

Package: matrixdist (via r-universe)

October 6, 2026

Type Package

Title Statistics for Matrix Distributions

Version 1.1.9

Date 2023-08-01

Maintainer Martin Bladt <martinbladt@gmail.com>

Author Martin Bladt [aut, cre], Jorge Yslas [aut], Alaric Müller [ctb]

URL https://github.com/martinbladt/matrixdist_1.0

BugReports https://github.com/martinbladt/matrixdist_1.0/issues

Description Tools for phase-type distributions including the following variants: continuous, discrete, multivariate, in-homogeneous, right-censored, and regression. Methods for functional evaluation, simulation and estimation using the expectation-maximization (EM) algorithm are provided for all models. The methods of this package are based on the following references. Asmussen, S., Nerman, O., & Olsson, M. (1996). Fitting phase-type distributions via the EM algorithm, Olsson, M. (1996). Estimation of phase-type distributions from censored data, Albrecher, H., & Bladt, M. (2019) <doi:10.1017/jpr.2019.60>, Albrecher, H., Bladt, M., & Yslas, J. (2022) <doi:10.1111/sjos.12505>, Albrecher, H., Bladt, M., Bladt, M., & Yslas, J. (2022) <doi:10.1016/j.insmatheco.2022.08.001>, Bladt, M., & Yslas, J. (2022) <doi:10.1080/03461238.2022.2097019>, Bladt, M. (2022) <doi:10.1017/asb.2021.40>, Bladt, M. (2023) <doi:10.1080/10920277.2023.2167833>, Albrecher, H., Bladt, M., & Mueller, A. (2023) <doi:10.1515/demo-2022-0153>, Bladt, M. & Yslas, J. (2023) <doi:10.1016/j.insmatheco.2023.02.008>.

Depends R (>= 4.2.0)

License GPL-3

Imports Rcpp, methods, nnet, reshape2

LinkingTo Rcpp, RcppArmadillo

Encoding UTF-8

Roxygen list(markdown = TRUE)
RoxygenNote 7.3.2
Config/pak/sysreqs libicu-dev
Repository https://martinbladt.r-universe.dev
Date/Publication 2026-03-06 01:20:18 UTC
RemoteUrl https://github.com/martinbladt/matrixdist_1.0
RemoteRef HEAD
RemoteSha 20ff43761bdb2667e3cb2ce1db0e9f342bdd8e15

Contents

matrixdist-package	9
+ ,dph,dph-method	10
+ ,ph,ph-method	11
a_rungekutta	11
bivdph	12
bivdph-class	12
bivdph_density	13
bivdph_tail	13
biviph	14
biviph-class	15
bivph	15
bivph-class	16
bivph_density	16
bivph_laplace	17
bivph_tail	17
cdf	18
cdf,dph-method	18
cdf,iph-method	19
cdf,miph-method	19
cdf,mph-method	20
cdf,ph-method	21
clone_matrix	21
clone_vector	22
coef,bivdph-method	22
coef,biviph-method	23
coef,bivph-method	23
coef,dph-method	24
coef,iph-method	24
coef,mdph-method	25
coef,ph-method	25
coef,sph-method	26
cor,bivdph-method	26
cor,bivph-method	27
cor,mdph-method	27

cor,mph-method	28
cor,MPHstar-method	29
csph_density	29
csph_density_par	30
csph_density_test	30
csph_densityrk	31
cumulate_matrix	31
cumulate_vector	32
data_aggregation	32
default_step_length	33
dens	33
dens,bivdph-method	34
dens,bivph-method	34
dens,bivph-method	35
dens,dph-method	35
dens,iph-method	36
dens,mdph-method	36
dens,miph-method	37
dens,mph-method	38
dens,ph-method	38
dph	39
dph-class	40
dph_pgf	40
dphcdf	41
dphdensity	41
EM_step_mPH_rc	42
embedded_mc	42
embedded_mc_cs	43
EMstep_bivdph	43
EMstep_bivdph_MoE	44
EMstep_bivph	44
EMstep_dph	45
EMstep_dph_MoE	45
EMstep_mdph	46
EMstep_mdph_MoE	46
EMstep_MoE_PADE	47
EMstep_PADE	47
EMstep_RK	48
EMstep_UNI	49
EMstep_UNI_intervalCensoring	49
evaluate	50
evaluate,sph-method	51
expm_terms	51
expmat	52
find_n	52
find_weight	53
Fisher	53
Fisher,sph-method	54

fit	54
fit,bivdph-method	55
fit,bivph-method	55
fit,dph-method	56
fit,mdph-method	57
fit,mph-method	58
fit,MPHstar-method	59
fit,ph-method	60
fit_multiple	62
haz	64
haz,ph-method	64
inf_norm	65
inhomogeneity_reg	65
inhomogeneity_reg,ph-method	66
initial_state	68
iph	68
iph-class	69
laplace	70
laplace,bivph-method	70
laplace,mph-method	71
laplace,ph-method	71
linCom	72
linCom,bivph-method	72
linCom,MPHstar-method	73
linear_combination	73
logLik,ph-method	74
logLikelihood_UNIs_PI	75
logLikelihoodbivDPH	76
logLikelihoodbivDPH_MoE	76
logLikelihoodbivPH	77
logLikelihoodDPH	77
logLikelihoodDPH_MoE	78
logLikelihoodmDPH	78
logLikelihoodmDPH_MoE	79
logLikelihoodMgev_PADE	79
logLikelihoodMgev_RK	80
logLikelihoodMgev_UNI	80
logLikelihoodMgev_UNI_inhom	81
logLikelihoodMgev_UNI_inhom_intCens	81
logLikelihoodMgev_UNI_intervalCensoring	82
logLikelihoodMgompertz_PADE	83
logLikelihoodMgompertz_PADEs	83
logLikelihoodMgompertz_RK	84
logLikelihoodMgompertz_RKs	85
logLikelihoodMgompertz_UNI	86
logLikelihoodMgompertz_UNI_inhom	86
logLikelihoodMgompertz_UNI_inhom_intCens	87
logLikelihoodMgompertz_UNI_intervalCensoring	88

logLikelihoodMgompertz_UNIs	88
logLikelihoodMgompertz_UNIs_inhom	89
logLikelihoodMgompertz_UNIs_inhom_intCens	90
logLikelihoodMgompertz_UNIs_intervalCensoring	91
logLikelihoodMloglogistic_PADE	92
logLikelihoodMloglogistic_PADEs	92
logLikelihoodMloglogistic_RK	93
logLikelihoodMloglogistic_RKs	94
logLikelihoodMloglogistic_UNI	95
logLikelihoodMloglogistic_UNI_inhom	95
logLikelihoodMloglogistic_UNI_inhom_intCens	96
logLikelihoodMloglogistic_UNI_intervalCensoring	97
logLikelihoodMloglogistic_UNIs	97
logLikelihoodMloglogistic_UNIs_inhom	98
logLikelihoodMloglogistic_UNIs_inhom_intCens	99
logLikelihoodMloglogistic_UNIs_intervalCensoring	100
logLikelihoodMlognormal_PADE	101
logLikelihoodMlognormal_PADEs	101
logLikelihoodMlognormal_RK	102
logLikelihoodMlognormal_RKs	103
logLikelihoodMlognormal_UNI	104
logLikelihoodMlognormal_UNI_inhom	104
logLikelihoodMlognormal_UNI_inhom_intCens	105
logLikelihoodMlognormal_UNI_intervalCensoring	106
logLikelihoodMlognormal_UNIs	106
logLikelihoodMlognormal_UNIs_inhom	107
logLikelihoodMlognormal_UNIs_inhom_intCens	108
logLikelihoodMlognormal_UNIs_intervalCensoring	109
logLikelihoodMpareto_PADE	110
logLikelihoodMpareto_PADEs	110
logLikelihoodMpareto_RK	111
logLikelihoodMpareto_RKs	112
logLikelihoodMpareto_UNI	113
logLikelihoodMpareto_UNI_inhom	113
logLikelihoodMpareto_UNI_inhom_intCens	114
logLikelihoodMpareto_UNI_intervalCensoring	115
logLikelihoodMpareto_UNIs	115
logLikelihoodMpareto_UNIs_inhom	116
logLikelihoodMpareto_UNIs_inhom_intCens	117
logLikelihoodMpareto_UNIs_intervalCensoring	118
logLikelihoodMweibull_PADE	119
logLikelihoodMweibull_PADEs	119
logLikelihoodMweibull_RK	120
logLikelihoodMweibull_RKs	121
logLikelihoodMweibull_UNI	122
logLikelihoodMweibull_UNI_inhom	122
logLikelihoodMweibull_UNI_inhom_intCens	123
logLikelihoodMweibull_UNI_intervalCensoring	124

logLikelihoodMweibull_UNIs	124
logLikelihoodMweibull_UNIs_inhom	125
logLikelihoodMweibull_UNIs_inhom_intCens	126
logLikelihoodMweibull_UNIs_intervalCensoring	127
logLikelihoodPH_MoE	128
logLikelihoodPH_PADE	128
logLikelihoodPH_PADEs	129
logLikelihoodPH_RK	130
logLikelihoodPH_RKs	130
logLikelihoodPH_UNI	131
logLikelihoodPH_UNI_intervalCensoring	131
logLikelihoodPH_UNIs	132
logLikelihoodPH_UNIs_intervalCensoring	133
LRT	133
LRT,ph,ph-method	134
m_exp_sum	134
marginal	135
marginal,bivdph-method	135
marginal,biviph-method	136
marginal,bivph-method	137
marginal,mdph-method	137
marginal,miph-method	138
marginal,mph-method	138
marginal,MPHstar-method	139
marginal_expectation	140
matrix_exponential	140
matrix_inverse	141
matrix_power	141
matrix_product	142
matrix_vanloan	142
max_diagonal	143
maximum	143
maximum,dph,dph-method	144
maximum,iph,iph-method	144
maximum,ph,ph-method	145
mdph	146
mdph-class	146
mdphdensity	147
mean,bivdph-method	147
mean,bivph-method	148
mean,dph-method	148
mean,mdph-method	149
mean,mph-method	149
mean,MPHstar-method	150
mean,ph-method	150
merge_matrices	151
merlang_dph	152
merlang_ph	152

mgevcdcf	153
mgevden	153
mgf	154
mgf,bivph-method	154
mgf,mph-method	155
mgf,ph-method	155
mgompertzcdf	156
mgompertzden	156
minimum	157
minimum,dph,dph-method	157
minimum,iph,iph-method	158
minimum,ph,ph-method	159
miph	159
miph-class	160
mixture	161
mixture,dph,dph-method	161
mixture,ph,ph-method	162
mloglogisticcdf	163
mloglogisticden	163
mlognormalcdf	164
mlognormalden	164
MoE	165
MoE,bivdph-method	165
MoE,dph-method	166
MoE,mdph-method	167
MoE,mph-method	168
MoE,ph-method	169
moment	171
moment,bivdph-method	171
moment,bivph-method	172
moment,dph-method	172
moment,mdph-method	173
moment,mph-method	173
moment,ph-method	174
mparetocdf	175
mparetoden	175
mph	176
mph-class	176
MPHstar	177
MPHstar-class	178
MPHstar_data_aggregation	178
MPHstar_EMstep_UNI	179
mweibullcdf	179
mweibullden	180
n_pos	180
new_state	181
Nfold	181
Nfold,dph-method	182

pgf	182
pgf,bivdph-method	183
pgf,dph-method	183
pgf,mdph-method	184
ph	184
ph-class	185
ph_laplace	186
phcdf	186
phdensity	187
plus_states	187
pow2_matrix	188
quan	188
quan,ph-method	189
random_reward	189
random_structure	190
random_structure_bivph	190
rcsph	191
rdphasetype	191
reg	192
reg,ph-method	192
revers_data_trans	194
rew_sanity_check	194
riph	195
rmatrixgev	195
rMDPHstar	196
rMIPHstar	196
rMPHstar	197
rphasetype	197
runge_kutta	198
show,bivdph-method	198
show,biviph-method	199
show,bivph-method	199
show,dph-method	199
show,iph-method	200
show,mdph-method	200
show,miph-method	201
show,mph-method	201
show,MPHstar-method	202
show,ph-method	202
show,sph-method	202
sim	203
sim,bivdph-method	203
sim,biviph-method	204
sim,bivph-method	204
sim,dph-method	205
sim,iph-method	205
sim,mdph-method	206
sim,miph-method	207

sim,mph-method	207
sim,MPHstar-method	208
sim,ph-method	209
sph	209
sph-class	210
sum_dph	210
sum_ph	211
TVR	211
TVR,dph-method	212
TVR,ph-method	212
tvr_dph	213
tvr_ph	213
var,bivdph-method	214
var,bivph-method	214
var,dph-method	215
var,mdph-method	215
var,mph-method	216
var,MPHstar-method	216
var,ph-method	217
vector_of_matrices	217
vector_of_matrices_2	218
vector_of_powers	218
Index	219

matrixdist-package	<i>Statistics for Matrix Distributions</i>
--------------------	--

Description

This package implements tools which are useful for the statistical analysis of discrete, continuous, multivariate, right-censored or regression variants of phase-type distributions. These distributions are absorption times of Markov jump processes, and thus the maximization of their likelihood for statistical estimation is best dealt with using the EM algorithm.

Author(s)

Martin Bladt and Jorge Yslas.

Maintainer: Martin Bladt <martinbladt@gmail.com>

References

- Asmussen, S., Nerman, O., & Olsson, M. (1996). Fitting phase-type distributions via the EM algorithm. *Scandinavian Journal of Statistics*, 23(4),419-441.
- Olsson, M. (1996). Estimation of phase-type distributions from censored data. *Scandinavian journal of statistics*, 24(4), 443-460.

Albrecher, H., & Bladt, M. (2019). Inhomogeneous phase-type distributions and heavy tails. *Journal of Applied Probability*, 56(4), 1044-1064.

Albrecher, H., Bladt, M., & Yslas, J. (2022). Fitting inhomogeneous Phase-Type distributions to data: The univariate and the multivariate case. *Scandinavian Journal of Statistics*, 49(1), 44-77

Albrecher, H., Bladt, M., Bladt, M., & Yslas, J. (2020). Mortality modeling and regression with matrix distributions. *Insurance: Mathematics and Economics*, 107, 68-87.

Bladt, M., & Yslas, J. (2022). Phase-type mixture-of-experts regression for loss severities. *Scandinavian Actuarial Journal*, 1-27.

Bladt, M. (2022). Phase-type distributions for claim severity regression modeling. *ASTIN Bulletin: The journal of the IAA*, 52(2), 417-448.

Bladt, M. (2023). A tractable class of Multivariate Phase-type distributions for loss modeling. *North American Actuarial Journal*, to appear.

Albrecher, H., Bladt, M., & Mueller, A. (2023). Joint lifetime modelling with matrix distributions. *Dependence Modeling*, 11(1), 1-22.

Bladt, M. & Yslas, J. (2023). Robust claim frequency modeling through phase-type mixture-of-experts regression. *Insurance: Mathematics and Economics*, 111, 1-22.

+,dph,dph-method

Sum method for discrete phase-type distributions

Description

Sum method for discrete phase-type distributions

Usage

```
## S4 method for signature 'dph,dph'
e1 + e2
```

Arguments

e1 An object of class [dph](#).
e2 An object of class [dph](#).

Value

An object of class [dph](#).

Examples

```
dph1 <- dph(structure = "general", dimension = 3)
dph2 <- dph(structure = "general", dimension = 5)
dph_sum <- dph1 + dph2
dph_sum
```

+,ph,ph-method	<i>Sum method for phase-type distributions</i>
----------------	--

Description

Sum method for phase-type distributions

Usage

```
## S4 method for signature 'ph,ph'
e1 + e2
```

Arguments

e1	An object of class ph .
e2	An object of class ph .

Value

An object of class [ph](#).

Examples

```
ph1 <- ph(structure = "general", dimension = 3)
ph2 <- ph(structure = "gcoxian", dimension = 5)
ph_sum <- ph1 + ph2
ph_sum
```

a_rungekutta	<i>Runge-Kutta for the calculation of the a vector in a EM step</i>
--------------	---

Description

Runge-Kutta for the calculation of the a vector in a EM step

Usage

```
a_rungekutta(avector, dt, h, S)
```

Arguments

avector	The a vector.
dt	Increment.
h	Step-length.
S	Sub-intensity matrix.

bivdph	<i>Constructor function for bivariate discrete phase-type distributions</i>
--------	---

Description

Constructor function for bivariate discrete phase-type distributions

Usage

```
bivdph(alpha = NULL, S11 = NULL, S12 = NULL, S22 = NULL, dimensions = c(3, 3))
```

Arguments

alpha	A probability vector.
S11	A sub-transition matrix.
S12	A matrix.
S22	A sub-transition matrix.
dimensions	The dimensions of the bivariate discrete phase-type (if no parameters are provided).

Value

An object of class `bivdph`.

Examples

```
bivdph(dimensions = c(3, 3))
S11 <- matrix(c(0.1, .5, .5, 0.1), 2, 2)
S12 <- matrix(c(.2, .3, .2, .1), 2, 2)
S22 <- matrix(c(0.2, 0, 0.1, 0.1), 2, 2)
bivdph(alpha = c(.5, .5), S11, S12, S22)
```

bivdph-class	<i>Bivariate discrete phase-type distributions</i>
--------------	--

Description

Class of objects for bivariate discrete phase-type distributions.

Value

Class object.

Slots

name Name of the discrete phase-type distribution.
pars A list comprising of the parameters.
fit A list containing estimation information.

bivdph_density	<i>Bivariate discrete phase-type joint density of the feed forward type</i>
----------------	---

Description

Bivariate discrete phase-type joint density of the feed forward type

Usage

```
bivdph_density(x, alpha, S11, S12, S22)
```

Arguments

x	Matrix of values.
alpha	Vector of initial probabilities.
S11	Sub-transition matrix.
S12	Matrix.
S22	Sub-transition matrix.

Value

Joint density at x.

bivdph_tail	<i>Bivariate discrete phase-type joint tail of the feed forward type</i>
-------------	--

Description

Bivariate discrete phase-type joint tail of the feed forward type

Usage

```
bivdph_tail(x, alpha, S11, S12, S22)
```

Arguments

x	Matrix of values.
alpha	Vector of initial probabilities.
S11	Sub-transition matrix.
S12	Matrix.
S22	Sub-transition matrix.

Value

Joint tail at x.

biviph	<i>Constructor function for bivariate inhomogeneous phase-type distributions</i>
--------	--

Description

Constructor function for bivariate inhomogeneous phase-type distributions

Usage

```
biviph(
  bivph = NULL,
  gfun = NULL,
  gfun_pars = NULL,
  alpha = NULL,
  S11 = NULL,
  S12 = NULL,
  S22 = NULL,
  dimensions = c(3, 3)
)
```

Arguments

bivph	An object of class bivph .
gfun	Vector of inhomogeneity transforms.
gfun_pars	List of parameters for the inhomogeneity functions.
alpha	A probability vector.
S11	A sub-intensity matrix.
S12	A matrix.
S22	A sub-intensity matrix.
dimensions	The dimensions of the bivariate phase-type (if no parameters are provided).

Value

An object of class `biviph`.

Examples

```
under_bivph <- bivph(dimensions = c(3, 3))
biviph(under_bivph, gfun = c("weibull", "pareto"), gfun_pars = list(c(2), c(3)))
```

biviph-class	<i>Bivariate inhomogeneous phase-type distributions</i>
--------------	---

Description

Class of objects for bivariate inhomogeneous phase-type distributions.

Value

Class object.

Slots

- name Name of the phase type distribution.
- gfun A list comprising of the parameters.

bivph	<i>Constructor function for bivariate phase-type distributions</i>
-------	--

Description

Constructor function for bivariate phase-type distributions

Usage

```
bivph(alpha = NULL, S11 = NULL, S12 = NULL, S22 = NULL, dimensions = c(3, 3))
```

Arguments

- alpha A probability vector.
- S11 A sub-intensity matrix.
- S12 A matrix.
- S22 A sub-intensity matrix.
- dimensions The dimensions of the bivariate phase-type (if no parameters are provided).

Value

An object of class `bivph`.

Examples

```
bivph(dimensions = c(3, 3))
S11 <- matrix(c(-1, .5, .5, -1), 2, 2)
S12 <- matrix(c(.2, .4, .3, .1), 2, 2)
S22 <- matrix(c(-2, 0, 1, -1), 2, 2)
bivph(alpha = c(.5, .5), S11, S12, S22)
```

bivph-class
Bivariate phase-type distributions

Description

Class of objects for bivariate phase-type distributions.

Value

Class object.

Slots

`name` Name of the phase-type distribution.
`pars` A list comprising of the parameters.
`fit` A list containing estimation information.

bivph_density
Bivariate phase-type joint density of the feed forward type

Description

Bivariate phase-type joint density of the feed forward type

Usage

```
bivph_density(x, alpha, S11, S12, S22)
```

Arguments

<code>x</code>	Matrix of values.
<code>alpha</code>	Vector of initial probabilities.
<code>S11</code>	Sub-intensity matrix.
<code>S12</code>	Matrix.
<code>S22</code>	Sub-intensity matrix.

Value

Joint density at x.

bivph_laplace	<i>Bivariate phase-type joint Laplace</i>
---------------	---

Description

Bivariate phase-type joint Laplace

Usage

```
bivph_laplace(r, alpha, S11, S12, S22)
```

Arguments

r	Matrix of values.
alpha	Vector of initial probabilities.
S11	Sub-intensity matrix.
S12	Matrix.
S22	Sub-intensity matrix.

Value

Joint laplace at r.

bivph_tail	<i>Bivariate phase-type joint tail of the feed forward type</i>
------------	---

Description

Bivariate phase-type joint tail of the feed forward type

Usage

```
bivph_tail(x, alpha, S11, S12, S22)
```

Arguments

x	Matrix of values.
alpha	Vector of initial probabilities.
S11	Sub-intensity matrix.
S12	Matrix.
S22	Sub-intensity matrix.

Value

Joint tail at x .

cdf

New generic for the distribution of matrix distributions

Description

Methods are available for objects of class [ph](#).

Usage

```
cdf(x, ...)
```

Arguments

x	An object of the model class.
$\dots$	Further parameters to be passed on.

Value

CDF from the matrix distribution.

cdf,dph-method

Distribution method for discrete phase-type distributions

Description

Distribution method for discrete phase-type distributions

Usage

```
## S4 method for signature 'dph'
cdf(x, q, lower.tail = TRUE)
```

Arguments

x	An object of class dph .
q	A vector of locations.
<code>lower.tail</code>	Logical parameter specifying whether lower tail (CDF) or upper tail is computed.

Value

A vector containing the CDF evaluations at the given locations.

Examples

```
obj <- dph(structure = "general")
cdf(obj, c(1, 2, 3))
```

cdf,iph-method

*Distribution method for inhomogeneous phase-type distributions***Description**

Distribution method for inhomogeneous phase-type distributions

Usage

```
## S4 method for signature 'iph'
cdf(x, q, lower.tail = TRUE)
```

Arguments

x	An object of class iph .
q	A vector of locations.
lower.tail	Logical parameter specifying whether lower tail (CDF) or upper tail is computed.

Value

A vector containing the CDF evaluations at the given locations.

Examples

```
obj <- iph(ph(structure = "general"), gfun = "weibull", gfun_pars = 2)
cdf(obj, c(1, 2, 3))
```

cdf,miph-method

*Distribution method for multivariate inhomogeneous phase-type distributions***Description**

Distribution method for multivariate inhomogeneous phase-type distributions

Usage

```
## S4 method for signature 'miph'
cdf(x, y, lower.tail = TRUE)
```

Arguments

x	An object of class <code>mph</code> .
y	A matrix of observations.
lower.tail	Logical parameter specifying whether lower tail (CDF) or upper tail is computed.

