

seqimpute: Treating Missing Data in Categorical Sequence Data Through Multiple Imputation

Kevin Emery  Anthony Guinchard Kamyar Taher André Berchtold
University of Geneva University of Lausanne University of Lausanne University of Lausanne

Abstract

This article describes the tools available in the **seqimpute** package. It focuses on the treatment of missing data in longitudinal categorical data. It mainly implements two methods to handle missing data by multiple imputation, namely MICT and MICT-timing. In addition, the package provides tools to visualise missing data in longitudinal categorical data. In particular, this article provides a practical demonstration of the imputation algorithms together on a typical sequence analysis workflow, namely deriving a typology and then using it as a variable in a regression analysis.

Keywords: missing data, multiple imputation, categorical longitudinal data, MICT, MICT-timing, sequence analysis.

1. Introduction

Missing data is a common issue that can significantly affect the results of statistical analyses. For example, [Lall \(2016\)](#) reanalysed quantitative studies published in two leading political science journals over a five-year period where missing values were initially handled by deleting all cases with missing values. When multiple imputation was applied, the statistical significance of the results changed in almost half of the studies, with some analyses even producing opposite conclusions. This highlights the strong influence that the method chosen to deal with missing data can have on research results.

While the simple approach of excluding cases with missing values may seem attractive, it is often inadequate. This method can result in a significant reduction in the number of cases available, thereby reducing the statistical power ([King, Honaker, Joseph, and Scheve 2001](#)) and hence the ability to detect effects. The problem is particularly pronounced for categorical sequence data, where even a single missing value can result in the removal of an entire trajectory. Furthermore, retaining only complete cases can introduce bias and lead to incorrect inferences. [Perkins, Cole, Harel, Tchetgen Tchetgen, Sun, Mitchell, and Schisterman \(2018\)](#) illustrated this by showing how removing cases with missing data incorrectly suggested a protective effect of smoking on the risk of spontaneous abortion.

Multiple imputation is a versatile approach for addressing missing values. It creates multiple complete datasets by replacing missing values with plausible values. Multiple complete datasets are created, rather than just one, to account for the inherent uncertainty associated with missing data. Statistical analysis is then performed independently on each complete dataset, and the results of these statistical analyses are finally combined ([Rubin 1987](#)). A

critical step in this process is determining how to generate plausible values to replace the missing ones. Various methods exist, implemented in packages such as **VIM** (Kowarik and Templ 2016) and **Hmisc** (Harrell Jr 2024) in R.

However, few approaches apply to longitudinal categorical data. For instance, **jomo** (Quartagno and Carpenter 2023) and **Amelia** (Honaker, King, and Blackwell 2011) rely on multivariate normal joint modeling, which assumes that the data structure can be adequately represented by a multivariate normal distribution. This approach can be extended to categorical data by decomposing it into multiple binary variables. However, this strategy has significant drawbacks: the number of model parameters grows rapidly, leading to overfitting, and the covariance matrix becomes non-invertible when collinearity exists between binary variables (Audigier, Husson, and Josse 2017)—a scenario that is particularly likely in longitudinal data. The **mice** package (Van Buuren and Groothuis-Oudshoorn 2011), which applies multiple imputation through Fully Conditional Specification (FCS), is highly flexible, supports a wide range of data types. Additionally, it includes tools to perform statistical analysis on multiple imputed datasets. Despite its versatility, **mice** does not explicitly account for the unique patterns of missingness commonly observed in longitudinal categorical data, such as gaps—consecutive missing values. The **PST** package (Gabadinho and Ritschard 2016) enables imputations using variable-length Markov Chain models. However, a Markovian model can be overly simplistic for complex longitudinal data, such as life-course trajectories. In particular, Markovian models rely solely on previous information to impute missing values, overlooking the potential benefits of incorporating future information to improve imputation quality.

To address these limitations, the **seqimpute** package introduces two methods specifically designed for longitudinal categorical data: the MICT algorithm (Halpin 2012) and the MICT-timing algorithm (Emery, Studer, and Berchtold 2024). These methods take a recursive approach to impute gaps of missing data, starting from the edges and working inward.

This document provides a practical demonstration of the imputation algorithms implemented in the package, together with some visualisation tools. The illustrative example shows a typical sequence analysis workflow: construction of a typology by cluster analysis and subsequent use of this typology in a regression analysis (Piccarreta and Studer 2019). For a general overview of sequence analysis, see Liao, Bolano, Brzinsky-Fay, Cornwell, Fasang, Helske, Piccarreta, Raab, Ritschard, Struffolino *et al.* (2022). Practical guidance for studying sequences in R is available in Gabadinho, Ritschard, Müller, and Studer (2011), while Studer (2013) more specifically focus on clustering. However, the imputation methods presented here are not limited to this specific application of sequence analysis, nor to sequences. They are broadly applicable to a wide range of statistical analyses involving longitudinal categorical data.

The rest of this document follows the following structure.

- Section 2 discusses the challenges posed by missing data and highlights the flexibility of multiple imputation as a treatment.
- Section 3 introduces the two algorithms implemented in the **seqimpute** package: MICT and MICT-timing.
- Section 4 describes the dataset used for illustration.
- Section 5 outlines the four steps involved in using clustering in a regression analysis when data are incomplete.

1. Description and visualization of missing data (subsection 5.1)
2. Multiple imputation (subsection 5.2)
3. Typology construction (subsection 5.3)
4. Regression (subsection 5.4)

- Section 6 provides guidelines tailored to the imputation algorithms implemented in this package.
- Section 7 highlights additional features of the package, including parallel computing and the use of random forest imputation models.
- Section 8 focuses on additional functionalities available in the package.
- Section 9 concludes the document with a brief summary.

2. Missing data issues and treatment

This section provides a broad overview of missing data challenges and the principles of multiple imputation. For a more comprehensive discussion on missing data, including detailed statistical explanations, refer to [Little and Rubin \(2019\)](#) or [Molenberghs, Fitzmaurice, Kenward, Tsiatis, and Verbeke \(2014\)](#), and for an in-depth treatment of multiple imputation, see [Van Buuren \(2018\)](#).

This section covers the following key points, illustrated with an example:

- The issues posed by the deletion of missing data, called “complete case analysis”.
- A brief introduction to multiple imputation.
- The application of clustering with multiple imputation.
- The scenarios in which multiple imputation is applicable.

To illustrate these concepts, we use a synthetic dataset composed of categorical sequence data measured across ten time points, with each measurement coded as either “state A” or “state B.”

This dataset includes missing values, presenting a challenge for analysis, since many statistical methods, such as regression and cluster analysis, are designed to operate on complete datasets.

Complete case analysis

The straightforward approach, often the default in many software packages such as `lm()` and `glm()` in R for linear and logistic regression, respectively, is to discard observations with missing values. This method is known as complete case analysis.

This approach has two main issues ([Little and Schenker 1995](#)). First, it reduces the amount of information available. In the example, only 6 trajectories remain in the dataset (Figure 2). Even if a trajectory has only one missing value, such as the first trajectory, it is completely

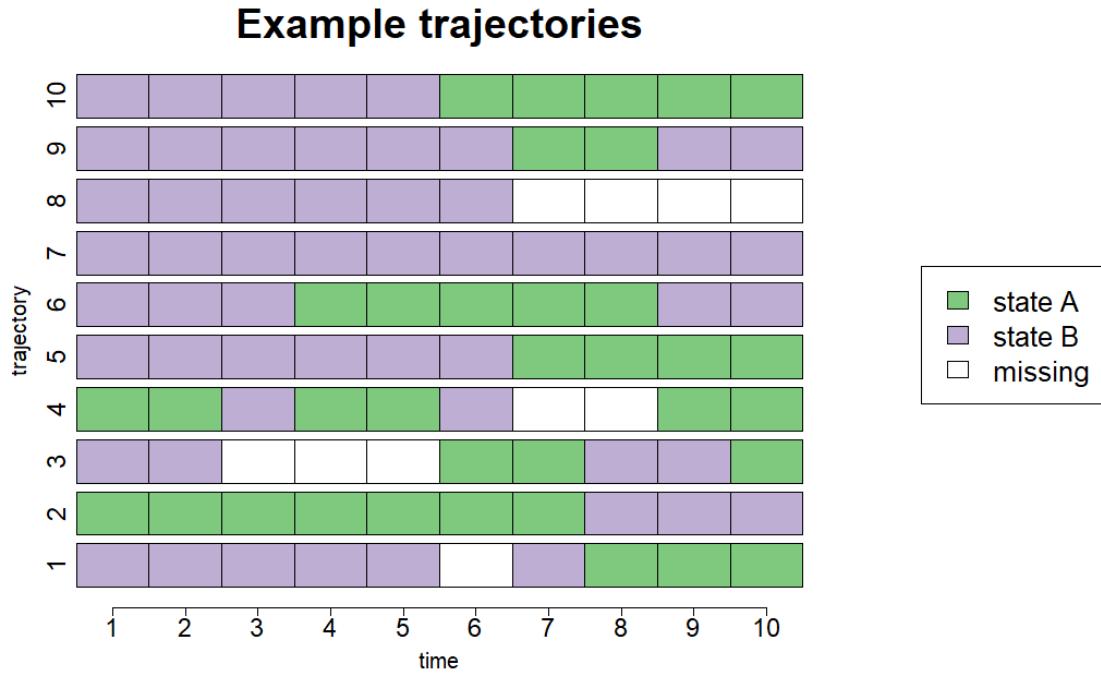


Figure 1: Dataset composed of 10 trajectories used for illustration.

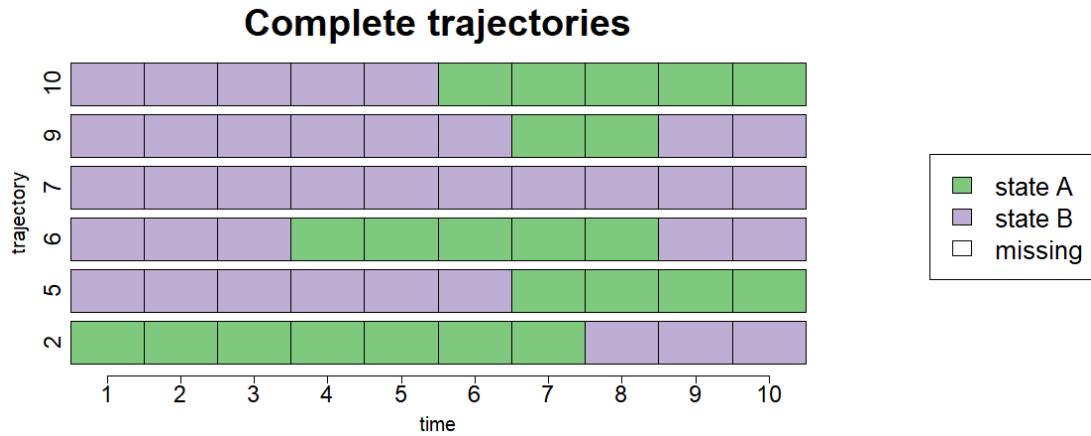


Figure 2: Remaining trajectories after a complete case analysis.

removed from the analysis. This can reduce statistical power and hinder the detection of effects.

