Amelia II: A Program for Missing Data

James Honaker
The Pennsylvania State University

Gary King
Harvard University

Matthew Blackwell
Harvard University

Abstract

Amelia II is a complete R package for multiple imputation of missing data. The package implements a new expectation-maximization with bootstrapping algorithm that works faster, with larger numbers of variables, and is far easier to use, than various Markov chain Monte Carlo approaches, but gives essentially the same answers. The program also improves imputation models by allowing researchers to put Bayesian priors on individual cell values, thereby including a great deal of potentially valuable and extensive information. It also includes features to accurately impute cross-sectional datasets, individual time series, or sets of time series for different cross-sections. A full set of graphical diagnostics are also available. The program is easy to use, and the simplicity of the algorithm makes it far more robust; both a simple command line and extensive graphical user interface are included.

Keywords: missing data, multiple imputation, bootstrap, R.

1. Introduction

Missing data is a ubiquitous problem in social science data. Respondents do not answer every question, countries do not collect statistics every year, archives are incomplete, subjects drop out of panels. Most statistical analysis methods, however, assume the absence of missing data, and are only able to include observations for which every variable is measured.

Amelia II performs multiple imputation, a general-purpose approach to data with missing values. This method creates multiple “filled in” or rectangularized versions of the incomplete data set so that analyses which require complete observations can appropriately use all the information present in a data set with missingness. Multiple imputation has been shown to reduce bias and increase efficiency compared to listwise deletion. Furthermore, ad-hoc methods of imputation, such as mean imputation, can lead to serious biases in variances and covariances. Unfortunately, creating multiple imputations can be a burdensome process due
to the technical nature of algorithms involved. Amelia II provides users with a simple way
to create and implement an imputation model, generate imputed datasets, and check its fit
using diagnostics.

Amelia II draws imputations of the missing values using a novel bootstrapping approach,
the EMB (expectation-maximization with bootstrapping) algorithm. The algorithm uses the
familiar EM (expectation-maximization) algorithm on multiple bootstrapped samples of the
original incomplete data to draw values of the complete-data parameters. The algorithm then
draws imputed values from each set of boostrapped parameters, replacing the missing values
with these draws.

The Amelia II program goes several significant steps beyond the capabilities of the first version
of Amelia (Honaker, Joseph, King, Scheve, and Singh 2002). For one, the bootstrap-based
EMB algorithm included in Amelia II can impute many more variables, with many more
observations, in much less time. The great simplicity and power of the EMB algorithm made
it possible to write Amelia II so that it virtually never crashes – which to our knowledge
makes it unique among all existing multiple imputation software – and is much faster than
the alternatives too. While creative applications of bootstrapping have been developed for
several application-specific missing data problems (Rubin and Schenker 1986; Rubin 1994;
Efron 1994; Shao and Sitter 1996; Lahiri 2003), but to our knowledge the technique has
not been used to develop and implement a general purpose multiple imputation algorithm.
Amelia II also has features to make valid and more accurate imputations for cross-sectional,
time-series, and time-series-cross-section data, and allows the incorporation of observation and
data-matrix-cell level prior information. The cell-level priors help users incorporate expert
knowledge about specific missing values, a task that is infeasible using other methods. In
addition to this, Amelia II provides many diagnostic functions that help users check the
validity of their imputation model. This software implements the ideas developed in Honaker
and King (2010).

2. What Amelia does

Multiple imputation involves imputing \( m \) values for each missing cell in a data matrix and
creating \( m \) “completed” data sets. Across these completed data sets, the observed values are
the same, but the missing values are filled in with a sample of values from the predictive
distribution of missing data. After imputation with the EMB algorithm of Amelia\(^1\), the user
can apply whatever statistical method he or she would have used if there had been no missing
values to each of the \( m \) data sets, and use a simple procedure, described below, to combine
the results.\(^2\) Under normal circumstances, you only need to run multiple imputation once
and can then analyze the \( m \) imputed data sets as many times and for as many purposes as
you wish. The advantage of Amelia is that it combines the comparative speed and ease-of-use
of our algorithm with the power of multiple imputation, to let you focus on your substantive
research questions rather than spending time developing complex application-specific models
for nonresponse in each new data set. Unless the rate of missingness is very high, \( m = 5 \) (the
program default) is probably adequate (Rubin 1987; Schafer 1997).

\(^1\)For simplicity, we refer to the package as Amelia for the rest of this document.
\(^2\)You can combine the results automatically by doing your data analyses within Zelig (Imai, King, and Lau
2008) for R (R Development Core Team 2011), or within Clarify (King, Tomz, and Wittenberg 2000) for Stata
(StataCorp. 2011).
2.1. Assumptions

The imputation model in Amelia assumes that the complete data (that is, both observed and unobserved) are multivariate normal. If we denote the \((n \times k)\) dataset as \(D\) (with observed part \(D^{\text{obs}}\) and unobserved part \(D^{\text{mis}}\)), then this assumption is

\[
D \sim N_k(\mu, \Sigma),
\]

which states that \(D\) has a multivariate normal distribution with mean vector \(\mu\) and covariance matrix \(\Sigma\). The multivariate normal distribution is often a crude approximation to the true distribution of the data, yet there is evidence that this model works as well as other, more complicated models even in the face of categorical or mixed data (see Schafer 1997; Schafer and Olsen 1998). Furthermore, transformations of many types of variables can often make this normality assumption more plausible (see Section 4.3 for more information on how to implement this in Amelia).

The essential problem of imputation is that we only observe \(D^{\text{obs}}\), not the entirety of \(D\). Amelia assumes, as most multiple imputation methods do, that the data are missing at random (MAR). This assumption means that the pattern of missingness only depends on the observed data \(D^{\text{obs}}\), not the unobserved data \(D^{\text{mis}}\). Let \(M\) to be the missingness matrix, with cells \(m_{ij} = 1\) if \(d_{ij} \in D^{\text{mis}}\) and \(m_{ij} = 0\) otherwise. Put simply, \(M\) is a matrix that indicates whether or not a cell is missing in the data. With this, we can define the MAR assumption as

\[
p(M|D) = p(M|D^{\text{obs}}).
\]

Note that MAR includes the case when missing values are created randomly by, say, coin flips, but it also includes many more sophisticated missingness models. When missingness is not dependent on the data at all, we say that the data are missing completely at random (MCAR). Amelia requires both the multivariate normality and the MAR assumption (or the simpler special case of MCAR). Note that the MAR assumption can be made more plausible by including additional variables in the dataset \(D\) in the imputation dataset than just those eventually envisioned to be used in the analysis model. This auxiliary information is useful when it either helps predict the value of the missing data or when missingness is likely to occur.

2.2. Algorithm

In multiple imputation, we are concerned with the complete-data parameters, \(\theta = (\mu, \Sigma)\). Note that the observed data is actually \(D^{\text{obs}}\) and \(M\), the missingness matrix. Thus, the likelihood of our observed data is \(p(D^{\text{obs}}, M|\theta)\). Using the MAR assumption\(^3\), we can break this up,

\[
p(D^{\text{obs}}, M|\theta) = p(M|D^{\text{obs}})p(D^{\text{obs}}|\theta).
\]

As we only care about inference on the complete data parameters, we can write the likelihood as

\[
L(\theta|D^{\text{obs}}) \propto p(D^{\text{obs}}|\theta),
\]

\(^3\)There is an additional assumption hidden here that \(M\) does not depend on the complete-data parameters.
which we can rewrite using the law of iterated expectations as
\[
p(D_{obs}|\theta) = \int p(D|\theta)dD_{mis}. \tag{5}
\]

With this likelihood and a flat prior on \(\theta\), we can see that the posterior is
\[
p(\theta|D_{obs}) \propto p(D_{obs}|\theta) = \int p(D|\theta)dD_{mis}. \tag{6}
\]

The main computational difficulty in the analysis of incomplete data is taking draws from this posterior. The EM algorithm (Dempster, Laird, and Rubin 1977) is a simple computational approach to finding the mode of the posterior. Our EMB algorithm combines the classic EM algorithm with a bootstrap approach to take draws from this posterior. For each draw, we bootstrap the data to simulate estimation uncertainty and then run the EM algorithm to find the mode of the posterior for the bootstrapped data, which gives us fundamental uncertainty too (see Honaker and King (2010) for details of the EMB algorithm).

Once we have draws of the posterior of the complete-data parameters, we create imputations by drawing values of \(D_{mis}\) from its distribution conditional on \(D_{obs}\) and the draws of \(\theta\).

