Imputing longitudinal categorical covariates in *R* using the `ImputeLongiCovs` package

Pavlos Mamouris, Vahid Nassiri

Introduction

The purpose of this vignette is to explain the theoretical background of the functions used in the R package `ImputeLongiCovs`. There are two functions inside the `ImputeLongiCovs` R package, namely `create_probMatrix` and `impute_categorical_covariates`. The `create_probMatrix` function creates a new column which represents the probability matrix, and the `impute_categorical_covariates` function imputes the longitudinal categorical covariates via a joint transition model.

Setup

Install and load the `ImputeLongiCovs` package

You can install and load the `ImputeLongiCovs` package directly from R. After you install the `ImputeLongiCovs` package, you can load it to use its functions using the library function. Note that you have to load the *R* package that you need to use each time you start a new *R* session, however installation only needs to occur once.

```
install.packages("ImputeLongiCovs")
```

```
library(ImputeLongiCovs)
```

Datasets

`initial_data` dataset

The `initial_data` dataset contains the pre-processed data and will be used to explain the function `create_probMatrix`. You can load the dataset and inspect the first 7 rows like this:

```
data(input_data = initial_data, package = "ImputeLongiCovs")
head(initial_data, 7)
#>     patient_id tran_Year transition_year   state_from     state_to
#> 1    patient_1         1  tran_2019_2020 never-smoker         <NA>
#> 2    patient_1         2  tran_2020_2021         <NA>         <NA>
#> 3   patient_10         1  tran_2019_2020         <NA>    ex-smoker
#> 4   patient_10         2  tran_2020_2021    ex-smoker         <NA>
#> 5  patient_100         1  tran_2019_2020         <NA>         <NA>
#> 6  patient_100         2  tran_2020_2021         <NA>       smoker
#> 7 patient_1000         1  tran_2019_2020 never-smoker never-smoker
#>   cardio_state_to cardio_state_from flu_vaccination_state_from
#> 1               0                 0                          0
#> 2               0                 0                          0
#> 3               0                 0                          0
#> 4               1                 0                          0
#> 5               0                 0                          0
#> 6               0                 0                          0
#> 7               0                 0                          0
#>   flu_vaccination_state_to
#> 1                        0
#> 2                        0
#> 3                        0
#> 4                        0
#> 5                        0
#> 6                        0
#> 7                        0
```

The `initial_data` dataset contains nine columns. The `patient_id` is the patient identification. `tranYear` is a numeric column starting from 1, i.e. the first transition up to the total number of transitions, namely 2 in our case. Finally, the `transition_year` is an auxiliary column that clarifies the `tranYear` column. For instance, `tranYear` = 1 is the `transition_year` = tran_2019_2020, namely the transition occurred from year 2019 to 2020. `tranYear` = 2 is the equivalent of `transition_year` = tran_2020_2021, namely the transition occurred from year 2020 to 2021. The `state_from` denotes the initial state of the transition, whereas the `state_to` denotes the end state of the transition. `cardio_state_from` is the cardiovascular disease at the beginning of the transition (1 == Yes, 0 == No), `cardio_state_to` is the ardiovascular disease at the end of the transition, `flu_vaccination_state_from`, `flu_vaccination_state_to` the flu vaccination status at the beginning and the end of the transition. These are the covariates to be used in this tutorial. We used a longitudinal data with 3 waves from 2019 until 2021, thus we end up with 2 transitions. Important to mention is that the user has to perform some minor data-manipulation to reach the data in this longitudinal format. Therefore, the following variables must be present in the dataset:

- `state_from` = a character variable (e.g., `c("smoker" "ex-smoker", "never-smoker")`)
- `state_to` = a character variable (e.g., `c("smoker" "ex-smoker", "never-smoker")`)
- `tranYear` = a numeric variable (1 : number of transitions)

The rest variables, i.e. (`patient_id` , covariates) can be determined by the user.

