Methods at Manchester November 2025

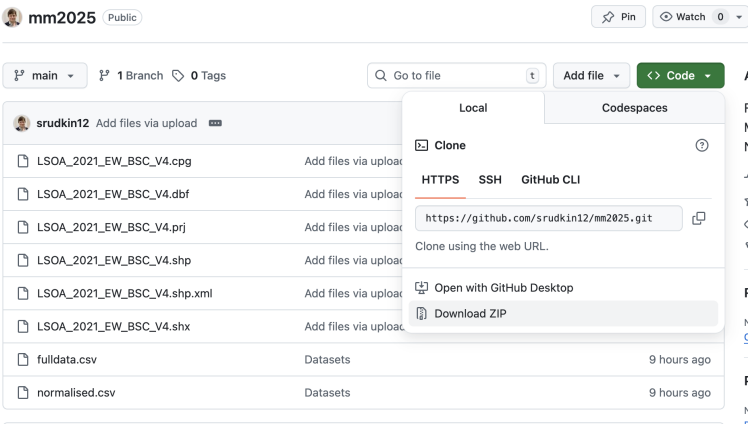
Welcome to the 2025 Methods at Manchester workshop on Topological Data Analysis Ball Mapper (TDABM). For more on the rationales for considering TDABM, you are encouraged to watch the seminar that was given within the Methods@Manchester seminar series https://www.youtube.com/watch?v=Zl24vEufLFw. There are two sessions within the workshop. The first half focuses on the algorithm and the inference that can be drawn from TDABM graphs. After the lunch break, the second half of the workshop provides an example based upon deprivation indicators for Greater Manchester. Our objective across the two sessions is to familiarise you with the ways in which TDABM can be used in your own research.

To help you to follow the sessions, code is provided in both Python and R. All code is provided in Jupyter Notebook form with the extension `.ipynb`. There are two .ipynb files for the applied section, one using Python `mm2025python.ipynb` and one using R `mm2025R.ipynb`. For the first part of the day it is not necessary to be able to use the code, however you may wish to follow along if you are comfortable with Python. The notebook for the first half is called `Session1Python.ipynb`. You should ensure that you have your computer set up ready for when we move into the application.

1 Working with .ipynb Files

To allow the replication of the material both of the sessions, the code is provided in .ipynb files. You can run these files from your own computer. The first step is to create a folder on your computer for the session, for example `/mmtda/` so that you can place all of the necessary files in that folder. You will need to download all of the files on the accompanying GitHub page. If you are happy with the naming then you can simply download everything as a `.zip` file and then extract onto your computer.

Figure 1: Downloading from GitHub



Notes: Screenshot showing the option to download the GitHub as a .zip file. When expanded, this .zip file becomes a folder containing all of the necessary files for the workshop.

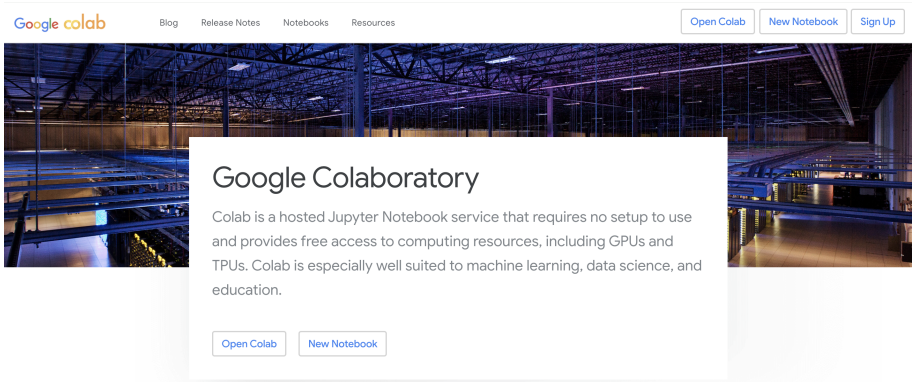
In the first half of the workshop the code creates artificial datasets. For the second half of the workshop, we will be working with dataset is based on the index of multiple deprivation data that was released recently. To ensure that the memory requirements are not too large, a subset of the data, specifically focusing on Manchester Lower Super Output Areas (LSOAs) will be used. The files that have been shared on the GitHub are already subsetted to Greater Manchester. There are two datasets provided. `fulldata.csv` is the main dataset, whilst `normalised.csv` has the variables already normalised onto $[0, 1]$. Because TDABM is a distance-based algorithm, having variables on the same scale is an essential part of the preparation process for any dataset.

From the downloaded files you can launch the .ipynb file directly as long as you have an installation of software that can handle .ipynb files. Examples include the Jupyter Notebook within Anaconda, or a coding suite like Visual Studio. Alternatively you can use a tool like Google Colab and then upload the .ipynb files there to view them. If you do not have either of these then you can view the codes on the GitHub and then copy the relevant lines of code into your R or Python command line.

