# Package ‘MVNBayesian’

August 16, 2018

**Type** Package

**Title** Bayesian Analysis Framework for MVN (Mixture) Distribution

**Version** 0.0.8-11

**Author** ZHANG Chen

**Maintainer** ZHANG Chen <447974102@qq.com>

**Description** Tools of Bayesian analysis framework using the method suggested by Berger (1985) <doi:10.1007/978-1-4757-4286-2> for multivariate normal (MVN) distribution and multivariate normal mixture (MixMVN) distribution:
a) calculating Bayesian posteriori of (Mix)MVN distribution;
b) generating random vectors of (Mix)MVN distribution;
c) Markov chain Monte Carlo (MCMC) for (Mix)MVN distribution.

**Imports** mvtnorm, plyr, stats

**Suggests** rgl, Rfast

**License** GPL-2

**URL** https://github.com/CubicZebra/MVNBayesian

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 6.1.0

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2018-08-16 10:40:07 UTC

## R topics documented:

MVNBayesian-package . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
Ascending_Num . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
dataset1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
dataset2 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
MatrixAlternative . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

MixMVN_BayesianPosteriori . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
MixMVN_GibbsSampler . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
MixMVN_MCMC . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8
MVN_BayesianIterator . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
MVN_BayesianPosteriori . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
MVN_FConditional . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13
MVN_GibbsSampler . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14
MVN_MCMC . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16

**Index** **18**

---

`MVNBayesian-package` *Bayesian Analysis Framework for MVN (Mixture) Distribution*

---

### Description

Tools of Bayesian analysis framework using the method suggested by Berger (1985) <doi:10.1007/978-1-4757-4286-2> for multivariate normal (MVN) distribution and multivariate normal mixture (MixMVN) distribution: a) calculating Bayesian posteriori of (Mix)MVN distribution; b) generating random vectors of (Mix)MVN distribution; c) Markov chain Monte Carlo (MCMC) for (Mix)MVN distribution.

### Details

This package is aimed to build a easy approach for MVN (mixture) distribution in Bayesian analysis framework. Bayesian posteriori MVN (mixture) distribution can be calculated in conditions of given priori MVN (mixture) informations. The conjugated property of MVN distribution makes it effective in parameter estimation using Bayesian iterator. Joint and marginal probability densities of a certain MVN (mixture) can be achieved through random vector generator, using Gibbs sampling. Conditional probability densities from a certain MVN (mixture) can be simulated using MCMC method.

### Author(s)

ZHANG Chen

Maintainer: ZHANG Chen <447974102@qq.com>

### References

"Statistical Inference" by George Casella. Roger L. Berger;
"Statistical Decision Theory and Bayesian Analysis" by James O. Berger;
"Matrix Computation" by Gee H. Golub. Charles F. Van Loan;
"Bayesian Statistics" by WEI Laisheng;
"Machine Learning" by NAKAGAWA Hiroshi.

### See Also

`stats`, `mvtnorm`

## Examples

```
library(Rfast)
library(mvtnorm)
library(plyr)

head(dataset1)

BP <- MVN_BayesianPosteriori(dataset1)
BP

BP_Gibbs <- MVN_GibbsSampler(5000, BP)
colMeans(BP_Gibbs)
colrange(BP_Gibbs)

result <- MVN_MCMC(BP, 5000, c(1), c(77.03))
result$Accept
```

---

# Ascending_Num *Renumbering vector by elemental frequency*

---

## Description

Renumbering vector by elemental frequency in ascending order.

## Usage

```
# Tidy vector by elemental frequency:
Ascending_Num(data)
```

## Arguments

| | |
|---|---|
| `data` | An 1d-vector. |

## Value

return a renumbered vector by elemental frequency. Factors will be positive integers arrayed in ascending order.

## Examples

```
library(plyr)

x <- c(1,2,2,2,2,2,2,2,3,3,3,1,3,3,3)
x
Ascending_Num(x)
```

## dataset1 — *Dataset for MVN test*

### Description

Dataset built for MVN mixture test, which contains 3 variables and 25 observations.

