**What is Topological Data Analysis Ball Mapper?**

**Part A: Summarising Data**

The rationale for using Topological Data Analysis Ball Mapper (TDABM) presented in the session is that there is value in visualising data. The argument goes back to the seminal contribution of Anscombe (1973). Our TDABM sessions are split into two sections. This first part creates the summaries of the data, the second part builds the TDABM plots themselves.

In this first part we will see how conventional statistics helps us to understand our data. This document supplements the code file that is available on the GitHub repository as Part A.txt and you are strongly encouraged not to try to copy and paste from this document. All code in this file appears in the courier new font

**Preliminaries**

If you have not previously used R then please make sure that you have a version of the software installed on your computer. You may choose whether you wish to use R or R studio. For those who prefer to code in Python then a separate package is available called pyBallMapper. A separate GitHub is available for pyBallMapper and is linked from the GitHub site for this session.

Packages

R uses packages of user developed code to run many of its’ useful functions therefore you will need to install the packages for the session. This only needs to be done once, so if you already have the packages installed you may skip this step.

install.packages("dplyr")

install.packages("BallMapper")

Working Directory

R requires you to set a working directory in which the data is located. This will also be the folder where all output is saved. In the example code the working directory is set to setwd("D://rmef/") but you should set this with the right path for your computer. If you wish you can set the working directory through the menus. On Windows you would select File and then Change dir… , whilst on a Mac you would select Misc and then Change Working Directory.

Before starting you must make sure that the **rmef2023.csv** file is in your working directory.

*For those using Google Colab the instructions are slightly different because Google creates a temporary working directory for you. There you will need to upload the data.*

**Setting Up the R Environment**

For this session we will need two packages and so will use the following code to bring them into the R environment

library(dplyr)

library(BallMapper)

Next we should read in the data using the read.table function. The options are the filename, the fact that the file uses a comma to separate columns, and the fact that the first row is the variable names.

dtx<-read.csv ("rmef2023.csv",header=TRUE)

If you wish to view the data then you can use the following code. Seeing the data like this is good to verify that what you have read in matches with your expectations.

head(dtx)

A screenshot of a computer

Description automatically generated with medium confidence

Figure 1: Screenshot of first 6 rows of the dtx data frame

The first two columns allow linking to other geographic databases and/or shapefiles for the construction of maps. Excellent resources to help with mapping in R are available online.

|  |  |
| --- | --- |
| Variable | Interpretation (All are percentages) |
| geog | Name of the Local Authority District |
| Armed | Respondents employed in the armed forces |
| Deprivation0 | Households with no deprivation as assessed against Income, health, Overcrowding and Education |
| Deprivation1 | Households defined as deprived on one of the four measures |
| Deprivation2Plus | Households defined as deprived on two or more of the four measures |
| HealthVeryGood | Respondents who self-identify as having very good health |
| HealthGood | Respondents who self-identify as having good health |
| HealthLow | Respondents who self-identify as having fair, bad or very bad health |
| Agriculture | Respondents working in the agriculture sector |
| Manufacturing | Respondents working in the manufacturing sector |
| Accommodation | Respondents working in the accommodation and travel sector |
| Married | Households where the owners are married |
| Cohabit | Households where the owners cohabit |
| Single | Households with one adult resident who is single |
| Other | Households with one adult resident in a relationship, widowed or divorced |
| QualNone | Highest level of qualification in household is below secondary school |
| QualLevel1 | 1-4 GCSEs at grade A-C |
| QualLevel2 | 5+ GCSEs at grade A-C |
| QualApprentice | Apprenticeships |
| QualLevel3 | Two or more A-Levels |
| QualLevel4 | University degree or higher – includes professional qualifications |
| QualOther | Includes vocational qualificiations |
| OwnedOutright | Household is owned outright |
| OwnedMortgage | Household is owned with support from a mortgage |
| SocialRental | Household is rented from a social housing agency (e.g council) |
| PrivateRental | Household is rented from a private individual or company |

Table 1: Variables used in this session

Table 1 is a useful reference for all of the variables used in this session. However, there are obviously large correlations between the variables.

Correlations

We may see the correlations using the code:

cor(dtx[,3:ncol(dtx)])

Note that this will be a very large correlation matrix, but you can see within it that there are a lot of correlations above 0.8. BallMapper can work with highly correlated variables, but to avoid any problems of correlation in the statistical analysis of the data we will restrict focus to a subset of variables. Selection is made using the following code (note each block is a single line):

dty<-cbind(dtx[,1:2],dtx$QualLevel1,dtx$Deprivation2Plus,dtx$Manufacturing,dtx$Single,dtx$HealthVeryGood,dtx$SocialRental)

names(dty)<-c("geog","geogcode","QualLevel1","Deprivation2","Manufacturing","Single","HealthLow","SocialRental")

These variables may be edited if you wish to try different combinations of the data.

