Package ‘SMC’

February 19, 2015

Type Package

Title Sequential Monte Carlo (SMC) Algorithm

Version 1.1

Date 2011-12-09

Author Gopi Goswami <goswami@stat.harvard.edu>

Maintainer Gopi Goswami <grgoswami@gmail.com>

Depends R (>= 1.9.0)

Description particle filtering, auxiliary particle filtering and sequential Monte Carlo algorithms

License GPL (>= 2)

Repository CRAN

Date/Publication 2011-12-11 10:41:18

NeedsCompilation yes

R topics documented:

auxiliaryParticleFilter 1
particleFilter 8
print 15
sequentialMonteCarlo 15
utilsForExamples 21

Index **23**

`auxiliaryParticleFilter` *The auxiliary particle filtering algorithm*

Description

Function for doing auxiliary particle filtering given the *state equation* (via `generateNextStreamsFunc`), the *stream representative* generation rule (via `generateStreamRepsFunc`), and the *observation equation* density (via `logObsDensFunc`).

See the sections *Details*, *Required Functions* and *Optional Functions* for explanation on the arguments and the return values of the *arguments that are themselves functions*.

Usage

```
auxiliaryParticleFilter(nStreams,
                        nPeriods,
                        dimPerPeriod,
                        generateStreamRepsFunc,
                        generateNextStreamsFunc,
                        logObsDensFunc,
                        resampCriterionFunc = NULL,
                        resampFunc          = NULL,
                        summaryFunc         = NULL,
                        nMHSteps            = 0,
                        MHUpdateFunc        = NULL,
                        nStreamsPreResamp   = NULL,
                        returnStreams       = FALSE,
                        returnLogWeights    = FALSE,
                        verboseLevel        = 0,
                        ...)
```

Arguments

`nStreams` integer > 0.

`nPeriods` integer > 0.

`dimPerPeriod` integer > 0.

`generateStreamRepsFunc`
function of five arguments (`currentPeriod, lag1Streams, lag1LogWeights, streamIndices,`

`generateNextStreamsFunc`
function of seven arguments (`currentPeriod, lag1Streams, lag1LogWeights, streamIndices,`

`logObsDensFunc` function of three arguments (`currentPeriod, currentStreams, ...`).

`resampCriterionFunc`
function of four arguments (`currentPeriod, currentStreams, currentLogWeights, ...`).

`resampFunc` function of four arguments (`currentPeriod, currentStreams, currentLogWeights, ...`).

`summaryFunc` function of four arguments (`currentPeriod, currentStreams, currentLogWeights, ...`).

`nMHSteps` integer ≥ 0.

`MHUpdateFunc` function of six arguments (`currentPeriod, nMHSteps, currentStreams, lag1Streams,      lag1`

`nStreamsPreResamp`
integer > 0.

`returnStreams` `logical`.

`returnLogWeights`
`logical`.

`verboseLevel` `integer`, a value ≥ 2 produces a lot of output.

`...` optional arguments to be passed to `generateStreamRepsFunc`, `generateNextStreamsFunc`, `logObsDensFunc`, `resampCriterionFunc`, `resampFunc`, `summaryFunc` and `MHUpdateFunc`.

Details

We introduce the following terms, which will be used in the sections *Required Function* and *Optional Function* below:

`stream` the state vector also called the particle, the hidden state or the latent variable. Below we will use the terms stream and state vector interchangeably.

`dimPerPeriod` the dimension of the space, the state vectors live in.

Value

This function returns a list with the following components:

`draws` a list with the following components: `summary`, `propUniqueStreamIds`, `streams`, `logWeights`, `acceptanceRates`. See the section *Note* for more details.

`nStreams` the `nStreams` argument.

`nPeriods` the `nPeriods` argument.

`dimPerPeriod` the `dimPerPeriod` argument.

`nStreamsPreResamp`
the `nStreamsPreResamp` argument.

`nMHSteps` the `nMHSteps` argument.

`filterType` type of the filter: “auxiliaryParticleFilter”.

`time` the time taken by the run.

Required function: generateStreamRepsFunc

Arguments: The following argument(s) require some explanation:

`lag1Streams` a matrix of dimension `nStreams` × `dimPerPeriod` of streams for `currentPeriod - 1`.

`lag1LogWeights` a vector of length `nStreams` of log weights corresponding to the streams in the argument matrix `lag1Streams`.

`streamIndices` a vector of length `nStreams` for which the stream representatives (μ_t^k of Pitt and Shephard, 1999) for `currentPeriod` are to be generated. See the sub-section *Note:* below.

Return value: a matrix of dimension `nStreamIndices` × `dimPerPeriod`. The rows of this matrix contain the stream representative for period `currentPeriod`, given the state vectors to be found in the `streamIndices` rows of the argument `lag1Streams` matrix. Here `nStreamIndices` is the length of the argument `streamIndices`.

