Treating missing data through multiple imputation with the seqimpute package

Kevin Emery, André Berchtold, Anthony Guinchard and Kamyar Taher

March 18, 2024

1 Introduction

This document focuses on addressing missing data in categorical sequence data, specifically through the application of multiple imputation algorithms integrated into the `seqimpute` package. In particular, it details the process of conducting a statistical analysis when missing data are managed using these algorithms, and it provides usage guidelines.

Missing data represent a prevalent issue, the treatment of which can significantly impact statistical analysis outcomes and the validity of research findings. For instance, Lall (2016) conducted a reanalysis of quantitative studies published in two prominent political science journals over five years, where missing value deletion was initially employed. Upon reanalysis using multiple imputation for missing data, significant statistical results were found to be affected in almost half of the studies, occasionally leading to contradictory conclusions

Although simply dropping cases with missing values may seem straightforward, it is often an inadequate approach that can introduce bias and potentially lead to erroneous conclusions. Perkins et al. (2018) demonstrated this by showing that deleting cases with missing data could erroneously suggest a protective effect of smoking on the risk of spontaneous abortion. Furthermore, this method is particularly inefficient in the context of categorical sequence data, as even a single missing value can result in the deletion of an entire trajectory's worth of information.

Multiple imputation is a versatile approach to dealing with missing values. It involves replacing missing values with plausible estimates, resulting in data sets with no missing values. Crucially, multiple imputation does not perform this replacement only once, but repeats the process several times. This approach recognizes the inherent uncertainty associated with missing data.

The `seqimpute` package implements two multiple imputation methods specifically designed to handle missing data in categorical sequence data: the MICT algorithm developed by Halpin (2012) and the MICT-timing algorithm developed by Emery (2023).

This document provides a practical demonstration of the use of these algorithms through a concrete case study that explores the relationship between various covariates and an established typology. It also provides clear usage guidelines for effectively handling missing data using the tools available in the package.

The example presented in this document serves as an illustration. The imputation methods presented in this document are versatile and applicable to most scenarios involving missing data in categorical sequence data.

The remainder of this document takes the following form.

- Section 2 explains why missing data is an important problem and highlights the versatility of multiple imputation as a solution.
- Section 3 provides insight into the two algorithms implemented in the `seqimpute` package: MICT and MICT-timing.
- Section 4 describes the sample used for illustration.
- Section 5 describes the four steps involved in using clustering in a regression analysis.
    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 a guide to the choices tailored to the imputation algorithms implemented in this package.
- Section 7 describes additional features of the package, namely parallel computing and random forest imputation models.
- A short conclusion concludes the document in section 8.

2 Missing data issues and treatment

The goal of this section is to provide a broad overview of missing data and multiple imputation without going into too much statistical detail. For more details on missing data, see the book by Molenberghs et al. (2014), and on multiple imputation in particular, Van Buuren (2018).

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

1. The problems posed by missing data and the general inappropriateness of complete case analysis

2. The mechanics of multiple imputation
3. The use of clustering with multiple imputation
4. The scenarios in which multiple imputation is applicable

We demonstrate these concepts using a synthetic example consisting of categorical sequence data spanning ten time points, with each measurement coded as either "A" or "B".

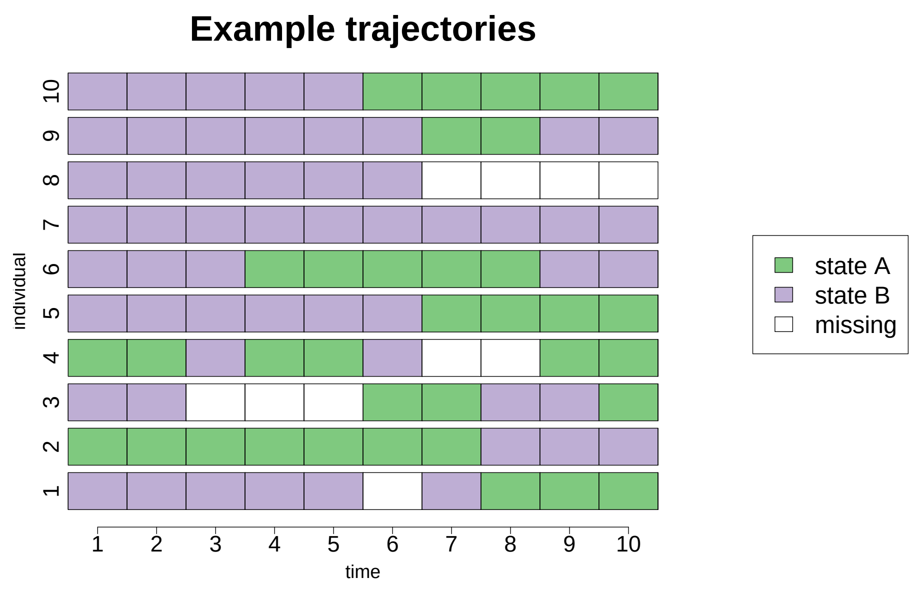


Figure 1: Dataset composed of 10 trajectories used for illustration.

This dataset is missing some values. The challenge is to determine how to deal with these missing observations. Many statistical methods, such as regression or cluster analysis, are designed primarily for complete datasets, which poses a challenge when applied to datasets with missing information.

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 introduces two main problems (Little and Schenker, 1995). First, it reduces the amount of information available. In the example, only 6 trajectories remain in the data set (Figure 2). Even if a trajectory has only one missing value, such as the first trajectory, it is completely removed from the analysis. This can reduce statistical power and hinder the detection of effects.

The second issue stemming from missing data relates to the reasons for non-response, which are often specific to certain individuals or circumstances. For example, individuals in vulnerable

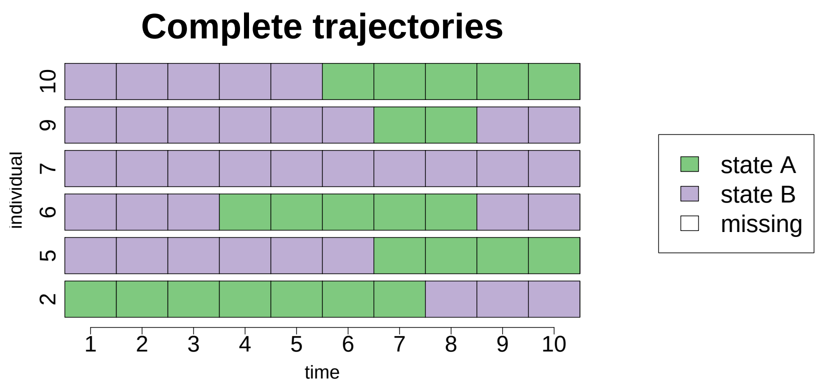


Figure 2: Remaining trajectories after a complete case analysis.

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 underrepresented in the subsample, leading to potential biases in various statistical measures related to vulnerability. In the illustrative example, trajectories with higher transition rates are more susceptible to missing values, resulting in an underrepresentation of volatile trajectories, which are often characteristic of unstable situations.

Imputation

Imputation is based on the idea of replacing missing values with some reasonable values. This is appealing because it would result in a data set 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, where missing values are imputed multiple times, resulting in multiple completed datasets.

We outline the general framework of multiple imputation, followed by an exploration of the challenges associated with using cluster analysis alongside multiple imputation.