Finally we create a dummy variable which takes the value 1 if the percentage of households where the highest qualified resident has a university degree or higher. The intuition of this code is that the value 1 is assigned wherever the condition is true.

dty$HL4<-as.numeric(dty$HealthLow>quantile(dty$HealthLow,0.75))

Our data is now ready for analysis. As a final verification the head of dty should appear as in Figure 2

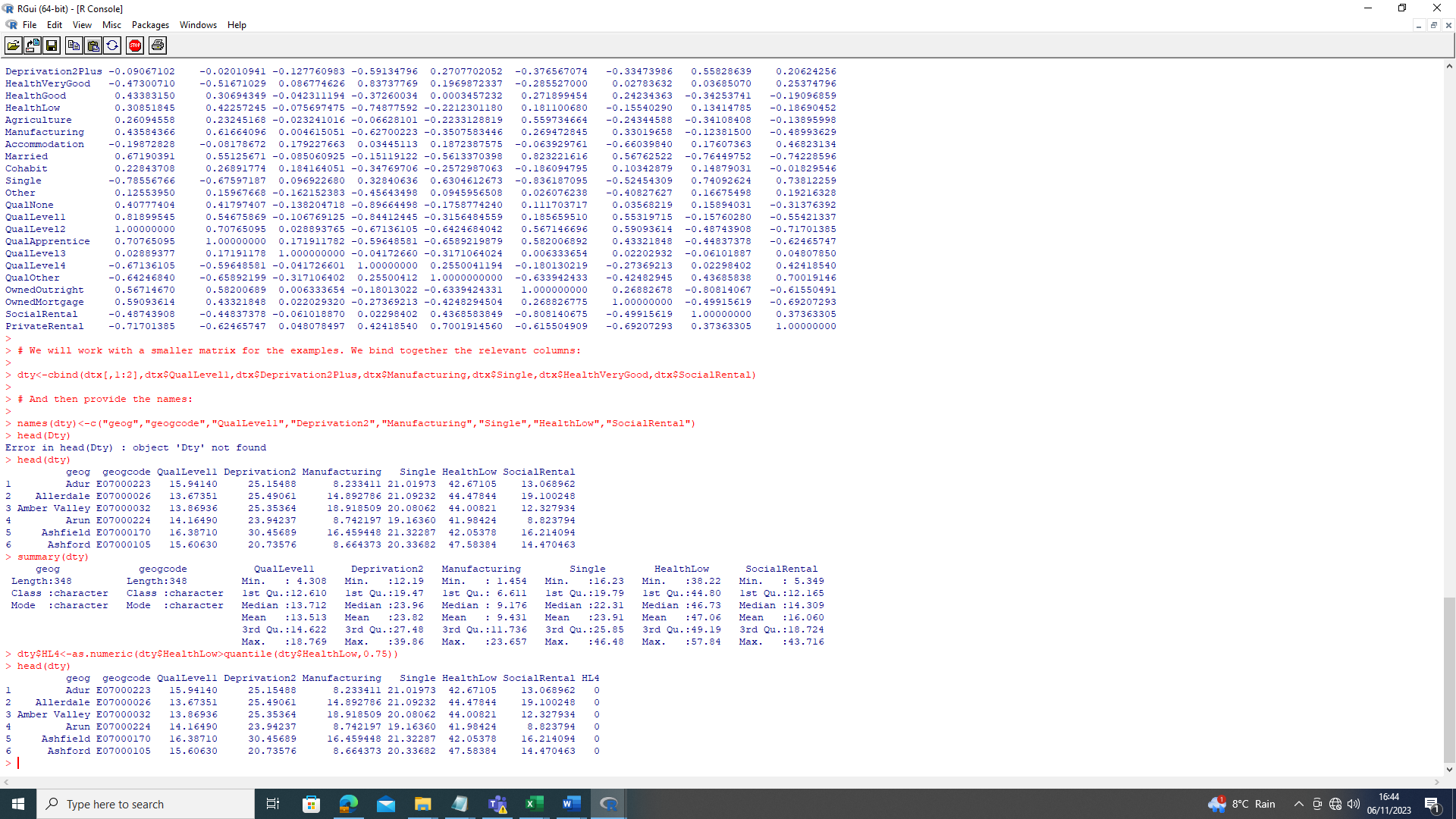


Figure 2: Head of the reduced dataset dty

**Summary Statistics**

As introduced in Session 1, the first step of an empirical analysis is to look at the summary statistics. The precise table used may include quantiles as well as the minimum and maximum. In some applications skewness and kurtosis for the variables may be tested. Here we present a simple summary statistics table which is created using the user defined function sumstats as on the top of the next page.