Note: The following points are in order:

- this function *should* distinguish the cases `currentPeriod == 1` and `currentPeriod > 1` inside of it.
- for details on the stream representatives (i.e., μ_t^k), see of Pitt and Shephard, 1999. The quantity μ_t^k could be the mean, the mode, a draw or some other likely value associated with the state density for period `currentPeriod` (i.e., $f(\alpha_t \mid \alpha_{t-1})$).
- this function is called by setting `streamIndices` to `1:nStreams`, i.e., stream representatives for all the streams in the argument `lag1Streams` matrix is generated.

Optional function: generateNextStreamsFunc

Arguments: The following argument(s) require some explanation:

`lag1Streams` a matrix of dimension `nStreams` × `dimPerPeriod` of streams for `currentPeriod - 1`.

`lag1LogWeights` a vector of length `nStreams` of log weights corresponding to the streams in the argument matrix `lag1Streams`.

`streamIndices` a vector of length ≥ `nStreams` which are to be updated from `currentPeriod - 1` to `currentPeriod`.

`streamReps` a matrix of dimension `nStreams` × `dimPerPeriod` of the stream representatives for `currentPeriod`.

`startingStreams` a matrix of dimension `nStreams` × `dimPerPeriod` to be used for `currentPeriod = 1`. If this is `NULL`, then the function should provide a way to generate streams for `currentPeriod = 1`.

Return value: a matrix of dimension `nStreamIndices` × `dimPerPeriod`. The rows of this matrix contain the state vectors for period `currentPeriod`, given the state vectors to be found in the `streamIndices` rows of the argument `lag1Streams` matrix. Here `nStreamIndices` is the length of the argument `streamIndices`.

Note: The following points are in order:

- this function *should* distinguish the cases `currentPeriod == 1` and `currentPeriod > 1` inside of it.
- this function is called by setting `streamIndices` such that `nStreamIndices` takes either of the two values `nStreams` or `nStreamsPreResamp` in different ocassions.

Optional function: logObsDensFunc

Arguments: The following argument(s) require some explanation:

`currentStreams` a matrix with `dimPerPeriod` columns, the rows containing the streams for `currentPeriod`.

Return value: a vector of length `nCurrentStreams`, where `nCurrentStreams` refers to the number of rows of the `currentStreams` matrix argument. This vector contains the observation equation density values for `currentPeriod` in the log scale, evaluated at the rows of `currentStreams`.

Note: `nCurrentStreams` might be ≥ `nStreams`.

Optional function: resampCriterionFunc

Arguments: The following argument(s) require some explanation:

`currentStreams` a matrix with `dimPerPeriod` columns, the rows containing the updated streams for `currentPeriod`.

`currentLogWeights` a vector of log weights corresponding to the streams in the argument matrix `currentStreams`.

Return value: `TRUE` or `FALSE` reflecting the decision of the resampling scheme implemented by this function.

Note: The following points are in order:

– resampling schemes manily depend on `currentLogWeights`, the other two arguments might come in handy for implementing period or stream specific resampling schemes.

– if `nStreamsPreResamp` > `nStreams`, then this function should always return `TRUE`.

Optional function: resampFunc

Arguments: see the sub-section *Arguments:* for section *Optional function: resampCriterionFunc*.

Return value: a *named* list with the following components:

`currentStreams` a matrix of dimension `nStreams` × `dimPerPeriod`. The rows of this matrix contain the streams for period `currentPeriod + 1` that were resampled from those of the argument `currentStreams` matrix, which may contain ≥ `nStreams` rows.

`currentLogWeights` The log weights vector of length `nStreams`, associated with the streams that were resampled in the returned `currentStreams` matrix. Note, after the resampling step, usually all the log weights are set to 0.

Note: the components of the list returned by this function and the arguments to this function have two common names, namely, `currentStreams` and `currentLogWeights`. These entities have different meanings, as explained above. For example, the argument matrix `currentStreams` could possibly have ≥ `nStreams` rows, whereas the returned `currentStreams` has exactly `nStreams` number of (resampled) streams in its rows.

Optional function: summaryFunc

Arguments: The following argument(s) require some explanation:

`currentStreams` a matrix of dimension `nStreams` × `dimPerPeriod` of streams for `currentPeriod`.

`currentLogWeights` a vector of log weights corresponding to the streams in the argument matrix `currentStreams`.

Return value: a vector of length of `dimSummPerPeriod` of summaries for `currentPeriod` given the `currentStreams` and the `currentLogWeights`.

Optional function: MHUpdateFunc

Arguments: The following argument(s) require some explanation:

nMHSteps the number of Metropolis Hastings (MH) steps (iterations) to be performed.

currentStreams a matrix of dimension nStreams × dimPerPeriod of streams for currentPeriod.

lag1Streams a matrix of dimension nStreams × dimPerPeriod of streams for currentPeriod - 1.

lag1LogWeights a vector of length nStreams of log weights corresponding to the streams in the argument matrix lag1Streams.

Return value: a *named* list with the following components:

currentStreams a matrix of dimension nStreams × dimPerPeriod. The rows of this matrix contain the streams for period currentPeriod that are (possibly) MH-updated versions of the rows of the *argument* currentStreams matrix.

acceptanceRates a vector of length nStreams, representing the acceptance rates of the nMHSteps MH steps for each of the streams in the rows of the argument currentStreams matrix.

