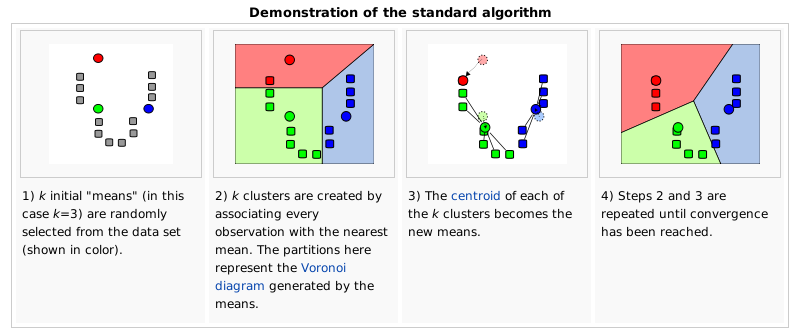
**An Introduction To k Means Clustering**

David Goody

1. k means clustering is a way of grouping and classifying data into a user specified number of groups (the k of the title). This technique is useful for trying to find underlying patterns or groups in datasets, particularly when we have large amounts of variables. Examples could include clustering schools based on educational performance or characterising wine through its chemical make up.
2. The technique works by calculating central points for each of clusters (or groups). These central points are the ones that have give the shortest overall distance between the data points and their associated cluster centres. This process doesn’t just classify the existing data points but provides boundaries that can be used to classify future data points[[1]](#footnote-1).
3. A simple example of how k means clustering works is looking at clustering people based on their height.
   1. Start by picking a value of k. Let’s go for 3.
   2. Randomly pick 3 points for our initial cluster centres. Let’s go for the minimum and maximum heights and a point in the middle.
   3. For each person work out which cluster centre they are closest to, then assign them to that cluster
   4. We’ve now got 3 clusters of people. Calculate the average height in each of these three clusters. These are our new cluster centres.
   5. For each person work out which of the new cluster centres they are closest to, then assign them to that cluster.
   6. Repeat steps d and e until our cluster centres don’t change anymore

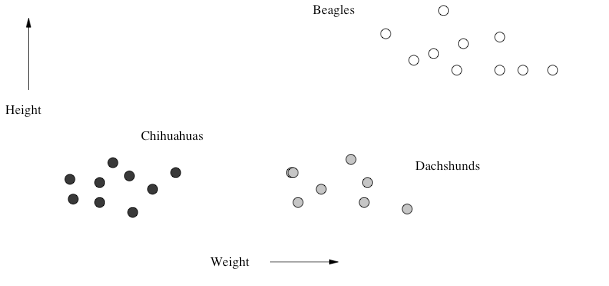
Annex A gives a graphical example of this. The end result is that our people are now in 3 cluster groups based on their height.

1. The last example was very simple and only used 1 dimension (height). We could extend this to 2 dimensions and the process is the same. Pick some cluster centres, assign the data points to the nearest cluster centre and then update the cluster centres to be the middle of the new cluster groups. Repeat the final two steps until the cluster centres don’t move anymore.



<https://en.wikipedia.org/wiki/K-means_clustering>

1. A practical example of this would be classifying dogs based on their height and weight. The following diagram shows some Beagles, Chihuahuas and Dachshunds plotted on a 2 dimensional chart. By using k-means clustering we’d not only cluster these initial points, but we’d the boundaries for each category. This means given the height and weight of a new dog we can predict whether it is a Beagle, Chihuahua and Dachshund.

****

1. In the 2 dimensional example with the height and weights of dogs, calculating the distance from each data point to the cluster centre is similar to looking at a map and working out where the nearest coffee shop is (assuming you can walk there in a straight line)[[2]](#footnote-2). We can extend this approach to any number of dimensions. If we had a dataset with 100 different variables of interest we’d be using k-means in 100 dimensional space. The underlying process is still the same.
2. K-means clustering typical requires data to:
   1. be clustered in spherical pattern around a central point
   2. have a similar level of spread around each central point

Where these conditions do not exist the method can struggle.

