An Introduction to Topological Data Analysis Ball Mapper in R

Simon Rudkin *[1]

[1]School of Social Sciences, University of Manchester, United Kingdom

Abstract

The Topological Data Analysis Ball Mapper (TDABM) algorithm of Dłotko (2019) provides a model free means to visualize multi-dimensional data. The visualizations are abstract two-dimensional representations of covers of the dataset. To construct a TDABM plot, each variable in the dataset should be ordinal and suitable for representing as an axis of a scatter plot. The graphs produced by TDABM provide a map of the dataset on which outcomes may be charted, models assessed and new models formed. The benefits of TDABM are powering a growing literature. This document provides a step-by-step introduction to the algorithm with code in R.

Keywords: Topological Data Analysis, Ball Mapper, Code

1 Introduction

Consider a dataset X formed of N observations on K variables. The dataset is also equipped with a further variable, Y, which has a numeric value for all N data points. Hence point i, $i \in \{1, N\}$ the values of the K variables are x_{ik}, $k \in \{1, K\}$. Data points can then be plotted as a point cloud where each axis of the point cloud represents one of the K variables. The location of a point in the K-dimensional space is defined by the values of x_{ik}. Topological Data Analysis (TDA) is based upon measures of the point cloud. Where the point cloud has more than two dimensions, $K > 2$, the visualization of the point cloud becomes difficult. This short guide explains the use of the Topological Data Analysis Ball Mapper (TDABM) algorithm of Dlotko (2019) using the *BallMapper* package in R[1].

Sources of data for the point cloud are many. There may be K different variables as used in Rudkin and Dlotko (2024) study of the digital divide at the dawn of the Covid-19 pandemic. The variables may be drawn

*Full Address: Department of Social Statistics, School of Social Sciences, University of Manchester, Oxford Road, Manchester, M13 9PL, United Kingdom. Email:simon.rudkin@manchester.ac.uk

[1]Throughout reference is made to the R computing language (R Core Team, 2023). The code accompanying this guide is available at https://github.com/srudkin12/BM-Guide.

as multiple regional aggregates on the same topic as in Rudkin et al. (2024a) study of voting patterns in the United Kingdom referendum on European Union membership. Otway and Rudkin (2024) uses a similar regional aggregate approach to study voting behaviours in the UK general elections. The K variables may be lags of the same time series, as seen in Rudkin et al. (2024b) plotting of the return trajectories of Bitcoin or Rudkin and Webber (2023) study of regional economic performance trajectories. Applications in finance include the consideration of credit risk across a space of financial ratios (Qiu et al., 2020), and stock returns across a joint distribution of firm characteristics (Dłotko et al., 2024).

The rationale for mapping the multidimensional space in a single plot is provided by the same arguments on data visualization as made by Anscombe (1973) and Matejka and Fitzmaurice (2017). Whether the aim is to develop an understanding of the structure of the data that is not given in the first and second moments, or understanding statistical models, the ability to visualize data is core. Although this guide does not provide the specific tools to evaluate models, the ability to map is an important first step. Readers are referred to the relevant methodological papers for more. See for example, Dłotko et al. (2024) and Rudkin et al. (2024b) for evaluation of model fit, and Rudkin et al. (2024b) for a naive forecaster based on TDABM. A useful discussion on the choice of parameters in TDABM is provided in Dlotko et al. (2022).

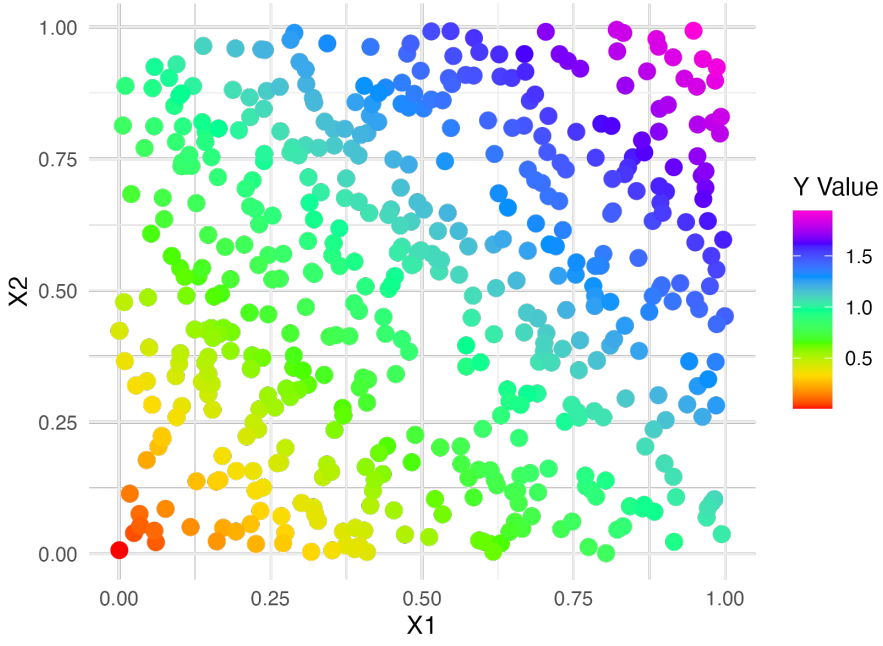
There are many alternative means for visualizing multivariate data. Alternatives include the original mapper algorithm of Singh et al. (2007), which is also topologically inspired. The original mapper requires a lens function, clustering algorithm and selection of number of clusters. A research agenda is exploring the choices for mapper, but there is still no definitive means for selecting the choice elements. The mapper graph can be very unstable and hence the single choice parameter in TDABM is preferable. Other methods such as Principal Components Analysis or T-SNE require the application of dimension reduction. Because TDABM does not generate data loss, TDABM provides a preferable means to visualize the full structure of the data. This guide focuses on TDABM and does not apply the alternative methods to the example data.

The remainder of the guide is organised as follows. Section 3 introduces the method from a theoretical standpoint. Section 2 introduces the artificial dataset used in this guide. Section 4 provides an explanation of the way in which the TDABM algorithm is implemented in R. Section 5 discusses the next steps and concludes.

2 Data

This note will use the same dataset for the demonstration of the TDABM algorithm. The dataset has $N = 500$ data points with coordinates in $K = 2$ dimensions. The variables are denoted as X_1 and X_2. Both variables are drawn independently form a uniform distribution on $[0, 1]$, $X_1 \sim U[0, 1]$ and $X_2 \sim U[0, 1]$. For

Figure 1: Raw Dataset



Notes: Dataset has $N = 500$ observations with $K = 2$ variables. Each variable is drawn at random from a uniform distribution such that $X_1 \sim U[0, 1]$ and $X_2 \sim U[0, 1]$. The outcome variable has $y_i = x_{i1} + x_{i2}$. Points are colored according to Y.

each data point there is an associated outcome variable Y where $y_i = x_{i1} + x_{i2}$. The resulting dataset is plotted in Figure 1.