1.1 Jupyter and Google Colab

We will show the use of the Jupyter notebooks on Google Colab. However, if you wish you can install Anaconda and open the Jupyter notebooks on your local computer. Depending on whether you have easy internet access, you may find that the local version is more convenient. However, if you are working in a university computer laboratory the online version has advantages.

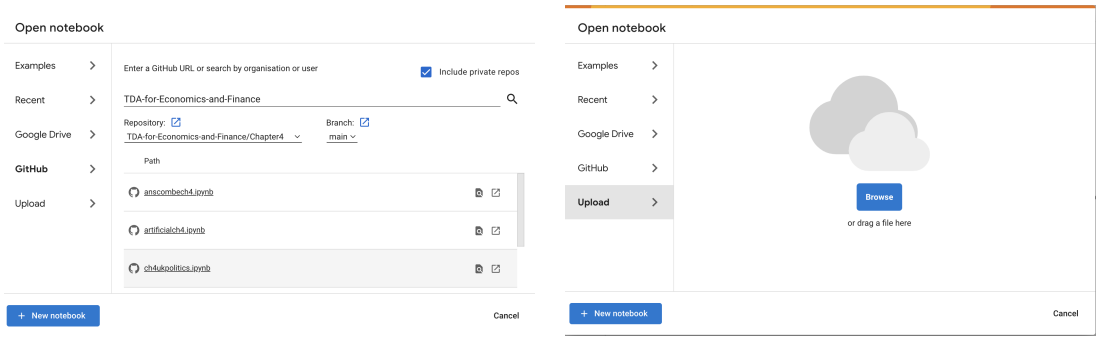
Figure 2: Google Colab



Notes: You can access Google Colab at https://colab.google/.

Figure 2 shows the landing page on Google Colab. Some of you may have an account, and others may wish to create one. For now we will work as a guest on Google colab. Click on `Open Colab` to open a new window with options to upload data. The next screens are shown in Figure 3. The precise screen displayed depends on the location of the file. Google Colab offers the choice to run directly from GitHub, or to upload a file from your computer. Here we will be using the upload option.

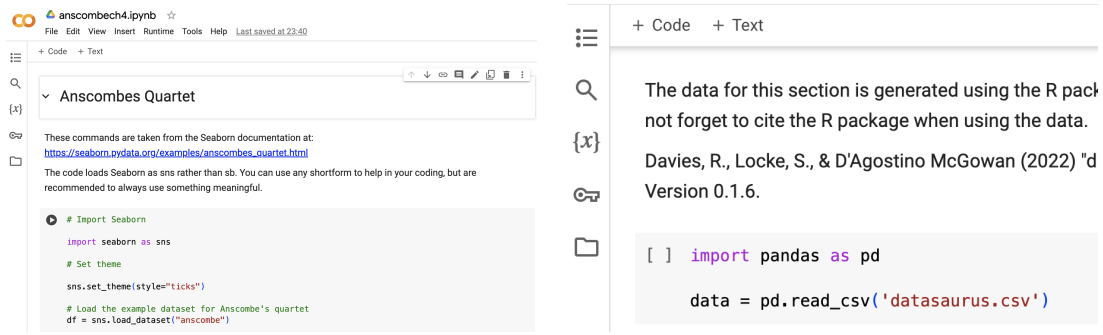
Figure 3: Using Google Colab to Load Files



(a) Loading from GitHub (b) Loading from a local file

Notes: The method to open a `.ipynb` file depends on whether your work is on your computer or whether you are connecting to part of the DATA70302 GitHub. We show the GitHub connection with part of the DATA70302 module in the example.

Figure 4: Using Google Colab with Jupyter Notebooks



(a) Loaded Notebook (b) Where files are required use folder

Notes: Panel (a) shows the Anscombe's quartet notebook which is from DATA70302. The top code block will run once the file has opened. Where a file is needed it must be uploaded to the colab. Upload files with the little folder icon seen in panel (b). Once uploaded the block with `data = pd.read_csv()` will work.

Having loaded a notebook following the approach in Figure 3, the next step is to work through the code blocks. Code blocks are executed by pressing `shift` and `enter` at the same time. It should also be possible to run a cell using the mouse to click on your own screen.

You should save your .ipynb files regularly. It is also important to download your files regularly to ensure that a lost internet connection does not delete your work.

2 Packages

In order to use TDABM, and to plot the results, we will be using a large set of packages. The specific packages used depend on the coding language that you will be using. To accommodate those who could only attend one half of the session, the packages are listed in both of the PDF files.

