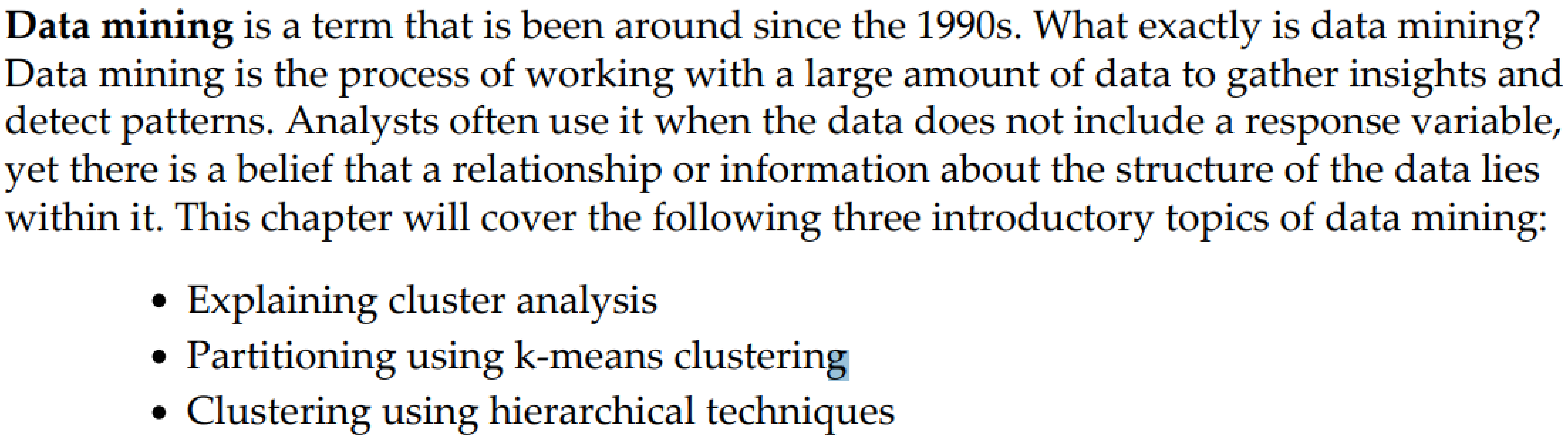
STQD6134: Business Analytics

Data Mining with Cluster Analysis

Introduction



Data mining is a term that is been around since the 1990s. What exactly is data mining? Data mining is the process of working with a large amount of data to gather insights and detect patterns. Analysts often use it when the data does not include a response variable, yet there is a belief that a relationship or information about the structure of the data lies within it. This chapter will cover the following three introductory topics of data mining:

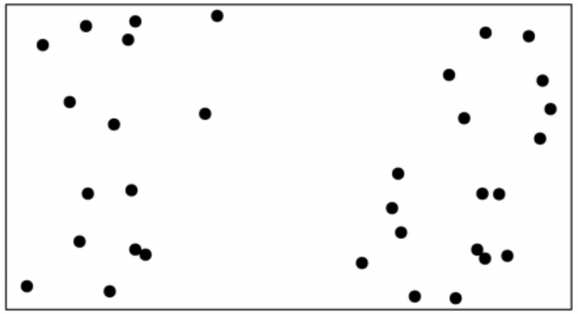
- Explaining cluster analysis
- Partitioning using k-means clustering
- Clustering using hierarchical techniques

Explaining cluster analysis

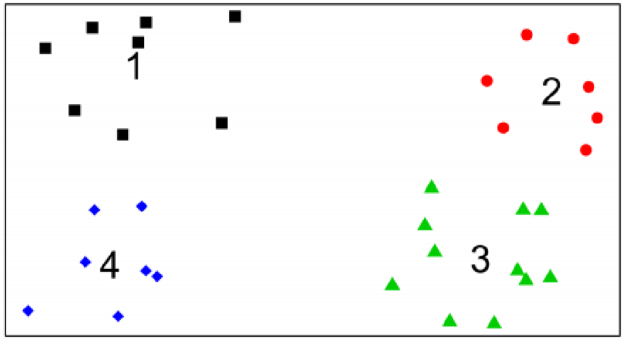
According to Han (2011), *Clustering is the process of grouping a set of data objects into multiple groups or clusters so that objects within a cluster have high similarity, but are very dissimilar to objects in other clusters (p. 443).*

Supervised learning is for cases when you have a response variable. You will use data features to predict that response. Unsupervised learning handles cases where you do not have a response variable. It provides a way to look at the people in that room (the observations) and determine ways to explain how they group together based on their features. Clusters form based on the similarity of observations compared to one another-the mean (central point).

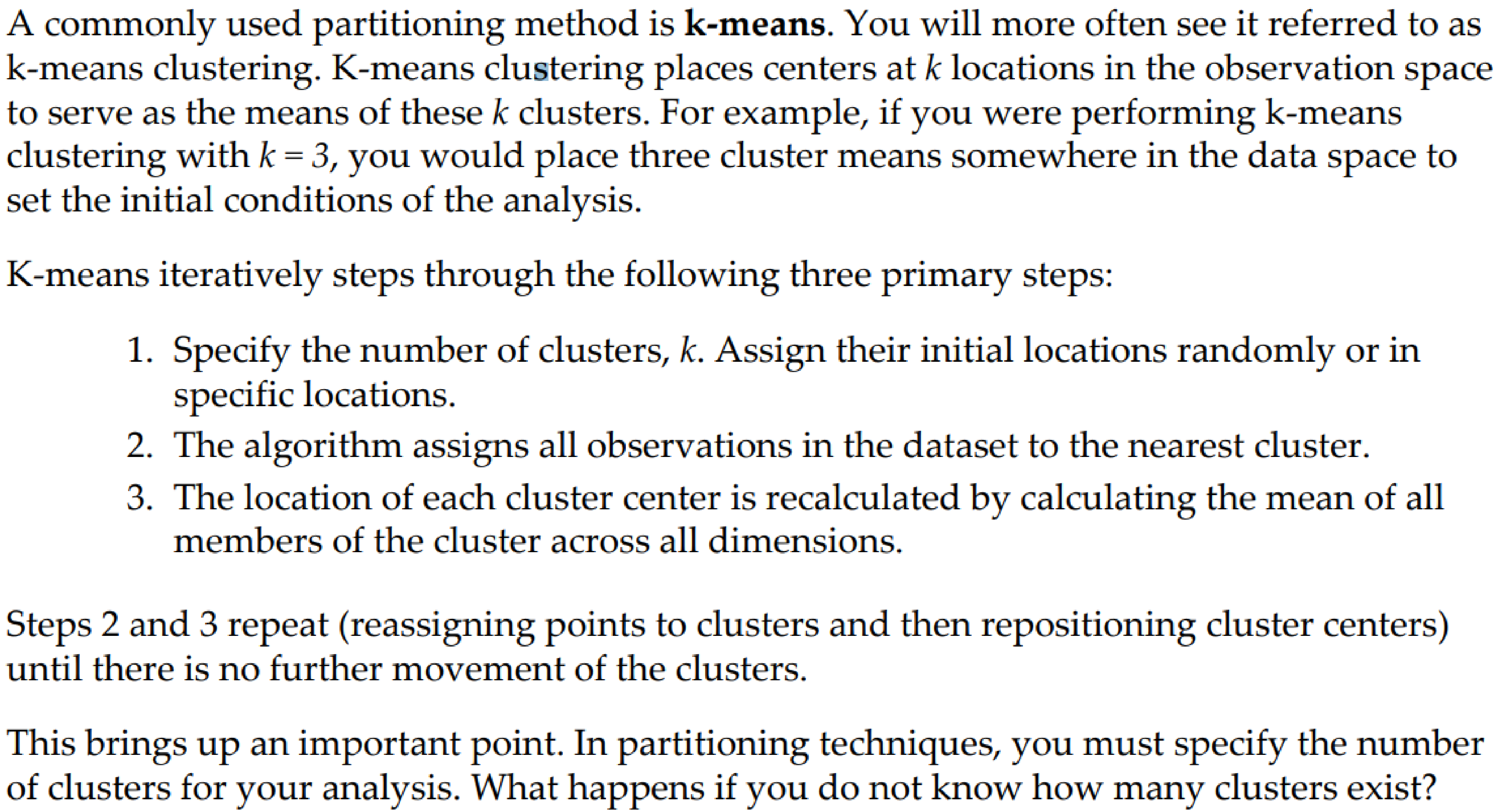
Dinner Party Guests



Guest Clusters



K- means clustering



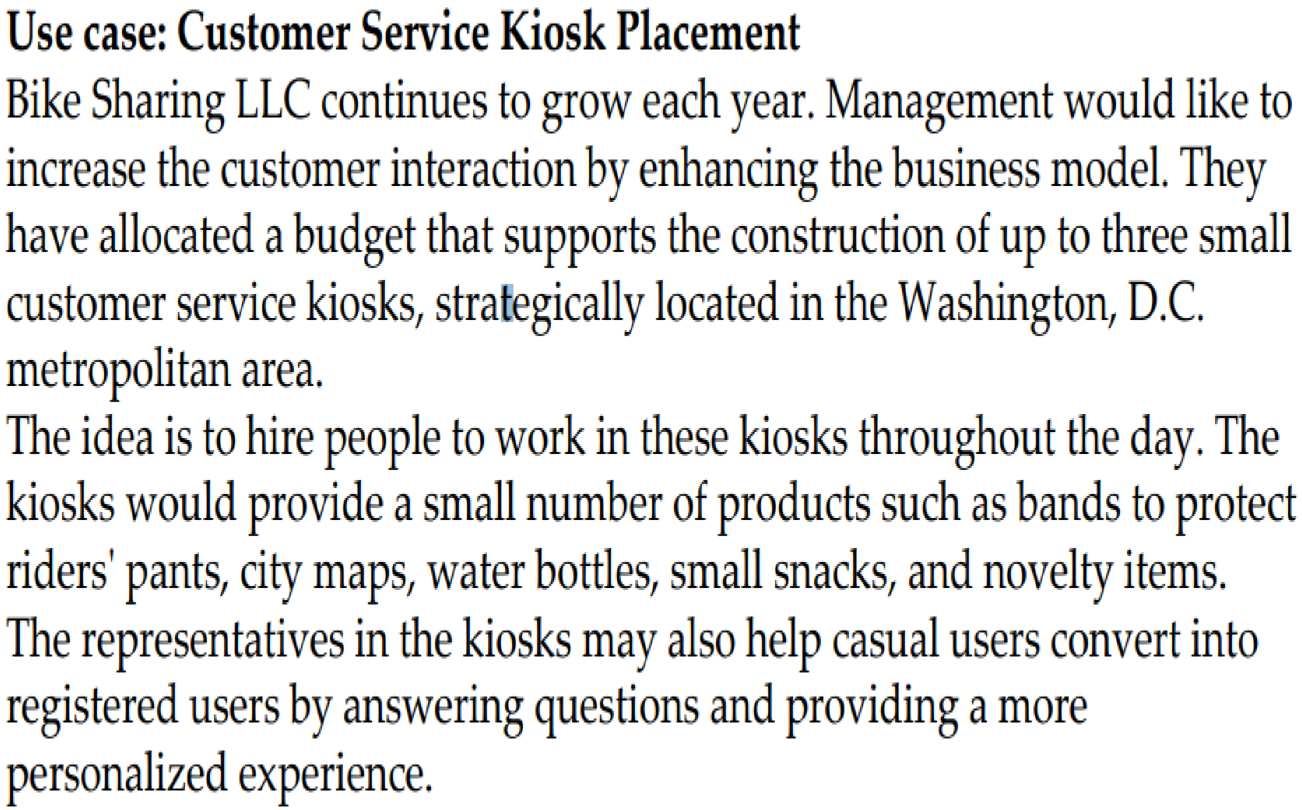
A commonly used partitioning method is **k-means**. You will more often see it referred to as k-means clustering. K-means clustering places centers at *k* locations in the observation space to serve as the means of these *k* clusters. For example, if you were performing k-means clustering with $k = 3$, you would place three cluster means somewhere in the data space to set the initial conditions of the analysis.

K-means iteratively steps through the following three primary steps:

1. Specify the number of clusters, *k*. Assign their initial locations randomly or in specific locations.
2. The algorithm assigns all observations in the dataset to the nearest cluster.
3. The location of each cluster center is recalculated by calculating the mean of all members of the cluster across all dimensions.