The coloration has a natural pattern with highest values occurring in the North East of the plot. Meanwhile the lowest values are observed to the South West. The 2-dimensional nature of the dataset means that we can see the full pattern of the data without requiring any advanced methods. In reality, there is limited motivation to use a visualisation algorithm like TDABM to represent a 2-dimensional dataset. However, the ability to plot the $K = 2$ case on a simple scatter plot allows the opportunity to demonstrate the algorithm clearly.

The decision to use the uniform distribution with the same minimum and maximum values as the axes means that there is no need to rescale the data prior to running the TDABM algorithm. Like many Machine Learning methods, TDABM is a distance based approach to analyzing data. Where variables exist on different scales then normalisation must be performed prior to running the TDABM algorithm on the data.

The code which is used to generate this dataset in R is provide in Box 2.1. The code demonstrates the simplicity of the construction of the dataset for this example document. Inclusion of Box 2.1 removes ambiguity about the data in the discussion that follows. The setting of the random seed is very important to ensure that reproduction of the examples in this guide.

Box 2.1: R Code for Dataset

This code will produce a bivariate dataset with two uniformly distributed variables and 500 observations:

```
set.seed(123)  # For reproducibility
x1<-runif(500,0,1)
x2<-runif(500,0,1)
```

We bind the two variables together to produce a `data.frame` object:

```
df1<-as.data.frame(cbind(x1,x2))
names(df1)<-c("X1","X2")
```

3 Methodology

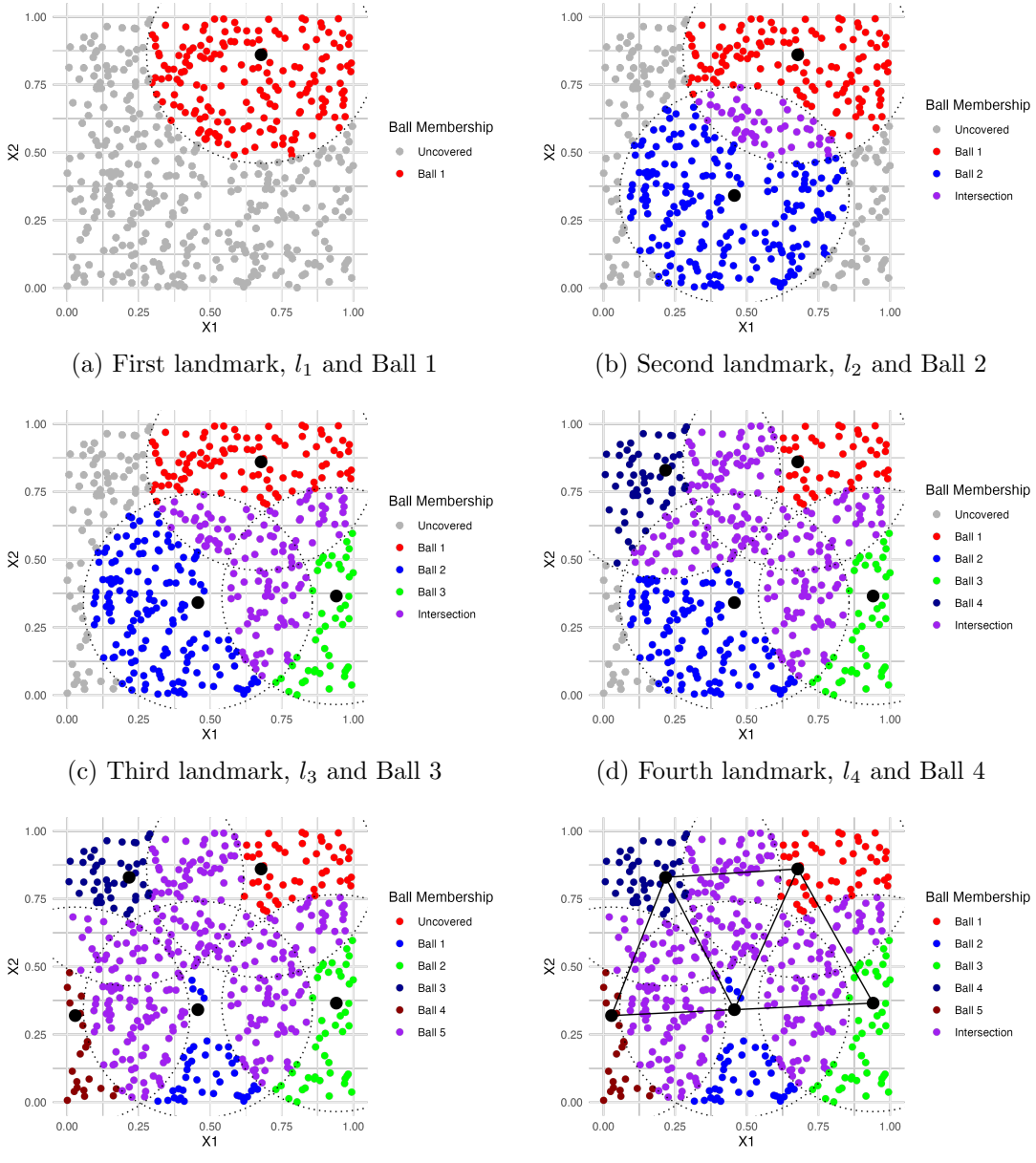
The TDABM algorithm begins with a dataset X with K variables. The data is converted into a point cloud in which the location of each data point i is determined by the value of x_{ik} on axis k, $k \in \{1, K\}$. The point cloud for the example dataset is represented in Figure 1. The single choice parameter for the TDABM algorithm is the radius of ball to use, ϵ. There is currently no algorithm with which to choose the optimal radius. As with maps of geographic space, there are times when the interest is in local structure, and cases where the interest is in the global structure of the data. The TDABM equivalent is to use a small ϵ to see local structures and a large ϵ to see the overall structure of the data. It is recommended that users consider many radii when looking at their data.

Step 1 is to choose a point at random from the dataset. This point becomes the first landmark of the TDABM plot. The landmark is labelled l_1. A ball of radius ϵ is drawn around l_1. Label this ball $B_1(X, \epsilon)$ All of the points within the ball are considered as being covered by ball 1. Panel (a) of Figure 2 shows l_1 as a black point. Ball 1 is shown as a dashed line. The points covered by Ball 1 are shown in red.

Step 2 is to select a second landmark, l_2, from the set of uncovered points. In panel (a) of Figure 2, the uncovered points are colored grey. The second selected data point is shown as a black dot in panel (b) of Figure 2. Again a ball of radius ϵ is drawn around l_2. There is now a second ball $B_2(X, \epsilon)$ to add to the set of balls. The combined set of balls can now be defined as $B(X, \epsilon)$. Panel (b) of Figure 2 shows both balls and their intersection. Points in the intersection are in both balls 1 and 2. Uncovered points continue to be shown in light grey. As the coloration of the balls attests, R is aware which datapoints are within which ball.

