Package ‘mapfit’

May 30, 2022

Version 0.9.9

Title A Tool for PH/MAP Parameter Estimation

Type Package

Maintainer Hiroyuki Okamura <okamu@hiroshima-u.ac.jp>

Description Estimation methods for phase-type distribution (PH) and Markovian arrival process (MAP) from empirical data (point and grouped data) and density function.

Encoding UTF-8

License MIT + file LICENSE

RoxygenNote 7.2.0

Imports deformula, Matrix, methods

URL https://github.com/okamumu/mapfit

BugReports https://github.com/okamumu/mapfit/issues

Suggests ggplot2

NeedsCompilation yes

Author Hiroyuki Okamura [aut, cre] (<https://orcid.org/0000-0001-6881-0593>)

Repository CRAN

Date/Publication 2022-05-30 06:40:11 UTC

R topics documented:

`mapfit-package` *mapfit: A Tool for PH/MAP Parameter Estimation*

Description

Estimation methods for phase-type distribution (PH) and Markovian arrival process (MAP) from empirical data (point and grouped data) and density function.

Author(s)

Maintainer: Hiroyuki Okamura <okamu@hiroshima-u.ac.jp> (ORCID)

See Also

Useful links:

- https://github.com/okamumu/mapfit
- Report bugs at https://github.com/okamumu/mapfit/issues

`BCpAug89` *Packet Trace Data*

Description

The data contains packet arrivals seen on an Ethernet at the Bellcore Morristown Research and Engineering facility. Two of the traces are LAN traffic (with a small portion of transit WAN traffic), and two are WAN traffic. The original trace BC-pAug89 began at 11:25 on August 29, 1989, and ran for about 3142.82 seconds (until 1,000,000 packets had been captured). The trace BC-pOct89 began at 11:00 on October 5, 1989, and ran for about 1759.62 seconds. These two traces captured all Ethernet packets. The number of arrivals in the original trace is one million.

Format

`BCpAug89` is a vector for the interarrival time in sencons for 1000 arrivals.

Source

The original trace data are published in http://ita.ee.lbl.gov/html/contrib/BC.html.

`cf1` *Canonical Form 1 for Phase-Type (PH) Distribution*

Description

A function to generate an object of `cf1`.

Usage

```
cf1(size, alpha, rate, class = "CsparseMatrix")
```

Arguments

`size`	A value for the number of phases.
`alpha`	A vector for the initial probabilities of PH distribution.
`rate`	A vector for transition rates to next phase (diagonal elements of Q).
`class`	Name of Matrix class for `Q`.

Details

- The PH distribution with parameters α, Q and $\xi = -Q1$:
- Cumulative probability function;

$$F(q) = 1 - \alpha \exp(Qq)1$$

- Probability density function;

$$f(x) = \alpha \exp(Qx)\xi,$$

  where Q is a bidiagonal matrix whose entries are sorted.

Value

`cf1` gives an object of canonical form 1 that is a subclass of PH distribution.

See Also

`ph`, `herlang`

Examples

```
## create a CF1 with 5 phases
(param1 <- cf1(5))

## create a CF1 with 5 phases
(param1 <- cf1(size=5))

## create a CF1 with specific parameters
(param2 <- cf1(alpha=c(1,0,0), rate=c(1.0,2.0,3.0)))

## p.d.f. for 0, 0.1, ..., 1
(dph(x=seq(0, 1, 0.1), ph=param2))

## c.d.f. for 0, 0.1, ..., 1
(pph(q=seq(0, 1, 0.1), ph=param2))

## generate 10 samples (this is quiker than rph with general ph)
(rph(n=10, ph=param2))
```

---

cf1-class *Class of canonical form 1 (PH distribution)*

Description

Parameters for a canonical form 1 which is a subclass of PH. This is extended from ph.

Slots

`rate` Transition rates to the next phase.

`size` The number of phases (transient states).

`alpha` A probability (row) vector to decide an initial phase.

`Q` A square matrix that means transition rates between phases.

`xi` A column vector for exiting rates from phases to an absorbing state.

`df` The number of free parameters.

Note

Objects are usually created by a cf1.

Methods:

`ph.moment` `signature(ph = "cf1")`: ...

emfit.init `signature(model = "cf1")`: ...

emfit.mstep `signature(model = "cf1")`: ...

See Also

Classes ph and herlang.

Examples

```
## create a CF1 with 5 phases
(param1 <- cf1(5))

## create a CF1 with 5 phases
(param1 <- cf1(size=5))

## create a CF1 with specific parameters
(param2 <- cf1(alpha=c(1,0,0), rate=c(1.0,2.0,3.0)))

## p.d.f. for 0, 0.1, ..., 1
(dph(x=seq(0, 1, 0.1), ph=param2))

## c.d.f. for 0, 0.1, ..., 1
(pph(q=seq(0, 1, 0.1), ph=param2))

## generate 10 samples (this is quiker than rph with general ph)
(rph(n=10, ph=param2))
```

`diag.padding.zero` *Padding zero to omitted diagonal elements*

Description

This function provides a sparse matrix whose all diagonal elements are not omitted even if they have zero values.