The basic concept of multiple imputation is illustrated in Figure 3. First, missing values are replaced with meaningful values multiple times to create multiple complete datasets. Next, statistical analyses are typically performed independently on each imputed dataset, yielding estimated parameters and their variance. Finally, these estimated parameters are combined into a single estimate and its variance.

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.

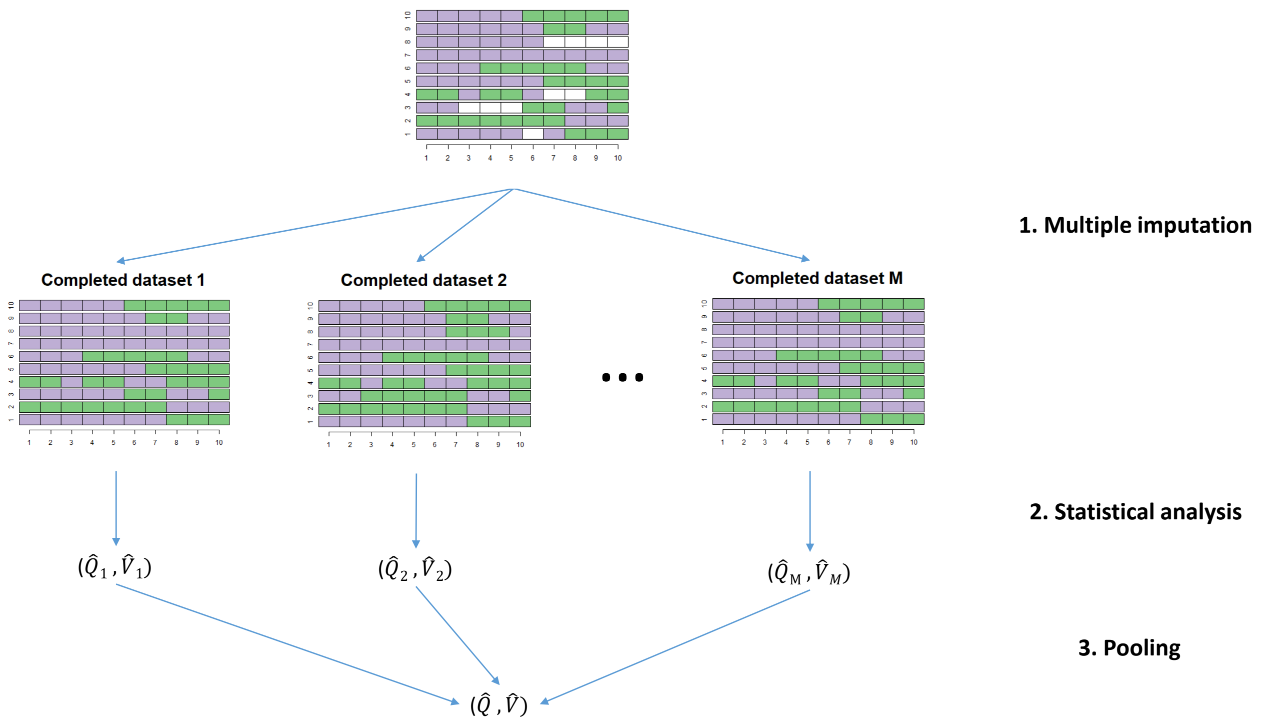


Figure 3: Illustration of three steps of a multiple imputation. The missing data are imputed several times, yielding M completed datasets. The estimated parameter and its variance is computed. These estimated parameters are then pooled in a single estimate and its variance.

Here is a brief overview of each of these three steps.

1. **Creation of completed datasets:** There are several methods for creating multiple completed datasets, with Fully Conditional Specification (FCS) being one of the most commonly used methods, implemented in the R package `mice` (Van Buuren and Groothuis-Oudshoorn, 2011). Although FCS is applicable to longitudinal data, it was not specifically designed for longitudinal data. On the other hand, the MICT (Halpin, 2012, 2013, 2016) and MICT-timing algorithms were specifically designed to impute longitudinal categorical data-a common format for life-course data-and exploit the unique characteristics of missing data in this context, which often manifest as gaps, i.e., consecutive time points that are missing.

   The description of MICT and MICT-timing is the subject of the next section.

2. **Statistical analysis:**The desired statistical analysis is usually performed independently on each completed dataset, and the estimated parameters are calculated.

3. **Combination of results:** In the final step, the estimated parameters computed on each completed dataset are combined into a single estimate and its variance is derived. Rubin

(1987) formulated rules for this: the value of the estimate is calculated as the mean across imputations, while the variance takes into account both within-imputation and between-imputation variance.

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 based on 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 address these challenges in clustering with multiple imputation. 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. This method is demonstrated in the illustrative example provided in this document.

Alternatively, consensus clustering methods aim to reconcile differences in clustering across completed datasets. For example, Basagaña et al. (2013) suggests determining the number of clusters based on the most frequently observed number across imputed datasets, and Faucheux et al. (2021) suggests identifying common clustering patterns across the multiple clusterings generated to build a single clustering.

Scenarios where multiple imputation is applicable

The appropriateness of a method for handling missing data depends on the mechanism governing the missing data. However, because the definition of the mechanisms relies on unobservable data, it is generally impossible to determine the mechanism with certainty.

Rubin (1976) distinguishes three mechanisms of missing data: Missing Completely at Random (MCAR), when the missing data happens randomly, Missing At Random (MAR), when there are specific reasons findable in the data why data is missing, and Missing Not at Random (MNAR), when the missing data depends on the value that is missing itself. For example, this could be

- *MCAR* if individuals simply forget to report their situation in some periods for no reason.
- *MAR* 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., each time someone experiences unemployment, a missing value is more likely the next month).
- *MNAR* if periods of unemployment are completely omitted.

Specifically, multiple imputation yields unbiased results when dealing with missing data completely at random (MCAR) or missing at random (MAR) scenarios (Little, 1992). The missing not at random (MNAR) case is more challenging because most methods for handling missing data, including multiple imputation, introduce bias in most cases. Real-world cases are likely to involve patterns of missing data caused by a mixture of mechanisms. Recent research by Gomer and Yuan (2023) suggests that multiple imputation may be an appropriate choice in such cases.

Thus, multiple imputation is particularly appropriate when the missing data are MCAR or MAR. A common strategy (see e.g. Van Buuren (2018)) when a MNAR mechanism is suspected is to consider adding more explanatory variables in the imputation models to explain what can be explained by other variables. Another strategy, called "sensitivity analysis", involves testing how results might be affected under MNAR scenarios.

We now provide a detailed look at the MICT and MICT-timing algorithms, which are the two methods implemented in this package.

3 Description of the imputation algorithms

As emphasized in the previous section, the first step in a multiple imputation process is the imputation itself. There are several methods for doing this. The `seqimpute` package implements two multiple imputation methods specifically designed to handle missing values in categorical sequence data, where missing values often occur as gaps rather than isolated points.

We provide a concise overview of the operating principles of the algorithms, focusing on their essence without delving into complicated details. Readers seeking a comprehensive understanding of the MICT algorithm are encouraged to refer to Halpin (2012), and for MICT timing to section 4.2 of Emery (2023).