`analyses_data` dataset

The `analyses_data` dataset contains the processed data and will be used to clarify the function `impute_categorical_covariates`. You can load the dataset and inspect the first 7 rows like this:

```
data(analyses_data, package = "ImputeLongiCovs")
head(analyses_data, 7)
#>     patient_id tran_Year transition_year   state_from     state_to prob_matrix
#> 1    patient_1         1  tran_2019_2020 never-smoker         <NA>     forward
#> 2    patient_1         2  tran_2020_2021         <NA>         <NA>     forward
#> 3   patient_10         1  tran_2019_2020         <NA>    ex-smoker    backward
#> 4   patient_10         2  tran_2020_2021    ex-smoker         <NA>     forward
#> 5  patient_100         1  tran_2019_2020         <NA>         <NA>    backward
#> 6  patient_100         2  tran_2020_2021         <NA>       smoker    backward
#> 7 patient_1000         1  tran_2019_2020 never-smoker never-smoker    observed
#>   cardio_state_to cardio_state_from flu_vaccination_state_from
#> 1               0                 0                          0
#> 2               0                 0                          0
#> 3               0                 0                          0
#> 4               1                 0                          0
#> 5               0                 0                          0
#> 6               0                 0                          0
#> 7               0                 0                          0
#>   flu_vaccination_state_to
#> 1                        0
#> 2                        0
#> 3                        0
#> 4                        0
#> 5                        0
#> 6                        0
#> 7                        0
```

The `analyses_data` dataset contains 10 columns. The same nine columns as the `initial_data` with an extra column, namely the `prob_matrix`. `prob_matrix` column was generated via the `create_probMatrix` function and accommodates all possible transitions. Those transitions include the `initial`, `forward`, `backward`, `intermittent`, and `observed`, which will be explained in details in the next section.

Function 1 (pre-processing): `create_probMatrix`

Let us return to the `initial_data` and inspect the first 7 rows:

```
data(initial_data, package = "ImputeLongiCovs")
head(initial_data, 7)
#>     patient_id tran_Year transition_year   state_from     state_to
#> 1    patient_1         1   tran_2019_2020 never-smoker         <NA>
#> 2    patient_1         2   tran_2020_2021         <NA>         <NA>
#> 3   patient_10         1   tran_2019_2020         <NA>    ex-smoker
#> 4   patient_10         2   tran_2020_2021    ex-smoker         <NA>
#> 5  patient_100         1   tran_2019_2020         <NA>         <NA>
#> 6  patient_100         2   tran_2020_2021         <NA>       smoker
#> 7 patient_1000         1   tran_2019_2020 never-smoker never-smoker
#>   cardio_state_to cardio_state_from flu_vaccination_state_from
#> 1               0                 0                          0
#> 2               0                 0                          0
#> 3               0                 0                          0
#> 4               1                 0                          0
#> 5               0                 0                          0
#> 6               0                 0                          0
#> 7               0                 0                          0
#>   flu_vaccination_state_to
#> 1                        0
#> 2                        0
#> 3                        0
#> 4                        0
#> 5                        0
#> 6                        0
#> 7                        0
```

The `create_probMatrix` has two arguments:

- `input_data` : A dataset in a format similar to `initial_data`
- `patient_id` : A character variable that specifies the column name with the unique Id of the patient