2.1 Python Packages

In order to use the code you will need to follow the code to add the packages to the appropriate workspace. For those following in Python, the code is placed in green boxes. The relevant .ipynb file is `Session2Python.ipynb`.

```
import numpy as np
 import pandas as pd
 import matplotlib.pyplot as plt
 import networkx as nx
 import pyballmapper as pbm
 import statistics
 import seaborn as sb
 import statsmodels.api as sm
 import random
 import sklearn
import geopandas as gpd
```

To make the TDABM plots fit closer to those generated in R, and seen in Rudkin et al. (2024), we will add alternative coloration. The colors are taken from `matplotlib`

```
from matplotlib.colors import ListedColormap
from matplotlib import colormaps as cm
```

2.2 R

For those who will follow the code in R. The code is placed into blue boxes. We include the `tidyverse` within the R code, but will not code fully with `tidy` commands. You are welcome to make the code more efficient using `tidy` commands if so desired. Because of the use of mapping, it is necessary to include the shapefile package `sf`. There are map plotting options within base R and also through the various geographic packages that are available. This session is not focused on mapping and so we do not argue to adopt the optimal map plotting tools.

```
install.packages(``BallMapper'')
install.packages(``tidyverse'')
install.packages(``sf'')
install.packages(``tmap'')
```

In order to process the outputs of the `BallMapper` package in R, there are a series of user defined functions available. These functions were written by Simon Rudkin as part

of his teaching and are made available via the `additional_functions.txt` within the GitHub. To finalise the setup of your R environment, please paste all of the contents of `additional_functions.txt` into your R command line.

3 Artificial Datasets

The first half of the class is based on a set of artificial datasets. You can follow the code used to create the slides in the .ipynb file on GitHub. Because the slides are produced with Python, only the Python code for the artificial datasets is included.

4 Data

The data that is used for this workshop is taken from the Index for Multiple Deprivation (IMD) which is available to download from the ONS. Data is available at Local Authority District level as well as the Lower Super Output Area (LSOA) level. A link is included within the GitHub.

```
df1 = pd.read_csv('fulldata.csv')
nd1 = pd.read_csv('normalised.csv')
```

```
dt1<-read.csv(``fulldata.csv'')
nd1<-read.csv(``normalised.csv'')
```

To help you to be able to implement the code, the data has been pre-prepared for the session. A single merged .csv file is created within the GitHub folder for expediency. To create the file, the individual `.csv` files from the IMD website were downloaded. Next the variables were renamed following the mapping in Table 1

The set of variables provides many options for analysis. Because we are working with the Lower Super Output Area (LSOA) dataset, there are also many more variables in the census that you may wish to consider. Merging in using the `LSOAcode` variable would be straightforward for any Census 2021 dataset that is at the LSOA level. You can also merge in any LSOA dataset.

The next step is to generate the summary statistics for the data. This is done using the following codes:

The resulting summary statistics can be seen in Table 2. It can be seen that these variables are indeed on different scales. This is why we adopt the normalisation onto $[0, 1]$ before fitting the TDABM code.

We also want to see the correlation structures within the data

From the correlation matrices we can see that there are strong correlations between many of the measures. The domains mostly correlate with the IMD, with the exception of

Table 1: Variables in the Index of Multiple Deprivation Dataset

Group	Name	Description
Identifiers	LSOA_2021	Geographic identification code for the LSOA
	LSOA_NAME_2021	Name of the LSOA
	LAD2024	Local Authority District geographic identification code
	LAD_NAME_2024	Name of the Local Authority District
Indicators	IMD_RANK	Index of Multiple Deprivation Rank
	IMD_DECILE	Decile of the LSOA in the overall IMD
	IDACI_RANK	Income Deprivation Affecting Children Index Rank
	IDACI_DECILE	Decile of the LSOA in the overall IDACI
	IDAOPI_RANK	Income Deprivation Affecting Older People Index Rank
	IDAOPI_DECILE	Decile of the LSOA in the overall IDAOPI
Dimensions	EST_RANK	Education, Skills and Training Rank
	EST_DECILE	Decile of the LSOA in the EST_RANK
	EST	Score for Education, Skills and Training
	CYP_RANK	Children and Young People Rank
	CYP_DECILE	Decile of the LSOA in the CYP_RANK
	CYP	Score for Children and Young People
	ADS_RANK	Adult Skills Rank
	ADS_DECILE	Decile of LSOA in ADS_RANK
	ADS	Adult Skills Score
	BHS_RANK	Barriers to Housing and Services Rank
	BHS_DECILE	Decile of LSOA in BHS_RANK
	BHS	Barries to Housing and Services Score
	GEO_RANK	Geographical Barriers Rank
	GEO_DECILE	Decile of LSOA in GEO_RANK
	GEO	Geographical Barriers Score
	WID_RANK	Wider Barriers Rank
	WID_DECILE	Decile of LSOA in WID_RANK
	WID	Wider Barriers Score
	LIV_RANK	Living Environment Rank
	LIV_DECILE	Decile of LSOA in LIV_RANK
	LIV	Living Environment Score
	IND_RANK	Indoor Environment Rank
	IND_DECILE	Decile of LSOA in IND_RANK
	IND	Indoor Environment Score
	OUT_RANK	Outdoor Environment Rank
	OUT_DECILE	Decile of LSOA in OUT_RANK
	OUT	Outdoor Environment Score

