# Sequence Analysis - Introduction

## Worksheet

Full resource, see:
https://www.ncrm.ac.uk/resources/online/all/?id=20853

This worksheet takes you through code used to create a typology of sequence data using the TraMineR package in R. Relevant R functions are highlighted throughout this worksheet. You can download the R code, from: <https://www.ncrm.ac.uk/resources/online/sequence_analysis_introduction/downloads/R%20code%20used%20in%20worksheet.zip>

Three packages are required for this analysis:

* ‘TraMineR’ = conduct sequence analysis
* ‘cluster’ = conduct cluster analysis
* ‘ggseqplot’ = data visualisation to aid interpretation of clusters

Initial installation of these packages can be achieved by running the **install.packages()** function.

install.packages("TraMineR")

install.packages("cluster")

install.packages ("ggseqplot")

Once installed, you will need to load these packages at the start of every R session using the **library()** function.

library(TraMineR)

library(cluster)

library(ggseqplot)

For the purposes of this worksheet, we will be using the ‘biofam’ data set which is included with the TraMineR package. The code snippet below will load the biofam data set.

data(biofam)

The biofam data set contains 27 variables (columns) and 2,000 rows. Each row represents responses from a single person. biofam data are a random sample of responses from retrospective biographical survey carried out by the Swiss Household Panel in 2002[[1]](#endnote-1). Respondents were asked to give details about their family formation in each year from the age of 15 to 30 years old. States within the data set are labelled from 0 to 7, and are composites of several pieces of information, as follows:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **State** | **Left parental home** | **Married** | **Children** | **Divorced** | **New label** |
| 0 | No | No | No | No | Parents |
| 1 | Yes | No | No | No | Left |
| 2 | No | Yes | Yes/No | No | Married |
| 3 | Yes | Yes | No | No | Left/Married |
| 4 | No | No | Yes | No | Child |
| 5 | Yes | No | Yes | No | Left/Child |
| 6 | Yes | Yes | Yes | No | Left/Married/Child |
| 7 | Yes/No | Yes/No | Yes/No | Yes | Divorced |

By the end of this worksheet, you will have created a typology of family life course trajectories between the ages of 15 and 30 years old.

**Step 1: Creating sequences**

The **seqdef()** function is used to create a sequence object (‘basic.sequence.object’). Columns 10 to 25 in the biofam data set contain the family states, which are called into this function using the ‘var=’ option.

basic.sequence.object = seqdef(biofam, var=10:25)

After running **seqdef()** the dialogue below will appear in the console screen. It summarises the sequence object, its contents, and how states are currently coded/labelled. The min/max sequence length indicates the number of time-points making up sequences. It is advisable to check that the min/max value and the number of sequences created conform to expected values. In this case, the biofam data set contained 2000 rows, and 2000 sequences are reported, whilst columns 10 to 25 inclusive were used to create sequences—this being 16 columns—which matches the min/max value (= 16).

[>] 8 distinct states appear in the data:

 1 = 0

 2 = 1

 3 = 2

 4 = 3

 5 = 4

 6 = 5

 7 = 6

 8 = 7

[>] state coding:

 [alphabet] [label] [long label]

 1 0 0 0

 2 1 1 1

 3 2 2 2

 4 3 3 3

 5 4 4 4

 6 5 5 5

 7 6 6 6

 8 7 7 7

[>] 2000 sequences in the data set

[>] min/max sequence length: 16/16

Options within the **seqdef()** function can be used to assign additional attributes to the sequence object. The code snippet below adds long labels (‘labels= long.labels’) and replaces the numerical numbering of states with abbreviations (‘states=states.biofam’) to make it easier to interpret sequences when they are extracted. Prior to running the updated **seqdef()** function, lists are created that contain the new long labels (labels.biofam) and states (states.biofam).

labels.biofam = c("Parents", "Left", "Married", "Left/Married", "Child", "Left/Child", "Left/Married/Child", "Divorced")

states.biofam = c("P","L","M","LM","C","LC", "LMC", "D")

complex.sequence.object = seqdef(biofam, var=10:25, states = states.biofam, labels = labels.biofam)

After running the updated **seddef()** function, check the summary in the console to ensure that the correct labels have been attributed to states.

