Using Multiple Imputation to Simulate Time Series

SEBASTIAN CANO
Universitat Rovira i Virgili
Departament d’Economia
Av. Universitat 1, 43204 Reus
SPAIN
sebastian.cano@urv.net

JORDI ANDREU
Universitat Rovira i Virgili
Departament de Gestió d’Empreses
Av. Universitat 1, 43204 Reus
SPAIN
jordi.andreuc@urv.net

Abstract: Multiple Imputation is a Markov chain Monte Carlo technique developed to work out missing data problems. This paper proposes a different point of view to use this technique with time series. The authors’ idea consists on an endogenous construction of the database to avoid noise in the simulations and reach the right convergence of the limit distribution of the chain. New computer code was designed to carry out all the simulations.


1 Introduction

Complex probability distributions, where the number of dimensions were a serious issue, were solved by physicists by simulation instead of direct calculation. This research was published in 1953 [7]. In that paper Metropolis Algorithm was presented (later generalized by Hastings [6]). The innovation of the commented algorithm was the use of Markov chains to look for probability distributions, making the target distribution the limit distribution of the chain. In the early years, MCMC was only developed theoretically due to the lack of computational power to run the algorithms. But since the late 1980s the huge development of computers allowed the use of MCMC again.

MCMC implied a real revolution in multiple fields, not only in physics. We may find them in Mechanical Statistics, Bayesian Statistics or Reconstruction Image Theory for example. Within Bayesian Statistics, MCMC was implemented to solve missing data problems, see [8]. Missing values represent an issue to analyze a database because one cannot use the standard analysis tools directly. To overcome this, Rubins idea was to combine MCMC algorithms such as EM, Data Augmentation or Gibbs Sampling to approximate the joint probability distribution of missing data and observed data. After this calculation, m plausible values are simulated to fill in every empty cell. Later on, these multiplicity is combined following special inference rules. This statistical technique is known as Multiple Imputation (MI).

The literature about Multiple Imputation has been focused on cross section studies, where missing data is more likely to appear. In [1] the authors test MI with financial time series paying attention on how simulations change when one varies the main parameters of the technique. After many simulations a distance effect was found: the longer the time series length the worse results of the simulation. Indeed, the problem was an inappropriate design of the database. A wrong data structure leads to a faulty convergence of the algorithm, and therefore to non plausible simulations. So, a new perspective must be considered to improve the performance of MI with time series. In this paper we use lags of a time series as supporting variables of the main series. Doing the simulations in this fashion lead to better results but cause even higher multiplicity (more plausible values) which need to be pooled.

In this research we explain the evolution of our approach, and provide some evidences with new simulations. The structure of the paper is as follows: in section 2 we review the main points of MCMC and Multiple Imputation. In section 3 the problem is defined. Section 4 presents a different point of view that lead to new simulations. Finally, in section 5 we shows results of some simulations and section 6 draws the main conclusion of this paper.

2 Methodology Review

2.1 Markov chain Monte Carlo

Markov Chain Monte Carlo (hereby MCMC) is a collection of methods to generate pseudorandom numbers via Markov Chains. MCMC works constructing a Markov chain which steady-state is the distribution of interest. Random Walks Markov are closely attached to MCMC. Indeed, this makes a division within the classification of MCMC algorithms. The well known Metropolis-Hastings and Gibbs Sampling are part of...
the Random Walk algorithms and their success depends on the number of iterations needed to explore the space, meanwhile the Hybrid Monte Carlo tries to avoid the random walk using hamiltonian dynamics.

The literature related to MCMC has raised in the last decades due to the improvement of computational tools\(^1\). Following that improvements, new fields for these methods have been discovered. For example one can find applications in Statistical Mechanics, Image Reconstruction and Bayesian Statistics.