Notes: Variables used within the sessions. These are taken from the Index for Multiple Deprivation. Further explanations of the variables may be found at THIS LINK

```
df2 = df1[['IMD','INC','EMP','EST','HDD','CRM','BHS','LIV','CYP','ADS']]
sdep = pd.DataFrame(columns=['Var', 'Mean', 'SD','Min','q25','q50','q75',
  'Max'], index=range(df2.shape[1]))
for i in range(df2.shape[1])
   sdep.loc[i,'Var'] = df2.columns[i]
   sdep.loc[i,'Mean'] = round(np.mean(df2.iloc[:,i]),3)
   sdep.loc[i,'SD'] = round(statistics.stdev(df2.iloc[:,i]),3)
   sdep.loc[i,'Min'] = round(min(df2.iloc[:,i]),3)
   sdep.loc[i,'q25'] = round(np.quantile(df2.iloc[:,i],0.25),3)
   sdep.loc[i,'q50'] = round(np.quantile(df2.iloc[:,i],0.50),3)
   sdep.loc[i,'q75'] = round(np.quantile(df2.iloc[:,i],0.75),3)
   sdep.loc[i,'Max'] = round(max(df2.iloc[:,i]),3)
```

```
df2<-as.data.frame(cbind.data.frame(df1$IMD,df1$INC,df1$EMP,df1$EST,
   df1$HDD, df1$CRM,df1$BHS,df1$LIV,df1$CYP,df1$ADS))
names(df2)<-c("IMD","INC","EMP","EST","HDD","CRM","BHS","LIV","CYP","ADS")
sdep<-as.data.frame(matrix(0,nrow=ncol(df2),ncol=8))
names(sdep)<-c("Var","Mean","SD","Min","q25","q50","q75","Max")
for(i in 1:ncol(df2)){
   sdep$Var[i]<-names(df2)[i]
   sdep$Mean[i]<-round(mean(df2[,i]),3)
   sdep$SD[i]<-round(sd(df2[,i]),3)
    sdep$Min[i]<-round(min(df2[,i]),3)
    sdep$q25[i]<-round(quantile(df2[,i],0.25),3)
    sdep$q50[i]<-round(quantile(df2[,i],0.5),3)
    sdep$q75[i]<-round(quantile(df2[,i],0.75),3)
    sdep$Max[i]<-round(max(df2[,i]),3)
}
```

```
Corr_Matrix = round(df2.corr(),3)
print(Corr_Matrix)
```

```
Corr_Matrix<-round(cor(df2),3)
Corr_Matrix
```

Table 2: Summary Statistics of Greater Manchester Data

Var	Mean	SD	Min	q25	q50	q75	Max
IMD	28.45	18.89	0.604	11.86	24.96	43.08	82.63
INC	0.283	0.187	0.011	0.111	0.254	0.435	0.812
EMP	0.170	0.098	0.007	0.085	0.154	0.239	0.555
EST	24.576	18.99	0.093	7.611	20.95	38.15	84.58
HDD	0.561	0.802	-1.970	-0.032	0.616	1.149	2.962
CRM	0.490	0.781	-1.950	-0.033	0.563	1.051	2.575
BHS	19.58	9.732	3.044	12.35	17.95	24.48	60.95
LIV	25.29	14.94	0.460	13.50	22.75	34.32	80.33
CYP	-0.012	0.925	-2.612	-0.646	0.083	0.651	2.752
ADS	0.264	0.114	0.022	0.177	0.253	0.344	0.601

Summary statistics for the Greater Manchester Index of Multiple Deprivation 2025 dataset. Variable names can be matched against Table 1. The statistics provided are the mean, standard deviation, minimum, lower quartile, median, upper quartile and maximum.

Table 3: Correlation Matrix

	IMD	INC	EMP	EST	HDD	CRM	BHS	LIV	CYP	ADS
IMD	1.000	0.975	0.962	0.927	0.888	0.841	0.564	0.210	0.771	0.895
INC	0.975	1.000	0.951	0.912	0.830	0.776	0.537	0.164	0.733	0.916
EMP	0.962	0.951	1.000	0.896	0.865	0.791	0.433	0.055	0.768	0.859
EST	0.927	0.912	0.896	1.000	0.810	0.745	0.446	0.125	0.856	0.918
HDD	0.888	0.830	0.865	0.810	1.000	0.832	0.441	0.131	0.812	0.751
CRM	0.841	0.776	0.791	0.745	0.832	1.000	0.404	0.250	0.750	0.682
BHS	0.564	0.537	0.433	0.446	0.441	0.404	1.000	0.116	0.301	0.525
LIV	0.210	0.164	0.055	0.125	0.131	0.250	0.116	1.000	0.066	0.196
CYP	0.771	0.733	0.768	0.856	0.812	0.750	0.301	0.066	1.000	0.694
ADS	0.895	0.916	0.859	0.918	0.751	0.682	0.525	0.196	0.694	1.000

Notes: Correlations between the subdomains of the Index for Multiple Deprivation

the living environment dimension. Correlation between LIV and IMD is just 0.210. To see these correlation structures more clearly we can create scatterplots.