Usage

```
diag.padding.zero(A)
```

Arguments

A	A sparse matrix

Value

A sparse matrix whose all diagonal elements are not omitted even if they have zero values.

erhmm *ER-HMM (HMM with Erlang outputs)*

Description

A function to generate an object of `erhmm`.

Usage

```
erhmm(shape, alpha, rate, P, class = "CsparseMatrix")
```

Arguments

`shape`	An integer vector of shape parameters of Erlang outputs.
`alpha`	A vector for initial probabilities of HMM states.
`rate`	A vector of rate parameters of Erlang outputs.
`P`	An object of Matrix class for a transition probability matrix of HMM.
`class`	Name of Matrix class for `P`.

Details

ER-HMM has parameters α, $shape$, $rate$ and P. HMM state chages according to a discrete-time Markov chain with transition matrix P. At each HMM state, there is an inherent Erlang distriution as an output. This model can be converted to a MAP.

Value

`erhmm` gives an object of ER-HMM.

Note

`erhmm` requires shape parameters. Other parameters have default values.

See Also

`map`, `gmmpp`, `map.mmoment`, `map.jmoment`, `map.acf`

Examples

```
## create an ER-HMM consisting of two Erlang components with
## shape parameters 2 and 3.
erhmm(c(2,3))

## create an ER-HMM consisting of two Erlang components with
## shape parameters 2 and 3.
erhmm(shape=c(2,3))
```

```
## create an ER-HMM with specific parameters
(param <- erhmm(shape=c(2,3), alpha=c(0.3,0.7),
                rate=c(1.0,10.0),
                P=rbind(c(0.3, 0.7), c(0.1, 0.9))))

## convert to a general MAP
as(param, "map")

## marginal moments of MAP
map.mmoment(k=3, map=as(param, "map"))

## joint moments of MAP
map.jmoment(lag=1, map=as(param, "map"))

## k-lag correlation
map.acf(map=as(param, "map"))
```

erhmm-class *Class of ER-HMM*

Description

Parameters for an ER-HMM (Hidden Markov Model with Erlang outputs).

Slots

`size` The number of HMM states.

`alpha` A vector of initial probabilities for HMM states.

`shape` Shape parameters for Erlang distributions. The sum of shape parameters is the number of phases of MAP.

`rate` Rate parameters for Erlang distributions.

`P` An object of Matrix class for a transition probability matrix of HMM.

Note

Objects are usually created by an erhmm.

This class can be converted to `map`.

Methods:

ph.moment `signature(ph = "herlang")`: ...

emfit.init `signature(model = "herlang", data = "phdata.wtime")`: ...

emfit.init `signature(model = "herlang", data = "phdata.group")`: ...

emfit.estep `signature(model = "herlang", data = "phdata.wtime")`: ...

emfit.estep `signature(model = "herlang", data = "phdata.group")`: ...

emfit.mstep `signature(model = "herlang")`: ...

See Also

Classes map and gmmpp.

Examples

```
## create an ER-HMM consisting of two Erlang components with
## shape parameters 2 and 3.
erhmm(c(2,3))

## create an ER-HMM consisting of two Erlang components with
## shape parameters 2 and 3.
erhmm(shape=c(2,3))

## create an ER-HMM with specific parameters
(param <- erhmm(shape=c(2,3), alpha=c(0.3,0.7),
                rate=c(1.0,10.0),
                P=rbind(c(0.3, 0.7), c(0.1, 0.9))))

## convert to a general MAP
as(param, "map")

## marginal moments of MAP
map.mmoment(k=3, map=as(param, "map"))

## joint moments of MAP
map.jmoment(lag=1, map=as(param, "map"))

## k-lag correlation
map.acf(map=as(param, "map"))
```

---

gmmpp *Markovian Arrival Process (MAP)*

Description

Functions to generate an object of map.

Usage

```
gmmpp(size, alpha, D0, D1, class = "dgeMatrix")

map(size, alpha, D0, D1, class = "CsparseMatrix")

mmpp(size, class = "CsparseMatrix")
```

Arguments

`size`	An integer for the number of phases.
`alpha`	A vector of probabilities for determing an initial phase.
`D0`	An object of Matrix class for the initesmal generator without arrivals.
`D1`	An object of Matrix class for the initesmal generator with arrivals.
`class`	Name of Matrix class for `D0` and `D1`.

Details

MAP parameters are $alpha$, D_0 and D_1. $alpha$ is the probability vector to determine an initial phase at time 0. D_0 is an infinitesimal generator of underlyinc continuous-time Markov chain (CTMC) without arrival. D_1 is an infinitesimal generator of CTMC with arrival. The infinitesimal generator of underlying CTMC becomes $D_0 + D_1$. In the stationary case, α is often given by a stationary vector satisfying $\alpha(D_0 + D_1) = \alpha$.

`mmpp` generates an object of a specific MAP called MMPP. MMPP (Markov modulated Poisson process) is an MAP whose D_1 is given by a diagonal matrix. Unlike to general MAPs, MMPP never changes the phase at which an arrival occurs.

`gmmpp` generates an object of `gmmpp`, which is exactly same as MMPP. In the estimation algorithm, `gmmpp` class uses an approximate method.

Value

`map` gives an object of general MAP. `mmpp` gives an object of MMPP with default parameters. `gmmpp` gives an object of MMPP which uses an approximate estimation algorithm.