### Usage

```
data("dataset1")
```

### Format

A data frame with 25 observations on 3 independent variables, named as `fac1`, `fac2` and `fac3`.

`fac1` The 1st factor.

`fac2` The 2nd factor.

`fac3` The 3rd factor.

### Examples

```
dataset1
```

## dataset2 — *Dataset for MVN mixture test*

### Description

Dataset built for MVN mixture test, which contains 4 variables (the first 4 columns), clustering (the last column) and 96 observations.

### Usage

```
data("dataset2")
```

### Format

A data frame with 96 pseudo-observations generated by random number generator. All observations come from 3 different centers which have been marked in the last column "species". More specifically, data of species=1 comes from the center (1,1,1,1); data of species=2 comes from the center (2,2,2,0); data of species=3 comes from the center (1,0,2,2).

`dimen1` the 1st variable

`dimen2` the 2nd variable

`dimen3` the 3rd variable

`dimen4` the 4th variable

`species` clustering label

## Examples

```
dataset2
```

---

`MatrixAlternative` *Interchanging specified rows and columns*

---

## Description

Interchange all elements between two specified rows and columns in a matrix.

## Usage

```
# A matrix-like data
MatrixAlternative(data, sub, rep)
```

## Arguments

| | |
|---|---|
| `data` | A matrix to be processed. |
| `sub` | A positive integer. The first selected dimension. |
| `rep` | A positive integer. The second selected dimension. Default value is 1. |

## Value

return a matrix with interchanged rows and columns in two specified dimensions.

## Examples

```
library(plyr)

M <- matrix(1:9,3,3,1)
M
MatrixAlternative(M, 2)
```

---

`MixMVN_BayesianPosteriori` *Calculate Bayesian posteriori MVN mixture distribution*

---

## Description

The function to export the mixture probabilities, the mean vectors and covariance matrices of Bayesian posteriori MVN mixture distribution in the basis of given priori information (priori MVN mixture) and observation data (a design matrix containing all variables).

### Usage

```
# paramtric columns-only as input data:
# data <- dataset2[,1:4]

# Specify species to get parameters of MVN mixture model:
MixMVN_BayesianPosteriori(data, species, idx)
```

### Arguments

| | |
|---|---|
| data | A data.frame or matrix-like data: obervations should be arrayed in rows while variables should be arrayed in columns. |
| species | A positive integer. The number of clusters for import data. It will be only called once by the next argument idx through kmeans clustering algrithm in this function. Default value is 1, which means no clustering algrithm is used. |
| idx | A vector-like data to import for accepting clustering result. Default value is generated by kmeans clustering. Notice the length of idx should be the same as observation numbers of data (rows). |

### Value

return a matrix-like result containing all parameters of Bayesian posteriori MVN mixture distribution: Clusters are arrayed in rows, while the mixture probabilities, posteriori mean vectors and posteriori covariance matrices are arrayed in columns.

### See Also

kmeans, MVN_BayesianPosteriori

### Examples

```
library(plyr)

# Design matrix should only contain columns of variables
# Export will be a matrix-like data
# Using kmeans (default) clustering algrithm
data_dim <- dataset2[,1:4]
result <- MixMVN_BayesianPosteriori(data=data_dim, species=3)
result

# Get the parameters of the cluster1:
result[1,]

# Get the mixture probability of cluster2:
# (Attention to the difference between
# result[2,1][[1]] and result[2,1])
result[2,1][[1]]

# Get the mean vector of cluster1:
result[1,2][[1]]
```

```
# Get the covariance matrix of cluster3:
result[3,3][[1]]
```

MixMVN_GibbsSampler *Gibbs sampler for MVN mixture distribution*

## Description

Generating random vectors on the basis of a given MVN mixture distribution, through Gibbs sampling algorithm or matrix factorization.