We apply the `create_probMatrix` function on the `initial_data` and store the result in the `initial_data_after_function` dataset. A new column `prob_matrix` is created:

```
initial_data_after_function <- create_probMatrix(initial_data, patient_id = "patient_id")
head(initial_data_after_function, 7)
#>     patient_id tran_Year transition_year   state_from     state_to
#> 1    patient_1         1  tran_2019_2020 never-smoker         <NA>
#> 2    patient_1         2  tran_2020_2021         <NA>         <NA>
#> 3   patient_10         1  tran_2019_2020         <NA>    ex-smoker
#> 4   patient_10         2  tran_2020_2021    ex-smoker         <NA>
#> 5  patient_100         1  tran_2019_2020         <NA>         <NA>
#> 6  patient_100         2  tran_2020_2021         <NA>       smoker
#> 7 patient_1000         1  tran_2019_2020 never-smoker never-smoker
#>   cardio_state_to cardio_state_from flu_vaccination_state_from
#> 1               0                 0                          0
#> 2               0                 0                          0
#> 3               0                 0                          0
#> 4               1                 0                          0
#> 5               0                 0                          0
#> 6               0                 0                          0
#> 7               0                 0                          0
#>   flu_vaccination_state_to prob_matrix
#> 1                        0     forward
#> 2                        0     forward
#> 3                        0    backward
#> 4                        0     forward
#> 5                        0    backward
#> 6                        0    backward
#> 7                        0    observed
```

Let us further use 4 patients to showcase this function:

```
initial_data_subset <- initial_data_after_function[which(initial_data_after_function$patient_id %in%
    c("patient_10", "patient_102", "patient_114", "patient_136")), ]
initial_data_subset <- initial_data_subset[, c(1:5, 10)]
initial_data_subset
#>     patient_id tran_Year transition_year   state_from     state_to  prob_matrix
#> 3   patient_10         1  tran_2019_2020         <NA>    ex-smoker     backward
#> 4   patient_10         2  tran_2020_2021    ex-smoker         <NA>      forward
#> 11 patient_102         1  tran_2019_2020 never-smoker never-smoker     observed
#> 12 patient_102         2  tran_2020_2021 never-smoker         <NA>      forward
#> 37 patient_114         1  tran_2019_2020         <NA>         <NA>      initial
#> 38 patient_114         2  tran_2020_2021         <NA>         <NA>      forward
#> 85 patient_136         1  tran_2019_2020    ex-smoker         <NA> intermittent
#> 86 patient_136         2  tran_2020_2021         <NA>    ex-smoker intermittent
```

- Patient `10` has two levels of the `prob_matrix`, namely `backward` for the transition 1, and `forward` for the transitions 2. We observe that the first observed smoking status occurred at year 2020 (i.e. `ex-smoker`).

We imputed the smoking status for the year before, i.e. 2019 (`backward`), and for the year after, i.e. 2021 (`forward`) via a joint transition model.

- Patient `102` has two levels of the `prob_matrix`, namely `observed` for the transition 1, and `forward` for the transitions 2. Here we have the `prob_matrix` as `observed` in transition 1 since the smoking status of both years 2019 & 2020 was observed.
- Patient `114` has two levels of the `prob_matrix`, namely `initial` for the transition 1, and `forward` for the transition 2. We observe that this patient had no observed smoking status at all. Thus, we apply a multinomial logistic regression in the transition 1 (`initial`) and for the years after, i.e. 2020-2021 (`forward`), forward transition probabilities were used for imputing the missing values.
- Patient `136` has `intermittent` missingness. We observe that this patient had two observed smoking status at years 2019 and 2021 (both `ex-smoker`). Therefore, year 2019 was imputed in a different way (`intermittent`). The intermittent imputation process will be discussed in details in the next section.

To conclude, if a smoking record exists in the longitudinal waves, imputation based on transition probabilities was applied. If no smoking status was recorded, a multinomial regression model was used in the first year, and forward transition probabilities from the starting year to each of the subsequent years were used for imputing the missing values.