To run the function paste it from the accompanying text file. Please note the next lines of code will not run without it.

sstatsmat<-function(characteristics,decp){

if(missing(decp)) decp <- 2

a001<-ncol(characteristics)

sstats<-matrix(0,nrow=a001,ncol=5)

for(i in 1:a001){

j<-i

sstats[i,1]<-names(characteristics)[j]

sstats[i,2]<-round(mean(characteristics[,j]),decp)

sstats[i,3]<-round(sd(characteristics[,j]),decp)

sstats[i,4]<-round(min(characteristics[,j]),decp)

sstats[i,5]<-round(max(characteristics[,j]),decp)

}

return(sstats)

}

s001<-sstatsmat(dty[,3:8]) # Creates an object with the summary statistics

s001<-as.data.frame(s001) # Convert to data frame

names(s001)<-c("Variable","Mean","s.d.","Min","Max")

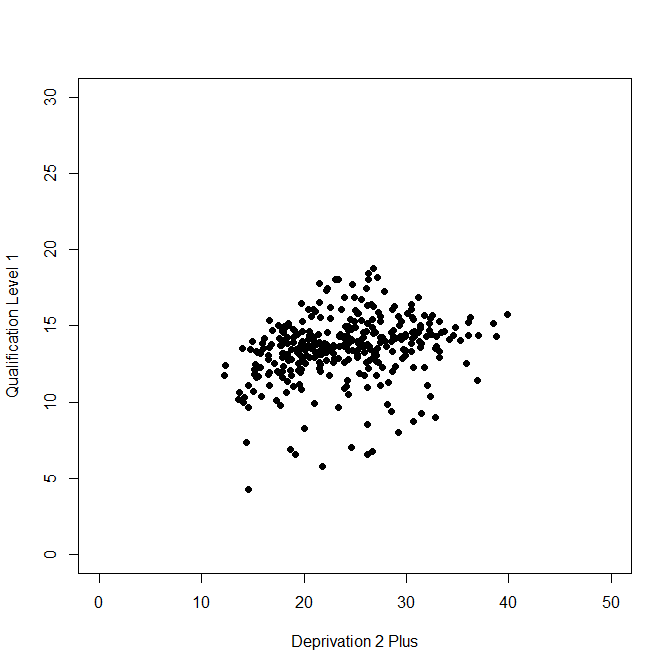
write.table(s001,"summarystats.csv",sep=",",row.names=FALSE)

s001

We now have the basic information about our dataset

Q1: What inference may be derived from the summary statistics?

Q2: What does the correlation matrix tell us about the dty dataset? (Hint: the correlation code for dtx can be adapted from above. A code to allow output to a csv file is included in the code file.)

**Scatter Plots for Visualisation**

The first argument of today’s session is that to understand data we must view its joint distribution. One way to see the distribution is to plot two variables as a scatter plot. Scatter plots are created by default from the plot() function of R. The plotting code below will give us a basic scatter plot. The scatter plot generated is seen in Figure 3.

The options for plotting are the variables to be plotted, the character used for the points (here a solid circle), the ranges for the X and Y axes and then labels for the axes. Much more may be added to enrich the plots but that is left for you to try.

Figure 3: Example scatter plot

The code for producing Figure 3 is:

plot(dty$Deprivation2,dty$QualLevel1,pch=16,xlim=c(0,50),ylim=c(0,30),xlab="Deprivation 2 Plus",ylab="Qualification Level 1")

In Figure 3 the axis limits are set based on the maximum value of the characteristics observed when we constructed the summary statistics. The range on the vertical axis is smaller since the highest percentage of qualification level 1 is just above 20%. The range for the horizontal axis is up to 50%

Our next task is to use the dummy variable QL4 to colour the plot. We use the dummy because colouring by the level variable will create too many subtleties of colour. Again, those interested can look at guides to plotting in R to see ways of colouring the points on colour scales etc.

As a first step to plotting the distribution of health levels on our scatter plots, we will create two subsets from the data, one where the HL4 dummy is 0 and one where the HL4 dummy is 1.

dty0<-subset(dty,dty$HL4==0)

dty1<-subset(dty,dty$HL4==1)

Here we could undertake some two-sample t-tests to test the differences in the five characteristic variables. However, because the choice of 33% was arbitrary there is less value in such tests. In other applications of BM it may be sensible to conduct such tests.