The second problem of missing data relates to the reasons for non-response, which are often specific to certain individuals or circumstances. For example, individuals in vulnerable situations such as unemployment, migrant background or poor health are more likely to drop out of longitudinal studies (Rothenbühler and Voorpostel 2016). As a result, these vulnerable situations may be under-represented in the subsample, leading to potential biases in various statistical measures of vulnerability. Similarly, trajectories with higher transition rates

are more prone to missing values (Müller, Sapin, Gauthier, Orita, and Widmer 2012). This behaviour is observed in trajectories 3 and 4 of the example.

Imputation

Imputation is based on the idea of replacing missing values with some reasonable values. This is appealing because it would result in a dataset with no more missing values. However, the problem with imputing missing values once, which is called “single imputation,” is that it treats missing values as if they were observed and overlooks the inherent uncertainty associated with missing data. To address this limitation, multiple imputation is used. As illustrated in Figure 3, the process involves three main steps:

1. **Creation of completed datasets:** Missing values are replaced with values multiple times, generating several complete datasets.
2. **Statistical analysis:** The statistical analysis that would have been applied if the data had been complete is usually carried out independently on each of the completed datasets.
3. **Combination of results:** In the final step, the results of the statistical analysis computed on each completed dataset are combined into a single result. Rubin (1987) formulated rules for this.

For example, we might be interested in the relative risk of moving to state B versus staying in state A while in state A. After multiple imputations, we would estimate this relative risk and its variance on each of the completed datasets. We would then combine these estimates and their variance into a single relative risk and its variance. The calculated variance would typically be useful in testing the hypothesis of whether or not this risk is significantly different from 1.

Clustering and multiple imputation

While the standard multiple imputation framework is well suited to most statistical analyses, clustering presents unique challenges. Unlike parameter estimation, clustering involves grouping observations on the basis of similarity, which complicates the pooling of results across multiple imputed datasets. Even small variations in the data can lead to significant differences in the resulting clusters and their number.

Several approaches have been proposed to overcome this challenge. One method, proposed by Halpin (2012), involves stacking all completed datasets together and clustering this stacked dataset. Observations are then assigned to clusters based on their frequency among the imputed replications.

Alternatively, consensus clustering methods attempt to reconcile differences between clusterings obtained from multiple imputed datasets. For example, Basagaña, Barrera-Gómez, Benet, Antó, and García-Aymerich (2013) suggests first determining the number of clusters by identifying the most frequently observed count across imputed datasets and then assigning each sequence to the cluster to which it most frequently belongs, while Faucheu, Resche-Rigon, Curis, Soumelis, and Chevret (2021) suggests identifying common clustering patterns across the multiple clusterings generated to build a single clustering.

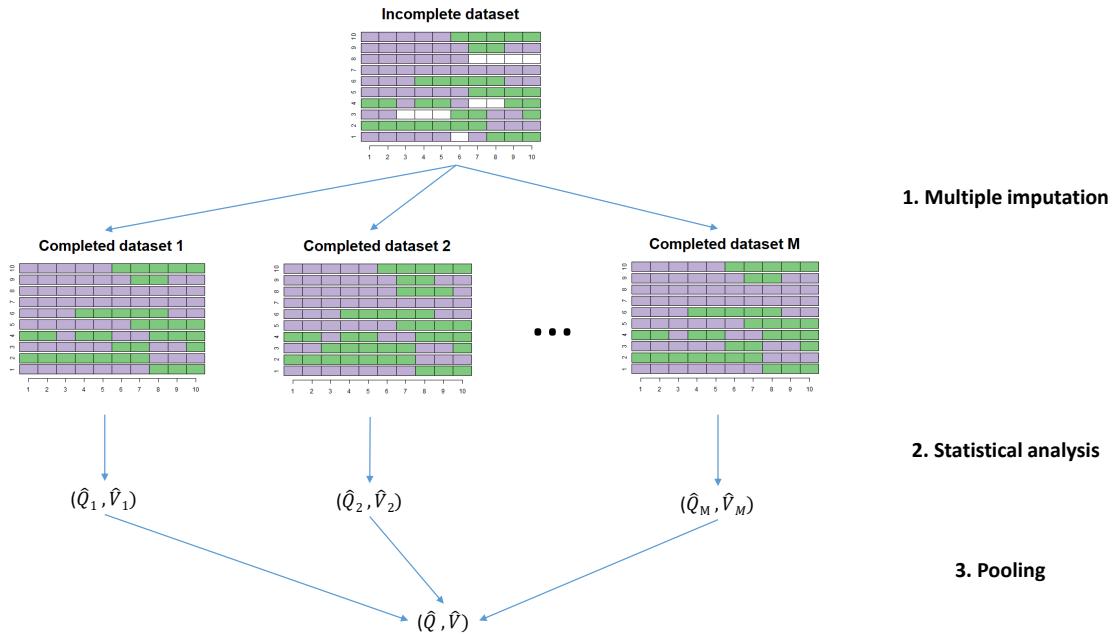


Figure 3: Illustration of three steps of a multiple imputation. The missing data are imputed several times, yielding M completed datasets. The estimated parameter and its variance is computed. These estimated parameters are then pooled in a single estimate and its variance.

In our illustrative example, we use the stacking method. However, consensus clustering is a promising alternative that may address some of the limitations of the stacking approach. Further research is required to fully evaluate the most appropriate alternative.

Scenarios where multiple imputation is applicable

Not all missing data are equivalent. At least two distinctions can be made. First, it is important to differentiate between structural missing data and genuine missing data. Second, the underlying mechanisms driving the missingness must be considered.

Structural missing data refer to values that are supposed to be missing and can be further categorized into two categories: cases where no value is meaningful and cases where the value can be determined with certainty. An example of the first category is a missing occupation for individuals who are still in education. An example of the second is individuals who have died and have missing values for subsequent health states. In such cases, it is appropriate to predefined a specific state, such as "dead," for these missing values.

The suitability of a method for dealing with missing data depends on the underlying mechanism causing the missing data. Rubin (1976) identified three mechanisms of missing data:

- Missing Completely at Random (MCAR): Data are missing entirely by chance. For example, individuals forgetting to report their situation for no apparent reason.
- Missing At Random (MAR): Data are missing for reasons explained by observed variables. For instance, if women are more likely to have missing data, or if the probability

of missing at time t depends on the state observed at time $t-1$ (e.g., unemployment at time $t-1$ increases the likelihood of missing data at t).

- Missing Not at Random (MNAR): Data are missing due to reasons related to the missing value itself, such as systematically omitting periods of unemployment.

Multiple imputation produces unbiased results under MCAR and MAR conditions ([Little 1992](#)). However, addressing MNAR data remains challenging, as most methods, including multiple imputation, are prone to bias in such cases.

Since these mechanisms depend on unobserved data, identifying the mechanism with certainty is generally impossible. MCAR is rarely a realistic assumption, and real-world datasets often involve a mix of MAR and MNAR mechanisms. Research by [Gomer and Yuan \(2023\)](#) suggests that multiple imputation can still be a viable approach in such scenarios.

When MNAR mechanisms are suspected, a common strategy, as described by [Van Buuren \(2018\)](#), is to include additional explanatory variables in the imputation models to account for relationships that might explain the missingness. Another approach is sensitivity analysis, which assesses how results change under different MNAR assumptions.

3. Description of the imputation algorithms

As discussed in the previous section, multiple imputation begins with the imputation process itself. The **seqimpute** package implements two methods specifically designed for categorical sequence data, where missing values often appear as gaps rather than isolated time points.

This section provides an overview of the principles underlying the two algorithms—MICT and MICT-timing. Readers seeking detailed explanations can refer to [Halpin \(2012\)](#) for MICT and to section 4.2 of [Emery et al. \(2024\)](#) for MICT-timing.

The MICT algorithm introduced by [Halpin \(2012\)](#) handles gaps by iteratively imputing missing values from their edges. However, it assumes homogeneous transition probabilities throughout a trajectory, which may not hold in real-world data. For example, in life course analysis, transition rates between education and work vary over time: transitions to work are rare in childhood but common between the ages of 16 and 30. This can lead to implausible imputations, such as transitions to work in childhood. The MICT-timing algorithm was developed to account for such temporal variations.

We first describe the MICT algorithm, before outlining what distinguishes MICT-timing from MICT.

MICT

We describe the main features of the MICT algorithm. We first consider the order in which missing values are imputed, starting at the dataset level, before focusing on individual gaps. Then, we examine the variables included in the imputation model for a given missing value and consider the construction of the data used to fit an imputation model. Next, we discuss how an imputed value is concretely obtained from the imputation model. Finally, we discuss the special considerations required for handling beginning and ending gaps due to their unique edge structure, and outline the process of creating multiple complete datasets.

1. Order of imputation

The order in which the missing data are imputed is shown in Figure 4. Let's highlight two characteristics. First, the missing gaps are imputed recursively from the edges. This approach serves two purposes: missing data near the edges of the gap are generally easier to impute because they are adjacent to observations, and by imputing from the edges inwards we maintain longitudinal consistency between the imputed values. Then only the position of a missing value in a gap is important, not its position in the sequence. For example, both missing values labelled “2” are imputed at the same time by the same imputation model, even though they do not occur at the same time, because once “1” is imputed, they are both the last value of a gap of length 2.

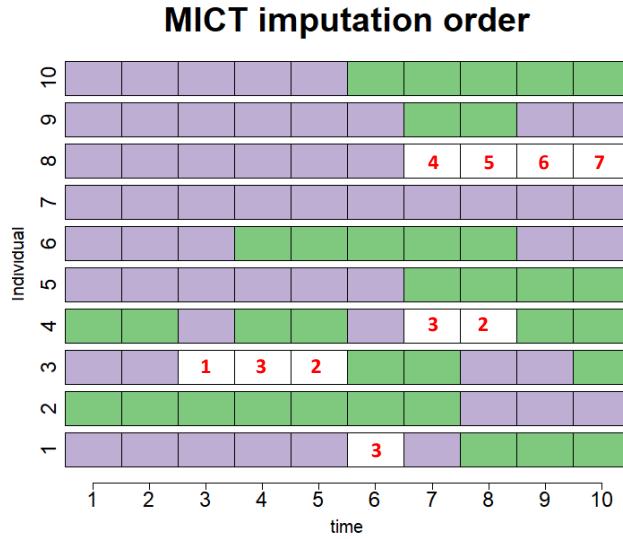
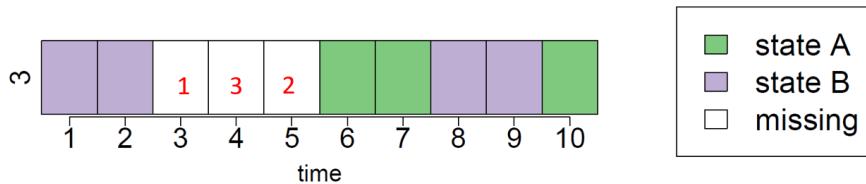


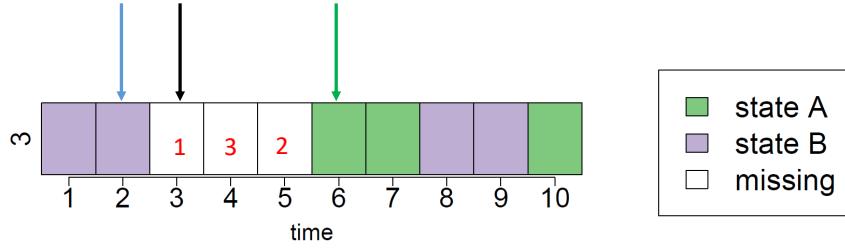
Figure 4: Order of the imputation for the MICT algorithm.

