Package 'LMfilteR'

May 7, 2026

Type Package

Title Filter Methods for Parameter Estimation in Linear and Non Linear Regression Models

Version 0.1.3.1

Description We present a method based on filtering algorithms to estimate the parameters of linear, i.e. the coefficients and the variance of the error term. The proposed algorithms make use of Particle Filters following Ristic, B., Arulampalam, S., Gordon, N. (2004, ISBN: 158053631X) resampling methods. Parameters of logistic regression models are also estimated using an evolutionary particle filter method.

License GPL (>= 2)

Imports MASS (>= 7.3-50), stats (>= 3.5.1)

Depends R (>= 3.6.0)

Encoding UTF-8

BugReports https://github.com/ChrissCod/LMfilteR/issues

RoxygenNote 7.2.3

NeedsCompilation no

Author Christian Llano Robayo [aut, cre],
Nazrul Shaikh [aut],
Pegu Nilutpal [aut]

Maintainer Christian Llano Robayo <info@cecareus.com>

Repository CRAN

Date/Publication 2023-02-02 09:10:06 UTC

Contents

`EPF_logist_compl` *Parameter Estimation Of Logistic Linear Models Using Evolutionary Particle Filters (EPF) Method*

Description

Estimation of the parameters of a logistic regression using a particle filter method that includes two evolutionary algorithm-based steps. See *Details*

Usage

```
EPF_logist_compl(
  Data,
  Y,
  nPart = 1000L,
  p_mut = 0.01,
  p_cross = 0.5,
  thr = 0.5,
  lmbd = 3,
  count = 10,
  initDisPar,
  resample_met = syst_rsmpl
)
```

Arguments

`Data`	matrix. The matrix containing the independent variables of the linear model
`Y`	numeric. The dependent variable
`nPart`	integer. The number of particles, by default 1000L
`p_mut`	numeric. The mutation probability in EPF, by default 0.01
`p_cross`	numeric. The cross-over probability in EPF, by default 0.5
`thr`	numeric. The threshold after which the estimated Y is classified as 1
`lmbd`	numeric. A number to be added and subtracted from the priors when `initDisPar` is not provided, by default 3
`count`	numeric. The number of replications within EPF, by default 10
`initDisPar`	matrix. Values a, b of the uniform distribution (via `runif`) for each parameter to be estimated, see more in *Details*
`resample_met`	function. The resampling method used in EPF, by default `syst_rsmpl`, see *Details*

Details

Estimation of the coefficients of a logistic linear regression using the evolutionary particle filter. EPF includes evolutionary algorithms (mutation and cross-over) within PF-base algorithm to avoid degeneracy of particles and prevent the curse of dimensionality. In mutation, `p_mut * nPart` particles are selected at random from the set of particles obtained in `k-1` PF-iteration. This particles will be replaced by fresh new particles taken from a uniform distribution. In cross-over, a pair of random particles is selected from the `k-1` propagated particles. The pair is combined into one particle (using the mean) and the result replaces a particle selected at random from the `k-1` propagated particles. This cross-over process is replicated `prob_cross * N` times on each iteration. `EPF_logist_compl` uses all the information to estimate the parameters.

The state-space equations of the logistic regression model are:

$(Eq.1) X_k = a_0 + a_1 * X1_{k-1} + ... + a_n * Xn_{k-1}$

$(Eq.2) Y_k = 1/(1 + exp(-X_k))$

$(Eq.3) Z_k \sim Ber(Y_k)$

where, k = 1, ... , number of observations; $a_0, ..., a_n$ are the parameters to be estimated (coefficients), and $Ber(\cdot)$ is the Bernoulli distribution.

The priors of the parameters are assumed uniformly distributed. In `initDisPar`, the number of rows is the number of independent variables plus one since the constant term of the regression is also estimated. The first and second column of `initDisPar` are the corresponding arguments `a` and `b` of the uniform distribution (`stats::runif`) for each parameter prior. The first row of `initDisPar` is the prior guess of the constant term. The following rows are the prior guesses of the coefficients. If `initDisPar` is missing, the initial priors are taken using `glm()` and `coeff()` plus-minus `lmbd` as a reference.