We now produce a scatter plot on the [0,60] scale where the points where the qualifications are low are coloured black and the points with higher proportions of level 4 qualifications are blue. We add a legend in the bottom left to help with the presentation of the plot.

plot(dty0$Deprivation2,dty0$QualLevel1,pch=16,xlim=c(0,50),ylim=c(0,30),xlab="Deprivation 2 Plus",ylab="Qualification Level 1")

points(dty1$Deprivation2,dty1$QualLevel1,pch=16,col="blue")

leg.text=c("Quartiles 1 to 3","Upper Quartile")

legend("bottomleft",leg.text,pch=16,col=c("black","blue"))

A gap is left here because the first line wraps around and should actually be a single line of entry in R. Remember it is always easier to take the code from the accompanying .txt file.

Q3. What does the plot tell us about the relationships between the proportion of households who are classed as deprived on two or more of the four deprivation measures, and the proportion of residents with between 1 and 4 GCSES?

R allows us to output graphs as .png files using the simple png() function. Below each block of code you will also see a second identical block which is prefaced by a png() command and ended with a dev.off() line. This creates a new graphic device in R, plots as instructed and then closes the device. As a result you will not see the graph, just a message that says the graph is complete.

png("Deprivation0Accommodation50.png")

plot(dty0$Deprivation2,dty0$QualLevel1,pch=16,xlim=c(0,50),ylim=c(0,30),xlab="Deprivation 2 Plus",ylab="Qualification Level 1")

points(dty1$Deprivation2,dty1$QualLevel1,pch=16,col="blue")

leg.text=c("Quartiles 1 to 3","Upper Quartile")

legend("bottomleft",leg.text,pch=16,col=c("black","blue"))

dev.off()

The 60 on this file name is to represent the fact that 50 was used as the axis limit. After running the code you will see that a new .png has appeared in your working directory. The .png file can then be pasted into a paper, report etc.

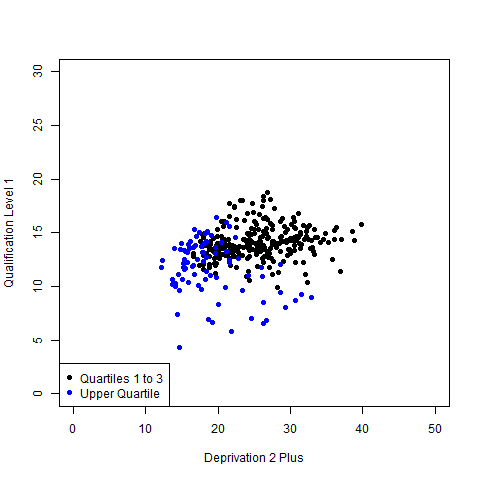
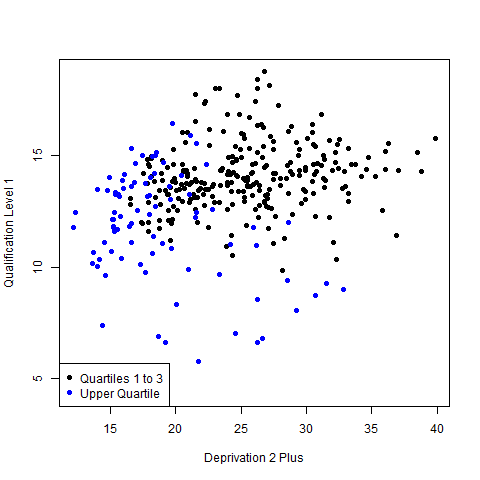


Figure 4: Scatter plots of the proportion of households recording 0 measures of deprivation and the proportion of households where a resident is employed in the accommodation and tourism sector. Left panel with both axes on scale 0 to 60. Right panel with ranges determined by values in the data set.

Figure 4 plots both the fixed range scatter plot and a scatter plot in which the range of values is determined by the values of the two variables independently. Code for producing the second plot requires first drawing all points in the dataset and then overlaying with the colour of the two datasets.

png("Deprivation2Qualification1.png")

plot(dty$Deprivation2,dty$QualLevel1,pch=16,xlab="Deprivation 2 Plus",ylab="Qualification Level 1")

points(dty0$Deprivation2,dty0$QualLevel1,pch=16,col="black")

points(dty1$Deprivation2,dty1$QualLevel1,pch=16,col="blue")

leg.text=c("Quartiles 1 to 3","Upper Quartile")

legend("bottomleft",leg.text,pch=16,col=c("black","blue"))

dev.off()

Q4. What inference may be derived from Figure 4?

