

# Package: eimpute (via r-universe)

August 22, 2024

**Type** Package

**Title** Efficiently Impute Large Scale Incomplete Matrix

**Version** 0.2.4

**Date** 2024-07-22

**Author** Zhe Gao [aut, cre], Jin Zhu [aut], Junxian Zhu [aut], Xueqin Wang [aut], Yixuan Qiu [cph], Gael Guennebaud [cph, ctb], Jitse Niesen [cph, ctb], Ray Gardner [ctb]

**Maintainer** Zhe Gao <gaozh8@mail.ustc.edu.cn>

**Description** Efficiently impute large scale matrix with missing values via its unbiased low-rank matrix approximation. Our main approach is Hard-Impute algorithm proposed in <<https://www.jmlr.org/papers/v11/mazumder10a.html>>, which achieves highly computational advantage by truncated singular-value decomposition.

**License** GPL-3 | file LICENSE

**Imports** Rcpp (>= 0.12.6)

**LinkingTo** Rcpp, RcppEigen

**Encoding** UTF-8

**RoxygenNote** 7.3.1

**NeedsCompilation** yes

**Suggests** knitr

**VignetteBuilder** knitr

**Repository** CRAN

**Date/Publication** 2024-07-22 22:10:05 UTC

## Contents

biscale . . . . .	2
biscale.control . . . . .	3
eimpute . . . . .	4
incomplete.generator . . . . .	5
r.search . . . . .	6

---

biscale	<i>Data standardization</i>
---------	-----------------------------

---

### Description

Standardize a matrix rows and/or columns to have zero mean or unit variance

### Usage

```
biscale(x, thresh.sd = 1e-05, maxit.sd = 100, control = list(...), ...)
```

### Arguments

<code>x</code>	an $m$ by $n$ matrix possibly with NAs.
<code>thresh.sd</code>	convergence threshold, measured as the relative change in the Frobenius norm between two successive estimates.
<code>maxit.sd</code>	maximum number of iterations.
<code>control</code>	a list of parameters that control details of standard procedure. See <a href="#">biscale.control</a> .
<code>...</code>	arguments to be used to form the default control argument if it is not supplied directly.

### Value

A list is returned

<code>x.st</code>	The matrix after standardization.
<code>alpha</code>	The row mean after iterative process.
<code>beta</code>	The column mean after iterative process.
<code>tau</code>	The row standard deviation after iterative process.
<code>gamma</code>	The column standard deviation after iterative process.

### References

Hastie, Trevor, Rahul Mazumder, Jason D. Lee, and Reza Zadeh. Matrix completion and low-rank SVD via fast alternating least squares. *The Journal of Machine Learning Research* 16, no. 1 (2015): 3367-3402.

**Examples**

```
##### Quick Start #####
m <- 100
n <- 100
r <- 10
x_na <- incomplete.generator(m, n, r)

##### Standardize both mean and variance
xs <- biscale(x_na)

##### Only standardize mean #####
xs_mean <- biscale(x_na, row.mean = TRUE, col.mean = TRUE)

##### Only standardize variance #####
xs_std <- biscale(x_na, row.std = TRUE, col.std = TRUE)
```

---

biscale.control	<i>Control for standard procedure</i>
-----------------	---------------------------------------

---

**Description**

Various parameters that control aspects of the standard procedure.

**Usage**

```
biscale.control(
  row.mean = FALSE,
  row.std = FALSE,
  col.mean = FALSE,
  col.std = FALSE
)
```

**Arguments**

row.mean	if row.mean = TRUE (the default), row centering will be performed resulting in a matrix with row means zero. If row.mean is a vector, it will be used in the iterative process. If row.mean = FALSE nothing is done.
row.std	if row.std = TRUE , row scaling will be performed resulting in a matrix with row variance one. If row.std is a vector, it will be used in the iterative process. If row.std = FALSE (the default) nothing is done.
col.mean	similar to row.mean.
col.std	similar to row.std.

**Value**

A list with components named as the arguments.

---

eimpute

*Efficiently impute missing values for a large scale matrix*


---

### Description

Fit a low-rank matrix approximation to a matrix with missing values. The algorithm iterates like EM: filling the missing values with the current guess, and then approximating the complete matrix via truncated SVD.

### Usage

```
eimpute(
  x,
  r,
  svd.method = c("tsvd", "rsvd"),
  noise.var = 0,
  thresh = 1e-05,
  maxit = 100,
  init = FALSE,
  init.mat = 0,
  override = FALSE,
  control = list(...),
  ...
)
```

### Arguments

<code>x</code>	an $m$ by $n$ matrix with NAs.
<code>r</code>	the rank of low-rank matrix for approximating $x$
<code>svd.method</code>	a character string indicating the truncated SVD method. If <code>svd.method = "rsvd"</code> , a randomized SVD is used, else if <code>svd.method = "tsvd"</code> , standard truncated SVD is used. Any unambiguous substring can be given. Default <code>svd.method = "tsvd"</code> .
<code>noise.var</code>	the variance of noise.
<code>thresh</code>	convergence threshold, measured as the relative change in the Frobenius norm between two successive estimates.
<code>maxit</code>	maximal number of iterations.
<code>init</code>	if <code>init = FALSE</code> (the default), the missing entries will initialize with mean.
<code>init.mat</code>	the initialization matrix.
<code>override</code>	logical value indicating whether the observed elements in $x$ should be overwritten by its low-rank approximation.
<code>control</code>	a list of parameters that control details of standard procedure, See <a href="#">biscale.control</a> .
<code>...</code>	arguments to be used to form the default control argument if it is not supplied directly.

