The Stata Journal (2016)
16, Number 3, pp. 590–612

Multiple imputation for categorical time series

Brendan Halpin
Department of Sociology
University of Limerick
Limerick, Ireland
brendan.halpin@ul.ie

Abstract. The `mict` package provides a method for multiple imputation of categorical time-series data (such as life course or employment status histories) that preserves longitudinal consistency, using a monotonic series of imputations. It allows flexible imputation specifications with a model appropriate to the target variable (`mlogit`, `ologit`, etc.). Where transitions in individual units' data are substantially less frequent than one per period and where missingness tends to be consecutive (as is typical of life course data), `mict` produces imputations with better longitudinal consistency than `mi impute` or `ice`.

Keywords: st0445, mict_impute, mict_prep, mict_model_gap, mict_model_initial, mict_model_terminal, multiple imputation, categorical time series

1 Missingness in longitudinal data

In this article, I describe an approach for multiple imputation of categorical cross-sectional time-series data, such as labor market or other life course histories. The method is implemented in the `mict` package. The approach focuses on filling gaps from their edges, using a monotonic series of imputations. It uses `mi impute mlogit` (or `mi impute ologit` if appropriate) to carry out single imputations, but it manages the sequencing of imputations independently of the `mi impute` infrastructure. It respects the longitudinal consistency of the data in a way that is difficult or impossible to achieve with standard `mi impute` or `ice` approaches, while allowing flexible imputation models and full access to the power of Stata's `mi` postimputation infrastructure.

The approach presented is well adapted for the sort of data typically seen in life course histories: a categorical state space observed on a regular basis (for example, monthly) over a reasonably extended period (for example, a small number of years), with transitions occurring at a relatively low rate. Hence, we observe spells in states that are significantly longer than one observation. Similarly, missing values will also tend to occur in runs (as gaps), due to a mixture of data-collection problems (respondent absence at consecutive data-collection points) and data structure (for example, part or complete spells being nonreported or misreported). In effect, these data are typified by having a discrete state space in continuous time approximated by discrete observations (for example, monthly) and will usually be collected retrospectively at one or more well-separated time points (for example, annually), often in terms of start and end dates of spells. Such data contain a relatively high amount of redundancy, such that information in the observed proportion is a good predictor of the missing part, and

the runs of missingness mean missing observations tend to have one or more missing neighbors.

Of course, forms of data other than life course histories may well also have these features, and the approach is equally valid for those data. Data with higher transition rates, or truly discrete time, will be harder to impute because there is more variation from observation to observation.

1.1 Longitudinal data and missingness

The availability of longitudinal data (such as labor market, family formation, or residential histories) is ever increasing, and methods for its analysis are becoming ever more common and widely used. However, longitudinal data are subject to missingness, often to a greater degree than cross-sectional data. While some methods can deal with missingness (for example, duration models can "censor" data from the first occurrence of missing observations onward), others cannot and so require full data. Throwing away individual histories because of a small proportion of missingness is wasteful (even for duration models), and if missingness is not "completely at random", to use Rubin's (1987) term, such deletion of cases may cause bias. Indeed, it is particularly likely that missingness is not random, in that volatile histories are disproportionately likely to result in missingness. This is not only because people who experience volatility will be more likely to miss data-recording opportunities (for example, annual interviews) but also because volatile histories have more opportunity for incompleteness. For instance, if you are in the same job for 10 years, missing an occasional interview will not impact the record, whereas if you have changed job five times since the last interview, there are far more opportunities for error to enter and for information to be lost.

2 Multiple imputation as a solution

Introduced by Rubin (1987), multiple imputation has become a standard way of dealing with missing data. Where data are missing in a manner that is random conditional on the fully observed variables (missing at random [MAR]), regression models can be used to predict the incompletely observed variables using the fully observed ones. These are then used to impute values to replace the missing, drawn at random from the prediction distribution of the fit regression model. Rubin's key insight is that if multiple imputations are drawn, creating multiple imputed datasets, and if the results of analyses on the multiple datasets are averaged, then unbiased estimates of the desired quantity (such as a regression coefficient) can be made.[1] Stata has implemented a package of commands for creating, managing, and analyzing such imputations (see `mi impute`).

However, cross-sectional time-series data such as life histories put a substantial strain on standard imputation. If in wide format (with one variable per time unit), there are many very similar variables, all likely to be subject to missingness. Consider the example

1. More strictly, parameter estimates are averaged across the multiple imputed datasets, and the variance depends on both the variance within and the variance between analyses.

of five years of monthly employment statuses, yielding 60 very similar observations. Imputing each on the basis of all others will be computationally challenging, if not impossible, and it is (at best) difficult to define selective-imputation models for each time-unit observation. Moreover, as will be shown below, while standard approaches perform well in terms of the distribution of individual variables, consecutive imputed variables will tend to vary too much relative to each other; thus, from a longitudinal point of view, the imputed data will have transition rates that are significantly biased upward.

Where a single variable is subject to missingness, imputation is straightforward. When multiple variables have missing values, the complication may arise that cases to be imputed have missing values in the predictor variables. It may be the case that variables to be imputed can be arranged in an order such that the first has no missing predictors, the next has missing predictors only on the first, the next only on the first and second, and so on. If such a monotonic pattern of missingness is present, imputation can proceed following the same order, such that at each predictive step, all the predictors are either fully observed or already imputed. If so, imputation with multiple variables to be imputed becomes a simple extension of imputation of a single variable. However, such a monotonic pattern is not likely to emerge without a structural reason for it (attrition in longitudinal data is a typical example). In its absence, there are two approaches to multiple imputation with multiple variables subject to missingness: modeling the joint distribution of the variables directly (JM) or multiple imputation by chained equations (MICE).

JM is attractive where the variables are, or can be, transformed such that the joint distribution is multivariate normal. It is efficient and has good theoretical foundations. Where some variables are categorical, conventional practice is to use linear predictions of dummy variables. However, this has been shown to produce poor results in certain conditions (Allison 2005; van Buuren 2007). The alternative is so-called fully conditional specification (FCS) where, rather than modeling the joint distribution, the conditional distributions are modeled. This allows variable-specific imputation models to be used (including models appropriate to categorical data) and is thus more flexible. MICE implements FCS.

Monotonic imputation, where applicable, is also flexible, allowing variable-specific imputation models and permitting forms such as logistic regression (van Buuren 2007).

MICE works as follows: In the first round, all missing values are imputed with a minimal model (hot-decking, or regressions using fully observed predictors only). Then, the imputations are replaced by better imputations based on a full model using observed and imputed values. This process is repeated for a number of cycles, effectively diluting the influence of the poor-quality first-round imputations. It has been implemented for R by van Buuren, Boshuizen, and Knook (1999), van Buuren (2007), and van Buuren and Groothuis-Oudshoorn (2011), and for Stata by Royston (2004, 2009).