2.3. Analysis

In order to combine the results across \(m\) data sets, first decide on the quantity of interest, \(q\), to compute, such as a univariate mean, regression coefficient, predicted probability, or first difference. Then, on each of the \(m\) imputed data sets, run the complete data analysis model—that is, the analysis model that would be appropriate if there had been no missing data. The easiest way to combine the results from each of these models is to draw \(1/m\) simulations of \(q\) from each of the \(m\) models, combine them into one set of \(m\) simulations, and then to use the standard simulation-based methods of interpretation common for single data sets (King et al. 2000).
Alternatively, you can combine the model results directly and use as the multiple imputation estimate of this parameter, $\bar{q}$, the average of the $m$ separate estimates, $q_j$ ($j = 1, \ldots, m$):

$$\bar{q} = \frac{1}{m} \sum_{j=1}^{m} q_j. \tag{7}$$

The variance of the point estimate is the average of the estimated variances from within each completed data set, plus the sample variance in the point estimates across the data sets (multiplied by a factor that corrects for the bias because $m < \infty$). Let $SE(q_j)^2$ denote the estimated variance (squared standard error) of $q_j$ from the data set $j$, and $S_q^2 = \Sigma_{j=1}^{m} (q_j - \bar{q})^2/(m - 1)$ be the sample variance across the $m$ point estimates. The standard error of the multiple imputation point estimate is the square root of

$$SE(q)^2 = \frac{1}{m} \sum_{j=1}^{m} SE(q_j)^2 + S_q^2 (1 + 1/m). \tag{8}$$

### 3. Versions of Amelia

Two versions of Amelia are available, each with its own advantages and drawbacks, but both of which use the same underlying code and algorithms. First, Amelia exists as a package for the R statistical software package (R Development Core Team 2011). Users can utilize their knowledge of the R language to run Amelia at the command line or to create scripts that will run Amelia and preserve the commands for future use. Alternatively, you may prefer AmeliaView, where an interactive Graphical User Interface (GUI) allows you to set options and run Amelia without any knowledge of the R programming language.

Both versions of Amelia are available on the Windows, Mac OS X, and Linux platforms and Amelia for R runs in any environment that R can. All versions of Amelia require the R software (version 2.10.0 or higher), which is freely available at http://www.R-project.org/.

#### 3.1. Installation and updates from R

To install the Amelia package on any platform, simply type the following at the R command prompt,

```r
R> install.packages("Amelia")
```

and R will automatically install the package to your system from CRAN (Comprehensive R Archive Network) at http://CRAN.R-project.org/package=Amelia. If you wish to use the most current beta version of Amelia, feel free to install the test version,

```r
R> install.packages("Amelia", repos = "http://r.iq.harvard.edu", + type = "source")
```

In order to keep your copy of Amelia completely up to date, you should use the command

```r
update.packages()
```
3.2. Installation in Windows of AmeliaView as a standalone program

To install a standalone version of AmeliaView in the Windows environment, simply download the installer setup.exe from http://gking.harvard.edu/amelia/ and run it. The installer will ask you to choose a location to install Amelia. If you have installed R with the default options, Amelia will automatically find the location of R. If the installer cannot find R, it will ask you to locate the directory of the most current version of R. Make sure you choose the directory name that includes the version number of R (for example, C:/Program Files/R/R-2.9.0) and contains a subdirectory named bin. The installer will also put shortcuts on your Desktop and Start Menu.

Even users familiar with the R language may find it useful to utilize AmeliaView to set options on variables, change arguments, or run diagnostics. From the command line, AmeliaView can be brought up with the call:

```r
R> library("Amelia")
R> AmeliaView()
```

4. A user’s guide

4.1. Data and initial results

We now demonstrate how to use Amelia using data from Milner and Kubota (2005) which studies the effect of democracy on trade policy. For the purposes of this user’s guide, we will use a subset restricted to nine developing countries in Asia from 1980 to 1999. This dataset includes 9 variables: year (year), country (country), average tariff rates (tariff), Polity IV score (polity), total population (pop), gross domestic product (GDP) per capita (gdp.pc), gross international reserves (intresmi), a dummy variable signifying whether the country had signed an IMF (International Monetary Fund) agreement in that year (signed), a measure of financial openness (fivop), and a measure of United States hegemony (ushag). These variables correspond to the variables used in the analysis model of Milner and Kubota (2005, Table 2).

We first load the Amelia and the data:

```r
R> library("Amelia")
```

```
## Amelia II: Multiple Imputation
## (Version 1.5-4, built: 2011-08-20)
## Copyright (C) 2005-2011 James Honaker, Gary King and Matthew Blackwell
## Refer to http://gking.harvard.edu/amelia/ for more information
```
R> data("freetrade")

We can check the summary statistics of the data to see that there is missingness on many of the variables:

R> summary(freetrade)

<table>
<thead>
<tr>
<th></th>
<th>Min.</th>
<th>1st Qu.</th>
<th>Median</th>
<th>Mean</th>
<th>3rd Qu.</th>
<th>Max.</th>
<th>NA's</th>
</tr>
</thead>
<tbody>
<tr>
<td>country</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2.000</td>
</tr>
<tr>
<td>tariff</td>
<td>7.10</td>
<td>16.30</td>
<td>25.20</td>
<td>31.65</td>
<td>40.80</td>
<td>100.00</td>
<td>58.00</td>
</tr>
<tr>
<td>polity</td>
<td>-8.000</td>
<td>-2.000</td>
<td>5.000</td>
<td>2.905</td>
<td>8.000</td>
<td>9.000</td>
<td>2.000</td>
</tr>
<tr>
<td>pop</td>
<td>14105080</td>
<td>19676715</td>
<td>52799040</td>
<td>149904501</td>
<td>120888400</td>
<td>997515200</td>
<td>13.0000</td>
</tr>
<tr>
<td>gdp.pc</td>
<td>149.5</td>
<td>420.1</td>
<td>814.3</td>
<td>1867.3</td>
<td>2462.9</td>
<td>12086.2</td>
<td>7.9346</td>
</tr>
<tr>
<td>intresmi</td>
<td>0.9036</td>
<td>2.2231</td>
<td>3.1815</td>
<td>3.3752</td>
<td>4.4063</td>
<td>4.4063</td>
<td>13.0000</td>
</tr>
<tr>
<td>signed</td>
<td>0.0000</td>
<td>12.30</td>
<td>0.2558</td>
<td>0.2558</td>
<td>12.50</td>
<td>12.50</td>
<td>0.2887</td>
</tr>
<tr>
<td>fiveop</td>
<td>0.0000</td>
<td>12.60</td>
<td>0.2756</td>
<td>0.2756</td>
<td>12.74</td>
<td>12.74</td>
<td>0.3083</td>
</tr>
<tr>
<td>usheg</td>
<td>0.0000</td>
<td>13.20</td>
<td>0.2887</td>
<td>0.2887</td>
<td>13.20</td>
<td>13.20</td>
<td>0.3083</td>
</tr>
<tr>
<td>NA's</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>18.00</td>
</tr>
</tbody>
</table>

In the presence of missing data, most statistical packages use listwise deletion, which removes any row that contains a missing value from the analysis. Using the base model of Milner and Kubota (2005, Table 2), we run a simple linear model in R, which uses listwise deletion:

R> summary(lm(tariff ~ polity + pop + gdp.pc + year + country, + data = freetrade))

Call:
  lm(formula = tariff ~ polity + pop + gdp.pc + year + country, 
     data = freetrade)

Residuals:
    Min     1Q    Median     3Q    Max
  -30.764  -3.2595    0.0868    2.5983  18.3097
Coefficients:                           Estimate  Std. Error t value Pr(>|t|)
(Intercept)                        1.973e+03   4.016e+02  4.912  3.61e-06
polity                          -1.373e-01   1.821e-01 -0.754   0.453
pop                          -2.021e-07   2.542e-08 -7.951  3.23e-12
gdp.pc                        6.096e-04    7.442e-04  0.819   0.415
year                         -8.705e-01   2.084e-01 -4.176  6.43e-05
countryIndonesia            -1.823e+02   1.857e+01 -9.819  2.98e-16
countryKorea                  -2.204e+02   2.078e+01 -10.608 < 2e-16
countryMalaysia             -2.245e+02   2.171e+01 -10.343 < 2e-16
countryNepal                 -2.163e+02   2.247e+01 -9.629  7.46e-16
countryPakistan             -1.554e+02   1.982e+01 -7.838  5.63e-12
countryPhilippines          -2.040e+02   2.088e+01 -9.774  3.75e-16
countrySriLanka              -2.091e+02   2.210e+01 -9.460  1.80e-15
countryThailand              -1.961e+02   2.095e+01 -9.358  2.99e-15

Residual standard error: 6.221 on 98 degrees of freedom
(60 observations deleted due to missingness)
Multiple R-squared: 0.9247, Adjusted R-squared: 0.9155
F-statistic: 100.3 on 12 and 98 DF, p-value: < 2.2e-16

Note that 60 of the 171 original observations are deleted due to missingness. These observations, however, are partially observed, and contain valuable information about the relationships between those variables which are present in the partially completed observations. Multiple imputation will help us retrieve that information and make better, more efficient, inferences.

4.2. Multiple imputation

When performing multiple imputation, the first step is to identify the variables to include in the imputation model. It is crucial to include at least as much information as will be used in the analysis model. That is, any variable that will be in the analysis model should also be in the imputation model. This includes any transformations or interactions of variables that will appear in the analysis model.

In fact, it is often useful to add more information to the imputation model than will be present when the analysis is run. Since imputation is predictive, any variables that would increase predictive power should be included in the model, even if including them in the analysis model would produce bias in estimating a causal effect (such as for post-treatment variables) or collinearity would preclude determining which variable had a relationship with the dependent variable (such as including multiple alternate measures of GDP). In our case, we include all the variables in freetrade in the imputation model, even though our analysis model focuses on polity, pop and gdp.pc.7

To create multiple imputations in Amelia, we can simply run

R> a.out <- amelia(fretrade, m = 5, ts = "year", cs = "country")

7Note that this specification does not utilize time or spatial data yet. The ts and cs arguments only have force when we also include polytime or intercs, discussed in Section 4.5.
R> a.out

Amelia output with 5 imputed datasets.
Return code: 1
Message: Normal EM convergence.