Value

A list containing the locations and corresponding CDF evaluations.

Examples

```
under_mph <- mph(structure = c("general", "general"))
obj <- mph(under_mph, gfun = c("weibull", "pareto"), gfun_pars = list(c(2), c(3)))
cdf(obj, c(1, 2))
```

cdf,mph-method

Distribution method for multivariate phase-type distributions

Description

Distribution method for multivariate phase-type distributions

Usage

```
## S4 method for signature 'mph'
cdf(x, y, lower.tail = TRUE)
```

Arguments

x	An object of class <code>mph</code> .
y	A matrix of observations.
lower.tail	Logical parameter specifying whether lower tail (CDF) or upper tail is computed.

Value

A list containing the locations and corresponding CDF evaluations.

Examples

```
obj <- mph(structure = c("general", "general"))
cdf(obj, matrix(c(0.5, 1), ncol = 2))
```

cdf,ph-method	<i>Distribution method for phase-type distributions</i>
---------------	---

Description

Distribution method for phase-type distributions

Usage

```
## S4 method for signature 'ph'
cdf(x, q, lower.tail = TRUE)
```

Arguments

x	An object of class ph .
q	A vector of locations.
lower.tail	Logical parameter specifying whether lower tail (CDF) or upper tail is computed.

Value

A vector containing the CDF evaluations at the given locations.

Examples

```
obj <- ph(structure = "general")
cdf(obj, c(1, 2, 3))
```

clone_matrix	<i>Clone a matrix</i>
--------------	-----------------------

Description

Clone a matrix

Usage

```
clone_matrix(m)
```

Arguments

m	A matrix.
---	-----------

Value

A clone of the matrix.

clone_vector	<i>Clone a vector</i>
--------------	-----------------------

Description

Clone a vector

Usage

```
clone_vector(v)
```

Arguments

v A vector.

Value

A clone of the vector.

coef,bivdph-method	<i>Coef method for bivdph class</i>
--------------------	-------------------------------------

Description

Coef method for bivdph class

Usage

```
## S4 method for signature 'bivdph'
coef(object)
```

Arguments

object An object of class [bivdph](#).

Value

Parameters of bivariate discrete phase-type model.

Examples

```
obj <- bivdph(dimensions = c(3, 3))
coef(obj)
```

coef,biviph-method	<i>Coef method for biviph class</i>
--------------------	-------------------------------------

Description

Coef method for biviph class

Usage

```
## S4 method for signature 'biviph'  
coef(object)
```

Arguments

object An object of class [biviph](#).

Value

Parameters of bivariate inhomogeneous phase-type model.

Examples

```
under_bivph <- bivph(dimensions = c(3, 3))  
obj <- biviph(under_bivph, gfun = c("weibull", "pareto"), gfun_pars = list(c(2), c(3)))  
coef(obj)
```

coef,bivph-method	<i>Coef method for biviph class</i>
-------------------	-------------------------------------

Description

Coef method for biviph class

Usage

```
## S4 method for signature 'biviph'  
coef(object)
```

Arguments

object An object of class [bivph](#).

Value

Parameters of bivariate phase-type model.

Value

Parameters of iph model.

Examples

```
obj <- iph(ph(structure = "general", dimension = 2), gfun = "lognormal", gfun_pars = 2)
coef(obj)
```

coef,mdph-method	<i>Coef method for mdph class</i>
------------------	-----------------------------------

Description

Coef method for mdph class

Usage

```
## S4 method for signature 'mdph'
coef(object)
```

Arguments

object An object of class [mdph](#).

Value

Parameters of multivariate discrete phase-type model.

Examples

```
obj <- mdph(structure = c("general", "general"))
coef(obj)
```

coef,ph-method	<i>Coef method for ph class</i>
----------------	---------------------------------

Description

Coef method for ph class

Usage

```
## S4 method for signature 'ph'
coef(object)
```

Arguments

object An object of class [ph](#).

Value

Parameters of ph model.

Examples

```
obj <- ph(structure = "general")
coef(obj)
```

coef , sph-method	<i>Coef method for sph Class</i>
-------------------	----------------------------------

Description

Coef method for sph Class

Usage

```
## S4 method for signature 'sph'
coef(object)
```

Arguments

object An object of class [sph](#).

Value

Parameters of sph model.

cor , bivdph-method	<i>Cor method for bivdph class</i>
---------------------	------------------------------------

Description

Cor method for bivdph class

Usage

```
## S4 method for signature 'bivdph'
cor(x)
```

Arguments

x An object of class [bivdph](#).

Value

The correlation matrix of the bivariate discrete phase-type distribution.

Examples

```
obj <- bivdph(dimensions = c(3, 3))
cor(obj)
```

cor,bivph-method	<i>Cor method for bivph class</i>
------------------	-----------------------------------

Description

Cor method for bivph class

Usage

```
## S4 method for signature 'bivph'
cor(x)
```

Arguments

x An object of class [bivph](#).

Value

The correlation matrix of the bivariate phase-type distribution.

Examples

```
obj <- bivph(dimensions = c(3, 3))
cor(obj)
```

cor,mdph-method	<i>Cor method for multivariate discrete phase-type distributions</i>
-----------------	--

Description

Cor method for multivariate discrete phase-type distributions

Usage

```
## S4 method for signature 'mdph'
cor(x)
```

Arguments

x An object of class `mdph`.

Value

The correlation matrix of the multivariate discrete phase-type distribution.

Examples

```
obj <- mdph(structure = c("general", "general"))
cor(obj)
```

cor,mph-method

Cor method for multivariate phase-type distributions

Description

Cor method for multivariate phase-type distributions

Usage

```
## S4 method for signature 'mph'
cor(x)
```

Arguments

x An object of class `mph`.

Value

The correlation matrix of the multivariate phase-type distribution.

Examples

```
obj <- mph(structure = c("general", "general"))
cor(obj)
```

cor,MPHstar-method	<i>Cor method for MPHstar class</i>
--------------------	-------------------------------------

Description

Cor method for MPHstar class

Usage

```
## S4 method for signature 'MPHstar'
cor(x)
```

Arguments

x An object of class [MPHstar](#).

Value

The correlation matrix of the MPHstar distribution.

Examples

```
obj <- MPHstar(structure = "general")
cor(obj)
```

csph_density	<i>Bivariate phase-type joint density of the common shock type</i>
--------------	--

Description

Bivariate phase-type joint density of the common shock type

Usage

```
csph_density(x, alpha, S, P, Q1, Q2)
```

Arguments

- x Matrix of values.
- alpha Vector of initial probabilities.
- S Sub-intensity matrix.
- P Matrix.
- Q1 Sub-intensity matrix.
- Q2 Sub-intensity matrix.

Value

Joint density at x.

csph_density_par	<i>Bivariate phase-type joint density of the common shock type</i>
------------------	--

Description

Bivariate phase-type joint density of the common shock type

Usage

```
csph_density_par(x, alpha, S, P, Q1, Q2)
```

Arguments

x	Matrix of values.
alpha	Vector of initial probabilities.
S	Sub-intensity matrix.
P	Matrix.
Q1	Sub-intensity matrix.
Q2	Sub-intensity matrix.

Value

Joint density at x.

csph_density_test	<i>Bivariate phase-type joint density of the common shock type</i>
-------------------	--

Description

Bivariate phase-type joint density of the common shock type

Usage

```
csph_density_test(x, alpha, S, P, Q1, Q2)
```

Arguments

x	Matrix of values.
alpha	Vector of initial probabilities.
S	Sub-intensity matrix.
P	Matrix.
Q1	Sub-intensity matrix.
Q2	Sub-intensity matrix.

Value

Joint density at x.

csph_densityrk	<i>Bivariate phase-type joint density of the common shock type</i>
----------------	--

Description

Bivariate phase-type joint density of the common shock type

Usage

```
csph_densityrk(x, alpha, S, P, Q1, Q2, h)
```

Arguments

- x Matrix of values.
- alpha Vector of initial probabilities.
- S Sub-intensity matrix.
- P Matrix.
- Q1 Sub-intensity matrix.
- Q2 Sub-intensity matrix.
- h step size.

Value

Joint density at x.

cumulate_matrix	<i>Cumulate matrix</i>
-----------------	------------------------

Description

Creates a new matrix with entries the cumulated rows of A.

Usage

```
cumulate_matrix(A)
```

Arguments

- A A matrix.

Value

The cumulated matrix.

cumulate_vector	<i>Cumulate vector</i>
-----------------	------------------------

Description

Creates a new vector with entries the cumulated entries of A.

Usage

cumulate_vector(A)

Arguments

A A vector.

Value

The cumulated vector.

data_aggregation	<i>Aggregate data to improve computation speed</i>
------------------	--

Description

Aggregate data to improve computation speed
Aggregate data to improve computation speed
Aggregate data to improve computation speed

Usage

data_aggregation(y, w)

data_aggregation(y, w)

data_aggregation(y, w)

Arguments

y Observations. Either a vector or a matrix.
w Respective weights of observations

Value

Returns a named list with unique observations and associated weights. If *y* is a vector then the unique observations are given, otherwise the unique rows are returned.

Returns a named list with unique observations and associated weights. If *y* is a vector then the unique observations are given, otherwise the unique rows are returned.

Returns a named list with unique observations and associated weights. If *y* is a vector then the unique observations are given, otherwise the unique rows are returned.

default_step_length	<i>Default size of the steps in the RK</i>
---------------------	--

Description

Computes the default step length for a matrix *S* to be employed in the RK method.

Usage

```
default_step_length(S)
```

Arguments

<i>S</i>	Sub-intensity matrix.
----------	-----------------------

Value

The step length for *S*.

dens	<i>New generic for the density of matrix distributions</i>
------	--

Description

Methods are available for objects of class [ph](#).

Usage

```
dens(x, ...)
```

Arguments

<i>x</i>	An object of the model class.
<i>...</i>	Further parameters to be passed on.

Value

Density from the matrix distribution.

dens,bivdph-method	<i>Density method for bivariate discrete phase-type distributions</i>
--------------------	---

Description

Density method for bivariate discrete phase-type distributions

Usage

```
## S4 method for signature 'bivdph'  
dens(x, y)
```

Arguments

x	An object of class <code>bivdph</code> .
y	A matrix of locations.

Value

A vector containing the joint density evaluations at the given locations.

Examples

```
obj <- bivdph(dimensions = c(3, 3))  
dens(obj, matrix(c(1, 2), ncol = 2))
```

dens,biviph-method	<i>Density method for bivariate inhomogeneous phase-type distributions</i>
--------------------	--

Description

Density method for bivariate inhomogeneous phase-type distributions

Usage

```
## S4 method for signature 'biviph'  
dens(x, y)
```

Arguments

x	An object of class <code>biviph</code> .
y	A matrix of locations.

Value

A vector containing the joint density evaluations at the given locations.

Examples

```
under_bivph <- bivph(dimensions = c(3, 3))
obj <- bivph(under_bivph, gfun = c("weibull", "pareto"), gfun_pars = list(c(2), c(3)))
dens(obj, matrix(c(0.5, 1), ncol = 2))
```

dens,bivph-method	<i>Density method for bivariate phase-type distributions</i>
-------------------	--

Description

Density method for bivariate phase-type distributions

Usage

```
## S4 method for signature 'bivph'
dens(x, y)
```

Arguments

x	An object of class bivph .
y	A matrix of locations.

Value

A vector containing the joint density evaluations at the given locations.

Examples

```
obj <- bivph(dimensions = c(3, 3))
dens(obj, matrix(c(0.5, 1), ncol = 2))
```

dens,dph-method	<i>Density method for discrete phase-type distributions</i>
-----------------	---

Description

Density method for discrete phase-type distributions

Usage

```
## S4 method for signature 'dph'
dens(x, y)
```

Arguments

x	An object of class dph .
y	A vector of locations.

Value

A vector containing the density evaluations at the given locations.

Examples

```
obj <- dph(structure = "general")
dens(obj, c(1, 2, 3))
```

dens,iph-method

Density method for inhomogeneous phase-type distributions

Description

Density method for inhomogeneous phase-type distributions

Usage

```
## S4 method for signature 'iph'
dens(x, y)
```

Arguments

x	An object of class iph .
y	A vector of locations.

Value

A vector containing the density evaluations at the given locations.

Examples

```
obj <- iph(ph(structure = "general"), gfun = "weibull", gfun_pars = 2)
dens(obj, c(1, 2, 3))
```

dens,mdph-method

Density method for multivariate discrete phase-type distributions

Description

Density method for multivariate discrete phase-type distributions

Usage

```
## S4 method for signature 'mdph'
dens(x, y)
```

Arguments

`x` An object of class `mdph`.
`y` A matrix of locations.

Value

A vector containing the joint density evaluations at the given locations.

Examples

```
obj <- mdph(structure = c("general", "general"))
dens(obj, matrix(c(1, 1), ncol = 2))
```

dens,miph-method	<i>Density method for multivariate inhomogeneous phase-type distributions</i>
------------------	---

Description

Density method for multivariate inhomogeneous phase-type distributions

Usage

```
## S4 method for signature 'miph'
dens(x, y, delta = NULL)
```

Arguments

`x` An object of class `miph`.
`y` A matrix of observations.
`delta` Matrix with right-censoring indicators (1 uncensored, 0 right censored).

Value

A list containing the locations and corresponding density evaluations.

Examples

```
under_mph <- mph(structure = c("general", "general"))
obj <- miph(under_mph, gfun = c("weibull", "pareto"), gfun_pars = list(c(2), c(3)))
dens(obj, c(1, 2))
```

dens, mph-method	<i>Density method for multivariate phase-type distributions</i>
------------------	---

Description

Density method for multivariate phase-type distributions

Usage

```
## S4 method for signature 'mph'
dens(x, y, delta = NULL)
```

Arguments

x	An object of class mph .
y	A matrix of observations.
delta	Matrix with right-censoring indicators (1 uncensored, 0 right censored).

Value

A list containing the locations and corresponding density evaluations.

Examples

```
obj <- mph(structure = c("general", "general"))
dens(obj, matrix(c(0.5, 1), ncol = 2))
```

dens, ph-method	<i>Density method for phase-type distributions</i>
-----------------	--

Description

Density method for phase-type distributions

Usage

```
## S4 method for signature 'ph'
dens(x, y)
```

Arguments

x	An object of class ph .
y	A vector of locations.

Value

A vector containing the density evaluations at the given locations.

Examples

```
obj <- ph(structure = "general")
dens(obj, c(1, 2, 3))
```

dph

*Constructor function for discrete phase-type distributions***Description**

Constructor function for discrete phase-type distributions

Usage

```
dph(
  alpha = NULL,
  S = NULL,
  structure = NULL,
  dimension = 3,
  block_sizes = NULL,
  probs = NULL,
  q = NULL
)
```

Arguments

alpha	A probability vector.
S	A sub-transition matrix.
structure	A valid dph structure: "general", "coxian", "hyperexponential", "gcoxian", "gerlang", "erlang", or "merlang".
dimension	The dimension of the dph structure (if structure is provided).
block_sizes	Optional integer vector with Erlang block sizes when structure = "merlang" (or aliases "mixederlang", "mixed_erlang").
probs	Optional mixture weights used only for structure = "merlang".
q	Optional forward probabilities used only for structure = "merlang".

Value

An object of class [dph](#).

Examples

```
dph(structure = "general", dim = 5)
dph(structure = "merlang", block_sizes = c(2, 3))
dph(alpha = c(0.5, 0.5), S = matrix(c(0.1, 0.5, 0.5, 0.2), 2, 2))
```

dph-class	<i>Discrete phase-type distributions</i>
-----------	--

Description

Class of objects for discrete phase-type distributions.

Value

Class object.

Slots

name Name of the discrete phase-type distribution.

pars A list comprising of the parameters.

fit A list containing estimation information.

dph_pgf	<i>Pgf of a discrete phase-type distribution</i>
---------	--

Description

Computes the pgf at z of a discrete phase-type distribution with parameters α and S .

Usage

```
dph_pgf(z, alpha, S)
```

Arguments

z Vector of real values.

α Vector of initial probabilities.

S Sub-transition matrix.

Value

Laplace transform at r .

dphcdf	<i>Discrete phase-type cdf</i>
--------	--------------------------------

Description

Computes the cdf (tail) of a discrete phase-type distribution with parameters `alpha` and `S` at `x`.

Usage

```
dphcdf(x, alpha, S, lower_tail = TRUE)
```

Arguments

<code>x</code>	Non-negative value.
<code>alpha</code>	Initial probabilities.
<code>S</code>	Sub-intensity matrix.
<code>lower_tail</code>	Cdf or tail.

Value

The cdf (tail) at `x`.

dphdensity	<i>Discrete phase-type density</i>
------------	------------------------------------

Description

Computes the density of discrete phase-type distribution with parameters `alpha` and `S` at `x`.

Usage

```
dphdensity(x, alpha, S)
```

Arguments

<code>x</code>	Non-negative value.
<code>alpha</code>	Initial probabilities.
<code>S</code>	Sub-transition matrix.

Value

The density at `x`.

EM_step_mPH_rc	<i>EM step for the mPH class with right-censoring, for different marginal sub-intensity matrices</i>
----------------	--

Description

EM step for the mPH class with right-censoring, for different marginal sub-intensity matrices

Usage

```
EM_step_mPH_rc(alpha, S_list, y, delta, h)
```

Arguments

alpha	Common initial distribution vector.
S_list	List of marginal sub-intensity matrices.
y	Matrix of marginal observations.
delta	Matrix with right-censoring indications (1 uncensored, 0 right-censored).
h	Tolerance of uniformization.

embedded_mc	<i>Embedded Markov chain of a sub-intensity matrix</i>
-------------	--

Description

Returns the transition probabilities of the embedded Markov chain determined the sub-intensity matrix.

Usage

```
embedded_mc(S)
```

Arguments

S	A sub-intensity matrix.
---	-------------------------

Value

The embedded Markov chain.

embedded_mc_cs	<i>Embedded Markov chain of a sub-intensity matrix</i>
----------------	--

Description

Returns the transition probabilities of the embedded Markov chain determined by the sub-intensity matrix.

Usage

```
embedded_mc_cs(S, P)
```

Arguments

S	A sub-intensity matrix.
P	A matrix of exit rates to external states.

Value

The embedded Markov chain.

EMstep_bivdph	<i>EM for discrete bivariate phase-type</i>
---------------	---

Description

EM for discrete bivariate phase-type

Usage

```
EMstep_bivdph(alpha, S11, S12, S22, obs, weight)
```

Arguments

alpha	Initial probabilities.
S11	Sub-transition matrix.
S12	Matrix.
S22	Sub-transition matrix.
obs	The observations.
weight	The weights for the observations.

EMstep_bivdph_MoE	<i>EM for discrete bivariate phase-type MoE</i>
-------------------	---

Description

EM for discrete bivariate phase-type MoE

Usage

```
EMstep_bivdph_MoE(alpha, S11, S12, S22, obs, weight)
```

Arguments

alpha	Initial probabilities.
S11	Sub-transition matrix.
S12	Matrix.
S22	Sub-transition matrix.
obs	The observations.
weight	The weights for the observations.

EMstep_bivph	<i>EM for bivariate phase-type distributions using Pade for matrix exponential</i>
--------------	--

Description

EM for bivariate phase-type distributions using Pade for matrix exponential

Usage

```
EMstep_bivph(alpha, S11, S12, S22, obs, weight)
```

Arguments

alpha	Initial probabilities.
S11	Sub-intensity.
S12	A matrix.
S22	Sub-intensity.
obs	The observations.
weight	The weights for the observations.

Value

Fitted alpha, S11, S12 and S22 after one iteration.

EMstep_dph	<i>EM for discrete phase-type</i>
------------	-----------------------------------

Description

EM for discrete phase-type

Usage

```
EMstep_dph(alpha, S, obs, weight, erlang = FALSE, merlang_blocks = integer(0))
```

Arguments

alpha	Initial probabilities.
S	Sub-transition matrix.
obs	The observations.
weight	The weights for the observations.
erlang	Logical. If TRUE, performs the exact Erlang-type M-step with one repeated transition probability and $\alpha = (1, 0, \dots, 0)$.
merlang_blocks	Optional integer vector of block sizes for exact mixture-of-Erlangs M-step.

EMstep_dph_MoE	<i>EM for discrete phase-type MoE</i>
----------------	---------------------------------------

Description

EM for discrete phase-type MoE

Usage

```
EMstep_dph_MoE(alpha, S, obs, weight)
```

Arguments

alpha	Initial probabilities.
S	Sub-transition matrix.
obs	The observations.
weight	The weights for the observations.

EMstep_mdph	<i>EM for multivariate discrete phase-type</i>
-------------	--

Description

EM for multivariate discrete phase-type

Usage

```
EMstep_mdph(alpha, S_list, obs, weight)
```

Arguments

alpha	Initial probabilities.
S_list	List of marginal sub-transition matrices.
obs	The observations.
weight	The weights for the observations.

EMstep_mdph_MoE	<i>EM for multivariate discrete phase-type MoE</i>
-----------------	--

Description

EM for multivariate discrete phase-type MoE

Usage

```
EMstep_mdph_MoE(alpha, S_list, obs, weight)
```

Arguments

alpha	Initial probabilities.
S_list	List of marginal sub-transition matrices.
obs	The observations.
weight	The weights for the observations.

EMstep_MoE_PADE	<i>EM for PH-MoE</i>
-----------------	----------------------

Description

No recycling of information

Usage

```
EMstep_MoE_PADE(alpha, S, obs, weight, rcens, rcweight)
```

Arguments

alpha	Initial probabilities.
S	Sub-intensity matrix.
obs	The observations.
weight	The weights for the observations.
rcens	Censored observations.
rcweight	The weights for the censored observations.

EMstep_PADE	<i>EM for phase-type distributions using Pade approximation for matrix exponential</i>
-------------	--

Description

EM for phase-type distributions using Pade approximation for matrix exponential

Usage

```
EMstep_PADE(
  h,
  alpha,
  S,
  obs,
  weight,
  rcens,
  rcweight,
  erlang = FALSE,
  merlang_blocks = integer(0)
)
```

Arguments

h	Nuisance parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
obs	The observations.
weight	The weights for the observations.
rcens	Censored observations.
rcweight	The weights for the censored observations.
erlang	Logical. If TRUE, performs the exact Erlang M-step with one repeated rate and $\alpha = (1, 0, \dots, 0)$.
merlang_blocks	Optional integer vector of block sizes for exact mixture-of-Erlangs M-step.

EMstep_RK

*EM step for phase-type using Runge-Kutta***Description**

Computes one step of the EM algorithm by using a Runge-Kutta method of fourth order.

Usage

```
EMstep_RK(
  h,
  alpha,
  S,
  obs,
  weight,
  rcens,
  rcweight,
  erlang = FALSE,
  merlang_blocks = integer(0)
)
```

Arguments

h	Step-length.
alpha	Initial probabilities.
S	Sub-intensity matrix.
obs	The observations.
weight	The weights for the observations.
rcens	Censored observations.
rcweight	The weights for the censored observations.
erlang	Logical. If TRUE, performs the exact Erlang M-step with one repeated rate and $\alpha = (1, 0, \dots, 0)$.
merlang_blocks	Optional integer vector of block sizes for exact mixture-of-Erlangs M-step.