2. Information from the trajectory used to determine an imputed value

Let's focus on the imputation of the first missing value of the third sequences:



The goal is to ensure consistency between the imputed values and the other observed values within the trajectory. Therefore, at least the preceding and subsequent observations around a gap are integrated in the determination of an imputed value. For the first missing value of sequence 3, the individual was in state B (indicated by the blue arrow) just before the gap and in state A three time points after (indicated by the green arrow):

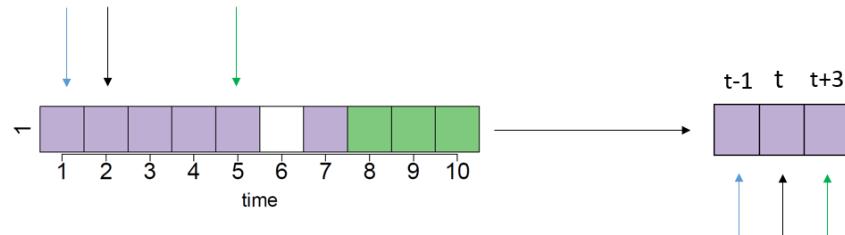


The values immediately before and after a gap represent the minimum information to be included in the prediction model. However, additional predictors, both before and after the gap, may be included if relevant. In addition, covariates related to the missing data mechanism or variables intended for subsequent statistical analysis should also be included. Section 6 provides guidelines for selecting the appropriate information to include in the imputation model.

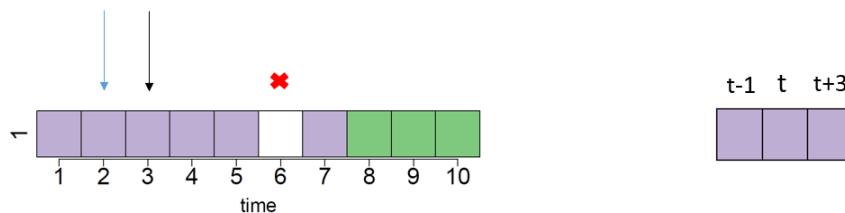
3. Build the training dataset

To construct the data used to fit the imputation model for the first missing value in sequence 3, the algorithm identifies instances where both the previous state and the state three time points later are observed.

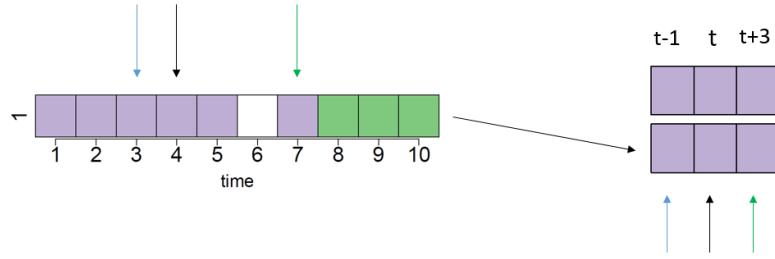
For example, in the case of the first sequence, the second time point is the first occurrence where both the preceding state and the state three time points after are observed:



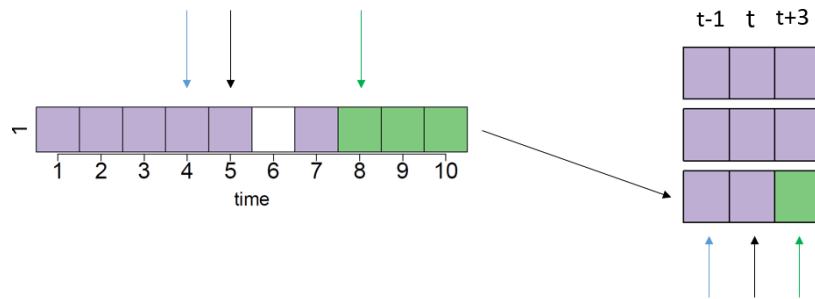
The time 3 has a missing value three time points later and is hence not included:



The fourth time point is also included:



and finally the fifth one:



The sixth time point is missing and is therefore not included, while subsequent time points have no value that occurs three time points later. In summary, the first sequence contains three observed patterns similar to the one we want to impute.

This process is applied to all sequences in the dataset, resulting in 44 observations used to fit the imputation model.

4. Determine an imputed value

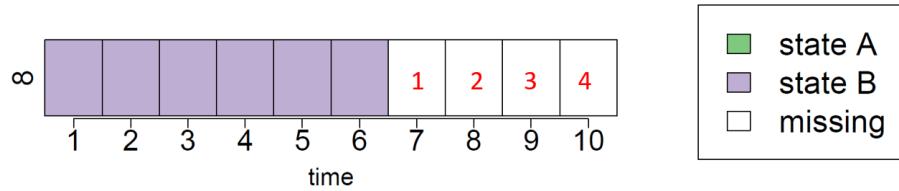
A logistic model, or a multinomial model for cases with more than two categories, is then fitted to these data. This model is used to determine the probabilities of belonging to state "A" or "B." Finally, a value is assigned to the missing observation based on these probabilities. We examine each step in details.

- (a) *Fitting the logistic model* A logistic model is fitted to the data extracted in point (a). The dependent variable is t and the two independent variables are $t-1$ and $t+3$.
- (b) *Probability of belonging to each state* Given that the previous state for the missing value to be imputed was "A" and that three time points later there is a state "B", the probabilities of being in state "A" or "B" are determined.
- (c) *Imputed value* A value is drawn at random based on the derived probabilities.

5. Starting and ending gaps

Because these gaps have only one edge, imputations start from the far right to the left for start gaps and from the far left to the right for end gaps.

For example, the imputation order for the end gap in the eighth sequence is:



Then, in a similar way to the imputation of middle gaps, the algorithm looks in each sequence for each time that an observed state also has the previously observed state. A logistic or multinomial model is fitted, probabilities are estimated and an imputed value is drawn.

6. Multiple imputed datasets

Imputed values are drawn based on probabilities computed from logistic or multinomial models, which introduces a random element into the imputation process. As a result, the imputation process is repeated several times to generate multiple imputed records. Choosing the appropriate number of imputed datasets is discussed in Section 6.

MICT-timing

The MICT-timing algorithm is an extension of the MICT algorithm designed to address a key limitation of the latter: its assumption that position in the trajectory is irrelevant. We go back to the different points that were described previously for the MICT algorithm in order to highlight the differences between MICT-timing and MICT.

1. Order of imputation

A key difference between the MICT and MICT-timing algorithms is illustrated in Figure 5, which shows the order in which missing values are imputed. We have seen previously that the third missing value in sequence 3 and the second missing value in sequence 4 (both labelled "2") are imputed at the same stage and by the same imputation model by the MICT algorithm. For MICT, it does not matter that they occur at different times. In contrast, the MICT-timing algorithm differentiates between these values, and impute them with two different models.

2. Information from the trajectory used to determine an imputed

Similarly to MICT, at least the time points preceding and following the gaps are included in the prediction model.

3. Build the training dataset

As noted above, the MICT algorithm identifies similar configurations in all sequences, regardless of their position, and fits a model using these data. In contrast, the MICT-timing algorithm only considers observations that occur at the same time in other sequences. Thus, unlike MICT, each sequence contributes at most one observation.

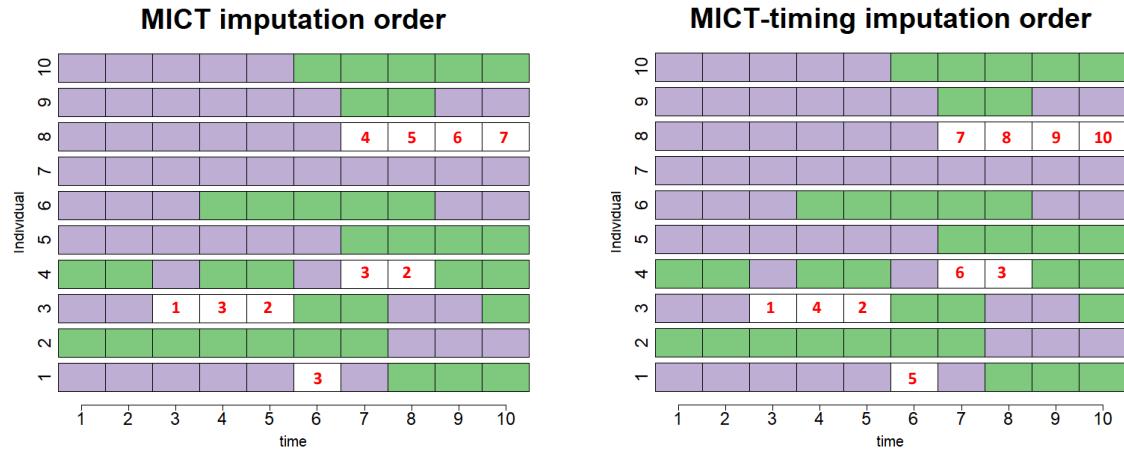
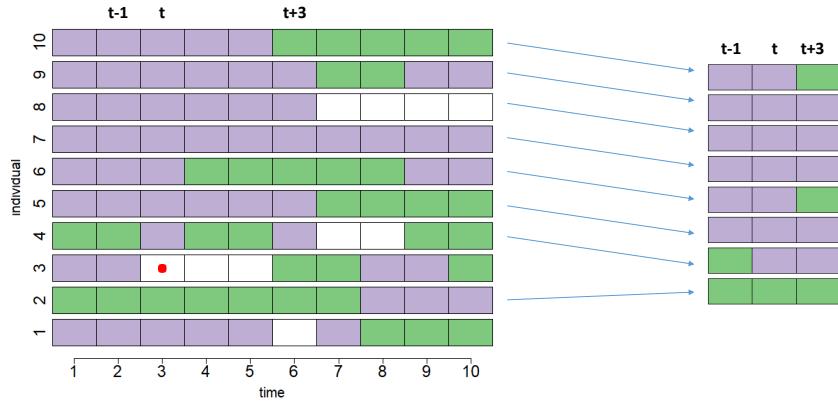


Figure 5: Comparison of the order of the imputation for the MICT algorithm (left) and the order of the imputation of the MICT-timing algorithm (right).