Q5. Repeat the analysis with Single and Manufacturing replacing Deprivation0 and Accommodation respectively. What inference can be taken from the resulting scatter plots?

Q6. Repeat the analysis with QualLevel1 and SocialRental as the variables. What is the inference in this third pair of plots?

**Summary**

This first half of the practical session has sought to provide an overview of the dataset using the methods that we would commonly use in statistics. We have seen how to read the data into R, perform simple dataset manipulation and prepare a subset of data for analysis. We then created a summary statistics matrix and a correlation matrix. We outputted the summary statistics and correlation matrix to a csv file so that they may be easily viewed in Excel and then cut and pasted into Word. Finally, we visualised the data using scatter plots. Colouring according to whether the local authority district had a proportion of residents with a university degree or higher was greater than 33% allowed discussion of the relationship between the two axis variables and qualification levels. Part B will introduce Topological Data Analysis Ball Mapper in R and create plots which allow us to see the distribution of qualifications across the joint distribution of local authority district characteristics.

**Part B: Topological Data Analysis Ball Mapper**

**Seeds**

TDABM, like many algorithms in statistics, uses a randomisation process. To ensure that results are reproducible it is a good idea to set the seed in R at the very start of the code file. To set the seed we use the following:

set.seed(1)

**Preparing the Data**

The functions within the BallMapper package all require that the data be in the data.frame format. Hence before we do anything else we will create data.frame for the two outcome variables in which we have interest.

y1<-as.data.frame(dty$HealthLow)

y2<-as.data.frame(dty$HL4)

We also need a data.frame of the variables that are going to be used as our axis variables for the BM plot. These variables are in columns 4 to 8 of the dty data

x1<-as.data.frame(dty[,4:8])

In Part A we noted that the different axes are on different scales, values for Accommodation being much smaller than other characteristics. To overcome this issue in almost all data science applications the first step is to normalise all variables onto the interval [0,1]. The formula for normalisation of observation of variable is:

In the BallMapper package there is a function normalize\_to\_min\_0\_max\_1 which performs the normalisation for us. Therefore we create a new data frame with all values in x1 normalised.

x2<-normalize\_to\_min\_0\_max\_1(x1)

We are now ready to produce our first BM plot.

**A First BM Plot**

The command for producing a BallMapper object has three arguments. The data.frame that will be the axes, the data frame that will be the outcome and the radius of the balls.

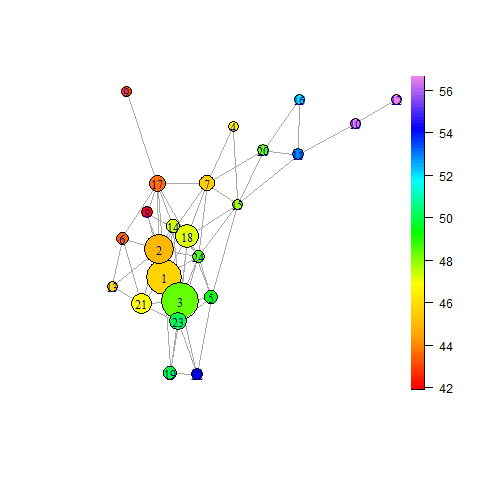
bm1<-BallMapper(x2,y1,0.3)

Here we are creating a BM plot from the normalised characteristics and colouring according to the percentage of residents in the local authority district who have a degree or higher.

To plot the BM we have to use a further command

ColorIgraphPlot(bm1,seed\_for\_plotting=123)

Note here that the seed\_for\_plotting is set to 123. You can change this as it is like a lens through which to view the BM graph. Choosing other values will alter the appearance as if looking at the plot from a different angle. Changing the seed\_for\_plotting can help when the balls are being placed on top of each other in a projection. Note that changing the seed\_for\_plotting does not change any of the sizes, colours and edges.



In Figure 2 we see the first BM plot that has been created. Colouring is according to the percentage of low self-reported health, with the values indicated on the colour scale. Remember that the colouration is the average value for the local authority districts contained within the ball.

We see that there are two areas where the levels of low health are high, and that these sit at the top right and to the lower right. On the left the levels of qualification are lower.

Figure 2: Initial Ball Mapper Plot

**The Role of Radius**

BM has a single parameter, the radius of the balls. We can easily see the effects of the radius by creating new BM plots with different radii. We will first create the BM graphs using the following code:

bm125<-BallMapper(x2,y1,0.25)

bm130<-BallMapper(x2,y1,0.30)

bm135<-BallMapper(x2,y1,0.35)

bm140<-BallMapper(x2,y1,0.40)

bm145<-BallMapper(x2,y1,0.45)

bm150<-BallMapper(x2,y1,0.50)

We may then plot these objects. It is recommended to run all of the plots and then use the up and down arrows to plot again to allow you to compare the different graphs.