More recent versions of Stata have incorporated equivalent functionality in the core `mi impute` infrastructure. These are excellent implementations, providing good imputations and tools that make `mi` easy to use, but they do not suit time-series data in either

wide or long format. In wide format, there are many poorly distinguished variables with wide incidence of missingness; it is hard to write adequate imputation models, and they will tend to have severe problems in estimation. In long format, the relevant predictors are lags and leads, and the infrastructure is not adapted to using and updating lagged and leading variables.

Two other packages address imputation of longitudinal data: Amelia (for R and Stata) (Honaker and King 2010), and `twofold` (for Stata) (Welch, Bartlett, and Petersen 2014; Nevalainen, Kenward, and Virtanen 2009). Amelia is written explicitly to respect the longitudinal logic of time series. However, it implements the JM approach to imputation. So, while its authors make passing reference to imputing categorical variables, it is likely to generate poorer imputations for categorical data than packages that can use appropriate forms of model, such as binary or multinomial logistic. The `twofold` package uses an FCS approach, using moving time-windows to restrict the number of predictor variables, in a chained-equation framework. It thus captures the longitudinal logic in a manner comparable but not identical to multiple imputation for categorical time series (MICT). While `twofold` is not specific to categorical time-series data, the FCS approach is capable of dealing with it well. `mict` and `twofold` also differ in that the former fits a monotonic series of models, while the latter estimates chained equations (requiring burn-in, etc.). The main benefit of `twofold` is that it makes it much easier to express MICE models with longitudinal data.

3 Filling gaps in life course data

When data contain multiple observations per individual, the long format (one observation per person–time-unit) is natural, though (particularly when all individuals should be observed for the same time span) the wide format is also appropriate, if a little less natural. However, Stata's `mi impute` infrastructure is not designed to deal with data in a long format, requiring imputation variables to be in the same record. The method reported here exploits the `mi impute mlogit` command, but it uses the long format and handles the management of predictor variables (in particular, lags and leads of the target state), the storing of imputed values, and the sequencing of the imputations independently of Stata's `mi` infrastructure.

Conceiving of the data as longitudinal in nature reduces the many variables to a single state variable (indexed by time) and places the focus on gaps rather than individual missing variables. We can use a single form of model to predict all candidate observations. However, the presence of gaps means that no single model can apply to all time points [for example, the state at $(t + 1)$ will not be observed for all cases]. By focusing directly on gaps, we can nonetheless define a family of models that can be fit in a monotonic series. We start by imputing the first (or, equally validly, the last) element of the longest gap with data from the nearest observed points, before and after. That is, the first element of a gap is predicted by data from the element immediately preceding the gap and the element immediately following the gap. If we restrict ourselves to internal gaps, the lag and lead data are guaranteed not to be missing.

Once the first element of the longest gap has been filled, it holds that for the next-longest gap, the lag and lead data are either observed or already imputed. Thus, we continue by imputing the last element of the next-longest gap, using the last observed (or imputed) value before the gap and the immediately subsequent value. The process continues (alternating between first and last elements of the gaps) until all internal gaps are filled. Gaps at the start and end of the series can be imputed using an analogous approach that uses only information from, respectively, after and before the gap.

We can illustrate gap filling with the following example, with two sequences containing a six-element and a three-element gap, respectively (see table 1). We begin (in line two) by imputing the first element of six-unit gaps, using information pertaining to the last observed $(t-1)$ and next observed $(t+6)$ time points. Nothing happens to the shorter sequence. Then, we impute the fifth element of five-unit gaps, using $(t+1)$ and $(t-5)$, and then the first element of four-unit gaps. The remaining steps affect both sequences because the six-unit gap has been reduced to three, and it imputes the third element of three-unit gaps. The process continues until all gaps are filled.

Table 1. Gap-filling sequence of imputation

	Six-unit gap	Three-unit gap
Observed data with gaps	XX.....XXX	XXX...XXXXX
Impute first element of 6-unit gaps	XXi.....XXX	XXX...XXXXX
Impute last element of 5-unit gaps	XXI....iXXX	XXX...XXXXX
Impute first element of 4-unit gaps	XXIi...IXXX	XXX...XXXXX
Impute last element of 3-unit gaps	XXII..iIXXX	XXX..iXXXXX
Impute first element of 2-unit gaps	XXIIi.IIXXX	XXXi.IXXXXX
Impute only element of 1-unit gaps	XXIIIiIIXXX	XXXIiIXXXXX

X: observed data; .: missing data; i: data being imputed; X: observed data used as predictor; I: imputed data used as predictor; I: previously imputed data

The data used to predict must at least include the state at the prior and subsequent observed time points. It should also include summaries of the prior and subsequent experience, and it can contain any other data keyed to these time points (that is, state in another time-dependent state space) and contain fixed individual-level information. All the considerations regarding the requirements of a good imputation model in conventional circumstances apply equally here—in particular, the imputation model should be "congenial" (Allison 2009, 84) with the analysis model—but with the additional requirement that longitudinal information in the imputed state must be included.

4 Syntax

`mict` consists of two key commands: `mict_prep` sets up the data and `mict_impute` carries out the imputations. The imputation models are specified in further helper commands (`mict_model_gap`, `mict_model_initial`, and `mict_model_terminal`), which the user will usually overwrite to specify more appropriate models (as outlined on page 598).

`mict_prep` must be called first, with the following syntax:

`mict_prep` *stublist*`, id(`*varname*`)`

The *stublist* refers (at a minimum) to the variables in the categorical time series, which should be consecutive and named from *stublist*`1` to *stublistN* in wide format. The *stublist* value will be passed internally to `reshape` (see [D] **reshape**). The `id(`*varname*`)` option designates an ID variable. `id()` is required. The `mict_prep` command puts the data in long format and creates several variables that will be used by `mict_impute`. If other time-dependent variables exist that might be used on the right-hand side of the imputation, they can be included in the *stublist*. For example, if the main variable is `state1` to `state`*N* and a parallel time series exists as `auxvar1` to `auxvar`*N*, then both can be made available as follows:

```
. mict_prep state auxvar, id(id)
```

The variable `auxvar` can then be used in the imputation model specification, but it needs to be fully observed.

`mict_prep` creates several internal variables that are used by `mict_impute`:

- `_mct_state`: A copy of the state variable (that is to be imputed).
- `_mct_next`: The next observed state (after the current gap).
- `_mct_last`: The prior observed state (before the current gap).
- `_mct_after*`: The proportion of time after the current gap in each state (one variable for each category of the state variable).
- `_mct_before*`: The proportion of time before the current gap in each state (one variable for each category of the state variable).
- `_mct_t`: A time index, running from 1 to the length of the time series.

