Akmedoids R-package for longitudinal dataset: A guide to measuring long-term inequality in the exposure to crime at micro-area levels

Adepeju, M., Langton, S., and Bannister, J.

Big Data Centre, Manchester Metropolitan University, Manchester, M15 6BH

2019-03-10

**Abstract**

The akmedoids package advances a set of R-functions for longitudinal clustering of long-term trajectories and determines the optimal solution based on the Calinski-Harabatz criterion (Caliński and Harabasz 1974). The package also includes a set of useful functions for addressing common data issues, such as missing entries and outliers, prior to conducting longitudinal data analysis. One of the key objectives of this package is to facilitate easy replication of a recent paper which examined small area inequalities in the crime drop (see Adepeju et al. 2019). This document is created to provide a guide towards accomplishing this objective. Many of the functions provided in the akmedoids package may be applied to longitudinal data in general.

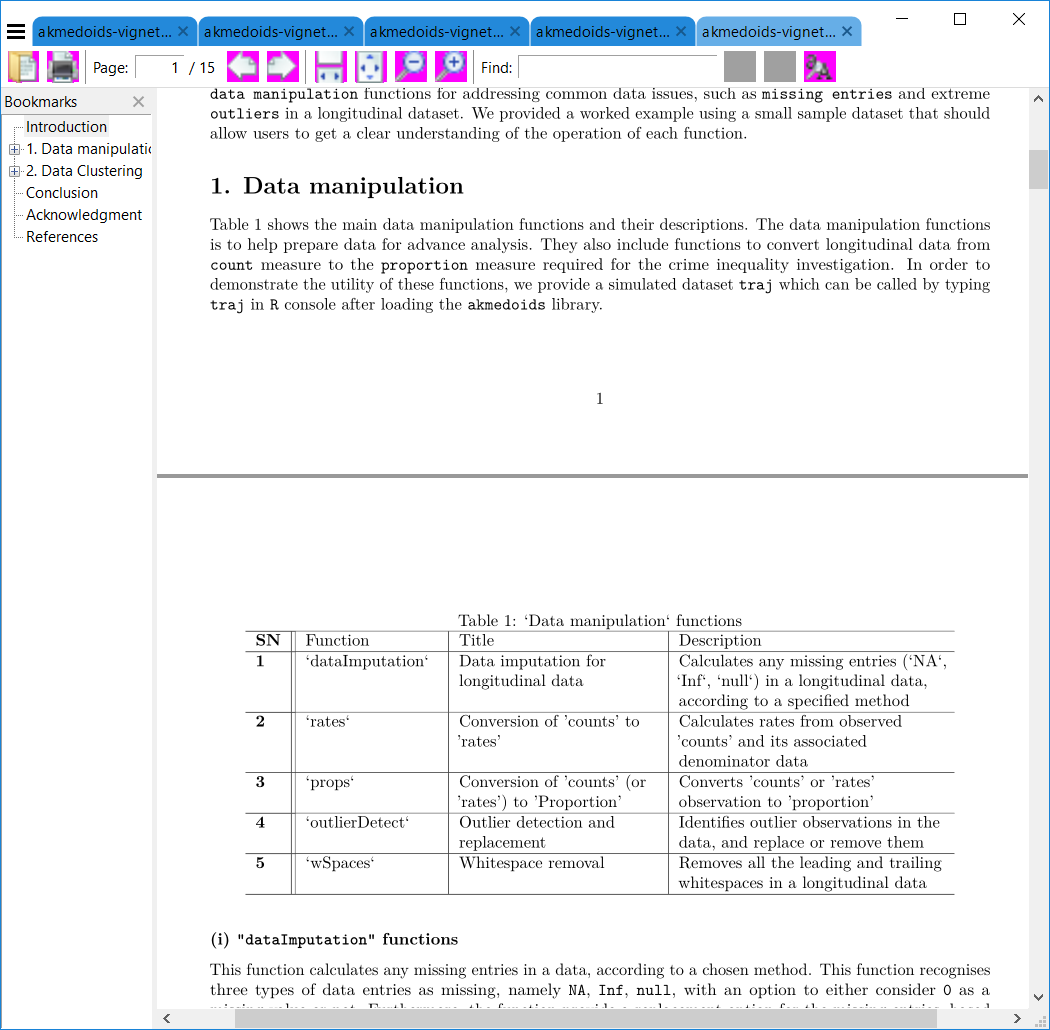
# Introduction

Longitudinal clustering analysis is ubiquitous in social and behavioural sciences for investigating the developmental processes of a phenomenon over time. Examples of the commonly used techniques in these areas include group-based trajectory modelling (GBTM) and the non-parametric kmeans method. Whilst kmeans has a number of benefits over GBTM, such as more relaxed statistical assumptions, generic implementations render it more sensitive to outliers and short-term fluctuations, which minimises its ability to identify long-term linear trends in data. In crime and place research, for example, the identification of such long-term linear trends may help to develop some theoretical understanding of criminal victimisation within a geographical space (Weisburd, et al., 2004; Griffith and Chavez 2004). In order to address this sensitivity problem, we advance a novel technique named anchored kmedoids (‘akmedoids’) which implements three key modifications to the existing longitudinal kmeans approach. First, it approximates trajectories using ordinary least square regression (OLS) and second, anchors the initialisation process with median observations. It then deploys the medoids observations as new anchors for each iteration of the expectation-maximisation procedure (Celeux and Govaert 1992). These modifications ensure that the impacts of short-term fluctuations and outliers are minimised. By linking the final groupings back to the original trajectories, a clearer delineation of the long-term linear trends of trajectories are obtained.