The MICT algorithm introduced by Halpin (2012) addresses these gaps by recursively imputing missing values from their edges. However, a limitation of the MICT algorithm lies in its assumption of homogeneity of transitions throughout the trajectory. In many real-world scenarios, this assumption does not hold. For example, in life course analysis, transition rates between education and work can vary significantly over time. Individuals typically remain in education during childhood and transition predominantly from education to work between the ages of 16 and 30. In such cases, the MICT algorithm may introduce excessive transitions early on, potentially leading to erroneous imputation of transitions to work during childhood. To overcome this limitation, the MICT timing algorithm was developed specifically to account for temporal variation in transitions.

We now provide a detailed look at the MICT and MICT-timing algorithms, which are the two methods implemented in this package.

We start by describing the MICT algorithm, which serves as the foundation for MICT-timing. Following that, we outline the distinctions between MICT-timing and MICT.

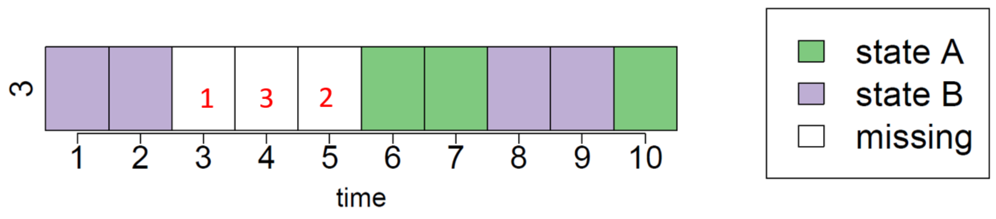
MICT

We outline the MICT algorithm by elucidating its essential operational components. First, we examine the order in which missing values are addressed within a missing data gap. We then examine the trajectory information used to determine the imputed values. Next, we examine the practical methodology for determining the imputed value. Finally, we consider the handling of starting and ending gaps of missing data, which pose unique considerations due to their singular edge structure, before outlining the process of constructing multiple completed datasets.

1. **Order of imputation**

   In the MICT algorithm, missing data gaps are recursively imputed from the edges. This approach serves two purposes. First, missing data near the edges of the gap are generally easier to impute because they are adjacent to observations. Second, by imputing from the edges inward, we maintain longitudinal consistency between the imputed values.

   For example, if we focus on the third sequences of the example:

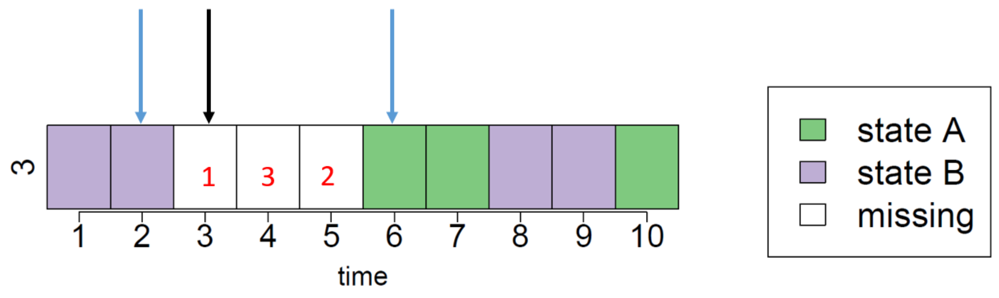


   The first missing value of the gap is imputed first, then the last value of the gaps, and finally the value in the middle.

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

   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 considered when determining 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):



This information is used to determine an imputed value. However, this is only a basic level of information incorporation. In real-world scenarios, trajectories may vary based on additional covariates that should also be included to improve the accuracy of the imputation. In section 6, we provide guidelines on the appropriate information to include in the imputation model.

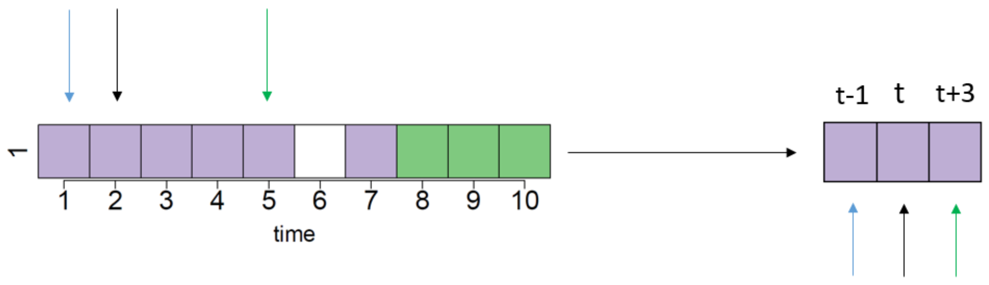
3. **Determine an imputed value**

   We know that we have a state B just before and a state A three times after. The general idea is to look in all other sequences for each time such a configuration is observed to determine an imputed value.

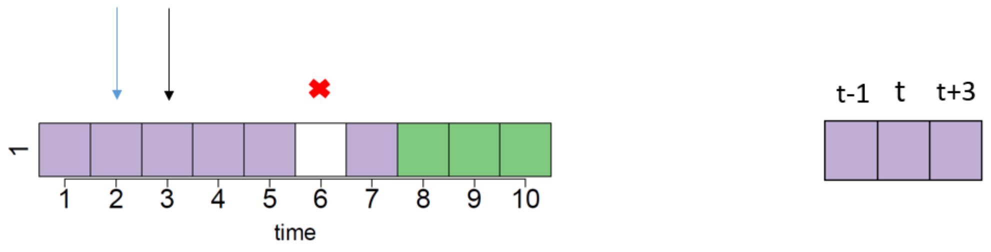
   Specifically, we look for every configuration where an observed state has the one before and the one three times after also observed, regardless of their values. We then fit a logistic model, or a multinomial model in the case of more than two categories, to these data. This model allows us to determine the probabilities of belonging to state "A" or "B". Finally, based on these probabilities, we assign a value to the missing observation. Let's examine these steps in detail.

   (a) *Extraction of the similar patterns* Let's identify the patterns in the first sequence that resemble the one we want to impute. We look for each observed time point where the previous time point and the one three time points later are observed.

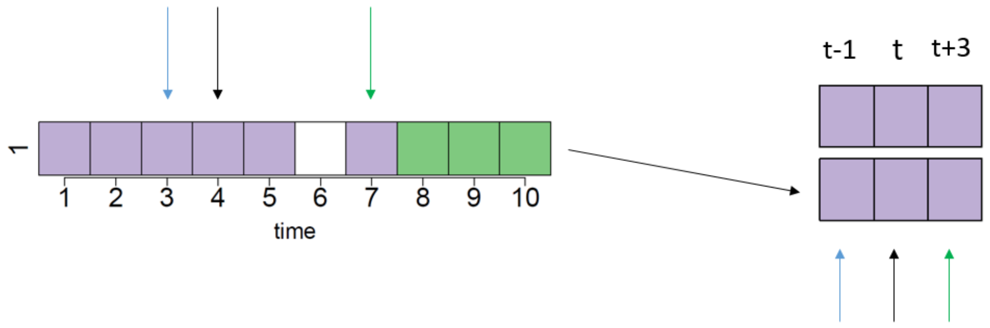
   The first similar pattern in the first sequence is the second time point:



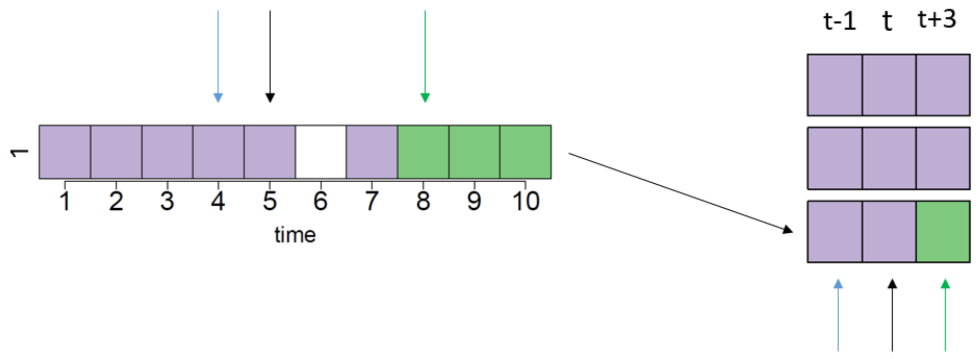
The time 3 has a missing value three timepoints after and is hence not considered:



The fourth timepoint also resemble the one we aim to impute



and finally the fiveth one:



Time points six and seven are not considered due to a missing value at time six, and 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.