Chain Lengths:
--------------
Imputation 1: 16
Imputation 2: 14
Imputation 3: 15
Imputation 4: 21
Imputation 5: 17

Note that our example dataset is deliberately small both in variables and in cross-sectional elements. Typical datasets may often have hundreds or possibly a couple thousand steps to the EM algorithm. Long chains should remind the analyst to consider whether transformations of the variables would more closely fit the multivariate normal assumptions of the model (correct but omitted transformations will shorten the number of steps and improve the fit of the imputations), but do not necessarily denote problems with the imputation model.

The output gives some information about how the algorithm ran. Each of the imputed datasets is now in the list `a.out$imputations`. Thus, we could plot a histogram of the `tariff` variable from the 3rd imputation,

R> hist(a.out$imputations[[3]]$tariff, col = "grey", border = "white")
Saving imputed datasets

If you need to save your imputed datasets, one direct method is to save the output list from amelia,

```r
R> save(a.out, file = "imputations.RData")
```

As in the previous example, the i-th imputed datasets can be retrieved from this list as `a.out$imputations[[i]]`

In addition, you can save each of the imputed datasets to its own file using the `write.amelia` command,

```r
R> write.amelia(obj = a.out, file.stem = "outdata")
```

This will create one comma-separated value file for each imputed dataset:

outdata1.csv
outdata2.csv
outdata3.csv
outdata4.csv
outdata5.csv

The `write.amelia` function can also save files in tab-delimited and Stata (.dta) file formats. For instance, to save Stata files, simply change the `format` argument to "dta",

```r
R> write.amelia(obj = a.out, file.stem = "outdata", format = "dta")
```
Combining multiple runs of `amelia`

The EMB algorithm is what computer scientists call *embarrassingly parallel*, meaning that it is simple to separate each imputation into parallel processes. With `Amelia` it is simple to run subsets of the imputations on different machines and then combine them after the imputation for use in analysis model. This allows for a huge increase in the speed of the algorithm.

Output lists from different `Amelia` runs can be combined together into a new list. For instance, suppose that we wanted to add another ten imputed datasets to our earlier call to `amelia`. First, run the function to get these additional imputations,

```R
R> a.out.more <- amelia(freetrade, m = 10, ts = "year", cs = "country", +       p2s = 0)
R> a.out.more

Amelia output with 10 imputed datasets.
Return code: 1
Message: Normal EM convergence.

Chain Lengths:
-------------
Imputation 1: 15
Imputation 2: 13
Imputation 3: 11
Imputation 4: 21
Imputation 5: 13
Imputation 6: 14
Imputation 7: 11
Imputation 8: 10
Imputation 9: 12
Imputation 10: 10
```

then combine this output with our original output using the `ameliabind` function,

```R
R> a.out.more <- ameliabind(a.out, a.out.more)
R> a.out.more

Amelia output with 15 imputed datasets.
Return code: 1
Message: Normal EM convergence

Chain Lengths:
-------------
Imputation 1: 16
Imputation 2: 14
Imputation 3: 15
Imputation 4: 21
Imputation 5: 17
```
This function binds the two outputs into the same output so that you can pass the combined imputations easily to analysis models and diagnostics. Note that `a.out.more` now has a total of 15 imputations.

A simple way to execute a parallel processing scheme with Amelia would be to run `amelia` with `m` set to 1 on `m` different machines or processors, save each output using the `save` function, load them all on the same R session using `load` command and then combine them using `ameliabind`. In order to do this, however, make sure to name each of the outputs a different name so that they do not overwrite each other when loading into the same R session. Also, some parallel environments will dump all generated files into a common directory, where they may overwrite each other. If it is convenient in a parallel environment to run a large number of `amelia` calls from a single piece of code, one useful way to avoid overwriting is to create the `file.stem` with a random suffix. For example:

```r
R> b <- round(runif(1, min = 1111, max = 9999))
R> random.name <- paste("am", b, sep = ")
R> amelia <- write.amelia(obj = a.out, file.stem = random.name)
```

**Screen output**

Screen output can be adjusted with the “print to screen” argument, `p2s`. At a value of 0, no screen printing will occur. This may be useful in large jobs or simulations where a very large number of imputation models may be required. The default value of 1, lists each bootstrap, and displays the number of iterations required to reach convergence in that bootstrapped dataset. The value of 2 gives more thorough screen output, including, at each iteration, the number of parameters that have significantly changed since the last iteration. This may be useful when the EM chain length is very long, as it can provide an intuition for many parameters still need to converge in the EM chain, and a sense of the time remaining. However, it is worth noting that the last several parameters can often take a significant fraction of the total number of iterations to converge. Setting `p2s` to 2 will also generate information on how EM algorithm is behaving, such as ! when the current estimated complete data covariance matrix is not invertible and * when the likelihood has not monotonically increased in that step. Having many of these two symbols in the screen output is an indication of a problematic imputation model.\(^8\)

---

\(^8\)Problems of non-invertible matrices often mean that current guess for the covariance matrix is singular. This is a sign that there may be two highly correlated variables in the model. One way to resolve is to use a ridge prior (see Section 4.6).
An example of the output when p2s is 2 would be

\[ R> amelia(freetrade, m = 1, ts = "year", cs = "country", p2s = 2) \]

amelia starting
beginning prep functions
Variables used:  tariff polity pop gdp.pc intresmi signed fiveop usheg
running bootstrap
-- Imputation 1 --
setting up EM chain indicies

1(44) 2(35) 3(26) 4(23) 5(18) 6(15) 7(15) 8(12) 9(10) 10(7)
11(5) 12(2) 13(0)

saving and cleaning

Amelia output with 1 imputed datasets.
Return code: 1
Message: Normal EM convergence.

Chain Lengths:
--------------
Imputation 1: 13

4.3. Imputation-improving transformations

Social science data commonly includes variables that fail to fit to a multivariate normal distribution. Indeed, numerous models have been introduced specifically to deal with the problems they present. As it turns out, much evidence in the literature (discussed in King, Honaker, Joseph, and Scheve 2001) indicates that the multivariate normal model used in Amelia usually works well for the imputation stage even when discrete or non-normal variables are included and when the analysis stage involves these limited dependent variable models. Nevertheless, Amelia includes some limited capacity to deal directly with ordinal and nominal variables and to modify variables that require other transformations. In general nominal and log transform variables should be declared to Amelia, whereas ordinal (including dichotomous) variables often need not be, as described below. (For harder cases, see (Schafer 1997), for specialized MCMC-based (Markov chain Monte Carlo) imputation models for discrete variables.)

Although these transformations are taken internally on these variables to better fit the data to the multivariate normal assumptions of the imputation model, all the imputations that are created will be returned in the original untransformed form of the data. If the user has already performed transformations on their data (such as by taking a log or square root prior to feeding the data to amelia) these do not need to be declared, as that would result in the transformation occurring \emph{doubly} in the imputation model. The fully imputed data sets that are returned will always be in the form of the original data that is passed to the amelia routine.
**Ordinal variables**

In much statistical research, researchers treat independent ordinal (including dichotomous) variables as if they were really continuous. If the analysis model to be employed is of this type, then nothing extra is required of the imputation model. Users are advised to allow *Amelia* to impute non-integer values for any missing data, and to use these non-integer values in their analysis. Sometimes this makes sense, and sometimes this defies intuition. One particular imputation of 2.35 for a missing value on a seven point scale carries the intuition that the respondent is between a 2 and a 3 and most probably would have responded 2 had the data been observed. This is easier to accept than an imputation of 0.79 for a dichotomous variable where a zero represents a male and a one represents a female respondent. However, in both cases the non-integer imputations carry more information about the underlying distribution than would be carried if we were to force the imputations to be integers. Thus whenever the analysis model permits, missing ordinal observations should be allowed to take on continuously valued imputations.

In the *freetrade* data, one such ordinal variable is *polity* which ranges from $-10$ (full autocracy) to 10 (full democracy). If we tabulate this variable from one of the imputed datasets,

```R
R> table(round(a.out$imputations[[3]]$polity, digits = 3))
   -8 -7 -6 -5 -4 -3.858 -2 -1 2 3
   1 22 4 7 3 1 9 1 7 7
   4 5 6 6.365 7 8 9
15 26 13 1 5 36 13
```

we can see that there is one imputation between $-4$ and $-3$ and one imputation between 6 and 7. Again, the interpretation of these values is rather straightforward even if they are not strictly in the coding of the original Polity data.

Often, however, analysis models require some variables to be strictly ordinal, as for example, when the dependent variable will be modeled in a logistical or Poisson regression. Imputations for variables set as ordinal are created by taking the continuously valued imputation and using an appropriately scaled version of this as the probability of success in a binomial distribution. The draw from this binomial distribution is then translated back into one of the ordinal categories. For our data we can simply add *polity* to the *ords* argument:

```R
R> a.out1 <- amelia(freetrade, m = 5, ts = "year", cs = "country",
+ ords = "polity", p2s = 0)
R> table(a.out1$imputations[[3]]$polity)
   -8 -7 -6 -5 -4 -2 -1 2 3 4 5 6 7 8 9
   1 22 4 8 3 9 1 7 15 27 13 5 36 13
```

Now, we can see that all of the imputations fall into one of the original polity categories.