If we focus on the imputation of the first missing value of sequence 3, any other sequence that does not have missing time points 2, 3 and 6 is included, which is the case for every sequence except the first:



Note that in this case the imputation is based on eight observations, whereas with the MICT algorithm it was 44. Since the performance of the logistic and multinomial models depends on the sample size, it may be necessary to include not only the observations that occur at the same time, but also the previous and subsequent observations if there are very few sequences in the dataset. This is discussed in Section 6.

4. Determine an imputed value

This part are similar to MICT. A logistic (or multinomial if there are more than two categories) model is fitted, probability are estimated, and an imputed value is drawn.

5. Starting and ending gaps

The order of imputation of starting and ending gaps is the same as for MICT, but as for middle gaps only consider the observations that occur at the same time points in other sequences to fit the logistic or multinomial imputation model.

6. Creation of multiple imputed datasets

As with MICT, the whole process of imputation is repeated several times to produce multiple complete datasets.

4. Sample application

The illustrative analysis detailed in section 5 aims to cluster the dataset and use the resulting clustering as a dependent variable in a regression analysis. This is a typical application in sequence analysis (Piccarreta and Studer 2019).

We assume that the reader is familiar with the **TraMineR** package (Gabadinho *et al.* 2011), a widely used tool for analysing sequence data. Importantly, many of the visualisation features in **seqimpute** extend and build upon the capabilities of **TraMineR**.

We use a dataset tracking the transition from education to employment within a cohort in Northern Ireland (McVicar and Anyadike-Danes 2002). This dataset, which is available in the **TraMineR** package, originally has no missing data. We deliberately introduce missing data to illustrate MICT and MICT-timing.

We begin by retrieving the dataset from the **TraMineR** package and creating a state sequence object, which is the format of sequences in **TraMineR**. We then introduce missing data using a dedicated function in the **seqimpute** package. Finally, we briefly discuss the limitations of relying solely on complete case analysis as a strategy for dealing with missing data. The statistical analysis itself is carried out in Section 5.

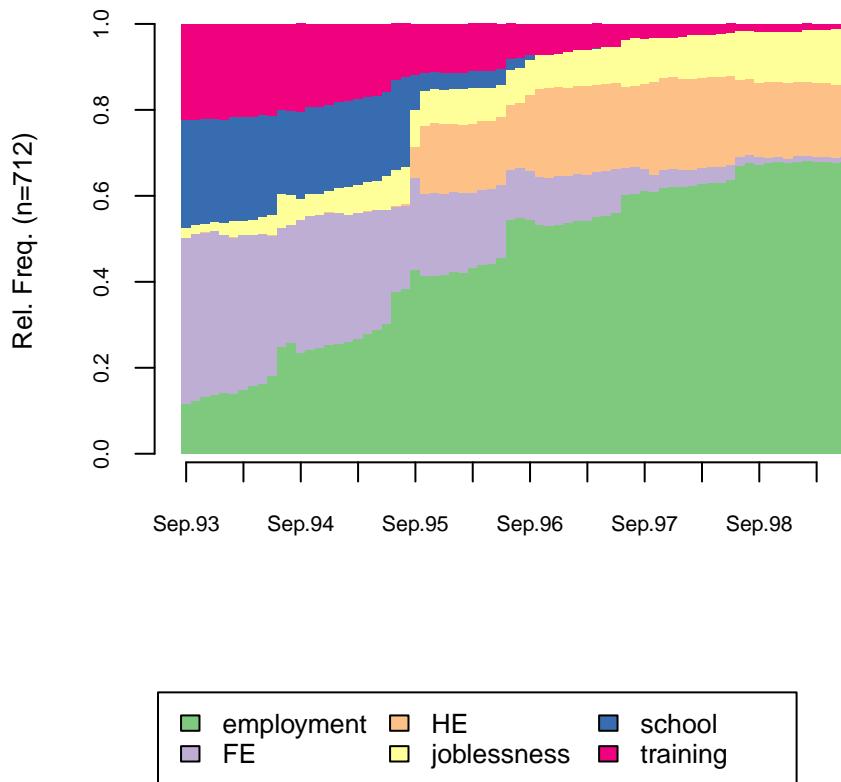
1. Preparation of the dataset

We load the **TraMineR** package and then fetch the data set named `mvad`. Using the `seqdef()` function from the **TraMineR** package, we create a state sequence object by specifying the columns containing the trajectories with the `var` argument, defining the possible states (the `alphabet`), and setting the interval between tick marks and labels in the plots with the `xtstep` argument.

```
R> library("TraMineR")
R> data(mvad)
R> mvad.alphab <- c("employment", "FE", "HE", "joblessness", "school", "training")
R> mvadseq <- seqdef(mvad, var=17:86, alphabet=mvad.alphab, xtstep=6,)
```

We visualize the cross-sectional state frequency at each time point with a state distribution plot (`seqdplot()` function of the **TraMineR** package):

```
R> seqdplot(mvadseq, border=NA)
```



We observe that at the beginning of the trajectories, the majority of individuals are engaged in education (school, training or further education), while towards the end of the trajectories, a large proportion have moved into employment. In particular, 13.1% of individuals are unemployed at the end of the trajectories.

3. Generation of missing data

We introduce small gaps of missing data into the `mvad` dataset, especially during periods of unemployment, reflecting real-world situations.

To achieve this, we load the `seqimpute` package and use the `seqaddNA()` function to simulate missing values. This function generates missing data following a Markovian logic, where the `states.high` argument specifies the states with a higher probability of the subsequent state being missing. In this example, we use the default parameters of the function. A detailed discussion of the `seqaddNA()` function is provided in Section 8.

As the process of simulation implies some randomness, we set a seed for reproducibility. Moreover, the `var` argument is used to specify the columns of the dataset that contain the trajectories.

```
R> library("seqimpute")
R> set.seed(2)
```

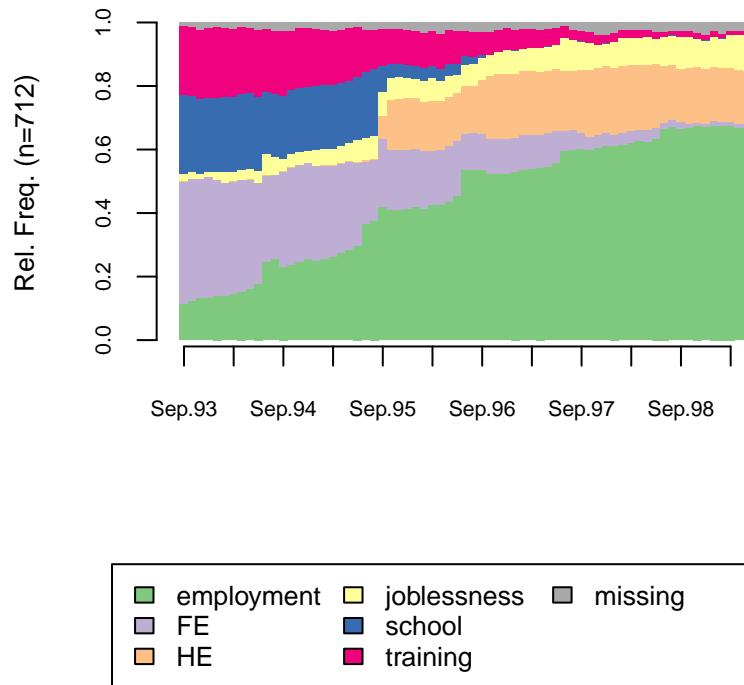
```
R> mvad.miss <- seqaddNA(mvad, var=17:86, states.high="joblessness")
```

We have generated small gaps of missing data, which are more likely to occur after unemployment. In the next section, we examine these missing values in more detail.

4. Limitations of complete case analysis

To illustrate the limitations of complete case analysis, we examine the state distribution at each time point of the complete trajectories. The `seqcomplete()` function allows to extract all the trajectories without missing data:

```
R> mvadseq.miss <- seqdef(mvad.miss, 17:86, xtstep=6, alphabet=mvad.alphab,
+ right=NA)
R> mvadseq.cca <- seqcomplete(mvadseq.miss)
R> seqdplot(mvadseq.cca, border=NA)
```



Unemployment is underrepresented in the trajectories with no missing data. For example, we observe 3.1% of jobless situations at the last time point, while it is 13.1% in the original dataset. Although these differences are clearly amplified by the way we generated missing values, they illustrate the kind of results that can emerge from a complete case analysis.

5. Step-by-step analysis

In this section, we illustrate the steps taken to cluster data, where missing data are treated by multiple imputation using the MICT algorithm, and then used in a subsequent regression analysis. We examine the social reproduction of unemployment.

The four main steps we follow, along with their objectives, are as follows:

1. Description and visualization of missing data

The goal here is multifaceted. First, we want to assess the magnitude of the missing data problem, focusing in particular on the proportion of missing data, its patterns, and the mechanism governing missing data. In addition, this step helps to identify values that should not be imputed, either because these values could be known with certainty or because no value makes sense.

2. Imputation

This is the first step in a multiple imputation process. We create multiple complete datasets, using MICT algorithm as the imputation method.

3. Typology creation

We extract a typology of typical patterns of transition from education to employment from the several complete datasets built in the previous step. In addition, we aim to identify the cluster that captures trajectories leading to unemployment.

4. Regression

In this step, we use logistic regression to study the social reproduction of unemployment. Specifically, the dependent variable is the cluster membership that captures trajectories leading to unemployment, while the father's unemployment serves as the independent variable.

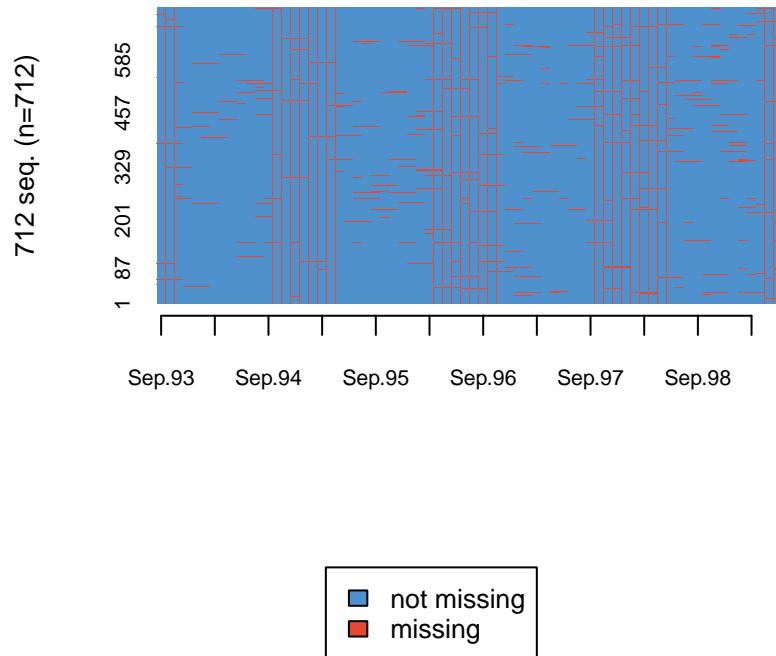
We provide detailed explanations of how to carry out each of these steps.