EMstep_UNI

*EM for phase-type using uniformization for matrix exponential***Description**

EM for phase-type using uniformization for matrix exponential

Usage

```
EMstep_UNI(
  h,
  alpha,
  S,
  obs,
  weight,
  rcens,
  rcweight,
  erlang = FALSE,
  merlang_blocks = integer(0)
)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
obs	The observations.
weight	The weights for the observations.
rcens	Censored observations.
rcweight	The weights for the censored observations.
erlang	Logical. If TRUE, performs the exact Erlang M-step with one repeated rate and $\alpha = (1, 0, \dots, 0)$.
merlang_blocks	Optional integer vector of block sizes for exact mixture-of-Erlangs M-step.

EMstep_UNI_intervalCensoring

*EM for phase-type using uniformization for matrix exponential***Description**

EM for phase-type using uniformization for matrix exponential

Usage

```
EMstep_UNI_intervalCensoring(  
  h,  
  alpha,  
  S,  
  obs,  
  weight,  
  rcens,  
  rcweight,  
  erlang = FALSE,  
  merlang_blocks = integer(0)  
)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
obs	The observations.
weight	The weights for the observations.
rcens	Matrix of censored observations' bounds. Each row corresponds to a given observation.
rcweight	The weights for the censored observations.
erlang	Logical. If TRUE, performs the exact Erlang M-step with one repeated rate and $\alpha = (1, 0, \dots, 0)$.
merlang_blocks	Optional integer vector of block sizes for exact mixture-of-Erlangs M-step.

evaluate	<i>New generic for evaluating survival matrix distributions</i>
----------	---

Description

Methods are available for objects of class [sph](#).

Usage

```
evaluate(x, subject, ...)
```

Arguments

x	An object of the model class.
subject	A vector of data.
...	Further parameters to be passed on.

evaluate,sph-method	<i>Evaluation method for sph Class</i>
---------------------	--

Description

Evaluation method for sph Class

Usage

```
## S4 method for signature 'sph'
evaluate(x, subject)
```

Arguments

- x An object of class [sph](#).
- subject Covariates of a single subject.

Value

A [ph](#) model.

expm_terms	<i>expm terms of phase-type likelihood using uniformization</i>
------------	---

Description

expm terms of phase-type likelihood using uniformization

Usage

```
expm_terms(h, S, obs)
```

Arguments

- h Positive parameter.
- S Sub-intensity matrix.
- obs The observations.

 expmat

Matrix exponential

Description

Armadillo matrix exponential implementation.

Usage

expmat(A)

Arguments

A A matrix.

Value

exp(A).

 find_n

Find n such that $P(N > n) = h$ with N Poisson distributed

Description

Find n such that $P(N > n) = h$ with N Poisson distributed

Usage

find_n(h, lambda)

Arguments

h Probability.
 lambda Mean of Poisson random variable.

Value

Integer satisfying condition.

find_weight	<i>Find weight of observations</i>
-------------	------------------------------------

Description

Find weight of observations

Usage

```
find_weight(x)
```

Arguments

x A vector of observations from which we want to know their weights.

Value

A matrix with unique observations as first column and associated weights for second column.

Fisher	<i>New generic for obtaining the Fisher information of survival matrix distributions</i>
--------	--

Description

Methods are available for objects of class [sph](#).

Usage

```
Fisher(x, ...)
```

Arguments

x An object of the model class.
... Further parameters to be passed on.

Fisher, sph-method	<i>Fisher information method for sph class</i>
--------------------	--

Description

Fisher information method for sph class

Usage

```
## S4 method for signature 'sph'
Fisher(x, y, X, w = numeric(0))
```

Arguments

x	An object of class sph .
y	Independent variate.
X	Matrix of covariates.
w	Weights.

Value

A matrix.

fit	<i>New generic for estimating matrix distributions</i>
-----	--

Description

Methods are available for objects of class [ph](#).

Usage

```
fit(x, y, ...)
```

Arguments

x	An object of the model class.
y	A vector of data.
...	Further parameters to be passed on.

Value

An object of the fitted model class.

fit,bivdph-method	<i>Fit method for bivdph Class</i>
-------------------	------------------------------------

Description

Fit method for bivdph Class

Usage

```
## S4 method for signature 'bivdph'  
fit(x, y, weight = numeric(0), stepsEM = 1000, every = 10)
```

Arguments

x	An object of class bivdph .
y	A matrix with the data.
weight	Vector of weights.
stepsEM	Number of EM steps to be performed.
every	Number of iterations between likelihood display updates.

Value

An object of class [bivdph](#).

Examples

```
obj <- bivdph(dimensions = c(3, 3))  
data <- sim(obj, n = 100)  
fit(obj, data, stepsEM = 100, every = 50)
```

fit,bivph-method	<i>Fit method for bivph Class</i>
------------------	-----------------------------------

Description

Fit method for bivph Class

Usage

```
## S4 method for signature 'bivph'
fit(
  x,
  y,
  weight = numeric(0),
  stepsEM = 1000,
  maxit = 100,
  reltol = 1e-08,
  every = 10
)
```

Arguments

x	An object of class bivph .
y	A matrix with the data.
weight	Vector of weights.
stepsEM	Number of EM steps to be performed.
maxit	Maximum number of iterations when optimizing g functions.
reltol	Relative tolerance when optimizing g functions.
every	Number of iterations between likelihood display updates.

Value

An object of class [bivph](#).

Examples

```
obj <- bivph(dimensions = c(3, 3))
data <- sim(obj, n = 100)
fit(obj, data, stepsEM = 100, every = 50)
```

fit,dph-method

Fit method for dph class

Description

Fit method for dph class

Usage

```
## S4 method for signature 'dph'
fit(
  x,
  y,
  weight = numeric(0),
  stepsEM = 1000,
  every = 100,
  erlang = NULL,
  merlang_blocks = NULL
)
```

Arguments

x	An object of class dph .
y	Vector or data.
weight	Vector of weights.
stepsEM	Number of EM steps to be performed.
every	Number of iterations between likelihood display updates.
erlang	Logical flag for exact Erlang-type EM updates with one repeated transition probability. If NULL, this is auto-detected from the model name.
merlang_blocks	Optional integer vector with Erlang block sizes for exact mixture-of-Erlangs EM updates. If NULL, it is auto-read from object attributes when available.

Value

An object of class [dph](#).

Examples

```
obj <- dph(structure = "general", dimension = 2)
data <- sim(obj, n = 100)
fit(obj, data, stepsEM = 100, every = 20)
```

fit,mdph-method

Fit method for mdph Class

Description

Fit method for mdph Class

Usage

```
## S4 method for signature 'mdph'
fit(x, y, weight = numeric(0), stepsEM = 1000, every = 10)
```

Arguments

x	An object of class <code>mdph</code> .
y	A matrix with the data.
weight	Vector of weights.
stepsEM	Number of EM steps to be performed.
every	Number of iterations between likelihood display updates.

Value

An object of class `mdph`.

Examples

```
obj <- mdph(structure = c("general", "general"))
data <- sim(obj, n = 100)
fit(obj, data, stepsEM = 100, every = 50)
```

fit,mph-method	<i>Fit method for mph Class</i>
----------------	---------------------------------

Description

Fit method for mph Class

Usage

```
## S4 method for signature 'mph'
fit(
  x,
  y,
  delta = numeric(0),
  stepsEM = 1000,
  equal_marginals = FALSE,
  r = 1,
  maxit = 100,
  reltol = 1e-08
)
```

Arguments

x	An object of class <code>mph</code> .
y	Matrix of data.
delta	Matrix with right-censoring indicators (1 uncensored, 0 right censored).
stepsEM	Number of EM steps to be performed.

equal_marginals	Logical. If TRUE, all marginals are fitted to be equal.
r	Sub-sampling parameter, defaults to 1.
maxit	Maximum number of iterations when optimizing g function.
reltol	Relative tolerance when optimizing g function.

Examples

```
obj <- mph(structure = c("general", "coxian"))
data <- sim(obj, 100)
fit(x = obj, y = data, stepsEM = 20)
```

fit,MPHstar-method	<i>Fit method for mph class</i>
--------------------	---------------------------------

Description

Fit method for mph class

Usage

```
## S4 method for signature 'MPHstar'
fit(
  x,
  y,
  weight = numeric(0),
  stepsEM = 1000,
  uni_epsilon = 1e-04,
  zero_tol = 1e-04,
  every = 100,
  plot = F,
  r = 1,
  replace = F
)
```

Arguments

x	An object of class MPHstar .
y	A matrix of marginal data.
weight	A matrix of marginal weights.
stepsEM	The number of EM steps to be performed, defaults to 1000.
uni_epsilon	The epsilon parameter for the uniformization method, defaults to 1e-4.
zero_tol	The smallest value that a reward can take (to avoid numerical instability), defaults to 1e-4.

every	The number of iterations between likelihood display updates. The originating distribution is used, given that there is no explicit density.
plot	Boolean that determines if the plot of the loglikelihood evolution is plotted, defaults to False.
r	The sub-sampling proportion for stochastic EM, defaults to 1.
replace	Boolean that determines if sub-sampling is done with replacement or not, defaults to False.

Value

An object of class [MPHstar](#).

Examples

```
set.seed(123)
obj <- MPHstar(structure = "general")
data <- sim(obj, 100)
fit(obj, data, stepsEM = 20)
```

fit,ph-method	<i>Fit method for ph class</i>
---------------	--------------------------------

Description

Fit method for ph class

Usage

```
## S4 method for signature 'ph'
fit(
  x,
  y,
  weight = numeric(0),
  rcen = numeric(0),
  rcenweight = numeric(0),
  stepsEM = 1000,
  methods = c("RK", "RK"),
  rkstep = NA,
  uni_epsilon = NA,
  maxit = 100,
  reltol = 1e-08,
  every = 100,
  optim_method = "BFGS",
  r = 1,
  burn = 0,
  erlang = NULL,
  merlang_blocks = NULL,
```

```

    plot_progress = FALSE,
    plot_every = 100,
    plot_breaks = 50,
    plot_pause = 0.01
  )

```

Arguments

x	An object of class ph .
y	Vector or data.
weight	Vector of weights.
rcen	Vector of right-censored observations. If a matrix is passed as argument, then observations are considered interval-censored
rcenweight	Vector of weights for right-censored observations.
stepsEM	Number of EM steps to be performed.
methods	Methods to use for matrix exponential calculation: RM, UNI or PADE.
rkstep	Runge-Kutta step size (optional).
uni_epsilon	Epsilon parameter for uniformization method.
maxit	Maximum number of iterations when optimizing g function.
reltol	Relative tolerance when optimizing g function.
every	Number of iterations between likelihood display updates.
optim_method	Optimization method for optim function.
r	Sub-sampling proportion for stochastic EM, defaults to 1.
burn	Percentage of iterations (from 0 to 100) run on histogram midpoints of y with bin counts as weights, before switching to the standard fit on original data. Available only for uncensored data.
erlang	Logical flag for exact Erlang EM updates with one repeated rate. If NULL, this is auto-detected from the model name.
merlang_blocks	Optional integer vector with Erlang block sizes for exact mixture-of-Erlangs EM updates. If NULL, it is auto-read from object attributes when available.
plot_progress	Logical flag. If TRUE, a progress plot is redrawn during EM: histogram plus fitted density for uncensored data, or Kaplan-Meier plus fitted survival for right-censored data.
plot_every	Positive integer. Plot refresh frequency in iterations when plot_progress = TRUE. Defaults to 100.
plot_breaks	Number of histogram breaks used in progress plotting.
plot_pause	Small pause (in seconds) after each plot refresh to let interactive devices repaint in real time.

Value

An object of class [ph](#).

Examples

```
obj <- iph(ph(structure = "general", dimension = 2), gfun = "weibull", gfun_pars = 2)
data <- sim(obj, n = 100)
fit(obj, data, stepsEM = 100, every = 20)
```

fit_multiple

*Fit Multiple Competing Phase-Type Models***Description**

Runs several competing ph/iph models from user-provided starting values. Progress is updated asynchronously when a model finishes its current EM chunk, so faster models refresh earlier without waiting for slower models.

Usage

```
fit_multiple(
  models,
  y,
  weight = numeric(0),
  rcen = numeric(0),
  rcenweight = numeric(0),
  stepsEM = 1000,
  methods = c("RK", "RK"),
  rkstep = NA,
  uni_epsilon = NA,
  maxit = 100,
  reltol = 1e-08,
  optim_method = "BFGS",
  r = 1,
  burn = 0,
  erlang = NULL,
  merlang_blocks = NULL,
  plot_progress = TRUE,
  plot_every = 100,
  plot_breaks = 50,
  plot_pause = 0.01,
  parallel = TRUE,
  cores = max(1L, parallel::detectCores(logical = FALSE) - 1L),
  colors = NULL
)
```

Arguments

models	List of competing models (objects inheriting from ph).
y	Vector of observations.

weight	Optional observation weights.
rcen	Optional right-censored observations (vector) or interval-censored observations (matrix).
rcenweight	Optional weights for censored observations.
stepsEM	Total EM iterations per model.
methods	Methods for matrix exponential calculation.
rkstep	Runge-Kutta step size.
uni_epsilon	Epsilon parameter for uniformization.
maxit	Maximum iterations in transformation-parameter optimization.
reltol	Relative tolerance in transformation-parameter optimization.
optim_method	Optimization method passed to optim.
r	Sub-sampling proportion.
burn	Burn percentage for histogram midpoint fitting (uncensored only).
erlang	Optional logical flag for exact Erlang updates.
merlang_blocks	Optional Erlang block sizes for exact mixed-Erlang updates.
plot_progress	Logical. If TRUE, a two-panel progress display is updated.
plot_every	Positive integer. EM chunk size per asynchronous worker update.
plot_breaks	Number of histogram breaks for uncensored plotting.
plot_pause	Pause in seconds after each plot refresh.
parallel	Logical. If TRUE, uses asynchronous parallel chunks when available.
cores	Number of worker processes in asynchronous mode.
colors	Optional vector of colors for competing models.

Value

A list with fitted models and trajectories.

Examples

```
set.seed(1)
truth <- ph(
  structure = "merlang",
  block_sizes = c(3, 4),
  probs = c(0.4, 0.6),
  rates = c(0.12, 0.03)
)
y <- sim(truth, n = 400)
cands <- list(
  merlang = ph(structure = "merlang", block_sizes = c(3, 4)),
  gcoxian = ph(structure = "gcoxian", dimension = 7),
  general = ph(structure = "general", dimension = 7)
)
fm <- fit_multiple(cands, y, stepsEM = 40, plot_progress = TRUE, plot_every = 10)
```

haz	<i>New generic for the hazard rate of matrix distributions</i>
-----	--

Description

Methods are available for objects of class [ph](#).

Usage

```
haz(x, ...)
```

Arguments

x	An object of the model class.
...	Further parameters to be passed on.

Value

Hazard rate from the matrix distribution.

haz,ph-method	<i>Hazard rate method for phase-type distributions</i>
---------------	--

Description

Hazard rate method for phase-type distributions

Usage

```
## S4 method for signature 'ph'
haz(x, y)
```

Arguments

x	An object of class ph .
y	A vector of locations.

Value

A vector containing the hazard rate evaluations at the given locations.

Examples

```
obj <- ph(structure = "general")
haz(obj, c(1, 2, 3))
```

inf_norm	<i>L inf norm of a matrix</i>
----------	-------------------------------

Description

Computes the L inf norm of a matrix A, which is defined as: $L_inf(A) = \max(1 \leq i \leq M) \sum(1 \leq j \leq N) \text{abs}(A(i,j))$.

Usage

```
inf_norm(A)
```

Arguments

A	A matrix.
---	-----------

Value

The L inf norm.

inhomogeneity_reg	<i>New generic for regression with matrix distributions, when the inhomogeneity parameter is also regressed</i>
-------------------	---

Description

Methods are available for objects of class [ph](#).

Usage

```
inhomogeneity_reg(x, y, ...)
```

Arguments

x	An object of the model class.
y	A vector of data.
...	Further parameters to be passed on.

Value

An object of the fitted model class.

Regression method for ph Class

Regression method for ph Class

```
## S4 method for signature 'ph'
inhomogeneity_reg(
  x,
  y,
  weight = numeric(0),
  rcen = numeric(0),
  rcenweight = numeric(0),
  X = numeric(0),
  intercept_X = F,
  X2 = NULL,
  intercept_X2 = F,
  prop_f = NULL,
  inhom_f = NULL,
  B0 = numeric(0),
  lasso_X = 0,
  lasso_X2 = 0,
  lasso_X_alpha = 1,
  lasso_X2_alpha = 1,
  stepsEM = 1000,
  methods = c("RK", "UNI"),
  rkstep = NA,
  uni_epsilon = NA,
  optim_method = "BFGS",
  maxit = 50,
  reltol = 1e-08,
  break_tol = 1e-08,
  save_tmp = F,
  assign_lab = "reg_tmp",
  break_n = stepsEM,
  every = 10,
  erlang = NULL,
  merlang_blocks = NULL
)
```

x An object of class `ph`.

y	Vector or data.
weight	Vector of weights.
rcen	Vector of right-censored observations.
rcenweight	Vector of weights for right-censored observations.
X	Model matrix (no intercept needed).
intercept_X	Logical value indicating if an intercept should be included in the model matrix X.
X2	Model matrix for the inhomogeneity parameter (no intercept needed).
intercept_X2	Logical value indicating if an intercept should be included in the model matrix X2.
prop_f	Regression function for the intensity function.
inhom_f	Regression function for the intensity function.
B0	Initial regression coefficients (optional).
lasso_X	Lasso penalty smoothness parameter for coefficients linked with X (optional).
lasso_X2	Lasso penalty smoothness parameter for coefficients linked with X2 (optional).
lasso_X_alpha	Lasso order for coefficients linked with X (optional).
lasso_X2_alpha	Lasso order for coefficients linked with X2 (optional).
stepsEM	Number of EM steps to be performed.
methods	Methods to use for matrix exponential calculation: RM, UNI, or PADE.
rkstep	Runge-Kutta step size (optional).
uni_epsilon	Epsilon parameter for uniformization method.
optim_method	Method to use in gradient optimization.
maxit	Maximum number of iterations when optimizing g function.
reltol	Relative tolerance when optimizing g function.
break_tol	Tolerance to stop the algorithm if the likelihood regresses.
save_tmp	Logical value indicating if the last iteration should be saved.
assign_lab	Label to assign the last iteration to the global environment.
break_n	Number of iterations to wait before stopping the algorithm if the likelihood decreases.
every	Number of iterations between likelihood display updates.
erlang	Logical flag for exact Erlang EM updates with one repeated rate. If NULL, this is auto-detected from the model name.
merlang_blocks	Optional integer vector with Erlang block sizes for exact mixture-of-Erlangs EM updates. If NULL, it is auto-read from object attributes when available.

Value

An object of class [sph](#).

Examples

```
set.seed(1)
obj <- iph(ph(structure = "general", dimension = 2), gfun = "weibull", gfun_pars = 2)
data <- sim(obj, n = 100)
X <- runif(100)
inhomogeneity_reg(x = obj, y = data, X = X, stepsEM = 10)
```

initial_state	<i>Initial state of Markov jump process</i>
---------------	---

Description

Given the accumulated values of the initial probabilities alpha and a uniform value u, it returns the initial state of a Markov jump process. This corresponds to the states satisfying $\text{cum_alpha}_{(k-1)} < u < \text{cum_alpha}_{(k)}$.

Usage

```
initial_state(cum_alpha, u)
```

Arguments

cum_alpha	A cummulated vector of initial probabilities.
u	Random value in (0,1).

Value

Initial state of the Markov jump process.

iph	<i>Constructor function for inhomogeneous phase-type distributions</i>
-----	--

Description

Constructor function for inhomogeneous phase-type distributions

Usage

```
iph(
  ph = NULL,
  gfun = NULL,
  gfun_pars = NULL,
  alpha = NULL,
  S = NULL,
  structure = NULL,
  dimension = 3,
  scale = 1
)
```

Arguments

ph	An object of class ph .
gfun	Inhomogeneity transform.
gfun_pars	The parameters of the inhomogeneity function.
alpha	A probability vector.
S	A sub-intensity matrix.
structure	A valid ph structure.
dimension	The dimension of the ph structure (if provided).
scale	Scale.

Value

An object of class [iph](#).

Examples

```
iph(ph(structure = "coxian", dimension = 4), gfun = "pareto", gfun_pars = 3)
```

 iph-class

Inhomogeneous phase-type distributions

Description

Class of objects for inhomogeneous phase-type distributions.

Value

Class object.

Slots

name Name of the phase-type distribution.
 gfun A list comprising of the parameters.
 scale Scale.

laplace	<i>New generic for Laplace transform of matrix distributions</i>
---------	--

Description

Methods are available for objects of class [ph](#).

Usage

```
laplace(x, ...)
```

Arguments

x	An object of the model class.
...	Further parameters to be passed on.

Value

Laplace transform of the matrix distribution.

laplace, bivph-method	<i>Laplace method for bivph class</i>
-----------------------	---------------------------------------

Description

Laplace method for bivph class

Usage

```
## S4 method for signature 'bivph'
laplace(x, r)
```

Arguments

x	An object of class mph .
r	A matrix of real values.

Value

A vector containing the corresponding Laplace transform evaluations.

Examples

```
obj <- bivph(dimensions = c(3, 3))
laplace(obj, matrix(c(0.5, 1), ncol = 2))
```

laplace,mph-method	<i>Laplace method for multivariate phase-type distributions</i>
--------------------	---

Description

Laplace method for multivariate phase-type distributions

Usage

```
## S4 method for signature 'mph'  
laplace(x, r)
```

Arguments

x	An object of class mph .
r	A matrix of real values.

Value

A vector containing the corresponding Laplace transform evaluations.

Examples

```
set.seed(123)  
obj <- mph(structure = c("general", "general"))  
laplace(obj, matrix(c(0.5, 1), ncol = 2))
```

laplace,ph-method	<i>Laplace method for phase-type distributions</i>
-------------------	--

Description

Laplace method for phase-type distributions

Usage

```
## S4 method for signature 'ph'  
laplace(x, r)
```

Arguments

x	An object of class ph .
r	A vector of real values.

Value

The Laplace transform of the [ph](#) (or underlying [ph](#)) object at the given locations.

Examples

```
set.seed(123)
obj <- ph(structure = "general", dimension = 3)
laplace(obj, 3)
```

linCom	<i>New generic for linear combinations of multivariate matrix distributions</i>
--------	---

Description

Methods are available for objects of multivariate classes.

Usage

```
linCom(x, ...)
```

Arguments

x	An object of the model class.
...	Further parameters to be passed on.

Value

Marginal of the matrix distribution.

linCom, bivph-method	<i>Linear combination method for bivariate phase-type distributions</i>
----------------------	---

Description

Linear combination method for bivariate phase-type distributions

Usage

```
## S4 method for signature 'bivph'
linCom(x, w = c(1, 1))
```

Arguments

x	An object of class bivph .
w	A vector with non-negative entries.

Value

An object of class `ph`.

Examples

```
obj <- bivph(dimensions = c(3, 3))
linCom(obj, c(1, 0))
```

linCom, MPHstar-method *Linear combination method for MPHstar class*

Description

Linear combination method for MPHstar class

Usage

```
## S4 method for signature 'MPHstar'
linCom(x, w)
```

Arguments

<code>x</code>	An object of class <code>MPHstar</code> .
<code>w</code>	A vector with non-negative entries.

Value

An object of class `ph`.

Examples

```
obj <- MPHstar(structure = "general")
linCom(obj, c(1, 0))
```

linear_combination *Computes PH parameters of a linear combination of vector from MPHstar*

Description

Computes PH parameters of a linear combination of vector from MPHstar

Usage

```
linear_combination(w, alpha, S, R)
```

Arguments

w	Vector with weights.
alpha	Initial distribution vector.
S	Sub-intensity matrix.
R	Reward matrix.

Value

A list of PH parameters.

logLik,ph-method	<i>Loglikelihood method for ph class</i>
------------------	--

Description

Loglikelihood method for ph class

Usage

```
## S4 method for signature 'ph'
logLik(object)
```

Arguments

object An object of class [ph](#).

Value

An object of class logLik.