## Usage

```
# Bayesian posteriori MVN mixture model as input data:
# data <- MixMVN_BayesianPosteriori(dataset2[,1:4], species=3)

# Generate random vectors based on Bayesian posteriori MVN mixture:
MixMVN_GibbsSampler(n, data, random_method = c("Gibbs", "Fast"), reject_rate=0, ...)
```

## Arguments

| | |
|---|---|
| `n` | A positive integer. The numbers of random vectors to be generated. |
| `data` | A matrix-like data which contains the mixture probability, mean vector and covariance matrix for each cluster in each row. |
| `random_method` | The method to generate random vectors. Options are `"Gibbs"`: Gibbs sampling for MVN mixture model; and `"Fast"`: call `rmvnorm()` to generate random vectors based on matrix factorization. |
| `reject_rate` | A numeric value which will be efficient if the `random_method` is "Gibbs": Determine the discarded items in burn-in period by ratio. Default value is 0. For details see `MVN_GibbsSampler`. |
| `...` | Other arguments to control the process in Gibbs sampling if the `random_method` is "Gibbs". |

## Details

It is recommanded using the random method of "Fast" due to the high efficiency. The time complexity of "Gibbs" method is O(k*n) where the k means dimensionality of MVN mixture model and n means generated numbers of random vectors; while that of the "Fast" method is only O(n), without considering the effect of burn-in period. this discrepancy will be even further significant when we use MCMC methods to do some further analysis in which random vectors will be generated every time when we set conditions.

## Value

return a series random vectors in the basis of given MVN mixture distribution.

### See Also

Ascending_Num, MixMVN_BayesianPosteriori, MVN_BayesianPosteriori

### Examples

```
library(plyr)
library(mvtnorm)
library(stats)

# Use dataset2 for demonstration. Get parameters of Bayesian
# posteriori multivariate normal mixture distribution
head(dataset2)
dataset2_par <- dataset2[,1:4] # only parameter columns are premitted
MixBPos <- MixMVN_BayesianPosteriori(dataset2_par, species=3)
MixBPos

# Generate random vectors using Gibbs sampling:
MixBPos_Gibbs <- MixMVN_GibbsSampler(5000, MixBPos, random_method = "Gibbs")
head(MixBPos_Gibbs)

# Compared generation speed of "Gibbs" to that of "Fast"
MixBPos_Fast <- MixMVN_GibbsSampler(5000, MixBPos, random_method = "Fast")
head(MixBPos_Fast)

# Visulization by clusters:
library(rgl)
dimen1 <- MixBPos_Gibbs[,1]
dimen2 <- MixBPos_Gibbs[,2]
dimen3 <- MixBPos_Gibbs[,3]
dimen4 <- MixBPos_Gibbs[,4]
plot3d(x=dimen1, y=dimen2, z=dimen3, col=MixBPos_Gibbs[,5], size=2)
```

---

## MixMVN_MCMC — *MCMC simulation for MVN mixture distribution*

---

### Description

Function to get a MCMC simulation results based on the imported MVN mixture distribution. It is commonly used for inquiring the specified conditional probability of MVN mixture distribuiton calculated through Bayesian posteriori.

### Usage

```
# Bayesian posteriori mix MVN as input data:
# data <- MixMVN_BayesianPosteriori(dataset2[,1:4], 3)

# run MCMC simulation based on Bayesian posteriori mix MVN:
MixMVN_MCMC(data, steps, pars, values, tol, random_method, ...)
```

### Arguments

- `data` — A matrix-like data containing the mixture probability, mean vector and covariance matrix for each cluster in each row.
- `steps` — A positive integer. The numbers of random vectors to be generated for MCMC step.
- `pars` — A integer vector to declare fixed dimension(s). For example if the desired dimensions are 1st=7 and 3rd=10, set this argument as c(1,3).
- `values` — A numeric vector to assign value(s) to declared dimension(s). For example if the desired dimensions are 1st=7 and 3rd=10, set this argument as c(7,10).
- `tol` — Tolerance. A numeric value to control the generated vectors to be accepted or rejected. Criterion uses Euclidean distance in declared dimension(s). Default value is 0.3.
- `random_method` — The method to generate random vectors. Options are `"Gibbs"`: Gibbs sampling for MVN mixture model; and `"Fast"`: call `rmvnorm()` to generate random vectors based on matrix factorization. Default option is `"Fast"`.
- `...` — Other arguments to control the process in Gibbs sampling if the `random_method` is "Gibbs".

