Creating a Geodemographic Classification Using K-means Clustering in R

Guy Lansley and James Cheshire

2018

Contents

Creating a Geodemographic Classification Using K-means Clustering in R

Guy Lansley[1] and James Cheshire[2]

University College London

Introduction

> "Geodemographic profiling (or"geodemographics") is the neighborhood scale analysis of people by where they live. Traditionally, it has fallen into the nomothetic approach to geographic enquiry - emphasizing the shared social, economic, and demographic characteristics of different types (or classes) of neighborhoods, independent of their locations relative to unique places. As such, a type of neighborhood may be widely scattered across a territory or jurisdiction. Geodemographic profiles thus provide summary indicators of the commonalities of social structure that link different locations." - Longley, 2017

Geodemographic classifications group neighbourhoods (or sometimes even indiviudal households) into types of similar characteristics based on a range of variables. They are a useful means on segmenting the population into distinctive groups in order to effectively channel resources. Such classifications have been effective deductive tools for marketing, retail and service planning industries due to the assumed association between geodemographics and behaviour. For instance, typically a classification at the broadest level may distinguish cosmopolitan neighbourhoods with high proportions of young and newly qualified workers from suburban neighbourhoods with high proportions of settled families. Such classifications work because people of like-minded characteristics tend to cluster within cities. Whilst most geodemographic products are built within the commercial sector and sold by vendors, open source alternatives are available.

The following tutorial will provide you with the basic skills to build your own geodemographic classification using R. All data and resources for this exercise are freely available. Those that are unfamiliar with R may find it useful to go through the Introduction to Spatial Data Analysis and Visualisation in R CDRC tutorial series first.

Getting started

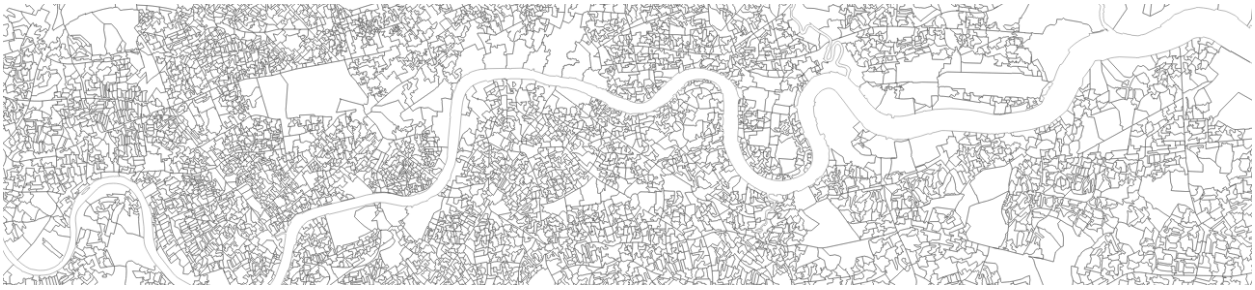
In this exercise, we will classify areas based on their demographic characteristics using a form of statistical clustering known as k-means. The clustering will group cases based on their statistical similarity as exerted by the inputted variables. It is a very effective means of reducing large multivariate databases into a singular, but informative, indicator. Cases in the same group may be similar in terms of their geodemographic compositions, but they may not be geographically proximal as the technique is aspatial. This exercise will take you through all the steps necessary to create a classification; from data selection, preparation, clustering and eventually interpretation. Although we have provided a pipeline of useful methods, you will see that the classification builder must make several analytical and subjective decisions in order to produce an optimum classification for a particular purpose. We have therefore provided some useful references at the end of this document for further reading.

The first step to creating a geodemographic classification is considering what data to include and at what granularity Finer level data will allow you to capture more intricate variations and reduce any issues of ecological fallacy. However, we also require a good number of useful variables in order to effectively segment neighbourhoods. The following data has been prepared for this exercise:

Study area: Greater London
Geography (spatial units): 2011 Output Areas (OAs)
Dataset: 2011 Census of Population

[1] g.lansley@ucl.ac.uk
[2] james.cheshire@ucl.ac.uk

2011 Output Areas are spatially aggregate units created for the dissemination of 2011 Census data. They are the lowest geographical level at which census estimates are provided with an average population of about 300 persons. As they were built to be within predefined maximum and minimum population size thresholds, they are typically smaller in area in places where there are high population densities and much larger in sparsely populated areas.



The following datasets have been provided for you for free on the CDRC website:

- *London-Census-Data.csv* - a selection of Census variables at the OA level for the UK
- *Data_Dictionary.csv* - metadata for the London-Census-Data.csv file
- *OA_2011_London_gen_MHW.shp* - a shapefile of generalised 2011 OA Boundaries for London

Like the 2011 Output Area classification, we are working with open data. However, you can also run this exercise with your own multivariate dataset.

Load the data into R

If you have never used R before you may find these short tutorials useful. R can be downloaded from https://www.r-project.org/ if it is not on your computer already. Although it is possible to conduct analysis on R directly, you may find it easier to run it via Rstudio which provides a user-friendly graphical user interface. After downloading R, Rstudio can be obtained for free from https://www.rstudio.com/.

Download the data from the CDRC website and save them to a new folder on your computer. We will now load the census data into R. We also need to set a working directory so that R knows where to open and save data on your computer. In this case, the new folder where you saved the data for this exercise will be the working directory.

```
# Set the working directory (change this to a directory on your computer)
setwd("C:/Users/Guy/Documents/Teaching/CDRC/Geodemographics")

# Load the population data from your directory
# make sure the London-Census-Data.csv is saved to your chosen working directory
pop_data <- read.csv(file = "London-Census-Data.csv",head = TRUE,sep = ",")
```

Selecting input variables

While many general purpose classifications use a wide range of variables on the population, those with a special application should be more selective. The most common uses of geodemographic classifications today are for marketing and service planning as generally consumer behaviour and service needs will vary by different geodemographic segments.

If we are classifying areas in the context of marketing we are interested developing a good understanding of residents' behaviour, their needs and their activities. Population data include several variables which can be used to provide useful inferences about consumer behaviour. For instance, areas with high proportions of families with young children will generally have different consumer traits to neighbourhoods with high

proportions of young adults. Likewise, other domains such as socio-economics, ethnicity, tenure, car ownership, etc... will also influence consumer behaviour.

The aim here is to select a number of useful variables for your geodemographic classification from the 2011 Census data. The dataset we provided contains 72 variables. You might not need all of them. Indeed, having too many similar variables may skew the results of the classification toward particular domains or subgroups. Furthermore, some variables may be of little value due to the phenomena they offer.

To aid your decision a metadata file which summarises the variables has been provided (see *Data_Dictionary.csv*). It will be worth considering what domain these variables represent before proceeding (E.g. age & sex, migration, family structure, ethnicity, employment & occupation, education and professional qualifications, housing, car ownership, health, mode of transport).