### 2.2 Gibbs Sampling

Gibbs Sampling, named after Josiah Willard Gibbs, is an MCMC algorithm created by Geman and Geman in 1984 [5]. Due to its simplicity is a common option for those who implement MI in a software package. Furthermore, Gibbs Sampling has had a vital importance in the later development of Bayesian Inference thanks to BUGS software (Bayesian Inference Using Gibbs Sampling). Owing to this fact, some authors have suggested to rename the algorithm to Bayesian Sampling. The process of the algorithm is as follows: let \( \pi(\theta) \) be the target distribution where \( \theta = (\theta_1, \theta_2, \ldots, \theta_d) \). Also let \( \pi_i(\theta_i) = \pi(\theta_i|\theta_{-i}) \) be the conditional distributions for \( i = 1, 2, \ldots, d \). Then, if the conditional distributions are available, we may approximate \( \pi(\theta) \) through an iterative process. Gibbs Sampling is performed by 3 steps,

1. Choose the initial values \( \theta^{(0)} = (\theta_1^{(0)}, \theta_2^{(0)}, \ldots, \theta_d^{(0)}) \) at the moment \( j \).

2. Calculate a new value of \( \theta^{(j)} \) from \( \theta^{(j-1)} \) by the following process,

\[
\begin{align*}
\theta_1^{(j)} & \sim \pi(\theta_1|\theta_2^{(j-1)}, \ldots, \theta_d^{(j-1)}) \\
\theta_2^{(j)} & \sim \pi(\theta_2|\theta_1^{(j)}, \ldots, \theta_d^{(j-1)}) \\
& \vdots \\
\theta_d^{(j)} & \sim \pi(\theta_d|\theta_1^{(j)}, \ldots, \theta_d^{(j-1)})
\end{align*}
\]

3. Change the counter from \( j \) to \( j + 1 \) and go to the second step until the convergence is reached.

### 2.3 Multiple Imputation

The presence of missingness is an issue to process data. Every empty cell is represented by software with \( na \) which cannot be treated until is replaced by a number. In such scenario the literature has developed many approaches to this problem: Case Deletion, Single Imputation and the more complex Multiple Imputation. For a brief introduction to this technique see [9] and [11]. For a more detailed description [8] and [10].

Multiple Imputation is a MCMC technique which tries to solve missing data problems in a different fashion. Instead of calculating the values of missing values directly, it carries many simulations to achieve plausible values. After that, there are many plausible values for every missing datum. This multiplicity of information needs to be summarized, and special rules of inference are defined to pool the results.\(^2\). MI is a 3 stage process:

**imputation:** The number \( m \) of imputations is set. The probability distribution \( Pr(X_{mis}|X_{obs}) \) is approximated through MCMC algorithms. Later on it will be used to Monte Carlo simulations.

**analysis:** Every simulated data set is analyzed using standard methods.

**pool:** At this point \( m \) results are available. They are combined with special inference rules.

This technique performs fine when the data missing mechanism is random. To see that, the probability distribution of the dummy \( R \) (it represents the missing data pattern) has to be analyzed. To do so, the connection between \( R \) (known information), missing information of the sample and a nuisance parameter \( \xi \) is studied through conditional probability,

\[
Pr(R|X_{obs}, X_{mis}, \xi)
\]

In case we have \( R \) \( X_{mis} = 0 \) the missing data process is considered to be random. Nowadays, this analysis lacks a formal test to be sure about the missing data process.

### 3 Definition of the problem

Andreu and Cano (2008) [1] perform some tests with Multiple Imputation and time series. The authors designed a database with prices of 10 stocks of DJIA.\(^3\) The paper shows different results when changing the number of imputations, the length and the number of iterations. The good point is that only a small number

\(^1\)see [3] and [4]

\(^2\)This inference calculates the uncertainty of the missing data via degrees of freedom.

\(^3\)Alcoa Inc (AA), Boeing Co (BA), Caterpillar Inc (CAT), Dupont (DD), Walt Disney (DIS), General Electric (GE), General Motors (GM), Hewlett Packard (HPQ), IBM and CocaCola (KO), 541 observations for monthly data, 2347 observations for weekly data and 11328 for daily data are available.
of iterations is needed to reach the convergence. The most important conclusion the authors draw from the study is a distance effect. When the size of the time series grows simulations smoothens the value. The error shows that from the 5th or 6th missing, the simulation is not a plausible value, because the Markov chain does not converge to the right limit distribution. Comparing both series (actual and simulated), one can see an early structural break. Authors found that the design of the database is the key of the issue. First the limit distribution of the chain is faulty. Second, every time series is adding noise to simulations. The new perspective should avoid this two problems.