Note: a positive value of nMHSteps performs as many MH steps on the rows of the argument currentStreams matrix. This is done to reduce the possible degeneracy after the resampling.

Warning

Using very small values ($\leq$ 1e3) for nStreams might not give reliable results.

Note

The effect of leaving the default value NULL for some of the arguments above are as follows:

resampCriterionFunc the builtin resampling criterion, namely, resample when square of the coefficient of variation of the weights ≥ 1, is used.

resampFunc the builtin resampling function, which resamples streams with probability proportional to their weights, is used.

summaryFunc the builtin summary function, which returns the weighted average of each of the dimPerPeriod dimensions, is used.

MHUpdateFunc *unlike*, particleFilter, there is no builtin Metropolis Hastings updating function, which generates proposals for currentPeriod streams using those of currentPeriod - 1. The user needs to implement this function if nMHSteps > 0.

nStreamsPreResamp it is set to nStreams.

Also, the following point is worth noting:

resampCriterionFunc, resampFunc, summaryFunc are only necessary when user wants to try out new resampling schemes or enhanced summary generation procedures, as part of their research. The default builtins take care of the typical problems.

This function returns a list with component called draw. The detailed description of this component, as promised in section *Value*, is as follows. It is a list itself with the following components:

`summary` a matrix of dimension `nPeriods` × `dimSummPerPeriod`.

`propUniqueStreamIds` a vector of length `nPeriods`. The values are either proportions of unique streams accpeted (at each period) if resampling was done or `NA`.

`streams` an array of dimension `nStreams` × `dimPerPeriod` × `nPeriods`. This is returned if `returnStreams = TRUE`.

`logWeights` a matrix of dimension `nStreams` × `nPeriods`. This is returned if `returnLogWeights = TRUE`.

`acceptanceRates` a matrix of dimension `nStreams` × `nPeriods`. This is returned if `nMHSteps > 0`.

Author(s)

Gopi Goswami <`goswami@stat.harvard.edu`>

References

Michael K. Pitt and Meil Shephard (1999). Filtering via Simulation: Auxiloary Particle Filters. Journal of the American Statistical Association 94(446): 590-599.

See Also

`particleFilter`

Examples

```
MSObj  <- MarkovSwitchingFuncGenerator(-2468)
smcObj <-
    with(MSObj,
     {
         auxiliaryParticleFilter(nStreams                = 5000,
                                 nPeriods                = nrow(yy),
                                 dimPerPeriod            = ncol(yy),
                                 generateStreamRepsFunc  = generateStreamRepsFunc,
                                 generateNextStreamsFunc = generateNextStreamsFunc,
                                 logObsDensFunc          = logObsDensFunc,
                                 returnStreams           = TRUE,
                                 returnLogWeights        = TRUE,
                                 verboseLevel            = 1)
     })
print(smcObj)
print(names(smcObj))
with(c(smcObj, MSObj),
 {
     par(mfcol = c(2, 1))
     plot(as.ts(yy),
          main     = expression('The data and the underlying regimes'),
          cex.main = 0.8,
          xlab     = 'period',
          ylab     = 'data and the regime means',
          cex.lab  = 0.8)
     lines(as.ts(mu), col = 2, lty = 2)
     plot(as.ts(draws$summary[1, ]),
          main     = expression('The underlying regimes and their estimates'),
          cex.main = 0.8,
          xlab     = 'period',
          ylab     = 'regime means',
          cex.lab  = 0.8)
     lines(as.ts(mu), col = 2, lty = 2)
 })

MSObj  <- MarkovSwitchingFuncGenerator(-8642)
smcObj <-
    with(MSObj,
     {
         auxiliaryParticleFilter(nStreams                = 5000,
                                 nPeriods                = nrow(yy),
                                 dimPerPeriod            = ncol(yy),
                                 generateStreamRepsFunc  = generateStreamRepsFunc,
                                 generateNextStreamsFunc = generateNextStreamsFunc,
                                 logObsDensFunc          = logObsDensFunc,
                                 returnStreams           = TRUE,
                                 returnLogWeights        = TRUE,
                                 verboseLevel            = 1)
     })
print(smcObj)
print(names(smcObj))
with(c(smcObj, MSObj),
 {
     par(mfcol = c(2, 1))
     plot(as.ts(yy),
          main     = expression('The data and the underlying regimes'),
          cex.main = 0.8,
          xlab     = 'period',
          ylab     = 'data and the regime means',
          cex.lab  = 0.8)
     lines(as.ts(mu), col = 2, lty = 2)
     plot(as.ts(draws$summary[1, ]),
          main     = expression('The underlying regimes and their estimates'),
          cex.main = 0.8,
          xlab     = 'period',
          ylab     = 'regime means',
          cex.lab  = 0.8)
     lines(as.ts(mu), col = 2, lty = 2)
 })
```

particleFilter *The particle filtering algorithm*

Description

Function for doing particle filtering given the *state equation* (via generateNextStreamFunc), and the *observation equation* density (via logObsDensFunc).