ColorIgraphPlot(bm125,seed\_for\_plotting=123)

ColorIgraphPlot(bm130,seed\_for\_plotting=123)

ColorIgraphPlot(bm135,seed\_for\_plotting=123)

ColorIgraphPlot(bm140,seed\_for\_plotting=123)

ColorIgraphPlot(bm145,seed\_for\_plotting=123)

ColorIgraphPlot(bm150,seed\_for\_plotting=123)

By going through the plots we can see that the inference is very consistent. Local authority districts with the highest levels of qualifications do not sit next to each other in the characteristic space considered in our example. The inference is particularly clear from the radius 0.3 case. We therefore use the bm130 plot. The following code outputs the BM graph to a .png file in exactly the way that we did for the scatter plots.

png("LowHealth33.png")

ColorIgraphPlot(bm1,seed\_for\_plotting=123)

dev.off()

Q7. Describe what happens as the radius of the balls increases. Is this what you would expect?

**Colouring by a Dummy Variable**

We now repeat our BM analysis but use the dummy variable QL4 instead of the continuous variable QualLevel4 used so far. When we calculate the colouration of a ball we are now computing the average of a dummy. The colouration of a ball represents the proportion of points within the ball that have the value 1 for the dummy variable.

bm2<-BallMapper(x2,y2,0.33)

bm225<-BallMapper(x2,y2,0.25)

bm230<-BallMapper(x2,y2,0.30)

bm235<-BallMapper(x2,y2,0.35)

bm240<-BallMapper(x2,y2,0.40)

bm245<-BallMapper(x2,y2,0.45)

bm250<-BallMapper(x2,y2,0.50)

Notice now that the ball mappers are labelled as bm2 followed by the radius, with the second argument of the BallMapper() function now being y2 instead of y1. As before we should run the different radii plots and then move through them to carry out a comparison.

ColorIgraphPlot(bm225,seed\_for\_plotting=123)

ColorIgraphPlot(bm230,seed\_for\_plotting=123)

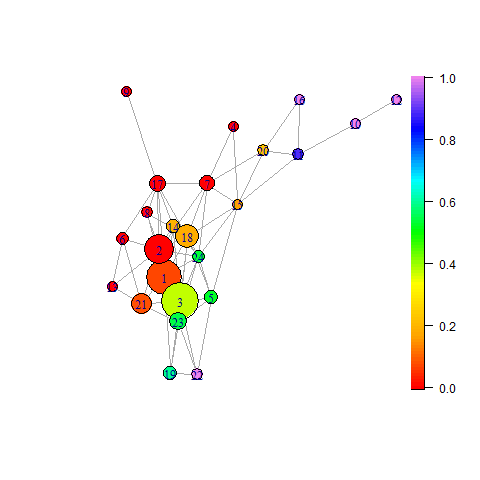
ColorIgraphPlot(bm235,seed\_for\_plotting=123)

ColorIgraphPlot(bm240,seed\_for\_plotting=123)

ColorIgraphPlot(bm245,seed\_for\_plotting=123)

ColorIgraphPlot(bm250,seed\_for\_plotting=123)

Again let us output the plot with radius 0.3, bm230 in this case, as a .png file.



There are some balls in which every local authority district exceeds the 33% of residents with a degree level qualification or higher. The highest proportions appear at the ends of the main shape and in the outliers. On the left we notice that there are many balls in which the percentage of households reporting low health is not in the upper quartile.

It should be remembered that not being in the upper quartile does not mean the local authority does not have households with low self-reported health.

Figure 3: Proportion of districts with more than 33% at Level 4

Q8. Does the colouration in Figure 3 make sense? How does this compare with Figure 2?

Q9. Is the message on proportions consistent across the radii?