Step 3 is the choice of a third landmark, l_3 from the uncovered set. Again a ball, $B_2(X, \epsilon)$, is created around l_3. There are now more points which are in the intersections of the balls. Panel (c) of Figure 2 shows the construction with ball 3 present. The algorithm continues to select landmarks at random from

Figure 2: Raw Dataset



(a) First landmark, l_1 and Ball 1

(b) Second landmark, l_2 and Ball 2

(c) Third landmark, l_3 and Ball 3

(d) Fourth landmark, l_4 and Ball 4

(e) Fifth landmark, l_5 and Ball 5

(f) Edges for non-empty intersections

Notes: Demonstration of the construction of a TDABM plot with random selection of landmark points. The dataset, X, comprises two variables, X_1 and X_2, which are drawn independently from $U[0,1]$. In each case a ball is drawn around a landmark point and all points within that ball are added to the covered set. Light grey points represent the uncovered set. After the selection of the 5th landmark, all points are covered. Edges are drawn between any pair of landmarks with a non-empty intersection.

the uncovered set until all points are covered. Figure 2 shows the process with the selection of landmark l_4 in panel (d) and finally l_5 in panel (e). As each ball is added the set of light grey uncovered points shrinks.

The balls have numbers owing to the order in which they are created. Ball 1 is the ball surrounding the first landmark selected, etc. Because the ball numbers link to the random selection of points, they have no interpretation as numbers. Ball 1 has no superiority, or inferiority, to Ball 2 by fact of having a lower ball number. The numbers are provided to discuss the balls only. For example, if discussing the members of a single ball, it is known that those data points are similar in X. The ball number facilitates the discussion of the collection of similar points. Ball numbering gains further relevance when the balls are given a coloration, as is discussed later in this guide.

In order to represent the dataset further it is necessary to know the relative location of the balls. The final essential step of the algorithm is to draw edges between the landmarks of balls that have a non-empty intersection. Figure 2 shows the network constructed by the edges within panel (b). We end the theoretical demonstration of the TDABM algorithm here.

4 Ball Mapper as Implemented in R

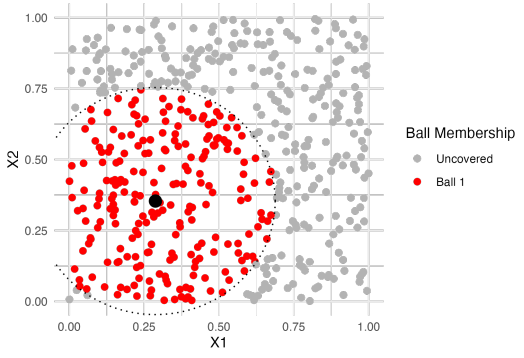
There are small differences in the construction of the TDABM graph in R, compared to the theoretical construction. The differentials are explored in the first subsection here. The next step is to consider the inference that can be drawn from TDABM graphs. A subsection progressing from the basic network to the TDABM representation used in the literature is provided.

4.1 Landmark Selection and Graph Construction

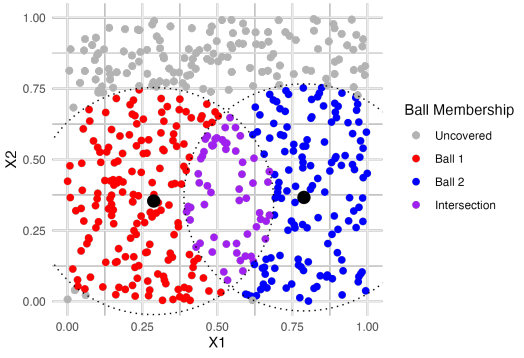
The R implementation of the TDABM algorithm is performed using the package `BallMapper` by Dlotko (2019). To allow the user to randomise the selection of landmarks, the R algorithm simply chooses the data point with the lowest index that is still within the uncovered set. Hence with a given seed you will always get the same landmarks. Likewise, by shuffling the order of the input data it is possible to obtain a different TDABM representation of the data. In this section I repeat the process from Figure 2, but by using first uncovered point as the landmark. Figure 3 provides the results.

Figure 3 shows the selection of the first landmark point, l_1 in panel (a). The points contained within $B_1(X, \epsilon)$ are shown in red. Panel (b) has the addition of the second landmark. We see the points which are only in $B_2(X, \epsilon)$ colored blue. The points in the intersection of the two balls are colored purple. This is the same as in Figure 2, but the landmarks selected are different. Continuing, the algorithm selects l_3, l_4, l_5, l_6 and l_7. Panels (c) to (g) show how the balls cover the full dataset. Panel (h) has the edges between any pair

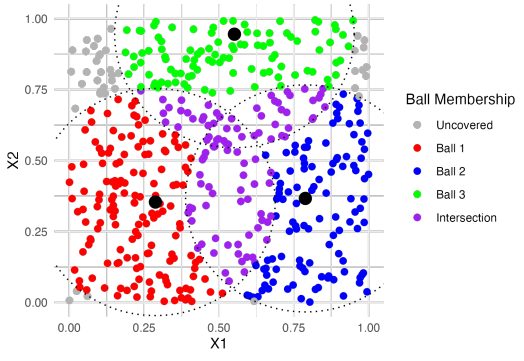
Figure 3: Selection by First Uncovered Point