4 A new point of view

Multiple Imputation has been designed for working on cross section databases. Making a design close to a cross section appearance seems not to work. A new point of view is needed in order to use the technique with time series. In the new approach 2 issues need to be considered:

1. Proper construction of the Markov chain.
2. Noise from other variables of the database.

Let’s see a normal time series \( X \), which is a matrix with \( t \) rows and one column,

\[
X = \begin{pmatrix}
    x_1 \\
    x_2 \\
    x_3 \\
    \vdots \\
    x_t
\end{pmatrix}
\]

One can add an auxiliary variable to the matrix, which is actually the first lag of \( X \). We call the new data structure \( X^* \), it has the shape,

\[
X^* = \begin{pmatrix}
    x_2 & x_1 \\
    x_3 & x_2 \\
    x_4 & x_3 \\
    \vdots & \vdots \\
    x_t & x_{t-1}
\end{pmatrix}
\]

Arranging the time series in this fashion we let the values of the past influence the recent values. One can add as many artificial variables he may consider. Now let’s think we have a missing value in our time series, 4

\[
X = \begin{pmatrix}
    x_1 \\
    x_2 \\
    na \\
    x_4 \\
    x_5 \\
    x_6 \\
    x_7 \\
    \vdots \\
    x_t
\end{pmatrix}
\]

One build the matrix \( X^* \) with two artificial variables,

\[
X^* = \begin{pmatrix}
    na & x_2 & x_1 \\
    x_4 & na & x_2 \\
    x_5 & x_4 & na \\
    \vdots & \vdots & \vdots \\
    x_t & x_{t-1} & x_{t-2}
\end{pmatrix}
\]

Notice that now the missing value appears 3 times and is across one diagonal of the matrix. In a more complete case one might have a matrix like,

\[
X^* = \begin{pmatrix}
    x_{t-2} & x_{t-3} & x_{t-4} & x_{t-5} \\
    x_{t-1} & x_{t-2} & x_{t-3} & x_{t-4} \\
    na & x_{t-1} & x_{t-2} & x_{t-3} \\
    na & na & x_{t-1} & x_{t-2} \\
    na & na & na & x_{t-1}
\end{pmatrix}
\]

Here one can identify 2 different submatrix, one is known information and the other one is missing information. Some considerations about the triangular matrix:

- if convergence is reached, values of the diagonal should be closer.
- the best simulation should be the one with more supporting information.
- when the missing is far away from the known information uncertainty will grow.

Now there are many plausible values which have to be pooled using Rubin’s inference. The scalar of interest (\( Q \)) will be the value of the missing cell we are looking for. First we need calculate the average value of the scalar of interest,

\[
\bar{Q} = \frac{1}{m} \sum_{t=1}^{m} \hat{Q}(t)
\]

and the total variance associated to \( Q \),

\[\text{Variance} = \frac{1}{m} \sum_{t=1}^{m} (Q(t) - \bar{Q})^2\]
\[ T = \frac{1}{m} \sum_{t=1}^{m} \hat{S}(t) + \left( 1 + \frac{1}{m} \right) \sum_{t=1}^{m} \frac{(\hat{Q}(t) - \bar{Q})^2}{m-1} \] (3)

Next step is to calculate the degrees of freedom for a small sample to carry out the inference,

\[ \frac{dfc}{(1-f)} - \frac{f - f^2}{m+1} \] (4)

After all this process inference based on \( t \) distribution can be calculated,

\[ T^{0.5}(Q - \bar{Q}) \sim t_{df} \] (5)

5 Empirical tests

To illustrate what this paper has explained we make some simulations with different time series. We use the \texttt{R} language to do the programming and to perform test. We use a library named \texttt{multiple imputation simulation for time series}.

There are 2 main commands in the library:

\texttt{mists()} performs the whole Multiple Imputation process and builds the \( X^* \) matrix. The sintax is: \texttt{mists(data, iterations, number of data simulations, number of artificial variables)}.

\texttt{rubin.value()} pools the simulations for each missing value and makes the inference calculation. The sintax is: \texttt{rubin.value(object, missing number to make the inference)}.

The tests have the following structure: first of all we run simulations with the \texttt{mists()} instruction and second we make the inference over each simulated value using 95% confidence level.