5.1. Description and visualization of missing data

We begin by demonstrating the use of various visual tools available in the **TraMineR** package, as well as those developed for **seqimpute**, to assess the prevalence of the missing data problem and to identify values that are best left unimputed. We then discuss specific methods for assessing the mechanisms underlying missing data, which is critical for evaluating the impact of any data handling method on the resulting analyses.

To get an initial overview of the proportion of missing data across the trajectories, the state distribution plot can be constructed with the `seqdplot()` function of the **TraMineR** package. This visualization provides the distribution of states at each time point.

```
R> seqdplot(mvadseq.miss, border=NA, with.missing=TRUE)
```

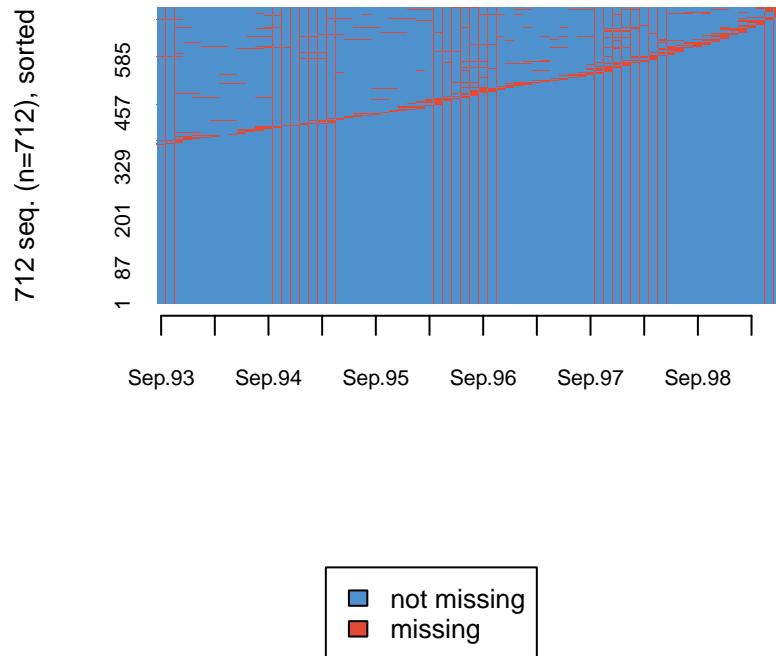


We observe that missing data increases slightly over time, but the proportion of missing values remains below 5% at each time point.

Some visualization tools are provided in the **seqimpute** package to better examine the patterns of missing data. These plotting functions are based on the **seqplot()** function from the **TraMineR** package.

To display all patterns of missing data among the trajectories, one can use the **seqmissIplot()** function.

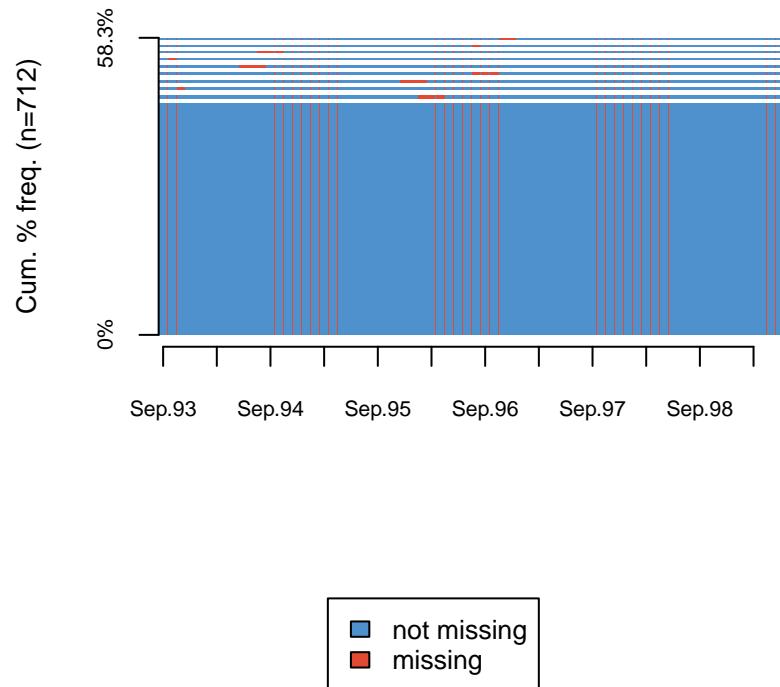
```
R> seqmissIplot(mvadseq.miss, border=NA)
```



This plot highlights that missing data tends to occur in the form of very short gaps.

`seqmissIplot()` also facilitates a closer examination of late entry and attrition patterns, thanks to its `sortv` argument. For example, by setting `sortv="from.end"`, the trajectories are organized according to when the last missing value occurs.

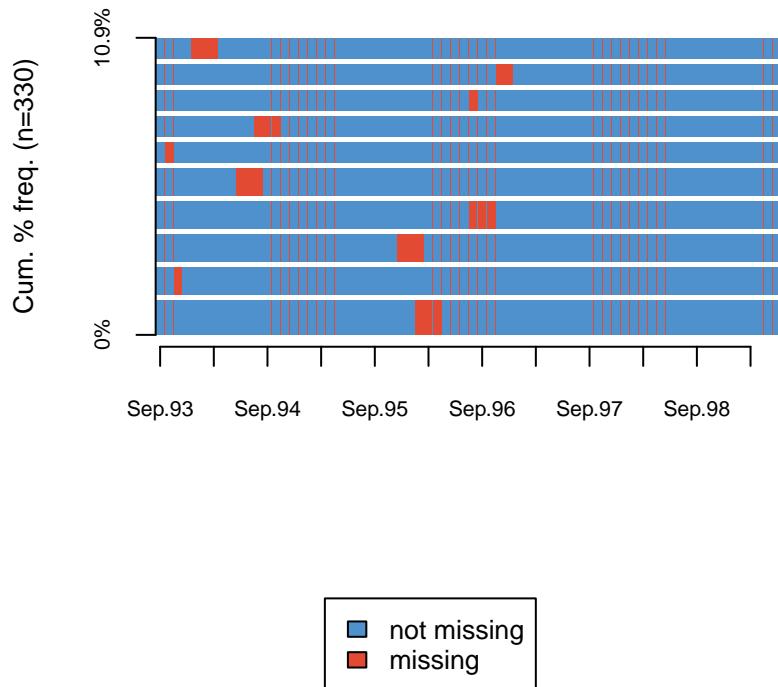
```
R> seqmissIplot(mvadseq.miss, sortv="from.end")
```



In this example, we do not observe anything systematic in terms of attrition. Patterns such as attrition need careful consideration. For example, in health trajectories, patterns of attrition may signal the death of an individual, making imputation inappropriate. Such cases need to be addressed before imputations are considered.

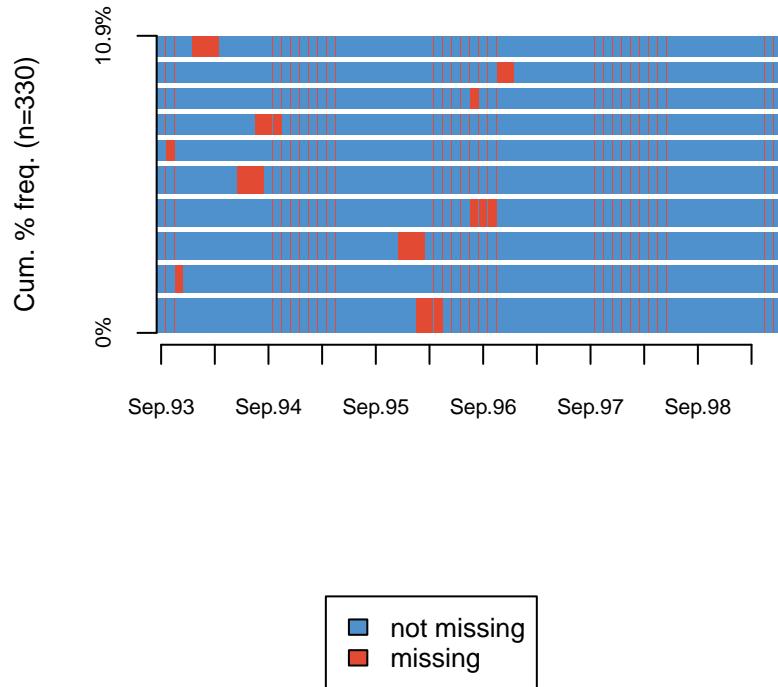
The `seqmissfplot()` shows the most frequent patterns of missing data within a sequence:

```
R> seqmissfplot(mvadseq.miss, border=NA)
```



We observe that the most common pattern is to have no missing value. This pattern makes it difficult to see the plot. Therefore, we can have a clearer view by displaying only trajectories with at least one missing value by setting the `with.complete` argument to `FALSE`.

```
R> seqmissfplot(mvadseq.miss, with.complete=FALSE, border=NA)
```



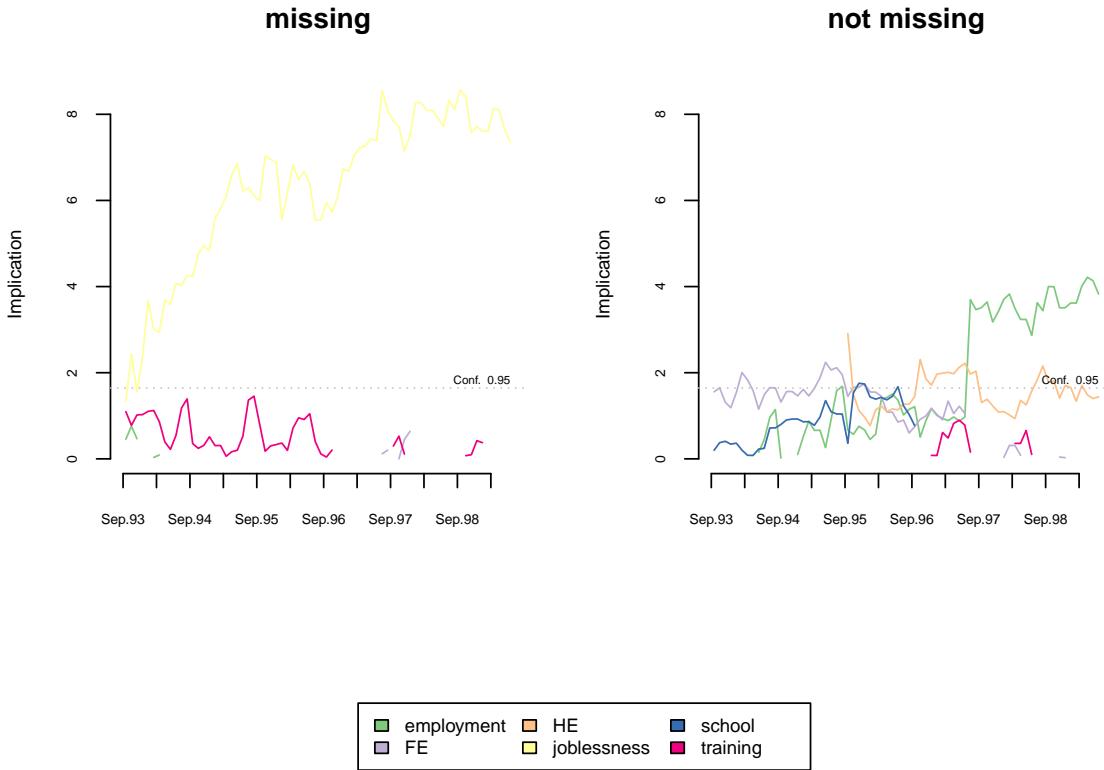
We do not observe any systematic patterns of missing data, since the 10 most frequent patterns of missing data account for only 10% of all patterns.