We facilitate the easy use of akmedoids through an open-source package using R. We encourage the use of the package outside of criminology, should it be appropriate. Before outlining the main clustering functions, we demonstrate the use of a few data manipulation functions that assist in data preparation. The worked demonstration uses a small example dataset which should allow users to get a clear understanding of the operation of each function.

# 1. Data manipulation

Table 1 shows the main data manipulation functions and their descriptions. These functions help to address common data issues prior to analysis, as well as basic data manipulation tasks such as converting longitudinal data from count to proportion measures (as per the crime inequality paper where akmedoids was first implemented). In order to demonstrate the utility of these functions, we provide a simulated dataset traj which can be called by typing traj in R console after loading the akmedoids library.



### (i) "dataImputation" function

Calculates any missing entries in a data, according to a chosen method. This function recognises three kinds of data entries as missing: NA, Inf, null, and an option of whether or not to consider 0‘s as missing values. The function provides a replacement option for the missing entries using two methods. First, an arithmetic method which uses the mean, minimum or maximum value of the corresponding rows or columns of the missing values. Second, a regression method which uses OLS regression lines to estimate the missing values. Using the regression method, only the missing data points derive values from the regression line while the remaining (observed) data points retain their original values. The function terminates if there is any trajectory with only one observation in it. Using the traj dataset, we demonstrate how the regression method estimates missing values.

#installing the `akmedoids` packages  
install.packages("devtools")  
devtools::install\_github("manalytics/packages/akmedoids")

#loading the package  
library(akmedoids)

#viewing the first 6 rows of 'traj' object  
head(traj)  
#> location\_ids X2001 X2002 X2003 X2004 X2005 X2006 X2007 X2008 X2009  
#> 1 E01012628 3 0 1 2 1 0 1 4 0  
#> 2 E01004768 9 NA 2 4 7 5 1 3 1  
#> 3 E01004803 4 3 0 10 2 3 6 6 8  
#> 4 E01004804 7 3 9 3 2 NA 6 3 2  
#> 5 E01004807 2 Inf 5 5 6 NA 3 5 4  
#> 6 E01004808 8 5 8 4 1 5 6 1 1  
  
#no. of rows  
nrow(traj)   
#> [1] 10  
  
#no. of columns  
ncol(traj)   
#> [1] 10

The first column of the traj object is the id (unique) field. In many applications, it is necessary to preserve the id column in order to allow linking of outputs to other external datasets, such as spatial location data. Most of the functions of the akmedoids provides an option to recognise the first column of an input dataset as the unique field. The dataImputation function can be used to imput the missing data point of traj object as follows:

imp\_traj <- dataImputation(traj, id\_field = TRUE, method = 2,   
 replace\_with = 1, fill\_zeros = FALSE)  
#> [1] "8 entries were found/filled!"  
  
#viewing the first 6 rows  
head(imp\_traj)  
#> location\_ids X2001 X2002 X2003 X2004 X2005 X2006 X2007 X2008 X2009  
#> 1 E01012628 3 0.00 1 2 1 0.00 1 4 0  
#> 2 E01004768 9 6.44 2 4 7 5.00 1 3 1  
#> 3 E01004803 4 3.00 0 10 2 3.00 6 6 8  
#> 4 E01004804 7 3.00 9 3 2 3.90 6 3 2  
#> 5 E01004807 2 3.92 5 5 6 4.36 3 5 4  
#> 6 E01004808 8 5.00 8 4 1 5.00 6 1 1

The argument method = 2 refers to the regression technique, while the argument replace\_with = 1 refers to the linear option (currently the only available option). Figure 1 is a graphical illustration of how this method approximates the missing values of the traj object.

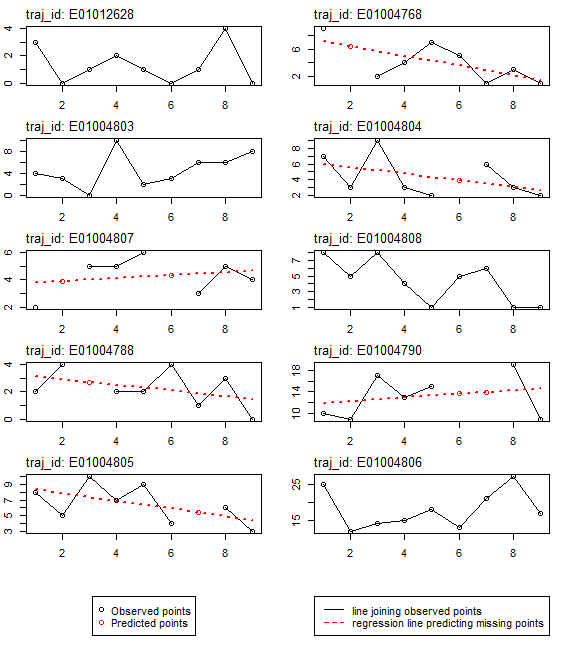


Figure 1: data imputation with regression

### A special use of ‘dataImputation’ function:

Obtaining the denominator information (e.g. population estimates to normalize counts) of local areas within a city for non-census years is problematic in longitudinal studies. This challenge poses a significant drawback to the accurate estimation of various measures, such as crime rates and population-at-risk of an infectious disease. Assuming a limited amount of denominator information is available, an alternative way of obtaining the missing data points is to interpolate and/or extrapolate the missing population information using the available data points. The dataImputation function can be used to perform this task.

The key step towards using the function for this purpose is to create a matrix containing both the available fields and the missing fields arranged in their appropriate order. All the entries of the missing fields can be filled with either NA or null. Below is a demonstration of this task with a sample population dataset with only two available data fields. The corresponding input matrix is constructed as shown.