5.1 Test 1


\texttt{mists(IBM,50,c(253:260),10)}

Results in the following table,

<table>
<thead>
<tr>
<th>Actual Q</th>
<th>Lower</th>
<th>Upper</th>
<th>Df</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>126,6</td>
<td>126,17</td>
<td>121,73</td>
<td>22</td>
<td>0.003</td>
</tr>
<tr>
<td>125,22</td>
<td>124,63</td>
<td>117,41</td>
<td>10</td>
<td>0.005</td>
</tr>
<tr>
<td>125,8</td>
<td>123,23</td>
<td>116,08</td>
<td>10</td>
<td>0.020</td>
</tr>
<tr>
<td>126,94</td>
<td>121,96</td>
<td>114,58</td>
<td>8</td>
<td>0.039</td>
</tr>
<tr>
<td>126,36</td>
<td>121,2</td>
<td>111,26</td>
<td>5</td>
<td>0.041</td>
</tr>
<tr>
<td>124,59</td>
<td>120,85</td>
<td>108,36</td>
<td>4</td>
<td>0.030</td>
</tr>
<tr>
<td>122,56</td>
<td>119,88</td>
<td>108,93</td>
<td>4</td>
<td>0.022</td>
</tr>
</tbody>
</table>

5.2 Test 2


\texttt{mists(Apple,50,c(253:260),10)}

Results in the following table,

<table>
<thead>
<tr>
<th>Actual Q</th>
<th>Lower</th>
<th>Upper</th>
<th>Df</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>173,56</td>
<td>170,24</td>
<td>163,72</td>
<td>31</td>
<td>0.020</td>
</tr>
<tr>
<td>176,73</td>
<td>170,30</td>
<td>164,36</td>
<td>27</td>
<td>0.036</td>
</tr>
<tr>
<td>179,30</td>
<td>168,99</td>
<td>162,22</td>
<td>16</td>
<td>0.058</td>
</tr>
<tr>
<td>179,32</td>
<td>168,52</td>
<td>161,45</td>
<td>19</td>
<td>0.060</td>
</tr>
<tr>
<td>175,74</td>
<td>169,27</td>
<td>161,67</td>
<td>12</td>
<td>0.037</td>
</tr>
<tr>
<td>175,39</td>
<td>167,66</td>
<td>151,89</td>
<td>4</td>
<td>0.044</td>
</tr>
<tr>
<td>173,53</td>
<td>167,55</td>
<td>138,48</td>
<td>2</td>
<td>0.034</td>
</tr>
</tbody>
</table>

5.3 Test 3


\texttt{mists(aqua,50,c(1:9),14)}

Results in the following table,

<table>
<thead>
<tr>
<th>Actual Q</th>
<th>Lower</th>
<th>Upper</th>
<th>Df</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>17,7</td>
<td>18,14</td>
<td>16,52</td>
<td>67</td>
<td>-0.025</td>
</tr>
<tr>
<td>17,8</td>
<td>18,13</td>
<td>16,48</td>
<td>63</td>
<td>-0.019</td>
</tr>
<tr>
<td>17,7</td>
<td>18,11</td>
<td>16,41</td>
<td>59</td>
<td>-0.023</td>
</tr>
<tr>
<td>17,5</td>
<td>18,10</td>
<td>16,28</td>
<td>45</td>
<td>-0.034</td>
</tr>
<tr>
<td>18,6</td>
<td>18,01</td>
<td>16,10</td>
<td>47</td>
<td>0.032</td>
</tr>
<tr>
<td>18,3</td>
<td>18,23</td>
<td>16,31</td>
<td>43</td>
<td>0.004</td>
</tr>
<tr>
<td>18,2</td>
<td>18,34</td>
<td>16,36</td>
<td>39</td>
<td>-0.008</td>
</tr>
<tr>
<td>18,1</td>
<td>18,42</td>
<td>16,41</td>
<td>35</td>
<td>-0.018</td>
</tr>
<tr>
<td>18,6</td>
<td>18,38</td>
<td>16,36</td>
<td>34</td>
<td>0.012</td>
</tr>
</tbody>
</table>

\footnote{This library has been programmed for the unpublished PhD Thesis “Imputación Múltiple: definición y aplicaciones”, see \cite{2}.}