Once `mict_prep` has been run, `mict_impute` carries out the imputation and has the following syntax:

`mict_impute` [`, maxgap(#) maxitgap(#) nimp(#) offset(#)`]

The option `maxgap()` sets the maximum length of the internal gap to impute (the default is `maxgap(12)`). `maxitgap()` sets the maximum length of the initial or terminal

gap to impute (the default is `maxitgap(6)`). `nimp()` specifies the number of imputations (the default is `nimp(5)`). And `offset()` numbers the imputations (the default is `offset(0)` and will number the imputations from 1 to 5); this is useful for parallel runs.

If performing multiple parallel runs, you should set a different random seed for each run; that is, use

```
set seed #
mict_impute ...
```

where # is different for each run. If you want reproducible imputations, you should also set the seed for a single run. See [R] **set seed**.

A minimal run will take the following form:

```
use mvadmar
mict_prep state, id(id)
mict_impute
```

Internally, the `mict_impute` command calls `mict_model_gap`, `mict_model_initial`, and `mict_model_terminal`, which define the imputation models. The models defined by default by these commands are excessively simple and need to be overwritten by the user, as described on page 598. For internal gaps, the default model definition is as follows:

```
program mict_model_gap
   mi impute mlogit _mct_state i._mct_next i._mct_last, add(1) force  ///
      augment iterate(40)
end
```

This declares `mlogit` as the imputation model. If the dependent variable is ordinal, this can be replaced by `ologit` or even `regress`; the user has full access to the power of the built-in `mi impute` at this point. The imputation model uses only the next and the prior state to predict. Of the options, `add(1)` and `force` are necessary, and `augment` is useful (it causes `mi impute mlogit` to use augmented logistic regression in the case of perfect prediction, which is likely to occur at least some of the time when imputing categorical data). The `iterate(40)` option is not necessary, except when trying to trap nonconvergence problems.

The model specification is presented as a user-definable program, to allow more complex code to be run, for instance, defining new variables (for example, recordings of `_mct_next` or `_mct_last`).

We redefine the imputation models in the following manner. The exact model specification will depend on the user; this one simply adds `_mct_before*` and `_mct_after*` (summaries of the proportion of time spent in each state before and after the current gap) to (the required) `_mct_next` and `_mct_last`. Note that the initial and terminal gap models are simplified versions of the internal gap model.

```
capture program drop mict_model_gap
program mict_model_gap
   mi impute mlogit _mct_state   ///
       i._mct_next i._mct_last   ///
       _mct_before* _mct_after*, ///
       add(1) force augment
end

capture program drop mict_model_initial
program mict_model_initial
   mi impute mlogit _mct_state i._mct_next _mct_after*, add(1) force augment
end

capture program drop mict_model_terminal
program mict_model_terminal
   mi impute mlogit _mct_state i._mct_last _mct_before*, add(1) force augment
end
```

4.1 Postimputation use

`mict` generates the imputations in memory, in what `mi impute` would consider `flong` format. To use these data in the standard `mi estimate` framework, you must `mi import` them:

```
mi import flong, id(id) m(_mct_iter) imputed(state*) clear
mi estimate: ...
```

where `id` is the ID variable designated in `mict_prep`'s option `id()`, `state*` refers to the sequence of state variables, and `_mct_iter` is the variable numbering the imputation (created by `_mict_impute` and equivalent to the `_mi_m` created by Stata's imputations).

If you wish to include the unimputed data alongside the imputations, then before doing `mi import`, you should `append` the original dataset to the imputed and set `_mct_iter` to 0 for the nonimputed cases.

5 Demonstrations

To demonstrate how `mict` works and assess its performance, I present some examples:

1. Real data (school-to-work transitions) with simulated longitudinal missingness, allowing comparison between the true state and the imputations.
2. Wholly simulated data using a simple structure, permitting comparison between MICT and MICE.
3. Real data (mothers' labor market histories) with real missingness, using a realistic model.
4. The same data as demonstration three, using an enhanced model that accounts for knowledge about the data-generation process to generate superior imputations.

5.1 Real data with simulated missingness

The first demonstration of the algorithm uses data from McVicar and Anyadike-Danes (2002), which report six years of the life course of Northern Irish young people, starting at the completion of compulsory schooling, in a state space concerned with the transition from school to work (the six states are secondary education, further education, higher education, training, unemployment, and employment). The 712 individuals are observed over 72 months. To demonstrate and assess the performance of the algorithm, we impose missingness at random such that each month has a 1.25% chance of being missing, but with a 66% chance if the previous month is missing. This generates a pattern of runs of missingness, which are MAR with respect to the observed data. The simulated data are stored in wide format (one variable per monthly observation, `state1` to `state72`, and one row per individual).

Redefining default imputation models

We carry out the imputation with the following Stata code:

```
// Load the modified data
use mvadmar
// Prepare it for imputation
mict_prep state, id(id)
// Run the imputation (accepting default options)
mict_impute
```

As described in section 4, the default imputation model is simple, using only the most recent and the nearest future observation as predictors:

```
mi impute mlogit _mct_state i._mct_next i._mct_last, add(1) force augment
```

(Initial and terminal gaps are imputed using only, respectively, subsequent and prior information.) In effect, the model assumes a zero-order Markov process where the transition rates are constant over time and across individuals, and it should normally be overridden by more adequate models, for example, to relax the zero-order assumption, to allow transition rates to change over time, or to incorporate other variables.

Before running the imputations, we can override the built-in models by redefining the commands `mict_model_gap`, `mict_model_initial`, and `mict_model_terminal`, as follows:

```
capture program drop mict_model_gap
program mict_model_gap
   mi impute mlogit _mct_state                        ///
       i._mct_next##c._mct_t i._mct_last##c._mct_t    ///
       _mct_before* _mct_after*,                      ///
       add(1) force augment
end

capture program drop mict_model_initial
program mict_model_initial
   mi impute mlogit _mct_state i._mct_next _mct_after*, add(1) force augment
end

capture program drop mict_model_terminal
program mict_model_terminal
   mi impute mlogit _mct_state i._mct_last _mct_before, add(1) force augment
end
```

Variables `_mct_before1` to `_mct_before`C and `_mct_after1` to `_mct_after`C (where C is the number of categories) are created by `mict_prep` and represent the proportion of time before and after the gap spent in each of the C categories of the state variable. Thus, these variables offer a means of including in the model some history prior to and after the nearest observed time units.