1. The following diagrams taken from the On My PhD site are good examples of this[[3]](#footnote-3). In the first example we have four very neat clusters with big gaps in between. If we set k=3 and colour code our clusters we end up with:
   1. a green group in the top left,
   2. a purple group in the top right
   3. a red group at the bottom (a yellow diamond is the cluster centre)

If we set k=4 then we split the bottom points into red and yellow groups. Whether k=3 or k=4 is “right” depends on the underlying data and how tightly we want to classify things. In both cases we have logical grouping because the centres are quite clear.

|  |  |
| --- | --- |
| **Where k-means works (k=3)** | **Where k-means works (k=4)** |

1. The following example wrong-foots the k-means algorithm[[4]](#footnote-4). Here we can see two curved lines of data. If we asked a human to cluster these they’d probably say cluster 1 is the line on the left and cluster 2 is the line on the right. The k-means algorithm just wants to find central points and so splits up the lines into different clusters. This could be very misleading if we are trying to understand the underlying data.

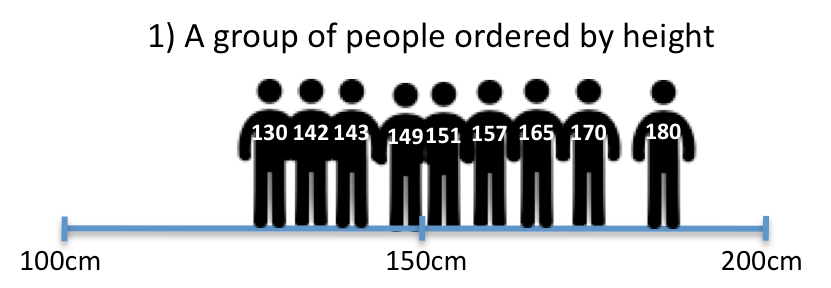
|  |  |
| --- | --- |
| **Where k-means doesn't work (k=2)** | **Where k-means doesn't work (k=3)** |

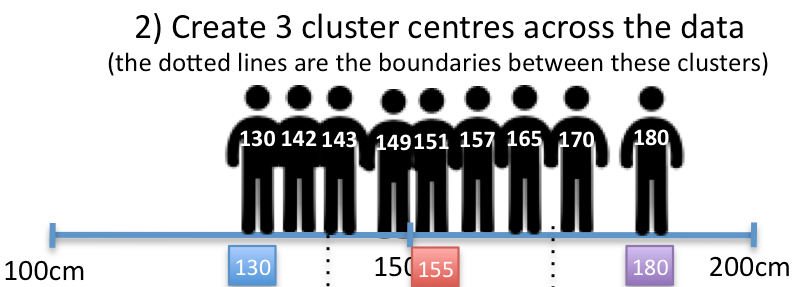
1. Other things you need to consider when working with the k-means clustering algorithm:
   1. It can be **an intensive calculation** so may take a long time to run for large datasets or and values of k
   2. The outcome may vary depending on the start points chosen and there is no guarantee that a single run will find the optimal solution. As such you may need to **run the algorithm multiple times** and compare the results
   3. You may need to run the algorithm for **multiple values of k** to find an optimal solution. One way to judge the optimal value of k is to calculate the silhouette of the data (see annex B)
   4. You need to [**standardize**](https://en.wikipedia.org/wiki/Standard_score) **your data** before running the algorithm. For example if you were using age and height or people you might find the ages had a far wider spread than heights. If you don’t standardise the data your results will be skewed by trying to minimise the big distances between ages.
   5. You need to **use continuous variables**. Height and weight are suitable. Categorical data such as gender and name are problematic because we can’t calculate the Euclidean distance between two different names[[5]](#footnote-5).
   6. There are **variants of k-means clustering** which may be more appropriate for particular datasets – including those with categorical data. These include [k-medians clustering](https://en.wikipedia.org/wiki/K-medians_clustering) and [k-mediods clustering](https://en.wikipedia.org/wiki/K-medoids).
2. Once you’ve run the clustering analysis you should inspect the values of the cluster centres to see what these tell you about the data. For example in the dogs example earlier in the paper the cluster centres tell us the typical heights and weight of breed of dog. Annex C and annex D have examples of how to apply k-means clustering in SPSS and R respectively. These include some interpretation of results.