Data exploration

It might is worth conducting some basic data exploration. First, the code below will list the names of all of the variables in the dataset.

```
# lists the column names and positions
names(pop_data)
```

```
##  [1] "oacode"                  "Male"
##  [3] "Age0to4"                 "Age5to9"
##  [5] "Age10to14"               "Age15to17"
##  [7] "Age20to24"               "Age25to29"
##  [9] "Age30to59"               "Age65to74"
## [11] "Age75"                   "Single"
## [13] "Married"                 "WhiteBritish"
## [15] "WhiteOther"              "Indian"
## [17] "Pakistani"               "Chinese"
## [19] "Black"                   "CoBoldEU"
## [21] "CoBnewEU"                "BornOther"
## [23] "Christian"               "NonRelig"
## [25] "GoodVeryGoodHealth"      "BadVeryBadHealth"
## [27] "PartTime"                "FullTime"
## [29] "Carer"                   "LLTI"
## [31] "AgricutureAndExtractive" "ManufactAndConstruct"
## [33] "Finance"                 "ProfTech"
## [35] "Admin"                   "Education"
## [37] "Health"                  "NoQual"
## [39] "Level4Qual"              "WorkHome"
## [41] "TTWCar"                  "TTWwalk"
## [43] "CoupleDepChild"          "CoupleNonDepChildren"
## [45] "CoupleNoChildren"        "LoneParentDP"
## [47] "X1PersonUnder65"         "X1PersonOld"
## [49] "Detached"                "Terrace"
## [51] "Flat"                    "OwnedOutright"
## [53] "OwnedMortgage"           "SocialRent"
## [55] "PrivateRent"             "NoCar"
## [57] "TwoOrMoreCars"           "Student"
## [59] "AB"                      "C1"
## [61] "C2"                      "DE"
## [63] "EmpAndMgr"               "HighProf"
## [65] "LowProf"                 "Intermed"
## [67] "SmEmpSeEm"               "LowSuper"
```

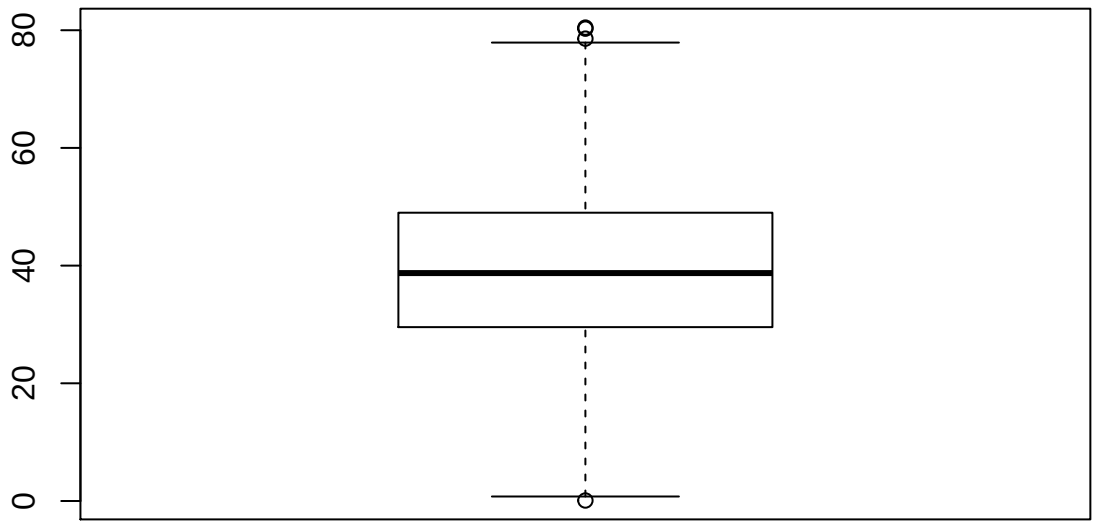
```
## [69] "SemiRout"                "Routine"
## [71] "NeverWork"               "LTUnemp"
## [73] "Density"
```

You can also produce descriptive statistics or univariate plots to observe how the data are distributed. Erratically distributed variables will influence the outputs of a classification. Some simple example techniques are demonstrated below. Here we have used the % Married variable. More details on descriptive statistics can be found in our data exploration in R tutorial.

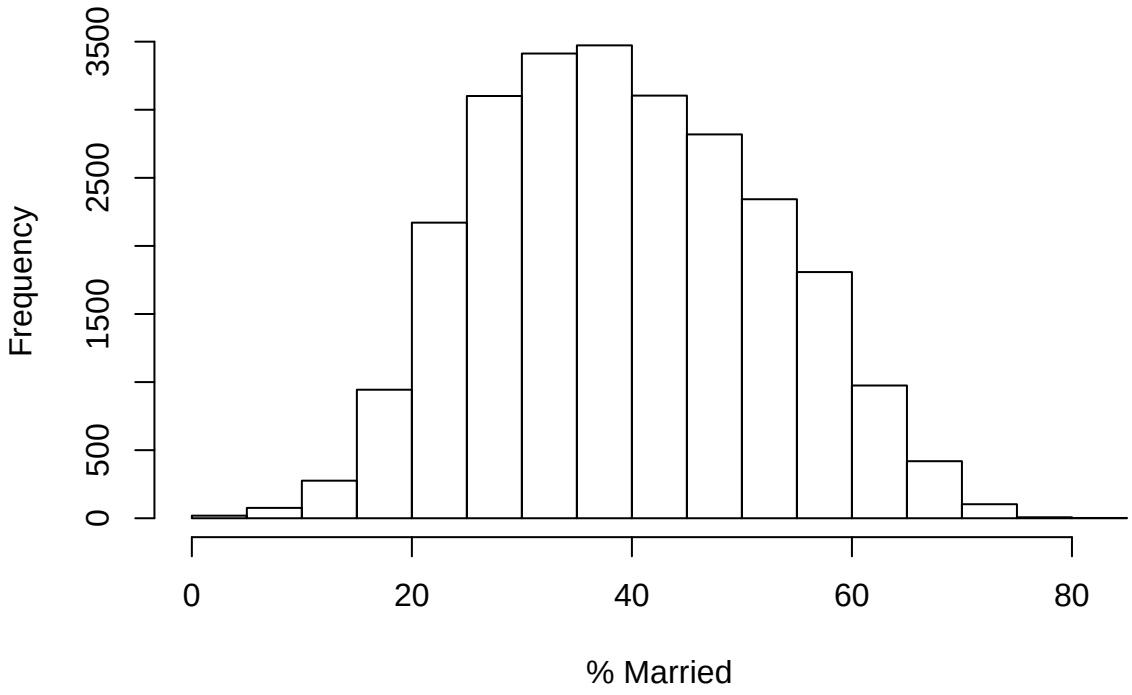
```
# summary statistics
summary(pop_data$Married)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##  0.08576 29.54545 38.72832 39.39022 48.98990 80.44693
```

```
# boxplot
boxplot(pop_data$Married)
```




```
# histogram - renamed the x axis and turned of the title
hist(pop_data$Married, xlab = "% Married", main = NULL)
```



Subsetting data

When you have decided which variables you are interested in, you can create a new subsetted data object. To select multiple columns in R you can adapt the code below to include the variables of your liking. Here we have selected 23 variables as an example.

```
# lists the column names and positions
my_data <- pop_data[,c("Age0to4", "Age15to17", "Age75", "Single", "WhiteBritish",
                       "WhiteOther", "Indian", "Pakistani","Chinese","Black",
                       "CoBoldEU","GoodVeryGoodHealth","LLTI","ManufactAndConstruct",
                       "Finance", "NoQual", "Level4Qual", "OwnedOutright","SocialRent",
                       "PrivateRent", "Student", "EmpAndMgr","LTUnemp")]
```

Data standardisation

Standardisation between areas

To reduce the effects of unbalanced base population sizes across each of the small area units (Output Areas) the variables all need to be transformed into percentages. Fortunately, this has already been done for you within the *London-Census-Data.csv* file.