To augment the scatterplots, and begin to say something about the deprivation within Greater Manchester, let us identify the 25% most deprived LSOAs using dummies. The code for the dummies is given in the next box.

```
df1['IMD25'] = (df1['IMD']>df1['IMD'].quantile(0.75))*1
df1['IDACI25'] = (df1['IDACI']>df1['IDACI'].quantile(0.75))*1
df1['IDAOPI25'] = (df1['IDAOPI']>df1['IDAOPI'].quantile(0.75))*1
```

```
df1$IMD25<-as.numeric(df1$IMD>quantile(df1$IMD,0.75))
 df1$IDACI25<-as.numeric(df1$IDACI>quantile(df1$IDACI,0.75))
 df1$IDAOPI25<-as.numeric(df1$IDAOPI>quantile(df1$IDAOPI,0.75))
```

4.1 Exploratory Data Analysis

In order to understand the data further, it is helpful to generate scatterplots. The code for these scatterplots is included in the relevant .ipynb files and you are encouraged to try other combinations of variables and colourations. For brevity, the plots included here are those which also appear in the presentation slides. The plots are generated using the Python code.

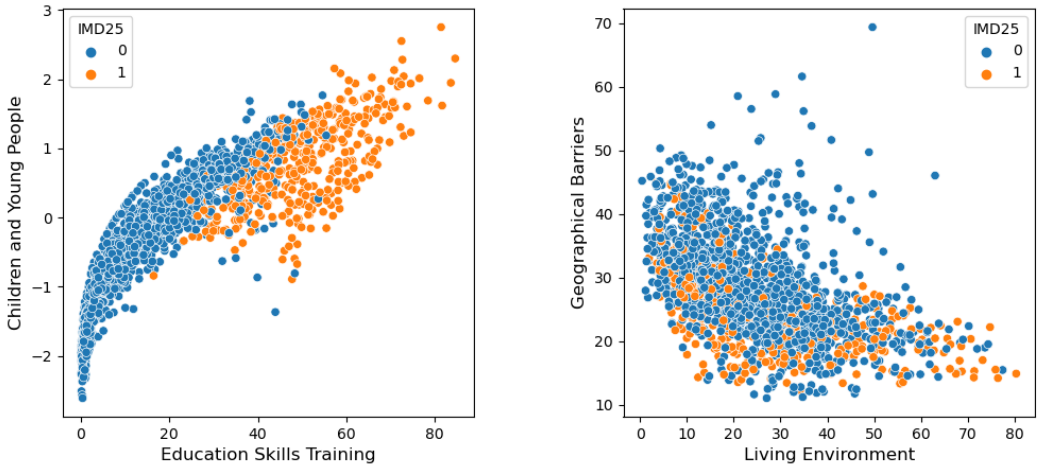
We can see more of the structure of the data by looking at the pairplots. Code for the scatterplots in Figure 5 and the pairplot is included within the Python .ipynb file. R does not have a direct comparison to the pairplot, although you will find that there are codes which attempt to replicate the seaborn offering. In this case we only provide the pairplot codes in the Python .ipynb file.

The pairplots in Figure 6 show that whilst there is a reasonably strong separation of the top 25% on each measure, there are still overlaps between the distributions of points. The leading diagonal shows the overlaps in the density functions. When combining two variables in a scatterplot, the overlaps become stronger. In this workshop we are looking at the picture across multiple dimensions simultaneously.

5 Mapping

For both Python and R, we provide code to generate maps of the Manchester data. These maps are helpful for understanding the spatial context of the data. Because the focus of todays workshop is not on mapping, we omit the detail of the code here and allow users to explore mapping functionalities themselves. Figure 7 provides an example of a map that has been generated in `geopandas` in Python. The yellow areas represent those LSOAs in the top 25% most deprived within Greater Manchester.

Figure 5: ScatterPlot of Greater Manchester IMD



(a) Education and Skills

(b) Living

Notes: Figures plot two measures of deprivation within the index of multiple deprivation. Colouration shows those LSOAs that are in the top 25% of LSOAs in Greater Manchester. Panel (a) is based on the Education, Skills and Training and Children and Young People dimensions. Panel (b) plots lving and the geographical barriers scores.