Steps 2 and 3 repeat (reassigning points to clusters and then repositioning cluster centers) until there is no further movement of the clusters.

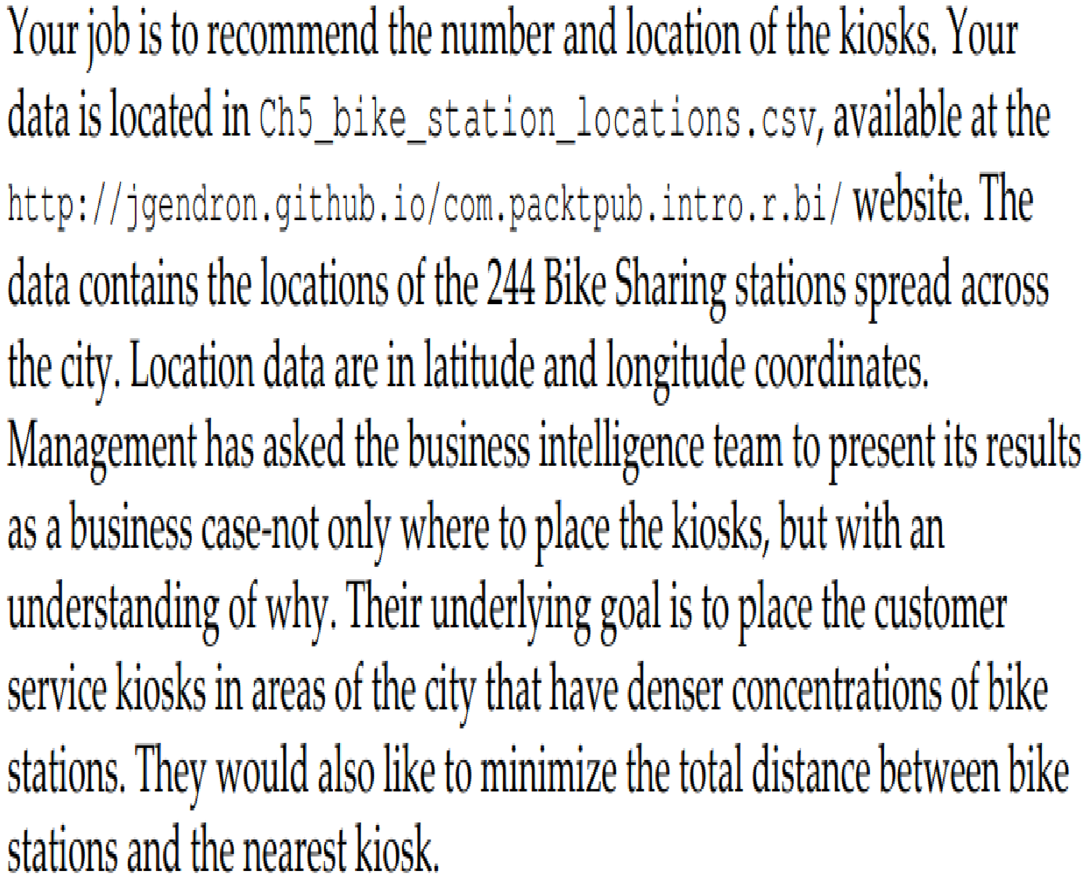
This brings up an important point. In partitioning techniques, you must specify the number of clusters for your analysis. What happens if you do not know how many clusters exist?



Use case: Customer Service Kiosk Placement

Bike Sharing LLC continues to grow each year. Management would like to increase the customer interaction by enhancing the business model. They have allocated a budget that supports the construction of up to three small customer service kiosks, strategically located in the Washington, D.C. metropolitan area.

The idea is to hire people to work in these kiosks throughout the day. The kiosks would provide a small number of products such as bands to protect riders' pants, city maps, water bottles, small snacks, and novelty items. The representatives in the kiosks may also help casual users convert into registered users by answering questions and providing a more personalized experience.



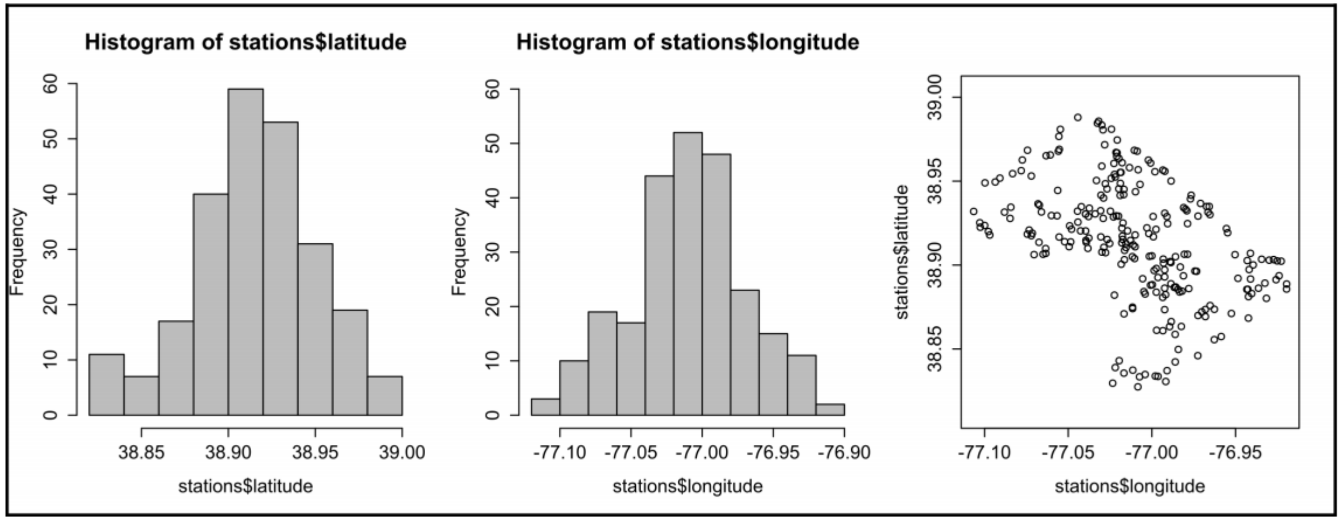
Your job is to recommend the number and location of the kiosks. Your data is located in `Ch5_bike_station_locations.csv`, available at the `http://jgendron.github.io/com.packtpub.intro.r.bi/` website. The data contains the locations of the 244 Bike Sharing stations spread across the city. Location data are in latitude and longitude coordinates. Management has asked the business intelligence team to present its results as a business case-not only where to place the kiosks, but with an understanding of why. Their underlying goal is to place the customer service kiosks in areas of the city that have denser concentrations of bike stations. They would also like to minimize the total distance between bike stations and the nearest kiosk.

Exploring the data

```
stations <- read.csv("./data/Ch5_bike_station_locations.csv")
summary(stations)
```

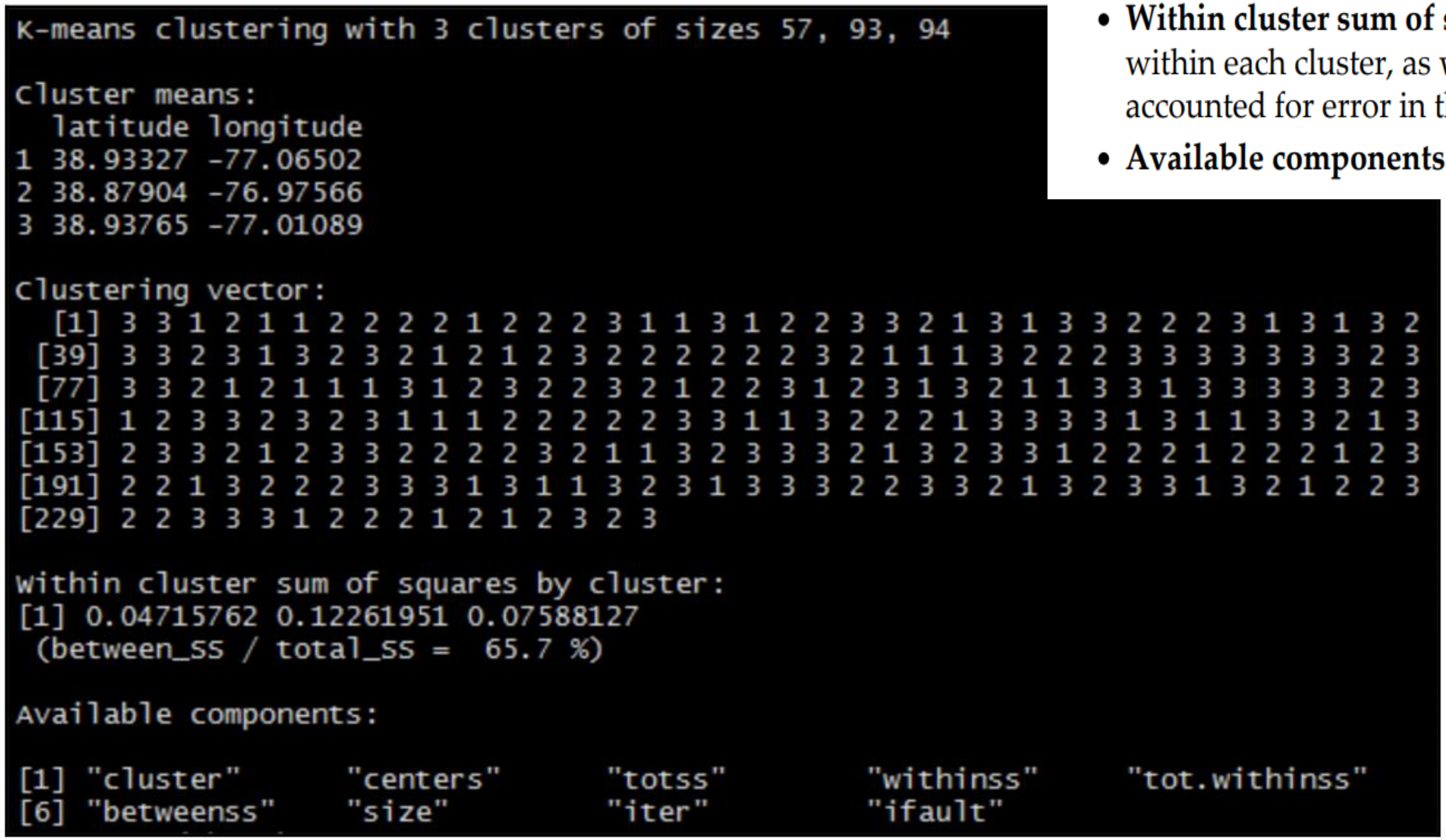
```
    latitude        longitude     
 Min.   :38.83   Min.   :-77.11  
 1st Qu.:38.89   1st Qu.:-77.03  
 Median :38.92   Median :-77.01  
 Mean   :38.91   Mean   :-77.01  
 3rd Qu.:38.94   3rd Qu.:-76.99  
 Max.   :38.99   Max.   :-76.92  
```

```
hist(stations$latitude, col = 'gray')
hist(stations$longitude, ylim = c(0, 60), col = 'gray')
plot(stations$longitude, stations$latitude, asp = 1)
```