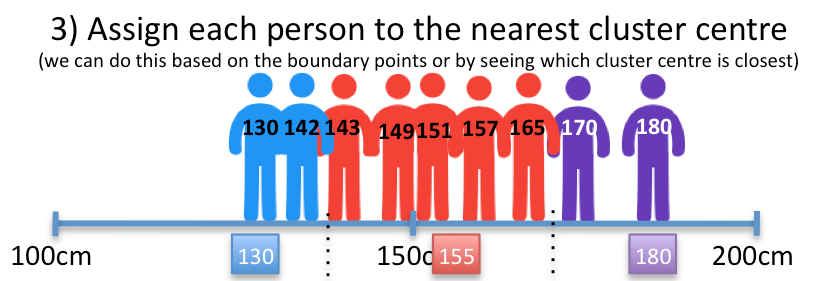
**Further reading:**

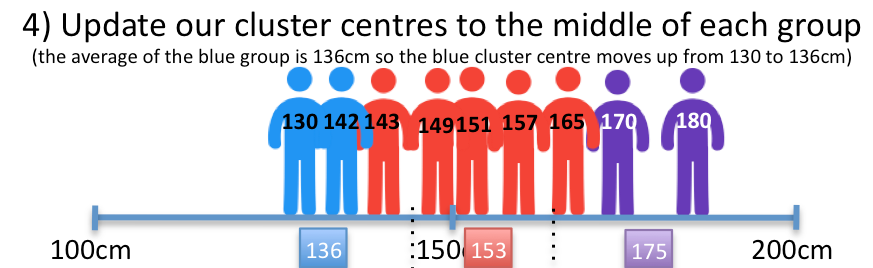
* Wikipedia: <https://en.wikipedia.org/wiki/K-means_clustering>
* An interactive visual example: <http://www.naftaliharris.com/blog/visualizing-k-means-clustering/>
* Another interactive example <http://www.onmyphd.com/?p=k-means.clustering>
* An example including Euclidean distance calculations: <http://people.revoledu.com/kardi/tutorial/kMean/NumericalExample.htm>
* A detailed explanation of clustering techniques <http://infolab.stanford.edu/~ullman/mmds/ch7.pdf>

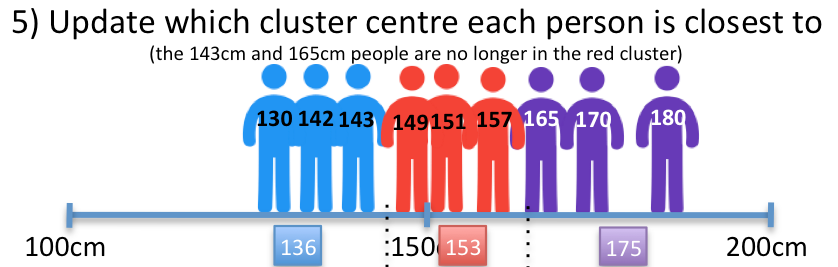
**Annex A – grouping people on height using k-means clustering**

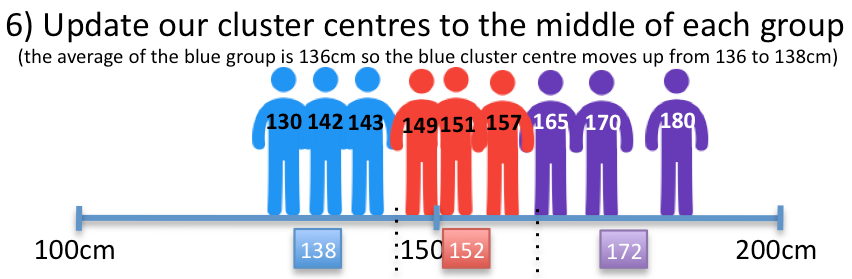


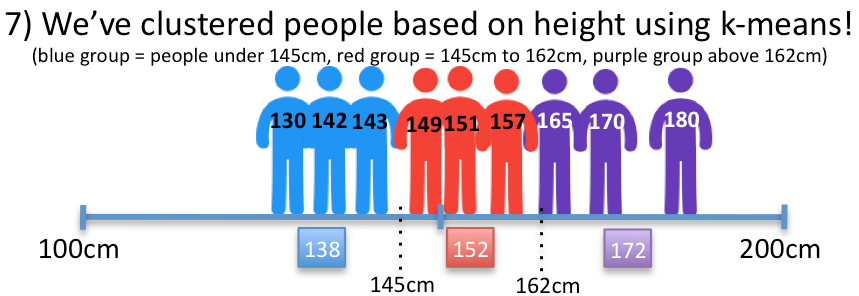












As well having clustered our 9 people we’ve created three areas (or Voronoi spaces) the we can use for classifying other people. These are:

* Blue group – People under 145cm
* Red group – People between 145cm to 162cm
* Purple group – People over 162cm

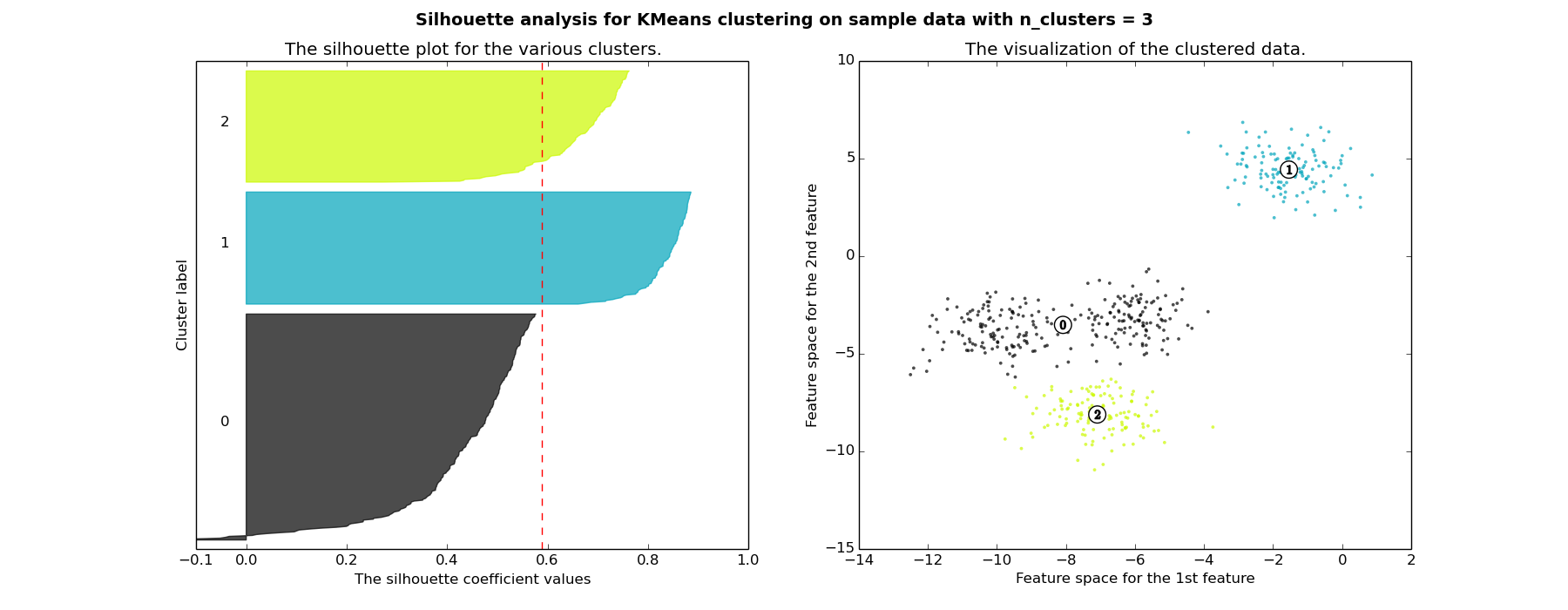
Icons made by [Freepik](http://www.flaticon.com/authors/freepik) from [www.flaticon.com](http://www.flaticon.com). Licensed by [CC BY 3.0](http://creativecommons.org/licenses/by/3.0/)

**Annex B – Silhouettes of clustered data**

1. Silhouettes are a useful way of judging how effective our clustering of the data has been. They are particularly useful when trying to decide what the optimal value of k is in k-means clustering. The silhouette for any given data point shows how close it is to points in its own cluster compared with points in the next nearest cluster. It is calculated using the formula:

* **Silhouette = ( b(i) – a(i) ) / max{a(i),b(i)}**
* **a(i)** = average distance to data points in our given data point cluster
* **b(i)** = average distance to data points in the next nearest cluster
* **max{a(i),b(i)}** = the maximum of a(i) and b(i)

