

Package ‘EMSNM’

July 21, 2025

Type Package

Title EM Algorithm for Sigmoid Normal Model

Version 1.0

Date 2019-04-19

Author Linsui Deng <denglinsui@gmail.com>

Maintainer Linsui Deng <denglinsui@gmail.com>

Description It provides a method based on EM algorithm to estimate the parameter of a mixture model, Sigmoid-Normal Model, where the samples come from several normal distributions (also call them subgroups) whose mean is determined by co-variable Z and coefficient alpha while the variance are homogeneous. Meanwhile, the subgroup each item belongs to is determined by co-variables X and coefficient eta through Sigmoid link function which is the extension of Logistic Link function. It uses bootstrap to estimate the standard error of parameters. When sample is indeed separable, removing estimation with abnormal sigma, the estimation of alpha is quite well. I used this method to explore the subgroup structure of HIV patients and it can be used in other domains where exists subgroup structure.

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2019-04-25 09:10:03 UTC

Contents

EMSNM-package	2
Ccompute	4
EMalgorithm	5
EMbootstrap	7
EMsimulation	8
EM_parameter_sd	10
EM_result_sort	11
fnorm	12
Ggenerate	13
softmax	14

standard	15
update_eta	15
update_gamma	17
weight_matrix	18
Wgenerate	19

Index	21
--------------	-----------

EMSNM-package	<i>EM Algorithm for Sigmoid Normal Model</i>
---------------	--

Description

It provides a method based on EM algorithm to estimate the parameter of a mixture model, Sigmoid-Normal Model, where the samples come from several normal distributions (also call them sub-groups) whose mean is determined by co-variable Z and coefficient alpha while the variance are homogeneous. Meanwhile, the subgroup each item belongs to is determined by co-variables X and coefficient eta through Sigmoid link function which is the extension of Logistic Link function. It uses bootstrap to estimate the standard error of parameters. When sample is indeed separable, removing estimation with abnormal sigma, the estimation of alpha is quite well. I used this method to explore the subgroup structure of HIV patients and it can be used in other domains where exists subgroup structure.

Details

The DESCRIPTION file:

Package: EMSNM
Type: Package
Title: EM Algorithm for Sigmoid Normal Model
Version: 1.0
Date: 2019-04-19
Author: Linsui Deng <denglinsui@gmail.com>
Maintainer: Linsui Deng <denglinsui@gmail.com>
Description: It provides a method based on EM algorithm to estimate the parameter of a mixture model, Sigmoid-Normal M
License: GPL(>=2)

Index of help topics:

Ccompute	Ccompute
EMSNM-package	EM Algorithm for Sigmoid Normal Model
EM_parameter_sd	Bootstrap Parameter Inference
EM_result_sort	Sort Parameter
EMalgorithm	Parameter Estimation
EMbootstrap	Bootstrap Method
EMsimulation	Simulation For Estimation
Ggenerate	Subgroup Determination

Wgenerate	Sigmoid Logistic Data Generation
fnorm	Density Value
softmax	Softmax Value
standard	Data Standardlization
update_eta	Udata Eta
update_gamma	Udata Alpha and Sigma
weight_matrix	Weighted Inner Product

The EMalgorithm is used to estimate the parameters, EMbootstrap is used to estimate the parameters with bootstrap method. In EMSimulation we can simulate the situation with given parameters, so parameter estimation can be verified.

Author(s)