Note

`map` and `gmmpp` require either `size` or (`alpha`, `D0`, `D1`).

See Also

`erhmm`, `map.mmoment`, `map.jmoment`, `map.acf`

Examples

```
## create an MAP (full matrix) with 5 phases
map(5)

## create an MAP (full matrix) with 5 phases
map(size=5)

## create an MMPP with 5 states
mmpp(5)

## create an MMPP with 5 states for approximate
## estimation
gmmpp(5)
```

```
## create an MAP with specific parameters
(param <- map(alpha=c(1,0,0),
              D0=rbind(c(-4,2,0),c(2,-5,1),c(1,0,-4)),
              D1=rbind(c(1,1,0),c(1,0,1),c(2,0,1))))

## marginal moments of MAP
map.mmoment(k=3, map=param)

## joint moments of MAP
map.jmoment(lag=1, map=param)

## k-lag correlation
map.acf(map=param)
```

---

herlang *Hyper-Erlang Distribution*

Description

Density function, distribution function and random generation for the hyper-Erlang distribution, and a function to generate an object of herlang.

Usage

```
herlang(
  shape,
  mixrate = rep(1/length(shape), length(shape)),
  rate = rep(1, length(shape))
)

dherlang(x, herlang = herlang(shape = c(1)), log = FALSE)

pherlang(q, herlang = herlang(shape = c(1)), lower.tail = TRUE, log.p = FALSE)

rherlang(n, herlang = herlang(shape = c(1)))
```

Arguments

shape	An integer vector of shape parameters of Erlang components.
mixrate	A vector for the initial probabilities of hyper-Erlang distribution.
rate	A vector of rate parameters of Erlang components.
x	Vectors of quantiles.
herlang	An object of S4 class of hyper Erlang (herlang).
log	Logical; if TRUE, the log density is returned.
q	Vectors of quantiles.

`lower.tail`	Logical; if `TRUE`, probabilities are `P[X <= x]`, otherwise, `P[X > x]`.
`log.p`	Logical; if `TRUE`, the log probability is returned.
`n`	Number of observations.
`p`	A vector of probabilities.

Details

The hyper-Erlang distribution with parameters m_i (`mixrate`), s_i (`shape`) and r_i (`rate`): Cumulative probability function;

$$F(q) = \sum_i \int_0^q m_i \frac{r_i^{s_i} x^{s_i-1} e^{-r_i x}}{(s_i-1)!} dx$$

Probability density function;

$$f(x) = \sum_i m_i \frac{r_i^{s_i} x^{s_i-1} e^{-r_i x}}{(s_i-1)!}$$

Value

`herlang` gives an object of hyper-Erlang distribution. `dherlang` gives the density function, `pherlang` gives the distribution function, and `rherlang` generates random samples.

Note

`herlang` requires shape parameters.

See Also

`ph`, `herlang`

Examples

```
## create a hyper Erlang consisting of two Erlang
## with shape parameters 2 and 3.
(param1 <- herlang(c(2,3)))

## create a hyper Erlang consisting of two Erlang
## with shape parameters 2 and 3.
(param1 <- herlang(shape=c(2,3)))

## create a hyper Erlang with specific parameters
(param2 <- herlang(shape=c(2,3), mixrate=c(0.3,0.7), rate=c(1.0,10.0)))

## convert to a general PH
as(param2, "ph")

## p.d.f. for 0, 0.1, ..., 1
(dherlang(x=seq(0, 1, 0.1), herlang=param2))

## c.d.f. for 0, 0.1, ..., 1
(pherlang(q=seq(0, 1, 0.1), herlang=param2))
```

```
## generate 10 samples
(rherlang(n=10, herlang=param2))
```

---

herlang-class *Class of hyper Erlang*

Description

Parameters for a hyper Erlang.

Slots

`size` The number of components (hyper Erlang components).

`mixrate` A vector of mixed rates (probability for selecting a component).

`shape` Shape parameters for Erlang distributions.

`rate` Rate parameters for Erlang distributions.

Note

Objects are usually created by a herlang. This class can be converted to ph.

Methods:

ph.moment `signature(ph = "herlang")`: ...

emfit.init `signature(model = "herlang", data = "phdata.wtime")`: ...

emfit.init `signature(model = "herlang", data = "phdata.group")`: ...

emfit.estep `signature(model = "herlang", data = "phdata.wtime")`: ...

emfit.estep `signature(model = "herlang", data = "phdata.group")`: ...

emfit.mstep `signature(model = "herlang")`: ...

See Also

Classes ph and cf1.

Examples

```
## create a hyper Erlang consisting of two Erlang
## with shape parameters 2 and 3.
(param1 <- herlang(c(2,3)))

## create a hyper Erlang consisting of two Erlang
## with shape parameters 2 and 3.
(param1 <- herlang(shape=c(2,3)))

## create a hyper Erlang with specific parameters
```

```
(param2 <- herlang(shape=c(2,3), mixrate=c(0.3,0.7), rate=c(1.0,10.0)))

## convert to a general PH
as(param2, "ph")

## p.d.f. for 0, 0.1, ..., 1
(dherlang(x=seq(0, 1, 0.1), herlang=param2))

## c.d.f. for 0, 0.1, ..., 1
(pherlang(q=seq(0, 1, 0.1), herlang=param2))

## generate 10 samples
(rherlang(n=10, herlang=param2))
```

`map-class` *Classes of MAP*

Description

Parameters for MAP and MMPP.

