Package 'FitDynMix'

January 20, 2025

Title Estimation of Dynamic Mixtures

Version 1.0.0

Description Estimation of a dynamic lognormal - Generalized Pareto mixture via the Approximate Maximum Likelihood and the Cross-Entropy methods. See Bee, M. (2023) <doi:10.1016/j.csda.2023.107764>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.3

Depends R (>= 2.10)

LazyData true

RdMacros Rdpack

Imports evir, MASS, pracma, Rdpack, parallel, ks, stats, graphics

URL https://github.com/marco-bee/FitDynMix

BugReports https://github.com/marco-bee/FitDynMix/issues

NeedsCompilation no

Author Marco Bee [aut, cre] (<https://orcid.org/0000-0002-9579-3650>)

Maintainer Marco Bee <marco.bee@unitn.it>

Repository CRAN

Date/Publication 2024-01-11 13:00:03 UTC

Contents

AMLEfit 2
AMLEmode 3
CENoisyFit 4
CENoisyFitBoot 6
cvm_stat_M 7
ddyn 8
dynloglik 8
dynloglikMC 9

Metro2019 10
MLEBoot 10
MLEfit 11
nConst_MC 12
rDynMix 13

Index **14**

`AMLEfit` *Estimating a dynamic mixture via AMLE*

Description

This function fits a dynamic mixture via Approximate Maximum Likelihood. Currently only implemented for the lognormal - generalized Pareto case, with Cauchy or exponential weight. The bootstrap estimation of the standard errors of the MLEs (used for finding the supports of the uniform priors) is carried out via parallel computing.

Usage

```
AMLEfit(yObs, epsilon, k, bootreps, intTol = 1e-04, weight)
```

Arguments

`yObs`	numerical vector: observed sample.
`epsilon`	non-negative scalar: scale parameter of the Markov kernel.
`k`	non-negative integer: number of samples generated in the AMLE approach, such that k*epsilon = ABC sample size.
`bootreps`	positive integer: number of bootstrap replications.
`intTol`	non-negative scalar: threshold for stopping the computation of the integral in the normalization constant: if the integral on the interval from n-1 to n is smaller than intTol, the approximation procedure stops.
`weight`	'cau' or 'exp': name of weight distribution.

Details

For the lognormal and GPD parameters, the support of the uniform prior is set equal to the 99% confidence interval of the bootstrap distribution after discarding the outliers. For the Cauchy parameters, the support is given by the range of the bootstrap distribution after discarding the outliers. Be aware that computing times are large when k and/or bootreps are large.

Value

A list with the following elements:

AMLEpars a list of four 6 or 5-dimensional vectors: approximate maximum likelihood estimates computed via sample mean, maxima of the marginal kernel densities, maximum of the multivariate kernel densities, maximum of the product of the marginal kernel densities.

ABCsam ((k x epsilon) x nc) matrix: ABC sample, where nc is 6 or 5, according to the weight.

MLEpars (np x 1) vector: maximum likelihood estimates and maximized log-likelihood, where np is 7 or 6, according to the weight.

MLEboot (bootreps x nc) matrix: maximum likelihood estimates obtained in each bootstrap replication. nc is 6 or 5, according to the weight.

References

Bee M (2023). "Unsupervised mixture estimation via approximate maximum likelihood based on the Cramér - von Mises distance." *Computational Statistics & Data Analysis*, **185**, 107764.

See Also

AMLEmode.

Examples

```
k <- 5000
epsilon <- .02
bootreps <- 2
res = AMLEfit(Metro2019, epsilon, k, bootreps, , 'cau')
```

`AMLEmode` *Approximating the mode of a multivariate empirical distribution*

Description

This function approximates the mode of a multivariate empirical distribution by means of:

1. the sample mean
2. the product of the maxima of the univariate kernel densities estimated using the marginals
3. the maximum of the multivariate kernel density
4. the maximum of the product of the univariate kernel densities Typically used in connection with AMLEfit (see AMLEfit for examples).