Linsui Deng <denglinsui@gmail.com>

Maintainer: Linsui Deng <denglinsui@gmail.com>

Examples

```
#parameter initialization
etasize <- 2
classsize <- 2
alphasize <- 3
samplesize <- 100
experiments <- 30

etatest <- matrix(c(1,1,
                    0,0),etasize,classsize)

alphatest <- matrix(c(1,0,2,
                     4,3,5),alphasize,classsize)

sigmatest <- 0.5

#test of EMSimulation
EMsimulation_result <- EMSimulation(eta=etatest,alpha=alphatest,sigma=sigmatest,
                                   samplesize=samplesize,experiments=experiments,
                                   compact_flag=TRUE,C0=5,C1=0.5,C2=5)
index <- which(EMsimulation_result$sigma<0.8)
EMsimulation_result_sort <- EM_result_sort(EMsimulation_result$alpha[index,,],
                                           EMsimulation_result$eta[index,,])
EM_parameter <- EM_parameter_sd(EMsimulation_result_sort$alpha,
                               EMsimulation_result_sort$eta,
                               EMsimulation_result$sigma[index])

#test of EMbootstrap
samplesize <- 1000
X <- matrix(c(matrix(1,samplesize),
                rnorm(samplesize*(etasize-1))+1),samplesize,etasize)
Z <- matrix(c(matrix(1,samplesize),rbinom(prob=1/2,size=1,n=samplesize),
                rnorm(samplesize*(alphasize-2))+1),samplesize,alphasize)
```

```
Wtest <- Wgenerate(alpha=alphatest,eta=etatest,sigma=sigmat, X=X,Z=Z)

boots_samplesize <- 100
boots_experiments <- 30
samplesize <- dim(Wtest$X)[1]
EMbootstrap_theta <- EMbootstrap(Wtest$X,Wtest$Y,Wtest$Z,samplesize,
                                boots_samplesize,boots_experiments,
                                classsize=2,compact_flag=TRUE,C0=5,C1=0.2,C2=5)
index <- which(EMbootstrap_theta$sigma<0.8)
EMsimulation_result_sort <- EM_result_sort(EMbootstrap_theta$alpha[index,,],
                                           EMbootstrap_theta$eta[index,,])
EM_parameter <- EM_parameter_sd(EMsimulation_result_sort$alpha,
                                EMsimulation_result_sort$eta,
                                EMbootstrap_theta$sigma[index])
```

Ccompute	Ccompute
----------	----------

Description

Compute the probability of Y in given parameters alphas, sigmas, etas and variables X, Z by the Bayesian Formula under the assumption of Sigmoid-Normal Model.

Usage

```
Ccompute(alphas, sigmas, etas, X, Z, Y)
```

Arguments

- | | |
|--------|---|
| alphas | the coefficients of the mean of each subgroup |
| sigmas | the variance of Y |
| etas | the coefficients determining subgroup |
| X | the covariables of the mean of each subgroup |
| Z | the covariables determining subgroup |
| Y | the respond variable |

Value

the probability of Y in given parameters alphas, sigmas, etas and variables X, Z under the assumption of Sigmoid-Normal Model.

Author(s)

Linsui Deng

Examples

```
#some variables
samplesize <- 1000
classsize <- 6
etasize <- 3
alphasize <- 2

Xtest <- data.frame(matrix(rnorm(samplesize*etasize),samplesize,etasize))
Ztest <- matrix(rnorm(samplesize*alphasize),samplesize,alphasize)

etatest <- matrix(seq(1.15,1,length=etasize*classsize),etasize,classsize)
alphatest <- matrix(seq(1.15,1,length=alphasize*classsize),alphasize,classsize)

Wtest <- Wgenerate(alpha=alphatest,eta=etatest,X=Xtest,Z=Ztest)

#test of Ccompute
sigmatetest <- 1
Ctest <-
  Ccompute(alphat=alphatest,sigmat=sigmatetest,
           etat=etatest,X=Wtest$X,Z=Wtest$Z,Y=Wtest$Y)
```

EMalgorithm

*Parameter Estimation***Description**

Estimate paramters value by EM algorithm under the assumption of Sigmoid-Normal Model.