**Nominal variables**

Nominal variables\(^9\) must be treated quite differently than ordinal variables. Any multinomial variables in the data set (such as religion coded 1 for Catholic, 2 for Jewish, and 3 for

\(^9\)Dichotomous (two category) variables are a special case of nominal variables. For these variables, the nominal and ordinal methods of transformation in *Amelia* agree.
Protestant) must be specified to Amelia. In our freetrade dataset, we have signed which is 1 if a country signed an IMF agreement in that year and 0 if it did not. Of course, our first imputation did not limit the imputations to these two categories

\begin{verbatim}
R> table(round(a.out1$imputations[[3]]$signed, digits = 3))

  0  0.005  0.065  0.185  1
142  1  1  1  26
\end{verbatim}

In order to fix this for a \( p \)-category multinomial variable, Amelia will determine \( p \) (as long as your data contain at least one value in each category), and substitute \( p - 1 \) binary variables to specify each possible category. These new \( p - 1 \) variables will be treated as the other variables in the multivariate normal imputation method chosen, and receive continuous imputations. These continuously valued imputations will then be appropriately scaled into probabilities for each of the \( p \) possible categories, and one of these categories will be drawn, where upon the original \( p \)-category multinomial variable will be reconstructed and returned to the user. Thus all imputations will be appropriately multinomial.

For our data we can simply add signed to the noms argument:

\begin{verbatim}
R> a.out2 <- amelia(freetrade, m = 5, ts = "year", cs = "country",
+ noms = "signed", p2s = 0)
R> table(a.out2$imputations[[3]]$signed)

  0  1
144 27
\end{verbatim}

Note that Amelia can only fit imputations into categories that exist in the original data. Thus, if there was a third category of signed, say 2, that corresponded to a different kind of IMF agreement, but it never occurred in the original data, Amelia could not match imputations to it.

Since Amelia properly treats a \( p \)-category multinomial variable as \( p - 1 \) variables, one should understand the number of parameters that are quickly accumulating if many multinomial variables are being used. If the square of the number of real and constructed variables is large relative to the number of observations, it is useful to use a ridge prior as in Section 4.6.

**Natural log**

If one of your variables is heavily skewed or has outliers that may alter the imputation in an unwanted way, you can use a natural logarithm transformation of that variable in order to normalize its distribution. This transformed distribution helps Amelia to avoid imputing values that depend too heavily on outlying data points. Log transformations are common in expenditure and economic variables where we have strong beliefs that the marginal relationship between two variables decreases as we move across the range.

For instance, Figure 3 show the tariff variable clearly has positive (or, right) skew while its natural log transformation has a roughly normal distribution.
Event count data is often heavily skewed and has nonlinear relationships with other variables. One common transformation to tailor the linear model to count data is to take the square roots of the counts. This is a transformation that can be set as an option in *Amelia*.

Logistic

Proportional data is sharply bounded between 0 and 1. A logistic transformation is one possible option in *Amelia* to make the distribution symmetric and relatively unbounded.

4.4. Identification variables

Datasets often contain identification variables, such as country names, respondent numbers, or other identification numbers, codes or abbreviations. Sometimes these are text and sometimes these are numeric. Often it is not appropriate to include these variables in the imputation model, but it is useful to have them remain in the imputed datasets (However, there are models that would include the ID variables in the imputation model, such as fixed effects model for data with repeated observations of the same countries). Identification variables which are not to be included in the imputation model can be identified with the argument `idvars`. These variables will not be used in the imputation model, but will be kept in the imputed datasets.

If the `year` and `country` contained no information except labels, we could omit them from the imputation:
R> amelia(freetrade, idvars = c("year", "country"))

Note that Amelia will return with an error if your dataset contains a factor or character variable that is not marked as a nominal or identification variable. Thus, if we were to omit the factor country from the cs or idvars arguments, we would receive an error:

R> a.out2 <- amelia(freetrade, idvars = c("year"))

Amelia Error Code: 38
The following variable(s) are characters:
country
You may have wanted to set this as a ID variable to remove it from the imputation model or as an ordinal or nominal variable to be imputed. Please set it as either and try again.

In order to conserve memory, it is wise to remove unnecessary variables from a data set before loading it into Amelia. The only variables you should include in your data when running Amelia are variables you will use in the analysis stage and those variables that will help in the imputation model. While it may be tempting to simply mark unneeded variables as IDs, it only serves to waste memory and slow down the imputation procedure.

4.5. Time series, or time-series cross-sectional data

Many variables that are recorded over time within a cross-sectional unit are observed to vary smoothly over time. In such cases, knowing the observed values of observations close in time to any missing value may enormously aid the imputation of that value. However, the exact pattern may vary over time within any cross-section. There may be periods of growth, stability, or decline; in each of which the observed values would be used in a different fashion to impute missing values. Also, these patterns may vary enormously across different cross-sections, or may exist in some and not others. Amelia can build a general model of patterns within variables across time by creating a sequence of polynomials of the time index. If, for example, tariffs vary smoothly over time, then we make the modeling assumption that there exists some polynomial that describes the economy in cross-sectional unit \(i\) at time \(t\) as:

\[
tariff_{ti} = \beta_0 + \beta_1 t + \beta_2 t^2 + \beta_3 t^3 \ldots
\]

And thus if we include enough higher order terms of time then the pattern between observed values of the tariff rate can be estimated. Amelia will create polynomials of time up to the user defined \(k\)-th order, \((k \leq 3)\).

We can implement this with the ts and polytime arguments. If we thought that a second-order polynomial would help predict we could run

R> a.out2 <- amelia(freetrade, ts = "year", cs = "country", polytime = 2)

With this input, Amelia will add covariates to the model that correspond to time and its polynomials. These covariates will help better predict the missing values.
Amelia II: A Program for Missing Data

Figure 4: The increase in predictive power when using polynomials of time. The panels show mean imputations with 95% bands (in red) and observed data point (in black). The left panel shows an imputation without using time and the right panel includes polynomials of time.

If cross-sectional units are specified these polynomials can be interacted with the cross-section unit to allow the patterns over time to vary between cross-sectional units. Unless you strongly believe all units have the same patterns over time in all variables (including the same constant term), this is a reasonable setting. When $k$ is set to 0, this interaction simply results in a model of fixed effects where every unit has a uniquely estimated constant term. Amelia does not smooth the observed data, and only uses this functional form, or one you choose, with all the other variables in the analysis and the uncertainty of the prediction, to impute the missing values.

In order to impute with trends specific to each cross-sectional unit, we can set intercs to TRUE:

```R
R> a.out.time <- amelia(freetrade, ts = "year", cs = "country", polytime = 2, + intercs = TRUE, p2s = 2)
```

Note that attempting to use polytime without the ts argument, or intercs without the cs argument will result in an error.

Using the tscsPlot function (discussed below), we can see in Figure 4 that we have a much better prediction about the missing values when incorporating time than when we omit it:

```R
R> tscsPlot(a.out, cs = "Malaysia", var = "tariff", ylim = c(-10, 60), + main = "Malaysia (no time settings)"
R> tscsPlot(a.out.time, cs = "Malaysia", var = "tariff", + ylim = c(-10, 60), main = "Malaysia (with time settings)"
```
Lags and leads

An alternative way of handling time-series information is to include lags and leads of certain variables into the imputation model. Lags are variables that take the value of another variable in the previous time period while leads take the value of another variable in the next time period. Many analysis models use lagged variables to deal with issues of endogeneity, thus using leads may seems strange. It is important to remember, however, that imputation models are predictive, not causal. Thus, since both past and future values of a variable are likely correlated with the present value, both lags and leads should improve the model.

If we wanted to include lags and leads of tariffs, for instance, we would simply pass this to the lags and leads arguments:

```R
R> a.out2 <- amelia(freetrade, ts = "year", cs = "country", lags = "tariff",
+                  leads = "tariff")
```

4.6. Including prior information

Amelia has a number of methods of setting priors within the imputation model. Two of these are commonly used and discussed below, ridge priors and observational priors.

Ridge priors for high missingness, small n’s, or large correlations

When the data to be analyzed contain a high degree of missingness or very strong correlations among the variables, or when the number of observations is only slightly greater than the number of parameters \( p(p + 3)/2 \) (where \( p \) is the number of variables), results from your analysis model will be more dependent on the choice of imputation model. This suggests more testing in these cases of alternative specifications under Amelia. This can happen when using the polynomials of time interacted with the cross section are included in the imputation model. In our running example, we used a polynomial of degree 2 and there are 9 countries. This adds \( 3 \times 9 - 1 = 17 \) more variables to the imputation model (One of the constant “fixed effects” will be dropped so the model will be identified). When these are added, the EM algorithm can become unstable, as indicated by the vastly differing chain lengths for each imputation:

```R
R> a.out.time

Amelia output with 5 imputed datasets.
Return code: 1
Message: Normal EM convergence.

Chain Lengths:
--------------
Imputation 1: 315
Imputation 2: 50
Imputation 3: 123
Imputation 4: 95
Imputation 5: 165
```
In these circumstances, we recommend adding a ridge prior which will help with numerical stability by shrinking the covariances among the variables toward zero without changing the means or variances. This can be done by including the `empri` argument. Including this prior as a positive number is roughly equivalent to adding `empri` artificial observations to the data set with the same means and variances as the existing data but with zero covariances. Thus, increasing the `empri` setting results in more shrinkage of the covariances, thus putting more a priori structure on the estimation problem: like many Bayesian methods, it reduces variance in return for an increase in bias that one hopes does not overwhelm the advantages in efficiency. In general, we suggest keeping the value on this prior relatively small and increase it only when necessary. A recommendation of 0.5 to 1 percent of the number of observations, \( n \), is a reasonable starting value, and often useful in large datasets to add some numerical stability. For example, in a dataset of two thousand observations, this would translate to a prior value of 10 or 20 respectively. A prior of up to 5 percent is moderate in most applications and 10 percent is reasonable upper bound.