However, please note that if you were to create your own percentages from aggregate population data it is important to ensure you are dividing the nominator by the correct dominator. The dominators create from

Census data will vary between data tables (i.e. the total population, the total number of households, the total of the economically active population, and so forth).

Standardisation between variables

So that erratic values within variables do not inadvertently dominate the clustering process, the input variables need to be standardised so that they each contribute an equal weight. Standardising the data will also make the final outputs much easier to interpret.

This can be achieved in various ways but here we will adopt a straightforward approach by calculating Z-scores for each variable.

Z-scores (or standard scores) describe a standardisation format where all values are represented as the number of standard deviations from the mean (0). Therefore, positive Z-scores indicate that the values are above the mean. Crucially, it is now easy to compare the varaibles on a common scale. Any of the displayed lines below will standardise the data into Z-scores, the example here being the "Male" column.

Remember: *z-score = (x- mean of the population)/standard deviation of the population*

```
stand_data <- scale(my_data$Student, center = TRUE, scale = TRUE)

# or simply...

stand_data <- scale(my_data$Student)
```

So let's loop this code so it restandardises all of your variables and produces a new data frame called standardised. The i represents what is being looped, in this case, it loops through all of the columns in the subsetted data frame (remember in our example we did not include the "oacode" column from the original data).

```
value <- colnames(my_data)

# creates a new data frame
stand_data <- my_data

# loops columns from position 1 : the last column
for(i in 1: ncol (my_data)){

 stand_data[, value[i]] <- scale(as.numeric(my_data[, value[i]]))

}
```

Measure the variables for association

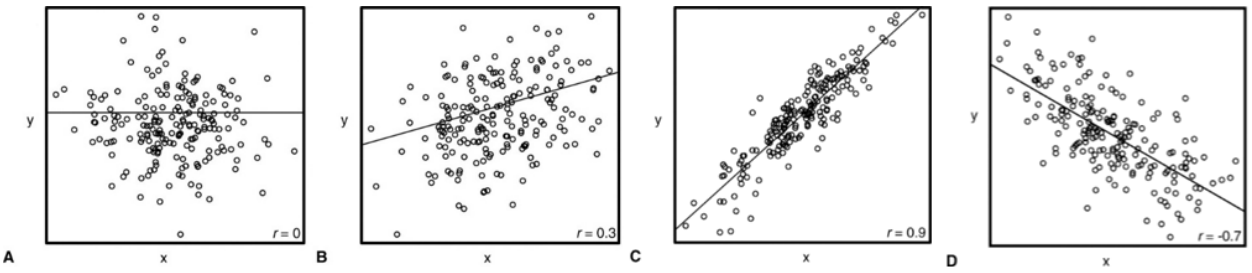
It is useful to test for associations between the final selection of variables. Variables that are collinear would essentially be conveying very similar distributions. This could give a particular phenomena a higher weighting in the final classification. To check for multicollinearity, create a Pearson's correlation matrix for the dataset.

The following instructions will produce a Pearson's correlation matrix and accompanying significance table.

```
# creates a pairwise correlation matrix for the dataframe
cor(stand_data, method  =  "pearson")
```

For every pair of data, a Pearson's coefficient and significance value are calculated. A Pearson's coefficient (or r-square value) is presented on a scale between -1 and 1. The greater the value the greater the association

between the two variables. The direction of the value equates to the direction of the relationship, negative values represent negative relationships for instance.

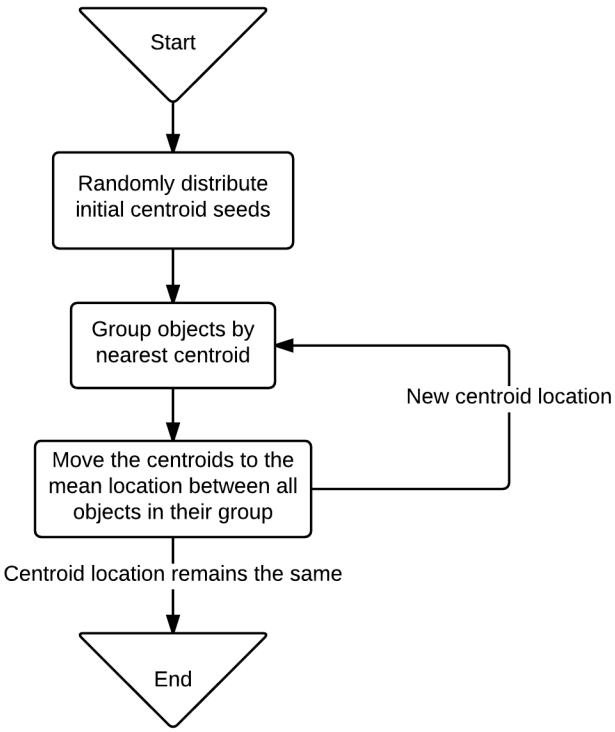


Look at the correlation matrix, do the associations seem logical?

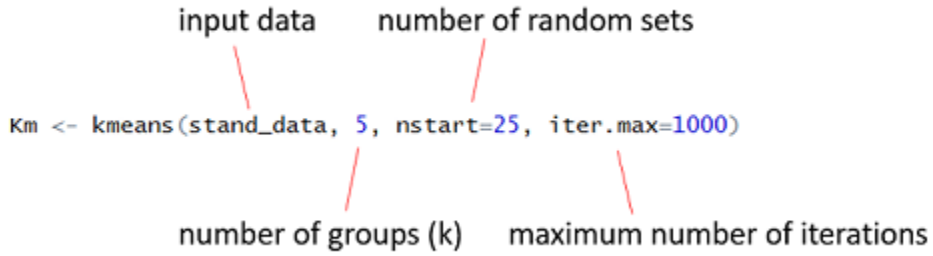
As a rule of thumb, two variables with coefficients greater than ± 0.8 can be considered to be highly correlated. In these cases, it might be reasonable to remove the variable which correlates the greatest with the other variables in the matrix so you are left with the most unique one. If you remoeved a varaible, you may wish to find new variables to replace it. However, of course, some researchers might argue that it is reasonable to retain correlated variables so long as each of them as unique value and meaning. Indeed, so long as the correlations are not quite perfect (i.e 1 or -1), both variables in the pairs may contain useful outliers which could help distinguish certain area types.

K-means clustering

We will use the k-means clustering method, the same approach used to produce the Output Area Classifications. It is a top-down approach whereby the number of cluster groups is predefined. K-means is an iterative relocation algorithm based on an error sum of squares measure. The algorithm seeks to reduce the sum distance between each data point and their respective cluster centre. The diagram below illustrates the basic algorithm process of k-means clustering. It starts by randomly allocating seeds across a multidimensional space as defined by the variables, each case is then assigned to the nearest seed centroid. In other words, the cases are assigned into cluster groups based on the seed they are nearest to across the multiple variables. Following the first iteration, a new seed is created at the centroid of each of the clusters. Each case is then re-assigned to clusters based on the distance to the nearest of these new centroids. This process repeats iteratively until the centroid seed locations cannot be moved as an optimum solution has been reached (See Harris *et al.*, 2005).