The interactions `i._mct_next##c._mct_t` and `i._mct_last##c._mct_t` allow the effects of the prior and the next state to vary in a linear fashion with time, relaxing the assumption that transition rates are constant. Depending on the data, it may be desirable to relax this constraint even further, perhaps with time as a quadratic. Other variables may also be entered, including fixed individual variables (such as gender or social class) and variables indicating time-dependent state in another domain. The imputation model must contain all the variables needed to satisfy the MAR assumption.

The options `add(1)`, `force`, and `augment` to `mi impute mlogit` are required, to make a single imputation, to force imputation even where the predictor variables are not fully observed, and to use augmented multinomial regression if perfect prediction is encountered. Other options may be used as appropriate.

Simulated missing on McVicar and Anyadike-Danes (2002) data: Some results

Using this model, we generate 10 imputations. Figure 1 shows four typical cases, with the fully observed data, the data with random runs of missingness imposed, and the 10 imputations shown as horizontal lines.[2] These display features that are typical of the full dataset. For example, short gaps with the same state before and after tend to get filled in with that state, which is often correct (for example, case 33 and the first two gaps in case 62). When longer, such gaps show a quite small tendency to have other states interpolated (imputation 10 for case 85 shows an example). Where a gap has two different states as neighbors, the majority of imputations show a single transition from one to the other, where the timing of the transition is randomly distributed in the gap

2. The figure, an indexplot, is created using the `sqindexplot` command from the `sq` package (Kohler and Brzinsky-Fay 2005; Brzinsky-Fay, Kohler, and Luniak 2006).

(case 13). Such gaps show a larger tendency to interpolate one or more other states, particularly as the gaps get longer (more so than gaps bracketed by a single state). However, if the run of missingness completely obliterates a spell in the original data (which is something we can only know because we have the fully observed data; see the third gap in case 62), the imputations are very unlikely to pick up that state (though in this case, one of the imputations does create such a state, if somewhat different in timing and duration).

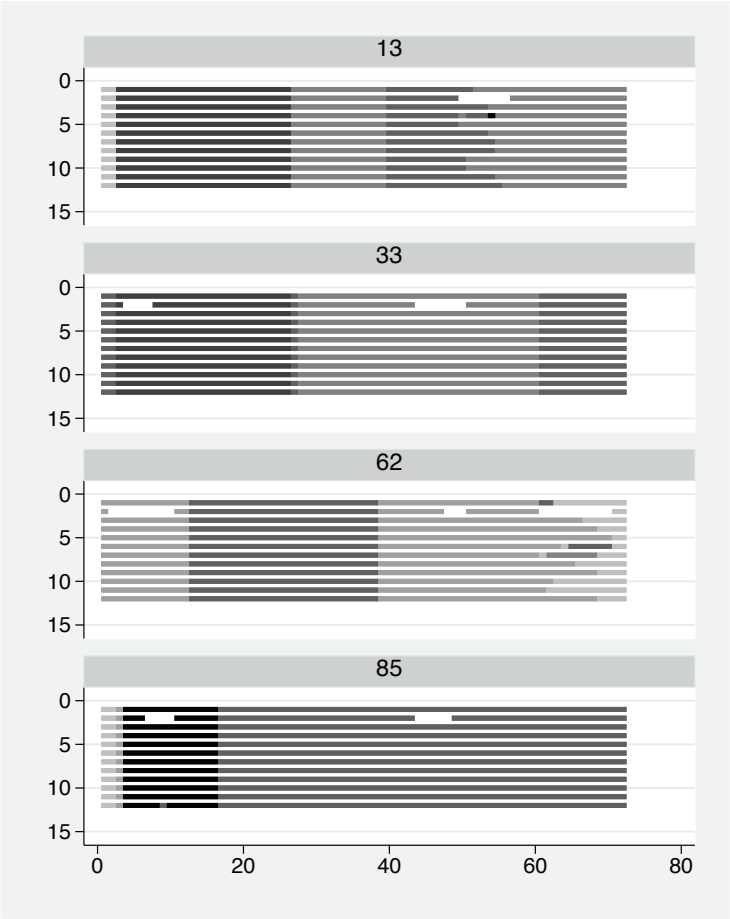

Figure 1. Sample imputations for four typical cases, with horizontal lines representing the fully observed data (first line), data with imposed missings (second line, missing is white), and 10 imputations

Thus, we see how to use MICT to program imputations by using a simple but more-or-less realistic predictive model. We also see that, in general, the imputations approximate the true data quite well because there is a lot of redundancy in data like these. However, where a gap envelopes a complete spell, the redundancy is markedly less.

5.2 Simulated data with simulated missingness

It is difficult to compare the performance of the gap-filling approach with MICE, either via the official `mi impute` or via Royston's `ice`, because it is difficult to specify analogous models. To facilitate comparison, I present a simulation that permits much simpler models. A zero-order Markov process with time-constant transition rates is used to create sequences that are 36 elements long and with four states, with random gaps imposed. Because the generating process is zero-order, only adjacent last and next observations carry information with which to impute. Thus, for the MICT approach, the only meaningful imputation model uses just `_mct_last` and `_mct_next` as predictors, while for MICE, only the immediately adjacent states, s_{t-1} and s_{t+1}, are used. Two thousand sequences are generated, and for each method—MICT, `mi impute chained`, and `ice`—10 imputations are made.

In what follows, I use both the `mi impute chained` and the `ice` implementations of MICE. In this more conventional framework, chained imputation takes care of the fact that, given the data in a wide format, missingness is nonmonotonic (that is, that predictors of missing values may well themselves be missing).

For `ice`, the models are defined as follows:

```
ice m.m1 m.m2 m.m3 m.m4 m.m5 m.m6 m.m7 m.m8 m.m9 m.m10          ///
    m.m11 m.m12 m.m13 m.m14 m.m15 m.m16 m.m17 m.m18 m.m19 m.m20 ///
    m.m21 m.m22 m.m23 m.m24 m.m25 m.m26 m.m27 m.m28 m.m29 m.m30 ///
    m.m31 m.m32 m.m33 m.m34 m.m35 m.m36,                        ///
  saving(puresim_ice_cycles, replace) persist m(10) cycles(10)  ///
  eq(m1:    i.m2      ,                                         ///
     m36:  i.m35      ,                                         ///
     m2:    i.m1  i.m3,                                         ///
     m3:    i.m2  i.m4,                                         ///
(code omitted)
     m35:  i.m34 i.m36)
```