### Value

return a list which contains:

- `AcceptRate` — Acceptance of declared conditions of MCMC
- `MCMCdata` — All generated random vectors in MCMC step based on MVN mixture distribution
- `Accept` — Subset of accepted sampling in MCMCdata
- `Reject` — Subset of rejected sampling in MCMCdata

### See Also

`MixMVN_BayesianPosteriori`, `MixMVN_GibbsSampler`, `MVN_GibbsSampler`, `MVN_FConditional`

### Examples

```
library(plyr)
library(mvtnorm)
library(stats)

# dataset2 has 4 parameters: dimen1, dimen2, dimen3 and dimen4:
head(dataset2)
dataset2_dim <- dataset2[,1:4] # extract parametric columns

# Get posteriori parameters of dataset2 using kmeans 3 clustering:
MixBPos <- MixMVN_BayesianPosteriori(dataset2_dim, 3)

# If we want to know when dimen1=1, which clusters are accepted, run:
MixBPos_MCMC <- MixMVN_MCMC(MixBPos, steps=5000, pars=c(1), values=c(1), tol=0.3)
```

```
MixBPos_MCMC$AcceptRate
result <- MixBPos_MCMC$MCMCdata
head(result)

# count accepted samples by clustering:
count(result[which(result[,7]==1),5])

library(rgl)
# Visualization using plot3d() if necessary:
# Clustering result in the rest 3 dimensions:
plot3d(result[,2], result[,3], z=result[,4], col=result[,5], size=2)

# Acceptance rejection visualization:
plot3d(result[,2], result[,3], z=result[,4], col=result[,7]+1, size=2)
```

---

`MVN_BayesianIterator` *Parameter estimation using Bayesian iteration*

---

## Description

Function to execute parameter estimation for MVN distribution, under Bayesian analysis framework.

## Usage

```
# Get parameters of Bayesian posteriori MVN:
MVN_BayesianIterator(data, pri_mean=colMeans(data), Gibbs_nums=5000,
pseudo_nums=dim(data)[1], threshold=1e-04, iteration=100, ...)
```

## Arguments

| | |
|---|---|
| `data` | A data.frame or matrix-like data: obervations should be arrayed in rows while variables should be arrayed in columns. |
| `pri_mean` | A numeric vector to assign priori mean for MVN. Default value applies `colMeans()` to data. |
| `Gibbs_nums` | A positive integer. The numbers of random vectors to be generated for each iteration step. Defult value is 5000. |
| `pseudo_nums` | A positive integer. The argument to determine numbers of generated vectors used for each iteration step. Default value keeps the same scale as input data. Notice that a too small value can result in singular matrix. |
| `threshold` | A numeric value to control stoping the iteration loop. Default value used 0.0001. While the Euclidean distance of mean vectors between pseudo-data (the last `pseudo_nums` items) and Bayesian posteriori is less than threshold, iteration stops. |
| `iteration` | A positive integer. Argument to assign the maximum steps for iteration. Default value is 100 after which the iteration loop will compulsively exit. |
| `...` | Other arguments to control the process in Gibbs sampling. |

### Details

Because that MVN distribution possess conjugated property in Bayesian analysis framework, the convergence of Bayesian iterator for MVN distribution can be ensured, accoumpanied with the shrink of 2nd-norm of Bayesian posteriori covariance matrix. But pay attention to the fact that pseudo-data leads to the randomness, the argument `pseudo_nums` should be set carefully.

### Value

return a double level list containing Bayesian posteriori after iteration process:

| | |
|---|---|
| `mean` | Bayesian posteriori mean vector |
| `var` | Bayesian posteriori covariance matrix |

### Note

If the parameter values are the only interested thing we concerned, this iterator makes sense. Since it can significantly help us decrease the scale of covariance matrix, to obtain a more reliable estimation for the parameters. However, in more cases, some correlationships of a certain group of pamameters are more valuable, which are usually clued by the covariance matrix.