Usage

```
AMLEmode(ABCsam)
```

Arguments

ABCsam	(m x k) matrix: ABC sample, where m is the ABC sample size and k is the number of parameters.

Details

The bandwidth is estimated via smoothed cross-validation

Value

A list containing the 4 approximate modes.

CENoisyFit *Cross-Entropy estimation*

Description

This function estimates a dynamic mixture by means of the noisy Cross-Entropy method. Currently only implemented for the lognormal - generalized Pareto case, with Cauchy or exponential weight. This is mainly an auxiliary function, not suitable for the final user. Use CeNoisyFitBoot instead.

Usage

```
CENoisyFit(
  x,
  rawdata,
  rho,
  maxiter,
  alpha,
  nsim,
  nrepsInt,
  xiInst,
  betaInst,
  eps,
  r = 5,
  weight
)
```

Arguments

x	list: sequence of integers 1,...,K, where K is the mumber of datasets. Set x = 1 in case of a single dataset.
rawdata	either a list of vectors or a vector: in the former case, each vector contains a dataset to be used for estimation.
rho	real in (0,1): parameter determining the quantile of the log-likelihood values to be used at each iteration.

maxiter	non-negative integer: maximum number of iterations.
alpha	real in (0,1): smoothing parameter.
nsim	non-negative integer: number of replications used in the normal and lognormal updating.
nrepsInt	non-negative integer: number of replications used in the Monte Carlo estimate of the normalizing constant.
xiInst	non-negative real: shape parameter of the instrumental GPD.
betaInst	non-negative real: scale parameter of the instrumental GPD.
eps	non-negative real: tolerance for the stopping criterion of the noisy Cross-Entropy method.
r	positive integer: length of window to be used in the stopping criterion.
weight	'cau' or 'exp': name of weight distribution.

Details

See Rubinstein and Kroese (2004, chap. 6).

Value

For each dataset, a list with the following elements is returned:

V (nreps x 12) matrix: updated mean and variance of the distributions used in the stochastic program. nit (positive integer): number of iterations needed for convergence. loglik (scalar): maximized log-likelihood.

References

Rubinstein RY, Kroese DP (2004). *The Cross-Entropy Method*. Springer.

See Also

CENoisyFitBoot

Examples

```
maxiter = 10
alpha = .5
rho = .05
eps = 1e-2
nsim = 1000
nrepsInt = 1000
xiInst = 3
betaInst = 3
r = 5
res <- CENoisyFit(1,Metro2019,rho,maxiter,alpha,nsim,nrepsInt,xiInst,betaInst,eps,r,'exp')
```

CENoisyFitBoot *Cross-Entropy estimation and bootstrap standard errors*

Description

This function estimates a dynamic mixture by means of the noisy Cross-Entropy method and computes bootstrap standard errors. Currently only implemented for the lognormal - generalized Pareto, with Cauchy or exponential weight. Bootstrap standard errors are computed in parallel.

Usage

```
CENoisyFitBoot(
  yObs,
  nboot,
  rho,
  maxiter,
  alpha,
  nsim,
  nrepsInt,
  xiInst,
  betaInst,
  eps,
  r = 5,
  weight
)
```

Arguments

yObs	numerical vector: observed random sample from the mixture.
nboot	integer: number of bootstrap replications for computing the standard errors. If nboot = 0, no standard errors are computed.
rho	real in (0,1): parameter determining the quantile of the log-likelihood values to be used at each iteration.
maxiter	non-negative integer: maximum number of iterations.
alpha	real in (0,1): smoothing parameter.
nsim	non-negative integer: number of replications used in the normal and lognormal updating.
nrepsInt	non-negative integer: number of replications used in the Monte Carlo estimate of the normalizing constant.
xiInst	non-negative real: shape parameter of the instrumental GPD.
betaInst	non-negative real: scale parameter of the instrumental GPD.
eps	non-negative real: tolerance for the stopping criterion of the noisy Cross-Entropy method.
r	positive integer: length of window to be used in the stopping criterion.
weight	'cau' or 'exp': name of weight distribution.