6 Topological Data Analysis Ball Mapper

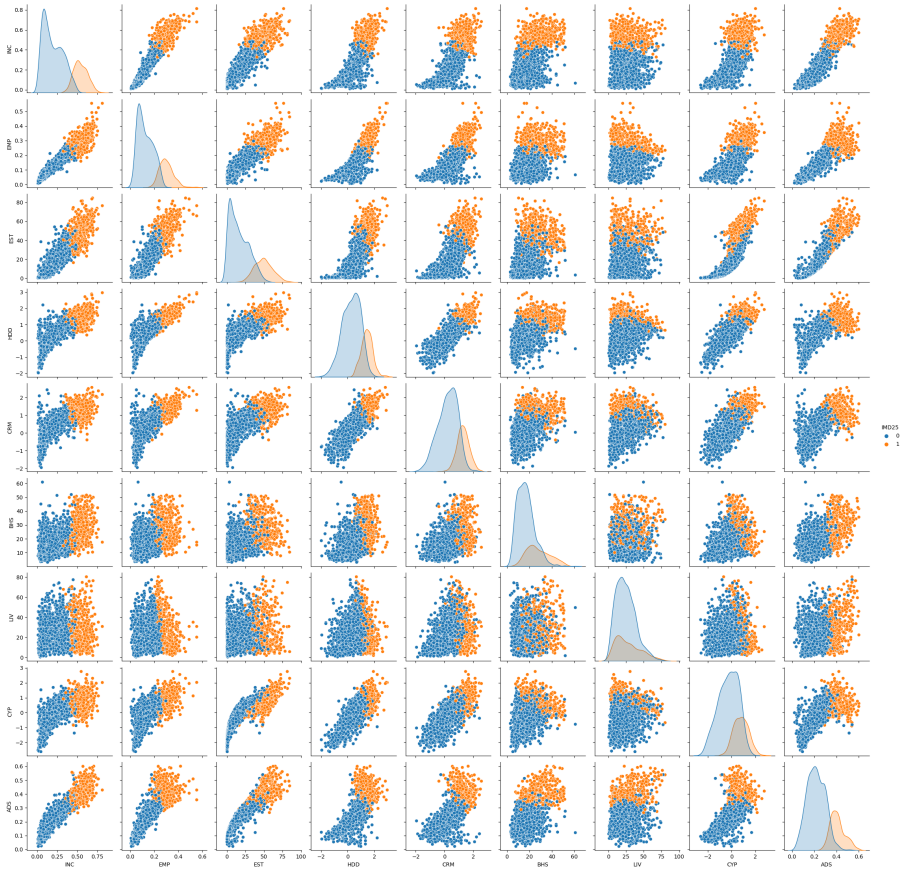
Because TDABM is a distance based method, we need to normalise the data onto a consistent scale. We will use the dataset `nd1` here which has already been normalised. To ensure that there will be consistency with the main dataframe we will sort this according to the LSOA2021 code and then add an index variable. The use of the index variable is to match the points to the balls in which they are contained. Codes for the data preparation ensure that both the reduced and full dataframes are sorted according to the LSOA code.

```
nd1 = nd1.sort_values(by="LSOA2021").reset_index(drop=True)
nd1["pt"] = nd1.index
nd1.head()
df1 = df1.sort_values(by="LSOA2021").reset_index(drop=True)
```

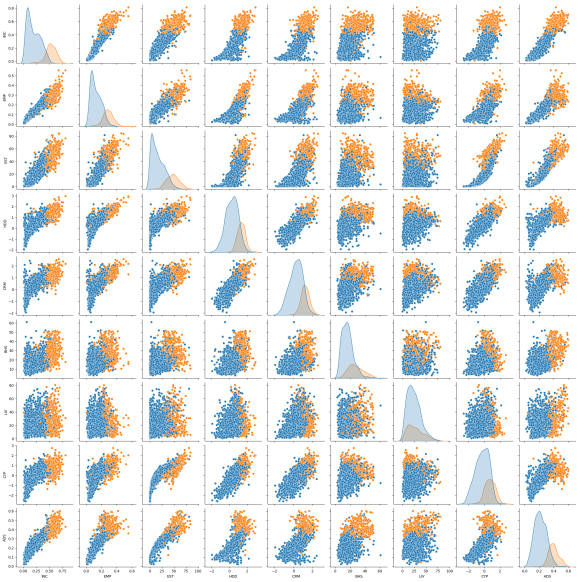
```
nd1$point<-seq(1:nrow(nd1))
```

Production of the TDABM object requires the generation of dataframes. We require a dataframe for the axes and a dataframe for the colouration variable. The code to achieve this is provided in the next set of boxes.

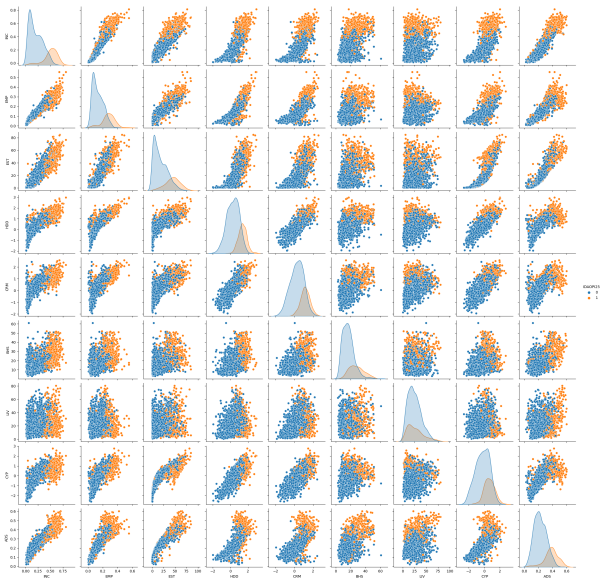
Figure 6: Pairplot using 25% Most Deprived