Similar patterns are extracted from all sequences in the data. In total, there are 44 such patterns across the 10 sequences.

(b) *Fitting the logistic model* A logistic model is fitted to the set of similar patterns extracted in point (a). The dependent variable is *t* and the two independent variables are *t-1* and *t+3*.

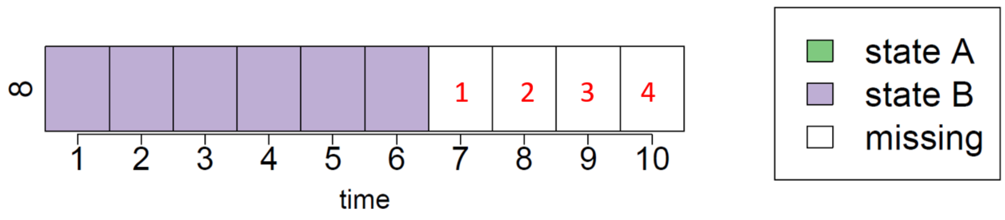
(c) *Probability of belonging to each state* Given that the previous state for the missing value to be imputed was "A" and that three time points later there is a state "B", the probabilities of being in state "A" or "B" are determined.

(d) *Imputed value* A value is drawn at random based on the derived probabilities.

4. **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 eight sequence is



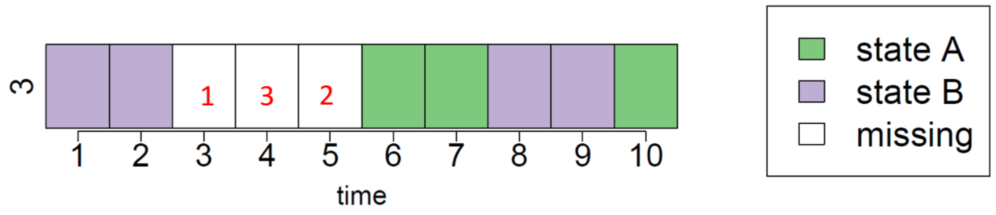
5. **How are multiple imputed data sets created?** Imputed values are drawn based on probabilities computed from logistic models, which introduces a random element into the imputation process. As a result, the imputation process is repeated several times to generate multiple imputed records. Determining the appropriate number of imputed records is discussed in the 6 section.

MICT-timing

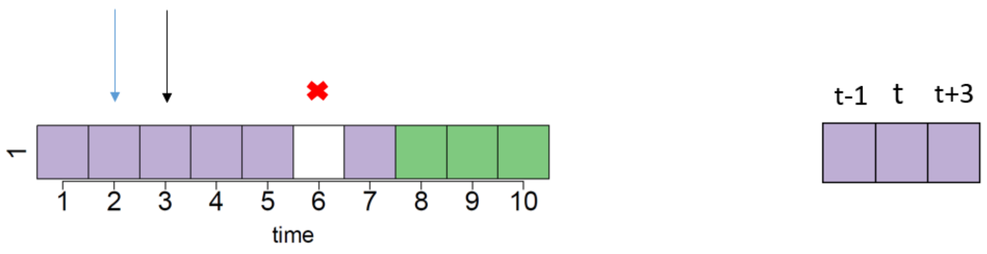
The MICT-timing algorithm is an extension of the MICT algorithm designed to address a key limitation of the latter: its assumption that position in the trajectory is irrelevant.

In practice, the primary difference between the MICT and MICT-timing algorithms lies in their approach to selecting patterns from other sequences for fitting the multinomial model. While the MICT algorithm considers all similar patterns regardless of their temporal placement, MICT-timing restricts pattern selection to those that are temporally closest to the missing value. This refinement ensures that the imputation process adequately accounts for temporal dynamics, resulting in more accurate imputed values.

As an illustrative example, let's examine the imputation of the first missing value in the third sequence. This missing value occurs at the third time point in the sequence.



To account for temporal heterogeneity, only patterns observed at time 3 on other trajectories are considered. In the case of the MICT algorithm, the first sequence contributes three patterns, whereas it would not contribute any pattern for MICT timing. In fact, the third time point, where the missing to impute occurs in sequence 3, has the time point located three times after the missing:



4 Sample application

To illustrate the application of the previously described imputation methods in statistical analysis, we use a dataset detailing 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 (Gabadinho et al., 2011), originally contains no missing data. However, to demonstrate the imputation techniques, we deliberately introduce missing data.

In our illustrative analysis, which is explained in section 5, we cluster the dataset and use these clusters as the dependent variable in a regression analysis, with the goal of exploring the social reproduction of unemployment.

We begin by retrieving the dataset from the `TraMineR` package and creating a state sequence object. Many of the visualization tools in the `seqimpute` package are based on the visualization tools in `TraMineR`, which use state sequence objects. We then introduce missing data using a function provided in the `seqimpute` package. In addition, we briefly highlight the limitations associated with relying on complete case analysis.

1. Preparation of the dataset

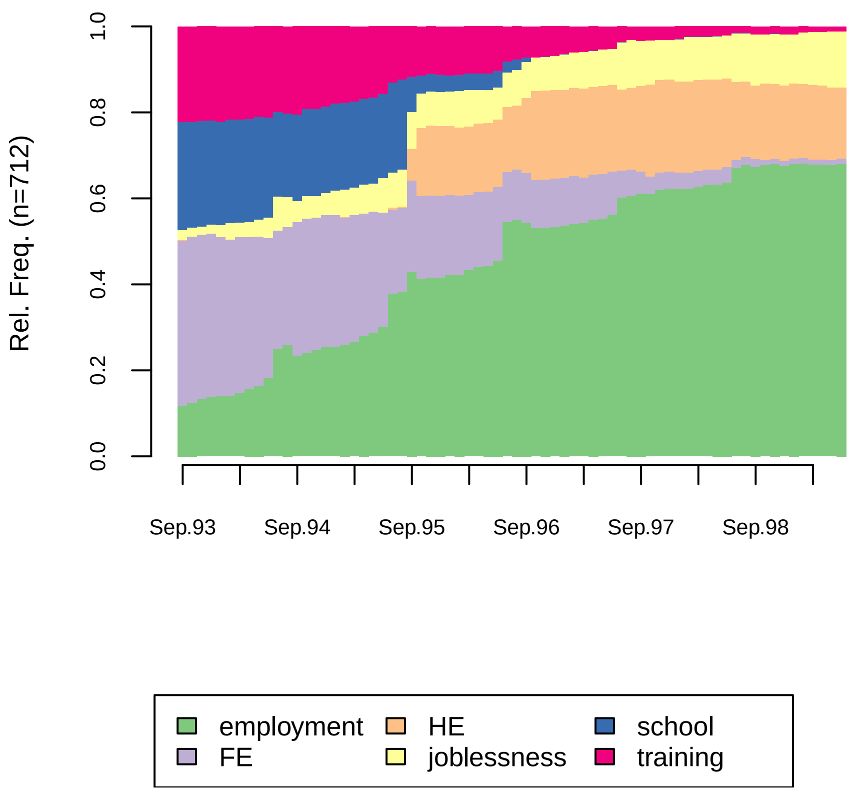
We load the `TraMineR` package and then fetch the data set named “mvad”. Using the `seqdef()` function from the `TraMineR` package, we create a state sequence object representing school-to-work trajectories. The monthly states are in columns 17 through 86 of the mvad dataset:

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

2. Visualization

We use a state distribution plot to visualize the cross-sectional state frequencies at each time point:

```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 significant proportion have moved into employment. Interestingly, about 20% 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. Studies have consistently shown a relationship between missing data and vulnerable situations such as unemployment.

To achieve this, we load the `seqimpute` package and use the `seqaddNA()` function to simulate missing values:

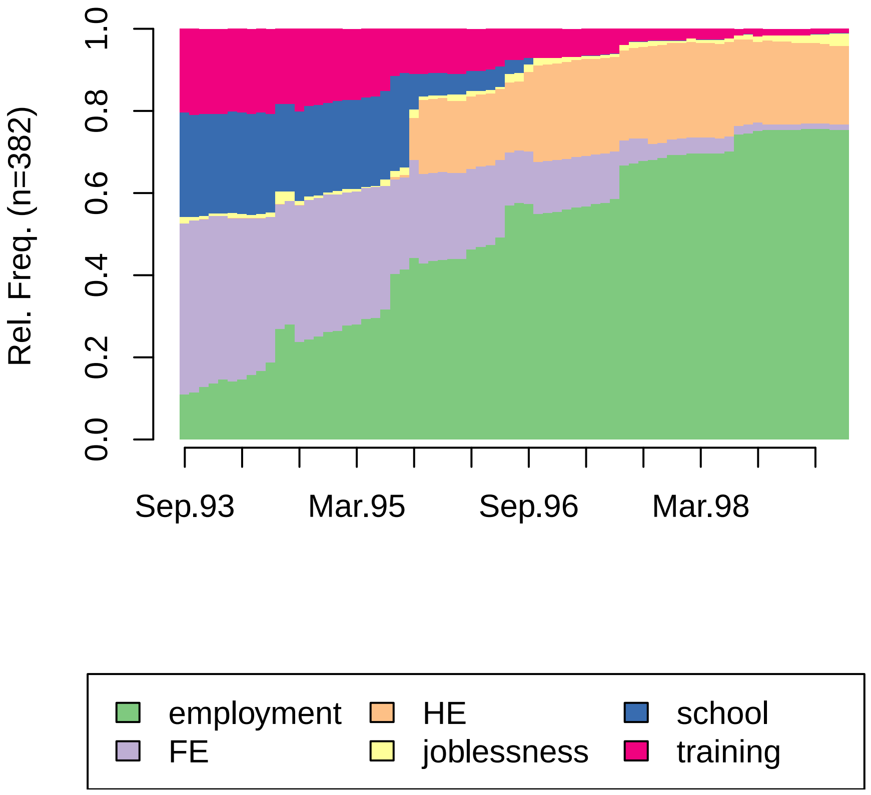
```r
library("seqimpute")
set.seed(2)
mvad.miss <- seqaddNA(mvad, var=17:86, states.high="joblessness")
```

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 limitation 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)
mvadseq.cca <- seqcomplete(mvadseq.miss)
seqdplot(mvadseq.cca, border=NA)
```



Unemployment is significantly underrepresented in the trajectories with no missing data. For example, we observe about 2% of jobless situations at the last time point, while it is almost 20% 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.

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 want to create multiple complete records. To do this, we use the MICT algorithm as the imputation method.

3. **Typology creation**

   The goal here is to 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.

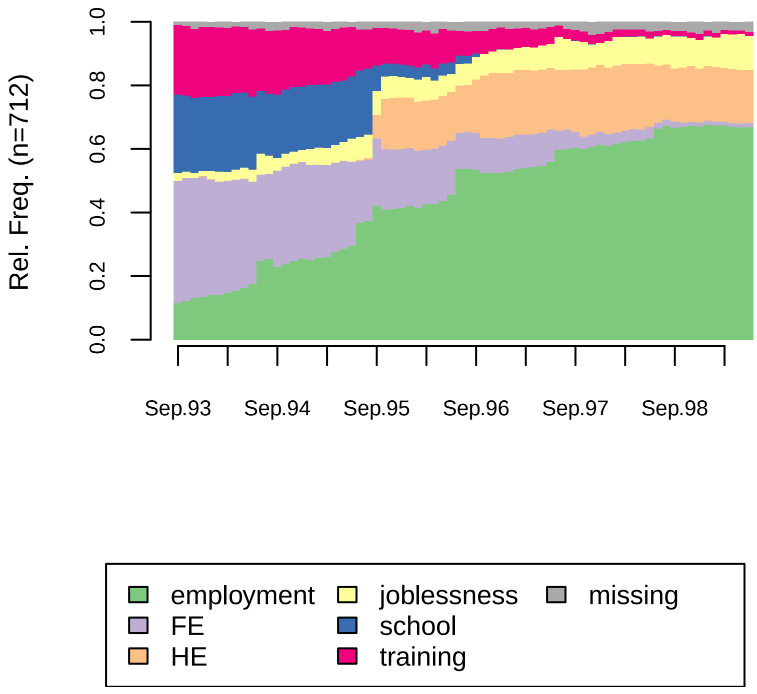
We provide detailed explanations of how to carry out each of these steps.

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 used. This visualization provides the distribution of states at each time point.

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

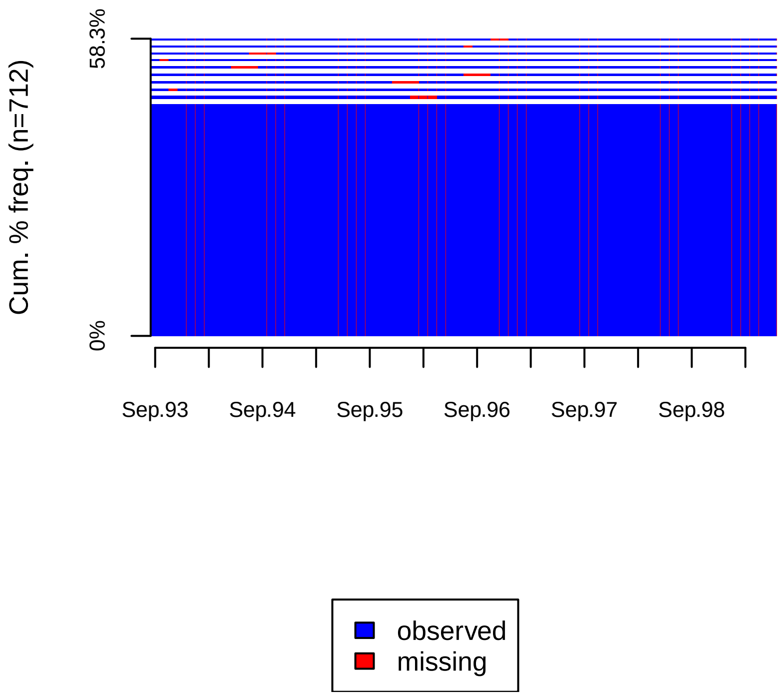


We observe that missing data increases slightly over time, but the proportion of missing values remains below 5% at each time point.