The code to produce Figure 3 is:

png("ProportionHealthLow33.png")

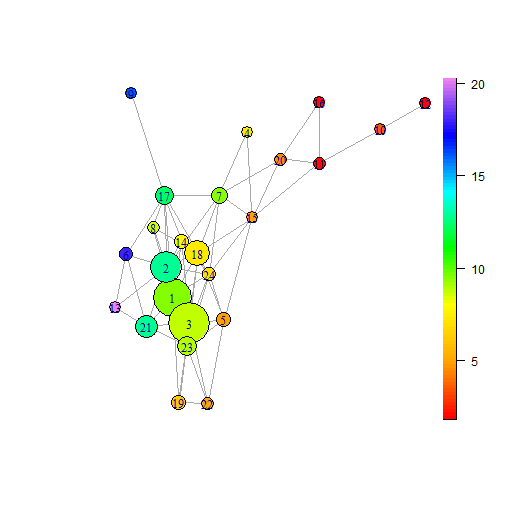
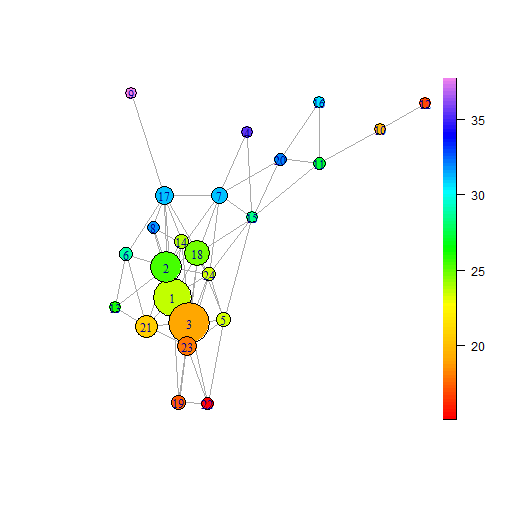
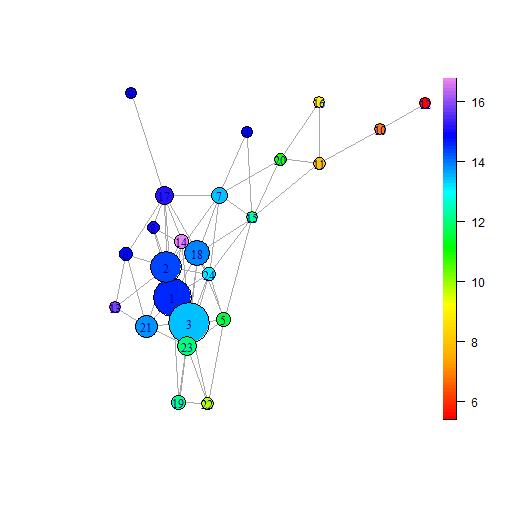
ColorIgraphPlot(bm2,seed\_for\_plotting=123)

dev.off()

The use of proportion 430 this file name is to represent the use of proportions of respondents with qualifications of level 4 or higher. The 30 is the radius used multiplied by 100 to make it suitable for use as a file name.

**Axis Variables**

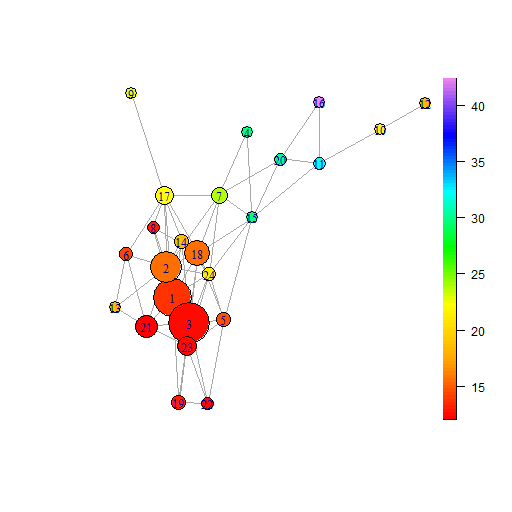
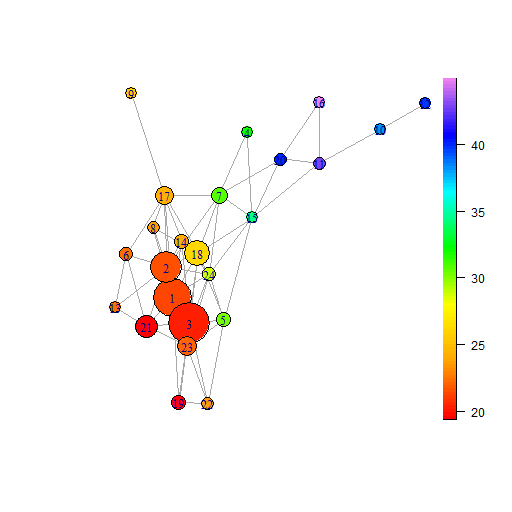
Because the BM graph is an abstract representation of a multi-dimensional dataset there are no axes on the plot. Instead we should colour the plot according to the values of each of the axes. Remember that we used normalised variables for the plot and these will not have meaning when viewed independently. Therefore we will colour by the values in x1 not x2.