Examples

```
obj <- iph(ph(structure = "general", dimension = 2), gfun = "weibull", gfun_pars = 2)
data <- sim(obj, n = 100)
fitted_ph <- fit(obj, data, stepsEM = 10)
logLik(fitted_ph)
```

logLikelihood_UNIs_PI *Loglikelihood of PI using uniformization, with customizable observation transformation, automatically handling right-censored or interval-censored data based on the structure of rcens.*

Description

Loglikelihood for a sample.

Usage

```
logLikelihood_UNIs_PI(  
  h,  
  alpha,  
  S,  
  beta1,  
  beta2,  
  obs,  
  weight,  
  rcens,  
  rcweight,  
  scale1,  
  scale2,  
  gfun_name  
)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta1	Parameter of transformation (uncensored observations).
beta2	Parameter of transformation (censored observations).
obs	The observations.
weight	Weights of the observations.
rcens	Censored observations (either numeric vector for right censoring or matrix for interval censoring).
rcweight	Weights of the censored observations.
scale1	Scale for observations.
scale2	Scale for censored observations.
gfun_name	Name of the transformation function to be applied to observations (e.g., "scale", "log", "sqrt").

Value

The loglikelihood of censored the sample.

logLikelihoodbivDPH	<i>Loglikelihood for bivariate discrete phase-type</i>
---------------------	--

Description

Loglikelihood for bivariate discrete phase-type

Usage

```
logLikelihoodbivDPH(alpha, S11, S12, S22, obs, weight)
```

Arguments

alpha	Initial probabilities.
S11	Sub-transition matrix.
S12	Matrix.
S22	Sub-transition matrix.
obs	The observations.
weight	The weights of the observations.

logLikelihoodbivDPH_MoE	<i>Loglikelihood for bivariate discrete phase-type MoE</i>
-------------------------	--

Description

Loglikelihood for bivariate discrete phase-type MoE

Usage

```
logLikelihoodbivDPH_MoE(alpha, S11, S12, S22, obs, weight)
```

Arguments

alpha	Initial probabilities.
S11	Sub-transition matrix.
S12	Matrix.
S22	Sub-transition matrix.
obs	The observations.
weight	The weights of the observations.

logLikelihoodbivPH	<i>Loglikelihood for Bivariate PH</i>
--------------------	---------------------------------------

Description

Loglikelihood for Bivariate PH

Usage

```
logLikelihoodbivPH(alpha, S11, S12, S22, obs, weight)
```

Arguments

alpha	Vector of initial probabilities.
S11	Sub-intensity matrix.
S12	Matrix.
S22	Sub-intensity matrix.
obs	The observations.
weight	The weights of the observations.

logLikelihoodDPH	<i>Loglikelihood for discrete phase-type</i>
------------------	--

Description

Loglikelihood for discrete phase-type

Usage

```
logLikelihoodDPH(alpha, S, obs, weight)
```

Arguments

alpha	Initial probabilities.
S	Sub-transition matrix.
obs	The observations.
weight	The weights of the observations.

logLikelihoodDPH_MoE	<i>Loglikelihood for discrete phase-type MoE</i>
----------------------	--

Description

Loglikelihood for discrete phase-type MoE

Usage

```
logLikelihoodDPH_MoE(alpha, S, obs, weight)
```

Arguments

alpha	Initial probabilities.
S	Sub-transition matrix.
obs	The observations.
weight	The weights of the observations.

logLikelihoodmDPH	<i>Loglikelihood for multivariate discrete phase-type</i>
-------------------	---

Description

Loglikelihood for multivariate discrete phase-type

Usage

```
logLikelihoodmDPH(alpha, S_list, obs, weight)
```

Arguments

alpha	Initial probabilities.
S_list	List of marginal sub-transition matrices.
obs	The observations.
weight	The weights of the observations.

logLikelihoodmDPH_MoE *Loglikelihood for multivariate discrete phase-type MoE*

Description

Loglikelihood for multivariate discrete phase-type MoE

Usage

```
logLikelihoodmDPH_MoE(alpha, S_list, obs, weight)
```

Arguments

alpha	Initial probabilities.
S_list	List of marginal sub-transition matrices.
obs	The observations.
weight	The weights of the observations.

logLikelihoodMgev_PADE

Loglikelihood of matrix-GEV using Pade

Description

Loglikelihood for a sample

Usage

```
logLikelihoodMgev_PADE(h, alpha, S, beta, obs, weight, rcens, rcweight)
```

Arguments

h	Nuisance parameter.
alpha	Initial probabilities.
S	sub-intensity matrix.
beta	Inhomogeneity parameter.
obs	The observations.
weight	The weights of the observations.
rcens	Censored observations.
rcweight	The weights of the censored observations.

logLikelihoodMgev_RK *Loglikelihood of matrix-GEV using Runge-Kutta*

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMgev_RK(h, alpha, S, beta, obs, weight, rcens, rcweight)
```

Arguments

h	Step-length.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation
obs	The observations.
weight	Weights of the observations.
rcens	Censored observations.
rcweight	Weights of the censored observations.

logLikelihoodMgev_UNI *Loglikelihood of matrix-GEV using uniformization*

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMgev_UNI(h, alpha, S, beta, obs, weight, rcens, rcweight)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	censored observations.
rcweight	Weights of the censored observations.

```
logLikelihoodMgev_UNI_inhom
```

Loglikelihood of matrix-GEV using uniformization, when the intensity function is regressed on covariate informatino

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMgev_UNI_inhom(h, alpha, S, beta, obs, weight, rcens, rcweight)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	censored observations.
rcweight	Weights of the censored observations.

```
logLikelihoodMgev_UNI_inhom_intCens
```

Loglikelihood of matrix-GEV using uniformization, when the intensity function is regressed on covariate informatino

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMgev_UNI_inhom_intCens(
  h,
  alpha,
  S,
  beta,
  obs,
  weight,
  rcens,
  rcweight
)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	Interval censored observations.
rcweight	Weights of the censored observations.

logLikelihoodMgev_UNI_intervalCensoring
Loglikelihood of matrix-GEV using uniformization

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMgev_UNI_intervalCensoring(
  h,
  alpha,
  S,
  beta,
  obs,
  weight,
  rcens,
  rcweight
)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	Interval censored observations.
rcweight	Weights of the censored observations.

logLikelihoodMgompertz_PADE

Loglikelihood of matrix-Gompertz using Pade

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMgompertz_PADE(h, alpha, S, beta, obs, weight, rcens, rcweight)
```

Arguments

h	Nuisance parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Inhomogeneity parameter.
obs	The observations.
weight	The weights of the observations.
rcens	Censored observations.
rcweight	The weights of the censored observations.

logLikelihoodMgompertz_PADEs

Loglikelihood of PI with matrix-Gompertz using Pade

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMgompertz_PADEs(
  h,
  alpha,
  S,
  beta,
  obs,
  weight,
  rcens,
  rcweight,
  scale1,
  scale2
)
```

Arguments

h	Nuisance parameter.
alpha	Initial probabilities.
S	Sub-intensity.
beta	Inhomogeneity parameter.
obs	The observations.
weight	Weights of the observations.
rcens	Censored observations.
rcweight	Weights of the censored observations.
scale1	Scale for observations.
scale2	Scale for censored observations.

logLikelihoodMgompertz_RK

Loglikelihood of matrix-Gompertz using Runge-Kutta

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMgompertz_RK(h, alpha, S, beta, obs, weight, rcens, rcweight)
```

Arguments

h	Step-length.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	Censored observations.
rcweight	Weights of the censored observations.

`logLikelihoodMgompertz_RKs`*Loglikelihood of PI with matrix-Gompertz using Runge-Kutta*

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMgompertz_RKs(  
  h,  
  alpha,  
  S,  
  beta,  
  obs,  
  weight,  
  rcens,  
  rcweight,  
  scale1,  
  scale2  
)
```

Arguments

<code>h</code>	Step-length.
<code>alpha</code>	Initial probabilities.
<code>S</code>	Sub-intensity matrix.
<code>beta</code>	Parameter of transformation.
<code>obs</code>	The observations.
<code>weight</code>	Weights of the observations.
<code>rcens</code>	Censored observations.
<code>rcweight</code>	Weights of the censored observations.
<code>scale1</code>	Scale for observations.
<code>scale2</code>	Scale for censored observations.

logLikelihoodMgompertz_UNI

Loglikelihood of matrix-Gompertz using uniformization

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMgompertz_UNI(h, alpha, S, beta, obs, weight, rcens, rcweight)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	censored observations.
rcweight	Weights of the censored observations.

logLikelihoodMgompertz_UNI_inhom

Loglikelihood of matrix-Gompertz using uniformization, when the intensity function is regressed on covariate informatino

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMgompertz_UNI_inhom(
  h,
  alpha,
  S,
  beta,
  obs,
  weight,
  rcens,
  rcweight
)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	censored observations.
rcweight	Weights of the censored observations.

logLikelihoodMgompertz_UNI_inhom_intCens

Loglikelihood of matrix-Gompertz using uniformization, when the intensity function is regressed on covariate informatino

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMgompertz_UNI_inhom_intCens(
  h,
  alpha,
  S,
  beta,
  obs,
  weight,
  rcens,
  rcweight
)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	Interval censored observations.
rcweight	Weights of the censored observations.

logLikelihoodMgompertz_UNI_intervalCensoring
<i>Loglikelihood of matrix-Gompertz using uniformization</i>

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMgompertz_UNI_intervalCensoring(  
  h,  
  alpha,  
  S,  
  beta,  
  obs,  
  weight,  
  rcens,  
  rcweight  
)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	Interval censored observations.
rcweight	Weights of the censored observations.

logLikelihoodMgompertz_UNIs
<i>Loglikelihood of PI with matrix-Gompertz using Uniformization</i>

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMgompertz_UNIs(
  h,
  alpha,
  S,
  beta,
  obs,
  weight,
  rcens,
  rcweight,
  scale1,
  scale2
)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	Censored observations.
rcweight	Weights of the censored observations.
scale1	Scale for observations.
scale2	Scale for censored observations.

```
logLikelihoodMgompertz_UNIs_inhom
```

Loglikelihood of PI with matrix-Gompertz using uniformization, when the intensity function is regressed on covariate informatino

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMgompertz_UNIs_inhom(
  h,
  alpha,
  S,
  beta,
  obs,
```

```

    weight,
    rcens,
    rcweight,
    scale1,
    scale2
  )

```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	Censored observations.
rcweight	Weights of the censored observations.
scale1	Scale for observations.
scale2	Scale for censored observations.

logLikelihoodMgompertz_UNIs_inhom_intCens

Loglikelihood of PI with matrix-Gompertz using uniformization, when the intensity function is regressed on covariate informatino

Description

Loglikelihood for a sample.

Usage

```

logLikelihoodMgompertz_UNIs_inhom_intCens(
  h,
  alpha,
  S,
  beta,
  obs,
  weight,
  rcens,
  rcweight,
  scale1,
  scale2
)

```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	Interval censored observations.
rcweight	Weights of the censored observations.
scale1	Scale for observations.
scale2	Scale for censored observations.

logLikelihoodMgompertz_UNIs_intervalCensoring

Loglikelihood of PI with matrix-Gompertz using Uniformization

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMgompertz_UNIs_intervalCensoring(
  h,
  alpha,
  S,
  beta,
  obs,
  weight,
  rcens,
  rcweight,
  scale1,
  scale2
)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.

weight	Weights of the observations.
rcens	Interval censored observations.
rcweight	Weights of the censored observations.
scale1	Scale for observations.
scale2	Scale for censored observations.

logLikelihoodMloglogistic_PADE

Loglikelihood of matrix-loglogistic using Pade

Description

Loglikelihood for a sample.

Usage

logLikelihoodMloglogistic_PADE(h, alpha, S, beta, obs, weight, rcens, rcweight)

Arguments

h	Nuisance parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Inhomogeneity parameter.
obs	The observations.
weight	The weights of the observations.
rcens	Censored observations.
rcweight	The weights of the censored observations.

logLikelihoodMloglogistic_PADEs

Loglikelihood of PI with matrix-loglogistic using Pade

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMloglogistic_PADEs(
  h,
  alpha,
  S,
  beta,
  obs,
  weight,
  rcens,
  rcweight,
  scale1,
  scale2
)
```

Arguments

h	Nuisance parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Inhomogeneity parameter.
obs	The observations.
weight	Weights of the observations.
rcens	Censored observations.
rcweight	Weights of the censored observations.
scale1	Scale for observations.
scale2	Scale for censored observations.

logLikelihoodMloglogistic_RK

Loglikelihood of matrix-loglogistic using Runge-Kutta

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMloglogistic_RK(h, alpha, S, beta, obs, weight, rcens, rcweight)
```

Arguments

h	Step-length.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameters of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	Censored observations.
rcweight	Weights of the censored observations.

logLikelihoodMloglogistic_RKs

Loglikelihood of PI with matrix-loglogistic using Runge-Kutta

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMloglogistic_RKs(
  h,
  alpha,
  S,
  beta,
  obs,
  weight,
  rcens,
  rcweight,
  scale1,
  scale2
)
```

Arguments

h	Step-length.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameters of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	Censored observations.
rcweight	Weights of the censored observations.
scale1	Scale for observations.
scale2	Scale for censored observations.

logLikelihoodMloglogistic_UNI

Loglikelihood of matrix-loglogistic using uniformization

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMloglogistic_UNI(h, alpha, S, beta, obs, weight, rcens, rcweight)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	censored observations.
rcweight	Weights of the censored observations.

logLikelihoodMloglogistic_UNI_inhom

Loglikelihood of matrix-loglogistic using uniformization, when the intensity function is regressed on covariate informatino

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMloglogistic_UNI_inhom(
  h,
  alpha,
  S,
  beta,
  obs,
  weight,
  rcens,
  rcweight
)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	censored observations.
rcweight	Weights of the censored observations.

logLikelihoodMloglogistic_UNI_inhom_intCens

Loglikelihood of matrix-loglogistic using uniformization, when the intensity function is regressed on covariate informatino

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMloglogistic_UNI_inhom_intCens(
  h,
  alpha,
  S,
  beta,
  obs,
  weight,
  rcens,
  rcweight
)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	Interval censored observations.
rcweight	Weights of the censored observations.

logLikelihoodMloglogistic_UNI_intervalCensoring
<i>Loglikelihood of matrix-loglogistic using uniformization</i>

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMloglogistic_UNI_intervalCensoring(  
  h,  
  alpha,  
  S,  
  beta,  
  obs,  
  weight,  
  rcens,  
  rcweight  
)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	Interval censored observations.
rcweight	Weights of the censored observations.

logLikelihoodMloglogistic_UNIs
<i>Loglikelihood of PI with matrix-loglogistic using uniformization</i>

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMloglogistic_UNIs(
  h,
  alpha,
  S,
  beta,
  obs,
  weight,
  rcens,
  rcweight,
  scale1,
  scale2
)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	Censored observations.
rcweight	Weights of the censored observations.
scale1	Scale for observations.
scale2	Scale for censored observations.

```
logLikelihoodMloglogistic_UNIs_inhom
```

Loglikelihood of PI with matrix-loglogistic using uniformization, when the intensity function is regressed on covariate informatino

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMloglogistic_UNIs_inhom(
  h,
  alpha,
  S,
  beta,
  obs,
```

```

    weight,
    rcens,
    rcweight,
    scale1,
    scale2
  )

```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	Censored observations.
rcweight	Weights of the censored observations.
scale1	Scale for observations.
scale2	Scale for censored observations.

```
logLikelihoodMloglogistic_UNIs_inhom_intCens
```

Loglikelihood of PI with matrix-loglogistic using uniformization, when the intensity function is regressed on covariate informatino

Description

Loglikelihood for a sample.

Usage

```

logLikelihoodMloglogistic_UNIs_inhom_intCens(
  h,
  alpha,
  S,
  beta,
  obs,
  weight,
  rcens,
  rcweight,
  scale1,
  scale2
)

```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	Interval censored observations.
rcweight	Weights of the censored observations.
scale1	Scale for observations.
scale2	Scale for censored observations.

logLikelihoodMloglogistic_UNIs_intervalCensoring

Loglikelihood of PI with matrix-loglogistic using uniformization

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMloglogistic_UNIs_intervalCensoring(
  h,
  alpha,
  S,
  beta,
  obs,
  weight,
  rcens,
  rcweight,
  scale1,
  scale2
)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.

weight	Weights of the observations.
rcens	Interval censored observations.
rcweight	Weights of the censored observations.
scale1	Scale for observations.
scale2	Scale for censored observations.

logLikelihoodMlognormal_PADE

Loglikelihood of matrix-lognormal using Pade

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMlognormal_PADE(h, alpha, S, beta, obs, weight, rcens, rcweight)
```

Arguments

h	Nuisance parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Inhomogeneity parameter.
obs	The observations.
weight	The weights of the observations.
rcens	Censored observations.
rcweight	The weights of the censored observations.

logLikelihoodMlognormal_PADEs

Loglikelihood of PI with matrix-lognormal using Pade

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMlognormal_PADEs(
  h,
  alpha,
  S,
  beta,
  obs,
  weight,
  rcens,
  rcweight,
  scale1,
  scale2
)
```

Arguments

h	Nuisance parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Inhomogeneity parameter.
obs	The observations.
weight	Weights of the observations.
rcens	Censored observations.
rcweight	Weights of the censored observations.
scale1	Scale for observations.
scale2	Scale for censored observations.

logLikelihoodMlognormal_RK

Loglikelihood of matrix-lognormal using Runge-Kutta

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMlognormal_RK(h, alpha, S, beta, obs, weight, rcens, rcweight)
```

Arguments

h	Step-length.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	Censored observations.
rcweight	Weights of the censored observations.

logLikelihoodMlognormal_RKs

Loglikelihood of PI matrix-lognormal using Runge-Kutta

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMlognormal_RKs(
  h,
  alpha,
  S,
  beta,
  obs,
  weight,
  rcens,
  rcweight,
  scale1,
  scale2
)
```

Arguments

h	Step-length.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	Censored observations.
rcweight	Weights of the censored observations.
scale1	Scale for observations.
scale2	Scale for censored observations.

logLikelihoodMlognormal_UNI

Loglikelihood of matrix-lognormal using uniformization

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMlognormal_UNI(h, alpha, S, beta, obs, weight, rcens, rcweight)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	censored observations.
rcweight	Weights of the censored observations.

logLikelihoodMlognormal_UNI_inhom

Loglikelihood of matrix-lognormal using uniformization, when the intensity function is regressed on covariate informatino

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMlognormal_UNI_inhom(
  h,
  alpha,
  S,
  beta,
  obs,
  weight,
  rcens,
  rcweight
)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	censored observations.
rcweight	Weights of the censored observations.

logLikelihoodMlognormal_UNI_inhom_intCens

Loglikelihood of matrix-lognormal using uniformization, when the intensity function is regressed on covariate informatino

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMlognormal_UNI_inhom_intCens(
  h,
  alpha,
  S,
  beta,
  obs,
  weight,
  rcens,
  rcweight
)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	Interval censored observations.
rcweight	Weights of the censored observations.

logLikelihoodMlognormal_UNI_intervalCensoring
<i>Loglikelihood of matrix-lognormal using uniformization</i>

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMlognormal_UNI_intervalCensoring(  
  h,  
  alpha,  
  S,  
  beta,  
  obs,  
  weight,  
  rcens,  
  rcweight  
)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	Interval censored observations.
rcweight	Weights of the censored observations.

logLikelihoodMlognormal_UNIs
<i>Loglikelihood of PI with matrix-lognormal using uniformization</i>

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMlognormal_UNIs(
  h,
  alpha,
  S,
  beta,
  obs,
  weight,
  rcens,
  rcweight,
  scale1,
  scale2
)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	Censored observations.
rcweight	Weights of the censored observations.
scale1	Scale for observations.
scale2	Scale for censored observations.

logLikelihoodMlognormal_UNIs_inhom

Loglikelihood of PI with matrix-lognormal using uniformization, when the intensity function is regressed on covariate informatino

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMlognormal_UNIs_inhom(
  h,
  alpha,
  S,
  beta,
  obs,
```

```

    weight,
    rcens,
    rcweight,
    scale1,
    scale2
  )

```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	Censored observations.
rcweight	Weights of the censored observations.
scale1	Scale for observations.
scale2	Scale for censored observations.

logLikelihoodMlognormal_UNIs_inhom_intCens

Loglikelihood of PI with matrix-lognormal using uniformization, when the intensity function is regressed on covariate informatino

Description

Loglikelihood for a sample.

Usage

```

logLikelihoodMlognormal_UNIs_inhom_intCens(
  h,
  alpha,
  S,
  beta,
  obs,
  weight,
  rcens,
  rcweight,
  scale1,
  scale2
)

```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	Interval censored observations.
rcweight	Weights of the censored observations.
scale1	Scale for observations.
scale2	Scale for censored observations.

logLikelihoodMlognormal_UNIs_intervalCensoring

Loglikelihood of PI with matrix-lognormal using uniformization

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMlognormal_UNIs_intervalCensoring(
  h,
  alpha,
  S,
  beta,
  obs,
  weight,
  rcens,
  rcweight,
  scale1,
  scale2
)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.

weight	Weights of the observations.
rcens	Interval censored observations.
rcweight	Weights of the censored observations.
scale1	Scale for observations.
scale2	Scale for censored observations.

logLikelihoodMpareto_PADE
<i>Loglikelihood of matrix-Pareto using Pade</i>

Description

Loglikelihood for a sample.

Usage

logLikelihoodMpareto_PADE(h, alpha, S, beta, obs, weight, rcens, rcweight)

Arguments

h	Nuisance parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Inhomogeneity parameter.
obs	The observations.
weight	The weights of the observations.
rcens	Censored observations.
rcweight	The weights of the censored observations.

logLikelihoodMpareto_PADEs
<i>Loglikelihood of PI with matrix-Pareto using Pade</i>

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMpareto_PADEs(
  h,
  alpha,
  S,
  beta,
  obs,
  weight,
  rcens,
  rcweight,
  scale1,
  scale2
)
```

Arguments

h	Nuisance parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Inhomogeneity parameter.
obs	The observations.
weight	Weights of the observations.
rcens	Censored observations.
rcweight	Weights of the censored observations.
scale1	Scale for observations.
scale2	Scale for censored observations.

logLikelihoodMpareto_RK

Loglikelihood of matrix-Pareto using Runge-Kutta

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMpareto_RK(h, alpha, S, beta, obs, weight, rcens, rcweight)
```

Arguments

h	Step-length.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	Censored observations.
rcweight	Weights of the censored observations.

logLikelihoodMpareto_RKs

Loglikelihood of PI with matrix-Pareto using Runge-Kutta

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMpareto_RKs(
  h,
  alpha,
  S,
  beta,
  obs,
  weight,
  rcens,
  rcweight,
  scale1,
  scale2
)
```

Arguments

h	Step-length.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	Censored observations.
rcweight	Weights of the censored observations.
scale1	Scale for observations.
scale2	Scale for censored observations.

logLikelihoodMpareto_UNI

Loglikelihood of matrix-Pareto using uniformization

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMpareto_UNI(h, alpha, S, beta, obs, weight, rcens, rcweight)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	censored observations.
rcweight	Weights of the censored observations.

logLikelihoodMpareto_UNI_inhom

Loglikelihood of matrix-Pareto using uniformization, when the intensity function is regressed on covariate informatino

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMpareto_UNI_inhom(h, alpha, S, beta, obs, weight, rcens, rcweight)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	censored observations.
rcweight	Weights of the censored observations.

logLikelihoodMpareto_UNI_inhom_intCens

Loglikelihood of matrix-Pareto using uniformization, when the intensity function is regressed on covariate informatino

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMpareto_UNI_inhom_intCens(
  h,
  alpha,
  S,
  beta,
  obs,
  weight,
  rcens,
  rcweight
)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	Interval censored observations.
rcweight	Weights of the censored observations.

logLikelihoodMpareto_UNI_intervalCensoring
<i>Loglikelihood of matrix-Pareto using uniformization</i>

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMpareto_UNI_intervalCensoring(  
  h,  
  alpha,  
  S,  
  beta,  
  obs,  
  weight,  
  rcens,  
  rcweight  
)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	Interval censored observations.
rcweight	Weights of the censored observations.

logLikelihoodMpareto_UNIs
<i>Loglikelihood of PI with matrix-Pareto using uniformization</i>

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMpareto_UNIs(
  h,
  alpha,
  S,
  beta,
  obs,
  weight,
  rcens,
  rcweight,
  scale1,
  scale2
)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	Censored observations.
rcweight	Weights of the censored observations.
scale1	Scale for observations.
scale2	Scale for censored observations.