Some visualization tools are provided in this package to better examine the patterns of missing data. These plotting functions are based on the `seqplot()` function from the `TraMineR` package. Therefore, most of the arguments of this function can be applied to the plots available in the package.

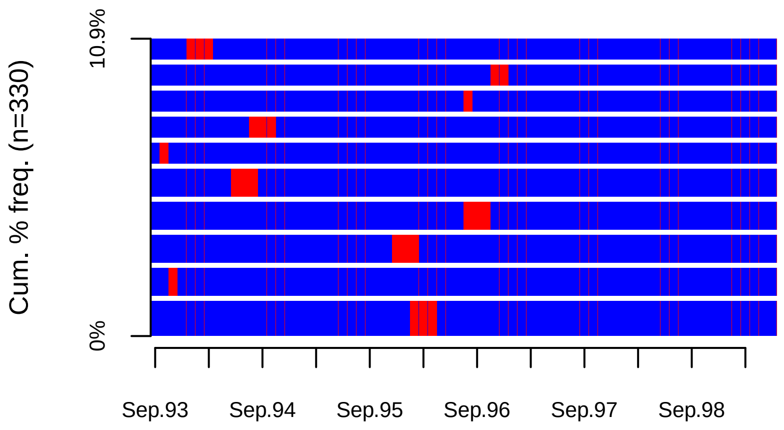
The `seqmissfplot()` shows the most frequent patterns of missing data within a sequence:

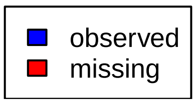
```
seqmissfplot(mvadseq.miss, border=NA)
```



We observe that the most common pattern is to have no missing value. This pattern hinders the visibility of the plot. Therefore, we can have a clearer view by displaying only trajectories with at least one missing value.

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



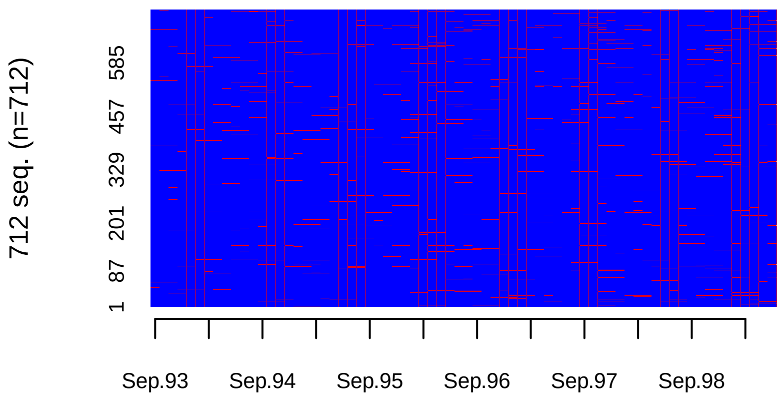


We do not observe any systematic patterns of missing data, since the 10 most frequent patterns of missing data account for only 10% of all patterns.

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.

To display all patterns of missing data among the trajectories, one can use the `seqmissIplot()` function. We show all trajectories, but as for `seqmissfplot()`, one could show only the trajectories with at least one missing value.

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

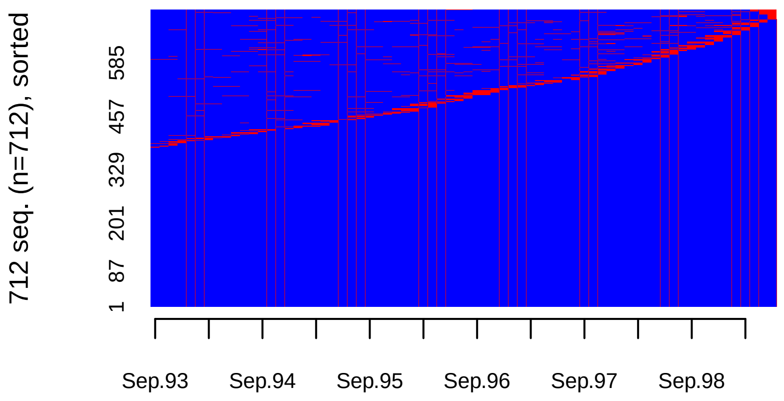


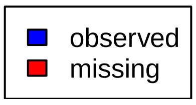
🟦 observed
🟥 missing

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.

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



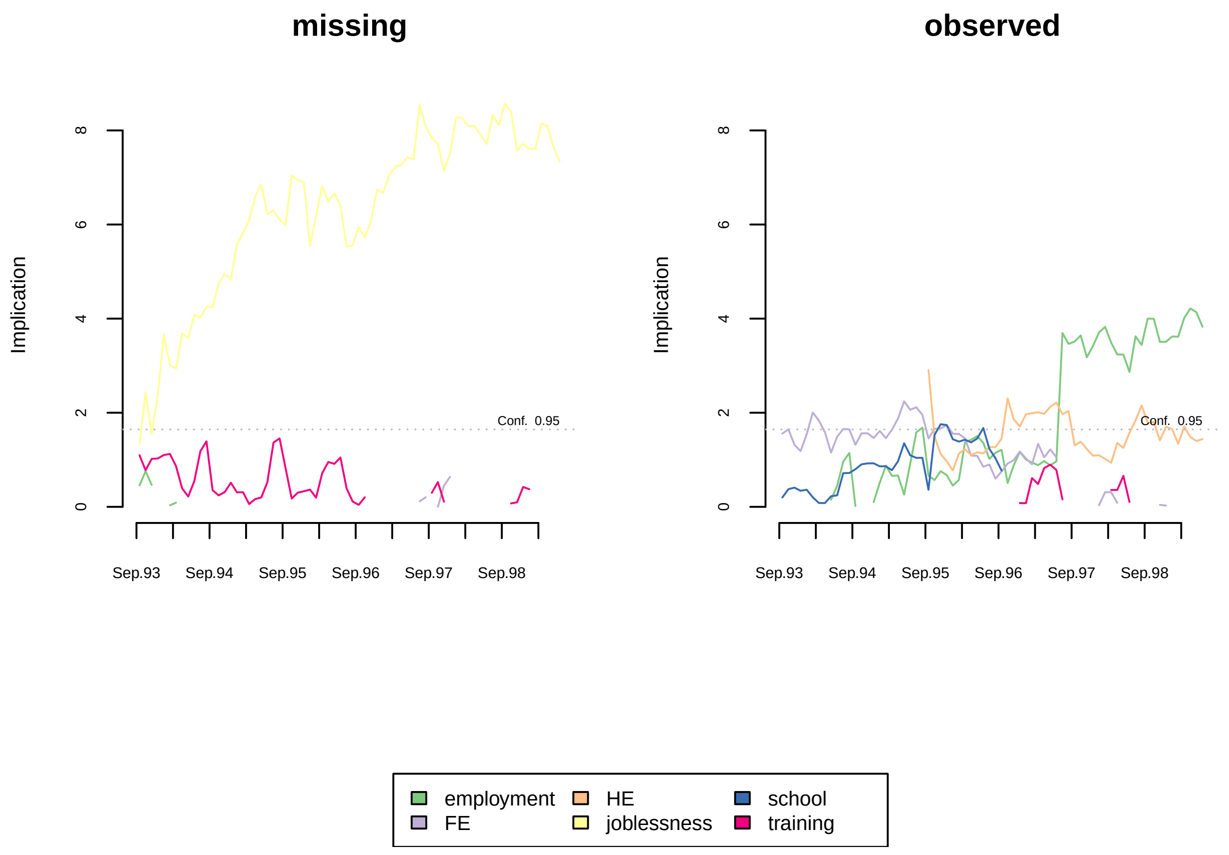


Patterns such as attrition need careful consideration. For example, in health trajectories, attrition patterns might signal the death of an individual, making imputation inappropriate. Such cases must be addressed before imputations are considered.