For `resample_met`, several resampling methods are used following Li, T., Bolic, M., & Djuric, P. M. (2015). Resampling methods for particle filtering: classification, implementation, and strategies. IEEE Signal processing magazine, 32(3), 70-86. Currently available resampling methods are: `syst_rsmpl` - systematic resampling (default), `multin_rsmpl` - multinomial resampling, `strat_rsmpl` - stratified resampling, and `simp_rsmpl` - simple resampling (similar to PF_lm_ss)

Value

A list with the following elements: `smmry`: A summary matrix of the estimated parameters; first column is the GLM estimation, and second column is the EPF estimation. `AIC`: Akaike information criteria of the EPF estimation. `Yhat`: Estimation of the observation Y. `sttp`: A matrix with the last iteration parameters-particles. `estm`: A vector with the final parameter EPF estimation. `c_time`: The time in seconds it takes to complete EPF.

Author(s)

Christian Llano Robayo, Nazrul Shaikh.

Examples

```
## Not run:
#Simulating logistic regression model
val_coef <- c(2, -1.25, 2.6, -0.7, -1.8)
Data1 <-  MASS::mvrnorm(100, mu = rep(0, 4),
```

```
                      Sigma = diag(4),
                      empirical = TRUE)
Y1 <- colSums(t(cbind(1,Data1))*val_coef)
prb <- 1 / (1 + exp(-Y1))
Y_ <- rbinom(100, size = 1, prob = prb) ##
#Run EPF
Res <- EPF_logist_compl(Data = Data1, Y = Y_)
#Summary EPF estimation
Res$smmry

## End(Not run)
```

`EPF_L_compl` *Parameter Estimation Of Linear Models Using Evolutionary Particle Filters (EPF) Method*

Description

Estimation of the parameters of a linear regression model using a particle filter method that includes two evolutionary algorithm-based steps. See *Details*

Usage

```
EPF_L_compl(
  Data,
  Y,
  nPart = 1000L,
  p_mut = 0.01,
  p_cross = 0.5,
  sigma_l = 1,
  lmbd = 3,
  count = 10,
  initDisPar,
  resample_met = syst_rsmpl
)
```

Arguments

`Data`	matrix. The matrix containing the independent variables of the linear model
`Y`	numeric. The dependent variable
`nPart`	integer. The number of particles, by default 1000L
`p_mut`	numeric. The mutation probability in EPF, by default 0.01
`p_cross`	numeric. The cross-over probability in EPF, by default 0.5
`sigma_l`	The value of the variance in the likelihood normal, 1 by default
`lmbd`	numeric. A number to be added and subtracted from the priors when `initDisPar` is not provided, by default 3

count	numeric. The number of replications within EPF, by default 10
initDisPar	matrix. Values a, b of the uniform distribution (via runif) for each parameter to be estimated, see more in *Details*
resample_met	function. The resampling method used in EPF, by default syst_rsmpl, see *Details*

Details

Estimation of the coefficients of a linear regression:

$Y = \beta_0 + \beta_1 * X_1 + ... + \beta_n * X_n + \epsilon, (\epsilon \sim N(0, 1))$

using Evolutionary Particle Filter. EPF includes additional steps within PF-base algorithm to avoid degeneracy of particles and prevent the curse of dimensionality. One step is the inclusion of two evolutionary algorithm - based processes: mutation and cross-over. In mutation, p_mut * nPart particles are selected at random from the set of particles obtained in k-1 PF-iteration. This particles will be replaced by fresh new particles taken from a uniform distribution. In cross-over, a pair of random particles is selected from the k-1 propagated particles. The pair is combined into one particle (using the mean function) and the result replaces a particle selected at random from the k-1 propagated particles. This cross-over process is replicated prob_cross * N times on each iteration. EPF_L_compl uses all the information to estimate the parameters, differs from function EPF_L_MovH by using movH equals nrows(Data)

Similarly as in PF_lm and PF_lm_ss, the state-space equations of the linear model are:

$(Eq.1) X_k = a_0 + a_1 * X1_{k-1} + ... + a_n * Xn_{k-1}$

$(Eq.2) Y_k = X_k + \epsilon, \epsilon \sim N(0, \sigma)$

where, k = 2, ... , number of observations; $a_0, ..., a_n$ are the parameters to be estimated (coefficients), and $\sigma = 1$.

The priors of the parameters are assumed uniformly distributed. initDisPar is a matrix; the number of rows is the number of independent variables plus one since the constant term of the regression is also estimated. The first and second column of initDisPar are the corresponding arguments a and b of the uniform distribution (stats::runif) for each parameter prior. The first row of initDisPar is the prior guess of the constant term. The following rows are the prior guesses of the coefficients. If initDisPar is missing, the initial priors are taken using lm() and coeff() plus-minus lmbd as a reference.