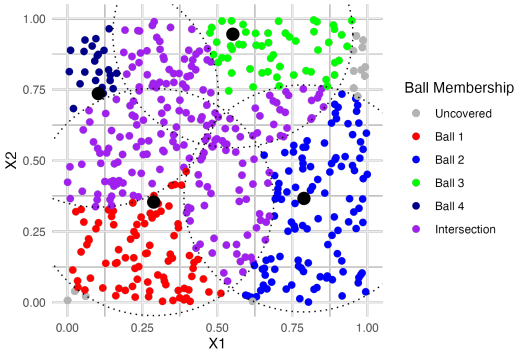
(a) First landmark, l_1 and Ball 1



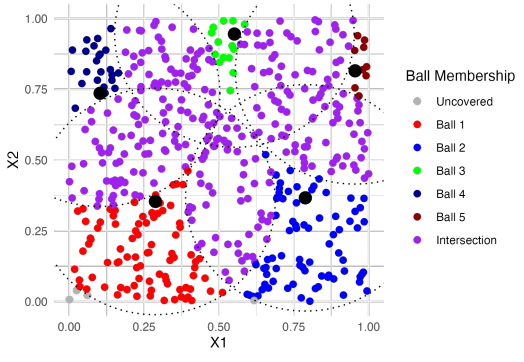
(b) Second landmark, l_2 and Ball 2



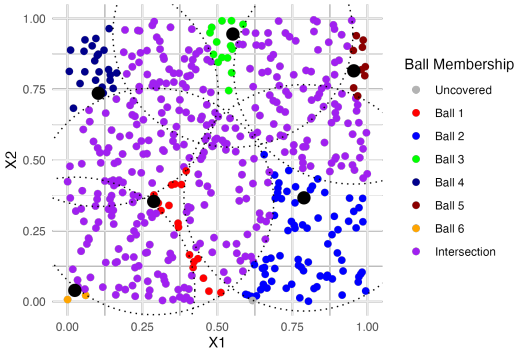
(c) Third landmark, l_3 and Ball 3



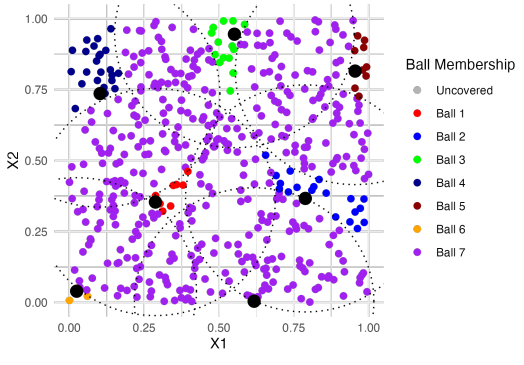
(d) Fourth landmark, l_4 and Ball 4



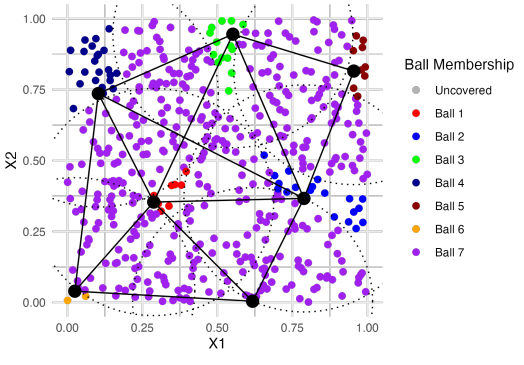
(e) Fifth landmark, l_5 and Ball 5



(f) Sixth landmark, l_6 and Ball 6



(g) Seventh landmark, l_7 and Ball 7



(h) Edges connecting non-empty intersections

Notes: Demonstration of the construction as performed in the `BallMapper` package in R (Dlotko, 2019). Data is 500 points with two variables, X_1 and X_2. X_1 and X_2 are drawn independently from $U[0,1]$. Panels (a) to (g) show the iterative selection of the point from the uncovered set with the lowest ID in the uncovered set. In each case the points solely within one ball are colored to indicate their membership. Intersection points are colored purple. Panel (h) adds edges between the landmarks that have non empty intersections. Where relevant, light grey coloration represents the uncovered set. IDs are in ascending order for the dataset, X, which is supplied to the algorithm.

of landmarks that have a non-empty intersection. In this case the number of balls needed to cover the full dataset is 7. The 7 compares to just 5 balls in the previous case. Dlotko et al. (2022), Rudkin et al. (2024a), Rudkin et al. (2024b) and Rudkin and Dlotko (2024) all show that there are small variations in the number of balls obtained when repeating the algorithm multiple times. However, in all cases the inference drawn is consistent.

Within the code for each step, the first task is to identify the uncovered points. Secondly, the distances between all data points and the selected landmark point are calculated. The points within the radius become the points in the ball that is being constructed. An example for constructing Ball 6 is seen in Box 4.1. A dataset of landmarks, here called *sel* is appended with the new landmark, l_6. The next block of code is designed to isolate all of the intersections to create the purple coloration on the illustration. Finally, the `member` dummy is created for any point which is either in one of the balls, or in the intersection of more than one ball. There may be a more efficient way to obtain a simulation of the TDABM algorithm. However, the aim here is not to provide efficient code as I am instead to explain what the R function `BallMapper` from Dlotko (2019) is doing under the hood.

Box 4.1: R Code for Imitating Creation of Ball 6

When selecting a landmark, R will look for the lowest id in the uncovered set:

```
unc<-subset(df1,df1$member==0)
minid<-min(unc$pt)
```

The landmark can be identified in the dataset and distances to that point measured:

```
selected_point6 <- df1[minid, c("X1", "X2")]
df1$distance <- sqrt((df1$X1 - selected_point6$X1)^2 + (df1$X2 -
    selected_point6$X2)^2)
df1$within_radius6 <- ifelse(df1$distance <= 0.4, 1, 0)
```

Convert the selected landmark point into a dataframe

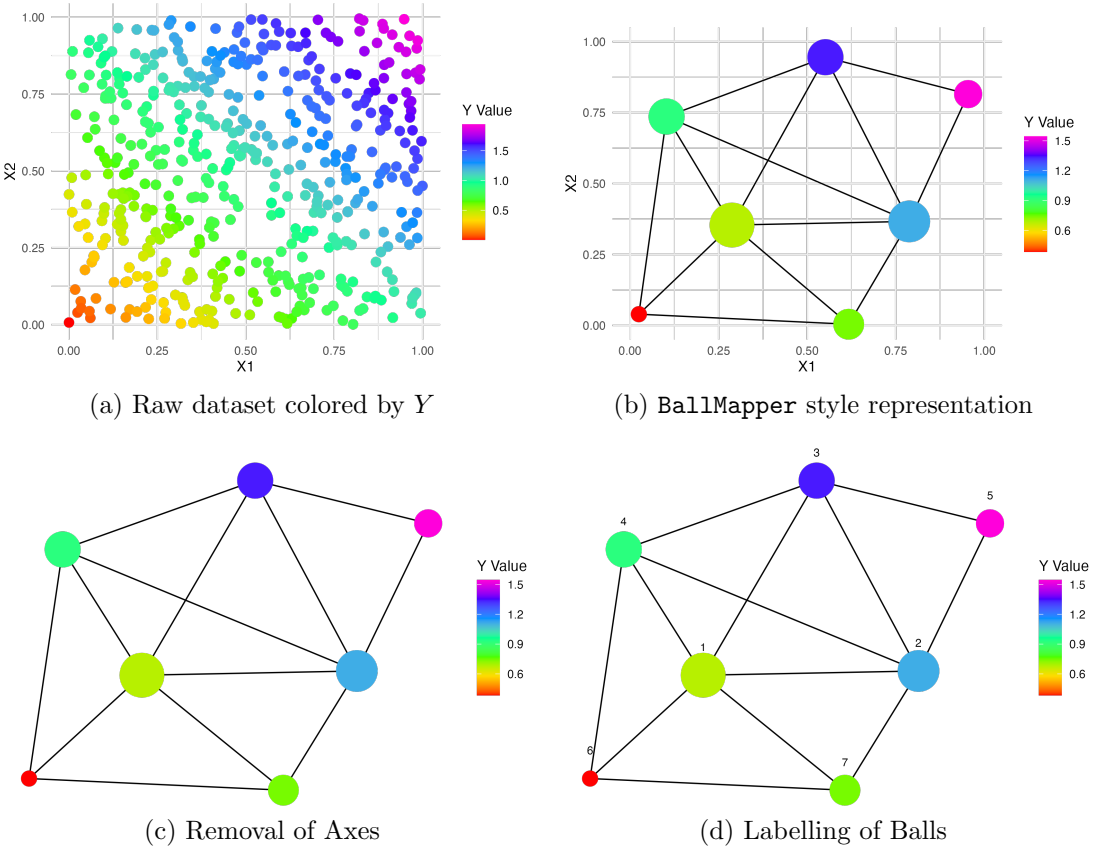
```
selected_point6<-as.data.frame(cbind(selected_point6$X1,selected_point6$X2))
names(selected_point6)<-c("X1","X2")
sel<-as.data.frame(rbind.data.frame(sel,selected_point6))
```