Value

If nboot > 0, a list with the following elements:

estPars: Cross-Entropy estimates.

nit: number of iterations needed for convergence.

loglik: maximized log-likelihood.

bootPars: parameter estimates obtained for each bootstrap sample.

stddev: bootstrap standard errors.

If nboot = 0, only estPars, nit and loglik are returned.

Examples

```
res = CENoisyFitBoot(Metro2019,0,.05,20,.5,500,500,3,3,.01,5,'exp')
```

`cvm_stat_M` *Computing the Cramér - von Mises distance between two samples*

Description

This function Computing the Cramér - von Mises distance between two samples.

Usage

```
cvm_stat_M(vec1, vec2, power)
```

Arguments

`vec1`	(n x 1) vector: first sample.
`vec2`	(n x 1) vector: second sample.
`power`	power of the integrand. Equal to 2 when using the L2 distance

Details

This function computes the Cramér - von Mises distance between two samples. See Drovandi and Frazier (2022, p. 7).

Value

out scalar: Cramér - von Mises distance between the two samples

References

Drovandi C, Frazier DT (2022). "A comparison of likelihood-free methods with and without summary statistics." *Statistics and Computing*, **32**, Paper No. 42.

Examples

```
out = cvm_stat_M(runif(100),rnorm(100),2)
```

ddyn *Density of a Lognormal-GPD dynamic mixture*

Description

This function evaluates the density of a Lognormal-GPD dynamic mixture, with Cauchy or exponential weight.

Usage

```
ddyn(x, pars, intTol, weight)
```

Arguments

x	non-negative vector: points where the density is evaluated.
pars	numerical vector: if weight is equal to 'cau', values of μ_c, τ, μ, σ, ξ, β; if weight is equal to 'exp', values of λ, μ, σ, ξ, β.
intTol	non-negative scalar: threshold for stopping the computation of the integral in the normalization constant: if the integral on the interval from n-1 to n is smaller than intTol, the approximation procedure stops.
weight	'cau' or 'exp': name of weight distribution.

Value

density of the lognormal-GPD mixture evaluated at x.

Examples

```
x <- seq(0,20,length.out=1000)
pars <- c(1,2,0,.5,.25,3.5)
dLNPar <- ddyn(x,pars,1e-04,'cau')
```

dynloglik *Log-likelihood of a Lognormal-GPD dynamic mixture*

Description

This function evaluates the log-likelihood of a Lognormal-GPD dynamic mixture, computing the integral in the normalizing constant via quadrature methods.

Usage

```
dynloglik(x, y, intTol, weight)
```

Arguments

x	if weight is equal to 'cau', (6 by 1) numerical vector: values of μ_c, τ, μ, σ, ξ, β; if weight is equal to 'exp', (5 by 1) numerical vector: values of λ, μ, σ, ξ, β.
y	vector: points where the function is evaluated.
intTol	non-negative scalar: threshold for stopping the computation of the integral in the normalization constant: if the integral on the interval from n-1 to n is smaller than intTol, the approximation procedure stops.
weight	'cau' or 'exp': name of weight distribution.

Value

log-likelihood of the lognormal-GPD mixture evaluated at y.

Examples

```
x <- c(1,2,0,.5,.25,3.5)
y <- rDynMix(100,x,'cau')
fit <- dynloglik(x,y,1e-04,'cau')
```

---

dynloglikMC *Log-likelihood of a Lognormal-GPD dynamic mixture*

Description

This function evaluates the log-likelihood of a Lognormal-GPD dynamic mixture, with Cauchy or exponential weight, approximating the normalizing constant via Monte Carlo simulation.

