|  |  |
| --- | --- |
| **Learning objectives:**   1. **K-means** 2. **Hierarchical clustering** | |
| Download the following data to your computer as a csv file:  <https://docs.google.com/spreadsheets/d/1K6smMZfLnK_j_2Fiy1ikVBeK40dmW3IbnP5e_1eoBNM/edit?usp=sharing>  These are the world value survey data. Import these data into a data frame called **df**. Use the next bit of code to eliminate any data with missing responses    Run the next chunk of code to reduce the datan and convert to numeric, rename variables and then. | |
| The World Values Survey (<https://www.worldvaluessurvey.org>) collects data from around the world. There is documentation about the survey online, and the dataset is downloadable. What you see here is a very small subset of the data.  In addition to crosstabulations and other simple descriptive analyses of survey data, we may wonder if there are natural groupings of respondents—people who tend to answer the questions in a similar way. This could help us understand ‘types’ of respondents that could be useful.  These groups, or *clusters*, can be found by finding the distance between respondents on all of the survey responses. To do this, we would need a method of measuring distance of responses to each other, and summarising that across all the questions. Most questions in the world values survey survey are categorical, and they many are also ordered, so similarities in answers can be (to some degree) based on how close the ordinal numbered responses are to each other. For example, imagine person A answers question 1 with ‘5’, person B with ‘4’, and person C with ‘2’. Persons A and B are in some sense ‘closer’ to each other in their responses to this question than either are to person C.  If we look at all the ‘distances’ between all answers for all participants, we have a measure of total similarity between participants. Two participants who answer all questions the same, will have a total distance of ‘0’. Participants who answer very differently have a total distance that is very large. | |
| Write and run the code to the right. The code creates a distance object out of all the response data. The R distance object contains distances between observations in a way that facilitates analysis. |  |
| The resulting file, m, is an object of class ‘dist’. This is not a very useful object for general analysis in R, but it is useful with certain libraries and certain forms of analysis. It contains the **total distances** between all observations across all survey variables. We can look at the distribution of distances |  |
| You can see the average distance between the observations. The average respondents are 3 to 4 units distance from each other, but some are much farther away as well. |  |
| Instead of using the dist() function, we could also manually create a matrix object with these distances. This is not something you would normally do, but it will give you a sense of how for-loops operate in R. | |
| ‘loops’ are often not efficient in high level languages like R and Python, and it’s often best to avoid them. However, understanding them can be useful for deepening our understanding of operations and data. Let’s remind ourselves how *for loops* work. Here the work between the braces is done a number of times equal to the number of rows in df. The ‘i’ is an index that increments with every loop from 1 to the number of rows in df2. This incrementing is useful because it can allow us to access certain elements in matrices and vectors using the index. |  |
| For example, the ‘i’ is being used to fill the vector v with values. The values are equal to the current value of i, squared. |  |
| **Q1. Write code that:**   1. **Creates a vector of ten elements each with a value equal to 1** 2. **Loops through the elements one at a tidme subtracting the value of i from the value of the element in the ith position. Use a *for loop* for this task.** | |
| Below is a nested for-loop. Within the inner braces, work will be done a number of times equal to length(v) \* length(v). Note that k is an index that is inside the inner loop. It is initialized = 1 outside the loop, and then increments by one for ever    **Q2. What does the vector vv contain?**  **Q3. Assume mm is a 5 by 5 matrix (mm <- matrix(0,nrow=5,ncol=5). Write code to fill it with values from 1 to 25 using a nested for loop?** | |
| In this assignment we are going to learn about cluster analysis. Cluster analysis involves finding groups of things with similar attributes. In some sense, it is the opposite of principal components analysis (PCA). PCA creates synthetic indices to reduce the complexity of attributes, and cluster analysis creates synthetic things/objects to reduce the number of ‘things’ in data.  In cluster analysis, we could analyze data on students to find groups of students with common characteristics. We could also use cluster analysis to identify markets of consumers with similar purchasing habits. There are many environmental/earth science applications too, but I can’t think of any at the moment, so you’ll have to use your imagination. | |
| Write and run the code to the right. This step creates an empty matrix of the size we will need to store the distance data. It also converts the dataframe to a matrix, which is faster to work with than a dataframe. |  |
| Copy and paste the code below. **Note that it may take a few seconds (or minutes) to complete depending on your computer…**  for(i in 1:nrow(df2)){  for(j in 1:nrow(df2)){  d[i,j] <-  sqrt((mm[i,1]-mm[j,1])\*\*2) +  sqrt((mm[i,2]-mm[j,2])\*\*2) +  sqrt((mm[i,3]-mm[j,3])\*\*2) +  sqrt((mm[i,4]-mm[j,4])\*\*2) +  sqrt((mm[i,5]-mm[j,5])\*\*2) +  sqrt((mm[i,6]-mm[j,6])\*\*2)  }  }  **Q4. Briefly describe the values of d[i,j]. What do they measure/represent?** | |