Running the kmeans function

```
set.seed(123)

two <- kmeans(stations, 2)
three <- kmeans(stations, 3)
```



```
K-means clustering with 3 clusters of sizes 57, 93, 94

Cluster means:
  latitude longitude
1 38.93327 -77.06502
2 38.87904 -76.97566
3 38.93765 -77.01089

Clustering vector:
  [1] 3 3 1 2 1 1 2 2 2 2 1 2 2 2 3 1 1 3 1 2 2 3 3 2 1 3 1 3 3 2 2 2 3 1 3 1 3 2
 [39] 3 3 2 3 1 3 2 3 2 1 2 1 2 3 2 2 2 2 2 2 3 2 1 1 1 3 2 2 2 3 3 3 3 3 3 3 2 3
 [77] 3 3 2 1 2 1 1 1 3 1 2 3 2 2 3 2 1 2 2 3 1 2 3 1 3 2 1 1 3 3 1 3 3 3 3 3 2 3
[115] 1 2 3 3 2 3 2 3 1 1 1 2 2 2 2 2 3 3 1 1 3 2 2 2 1 3 3 3 3 1 3 1 1 3 3 2 1 3
[153] 2 3 3 2 1 2 3 3 2 2 2 2 3 2 1 1 3 2 3 3 3 2 1 3 2 3 3 1 2 2 2 1 2 2 2 1 2 3
[191] 2 2 1 3 2 2 2 3 3 3 1 3 1 1 3 2 3 1 3 3 3 2 2 3 3 2 1 3 2 3 3 1 3 2 1 2 2 3
[229] 2 2 3 3 3 1 2 2 2 1 2 1 2 3 2 3

Within cluster sum of squares by cluster:
[1] 0.04715762 0.12261951 0.07588127
 (between_SS / total_SS =  65.7 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

- This is a three-cluster model. The size of each cluster is **57**, **93**, and **94**.
- **Cluster means** provides the location of the center of each cluster.
- **Clustering vector** shows the cluster to which each data point is assigned.
- **Within cluster sum of squares by cluster** provides the sum of square error within each cluster, as well as a percentage showing how well the model accounted for error in the model. This model explains **65.7%** of the error.
- **Available components** shows all the items you can access for computation.

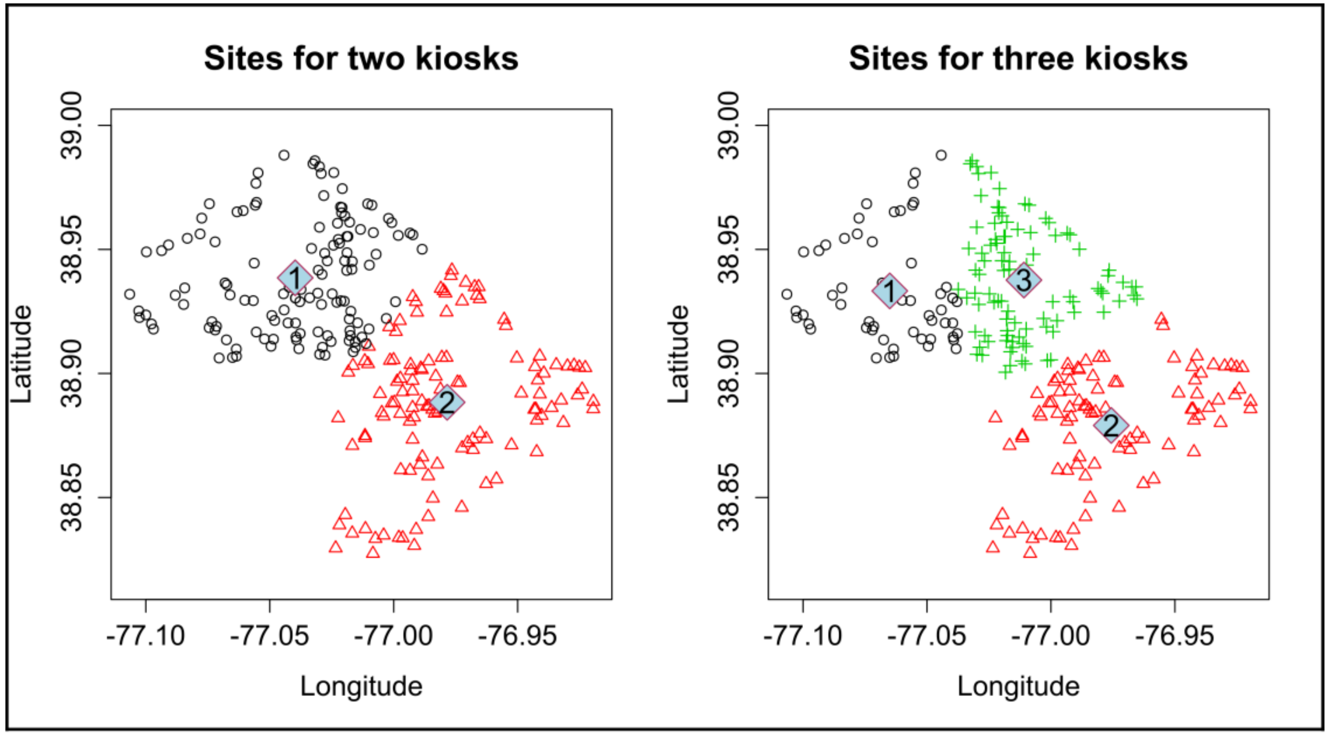
```
clus <- cbind(stations, clus2 = two$cluster,
                        clus3 = three$cluster)
head(clus)
```

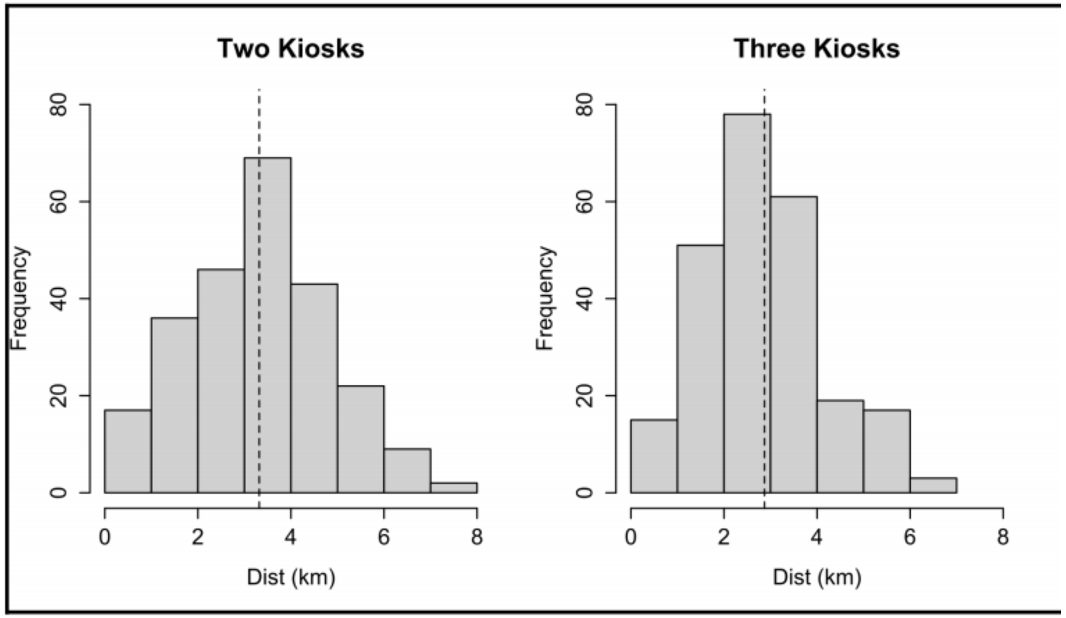
The output is shown here:

```
  latitude longitude clus2 clus3
1 38.95659 -76.99344     1     3
2 38.90522 -77.00150     2     3
3 38.98086 -77.05472     1     1
4 38.90293 -76.92991     2     2
5 38.94950 -77.09362     1     1
6 38.92780 -77.08474     1     1
```

Developing a business case

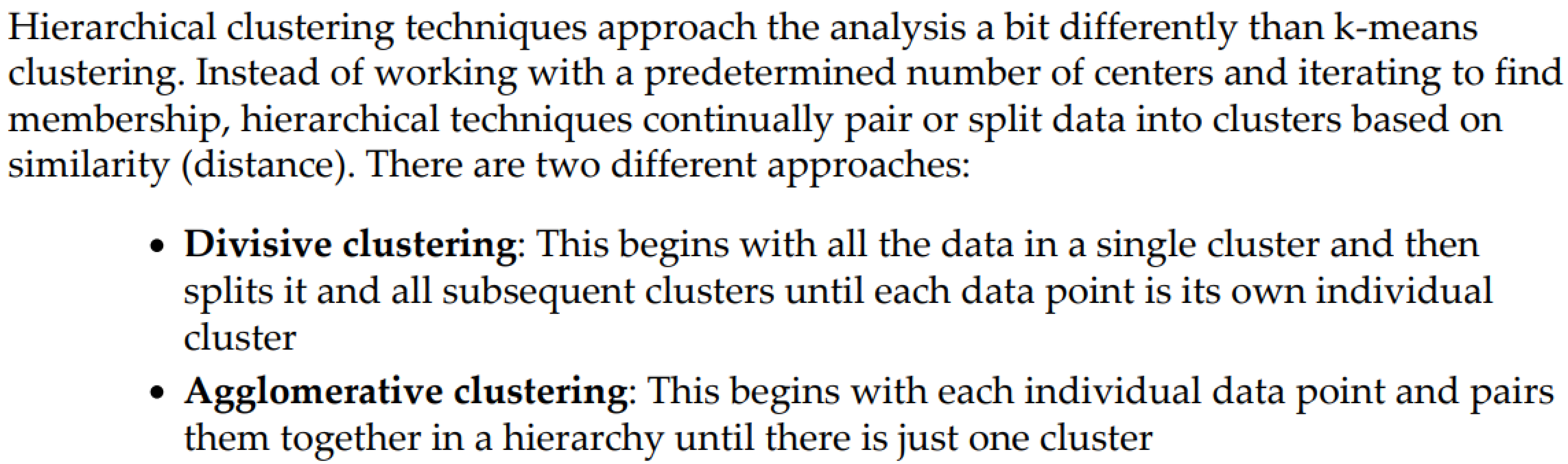
```
plot(clus$longitude, clus$latitude, col = two$cluster, asp = 1,
     pch = two$cluster, main = "Sites for two kiosks",
     xlab = "Longitude",  ylab = "Latitude")
points(two$centers[ ,2], two$centers[ ,1], pch = 23,
       col = 'maroon', bg = 'lightblue', cex = 3)
text(two$centers[ ,2], two$centers[ ,1], cex = 1.1,
     col = 'black', attributes(two$centers)$dimnames[[1]])

plot(clus$longitude, clus$latitude, col = three$cluster, asp = 1,
     pch = three$cluster, main = "Sites for three kiosks",
     xlab = "Longitude",  ylab = "Latitude")
points(three$centers[ ,2], three$centers[ ,1],
       pch = 23, col = 'maroon', bg = 'lightblue', cex = 3)
text(three$centers[ ,2], three$centers[ ,1], cex = 1.1,
     col = 'black', attributes(three$centers)$dimnames[[1]])
```





measure	**2-cluster**	**3-cluster**
mean distance	3.32 km	2.87 km
maximum distance	7.61 km	6.88 km

Clustering using hierarchical techniques



Hierarchical clustering techniques approach the analysis a bit differently than k-means clustering. Instead of working with a predetermined number of centers and iterating to find membership, hierarchical techniques continually pair or split data into clusters based on similarity (distance). There are two different approaches:

- **Divisive clustering**: This begins with all the data in a single cluster and then splits it and all subsequent clusters until each data point is its own individual cluster
- **Agglomerative clustering**: This begins with each individual data point and pairs them together in a hierarchy until there is just one cluster

Use case: Targeted Marketing Segments

Nice job on the kiosk project. Word gets around and the marketing group wants you to help them define customer segments and target advertising campaigns. Bike Sharing ridership has increased over 65% from 1,221,270 uses in 2011 to 2,028,912 uses in 2012, representing over 3.2 million bike shares.

You have been given the age and income data for over 8,000 existing customers. This `Ch5_age_income_data.csv` data is available on the book's website at `http://jgendron.github.io/com.packtpub.intro.r.bi/`.

Your job is to use your skills to inspect and explore the data. Then, you will create various hierarchical clustering models and evaluate them to determine the number and characteristics of the customer segments. As you work through the problem, observe the number of analytic decisions you make-all affecting the final product. This will require you to use your analytic skills and business sense, blending science and art.

To prepare your analysis for the marketing group, do the following exercise:

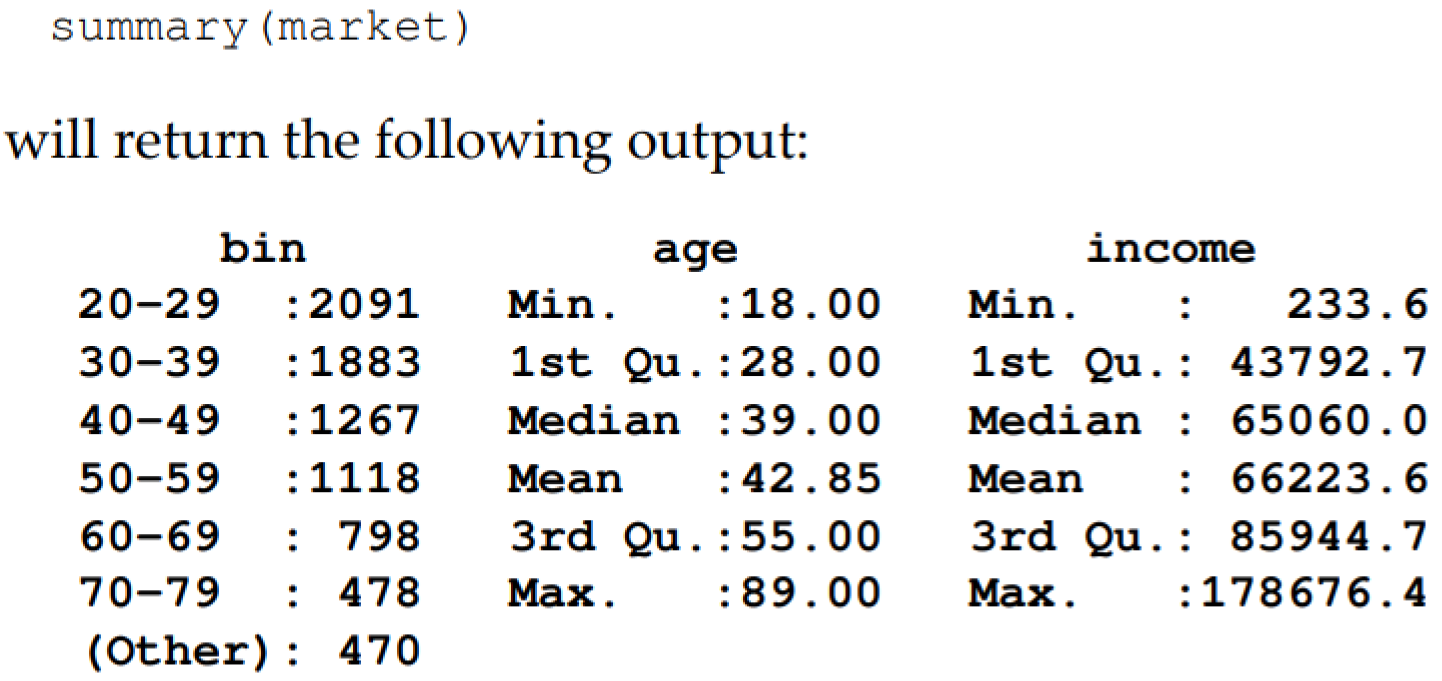
1. Create a visualization of the data plotted as income versus age and color-code the points based on cluster membership
2. Provide tables indicating the relative size of each cluster, as well as the minimum, median, and maximum age and income within each cluster

Load the data into a `market` data frame:

```
market <- read.csv("./data/Ch5_age_income_data.csv")
str(market)
```

The output is as follows:

```
'data.frame': 8105 obs. of  3 variables:
 $ bin   : Factor w/ 8 levels "10-19","20-29",..: 6 3 2 3 7 6 8 5 ...
 $ age   : int  64 33 24 33 78 62 88 54 54 31 ...
 $ income: num  87083 76808 12044 61972 60120 ...
```



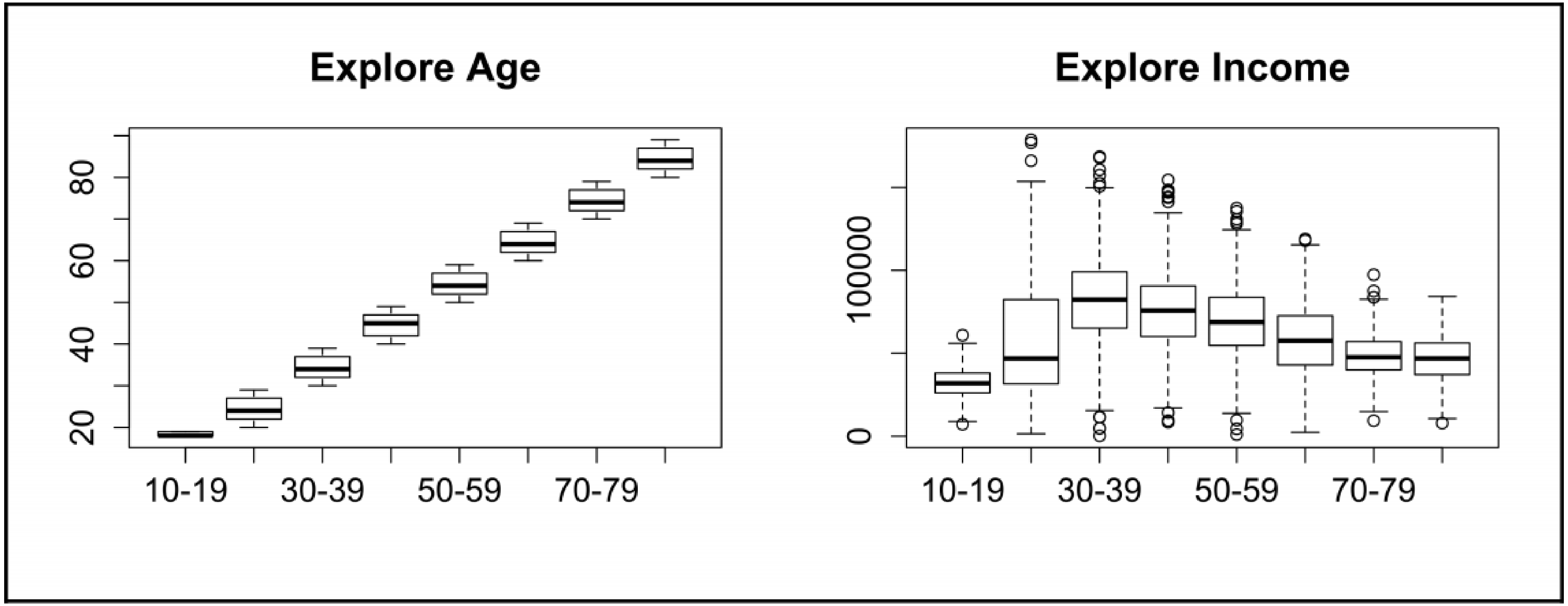
```
summary(market)
```

It will return the following output:

```
      bin            age             income
 20-29   :2091   Min.   :18.00   Min.   :   233.6
 30-39   :1883   1st Qu.:28.00   1st Qu.: 43792.7
 40-49   :1267   Median :39.00   Median : 65060.0
 50-59   :1118   Mean   :42.85   Mean   : 66223.6
 60-69   : 798   3rd Qu.:55.00   3rd Qu.: 85944.7
 70-79   : 478   Max.   :89.00   Max.   :178676.4
 (Other): 470
```

```
boxplot(market$age ~ market$bin, main = "Explore Age")
boxplot(market$income ~ market$bin, main = "Explore Income")
```

The output is as shown here:

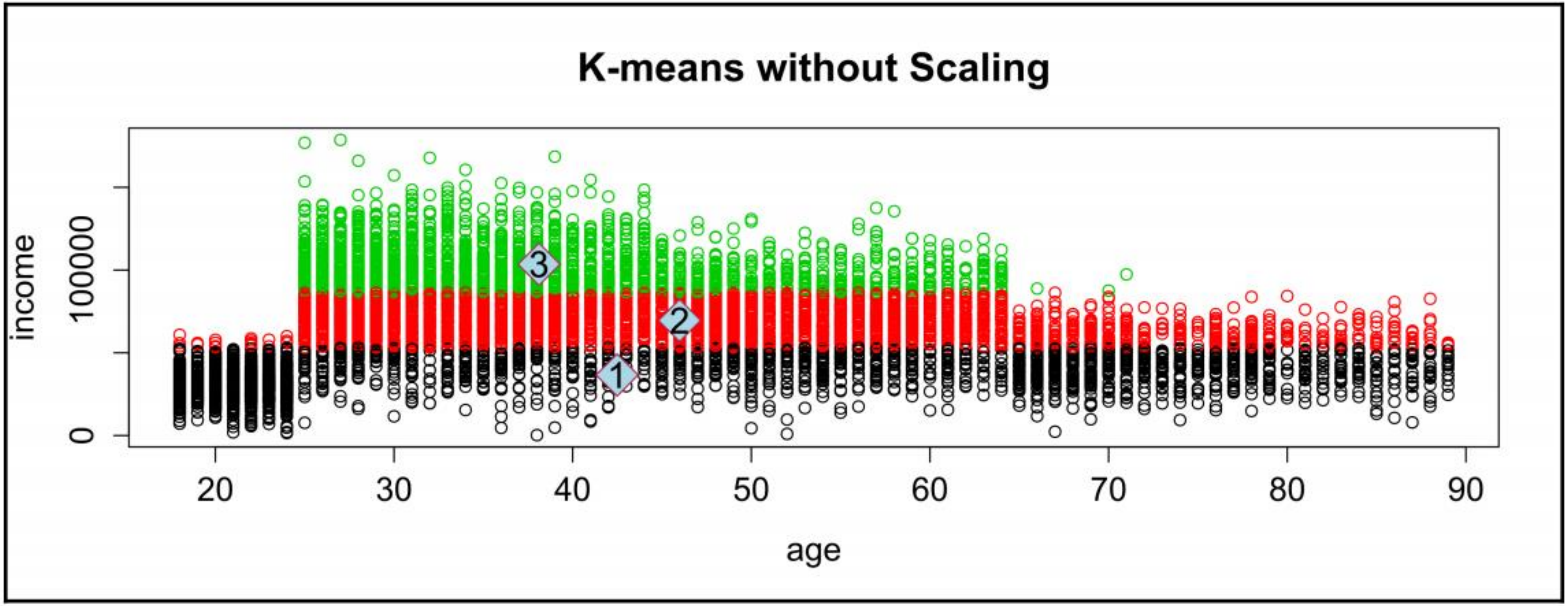


Explore Age

Explore Income

The left panel shows no improperly binned ages. The right panel shows a relationship between `age` and `income`, and this is not surprising. Is there a correlation?

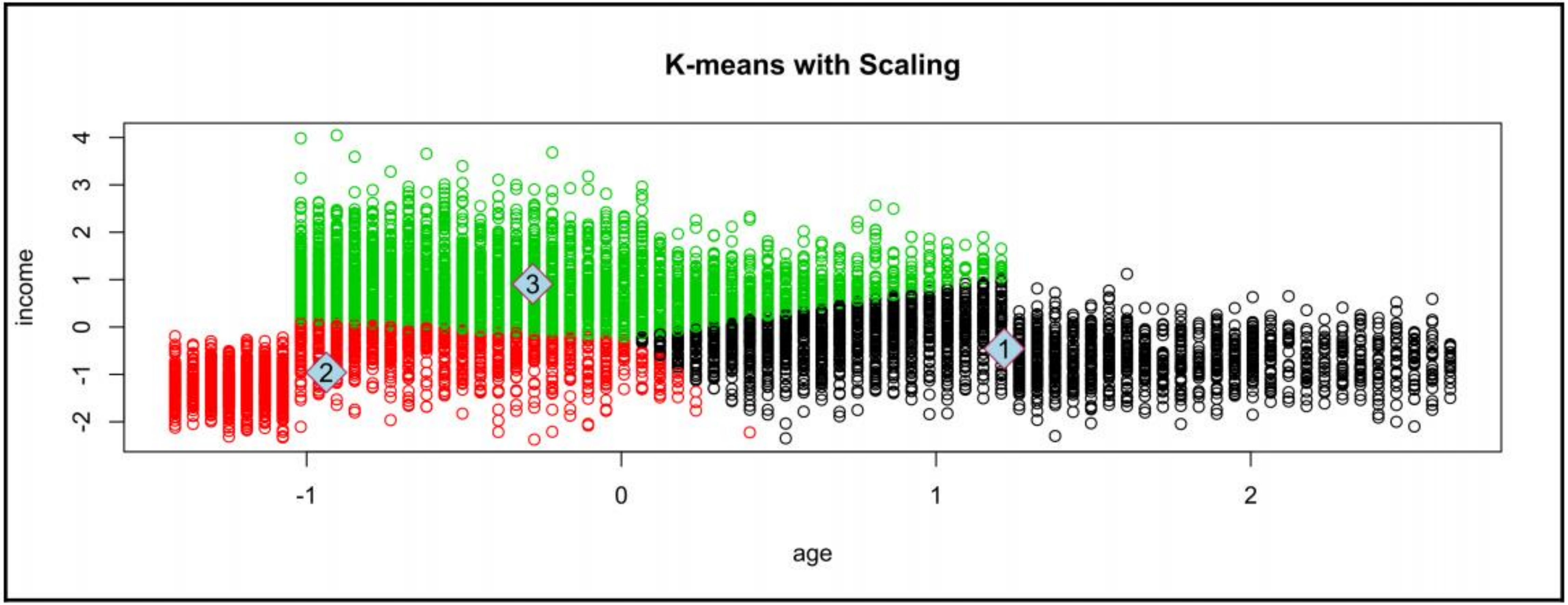
Good. There is only one thing left for you to check. Do you need to adapt any data to suit your analytic needs? This is a tough question. Consider the units (scale) of the two variables. We have `age` in tens and `income` in tens of thousands. Clustering techniques work on distance measures, so what happens with the `income` variables that are 1,000 times larger than `age`? A demonstration using a k-means plot shows the effect:



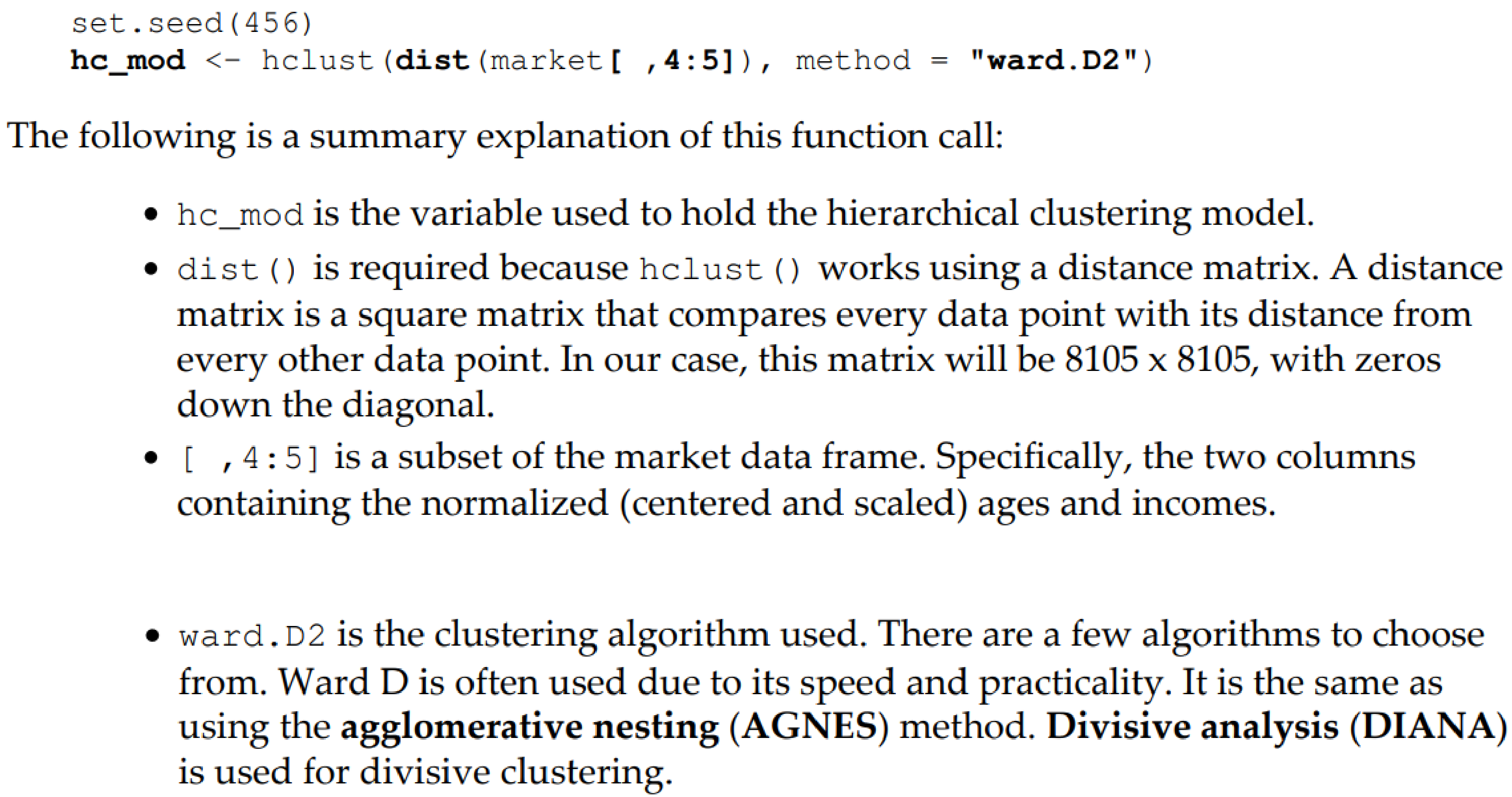
Hmmm…that is not what we expected. Why? Clustering relies on a measure of distance, and in this case, the scale of the `age` variable is smaller, relative to `income`, so it dominates the clustering. Each cluster grabbed all the data across the entire range of `age` because they are all closer to one another (in value) than to any `income` data points. You will need to normalize the data by scaling and centering it. R has a function called `scale()`. This function centers the data by subtracting the variable's mean from each observation. Likewise, it scales the data by dividing each observation by a scaling factor.

You can add the centered and scaled transformation as new columns (variables) to the data frame. The `scale()` function outputs a matrix, so you need to convert it using the `as.numeric()` function into a data type that can be added to the `market` data frame:

```
    market$age_scale <- as.numeric(scale(market$age))
    market$inc_scale <- as.numeric(scale(market$income))
```



Running the hclust() function



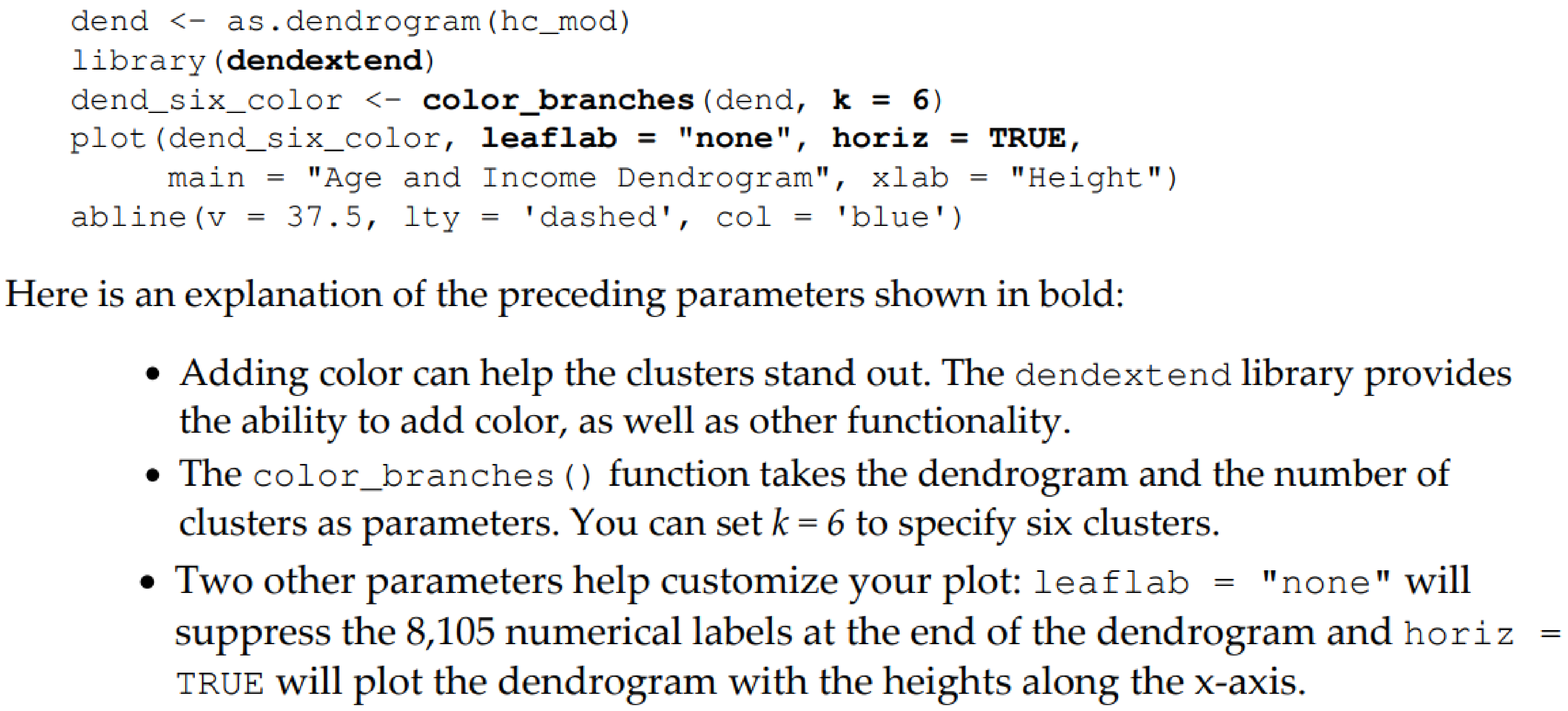
```
set.seed(456)
hc_mod <- hclust(dist(market[ ,4:5]), method = "ward.D2")
```

The following is a summary explanation of this function call:

- `hc_mod` is the variable used to hold the hierarchical clustering model.
- `dist()` is required because `hclust()` works using a distance matrix. A distance matrix is a square matrix that compares every data point with its distance from every other data point. In our case, this matrix will be 8105 x 8105, with zeros down the diagonal.
- `[ ,4:5]` is a subset of the market data frame. Specifically, the two columns containing the normalized (centered and scaled) ages and incomes.

- `ward.D2` is the clustering algorithm used. There are a few algorithms to choose from. Ward D is often used due to its speed and practicality. It is the same as using the **agglomerative nesting** (AGNES) method. **Divisive analysis** (DIANA) is used for divisive clustering.

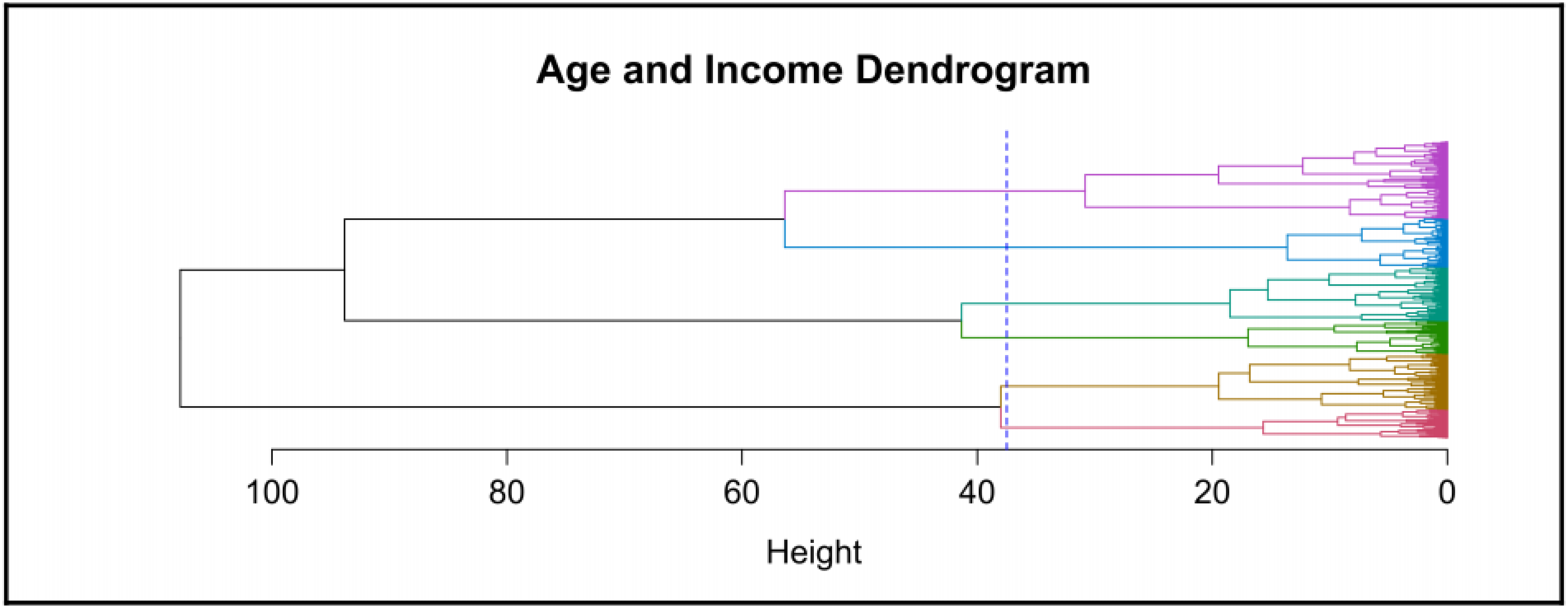
Visualizing the model output



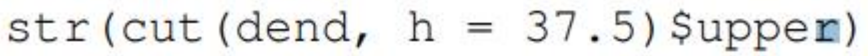
```
dend <- as.dendrogram(hc_mod)
library(dendextend)
dend_six_color <- color_branches(dend, k = 6)
plot(dend_six_color, leaflab = "none", horiz = TRUE,
     main = "Age and Income Dendrogram", xlab = "Height")
abline(v = 37.5, lty = 'dashed', col = 'blue')
```

Here is an explanation of the preceding parameters shown in bold:

- Adding color can help the clusters stand out. The `dendextend` library provides the ability to add color, as well as other functionality.
- The `color_branches()` function takes the dendrogram and the number of clusters as parameters. You can set *k = 6* to specify six clusters.
- Two other parameters help customize your plot: `leaflab = "none"` will suppress the 8,105 numerical labels at the end of the dendrogram and `horiz = TRUE` will plot the dendrogram with the heights along the x-axis.

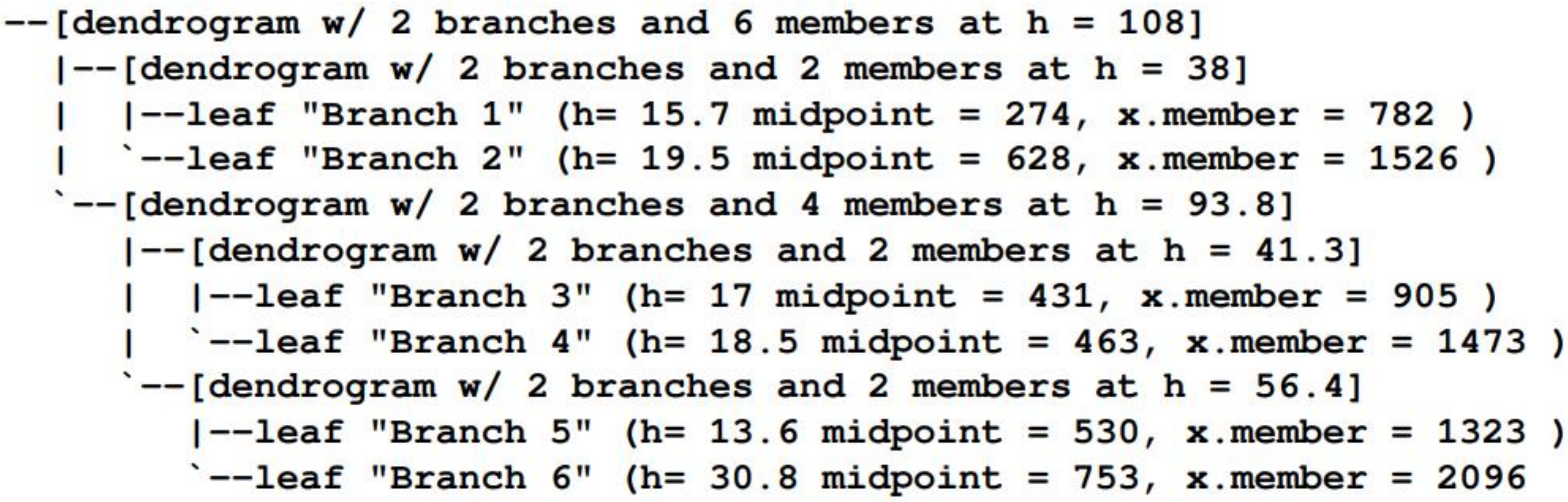

Age and Income Dendrogram
100
80
60
40
20
0
Height

Lastly, running `str()` for any height provides additional information on the dendrogram:



```
str(cut(dend, h = 37.5)$upper)
```

It will return the following output:

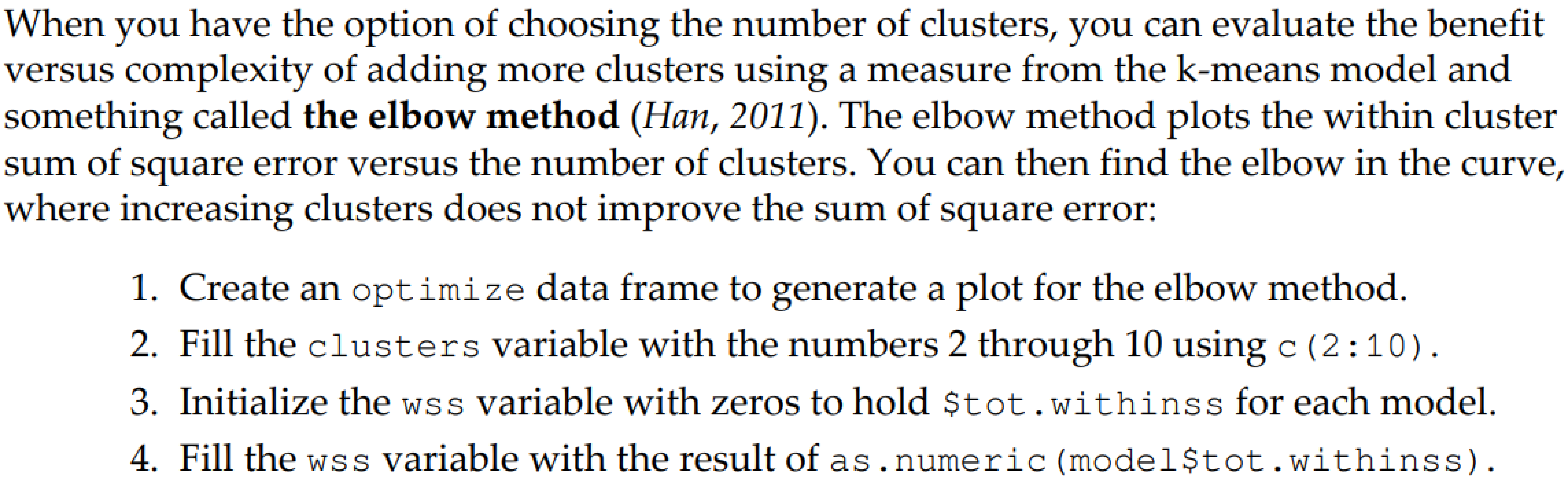


```
--[dendrogram w/ 2 branches and 6 members at h = 108]
  |--[dendrogram w/ 2 branches and 2 members at h = 38]
  |  |--leaf "Branch 1" (h= 15.7 midpoint = 274, x.member = 782 )
  |  `--leaf "Branch 2" (h= 19.5 midpoint = 628, x.member = 1526 )
  `--[dendrogram w/ 2 branches and 4 members at h = 93.8]
     |--[dendrogram w/ 2 branches and 2 members at h = 41.3]
     |  |--leaf "Branch 3" (h= 17 midpoint = 431, x.member = 905 )
     |  `--leaf "Branch 4" (h= 18.5 midpoint = 463, x.member = 1473 )
     `--[dendrogram w/ 2 branches and 2 members at h = 56.4]
        |--leaf "Branch 5" (h= 13.6 midpoint = 530, x.member = 1323 )
        `--leaf "Branch 6" (h= 30.8 midpoint = 753, x.member = 2096
```

Sometimes hierarchical clustering helps determine the number of clusters. You are going to build nine different k-means models and compare how well they explain the variance among group membership. We are doing this for the following two reasons:

- To show you a way to evaluate the increased accuracy (reduction in error) by increasing clusters in order to find an appropriate number of clusters
- To compare the clustering patterns that emerge from the two methods and give you options for developing your business case to the marketing team

Evaluating the models

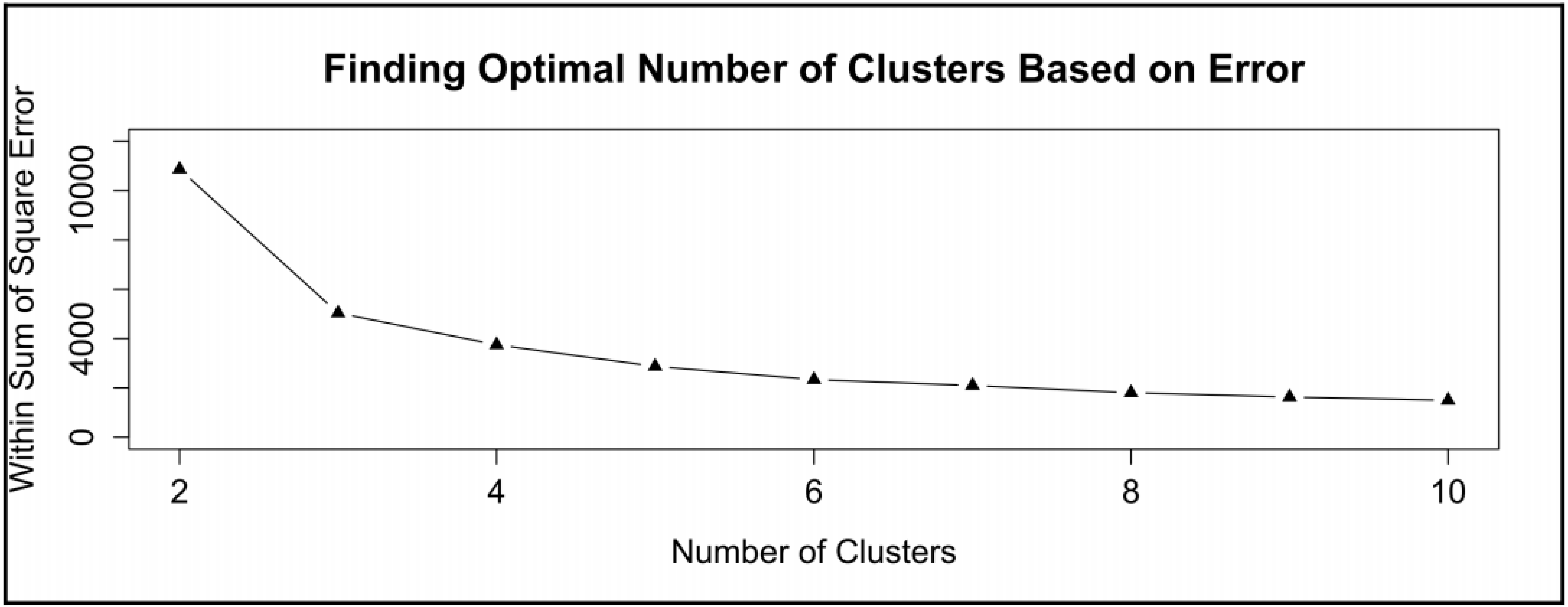


When you have the option of choosing the number of clusters, you can evaluate the benefit versus complexity of adding more clusters using a measure from the k-means model and something called **the elbow method** (*Han, 2011*). The elbow method plots the within cluster sum of square error versus the number of clusters. You can then find the elbow in the curve, where increasing clusters does not improve the sum of square error:

1. Create an `optimize` data frame to generate a plot for the elbow method.
2. Fill the `clusters` variable with the numbers 2 through 10 using `c(2:10)`.
3. Initialize the `wss` variable with zeros to hold `$tot.withinss` for each model.
4. Fill the `wss` variable with the result of `as.numeric(model$tot.withinss)`.

Let's look at the code:

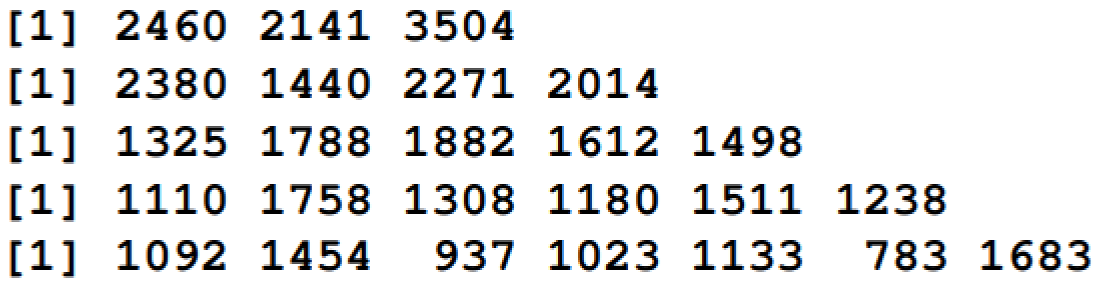
```
optimize <- data.frame(clusters = c(2:10), wss = rep(0, 9))
optimize[1, 2] <- as.numeric(two$tot.withinss)
optimize[2, 2] <- as.numeric(three$tot.withinss)
optimize[3, 2] <- as.numeric(four$tot.withinss)
optimize[4, 2] <- as.numeric(five$tot.withinss)
optimize[5, 2] <- as.numeric(six$tot.withinss)
optimize[6, 2] <- as.numeric(seven$tot.withinss)
optimize[7, 2] <- as.numeric(eight$tot.withinss)
optimize[8, 2] <- as.numeric(nine$tot.withinss)
optimize[9, 2] <- as.numeric(ten$tot.withinss)
plot(optimize$wss ~ optimize$clusters, type = "b",
     ylim = c(0, 12000), ylab = 'Within Sum of Square Error',
     main = 'Finding Optimal Number of Clusters Based on Error',
     xlab = 'Number of Clusters', pch = 17, col = 'black')
```


Finding Optimal Number of Clusters Based on Error
Within Sum of Square Error
0
4000
10000
2
4
6
8
10
Number of Clusters

Based on the elbow method, there is little apparent benefit from having more than six clusters. A reasonable question is whether that is too many marketing campaigns to manage. There were over three million individual rentals over the two-year period. That is a large enough population to explore as many as six clusters for marketing. How did k-means breakdown these clusters? You can use the `$size` attribute to see:

```
three$size; four$size; five$size; six$size; seven$size
```

The output is as follows:



```
[1] 2460 2141 3504
[1] 2380 1440 2271 2014
[1] 1325 1788 1882 1612 1498
[1] 1110 1758 1308 1180 1511 1238
[1] 1092 1454  937 1023 1133  783 1683
```

It looks like five or six clusters give reasonable target markets and reduced error. We can compare them by getting a sense of the demographics in five-cluster versus six-cluster modeling:

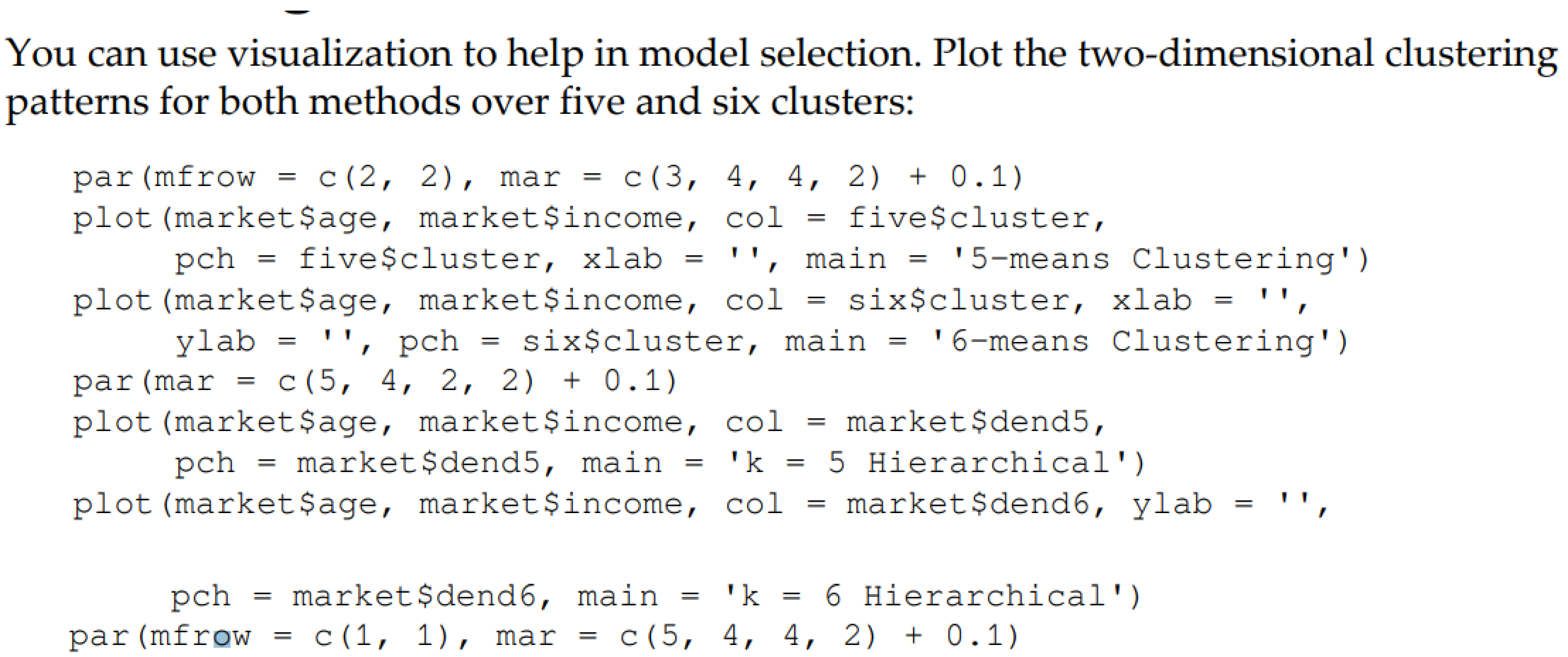
1. Add a `clus5` variable to the `market` data frame and add cluster assignments.
2. Create a five-cluster pruned dendrogram using `cutree` and `k = 5`.
3. Add a `dend5` variable and five-cluster assignments from the dendrogram.
4. Repeat steps **1 – 3** for a six-cluster k-means and dendrogram using `k = 6`.

This is shown in the following code snippet:

```
market$clus5 <- five$cluster
dend_five <- cutree(dend, k = 5)
market$dend5 <- dend_five

market$clus6 <- six$cluster
dend_six <- cutree(dend, k = 6)
market$dend6 <- dend_six
```

You have added four columns to the `market` data frame to signify the cluster each observation was assigned to, under all four models.

Choosing a model

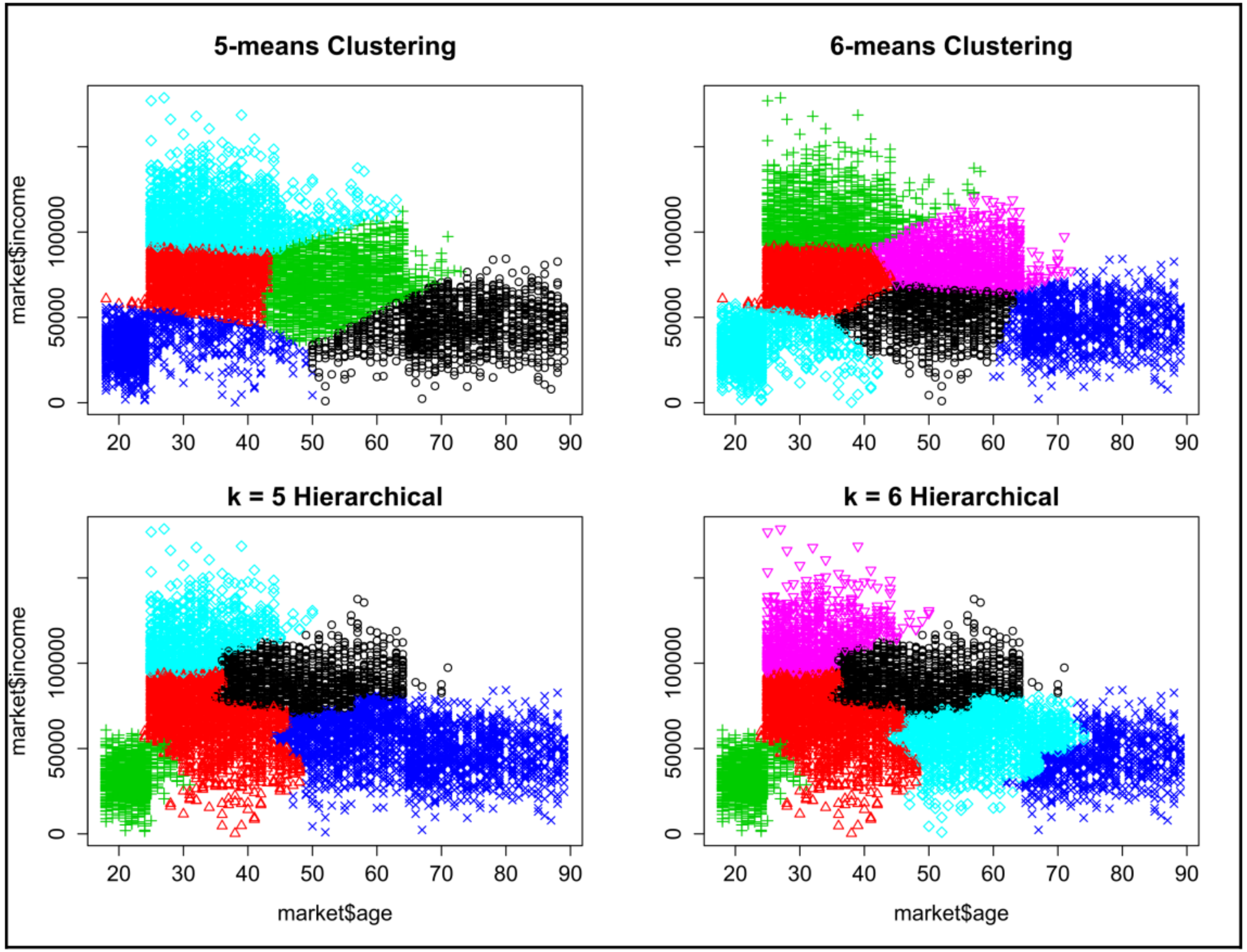


You can use visualization to help in model selection. Plot the two-dimensional clustering patterns for both methods over five and six clusters:

```
par(mfrow = c(2, 2), mar = c(3, 4, 4, 2) + 0.1)
plot(market$age, market$income, col = five$cluster,
     pch = five$cluster, xlab = '', main = '5-means Clustering')
plot(market$age, market$income, col = six$cluster, xlab = '',
     ylab = '', pch = six$cluster, main = '6-means Clustering')
par(mar = c(5, 4, 2, 2) + 0.1)
plot(market$age, market$income, col = market$dend5,
     pch = market$dend5, main = 'k = 5 Hierarchical')
plot(market$age, market$income, col = market$dend6, ylab = '',