Create the intersections and membership

```
df1$ball6<-ifelse(df1$within_radius6==1&df1$ball1==0&df1$ball2==0&
    df1$ball3==0&df1$ball4==0&df1$ball5==0&df1$intersection99==0,6,0)
df1$intersection5<-ifelse(df1$within_radius6==1&df1$within_radius==1,99,0)
df1$intersection5<-ifelse(df1$within_radius6==1&df1$within_radius2==1,99,
    df1$intersection5)
df1$intersection5<-ifelse(df1$within_radius6==1&df1$within_radius3==1,99,
    df1$intersection5)
df1$intersection5<-ifelse(df1$within_radius6==1&df1$within_radius4==1,99,
    df1$intersection5)
df1$intersection5<-ifelse(df1$within_radius6==1&df1$within_radius5==1,99,
    df1$intersection5)
df1$intersection99<-df1$intersection+df1$intersection2+df1$intersection3+
    df1$intersection4+df1$intersection5
df1$intersection99<-ifelse(df1$intersection99>0,99,0)
df1$member<-df1$ball1+df1$ball2+df1$ball3+df1$ball4+df1$ball5+df1$ball6+
    df1$intersection99
df1$member<-ifelse(df1$member<8,df1$member,100)
```

This example has produced the 6th landmark within the example. There may be more efficient coding mechanisms to obtain the same result.

Figure 4: Annotating TDABM Plots



(a) Raw dataset colored by Y

(b) `BallMapper` style representation

(c) Removal of Axes

(d) Labelling of Balls

Notes: Demonstration of the construction of a TDABM plot for the dataset shown in panel (a). Panel (a) shows the $N = 500$ data points colored according to the value $y_i = x_{1i} + x_{2i}$. Variables X_1 and X_2 are drawn independently from $U[0, 1]$. Panel (b) shows the balls as constructed in Figure 3. The ball size corresponds to the number of points within the ball normalised onto an interval of sizes between 5 and 15. Coloration of the balls is according to the average value of y_i across all points within the ball. Panel (c) shows the same plot without the axes. Panel (d) has numbers corresponding to the order in which the landmarks are selected.

4.2 Interpreting TDABM Graphs

To this point, the demonstration has focused on the construction of the TDABM graph. Because TDABM knows which data points are in which ball, it is possible to enhance the visualization to represent the density of the joint distribution of X. It is also possible to map Y across the space using the TDABM visualization. Figure 4 demonstrates the process of getting from the raw data to the final representation.

Figure 4 panel (a) repeats the dataset picture from Figure 1 as a comparator for the other panels. Panel (b) is based upon panel (h) of Figure 3. Here the landmarks have been resized to reflect the number of data points that appear within each ball. The sizes are normalized to ensure that the representation of the ball does not become too large. Instead there is a minimum size associated with the smallest number of points and a maximum associated with the largest number of points. By resizing the data, an impression of the density of the point cloud in the area surrounding each landmark is provided.

The coloration is based upon the average value of Y across all of the points within the ball. Here, advantage is taken of the fact that the algorithm knows exactly which points are within each ball. In the code dummies have been created for points within the fixed radius of each landmark. To obtain the average value of Y, it is simply a case of subsetting to just those points in the ball and finding the mean. Code is omitted from this section for brevity. Because TDABM plots are abstract, coloration by the X axes can be used to understand how each varies across the TDABM plot. In this case, the balls remain at the correct co-ordinates in X_1, X_2 space such that coloration by either X_1 or X_2 is not necessary.

Once the coloration has been added the ball numbers become useful. In this example, the ball numbers are added manually in panel (d) of Figure 4. Now it is possible to say that Ball 1 has the highest value of the coloration function. Ball 5 has the lowest number of observations contained within, Ball 2 is connected to Ball 3 etc. It is through the discussion of ball numbers that more is understood about the structure of the data. When the coloration of the TDABM is generated from real-world data the discussion of the colors of balls has more meaning. Low, or high, coloration balls amonst otherwise high, low, balls would indicate an important variation in the outcome surface that warrants attention. The ball numbers have no further interpretation beyond being devices to describe the plot.

4.3 TDABM Graph

The previous subsections have documented the process through which the TDABM graph is constructed. However, when implementing TDABM in R, a single command is used. Code is provided in Box 4.2 for the production of the datasets from the `df1` dataset that has been built in previous subsections. Secondly, Box 4.3 shows the implementation of the TDABM algorithm.

Box 4.2: Preparing Data for TDABM

First produce the requisite data sets for the axes and outcomes:

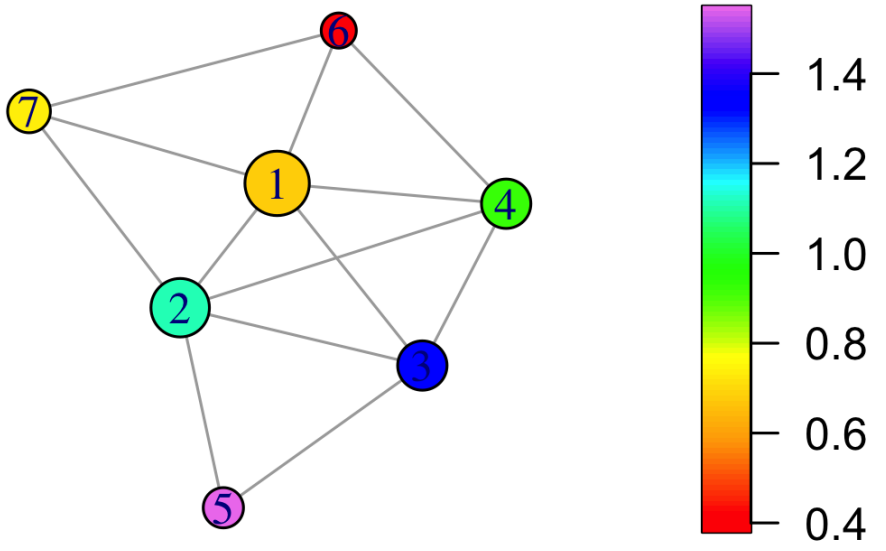
```
xd<-as.data.frame(cbind.data.frame(df1$X1,df1$X2))
yd<-as.data.frame(df1$Y)
```

We bind the two variables together to produce a `data.frame` object:

```
names(xd)<-c("X1","X2")
names(yd)<-"Y"
```

Having created a dataframe for the axes, a dataframe for the outcome and chosen a radius, it is possible to generate the TDABM graph. The code for generating the TDABM graph is provided in Box 4.3. The simplicity of the `BallMapper` function itself is clear.