### See Also

`MVN_BayesianPosteriori`, `MVN_GibbsSampler`, `MVN_FConditional`, `MatrixAlternative`

### Examples

```
library(mvtnorm)

# Bayesian posteriori before iteration using dataset1 as example,
# c(80, 16, 3) as priori mean:
# View 2-norm of covariance matrix of Bayesian posteriori:
BPos_init <- MVN_BayesianPosteriori(dataset1, c(80,16,3))
BPos_init
norm(as.matrix(BPos_init$var), type = "2")

# Bayesian posteriori after iteration using c(80,16,3) as priori
# Using 30 last samples generated by GibbsSampler for each step:
BPos_fina1 <- MVN_BayesianIterator(dataset1, c(80,16,3), 5000, 30)
BPos_fina1
norm(as.matrix(BPos_fina1$var), type = "2")

# Too small pseudo_nums setting can results in singular system, try:
MVN_BayesianIterator(dataset1, pseudo_nums=3)
```

MVN_BayesianPosteriori
*Calculate Bayesian posteriori MVN distribution*

## Description

The function to export the mean vector and covariance matrix of Bayesian posteriori MVN distribution in the basis of given priori information (priori MVN) and observation data (a design matrix containing all variables).

## Usage

```
# Given the data as design matrix, priori mean vector and priori covariance
# matrix, this function will export a list which contains mean ($mean) and
# covariance ($var) of Bayesian posteriori multivariate normal distribution.

MVN_BayesianPosteriori(data, pri_mean, pri_var)
```

## Arguments

| | |
|---|---|
| `data` | A data.frame or matrix-like data: obervations should be arrayed in rows while variables should be arrayed in columns. |
| `pri_mean` | A numeric vector to assign priori mean for MVN. Default value applies `colMeans()` to data. |
| `pri_var` | A matrix-like parameter to assign priori covariance matrix. Default value uses unit matrix. |

## Value

return a double level list containing:

| | |
|---|---|
| `mean` | mean vector of Bayesian posteriori MVN distribution |
| `var` | covariance of Bayesian posteriori MVN distribution |

## Note

It is strongly recommanded that users should have some prior knowledge of ill-conditioned system before using this function. Simply, ill-conditioned system, or singular matrix, is caused by a) insufficient data or b) almostly linear dependency of two certain parameters, which two can result in a excessively small eigenvalue then cause a ill-conditioned (singular) system. Therefore users must diagnose their data firstly to confirm the fact that the it contains enough observations, and the degree of freedom is strictly equal to the number of parameters as well. Additionally, for the argument `pri_var`, a real symmetric matrix is desired by definition.

### Examples

```
# Demo using dataset1:
head(dataset1)
BPos <- MVN_BayesianPosteriori(dataset1, c(80,16,3))
BPos$mean
BPos$var

# Singular system caused by insufficient data
eigen(var(dataset1[1:3,]))$values
rcond(var(dataset1[1:3,]))
eigen(var(dataset1[1:6,]))$values
rcond(var(dataset1[1:6,]))

# Singular system caused by improper degree of freedom
K <- cbind(dataset1, dataset1[,3]*(-2)+3)
eigen(var(K[,2:4]))$values
rcond(var(K[,2:4]))
```

---

## MVN_FConditional — *Calculate full conditional normal ditribution of MVN*

---

### Description

Function to export parameters of full conditional normal distribution in basis of given MVN distribution, the undecided dimension, as well as all values in the rest dimensions.

### Usage

```
# Bayesian posteriori as input data:
# data <- MVN_BayesianPosteriori(dataset1, c(80,16,3))

# inquire parameters of full-conditional distribution based on Bayesian posteriori:
MVN_FConditional(data, variable, z)
```

### Arguments

| | |
|---|---|
| data | A double level list containing all parameters of MVN distribution: mean vector (data$mean) and covariance matrix (data$var). |
| variable | A integer to specify the undecided dimension. |
| z | A nd-vector to assign conditions (n = dimensions of given MVN distribution). It should be noted that the value in dimension specified by variable doesn't participate in the calculation. |

### Details

It can be proved that any full conditional distribution from a given MVN will degenerate to an 1d-normal distribution.