Slots

`size` The number of phases (internal states).

`alpha` A probability (row) vector to decide an initial phase.

`D0` A square matrix that means transition rates without arrivals.

`D1` A square matrix that means transition rates with arrivals. In the case of MMPP, D1 should be a diagonal matrix.

`df` The number of free parameters.

Note

Objects are usually created by map, mmpp or gmmpp.

See Also

Classes `erhmm`.

Examples

```
## create an MAP (full matrix) with 5 phases
map(5)

## create an MAP (full matrix) with 5 phases
map(size=5)

## create an MMPP with 5 states
```

```
mmpp(5)

## create an MMPP with 5 states for approximate
## estimation
gmmpp(5)

## create an MAP with specific parameters
(param <- map(alpha=c(1,0,0),
              D0=rbind(c(-4,2,0),c(2,-5,1),c(1,0,-4)),
              D1=rbind(c(1,1,0),c(1,0,1),c(2,0,1))))

## marginal moments of MAP
map.mmoment(k=3, map=param)

## joint moments of MAP
map.jmoment(lag=1, map=param)

## k-lag correlation
map.acf(map=param)
```

---

map.mmoment *Moments for Markovian arrival pcess (MAP)*

Description

Moments for MAP.

Usage

```
map.mmoment(k, map)

map.jmoment(lag, map)

map.acf(map)
```

Arguments

k	An integer of dgrees of moments.
map	An object of S4 class of MAP (map, gmmpp).
lag	An integer of time lag for corrleation.

Details

MAP parameters are α, D_0 and D_1;

$$P = (-D_0)^{-1} D_1$$

and

$$sP = s.$$

Then the moments for MAP are marginal moment;

$$m_k = k!s(-D_0)^{-k}1,$$

joint moment;

$$s_{ij}(lag) = i!j!s(-D_0)^{-i}P^{lag}(-D_0)^{-j}1,$$

k-lag correlation (autocorrelation);

$$rho(lag) = (s_{11}(lag) - m_1^2)/(m_2 - m_1^2)$$

Value

`map.mmoment` gives a vector of up to k moments. `map.jmoment` gives a matrix of $s_{ij}(lag), i = 1, .., n, j = 1, .., n$ where n is the size of phases. `map.acf` gives a vector of up to n-lag correlation, where n is the size of phases.

Note

`map.mmoment` is a generic function for `ph` and `herlang`.

See Also

`map`, `gmmpp`, `erhmm`

Examples

```
## create an MAP with specific parameters
(param1 <- map(alpha=c(1,0,0),
               D0=rbind(c(-4,2,0),c(2,-5,1),c(1,0,-4)),
               D1=rbind(c(1,1,0),c(1,0,1),c(2,0,1))))

## create an ER-HMM with specific parameters
(param2 <- erhmm(shape=c(2,3), alpha=c(0.3,0.7),
                 rate=c(1.0,10.0),
                 P=rbind(c(0.3, 0.7), c(0.1, 0.9))))

## marginal moments of MAP
map.mmoment(k=3, map=param1)
map.mmoment(k=3, map=as(param2, "map"))

## joint moments of MAP
map.jmoment(lag=1, map=param1)
map.jmoment(lag=1, map=as(param2, "map"))

## k-lag correlation
map.acf(map=param1)
map.acf(map=as(param2, "map"))
```

`mapfit.group` *MAP fitting with grouped data*

Description

Estimates MAP parameters from grouped data.

Usage

```
mapfit.group(
  map,
  counts,
  breaks,
  intervals,
  instant,
  stationary = TRUE,
  control = list(),
  verbose = list(),
  ...
)
```

Arguments

`map`	S4 class for MAP. The estimation algorithm is selected depending on thie class.
`counts`	A vector for the number of arrivals in time interval.
`breaks`	A vector for time sequence to determine time interval. This is equivalent to `c(0,cumsum(intervals))`. If this is missing, it is assigned to `0:length(counts)`.
`intervals`	A vector for a sequence of time length for intervals. This is equivalent to `diff(breaks))`. If this is missing, it is assigned to `rep(1,length(counts))`.
`instant`	A vector of integer to indicate whether an arrival occurs at the last time of interval. If instant is 1, an arrival occurs at the last time of interval. If instant is 0, no arrival occurs at the last time of interval. By using instant, time point data can be expressed by grouped data class. If instant is missing, it is given by `rep(0,length(counts))`, i.e., there are no arrivals at the end of interval.
`stationary`	A logical value that determine whether initial probability is given by a stationary vector of underlying Markov process or not.
`control`	A list of parameters for controlling the fitting process.
`verbose`	A list of parameters for displaying the fitting process.
`...`	Further arguments for methods.

Value

Returns a list with components, which is an object of S3 class `mapfit.result`;

`model`	an object for estimated MAP class (`map`, `erhmm`).