For resample_met, several resampling methods are used following Li, T., Bolic, M., & Djuric, P. M. (2015). Resampling methods for particle filtering: classification, implementation, and strategies. IEEE Signal processing magazine, 32(3), 70-86. Currently available resampling methods are: syst_rsmpl - systematic resampling (default), multin_rsmpl - multinomial resampling, strat_rsmpl - stratified resampling, and simp_rsmpl - simple resampling (similar to PF_lm_ss)

Value

A list with the following elements: smmry: A summary matrix of the estimated parameters; first column is the OLS estimation, and second column is the EPF estimation. SSE: The sum of squared errors of the EPF estimation. Yhat: Estimation of the observation Y. sttp: A matrix with the last iteration particles. estm: A vector with the final parameter EPF estimation. c_time: The time in seconds it takes to complete EPF.

Author(s)

Christian Llano Robayo, Nazrul Shaikh.

Examples

```
## Not run:
#Example using linear model with 5 variables
#Using rcorrmatrix from cluterGeneration to generate a random correlation matrix
realval_coef <- c(2, -1.25, 2.6, -0.7, -1.8, 0.6)
Data1 <-  MASS::mvrnorm(100, mu = rep(0, 5),
                        Sigma = clusterGeneration::rcorrmatrix(5),
                        empirical = TRUE)
eps <- stats::rnorm(100) # error term
Y_ <- colSums(t(cbind(1,Data1)) * realval_coef) + eps
#run EPF
Res1 <- EPF_L_compl(Data = Data1, Y = Y_)
#Summary EPF estimation
Res1$smmry

## End(Not run)
```

`PF_lm` *Parameter Estimation Of Linear Regression Using Particle Filters*

Description

Estimation of the coefficients of a linear regression based on the particle filters algorithm (PF); given `Data1`, the function estimates the coefficients of the regression as the samples (particles) of coefficients that are more likely to occur. Optional, the standard deviation of the error term can also be estimated. Synthetic data is generated in case no provided `Data1`.

Usage

```
PF_lm(Y, Data1, n = 500L, sigma_l = 1, sigma_est = FALSE, initDisPar, lbd = 2)
```

Arguments

`Y`	numeric. The response variable
`Data1`	matrix. The matrix containing the independent variables
`n`	integer. Number of particles, by default 500
`sigma_l`	The value of the variance in the likelihood normal, 1 by default
`sigma_est`	logical. If `TRUE` takes the last row of `initDisPar` as prior estimation of the standard deviation, see more in *Details*
`initDisPar`	matrix. Values a, b of the uniform distribution (via `runif`) for each parameter to be estimated, see more in *Details*
`lbd`	numeric. A number to be added and substracted from the priors when `initDisPar` is not provided

Details

Estimation of the coefficients of a linear regression:

$Y = \beta_0 + \beta_1 * X_1 + ... + \beta_n * X_n + \epsilon$, $(\epsilon \sim N(0,1))$

using particle filter methods. The implementation follows An, D., Choi, J. H., Kim, N. H. (2013) adapted for the case of linear models.

The state-space equations are:

$(Eq.1) X_k = a_0 + a_1 * X1_{k-1} + ... + a_n * Xn_{k-1}$

$(Eq.2) Y_k = X_k + \epsilon, \epsilon \sim N(0, \sigma)$

where, k = 2, ... , number of observations; $a_0, ..., a_n$ are the parameters to be estimated (coefficients), and $\sigma = 1$.

The priors of the parameters are assumed uniformly distributed. `initDisPar` is a matrix for which the number of rows is the number of independent variables plus one when `sigma_est` = `FALSE`, (plus one since we also estimate the constant term of the regression), or plus two when `sigma_est` = `TRUE` (one for the constant term of the regression, and one for the estimation of sigma). The first and second column of `initDisPar` are the corresponding arguments `a` and `b` of the uniform distribution (`stats::runif`) of each parameter prior. The first row of `initDisPar` is the prior guess of the constant term. The following rows are the prior guesses of the coefficients. When sigma is estimated, i.e., if `sigma_est` = `TRUE` the last row of `initDisPar` corresponds to the prior guess for the standard deviation. If `sigma_est` = `FALSE`, then the standard deviation of the likelihood is `sigma_l`. If `sigma_est` = `TRUE`, the algorithm estimates the standard deviation of ϵ together with the coefficients. If `initDisPar` is missing, the initial priors are taken using `lm()` and `coeff()` plus-minus `lbd` as a reference.