(c) Manufacturing

(b) Deprivation 2+

(a) Qualification 1



(e) Social Rental

(d) Single

Figure 4: BallMapper plots coloured by the average value of the five axis variables within each ball. Scales for each plot inform of the value ranges. These plots may be compared with the outcomes in Figures 2 and 3.

Figure 4 provides 5 panels and is generating by editing the output from the colorByAllVariables function:

colorByAllVariables(bm1,x1,"bm1axis")

Here the inputs to the function are a BallMapper object, in this case bm130, the variables that will be used for colouration, in this case x1, and the prefix that will be given to the output files. The code will produce one BM plot for each of the axes and give the file a name with the prefix stated followed by the axis number. Since the first axis variable is Qualification Level 1, the output bm1axis1.png is coloured by Deprivation0.

Q10. What inference may we draw from our BM graph using Figures 2, 3 and 4?

**Points in Balls**

One of the big advantages of BM is that we know exactly which points are in each ball. Therefore if we want to know more about a particular ball we only need to go back to the R output and find the points associated with that ball. In this final section of Part B we will see how to uncover which regions are in each of the balls.

The first step is to prepare the data with a point number. The BallMapper package treats the data it receives in order, with the output recording a point number and a ball number. Where a point appears in the overlap of two or more balls separate lines appear in the output associating that point with each of the balls. Our task is to extract the coverage information. For this we use a user generated function:

points\_to\_balls<-function(l){

a001<-length(l$landmarks)

a1<-matrix(0,nrow=a001,ncol=2)

a1<-as.data.frame(a1)

names(a1)<-c("pt","ball")

for(i in 1:a001){

a<-as.data.frame(l$points\_covered\_by\_landmarks[i])

names(a)<-"pt"

a$ball<-i

a1<-rbind.data.frame(a1,a)

}

a1<-a1[2:nrow(a1),]

return(a1)

}

Adding a column to the dty dataset to allow merging with the points\_to\_balls output is straightforward with the following code:

dty$pt<-seq(1:nrow(dty))

We may now apply the points\_to\_balls function on our BallMapper object bm130, storing the outcome as bmp1.

bmp1<-points\_to\_balls(bm130)

The resulting output is turned into a dataframe and the columns are named for easier merging with the dataset.

bmp1<-as.data.frame(bmp1)

names(bmp1)<-c("pt","ball")

Finally, we use the merge function from R to connect together the data with the points\_to\_balls list.

dtz<-merge(dty,bmp1,by="pt")

It is useful to have this list to query later. As such let us save dtz as a .csv file as follows:

write.table(dtz,"Ballmembers130.csv",sep=",",row.names=FALSE)

We may now choose to switch to a spreadsheet programme like Excel to query the .csv file that has been produced. However, R has the functionality to let us look at the members of each ball. We will do this using the subset function. We look at ball 12 up in the top right.

ball12<-subset(dtz,dtz$ball==12)

The object ball12 is simply the dataset dtz but only containing observations that are in ball 12. Hence, we may simply type ball12 to see the full details of the members. We may also use ball12$geog to just see the list of local authority names.

ball12$geog

ball12

Q11. Repeat the analysis with Ball 22. What can we say about the two ends of the main shape in Figures 2 to 4?

Q13. Which local authority districts are in Ball 13. What can we say about the differences between Ball 13 and Balls 12 and 22?

Q14. How would we construct summary statistics for a ball? (Hint: Refer to Part A)

The process of analysis can continue in many ways, but we leave the discussion here. Session 4 will offer a chance to talk about additional options.

**Summary**

This second half of the practical session has been targeted at giving a basic appreciation of how to use the BallMapper package in R to perform BM analysis. We have continued with the same five variables from Part A and gained further insight into the relationship between the characteristics of local authority districts and the self-reported health of the residents thereof. In many cases simply looking at the data produces new stories that would be of value to social scientists. If you would like to know more about TDABM and the functionality that can be applied, please do not hesitate to get in touch.

**Sources:**

The BallMapper package should be cited as:

Pawel Dlotko (2019). BallMapper: The Ball Mapper Algorithm. R package version 0.2.0. <https://CRAN.R-project.org/package=BallMapper>

The original BallMapper algorithm may be found in the working paper of Dlotko (2019):

Dłotko, P. (2019). Ball mapper: A shape summary for topological data analysis. *arXiv preprint arXiv:1901.07410*.

The data used in this session is taken from Nomis and can be extended and explored through [2011 Census - Census of Population - Data Sources - home - Nomis - Official Census and Labour Market Statistics (nomisweb.co.uk)](https://www.nomisweb.co.uk/sources/census_2011?release=5.1a)