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 et al., 2024). It allows to display the states that better characterize, at each time point, sequences with missing data vs. sequences without missing data.

```
imp <- seqmissimplic(mvadseq.miss)
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 and in trajectories with missing values.

5.2 Multiple imputation

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

During the implementation of this process, several decisions must be made regarding the specification of the imputation model. For illustrative purposes, we use a simplified imputation method with standard arguments. However, this model may prove to be too simplistic for real-world applications. In section 6, we provide comprehensive guidelines for navigating the complexities of choosing an appropriate imputation model.

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 using a small number of imputations, as in this example. To reproduce the same results presented here, one should set a seed value before starting the imputation process.

Therefore, we set a seed value before generating two complete records with the `seqimpute()` function.

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

5.3 Typology creation

The next step is to cluster these multiple imputed datasets. In this document we follow the strategy of Halpin (2016). All imputed datasets are stacked together, the clustering is then identified, and finally it is re-distributed among the imputed datasets. As discussed in the 2 section, this strategy is not the only one available. Clustering sequences with missing data is mostly an open problem.

To accomplish this step, we first organize the imputed datasets into the desired format. Then we compute pairwise dissimilarities. Next, we determine different clusterings and select the best one among them based on cluster quality indices. Finally, we store the cluster labels within the imputed datasets for the next step. After selecting the clustering, we display it and identify the cluster that captures the unemployment trajectories.

For this illustration, we follow the procedure of the `WeightedCluster` vignette (Studer, 2013). We use hierarchical clustering with Ward's linkage, and dissimilarities are computed using Hamming distance.

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")
   stackedimpseq <- seqdef(stackedimp, xtstep=6)
   ```

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

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

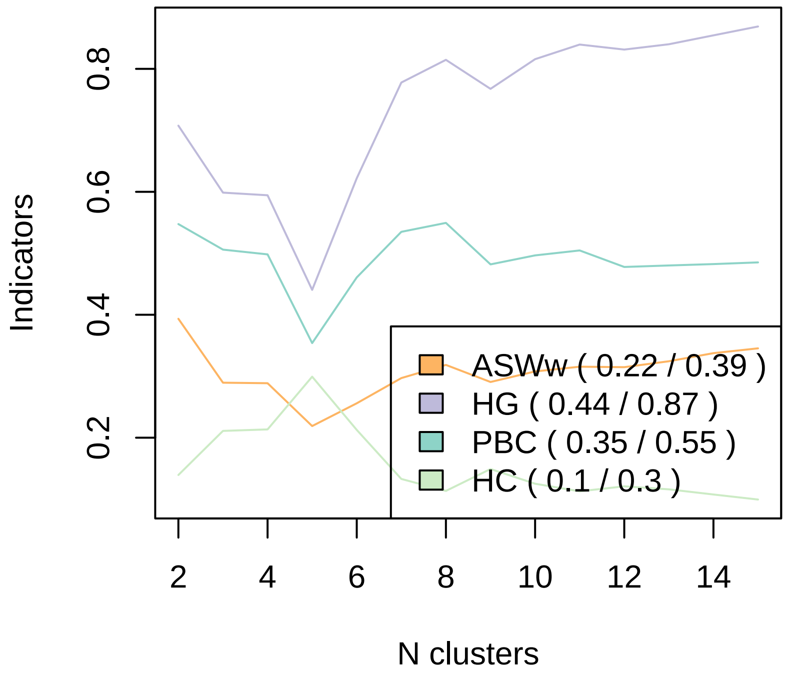
3. The hierarchical clustering with Ward linkage is computed.

```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")
wardRange <- as.clustrange(wardCluster, diss=ham, ncluster=15)

plot(wardRange, stat=c("ASWw", "HG", "PBC", "HC"))
```



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

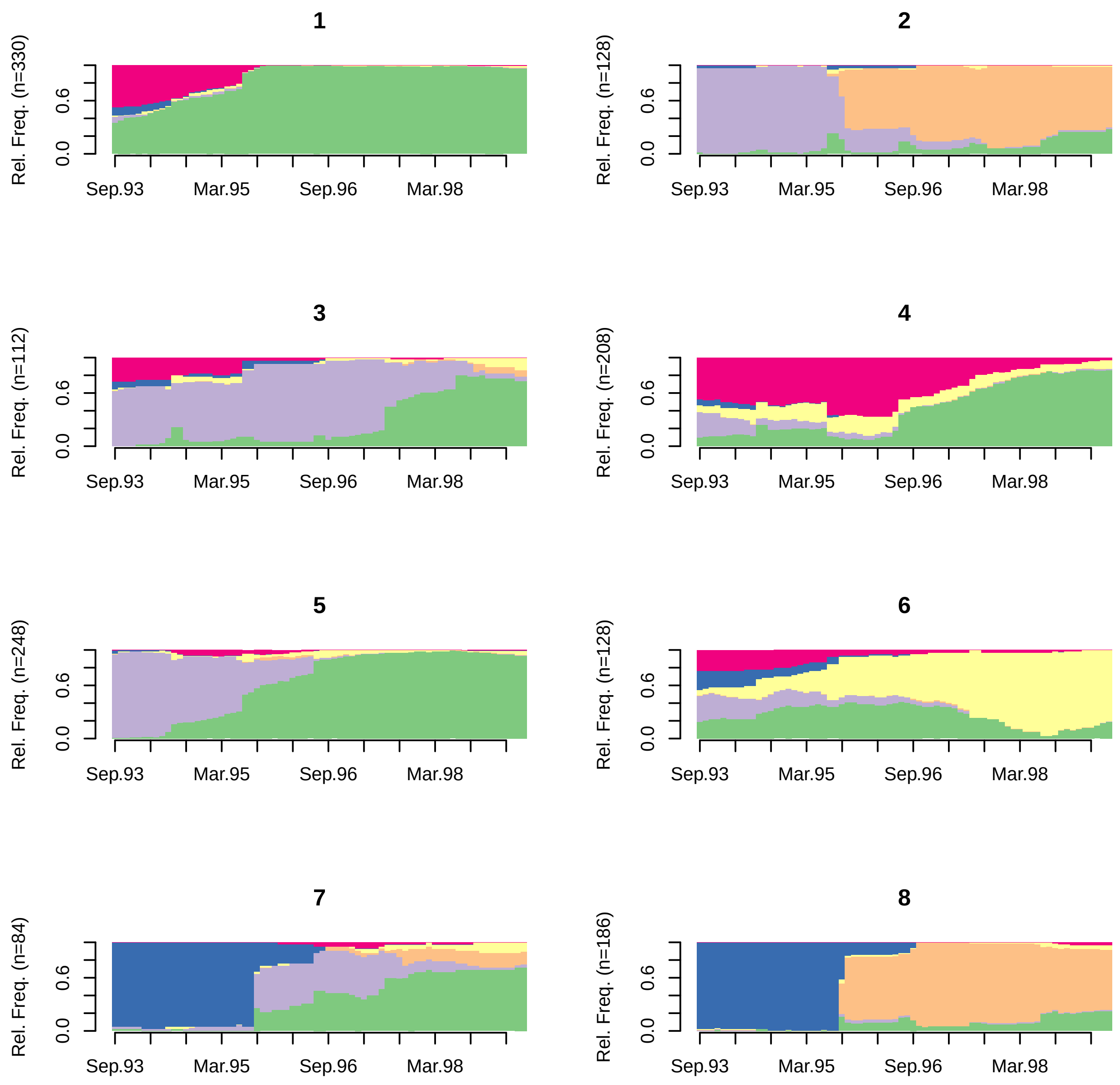
5. After performing hierarchical clustering and selecting eight clusters, we proceed to assign these clusters to the *seqimp* object with the `addcluster()` function.