The process of the k-means algorithm (excerpted from Lansley et al, 2015). The code is annotated below. You can view the full parameters by opening the k-means documentation (run `?kmeans`).




```
# runs k-means clustering
Km <- kmeans(stand_data, 5, nstart = 25, iter.max = 1000)
```

The `kmeans()` function returns a list with the following components:

- cluster - A vector of integers (from 1:k) indicating the cluster to which each point is allocated.
- centers - A matrix of cluster centres.
- totss - The total sum of squares.
- withinss - Vector of within-cluster sum of squares, one component per cluster.
- tot.withinss - Total within-cluster sum of squares, i.e. sum(withinss).
- betweenss - The between-cluster sum of squares, i.e. totss-tot.withinss.
- size - The number of points in each cluster.
- iter - The number of (outer) iterations.
- ifault - integer: indicator of a possible algorithm problem - for experts.

We shall extract the cluster names for each Output Area and also the cluster centres for each cluster.

```
# the cluster membership for each case
KmClusters <- as.matrix(Km$cluster)
KmClusters <- as.data.frame(KmClusters)

# the cluster centres
KmCenters <- as.matrix(Km$centers)
KmCenters <- as.data.frame(KmCenters)
```

If you would like to look at the core characteristics of each of the cluster groups, you can open the *KmCenters* data frame. The cluster centres simple indicate the locations of each of the cluster centroids at the last iteration of the k-means. We will come back to this in the interpreting the cluster centres later in this tutorial.

Now we will look at the number of cases (Output Areas) in each cluster. This can be done by simply opening the *size* component from the k-means list (i.e. *Km$size*). However, this approach does not explicitly label the clusters (although they are printed in chronological order). So below we have instead created a frequency table using cluster membership table.

```
# the frequency of cases in each cluster
table(KmClusters)
```

```
## KmClusters
##    1    2    3    4    5
## 7196 6628 5990 3135 2104
```

What is the optimum number of clusters?

There is no right answer to this question. Even making judgements using some guidance on criteria involves a level of subjectivity. Your task is now to choose an appropriate number of clusters for your geodemographic classification.

Aims of the cluster analysis:

- Each cluster should be homogeneous as possible
- Each cluster group should be distinct from the other groups
- The groups should be as evenly sized as possible

In addition, to each of these, we must also consider the compositions of the cluster groups. It is important that each of the characteristics of each cluster are distinguishable and relatable to real-life neighbourhood types. We will cover interpreting the cluster centres later in this tutorial.

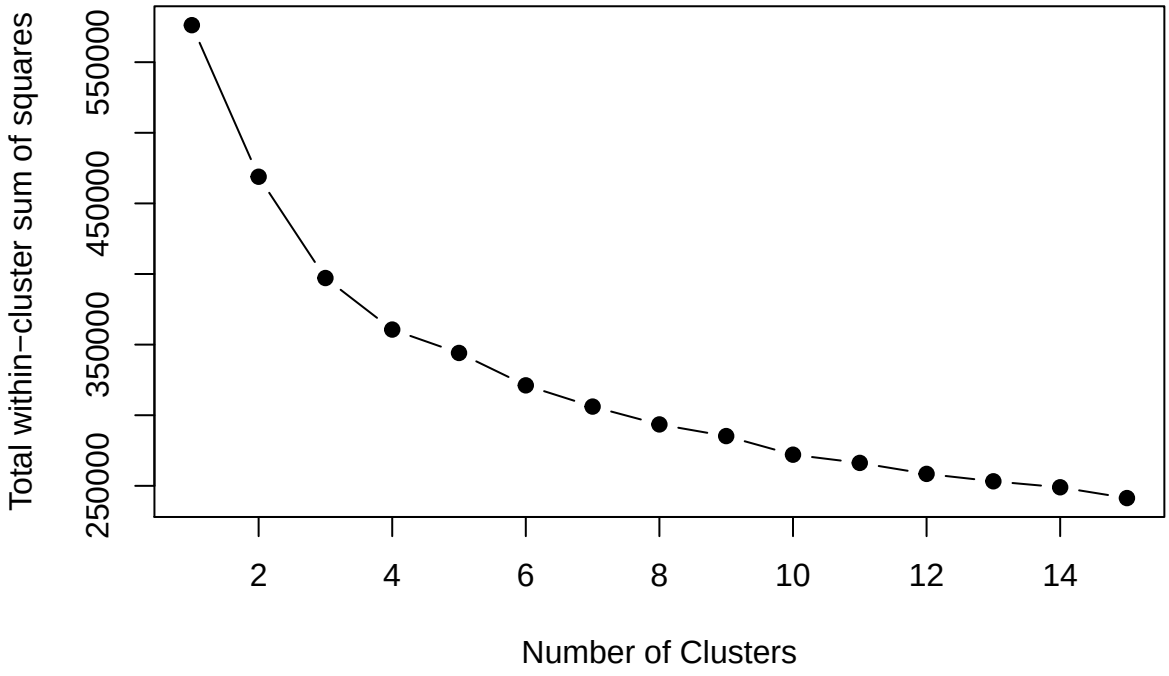
Each cluster should be homogeneous as possible Generally, the greater the number of clusters the closer the each case is to their cluster centroid on average. However, of course, with more groups the classification becomes more difficult to interpret and the differences between some groups may become more subtle. A measure we can use to check this is the within-cluster sum of squares which is the sum of the squared deviations from each observation to the cluster centroid. So larger sums indicate that the cluster is more dispersed and less homogenous.

A plot of the within groups sum of squares by number of clusters extracted can help determine the appropriate number of clusters (k). The tip is to look for a bend in the plot similar to a scree test in factor analysis.

```
# function to compute total within-cluster sum of square
# creates an empty data object first
wss <- NULL

# finding the tot.withinss for 15 kmeans models
for (i in 1:15) wss[i] <- kmeans(stand_data,centers = i,iter.max = 1000)$tot.withinss
```

```
plot(1:15, wss, type = "b", pch = 19, xlab = "Number of Clusters",
     ylab = "Total within-cluster sum of squares")
```



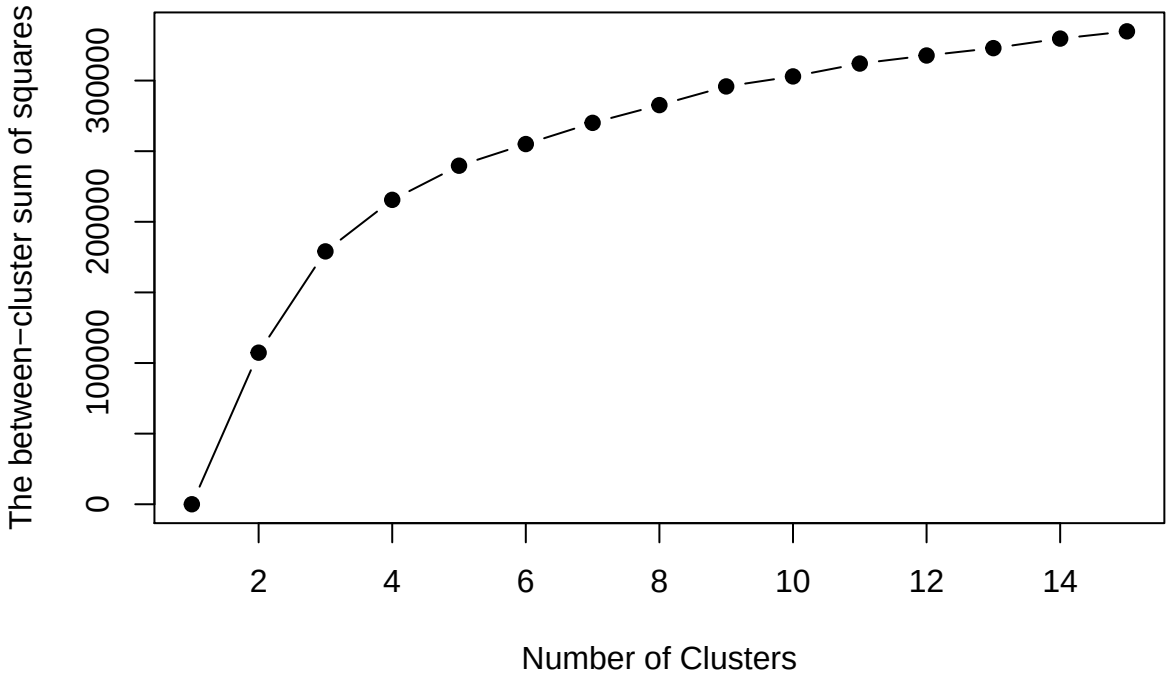
Each cluster group should be distinct from the other groups We can also measure how distinctive each of the clusters are in each model overall by looking at the between-cluster sum of squares. This value essentially measures how far apart clusters from different centres are. If the value is low then cases from different clusters will not be that distinctive. It is also important to observe the cluster centres to ensure that the compositions of each of the groups are logical and sufficiently unique.