Usage

```
EMalgorithm(X, Y, Z, etat, alphasize, sigmat, classsize = 2, learning_rate = 0.1,
            regular_parameter_eta = 0.001, max_iteration = 10000,
            max_iteration_eta = 10000, compact_flag = FALSE, C0 = 5, C1 = 2, C2 = 9)
```

Arguments

X	the covariables of the mean of each subgroup
Y	the respond variable
Z	the covaraibles determining subgroup
etat	the coeffients determining subgroup
alphat	the coeffients of the mean of each subgroup
sigmat	the variance of Y
classsize	the number of subgroup types in your model assumption
learning_rate	learning rate of updating eta
regular_parameter_eta	regular value of updating eta by gradiant descending method.

max_iteration maximum steps of iteration to avoid unlimited looping.
 max_iteration_eta maximal steps of eta iteration to avoid unlimited looping.
 compact_flag if the value of eta is limited in a compact set, set it TRUE
 C0 the maximum of intercept of eta.
 C1 the minimum of the norm of slope of eta
 C2 the maximum of the norm of slope of eta

Value

alpha alpha estimation
 eta eta estimation
 sigma sigma estimation

Author(s)

Linsui Deng

Examples

```

#data generation
samplesize <- 1000
classsize <- 2
etasize <- 3
alphasize <- 3

set.seed(1)
Xtest <- data.frame(matrix(rnorm(samplesize*etasize),samplesize,etasize))
etatest <- matrix(c(1,2,-1,
                   0,0,0),etasize,classsize)

Ztest <- matrix(rnorm(samplesize*alphasize),samplesize,alphasize)
alphatest <- matrix(c(1,0,2,
                     5,0,7),alphasize,classsize)

sigmatetest <- 5

Wtest <- Wgenerate(alpha=alphatest,eta=etatest,X=Xtest,Z=Ztest,sigma=sigmatetest)

eta_initial <- matrix(c(rnorm(3),0,0,0),etasize,classsize)
alpha_initial<- matrix(rnorm(alphasize*classsize)*3,alphasize,classsize)
sigma_initial <- 1

EMtheta <- EMalgorithm(X=Wtest$X,Z=Wtest$Z,Y=Wtest$Y,classsize=2,
                      etat=eta_initial,alpat=alpha_initial,sigmat=sigma_initial,
                      learning_rate=0.01,regular_parameter_eta=0.001,
                      max_iteration=1000,max_iteration_eta=10000,
                      compact_flag = TRUE, C0 = 5, C1 = 2, C2 = 9)

```

EMbootstrap

*Bootstrap Method***Description**

Estimate the value of parameters several times by bootstrap method where the parameters are estimated by EM algorithm. In this way, we can observe the distribution of parameter.

Usage

```
EMbootstrap(X, Y, Z, samplesize, boots_samplesize, boots_experiments, classsize = 2,
            learning_rate = 0.1, regular_parameter_eta = 0.001, max_iteration = 10000,
            max_iteration_eta = 10000, compact_flag = FALSE, C0 = 5, C1 = 2, C2 = 9)
```

Arguments

X	the co-variables of the mean of each subgroup
Y	the respond variable
Z	the co-variables determining subgroup
samplesize	the size of this sample.
boots_samplesize	the size of chosen sample in one step of bootstrap
boots_experiments	the steps of bootstrap
classsize	the number of subgroup types in your model assumption
learning_rate	learning rate of updating eta
regular_parameter_eta	regular value of updating eta by gradient descending method.
max_iteration	maximum steps of iteration to avoid unlimited looping.
max_iteration_eta	maximal steps of eta iteration to avoid unlimited looping.
compact_flag	if the value of eta is limited in a compact set, set it TRUE
C0	the maximum of intercept of eta.
C1	the minimum of the norm of slope of eta
C2	the maximum of the norm of slope of eta

Details

Actually, the method can be extended to other parameter estimation where the standard error of parameter can't be calculated in a simple way.

Value

alpha	alpha estimated by bootstrap method.
eta	eta estimated by bootstrap method.
sigma	sigma estimated by bootstrap method.