(a) Top 25% Most Deprived Overall



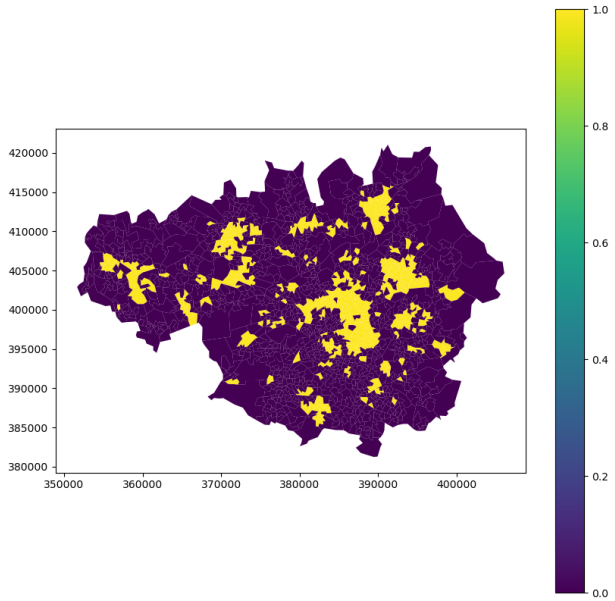
(b) IDACI



(c) IDAOPI

Notes: Figures plot three outcomes based upon index of multiple deprivation. Colouration shows those LSOAs that are in the top 25% of LSOAs in Greater Manchester for each outcome. Panel (a) is based on the overall IMD. Panel (b) shades according to deprivation affecting children index (IDACI). Panel (c) shades according to the deprivation impact on older people (IDAOPI).

Figure 7: Most Deprived LSOAs in Greater Manchester (Top 25%)



Notes: LSOAs in Greater Manchester with shading denoting whether an individual LSOA is in the top 25% of most deprived LSOAs according to the Index of Multiple Deprivation 2025.

```
adf = nd1[['INCN','EMPN','ESTN','HDDN','CRMN','BHSN','LIVN','CYPN','ADSN']]
cdf1 = df1[['IMD']]
cdf2 = df1[['IMD25']]
cdf3 = df1[['IDACI25']]
cdf4 = df1[['IDAOPI25']]
```

```
adf<-nd1[,-1]
adf<-adf[,-10]
cdf1<-as.data.frame(df1$IMD)
cdf2<-as.data.frame(df1$IMD25)
cdf3<-as.data.frame(df1$IDACI25)
cdf4<-as.data.frame(df1$IDAOPI25)
```

We are now ready to generate the BallMapper object. In both Python and R, the `BallMapper` function is used, but Python requires that we specify the package from which the function is taken. The resulting codes are then just one line. We will work with a radius of 0.5 in this example, but you can easily change the value of the radius to see the impact.

```
bm1=pbm.BallMapper(X=adf, coloring_df=cdf1, eps=0.5)
```

```
bm1<-BallMapper(adf,cdf1,0.5)
```

To actually visualise the output of the code we need to make use of a graphing package. In Python, the `networkx` package is used, whilst in R the plotting is a wrapper on `igraph`. In the R installation the `igraph` package is a dependency and so we do not need to load it separately.

```
hsvp = cm.get_cmap("gist_rainbow")
bm1.draw_networkx(coloring_variable='IMD', color_palette=hsvp,
  colorbar=True, colorbar_label="IMD Rank")
```

For brevity the codes for the individual plots are not included within this document, but you can find all of the codes within the accompanying .ipynb files. The output that is generated from the Python code is shown in Figure 8

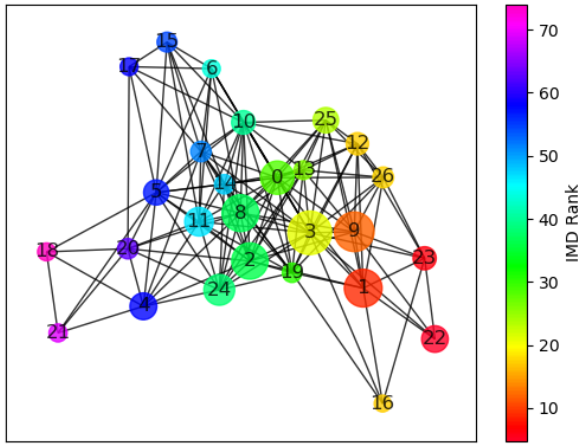
In order to understand the context behind the pictures, it is helpful to be able to extract the ball memberships. This is done using the `points_and_balls` function within Python. For R the user defined function `pointstoballs` is used.

```
pb1 = bm1.points_and_balls()
nd2 = nd1.merge(pb1, on='point')
df5 = df1.merge(nd2, on='LSOA2021')
```

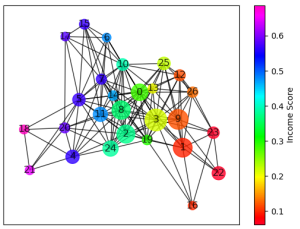
Once these merges are made we have dataframes with 1 line for each point-ball combination. There will be more lines in the dataframe than there are observations since many LSOAs appear in the intersection of the balls. We can export the full list to a .csv file. Here we only keep relevant information including the LSOA name and the Local Authority District name. The overall IMD rank is added, as is the IMD score. The 5 in the filename is added to show that the radius was 0.5.