```
logLikelihoodMpareto_UNIs_inhom
```

Loglikelihood of PI with matrix-Pareto using uniformization, when the intensity function is regressed on covariate informatino

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMpareto_UNIs_inhom(
  h,
  alpha,
  S,
  beta,
  obs,
```

```

    weight,
    rcens,
    rcweight,
    scale1,
    scale2
  )

```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	Censored observations.
rcweight	Weights of the censored observations.
scale1	Scale for observations.
scale2	Scale for censored observations.

logLikelihoodMpareto_UNIs_inhom_intCens

Loglikelihood of PI with matrix-Pareto using uniformization, when the intensity function is regressed on covariate informatino

Description

Loglikelihood for a sample.

Usage

```

logLikelihoodMpareto_UNIs_inhom_intCens(
  h,
  alpha,
  S,
  beta,
  obs,
  weight,
  rcens,
  rcweight,
  scale1,
  scale2
)

```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	Interval censored observations.
rcweight	Weights of the censored observations.
scale1	Scale for observations.
scale2	Scale for censored observations.

logLikelihoodMpareto_UNIs_intervalCensoring

Loglikelihood of PI with matrix-Pareto using uniformization

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMpareto_UNIs_intervalCensoring(
  h,
  alpha,
  S,
  beta,
  obs,
  weight,
  rcens,
  rcweight,
  scale1,
  scale2
)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.

weight	Weights of the observations.
rcens	Interval censored observations.
rcweight	Weights of the censored observations.
scale1	Scale for observations.
scale2	Scale for censored observations.

logLikelihoodMweibull_PADE

Loglikelihood of matrix-Weibull using Pade

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMweibull_PADE(h, alpha, S, beta, obs, weight, rcens, rcweight)
```

Arguments

h	Nuisance parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Inhomogeneity parameter.
obs	The observations.
weight	The weights of the observations.
rcens	Censored observations.
rcweight	The weights of the censored observations.

logLikelihoodMweibull_PADEs

Loglikelihood of PI with matrix-Weibull using Pade

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMweibull_PADEs(  
  h,  
  alpha,  
  S,  
  beta,  
  obs,  
  weight,  
  rcens,  
  rcweight,  
  scale1,  
  scale2  
)
```

Arguments

h	Nuisance parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Inhomogeneity parameter.
obs	The observations.
weight	The weights of the observations.
rcens	Censored observations.
rcweight	The weights of the censored observations.
scale1	Scale for observations.
scale2	Scale for censored observations.

logLikelihoodMweibull_RK
<i>Loglikelihood of matrix-Weibull using Runge-Kutta</i>

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMweibull_RK(h, alpha, S, beta, obs, weight, rcens, rcweight)
```

Arguments

h	Step-length.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	Censored observations.
rcweight	Weights of the censored observations.

logLikelihoodMweibull_RKs

Loglikelihood of PI with matrix-Weibull using Runge-Kutta

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMweibull_RKs(
  h,
  alpha,
  S,
  beta,
  obs,
  weight,
  rcens,
  rcweight,
  scale1,
  scale2
)
```

Arguments

h	Step-length.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	Censored observations.
rcweight	Weights of the censored observations.
scale1	Scale for observations.
scale2	Scale for censored observations.

logLikelihoodMweibull_UNI
<i>Loglikelihood of matrix-Weibull using uniformization</i>

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMweibull_UNI(h, alpha, S, beta, obs, weight, rcens, rcweight)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	censored observations.
rcweight	Weights of the censored observations.

logLikelihoodMweibull_UNI_inhom
<i>Loglikelihood of matrix-Weibull using uniformization, when the intensity function is regressed on covariate informatino</i>

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMweibull_UNI_inhom(  
  h,  
  alpha,  
  S,  
  beta,  
  obs,  
  weight,  
  rcens,  
  rcweight  
)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	censored observations.
rcweight	Weights of the censored observations.

logLikelihoodMweibull_UNI_inhom_intCens

Loglikelihood of matrix-Weibull using uniformization, when the intensity function is regressed on covariate informatino

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMweibull_UNI_inhom_intCens(
  h,
  alpha,
  S,
  beta,
  obs,
  weight,
  rcens,
  rcweight
)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	Interval censored observations.
rcweight	Weights of the censored observations.

logLikelihoodMweibull_UNI_intervalCensoring
<i>Loglikelihood of matrix-Weibull using uniformization</i>

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMweibull_UNI_intervalCensoring(  
  h,  
  alpha,  
  S,  
  beta,  
  obs,  
  weight,  
  rcens,  
  rcweight  
)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	Interval censored observations.
rcweight	Weights of the censored observations.

logLikelihoodMweibull_UNIs
<i>Loglikelihood of PI with matrix-Weibull using uniformization</i>

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMweibull_UNIs(
  h,
  alpha,
  S,
  beta,
  obs,
  weight,
  rcens,
  rcweight,
  scale1,
  scale2
)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	Censored observations.
rcweight	Weights of the censored observations.
scale1	Scale for observations.
scale2	Scale for censored observations.

```
logLikelihoodMweibull_UNIs_inhom
```

Loglikelihood of PI with matrix-Weibull using uniformization, when the intensity function is regressed on covariate informatino

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMweibull_UNIs_inhom(
  h,
  alpha,
  S,
  beta,
  obs,
```

```

    weight,
    rcens,
    rcweight,
    scale1,
    scale2
  )

```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	Censored observations.
rcweight	Weights of the censored observations.
scale1	Scale for observations.
scale2	Scale for censored observations.

logLikelihoodMweibull_UNIs_inhom_intCens

Loglikelihood of PI with matrix-Weibull using uniformization, when the intensity function is regressed on covariate informatino

Description

Loglikelihood for a sample.

Usage

```

logLikelihoodMweibull_UNIs_inhom_intCens(
  h,
  alpha,
  S,
  beta,
  obs,
  weight,
  rcens,
  rcweight,
  scale1,
  scale2
)

```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.
weight	Weights of the observations.
rcens	Interval censored observations.
rcweight	Weights of the censored observations.
scale1	Scale for observations.
scale2	Scale for censored observations.

logLikelihoodMweibull_UNIs_intervalCensoring

Loglikelihood of PI with matrix-Weibull using uniformization

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodMweibull_UNIs_intervalCensoring(
  h,
  alpha,
  S,
  beta,
  obs,
  weight,
  rcens,
  rcweight,
  scale1,
  scale2
)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Parameter of transformation.
obs	The observations.

weight	Weights of the observations.
rcens	Interval censored observations.
rcweight	Weights of the censored observations.
scale1	Scale for observations.
scale2	Scale for censored observations.

logLikelihoodPH_MoE	<i>Loglikelihood for PH-MoE</i>
---------------------	---------------------------------

Description

Loglikelihood for PH-MoE

Usage

logLikelihoodPH_MoE(alpha1, alpha2, S, obs, weight, rcens, rcweight)

Arguments

alpha1	Initial probabilities for non-censored data.
alpha2	Initial probabilities for censored data.
S	Sub-intensity matrix.
obs	The observations.
weight	The weights of the observations.
rcens	Censored observations.
rcweight	The weights of the censored observations.

logLikelihoodPH_PADE	<i>Loglikelihood of phase-type using Pade approximation</i>
----------------------	---

Description

Loglikelihood for a sample.

Usage

logLikelihoodPH_PADE(h, alpha, S, obs, weight, rcens, rcweight)

Arguments

h	Nuisance parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
obs	The observations.
weight	The weights of the observations.
rcens	Censored observations.
rcweight	The weights of the censored observations.

logLikelihoodPH_PADEs *Loglikelihood of PI with phase-type using Pade*

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodPH_PADEs(
  h,
  alpha,
  S,
  obs,
  weight,
  rcens,
  rcweight,
  scale1,
  scale2
)
```

Arguments

h	Nuisance parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
obs	The observations.
weight	The weights of the observations.
rcens	Censored observations.
rcweight	The weights of the censored observations.
scale1	Scale for observations.
scale2	Scale for censored observations.

logLikelihoodPH_RK	<i>Loglikelihood of phase-type using Runge-Kutta</i>
--------------------	--

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodPH_RK(h, alpha, S, obs, weight, rcens, rcweight)
```

Arguments

h	Step-length.
alpha	Initial probabilities.
S	Sub-intensity matrix.
obs	The observations.
weight	Weights of the observations.
rcens	Censored observations.
rcweight	Weights of the censored observations.

logLikelihoodPH_RKs	<i>Loglikelihood of PI with phase-type using Runge-Kutta</i>
---------------------	--

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodPH_RKs(h, alpha, S, obs, weight, rcens, rcweight, scale1, scale2)
```

Arguments

h	Step-length.
alpha	Initial probabilities.
S	Sub-intensity matrix.
obs	The observations.
weight	Weights of the observations.
rcens	Censored observations.
rcweight	Weights of the censored observations.
scale1	Scale for observations.
scale2	Scale for censored observations.

logLikelihoodPH_UNI	<i>Loglikelihood of phase-type using uniformization</i>
---------------------	---

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodPH_UNI(h, alpha, S, obs, weight, rcens, rcweight)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
obs	The observations.
weight	Weights of the observations.
rcens	Censored observations.
rcweight	Weights of the censored observations.

logLikelihoodPH_UNI_intervalCensoring	<i>Loglikelihood of phase-type using uniformization</i>
---------------------------------------	---

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodPH_UNI_intervalCensoring(  
  h,  
  alpha,  
  S,  
  obs,  
  weight,  
  rcens,  
  rcweight  
)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
obs	The observations.
weight	Weights of the observations.
rcens	Interval censored observations.
rcweight	Weights of the censored observations.

logLikelihoodPH_UNIs *Loglikelihood of PI with phase-type using uniformization*

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodPH_UNIs(h, alpha, S, obs, weight, rcens, rcweight, scale1, scale2)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
obs	The observations.
weight	Weights of the observations.
rcens	Censored observations.
rcweight	Weights of the censored observations.
scale1	Scale for observations.
scale2	Scale for censored observations.

logLikelihoodPH_UNIs_intervalCensoring
<i>Loglikelihood of PI with phase-type using uniformization</i>

Description

Loglikelihood for a sample.

Usage

```
logLikelihoodPH_UNIs_intervalCensoring(  
  h,  
  alpha,  
  S,  
  obs,  
  weight,  
  rcens,  
  rcweight,  
  scale1,  
  scale2  
)
```

Arguments

h	Positive parameter.
alpha	Initial probabilities.
S	Sub-intensity matrix.
obs	The observations.
weight	Weights of the observations.
rcens	Interval censored observations.
rcweight	Weights of the censored observations.
scale1	Scale for observations.
scale2	Scale for censored observations.

LRT	<i>New generic for likelihood ratio test between two matrix distribution models</i>
-----	---

Description

Methods are available for objects of class [ph](#).

Usage

LRT(x, y, ...)

Arguments

- x, y Objects of the model class.
- ... Further parameters to be passed on.

Value

A likelihood ratio test result.

LRT, ph, ph-method	<i>LRT method for ph class</i>
--------------------	--------------------------------

Description

LRT method for ph class

Usage

```
## S4 method for signature 'ph,ph'  
LRT(x, y)
```

Arguments

- x, y Objects of class [ph](#).

Value

LRT between the models.

m_exp_sum	<i>Computes exp(Sx) via series representation</i>
-----------	---

Description

Computes exp(Sx) via series representation

Usage

m_exp_sum(x, n, pow_vector, a)

Arguments

x	A number.
n	An integer.
pow_vector	A vector.
a	A number.

marginal

New generic for the marginals of multivariate matrix distributions

Description

Methods are available for objects of multivariate classes.

Usage

```
marginal(x, ...)
```

Arguments

x	An object of the model class.
...	Further parameters to be passed on.

Value

Marginal of the matrix distribution.

marginal, bivdph-method

Marginal method for bivdph class

Description

Marginal method for bivdph class

Usage

```
## S4 method for signature 'bivdph'
marginal(x, mar = 1)
```

Arguments

x	An object of class bivdph .
mar	Indicator of which marginal.

Value

An object of the of class [dph](#).

Examples

```
obj <- bivdph(dimensions = c(3, 3))
marginal(obj, 1)
```

marginal,biviph-method

Marginal method for biviph class

Description

Marginal method for biviph class

Usage

```
## S4 method for signature 'biviph'
marginal(x, mar = 1)
```

Arguments

x	An object of class biviph .
mar	Indicator of which marginal.

Value

An object of the of class [iph](#).

Examples

```
under_bivph <- bivph(dimensions = c(3, 3))
obj <- biviph(under_bivph, gfun = c("weibull", "pareto"), gfun_pars = list(c(2), c(3)))
marginal(obj, 1)
```

marginal,bivph-method *Marginal method for bivph class*

Description

Marginal method for bivph class

Usage

```
## S4 method for signature 'bivph'  
marginal(x, mar = 1)
```

Arguments

x	An object of class bivph .
mar	Indicator of which marginal.

Value

An object of the of class [ph](#).

Examples

```
obj <- bivph(dimensions = c(3, 3))  
marginal(obj, 1)
```

marginal,mdph-method *Marginal method for mdph class*

Description

Marginal method for mdph class

Usage

```
## S4 method for signature 'mdph'  
marginal(x, mar = 1)
```

Arguments

x	An object of class mdph .
mar	Indicator of which marginal.

Value

An object of the of class [dph](#).

Examples

```
obj <- mdph(structure = c("general", "general"))
marginal(obj, 1)
```

marginal,miph-method	<i>Marginal method for multivariate inhomogeneous phase-type distributions</i>
----------------------	--

Description

Marginal method for multivariate inhomogeneous phase-type distributions

Usage

```
## S4 method for signature 'miph'
marginal(x, mar = 1)
```

Arguments

x	An object of class miph .
mar	Indicator of which marginal.

Value

An object of the of class [iph](#).

Examples

```
under_mph <- mph(structure = c("general", "general"))
obj <- miph(under_mph, gfun = c("weibull", "pareto"), gfun_pars = list(c(2), c(3)))
marginal(obj, 1)
```

marginal,mph-method	<i>Marginal method for multivariate phase-type distributions</i>
---------------------	--

Description

Marginal method for multivariate phase-type distributions

Usage

```
## S4 method for signature 'mph'
marginal(x, mar = 1)
```

Arguments

x	An object of class mph .
mar	Indicator of which marginal.

Value

An object of the of class [ph](#).

Examples

```
obj <- mph(structure = c("general", "general"))
marginal(obj, 1)
```

marginal, MPHstar-method

Marginal method for MPHstar class

Description

Marginal method for MPHstar class

Usage

```
## S4 method for signature 'MPHstar'
marginal(x, mar = 1)
```

Arguments

x	An object of class MPHstar .
mar	Indicator of which marginal.

Value

An object of the of class [ph](#).

Examples

```
obj <- MPHstar(structure = "general")
marginal(obj, 1)
```

marginal_expectation	<i>Marginal conditional expectations</i>
----------------------	--

Description

Marginal conditional expectations

Usage

marginal_expectation(rew, pos, N, alpha, S, obs, weight)

Arguments

- rew Column of the reward matrix corresponding to its marginal.
- pos Vector that indicates which state is associated to a positive reward.
- N Uniformization parameter.
- alpha Marginal initial distribution vector.
- S Marginal sub-intensity matrix.
- obs Marginal observations.
- weight Marginal weights.

Value

A vector with the expected time spent in each state by the marginal, conditional on the observations.

matrix_exponential	<i>Matrix exponential</i>
--------------------	---------------------------

Description

MATLAB's built-in algorithm for matrix exponential - Pade approximation.

Usage

matrix_exponential(A)

Arguments

- A A matrix.

Value

exp(A).

matrix_inverse	<i>Inverse of a matrix</i>
----------------	----------------------------

Description

Inverse of a matrix

Usage

```
matrix_inverse(A)
```

Arguments

A	A matrix.
---	-----------

Value

Inverse of A.

matrix_power	<i>Computes A^n</i>
--------------	----------------------------------

Description

Computes A^n

Usage

```
matrix_power(n, A)
```

Arguments

n	An integer.
A	A matrix.

Value

A^n .

matrix_product	<i>Product of two matrices</i>
----------------	--------------------------------

Description

Product of two matrices

Usage

```
matrix_product(A1, A2)
```

Arguments

- A1 A matrix.
- A2 A matrix.

Value

Computes $A1 * A2$.

matrix_vanloan	<i>Creates the matrix (A1, B1 ; 0, A2)</i>
----------------	--

Description

Creates the matrix (A1, B1 ; 0, A2)

Usage

```
matrix_vanloan(A1, A2, B1)
```

Arguments

- A1 Matrix.
- A2 Matrix.
- B1 Matrix.

Value

Computes (A1, B1 ; 0, A2).

max_diagonal	<i>Maximum diagonal element of a matrix</i>
--------------	---

Description

Maximum diagonal element of a matrix

Usage

```
max_diagonal(A)
```

Arguments

A	Matrix.
---	---------

Value

The maximum value in the diagonal.

maximum	<i>New generic for maximum of two matrix distributions</i>
---------	--

Description

Methods are available for objects of class [ph](#).

Usage

```
maximum(x1, x2, ...)
```

Arguments

x1	An object of the model class.
x2	An object of the model class.
...	Further parameters to be passed on.

Value

An object of the model class.

`maximum,dph,dph-method`*Maximum method for discrete phase-type distributions*

Description

Maximum method for discrete phase-type distributions

Usage

```
## S4 method for signature 'dph,dph'
maximum(x1, x2)
```

Arguments

<code>x1</code>	An object of class <code>dph</code> .
<code>x2</code>	An object of class <code>dph</code> .

Value

An object of class `dph`.

Examples

```
dph1 <- dph(structure = "general", dimension = 3)
dph2 <- dph(structure = "general", dimension = 5)
dph_max <- maximum(dph1, dph2)
dph_max
```

`maximum,iph,iph-method`*Maximum method for inhomogeneous phase-type distributions*

Description

Maximum method for inhomogeneous phase-type distributions

Usage

```
## S4 method for signature 'iph,iph'
maximum(x1, x2)
```

Arguments

<code>x1</code>	An object of class <code>iph</code> .
<code>x2</code>	An object of class <code>iph</code> .

Value

An object of class `iph`.

Examples

```
iph1 <- iph(ph(structure = "general", dimension = 3), gfun = "weibull", gfun_pars = 2)
iph2 <- iph(ph(structure = "gcoxian", dimension = 5), gfun = "weibull", gfun_pars = 2)
iph_min <- maximum(iph1, iph2)
iph_min
```

maximum,ph,ph-method	<i>Maximum method for phase-type distributions</i>
----------------------	--

Description

Maximum method for phase-type distributions

Usage

```
## S4 method for signature 'ph,ph'
maximum(x1, x2)
```

Arguments

x1	An object of class <code>ph</code> .
x2	An object of class <code>ph</code> .

Value

An object of class `ph`.

Examples

```
ph1 <- ph(structure = "general", dimension = 3)
ph2 <- ph(structure = "gcoxian", dimension = 5)
ph_max <- maximum(ph1, ph2)
ph_max
```

mdph	<i>Constructor function for multivariate discrete phase-type distributions</i>
------	--

Description

Constructor function for multivariate discrete phase-type distributions

Usage

```
mdph(alpha = NULL, S = NULL, structure = NULL, dimension = 3, variables = NULL)
```

Arguments

alpha	A probability vector.
S	A list of sub-transition matrices.
structure	A vector of valid ph structures.
dimension	The dimension of the dph structure (if provided).
variables	The dimension of the multivariate discrete phase-type.

Value

An object of class [mdph](#).

Examples

```
mdph(structure = c("general", "general"), dimension = 5)
```

mdph-class	<i>Multivariate discrete phase-type distributions</i>
------------	---

Description

Class of objects for multivariate discrete phase-type distributions.

Value

Class object.

Slots

name	Name of the discrete phase-type distribution.
pars	A list comprising of the parameters.
fit	A list containing estimation information.

mdphdensity	<i>Multivariate discrete phase-type density</i>
-------------	---

Description

Computes the density of multivariate discrete phase-type distribution with parameters α and S at x .

Usage

```
mdphdensity(x, alpha, S_list)
```

Arguments

x	Matrix of positive integer values.
α	Initial probabilities.
S_list	List of marginal sub-transition matrices.

Value

The density at x .

mean,bivdph-method	<i>Mean method for bivdph class</i>
--------------------	-------------------------------------

Description

Mean method for bivdph class

Usage

```
## S4 method for signature 'bivdph'
mean(x)
```

Arguments

x	An object of class bivdph .
-----	---

Value

The mean of the bivariate discrete phase-type distribution.

Examples

```
obj <- bivdph(dimensions = c(3, 3))
mean(obj)
```

mean,bivph-method	<i>Mean Method for bivph class</i>
-------------------	------------------------------------

Description

Mean Method for bivph class

Usage

```
## S4 method for signature 'bivph'  
mean(x)
```

Arguments

x An object of class [bivph](#).

Value

The mean of the bivariate phase-type distribution.

Examples

```
obj <- bivph(dimensions = c(3, 3))  
mean(obj)
```

mean,dph-method	<i>Mean method for discrete phase-type distributions</i>
-----------------	--

Description

Mean method for discrete phase-type distributions

Usage

```
## S4 method for signature 'dph'  
mean(x)
```

Arguments

x An object of class [dph](#).

Value

The raw first moment of the [dph](#) object.

Examples

```
set.seed(123)
obj <- dph(structure = "general", dimension = 3)
mean(obj)
```

Value

The mean of the multivariate phase-type distribution.

Examples

```
obj <- mph(structure = c("general", "general"))
mean(obj)
```

mean,MPHstar-method	<i>Mean method for MPHstar class</i>
---------------------	--------------------------------------

Description

Mean method for MPHstar class

Usage

```
## S4 method for signature 'MPHstar'
mean(x)
```

Arguments

x An object of class [MPHstar](#).

Value

The mean of MPHstar distribution.

Examples

```
obj <- MPHstar(structure = "general")
mean(obj)
```

mean,ph-method	<i>Mean method for phase-type distributions</i>
----------------	---

Description

Mean method for phase-type distributions

Usage

```
## S4 method for signature 'ph'
mean(x)
```

Arguments

x An object of class [ph](#).

Value

The raw first moment of the [ph](#) (or underlying [ph](#)) object.

Examples

```
set.seed(123)
obj <- ph(structure = "general", dimension = 3)
mean(obj)
```

merge_matrices	<i>Merges the matrices S11, S12 and S22 into a sub-intensity matrix</i>
----------------	---

Description

Merges the matrices S11, S12 and S22 into a sub-intensity matrix

Usage

```
merge_matrices(S11, S12, S22)
```

Arguments

S11	A sub-intensity matrix.
S12	A matrix.
S22	A sub-intensity matrix.

Value

A sub-intensity matrix.

merlang_dph

Mixture of Erlang (discrete) constructor

Description

Builds a discrete phase-type model with block-diagonal Erlang components.