Figure 5: TDABM Plot



Notes: TDABM plot of a bivariate dataset X in which X_1 and X_2 are drawn independently from $U(0,1)$. TDABM graph constructed with ball radius 0.4. The generation process for the outcome Y has $y_i = x_{i1} + x_{i2}$. Coloration is according to the average value of Y within each ball. TDABM implemented using the R package `BallMapper` by Dlotko (2019).

Box 4.3: R Code for Implementing TDABM

Having constructed the axes as `xd` and outcome as `yd`, the TDABM algorithm with radius $\epsilon = 0.4$ is implemented using

```
bm1<-BallMapper(xd,yd,0.4)
```

The result is a BallMapper object, `bm1`.

Plotting the TDABM object is done using the `ColorIgraphPlot` command. The plotting makes use of the `igraph` package of Csárdi et al. (2025), based on the paper Csardi and Nepusz (2006). The `igraph` package is also the workhorse package for social network analysis in R. For the purposes of plotting TDABM, the ability to recreate the TDABM graph as a network is the key advantage. When plotting more than 2 dimensions, the result is necessarily abstract. The parameter `seed_for_plotting` allows the user to change the representation for readability without altering the structure. Box 4.4 shows the code for plotting the `bm1` object constructed following Box 4.3. The resulting TDABM graph is shown in Figure 5.

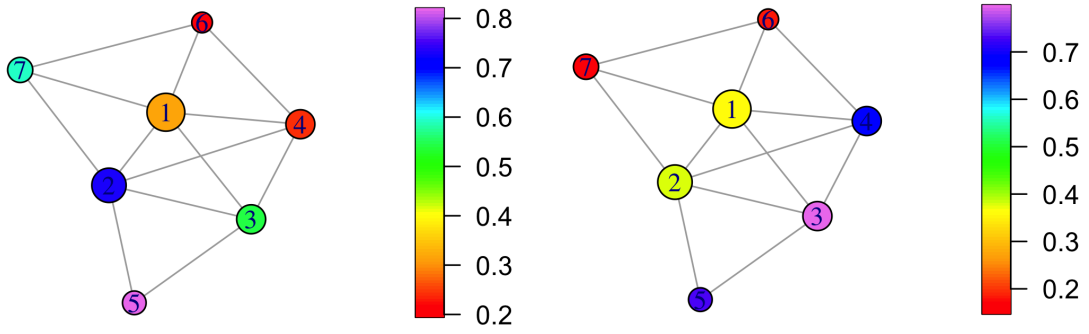
Box 4.4: R Code for Generating a Simple TDABM Plot

The TDABM object, `bm1`, can be plotted in R using:

```
ColorIgraphPlot(bm1,seed_for_plotting=123)
```

Many similarities can be seen between the plot of Figure 5 and panel (d) of Figure 4. The largest balls are

Figure 6: TDABM Plot Axes



(a) Colored by X_1 (b) Colored by X_2

Notes: TDABM plot of a bivariate dataset X in which X_1 and X_2 are drawn independently from $U(0,1)$. TDABM graph constructed with ball radius 0.4. Coloration of the two panels is according to the average value of the respective axes within each ball. TDABM implemented using the R package `BallMapper` by Dlotko (2019).

numbers 1 and 2, as was seen in Figure 4. Ball 6 has the lowest value of the coloration function, whilst Ball 5 has the highest coloration. In the plot of panel (a), it can be seen that Ball 5 is to the North East where $X_1 + X_2$ is at the maximum. Ball 6 is located to the lower left where $X_1 + X_2$ is at its lowest. Because the artificial simulation of TDABM did not make the plot abstract, the mapping from the balls to the original dataset is easier to understand. However, the abstract representation is still easy to follow.

The challenge of understanding the relationship between the balls and the X axis values stems from the ability of TDABM to handle multiple dimensions. In the demonstration, links with the X variables is made by coloration of the balls by values of X_1 and X_2 is undertaken. Figure 6 has two panels in which each panel plots a different axis value.

Figure 6 shows that the highest values of X_1 can be found to the lower left of the TDABM plot. Meanwhile the highest values of X_2 are to the lower right. The bottom corner contains ball 5. Ball 5 is high for both X_1 and X_2. From Figure 4, it is understood that Ball 5 sits in the North East of the space. It is seen too that Ball 3 has higher values of X_2, but that no balls have higher X_1 than Ball 5. Ball 4 has low X_1, whilst ball 7 has low X_1. The lowest value of both variables in Ball 6. Again comparisons across Figures 4 and 6 are clear.

5 Analyzing BallMapper Output

5.1 Elements of BallMapper Output

The `BallMapper` object that is generated through the implementation of the `BallMapper` function contains further elements which can be used to support analysis. In this section I explore the application of these

Table 1: Elements of a `BallMapper` object

Name	Description
vertices	These are the balls in the graph. For each vertex the number of points contained within the ball is reported
edges	A list of the from ball and to ball for all of the edges that are included in the TDABM plot
edges_strength	For each edge the strength is defined as the number of points in the intersection of the to and from ball. To avoid structure created by a single point in the intersection, the edge strength can be used to manually adjust the edges list prior to passing to the plotting function. In most cases the edges_strength element is not used.
points_covered_by_landmarks	Produces a list of the points within each ball. The points are the row numbers of each data point in the original axis dataset. In most cases the format is not useful so a separate conversion function is provided in this guide.
landmarks	These are the point numbers from the original axis dataset of the landmarks. Using the landmark list it is possible to merge back with the underlying dataset to learn more of the specific points that have been chosen.
coloring	A list of the coloration values for each of the balls. If you wish to provide a different value of coloring then you can do so by setting the coloring value equal to a new vector. The requirement here is that each element of the vector matches the corresponding ball in the graph.
coverage	A list which has an element for each data point and then a sub-list within each point for the balls that contain that point. The formatting of the list is also awkward to work with so it is often easier to work from the function suggested in this guide.