For `mi impute chained`, the following, rather verbose, code is used (note omissions):

```
mi set flong
mi register imputed m*
mi impute chained                                                    ///
  (mlogit, omit(          i.m3 i.m4   [...]  i.m34 i.m35 i.m36 ))  m1   ///
  (mlogit, omit(               i.m4   [...]  i.m34 i.m35 i.m36 ))  m2   ///
  (mlogit, omit(i.m1                  [...]  i.m34 i.m35 i.m36 ))  m3   ///
  (mlogit, omit(i.m1 i.m2             [...]  i.m34 i.m35 i.m36 ))  m4   ///
  (mlogit, omit(i.m1 i.m2 i.m3        [...]  i.m34 i.m35 i.m36 ))  m5   ///
 [...]
  (mlogit, omit(i.m1 i.m2 i.m3 i.m4   [...]                    ))  m35  ///
  (mlogit, omit(i.m1 i.m2 i.m3 i.m4   [...]  i.m34             ))  m36, ///
  add(10) force augment
```

For MICT, the Markov process generating the data is captured fully by the default imputation models, so it is enough to do the following:

```
mict_prep m, id(id)
mict_impute, maxgap(21) maxitgap(14) nimp(10)
```

The `maxgap()` and `maxitgap()` options need to be set, because the maximum gap lengths in the simulated data are longer than the defaults.

Given the way the data are simulated, each of the three imputations has the best possible model in its framework.

Figure 2 shows some example imputations for a handful of cases across the three strategies. As can be seen, `mi impute` and `ice` show somewhat more transitions in the imputed sections. To test whether this is a systematic feature, we compare the number of spells in the imputed sequences with the true number (in the simulated data, before imposition of missingness). We can use `mi estimate` to test the null that the difference is 0 by running a null regression and looking at the t statistic for `_cons` (this is a way of getting `mi estimate` to run a t test).

Method	`_cons`	Std. err.	t	p
MICT	0.01025	0.0253	0.40	0.687
`mi impute`	0.2878	0.0437	6.59	0.000
`ice`	0.31645	0.0347	9.12	0.000

Over the 10 imputations of 2,000 sequences, MICT does not significantly increase the mean number of spells, while `mi impute` and `ice` do (by 0.29 and 0.32, respectively). Thus, in this simple comparison, MICT retains longitudinal consistency whereas MICE does not.[3] Analysis using `twofold` (not presented) shows that it behaves in a similar manner to the two MICE implementations.[4]

3. Experiments with increasing the number of cycles in the MICE chains were attempted, to see if greater consistency could be achieved with a longer burn-in, but there was no tendency to a systematic improvement.
4. The main, and substantial, advantage of `twofold` in this context is that it makes it easy to express a good imputation model for longitudinal data in the MICE framework, unlike the unwieldy examples above.

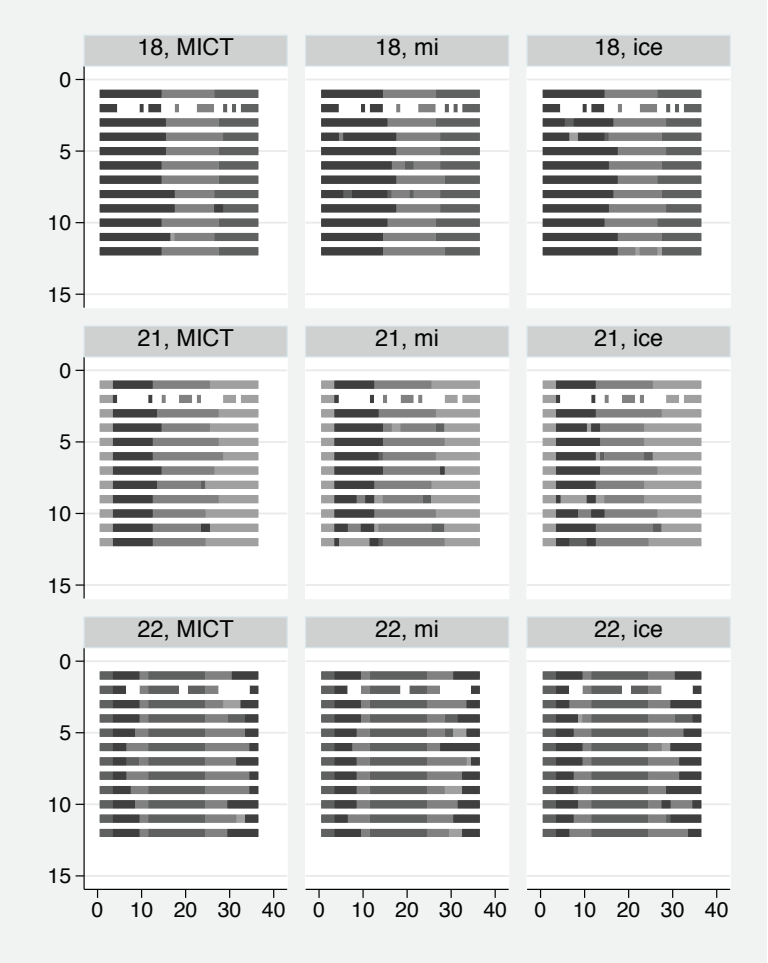

Figure 2. Three example sequences of imputations by MICT, `mi impute chained`, and `ice`

5.3 Mothers' labor market histories

Let us now consider how MICT performs with real missingness. Using data drawn from the British Household Panel Survey (BHPS, a large annual household panel study, collected in the UK from 1991 to 2008), I created six-year monthly employment status histories for women who have a birth at the end of the second year (Taylor et al. 2010; Halpin 1998). The state space consists of full-time and part-time employment, unemployment, and nonemployment. This dataset contains 706 fully observed sequences, 190 with gaps of up to 12 months, and 425 with longer gaps. I choose to impute gaps of up to 12 months but use data from sequences with longer gaps to provide information for

the imputation models. See figure 3, which shows the overall picture of a retreat from paid work as the birth approaches, and a qualified return afterward, predominantly to part-time work.