If we wish to look at a single ball then we can subset and then view the resulting dataframe. If we are just interested in the identity of the ball members then the ball list is subsetted. Maps based on ball memberships can be constructed by adding dummies to `df5` and then merging with the map file. We show this in the .ipynb files.

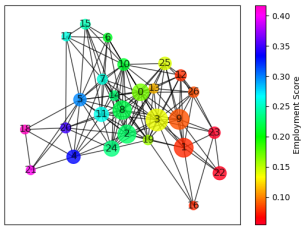
Figure 8: TDABM plot of Index of Multiple Deprivation



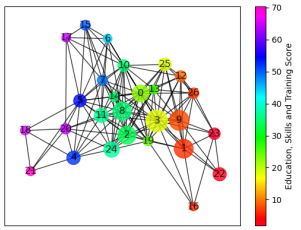
(a) Index of Multiple Deprivation



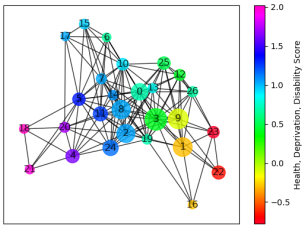
(b) Income



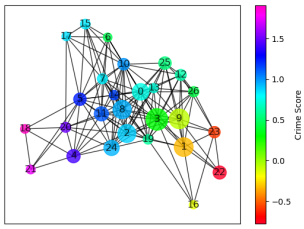
(c) Employment



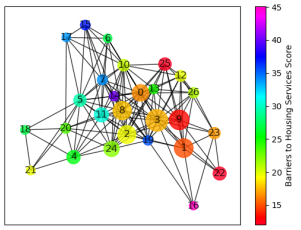
(d) Education



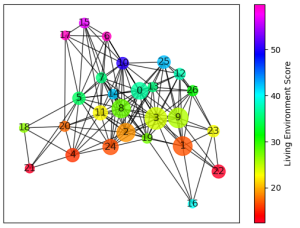
(e) HDD



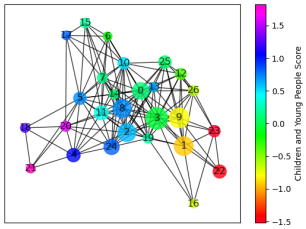
(f) Crime



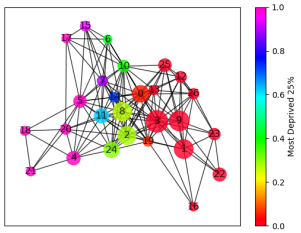
(g) Barriers to Housing



(h) Living Environment



(i) Children



(j) Top 25%

```
ballslist = pd.DataFrame(df5[["ball","LSOA2021","LSOA_NAME_2021",
  "LAD_NAME_2024","IMD_RANK","IMD"]])
ballslist.to_csv('ballslistr5.csv', index=False)
```

```
ball16 = ballslist[ballslist["ball"] == 16]
ball16.head()
```

```
imdm = df5.groupby('ball')['IMD'].mean()
incm = df5.groupby('ball')['INC'].mean()
empm = df5.groupby('ball')['EMP'].mean()
estm = df5.groupby('ball')['EST'].mean()
hddm = df5.groupby('ball')['HDD'].mean()
crmm = df5.groupby('ball')['CRM'].mean()
bhsm = df5.groupby('ball')['BHS'].mean()
livm = df5.groupby('ball')['LIV'].mean()
cypm = df5.groupby('ball')['CYP'].mean()
adsm = df5.groupby('ball')['ADS'].mean()
idacim = df5.groupby('ball')['IDACI'].mean()
idaopim = df5.groupby('ball')['IDAOPI'].mean()
ballmeans = pd.DataFrame([imdm,incm,empm,estm,hddm,crmm,bhsm,livm,
cypm,adsm,idacim,idaopim])
ballmeans = ballmeans.transpose()
ballmeans.to_csv("ballmeans.csv", index=False)
```

We can also generate summary statistics for the balls using the expanded dataframe.

Once we are happy that we have the correct axis variables, and that the colouration is of interest, we can proceed to conduct a more in depth analysis of the TDABM plots. In this case we are just illustrating what can be done and so no detailed analysis is undertaken.

7 Summary

This half of the workshop has been designed to show how the TDABM algorithm is implemented in R and Python. We have seen how to extract information about the membership of the balls. With artificial data there is limited interpretation to the membership. However, when applied on real data there is value in the ball memberships. A ball in TDABM constitutes datapoints which are similar in all dimensions, as we demonstrated with the summary statistics here. The example presented concerns the Index of Multiple Deprivation and the axes have been the subdomains of deprivation. We can see that there are clear linkages between the dimensions and the overall index, but that the top 25% of the index, and the areas where children or older people are impacted most, do not overlap. The TDABM plots have allowed us to extract more information about these areas.

References

Rudkin, S., Barros, L., Dłotko, P., and Qiu, W. (2024). An economic topology of the brexit vote. *Regional Studies*, 58(3):601–618.