Notes: Table provides details of the elements that are contained within a `BallMapper` object in R. TDABM is implemented using the `BallMapper` package of Dlotko (2019).

additional elements for preliminary analysis of TDABM plots. The wealth of possibilities facilitated by the output go well beyond the scope of this introductory guide and readers are directed to the various papers mentioned in the introduction for inspiration. Here I offer a basic summary to help users get started analysing their data. A list of elements in the `BallMapper` object is given in Table 1

5.2 Ball Membership

Box 5.2 includes the `points_to_balls()` function which was developed by the author for the purpose of converting the output from `BallMapper` into a format that can be used to merge with the underlying dataset. The function makes use of `points_covered_by_landmarks`, looping over each of the landmarks to extract the information about the points within the ball. This function can be updated for speed, but is still very quick to run in any case. The resulting dataframe, named `a1` within the function, is a long list of points for each ball starting with Ball 1. The `pt` column is the point number in the order that the original

axis dataset was provided to the algorithm.

Box 5.1: R Code for Implementing TDABM

The points contained within each ball of the TDABM plot, `bm1`, are recovered with a user defined function:

```
points_to_balls<-function(bm1){
  a001<-length(l$landmarks)
  a1<-matrix(0,nrow=1,ncol=2)
  a1<-as.data.frame(a1)
  names(a1)<-c("pt","ball")
  for(i in 1:a001){
    a<-as.data.frame(bm1$points_covered_by_landmarks[i])
    names(a)<-"pt"
    a$ball<-i
    a1<-rbind.data.frame(a1,a)
  }
  a1<-a1[2:nrow(a1),]
  return(a1)
}
```

Because the output of the `points_to_balls` function is a long list of point numbers, the output can be merged with the underlying dataset easily. To enact the merge a column with the name `pt` is needed in the underlying dataset. The code in Box 5.2 provides the necessary steps to get from the `BallMapper` object `bm1` to the merged dataset. The steps in Box 5.2 include the application of the `points_to_balls` function.

Box 5.2: Merging Balls and Underlying Data

We apply the function to the `bm1` object and merge with the data

```
pb1<-points_to_balls(bm1)
names(pb1)<-c("pt","Ball")
df1$pt<-seq(1:nrow(df1))
df2<-merge(df1,pb1,by="pt")
```

Note here the use of `df2` recognises that points appear in multiple balls

The block of code creates a new dataframe, `df2`, which has one row for each of the points in each ball. Hence where a point appears in more than one ball, that point will appear more than once in `df2`. By keeping the merged dataset under a different name to the original data, the original data is preserved for future application.

5.3 Coloration

Box 5.3 provides the code for extracting the coloration of the balls in the TDABM plot and creating a data frame with the coloration values. This is useful for verifying the inference provided by the coloration in the TDABM graph itself.

Box 5.3: R Code for Implementing TDABM

The TDABM object, `bm1`, can be plotted in R using:

```
cd1<-as.data.frame(bm1$color)
cd1$Ball<-seq(1:nrow(cd1))
names(cd1)<-c("TDABM_Coloration","Ball")
```

The coloration element can also be used in reverse to generate a new coloration variable for the plot. Using the merged dataset, summaries for each ball can be generated by using `dplyr` to group on the ball number. If a new coloration dataset emerges then the coloring element can be udpated by assignment in R. The code in Box 5.4 shows the construction of the standard deviation of Y for each data point and then the subsequent assignment of the coloration to the `bm1` object. Figure 7 gives the updated TDABM plot.

Box 5.4: R Code for Generating a new Coloration

First summarise the extended dataframe by grouping on the `Ball` variable:

```
df2g<-group_by(df2,Ball)
df2s<-summarise(df2g,sdy = sd(Y), .groups="drop")
df2s<-as.data.frame(df2s)
names(df2s)<-c("Ball","SDY")
```

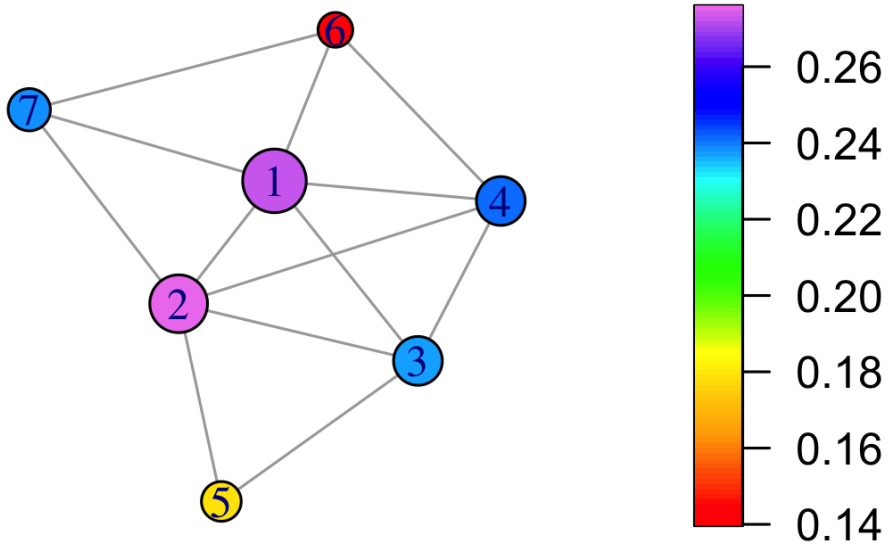
Next update the coloring variable

```
bm1$coloring<-df2s$SDY
```

The plot of `bm1` can then be generated in the usual way.

Figure 7 reveals that there is a large variation in Y within each ball. The result is unsurprising since a large radius is being used and the values of Y are tied to the values of both axis variables. The smallest standard deviation appears in the smaller balls at the edges of the plot. Balls 5 and 6 both have lower standard deviations. Here the summary function is purely for illustration. In other contexts, the extent of within ball variation in outcome may be used as a guide on radius selection, or as a way to understand the ability of the variables in X to explain the variation in Y. Note that TDABM is only showing the extent of variation of Y conditional on the X within the ball, there is no causality implied. Adding additional coloration variables requires updating the code in Box 5.4.

Figure 7: TDABM Plot



Notes: TDABM plot of a bivariate dataset X in which X_1 and X_2 are drawn independently from $U(0,1)$. TDABM graph constructed with ball radius 0.4. The generation process for the outcome Y has $y_i = x_{i1} + x_{i2}$. Coloration is according to the standard deviation of Y within each ball. TDABM implemented using the R package `BallMapper` by Dlotko (2019).

5.4 Updating the Plot Function

Finally, this guide considers the ways in which the `ColorIgraphPlot` function may be updated to provide more meaningful plots. The inputs to the `ColorIgraphPlot` function are provided in Table 2.