Author(s)

Linsui Deng

Examples

```
#parameter initialization
etasize <- 2
classsize <- 2
alphasize <- 3
samplesize <- 1000

etatest <- matrix(c(1,1,
                    0,0),etasize,classsize)

alphatest <- matrix(c(1,0,2,
                     4,3,5),alphasize,classsize)
sigmatest <- 0.5

#test of EMbootstrap
X <- matrix(c(matrix(1,samplesize),
               rnorm(samplesize*(etasize-1))+1),samplesize,etasize)
Z <- matrix(c(matrix(1,samplesize),rbinom(prob=1/2,size=1,n=samplesize),
               rnorm(samplesize*(alphasize-2))+1),samplesize,alphasize)

Wtest <- Wgenerate(alpha=alphatest,eta=etatest,sigma=sigmatest,X=X,Z=Z)

boots_samplesize <- 100
boots_experiments <- 30
samplesize <- dim(Wtest$X)[1]
EMbootstrap_theta <- EMbootstrap(Wtest$X,Wtest$Y,Wtest$Z,samplesize,
                                boots_samplesize,boots_experiments,
                                classsize=2,compact_flag=TRUE,C0=5,C1=0.2,C2=5)
index <- which(EMbootstrap_theta$sigma<0.8)
EMsimulation_result_sort <- EM_result_sort(EMbootstrap_theta$alpha[index,,],
                                           EMbootstrap_theta$eta[index,,])
EM_parameter <- EM_parameter_sd(EMsimulation_result_sort$alpha,
                               EMsimulation_result_sort$eta,
                               EMbootstrap_theta$sigma[index])
```

Description

It simulates the experiments with given alpha, eta and sigma to verify the EMalgorithm

Usage

```
EMsimulation(eta, alpha, sigma, samplesize, expriments,
             compact_flag = FALSE, C0 = 5, C1 = 2, C2 = 9)
```

Arguments

eta	the true value of eta
alpha	the true value of alpha
sigma	the true value of sigma
samplesize	the size of sample
expriments	the times of experiments
compact_flag	if the value of eta is limited in a compact set, set it TRUE
C0	the maximum of intercept of eta.
C1	the minimum of the norm of slope of eta
C2	the maximum of the norm of slope of eta

Value

alpha	alpha estimated in simulation.
eta	eta estimated in simulation.
sigma	sigma estimated in simulation.

Author(s)

Linsui Deng

Examples

```
#parameter initialization
etasize <- 2
classsize <- 2
alphasize <- 3
samplesize <- 100
expriments <- 30

etatest <- matrix(c(1,1,
                   0,0),etasize,classsize)

alphatest <- matrix(c(1,0,2,
                     4,3,5),alphasize,classsize)
sigmatest <- 0.5

#test of EMsimulation
```

```
EMsimulation_result <- EMsimulation(eta=etatest,alpha=alphatest,sigma=sigmatetest,
                                   samplesize=samplesize,experiments=experiments,
                                   compact_flag=TRUE,C0=5,C1=0.5,C2=5)
```

EM_parameter_sd	<i>Bootstrap Parameter Inference</i>
-----------------	--------------------------------------

Description

Estimating the parameters and their stand error through the sorted parameters estimated by bootstrap method.

Usage

```
EM_parameter_sd(EMsimulation_sort_alpha, EMsimulation_sort_eta, EMsimulation_sort_sigma)
```

Arguments

```
EMsimulation_sort_alpha
    sorted alpha estimated by bootstrap method.
EMsimulation_sort_eta
    sorted eta estimated by bootstrap method.
EMsimulation_sort_sigma
    sorted sigma estimated by bootstrap method.
```

Value

sigma	the estimation of sigma
sigma_sd	the estimation of standard error of sigma
alpha	the estimation of alpha
alpha_sd	the estimation of standard error of alpha
eta	the estimation of eta
eta_sd	the estimation of standard error of eta

Author(s)