**Value**

A list containing the following components

<code>x.imp</code>	the matrix after completion.
<code>rmse</code>	the relative mean square error of matrix completion, i.e., training error.
<code>iter.count</code>	the number of iterations.

**References**

Rahul Mazumder, Trevor Hastie and Rob Tibshirani (2010) Spectral Regularization Algorithms for Learning Large Incomplete Matrices, *Journal of Machine Learning Research* 11, 2287-2322

Nathan Halko, Per-Gunnar Martinsson, Joel A. Tropp (2011) Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions, *Siam Review* Vol. 53, num. 2, pp. 217-288

**Examples**

```
##### Quick Start #####
m <- 100
n <- 100
r <- 10
x_na <- incomplete.generator(m, n, r)
head(x_na[, 1:6])
x_impute <- eimpute(x_na, r)
head(x_impute[["x.imp"]][, 1:6])
x_impute[["rmse"]]
```

---

`incomplete.generator` *Incomplete data generator*

---

**Description**

Generate a matrix with missing values, where the indices of missing values are uniformly randomly distributed in the matrix.

**Usage**

```
incomplete.generator(m, n, r, snr = 3, prop = 0.5, seed = 1)
```

**Arguments**

<code>m</code>	the rows of the matrix.
<code>n</code>	the columns of the matrix.
<code>r</code>	the rank of the matrix.
<code>snr</code>	the signal-to-noise ratio in generating the matrix. Default <code>snr = 3</code> .
<code>prop</code>	the proportion of missing observations. Default <code>prop = 0.5</code> .
<code>seed</code>	the random seed. Default <code>seed = 1</code> .

**Details**

We generate the matrix by  $UV + \epsilon$ , where  $U, V$  are  $m$  by  $r, r$  by  $n$  matrix satisfy standard normal distribution.  $\epsilon$  has a normal distribution with mean 0 and variance  $\frac{\sigma^2}{snr}$ .

**Value**

A matrix with missing values.

**Examples**

```
m <- 100
n <- 100
r <- 10
x_na <- incomplete.generator(m, n, r)
head(x_na[, 1:6])
```

---

r.search

*Search rank magnitude of the best approximating matrix*


---

**Description**

Estimate a preferable matrix rank magnitude for fitting a low-rank matrix approximation to a matrix with missing values. The algorithm use GIC/CV to search the rank in a given range, and then fill the missing values with the estimated rank.

**Usage**

```
r.search(
  x,
  r.min = 1,
  r.max = "auto",
  svd.method = c("tsvd", "rsvd"),
  rule.type = c("gic", "cv"),
  noise.var = 0,
  init = FALSE,
  init.mat = 0,
  maxit.rank = 1,
  nfolds = 5,
  thresh = 1e-05,
  maxit = 100,
  override = FALSE,
  control = list(...),
  ...
)
```

**Arguments**

<code>x</code>	an $m$ by $n$ matrix with NAs.
<code>r.min</code>	the start rank for searching. Default <code>r.min = 1</code> .
<code>r.max</code>	the max rank for searching.
<code>svd.method</code>	a character string indicating the truncated SVD method. If <code>svd.method = "rsvd"</code> , a randomized SVD is used, else if <code>svd.method = "tsvd"</code> , standard truncated SVD is used. Any unambiguous substring can be given. Default <code>svd.method = "tsvd"</code> .
<code>rule.type</code>	a character string indicating the information criterion rule. If <code>rule.type = "gic"</code> , generalized information criterion rule is used, else if <code>rule.type = "cv"</code> , cross validation is used. Any unambiguous substring can be given. Default <code>rule.type = "gic"</code> .
<code>noise.var</code>	the variance of noise.
<code>init</code>	if <code>init = FALSE</code> (the default), the missing entries will initialize with mean.
<code>init.mat</code>	the initialization matrix.
<code>maxit.rank</code>	maximal number of iterations in searching rank. Default <code>maxit.rank = 1</code> .
<code>nfolds</code>	number of folds in cross validation. Default <code>nfolds = 5</code> .
<code>thresh</code>	convergence threshold, measured as the relative change in the Frobenius norm between two successive estimates.
<code>maxit</code>	maximal number of iterations.
<code>override</code>	logical value indicating whether the observed elements in <code>x</code> should be overwritten by its low-rank approximation.
<code>control</code>	a list of parameters that control details of standard procedure, See <a href="#">biscale.control</a> .
<code>...</code>	arguments to be used to form the default control argument if it is not supplied directly.

**Value**

A list containing the following components

<code>x.imp</code>	the matrix after completion with the estimated rank.
<code>r.est</code>	the rank estimation.
<code>rmse</code>	the relative mean square error of matrix completion, i.e., training error.
<code>iter.count</code>	the number of iterations.

**Examples**

```
##### Quick Start #####
m <- 100
n <- 100
r <- 10
x_na <- incomplete.generator(m, n, r)
head(x_na[, 1:6])
x_impute <- r.search(x_na, 1, 15, "rsvd", "gic")
x_impute[["r.est"]]
```

# Index

biscale, [2](#)

biscale.control, [2](#), [3](#), [4](#), [7](#)

eimpute, [4](#)

incomplete.generator, [5](#)

r.search, [6](#)