`llf`	a value of the maximum log-likelihood.
`df`	a value of degrees of freedom of the model.
`aic`	a value of Akaike information criterion.
`iter`	the number of iterations.
`convergence`	a logical value for the convergence of estimation algorithm.
`ctime`	computation time (user time).
`stationary`	a logical value for the argument `stationary`.
`data`	an object for MAP data class
`aerror`	a value of absolute error for llf at the last step of algorithm.
`rerror`	a value of relative error for llf at the last step of algorithm.
`control`	a list of the argument of `control`.
`verbose`	a list of the argument of `verbose`.
`call`	the matched call.

See Also

`mapfit.point`, `map` and `gmmpp`

Examples

```
## load trace data
data(BCpAug89)
BCpAug89s <- head(BCpAug89, 50)

## make grouped data
BCpAug89.group <- hist(cumsum(BCpAug89s),
                        breaks=seq(0, 0.15, 0.005),
                        plot=FALSE)

## MAP fitting for general MAP
(result1 <- mapfit.group(map=map(2),
                        counts=BCpAug89.group$counts,
                        breaks=BCpAug89.group$breaks))
## MAP fitting for MMPP
(result2 <- mapfit.group(map=mmpp(2),
                         counts=BCpAug89.group$counts,
                         breaks=BCpAug89.group$breaks))

## MAP fitting with approximate MMPP
(result3 <- mapfit.group(map=gmmpp(2),
                         counts=BCpAug89.group$counts,
                         breaks=BCpAug89.group$breaks))

## marginal moments for estimated MAP
map.mmoment(k=3, map=result1$model)
map.mmoment(k=3, map=result2$model)
```

```
map.mmoment(k=3, map=result3$model)

## joint moments for estimated MAP
map.jmoment(lag=1, map=result1$model)
map.jmoment(lag=1, map=result2$model)
map.jmoment(lag=1, map=result3$model)

## lag-k correlation
map.acf(map=result1$model)
map.acf(map=result2$model)
map.acf(map=result3$model)
```

---

mapfit.point *MAP fitting with time point data*

Description

Estimates MAP parameters from time point data.

Usage

```
mapfit.point(
  map,
  x,
  intervals,
  stationary = TRUE,
  method = c("all", "increment"),
  lbound = 1,
  ubound = NULL,
  control = list(),
  verbose = list(),
  ...
)
```

Arguments

map	An object of S4 class for MAP. The estimation algorithm is selected depending on thie class.
x	A vector for time sequence of arrivals. This is equivalent to `cumsum(intervals)`. Either time or difftime should be given.
intervals	A vector for the data for intrarrival time. This is equivalent to `diff(c(0,x))`. Either time or difftime should be given.
stationary	A logical value that determine whether initial probability is given by a stationary vector of underlying Markov process or not.
method	The name of estimation method for ER-HMM (erhmm).
lbound	A value for lower limit for the number of states in ER-HMM (erhmm).

ubound	A value for upper limit for the number of states in ER-HMM (erhmm).
control	A list of parameters for controlling the fitting process.
verbose	A list of parameters for displaying the fitting process.
...	Further arguments for methods.

Value

Returns a list with components, which is an object of S3 class `mapfit.result`;

model	an object for estimated MAP class (map, erhmm).
llf	a value of the maximum log-likelihood.
df	a value of degrees of freedom of the model.
aic	a value of Akaike information criterion.
iter	the number of iterations.
convergence	a logical value for the convergence of estimation algorithm.
ctime	computation time (user time).
stationary	a logical value for the argument `stationary`.
data	an object for MAP data class
aerror	a value of absolute error for llf at the last step of algorithm.
rerror	a value of relative error for llf at the last step of algorithm.
control	a list of the argument of `control`.
verbose	a list of the argument of `verbose`.
call	the matched call.

See Also

mapfit.group, map and erhmm

Examples

```
## load trace data
data(BCpAug89)
BCpAug89s <- head(BCpAug89, 50)

## MAP fitting for general MAP
(result1 <- mapfit.point(map=map(2), x=cumsum(BCpAug89s)))

## MAP fitting for MMPP
(result2 <- mapfit.point(map=mmpp(2), x=cumsum(BCpAug89s)))

## MAP fitting for ER-HMM
(result3 <- mapfit.point(map=erhmm(3), x=cumsum(BCpAug89s)))

## marginal moments for estimated MAP
map.mmoment(k=3, map=result1$model)
map.mmoment(k=3, map=result2$model)
```

```
map.mmoment(k=3, map=as(result3$model, "map"))

## joint moments for estimated MAP
map.jmoment(lag=1, map=result1$model)
map.jmoment(lag=1, map=result2$model)
map.jmoment(lag=1, map=as(result3$model, "map"))

## lag-k correlation
map.acf(map=result1$model)
map.acf(map=result2$model)
map.acf(map=as(result3$model, "map"))
```

---

ph *Phase-Type (PH) Distribution*

Description

Density function, distribution function and random generation for the PH distribution, and a function to generate an object of ph.