[>] state coding:

 [alphabet] [label] [long label]

 1 0 P Parents

 2 1 L Left

 3 2 M Married

 4 3 LM Left/Married

 5 4 C Child

 6 5 LC Left/Child

 7 6 LMC Left/Married/Child

 8 7 D Divorced

[>] 2000 sequences in the data set

[>] min/max sequence length: 16/16

**Step 2: Comparing sequences**

The **seqdist()** function is used to calculate the distance between each sequence in a specified sequence object (‘complex.sequence.object’). The resultant distance matrix is saved (‘distance.matrix’). In this example, Hamming distances are calculated (method="HAM"), a method which only uses substitutions to align sequences. Substitution costs have been based on transition rates between states using the ‘sm = "TRATE"’ option.

distance.matrix = seqdist(complex.sequence.object, method="HAM", sm = "TRATE")

After running **seqdist()**, the dialogue below will appear in the console. It is useful to review this dialogue to check that the function has executed as expected.

[>] 2000 sequences with 8 distinct states

[>] Computing sm with seqcost using TRATE

[>] creating substitution-cost matrix using transition rates ...

[>] computing transition probabilities for states P/L/M/LM/C/LC/LMC/D ...

[>] generated an indel of type number

[>] 537 distinct sequences

[>] min/max sequence lengths: 16/16

[>] computing distances using the HAM metric

[>] elapsed time: 0.27 secs

**Step 3: Cluster analysis**

The ‘cluster’ package enables several different forms of cluster analysis to be performed. The **hclust()** function is used to conduct Ward’s hierarchical clustering (‘method = "ward.D2"’) of a specified distance matrix (‘distance.matrix’). **hclust()** was designed to accept distance matrixes in a different format to that output by the **seqdist()** command; hence the need to do some data wrangling to make the distance matrix conform to the required format (‘as.dist(distance.matrix)’).

ward.hclust = hclust(as.dist(distance.matrix), method = "ward.D2")

**hclust()** can be used to undertake other forms of hierarchical clustering, which can be selected by changing the ‘method =’ option. Running this command generates several different pieces of information, which are saved as ‘ward.hclust’ so that they can be called into different commands later.

As Ward’s method is hierarchical, a useful visual tool for deciding on the number of clusters to retain is called a ‘dendrogram’ or cluster-tree. This visualisation shows how sequences combine into an initial set of clusters, and how these subsequent clusters can be combined at differing levels. The cluster tree (Figure 1) can be plotted using the **plot()** function, drawing on the result of the cluster analysis saved earlier.

plot(ward.hclust)

Height in the dendrogram represents the distance at which clusters join. Clusters are chosen by ‘cutting’ the tree at a preferred height, i.e., drawing a horizontal line across the diagram, with the intersection of that line and branches being the clusters retained. Researchers should look for large step-changes in the levels at which clusters combine.

**Figure 1: Cluster tree generated of sequences from the biofam data set, and how to choose the number of clusters to retain by cutting the cluster tree**



For the biofam data, a 4 or 5-cluster solution may be appropriate for further exploration. 5-clusters were selected in the paper from which the biofam data originate[[2]](#endnote-2). For illustrative purposes, we extract 5-clusters.

Running the **cutree()** function cuts the dendrogram at the appropriate level to produce the number of clusters indicated using the ‘k =’ option. The cluster that each case/person in the data set is assigned to is saved as ‘cluster.solution’.

cluster.solution = cutree(ward.hclust, k = 5)

Cluster membership can be added to the biofam data set as a new column (‘cluster.solution’) using the **cbind()** function, as shown in the code snippet below. Adding cluster membership onto the original data set enables cross-tabulation, ANOVA, regressions and other analyses to explore variation of characteristics not included in the sequence analysis, e.g., gender, ethnicity, and age.

biofam = cbind(biofam, cluster.solution)

At this point it is useful to look at the size of clusters. Using the **table()** function will give a breakdown of the number of people in each cluster, whilst the **prop.table()** function gives the proportion of people in each cluster.

table(cluster.solution)

prop.table(table(cluster.solution))

After running the **table()** function, the following dialogue will appear in the console screen. We can see that Cluster 1 was the largest cluster, containing 620 people, whilst Cluster 4 was the smallest cluster, with 139 people.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |   |   |   |   |
| 620 | 544 | 376 | 139 | 321 |   |   |   |   |

The following dialogue appears on the console screen after running the **prop.table()** function. Proportions sum to 1. Cluster 1 makes up roughly 31% of the sample (0.3100 \* 100), whilst Cluster 4 make up 7% (0.0695 \* 100).

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |   |   |   |   |
| 0.3100 | 0.2720 | 0.1808 | 0.0695 | 0.1605 |   |   |   |   |

**Step 4: Interpretation of clusters**

This section covers a limited number data visualisations to aid interpretation of clusters. The TraMineR website (https://traminer.unige.ch/) and associated publication[[3]](#endnote-3) provide a comprehensive overview of methods available to researchers.