#viewing the data first 6 rows  
head(population)  
#> location\_id census\_2003 census\_2007  
#> 1 E01004809 300 200  
#> 2 E01004807 550 450  
#> 3 E01004788 150 250  
#> 4 E01012628 100 100  
#> 5 E01004805 400 350  
#> 6 E01004790 750 850  
  
nrow(population) #no. of rows  
#> [1] 11  
  
ncol(population) #no. of columns  
#> [1] 3

The corresponding input dataset is prepared as follows and saved as population2:

#> location\_ids X2001 X2002 X2003 X2004 X2005 X2006 X2007 X2008 X2009  
#> 1 E01004809 NA NA 300 NA NA NA 200 NA NA  
#> 2 E01004807 NA NA 550 NA NA NA 450 NA NA  
#> 3 E01004788 NA NA 150 NA NA NA 250 NA NA  
#> 4 E01012628 NA NA 100 NA NA NA 100 NA NA  
#> 5 E01004805 NA NA 400 NA NA NA 350 NA NA  
#> 6 E01004790 NA NA 750 NA NA NA 850 NA NA

The missing values are estimated as follows using the regression method of the dataImputation function:

pop\_imp\_result <- dataImputation(population2, id\_field = TRUE, method = 2,   
 replace\_with = 1, fill\_zeros = FALSE)  
#> [1] "77 entries were found/filled!"  
  
#viewing the first 6 rows  
head(pop\_imp\_result)  
#> location\_ids X2001 X2002 X2003 X2004 X2005 X2006 X2007 X2008 X2009  
#> 1 E01004809 350 325 300 275 250 225 200 175 150  
#> 2 E01004807 600 575 550 525 500 475 450 425 400  
#> 3 E01004788 100 125 150 175 200 225 250 275 300  
#> 4 E01012628 100 100 100 100 100 100 100 100 100  
#> 5 E01004805 425 412.5 400 387.5 375 362.5 350 337.5 325  
#> 6 E01004790 700 725 750 775 800 825 850 875 900

Given that there are only two data points in each row, the regression method will simply generate the missing values by fitting a straight line to the available data points. In other words, the higher the number of available data points for any trajectory the better the estimation of the missing points. Figure 1 illustrates this estimation process.

### (ii) "rates" function

Given a longitudinal data () and its associated denominator data (), the ‘rates’ function converts the longitudinal data to ‘rates’ measures (e.g. counts per 200 residents). Both the longitudinal and the denominator data may contain different number of rows, but need to have the same number of columns, and must include the id (unique) field as their first column. The rows do not have to be sorted in any particular order. The ‘rates’ measures (i.e. the output) will contain only rows whose id’s match from both datasets. We demonstrate the utility of this function using the imp\_traj object (above) and the estimated population data (i.e. ‘pop\_imp\_result’) above.

#example of estimation of 'crimes per 200 residents'  
crime\_per\_200\_people <- rates(imp\_traj, denomin=pop\_imp\_result, id\_field=TRUE, multiplier = 200)  
  
#view the full output  
crime\_per\_200\_people  
#> location\_ids X2001 X2002 X2003 X2004 X2005 X2006 X2007 X2008 X2009  
#> 1 E01012628 6 0 2 4 2 0 2 8 0  
#> 2 E01004768 3.6 2.86 1 2.29 4.67 4 1 4 2  
#> 3 E01004803 2.29 1.6 0 4.71 0.89 1.26 2.4 2.29 2.91  
#> 4 E01004804 5.09 1.92 5.14 1.55 0.94 1.69 2.4 1.12 0.7  
#> 5 E01004807 0.67 1.36 1.82 1.9 2.4 1.84 1.33 2.35 2  
#> 6 E01004808 6.4 3.64 5.33 2.46 0.57 2.67 3 0.47 0.44  
#> 7 E01004788 4 6.4 3.63 2.29 2 3.56 0.8 2.18 0  
#> 8 E01004790 2.86 2.48 4.53 3.35 3.75 3.3 3.29 4.34 2  
#> 9 E01004805 3.76 2.42 5 3.61 4.8 2.21 3.09 3.56 1.85  
  
#check the number of rows  
nrow(crime\_per\_200\_people)  
#> [1] 9

From the output, it can be observed that the number of rows of the output data is 9. This implies that only 9 location\_ids match between the two datasets. The unmatched ids are ignored. **Note**: the calculation of rates often returns outputs with some of the cell entries having Inf and NA values, due to calculation errors and character values in the data. We therefore recommend that users re-run the dataImputation function after generating rates measures, especially for a large data matrix.

### (iii) "props" function

Given a longitudinal data, the props function converts each data point (i.e. entry in each cell) to the proportion of the sum of their corresponding column. Using the crime\_per\_200\_people estimated above, we can derive the proportion of crime per 200 people for each entry as follows:

#Proportions of crimes per 200 residents  
prop\_crime\_per200\_people <- props(crime\_per\_200\_people, id\_field = TRUE)  
  
#view the full output  
prop\_crime\_per200\_people  
#> location\_ids X2001 X2002 X2003 X2004 X2005 X2006 X2007 X2008 X2009  
#> 1 E01012628 0.17 0.00 0.07 0.15 0.09 0.00 0.10 0.28 0.00  
#> 2 E01004768 0.10 0.13 0.04 0.09 0.21 0.19 0.05 0.14 0.17  
#> 3 E01004803 0.07 0.07 0.00 0.18 0.04 0.06 0.12 0.08 0.24  
#> 4 E01004804 0.15 0.08 0.18 0.06 0.04 0.08 0.12 0.04 0.06  
#> 5 E01004807 0.02 0.06 0.06 0.07 0.11 0.09 0.07 0.08 0.17  
#> 6 E01004808 0.18 0.16 0.19 0.09 0.03 0.13 0.16 0.02 0.04  
#> 7 E01004788 0.12 0.28 0.13 0.09 0.09 0.17 0.04 0.08 0.00  
#> 8 E01004790 0.08 0.11 0.16 0.13 0.17 0.16 0.17 0.15 0.17  
#> 9 E01004805 0.11 0.11 0.18 0.14 0.22 0.11 0.16 0.13 0.16  
  