For our data, it is easy to code up a 5 percent ridge prior:

```r
R> a.out.time2 <- amelia(freetrade, ts = "year", cs = "country",
+   polytime = 2, intercs = TRUE, p2s = 0, empri = 0.01 * nrow(freetrade))
R> a.out.time2

Amelia output with 5 imputed datasets.
Return code: 1
Message: Normal EM convergence.

Chain Lengths:
-------------
Imputation 1: 13
Imputation 2: 17
Imputation 3: 18
Imputation 4: 17
Imputation 5: 24

This new imputation model is much more stable and, as shown by using `tscsPlot`, produces about the same imputations as the original model (see Figure 5):

```r
R> tscsPlot(a.out.time, cs = "Malaysia", var = "tariff",
+   ylim = c(-10, 60), main = "Malaysia (no ridge prior)"
R> tscsPlot(a.out.time2, cs = "Malaysia", var = "tariff",
+   ylim = c(-10, 60), main = "Malaysia (with ridge prior)"
```

**Observation-level priors**

Researchers often have additional prior information about missing data values based on previous research, academic consensus, or personal experience. *Amelia* can incorporate this information to produce vastly improved imputations. The *Amelia* algorithm allows users to include informative Bayesian priors about individual missing data cells instead of the more general model parameters, many of which have little direct meaning.
The incorporation of priors follows basic Bayesian analysis where the imputation turns out to be a weighted average of the model-based imputation and the prior mean, where the weights are functions of the relative strength of the data and prior: when the model predicts very well, the imputation will down-weight the prior, and vice versa (Honaker and King 2010).

The priors about individual observations should describe the analyst’s belief about the distribution of the missing data cell. This can either take the form of a mean and a standard deviation or a confidence interval. For instance, we might know that 1986 tariff rates in Thailand around 40%, but we have some uncertainty as to the exact value. Our prior belief about the distribution of the missing data cell, then, centers on 40 with a standard deviation that reflects the amount of uncertainty we have about our prior belief.

To input priors you must build a priors matrix with either four or five columns. Each row of the matrix represents a prior on either one observation or one variable. In any row, the entry in the first column is the row of the observation and the entry is the second column is the column of the observation. In the four column priors matrix the third and fourth columns are the mean and standard deviation of the prior distribution of the missing value.

For instance, suppose that we had some expert prior information about tariff rates in Thailand.

For instance, suppose that we had some expert prior information about tariff rates in Thailand.

```R
> freetrade[freetrade$country == "Thailand", c("year", "country", "tariff")]
```

<table>
<thead>
<tr>
<th>year</th>
<th>country</th>
<th>tariff</th>
</tr>
</thead>
<tbody>
<tr>
<td>153</td>
<td>1981</td>
<td>Thailand</td>
</tr>
<tr>
<td>154</td>
<td>1982</td>
<td>Thailand</td>
</tr>
</tbody>
</table>

Figure 5: The difference in imputations when using no ridge prior (left) and when using a ridge prior set to 1% of the data (right).
Suppose that we had expert information that tariff rates were roughly 40% in Thailand between 1986 and 1988 with about a 6% margin of error. This corresponds to a standard deviation of about 3. In order to include this information, we must form the priors matrix:

```R
R> pr <- matrix(c(158, 159, 160, 3, 3, 3, 40, 40, 40, 3, 3, 3),
                 + nrow = 3, ncol = 4)
R> pr
```

```
[1,] 158  3  40  3
[2,] 159  3  40  3
[3,] 160  3  40  3
```

The first column of this matrix corresponds to the row numbers of Thailand in these three years, the second column refers to the column number of tariff in the data and the last two columns refer to the actual prior. Once we have this matrix, we can pass it to `amelia`.

```R
R> a.out.pr <- amelia(freetrade, ts = "year", cs = "country", priors = pr)
```

In the five column matrix, the last three columns describe a confidence range of the data. The columns are a lower bound, an upper bound, and a confidence level between 0 and 1, exclusive. Whichever format you choose, it must be consistent across the entire matrix. We could get roughly the same prior as above by utilizing this method. Our margin of error implies that we would want imputations between 34 and 46, so our matrix would be

```R
R> pr.2 <- matrix(c(158, 159, 160, 3, 3, 3, 34, 34, 34, 46, 46, 46, 0.95, + 0.95, 0.95), nrow = 3, ncol = 5)
R> pr.2
```
These priors indicate that we are 95% confident that these missing values are in the range 34 to 46.

If a prior has the value 0 in the first column, this prior will be applied to all missing values in this variable, except for explicitly set priors. Thus, we could set a prior for the entire tariff variable of 20, but still keep the above specific priors with the following code:

```r
R> pr.3 <- matrix(c(158, 159, 160, 0, 3, 3, 3, 3, 40, 40, 40, 20, 3, 3, 3, 5), nrow = 4, ncol = 4)
R> pr.3
[1,] 158  3  34  46  0.95
[2,] 159  3  34  46  0.95
[3,] 160  3  34  46  0.95
```

**Logical bounds**

In some cases, variables in the social sciences have known logical bounds. Proportions must be between 0 and 1 and duration data must be greater than 0, for instance. Many of these logical bounds can be handled by using the correct transformation for that type of variable (see Section 4.3 for more details on the transformations handled by Amelia). In the occasional case that imputations must satisfy certain logical bounds not handled by these transformations, Amelia can take draws from a truncated normal distribution in order to achieve imputations that satisfy the bounds. Note, however, that this procedure imposes extremely strong restrictions on the imputations and can lead to lower variances than the imputation model implies. The mean value across all the imputed values of a missing cell is the best guess from the imputation model of that missing value. The variance of the distribution across imputed datasets correctly reflects the uncertainty in that imputation. It is often the mean imputed value that should conform to the any known bounds, even if individual imputations are drawn beyond those bounds. The mean imputed value can be checked with the diagnostics presented in the next section. In general, building a more predictive imputation model will lead to better imputations than imposing bounds.

Amelia implements these bounds by rejection sampling. When drawing the imputations from their posterior, we repeatedly resample until we have a draw that satisfies all of the logical constraints. You can set an upper limit on the number of times to resample with the max.resample arguments. Thus, if after max.resample draws, the imputations are still outside the bounds, Amelia will set the imputation at the edge of the bounds. Thus, if the bounds were 0 and 100 and all of the draws were negative, Amelia would simply impute 0.

As an extreme example, suppose that we know, for certain that tariff rates had to fall between 30 and 40. This, obviously, is not true, but we can generate imputations from this model. In
order to specify these bounds, we need to generate a matrix of bounds to pass to the `bounds` argument. This matrix will have 3 columns: the first is the column for the bounded variable, the second is the lower bound and the third is the upper bound. Thus, to implement our bound on tariff rates (the 3rd column of the dataset), we would create the matrix,

\[
R\> bds <- \text{matrix}(c(3, 30, 40), nrow = 1, ncol = 3)
\]

\[
\begin{bmatrix}
[1,] & 3 & 30 & 40
\end{bmatrix}
\]

which we can pass to the `bounds` argument,

\[
R\> \text{amelia}(\text{freetrade}, ts = "year", cs = "country", + \text{bounds} = bds, \text{max.resample} = 1000)
\]

The difference in results between the bounded and unbounded model are not obvious from the output, but inspection of the imputed tariff rates for Malaysia in Figure 6 shows that there has been a drastic restriction of the imputations to the desired range:

\[
R\> \text{tscsPlot}(\text{a.out}, cs = "Malaysia", main = "No logical bounds", + \text{var} = "tariff", \text{ylim} = c(-10, 60))
\]

\[
R\> \text{tscsPlot}(\text{a.out.bds}, cs = "Malaysia", main = "Bounded between 30 and 40", + \text{var} = "tariff", \text{ylim} = c(-10, 60))
\]

Again, analysts should be extremely cautious when using these bounds as they can seriously affect the inferences from the imputation model, as shown in this example. Even when logical
Figure 6: On the left are the original imputations without logical bounds and on the right are the imputation after imposing the bounds.

bounds exist, we recommend simply imputing variables normally, as the violation of the logical bounds represents part of the true uncertainty of imputation.

4.7. Diagnostics

Amelia currently provides a number of diagnostic tools to inspect the imputations that are created.

Comparing densities

One check on the plausibility of the imputation model is check the distribution of imputed values to the distribution of observed values. Obviously we cannot expect, *a priori*, that these distribution will be identical as the missing values may differ systematically from the observed value—this is fundamental reason to impute to begin with! Imputations with strange distributions or those that are far from the observed data may indicate that imputation model needs at least some investigation and possibly some improvement.

The plot method works on output from Amelia and, by default, shows for each variable a plot of the relative frequencies of the observed data with an overlay of the relative frequency of the imputed values.

\[ R > \text{plot(a.out, which.vars = 3:6)} \]

where the argument `which.vars` indicates which of the variables to plot (in this case, we are taking the 3rd through the 6th variables).
Figure 7: The output of the plot method as applied to output from amelia. In the upper panels, the distribution of mean imputations (in red) is overlayed on the distribution of observed values (in black) for each variable. In the lower panels, there are no missing values and the distribution of observed values is simply plotted (in blue). Note that now imputed tariff rates are very similar to observed tariff rates, but the imputation of the Polity score are quite different. This is plausible if different types of regimes tend to be missing at different rates.

The imputed curve (in red) plots the density of the mean imputation over the m datasets. That is, for each cell that is missing in the variable, the diagnostic will find the mean of that cell across each of the m datasets and use that value for the density plot. The black distributions are the those of the observed data. When variables are completely observed, their densities are plotted in blue. These graphs will allow you to inspect how the density of imputations compares to the density of observed data. Some discussion of these graphs can be found in Abayomi, Gelman, and Levy (2008). Minimally, these graphs can be used to check that the mean imputation falls within known bounds, when such bounds exist in certain variables or settings.
Overimputation is a technique we have developed to judge the fit of the imputation model. Because of the nature of the missing data mechanism, it is impossible to tell whether the mean prediction of the imputation model is close to the unobserved value that is trying to be recovered. By definition this missing data does not exist to create this comparison, and if it existed we would no longer need the imputations or care about their accuracy. However, a natural question the applied researcher will often ask is how accurate are these imputed values?

Overimputing involves sequentially treating each of the observed values as if they had actually
been missing. For each observed value in turn we then generate several hundred imputed values of that observed value, *as if it had been missing*. While \( m = 5 \) imputations are sufficient for most analysis models, this large number of imputations allows us to construct a confidence interval of what the imputed value would have been, had any of the observed data been missing. We can then graphically inspect whether our observed data tends to fall within the region where it would have been imputed had it been missing.

For example, we can run the overimputation diagnostic on our data by running