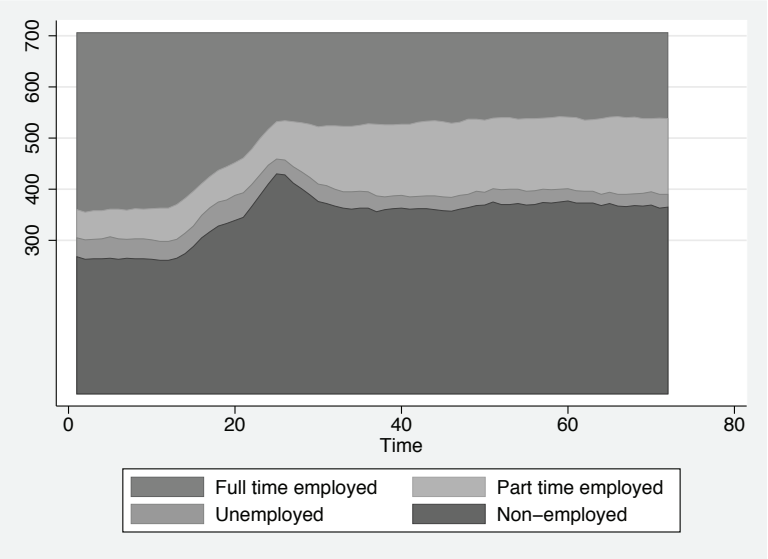

Figure 3. State distribution of fully observed mothers' labor market sequences

We use the following predictive model for imputing the internal gaps (analogous models are used for initial and terminal gaps):

```
capture program drop mict_model_gap
program mict_model_gap
   mi impute mlogit _mct_state                                          ///
     i._mct_next##c._mct_t##c._mct_t i._mct_last##c._mct_t##c._mct_t ///
     _mct_before* _mct_after*
end
```

This is a relatively simple model, implying that transition patterns vary in a nonlinear pattern, and using the prior and subsequent cumulative distributions of states. That is, the effects of the last and next observed states vary in interaction with a quadratic time term, and the `_mct_before*` and `_mct_after*` terms represent the proportion of time spent in the various states before and after the gap.

The following code runs the whole example, with the `maxgap(12)` and `maxitgap(6)` options to `mict_impute` limiting the imputations to cases with maximum internal gap lengths of 12 and initial/terminal gaps of 6 months. The `nimp(10)` option causes it to generate 10 imputations.

```
mict_prep state, id(pid)

capture program drop mict_model_gap
program mict_model_gap
   mi impute mlogit _mct_state i._mct_next##c._mct_t##c._mct_t ///
      i._mct_last##c._mct_t##c._mct_t                          ///
      _mct_before* _mct_after*,                                ///
      add(1) force augment noisily iterate(40)
end

capture program drop mict_model_initial
program mict_model_initial
   mi impute mlogit _mct_state i._mct_next##c._mct_t _mct_after*, ///
         add(1) force augment iterate(40)
end

capture program drop mict_model_terminal
program mict_model_terminal
   mi impute mlogit _mct_state i._mct_last##c._mct_t _mct_before*, ///
         add(1) force augment iterate(40)
end

mict_impute, maxgap(12) maxitgap(6) nimp(10)
```

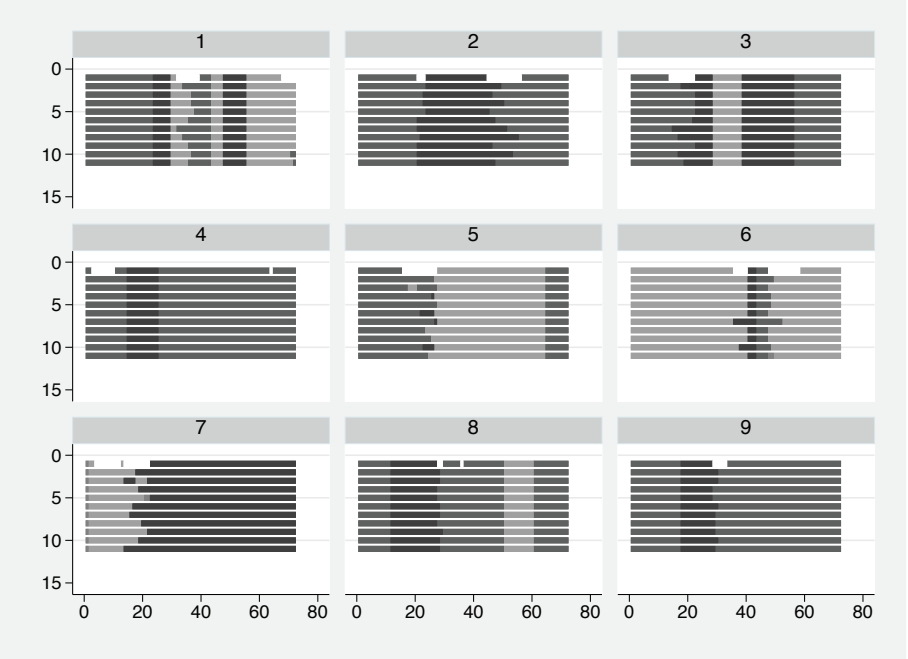

Figure 4. Imputations for selected cases, mothers' labor market history, first model

We can examine the performance of the imputation model in figure 4. In outline, the performance is similar to the simulations: gaps bracketed by a single state tend to be filled in by that state, with a certain amount of interpolation of other states that

increases as the gap length increases; gaps between different states are often filled by the two states with a distribution of transition points but are also quite likely to feature extra transitions and third states, again more so for longer gaps.

Which sequences get imputed?

In real-life data, there tends to be a strong relationship between the nature of a life course sequence and its probability of having gaps. In particular, volatile life courses are more subject to interruptions in data collection, and in being complex are more vulnerable to missed measurement opportunities. We can see this by looking at indicators of complexity, such as the number of spells, or the Shannon entropy.[5]

```
. table gap, c(n ent mean ent mean nsp) format(%5.2f)
```

gap	N(ent)	mean(ent)	mean(nsp)
0	7,060	0.53	2.39
1	1,900	1.03	3.80

Fully observed sequences have a mean of 2.39 spells, compared with 3.80 for sequences that have been imputed. Similarly, fully observed sequences have a mean entropy of 0.53 compared with 1.03 for imputed. The number of spells tracks how transition-prone sequences are, and the entropy additionally indicates how diverse the states visited are. On average, gappy sequences that are imputed have more transitions and are more diverse.[6] This is important because complex sequences are often substantively interesting and their exclusion reduces the useful information in the dataset, as well as potentially introducing bias.

5.4 Refining models

The model used above is relatively simple. Various ways to improve its quality as an imputation model are available, most readily the inclusion of fixed individual-level variables and the incorporation of more interactions. Fully observed time-dependent variables in another domain may also be included, such as residential or marital histories. Where that variable is also subject to missing, a simple imputation strategy such as carry forward may be adequate.[7]

5. Commands to calculate the number of spells and the Shannon entropy are available in the `sadi` package (Halpin 2014).
6. This is not an artifact of the imputation process. If instead we mechanically carry forward the previous state to fill gaps, thus not increasing the number of spells or the diversity, we get very similar results.
7. In the typical case where data in the other domain are missing at the same time as the main variable, MICT could in principle draw information in that domain from the prior and subsequent time points determined by the current gap length, but that would require updating (imputing) the observations for the other domain as the process fills in the gaps. This would add another layer of complexity to the package, so simpler imputations are preferred.

