# Package 'LMfilteR'

August 14, 2018

**Type** Package

**Title** Filter Methods for Parameter Estimation in Linear Regression Models

**Version** 0.1.2

**Description** We present a method based on filtering algorithms to estimate the parameters of linear regressions, 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.

**License** GPL (>= 2)

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

**Depends** R (>= 3.5.0)

**Encoding** UTF-8

**LazyData** true

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

**RoxygenNote** 6.0.1

**NeedsCompilation** no

**Author** Christian Llano Robayo [aut, cre], Nazrul Shaikh [aut]

**Maintainer** Christian Llano Robayo <cxl985@miami.edu>

**Repository** CRAN

**Date/Publication** 2018-08-14 04:20:25 UTC

## R topics documented:

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(Data1, n = 500L, sigma_l = 1, sigma_est = FALSE, initDisPar)
```

## Arguments

| | |
|---|---|
| `Data1` | matrix. A matrix containing the dependent and independent variables. The first column is the dependent variable. |
| `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*. |

## 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 independient variables plus two when `sigma_est = FALSE`, (one for the state and one for the constant of the regression), or plus three when `sigma_est = TRUE` (one for the state, one for the constant 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 of the state equation. The second row is the prior guess of the regression interecept term. The following rows are the prior guesses of the coefficients, or when is estimated, of sigma, i.e., if

`sigma_est = TRUE` the last row of `initDisPar` is the uniform prior of the standard deviation. If `sigma_est = FALSE`, then standard deviation of the likelihood is `sigma_l`. If `sigma_est = TRUE`, the algorithm estimates the standard deviation of $\epsilon$ together with the coefficients. If `initDisPar` is missing, the initial priors are taken using `lm()` and `coeff()` plus-minus 1 as a reference.

The CDF 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`.

## 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.
sumRes <- sapply(2:5, function(i)
summary(apply(Res$stateP_res[[i]],1,mean))) # Summary of estimated parameters
colnames(sumRes) <-  c("a0", "a1", "a2", "a3")
sumRes
par(mfrow=c(2, 2)) #Evolution of the mean of the estimation on each time
for (i in 2:5){
plot(apply(Res$stateP_res[[i]],1,mean), main = colnames(sumRes)[i-1], 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(apply(Res2$stateP_res[[i]],1,mean)))# Summary of estimated parameters
colnames(sumRes2) <-  c("a0", "a1", "a2", "a3", "s")
sumRes2

par(mfrow=c(2, 3)) #Evolution of the mean of the particle estimation
for (i in 2:6){
plot(apply(Res2$stateP_res[[i]],1,mean), main = colnames(sumRes2)[i-1], col="blue",
       xlab =  "", ylab = "",type="l")
}


#### Using default Data1, given initDisPar ####
b0 <- matrix(c( -2, 11, # Prior of the state equation
                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=1000L, sigma_est = FALSE, initDisPar = b0)
lapply(Res3,class) # Structure of returning list.
sumRes3 <- sapply(2:5, function(i)
          summary(apply(Res3$stateP_res[[i]],1,mean))) # Summary of estimated parameters
colnames(sumRes3) <-  c("a0", "a1", "a2", "a3")
sumRes3

par(mfrow=c(2, 2)) #Evolution of the mean of the particle estimation
for (i in 2:5){
plot(apply(Res3$stateP_res[[i]],1,mean), main = colnames(sumRes3)[i-1], 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 result, the user can try higher number of particles.

## Usage

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

## Arguments

| | |
|---|---|
| Data1 | matrix. A matrix with the dependent and independent variables. The first column should contain the dependent variable. |
| 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*. |

## 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 independient variables plus two when sigma_est = FALSE, (one for the state and one for the constant of the regression), or plus three when sigma_est = TRUE (one for the state, one for the constant 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 of the state equation. The second row is the prior guess of the regression interecept term. The following rows are the prior guesses of the coefficients, or when is estimated, of sigma, i.e., if sigma_est = TRUE the last row of initDisPar is the uniform prior of the standard deviation. If sigma_est = FALSE, then standard deviation of the likelihood is sigma_l. If sigma_est = TRUE, the algorithm estimates the standard deviation of $\epsilon$ together with the coefficients. If initDisPar is missing, the initial priors are taken using lm() and coeff() plus-minus 1 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.
sumRes <- sapply(2:5, function(i)
summary(apply(Res$stateP_res[[i]],1,mean))) # Summary of estimated parameters
colnames(sumRes) <-  c("a0", "a1", "a2", "a3")
sumRes
par(mfrow=c(2, 2)) #Evolution of the mean of PF estimation on each time
for (i in 2:5){
plot(apply(Res$stateP_res[[i]],1,mean), main = colnames(sumRes)[i-1], col="blue",
      xlab =  "", ylab = "",type="l")
}

dev.off()
#### Using default Data1, with sigma estimation ####
Res2 <- PF_lm_ss(n = 1000L, sigma_est = TRUE)
lapply(Res2,class) # Structure of returning list
sumRes2 <- sapply(2:6, function(i)
  summary(apply(Res2$stateP_res[[i]],1,mean)))# Summary of estimated parameters
colnames(sumRes2) <-  c("a0", "a1", "a2", "a3", "s")
sumRes2

par(mfrow=c(2, 3)) #Evolution of the mean of PF estimation
for (i in 2:6){
plot(apply(Res2$stateP_res[[i]],1,mean), main = colnames(sumRes2)[i-1], col="blue",
       xlab =  "", ylab = "",type="l")
}
dev.off()

#### Using default Data1, given initDisPar ####
b0 <- matrix(c( -2, 11, # Prior of the state equation
                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.
sumRes3 <- sapply(2:5, function(i)
          summary(apply(Res3$stateP_res[[i]],1,mean))) # Summary of estimated parameters
colnames(sumRes3) <-  c("a0", "a1", "a2", "a3")
sumRes3

par(mfrow=c(2, 2)) #Evolution of the mean of PF estimation
for (i in 2:5){
plot(apply(Res3$stateP_res[[i]],1,mean), main = colnames(sumRes3)[i-1], col="blue",
    xlab =  "", ylab = "",type="l")
    }
dev.off()

## End(Not run)
```

# Index