1. Where clustering works well the average distance to points within a cluster (a(i)) is much smaller than the points within the neighbouring cluster (b(i)). This gives us a positive value with a maximum of 1. The closer this value is to 1 the better our clustering is. If the value is close to 0 then our clustering is fairly arbitary as data points are as close to neighbouring clusters as they are to their own clusters. A negative value implies the data point is in the wrong cluster. All silhouette values are between 1 and -1.
2. By taking the average of the silhouette values for all the points in our data set we can judge how well our clustering has worked overall. There are no strict cut-offs for clustering, and generally the higher the silhouette the better, but some guidelines are[[6]](#footnote-6):
   1. Silhouette of 0.70-1.00: a strong structure has been found
   2. Silhouette of 0.50-0.70: a reasonable structure has been found
   3. Silhouette of 0.25-0.50: structure is weak and could be artificial
   4. Silhouette of 0.25 or less: no substantial structure has been found
3. We can also plot the silhouette values to see which clusters are performing best. In the example below[[7]](#footnote-7) cluster 1 (the blue cluster) has the highest silouhette values so is the best formed cluster. Cluster 0 (the black cluster) has the lowest silhouette values as the points are widely spread and close to the yellow dots in cluster 2.

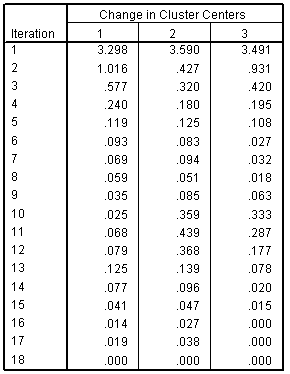


1. For more on silhouettes see:

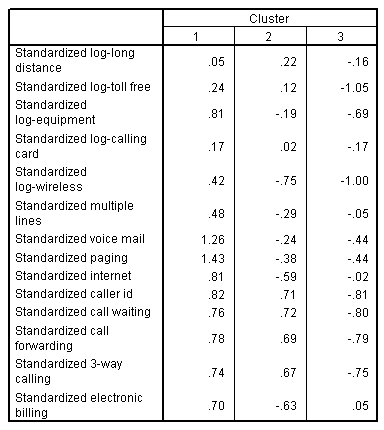
* <http://blog.data-miners.com/2011/03/cluster-silhouettes.html>
* <https://en.wikipedia.org/wiki/Silhouette_%28clustering%29>
* <http://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html>
* <https://pafnuty.wordpress.com/2013/02/04/interpretation-of-silhouette-plots-clustering/>

**Annex C – Applying k Means Clustering In SPSS**

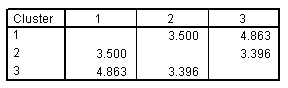
1. To apply k-means clustering in SPSS go to Analyze > Classify > K-Means Cluster… in the menus. In the dialogue box you will be asked to specify:
   1. **Which variables should be used** in the clusters (the different dimensions you are using such as height and weight)
   2. **The number of clusters** (your k value)
   3. The “**Iterate…**” options. These can control the length of time the calculation will run for. Typically the default options will be OK.
   4. The “**Save…**” options. These include whether SPSS should add an extra variables saying
      1. which cluster each case is in (you should typically include this)
      2. the distance each case is from the cluster center
   5. The “**Options…**” table. These include:
      1. Initial cluster centres – this is where the iterations started from. This is only used where you want to identify potential issues with your clustering results
      2. ANOVA – Statistics on the results. The higher F values highlight variables that create the greatest seperation between clusters. SPSS say that “the F tests are only descriptive and the resulting probabilities should not be interpreted” so this is of little value
      3. Cluster information for each case – Only useful if you have a small dataset, otherwise this table will be large. It is often best to use the save command to match this information onto your data file using the “Save…” options
2. If you selected Iterate & Classify you’ll get a table showing the amount of change in the cluster centres between each iteration. This will let you know whether the algorithm ran for enough iterations. Here the 3 clusters (k=3) stopped moving after 18 iterations, but there were minimal changes after iteration 14.