Function 2 (Imputation stage): `impute_categorical_covariates`

Let us revisit the `analyses_data` and inspect the first 7 rows:

```
data(analyses_data, package = "ImputeLongiCovs")
head(analyses_data, 7)
#>     patient_id tran_Year transition_year   state_from     state_to prob_matrix
#> 1    patient_1         1  tran_2019_2020 never-smoker         <NA>     forward
#> 2    patient_1         2  tran_2020_2021         <NA>         <NA>     forward
#> 3   patient_10         1  tran_2019_2020         <NA>    ex-smoker    backward
#> 4   patient_10         2  tran_2020_2021    ex-smoker         <NA>     forward
#> 5  patient_100         1  tran_2019_2020         <NA>         <NA>    backward
#> 6  patient_100         2  tran_2020_2021         <NA>       smoker    backward
#> 7 patient_1000         1  tran_2019_2020 never-smoker never-smoker    observed
#>   cardio_state_to cardio_state_from flu_vaccination_state_from
#> 1               0                 0                          0
#> 2               0                 0                          0
#> 3               0                 0                          0
#> 4               1                 0                          0
#> 5               0                 0                          0
#> 6               0                 0                          0
#> 7               0                 0                          0
#>   flu_vaccination_state_to
#> 1                        0
#> 2                        0
#> 3                        0
#> 4                        0
#> 5                        0
#> 6                        0
#> 7                        0
```

The `impute_categorical_covariates` function uses the categories of the `prob_matrix` column and implements a joint longitudinal transition model that accommodates different scenarios (`initial`, `forward`, `backward`, `intermittent`). As we explained previously, the `state_from` variable denotes the initial state of the transition, whereas the `state_to` denotes the end state of the transition. The smoking outcome has 3 states, say (r), (`smoker`, `ex-smoker` ,`never-smoker`) at the beginning of the transition and 3 states, say (s) (`smoker`, `ex-smoker` ,`never-smoker`) at the end of the transition. Inside the `impute_categorical_covariates`, we further enclose 3 different functions:

- `initial_forward_function`: Here we impute the smoking status based on whether in that transition the `prob_matrix` of a patient was `initial` or `forward`. For `initial`, we used a multinomial logistic regression model to derive the initial probability of the r state given covariates X, $P(r|X)$. For `forward`, we used another multinomial logistic regression model to derive the forward transition probabilities $P(s|r, X)$.
- `imputeIntermittent` function: To derive the intermittent transition probabilities from state r to state s requires an extra effort since there is an open and close condition. Suppose a patient that is `smoker` in year 2010 and `ex-smoker` in year 2013. Therefore we have 3 transitions, namely from 2010 to 2011, 2011 to 2012 and 2012 to 2013. We estimated all the in-between paths and stored them in a sequence. This should a priory sum up to $3^3 = 27$ sequences, however knowing the starting condition (smoker) and the ending one (ex-smoker), we have 9 sequences left. For these 9 sequences, once we calculated their probability, we re-standardized them back to 1. Finally, we sampled one of these sequences and filled in the intermittent path.
- `backward_function`: We used two multinomial logistic regression models. First, we derived the initial probability of the r state, $P(r|X)$, and the forward transition probabilities $P(s|r, X)$. Subsequently, we calculated the joint probability $P(r, s|X) = P(s|r, X) \times P(r|X)$. Having the joint probability distribution of the 3 states, we could compute the initial probabilities $P(s|X)$. Now, the backward transition probabilities $P(r|s, X)$ could be computed as follows:

$$P(r|s, X) = \frac{P(r, s|X)}{p(s|X)}$$

The `impute_categorical_covariates` has 9 arguments:

- `input_data` : A dataset in a format similar to `analyses_data`
- `patient_id` : A character variable that specifies the column name with the unique Id of the patient
- `number_of_transitions` : The number of transitions needed. The maximum of the `tranYear` column.
- `covariates_initial = NULL` : The covariates to be used in the initial model
- `covariates_transition = NULL` : The covariates to be used in the transition model
- `missing_variable_levels` : The levels of the missing categorical outcome (e.g. `c("smoker" "exsmoker", "neversmoker")`)
- `startingyear = NULL` : If the starting year per patient has no missing values, specify it
- `without_trans_prob` : This statement is useful when there are very high proportions of missing data and our initial and transition model cannot converge. It provides the user with two options. One, to "notImpute", namely to return NA and two, to "ImputeEqualProbabilities", i.e., the user can sample with equal probabilities.
- `m = 1` : A numeric variable that specifies the number of imputed datasets. Default is m = 1.