See the sections *Details*, *Required Functions* and *Optional Functions* for explanation on the arguments and the return values of the *arguments that are themselves functions*.

Usage

```
particleFilter(nStreams,
               nPeriods,
               dimPerPeriod,
               generateNextStreamsFunc,
               logObsDensFunc,
               resampCriterionFunc = NULL,
               resampFunc          = NULL,
               summaryFunc         = NULL,
               nMHSteps            = 0,
               MHUpdateFunc        = NULL,
               nStreamsPreResamp   = NULL,
               returnStreams       = FALSE,
               returnLogWeights    = FALSE,
               verboseLevel        = 0,
               ...)
```

Arguments

`nStreams` `integer` > 0.

`nPeriods` `integer` > 0.

`dimPerPeriod` `integer` > 0.

`generateNextStreamsFunc`
function of six arguments (`currentPeriod, lag1Streams, lag1LogWeights, streamIndices,`

`logObsDensFunc` function of three arguments (`currentPeriod, currentStreams, ...`).

`resampCriterionFunc`
function of four arguments (`currentPeriod, currentStreams, currentLogWeights, ...`).

`resampFunc` function of four arguments (`currentPeriod, currentStreams, currentLogWeights, ...`).

`summaryFunc` function of four arguments (`currentPeriod, currentStreams, currentLogWeights, ...`).

`nMHSteps` `integer` ≥ 0.

`MHUpdateFunc` function of six arguments (`currentPeriod, nMHSteps, currentStreams, lag1Streams, lag1`

`nStreamsPreResamp`
`integer` > 0.

`returnStreams` `logical`.

`returnLogWeights`
`logical`.

`verboseLevel` `integer`, a value ≥ 2 produces a lot of output.

`...` optional arguments to be passed to `generateNextStreamsFunc`, `logObsDensFunc`, `resampCriterionFunc`, `resampFunc`, `summaryFunc` and `MHUpdateFunc`.

Details

We introduce the following terms, which will be used in the sections *Required Function* and *Optional Function* below:

`stream` the state vector also called the particle, the hidden state or the latent variable. Below we will use the terms stream and state vector interchangeably.

`dimPerPeriod` the dimension of the space, the state vectors live in.

Value

This function returns a list with the following components:

`draws` a list with the following components: `summary`, `propUniqueStreamIds`, `streams`, `logWeights`, `acceptanceRates`. See the section *Note* for more details.

`nStreams` the `nStreams` argument.

`nPeriods` the `nPeriods` argument.

`dimPerPeriod` the `dimPerPeriod` argument.

`nStreamsPreResamp` the `nStreamsPreResamp` argument.

`nMHSteps` the `nMHSteps` argument.

`filterType` type of the filter: “particleFilter”.

`time` the time taken by the run.

Optional function: generateNextStreamsFunc

Arguments: The following argument(s) require some explanation:

`lag1Streams` a matrix of dimension `nStreams` × `dimPerPeriod` of streams for `currentPeriod - 1`.

`lag1LogWeights` a vector of length `nStreams` of log weights corresponding to the streams in the argument matrix `lag1Streams`.

`streamIndices` a vector of length ≥ `nStreams` which are to be updated from `currentPeriod - 1` to `currentPeriod`.

`startingStreams` a matrix of dimension `nStreams` × `dimPerPeriod` to be used for `currentPeriod = 1`. If this is `NULL`, then the function should provide a way to generate streams for `currentPeriod = 1`.

Return value: a matrix of dimension `nStreamIndices` × `dimPerPeriod`. The rows of this matrix contain the state vectors for period `currentPeriod` given the state vectors to be found in the `streamIndices` rows of the argument `lag1Streams` matrix. Here `nStreamIndices` is the length of the argument `streamIndices`.

Note: this function *should* distinguish the cases `currentPeriod == 1` and `currentPeriod > 1` inside of it.

Optional function: logObsDensFunc

Arguments: The following argument(s) require some explanation:

`currentStreams` a matrix with `dimPerPeriod` columns, the rows containing the streams for `currentPeriod`.

Return value: a vector of length `nCurrentStreams`, where `nCurrentStreams` refers to the number of rows of the `currentStreams` matrix argument. This vector contains the observation equation density values for `currentPeriod` in the log scale, evaluated at the rows of `currentStreams`.

Note: `nCurrentStreams` might be $\geq$ `nStreams`.

Optional function: resampCriterionFunc

Arguments: The following argument(s) require some explanation:

`currentStreams` a matrix with `dimPerPeriod` columns, the rows containing the updated streams for `currentPeriod`.

`currentLogWeights` a vector of log weights corresponding to the streams in the argument matrix `currentStreams`.

Return value: `TRUE` or `FALSE` reflecting the decision of the resampling scheme implemented by this function.

Note: The following points are in order:

- resampling schemes manily depend on `currentLogWeights`, the other two arguments might come in handy for implementing period or stream specific resampling schemes.
- if `nStreamsPreResamp` > `nStreams`, then this function should always return `TRUE`.

