

# Package ‘DOEM’

March 30, 2022

**Title** The Distributed Online Expectation Maximization Algorithms to Solve Parameters of Poisson Mixture Models

**Version** 0.0.0.1

**Description** The distributed online expectation maximization algorithms are used to solve parameters of Poisson mixture models. The philosophy of the package is described in Guo, G. (2022) <[doi:10.1080/02664763.2022.2053949](https://doi.org/10.1080/02664763.2022.2053949)>.

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.1.2

**Imports** stats

**Suggests** testthat (>= 3.0.0)

**Config/testthat/edition** 3

**NeedsCompilation** no

**Author** Qian Wang [aut, cre],  
Guangbao Guo [aut],  
Guoqi Qian [aut]

**Maintainer** Qian Wang <[waqian0715@163.com](mailto:waqian0715@163.com)>

**Depends** R (>= 3.5.0)

**Repository** CRAN

**Date/Publication** 2022-03-30 07:20:13 UTC

## R topics documented:

DE_OEM	2
DMOEM	3
DM_OEM	5
E_OEM	6
M_OEM	7
PeMSD3	8
PeMSD7	9

PeMSD7_weekend . . . . .	11
PeMSD7_workday . . . . .	12

<b>Index</b>	<b>14</b>
--------------	-----------

---

DE_OEM	<i>The DE-OEM algorithm replaces E-step with stochastic step in distributed manner, which is used to solve the parameter estimation of Poisson mixture model.</i>
--------	---

---

### Description

The DE-OEM algorithm replaces E-step with stochastic step in distributed manner, which is used to solve the parameter estimation of Poisson mixture model.

### Usage

```
DE_OEM(y, M, K, seed, alpha0, lambda0, a, b)
```

### Arguments

y	is a vector
M	is the number of subsets
K	is the number of Poisson distribution
seed	is the recommended way to specify seeds
alpha0	is the initial value of the mixing weight
lambda0	is the initial value of the mean
a	represents the power of the reciprocal of the step size
b	indicates that the M-step is not implemented for the first b data points

### Value

DE\_OEMtime, DE\_OEMalpha, DE\_OEMlambda

### Examples

```
library(stats)
set.seed(637351)
K=5
alpha1=c(rep(1/K,K))
lambda1=c(1,2,3,4,5)
n=300
U=sample(c(1:n),n,replace=FALSE)
y= c(rep(0,n))
for(i in 1:n){
  if(U[i]<=0.2*n){
    y[i] = rpois(1,lambda1[1])}
  else if(U[i]>0.2*n & U[i]<=0.4*n){
```

```

y[i] = rpois(1,lambda1[2])}
else if(U[i]>0.4*n & U[i]<=0.6*n){
y[i] = rpois(1,lambda1[3])}
else if(U[i]>0.6*n & U[i]<=0.8*n){
y[i] = rpois(1,lambda1[4])}
else if(U[i]>0.8*n ){
y[i] = rpois(1,lambda1[5])}
}
M=5
seed=637351
set.seed(123)
e=sample(c(1:n),K)
alpha0=e/sum(e)
lambda0=c(1.5,2.5,3.5,4.5,5.5)
a=1
b=5
DE_OEM(y,M,K,seed,alpha0,lambda0,a,b)

```

DMOEM

*The DMOEM is an overrelaxation algorithm in distributed manner, which is used to solve the parameter estimation of Poisson mixture model.*

### Description

The DMOEM is an overrelaxation algorithm in distributed manner, which is used to solve the parameter estimation of Poisson mixture model.

### Usage

```

DMOEM(
  y,
  M,
  K,
  seed,
  alpha0,
  lambda0,
  MOEMalpha0,
  MOEMlambda0,
  omega,
  T,
  epsilon
)

```

### Arguments

**y** is a data matrix  
**M** is the number of subsets

K	is the number of Poisson distribution
seed	is the recommended way to specify seeds
alpha0	is the initial value of the mixing weight under the EM algorithm
lambda0	is the initial value of the mean under the EM algorithm
MOEMalpha0	is the initial value of the mixing weight under the monotonically overrelaxed EM algorithm
MOEMlambda0	is the initial value of the mean under the monotonically overrelaxed EM algorithm
omega	is the overrelaxation factor
T	is the number of iterations
epsilon	is the threshold value