  
#A quick check that sum of each column of proportion measures adds up to 1.   
colSums(prop\_crime\_per200\_people[,2:ncol(prop\_crime\_per200\_people)])  
#> X2001 X2002 X2003 X2004 X2005 X2006 X2007 X2008 X2009   
#> 1.00 1.00 1.01 1.00 1.00 0.99 0.99 1.00 1.01

In line with the demonstration in Adepeju, Langton, and Bannister (2019), we will use these proportion measures to demonstrate the main clustering function of this package.

### (iv) "outlierDetect" function

This function is aimed at allowing users to identify any outlier observations in their longitudinal data, and replace or remove them accordingly. The first step towards identifying outliers in any data is to visualise the data. A user can then decide the cut-off value for isolating the outliers. The outlierDetect function provides two options for doing this: (i) a quantile method, which isolates any observations with values higher than a specified quantile of the data values distribution, and (ii) a manual method, in which a user specifies the cut-off value. The ‘replace\_with’ argument is used to determine whether an outlier value should be replaced with the mean value of the row or the mean value of the column in which the outlier is located. The user also has the option to simply remove the trajectory that contains an outlier value. In deciding whether a trajectory contains outlier or not, the count argument allows the user to set the horizontal threshold (i.e. number of outlier values that must occur in a trajectory) in order for the trajectory to be considered as having outlier observations. Below, we demonstrate the utility of the outlierDetect function using the imp\_traj data above.

#Plotting the data using ggplot library  
library(ggplot2)  
library(reshape2)  
  
#converting the wide data format into stacked format for plotting  
imp\_traj\_long <- melt(imp\_traj, id="location\_ids")   
  
#view the first 6 rows  
head(imp\_traj\_long)  
#> location\_ids variable value  
#> 1 E01012628 X2001 3  
#> 2 E01004768 X2001 9  
#> 3 E01004803 X2001 4  
#> 4 E01004804 X2001 7  
#> 5 E01004807 X2001 2  
#> 6 E01004808 X2001 8  
  
#plot function  
p <- ggplot(imp\_traj\_long, aes(x=variable, y=value,  
 group=location\_ids, color=location\_ids)) +   
 geom\_point() +   
 geom\_line()  
  
print(p)

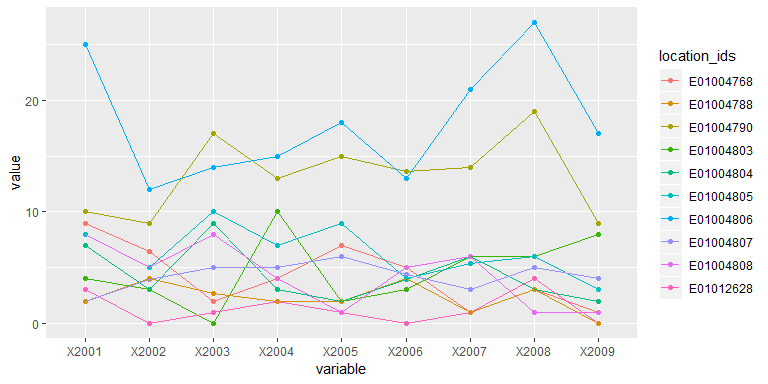


Figure 2 Identifying outliers

Figure 2 is the output of the above plot function.

Based on Figure 2, if we assume that observations of x2001, x2007 and x2008 of trajectory id E01004806 are outliers, we can set the threshold argument as 20. In this case, setting count=1 will suffice as the trajectory is clearly separable from the rest of the trajectories. Setting replace\_with = 2, that is to replace the outlier points with the ‘mean of the row observations’, the function generates outputs re-plotted in Figure 3.

imp\_traj\_New <- outlierDetect(imp\_traj, id\_field = TRUE, method = 2,   
 threshold = 20, count = 1, replace\_with = 2)  
#> [1] "1 trajectories were found to contain outlier observations and replaced accordingly!"  
#> [1] "Summary:"  
#> [1] "\*--Outlier observation(s) was found in trajectory 10 --\*"  
  
imp\_traj\_New\_long <- melt(imp\_traj\_New, id="location\_ids")   
  
#plot function  
p <- ggplot(imp\_traj\_New\_long, aes(x=variable, y=value,  
 group=location\_ids, color=location\_ids)) +   
 geom\_point() +   
 geom\_line()  
  
print(p)

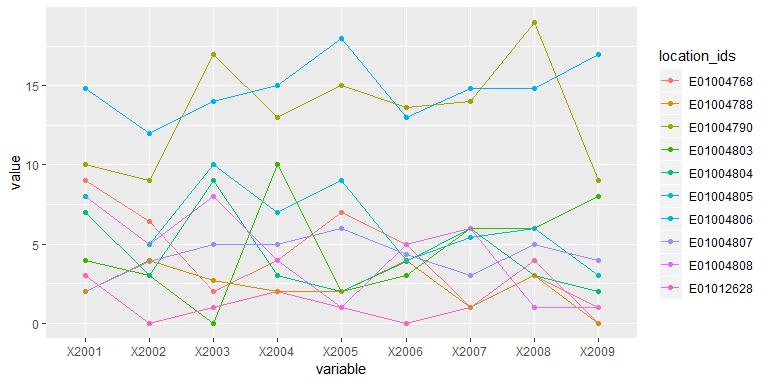


Figure 3: Replacing outliers with mean observation

### (v) ‘Other’ functions

Please see the akmedoids user manual for the remaining data manipulation functions.