Optional function: resampFunc

Arguments: see the sub-section *Arguments:* for section *Optional function: resampCriterionFunc*.

Return value: a *named* list with the following components:

`currentStreams` a matrix of dimension `nStreams` × `dimPerPeriod`. The rows of this matrix contain the streams for period `currentPeriod + 1` that were resampled from those of the argument `currentStreams` matrix, which may contain $\geq$ `nStreams` rows.

`currentLogWeights` The log weights vector of length `nStreams`, associated with the streams that were resampled in the returned `currentStreams` matrix. Note, after the resampling step, usually all the log weights are set to 0.

Note: the components of the list returned by this function and the arguments to this function have two common names, namely, `currentStreams` and `currentLogWeights`. These entities have different meanings, as explained above. For example, the argument matrix `currentStreams` could possibly have $\geq$ `nStreams` rows, whereas the returned `currentStreams` has exactly `nStreams` number of (resampled) streams in its rows.

Optional function: summaryFunc

Arguments: The following argument(s) require some explanation:

- `currentStreams` a matrix of dimension `nStreams` × `dimPerPeriod` of streams for `currentPeriod`.
- `currentLogWeights` a vector of log weights corresponding to the streams in the argument matrix `currentStreams`.

Return value: a vector of length of `dimSummPerPeriod` of summaries for `currentPeriod` given the `currentStreams` and the `currentLogWeights`.

Optional function: MHUpdateFunc

Arguments: The following argument(s) require some explanation:

- `nMHSteps` the number of Metropolis Hastings (MH) steps (iterations) to be performed.
- `currentStreams` a matrix of dimension `nStreams` × `dimPerPeriod` of streams for `currentPeriod`.
- `lag1Streams` a matrix of dimension `nStreams` × `dimPerPeriod` of streams for `currentPeriod - 1`.
- `lag1LogWeights` a vector of length `nStreams` of log weights corresponding to the streams in the argument matrix `lag1Streams`.

Return value: a *named* list with the following components:

- `currentStreams` a matrix of dimension `nStreams` × `dimPerPeriod`. The rows of this matrix contain the streams for period `currentPeriod` that are (possibly) MH-updated versions of the rows of the *argument* `currentStreams` matrix.
- `acceptanceRates` a vector of length `nStreams`, representing the acceptance rates of the `nMHSteps`-many MH steps for each of the streams in the rows of the argument `currentStreams` matrix.

Note: a positive value of `nMHSteps` performs as many MH steps on the rows of the argument `currentStreams` matrix. This is done to reduce the possible degeneracy after the resampling.

Warning

Using very small values ($\leq$ `1e3`) for `nStreams` might not give reliable results.

Note

The effect of leaving the default value `NULL` for some of the arguments above are as follows:

- `resampCriterionFunc` the builtin resampling criterion, namely, resample when square of the coefficient of variation of the weights ≥ 1, is used.
- `resampFunc` the builtin resampling function, which resamples streams with probability proportional to their weights, is used.
- `summaryFunc` the builtin summary function, which returns the weighted average of each of the `dimPerPeriod` dimensions, is used.
- `MHUpdateFunc` the builtin Metropolis Hastings updating function, which generates proposals for `currentPeriod` streams using those of `currentPeriod - 1`, is used.

nStreamsPreResamp it is set to nStreams.

Also, the following point is worth noting:

resampCriterionFunc, resampFunc, summaryFunc **and** MHUpdateFunc are only necessary when user wants to try out new resampling schemes, enhanced summary generation procedures or more efficient MH updating rules, as part of their research. The default builtins take care of the typical problems.

This function returns a list with component called draw. The detailed description of this component, as promised in section *Value*, is as follows. It is a list itself with the following components:

summary a matrix of dimension nPeriods × dimSummPerPeriod.

propUniqueStreamIds a vector of length nPeriods. The values are either proportions of unique stream ids accpeted (at each period) if resampling was done or NA.

streams an array of dimension nStreams × dimPerPeriod × nPeriods. This is returned if returnStreams = TRUE.

logWeights a matrix of dimension nStreams × nPeriods. This is returned if returnLogWeights = TRUE.

acceptanceRates a matrix of dimension nStreams × nPeriods. This is returned if nMHSteps > 0.

Author(s)

Gopi Goswami <goswami@stat.harvard.edu>

References

Jun S. Liu (2001). Monte Carlo strategies for scientific computing. *Springer. Page 66.*