Running the **seqmtplot()** function generates mean time plots for each cluster summarising the average time cluster members spent in each state. Time refers to the unit used when ordering states, e.g., survey waves, years, calendar months, days etc. For the biofam data, time was single year of age between 15 (a15) and 30 (a30) years old. The ‘group =’ option is used to split the visualisation by cluster membership (‘cluster.solution’). Appendix 1 provides the resultant plot.

seqmtplot(complex.sequence.object, group = cluster.solution)

Depending on the context, states that have extreme mean time values relative to others may be of interest. For example, Cluster 3 appears to be people who, on average, remain with their parents for prolonged periods of time. In contrast, Cluster 2 may represent a group of people who leave the parental home and who do not marry—owing to the relatively low mean time spent in any state other than ‘Left’.

The **seqdplot()** functiongenerates state distribution plots, split by cluster (see Appendix 2 for the resultant plot).

seqdplot(complex.sequence.object, group = cluster.solution)

State distribution plots are useful for describing changes in the composition of each cluster over time. For example, people in Cluster 1 appear to transition from living with their parents into living independently, being married, and having children by the age of 30 years old. However, because these plots aggregate information at each time point, they potentially mask nuances in people’s actual transitions between states.

The **seqrplot()** function extracts representative sequence plots for each of the clusters (see Appendix 3 for plots). This command uses the distance matrix (‘diss = distance.matrix’) to calculate the most ‘central’ and therefore representative sequence. Multiple sequences can be extracted by changing the ‘nrep =’ option.

seqrplot(complex.sequence.object, group = cluster.solution, diss = distance.matrix, nrep = 1)

The visualisation generated after running **seqrplot()** includes several pieces of information, discussed in greater detail in an associated publication on this type of visualisation[[4]](#endnote-4). Simply focusing on the sequence(s) represented in these plots is instructive for cluster interpretation. For example, the most central sequence for Cluster 3 was to remain in the parental home until the age of 29 years old, before moving out at age 30 years old.

The **ggseqtrplot()** function can be used to visualise transitions between different states, by plotting the transition matrix. Rather than creating a single visualisation containing transition matrixes for each cluster, the code snippet below will generate a transition matrix plot for a specified cluster (see Appendix 4 for the resultant plot). Altering the number after the ‘cluster.solution==’ option will generate the transition matrix for the corresponding cluster, i.e., ‘cluster.solution==2’ for Cluster 2 etc.

Setting ‘dss = TRUE’ calculates transitions between distinct states, e.g., in line with the definition of transitions as change in states between time-points. The ‘x\_n.dodge = 2’ option stops the labels on the x-axis from overlapping and becoming unreadable.

ggseqtrplot(complex.sequence.object[cluster.solution==1,], x\_n.dodge = 2, dss = TRUE)

Cells sum across rows in the transition matrix plot and indicate the proportion of people who transition from one state in time t to different states in the following year (t + 1). As an example of how to interpret these plots, for Cluster 1, of transitions from the parental home (‘Parents’), most were into independent living and marriage, 40% (0.40 \* 100) without having children and 28% having children (0.28 \* 100).

**Appendix 1: Mean time spent in each family state by cluster**

****

**Appendix 2: Distribution of family life states** **by cluster, between the ages of 15 (a15) and age 30 (a30) years old**

****

**Appendix 3: Medoid** **family life state sequences by cluster, between the ages of 15 (a15) and age 30 (a30) years old**



**Appendix 4: Family state transition rate matrix for Cluster 1, showing transitions between states at a time-point t and in the following time-point (t+1)**

****

1. More detail on the biofam data set can be found at: https://www.rdocumentation.org/packages/TraMineR/versions/1.1/topics/biofam [↑](#endnote-ref-1)
2. Müller NS, Studer M & Ritschard G (2007) Classification de parcours de vie à l’aide de l’optimal matching. In: XIVe Rencontre de la Société francophone de classification (SFC 2007), Paris, 5 - 7 September 2007, pp. 157–160. [↑](#endnote-ref-2)
3. Gabadinho, A., Ritschard, G., Müller, N. & Studer, M. (2011) Analyzing and Visualizing State Sequences in R with TraMineR.” Journal of Statistical Software. 40(4):1–37 [↑](#endnote-ref-3)
4. Gabadinho, A. & Ritschard, G. (2013) Searching for typical life trajectories applied to child birth histories. In: Lévy, R. & Widmer, E. eds. Gendered life courses. pp.287-312. Vienna: LIT

National Centre for Research Methods (NCRM)
Social Sciences
Murray Building (Bldg 58)
University of Southampton
Southampton SO17 1BJ
United Kingdom

**Web** www.ncrm.ac.uk
**Email** info@ncrm.ac.uk
**Tel** +44 23 8059 4539
**Twitter** @NCRMUK [↑](#endnote-ref-4)