Usage

```
merlang_dph(block_sizes, probs = NULL, q = NULL)
```

Arguments

block_sizes	Integer vector with Erlang orders for each component.
probs	Optional mixture weights. If omitted, random normalized weights are used.
q	Optional forward probabilities in (0,1) for each component. If omitted, random probabilities are used.

Value

An object of class [dph](#).

merlang_ph

Mixture of Erlang (continuous) constructor

Description

Builds a continuous phase-type model with block-diagonal Erlang components (a.k.a. hyper-Erlang with fixed orders).

Usage

```
merlang_ph(block_sizes, probs = NULL, rates = NULL)
```

Arguments

block_sizes	Integer vector with Erlang orders for each component.
probs	Optional mixture weights. If omitted, random normalized weights are used.
rates	Optional positive rates for each component. If omitted, random rates are used.

Value

An object of class [ph](#).

mgevpdf	<i>Matrix-GEV cdf</i>
---------	-----------------------

Description

Computes the cdf (tail) of a matrix-GEV distribution with parameters alpha, S and beta at x.

Usage

```
mgevpdf(x, alpha, S, beta, lower_tail = TRUE)
```

Arguments

x	Non-negative value.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Transformation parameters.
lower_tail	Cdf or tail.

Value

The cdf (tail) at x.

mgevden	<i>Matrix-GEV density</i>
---------	---------------------------

Description

Computes the density of a matrix-GEV distribution with parameters alpha, S and beta at x. Does not allow for atoms in zero.

Usage

```
mgevden(x, alpha, S, beta)
```

Arguments

x	Non-negative value.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Transformation parameters.

Value

The density at x.

mgf	<i>New generic for mgf of matrix distributions</i>
-----	--

Description

Methods are available for objects of class [ph](#).

Usage

```
mgf(x, ...)
```

Arguments

x	An object of the model class.
...	Further parameters to be passed on.

Value

Mgf of the matrix distribution.

mgf,bivph-method	<i>Mgf method for bivph class</i>
------------------	-----------------------------------

Description

Mgf method for bivph class

Usage

```
## S4 method for signature 'bivph'
mgf(x, r)
```

Arguments

x	An object of class mph .
r	A matrix of real values.

Value

A vector containing the corresponding mgf evaluations.

Examples

```
set.seed(123)
obj <- bivph(dimensions = c(3, 3))
mgf(obj, matrix(c(0.5, 0.1), ncol = 2))
```

mgf,mph-method	<i>Mgf method for multivariate phase-type distributions</i>
----------------	---

Description

Mgf method for multivariate phase-type distributions

Usage

```
## S4 method for signature 'mph'
mgf(x, r)
```

Arguments

x	An object of class mph .
r	A matrix of real values.

Value

A vector containing the corresponding mgf evaluations.

Examples

```
set.seed(124)
obj <- mph(structure = c("general", "general"))
mgf(obj, matrix(c(0.5, 0.3), ncol = 2))
```

mgf,ph-method	<i>Mgf method for phase-type distributions</i>
---------------	--

Description

Mgf method for phase-type distributions

Usage

```
## S4 method for signature 'ph'
mgf(x, r)
```

Arguments

x	An object of class ph .
r	A vector of real values.

Value

The mgf of the [ph](#) (or underlying [ph](#)) object at the given locations.

Examples

```
set.seed(123)
obj <- ph(structure = "general", dimension = 3)
mgf(obj, 0.4)
```

mgompertzcdf	<i>Matrix-Gompertz cdf</i>
--------------	----------------------------

Description

Computes the cdf (tail) of a matrix-Gompertz distribution with parameters alpha, S and beta at x.

Usage

```
mgompertzcdf(x, alpha, S, beta, lower_tail = TRUE)
```

Arguments

x	Non-negative value.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Shape parameter.
lower_tail	Cdf or tail.

Value

The cdf (tail) at x.

mgompertzden	<i>Matrix-Gompertz density</i>
--------------	--------------------------------

Description

Computes the density of a matrix-Gompertz distribution with parameters alpha, S and beta at x.

Usage

```
mgompertzden(x, alpha, S, beta)
```

Arguments

x	Non-negative value.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Shape parameter.

Value

The density at x.

minimum	<i>New generic for minimum of two matrix distributions</i>
---------	--

Description

Methods are available for objects of class [ph](#).

Usage

```
minimum(x1, x2, ...)
```

Arguments

x1	An object of the model class.
x2	An object of the model class.
...	Further parameters to be passed on.

Value

An object of the model class.

minimum,dph,dph-method	<i>Minimum method for discrete phase-type distributions</i>
------------------------	---

Description

Minimum method for discrete phase-type distributions

Usage

```
## S4 method for signature 'dph,dph'
minimum(x1, x2)
```

Arguments

x1 An object of class `dph`.
x2 An object of class `dph`.

Value

An object of class `dph`.

Examples

```
dph1 <- dph(structure = "general", dimension = 3)
dph2 <- dph(structure = "general", dimension = 5)
dph_min <- minimum(dph1, dph2)
dph_min
```

minimum,iph,iph-method

Minimum method for inhomogeneous phase-type distributions

Description

Minimum method for inhomogeneous phase-type distributions

Usage

```
## S4 method for signature 'iph,iph'
minimum(x1, x2)
```

Arguments

x1 An object of class `iph`.
x2 An object of class `iph`.

Value

An object of class `iph`.

Examples

```
iph1 <- iph(ph(structure = "general", dimension = 3), gfun = "weibull", gfun_pars = 2)
iph2 <- iph(ph(structure = "gcoxian", dimension = 5), gfun = "weibull", gfun_pars = 2)
iph_min <- minimum(iph1, iph2)
iph_min
```

minimum,ph,ph-method	<i>Minimum method for phase-type distributions</i>
----------------------	--

Description

Minimum method for phase-type distributions

Usage

```
## S4 method for signature 'ph,ph'
minimum(x1, x2)
```

Arguments

x1	An object of class ph .
x2	An object of class ph .

Value

An object of class [ph](#).

Examples

```
ph1 <- ph(structure = "general", dimension = 3)
ph2 <- ph(structure = "gcoxian", dimension = 5)
ph_min <- minimum(ph1, ph2)
ph_min
```

miph	<i>Constructor function for multivariate inhomogeneous phase-type distributions</i>
------	---

Description

Constructor function for multivariate inhomogeneous phase-type distributions

Usage

```
miph(
  mph = NULL,
  gfun = NULL,
  gfun_pars = NULL,
  alpha = NULL,
  S = NULL,
  structure = NULL,
  dimension = 3,
```

```
variables = NULL,  
scale = 1  
)
```

Arguments

mph	An object of class mph .
gfun	Vector of inhomogeneity transforms.
gfun_pars	List of parameters for the inhomogeneity functions.
alpha	A probability vector.
S	A list of sub-intensity matrices.
structure	A vector of valid ph structures.
dimension	The dimension of the ph structure (if provided).
variables	Number of marginals.
scale	Scale.

Value

An object of class [iph](#).

Examples

```
under_mph <- mph(structure = c("gcoxian", "general"), dimension = 4)  
miph(under_mph, gfun = c("weibull", "pareto"), gfun_pars = list(c(2), c(3)))
```

miph-class	<i>Multivariate inhomogeneous phase-type distributions</i>
------------	--

Description

Class of objects for multivariate inhomogeneous phase-type distributions.

Value

Class object.

Slots

- name Name of the phase type distribution.
- gfun A list comprising of the parameters.
- scale Scale.

`mixture`*New generic for mixture of two matrix distributions*

Description

Methods are available for objects of classes `ph` and `dph`.

Usage

```
mixture(x1, x2, ...)
```

Arguments

<code>x1</code>	An object of the model class.
<code>x2</code>	An object of the model class.
<code>...</code>	Further parameters to be passed on.

Value

An object of the model class.

`mixture,dph,dph-method`*Mixture method for phase-type distributions*

Description

Mixture method for phase-type distributions

Usage

```
## S4 method for signature 'dph,dph'  
mixture(x1, x2, prob)
```

Arguments

<code>x1</code>	An object of class <code>dph</code> .
<code>x2</code>	An object of class <code>dph</code> .
<code>prob</code>	Probability for first object.

Value

An object of class `dph`.

Examples

```
dph1 <- dph(structure = "general", dimension = 3)
dph2 <- dph(structure = "general", dimension = 5)
dph_mix <- mixture(dph1, dph2, 0.5)
dph_mix
```

mixture,ph,ph-method	<i>Mixture method for phase-type distributions</i>
----------------------	--

Description

Mixture method for phase-type distributions

Usage

```
## S4 method for signature 'ph,ph'
mixture(x1, x2, prob)
```

Arguments

x1	An object of class ph .
x2	An object of class ph .
prob	Probability for first object.

Value

An object of class [ph](#).

Examples

```
ph1 <- ph(structure = "general", dimension = 3)
ph2 <- ph(structure = "gcoxian", dimension = 5)
ph_mix <- mixture(ph1, ph2, 0.5)
ph_mix
```

mloglogisticcdf	<i>Matrix-loglogistic cdf</i>
-----------------	-------------------------------

Description

Computes the cdf (tail) of a matrix-loglogistic distribution with parameters alpha, S and beta at x.

Usage

```
mloglogisticcdf(x, alpha, S, beta, lower_tail = TRUE)
```

Arguments

x	Non-negative value.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Transformation parameters.
lower_tail	Cdf or tail.

Value

The cdf (tail) at x.

mloglogisticden	<i>Matrix-loglogistic density</i>
-----------------	-----------------------------------

Description

Computes the density of a matrix-loglogistic distribution with parameters alpha, S and beta at x.

Usage

```
mloglogisticden(x, alpha, S, beta)
```

Arguments

x	Non-negative value.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Transformation parameters.

Value

The density at x.

mlognormalcdf	<i>Matrix-lognormal cdf</i>
---------------	-----------------------------

Description

Computes the cdf (tail) of a matrix-lognormal distribution with parameters alpha, S and beta at x.

Usage

```
mlognormalcdf(x, alpha, S, beta, lower_tail = TRUE)
```

Arguments

x	Non-negative value.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Shape parameter.
lower_tail	Cdf or tail.

Value

The cdf (tail) at x.

mlognormalden	<i>Matrix-lognormal density</i>
---------------	---------------------------------

Description

Computes the density of a matrix-lognormal distribution with parameters alpha, S and beta at x.

Usage

```
mlognormalden(x, alpha, S, beta)
```

Arguments

x	Non-negative value.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Shape parameter.

Value

The density at x.

MoE	<i>New generic for mixture-of-experts regression with matrix distributions</i>
-----	--

Description

Methods are available for objects of class `ph`

Usage

```
MoE(x, y, ...)
```

Arguments

- `x` An object of the model class.
- `y` A vector of data.
- `...` Further parameters to be passed on.

Value

An object of the fitted model class.

MoE, bivdph-method	<i>MoE method for bivdph Class</i>
--------------------	------------------------------------

Description

MoE method for bivdph Class

Usage

```
## S4 method for signature 'bivdph'
MoE(
  x,
  formula,
  y,
  data,
  alpha_vecs = NULL,
  weight = numeric(0),
  stepsEM = 1000,
  every = 10,
  rand_init = TRUE
)
```

Arguments

<code>x</code>	An object of class <code>bivdph</code> .
<code>formula</code>	A regression formula.
<code>y</code>	A matrix of observations.
<code>data</code>	A data frame of covariates.
<code>alpha_vecs</code>	Matrix of initial probabilities.
<code>weight</code>	Vector of weights.
<code>stepsEM</code>	Number of EM steps to be performed.
<code>every</code>	Number of iterations between likelihood display updates.
<code>rand_init</code>	Random initiation in the R-step.

Value

An object of class `sph`.

Examples

```
x <- bivdph(dimensions = c(3, 3))
n <- 100
responses <- cbind(rpois(n, 3) + 1, rbinom(n, 5, 0.5))
covariates <- data.frame(age = sample(18:65, n, replace = TRUE) / 100, income = runif(n, 0, 0.99))
f <- responses ~ age + income
MoE(x = x, formula = f, y = responses, data = covariates, stepsEM = 20)
```

MoE,dph-method

MoE method for dph Class

Description

MoE method for dph Class

Usage

```
## S4 method for signature 'dph'
MoE(
  x,
  formula,
  data,
  alpha_vecs = NULL,
  weight = numeric(0),
  stepsEM = 1000,
  every = 10,
  rand_init = TRUE,
  maxWts = 1000
)
```

Arguments

x	An object of class dph .
formula	A regression formula.
data	A data frame.
alpha_vecs	Matrix of initial probabilities.
weight	Vector of weights.
stepsEM	Number of EM steps to be performed.
every	Number of iterations between likelihood display updates.
rand_init	Random initiation in the R-step.
maxWts	Maximal number of weights in the nnet function.

Value

An object of class [sph](#).

Examples

```
x <- dph(structure = "general")
n <- 100
responses <- rpois(n, 3) + 1
covariate <- data.frame(age = sample(18:65, n, replace = TRUE) / 100, income = runif(n, 0, 0.99))
f <- responses ~ age + income # regression formula
MoE(x = x, formula = f, y = responses, data = covariate, stepsEM = 20)
```

MoE,mdph-method

MoE method for mdph Class

Description

MoE method for mdph Class

Usage

```
## S4 method for signature 'mdph'
MoE(
  x,
  formula,
  y,
  data,
  alpha_vecs = NULL,
  weight = numeric(0),
  stepsEM = 1000,
  every = 10,
  rand_init = TRUE,
  maxWts = 1000
)
```

Arguments

<code>x</code>	An object of class <code>mdph</code> .
<code>formula</code>	A regression formula.
<code>y</code>	A matrix of observations.
<code>data</code>	A data frame of covariates.
<code>alpha_vecs</code>	Matrix of initial probabilities.
<code>weight</code>	Vector of weights.
<code>stepsEM</code>	Number of EM steps to be performed.
<code>every</code>	Number of iterations between likelihood display updates.
<code>rand_init</code>	Random initiation in the R-step.
<code>maxWts</code>	Maximal number of weights in the <code>nnet</code> function.

Value

An object of class `sph`.

Examples

```
x <- mdph(structure = c("general", "general"))
n <- 100
responses <- cbind(rpois(n, 3) + 1, rbinom(n, 5, 0.5))
covariates <- data.frame(age = sample(18:65, n, replace = TRUE) / 100, income = runif(n, 0, 0.99))
f <- responses ~ age + income
MoE(x = x, formula = f, y = responses, data = covariates, stepsEM = 20)
```

MoE,mph-method

Fit method for mph/miph class, using mixture-of-experts regression

Description

Fit method for mph/miph class, using mixture-of-experts regression

Usage

```
## S4 method for signature 'mph'
MoE(
  x,
  formula,
  y,
  data,
  alpha_mat = NULL,
  delta = numeric(0),
  stepsEM = 1000,
  r = 1,
  maxit = 100,
```

```

    reltol = 1e-08,
    rand_init = T
  )

```

Arguments

<code>x</code>	An object of class <code>mph</code> .
<code>formula</code>	a regression formula.
<code>y</code>	A matrix of observations.
<code>data</code>	A data frame of covariates (they need to be scaled for the regression).
<code>alpha_mat</code>	Matrix with initial distribution vectors for each row of observations.
<code>delta</code>	Matrix with right-censoring indicators (1 uncensored, 0 right censored).
<code>stepsEM</code>	Number of EM steps to be performed.
<code>r</code>	Sub-sampling parameter, defaults to 1 (not supported for this method).
<code>maxit</code>	Maximum number of iterations when optimizing the g function (inhomogeneous likelihood).
<code>reltol</code>	Relative tolerance when optimizing g function.
<code>rand_init</code>	Random initiation in the R-step of the EM algorithm.

Examples

```

under_mph <- mph(structure = c("general", "general"), dimension = 3)
x <- mph(under_mph, gfun = c("weibull", "weibull"), gfun_pars = list(c(2), c(3)))
n <- 100
responses <- cbind(rexp(n), rweibull(n, 2, 3))
covariates <- data.frame(age = sample(18:65, n, replace = TRUE) / 100, income = runif(n, 0, 0.99))
f <- responses ~ age + income
MoE(x = x, formula = f, y = responses, data = covariates, stepsEM = 20)

```

MoE,ph-method

MoE method for ph Class

Description

MoE method for ph Class

Usage

```

## S4 method for signature 'ph'
MoE(
  x,
  formula,
  data,
  inhom = NULL,
  alpha_vecs = NULL,

```

```

weight = numeric(0),
delta = numeric(0),
stepsEM = 1000,
optim_method = "BFGS",
maxit = 50,
reltol = 1e-08,
every = 10,
rand_init = TRUE
)

```

Arguments

x	An object of class ph .
formula	A regression formula.
data	A data frame.
inhom	A list with the inhomogeneity functions.
alpha_vecs	Matrix of initial probabilities.s
weight	Vector of weights.
delta	Right-censoring indicator.
stepsEM	Number of EM steps to be performed.
optim_method	Method to use in gradient optimization.
maxit	Maximum number of iterations when optimizing g function.
reltol	Relative tolerance when optimizing g function.
every	Number of iterations between likelihood display updates.
rand_init	Random initiation in the R-step.

Value

An object of class [sph](#).

Examples

```

x <- iph(ph(structure = "general"), gfun = "weibull")
n <- 100
responses <- rweibull(n, 2, 3)
covariate <- data.frame(age = sample(18:65, n, replace = TRUE) / 100, income = runif(n, 0, 0.99))
f <- responses ~ age + income # regression formula
MoE(x = x, formula = f, y = responses, data = covariate, stepsEM = 20)

```

moment	<i>New generic for moments of matrix distributions</i>
--------	--

Description

Methods are available for objects of class [ph](#).

Usage

```
moment(x, ...)
```

Arguments

x	An object of the model class.
...	Further parameters to be passed on.

Value

Moment of the matrix distribution.

moment, bivdph-method	<i>Moment method for bivdph class</i>
-----------------------	---------------------------------------

Description

Moment method for bivdph class

Usage

```
## S4 method for signature 'bivdph'
moment(x, k = c(1, 1))
```

Arguments

x	An object of class bivdph .
k	A vector with the location.

Value

An real value.

Examples

```
obj <- bivdph(dimensions = c(3, 3))
moment(obj, c(1, 1))
```

moment,bivph-method *Moment method for bivph class*

Description

Moment method for bivph class

Usage

```
## S4 method for signature 'bivph'
moment(x, k = c(1, 1))
```

Arguments

x An object of class [bivph](#).
k A vector with the location.

Value

An real value.

Examples

```
obj <- bivph(dimensions = c(3, 3))
moment(obj, c(1, 1))
```

moment,dph-method *Moment method for discrete phase-type distributions*

Description

Moment method for discrete phase-type distributions

Usage

```
## S4 method for signature 'dph'
moment(x, k = 1)
```

Arguments

x An object of class [dph](#).
k A positive integer (moment order).

Value

The factional moment of the [dph](#) object.

Examples

```
set.seed(123)
obj <- dph(structure = "general", dimension = 3)
moment(obj, 2)
```

moment,mdph-method	<i>Moment method for multivariate discrete phase-type distributions</i>
--------------------	---

Description

Moment method for multivariate discrete phase-type distributions

Usage

```
## S4 method for signature 'mdph'
moment(x, k)
```

Arguments

x	An object of class mdph .
k	A vector of positive integer values.

Value

The corresponding joint factorial moment evaluation.

Examples

```
obj <- mdph(structure = c("general", "general"))
moment(obj, c(2, 1))
```

moment,mph-method	<i>Moment method for multivariate phase-type distributions</i>
-------------------	--

Description

Moment method for multivariate phase-type distributions

Usage

```
## S4 method for signature 'mph'
moment(x, k)
```

Arguments

x	An object of class mph .
k	A vector of non-negative integer values.

Value

The corresponding joint moment evaluation.

Examples

```
obj <- mph(structure = c("general", "general"))
moment(obj, c(2, 1))
```

moment,ph-method	<i>Moment method for phase-type distributions</i>
------------------	---

Description

Moment method for phase-type distributions

Usage

```
## S4 method for signature 'ph'
moment(x, k = 1)
```

Arguments

x	An object of class ph .
k	A positive integer (moment order).

Value

The raw moment of the [ph](#) (or underlying [ph](#)) object.

Examples

```
set.seed(123)
obj <- ph(structure = "general", dimension = 3)
moment(obj, 2)
```

mparetocdf	<i>Matrix-Pareto cdf</i>
------------	--------------------------

Description

Computes the cdf (tail) of a matrix-Pareto distribution with parameters alpha, S and beta at x.

Usage

```
mparetocdf(x, alpha, S, beta, lower_tail = TRUE)
```

Arguments

x	Non-negative value.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Scale parameter.
lower_tail	Cdf or tail.

Value

The cdf (tail) at x.

mparetoden	<i>Matrix-Pareto density</i>
------------	------------------------------

Description

Computes the density of a matrix-Pareto distribution with parameters alpha, S and beta at x.

Usage

```
mparetoden(x, alpha, S, beta)
```

Arguments

x	Non-negative value.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Scale parameter.

Value

The density at x.

mph	<i>Constructor function for multivariate phase-type distributions</i>
-----	---

Description

Constructor function for multivariate phase-type distributions

Usage

```
mph(alpha = NULL, S = NULL, structure = NULL, dimension = 3, variables = NULL)
```

Arguments

- alpha A probability vector.
- S A list of sub-intensity matrices.
- structure A vector of valid ph structures.
- dimension The dimension of the ph structure (if provided).
- variables The dimension of the multivariate phase-type.

Value

An object of class [mph](#).

Examples

```
mph(structure = c("gcoxian", "general"), dimension = 5)
```

mph-class	<i>Multivariate phase-type distributions</i>
-----------	--

Description

Class of objects for multivariate phase-type distributions.

Value

Class object.

Slots

- name Name of the phase type distribution.
- pars A list comprising of the parameters.
- fit A list containing estimation information.

MPHstar	<i>Constructor function for multivariate phase-type distributions (MPH* class)</i>
---------	--

Description

Constructor function for multivariate phase-type distributions (MPH* class)

Usage

```
MPHstar(  
  alpha = NULL,  
  S = NULL,  
  structure = NULL,  
  dimension = 3,  
  R = NULL,  
  variables = 2  
)
```

Arguments

alpha	A probability vector.
S	A sub-intensity matrix.
structure	A valid ph structure.
dimension	The dimension of the ph structure (if provided).
R	A compatible (non-negative) reward matrix.
variables	The number of desired marginals.

Value

An object of class [MPHstar](#).

Examples

```
MPHstar(structure = "general", dimension = 4, variables = 3)
```

MPHstar-class	<i>Multivariate phase-type distributions obtained by transformation via rewards</i>
---------------	---

Description

Class of objects for multivariate phase type distributions.

Slots

- name Name of the phase type distribution.
- pars A list comprising of the parameters.

MPHstar_data_aggregation	<i>Prepare data for the MPHstar_EMstep_UNI</i>
--------------------------	--

Description

Prepare data for the MPHstar_EMstep_UNI

Usage

MPHstar_data_aggregation(y, w = numeric(0))

Arguments

- y A matrix with marginal observations, each column corresponds to a marginal.
- w A matrix of weights, each column corresponds to a marginal.

Value

For summed and marginal observations we have a list with matrices of unique observations and their associated weights, separated by uncensored and right-censored data.

MPHstar_EMstep_UNI	<i>EM step using Uniformization for MPHstar class</i>
--------------------	---

Description

EM step using Uniformization for MPHstar class

Usage

```
MPHstar_EMstep_UNI(h, Rtol, alpha, S, R, mph_obs)
```

Arguments

h	positive parameter for precision of uniformization method.
Rtol	The smallest value that a reward can take.
alpha	Vector of initial probabilities of the originating distribution.
S	The sub-intensity matrix of the originating distribution.
R	The reward matrix.
mph_obs	The list of summed, marginal observations with associated weights.

mweibullcdf	<i>Matrix-Weibull cdf</i>
-------------	---------------------------

Description

Computes the cdf (tail) of a matrix-Weibull distribution with parameters alpha, S and beta at x.