Usage

```
ph(size, alpha, Q, xi, class = "CsparseMatrix")

dph(x, ph = ph(1), log = FALSE)

pph(q, ph = ph(1), lower.tail = TRUE, log.p = FALSE)

rph(n, ph = ph(1))
```

Arguments

size	A value for the number of phases.
alpha	A vector for the initial probabilities of PH distribution.
Q	An object of Matrix class for the initesmal generator of PH distribution.
xi	A vector for the exit rates of PH distribution.
class	Name of Matrix class for Q.
x	Vectors of quantiles.
ph	An object of S4 class of PH (ph).
log	Logical; if TRUE, the log density is returned.
q	Vectors of quantiles.
lower.tail	Logical; if TRUE, probabilities are P[X <= x], otherwise, P[X > x].
log.p	Logical; if TRUE, the log probability is returned.
n	Number of observations.
p	A vector of probabilities.

Details

The PH distribution with parameters α, Q and ξ: Cumulative probability function;

$$F(q) = 1 - \alpha \exp(Qq)1$$

Probability density function;

$$f(x) = \alpha \exp(Qx)\xi$$

Value

ph gives an object of general PH distribution. dph gives the density function, pph gives the distribution function, and rph generates random samples.

Note

ph requires either size or (alpha, Q, xi). rph for ph is too slow. It is recommended to use rph for cf1.

See Also

cf1, herlang

Examples

```
## create a PH (full matrix) with 5 phases
(param1 <- ph(5))

## create a PH (full matrix) with 5 phases
(param1 <- ph(size=5))

## create a PH with specific parameters
(param2 <- ph(alpha=c(1,0,0),
              Q=rbind(c(-4,2,0),c(2,-5,1),c(1,0,-1)),
              xi=c(2,2,0)))

## p.d.f. for 0, 0.1, ..., 1
(dph(x=seq(0, 1, 0.1), ph=param2))

## c.d.f. for 0, 0.1, ..., 1
(pph(q=seq(0, 1, 0.1), ph=param2))

## generate 10 samples
(rph(n=10, ph=param2))
```

ph-class *Class of general PH distributions*

Description

Parameters for a general PH distribution.

Slots

`size` The number of phases (transient states).

`alpha` A probability (row) vector to decide an initial phase.

`Q` A square matrix that means transition rates between phases.

`xi` A column vector for exiting rates from phases to an absorbing state.

`df` The number of free parameters.

Note

Objects are usually created by a ph.

The methods of this class:

ph.moment `signature(ph = "ph")`: ...

emfit.init `signature(model = "ph")`: ...

emfit.estep `signature(model = "ph", data = "phdata.wtime")`: ...

emfit.estep `signature(model = "ph", data = "phdata.group")`: ...

emfit.mstep `signature(model = "ph")`: ...

See Also

Classes `cf1` and `herlang`.

Examples

```
## create a PH (full matrix) with 5 phases
(param1 <- ph(5))

## create a PH (full matrix) with 5 phases
(param1 <- ph(size=5))

## create a PH with specific parameters
(param2 <- ph(alpha=c(1,0,0),
              Q=rbind(c(-4,2,0),c(2,-5,1),c(1,0,-1)),
              xi=c(2,2,0)))

## p.d.f. for 0, 0.1, ..., 1
(dph(x=seq(0, 1, 0.1), ph=param2))
```

```
## c.d.f. for 0, 0.1, ..., 1
(pph(q=seq(0, 1, 0.1), ph=param2))

## generate 10 samples
(rph(n=10, ph=param2))
```

ph.moment *Moments for Phase-Type (PH) Distribution*

Description

Moments for PH distribution.

Usage

```
## S4 method for signature 'ANY,ph'
ph.moment(k, ph, ...)

ph.mean(ph)

ph.var(ph)
```

Arguments

k	An integer of dgrees of moments.
ph	An object of S4 class of PH (ph) or Hyper-Erlang (herlang).
...	Further arguments for methods.

Details

The PH distribution with parameters $alpha$, Q and xi: k-th moment;

$$k!\alpha(-Q)^{-k}1$$

Value

ph.mean and ph.var give mean and variance of PH. ph.moment gives a vector of up to k moments.

Note

ph.moment is a generic function for ph and herlang.

See Also

ph, cf1, herlang

Examples

```
## create a PH with specific parameters
(param1 <- ph(alpha=c(1,0,0),
              Q=rbind(c(-4,2,0),c(2,-5,1),c(1,0,-1)),
              xi=c(2,2,0)))

## create a CF1 with specific parameters
(param2 <- cf1(alpha=c(1,0,0), rate=c(1.0,2.0,3.0)))

## create a hyper Erlang with specific parameters
(param3 <- herlang(shape=c(2,3), mixrate=c(0.3,0.7), rate=c(1.0,10.0)))

## mean
ph.mean(param1)
ph.mean(param2)
ph.mean(param3)

## variance
ph.var(param1)
ph.var(param2)
ph.var(param3)

## up to 5 moments
ph.moment(5, param1)
ph.moment(5, param2)
ph.moment(5, param3)
```

---

phfit.3mom *PH fitting with three moments*

Description

Estimates PH parameters from three moments.

Usage

```
phfit.3mom(
  m1,
  m2,
  m3,
  method = c("Osogami06", "Bobbio05"),
  max.phase = 50,
  epsilon = sqrt(.Machine$double.eps)
)
```

Arguments

m1	A value of the first moment.
m2	A value of the second moment.
m3	A value of the third moment.
method	The name of moment matching method.
max.phase	An integer for the maximum number of phases in the method "Osogami06".
epsilon	A value of precision in the method "Osogami06".