Frequent patterns of missing data should be considered. For example, when considering educational trajectories, it may appear that missing data occur systematically in the summer months when individuals are in transition between two situations. Therefore, it may be appropriate to treat these cases systematically before considering imputations.

The amount of missing data and the patterns are a first indicator to gauge how pervasive the issue of missing data is, but it is not enough. Along this line, the concept of mechanism of missing data is crucial.

A tool available in the package that helps to grasp the mechanism is the `seqmissimplic()` function, which is based on the `seqimplic()` function of the **TraMineRextras** package (Ritschard, Studer, Buergin, Liao, Gabadinho, Fonta, Muller, and Rousset 2024). It allows to display the states that better characterize, at each time point, sequences with missing data vs. sequences without missing data.

```
R> imp <- seqmissimplic(mvadseq.miss)
R> plot(imp, legend.prop=0.22, cex.axis=0.65, cex.legend=0.8, cex.lab=0.8)
```



In the left panel, we observe that trajectories with at least one missing value are characterized by unemployment at almost every time point. Indeed, we observe that the implication statistic for the unemployment situation, shown in yellow, is above, and thus outside, the 95% confidence interval. On the other hand, the observed trajectories are characterized (right panel) by employment in the later time points, as we see that the value of the implication statistic is outside the 95% confidence interval. This allows us to highlight that the mechanism is not MCAR and that there tends to be an overrepresentation of employment in complete trajectories and, conversely, an overrepresentation of unemployment in trajectories with missing values.

5.2. Multiple imputation

The second step is to handle missing data through multiple imputation. We describe in detail how this step can be carried out with this package and, once imputation has been carried out, we briefly examine the imputed datasets.

During the implementation of this process, several decisions need to be made regarding the specification of the imputation model. For illustrative purposes, we use a simplified imputation method with default arguments. However, this model may prove to be too simplistic for real applications. In section 6 we provide comprehensive guidelines on how to choose an appropriate imputation model. In particular, we focus on the number of previous and subsequent time points to include, the number of multiple imputations, and the covariates to include. In addition, in section 7 we discuss additional functionalities: parallel computing to speed up computations and the use of random forest imputation models instead of

multinomial models.

The imputation process introduces an element of randomness. Running the imputation process multiple times will result in different imputed datasets and consequently may lead to different clusterings. This variability is particularly pronounced when a small number of imputations are used, as in this example. In order to reproduce the same results presented here, a seed value should be set before starting the imputation process.

We set a seed value before generating complete datasets with the `seqimpute()` function. The argument `var` specifies which columns of the datasets contain the trajectories to be imputed (here 17 to 86), the argument `np` specifies how many previous observations to include in the imputation models, `nf` specifies how many subsequent observations to include, and the argument `m` specifies the number of multiple imputations.

```
R> set.seed(1)
R> imputed <- seqimpute(mvad.miss, var=17:86, np=1, nf=1, m=2)
```

5.3. Typology creation

The next step is to derive the clustering from these multiple imputed datasets. In this document, we follow the strategy of stacking all the imputed datasets, then identifying the clustering, and finally redistributing it among the imputed datasets in order to apply the regression analysis to each imputed dataset. This is not the only strategy available. Consensus clustering may be a very promising approach in this respect.

The clustering of sequences follows three steps: first, a dissimilarity is computed between each pair of sequences, then different groupings are computed with a clustering algorithm based on these dissimilarities, and finally the best one is selected based on cluster quality indices. For this illustration we follow the procedure of the **WeightedCluster** vignette (Studer 2013). The pairwise dissimilarities are computed using Hamming distance (see Studer and Ritschard (2016) for recommendations on choosing dissimilarity measures), hierarchical clustering with Ward's linkage is applied, and the best clustering is selected based on ASWw, HG, PBC and HC.

1. The `fromseqimp()` function allow to transform the `seqimp` object obtained with the `seqimpute()` function into various formats. In particular, by stating `format` to "stacked", we obtain a dataset where the imputed datasets are stacked vertically, which is the form needed to apply Halpin's clustering strategy.

```
R> stackedimp <- fromseqimp(imputed, format="stacked")
R> stackedimpseq <- seqdef(stackedimp, xtstep=6)
```

2. The dissimilarity matrix is computed using the Hamming distance.

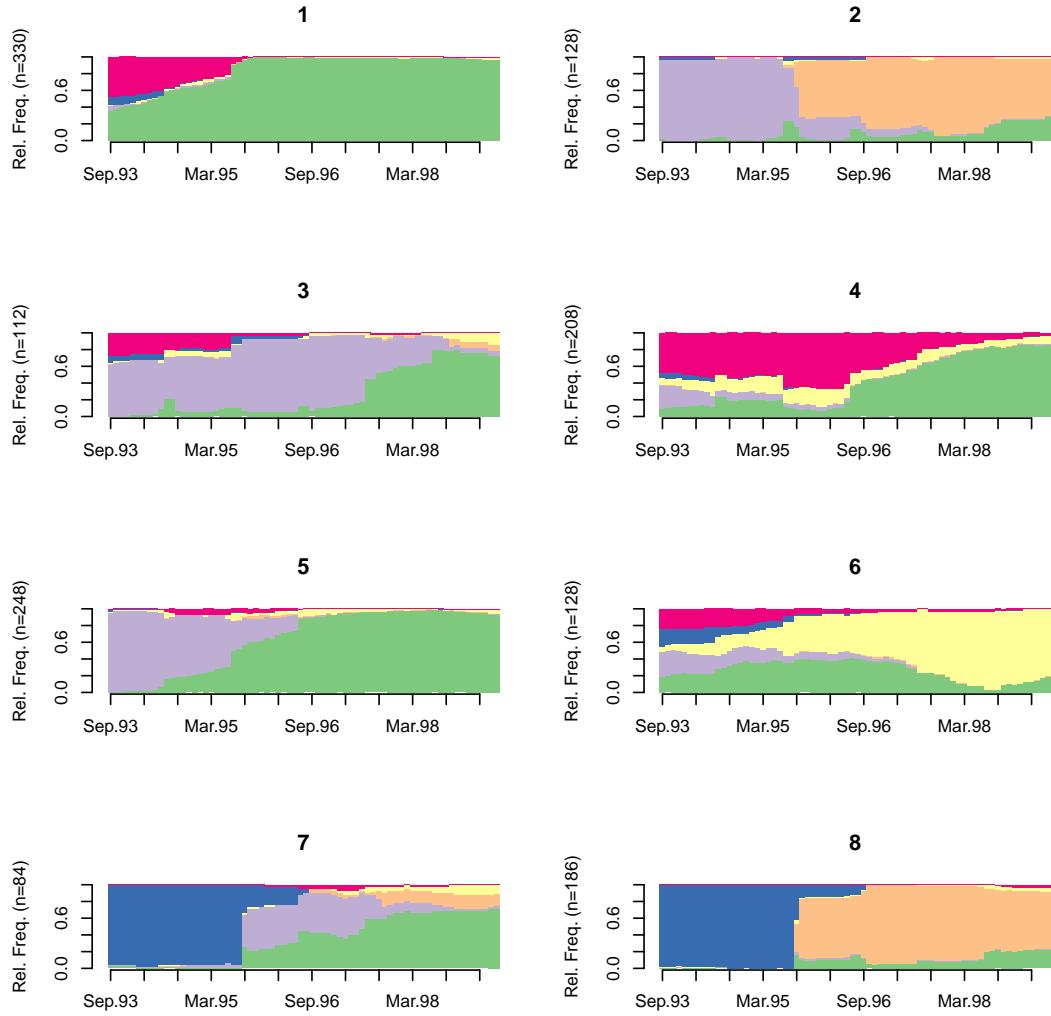
```
R> ham <- seqdist(stackedimpseq, method = "HAM")
```

3. The hierarchical clustering with Ward linkage is computed.

```
R> wardCluster <- hclust(as.dist(ham), method="ward.D")
```

4. The partition is chosen with the help of clustering quality measures, which can be done with the **WeightedCluster** library.

```
R> library("WeightedCluster")
R> wardRange <- as.clustrange(wardCluster, diss=ham, ncluster=15)
R>
R> plot(wardRange, stat=c("ASWw", "HG", "PBC", "HC"))
```



The clustering in eight groups is a local extremum for each of the clustering quality measures.