To apply the `impute_categorical_covariates` function in the `analyses_data`, we can use the following arguments. Note that we used a seed for reproducibility reasons.

```r
set.seed(123)
imputed_smoking_status <- impute_categorical_covariates(input_data = analyses_data,
    patient_id = "patient_id", number_of_transitions = 2, covariates_initial = c("cardio_state_from",
        "flu_vaccination_state_from"), covariates_transition = c("cardio_state_to",
        "flu_vaccination_state_to"), missing_variable_levels = c("never-smoker",
        "smoker", "ex-smoker"), startingyear = NULL, without_trans_prob = "notImpute",
    m = 1)
```

Here, we imputed our dataset only once (m =1). The user can choose for more imputations by changing the `m` argument. This function returns a list of m (in this example m = 1) data frames with no missing values in the smoking outcome Let us inspect the first 21 rows from the imputed dataset.

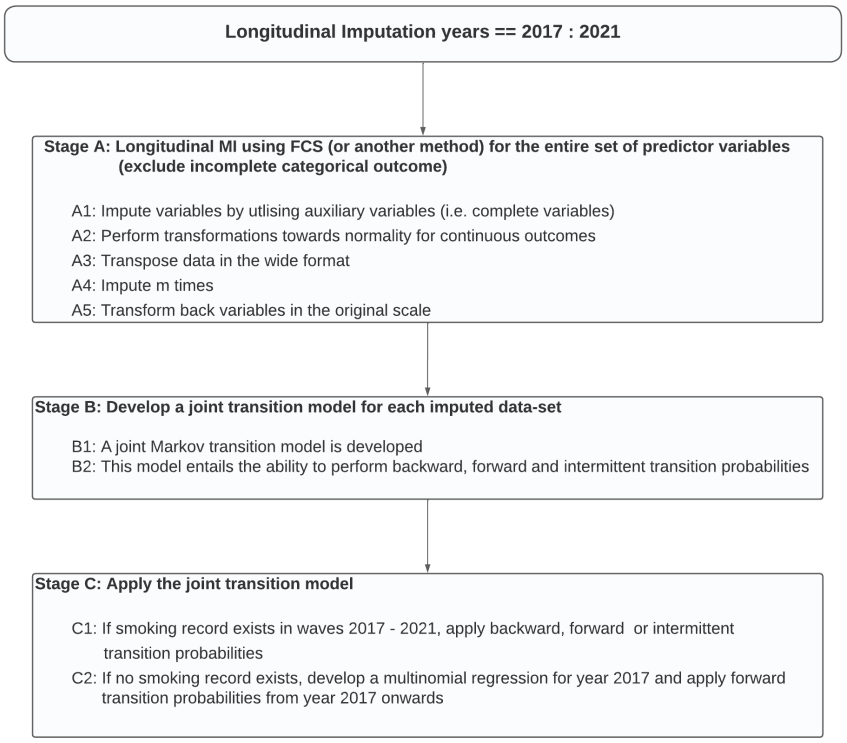
```r
imputed_smoking_status <- imputed_smoking_status[[1]]$input_data
imputed_smoking_status <- imputed_smoking_status[, c(1:5)]

head(imputed_smoking_status, 21)
#>      patient_id tran_Year transition_year   state_from     state_to
#> 1     patient_1         1  tran_2019_2020 never-smoker never-smoker
#> 2     patient_1         2  tran_2020_2021 never-smoker never-smoker
#> 3    patient_10         1  tran_2019_2020    ex-smoker    ex-smoker
#> 4    patient_10         2  tran_2020_2021    ex-smoker    ex-smoker
#> 5   patient_100         1  tran_2019_2020       smoker       smoker
#> 6   patient_100         2  tran_2020_2021       smoker       smoker
#> 7  patient_1000         1  tran_2019_2020 never-smoker never-smoker
#> 8  patient_1000         2  tran_2020_2021 never-smoker never-smoker
#> 9   patient_101         1  tran_2019_2020 never-smoker never-smoker
#> 10  patient_101         2  tran_2020_2021 never-smoker never-smoker
#> 11  patient_102         1  tran_2019_2020 never-smoker never-smoker
#> 12  patient_102         2  tran_2020_2021 never-smoker never-smoker
#> 13  patient_103         1  tran_2019_2020 never-smoker    ex-smoker
#> 14  patient_103         2  tran_2020_2021    ex-smoker    ex-smoker
#> 15  patient_104         1  tran_2019_2020       smoker       smoker
#> 16  patient_104         2  tran_2020_2021       smoker    ex-smoker
#> 17  patient_105         1  tran_2019_2020 never-smoker never-smoker
#> 18  patient_105         2  tran_2020_2021 never-smoker never-smoker
#> 19  patient_106         1  tran_2019_2020       smoker       smoker
#> 20  patient_106         2  tran_2020_2021       smoker    ex-smoker
#> 21  patient_107         1  tran_2019_2020    ex-smoker    ex-smoker
```

