

# Multiple Imputation in the Presence of High-dimensional data

Yi Deng  
Yize Zhao  
Qi Long

September 22, 2015

## 1 Introduction

Missing data are often encountered in medical and epidemiological research and present challenges for data analysis. There are numerous reasons for missing data; it could be due to the declination of a subject or to errors made during data collection. It is well known that inadequate handling of missing data may lead to biased estimation and inference. A number of statistical methods have been developed for handling missing data. Largely due to its ease of use, multiple imputation (MI) has been arguably the most popular method for handling missing data in practice. The basic idea underlying MI is to replace each missing data point with a set of values drawn from its predictive distribution given observed data and generate multiply imputed datasets to account for uncertainty of imputation. Each imputed data set is then analyzed separately using standard complete-data analysis methods and the results are combined across all imputed data sets using Rubin's rule.

While methods and applications of MI are well implemented in some existing softwares and R packages, no package has been focused on MI in the presence of high-dimensional data. In real world, we might encounter data with dimensions ranging from a few dozen to several thousand. For instance, DNA microarrays measure the expression levels of large numbers of genes simultaneously, which lead to the fact that the number of variables  $p$  is much larger than the number of observations. Existing methods and their software packages of MI cannot handle the 'p larger than n' issue and were shown to have poor performance for other high-dimensional data in numerical studies.

Zhao and Long (2013) developed three approaches for multiple imputations in the presence of high-dimensional data of univariate missing patterns. In their paper, regularized regressions are applied in order to fit a penalized imputation model. We implement and include their algorithms in this package. In addition, we extend their methods to be applicable for general missing data patterns based on the idea of multiple imputation by chained equations (MICE).

In this document, we will give a short tutorial on using the functions in the package to conduct imputations for incomplete data sets with univariate or general missing patterns. In Section 2, we illustrate three datasets that have univariate missing patterns with one single partially observed variable following gaussian, binary, and poisson distributions respectively. We give examples of using our functions to conduct multiple imputation for different types of data set. In Section 3, synthetic data set is presented and is used to illustrate multiple imputation for general missing data patterns in the presence of high-dimensional data. All functions offer

three regularized regression to choose: lasso, elastic net, adaptive lasso. Also, bayesian lasso is implemented in the case of univariate missing patterns.

## 2 Univariate missing pattern

The section presents three examples incorporating three types of data sets that have univariate missing patterns. After installing the R package **MIHD** as well as its dependency packages, load the package.

```
> library('MIHD')
```

This document uses the features of **MIHD** 3.0, which contains three data sets for univariate missing patterns. Missing values are represented as **NA**. We first load these three data sets so that they can be used for imputations.

```
> data(datagaussian)
> data(databinary)
> data(datapoisson)
```

Take the data frame **datagaussian** for instance. Missing values appear only in the first column, which corresponds to a partially observed variable following gaussian distribution. **datagaussian** has 100 subjects and 1001 variables, thus can be viewed as a high-dimensional data.

```
> sum(is.na(datagaussian))
[1] 35

> sum(is.na(datagaussian))/dim(datagaussian)[1]
[1] 0.35
```

Among 100 subjects, 35 (35%) of them are missing their first column values. With the help of **MIdurr** and **MIiurr** functions in this packages, we could impute these missing values.

Creating imputations can be done with a call to **MIdurr()** as follows”

```
> imp=MIdurr(data=datagaussian,method="lasso",family="gaussian",m=5)
```

where the multiply imputed data set is stored in the object **imp**. Inspect what the result looks like

```
> imp$n.missing
[1] 35

> imp$col.missing
[1] 1

> head(imp$impute)
      [,1]      [,2]      [,3]      [,4]      [,5]
[1,] -6.1929729 -6.1929729 -6.1929729 -6.1929729 -6.1929729
[2,] -1.1581383 -1.1581383 -1.1581383 -1.1581383 -1.1581383
[3,] -6.9837959  1.6403705  1.4277930  0.5273947 -9.5134948
[4,]  0.5795488  0.5795488  0.5795488  0.5795488  0.5795488
[5,] 16.1040824 16.1040824 16.1040824 16.1040824 16.1040824
[6,] -19.3543021 -19.3543021 -19.3543021 -19.3543021 -19.3543021
```

In the output, `n.missing` means the number of missing values in the data. `col.missing` gives the value of which column contains the missing values. The result shows that the first column has missing values. Imputations are generated according to the lasso regression for gaussian outcome. `impute` gives us imputations matrix with column number equals to `m`. Note, each column is an imputation.