We can look at the between-cluster sum of squares to understand how discriminatory the models are when different numbers of groups (k) are produced.

```
# we can also look at the betweenss values to observe how distinctive the clusters are
bss <- NULL

for (i in 1:15) bss[i] <- kmeans(stand_data,centers = i,iter.max = 1000)$betweenss

plot(1:15, bss, type = "b", pch = 19, xlab = "Number of Clusters",
     ylab = "The between-cluster sum of squares", ,)
```



Using the two plots, is it possible to determine an appropriate number of groups?

There are methods for identifying statistically optimal groups such as the average silhoutette method and gap statistic method. The NbClust package provides 30 indexes for determining the optimal number of clusters in a dataset (including both of the forementioned methods). However, they are computationally inefficient to run on very large datasets. In our example we have over 25,000 cases.

The groups should be as evenly sized as possible This is more of a rule of thumb for geodemographics. However, if a single cluster group comprises a large share of the data, then it is obviously undesirable for practicioners hoping to glean a better understanding of how the needs of neighbourhoods may vary. You might notice that your K-means has produced a small cluster group which has exagerated cluster centre values for a small number of variables, this is not unusual when attempting to segment social groups.

There are two main ways to edit the size of the groups without manipulating the variables: joining similar groups or splitting large groups.

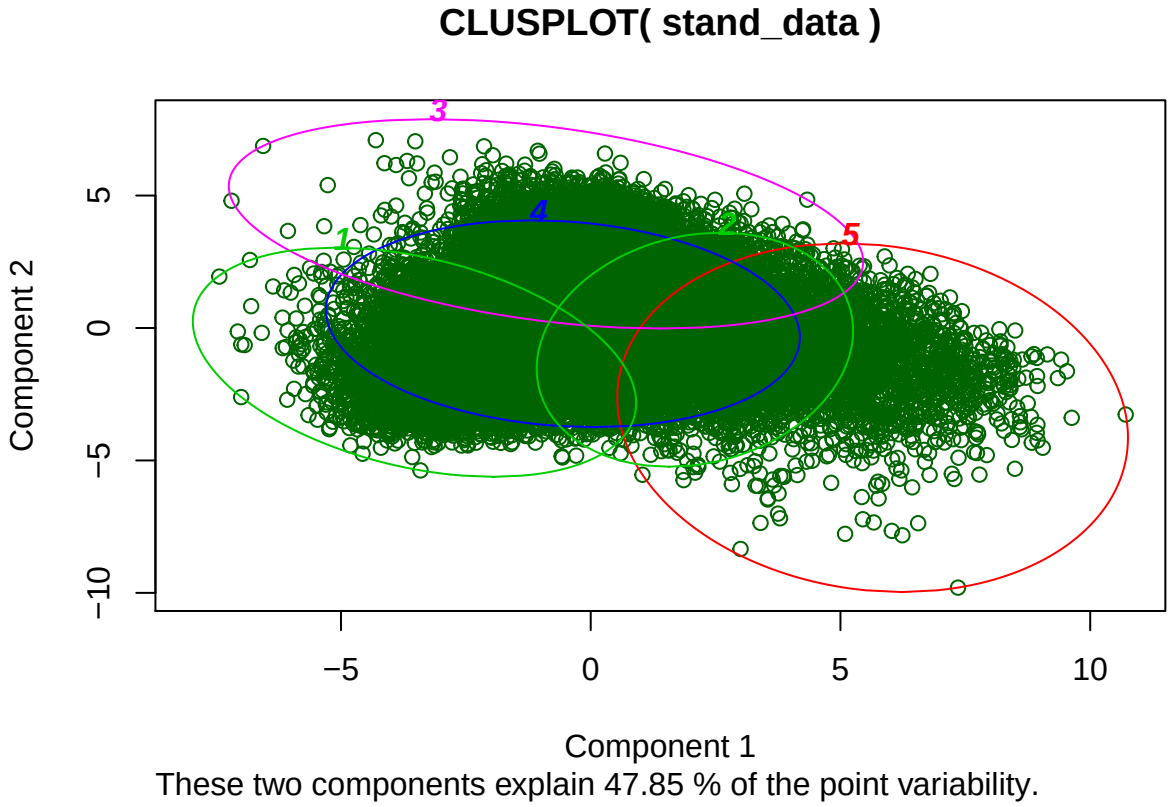
If one cluster group is particularly dominant in size, you can force it into two groups by isolating the cases (output areas) from that group only and running a 2 group solution k-means. With only one group selected, you can run a k-means as you did before, but change the number of groups produced to 2. As only one group is selected, the output areas from other groups will be excluded from the analysis. If you are happy with the results you now need to combine the two new groups with the original group membership variable (*KmClusters*). In the creation of OAC, this same technique was used for each supergroup to create the groups and subgroups.

Evaluating a classification

Having selected the number of clusters for the final model, below are two means of visualising the fit of a k-means model, you may need to install the packages first (as described earlier). In the example 6 = k, and the data corresponds with all of the Output Areas of London (25,053).