\footnote{We have simulated missing values this approac for different time series. Only 4 examples are provided here to show the application of this technique.}
5.4 Test 4

\[ \text{mists(usq,50,c(1:5),9)} \]
Results in the following table,

<table>
<thead>
<tr>
<th>Actual</th>
<th>Q</th>
<th>Lower</th>
<th>Upper</th>
<th>Df</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.680</td>
<td>0.719</td>
<td>0.320</td>
<td>1.118</td>
<td>25</td>
<td>-0.057</td>
</tr>
<tr>
<td>0.652</td>
<td>0.722</td>
<td>0.332</td>
<td>1.121</td>
<td>20</td>
<td>-0.107</td>
</tr>
<tr>
<td>0.647</td>
<td>0.712</td>
<td>0.308</td>
<td>1.101</td>
<td>15</td>
<td>-0.100</td>
</tr>
<tr>
<td>0.616</td>
<td>0.710</td>
<td>0.290</td>
<td>1.134</td>
<td>10</td>
<td>-0.153</td>
</tr>
<tr>
<td>0.623</td>
<td>0.718</td>
<td>0.234</td>
<td>1.201</td>
<td>5</td>
<td>-0.152</td>
</tr>
</tbody>
</table>

6 Conclusion
Multiple Imputation is a MCMC technique developed to work out missing data problems via simulation of \( m \) simulated values. After this step, special inference rules are applied to calculate the uncertainty of missing data over the scalar of interest (it may be any value of the model or individual cells of the database). MI is basically used with cross-sectional data. In [1] simulations depend on the time series length, concluding a distance effect.

In this article we have given a further step using Multiple Imputation on time series. MI may be successfully applied on time series using the right data structure. Particularly, in this research an endogenous perspective has been used to design the database. This strategy leads to a better construction of the Markov chain and avoids the noise in the simulations. Focusing on the performed tests, the actual value is inside the confidence interval, and \( Q \) is quite close to the real value. We can see things to improve: data frequency really matters; more frequency improve simulations (best results are on water temperature, worst results are on annual frequency). That means more uncertainty on the simulations, so the lose of freedom degrees is quite noticeable. So, more effort has to be put on this side.

References:


Appendix

\[
\text{CODE FOR THE INSTRUCTION mists()} \]
1: \text{mists <- function(x,y,z,l)}
2: \text{require(mice)}
3: \text{x -> data}
4: \text{embed(data,l) -> MRT}
5: \text{x[z] <- NA}
6: \text{embed(x,l) -> MR}
7: \text{mice(MR,maxit=y) -> MRS}
8: \text{complete(MRS,1) -> M1}
9: \text{complete(MRS,2) -> M2}
10: \text{complete(MRS,3) -> M3}
11: \text{complete(MRS,4) -> M4}
12: \text{complete(MRS,5) -> M5}
13: \text{rbind(M1[z,l],M2[z,l],M3[z,l],M4[z,l],M5[z,l]) -> Valor}
14: \text{data.frame(Valor) -> V}
15: \text{names(Valor) -> P}
16: midsobj←list(
17:  data=(data[z]),
18:  imp=MRS,
19:  M1=as.matrix(M1[z,1:l]),
20:  M2=as.matrix(M2[z,1:l]),
21:  M3=as.matrix(M3[z,1:l]),
22:  M4=as.matrix(M4[z,1:l]),
23:  M5=as.matrix(M5[z,1:l]),
24: return(midsobj)
25: cat(agm(Valor))
26: }

CODE FOR THE INSTRUCTION RUBIN.VALUE()
1: rubin.value=function(x,y){
2:  mdm(x$M1)→c1
3:  mdm(x$M2)→c2
4:  mdm(x$M3)→c3
5:  mdm(x$M4)→c4
6:  mdm(x$M5)→c5
7:  sddm(x$M1)→cp1
8:  sddm(x$M2)→cp2
9:  sddm(x$M3)→cp3
10: sddm(x$M4)→cp4
11: sddm(x$M5)→cp5
12: rbind([i]$STATSM[y])→mm
13: rbind([p]$STATSSD[y])→ss
14: dim(x$M1)→D
15: D1←D+1-y
16: M1inference(mm,ss,D1)→inference
17: print(inference)
18: list(inference=inference)
19: }