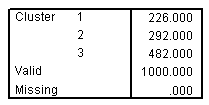
1. The Final Cluster Centre table shows the centre of the clusters based on the values for each of the variables. The table below is based on SPSS example data for customer spending on telephone services. Note that all the data here has been standardised first so positive values are above average spend and negative values are below average spend.
   1. The first cluster has a lot of high values implying this cluster is for big spenders.
   2. The second cluster are moderate spenders who mainly purchase the calling services (note the positive values towards the bottom of the table
   3. The third cluster is mainly negative values so these are low spenders



1. SPSS also shows you:
   1. The distance between the cluster centres. Here clusters 1 and 3 are furthest apart



* 1. The number of cases in each cluster. Here we have 226 cases in cluster 1



1. SPSS will not calculate the silouhette of the data automatically, but the tutorial linked below shows other methods of determining the optimal value of k.
2. For more detail of applying k-means clustering in SPSS see:
   * SPSS tutorial on cluster analysis: [http://www-01.ibm.com/support/knowledgecenter/api/content/nl/en- us/SSLVMB\_20.0.0/com.ibm.spss.statistics.cs/quickcluster\_table.htm](http://www-01.ibm.com/support/knowledgecenter/api/content/nl/en-%20us/SSLVMB_20.0.0/com.ibm.spss.statistics.cs/quickcluster_table.htm)
   * SPSS documentation on k-means: <http://www-01.ibm.com/support/knowledgecenter/SSLVMB_20.0.0/com.ibm.spss.statistics.help/idh_quic.htm>
   * Illustrative example of k-means and clustering analysis in SPSS <http://www.mvsolution.com/wp-content/uploads/SPSS-Tutorial-Cluster-Analysis.pdf>
   * Detailed explanation of cluster analysis in SPSS: [www.norusis.com/pdf/SPC\_v19.pdf](http://www.norusis.com/pdf/SPC_v19.pdf)

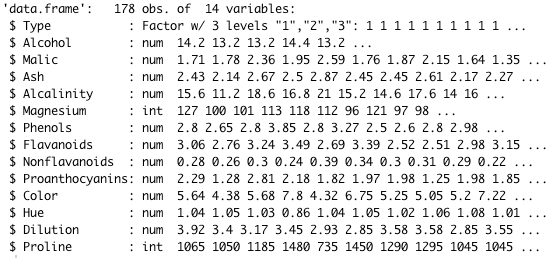
**Annex D – Applying k Means Clustering In R**

1. In this example we will use a dataset which has information on 178 wines. These 178 wines are from three different wine varieties. Here will be see how effective the k-means clustering algorithm is at clustering the 178 wines into these three varieties based on the wines characteristics (level of alcohol, malic acid, magnesium etc).
2. To access the data we need to load the rattle package

|  |
| --- |
| #Load the rattle package to access the wine data  install.packages("rattle")  require(rattle)  data(wine, package="rattle") |

1. Next we have a quick look at our wine dataset to see what we are working with. There are 14 variables. The wine variety (type) is the first value.

|  |
| --- |
| #View the structure of the wine database  str(wine) |



1. We’ll need to standardise the data before we can run the clustering. We don’t want to use the wine variety (type) in the cluster as this is giving the algorithm the answer so we only keep variables 2 to 14.

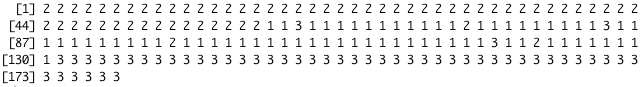
|  |
| --- |
| #Standardise the wine data  stand\_wine <- scale(wine[,2:14]) |

1. Now we’ve got the standardised data we can run the kmeans function that is part of the core R package. We need to tell the function the dataset to run on (stand\_wine), the number of clusters to create (3) and how many times to run the algorithm (nstart). In this case it will run the algorithm 25 times and keep the best result.

|  |
| --- |
| #Run k-means clustering on the wine dataset for k=3  wine.km3 <- kmeans(stand\_wine, 3, nstart=25) |

1. Our results are saved in the wine.km3 object. We can see which cluster each of the 178 cases is in by looking at the cluster element

|  |
| --- |
| #See which cluster each case is in  wine.km3$cluster |



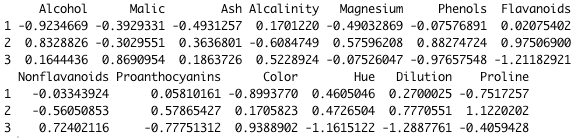
1. We can see how many wines have been put into each cluster by looking at the size element. This shows us that there are:
   1. 65 wines in cluster 1,
   2. 62 wines in cluster 2,
   3. 51 wines in cluster 3.