### Value

DMOEMtime,DMOEMalpha,DMOEMlambda

### Examples

```

library(stats)
set.seed(637351)
K=5
alpha1=c(rep(1/K,K))
lambda1=c(1,2,3,4,5)
n=300
U=sample(c(1:n),n,replace=FALSE)
y= c(rep(0,n))
for(i in 1:n){
  if(U[i]<=0.2*n){
    y[i] = rpois(1,lambda1[1])}
  else if(U[i]>0.2*n & U[i]<=0.4*n){
    y[i] = rpois(1,lambda1[2])}
  else if(U[i]>0.4*n & U[i]<=0.6*n){
    y[i] = rpois(1,lambda1[3])}
  else if(U[i]>0.6*n & U[i]<=0.8*n){
    y[i] = rpois(1,lambda1[4])}
  else if(U[i]>0.8*n ){
    y[i] = rpois(1,lambda1[5])}
  }
M=5
seed=637351
set.seed(123)
e=sample(c(1:n),K)
alpha0= MOEMalpha0=e/sum(e)
lambda0= MOEMlambda0=c(1.5,2.5,3.5,4.5,5.5)
omega=0.8
T=10
epsilon=0.005
DMOEM(y,M,K,seed,alpha0,lambda0,MOEMalpha0,MOEMlambda0,omega,T,epsilon)

```

---

DM_OEM	<i>The DM-OEM algorithm replaces M-step with stochastic step in distributed manner, which is used to solve the parameter estimation of Poisson mixture model.</i>
--------	---

---

### Description

The DM-OEM algorithm replaces M-step with stochastic step in distributed manner, which is used to solve the parameter estimation of Poisson mixture model.

### Usage

```
DM_OEM(y, M, K, seed, alpha0, lambda0, a, b)
```

### Arguments

y	is a vector
M	is the number of subsets
K	is the number of Poisson distribution
seed	is the recommended way to specify seeds
alpha0	is the initial value of the mixing weight
lambda0	is the initial value of the mean
a	represents the power of the reciprocal of the step size
b	indicates that the M-step is not implemented for the first b data points

### Value

DM\_OEMtime,DM\_OEMalpha,DM\_OEMlambda

### Examples

```
library(stats)
set.seed(637351)
K=5
alpha1=c(rep(1/K,K))
lambda1=c(1,2,3,4,5)
n=300
U=sample(c(1:n),n,replace=FALSE)
y= c(rep(0,n))
for(i in 1:n){
  if(U[i]<=0.2*n){
    y[i] = rpois(1,lambda1[1])}
  else if(U[i]>0.2*n & U[i]<=0.4*n){
    y[i] = rpois(1,lambda1[2])}
  else if(U[i]>0.4*n & U[i]<=0.6*n){
    y[i] = rpois(1,lambda1[3])}
  else if(U[i]>0.6*n & U[i]<=0.8*n){
```

```

y[i] = rpois(1,lambda1[4])}
else if(U[i]>0.8*n ){
y[i] = rpois(1,lambda1[5])}
}
M=5
seed=637351
set.seed(123)
e=sample(c(1:n),K)
alpha0=e/sum(e)
lambda0=c(1.5,2.5,3.5,4.5,5.5)
a=1
b=5
DM_OEM(y,M,K,seed,alpha0,lambda0,a,b)

```

---

E\_OEM

*The E-OEM algorithm replaces E-step with stochastic step, which is used to solve the parameter estimation of Poisson mixture model.*

---

### Description

The E-OEM algorithm replaces E-step with stochastic step, which is used to solve the parameter estimation of Poisson mixture model.

### Usage

```
E_OEM(y, K, alpha0, lambda0, a, b)
```

### Arguments

y	is a data vector
K	is the number of Poisson distribution
alpha0	is the initial value of the mixing weight
lambda0	is the initial value of the mean
a	represents the power of the reciprocal of the step size
b	indicates that the M-step is not implemented for the first b data points