# 2. Data Clustering

Table 2 shows the two main functions required to carry out the longitudinal clustering and generate the descriptive statistics of the resulting groups. The relevant functions are akmedoids.clust and statPrint. The akmedoids.clust function clusters trajectories according to the similarities of their long-term trends, while the statPrint function extracts descriptive and change statistics for each of the clusters. The latter also generates performance plots for the best cluster solution.

The long-term trends of trajectories are defined in terms of a set of OLS regression lines. This allows the clustering function to classify the final groupings in terms of their slopes as rising, stable, and falling. The key benefit of this implementation is that it allows the clustering function to ignore the short-term fluctuations of the actual trajectories and focus on their long-term linear trends. Adepeju and colleagues (2019) applied this approach in crime concentration research for measuring long-term inequalities in the exposure to crime at fine-grained spatial scales. Their implementation was informed by the conceptual (inequality) framework shown in Figure 4. That said, akmedoids can be deployed on any measure (counts, rates) and is not limited to criminology, but rather, any field where the aim is to cluster longitudinal data based on long-term trajectories. By mapping the resulting trend lines grouping to the original trajectories, various performance statistics can be generated.

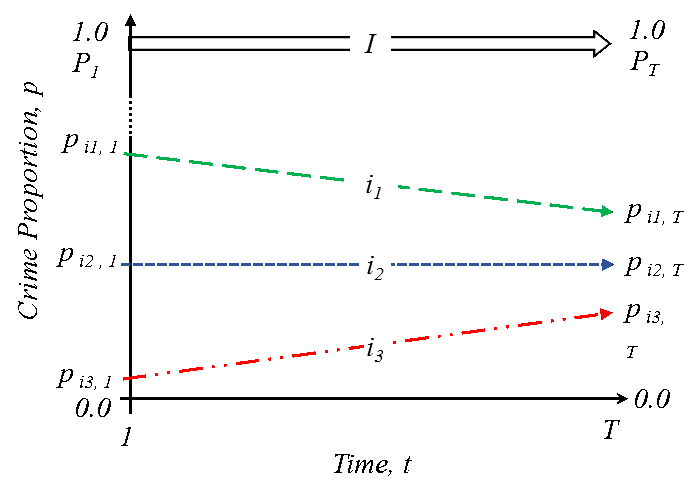
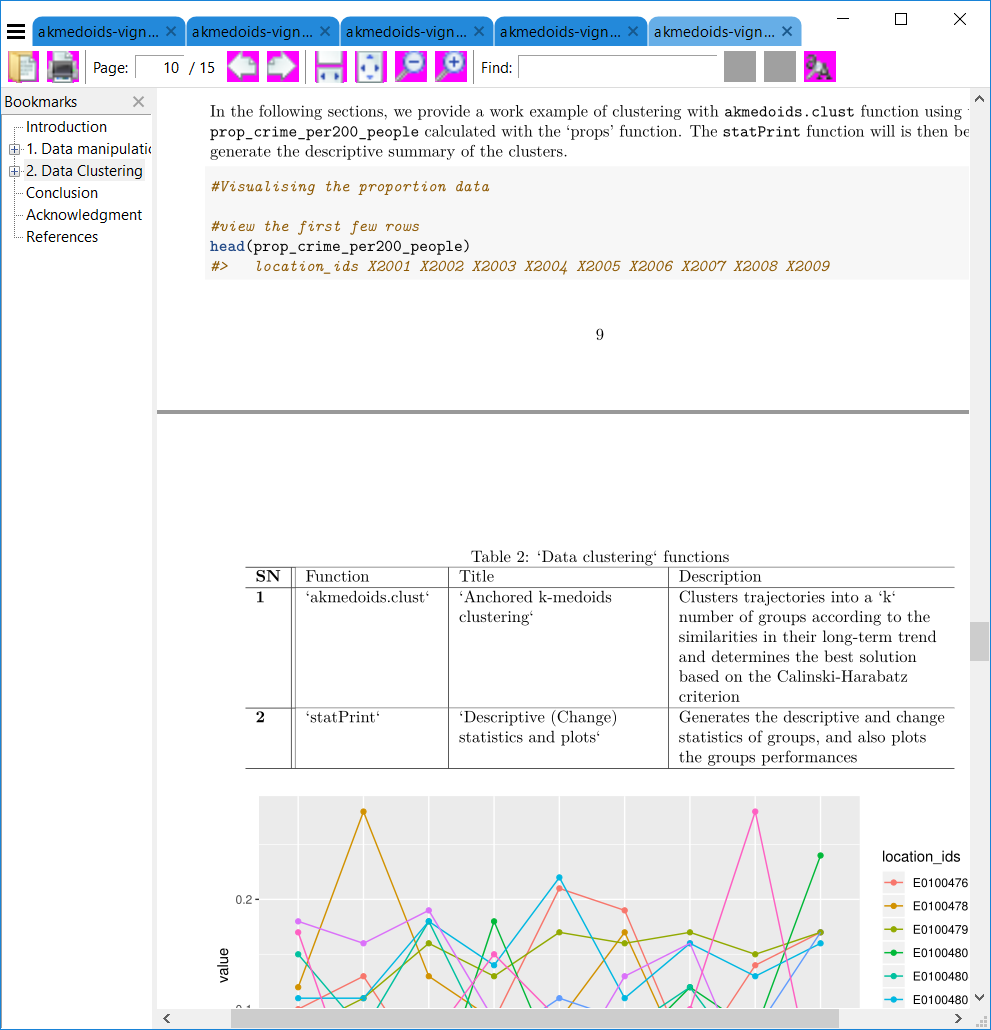