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

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

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

```
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 cluster membership and the independent variable is the 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.

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

2. 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.

   ```
   library("mice")
   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.

   ```
   summary(pool(fit))

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

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

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

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

6 Guidelines related to the imputation model

In the previous section, 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, deciding on the number of multiple imputations, and considering the inclusion of covariates.

For each decision, we provide practical guidelines and insights into their application.

In the context of the mvad dataset, the majority of transitions occur during the summer months. The MICT does not discriminate between summer and other months, resulting in a shortage 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.

1. **MICT or MICT timing** If the dataset has a timing structure, MICT-timing multinomial should be used, while if it does not, MICT is preferred.

   To apply the MICT-timing algorithm, the argument *timing* must be set to *TRUE*.The MICT-timing algorithm requires an additional parameter, the radius of the timeframe. We recommend using a small radius (0 or 1) except when the number of trajectories is very small (less than 200 in our experience). Otherwise, its use is similar to the MICT algorithm. One must specify the number of previous observations (np), the number of subsequent observations, and the number of imputations (m).

   ```r
   imputed <- seqimpute(mvad.miss, var=17:86, np=1, nf=1, m=3, timing=TRUE,
   frame.radius=0)
   ```

2. **Number of previous and subsequent predictors** In the minimalist example, we relied on only one observation in the past and one in the future for the prediction. The underlying assumption of this approach was that using only past and future observations would be sufficient for accurate value prediction. However, in many real-world scenarios, it's likely that long-term effects are at play.

   ```r
   imputed <- seqimpute(mvad.miss, var=17:86, np=5, nf=5, m=3, timing=TRUE,
   frame.radius=0)
   ```

3. **Number of imputations**With multiple imputation, we create multiple complete records with no missing data. Therefore, one question is how many imputed records 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.

4. **Inclusion of covariates** Incorporating carefully chosen covariates can significantly enhance the quality of imputations, thereby mitigating bias in subsequent statistical analyses. We will elaborate on the specific covariates that should be included and the rationale behind their selection. Additionally, we will 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)):

   - Elicit distinct unfolding patterns in trajectories,
   - Potentially relate to the missing data mechanism,
   - Will be considered in the subsequent statistical analysis.

   For instance, professional trajectories differ between men and women, with men often characterized by full-time employment and women more inclined towards part-time employment. Neglecting to include the variable of sex in the imputation process would result in imputed values that fail to account for these notable differences. Along the same line, if missing data depends on some covariates, including these in model is crucial. For example, Finally, including the covariates that will be used in the subsequent statistical analysis 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. Additionally, considering that transitions may vary based on the type of secondary education, we include this covariate in the imputation models as well.

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

```r
imputed.cov <- seqimpute(mvad.miss, var=17:86, np=5, nf=5, m=3, timing=TRUE,
frame.radius=0, 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.

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 et al., 2014; Doove et al., 2014). They possess several advantageous features such as the ability to capture non-linear relationships, interactions, and immunity to irrelevant predictors (Friedman 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:

```r
set.seed(2)
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

Imputing data through multiple imputations can impose a considerable computational burden. However, the seqimpute() function provides a valuable solution by allowing for parallel computing to expedite these calculations. Since each individual imputation is entirely independent of the others, it becomes feasible to execute multiple imputations simultaneously in parallel, significantly enhancing computational efficiency.

```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 Conclusion

This document described the functioning of the `seqimpute` package, which implements the MICT and MICT-timing imputation methods.

9 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.

Bibliography

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

Burgette, L. F. and Reiter, J. P. (2010). Multiple imputation for missing data via sequential regression trees. *American journal of epidemiology*, 172(9):1070–1076.

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

Emery, K. (2023). *Handling missing data in multichannel life course analysis.* PhD thesis, University of Lausanne.

Faucheux, L., Resche-Rigon, M., Curis, E., Soumelis, V., and 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., Müller, N. S., and Studer, M. (2011). Analyzing and visualizing state sequences in R with TraMineR. *Journal of Statistical Software*, 40(4):1–37.

Gomer, B. and Yuan, K.-H. (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.

Halpin, B. (2013). Imputing sequence data: Extensions to initial and terminal gaps, Stata's *mi*. *Department of Sociology Working Paper Series*, University of Limerick.

Halpin, B. (2016). Multiple imputation for categorical time series. *The Stata Journal*, 16(3):590–612.

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

Little, R. J. (1992). Regression with missing x's: a review. *Journal of the American statistical association*, 87(420):1227–1237.

Little, R. J. and Schenker, N. (1995). Missing data. In Arminger, G., Clogg, C. C., and Sobel, M. E., editors, *Handbook of statistical modeling for the social and behavioral sciences*, pages 39–75. Springer.

McVicar, D. and 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, M. G., Tsiatis, A., and Verbeke, G. (2014). *Handbook of missing data methodology*. Chapman & Hall/CRC handbooks of modern statistical methods. CRC Press.

Perkins, N. J., Cole, S. R., Harel, O., Tchetgen Tchetgen, E. J., Sun, B., Mitchell, E. M., and Schisterman, E. F. (2018). Principled approaches to missing data in epidemiologic studies. *American journal of epidemiology*, 187(3):568–575.

Ritschard, G., Studer, M., Buergin, R., Liao, T. F., Gabadinho, A., Fonta, P.-A., Muller, N. S., and Rousset, P. (2024). Package 'traminerextras'.

Rothenbühler, M. and Voorpostel, M. (2016). Attrition in the swiss household panel: Are vulnerable groups more affected than others? In Oris, M., Roberts, C., Joye, D., and Ernst Stähli, M., editors, *Surveying Human Vulnerabilities across the Life Course*, pages 223–244. Springer.

Rubin, D. B. (1976). Inference and missing data. *Biometrika*, 63(3):581–592.

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

Shah, A. D., Bartlett, J. W., Carpenter, J., Nicholas, O., and 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. Swiss National Centre of Competence in Research LIVES, Geneva.

Van Buuren, S. (2018). *Flexible Imputation of Missing Data*. Chapman & Hall/CRC Interdisciplinary Statistics. CRC Press.

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