The cumulative density function method is used for resampling.

In case no `Data1` is provided, a synthetic data set is generated automatically taking three normal i.i.d. variables, and the dependent variable is computed as in $Y = 2 + 1.25 * X_1 + 2.6 * X_2 - 0.7 * X_3 + \epsilon$

Value

A list with the following elements:

`stateP_res`: A list of matrices with the PF estimation of the parameters on each observation; the number of rows is the number of observations in `Data1` and the number of columns is `n`.

`Likel`: A matrix with the likelihood of each particle obtained on each observation.

`CDF`: A matrix with the cumulative distribution function for each column of `Likel`.

`summ`: A summary matrix with statistics of the estimated parameters.

Author(s)

Christian Llano Robayo, Nazrul Shaikh.

References

An, D., Choi, J. H., Kim, N. H. (2013). Prognostics 101: A tutorial for particle filter-based prognostics algorithm using Matlab. Reliability Engineering & System Safety, 115, DOI: https://doi.org/10.1016/j.ress.2013.02.019

Ristic, B., Arulampalam, S., Gordon, N. (2004). Beyond the Kalman filter: particle filters for tracking applications. Boston, MA: Artech House. ISBN: 158053631X.

West, M., Harrison, J. (1997). Bayesian forecasting and dynamic models (2nd ed.). New York: Springer. ISBN: 0387947256.

Examples

```
## Not run:
#### Using default Data1, no sigma estimation ####
Res <- PF_lm(n=1000L, sigma_est = FALSE)
lapply(Res,class) # Structure of returning list.
###Summary of estimated parameters
Res$summ
par(mfrow=c(2, 2)) #Evolution of the estimated parameters
for (i in 1:4){
plot(apply(Res$stateP_res[[i]],1,mean), main = colnames(Res$summ)[i], col="blue",
      xlab =  "", ylab = "",type="l")
}


#### Using default Data1, with sigma estimation ####
Res2 <- PF_lm(n=1000L, sigma_est = TRUE)
lapply(Res2,class) # Structure of returning list
sumRes2 <- sapply(2:6, function(i)
###Summary of estimated parameters
Res2$summ

par(mfrow=c(2, 3)) #Evolution of the estimated parameters
for (i in 1:5){
plot(apply(Res2$stateP_res[[i]],1,mean), main = colnames(Res2$summ)[i], col="blue",
       xlab =  "", ylab = "",type="l")
}


#### Using default Data1, given initDisPar ####
b0 <- matrix(c( 1.9, 2, # Prior of a_0
                1, 1.5, # Prior of a_1
                2, 3,   # Prior of a_2
                -1, 0)  # Prior of a_3
                ,ncol = 2, byrow = TRUE )
Res3 <- PF_lm(n=500L, sigma_est = FALSE, initDisPar = b0)
lapply(Res3,class) # Structure of returning list.

###Summary of estimated parameters
Res3$summ
par(mfrow=c(2, 2)) #Evolution of the estimated parameters
for (i in 1:4){
plot(apply(Res3$stateP_res[[i]],1,mean), main = colnames(Res3$summ)[i], col="blue",
      xlab =  "", ylab = "",type="l")
}

## End(Not run)
```

PF_lm_ss *Parameter Estimation Of Linear Regression Using Particle Filters With Simple Sampling*

Description

Estimation of the coefficients of a linear regression based on the particle filters algorithm. This function is similar to PF_lm except for the resampling method which in this case is the simple sampling. As a result, the user can try higher number of particles.

Usage

```
PF_lm_ss(
  Y,
  Data1,
  n = 500L,
  sigma_l = 1,
  sigma_est = FALSE,
  initDisPar,
  lbd = 2
)
```

Arguments

Y	numeric. The response variable
Data1	matrix. The matrix containing the independent variables
n	integer. Number of particles, by default 500
sigma_l	The variance of the normal likelihood, 1 by default
sigma_est	logical. If TRUE takes the last row of initDisPar as prior estimation of the standard deviation, see more in *Details*
initDisPar	matrix. Values a, b of the uniform distribution (via runif) for each parameter to be estimated, see more in *Details*
lbd	numeric. A number to be added and substracted from the priors when initDisPar is not provided

Details

Estimation of the coefficients of a linear regression:

$Y = \beta_0 + \beta_1 * X_1 + ... + \beta_n * X_n + \epsilon$, $(\epsilon \sim N(0, 1))$

using particle filter methods. The state-space equations are:

$(Eq.1) X_k = a_0 + a_1 * X1_{k-1} + ... + a_n * Xn_{k-1}$

$(Eq.2) Y_k = X_k + \epsilon, \epsilon \sim N(0, \sigma)$

where, k = 2, ... , number of observations; $a_0, ..., a_n$ are the parameters to be estimated (coefficients), and $\sigma = 1$.