Figure 4: Long-time linear trends of relative (proportion, p) crime exposure. Three inequality trends: trajectory i1: crime exposure is falling faster, i2, crime exposure is falling at the same rate, and i3, crime exposure is falling slower or increasing, relatively to the citywide trend. (Source: Adepeju et al. 2019)

In addition to using the trend lines, the `akmedoids` makes two other modifications to the expectation-maximisation clustering routines (Celeux and Govaert 1992). First, the akmedoids implements an anchored median-based initialisation strategy for the clustering to begin. The purpose of this step is to give the algorithm a theoretically-driven starting point and try and ensure that heterogenous trend slopes end up in different clusters (Khan and Ahmad, 2004; Steinley and Brusco 2007). Second, instead of recomputing centroids based on the mean distances between each trajectory trend lines and the cluster centers, the median of each cluster is selected and then used as the next starting point. This then becomes the new anchor for the current iteration of the expectation-maximisation step (Celeux and Govaert 1992). This strategy is implemented in order to minimise the impact of outliers. The iteration then continues until an objective function is maximised.

In the following sections, we provide a worked example of clustering with `akmedoids.clust` function using the `prop\_crime\_per200\_people` object. The `statPrint` function will then generate the descriptive summary of the clusters. The `prop\_crime\_per200\_people` object is plotted in Figure 5 as follows:



#Visualising the proportion data  
  
#view the first few rows  
head(prop\_crime\_per200\_people)  
#> location\_ids X2001 X2002 X2003 X2004 X2005 X2006 X2007 X2008 X2009  
#> 1 E01012628 0.17 0.00 0.07 0.15 0.09 0.00 0.10 0.28 0.00  
#> 2 E01004768 0.10 0.13 0.04 0.09 0.21 0.19 0.05 0.14 0.17  
#> 3 E01004803 0.07 0.07 0.00 0.18 0.04 0.06 0.12 0.08 0.24  
#> 4 E01004804 0.15 0.08 0.18 0.06 0.04 0.08 0.12 0.04 0.06  
#> 5 E01004807 0.02 0.06 0.06 0.07 0.11 0.09 0.07 0.08 0.17  
#> 6 E01004808 0.18 0.16 0.19 0.09 0.03 0.13 0.16 0.02 0.04  
  
prop\_crime\_per200\_people\_melt <- melt(prop\_crime\_per200\_people, id="location\_ids")   
  
#plot function  
p <- ggplot(prop\_crime\_per200\_people\_melt, aes(x=variable, y=value,  
 group=location\_ids, color=location\_ids)) +   
 geom\_point() +   
 geom\_line()  
  
print(p)

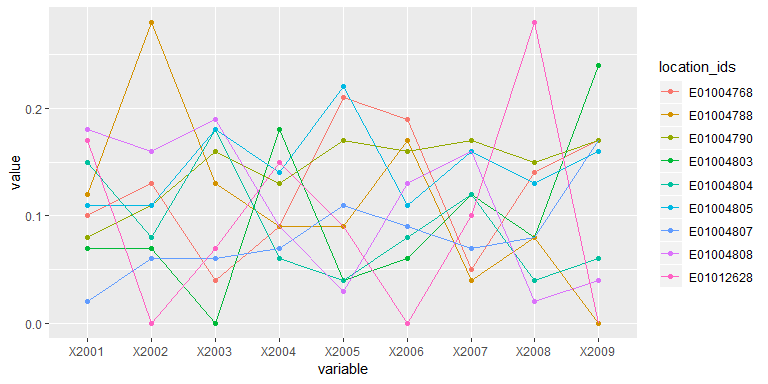


Figure 5: Trajectory of crime proportions over time

### (i) akmedoids.clust function

***Dataset***:

Each trajectory in Figure 5 represents the proportion of crimes per 200 residents in each location over time. This represents the relative exposure to crime of each observation over time. The goal is to first extract the inequality trend lines such as in Figure (4) and then cluster them according to the similarity of their slopes. For the akmedoids.clust function, a user sets the k value which may be an integer or a vector of length two specifying the minimum and maximum numbers of clusters to loop through. In the latter case, the akmedoids.clust function employs the Calinki-Harabatz score (Caliński and Harabasz, 1974; Genolini and Falissard, 2010) to determine the best cluster solution. The function is executed as follows:

#clustering  
cluster\_output <- akmedoids.clust(prop\_crime\_per200\_people, id\_field = TRUE,   
 method = "linear", k = c(3,8))  
#> [1] "solution of k = 3 determined!"  
#> [1] "solution of k = 4 determined!"  
#> [1] "solution of k = 5 determined!"  
#> [1] "solution of k = 6 determined!"  
#> [1] "solution of k = 7 determined!"  
#> [1] "solution of k = 8 determined!"  
  