Linsui Deng

Examples

```
#parameter initialization
etasize <- 2
classsize <- 2
alphasize <- 3
samplesize <- 100
experiments <- 30
```

```

etatest <- matrix(c(1,1,
                    0,0),etasize,classsize)

alphatest <- matrix(c(1,0,2,
                     4,3,5),alphasize,classsize)
sigmatest <- 0.5

EMsimulation_result <- EMsimulation(eta=etatest,alpha=alphatest,sigma=sigmatest,
                                   samplesize=samplesize,experiments=experiments,
                                   compact_flag=TRUE,C0=5,C1=0.5,C2=5)
index <- which(EMsimulation_result$sigma<0.8)
EMsimulation_result_sort <- EM_result_sort(EMsimulation_result$alpha[index,,],
                                           EMsimulation_result$eta[index,,])

#test of EM_parameter_sd
EM_parameter <- EM_parameter_sd(EMsimulation_result_sort$alpha,
                               EMsimulation_result_sort$eta,
                               EMsimulation_result$sigma[index])

```

EM_result_sort	<i>Sort Parameter</i>
----------------	-----------------------

Description

Since the number of subgroup types is always beyond 1, the order of subgroup may be different, we can sort them in this function.

Usage

```
EM_result_sort(EMsimulation_result_alpha, EMsimulation_result_eta)
```

Arguments

```

EMsimulation_result_alpha
    alpha estimated by bootstrap method.
EMsimulation_result_eta
    eta estimated by bootstrap method.

```

Value

alpha	sorted alpha estimated by bootstrap method.
eta	sorted eta estimated by bootstrap method.

Author(s)

Linsui Deng

Examples

```
#parameter initialization
etasize <- 2
classsize <- 2
alphasize <- 3
samplesize <- 100
experiments <- 30

etatest <- matrix(c(1,1,
                    0,0),etasize,classsize)

alphatest <- matrix(c(1,0,2,
                     4,3,5),alphasize,classsize)

sigmatest <- 0.5

#test of EMsimulation
EMsimulation_result <- EMsimulation(eta=etatest,alpha=alphatest,sigma=sigmatest,
                                   samplesize=samplesize,experiments=experiments,
                                   compact_flag=TRUE,C0=5,C1=0.5,C2=5)

#test of EM_result_sort
EMsimulation_result_sort <- EM_result_sort(EMsimulation_result$alpha,
                                           EMsimulation_result$eta)
```

fnorm

*Density Value***Description**

Calculate the density value of respond value Y under each mean and homogeneous variance.

Usage

```
fnorm(Y, mu, sigma)
```

Arguments

Y	the respond variable
mu	different mean of each subgroup
sigma	standard error

Value

the density value of Y under different mu and common sigma.

Author(s)

Linsui Deng

Examples

```
fnormtest <- fnorm(matrix(1:6,3,2),matrix(seq(1,3,length=6),3,2),1)
```

Ggenerate

*Subgroup Determination***Description**

In data generation, determining the subgroup of each item belonging to through random number and Sigmoid Link function.

Usage

```
Ggenerate(eta, X, seed = 0)
```

Arguments

eta	the coefficients determining subgroup
X	the covariables determining subgroup
seed	random seed

Value

the classes items belonging to, it's a vector. If X1 belongs to class 3, then the 1st row 3rd column is 1 and the rest of 1st row are 0.

Author(s)

Linsui Deng

Examples

```
#some variables
samplesize <- 1000
classsize <- 6
etasize <- 3
alphasize <- 2

#test of Ggenerate
Xtest <- data.frame(matrix(rnorm(samplesize*etasize),samplesize,etasize))
etatest <- matrix(seq(1.15,1,length=etasize*classsize),etasize,classsize)

Gtest1 <- Ggenerate(etatest,Xtest)
Gtest2 <- Ggenerate(etatest,Xtest,1)
```

`softmax`*Softmax Value*

Description

Calculate the Softmax Value of each subgroup to represent the probability of items belonging to specific class.