We can also conduct multiple imputations for other types of data using different methods of regularized regressions.

```
> MIdurr(data=datagaussian,method="adaptive lasso",family="gaussian")
> MIdurr(data=datagaussian,method="blasso",family="gaussian")
> MIdurr(data=databinary,method="lasso",family="binary")
> MIdurr(data=databinary,method="adaptive lasso",family="binary")
> MIdurr(data=databinary,method="blasso",family="binary")
```

Another approach is IURR:

```
> MIiurr(data=datagaussian,method="lasso",family="gaussian")
> MIiurr(data=databinary,method="adaptive lasso",family="binary")
```

### 3 General missing pattern

In this section, we give a complete example of analyze an incomplete high-dimensional data with general missing pattern following three steps: multiple imputations using DURR or IURR; converting the imputation results into `mids` object; analysis of converted data. To illustrate these three steps, we first load our synthetic data set `GMdata`

```
> data(GMdata)
```

#### 3.1 Creating imputations

Similar to what we did in the last section, we can create imputations with a call to `GMdurr` if we apply DURR method (IURR can be done very similarly).

```
> GM.imp=GMdurr(data=GMdata)
```

The default number of multiple imputations is 5. We burn 20 iterations by default for the consideration of convergence.

#### 3.2 Converting into `mids` object

Sometimes, obtaining the imputed data sets is not enough. We also want to do the analysis using these multiple data sets. `as.mice` function is built to convert an multiply imputed dataset obtained from DURR and IURR functions into a `mids` object.

```
> imp=as.mice(data=GMdata,imp=GM.imp)
> head(imp$nmis)
```

```
z1 z2 z3 z4 z5 z6
28 30 28 0 0 0
```

The imputations for `z1` are stored as

```
> imp$imp$z1
```

	impute	impute	impute	impute	impute
2	5.87852751	5.38246231	4.372164903	6.3948673	5.6518089
3	-2.02274415	6.58249856	3.028553873	1.1997224	-1.6876713
4	2.25568208	1.45087505	1.893568093	3.5869057	3.0902678
7	7.92923590	5.18331136	6.112921964	7.4710834	4.1372418
13	0.80177904	3.68779456	6.820840349	1.7231671	0.3604716
14	2.46127155	4.13688711	3.734073891	1.9511776	3.1488606
21	-2.29833555	2.85796181	-2.016519223	-1.8794955	-1.9893748
22	2.00207725	5.35389763	6.471696818	2.2327603	7.4614313
26	4.84569150	1.84202025	1.131701996	5.6774222	4.7943549
27	-0.95883080	-1.65888514	1.428926607	0.7961584	4.5400515
28	-0.81216288	3.73231504	1.248105146	5.5203996	0.8082905
29	1.05621397	1.97664024	5.654326550	4.5044610	4.8111214
31	-3.97587304	4.23938271	1.832888924	4.7337979	2.8733233
36	-0.30596101	0.08403211	-2.656168798	-0.5592650	-0.4451564
44	1.99554598	3.87831288	2.278475671	1.6160345	1.7314051
45	3.80803262	6.28265434	4.507824058	3.7684701	2.4105505
46	1.76937817	2.85354085	-0.005574403	-0.5516761	1.6464536
47	4.91918840	5.35966230	6.268155077	0.2706192	5.0349794
48	5.23698799	3.12058397	3.350847017	1.7347578	3.3676734
52	0.07145838	1.37540733	1.663528228	1.2155674	-0.7935056
57	-3.80178362	1.60956102	0.410145784	-1.9928960	-0.3265307
61	0.73332957	2.43253867	0.895059941	3.9261113	0.5507546
65	-0.27057174	-0.01741506	0.993918146	1.8811334	1.5780943
66	0.76907091	1.29221819	2.096966151	-0.2618749	0.5758993
71	1.57517705	3.89325017	2.766233238	-2.3312347	1.3420810
73	5.24085906	0.92747622	2.277394401	0.8035698	2.0612249
82	4.54789216	1.90056019	2.075113707	1.4918078	4.1835914
98	0.46711850	-2.46015918	0.970619699	1.7219256	1.0802928

Each row corresponds to a missing entry in `z1`. The columns contain the multiple imputations. The completed data set combines the observed and imputed values. The (first) completed data set can be obtained as

```
> complete(imp)
```

### 3.3 Analysis of imputed data

Suppose that we are interested in a linear regression of `z1` on `z2` and `z4` in the complete-data analysis. For this purpose, we can use the function `with()`.

```
> fit=with(imp, lm(z1~z2+z4))
```

The `fit` object contains the results of five complete-data analyses and can be pooled as follows:

```
> print(pool(fit))
```

```
Call: pool(object = fit)
```

```
Pooled coefficients:
```

```
(Intercept)      z2      z4
  1.6964981    0.2719925    0.6149610
```

```
Fraction of information about the coefficients missing due to nonresponse:
```

```
(Intercept)          z2          z4
      0.4020935    0.2842725    0.1271962
```

More detailed output can be obtained with `summary()`, i.e.,

```
> round(summary(pool(fit)),3)
```

	est	se	t	df	Pr(> t )	lo 95	hi 95	nmis	fmi	lambda
(Intercept)	1.696	0.404	4.199	21.467	0.000	0.857	2.536	NA	0.402	0.349
z2	0.272	0.118	2.296	34.695	0.028	0.031	0.513	30	0.284	0.244
z4	0.615	0.269	2.283	69.696	0.025	0.078	1.152	0	0.127	0.103

## 4 References

Zhao, Y., and Long, Q. (2013). Multiple imputation in the presence of high-dimensional data. Statistical Methods in Medical Research, in press.