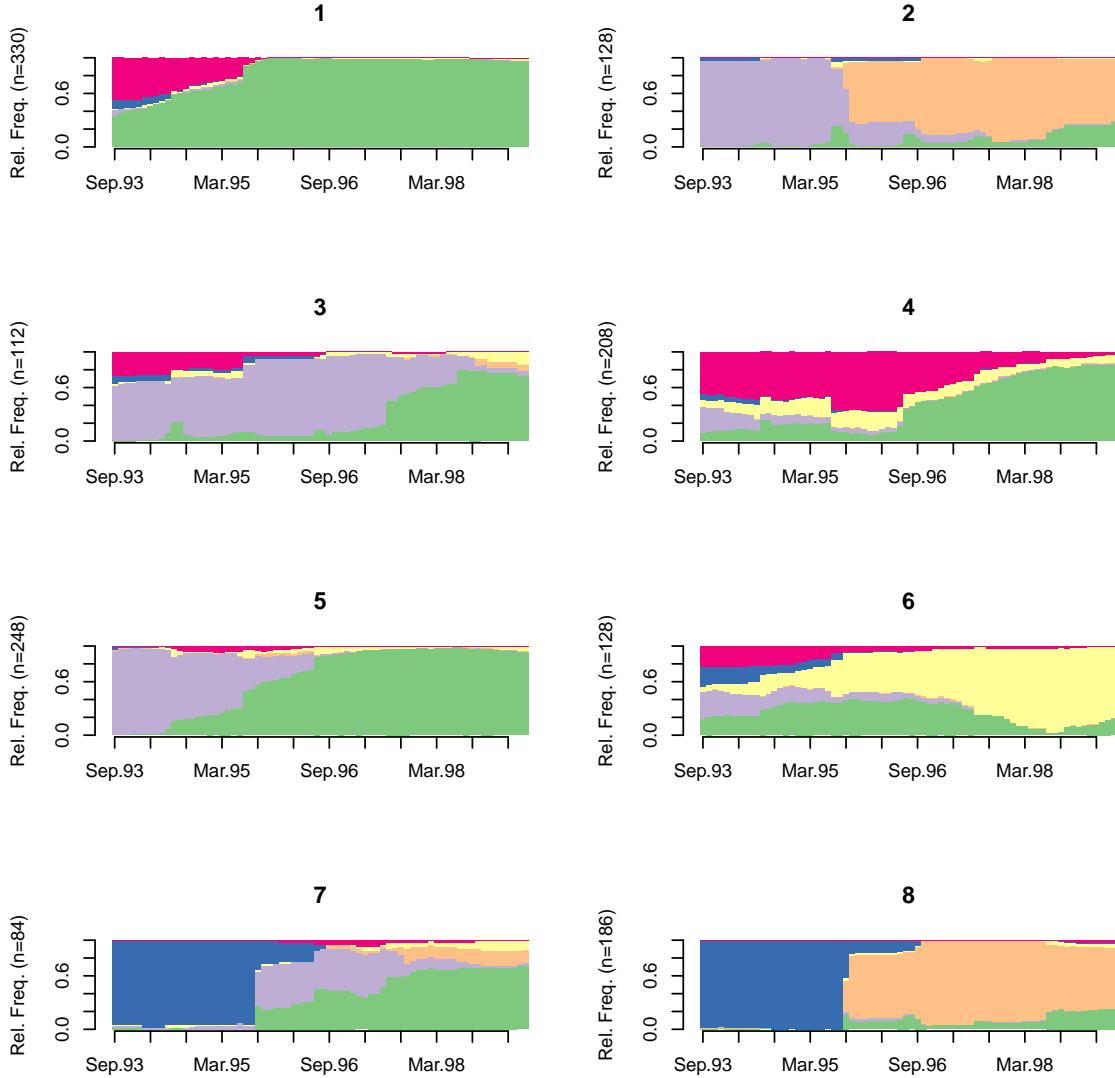
5. After performing hierarchical clustering and selecting eight clusters, we proceed to assign these clusters to the **seqimp** object with the **addcluster()** function.

The resulting cluster labels are stored as an additional column called **cluster** within the imputed datasets, setting the stage for the subsequent regression analysis.

```
R> imputed <- addcluster(imputed, clustering=wardRange$clustering$cluster8)
```

Now that the clustering has been determined, we visualize the eight groups that comprise it:

```
R> seqdplot(stackedimpseq, group = wardRange$clustering$cluster8,
+ border = NA, with.legend=FALSE)
```



Unemployment trajectories are mostly captured by the sixth group. We also observe some trajectories with unemployment in other groups, such as the fourth, but focus only on the sixth group for the sake of illustration.

5.4. Regression analysis on clustered trajectories

The final step in the analysis is to run the regression analysis separately on each completed dataset and pool the results.

The sixth cluster is the one that mostly captures trajectories with spells of unemployment. Therefore, we apply a logistic regression where the dependent variable is the membership to

this cluster and the independent variable is father's unemployment. We also use sex as a control variable.

We use the **mice** package, which provides many tools for dealing with multiple imputed datasets. We first transform the imputed datasets into an object that can be manipulated by **mice**.

1. We transform the object containing the imputed datasets with the `fromseqimp()` function.

```
R> imputed.mids <- fromseqimp(imputed, format="mids")
```

2. We apply a logistic regression where the dependent variable indicates membership in the sixth cluster, which represents trajectories involving unemployment. The independent variables are sex and whether the respondent's father was unemployment.

To fit the regression models within the multiple imputation framework, we use the `with()` function of the **mice** (Van Buuren and Groothuis-Oudshoorn 2011) package.

```
R> library("mice")
R> fit <- with(imputed.mids, glm(I(cluster == 6) ~ mvad$male + mvad$funemp,
+ family = binomial))
```

3. To pool the results of the regression models, we use the `pool()` function, also coming from the **mice** package.

```
R> summary(pool(fit))
```

	term	estimate	std.error	statistic	df
1	(Intercept)	-2.3649195	0.1968796	-12.012010	706.9327
2	mvad\$maleyes	-0.5755973	0.2722765	-2.114018	706.9327
3	mvad\$funempyes	1.2762176	0.2829957	4.509671	706.9327
	p.value				
1	2.225536e-30				
2	3.486334e-02				
3	7.599971e-06				

The interpretation of the regression model is similar as with standard regression.

Therefore, individuals who had an unemployed father are more likely to belong to the cluster that captures unemployment trajectories.

Of course, this analysis is only an illustration of the functions in this package. It is oversimplified. No firm conclusions can be drawn from this analysis.

6. Guidelines related to the imputation model

In the illustrative example, we used MICT with a simple imputation model for illustrative purposes. However, such a model proves too simplistic for most real-world applications. In this section, we examine four crucial decisions in selecting imputation models: choosing

between MICT and MICT-timing, determining the number of prior and subsequent time points to include, considering the inclusion of covariates, and deciding on the number of imputations.

1. MICT or MICT-timing

If the transitions may differ along the trajectories, MICT-timing should be used, while if this is not the case, MICT is preferred.

For example, in the context of our example application, the majority of transitions occur during the summer months. MICT does not discriminate between summer and other months, resulting in a lack of transitions in summer and an excess of transitions in other months. Conversely, MICT-timing makes this distinction, making it more appropriate for this particular case.

To apply the MICT-timing algorithm with the `seqimpute()` function, the argument `timing` must be set to `TRUE`. Otherwise, its use is similar to the MICT algorithm. One must specify the number of previous observations (`np`), the number of subsequent observations (`nf`), and the number of imputations (`m`).

```
R> imputed <- seqimpute(mvad.miss, var=17:86, np=1, nf=1, m=3,
+ timing=TRUE)
```

When the number of trajectories is very small (fewer than 200, based on our experience), it can be advantageous to include not only observations from the same time point across sequences but also those from adjacent time points. This can be achieved using the `frame.radius` argument. For instance, setting `frame.radius` to 1 incorporates observations from both the preceding and following time points into the imputation models.

```
R> imputed <- seqimpute(mvad.miss, var=17:86, np=1, nf=1, m=3,
+ timing=TRUE, frame.radius=1)
```

2. Inclusion of covariates

Incorporating carefully chosen covariates can enhance the quality of imputations. We elaborate on the specific covariates that should be included and the rationale behind their selection. Additionally, we focus on the practical implementation of these covariates within the `seqimpute()` function. Finally, we briefly discuss some open issues related to the inclusion of covariates.

Guidance regarding the incorporation of covariates in imputation models suggests including those that (see e.g. [Van Buuren \(2018\)](#)):

- Are subject to different trajectories. For example, career paths differ between men and women, with men often working full-time and women more likely to work part-time (see e.g. [Widmer and Ritschard \(2009\)](#)). Failure to include the gender variable in the imputation process would result in imputed values that do not take these notable differences into account. In the sample application of this article, transitions may vary according to the type of secondary education.

- Potentially relate to the missing data mechanism. For example, several studies have shown that less-educated individuals are more likely to have missing values (e.g. [Voorpostel 2010](#); [Müller *et al.* 2012](#)).
- Will be considered in the subsequent statistical analysis. This allows to maintain the relations between the corresponding covariates after the imputation. In our illustrative example, we included father unemployment and sex as covariates in the statistical analysis. Hence, these variables should be incorporated into the imputation models.

Covariates should be included by specifying the names of the columns in the data set that contain the covariates we want to include. We are including gender, type of secondary education, and whether the father was unemployed. These are in the columns named *male*, *Grammar* and *funemp*.

```
R> imputed <- seqimpute(mvad.miss, var=17:86, np=5, nf=5, m=3,
+ timing=TRUE, covariates=c("male", "Grammar", "funemp"))
```

The `seqimpute()` function can also be provided with time-varying covariates, supplied either as a list of sequence objects (each corresponding to a time-varying covariate) or as a list of dataframes.

Note that the `seqimpute()` function exclusively handles covariates without missing data.

3. Number of previous and subsequent predictors

In the example, we relied on only one observation in the past and one in the future for the prediction. If long-term effects are suspected, it may be worth increasing the number of past and future observations included.

```
R> imputed <- seqimpute(mvad.miss, var=17:86, np=5, nf=5,
+ m=3, timing=TRUE)
```

4. Number of imputations

With multiple imputation, we create multiple complete datasets with no missing data. Therefore, one question is how many imputed datasets are needed. By increasing the number of imputations, we reduce the variability introduced into the statistical results. However, we may be limited by the computational burden, and the gain from a very large number of imputations may be limited.

In practice, it is often recommended to use between 5 and 20, depending on the amount of missing data (see e.g. [Van Buuren \(2018\)](#) for a discussion on the number of imputations). If the interest is in detecting small effects, it may be worth increasing the number of imputations. It is important to strike a balance between the statistical benefits and the computational constraints.

7. Other features of the imputation through *seqimpute*

We delve into two other features available for `seqimpute()`, namely the application of random forests as imputation models and parallel computing.

7.1. Random forests

The package offers the flexibility to employ random forests as an alternative to multinomial imputation models. Random forests are a widely used method for handling missing data in various domains (Burgette and Reiter 2010; Shah, Bartlett, Carpenter, Nicholas, and Hemingway 2014; Doove, Van Buuren, and Dusseldorp 2014). They possess several advantageous features such as the ability to capture non-linear relationships, interactions, and immunity to irrelevant predictors (Friedman, Hastie, Tibshirani *et al.* 2001). These attributes are particularly valuable when dealing with longitudinal data, where specific state combinations can trigger long-term effects. However, despite these theoretically appealing characteristics, the suitability of random forests as imputation models still need to be demonstrated. We recommend users to apply by default multinomial imputation models unless they have specific reasons to explore alternatives.

To apply the imputation model defined in the last section with random forests imputation models, one needs to set the `regr` argument to "rf"

```
R> set.seed(2)
R> imputed.rf <- seqimpute(mvad.miss, var=17:86, np=5, nf=5, m=3,
+ covariates=c("male", "Grammar", "funemp"), regr="rf")
```

7.2. Parallel computing

Multiple imputation can be computationally intensive. To address this, the `seqimpute()` function supports parallel computing. Since each imputation is completely independent of the others, it is possible to run each of the imputations simultaneously in parallel, providing a practical solution to reduce processing time.

To enable parallel computing, the `ParExec` argument must be set to `TRUE`, and the `ncores` argument specifies the number of cores to use. If the number of cores is not specified, it is set as the maximum number of available cores minus 1. For reproducibility, a seed should be provided using the `SetRNGSeed` argument. Note that using `set.seed()` alone before computations does not ensure reproducibility; the `SetRNGSeed` argument must be explicitly set.

```
R> imputed.par <- seqimpute(mvad.miss, var=17:86, np=5, nf=5, m=3,
+ covariates=c("male", "Grammar", "funemp"), ParExec = TRUE, ncores=3,
+ SetRNGSeed = 2)
```

8. Additional Functions

This section outlines additional functionalities provided by the package. We focus on two functions: `seqaddNA()`, which simulates missing data and `seqQuickLook()`, which provides an overview of the number and sizes of different types of gaps in the original dataset.