Usage

```
softmax(eta, X)
```

Arguments

<code>eta</code>	the coefficients determining subgroup
<code>X</code>	the covariables determining subgroup

Value

Softmax Value of each subgroup

Author(s)

Linsui Deng

Examples

```
#some variables
samplesize <- 1000
classsize <- 6
etasize <- 3
alphasize <- 2

#test of softmax
Xtest <- data.frame(matrix(rnorm(samplesize*etasize),samplesize,etasize))
etatest <- matrix(seq(1.15,1,length=etasize*classsize),etasize,classsize)
softmax_value <- softmax(etatest,Xtest)
```

`standard`*Data Standardlization*

Description

Standardize the data to elimilate the effect of scale

Usage

```
standard(X)
```

Arguments

X the original data

Value

standarlized data

Author(s)

Linsui Deng

Examples

```
#data generata  
Y <- rnorm(100)*2+5  
Y_sta <- standard(Y)
```

`update_eta`*Updata Eta*

Description

Updata eta in step t+1 with given data and coeffients estimated in step t.

Usage

```
update_eta(fun, alphas, sigmas, etas, X, Y, Z, learning_rate_eta = 0.001,  
           regular_parameter_eta = 0.001, max_iteration_eta = 10000)
```

Arguments

fun	the function updata eta
alphan	the estimated coefficients of the mean of each subgroup in step t
sigmat	the estimated standard error of Y in step t
etat	the estimated coefficients determining subgroup in step t
X	the covariables of the mean of each subgroup
Z	the covariables determining subgroup
Y	the respond variable
learning_rate_eta	learning rate of updating eta
regular_parameter_eta	regular value of updating eta by gradient descending method.
max_iteration_eta	maximal steps of eta iteration to avoid unlimited looping.

Value

alpha	alpha estimated in step t.
eta	eta estimated in step t+1.
sigma	sigma estimated in step t.

Author(s)

Linsui Deng

Examples

```
#some variables
samplesize <- 1000
classsize <- 6
etasize <- 3
alphasize <- 2

Xtest <- data.frame(matrix(rnorm(samplesize*etasize),samplesize,etasize))
Ztest <- matrix(rnorm(samplesize*alphasize),samplesize,alphasize)

etatest <- matrix(seq(1.15,1,length=etasize*classsize),etasize,classsize)
alphatest <- matrix(seq(1.15,1,length=alphasize*classsize),alphasize,classsize)
sigmatest <- 0.1

Wtest <- Wgenerate(alpha=alphatest,eta=etatest,X=Xtest,Z=Ztest)

#test of update_eta
thetupdate_eta <- update_eta(fun=eta_gradient_fun,alphan=alphatest,sigmat=sigmatest,
                             etat=etatest,X=Wtest$X,Z=Wtest$Z,Y=Wtest$Y,
                             learning_rate=0.1,regular_parameter=0.001,max_iteration=10000)
```

update_gamma	<i>Udata Alpha and Sigma</i>
--------------	------------------------------

Description

Udata alpha and sigma in t+1 step with given data and coefficients estimated in step t.

Usage

```
update_gamma(alphat, sigmat, etat, X, Z, Y)
```

Arguments

alphat	the estimated coefficients of the mean of each subgroup in step t
sigmat	the estimated standard error of Y in step t
etat	the estimated coefficients determining subgroup in step t
X	the covariables of the mean of each subgroup
Z	the covariables determining subgroup
Y	the respond variable

Value

alpha	alpha estimated in step t+1.
eta	eta estimated in step t.
sigma	sigma estimated in step t+1.

Author(s)

Linsui Deng

Examples

```
#some variables
samplesize <- 1000
classsize <- 6
etasize <- 3
alphasize <- 2

Xtest <- data.frame(matrix(rnorm(samplesize*etasize),samplesize,etasize))
Ztest <- matrix(rnorm(samplesize*alphasize),samplesize,alphasize)

etatest <- matrix(seq(1.15,1,length=etasize*classsize),etasize,classsize)
alphatest <- matrix(seq(1.15,1,length=alphasize*classsize),alphasize,classsize)
sigmatest <- 0.1

Wtest <- Wgenerate(alpha=alphatest,eta=etatest,X=Xtest,Z=Ztest)
```

```
#test of updata_gamma
thetaupdate_gamma <- update_gamma(alphat=alphatest,sigmat=sigmatetest,
                                   etat=etatest,X=Wtest$X,Z=Wtest$Z,Y=Wtest$Y)
```

weight_matrix	<i>Weighted Inner Product</i>
---------------	-------------------------------

Description

Calculate the weighted inner product of A and B with the weight W. This result is useful in Logistic Regression.