#print cluster solution  
cluster\_output  
#> [[1]]  
#>   
#> $qualitycriterion  
#> [1] "Quality criterion: Calinski-Harabatz criterion"  
#>   
#> $optimSolution  
#> [1] "C" "D" "E" "B" "E" "A" "A" "D" "C"  
#> attr(,"k")  
#> [1] 5

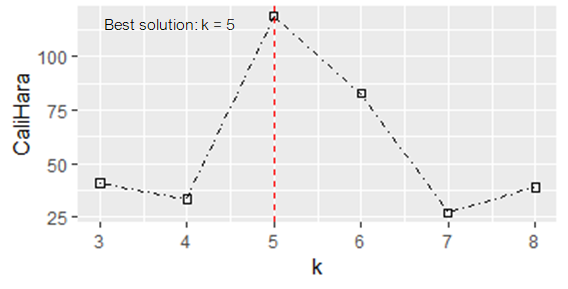


Figure 6: Clustering performance at different values of k

In addition to the output messages (as shown above), the akmedoids.clust function generates a performance plot (Figure 6) that shows the Calinki-Harabatz scores at different values of k. From the plot, the best value of k is highest at k=5, and therefore determined the best solution. Note that the group membership (labels) listed in the output message is that of the best solution determined, which contains five groups labelled from A to E according to increasing slopes. These labels can be extracted by typing the following command:

#vector of group memberships  
as.vector(cluster\_output$optimSolution)   
#> [1] "C" "D" "E" "B" "E" "A" "A" "D" "C"

Also, note that the indexes of the group memberships correspond to that of the trajectory object (prop\_crime\_per200\_people) inputted into the function. That is, the membership labels, "C", "D", "E", .... are the group membership of the trajectories "E01012628","E01004768","E01004803",... of the object prop\_crime\_per200\_people.

### (ii) statPrint function:

Given the vector of group membership (labels), such as = c("C", "D", "E", "B",....) in the example above, and its corresponding trajectory object, prop\_crime\_per200\_people, the statPrint function generates both the descriptive and the change statistics of the groups. The function also generates the plots of the group memberships and their performances in terms of their shares of the proportion measure captured over time. An important argument of statPrint function is the bandw parameter which determines the final classification of the groups in terms of slope. The bandw argument classifies each group into Rising, Stable, or Falling class. We refer users to the package user manual for more details about this parameter. Using the current example, the function can be ran as follows:

#assigning cluster membership to a variable  
clustr <- as.vector(cluster\_output$optimSolution)   
  
#plotting the group membership  
print(statPrint(clustr, prop\_crime\_per200\_people, id\_field=TRUE,   
 bandw = 0.40, type="lines", y.scaling="fixed"))  
#> $descriptiveStats  
#> group n n(%) %Prop.time1 %Prop.timeT Change %Change  
#> 1 A 2 22.2 30 4 -26 -650  
#> 2 B 1 11.1 15 5.9 -9.1 -154.2  
#> 3 C 2 22.2 28 15.8 -12.2 -77.2  
#> 4 D 2 22.2 18 33.7 15.7 46.6  
#> 5 E 2 22.2 9 40.6 31.6 77.8  
#>   
#> $changeStats  
#> group sn %+ve Traj. %-ve Traj. class  
#> 1 A 1 0 100 Rising  
#> 2 B 2 0 100 Rising  
#> 3 C 3 100 0 Stable  
#> 4 D 4 100 0 Falling  
#> 5 E 5 100 0 Falling

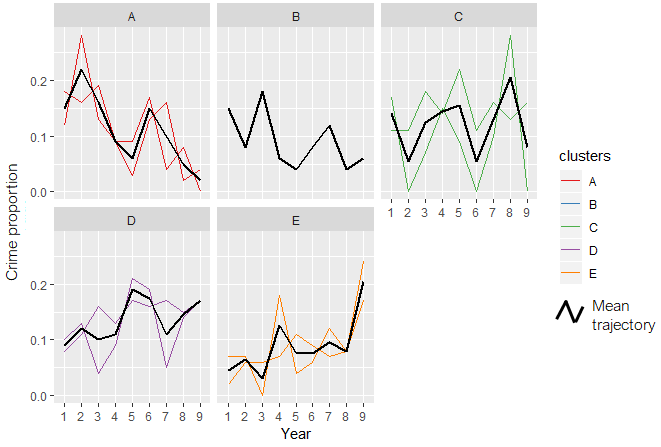


Figure 7: group memberships

See Table 3 for the description of the output table fields. These outputs are generated along with the plot of group memberships as shown in Figure 7. By changing the argument type="line" to type="stacked", a performance plot is generated instead (see Figure 8). Note that these plots make use of functions within the ggplot2 library (Wickham 2016). For a more customised visualisation, we recommend that users deploy the ggplot2 library directly.

In the context of the long-term inequality study, these outputs allow broad conclusions to be made regarding relative crime exposure of crime in the area represented by each group or class (Adepeju, Langton, and Bannister 2019). For example, whilst relative crime exposure have declined in 33.3% (groups A and B) of the study area, the relative crime exposure have risen in 44.4% (groups D and E) of the area. The relative crime exposure can be said to be stable in 22.2% (group C) of the area, based on the bandw parameter.