To add a completely observed (or filled-in) time series in another domain, add that variable name (as a `reshape`-style stub) to the `mict_prep` statement. For example, if the variable to be imputed is `state1` to `state`*N* and the second variable is `resid1` to `resid`*N*, the command would take this form:

```
mict_prep state resid, id(pid)
```

The `resid` variable is then available for use in the model statement. More than one extra time-series variable can be added in this fashion.

Picking up the missingness process

In what follows, I demonstrate the inclusion of an extra time-dependent variable, and address a substantive issue raised above, by the simulations. It was observed that in general there tends to be a high degree of redundancy in gappy life course data like these, such that imputations resemble the observed data well except where the gap overlaps a complete spell. If this happens at random, it is not a serious problem for the imputation/estimation process; but if the occurrence of gaps is somehow associated with the spell structure of the history represented by the data, then the imputations will understate the true variability. Depending on the domain, and on how data are collected, this is quite likely to occur.

The mothers' labor market histories are drawn from the BHPS, whose data collection is annual, with retrospective accounts covering the period between the interview and the start date of the last year's fieldwork (Halpin 1998). This assures continuity if no interviews are missed and the retrospective accounts are without measurement error (assumptions that are frequently violated). In the BHPS work-life histories, gaps may occur because of omitted spells, spells whose start or end date is misreported, or missed data-collection points. In the first case, the gap may be exactly coterminous with the spell; in the second, the gap will start at the beginning of the spell if the misreported date is late; and in the third, the gap will begin just after the previous data-collection point and end at the start of the account at the next data-collection point. This means that the process of missingness is quite structured.

We have data relating to this process on which we can draw. If the previously observed state was reported as a spell end in the retrospective account, we know there is greater probability that the current state is different because a transition is indicated (although it could be a transition to the same state); likewise if the next observed state is reported as a spell start. If the previous observation was the date of an interview, then the current state is more likely to be the same as that at the interview because no transition is reported. However, in the observed data, there is a distinct pattern of seam effects, where the account from the following interview clashes with the prior data, often by backdating a spell start such that it precedes the earlier interview. In the data used, this is represented as a transition immediately after the interview, on the logic that the current state reported at time $(t-1)$ is more authoritative than the retrospective report from time t (Halpin 1998). Thus, whether months of interview tend to be followed by elevated transition risk is an empirical question.

To account for these effects, a variable is created that distinguishes between "neutral" months, explicitly reported spell starts from the interwave job history (where a start may be a reported start from a spell with missing state information but valid dates, or one month after the end of a fully reported spell), explicitly reported spell starts of the spell current at interview, and months in which the annual interview fell. Where no information is available, it is allowed to default to neutral. This variable, `obstype`, is incorporated in the imputation model as follows:

```
program mict_model_gap
   capture drop _mct_n2 _mct_l2
   recode _mct_last 3=2, generate(_mct_l2)
   recode _mct_next 3=2, generate(_mct_n2)
   mi impute mlogit _mct_state i._mct_next##c._mct_t##c._mct_t  ///
                           i._mct_last##c._mct_t##c._mct_t     ///
                           _mct_before* _mct_after*            ///
                           i.obstype##i._mct_n2                ///
                           i.obstype##i._mct_l2,               ///
                           add(1) force augment
end
```

It is modeled in interaction with the next and prior states because its effect is via the transition pattern. However, in these data, there are relatively few transitions to and from unemployment, which causes problems in estimation. Therefore, `obstype` is interacted with recoded versions of these variables, which are re-created each iteration.[8]

Sample imputations are shown in figure 5 for the models with and without the observation structure variable. In each panel, the first row indicates the structure (black is the month of interview, dark gray is a spell start in the interwave job history, and light gray is the spell start current at the interview). As can be seen, taking account of the meta-information regarding spell structure has a systematic effect: first, the imputations are more likely to contain transitions at the key points, and second, they are more likely to interpolate spells in third states. Both of these are desirable because the model without the meta-information understates not only the transition rates around these key points but also the variability of the imputations.

8. In general, creation of any sort of transformed variable can be carried out within the `mict_model_gap`, `mict_model_initial`, and `mict_model_terminal` commands.

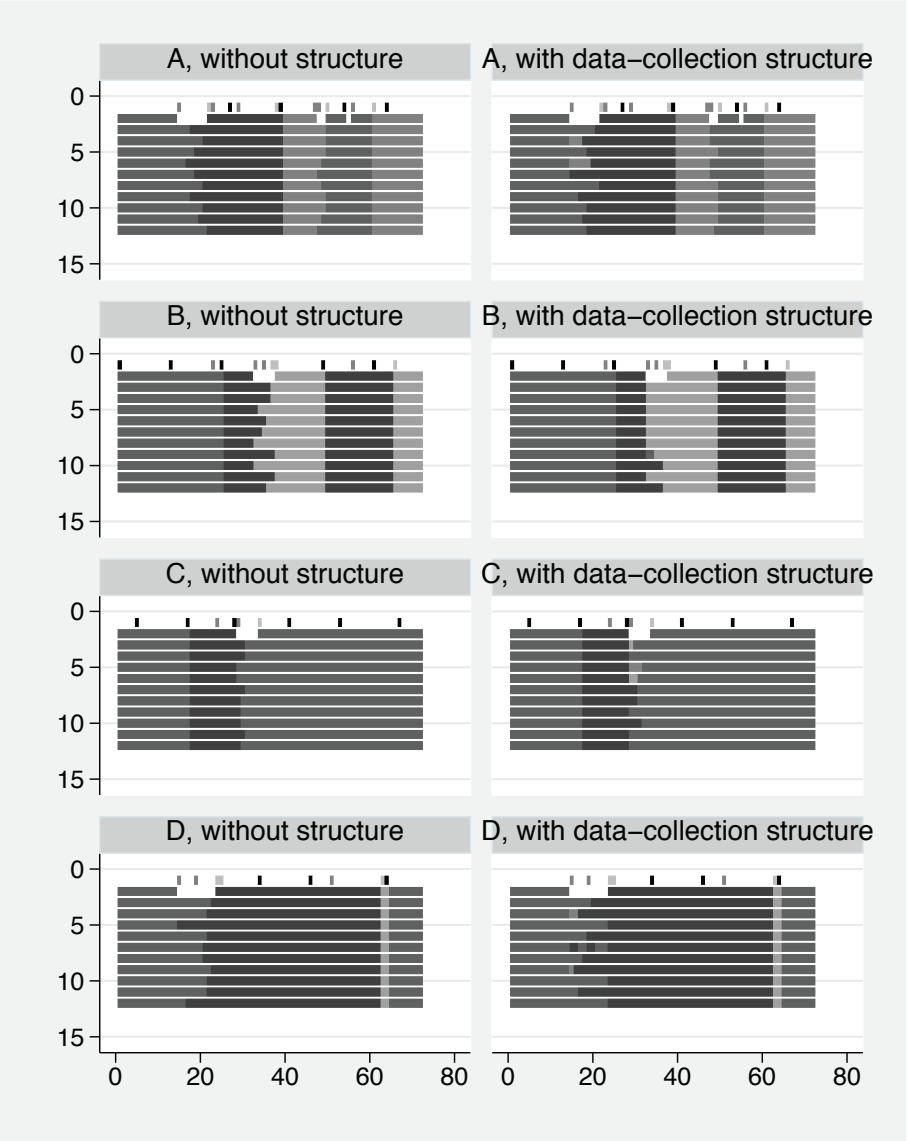

Figure 5. Selected imputations without (left) and with (right) data-collection information