|  |
| --- |
| #See how many wines are in each of the three clusters  wine.km3$size |



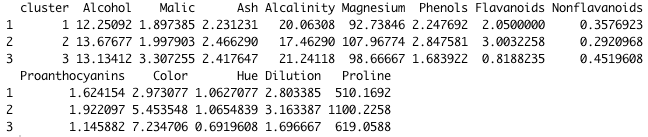
1. We can also see the centre of each of the clusters from the calculation. This will gives us the values as standardised scores.

|  |
| --- |
| #See the centre of each of the clusters (based on standardise scores)  wine.km3$centers |



1. If we want to see the cluster centres based on the raw data we match the results back onto the raw data and calculate averages. Looking at the results we see:
   1. Cluster 1 is wines with low alcohol, magnesium, colour and proline
   2. Cluster 2 is wines with high alcohol, magnesium and flavanoids
   3. Cluster 3 is wines with low phenols, flavanoids and dilution and high colour

|  |
| --- |
| #See the centre of each cluster based on the raw data  aggregate(wine[-1], by=list(cluster=wine.km3$cluster), mean) |



1. We can compare our clusters with the actual wine varieties. Here we see how well is has done.
   1. All 59 cases of wine type 1 are in cluster 2. Looks good!
   2. All 48 cases of wine type 3 are in cluster 3. K-means is doing well!
   3. The 71 cases of wine type 2 were a bit harder. 65 of these are in a cluster 1 which is pretty good but…
      1. 3 are mixed in with cluster 2 and
      2. 3 are mixed in with cluster 3.

|  |
| --- |
| #See how our 3 wine type compare with our 3 clusters  table(wine$Type, wine.km3$cluster) |



1. We can also see what would have happened if we re-ran the algorithm for k=2 and k=4.
   1. For k=2 wine types 1 and 3 end up in different clusters – this is what we want. Wine type 2 gets split across 2 categories.
   2. For k=4 we categorise wine types 1 and 3 pretty well. Wine type 2 ends up in two categories. This might be useful if we wanted to try and sub-divide that variety of wine.
      1. cluster 1 mainly has wine type 3 and
      2. cluster 3 mainly has wine type 1
      3. wine type 2 is split across cluster 2 and 4

Overall k=3 is still giving us the best match against our 3 categories (as we might expect.

|  |  |  |
| --- | --- | --- |
| #Re-run for k=2 and k=4 to see how well they would have fitted  wine.km2 <- kmeans(stand\_wine, 2, nstart=25)  wine.km4 <- kmeans(stand\_wine, 4, nstart=25)  table(wine$Type, wine.km2$cluster)  table(wine$Type, wine.km4$cluster) | | |
| **K=2** | **K=3** | **K=4** |

1. We can also use R to calculate silhouettes using the cluster package. First we install the package.

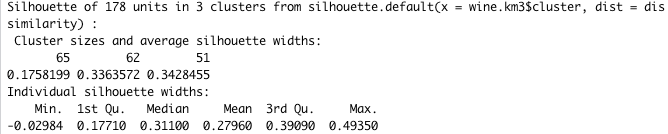
|  |
| --- |
| #Install cluster package to look at silohuette of the data  install.packages("cluster")  library(cluster) |

1. Next we calculate the dissimilarity of all the points in the data (the Euclidean distance between them) using the daisy function[[8]](#footnote-8). We’ll combine this with our clustering in a later step.

|  |
| --- |
| #Calculate the dissimialirity with the data based on euclidean distance  #Here we tell the function that we've already standardised the data (stand = TRUE)  dissimilarity <- daisy(stand\_wine, metric="euclidean", stand = TRUE) |

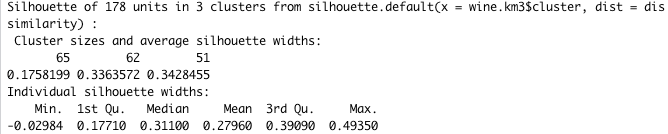
1. We can now calculate the silhouette by combining our dissimilarity data with our clustering data and applying the cluster packages silhouette function.

|  |
| --- |
| #Calculate the silhouette for our k=3 clustering  wine.km3.sil <- silhouette(wine.km3$cluster, dissimilarity) |