Usage

```
weight_matrix(A, W, B)
```

Arguments

A	the left matrix
W	the weight
B	the right matrix

Value

the wighted inner product, noticing it can be vector when A or B is 2 dimension.

Author(s)

Linsui Deng

Examples

```
#data generation
A <- matrix(rnorm(200),100,2)
W <- rnorm(100)
B <- matrix(rnorm(100),100,1)
weighted <- weight_matrix(A,W,B)
```

Wgenerate

Sigmoid Logistic Data Generation

Description

Generate data satisfies Sigmoid Logistic Model to check EMalgorithm.

Usage

```
Wgenerate(alpha, sigma = 1, eta, samplesize = 0, X, Z, seed1 = 0, seed2 = 0)
```

Arguments

alpha	the coefficients of the mean of each subgroup
sigma	the variance of Y
eta	the coefficients determining subgroup
samplesize	the size of sample you wanna generate
X	the covariables of the mean of each subgroup
Z	the covariables determining subgroup
seed1	random seed of generating Y
seed2	random seed of generating G

Value

X	the covariables of the mean of each subgroup
Z	the covariables determining subgroup
Y	the generated respond variable
G	the classes items belonging to

Author(s)

Linsui Deng

Examples

```
#some variables
samplesize <- 1000
classsize <- 6
etasize <- 3
alphasize <- 2

#test of Wgenerate
Xtest <- data.frame(matrix(rnorm(samplesize*etasize),samplesize,etasize))
Ztest <- matrix(rnorm(samplesize*alphasize),samplesize,alphasize)
```

```
etatest <- matrix(seq(1.15,1,length=etasize*classsize),etasize,classsize)
alphatest <- matrix(seq(1.15,1,length=alphasize*classsize),alphasize,classsize)

Wtest <- Wgenerate(alpha=alphatest,eta=etatest,X=Xtest,Z=Ztest)
```

Index

- * **algebra**
 - softmax, [14](#)
 - standard, [15](#)
 - weight_matrix, [18](#)
 - * **arith**
 - EM_result_sort, [11](#)
 - * **array**
 - softmax, [14](#)
 - standard, [15](#)
 - weight_matrix, [18](#)
 - * **datagen**
 - Ggenerate, [13](#)
 - Wgenerate, [19](#)
 - * **design**
 - EMsimulation, [8](#)
 - * **distribution**
 - fnorm, [12](#)
 - * **htest**
 - EM_parameter_sd, [10](#)
 - * **linear**
 - update_gamma, [17](#)
 - * **math**
 - Ccompute, [4](#)
 - * **models**
 - EMalgorithm, [5](#)
 - update_eta, [15](#)
 - update_gamma, [17](#)
 - * **multivariate**
 - EMalgorithm, [5](#)
 - update_eta, [15](#)
 - * **nonlinear**
 - EMalgorithm, [5](#)
 - update_eta, [15](#)
 - * **nonparametric**
 - EM_parameter_sd, [10](#)
 - EMbootstrap, [7](#)
 - * **package**
 - EMSNM-package, [2](#)
- EM_parameter_sd, [10](#)
- EM_result_sort, [11](#)
- EMalgorithm, [5](#)
- EMbootstrap, [7](#)
- EMsimulation, [8](#)
- EMSNM (EMSNM-package), [2](#)
- EMSNM-package, [2](#)
- fnorm, [12](#)
- Ggenerate, [13](#)
- softmax, [14](#)
- standard, [15](#)
- update_eta, [15](#)
- update_gamma, [17](#)
- weight_matrix, [18](#)
- Wgenerate, [19](#)
- Ccompute, [4](#)