Usage

```
dynloglikMC(x, y, nreps, xiInst, betaInst, weight)
```

Arguments

x	if weight is equal to 'cau', (6 by 1) numerical vector: values of μ_c, τ, μ, σ, ξ, β; if weight is equal to 'exp', (5 by 1) numerical vector: values of λ, μ, σ, ξ, β.
y	vector: points where the function is evaluated.
nreps	non-negative integer: number of replications to be used in the computation of the integral in the normalizing constant.
xiInst	non-negative real: shape parameter of the instrumental GPD.
betaInst	non-negative real: scale parameter of the instrumental GPD.
weight	'cau' or 'exp': name of weight distribution.

Value

Log-likelihood of the lognormal-GPD mixture evaluated at y.

Examples

```
llik <- dynloglikMC(c(1,2,0,1,.25,3.5),Metro2019,10000,3,3,'exp')
```

`Metro2019` *Population estimate of the US metropolitan areas*

Description

A dataset cotaining the 2019 population estimate, divided by 10000, of the 415 US metropolitan areas computed by the US Census Bureau.

Usage

```
Metro2019
```

Format

A numerical vector with 415 rows and 1 column.

Source

https://www.census.gov

`MLEBoot` *Bootstrap standard errors of MLEs*

Description

This function creates bootstrap samples of input data and fits a dynamic mixture via maximum likelihood.

Usage

```
MLEBoot(x, y, intTol, weight)
```

Arguments

`x`	list of integers: indices of replications.
`y`	numerical vector: observed data.
`intTol`	threshold for stopping the computation of the integral in the normalization constant: if the integral on the interval from n-1 to n is smaller than intTol, the approximation procedure stops.
`weight`	'cau' or 'exp': name of weight distribution.

Details

MLEs are computed by means of the optim function. When it breaks down, the sample is discarded and a new one is generated. The function keeps track of the number of times this happens.

Value

A list with the following elements:

MLE: maximum likelihood estimates obtained from each bootstrap sample.

errors: number of times the MLE algorithm breaks down.

Examples

```
bootMLEs <- MLEBoot(1,Metro2019,1e-04,'exp')
```

`MLEfit` *Estimating a dynamic mixture via MLE*

Description

This function fits a dynamic mixture via standard maximum likelihood. Currently only implemented for the lognormal - generalized Pareto case, with Cauchy or exponential weight.

Usage

```
MLEfit(yObs, bootreps, intTol = 1e-04, weight)
```

Arguments

`yObs`	numerical vector: observed sample.
`bootreps`	non-negative integer: number of bootstrap replications. If equal to 0, no standard errors are computed.
`intTol`	non-negative scalar: threshold for stopping the computation of the integral in the normalization constant: if the integral on the interval from n-1 to n is smaller than intTol, the approximation procedure stops.
`weight`	'cau' or 'exp': name of weight distribution.

Details

Starting values for mu and sigma are the lognormal MLEs computed with the observations below the median. Initial values for xi and tau are the GPD MLEs obtained with the observations above the median. For the location and scale parameter of the Cauchy, we respectively use the first quartile and abs(log(sd(x)/2)). For the parameter of the exponential, we use abs(log(sd(x)/2)).

Value

MLEpars vector: maximum likelihood estimates and maximized log-likelihood.

MLEboot matrix: maximum likelihood estimates obtained in each bootstrap replication.

sdMLE vector: bootstrap standard deviation of the MLEs.

References

Bee M (2023). "Unsupervised mixture estimation via approximate maximum likelihood based on the Cramér - von Mises distance." *Computational Statistics & Data Analysis*, **185**, 107764.

See Also

AMLEfit

Examples

```
mixFit <- MLEfit(Metro2019,0,,'cau')
```

`nConst_MC` *Monte carlo approximation of the normalizing constant of a Lognormal-GPD dynamic mixture*

Description

This function evaluates via Monte Carlo simulation the normalizing constant of a Lognormal-GPD dynamic mixture, with Cauchy or exponential weight.