1. We can view a summary of all the silhouettes and the average silhouette for each group. This shows that cluster 1 is the weakest cluster (it has the lowest silhouette width). We can also see that some data points have negative silouhette values which that they don’t fit their cluster well (the minimum value is below 0.

|  |
| --- |
| #View a summary of the silhouettes  summary(wine.km3.sil) |



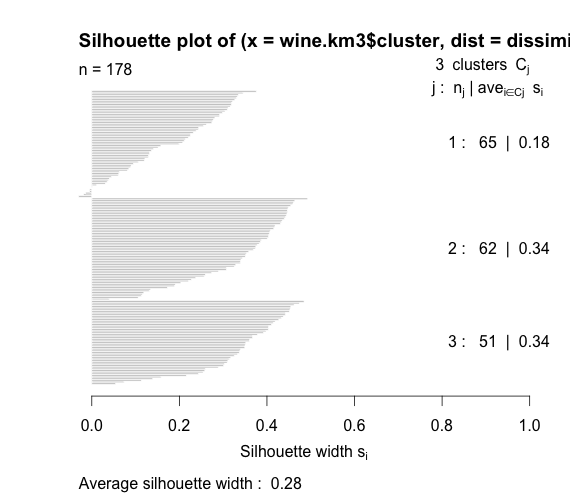
1. We can calculate the average silhouette for the entire dataset - which turns out to be 0.28 for our k=3 model, which is fairly low. This implies that there is a quite a bit of overlap within our data.

|  |
| --- |
| #View a summary of the silhouettes  summary(wine.km3.sil)$avg.width |



1. We can also plot silhouette charts. This again highlights that cluster 1 is the weakest cluster as it has the lowest silhouette values and contains some data points with negative silhouettes.

|  |
| --- |
| #Plot the silohouette charts  plot(wine.km3.sil) |



1. We can also plot the charts for our k=2 and k=4 clusters. We can see here that the average silouhette value doesn’t vary much, but the values of 0.25 and 0.26 are lower than the 0.28 we got for k=3, so k=3 is a better fitted set of clusters. For the k=4 chat the second cluster has a negative silhouette value of -0.02 which is a bad sign..

|  |  |
| --- | --- |
| #Plot charts for the k=2 and k=4 clusters  wine.km2.sil <- silhouette(wine.km2$cluster, dissimilarity)  wine.km4.sil <- silhouette(wine.km4$cluster, dissimilarity)  plot(wine.km2.sil)  plot(wine.km4.sil) | |
|  |  |

1. For more detail of applying k-means clustering in R see:
   * K-means clustering tutorial (this includes details of using the NbClust package to quickly find the optimal value of k): <http://www.r-statistics.com/2013/08/k-means-clustering-from-r-in-action/>
   * K-means function in core R package: <https://stat.ethz.ch/R-manual/R-devel/library/stats/html/kmeans.html>
   * Daisy (dissimilarity) function in cluster package: <https://stat.ethz.ch/R-manual/R-devel/library/cluster/html/daisy.html>
   * Silhouette function in cluster package: <https://stat.ethz.ch/R-manual/R-devel/library/cluster/html/silhouette.html>

1. The areas within cluster boundaries are known as [Voronoi cells](https://en.wikipedia.org/wiki/Voronoi_diagram). [↑](#footnote-ref-1)
2. This is known as [Euclidean distance](http://www.cut-the-knot.org/pythagoras/DistanceFormula.shtml). The algorithm can be applied using other distance calculations such as [Manhattan distance](https://en.wikipedia.org/wiki/Taxicab_geometry) or [spherical distance](http://epub.wu.ac.at/4000/). [↑](#footnote-ref-2)
3. <http://www.onmyphd.com/?p=k-means.clustering> [↑](#footnote-ref-3)
4. More examples of how k-means works with different types of data can be found here: <http://www.r-bloggers.com/k-means-clustering-is-not-a-free-lunch/> [↑](#footnote-ref-4)
5. For categorical data you can use [hamming distance](https://en.wikipedia.org/wiki/Hamming_distance) [↑](#footnote-ref-5)
6. <http://www.stat.berkeley.edu/~spector/s133/Clus.html> [↑](#footnote-ref-6)
7. <http://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html> [↑](#footnote-ref-7)
8. This function also support other distance calculations such as Manhattan distance [↑](#footnote-ref-8)