### Value

E\_OEMtime,E\_OEMalpha,E\_OEMlambda

### Examples

```

library(stats)
set.seed(637351)
K=5
alpha1=c(rep(1/K,K))
lambda1=c(1,2,3,4,5)
n=300

```

```

U=sample(c(1:n),n,replace=FALSE)
y= c(rep(0,n))
for(i in 1:n){
  if(U[i]<=0.2*n){
    y[i] = rpois(1,lambda1[1])}
  else if(U[i]>0.2*n & U[i]<=0.4*n){
    y[i] = rpois(1,lambda1[2])}
  else if(U[i]>0.4*n & U[i]<=0.6*n){
    y[i] = rpois(1,lambda1[3])}
  else if(U[i]>0.6*n & U[i]<=0.8*n){
    y[i] = rpois(1,lambda1[4])}
  else if(U[i]>0.8*n ){
    y[i] = rpois(1,lambda1[5])}
}
M=5
seed=637351
set.seed(123)
e=sample(c(1:n),K)
alpha0=e/sum(e)
lambda0=c(1.5,2.5,3.5,4.5,5.5)
a=0.75
b=5
E_OEM(y,K,alpha0,lambda0,a,b)

```

---

M\_OEM

*The M-OEM algorithm replaces M-step with stochastic step, which is used to solve the parameter estimation of Poisson mixture model.*

---

### Description

The M-OEM algorithm replaces M-step with stochastic step, which is used to solve the parameter estimation of Poisson mixture model.

### Usage

M\_OEM(y, K, alpha0, lambda0, a, b)

### Arguments

y	is a data vector
K	is the number of Poisson distribution
alpha0	is the initial value of the mixing weight
lambda0	is the initial value of the mean
a	represents the power of the reciprocal of the step size
b	indicates that the M-step is not implemented for the first b data points

### Value

M\_OEMtime,M\_OEMalpha,M\_OEMlambda

**Examples**

```

library(stats)
set.seed(637351)
K=5
alpha1=c(rep(1/K,K))
lambda1=c(1,2,3,4,5)
n=300
U=sample(c(1:n),n,replace=FALSE)
y= c(rep(0,n))
for(i in 1:n){
  if(U[i]<=0.2*n){
    y[i] = rpois(1,lambda1[1])}
  else if(U[i]>0.2*n & U[i]<=0.4*n){
    y[i] = rpois(1,lambda1[2])}
  else if(U[i]>0.4*n & U[i]<=0.6*n){
    y[i] = rpois(1,lambda1[3])}
  else if(U[i]>0.6*n & U[i]<=0.8*n){
    y[i] = rpois(1,lambda1[4])}
  else if(U[i]>0.8*n ){
    y[i] = rpois(1,lambda1[5])}
}
M=5
seed=637351
set.seed(123)
e=sample(c(1:n),K)
alpha0=e/sum(e)
lambda0=c(1.5,2.5,3.5,4.5,5.5)
a=0.75
b=5
M_OEM(y,K,alpha0,lambda0,a,b)

```

---

 PeMSD3

*PeMSD3*


---

**Description**

The PeMSD3 data

**Usage**

```
data("PeMSD3")
```

**Format**

A data frame with 26208 observations on the following 12 variables.

X315836 a numeric vector

X315837 a numeric vector

X315838 a numeric vector



X315841 a numeric vector  
X315842 a numeric vector  
X315839 a numeric vector  
X315843 a numeric vector  
X315846 a numeric vector  
X315847 a numeric vector  
X317895 a numeric vector  
X315849 a numeric vector  
X315850 a numeric vector

### Details

It is the traffic data of Sacramento in California, the United States.

### Source

Song, C., Lin, Y., Guo, S., Wan, H. Spatial-temporal synchronous graph convolutional networks: a new framework for spatial-temporal network data forecasting[C]. Proceedings of the AAAI Conference on Artificial Intelligence, 34(1), 914-921.

### References

Song, C., Lin, Y., Guo, S., Wan, H. Spatial-temporal synchronous graph convolutional networks: a new framework for spatial-temporal network data forecasting[C]. Proceedings of the AAAI Conference on Artificial Intelligence, 34(1), 914-921.

### Examples

```
data(PeMSD3)
## maybe str(PeMSD3) ; plot(PeMSD3) ...
```

---

PeMSD7

*PeMSD7*

---

### Description

The PeMSD7 data

### Usage

```
data("PeMSD7")
```

**Format**

A data frame with 17568 observations on the following 20 variables.

X773656 a numeric vector  
X760074 a numeric vector  
X760080 a numeric vector  
X716414 a numeric vector  
X760101 a numeric vector  
X760112 a numeric vector  
X716419 a numeric vector  
X716421 a numeric vector  
X716424 a numeric vector  
X765476 a numeric vector  
X760167 a numeric vector  
X716427 a numeric vector  
X716431 a numeric vector  
X716433 a numeric vector  
X760187 a numeric vector  
X760196 a numeric vector  
X716440 a numeric vector  
X760226 a numeric vector  
X760236 a numeric vector  
X716449 a numeric vector

**Details**

It is the traffic data of Los Angeles in California, the United States.