Value

An object of S4 class of general PH ph.

Note

The method "Osogami06" checks the first three moements on whether there exists a PH whose three moements match to them. In such case, the method "Bobbio05" often returns an error.

References

Osogami, T. and Harchol-Balter, M. (2006) Closed Form Solutions for Mapping General Distributions to Minimal PH Distributions. *Performance Evaluation*, **63**(6), 524–552.

Bobbio, A., Horvath, A. and Telek, M. (2005) Matching Three Moments with Minimal Acyclic Phase Type Distributions. *Stochastic Models*, **21**(2-3), 303–326.

See Also

ph, ph.moment

Examples

```
## Three moment matching
## Moments of Weibull(shpape=2, scale=1); (0.886227, 1.0, 1.32934)
(result1 <- phfit.3mom(0.886227, 1.0, 1.32934))

## Three moment matching
## Moments of Weibull(shpape=2, scale=1); (0.886227, 1.0, 1.32934)
(result2 <- phfit.3mom(0.886227, 1.0, 1.32934, method="Bobbio05"))

## mean
ph.mean(result1)
ph.mean(result2)

## variance
ph.var(result1)
ph.var(result2)

## up to 5 moments
ph.moment(5, result1)
ph.moment(5, result2)
```

phfit.density *PH fitting with density function*

Description

Estimates PH parameters from density function.

Usage

```
phfit.density(
  ph,
  f,
  method = c("all", "increment"),
  lbound = 1,
  ubound = NULL,
  deformula = deformula.zeroinf,
  weight.zero = 1e-12,
  weight.reltol = 1e-08,
  start.divisions = 8,
  max.iter = 12,
  control = list(),
  verbose = list(),
  ...
)
```

Arguments

ph	An object of S4 class for MAP. The estimation algorithm is selected depending on thie class.
f	A faunction object for a density function.
method	The name of estimation method for hyper Erlang (herlang).
lbound	A value for lower limit for the number of states in hyper Erlang (herlang).
ubound	A value for upper limit for the number of states in hyper Erlang (herlang).
deformula	An object for formulas of numerical integration. It is not necessary to change it when the density function is defined on the positive domain [0,infinity).
weight.zero	A absolute value which is regarded as zero in numerical integration.
weight.reltol	A value for precision of numerical integration.
start.divisions	A value for starting value of divitions in deformula.
max.iter	A value for the maximum number of iterations to increase divisions in deformula.
control	A list of parameters for controlling the fitting process.
verbose	A list of parameters for displaying the fitting process.
...	Further arguments for methods, which are also used to send the arguments to density function.

Value

Returns a list with components, which is an object of S3 class `phfit.result`;

`model`	an object for estimated PH class (`ph`, `cf1`, `herlang`).
`llf`	a value of the maximum log-likelihood (a netative value of the cross entropy).
`df`	a value of degrees of freedom of the model.
`aic`	a value of Akaike information criterion (this is not meaningless in this case).
`KL`	a value of Kullback-Leibler divergence.
`iter`	the number of iterations.
`convergence`	a logical value for the convergence of estimation algorithm.
`ctime`	computation time (user time).
`data`	an object for MAP data class
`aerror`	a value of absolute error for llf at the last step of algorithm.
`rerror`	a value of relative error for llf at the last step of algorithm.
`control`	a list of the argument of `control`.
`verbose`	a list of the argument of `verbose`.
`call`	the matched call.

Note

Any of density function can be applied to the argument `f`, where `f` should be defined `f <- function(x, ...)`. The first argument of `f` should be an integral parameter. The other parameters are set in the argument `...` of `phfit.density`. The truncated density function can also be used directly.

See Also

`phfit.point`, `phfit.group`, `ph`, `cf1` and `herlang`

Examples

```
####################
##### truncated density
####################

## PH fitting for general PH
(result1 <- phfit.density(ph=ph(2), f=dnorm, mean=3, sd=1))

## PH fitting for CF1
(result2 <- phfit.density(ph=cf1(2), f=dnorm, mean=3, sd=1))

## PH fitting for hyper Erlang
(result3 <- phfit.density(ph=herlang(3), f=dnorm, mean=3, sd=1))

## mean
ph.mean(result1$model)
ph.mean(result2$model)
```

```
ph.mean(result3$model)

## variance
ph.var(result1$model)
ph.var(result2$model)
ph.var(result3$model)

## up to 5 moments
ph.moment(5, result1$model)
ph.moment(5, result2$model)
ph.moment(5, result3$model)
```

---

phfit.group *PH fitting with grouped data*

Description

Estimates PH parameters from grouped data.

Usage

```
phfit.group(
  ph,
  counts,
  breaks,
  intervals,
  instant,
  method = c("all", "increment"),
  lbound = 1,
  ubound = NULL,
  control = list(),
  verbose = list(),
  ...
)
```

Arguments

ph	An object of S4 class for MAP. The estimation algorithm is selected depending on thie class.
counts	A vector of the number of points in intervals.
breaks	A vector for a sequence of points of boundaries of intervals. This is equivalent to `c(0,cumsum(intervals))`. If this is missing, it is assigned to `0:length(counts)`.
intervals	A vector of time lengths for intervals. This is equivalent to `diff(breaks))`. If this is missing, it is assigned to `rep(1,length(counts))`.