We can readily observe that the imputation of smoking outcome is executed successfully.

Extensions and Summary

In the previous sections, we considered that our covariates are complete. However, this is not always the case, and missingness can equally occur in other variables like body mass index, systolic and diastolic blood pressure and more. For this scenario, we developed a 3-stage methodology presented in our paper *"A longitudinal transition imputation model for categorical data applied to a large registry data set"*, from which we borrowed the following flowchart:


Longitudinal Imputation years == 2017 : 2021

Stage A: Longitudinal MI using FCS (or another method) for the entire set of predictor variables (exclude incomplete categorical outcome)

A1: Impute variables by utlising auxiliary variables (i.e. complete variables)
A2: Perform transformations towards normality for continuous outcomes
A3: Transpose data in the wide format
A4: Impute m times
A5: Transform back variables in the original scale

Stage B: Develop a joint transition model for each imputed data-set

B1: A joint Markov transition model is developed
B2: This model entails the ability to perform backward, forward and intermittent transition probabilities

Stage C: Apply the joint transition model

C1: If smoking record exists in waves 2017 - 2021, apply backward, forward or intermittent transition probabilities
C2: If no smoking record exists, develop a multinomial regression for year 2017 and apply forward transition probabilities from year 2017 onwards

In the first stage, we performed multiple imputation by using fully conditional specification (FCS) for the entire set of predictor variables and waves. It is important to note that FCS is not the only option here. The user can impute these variables in stage A with the imputation method of their choice, say (Multivariate normal Imputation, Bayesian Imputation and more). We included the continuous partially missing covariates, accompanied with selected auxiliary variables (categorical and continuous), which is shown to improve the accuracy of the imputations. For the missing continuous variables, we performed transformations towards normality to preserve their range (e.g., logarithmic transformation for triglycerides, inverse squared transformation for glucose, etc.) and transposed them using the wide format. Wide format is useful when performing longitudinal MI since the earlier and later information of the same patient is utilized. Next, we determined the appropriate imputation method for each variable, calculated the prediction matrix and, finally, generated 20 imputations, which is prudent as the percentage of missing values was substantial. Once the imputations were executed, we transformed the continuous variables back to their original stage. In this first stage, we did not impute the smoking variable.

Thus, the user should first perform m imputations using FCS (or another strategy) with their statistical package of choice, namely `mice`, `Amelia` or more. Subsequently, within each of the m imputed datasets, the user can apply the `ImputeLongiCovs` R package and its functions to execute the stages B and C.

In this tutorial we used as the longitudinal outcome of interest the smoking outcome with 3 states, (“smoker” “exsmoker”, “neversmoker”). Nevertheless, this package can be applied to other outcomes, say alcohol, `c("alcohol", "exalcohol", "neveralcohol")` or even with outcomes that have more than 3 states.