**Source**

Xu, D., Wei, C., Peng, P., Xuan, Q., Guo, H. Ge-gan: a novel deep learning framework for road traffic state estimation[J]. Transportation Research Part C Emerging Technologies, 2020, 117, 102635.

**References**

Xu, D., Wei, C., Peng, P., Xuan, Q., Guo, H. Ge-gan: a novel deep learning framework for road traffic state estimation[J]. Transportation Research Part C Emerging Technologies, 2020, 117, 102635.

**Examples**

```
data(PeMSD7)  
## maybe str(PeMSD7) ; plot(PeMSD7) ...
```

---

PeMSD7_weekend	<i>PeMSD7 on weekend</i>
----------------	--------------------------

---

**Description**

The weekend data in PeMSD7

**Usage**

```
data("PeMSD7_weekend")
```

**Format**

A data frame with 5184 observations on the following 20 variables.

X773656 a numeric vector  
X760074 a numeric vector  
X760080 a numeric vector  
X716414 a numeric vector  
X760101 a numeric vector  
X760112 a numeric vector  
X716419 a numeric vector  
X716421 a numeric vector  
X716424 a numeric vector  
X765476 a numeric vector  
X760167 a numeric vector  
X716427 a numeric vector  
X716431 a numeric vector  
X716433 a numeric vector  
X760187 a numeric vector  
X760196 a numeric vector  
X716440 a numeric vector  
X760226 a numeric vector  
X760236 a numeric vector  
X716449 a numeric vector

**Details**

The weekend data set only records traffic flow data on weekends.

**Source**

Xu, D., Wei, C., Peng, P., Xuan, Q., Guo, H. Ge-gan: a novel deep learning framework for road traffic state estimation[J]. Transportation Research Part C Emerging Technologies, 2020, 117, 102635.

**References**

Xu, D., Wei, C., Peng, P., Xuan, Q., Guo, H. Ge-gan: a novel deep learning framework for road traffic state estimation[J]. Transportation Research Part C Emerging Technologies, 2020, 117, 102635.

**Examples**

```
data(PeMSD7_weekend)
## maybe str(PeMSD7_weekend) ; plot(PeMSD7_weekend) ...
```

---

PeMSD7_workday	<i>PeMSD7 on workday</i>
----------------	--------------------------

---

**Description**

The workday data in PeMSD7

**Usage**

```
data("PeMSD7_workday")
```

**Format**

A data frame with 12384 observations on the following 20 variables.

X773656 a numeric vector  
 X760074 a numeric vector  
 X760080 a numeric vector  
 X716414 a numeric vector  
 X760101 a numeric vector  
 X760112 a numeric vector  
 X716419 a numeric vector  
 X716421 a numeric vector  
 X716424 a numeric vector  
 X765476 a numeric vector  
 X760167 a numeric vector  
 X716427 a numeric vector  
 X716431 a numeric vector  
 X716433 a numeric vector

X760187 a numeric vector  
X760196 a numeric vector  
X716440 a numeric vector  
X760226 a numeric vector  
X760236 a numeric vector  
X716449 a numeric vector

### Details

The workday data set only records traffic flow data on workdays.

### Source

Xu, D., Wei, C., Peng, P., Xuan, Q., Guo, H. Ge-gan: a novel deep learning framework for road traffic state estimation[J]. Transportation Research Part C Emerging Technologies, 2020, 117, 102635.

### References

Xu, D., Wei, C., Peng, P., Xuan, Q., Guo, H. Ge-gan: a novel deep learning framework for road traffic state estimation[J]. Transportation Research Part C Emerging Technologies, 2020, 117, 102635.

### Examples

```
data(PeMSD7_workday)  
## maybe str(PeMSD7_workday) ; plot(PeMSD7_workday) ...
```

# Index

## \* datasets

PeMSD3, [8](#)

PeMSD7, [9](#)

PeMSD7\_weekend, [11](#)

PeMSD7\_workday, [12](#)

DE\_OEM, [2](#)

DM\_OEM, [5](#)

DMOEM, [3](#)

E\_OEM, [6](#)

M\_OEM, [7](#)

PeMSD3, [8](#)

PeMSD7, [9](#)

PeMSD7\_weekend, [11](#)

PeMSD7\_workday, [12](#)