See Also

auxiliaryParticleFilter

Examples

```
MSObj  <- MarkovSwitchingFuncGenerator(-13579)
smcObj <-
    with(MSObj,
     {
         particleFilter(nStreams                = 5000,
                        nPeriods                = nrow(yy),
                        dimPerPeriod            = ncol(yy),
                        generateNextStreamsFunc = generateNextStreamsFunc,
                        logObsDensFunc          = logObsDensFunc,
                        returnStreams           = TRUE,
                        returnLogWeights        = TRUE,
                        verboseLevel            = 1)
     })
print(smcObj)
print(names(smcObj))
with(c(smcObj, MSObj),
 {
     par(mfcol = c(2, 1))
     plot(as.ts(yy),
          main     = expression('The data and the underlying regimes'),
          cex.main = 0.8,
          xlab     = 'period',
          ylab     = 'data and the regime means',
          cex.lab  = 0.8)
     lines(as.ts(mu), col = 2, lty = 2)
     plot(as.ts(draws$summary[1, ]),
          main     = expression('The underlying regimes and their estimates'),
          cex.main = 0.8,
          xlab     = 'period',
          ylab     = 'regime means',
          cex.lab  = 0.8)
     lines(as.ts(mu), col = 2, lty = 2)
 })

MSObj  <- MarkovSwitchingFuncGenerator(-97531)
smcObj <-
    with(MSObj,
     {
         particleFilter(nStreams                = 5000,
                        nPeriods                = nrow(yy),
                        dimPerPeriod            = ncol(yy),
                        generateNextStreamsFunc = generateNextStreamsFunc,
                        logObsDensFunc          = logObsDensFunc,
                        nMHSteps                = 10,
                        returnStreams           = TRUE,
                        returnLogWeights        = TRUE,
                        verboseLevel            = 1)
     })
print(smcObj)
print(names(smcObj))
with(c(smcObj, MSObj),
 {
     par(mfcol = c(2, 1))
     plot(as.ts(yy),
          main     = expression('The data and the underlying regimes'),
          cex.main = 0.8,
          xlab     = 'period',
          ylab     = 'data and the regime means',
          cex.lab  = 0.8)
     lines(as.ts(mu), col = 2, lty = 2)
     plot(as.ts(draws$summary[1, ]),
          main     = expression('The underlying regimes and their estimates'),
          cex.main = 0.8,
          xlab     = 'period',
          ylab     = 'regime means',
          cex.lab  = 0.8)
     lines(as.ts(mu), col = 2, lty = 2)
 })
```

`print` *The printing family of functions*

Description

The printing family of functions for this package.

Usage

```
## S3 method for class 'SMC'
print(x, ...)
```

Arguments

`x`	an object inheriting from class `SMC` (generated by functions `particleFilter`, `auxiliaryParticleFilter` and `sequentialMonteCarlo`).
`...`	optional arguments passed to `print.default`; see its documentation.

Author(s)

Gopi Goswami <`goswami@stat.harvard.edu`>

See Also

`particleFilter`, `auxiliaryParticleFilter`, `sequentialMonteCarlo`

`sequentialMonteCarlo` *The sequential Monte Carlo (SMC) algorithm*

Description

Function for the doing sequential Monte Carlo algorithm given the propagation rule over time (via `propagateFunc`). This is the most general interface for implementing a new SMC strategy, by providing a new propagation rule.

See the sections *Details*, *Required Functions* and *Optional Functions* for explanation on the arguments and the return values of the *arguments that are themselves functions*.

Usage

```
sequentialMonteCarlo(nStreams,
                     nPeriods,
                     dimPerPeriod,
                     propagateFunc,
                     resampCriterionFunc = NULL,
                     resampFunc          = NULL,
                     summaryFunc         = NULL,
                     nMHSteps            = 0,
                     MHUpdateFunc        = NULL,
                     nStreamsPreResamp   = NULL,
                     returnStreams       = FALSE,
                     returnLogWeights    = FALSE,
                     verboseLevel        = 0,
                     ...)
```

Arguments

nStreams integer > 0.

nPeriods integer > 0.

dimPerPeriod integer > 0.

propagateFunc function of six arguments (currentPeriod, nStreamsToGenerate, lag1Streams, lag1LogWeig

resampCriterionFunc
function of four arguments (currentPeriod, currentStreams, currentLogWeights, ...).

resampFunc function of four arguments (currentPeriod, currentStreams, currentLogWeights, ...).

summaryFunc function of four arguments (currentPeriod, currentStreams, currentLogWeights, ...).

nMHSteps integer ≥ 0.

MHUpdateFunc function of six arguments (currentPeriod, nMHSteps, currentStreams, lag1Streams, lag1

nStreamsPreResamp
integer > 0.

returnStreams logical.

returnLogWeights
logical.

verboseLevel integer, a value ≥ 2 produces a lot of output.

... optional arguments to be passed to propagateFunc, resampCriterionFunc, resampFunc, summaryFunc and MHUpdateFunc.

Details

We introduce the following terms, which will be used in the sections *Required Function* and *Optional Function* below:

stream the state vector also called the particle, the hidden state or the latent variable. Below we will use the terms stream and state vector interchangeably.

dimPerPeriod the dimension of the space, the state vectors live in.

Value

This function returns a list with the following components:

`draws` a list with the following components: `summary`, `propUniqueStreamIds`, `streams`, `logWeights`, `acceptanceRates`. See the section *Note* for more details.

`nStreams` the `nStreams` argument.

`nPeriods` the `nPeriods` argument.

`dimPerPeriod` the `dimPerPeriod` argument.

`nStreamsPreResamp` the `nStreamsPreResamp` argument.