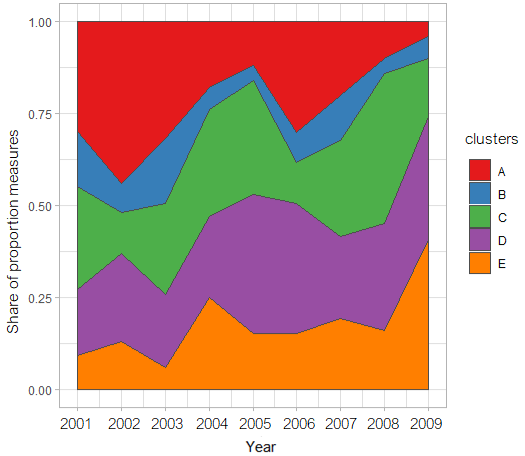
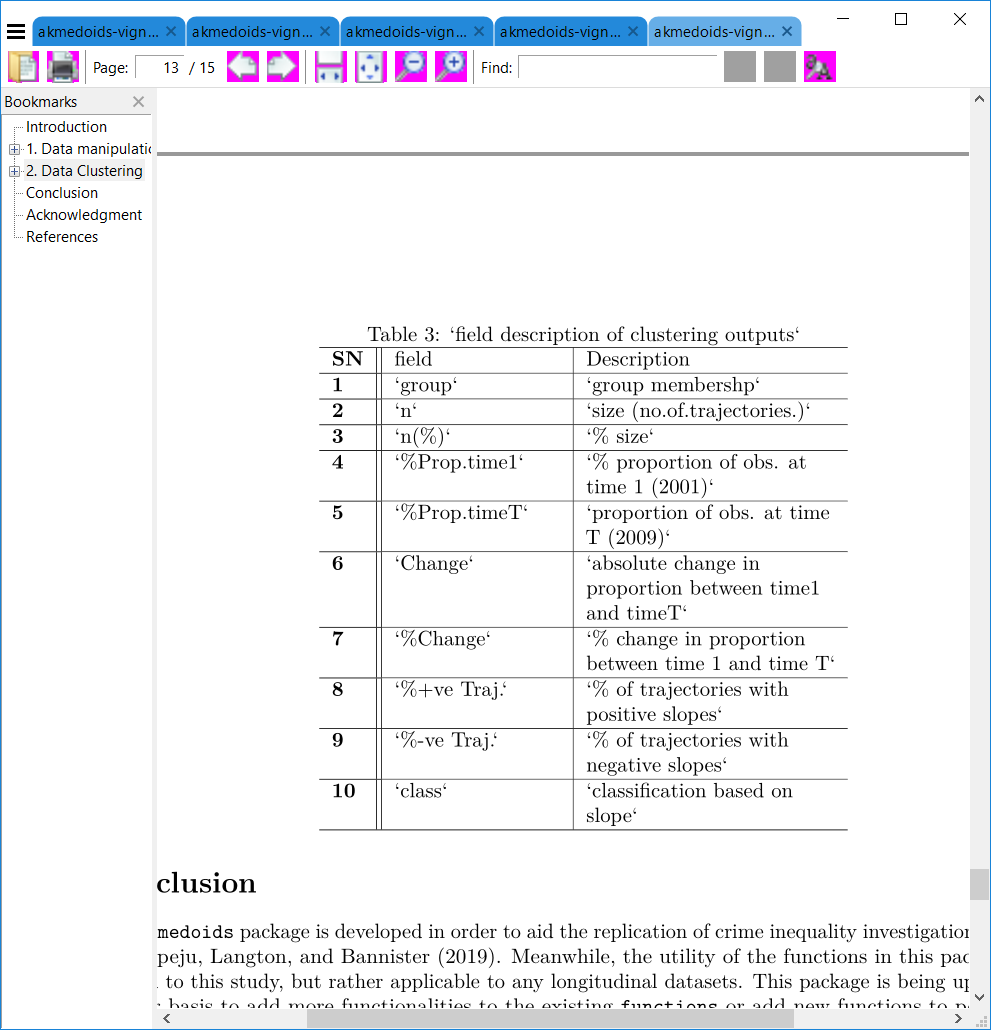


Figure 8: group performance over time



# 

# Conclusion

The akmedoids package has been developed in order to aid the replication of a place-based crime inequality investigation conducted in Adepeju, Langton, and Bannister (2019). Meanwhile, the utility of the functions in this package are not limited to criminology, but rather can be applicable to longitudinal datasets more generally. This package is being updated on a regular basis to add more functionalities to the existing functions and/or add new functions to carry out other longitudinal data analysis.

We encourage users to report any bugs encountered while using the package so that they can be fixed immediately. We welcome contributions to this package which will be acknowledged accordingly.

# Acknowledgment

# References

Adepeju, M., S. Langton, and J. Bannister. 2019. “Anchored K-Medoids: A Longitudinal Clustering Technique for Measuring Long-Term Inequality in the Exposure to Crime at the Micro-Area Levels.” *Journal of Quantitative Criminology*. [(Submitted)](file:///C:\Users\monsu\AppData\Local\Temp\RtmpwtO6bc\preview-5710779161cf.dir\(Submitted)).

Caliński, T., and J. Harabasz. 1974. “A Dendrite Method for Cluster Analysis.” *Communications in Statistics-Theory and Methods* 3(1): 1–27.

Celeux, G., and G. Govaert. 1992. “A Classification Em Algorithm for Clustering and Two Stochastic Versions.” *Computational Statistics* 14(3): 315–32.

Genolini, C., and B. Falissard. 2010. “Kml and Kml3d: R Packages to Cluster Longitudinal Data.” *Computational Statistics* 25(2): 317–28.

Griffith, E., and J.M. Chavez. 2004. “Communities, Street Guns and Homicide Trajectories in Chicago, 1980–1995: Merging Methods for Examining Homicide Trends Across Space and Time.” *Criminology* 42(4): 941–78.

Khan, S. S, and A. Ahmad. 2004. “Cluster Center Initialization Algorithm for K-Means Clustering.” *Pattern Recognition Letters* 25(11): 1293–1302.

Steinley, D., and M. J. Brusco. 2007. “Initializing K-Means Batch Clustering: A Critical Evaluation of Several Techniques.” *Journal of Classification* 24(1): 99–121.

Wickham, H. 2016. *Elegant Graphics for Data Analysis*. Spring-Verlag New York.

Weisburd, D., Bushway, S., Lum, C. and Yang, S.M., 2004. Trajectories of crime at places: A longitudinal study of street segments in the city of Seattle. *Criminology*, *42*(2), pp.283-322.