```r
R> overimpute(a.out, var = "tariff")
```

Our overimputation diagnostic, shown in Figure 8, runs this procedure through all of the observed values for a user selected variable. We can graph the estimates of each observation against the true values of the observation. On this graph, a \( y = x \) line indicates the line of perfect agreement; that is, if the imputation model was a perfect predictor of the true value, all the imputations would fall on this line. For each observation, Amelia also plots 90% confidence intervals that allows the user to visually inspect the behavior of the imputation model. By checking how many of the confidence intervals cover the \( y = x \) line, we can tell how often the imputation model can confidently predict the true value of the observation.

Occasionally, the overimputation can display unintuitive results. For example, different ob-

![Observed versus Imputed Values](image)
servations may have different numbers of observed covariates. If covariates that are useful to
the prediction are themselves missing, then the confidence interval for this observation will
be much larger. In the extreme, there may be observations where the observed value we are
trying to overimpute is the only observed value in that observation, and thus there is noth-
ing left to impute that observation with when we pretend that it is missing, other than the
mean and variance of that variable. In these cases, we should correctly expect the confidence
interval to be very large.

An example of this graph is shown in Figure 9. In this simulated bivariate dataset, one
variable is overimputed and the results displayed. The second variable is either observed,
in which case the confidence intervals are very small and the imputations (yellow) are very
accurate, or the second variable is missing in which case this variable is being imputed simply
from the mean and variance parameters, and the imputations (red) have a very large and
encompassing spread. The circles represent the mean of all the imputations for that value.
As the amount of missing information in a particular pattern of missingness increases, we
expect the width of the confidence interval to increase. The color of the confidence interval
reflects the percent of covariates observed in that pattern of missingness, as reflected in the
legend at the bottom.

Overdispersed starting values
If the data given to Amelia has a poorly behaved likelihood, the EM algorithm can have
problems finding a global maximum of the likelihood surface and starting values can begin
to effect imputations. Because the EM algorithm is deterministic, the point in the parameter
space where you start it can impact where it ends, though this is irrelevant when the likelihood
has only one mode. However, if the starting values of an EM chain are close to a local
maximum, the algorithm may find this maximum, unaware that there is a global maximum
farther away. To make sure that our imputations do not depend on our starting values, a
good test is to run the EM algorithm from multiple, dispersed starting values and check their
convergence. In a well behaved likelihood, we will see all of these chains converging to the
same value, and reasonably conclude that this is the likely global maximum. On the other
hand, we might see our EM chain converging to multiple locations. The algorithm may also
wander around portions of the parameter space that are not fully identified, such as a ridge of
equal likelihood, as would happen for example, if the same variable were accidentally included
in the imputation model twice.

Amelia includes a diagnostic to run the EM chain from multiple starting values that are
overdispersed from the estimated maximum. The overdispersion diagnostic will display a
graph of the paths of each chain. Since these chains move through spaces that are in an
extremely high number of dimensions and can not be graphically displayed, the diagnostic
reduces the dimensionality of the EM paths by showing the paths relative to the largest princi-
ple components of the final mode(s) that are reached. Users can choose between graphing the
movement over the two largest principal components, or more simply the largest dimension
with time (iteration number) on the x-axis. The number of EM chains can also be adjusted.
Once the diagnostic draws the graph, the user can visually inspect the results to check that
all chains convergence to the same point.

For our original model, this is a simple call to disperse:

\[ R> \text{disperse(a.out, dims = 1, m = 5)} \]
**Figure 10:** A plot from the overdispersion diagnostic where all EM chains are converging to the same mode, regardless of starting value. On the left, the $y$-axis represents movement in the (very high dimensional) parameter space, and the $x$-axis represents the iteration number of the chain. On the right, we visualize the parameter space in two dimensions using the first two principal components of the end points of the EM chains. The iteration number is no longer represented on the $y$-axis, although the distance between iterations is marked by the distance between arrowheads on each chain.

\[ R > \text{disperse}(a.\text{out}, \text{dims} = 2, m = 5) \]

where \( m \) designates the number of places to start EM chains from and \( \text{dims} \) are the number of dimensions of the principal components to show.

In one dimension, the diagnostic plots movement of the chain on the $y$-axis and time, in the form of the iteration number, on the $x$-axis. Figure 10 shows two examples of these plots. The first shows a well behaved likelihood, as the starting values all converge to the same point. The black horizontal line is the point where **Amelia** converges when it uses the default method for choosing the starting values. The diagnostic takes the end point of this chain as the possible maximum and disperses the starting values away from it to see if the chain will ever finish at another mode.

A few of the iterations of this diagnostic can ending up in vastly different locations of the parameter space. This can happen for a variety of reasons. For instance, suppose that we created another dataset and accidently included a linear function of another variable in this dataset:

\[ R > \text{freetrade2} \leftarrow \text{freetrade} \]
\[ R > \text{freetrade2}{}^\ast \text{tariff2} \leftarrow \text{freetrade2}{}^\ast \text{tariff} \ast 2 + 3 \]
If we tried to impute this dataset, Amelia could draw imputations without any problems:

```r
R> a.out.bad <- amelia(freetrade2, ts = "year", cs = "country")
```

```
-- Imputation 1 --
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
-- Imputation 2 --
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
-- Imputation 3 --
1 2 3 4 5 6 7 8 9 10 11 12 13 14
-- Imputation 4 --
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
-- Imputation 5 --
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
```

```r
R> a.out.bad
```

Amelia output with 5 imputed datasets.
Return code: 1
Message: Normal EM convergence.

Chain Lengths:
--------------
Imputation 1: 17
Imputation 2: 16
Imputation 3: 14
Imputation 4: 16
Imputation 5: 21

But if we were to run disperse, we would end up with the problematic Figure 11:

```r
R> disperse(a.out.bad, dims = 1, m = 5)
```

While this is a special case of a problematic likelihood, situations very similar to this can go undetected without using the proper diagnostics. More generally, an unidentified imputation model will lead to non-unique maximum likelihood estimates (see King (1989) for a more detailed discussion of identification and likelihoods).
Time-series plots

As discussed above, information about time trends and fixed effects can help produce better imputations. One way to check the plausibility of our imputation model is to see how it predicts missing values in a time series. If the imputations for the Malaysian tariff rate were drastically higher in 1990 than the observed years of 1989 or 1991, we might worry that there is a problem in our imputation model. Checking these time series is easy to do with the `tscsPlot` command. Simply choose the variable (with the `var` argument) and the cross-section (with the `cs` argument) to plot the observed time-series along with distributions of the imputed values for each missing time period. For instance, we can run