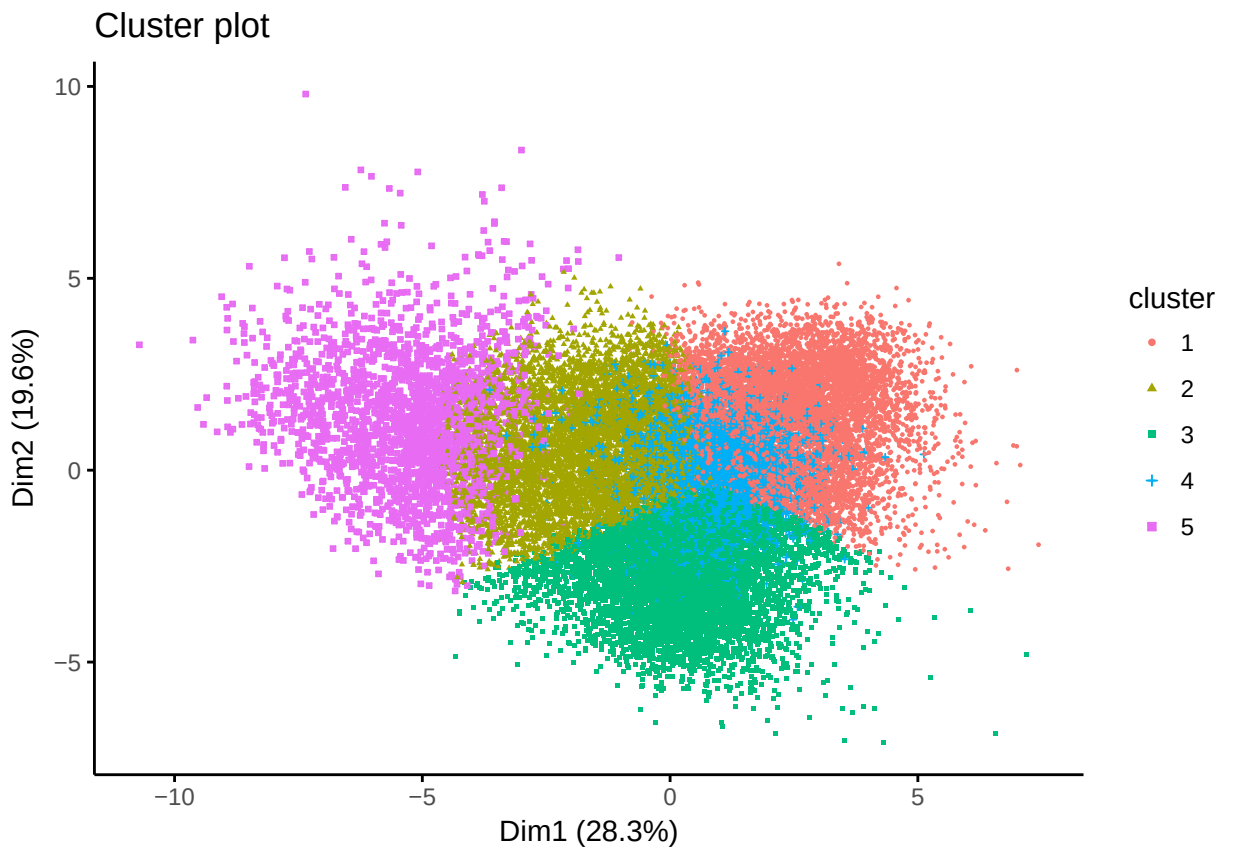
Firstly we shall us 'clusplot()', this Creates a bivariate plot visualising a partition (clustering) of the data. All observation are represented by points in the plot, using principal components or multidimensional scaling. Around each cluster, an ellipse is drawn.

```
# Cluster Plot against 1st 2 principal components
library(cluster)
clusplot(stand_data,Km$cluster, color = TRUE, shade = FALSE,
    labels = 4, lines = 0, plotchar = FALSE)
```



Perhaps a clearer visualision is offered by `fviz_cluster()` (from the factoextra package). It plots each of the cases in our data against the first 2 principal components and colours them by their cluster group from our K-means model. You could also try rerunning the K-means again with different number of groups to see how well the models disciminate based on two principal components.

```
# plots cases in the kmeans against 1st 2 principal components
library(ggplot2)
library(factoextra) # this may require you to install additional packages
fviz_cluster(Km, data = stand_data, geom = "point", ellipse = F, pointsize = 0.5,
             ggtheme = theme_classic())
```



You might notice that although there are obvious clusters in the graphs, there are some points that may overlap with other clusters. This is because the data we used is complex and includes several variables which may be exerting unique patterns, therefore, unrepresented by 2 principal components alone.

Interpreting the cluster centres

Before any efforts are made to visualise the data, it is important you understand what it represents. The cluster centres (*kmCenters*), indicates the coordinates of the centroid for each cluster group once the k-means had reached its optimum solution. It, therefore, is a good indicator of the average characteristics of each group based on the n variables that were included in the original model.

We inputted Z-score standardised data into the model, therefore the cluster centres are still represented as Z-scores. Zero represents the mean for each variable and values above or below indicate the number of standard deviations away from the average. The values can, therefore, be used to very easily understand how unique each group is relative to the whole sample (in this case, all of the pop_data you inputted).

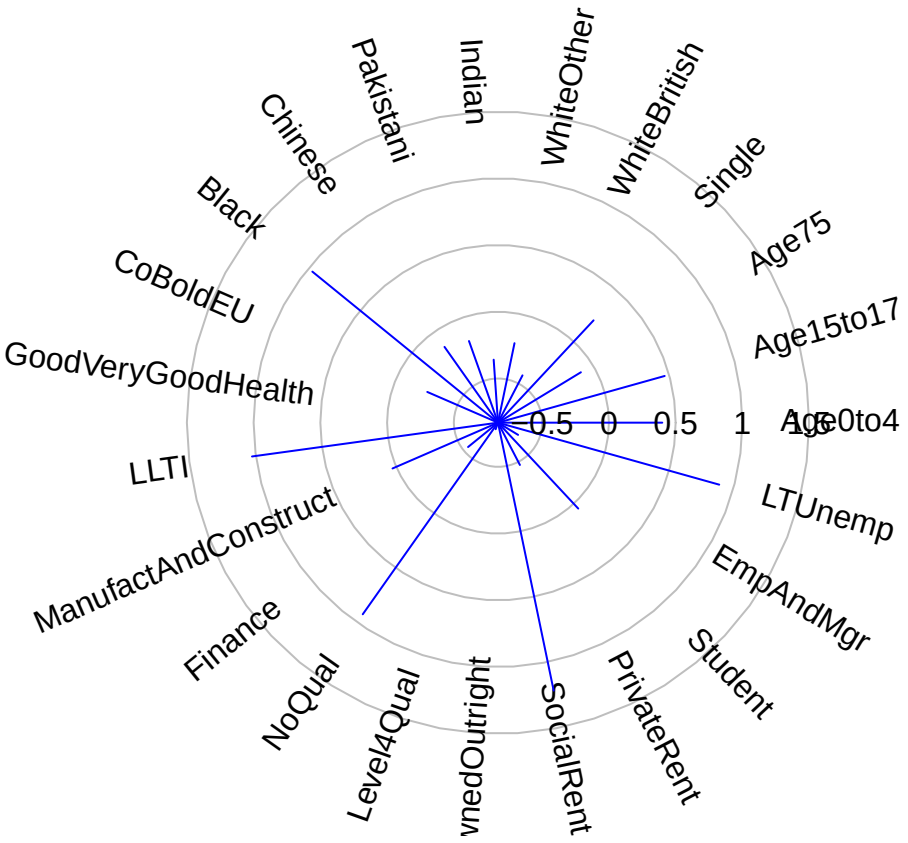
Open the KmCenters file to view the cluster centres for all of your groups. Is it immediately clear what your groups represent? It might be easier to create charts to visualise the characteristics of each cluster group. In the example below we will create radial plots, using the first group as our example.