Usage

```
mweibullcdf(x, alpha, S, beta, lower_tail = TRUE)
```

Arguments

x	Non-negative value.
alpha	Initial probabilities.
S	Sub-intensity matrix.
beta	Shape parameter.
lower_tail	Cdf or tail.

Value

The cdf (tail) at x.

mweibullden	<i>Matrix-Weibull density</i>
-------------	-------------------------------

Description

Computes the density of a matrix-Weibull distribution with parameters alpha, S and beta at x.

Usage

mweibullden(x, alpha, S, beta)

Arguments

- x Non-negative value.
- alpha Initial probabilities.
- S Sub-intensity matrix.
- beta Shape parameter.

Value

The density at x.

n_pos	<i>Find how many states have positive reward</i>
-------	--

Description

Find how many states have positive reward

Usage

n_pos(R)

Arguments

- R reward vector

Value

The number of states with positive rewards

new_state	<i>New state in a Markov jump process</i>
-----------	---

Description

Given a transition matrix Q , a uniform value u , and a previous state k , it returns the new state of a Markov jump process.

Usage

```
new_state(prev_state, cum_embedded_mc, u)
```

Arguments

prev_state	Previous state of the Markov jump process.
cum_embedded_mc	Transition matrix.
u	Random value in (0,1).

Value

Next state of the Markov jump process.

Nfold	<i>New generic for N-fold convolution of two matrix distributions</i>
-------	---

Description

Methods are available for objects of classes [ph](#) and [dph](#).

Usage

```
Nfold(x1, x2, ...)
```

Arguments

x1	An object of the class dph .
x2	An object of the model class.
...	Further parameters to be passed on.

Value

An object of the model class.

Nfold, dph-method	<i>Nfold method for phase-type distributions</i>
-------------------	--

Description

Nfold method for phase-type distributions

Usage

```
## S4 method for signature 'dph'
Nfold(x1, x2)
```

Arguments

x1	An object of class ph .
x2	An object of class dph .

Value

An object of class [ph](#).

Examples

```
dph1 <- dph(structure = "general", dimension = 3)
dph2 <- dph(structure = "general", dimension = 2)
ph0 <- ph(structure = "general", dimension = 2)
Nfold(dph1, ph0)
Nfold(dph1, dph2)
```

pgf	<i>New generic for pgf of matrix distributions</i>
-----	--

Description

Methods are available for objects of class [dph](#).

Usage

```
pgf(x, ...)
```

Arguments

x	An object of the model class.
...	Further parameters to be passed on.

Value

Pgf of the matrix distribution.

pgf,bivdph-method	<i>Pgf method for bivariate discrete phase-type distributions</i>
-------------------	---

Description

Pgf method for bivariate discrete phase-type distributions

Usage

```
## S4 method for signature 'bivdph'
pgf(x, z)
```

Arguments

x	An object of class bivdph .
z	A vector of real values.

Value

The joint pdf of the [dph](#) object at the given location.

Examples

```
obj <- bivdph(dimensions = c(3, 3))
pgf(obj, c(0.5, 0.2))
```

pgf,dph-method	<i>Pgf Method for discrete phase-type distributions</i>
----------------	---

Description

Pgf Method for discrete phase-type distributions

Usage

```
## S4 method for signature 'dph'
pgf(x, z)
```

Arguments

x	An object of class dph .
z	A vector of real values.

Value

The probability generating of the [dph](#) object at the given locations.

Examples

```
set.seed(123)
obj <- dph(structure = "general", dimension = 3)
pgf(obj, 0.5)
```

pgf,mdph-method

Pgf method for multivariate discrete phase-type distributions

Description

Pgf method for multivariate discrete phase-type distributions

Usage

```
## S4 method for signature 'mdph'
pgf(x, z)
```

Arguments

x An object of class [mdph](#).
z A matrix of real values.

Value

A vector containing the corresponding pgf evaluations.

Examples

```
obj <- mdph(structure = c("general", "general"))
pgf(obj, matrix(c(0.5, 1), ncol = 2))
```

ph

Constructor function for phase-type distributions

Description

Constructor function for phase-type distributions

Usage

```
ph(
  alpha = NULL,
  S = NULL,
  structure = NULL,
  dimension = 3,
  block_sizes = NULL,
  probs = NULL,
  rates = NULL
)
```

Arguments

alpha	A probability vector.
S	A sub-intensity matrix.
structure	A valid ph structure: "general", "coxian", "hyperexponential", "gcoxian", "gerlang", "erlang", or "merlang".
dimension	The dimension of the ph structure (if structure is provided).
block_sizes	Optional integer vector with Erlang block sizes when structure = "merlang" (or aliases "mixederlang", "mixed_erlang").
probs	Optional mixture weights used only for structure = "merlang".
rates	Optional Erlang rates used only for structure = "merlang".

Value

An object of class `ph`.

Examples

```
ph(structure = "gcoxian", dimension = 5)
ph(structure = "merlang", block_sizes = c(2, 3))
ph(alpha = c(.5, .5), S = matrix(c(-1, .5, .5, -1), 2, 2))
```

ph-class	<i>Phase-type distributions</i>
----------	---------------------------------

Description

Class of objects for phase-type distributions.

Value

Class object.

Slots

- name Name of the phase-type distribution.
- pars A list comprising of the parameters.
- fit A list containing estimation information.

ph_laplace	<i>Laplace transform of a phase-type distribution</i>
------------	---

Description

Computes the Laplace transform at r of a phase-type distribution with parameters α and S .

Usage

```
ph_laplace(r, alpha, S)
```

Arguments

r	Vector of real values.
α	Vector of initial probabilities.
S	Sub-intensity matrix.

Value

Laplace transform at r .

phcdf	<i>Phase-type cdf</i>
-------	-----------------------

Description

Computes the cdf (tail) of a phase-type distribution with parameters α and S at x .

Usage

```
phcdf(x, alpha, S, lower_tail = TRUE)
```

Arguments

x	Non-negative value.
α	Initial probabilities.
S	Sub-intensity matrix.
<code>lower_tail</code>	Cdf or tail.

Value

The cdf (tail) at x .

phdensity	<i>Phase-type density</i>
-----------	---------------------------

Description

Computes the density of a phase-type distribution with parameters alpha and S at x.

Usage

```
phdensity(x, alpha, S)
```

Arguments

x	Non-negative value.
alpha	Initial probabilities.
S	Sub-intensity matrix.

Value

The density at x.

plus_states	<i>Find which states have positive reward</i>
-------------	---

Description

Find which states have positive reward

Usage

```
plus_states(R)
```

Arguments

R	reward vector
---	---------------

Value

A vector with the states (number) that are associated with positive rewards

pow2_matrix	<i>Computes $A^{(2^n)}$</i>
-------------	--

Description

Computes $A^{(2^n)}$

Usage

pow2_matrix(n, A)

Arguments

- n An integer.
- A A matrix.

Value

$A^{(2^n)}$.

quan	<i>New generic for the quantile of matrix distributions</i>
------	---

Description

Methods are available for objects of class [ph](#).

Usage

quan(x, ...)

Arguments

- x An object of the model class.
- ... Further parameters to be passed on.

Value

Quantile from the matrix distribution.

quan,ph-method	<i>Quantile method for phase-type distributions</i>
----------------	---

Description

Quantile method for phase-type distributions

Usage

```
## S4 method for signature 'ph'
quan(x, p)
```

Arguments

x	An object of class ph .
p	A vector of probabilities.

Value

A vector containing the quantile evaluations at the given locations.

Examples

```
obj <- ph(structure = "general")
quan(obj, c(0.5, 0.9, 0.99))
```

random_reward	<i>Random reward matrix</i>
---------------	-----------------------------

Description

Generates a random reward matrix for a multivariate phase-type distribution with p states and d marginals.

Usage

```
random_reward(p, d)
```

Arguments

p	Number of transient states in the sub-intensity matrix.
d	Number of marginals.

Value

A random reward matrix.

random_structure	<i>Random structure of a phase-type</i>
------------------	---

Description

Generates random parameters α and S of a phase-type distribution of dimension p with chosen structure.

Usage

```
random_structure(p, structure = "general", scale_factor = 1)
```

Arguments

p	Dimension of the phase-type.
structure	Type of structure: "general", "hyperexponential", "gerlang", "erlang", "coxian" or "gcoxian".
scale_factor	A factor that multiplies the sub-intensity matrix.

Value

Random parameters α and S of a phase-type.

random_structure_bivph	<i>Random structure of a bivariate phase-type</i>
------------------------	---

Description

Generates random parameters α , S_{11} , S_{12} , and S_{22} of a bivariate phase-type distribution of dimension $p = p_1 + p_2$.

Usage

```
random_structure_bivph(p1, p2, scale_factor = 1)
```

Arguments

p_1	Dimension of the first block.
p_2	Dimension of the second block.
scale_factor	A factor that multiplies the sub-intensity matrix.

Value

Random parameters α , S_{11} , S_{12} , and S_{22} of a bivariate phase-type.

`rcsph`*Simulate common-shock phase-type*

Description

Generates a sample of size n from a phase-type distribution with parameters α , S , P , Q_1 , Q_2 , a_1 , and a_2

Usage

```
rcsph(n, alpha, S, P, Q1, Q2, a1, a2)
```

Arguments

<code>n</code>	Sample size.
<code>alpha</code>	Vector of initial probabilities.
<code>S</code>	Sub-intensity matrix.
<code>P</code>	Matrix.
<code>Q1</code>	Sub-intensity matrix.
<code>Q2</code>	Sub-intensity matrix.
<code>a1</code>	Multiplicative constant for margin 1.
<code>a2</code>	Multiplicative constant for margin 1.

Value

Simulated sample.

`rdphasetype`*Simulate discrete phase-type*

Description

Generates a sample of size n from a discrete phase-type distribution with parameters α and S .

Usage

```
rdphasetype(n, alpha, S)
```

Arguments

<code>n</code>	Sample size.
<code>alpha</code>	Vector of initial probabilities.
<code>S</code>	Sub-transition matrix.

Value

Simulated sample.

reg	<i>New generic for regression with matrix distributions</i>
-----	---

Description

Methods are available for objects of class [ph](#).

Usage

```
reg(x, y, ...)
```

Arguments

- x An object of the model class.
- y A vector of data.
- ... Further parameters to be passed on.

Value

An object of the fitted model class.

reg,ph-method	<i>Regression method for ph Class</i>
---------------	---------------------------------------

Description

Regression method for ph Class

Usage

```
## S4 method for signature 'ph'
reg(
  x,
  y,
  weight = numeric(0),
  rcen = numeric(0),
  rcenweight = numeric(0),
  X = numeric(0),
  B0 = numeric(0),
  stepsEM = 1000,
  methods = c("RK", "UNI"),
  rkstep = NA,
```

```

uni_epsilon = NA,
optim_method = "BFGS",
maxit = 50,
reltol = 1e-08,
every = 10,
erlang = NULL,
merlang_blocks = NULL
)

```

Arguments

x	An object of class ph .
y	Vector or data.
weight	Vector of weights.
rcen	Vector of right-censored observations.
rcenweight	Vector of weights for right-censored observations.
X	Model matrix (no intercept needed).
B0	Initial regression coefficients (optional).
stepsEM	Number of EM steps to be performed.
methods	Methods to use for matrix exponential calculation: RM, UNI, or PADE.
rkstep	Runge-Kutta step size (optional).
uni_epsilon	Epsilon parameter for uniformization method.
optim_method	Method to use in gradient optimization.
maxit	Maximum number of iterations when optimizing g function.
reltol	Relative tolerance when optimizing g function.
every	Number of iterations between likelihood display updates.
erlang	Logical flag for exact Erlang EM updates with one repeated rate. If NULL, this is auto-detected from the model name.
merlang_blocks	Optional integer vector with Erlang block sizes for exact mixture-of-Erlangs EM updates. If NULL, it is auto-read from object attributes when available.

Value

An object of class [sph](#).

Examples

```

set.seed(1)
obj <- iph(ph(structure = "general", dimension = 2), gfun = "weibull", gfun_pars = 2)
data <- sim(obj, n = 100)
X <- runif(100)
reg(x = obj, y = data, X = X, stepsEM = 10)

```

revers_data_trans	<i>Applies the inverse of the GEV transformation but giving back the resulting vector in reverse order</i>
-------------------	--

Description

Used for EM step in RK.

Usage

```
revers_data_trans(obs, weights, beta)
```

Arguments

obs	The observations.
weights	Weights of the observations.
beta	Parameters of the GEV.

rew_sanity_check	<i>Transform a reward matrix with very small rewards to avoid numerical problems</i>
------------------	--

Description

Transform a reward matrix with very small rewards to avoid numerical problems

Usage

```
rew_sanity_check(R, tol)
```

Arguments

R	Reward matrix
tol	Lower bound considered for a reward

Value

A reward matrix that does not cause issues with uniformization

riph	<i>Random inhomogeneous phase-type</i>
------	--

Description

Generates a sample of size n from an inhomogeneous phase-type distribution with parameters α , S and β .

Usage

```
riph(n, dist_type, alpha, S, beta)
```

Arguments

<code>n</code>	Sample size.
<code>dist_type</code>	Type of IPH.
<code>alpha</code>	Initial probabilities.
<code>S</code>	Sub-intensity matrix.
<code>beta</code>	Parameter of the transformation.

Value

The simulated sample.

rmatrixgev	<i>Random matrix GEV</i>
------------	--------------------------

Description

Generates a sample of size n from an inhomogeneous phase-type distribution with parameters α , S and β .

Usage

```
rmatrixgev(n, alpha, S, mu, sigma, xi = 0)
```

Arguments

<code>n</code>	Sample size.
<code>alpha</code>	Initial probabilities.
<code>S</code>	Sub-intensity matrix.
<code>mu</code>	Location parameter.
<code>sigma</code>	Scale parameter.
<code>xi</code>	Shape parameter: Default 0 which corresponds to the Gumbel case.

Value

The simulated sample.

rMDPHstar	<i>Simulate MDPH*</i>
-----------	-----------------------

Description

Generates a sample of size n from a MDPH* distribution with parameters alpha, S, and R.

Usage

rMDPHstar(n, alpha, S, R)

Arguments

- | | |
|-------|----------------------------------|
| n | Sample size. |
| alpha | Vector of initial probabilities. |
| S | Sub-transition matrix. |
| R | Reward matrix. |

Value

Simulated sample.

rMIPHstar	<i>Simulate a MIPH* random vector</i>
-----------	---------------------------------------

Description

Generates a sample of size n from a MIPH* distribution with parameters alpha, S and R.

Usage

rMIPHstar(n, alpha, S, R, gfun, gfun_par)

Arguments

- | | |
|----------|---------------------------------------|
| n | Sample size. |
| alpha | Initial probabilities. |
| S | Sub-intensity matrix. |
| R | Reward matrix. |
| gfun | Vector with transformations names. |
| gfun_par | List with transformations parameters. |

Value

The simulated sample.

rMPHstar	<i>Simulate a MPH* random vector</i>
----------	--------------------------------------

Description

Generates a sample of size n from a MPH* distribution with parameters alpha, S and R.

Usage

```
rMPHstar(n, alpha, S, R)
```

Arguments

n	Sample size.
alpha	Initial probabilities.
S	Sub-intensity matrix.
R	Reward matrix.

Value

The simulated sample.

rphasetype	<i>Simulate phase-type</i>
------------	----------------------------

Description

Generates a sample of size n from a phase-type distribution with parameters alpha and S.

Usage

```
rphasetype(n, alpha, S)
```

Arguments

n	Sample size.
alpha	Vector of initial probabilities.
S	Sub-intensity matrix.

Value

Simulated sample.

runge_kutta	<i>Runge-Kutta for the calculation of the a and b vectors and the c matrix in a EM step</i>
-------------	---

Description

Performs the Runge-Kutta method of fourth order.

Usage

```
runge_kutta(avector, bvector, cmatrix, dt, h, S, s)
```

Arguments

- | | |
|---------|-----------------------|
| avector | The a vector. |
| bvector | The b vector. |
| cmatrix | The c matrix. |
| dt | The increment. |
| h | Step-length. |
| S | Sub-intensity matrix. |
| s | Exit rates. |

show,bivdph-method	<i>Show method for bivariate discrete phase-type distributions</i>
--------------------	--

Description

Show method for bivariate discrete phase-type distributions

Usage

```
## S4 method for signature 'bivdph'  
show(object)
```

Arguments

- | | |
|--------|---|
| object | An object of class bivdph . |
|--------|---|

show,biviph-method	<i>Show method for bivariate inhomogeneous phase-type distributions</i>
--------------------	---

Description

Show method for bivariate inhomogeneous phase-type distributions

Usage

```
## S4 method for signature 'biviph'  
show(object)
```

Arguments

object	An object of class biviph .
--------	---

show,bivph-method	<i>Show method for bivariate phase-type distributions</i>
-------------------	---

Description

Show method for bivariate phase-type distributions

Usage

```
## S4 method for signature 'bivph'  
show(object)
```

Arguments

object	An object of class bivph .
--------	--

show,dph-method	<i>Show method for discrete phase-type distributions</i>
-----------------	--

Description

Show method for discrete phase-type distributions

Usage

```
## S4 method for signature 'dph'  
show(object)
```

Arguments

object	An object of class dph .
--------	--

show,iph-method	<i>Show method for inhomogeneous phase-type distributions</i>
-----------------	---

Description

Show method for inhomogeneous phase-type distributions

Usage

```
## S4 method for signature 'iph'  
show(object)
```

Arguments

object An object of class [iph](#).

show,mdph-method	<i>Show method for multivariate discrete phase-type distributions</i>
------------------	---

Description

Show method for multivariate discrete phase-type distributions

Usage

```
## S4 method for signature 'mdph'  
show(object)
```

Arguments

object An object of class [mdph](#).

show,miph-method	<i>Show method for multivariate inhomogeneous phase-type distributions</i>
------------------	--

Description

Show method for multivariate inhomogeneous phase-type distributions

Usage

```
## S4 method for signature 'miph'  
show(object)
```

Arguments

object An object of class [miph](#).

show,mph-method	<i>Show method for multivariate phase-type distributions</i>
-----------------	--

Description

Show method for multivariate phase-type distributions

Usage

```
## S4 method for signature 'mph'  
show(object)
```

Arguments

object An object of class [mph](#).

show,MPHstar-method	<i>Show method for multivariate phase-type distributions</i>
---------------------	--

Description

Show method for multivariate phase-type distributions

Usage

```
## S4 method for signature 'MPHstar'
show(object)
```

Arguments

object	An object of class MPHstar .
--------	--

show,ph-method	<i>Show method for phase-type distributions</i>
----------------	---

Description

Show method for phase-type distributions

Usage

```
## S4 method for signature 'ph'
show(object)
```

Arguments

object	An object of class ph .
--------	---

show,sph-method	<i>Show method for survival phase-type objects</i>
-----------------	--

Description

Show method for survival phase-type objects

Usage

```
## S4 method for signature 'sph'
show(object)
```

Arguments

object	An object of class sph .
--------	--

sim	<i>New generic for simulating matrix distributions</i>
-----	--

Description

Methods are available for objects of class [ph](#).

Usage

```
sim(x, ...)
```

Arguments

x	An object of the model class.
...	Further parameters to be passed on.

Value

A realization from the matrix distribution.

sim,bivdph-method	<i>Simulation method for bivariate discrete phase-type distributions</i>
-------------------	--

Description

Simulation method for bivariate discrete phase-type distributions

Usage

```
## S4 method for signature 'bivdph'
sim(x, n = 1000)
```

Arguments

x	An object of class bivdph .
n	An integer of length of realization.

Value

A realization of independent and identically distributed bivariate discrete phase-type vector.

Examples

```
obj <- bivdph(dimensions = c(3, 3))
sim(obj, n = 100)
```

sim,bivph-method	<i>Simulation method for bivariate inhomogeneous phase-type distributions</i>
------------------	---

Description

Simulation method for bivariate inhomogeneous phase-type distributions

Usage

```
## S4 method for signature 'biviph'
sim(x, n = 1000)
```

Arguments

x	An object of class biviph .
n	An integer of length of realization.

Value

A realization of independent and identically distributed bivariate inhomogeneous phase-type vector.

Examples

```
under_bivph <- bivph(dimensions = c(3, 3))
obj <- bivph(under_bivph, gfun = c("weibull", "pareto"), gfun_pars = list(c(2), c(3)))
sim(obj, n = 100)
```

sim,bivph-method	<i>Simulation method for bivariate phase-type distributions</i>
------------------	---

Description

Simulation method for bivariate phase-type distributions

Usage

```
## S4 method for signature 'bivph'
sim(x, n = 1000)
```

Arguments

x	An object of class bivph .
n	An integer of length of realization.

Value

A realization of independent and identically distributed bivariate phase-type vector.

Examples

```
obj <- bivph(dimensions = c(3, 3))
sim(obj, n = 100)
```

 sim,dph-method

Simulation method for phase-type distributions

Description

Simulation method for phase-type distributions

Usage

```
## S4 method for signature 'dph'
sim(x, n = 1000)
```

Arguments

x	An object of class dph .
n	An integer of length of realization.

Value

A realization of independent and identically distributed discrete phase-type variables.

Examples

```
obj <- dph(structure = "general")
sim(obj, n = 100)
```

 sim,iph-method

Simulation method for inhomogeneous phase-type distributions

Description

Simulation method for inhomogeneous phase-type distributions

Usage

```
## S4 method for signature 'iph'
sim(x, n = 1000)
```

Arguments

x An object of class `iph`.

n An integer of length of realization.

Value

A realization of independent and identically distributed inhomogeneous phase-type variables.

Examples

```
obj <- iph(ph(structure = "general"), gfun = "lognormal", gfun_pars = 2)
sim(obj, n = 100)
```

sim,mdph-method	<i>Simulation method for multivariate discrete phase-type distributions</i>
-----------------	---

Description

Simulation method for multivariate discrete phase-type distributions

Usage

```
## S4 method for signature 'mdph'
sim(x, n = 1000, equal_marginals = 0)
```

Arguments

x An object of class `mdph`.

n Length of realization.

equal_marginals Non-negative integer. If positive, it specifies the number of marginals to simulate from, all from the first matrix.

Value

A realization of a multivariate discrete phase-type distribution.

Examples

```
obj <- mdph(structure = c("general", "general"))
sim(obj, 100)
```

sim,miph-method	<i>Simulation method for inhomogeneous multivariate phase-type distributions</i>
-----------------	--

Description

Simulation method for inhomogeneous multivariate phase-type distributions

Usage

```
## S4 method for signature 'miph'
sim(x, n = 1000)
```

Arguments

x	An object of class miph .
n	An integer of length of realization.

Value

A realization of independent and identically distributed inhomogeneous multivariate phase-type variables. If x is a MoE miph an array of dimension c(n,d,m) is returned, with d the number of marginals and m the number of initial distribution vectors.

Examples

```
under_mph <- mph(structure = c("general", "general"))
obj <- miph(under_mph, gfun = c("weibull", "pareto"), gfun_pars = list(c(2), c(3)))
sim(obj, 100)
```

sim,mph-method	<i>Simulation method for multivariate phase-type distributions</i>
----------------	--

Description

Simulation method for multivariate phase-type distributions

Usage

```
## S4 method for signature 'mph'
sim(x, n = 1000, equal_marginals = 0)
```

Arguments

`x` An object of class `mph`.

`n` Length of realization.