| The calculations above are for illustration, and unnecessary (and inefficient) for analyzing clustering in data. However, you’ll find there are many ways to calculate distance, and this can affect the results of any cluster analysis.  We are going to use **k-means** clustering to find groups of respondents that are similar to each other. By default, K-means iteratively searches for centroids (centres) within groups of observations that minimize the distance between those centroids and the members of the groups. Write and run the code below.    K-means clustering is a simple unsupervised machine learning method for finding clusters in data. It is unsupervised because we are not telling it what is ‘correct’ ahead of time (something called ‘labeling’). It simply follows an algorithm to find clusters with desirable criteria. We use **hyperparameters** to specify how the algorithm works:  The **centers** parameter indicates how many clusters you are looking for. Here, we’re assuming only two cluster groups.  **Iter.max** = 50 says we should iterate 50 times for each run of the algorithm. More iterations takes more time, but it makes it more likely we find a nice solution.  **nstart**=50 says that we want the computer to restart the search 50 times and give us the best final solution. For some algorithms, the quality of the final solution depends on the starting conditions. By having many random starts, we make it more likely that we are not getting a bad solution because of some inferior starting conditions.  **Algorithm** = “Lloyd”. This is the algorithm used to solve the k-means problem. It is not the default algorithm, and may not even be the best, but it is reliable. If you leave this unspecified, it uses a different algorithm by default. | |
| **out1** is an object with information about the results of the k-means analysis.  We can look at the coordinateos of the two centres using the code to the right. Note that there are 6 coordinates for each centroid—this is because there are 6 variables, and the data are in 6-dimensional space. |  |
|  | |
| We can now assign the groups to the original data frame. |  |
| Take a look at the original data frame with the new member variable added (indicating which cluster each respondent belongs to).  **Q5. Using the group assignment indicator (called ‘member’) you’ve merged to the data frame df, explore the data and do some analysis to understand what features members of the two groups (1 & 2) make them distinct from one another. Ensure that you use some visualisation (base R only) to assist you in your analysis. Complete this analysis in no more than four sentences and one or two graphics.** | |
| Clustering is also used to describe hierarchical structure in data. In hierarchical clustering, membership within cluster groups is organized hierarchically—like branches in a tree. But before we get to this, write and run the code to the right to generate synthetic data.  This code is a custom function named ‘new\_data’. It has one argument, which specifies the number of new records to create in a synthetic data set. The return function (within this function) specifies what (if anything) the function returns.t  When you run the code, a new function is loaded into R, but nothing else happens… |  |
| To use the function, we have to call it. Notice that **m** receives whatd is returned from the function! |  |
| Now let’s create a distance matrix, and we’ll select the specific distance metric |  |
| Let’s create another distance matrix using another distance metric |  |
| **Q6. Create a scatterplot of these two distance matrices. Briefly comment on the pattern you see, and describe why you may be seeing this pattern (in less than 2 sentences).  *Hint: you may need to read up on these different distance metrics.*** | |
| You can fit and plot a dendrogram with the code to the right |  |
| At the bottom of the dendrogram are the observation labels. Working from the top to the bottom, but you can see how the observations are grouped into 2 groups, then 4, then 8 all the way into groups of 1. The height is an indicator of distance between the clusters. The horizontal branches show us at what point the clusters form. For reference, the pattern you see above is generated from randomly generated data. As you can imagine, interpreting these data needs to be contextualized with the data and what you might expect to see. | |
| Use this code to the right to generate a new set of random data that are synthesized with two clusters. Note how clear the clusters appear here—the two clusters are formed at relatively short distances. It looks quite different from the dedndrogram above. |  |
| **Download these data locally:**  <https://docs.google.com/spreadsheets/d/1HPxesmTAWXAhpQTxQLlwC_jJmLfqDfE9VGAu5Xjqf9I/edit?usp=sharing>  Each row is a sample and each column measures the % of the sample composed of a specific oxide.  **Q7. Use principal components analysis to reduce the data to fewer dimensions (PCs) that capture essential variation in the data, and then use k-means and/or hierarchical clustering to explore the composition of these rock samples using these PCs. Describe what you can learn from this process.**  **Hint! (next page)**  **Once you’ve done PCA and picked how many components you think are important, you can extract components and store them in a data object. For example, say the output of prcomp() is out1 and we want to extract 3 components:**  **key\_pcs <- out1$x[,1:4]**  **We can then use clustering methods on *these* data.** | |