8.1. Simulating missing data

The `seqaddNA()` function enables the simulation of missing data in sequences using a Markovian approach, where the probability of a value being missing depends on the preceding time point. The simulation process works as follows:

1. The first time point of a sequence has a `p.low` probability of being missing.
2. For subsequent time points:
 - (a) If the previous time point is missing, the current time point has a `pcont` probability of being missing.
 - (b) If the previous time point is observed, the current time point has a `p.high` probability of being missing if the previous state belongs to the set of states specified in the `states.high` argument. Otherwise, the probability of being missing is `p.low`.

The function also includes several additional arguments to customize the simulation:

- `propdata`: Specifies the proportion of trajectories on which missing data will be simulated, as a decimal between 0 and 1. By default, all sequences (`propdata = 1`) are selected for simulation.
- `maxgap`: Defines the maximum length of a gap. If this limit is reached, the next time point is automatically set as observed. The default value is 3.
- `maxprop`: Sets the maximum proportion of missing data allowed within a sequence. If this limit is exceeded, the missing data simulation is repeated for that sequence. The default value is 0.75.

8.2. Overview of the gaps of missing data

The function `seqQuickLook()` provides an overview of the number and size of the different types of gaps spread in the original dataset. Note that both MICT and MICT-timing differentiates between six different types of gaps. The definition of several of them depends on the number of previous (`np`) and future (`nf`) observations that are set for the MICT and MICT-timing algorithms.

- **Internal gap** have at least `np` observations before and `nf` after the gap
- **Initial gap** are situated at the very beginning of a trajectory
- **Terminal gap** are situated at the very end of a trajectory
- **Left-hand side specifically located gap (SLG)** have at least `nf` observations after the gap, but less than `np` observation before it
- **Right-hand side specifically located gap (SLG)** have at least `np` observations before the gap, but less than `nf` observation after it
- **Both-hand side specifically located gap (SLG)** have less than `np` observations before the gap, and less than `nf` observations after it

The function returns a data frame that summarizes the characteristics of each type of gap. For each gap type, it provides the minimum and maximum lengths, the total number of gaps, and the total number of missing values they contain.

For instance, consider the dataset with missing data generated in Section 4. Using an imputation model with `np=5` and `nf=5`, we have:

```
R> seqQuickLook(mvad.miss, var = 17:86, np = 5, nf = 5)
```

	MinGapSize	MaxGapSize	numbOfGaps	sumNAGaps
Internal Gaps	1	3	481	1028
Initial Gaps	1	3	7	16
Terminal Gaps	1	3	23	38
LEFT-hand side SLG	2	3	25	50
RIGHT-hand side SLG	1	3	44	81
BOTH-hand side SLG	0	0	0	0

Therefore, for example, we observe that internal gaps have a length between 1 and 3. There are 481 such gaps, and they account for 1028 missing values.

9. Conclusion

This document describes the functionality of the **seqimpute** package, which mainly implements two multiple imputation methods tailored to categorical sequence data: MICT and MICT-timing. In addition, several functions for displaying patterns of missing data within sequences are included. These tools are illustrated with an example focusing on a typical application in sequence analysis, namely the use of a derived typology in a regression analysis.

10. Acknowledgements

This publication benefited from the support of the Swiss National Science Foundation (project “Strengthening Sequence Analysis,” grant number: 10001A_204740). The authors are grateful to the Swiss National Science Foundation for its financial assistance.

References

Audigier V, Husson F, Josse J (2017). “MIMCA: multiple imputation for categorical variables with multiple correspondence analysis.” *Statistics and computing*, **27**, 501–518.

Basagaña X, Barrera-Gómez J, Benet M, Antó JM, Garcia-Aymerich J (2013). “A framework for multiple imputation in cluster analysis.” *American journal of epidemiology*, **177**(7), 718–725.

Burgette LF, Reiter JP (2010). “Multiple imputation for missing data via sequential regression trees.” *American journal of epidemiology*, **172**(9), 1070–1076.

Doove LL, Van Buuren S, Dusseldorp E (2014). “Recursive partitioning for missing data imputation in the presence of interaction effects.” *Computational statistics & data analysis*, **72**, 92–104.

Emery K, Studer M, Berchtold A (2024). “Comparison of imputation methods for univariate categorical longitudinal data.” *Quality & Quantity*.

Faucheu L, Resche-Rigon M, Curis E, Soumelis V, Chevret S (2021). “Clustering with missing and left-censored data: A simulation study comparing multiple-imputation-based procedures.” *Biometrical Journal*, **63**(2), 372–393.

Friedman J, Hastie T, Tibshirani R, *et al.* (2001). *The elements of statistical learning*. Springer.

Gabadinho A, Ritschard G (2016). “Analyzing state sequences with probabilistic suffix trees: the PST R package.” *Journal of statistical software*, **72**(1), 1–39.

Gabadinho A, Ritschard G, Müller NS, Studer M (2011). “Analyzing and visualizing state sequences in R with TraMineR.” *Journal of Statistical Software*, **40**(4), 1–37.

Gomer B, Yuan KH (2023). “A realistic evaluation of methods for handling missing data when there is a mixture of MCAR, MAR, and MNAR mechanisms in the same dataset.” *Multivariate Behavioral Research*, **58**(5), 988–1013.

Halpin B (2012). “Multiple imputation for life-course sequence data.” *Department of Sociology Working Paper Series*, University of Limerick.

Harrell Jr FE (2024). *Hmisc: Harrell Miscellaneous*. R package version 5.1-3, URL <https://CRAN.R-project.org/package=Hmisc>.

Honaker J, King G, Blackwell M (2011). “Amelia II: a program for missing data.” *Journal of statistical software*, **45**(1), 1–47.

King G, Honaker J, Joseph A, Scheve K (2001). “Analyzing incomplete political science data: An alternative algorithm for multiple imputation.” *American political science review*, **95**(1), 49–69.

Kowarik A, Templ M (2016). “Imputation with the R Package VIM.” *Journal of Statistical Software*, **74**(7), 1–16. doi:[10.18637/jss.v074.i07](https://doi.org/10.18637/jss.v074.i07).

Lall R (2016). “How multiple imputation makes a difference.” *Political Analysis*, **24**(4), 414–433.

Liao TF, Bolano D, Brzinsky-Fay C, Cornwell B, Fasang AE, Helske S, Piccarreta R, Raab M, Ritschard G, Struffolino E, *et al.* (2022). “Sequence analysis: its past, present, and future.” *Social science research*, **107**, 102772.

Little RJ (1992). “Regression with missing X’s: a review.” *Journal of the American statistical association*, **87**(420), 1227–1237.

Little RJ, Rubin DB (2019). *Statistical analysis with missing data*, volume 793. John Wiley & Sons.

Little RJ, Schenker N (1995). “Missing data.” In G Arminger, CC Clogg, ME Sobel (eds.), *Handbook of statistical modeling for the social and behavioral sciences*, pp. 39–75. Springer.

McVicar D, Anyadike-Danes M (2002). “Predicting successful and unsuccessful transitions from school to work by using sequence methods.” *Journal of the Royal Statistical Society Series A: Statistics in Society*, **165**(2), 317–334.

Molenberghs G, Fitzmaurice G, Kenward MG, Tsiatis A, Verbeke G (2014). *Handbook of missing data methodology*. Chapman & Hall/CRC handbooks of modern statistical methods. CRC Press.

Müller NS, Sapin M, Gauthier JA, Orita A, Widmer ED (2012). “Pluralized life courses? An exploration of the life trajectories of individuals with psychiatric disorders.” *International Journal of Social Psychiatry*, **58**(3), 266–277.

Perkins NJ, Cole SR, Harel O, Tchetgen Tchetgen EJ, Sun B, Mitchell EM, Schisterman EF (2018). “Principled approaches to missing data in epidemiologic studies.” *American journal of epidemiology*, **187**(3), 568–575.

Piccarreta R, Studer M (2019). “Holistic analysis of the life course: Methodological challenges and new perspectives.” *Advances in Life Course Research*, **41**, 100251.

Quartagno M, Carpenter J (2023). *jomo: A package for Multilevel Joint Modelling Multiple Imputation*. URL <https://CRAN.R-project.org/package=jomo>.

Ritschard G, Studer M, Buergin R, Liao TF, Gabadinho A, Fonta PA, Muller NS, Rousset P (2024). “Package ‘TraMineRextras’.”

Rothenbühler M, Voorpostel M (2016). “Attrition in the Swiss Household Panel: Are Vulnerable Groups more Affected than Others?” In M Oris, C Roberts, D Joye, M Ernst Stähli (eds.), *Surveying Human Vulnerabilities across the Life Course*, pp. 223–244. Springer. ISBN 978-3-319-24157-9. doi:10.1007/978-3-319-24157-9_10. URL https://doi.org/10.1007/978-3-319-24157-9_10.

Rubin DB (1976). “Inference and Missing Data.” *Biometrika*, **63**(3), 581–592. doi:10.1093/biomet/63.3.581.

Rubin DB (1987). *Multiple Imputation for Nonresponse in Surveys*. Wiley Series in Probability and Statistics. John Wiley & Sons.

Shah AD, Bartlett JW, Carpenter J, Nicholas O, Hemingway H (2014). “Comparison of random forest and parametric imputation models for imputing missing data using MICE: a CALIBER study.” *American journal of epidemiology*, **179**(6), 764–774.

Studer M (2013). “WeightedCluster Library Manual: A practical guide to creating typologies of trajectories in the social sciences with R.” *LIVES Working papers*, **24**. ISSN 2296-1658. [doi:10.12682/lives.2296-1658.2013.24](https://doi.org/10.12682/lives.2296-1658.2013.24). Swiss National Centre of Competence in Research LIVES, Geneva, URL <https://doi.org/10.12682/lives.2296-1658.2013.24>.

Studer M, Ritschard G (2016). “What matters in differences between life trajectories: A comparative review of sequence dissimilarity measures.” *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, **179**(2), 481–511. [doi:10.1111/rssa.12125](https://doi.org/10.1111/rssa.12125).

Van Buuren S (2018). *Flexible Imputation of Missing Data*. Chapman & Hall/CRC Interdisciplinary Statistics. CRC Press. ISBN 9781138588318. URL <https://books.google.ch/books?id=bLmItgEACAAJ>.

Van Buuren S, Groothuis-Oudshoorn K (2011). “mice: Multivariate imputation by chained equations in R.” *Journal of statistical software*, **45**, 1–67.

Voorpostel M (2010). “Attrition patterns in the Swiss household panel by demographic characteristics and social involvement.” *Swiss Journal of Sociology*, **36**(2), 359–377.

Widmer ED, Ritschard G (2009). “The de-standardization of the life course: are men and women equal?” *Advances in Life Course Research*, **14**(1-2), 28–39.

Affiliation:

Kevin Emery

Swiss Centre of Expertise in Life Course Research LIVES
and

Faculty of Social Sciences

University of Geneva
Geneva, Switzerland

E-mail: kevin.emery@unige.ch