     pch = market$dend6, main = 'k = 6 Hierarchical')
par(mfrow = c(1, 1), mar = c(5, 4, 4, 2) + 0.1)
```

This is where the power of visualization and the art of business will influence the numerical aspects of what you have been doing. This phase can take some time in real-world cases. It is a thoughtful process. Let's point out two characteristics of the different modeling approaches as shown in the following figure:

- K-means edges are not as jagged. It works based on distance from a center.
- Agglomerative hierarchical clustering groups by similarity, but you can find patterns that are not as spherical.



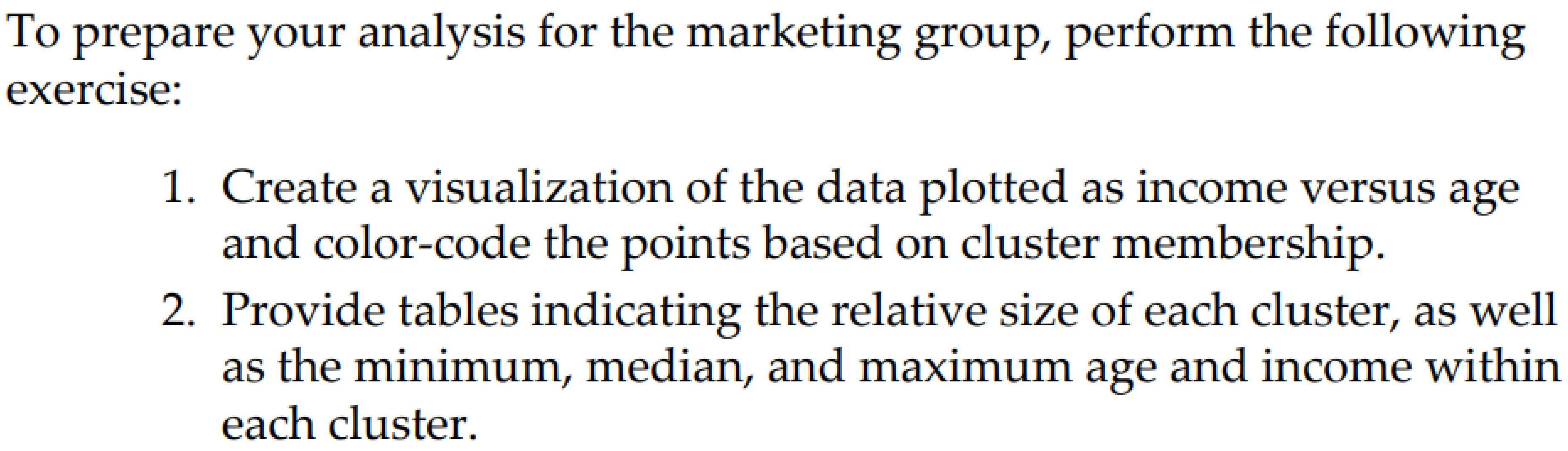
For example, hierarchical clustering gives a more ragged edge, such as the *noses* in the hierarchical plots at *age = 45* and *income = 60,000*. This also makes a difference in how the region between 30-50 years of age from 0-50,000 income appears. You will also see more income banding in k-means plots versus hierarchical plots. This raises the practical question: which of these represents a business understanding of the market more closely? The following are some observations and considerations that you will notice:

- In bike riding, age is an important factor for marketing
- The hierarchical models separate the lower income band into more age bins:
    - Younger users, as well as middle aged in the 30's through 50's
    - Older users display some type of division around retirement age
    - A *stovepipe* in the 30-35 range with only two bands of income
    - The 35-60 range adds a specific cluster from $70-140 thousand

These observations make some practical marketing sense. Regarding the 30-something aged customers, perhaps there is some wisdom in thinking of them more in terms of age and less in terms of income. On the other hand, the k-means approach has a large cluster in the lower left, hierarchical modeling creates a smaller demographic representing the youngest and lowest income. This could be the decisive tone of a successful marketing campaign.

You can certainly develop alternative conclusions, but after some time looking at these, we will decide to focus on the results from the six-cluster hierarchical model. Now, you will prepare the material for your final recommendation to the marketing group.

Preparing the results



To prepare your analysis for the marketing group, perform the following exercise:

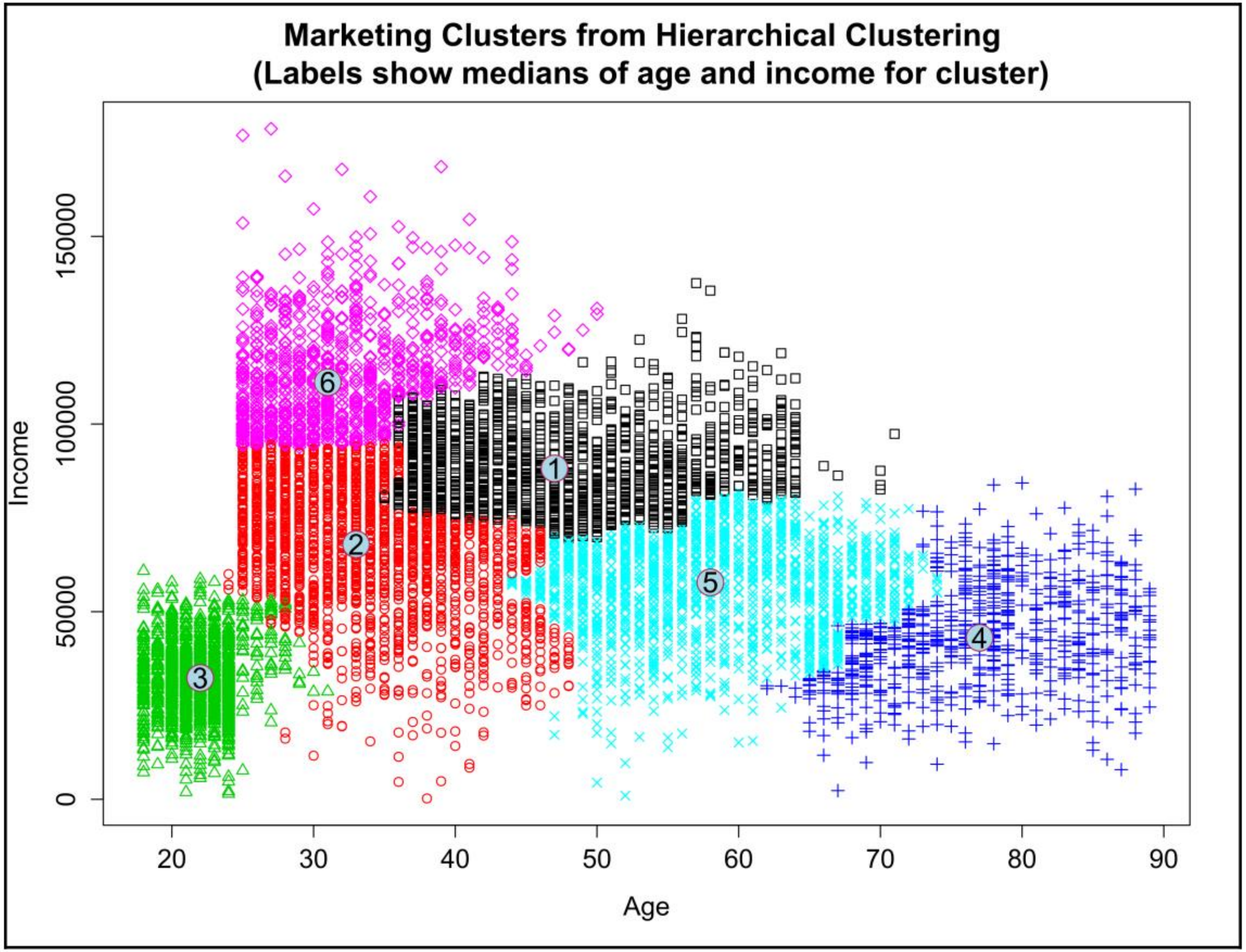
1. Create a visualization of the data plotted as income versus age and color-code the points based on cluster membership.
2. Provide tables indicating the relative size of each cluster, as well as the minimum, median, and maximum age and income within each cluster.

For visualization, you will need to generate the cluster centers manually as the `hclust` object does not have an element such as `$centers` from a `kmeans` object. You can do this by using the `dplyr` package you know and creating a `labels` data frame with the medians for `age` and `income` grouped by cluster assignment (found in the `dend6` variable):

```
library(dplyr)
labels <- as.data.frame(market %>%
     group_by(dend6) %>%
     summarise(avg_age = median(age), avg_inc = median(income)))
```

The resulting data frame will have six rows and three columns: a column with each of the six cluster numbers, a column with the median age of all assigned to each cluster, and a column with the median income for all assigned to each cluster. This is similar to the output that a k-means model produces, and we will use this data to plot our cluster labels:

```
plot(market$age, market$income, col = market$dend6,
     pch = market$dend6 - 1, xlab = "Age", ylab = "Income",
     main = 'Marketing Clusters from Hierarchical Clustering \n (Labels
show medians of age and income for cluster)')
points(labels[ ,2], labels[ ,3], pch = 21, col = 'maroon',
       bg = 'white', cex = 3)
text(labels[ ,2], labels[ ,3], cex = 1.1, col = 'black',
     labels[ ,1])
```


Marketing Clusters from Hierarchical Clustering
(Labels show medians of age and income for cluster)
Income
0
50000
100000
150000
Age
20
30
40
50
60
70
80
90
1
2
3
4
5
6

```
market %>% group_by(dend6) %>% summarise(ClusterSize = n())
```

We will get the following output:

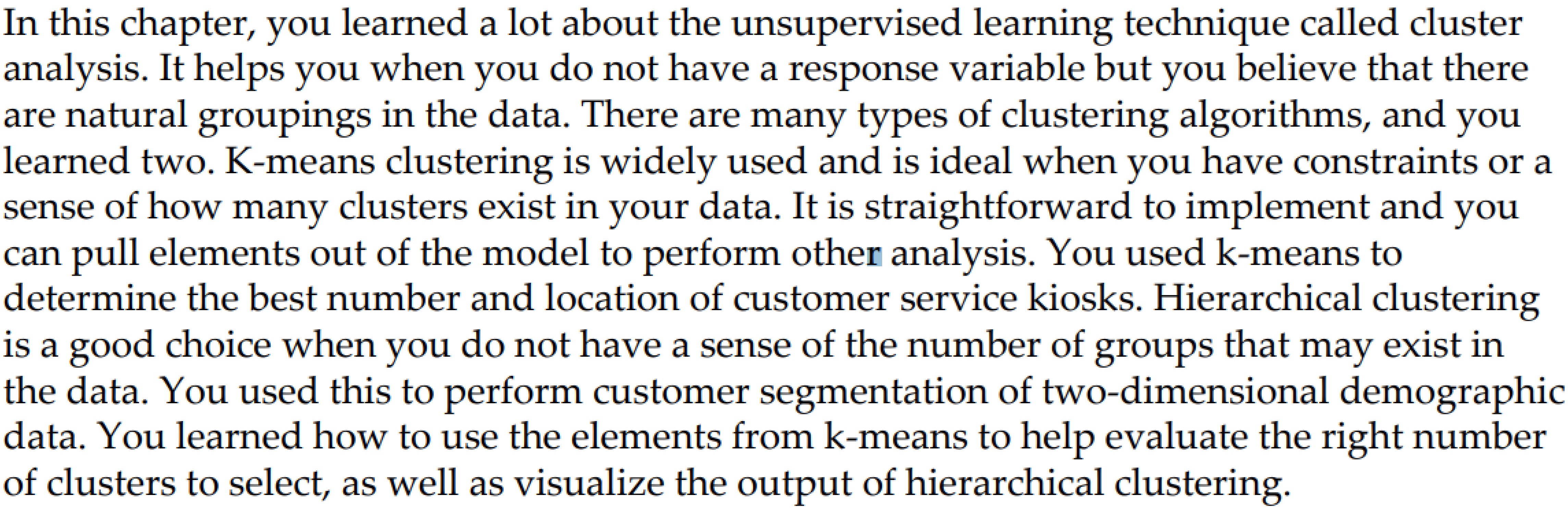
```
  dend6 ClusterSize
  (int)       (int)
1     1        1473
2     2        2096
3     3        1323
4     4         782
5     5        1526
6     6         905
```

```
arket %>% group_by(dend6) %>%
    summarise(min_age = min(age), med_age = median(age),
              max_age = max(age), med_inc = median(income),
              min_inc = min(income), max_inc = max(income))
```

This returns the table requested by marketing as shown in the following output:

```
  dend6 min_age med_age max_age   med_inc    min_inc   max_inc
  (int)   (int)   (dbl)   (int)     (dbl)      (dbl)     (dbl)
1     1      35      47      71  88170.32 69491.7763 137557.18
2     2      24      33      48  67957.66   233.6338  94708.92
3     3      18      22      31  32329.49  1484.8486  60887.37
4     4      62      77      89  43044.21  2319.2740  84300.56
5     5      44      58      74  57806.34   973.4146  81988.14
6     6      25      31      50 111124.93 93826.6611 178676.37
```

Summary



In this chapter, you learned a lot about the unsupervised learning technique called cluster analysis. It helps you when you do not have a response variable but you believe that there are natural groupings in the data. There are many types of clustering algorithms, and you learned two. K-means clustering is widely used and is ideal when you have constraints or a sense of how many clusters exist in your data. It is straightforward to implement and you can pull elements out of the model to perform other analysis. You used k-means to determine the best number and location of customer service kiosks. Hierarchical clustering is a good choice when you do not have a sense of the number of groups that may exist in the data. You used this to perform customer segmentation of two-dimensional demographic data. You learned how to use the elements from k-means to help evaluate the right number of clusters to select, as well as visualize the output of hierarchical clustering.