Table 2 demonstrates the flexibility of the plotting of TDABM graphs. It is possible to dig further into the options by going into the R function itself and making adjustments. For most purposes, the default inputs to the function are sufficient. Elements such as turning the ball labels off can be used to help readability of plots. The numbers are for discussing the balls, they do not have any value relevant function in the analysis. Hence, provided there is at least one plot with numbers it is possible to have other versions of the same graph without the numbers. The options to specify graphic parameters are also useful, but here the code is used to generate the plot as a new graphic device in R. The saving of plots is done using separate blocks of code.

6 Summary

This guide is an introduction to the `BallMapper` function from the `BallMapper` package in R (Dlotko, 2019). Using a simple bivariate dataset, it is shown how the BallMapper algorithm of Dłotko (2019) is implemented. It is then shown how the R package interprets the selection of landmarks and how the resulting difference in TDABM graph appearance results. The dataset is artificial and as such there is limited interpretation to

Table 2: Inputs to ColorIgraphPlot

Input Element `outputFromBallMapper`	Default	Description `BallMapper` object as generated by the `BallMapper` function
`showVertexLabels`	TRUE	A TRUE/FALSE indicator that states whether the numbers should be added to the balls. Numbers are useful for discussing the balls, but may distract in some applications. This option can be useful when plotting axes and having one plot with ball numbers against which the others can be directly compared.
`showLegend`	FALSE	This will add a list of balls to the plot. Typically, only the color bar is needed for interpreting the TDABM graph
`minimal_ball_radius`	7	Controls the size of the smallest ball. The size should be sufficient to include the label of the ball. Hence the minimum is defaulted to 7.
`maximal_ball_scale`	20	Controls the maximum size of a ball in the TDABM plot. This value can be made larger, but it must be remembered that large balls will start to dominate the space.
`maximal_color_scale`	10	
`seed_for_plotting`	-1	Because TDABM creates abstract visualisations, the process through which the balls are represented in two dimensions applies a spring algorithm. Setting the seed ensures that reproducability of the spring algorithm. Hence set the seed to have a consistent TDABM outcome.
`store_in_file = ""`		Allows the specification of a filename to store the resulting TDABM plot. In this guide the code is provided to open a new device and then plot the TDABM graph. You may prefer to set the inputs here.
`default_x_image_resolution = 512`	512	The resolution used on the horizontal direction defines the quality of the TDABM plot when it appears in documents. Higher resolutions improve clarity but also generate larger files. Use of little `x` here is because of the R function thinking of the horizontal axis.
`default_y_image_resolution = 512`	512	The resolution used on the vertical axis further defines the quality of the TDABM plot when it appears in documents. Higher resolutions improve clarity but also generate larger files. Use of little `y` here is because of the R function thinking of the vertical axis.
`number_of_colors`	100	The number of colors is a parameter of the colorbar in R. Increasing the number allows for more subtle variations between the colors of balls.

Notes: Table provides details of the inputs that are are provided to the `ColorIgraphPlot` function as implemented in the `BallMapper` package of Dlotko (2019).

be gained from the balls that are generated. However, as demonstrated in the literature, the plots do have meaning once the data is from the real-world (Rudkin et al., 2024a; Otway and Rudkin, 2024; Rudkin and Dlotko, 2024, amongst others). Readers are encouraged to apply TDABM to their own datasets as part of appreciating the structures that exist within their data.

There are many alternative means to visualize multidimensional data. However, the simplicity of the TDABM approach gives a ready interpretability. Users may compare the results of the TDABM against alternatives on their data. In the literature, it is the stability of the TDABM graphs, the simplicity of having a single parameter, and the neat interpretation of the similarity of data points within each ball, that motivates adoption. The method also begs comparison with clustering algorithms, but it must be stressed that TDABM is not a clustering algorithm. The grouping of observations into balls is simply to provide a reduction in the number of points for visualizing. Unlike almost all clustering algorithms, the balls of TDABM are all equal in size within the characteristic variable space. Comparisons between balls and clusters is also commonly incorporated into papers applying TDABM. The toolkit built upon TDABM is constantly evolving. Functionality is available in both Python and R. This guide is based upon R, but the principles can be readily matched to Python.

The need to visualize data is emphasised by Matejka and Fitzmaurice (2017), whilst the need to evaluate statistical models with visualization is shown in Anscombe (1973). This guide shows how TDABM can produce data visualizations, and how the algorithm allows coloration. As Qiu et al. (2020), Dłotko et al. (2024), Rudkin et al. (2024b) and others discuss, TDABM offers a means to evaluate models. TDABM has a contribution at the start of the modelling process to understand data structure, and at the end to plot model performance. In all analysis conducted to date, the application of TDABM has revealed new information in the data. Future research will continue to expand upon the role of TDABM in statistical analysis and as a visualization tool. This guide shows how the `BallMapper` function in R (Dlotko, 2019) provides foundation to that research agenda.

References

Anscombe, F. J. (1973). Graphs in statistical analysis. The American Statistician, 27(1):17–21.

Csardi, G. and Nepusz, T. (2006). The igraph software package for complex network research. InterJournal, Complex Systems:1695.

Csárdi, G., Nepusz, T., Traag, V., Horvát, S., Zanini, F., Noom, D., and Müller, K. (2025). igraph: Network Analysis and Visualization in R. R package version 1.6.0.

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

Dlotko, P. (2019). BallMapper: Create a Ball Mapper graph of the input data. R package version 0.1.0.

Dlotko, P., Qiu, W., and Rudkin, S. (2022). Topological data analysis ball mapper for finance. arXiv preprint arXiv:2206.03622.

Dłotko, P., Qiu, W., and Rudkin, S. (2024). Financial ratios and stock returns reappraised through a topological data analysis lens. The European Journal of Finance, pages 1–25.

Matejka, J. and Fitzmaurice, G. (2017). Same stats, different graphs: generating datasets with varied appearance and identical statistics through simulated annealing. In Proceedings of the 2017 CHI conference on human factors in computing systems, pages 1290–1294.

Otway, A. and Rudkin, S. (2024). The shape of left-behindedness: The 2019 uk general election and britain's changing electoral geography. Available at SSRN 4877434.

Qiu, W., Rudkin, S., and Dłotko, P. (2020). Refining understanding of corporate failure through a topological data analysis mapping of altman's z-score model. Expert Systems with Applications, page 113475.

R Core Team (2023). R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria.

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

Rudkin, S. and Dlotko, P. (2024). A topology of inclusion: European regions and the digital divide. In 2024 Fourth International Conference on Digital Data Processing (DDP), pages 123–128. IEEE.

Rudkin, S., Rudkin, W., and Dłotko, P. (2024b). Return trajectory and the forecastability of bitcoin returns. Financial Review.

Rudkin, S. and Webber, D. J. (2023). Regional growth paths and regional resilience. Available at SSRN 4333276.

Singh, G., Mémoli, F., and Carlsson, G. E. (2007). Topological methods for the analysis of high dimensional data sets and 3d object recognition. SPBG, 91:100.