This illustration is guided by the particular data-collection structure of the BHPS, but it is likely that many longitudinal datasets will contain meta-information that could inform the imputation model in an analogous way.

6 Conclusion

`mict` offers a flexible and longitudinally consistent means of multiply imputing categorical time-series data, particularly when it is characterized by relatively long spells in states and consecutive runs of missingness, as is typically the case in life course data. As the second simulation (section 5.2) shows, `mict` produces imputations with largely plausible patterns of transitions, while MICE (via either `mi impute chained` or `ice`) generates unrealistically elevated rates of transition.

The key advantage is that `mict` structures the imputations as a series of monotonic imputations, focusing on filling gaps. This allows a good deal of flexibility, including the use of binary, multinomial, or ordinal logistic regression models as appropriate. `mict` offers a reasonably user-friendly interface to defining models for `mi impute`, such that it is relatively easy to define good imputation models incorporating fixed and time-dependent effects. It produces imputations that can be used by the `mi impute` postimputation infrastructure.

`mict` does have some disadvantages, not least that it presents yet another interface to imputation, a compatible but separate solution to `mi impute`. It can deal with only a single target variable for imputation. It does not update created variables such as `_mct_before*`, nor does it update other time-series variables that may be used in the model (thus effectively falling back on carry-forward imputation for these variables). Extending the package to cope with these would be complicated. However, it presents a relatively lightweight and effective solution for imputing single categorical time-series variables.

6.1 Existing applications

This approach has already been used in practice (based on the functionally equivalent but less user-friendly approach described in Halpin [2012, 2013]). Fuller and Stecy-Hildebrandt (2015) apply it to the Canadian Survey of Labour and Income Dynamics, and McMunn et al. (2015) and Lacey et al. (2016) apply it to the British cohort study datasets (the MRC National Survey of Health and Development 1946 birth cohort, the National Child Development Study 1958 birth cohort, and the British Cohort Study 1970 birth cohort).

7 References

Allison, P. D. 2005. Imputation of categorical variables with PROC MI. SAS Users Group International Proceedings. http://www2.sas.com/proceedings/sugi30/113-30.pdf.

———. 2009. Missing data. In *The SAGE Handbook of Quantitative Methods in Psychology*, ed. R. E. Millsap and A. Maydeu-Olivares, 72–89. London: Sage.

Brzinsky-Fay, C., U. Kohler, and M. Luniak. 2006. Sequence analysis with Stata. *Stata Journal* 6: 435–460.

Fuller, S., and N. Stecy-Hildebrandt. 2015. Career pathways for temporary workers: Exploring heterogeneous mobility dynamics with sequence analysis. *Social Science Research* 50: 76–99.

Halpin, B. 1998. Unified BHPS work-life histories: Combining multiple sources into a user-friendly format. *Bulletin of Sociological Methodology* 60: 34–79.

———. 2012. Multiple imputation for life-course sequence data. Working Paper WP2012-01, Department of Sociology, University of Limerick. http://hdl.handle.net/10344/3639.

———. 2013. Imputing sequence data: Extensions to initial and terminal gaps, Stata's mi. Working Paper WP2013-01, Department of Sociology, University of Limerick. http://hdl.handle.net/10344/3620.

———. 2014. SADI: Sequence analysis tools for Stata. Working Paper WP2014-03, Department of Sociology, University of Limerick. http://hdl.handle.net/10344/3783.

Honaker, J., and G. King. 2010. What to do about missing values in time-series cross-section data. *American Journal of Political Science* 54: 561–581.

Kohler, U., and C. Brzinsky-Fay. 2005. Stata tip 25: Sequence index plots. *Stata Journal* 5: 601–602.

Lacey, R., M. Stafford, A. Sacker, and A. McMunn. 2016. Work-family life courses and subjective wellbeing in the MRC National Survey of Health and Development (the 1946 British birth cohort study). *Journal of Population Ageing* 9: 69–89.

McMunn, A., R. Lacey, D. Worts, P. McDonough, M. Stafford, C. Booker, M. Kumari, and A. Sacker. 2015. De-standardization and gender convergence in work–family life courses in Great Britain: A multi-channel sequence analysis. *Advances in Life Course Research* 26: 60–75.

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

Nevalainen, J., M. G. Kenward, and S. M. Virtanen. 2009. Missing values in longitudinal dietary data: A multiple imputation approach based on a fully conditional specification. *Statistics in Medicine* 28: 3657–3669.

Royston, P. 2004. Multiple imputation of missing values. *Stata Journal* 4: 227–241.

———. 2009. Multiple imputation of missing values: Further update of ice, with an emphasis on categorical variables. *Stata Journal* 9: 466–477.

Rubin, D. B. 1987. *Multiple Imputation for Nonresponse in Surveys*. New York: Wiley.

Taylor, M., J. Brice, N. Buck, and E. Prentice-Lane, eds. 2010. *British Household Panel Survey User Manual*. Colchester: University of Essex.

van Buuren, S. 2007. Multiple imputation of discrete and continuous data by fully conditional specification. *Statistical Methods in Medical Research* 16: 219–242.

van Buuren, S., H. C. Boshuizen, and D. L. Knook. 1999. Multiple imputation of missing blood pressure covariates in survival analysis. *Statistics in Medicine* 18: 681–694.

van Buuren, S., and K. Groothuis-Oudshoorn. 2011. mice: Multivariate imputation by chained equations in R. *Journal of Statistical Software* 45(3): 1–67.

Welch, C., J. Bartlett, and I. Petersen. 2014. Application of multiple imputation using the two-fold fully conditional specification algorithm in longitudinal clinical data. *Stata Journal* 14: 418–431.

About the author

Brendan Halpin is a senior lecturer and head of the Department of Sociology at the University of Limerick in Ireland and has a longstanding interest in longitudinal social science data.