The priors of the parameters are assumed uniformly distributed. `initDisPar` is a matrix for which the number of rows is the number of independent variables plus one when `sigma_est` = `FALSE`, (plus one since we also estimate the constant term of the regression), or plus two when `sigma_est` = `TRUE` (one for the constant term of the regression, and one for the estimation of sigma). The first and second column of `initDisPar` are the corresponding arguments `a` and `b` of the uniform distribution (`stats::runif`) of each parameter prior. The first row `initDisPar` is the prior guess of the constant term. The following rows are the prior guesses of the coefficients. When sigma is estimated, i.e., if `sigma_est` = `TRUE` the last row of `initDisPar` corresponds to the prior guess for the standard deviation. If `sigma_est` = `FALSE`, then the standard deviation of the likelihood is `sigma_l`. If `sigma_est` = `TRUE`, the algorithm estimates the standard deviation of ϵ together with the coefficients. If `initDisPar` is missing, the initial priors are taken using `lm()` and `coeff()` plus-minus `lbd` as a reference.

The resampling method used corresponds to the simple sampling, i.e., we take a sample of `n` particles with probability equals the likelihood computed on each iteration. In addition, on each iteration white noise is added to avoid particles to degenerate.

In case no `Data1` is provided, a synthetic data set is generated automatically taking three normal i.i.d. variables, and the dependent variable is computed as in $Y = 2 + 1.25 * X_1 + 2.6 * X_2 - 0.7 * X_3 + \epsilon$

Value

A list containing the following elemnts:

`stateP_res`: A list of matrices with the PF estimation of the parameters on each observation; the number of rows is the number of observations in `Data1` and the number of columns is `n`.

`Likel`: A matrix with the likelihood of each particle obtained on each observation.

Author(s)

Christian Llano Robayo, Nazrul Shaikh.

References

Ristic, B., Arulampalam, S., Gordon, N. (2004). Beyond the Kalman filter: particle filters for tracking applications. Boston, MA: Artech House. ISBN: 158053631X.

West, M., Harrison, J. (1997). Bayesian forecasting and dynamic models (2nd ed.). New York: Springer. ISBN: 0387947256.

Examples

```
## Not run:
#### Using default Data1, no sigma estimation ####
Res <- PF_lm_ss(n = 10000L, sigma_est = FALSE) #10 times more than in PF_lm
lapply(Res,class) # Structure of returning list.
###Summary of estimated parameters
Res$summ
#Evolution of the estimated parameters
```

```
par(mfrow=c(2, 2))
for (i in 1:4){
plot(apply(Res$stateP_res[[i]],1,mean), main = colnames(Res$summ)[i], col="blue",
      xlab =  "", ylab = "",type="l")
}


#### Using default Data1, with sigma estimation ####
Res2 <- PF_lm_ss(n = 1000L, sigma_est = TRUE)
lapply(Res2,class) # Structure of returning list
###Summary of the estimated parameters
Res2$summ
#Evolution of the estimated parameters
par(mfrow=c(2, 3))
for (i in 1:5){
plot(apply(Res2$stateP_res[[i]],1,mean), main = colnames(Res2$summ)[i], col="blue",
       xlab =  "", ylab = "",type="l")
}


#### Using default Data1, given initDisPar ####
b0 <- matrix(c(1.9, 2, # Prior of a_0
               1, 1.5, # Prior of a_1
               2, 3,   # Prior of a_2
               -1, 0),  # Prior of a_3
               ncol = 2, byrow = TRUE )
Res3 <- PF_lm_ss(n = 10000L, sigma_est = FALSE, initDisPar = b0)
lapply(Res3,class) # Structure of returning list.
###Summary of the estimated parameters
Res3$summ
#Evolution of the estimated parameters
par(mfrow=c(2, 2))
for (i in 1:4){
plot(apply(Res3$stateP_res[[i]],1,mean), main = colnames(Res3$summ)[i], col="blue",
    xlab =  "", ylab = "",type="l")
    }

## End(Not run)
```

Index