```r
R> tscsPlot(a.out.time, cs = "Malaysia", var = "tariff",
+ ylim = c(-10, 60), main = "Malaysia (with time settings)"
)
```

to get the plot in Figure 12. Here, the black point are observed tariff rates for Malaysia from 1980 to 2000. The red points are the mean imputation for each of the missing values, along with their 95% confidence bands. We draw these bands by imputing each of missing values 100 times to get the imputation distribution for that observation.

In Figure 12, we can see that the imputed 1990 tariff rate is quite in line with the values around it. Notice also that values toward the beginning and end of the time series have higher imputation variance. This occurs because the fit of the polynomials of time in the imputation model have higher variance at the beginning and end of the time series. This is
intuitive because these points have fewer neighbors from which to draw predictive power.

A word of caution is in order. As with comparing the histograms of imputed and observed values, there could be reasons that the missing values are systematically different than the observed time series. For instance, if there had been a major financial crisis in Malaysia in 1990 which caused the government to close off trade, then we would expect that the missing tariff rates should be quite different than the observed time series. If we have this information in our imputation model, we might expect to see out-of-line imputations in these time-series plots. If, on the other hand, we did not have this information, we might see “good” time-series plots that fail to point out this violation of the MAR assumption. Our imputation model would produce poor estimates of the missing values since it would be unaware that both the missingness and the true unobserved tariff rate depend on another variable. Hence, the \texttt{tscsPlot} is useful for finding obvious problems in imputation model and comparing the efficiency of various imputation models, but it cannot speak to the untestable assumption of MAR.

\textit{Missingness maps}

One useful tool for exploring the missingness in a dataset is a \textit{missingness map}. This is a map that visualizes the dataset a grid and colors the grid by missingness status. The column of
Figure 13: Missingness map of the freetrade data. Missing values are in tan and observed values are in red.

the grid are the variables and the rows are the observations, as in any spreadsheet program. This tool allows for a quick summary of the patterns of missingness in the data. If we simply call the missmap function on our output from amelia,

\[ R> \text{missmap}(\text{a.out}) \]

we get the plot in Figure 13. The missmap function arrange the columns so that the variables are in decreasing order of missingness from left to right. If the cs argument was set in the amelia function, the labels for the rows will indicate where each of the cross-sections begin.

In Figure 13, it is clear that the tariff rate is the variable most missing in the data and it tends to be missing in blocks of a few observations. Gross international reserves (intresmi) and financial openness (fivop), on the other hand, are missing mostly at the end of each cross-section. This suggests missingness by merging, when variables with different temporal coverages are merged to make one dataset. Sometimes this kind of missingness is an artifact of the date at which the data was merged and researchers can resolve it by finding updated versions of the relevant variables.
The missingness map is an important tool for understanding the patterns of missingness in the data and can often indicate potential ways to improve the imputation model or data collection process.

### 4.8. Analysis Models

Imputation is most often a data processing step as opposed to a final model in of itself. To this end, it is easy to pass output from `amelia` to other functions. The easiest and most integrated way to run an analysis model is to pass the output to the `zelig` function from the `Zelig` package. For example, in Milner and Kubota (2005), the dependent variable was tariff rates. We can replicate Table 5.1 from their analysis with the original data simply by running

```R
R> require("Zelig")
R> z.out <- zelig(tariff ~ polity + pop + gdp.pc + year + country, + data = freetrade, model = "ls", cite = FALSE)
R> summary(z.out)
```

```
Call:
zelig(formula = tariff ~ polity + pop + gdp.pc + year + country, + model = "ls", data = freetrade, cite = FALSE)

Residuals:
Min 1Q Median 3Q Max
-30.7640 -3.2595 0.0868 2.5983 18.3097

Coefficients:

| Estimate | Std. Error | t value | Pr(>|t|) |
|----------|------------|---------|----------|
| (Intercept) | 1.973e+03 | 4.016e+02 | 4.912 | 3.61e-06 |
| polity | -1.373e-01 | 1.821e-01 | -0.754 | 0.453 |
| pop | -2.021e-07 | 2.542e-08 | -7.951 | 3.23e-12 |
| gdp.pc | 6.096e-04 | 7.442e-04 | 0.819 | 0.415 |
| year | -8.705e-01 | 2.084e-01 | -4.176 | 6.43e-05 |
| countryIndonesia | -1.823e+02 | 1.857e+01 | -9.819 | 2.98e-16 |
| countryKorea | -2.204e+02 | 2.078e+01 | -10.608 | < 2e-16 |
| countryMalaysia | -2.245e+02 | 2.17e+01 | -10.343 | < 2e-16 |
| countryNepal | -2.163e+02 | 2.247e+01 | -9.529 | 2.98e-16 |
| countryPakistan | -1.554e+02 | 1.982e+01 | -7.838 | 5.63e-12 |
| countryPhilippines | -2.040e+02 | 2.088e+01 | -9.774 | 3.75e-16 |
| countrySriLanka | -2.091e+02 | 2.210e+01 | -9.460 | 1.80e-15 |
| countryThailand | -1.961e+02 | 2.095e+01 | -9.358 | 2.99e-15 |