### Value

return a double level list containing the following parameters of full conditional normal distributions of given MVN in specified dimension:

| | |
|---|---|
| mean | a numberic mean of a normal distribution |
| var | a numberic variance of a normal distribution |

### See Also

MVN_BayesianPosteriori, MatrixAlternative

### Examples

```
head(dataset1)
BPos <- MVN_BayesianPosteriori(dataset1, c(80,16,3))
BPos # Bayesian Posteriori
result <- MVN_FConditional(BPos, variable = 1, z=c(75, 13, 4))
result$mean
class(result$mean)
result$var
class(result$var)

# compare the following results:
MVN_FConditional(BPos, variable = 2, z=c(75, 13, 4))
MVN_FConditional(BPos, variable = 2, z=c(75, 88, 4))
MVN_FConditional(BPos, variable = 1, z=c(75, 88, 4))
```

---

## MVN_GibbsSampler *Gibbs sampler for MVN distribution*

---

### Description

Generating random vectors on the basis of a given MVN distribution, through Gibbs sampling algorithm.

### Usage

```
# Bayesian posteriori as data
# data <- MVN_BayesianPosteriori(dataset1)

# Using Gibbs sampler to generate random vectors based on Bayesian posteriori:
MVN_GibbsSampler(n, data, initial, reject_rate, burn)
```

### Arguments

| | |
|---|---|
| n | A positive integer. The numbers of random vectors to be generated. |
| data | A double level list which contains the mean vector (data$mean) and the covariance matrix (data$var) of a given MVN distribution. |
| initial | Initial vector where Markov chain starts. Default value use a random vector generated by rmvnorm(). |
| reject_rate | A numeric to control burn-in period by ratio. Default value is 0.2, namely the first 20% generated vectors will be rejected. If this arg was customized, the next arg burn should maintain the default value. |
| burn | A numeric to control burn-in period by numbers. If this arg was customized, final result will be generated by this manner in which it will drop the first n numbers (n=burn). |

### Details

There're also some literatures suggest using the mean or mode of priori as initial vector. Users can customize this setting according to their own needs.

### Value

return a series random vectors in the basis of given MVN distribution.

### See Also

MVN_FConditional, MatrixAlternative

### Examples

```
library(mvtnorm)

# Get parameters of Bayesian posteriori multivariate normal distribution
BPos <- MVN_BayesianPosteriori(dataset1)
BPos

# Using previous result (BPos) to generate random vectors through Gibbs
# sampling: 7000 observations, start from c(1,1,2), use 0.3 burning rate
BPos_Gibbs <- MVN_GibbsSampler(7000, BPos, initial=c(1,1,2), 0.3)
tail(BPos_Gibbs)

# Check for convergence of Markov chain
BPos$mean
colMeans(BPos_Gibbs)
BPos$var
var(BPos_Gibbs)

# 3d Visulization:
library(rgl)
fac1 <- BPos_Gibbs[,1]
fac2 <- BPos_Gibbs[,2]
```

```
fac3 <- BPos_Gibbs[,3]
plot3d(x=fac1, y=fac2, z=fac3, col="red", size=2)
```

---

MVN_MCMC *MCMC simulation for MVN distribution*

---

## Description

Function to get a MCMC simulation results based on the imported MVN distribution. It is commonly used for inquiring the specified conditional probability of MVN distribuiton calculated through Bayesian posteriori.

## Usage

```
# Bayesian posteriori as input data
# data <- MVN_BayesianPosteriori(dataset1, pri_mean=c(80,16,3))

# run MCMC simulation using Bayesian posteriori:
MVN_MCMC(data, steps, pars, values, tol, ...)
```

## Arguments

| | |
|---|---|
| data | A double level list which contains the mean vector (data$mean) and the covariance matrix (data$var) of a given MVN distribution. |
| steps | A positive integer. The numbers of random vectors to be generated for MCMC step. |
| pars | A integer vector to declare fixed dimension(s). For example if the desired dimensions are 1st=7 and 3rd=10, set this argument as c(1,3). |
| values | A numeric vector to assign value(s) to declared dimension(s). For example if the desired dimensions are 1st=7 and 3rd=10, set this argument as c(7,10). |
| tol | Tolerance. A numeric value to control the generated vectors to be accepted or rejected. Criterion uses Euclidean distance in declared dimension(s). Default value is 0.3. |
| ... | Other arguments to control the process in Gibbs sampling. |