`nMHSteps` the `nMHSteps` argument.

`filterType` type of the filter: “sequentialMonteCarlo”.

`time` the time taken by the run.

Required function: propagateFunc

Arguments: The following argument(s) require some explanation:

`nStreamsToGenerate` the number of streams to generate for propagating from `currentPeriod - 1` to `currentPeriod`. This function is usally called by setting `nStreamsToGenerate` to `nStreamsPreResamp`.

`lag1Streams` a matrix of dimension `nStreams` × `dimPerPeriod` of streams for `currentPeriod - 1`.

`lag1LogWeights` a vector of length `nStreams` of log weights corresponding to the streams in the argument matrix `lag1Streams`.

`startingStreams` a matrix of dimension `nStreams` × `dimPerPeriod` to be used for `currentPeriod = 1`. If this is `NULL`, then the function should provide a way to generate streams for `currentPeriod = 1`.

Return value: a *named* list with the following components:

`currentStreams` a matrix of dimension `nStreamsToGenerate` × `dimPerPeriod`. The rows of this matrix contain the propagated (updated) streams for period `currentPeriod`, given the argument `lag1Streams` matrix and the argument `lag1LogWeights` vector for `currentPeriod - 1`.

`currentLogWeights` the propagated (updated) log weights vector of length `nStreamsToGenerate`, associated with the streams in the rows of the returned `currentStreams` matrix.

Optional function: resampCriterionFunc

Arguments: The following argument(s) require some explanation:

`currentStreams` a matrix with `dimPerPeriod` columns, the rows containing the updated streams for `currentPeriod`.

`currentLogWeights` a vector of log weights corresponding to the streams in the argument matrix `currentStreams`.

Return value: `TRUE` or `FALSE` reflecting the decision of the resampling scheme implemented by this function.

Note: The following points are in order:

- resampling schemes manily depend on currentLogWeights, the other two arguments might come in handy for implementing period or stream specific resampling schemes.
- if nStreamsPreResamp > nStreams, then this function should always return TRUE.

Optional function: resampFunc

Arguments: see the sub-section *Arguments:* for section *Optional function: resampCriterionFunc*.

Return value: a *named* list with the following components:

currentStreams a matrix of dimension nStreams × dimPerPeriod. The rows of this matrix contain the streams for period currentPeriod + 1 that were resampled from those of the argument currentStreams matrix, which may contain $\geq$ nStreams rows.

currentLogWeights The log weights vector of length nStreams, associated with the streams that were resampled in the returned currentStreams matrix. Note, after the resampling step, usually all the log weights are set to 0.

Note: the components of the list returned by this function and the arguments to this function have two common names, namely, currentStreams and currentLogWeights. These entities have different meanings, as explained above. For example, the argument matrix currentStreams could possibly have $\geq$ nStreams rows, whereas the returned currentStreams has exactly nStreams number of (resampled) streams in its rows.

Optional function: summaryFunc

Arguments: The following argument(s) require some explanation:

currentStreams a matrix of dimension nStreams × dimPerPeriod of streams for currentPeriod.

currentLogWeights a vector of log weights corresponding to the streams in the argument matrix currentStreams.

Return value: a vector of length of dimSummPerPeriod of summaries for currentPeriod given the currentStreams and the currentLogWeights.

Optional function: MHUpdateFunc

Arguments: The following argument(s) require some explanation:

nMHSteps the number of Metropolis Hastings (MH) steps (iterations) to be performed.

currentStreams a matrix of dimension nStreams × dimPerPeriod of streams for currentPeriod.

lag1Streams a matrix of dimension nStreams × dimPerPeriod of streams for currentPeriod - 1.

lag1LogWeights a vector of length nStreams of log weights corresponding to the streams in the argument matrix lag1Streams.

Return value: a *named* list with the following components:

`currentStreams` a matrix of dimension `nStreams` × `dimPerPeriod`. The rows of this matrix contain the streams for period `currentPeriod` that are (possibly) MH-updated versions of the rows of the argument `currentStreams` matrix.

`acceptanceRates` a vector of length `nStreams`, representing the acceptance rates of the `nMHSteps`-many MH steps for each of the streams in the rows of the argument `currentStreams` matrix.

Note: a positive value of `nMHSteps` performs as many MH steps on the rows of the argument `currentStreams` matrix. This is done to reduce the possible degeneracy after the resampling.

Warning

Using very small values ($\leq$ `1e3`) for `nStreams` might not give reliable results.

Note

The effect of leaving the default value `NULL` for some of the arguments above are as follows:

`resampCriterionFunc` the builtin resampling criterion, namely, resample when square of the coefficient of variation of the weights ≥ 1, is used.

`resampFunc` the builtin resampling function, which resamples streams with probability proportional to their weights, is used.

`summaryFunc` the builtin summary function, which returns the weighted average of each of the `dimPerPeriod` dimensions, is used.

`MHUpdateFunc` *unlike*, `particleFilter`, there is no builtin Metropolis Hastings updating function, which generates proposals for `currentPeriod` streams using those of `currentPeriod - 1`. The user needs to implement this function if `nMHSteps` > 0.