Residual standard error: 6.221 on 98 degrees of freedom
Multiple R-squared: 0.9247, Adjusted R-squared: 0.9155
F-statistic: 100.3 on 12 and 98 DF, p-value: < 2.2e-16
```

Running the same model with imputed data is almost identical. Simply replace the original data set with the imputations from the `amelia` output:
R> z.out.imp <- zelig(tariff ~ polity + pop + gdp.pc + year + country, + data = a.out$imputations, model = "ls", cite = FALSE)
R> summary(z.out.imp)

Model: ls
Number of multiply imputed data sets: 5

Combined results:

Call:
zelig(formula = tariff ~ polity + pop + gdp.pc + year + country, + model = "ls", data = a.out$imputations, cite = FALSE)

Coefficients:
                      Value Std. Error t-stat  p-value
(Intercept)        2.391742e+03 7.394054e+02 3.234683 0.004912932
polity             4.475188e-02 3.681170e-01 0.12156972 0.904072397
pop                -8.198033e-08 5.642752e-08 -1.452843 0.171140087
gdp.pc             7.329640e-05 1.625340e-03 0.04509605 0.964523211
year               -1.137377e+00 3.804230e-01 -2.989768 0.008001366
countryIndonesia -8.541117e+01 4.116937e+01 -2.074629 0.061959424
countryKorea     -1.092425e+02 4.399060e+01 -2.483314 0.026661681
countryMalaysia  -1.103849e+02 4.981854e+01 -2.256505 0.035953333
countryNepal     -1.080707e+02 4.765556e+01 -2.266318 0.024191714
countryPakistan  -6.122513e+01 4.453178e+01 -1.374636 0.171400877
countryPhilippines -9.968491e+01 4.664926e+01 -2.136921 0.055662060
countrySriLanka  -9.677452e+01 5.008941e+01 -1.932035 0.053599991
countryThailand  -9.499329e+01 4.519723e+01 -2.101750 0.057144006

For combined results from datasets i to j, use summary(x, subset = i:j).
For separate results, use print(summary(x), subset = i:j).

Zelig is one way to run analysis models on imputed data, but certainly not the only way. The imputations list in the amelia output contains each of the imputed datasets. Thus, users could simply program a loop over the number of imputations and run the analysis model on each imputed dataset and combine the results using the rules described in King et al. (2001) and Schafer (1997). Furthermore, users can easily export their imputations using the write.amelia function as described in Section 4.2 and use statistical packages other than R for the analysis model.

4.9. The amelia class

The output from the amelia function is an instance of the S3 class amelia. Instances of the amelia class contain much more than simply the imputed datasets. The mu object of the class contains the posterior draws of the means of the complete data. The covMatrices contains the posterior draws of the covariance matrices of the complete data. Note that these correspond to the variables as they are sent to the EM algorithm. Namely, they refer to the variables after being transformed, centered and scaled.
The `iterHist` object is a list of \( m \) 3-column matrices. Each row of the matrices corresponds to an iteration of the EM algorithm. The first column indicates how many parameters had yet to converge at that iteration. The second column indicates if the EM algorithm made a step that decreased the number of converged parameters. The third column indicates whether the covariance matrix at this iteration was singular. Clearly, the last two columns are meant to indicate when the EM algorithm enters a problematic part of the parameter space.

5. AmeliaView menu guide

Below is a guide to the AmeliaView menus with references back to the user’s guide. The same principles from the user’s guide apply to AmeliaView. The only difference is how you interact with the program. Whether you use the GUI or the command line versions, the same underlying code is being called, and so you can read the command line-oriented discussion above even if you intend to use the GUI.

5.1. Loading AmeliaView

The easiest way to load AmeliaView is to open an R session and type the following two commands which will bring up the AmeliaView window on any platform.

\begin{verbatim}
R> library("Amelia")
R> AmeliaView()
\end{verbatim}

On the Windows operating system, there is an alternative way to start AmeliaView from the Desktop. See Section 3.2 for a guide on how to install this version. Once installed, there should be a Desktop icon for AmeliaView. Simply double-click this icon and the AmeliaView window should appear. If, for some reason, this approach does not work, simply open an R session and use the approach above.

5.2. Loading a data set into AmeliaView

AmeliaView load with a welcome screen (Figure 14) that has buttons which can load a data in many of the common formats. Each of these will bring up a window for choosing your dataset. Note that these buttons are only a subset of the possible ways to load data in AmeliaView. Under the File menu (shown in Figure 15), you will find more options, including the datasets included in the package (africa and freetrade). You will also find import commands for comma-separated values (.csv), tab-delimited text (.txt), Stata 5–10 (.dta), SPSS (.dat), and SAS transport (.xport). Note that when using a .csv file, Amelia assumes that your file has a header (that is, a row at the top of the data indicating the variable names). You can also load data from an .RData file. If the RData file contains more than one data.frame, a pop-up window will ask to you find the dataset you would like to load. In the file menu, you can also change the underlying working directory. This is where AmeliaView will look for data by default and where it will save imputed datasets.

5.3. Variable dashboard

Once a dataset is loaded, AmeliaView will show the variable dashboard (Figure 16). In this mode, you will see a table of variables, with the current options for each of them shown,
Amelia II: A Program for Missing Data

Figure 14: **AmeliaView** welcome screen.

Figure 15: **AmeliaView** file and import menu.

along with a few summary statistics. You can reorder this table by any of these columns by clicking on the column headings. This might be helpful to, say, order the variables by mean or amount of missingness.

You can set options for individual variables by the right-click context menu (Figure 17) or through the **Variables** menu. For instance, clicking **Set as Time-Series Variable** will set the currently selected variable in the dashboard as the time-series variable. Certain options are disabled until other options are enabled. For instance, you cannot add a lagged variable to the imputation until you have set the time-series variable. Note that any **factor** in the data is marked as a ID variable by default, since a **factor** cannot be included in the imputation.
without being set as an identification variable, a nominal variable, or the cross-section variable. If there is a factor that fails to meet one of these conditions, a red flag will appear next to the variable name.
1. **Set as Time-Series Variable**: Sets the currently selected variable to the time-series variable. Disabled when more than one variable is selected. Once this is set, you can add lags and leads and add splines of time. The time-series variable will have a clock icon next to it.

2. **Set as Cross-Section Variable**: Sets the currently selected variable to the cross-section variable. Disabled when more than one variable is selected. Once this is set, you can interact the splines of time with the cross-section. The cross-section variable will have a person icon next to it.

3. **Unset as Time-Series Variable**: Removes the time-series status of the variable. This will remove any lags, leads, or splines of time.

4. **Unset as Cross-Section Variable**: Removes the cross-section status of the variable. This will remove any intersection of the splines of time and the cross-section.

5. **Add Lag/Lead**: Adds versions of the selected variables either lagged back ("lag") or forward ("lead"). See Section 4.5 above.

6. **Remove Lag/Lead**: Removes any lags or leads on the selected variables.

7. **Plot Histogram of Selected**: Plots a histogram of the selected variables. This command will attempt to put all of the histograms on one page, but if more than nine histograms are requested, they will appear on multiple pages.

8. **Add Transformation**: Adds a transformation setting for the selected variables. Note that each variable can only have one transformation and the time-series and cross-section variables cannot be transformed.

9. **Remove Transformation**: Removes any transformation for the selected variables.

10. **Add or Edit Bounds**: Opens a dialog box to set logical bounds for the selected variable.

### 5.4. Amelia options

The Variable menu and the variable dashboard are the place to set variable-level options, but global options are set in the **Options** menu.

1. **Splines of Time with...**: This option, if activated, will have Amelia use flexible trends of time with the specified number of knots in the imputation. The higher the number of knots the greater the variation in the trend structure, yet it will take more degrees of freedom to estimate. For more information see Section 4.5 above.

2. **Interact with Cross-Section?**: Include and interaction of the cross-section with the time trends. This interaction is way of allowing the trend of time to vary across cases as well. Using a 0-level spline of time and interacting with the cross section is the equivalent of using a fixed effects. For more information see Section 4.5 above.

3. **Add Observational Priors...**: Brings a dialog window to set prior beliefs about ranges for individual missing observations. For more information about observational priors, see Section 4.6.
4. **Numerical Options**: Brings a dialog window to set the tolerance of the EM algorithm, the seed of the random number generator, the ridge prior for numerical stability, and the maximum number of redraws for the logical bounds.

5. **Draw Missingness Map**: Draws a missingness map. See Section 4.7.5 for more details on missingness maps.

6. **Output File Options**: Bring a dialog to set the stub of the prefix of the imputed data files and the number of imputations. If you set the prefix to `mydata`, your output files will be `mydata1.csv`, `mydata2.csv`... etc.

7. **Output File Type**: Sets the format of imputed data. If you would like to not save any output data sets (if you wanted, for instance, to simply look at diagnostics), set this option to `(no save)`. Currently, you can save the output data as: Comma-separated values, tab-delimited text, Stata, R save object (.RData), or to hold it in R memory. This last option will only work if you have called AmeliaView from an R session and want to return to the R command line to work with the output. Its name in R workspace will be the file prefix.

### Numerical options

1. **Seed**: Sets the seed for the random number generator used by Amelia. Useful if you need to have the same output twice.

2. **Tolerance**: Adjust the level of tolerance that Amelia uses to check convergence of the EM algorithm. In very large datasets, if your imputation chains run a long time without converging, increasing the tolerance will allow a lower threshold to judge convergence and end chains after fewer iterations.

3. **Empirical Prior**: A prior that adds observations to your data in order to shrink the covariances. A useful place to start is around 0.5% of the total number of observations in the dataset (see Section 4.6).
Amelia II: A Program for Missing Data

4. **Maximum Resample for Bounds**: Amelia fits logical bounds by rejecting any draws that do not fall within the bounds. This value sets the number of times Amelia should attempt to resample to fit the bounds before setting the imputation to the bound.

*Add distribution prior*

1. **Current Priors**: A table of current priors in distributional form, with the variable and case name. You can remove priors by selecting them and using the right-click context menu.

2. **Case**: Select the case name or number you wish to set the prior about. You can also choose to make the prior for the entire variable, which will set the prior for any missing
cell in that variable. The case names are generated from the row name of the observation, the value of the cross-section variable of the observation and the value of the time series variable of the observation.

3. **Variable**: The variable associated with the prior you would like specify. The list provided only shows the missing variables for the currently selected observation.

4. **Mean**: The mean value of the prior. The textbox will not accept letters or out of place punctuation.

5. **Standard Deviation**: The standard deviation of the prior. The textbox will only accept positive non-zero values.

**Add range prior**

1. **Case**: Select the case name or number you wish to set the prior about. You can also choose to make the prior for the entire variable, which will set the prior for any missing cell in that variable. The case names are generated from the row name of the observation, the value of the cross-section variable of the observation and the value of the time series variable of the observation.

2. **Variable**: The variable associated with the prior you would like specify. The list provided only shows the missing variables for the currently selected observation.

3. **Minimum**: The minimum value of the prior. The textbox will not accept letters or out of place punctuation.
4. **Maximum**: The maximum value of the prior. The textbox will not accept letters or out of place punctuation.

5. **Confidence**: The confidence level of the prior. This should be between 0 and 1, non-inclusive. This value represents how certain your priors are. This value cannot be 1, even if you are absolutely certain of a give range. This is used to convert the range into an appropriate distributional prior.

### 5.5. Imputing and checking diagnostics

Once you have set all the relevant options, you can impute your data by clicking the **Impute!** button in the toolbar. In the bottom right corner of the window, you will see a progress bar that indicates the progress of the imputations. For large datasets this could take some time. Once the imputations are complete, you should see a **Successful Imputation!** message appear where the progress bar was. You can click on this message to open the folder containing the imputed datasets.

If there was an error during the imputation, the output log will pop-up and give you the error message along with some information about how to fix the problem. Once you have fixed the problem, simply click **Impute!** again. Even if there was no error, you may want to view the
output log to see how Amelia ran. To do so, simply click the Show Output Log button. The log also shows the call to the amelia function in R. You can use this code snippet to run the same imputation from the R command line.\(^\text{10}\)

**Diagnostics dialog**

Upon the successful completion of an imputation, the diagnostics menu will become available. Here you can use all of the diagnostics available at the command-line.

1. **Compare Plots**: This will display the relative densities of the observed (red) and imputed (black) data. The density of the imputed values are the average imputations across all of the imputed datasets.

2. **Overimpute**: This will run Amelia on the full data with one cell of the chosen variable artificially set to missing and then check the result of that imputation against the truth. The resulting plot will plot average imputations against true values along with 90% confidence intervals. These are plotted over a $y = x$ line for visual inspection of the imputation model.

3. **Number of overdispersions**: When running the overdispersion diagnostic, you need to run the imputation algorithm from several overdispersed starting points in order to get a clear idea of how the chain are converging. Enter the number of imputations here.

4. **Number of dimensions**: The overdispersion diagnostic must reduce the dimensionality of the paths of the imputation algorithm to either one or two dimensions due to graphical restraints.

\(^{10}\text{You will have to replace the x argument in the amelia call to the name of you dataset in the R session.}\)
5. **Overdisperse**: Run overdispersion diagnostic to visually inspect the convergence of the *Amelia* algorithm from multiple start values that are drawn randomly.

5.6. **Sessions**

It is often useful to save a session of *AmeliaView* to save time if you have impute the same data again. Using the **Save Session** button will do just that, saving all of the current settings (including the original and any imputed data) to an RData file. You can then reload your session, on the same computer or any other, simply by clicking the **Load Session** button and finding the relevant RData file. All of the settings will be restored, including any completed imputations. Thus, if you save the session after imputing, you can always load up those imputations and view their diagnostics using the sessions feature of *AmeliaView*.

**References**


York.


StataCorp (2011). *Stata Data Analysis Statistical Software: Release 12*. StataCorp LP, College
Station, TX. URL http://www.stata.com/.

Affiliation:

James Honaker
Department of Political Science
The Pennsylvania State University
Pond Laboratory
University Park PA 16802, United States of America
Email: tercer@psu.edu

Gary King, Matthew Blackwell
Institute for Quantitative Social Science
Harvard University
1737 Cambridge Street
Cambridge, MA 02138, United States of America
Email: King@Harvard.edu, mblackwell@iq.harvard.edu