Creating charts in R: Radial Plots

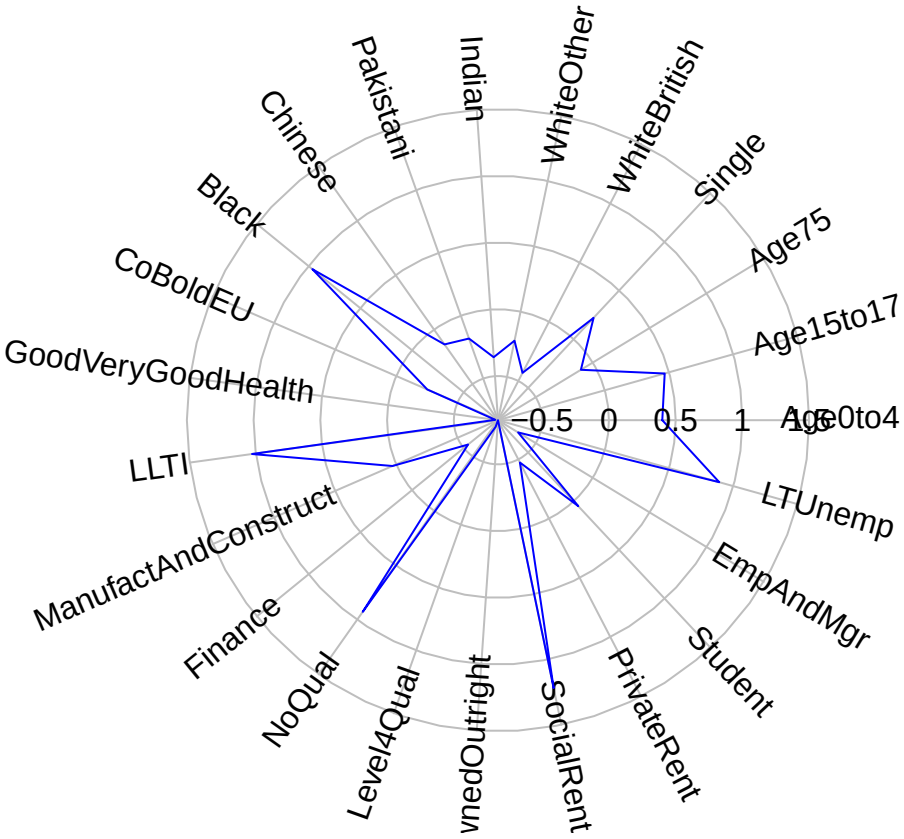
The following will produce a radial plot of the first group from the *KmCenters* data frame.

```
library(plotrix)

# creates a radial plot for the first group [1,]
# the boxed.radial (False) prevents white boxes forming under labels
# radlab rotates the labels
radial.plot(KmCenters[1,], labels = colnames(KmCenters),
            boxed.radial = FALSE, show.radial.grid = FALSE,
            line.col = "blue", radlab = TRUE )
```




```
# Creats a polygon(p) plot
radial.plot(KmCenters[1,], labels =  colnames(KmCenters),
            boxed.radial = FALSE, show.radial.grid = TRUE,
            line.col = "blue",  radlab = TRUE,
            rp.type = "p")
```

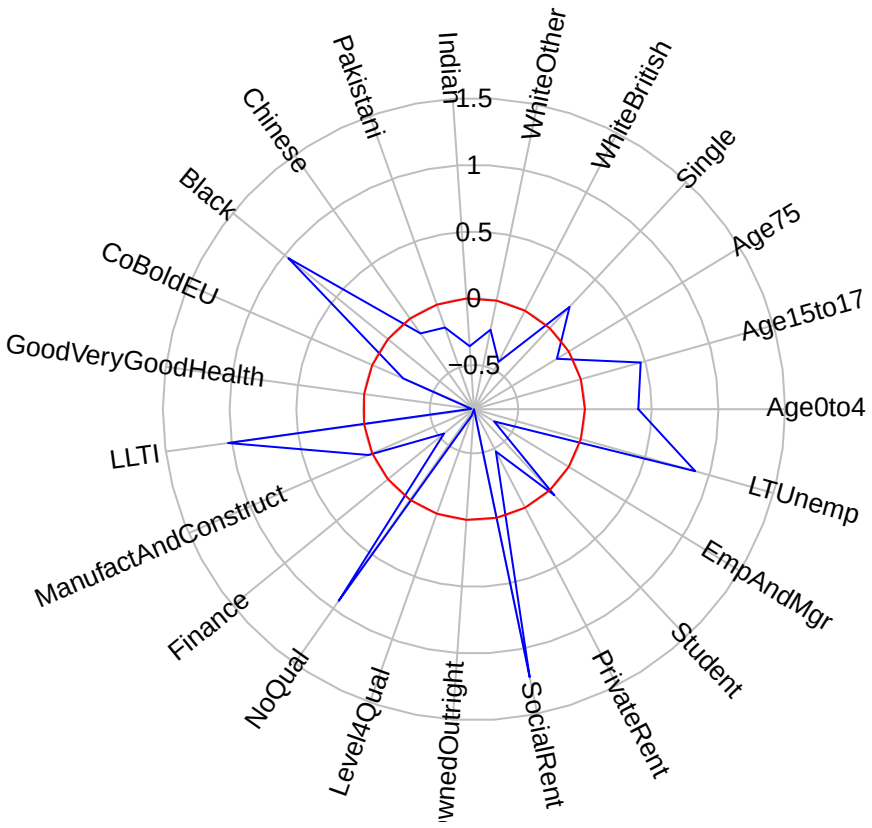


Now lets plot a zero line to indicate the average for the whole data. We have only explored a couple of parameters here, the others can be found in the function documentation (`?radial.plot`).

```
# creates a object of six zeros (remember we have 6 groups)
KmCenters[6,]<- c(0)

# this reduces the size of grid and axis labels in upcoming plots
par(cex.axis =  0.8, cex.lab = 0.8)

# creates a radial plot for the two columns
# moves the grid labels to position 3 (centre to top)
radial.plot(KmCenters[c(1,6),], labels =  colnames(KmCenters),
            boxed.radial = FALSE, show.radial.grid = TRUE,
            line.col = c("blue", "red"),  radlab = TRUE,
            rp.type = "p", show.grid.labels = 3)
```



If you want to title the charts you can include , `main = <>name of your cluster<>` within the code. You can now repeat this for the other cluster groups. Can you create meaningful labels for your groups based on their characteristics?

Mapping the clusters

For more an introduction on how to create maps in R please see our Mapping Maps in in R practical from the Introduction to Spatial Analysis and Visualisation in R tutorial series.

First, let's export the classification lookup.

```
#Extract join the cluster labels to the first column of the pop_data file (OA codes)
Classification <- as.data.frame(cbind(as.character(pop_data[,1]), KmClusters[,1]))
#we need to rename the column headers
names(Classification) <- c("oacode", "Classification")
```

```
#now lets export it as a CSV
write.csv(Classification, "Classification.csv")
```

We need to (install and) load the packages which allow us to handle spatial data in R. Next we will load the output area shapefile. Download the shapefile from the CDRC website, unzip the data and ensure each component file is in your working directory. The code below will load the output area boundaries for London.

```
library("sp")
library("rgdal")
library("rgeos")
```

```
#Load the output area shapefiles
Output.Areas<- readOGR(".", "OA_2011_London_gen_MHW")
```