## Value

return a list which contains:

| | |
|---|---|
| AcceptRate | Acceptance of declared conditions of MCMC |
| MCMCdata | All generated random vectors in MCMC step based on MVN distribution |
| Accept | Subset of accepted sampling in MCMCdata |
| Reject | Subset of rejected sampling in MCMCdata |

## See Also

MVN_GibbsSampler, MVN_FConditional

## Examples

```
library(mvtnorm)
library(plyr)

# dataset1 has three parameters: fac1, fac2 and fac3:
head(dataset1)

# Get posteriori parameters of dataset1 using prior of c(80,16,3):
BPos <- MVN_BayesianPosteriori(dataset1, pri_mean=c(80,16,3))

# If we want to know when fac1=78, how fac2 responses to fac3, run:
BPos_MCMC <- MVN_MCMC(BPos, steps=8000, pars=c(1), values=c(78), tol=0.3)
MCMC <- BPos_MCMC$MCMCdata
head(MCMC)

# Visualization using plot3d() if necessary:
library(rgl)
plot3d(MCMC[,1], MCMC[,2], z=MCMC[,3], col=MCMC[,5]+1, size=2)

# Visualization: 2d scatter plot
MCMC_2d <- BPos_MCMC$Accept
head(MCMC_2d)
plot(MCMC_2d[,3], MCMC_2d[,2], pch=20, col="red", xlab = "fac3", ylab = "fac2")

# Compared to the following scatter plot when fac1 is not fixed:
plot(BPos_MCMC$MCMCdata[,3], BPos_MCMC$MCMCdata[,2], pch=20, col="red", xlab = "fac3",
ylab = "fac2")
```

# Index


∗Topic **Bayesian posteriori**
MixMVN_BayesianPosteriori, 5
MVN_BayesianIterator, 10
MVN_BayesianPosteriori, 12
MVNBayesian-package, 2
∗Topic **Full conditional distribution**
MVN_FConditional, 13
∗Topic **Gibbs sampling**
MixMVN_GibbsSampler, 7
MixMVN_MCMC, 8
MVN_BayesianIterator, 10
MVN_GibbsSampler, 14
MVN_MCMC, 16
MVNBayesian-package, 2
∗Topic **MCMC**
MixMVN_MCMC, 8
MVN_MCMC, 16
MVNBayesian-package, 2
∗Topic **MVN distribution**
dataset1, 4
MVN_BayesianIterator, 10
MVN_BayesianPosteriori, 12
MVN_FConditional, 13
MVN_GibbsSampler, 14
MVN_MCMC, 16
MVNBayesian-package, 2
∗Topic **MVN mixture distribution**
dataset2, 4
MixMVN_BayesianPosteriori, 5
MixMVN_GibbsSampler, 7
MixMVN_MCMC, 8
MVNBayesian-package, 2
∗Topic **Matrix preprocessing**
MatrixAlternative, 5
∗Topic **Renumbering index**
Ascending_Num, 3
∗Topic **datasets**
dataset1, 4
dataset2, 4
∗Topic **package**
MVNBayesian-package, 2

Ascending_Num, 3, *8*

colMeans(), *10*, *12*

dataset1, 4
dataset2, 4

kmeans, *6*

MatrixAlternative, 5, *11*, *14*, *15*
MixMVN_BayesianPosteriori, 5, *8*, *9*
MixMVN_GibbsSampler, 7, *9*
MixMVN_MCMC, 8
MVN_BayesianIterator, 10
MVN_BayesianPosteriori, *6*, *8*, *11*, 12, *14*
MVN_FConditional, *9*, *11*, 13, *15*, *17*
MVN_GibbsSampler, *7*, *9*, *11*, 14, *17*
MVN_MCMC, 16
MVNBayesian *(*MVNBayesian-package*)*, 2
MVNBayesian-package, 2
mvtnorm, *2*

rmvnorm(), *7*, *9*, *15*

stats, *2*