Usage

```
nConst_MC(x, nreps, xiInst = 3, betaInst = 3, weight)
```

Arguments

`x`	if weight is equal to 'cau', (6 by 1) numerical vector: values of μ_c, τ, μ, σ, ξ, β; if weight is equal to 'exp', (5 by 1) numerical vector: values of λ, μ, σ, ξ, β.
`nreps`	number of replications to be used in the computation of the integral in the normalizing constant.
`xiInst`	real: value of the shape parameter of the instrumental density. Default value equal to 3.
`betaInst`	non-negative real: value of the scale parameter of the instrumental density. Default value equal to 3.
`weight`	'cau' or 'exp': name of weight distribution.

Value

Normalizing constant of the density of the lognormal-GPD mixture.

Examples

```
nconst <- nConst_MC(c(1,2,0,0.5,0.25,3.5), 10000, 3, 3, 'cau')
```

---

rDynMix *Simulating a dynamic lognormal-Pareto mixture*

Description

This function fits a dynamic mixture by Approximate Maximum Likelihood and by standard maximum likelihood. Currently only implemented for the lognormal - generalized Pareto case, with Cauchy or exponential weight.

Usage

```
rDynMix(nreps, x, weight)
```

Arguments

nreps	integer: number of observations sampled from the mixture.
x	numerical vector: if weight = 'cau', values of mu_c, τ, μ, σ, ξ, β; if weight = 'exp', values of λ, μ, σ, ξ, β.
weight	'cau' or 'exp': name of weight distribution.

Details

This function simulates a dynamic lognormal-GPD mixture using the algorithm of Frigessi et al. (2002, p. 221).

Value

ysim (nreps x 1) vector: nreps random numbers from the lognormal-GPD dynamic mixture.

References

Frigessi A, Haug O, Rue H (2002). "A Dynamic Mixture Model for Unsupervised Tail Estimation without Threshold Selection." *Extremes*, **3**(5), 219–235.

Examples

```
ysim <- rDynMix(100,c(1,2,0,0.5,0.25,3),'cau')
```

Index

∗ **Cross-Entropy;**
CENoisyFitBoot, 6
∗ **MLE;**
MLEBoot, 10
∗ **approximate**
AMLEfit, 2
AMLEmode, 3
∗ **bootstrap.**
CENoisyFitBoot, 6
MLEBoot, 10
∗ **cross-entropy.**
CENoisyFit, 4
∗ **datasets**
Metro2019, 10
∗ **dynamic**
AMLEfit, 2
AMLEmode, 3
CENoisyFit, 4
CENoisyFitBoot, 6
cvm_stat_M, 7
ddyn, 8
dynloglik, 8
dynloglikMC, 9
MLEBoot, 10
MLEfit, 11
nConst_MC, 12
rDynMix, 13
∗ **likelihood.**
AMLEfit, 2
AMLEmode, 3
MLEfit, 11
∗ **maximum**
AMLEfit, 2
AMLEmode, 3
MLEfit, 11
∗ **mixture.**
ddyn, 8
dynloglik, 8
dynloglikMC, 9
nConst_MC, 12
∗ **mixture;**
AMLEfit, 2
AMLEmode, 3
CENoisyFit, 4
CENoisyFitBoot, 6
cvm_stat_M, 7
MLEBoot, 10
MLEfit, 11
rDynMix, 13
∗ **non-parametric**
CENoisyFitBoot, 6
MLEBoot, 10
∗ **simulation.**
cvm_stat_M, 7
rDynMix, 13

AMLEfit, 2, *12*
AMLEmode, *3*, 3

CENoisyFit, 4
CENoisyFitBoot, 6
cvm_stat_M, 7

ddyn, 8
dynloglik, 8
dynloglikMC, 9

Metro2019, 10
MLEBoot, 10
MLEfit, 11

nConst_MC, 12

rDynMix, 13