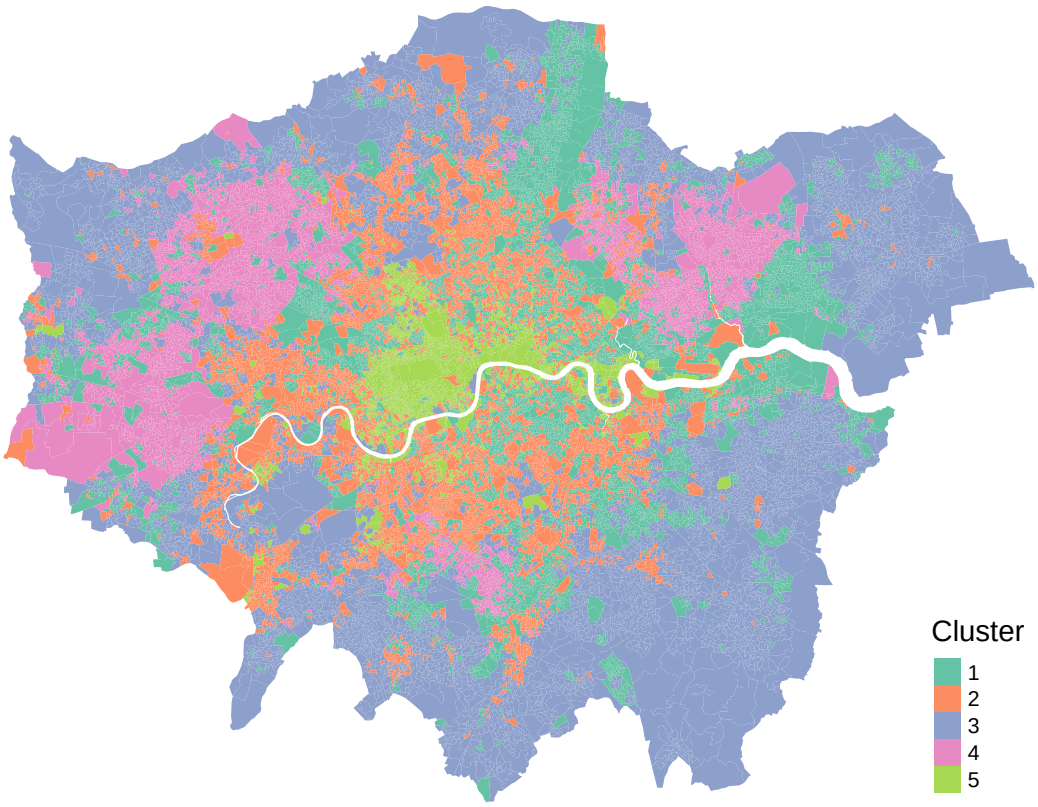
```
## OGR data source with driver: ESRI Shapefile
## Source: "C:\Users\Guy\Documents\Teaching\CDRC\Geodemographics", layer: "OA_2011_London_gen_MHW"
## with 25053 features
## It has 17 fields
```

And now we need to join our classification to the shapefile.

```
#Load the output area shapefiles
OA.Class<- merge(Output.Areas, Classification, by.x = "OA11CD", by.y = "oacode")
```

We will now map the data using the tmap package. A basic introduction to using the tmap package is provided in our Mapping Maps in in R tutorial.

```
library(tmap)
# creates a map in R
tm_shape(OA.Class) + tm_fill("Classification", palette  =  "Set2",
                             style  =  "quantile", title = "Cluster") +
  tm_layout(frame = FALSE)
```



We can make the maps interactive by switching tmap to view mode before running the map code. By using this fuction you can zoom in the explore neighbourhoods you might be familar with.

```
library(leaflet)
# turns view map on, enter tmap_mode("plot") to turn it off
tmap_mode("view") # now simply run the code to create your map again
```

Naming the clusters

Formulating names for geodemographic groups is not easy and be contentious. Essentially the names are derivative of the researcher's perceptions of the cluster centres and sometimes also consider their geographic distributions. Names should be informative, short and memorable. They should also not make unnecessary assumptions or be derogatory in any way. Look at your radial plots, can you derive names for your clusters? It is not uncommon for one of your cluster groups to represent the typical traits of the entire dataset. This group would, therefore, have very moderate Z-scores.

As an example, the names and pen portraits for the supergroups, groups and subgroups of the 2011 Output Area Classification can be downloaded from this link.

You can create labels by using the `gsub()` function to find and replace the codes within your data. It can be built into a function as demonstrated in this code below - you would need to tailor it for your own data. The data can then be mapped in tmap.

```
rename <-  function(x){

  x <- gsub("1", "Your Name", x)
  x <- gsub("2", "Your second name", x)
...

  return(x)
}

OA.Class$label <- rename(OA.Class$Classification)
```

As a final step, try renaming your final clusters then mapping the results.

Conclusion

From this exercise, you should have produced a geodemographic classification with a set number of distinctive groups. In almost every case, geodemographic groups should display some spatial pattern within urban areas, this is because of the tendency for social groups to cluster. This tutorial introduced a number of useful techniques, but not all of the possible methods that are used to create geodemographic classifications.

At each of the steps you, the classification builder, had to make analytical methodological decisions. Each step should be very carefully thought through in order to produce the optimum classification to be used for a particular purpose. For instance, the inclusion of a single additional variable may influence which cluster a particular case is assigned to. Therefore, we recommend further reading on the theory and statistics behind techniques used in geodemographic clustering. Some suggestions have been provided below.

References

Arabie, P., Hubert, L.J. and De Soete, G. Eds. (1996). *Clustering and Classification.* Singapore, World Scientific

Debenham, J. (2002) *Understanding Geodemographic Classification: Creating The Building Blocks For An Extension.* Working Paper. School of Geography, University of Leeds.

Everitt, B.S. & T. Hothorn. (2014). *A Handbook of Statistical Analyses Using R (3rd ed.)* Boca Raton, Chapman & Hall

Everitt, B.S., Landau, S. and Leese, M. (2001). *Cluster Analysis 4th Ed.* London, Arnold

Gale, C.G., Singleton, A., Bates, A.G. and Longley, P.A., 2016. Creating the 2011 area classification for output areas (2011 OAC). *Journal of Spatial Information Science*, 12, pp.1-27.

Harris R, Sleight P, Webber R. (2005). *Geodemographics: Neighbourhood Targeting and GIS.* Chichester, UK: John Wiley and Sons.

Kabacoff, R. (2015). *R in Action: Data Analysis and Graphics with R.* Manning Publications Co., Greenwich, USA

Lansley, G and Cheshire, J (2016). *An Introduction to Spatial Data Analysis and Visualisation in R.* CDRC Learning Resources. Online:
https://data.cdrc.ac.uk/tutorial/an-introduction-to-spatial-data-analysis-and-visualisation-in-r

Leventhal, B. (2016). *Geodemographics for Marketers: Using Location Analysis for Research and Marketing.* London, Kogan Page Publishers.

Longley, P.A. (2017). Geodemographic Profiling. *The International Encyclopedia of Geography* Wiley and the American Association of Geographers (AAG).

Longley, P.A., Goodchild, M., Maguire, D.J. and Rhind, D.W. (2010). *Geographic Information Systems and Science.* 4th Edition. John Wiley & Sons

Spielman, S. and A.D. Singleton. (2015). An Open Geodemographic Classification of US Census Tracts." *Annals of the Association of American Geographers.*

Usuelli, M. (2014). *R Machine Learning Essentials.* Birmingham, Packt

http://geogale.github.io/2011OAC/ - this webpage provides additional material and information to aid use of the 2011 OAC, including the R code used to build it

https://www.opengeodemographics.com/ - for details on open geodemographic products made in the UK

https://www.geodemographics.org.uk/ - a comprehensive directory of hand-selected websites for people interested in the application of geodemographics and geo-spatial analysis