`equal_marginals` Non-negative integer. If positive, it specifies the number of marginals to simulate from, all from the first matrix.

Value

A realization of a multivariate phase-type distribution.

Examples

```
obj <- mph(structure = c("general", "general"))
sim(obj, 100)
```

sim,MPHstar-method *Simulation method for multivariate phase-type distributions*

Description

Simulation method for multivariate phase-type distributions

Usage

```
## S4 method for signature 'MPHstar'
sim(x, n = 1000)
```

Arguments

`x` An object of class `MPHstar`.

`n` Desired sample size for each marginal.

Value

A matrix of sample data for each marginal.

Examples

```
obj <- MPHstar(structure = "general")
sim(obj, 100)
```

sim,ph-method

Simulation method for phase-type distributions

Description

Simulation method for phase-type distributions

Usage

```
## S4 method for signature 'ph'
sim(x, n = 1000)
```

Arguments

x An object of class [ph](#).
n An integer of length of realization.

Value

A realization of independent and identically distributed phase-type variables.

Examples

```
obj <- ph(structure = "general")
sim(obj, n = 100)
```

sph

Constructor function for survival phase-type objects

Description

Constructor function for survival phase-type objects

Usage

```
sph(x = NULL, coefs = list(B = numeric(0), C = numeric(0)), type = "reg")
```

Arguments

x An object of class [ph](#).
coefs Coefficients of the survival regression object.
type Type of survival object.

Value

An object of class [sph](#).

sph-class	<i>Survival analysis for phase-type distributions</i>
-----------	---

Description

Class of objects for inhomogeneous phase-type distributions

Value

Class object

Slots

coefs Coefficients of the survival regression object.

type Type of survival object.

sum_dph	<i>Computes the initial distribution and sub-intensity of the sum of two discrete phase-type distributed random variables</i>
---------	---

Description

Computes the initial distribution and sub-intensity of the sum of two discrete phase-type distributed random variables

Usage

```
sum_dph(alpha1, S1, alpha2, S2)
```

Arguments

alpha1	Initial distribution.
S1	Sub-transition matrix.
alpha2	Initial distribution.
S2	Sub-transition matrix.

sum_ph	<i>Computes the initial distribution and sub-intensity of the sum of two phase-type distributed random variables.</i>
--------	---

Description

Computes the initial distribution and sub-intensity of the sum of two phase-type distributed random variables.

Usage

```
sum_ph(alpha1, S1, alpha2, S2)
```

Arguments

alpha1	Initial distribution.
S1	Sub-intensity matrix.
alpha2	Initial distribution.
S2	Sub-intensity matrix.

TVR	<i>New generic for transformation via rewards of a matrix distribution</i>
-----	--

Description

Methods are available for objects of class [ph](#)

Usage

```
TVR(x, ...)
```

Arguments

x	An object of the model class.
...	Further parameters to be passed on.

Value

An object of the model class.

TVR,dph-method	<i>TVR Method for dph Class</i>
----------------	---------------------------------

Description

TVR Method for dph Class

Usage

```
## S4 method for signature 'dph'
TVR(x, rew)
```

Arguments

- x An object of class [dph](#).
- rew A vector of rewards.

Value

An object of the of class [dph](#).

Examples

```
obj <- dph(structure = "general")
TVR(obj, c(1, 0, 1))
```

TVR,ph-method	<i>TVR method for ph class</i>
---------------	--------------------------------

Description

TVR method for ph class

Usage

```
## S4 method for signature 'ph'
TVR(x, rew)
```

Arguments

- x An object of class [ph](#).
- rew A vector of rewards.

Value

An object of the of class [ph](#).

Examples

```
obj <- ph(structure = "general")
TVR(obj, c(1, 2, 3))
```

`tvr_dph`*Performs TVR for discrete phase-type distributions*

Description

Performs TVR for discrete phase-type distributions

Usage

```
tvr_dph(alpha, S, R)
```

Arguments

<code>alpha</code>	Initial distribution vector.
<code>S</code>	Sub-intensity matrix.
<code>R</code>	Reward vector.

Value

A list of PH parameters.

`tvr_ph`*Performs TVR for phase-type distributions*

Description

Performs TVR for phase-type distributions

Usage

```
tvr_ph(alpha, S, R)
```

Arguments

<code>alpha</code>	Initial distribution vector.
<code>S</code>	Sub-intensity matrix.
<code>R</code>	Reward vector.

Value

A list of phase-type parameters.

var,bivdph-method	<i>Var method for bivdph class</i>
-------------------	------------------------------------

Description

Var method for bivdph class

Usage

```
## S4 method for signature 'bivdph'
var(x)
```

Arguments

x An object of class [bivdph](#).

Value

The covariance matrix of the bivariate discrete phase-type distribution.

Examples

```
obj <- bivdph(dimensions = c(3, 3))
var(obj)
```

var,bivph-method	<i>Var method for bivph class</i>
------------------	-----------------------------------

Description

Var method for bivph class

Usage

```
## S4 method for signature 'bivph'
var(x)
```

Arguments

x An object of class [bivph](#).

Value

The covariance matrix of the bivariate phase-type distribution.

Examples

```
obj <- bivph(dimensions = c(3, 3))
var(obj)
```

var , dph-method	<i>Var method for discrete phase-type distributions</i>
------------------	---

Description

Var method for discrete phase-type distributions

Usage

```
## S4 method for signature 'dph'  
var(x)
```

Arguments

x An object of class [dph](#).

Value

The variance of the [dph](#) object.

Examples

```
set.seed(123)  
obj <- dph(structure = "general", dimension = 3)  
var(obj)
```

var , mdph-method	<i>Var method for multivariate discrete phase-type distributions</i>
-------------------	--

Description

Var method for multivariate discrete phase-type distributions

Usage

```
## S4 method for signature 'mdph'  
var(x)
```

Arguments

x An object of class [mdph](#).

Value

The covariance matrix of the multivariate discrete phase-type distribution.

Examples

```
obj <- mdph(structure = c("general", "general"))
var(obj)
```

var, mph-method	<i>Var method for multivariate phase-type distributions</i>
-----------------	---

Description

Var method for multivariate phase-type distributions

Usage

```
## S4 method for signature 'mph'
var(x)
```

Arguments

x An object of class [mph](#).

Value

The covariance matrix of the multivariate phase-type distribution.

Examples

```
obj <- mph(structure = c("general", "general"))
var(obj)
```

var, MPHstar-method	<i>Var method for MPHstar class</i>
---------------------	-------------------------------------

Description

Var method for MPHstar class

Usage

```
## S4 method for signature 'MPHstar'
var(x)
```

Arguments

x An object of class [MPHstar](#).

Value

The covariance matrix of the MPHstar distribution.

Examples

```
obj <- MPHstar(structure = "general")
var(obj)
```

var,ph-method	<i>Var method for phase-type distributions</i>
---------------	--

Description

Var method for phase-type distributions

Usage

```
## S4 method for signature 'ph'
var(x)
```

Arguments

x An object of class [ph](#).

Value

The variance of the [ph](#) (or underlying [ph](#)) object.

Examples

```
set.seed(123)
obj <- ph(structure = "general", dimension = 3)
var(obj)
```

vector_of_matrices	<i>Computes the elements $S^n / n!$ until the a given size</i>
--------------------	---

Description

Computes the elements $S^n / n!$ until the a given size

Usage

```
vector_of_matrices(vect, S, a, vect_size)
```

Arguments

vect	A vector.
S	Sub-intensity matrix.
a	A number.
vect_size	Size of vector.

vector_of_matrices_2	<i>Computes the elements $S^n / n!$ until given value of n</i>
----------------------	--

Description

Computes the elements $S^n / n!$ until given value of n

Usage

vector_of_matrices_2(vect, S, vect_size)

Arguments

vect	A vector.
S	Sub-intensity matrix.
vect_size	Size of vector.

vector_of_powers	<i>Computes elements A^n until the given size</i>
------------------	--

Description

Computes elements A^n until the given size

Usage

vector_of_powers(A, vect_size)

Arguments

A	A matrix.
vect_size	Size of vector.

Index

- * **matrixdist**
 - matrixdist-package, 9
- +, dph, dph-method, 10
- +, ph, ph-method, 11
- a_rungekutta, 11
- bivdph, 12, 12, 22, 26, 34, 55, 135, 147, 166, 171, 183, 198, 203, 214
- bivdph-class, 12
- bivdph_density, 13
- bivdph_tail, 13
- biviph, 14, 15, 23, 34, 136, 199, 204
- biviph-class, 15
- biviph, 14, 15, 16, 23, 27, 35, 56, 72, 137, 148, 172, 199, 204, 214
- biviph-class, 16
- biviph_density, 16
- biviph_laplace, 17
- biviph_tail, 17
- cdf, 18
- cdf, dph-method, 18
- cdf, iph-method, 19
- cdf, miph-method, 19
- cdf, mph-method, 20
- cdf, ph-method, 21
- clone_matrix, 21
- clone_vector, 22
- coef, bivdph-method, 22
- coef, biviph-method, 23
- coef, biviph-method, 23
- coef, dph-method, 24
- coef, iph-method, 24
- coef, mdph-method, 25
- coef, ph-method, 25
- coef, sph-method, 26
- cor, bivdph-method, 26
- cor, biviph-method, 27
- cor, mdph-method, 27
- cor, mph-method, 28
- cor, MPHstar-method, 29
- csph_density, 29
- csph_density_par, 30
- csph_density_test, 30
- csph_densityrk, 31
- cumulate_matrix, 31
- cumulate_vector, 32
- data_aggregation, 32
- default_step_length, 33
- dens, 33
- dens, bivdph-method, 34
- dens, biviph-method, 34
- dens, biviph-method, 35
- dens, dph-method, 35
- dens, iph-method, 36
- dens, mdph-method, 36
- dens, miph-method, 37
- dens, mph-method, 38
- dens, ph-method, 38
- dph, 10, 18, 24, 35, 39, 39, 57, 136, 137, 144, 148, 152, 158, 161, 167, 172, 181–183, 199, 205, 212, 215
- dph-class, 40
- dph_pgf, 40
- dphcdf, 41
- dphdensity, 41
- EM_step_mPH_rc, 42
- embedded_mc, 42
- embedded_mc_cs, 43
- EMstep_bivdph, 43
- EMstep_bivdph_MoE, 44
- EMstep_bivph, 44
- EMstep_dph, 45
- EMstep_dph_MoE, 45
- EMstep_mdph, 46
- EMstep_mdph_MoE, 46
- EMstep_MoE_PADE, 47

- EMstep_PADE, [47](#)
- EMstep_RK, [48](#)
- EMstep_UNI, [49](#)
- EMstep_UNI_intervalCensoring, [49](#)
- evaluate, [50](#)
- evaluate, sph-method, [51](#)
- expm_terms, [51](#)
- expmat, [52](#)
- find_n, [52](#)
- find_weight, [53](#)
- Fisher, [53](#)
- Fisher, sph-method, [54](#)
- fit, [54](#)
- fit, bivdph-method, [55](#)
- fit, bivph-method, [55](#)
- fit, dph-method, [56](#)
- fit, mdph-method, [57](#)
- fit, mph-method, [58](#)
- fit, MPHstar-method, [59](#)
- fit, ph-method, [60](#)
- fit_multiple, [62](#)
- haz, [64](#)
- haz, ph-method, [64](#)
- inf_norm, [65](#)
- inhomogeneity_reg, [65](#)
- inhomogeneity_reg, ph-method, [66](#)
- initial_state, [68](#)
- iph, [19](#), [24](#), [36](#), [68](#), [69](#), [136](#), [138](#), [144](#), [145](#), [158](#), [160](#), [200](#), [206](#)
- iph-class, [69](#)
- laplace, [70](#)
- laplace, bivph-method, [70](#)
- laplace, mph-method, [71](#)
- laplace, ph-method, [71](#)
- linCom, [72](#)
- linCom, bivph-method, [72](#)
- linCom, MPHstar-method, [73](#)
- linear_combination, [73](#)
- logLik, ph-method, [74](#)
- logLikelihood_UNIs_PI, [75](#)
- logLikelihoodbivDPH, [76](#)
- logLikelihoodbivDPH_MoE, [76](#)
- logLikelihoodbivPH, [77](#)
- logLikelihoodDPH, [77](#)
- logLikelihoodDPH_MoE, [78](#)
- logLikelihoodmDPH, [78](#)
- logLikelihoodmDPH_MoE, [79](#)
- logLikelihoodMgev_PADE, [79](#)
- logLikelihoodMgev_RK, [80](#)
- logLikelihoodMgev_UNI, [80](#)
- logLikelihoodMgev_UNI_inhom, [81](#)
- logLikelihoodMgev_UNI_inhom_intCens, [81](#)
- logLikelihoodMgev_UNI_intervalCensoring, [82](#)
- logLikelihoodMgompertz_PADE, [83](#)
- logLikelihoodMgompertz_PADEs, [83](#)
- logLikelihoodMgompertz_RK, [84](#)
- logLikelihoodMgompertz_RKs, [85](#)
- logLikelihoodMgompertz_UNI, [86](#)
- logLikelihoodMgompertz_UNI_inhom, [86](#)
- logLikelihoodMgompertz_UNI_inhom_intCens, [87](#)
- logLikelihoodMgompertz_UNI_intervalCensoring, [88](#)
- logLikelihoodMgompertz_UNIs, [88](#)
- logLikelihoodMgompertz_UNIs_inhom, [89](#)
- logLikelihoodMgompertz_UNIs_inhom_intCens, [90](#)
- logLikelihoodMgompertz_UNIs_intervalCensoring, [91](#)
- logLikelihoodMloglogistic_PADE, [92](#)
- logLikelihoodMloglogistic_PADEs, [92](#)
- logLikelihoodMloglogistic_RK, [93](#)
- logLikelihoodMloglogistic_RKs, [94](#)
- logLikelihoodMloglogistic_UNI, [95](#)
- logLikelihoodMloglogistic_UNI_inhom, [95](#)
- logLikelihoodMloglogistic_UNI_inhom_intCens, [96](#)
- logLikelihoodMloglogistic_UNI_intervalCensoring, [97](#)
- logLikelihoodMloglogistic_UNIs, [97](#)
- logLikelihoodMloglogistic_UNIs_inhom, [98](#)
- logLikelihoodMloglogistic_UNIs_inhom_intCens, [99](#)
- logLikelihoodMloglogistic_UNIs_intervalCensoring, [100](#)
- logLikelihoodMlognormal_PADE, [101](#)
- logLikelihoodMlognormal_PADEs, [101](#)
- logLikelihoodMlognormal_RK, [102](#)
- logLikelihoodMlognormal_RKs, [103](#)

- logLikelihoodMlognormal_UNI, [104](#)
- logLikelihoodMlognormal_UNI_inhom, [104](#)
- logLikelihoodMlognormal_UNI_inhom_intCens, [105](#)
- logLikelihoodMlognormal_UNI_intervalCensoring, [106](#)
- logLikelihoodMlognormal_UNIs, [106](#)
- logLikelihoodMlognormal_UNIs_inhom, [107](#)
- logLikelihoodMlognormal_UNIs_inhom_intCens, [108](#)
- logLikelihoodMlognormal_UNIs_intervalCensoring, [109](#)
- logLikelihoodMpareto_PADE, [110](#)
- logLikelihoodMpareto_PADEs, [110](#)
- logLikelihoodMpareto_RK, [111](#)
- logLikelihoodMpareto_RKs, [112](#)
- logLikelihoodMpareto_UNI, [113](#)
- logLikelihoodMpareto_UNI_inhom, [113](#)
- logLikelihoodMpareto_UNI_inhom_intCens, [114](#)
- logLikelihoodMpareto_UNI_intervalCensoring, [115](#)
- logLikelihoodMpareto_UNIs, [115](#)
- logLikelihoodMpareto_UNIs_inhom, [116](#)
- logLikelihoodMpareto_UNIs_inhom_intCens, [117](#)
- logLikelihoodMpareto_UNIs_intervalCensoring, [118](#)
- logLikelihoodMweibull_PADE, [119](#)
- logLikelihoodMweibull_PADEs, [119](#)
- logLikelihoodMweibull_RK, [120](#)
- logLikelihoodMweibull_RKs, [121](#)
- logLikelihoodMweibull_UNI, [122](#)
- logLikelihoodMweibull_UNI_inhom, [122](#)
- logLikelihoodMweibull_UNI_inhom_intCens, [123](#)
- logLikelihoodMweibull_UNI_intervalCensoring, [124](#)
- logLikelihoodMweibull_UNIs, [124](#)
- logLikelihoodMweibull_UNIs_inhom, [125](#)
- logLikelihoodMweibull_UNIs_inhom_intCens, [126](#)
- logLikelihoodMweibull_UNIs_intervalCensoring, [127](#)
- logLikelihoodPH_MoE, [128](#)
- logLikelihoodPH_PADE, [128](#)
- logLikelihoodPH_PADEs, [129](#)
- logLikelihoodPH_RK, [130](#)
- logLikelihoodPH_RKs, [130](#)
- logLikelihoodPH_UNI, [131](#)
- logLikelihoodPH_UNI_intervalCensoring, [131](#)
- logLikelihoodPH_UNIs, [132](#)
- logLikelihoodPH_UNIs_intervalCensoring, [133](#)
- LRT, [133](#)
- LRT, ph, ph-method, [134](#)
- m_exp_sum, [134](#)
- marginal, [135](#)
- marginal, bivdph-method, [135](#)
- marginal, biviph-method, [136](#)
- marginal, bivph-method, [137](#)
- marginal, mdph-method, [137](#)
- marginal, mph-method, [138](#)
- marginal, MPHstar-method, [139](#)
- marginal_expectation, [140](#)
- matrix_exponential, [140](#)
- matrix_inverse, [141](#)
- matrix_power, [141](#)
- matrix_product, [142](#)
- matrix_vanloan, [142](#)
- matrixdist (matrixdist-package), [9](#)
- matrixdist-package, [9](#)
- max_diagonal, [143](#)
- maximum, [143](#)
- maximum, dph, dph-method, [144](#)
- maximum, iph, iph-method, [144](#)
- maximum, ph, ph-method, [145](#)
- mdph, [25](#), [28](#), [37](#), [58](#), [137](#), [146](#), [146](#), [149](#), [168](#), [173](#), [184](#), [200](#), [206](#), [215](#)
- mdph-class, [146](#)
- mdphdensity, [147](#)
- mean, bivdph-method, [147](#)
- mean, bivph-method, [148](#)
- mean, dph-method, [148](#)
- mean, mdph-method, [149](#)
- mean, mph-method, [149](#)
- mean, MPHstar-method, [150](#)
- mean, ph-method, [150](#)
- merge_matrices, [151](#)
- merlang_dph, [152](#)
- merlang_ph, [152](#)
- mgevcd, [153](#)
- mgevden, [153](#)

- mgf, [154](#)
- mgf,bivph-method, [154](#)
- mgf,mph-method, [155](#)
- mgf,ph-method, [155](#)
- mgompertzcdf, [156](#)
- mgompertzden, [156](#)
- minimum, [157](#)
- minimum,dph,dph-method, [157](#)
- minimum,iph,iph-method, [158](#)
- minimum,ph,ph-method, [159](#)
- miph, [20](#), [37](#), [138](#), [159](#), [201](#), [207](#)
- miph-class, [160](#)
- mixture, [161](#)
- mixture,dph,dph-method, [161](#)
- mixture,ph,ph-method, [162](#)
- mloglogisticcdf, [163](#)
- mloglogisticden, [163](#)
- mlognormalcdf, [164](#)
- mlognormalden, [164](#)
- MoE, [165](#)
- MoE,bivdph-method, [165](#)
- MoE,dph-method, [166](#)
- MoE,mdph-method, [167](#)
- MoE,mph-method, [168](#)
- MoE,ph-method, [169](#)
- moment, [171](#)
- moment,bivdph-method, [171](#)
- moment,bivph-method, [172](#)
- moment,dph-method, [172](#)
- moment,mdph-method, [173](#)
- moment,mph-method, [173](#)
- moment,ph-method, [174](#)
- mparetocdf, [175](#)
- mparetoden, [175](#)
- mph, [20](#), [28](#), [38](#), [58](#), [70](#), [71](#), [139](#), [149](#), [154](#), [155](#), [160](#), [169](#), [173](#), [176](#), [176](#), [201](#), [208](#), [216](#)
- mph-class, [176](#)
- MPHstar, [29](#), [59](#), [60](#), [73](#), [139](#), [150](#), [177](#), [177](#), [202](#), [208](#), [216](#)
- MPHstar-class, [178](#)
- MPHstar_data_aggregation, [178](#)
- MPHstar_EMstep_UNI, [179](#)
- mweibullcdf, [179](#)
- mweibullden, [180](#)
- n_pos, [180](#)
- new_state, [181](#)
- Nfold, [181](#)
- Nfold,dph-method, [182](#)
- pgf, [182](#)
- pgf,bivdph-method, [183](#)
- pgf,dph-method, [183](#)
- pgf,mdph-method, [184](#)
- ph, [11](#), [18](#), [21](#), [26](#), [33](#), [38](#), [51](#), [54](#), [61](#), [64–66](#), [69–74](#), [133](#), [134](#), [137](#), [139](#), [143](#), [145](#), [151](#), [152](#), [154–157](#), [159](#), [161](#), [162](#), [165](#), [170](#), [171](#), [174](#), [181](#), [182](#), [184](#), [185](#), [188](#), [189](#), [192](#), [193](#), [202](#), [203](#), [209](#), [211](#), [212](#), [217](#)
- ph-class, [185](#)
- ph_laplace, [186](#)
- phcdf, [186](#)
- phdensity, [187](#)
- plus_states, [187](#)
- pow2_matrix, [188](#)
- quan, [188](#)
- quan,ph-method, [189](#)
- random_reward, [189](#)
- random_structure, [190](#)
- random_structure_bivph, [190](#)
- racsph, [191](#)
- rdphasetype, [191](#)
- reg, [192](#)
- reg,ph-method, [192](#)
- revers_data_trans, [194](#)
- rew_sanity_check, [194](#)
- riph, [195](#)
- rmatrixgev, [195](#)
- rMDPHstar, [196](#)
- rMIPHstar, [196](#)
- rMPHstar, [197](#)
- rphasetype, [197](#)
- runge_kutta, [198](#)
- show,bivdph-method, [198](#)
- show,biviph-method, [199](#)
- show,bivph-method, [199](#)
- show,dph-method, [199](#)
- show,iph-method, [200](#)
- show,mdph-method, [200](#)
- show,miph-method, [201](#)
- show,mph-method, [201](#)
- show,MPHstar-method, [202](#)
- show,ph-method, [202](#)

show, sph-method, [202](#)
sim, [203](#)
sim, bivdph-method, [203](#)
sim, biviph-method, [204](#)
sim, bivph-method, [204](#)
sim, dph-method, [205](#)
sim, iph-method, [205](#)
sim, mdph-method, [206](#)
sim, miph-method, [207](#)
sim, mph-method, [207](#)
sim, MPHstar-method, [208](#)
sim, ph-method, [209](#)
sph, [26](#), [50](#), [51](#), [53](#), [54](#), [67](#), [166–168](#), [170](#), [193](#),
[202](#), [209](#), [209](#)
sph-class, [210](#)
sum_dph, [210](#)
sum_ph, [211](#)

TVR, [211](#)
TVR, dph-method, [212](#)
TVR, ph-method, [212](#)
tvr_dph, [213](#)
tvr_ph, [213](#)

var, bivdph-method, [214](#)
var, bivph-method, [214](#)
var, dph-method, [215](#)
var, mdph-method, [215](#)
var, mph-method, [216](#)
var, MPHstar-method, [216](#)
var, ph-method, [217](#)
vector_of_matrices, [217](#)
vector_of_matrices_2, [218](#)
vector_of_powers, [218](#)