`nStreamsPreResamp` it is set to `nStreams`.

Also, the following point is worth noting:

`resampCriterionFunc`**,** `resampFunc`**,** `summaryFunc` are only necessary when user wants to try out new resampling schemes or enhanced summary generation procedures, as part of their research. The default builtins take care of the typical problems.

This function returns a list with component called `draw`. The detailed description of this component, as promised in section *Value*, is as follows. It is a list itself with the following components:

`summary` a matrix of dimension `nPeriods` × `dimSummPerPeriod`.

`propUniqueStreamIds` a vector of length `nPeriods`. The values are either proportions of unique stream ids accpeted (at each period) if resampling was done or `NA`.

`streams` an array of dimension `nStreams` × `dimPerPeriod` × `nPeriods`. This is returned if `returnStreams = TRUE`.

`logWeights` a matrix of dimension `nStreams` × `nPeriods`. This is returned if `returnLogWeights = TRUE`.

`acceptanceRates` a matrix of dimension `nStreams` × `nPeriods`. This is returned if `nMHSteps > 0`.

Author(s)

Gopi Goswami <`goswami@stat.harvard.edu`>

References

Jun S. Liu (2001). Monte Carlo strategies for scientific computing. *Springer. Chapter 3.*

Jun S. Liu and Rong Chen (1998). Sequential Monte Carlo methods for dynamical systems. Journal of the American Statistical Association 98(443): 1032-1044.

See Also

particleFilter, auxiliaryParticleFilter

Examples

```
MSObj  <- MarkovSwitchingFuncGenerator(-12345)
smcObj <-
    with(MSObj,
     {
         sequentialMonteCarlo(nStreams         = 5000,
                              nPeriods         = nrow(yy),
                              dimPerPeriod     = ncol(yy),
                              propagateFunc    = propagateFunc,
                              returnStreams    = TRUE,
                              returnLogWeights = TRUE,
                              verboseLevel     = 1)
     })
print(smcObj)
print(names(smcObj))
with(c(smcObj, MSObj),
 {
     par(mfcol = c(2, 1))
     plot(as.ts(yy),
          main     = expression('The data and the underlying regimes'),
          cex.main = 0.8,
          xlab     = 'period',
          ylab     = 'data and the regime means',
          cex.lab  = 0.8)
     lines(as.ts(mu), col = 2, lty = 2)
     plot(as.ts(draws$summary[1, ]),
          main     = expression('The underlying regimes and their estimates'),
          cex.main = 0.8,
          xlab     = 'period',
          ylab     = 'regime means',
          cex.lab  = 0.8)
     lines(as.ts(mu), col = 2, lty = 2)
 })

MSObj  <- MarkovSwitchingFuncGenerator(-54321)
smcObj <-
    with(MSObj,
     {
         sequentialMonteCarlo(nStreams         = 5000,
                              nPeriods         = nrow(yy),
                              dimPerPeriod     = ncol(yy),
                              propagateFunc    = propagateFunc,
                              returnStreams    = TRUE,
                              returnLogWeights = TRUE,
                              verboseLevel     = 1)
     })
print(smcObj)
print(names(smcObj))
with(c(smcObj, MSObj),
 {
     par(mfcol = c(2, 1))
     plot(as.ts(yy),
          main     = expression('The data and the underlying regimes'),
          cex.main = 0.8,
          xlab     = 'period',
          ylab     = 'data and the regime means',
          cex.lab  = 0.8)
     lines(as.ts(mu), col = 2, lty = 2)
     plot(as.ts(draws$summary[1, ]),
          main     = expression('The underlying regimes and their estimates'),
          cex.main = 0.8,
          xlab     = 'period',
          ylab     = 'regime means',
          cex.lab  = 0.8)
     lines(as.ts(mu), col = 2, lty = 2)
 })
```

---

utilsForExamples *The utility function(s) for examples*

Description

The utility function(s) that are used in the example sections of the exported functions in this package.

Usage

```
MarkovSwitchingFuncGenerator(seed = -975313579)
```

Arguments

seed	the seed for random number generation.

Value

A list containing the objects to be used as arguments to the exported functions in the respective example sections of this package.

Author(s)

Gopi Goswami <goswami@stat.harvard.edu>

See Also

`particleFilter`, `auxiliaryParticleFilter`, `sequentialMonteCarlo`

Index

∗Topic **datagen**
    `utilsForExamples`, 21
∗Topic **methods**
    `auxiliaryParticleFilter`, 1
    `particleFilter`, 8
    `sequentialMonteCarlo`, 15
∗Topic **print**
    `print`, 15

`auxiliaryParticleFilter`, 1, *13*, *15*, *20*, *22*

`MarkovSwitchingFuncGenerator`
        (`utilsForExamples`*)*, 21

`particleFilter`, *6*, *7*, 8, *15*, *19*, *20*, *22*
`print`, 15

`sequentialMonteCarlo`, *15*, 15, *22*

`utilsForExamples`, 21