`instant`	A vector of integers to indicate whether sample is drawn at the last of interval. If instant is 1, a sample is drawn at the last of interval. If instant is 0, no sample is drawn at the last of interval. By using instant, point data can be expressed by grouped data. If instant is missing, it is given by `rep(0L,length(counts))`, i.e., there are no sampels at the last of interval.
`method`	The name of estimation method for hyper Erlang (`herlang`).
`lbound`	A value for lower limit for the number of states in hyper Erlang (`herlang`).
`ubound`	A value for upper limit for the number of states in hyper Erlang (`herlang`).
`control`	A list of parameters for controlling the fitting process.
`verbose`	A list of parameters for displaying the fitting process.
`...`	Rurther arguments for methods.

Value

Returns a list with components, which is an object of S3 class `phfit.result`;

`model`	an object for estimated PH class (`ph`, `cf1`, `herlang`).
`llf`	a value of the maximum log-likelihood.
`df`	a value of degrees of freedom of the model.
`aic`	a value of Akaike information criterion.
`iter`	the number of iterations.
`convergence`	a logical value for the convergence of estimation algorithm.
`ctime`	computation time (user time).
`data`	an object for MAP data class
`aerror`	a value of absolute error for llf at the last step of algorithm.
`rerror`	a value of relative error for llf at the last step of algorithm.
`control`	a list of the argument of `control`.
`verbose`	a list of the argument of `verbose`.
`call`	the matched call.

Note

In this method, we can handle truncated data using `NA` and `Inf`; `phfit.group(ph=cf1(5), counts=c(countsdata, NA), breaks=c(breakdata, +Inf))` `NA` means missing of count data at the conrredponding interval, and `Inf` ia allowed to put the last of breaks or intervals which represents a special interval [the last break point,infinity).

See Also

`phfit.point`, `phfit.density`, `ph`, `cf1` and `herlang`

Examples

```
## make sample
wsample <- rweibull(n=100, shape=2, scale=1)
wgroup <- hist(x=wsample, breaks="fd", plot=FALSE)

## PH fitting for general PH
(result1 <- phfit.group(ph=ph(2), counts=wgroup$counts, breaks=wgroup$breaks))

## PH fitting for CF1
(result2 <- phfit.group(ph=cf1(2), counts=wgroup$counts, breaks=wgroup$breaks))

## PH fitting for hyper Erlang
(result3 <- phfit.group(ph=herlang(3), counts=wgroup$counts, breaks=wgroup$breaks))

## mean
ph.mean(result1$model)
ph.mean(result2$model)
ph.mean(result3$model)

## variance
ph.var(result1$model)
ph.var(result2$model)
ph.var(result3$model)

## up to 5 moments
ph.moment(5, result1$model)
ph.moment(5, result2$model)
ph.moment(5, result3$model)
```

`phfit.point` *PH fitting with point data*

Description

Estimates PH parameters from point data.

Usage

```
phfit.point(
  ph,
  x,
  weights,
  method = c("all", "increment"),
  lbound = 1,
  ubound = NULL,
  control = list(),
  verbose = list(),
  ...
)
```

Arguments

ph	An object of S4 class for MAP. The estimation algorithm is selected depending on thie class.
x	A vector for point data.
weights	A vector of weights for points.
method	The name of estimation method for hyper Erlang (herlang).
lbound	A value for lower limit for the number of states in hyper Erlang (herlang).
ubound	A value for upper limit for the number of states in hyper Erlang (herlang).
control	A list of parameters for controlling the fitting process.
verbose	A list of parameters for displaying the fitting process.
...	Further arguments for methods.

Value

Returns a list with components, which is an object of S3 class `phfit.result`;

model	an object for estimated PH class (ph, cf1, herlang).
llf	a value of the maximum log-likelihood.
df	a value of degrees of freedom of the model.
aic	a value of Akaike information criterion.
iter	the number of iterations.
convergence	a logical value for the convergence of estimation algorithm.
ctime	computation time (user time).
data	an object for MAP data class
aerror	a value of absolute error for llf at the last step of algorithm.
rerror	a value of relative error for llf at the last step of algorithm.
control	a list of the argument of `control`.
verbose	a list of the argument of `verbose`.
call	the matched call.

See Also

phfit.group, phfit.density, ph, cf1 and herlang

Examples

```
## make sample
wsample <- rweibull(n=100, shape=2, scale=1)

## PH fitting for general PH
(result1 <- phfit.point(ph=ph(2), x=wsample))

## PH fitting for CF1
```

```
(result2 <- phfit.point(ph=cf1(2), x=wsample))

## PH fitting for hyper Erlang
(result3 <- phfit.point(ph=herlang(3), x=wsample))

## mean
ph.mean(result1$model)
ph.mean(result2$model)
ph.mean(result3$model)

## variance
ph.var(result1$model)
ph.var(result2$model)
ph.var(result3$model)

## up to 5 moments
ph.moment(5, result1$model)
ph.moment(5, result2$model)
ph.moment(5, result3$model)
```

Index