Package ‘smcUtils’

February 20, 2015

Type Package
Title Utility functions for sequential Monte Carlo
Version 0.2.2
Author Author: Jarad Niemi
Maintainer Jarad Niemi <niemi@iastate.edu>
Date 2012-12-27
Description Provides resampling functions (stratified, residual, multinomial, systematic, and branching), measures of weight uniformity (coefficient of variation, effective sample size, and entropy), and a weight renormalizing function.
License GPL-3
URL https://github.com/jarad/smcUtils
LazyLoad yes
Depends stats
Suggests testthat, Hmisc
NeedsCompilation yes
Repository CRAN
Date/Publication 2013-02-25 20:16:36

R topics documented:

ess.weights .......................................................... 2
renormalize .......................................................... 3
renormalize.weights-deprecated ................................ 4
resample .............................................................. 4
smcUtils-deprecated .............................................. 6
Unbiased resampling .............................................. 6

Index 8
**ess.weights**  
*Measures of weight non-uniformity*

**Description**
These functions calculate heuristic measures of the effective number of samples given a set of weights (probabilities).

**Usage**
- `ess.weights(weights, engine="R")`
- `cov.weights(weights, engine="R")`
- `ent.weights(weights, engine="R")`

**Arguments**
- `weights`: a vector of weights (probabilities)
- `engine`: run using "R" or "C" code

**Details**
- `ess.weights` calculates the effective sample size, namely `1/(sum(weights^2))`. ESS has a minimum of 1 and a maximum equal to `length(weights)` when weights are uniform.
- `cov.weights` calculates the coefficient of variation of the weights, namely `var(weights)/mean(weights)^2`. CoV has a minimum of 0 when weights are uniform and a maximum equal to `length(weights)`.
- `ent.weights` calculates the entropy of the weights, namely `-sum(weights * log2(weights))`. Entropy has a minimum of 0 and a maximum equal to `log2(length(weights))` when weights are uniform. (For numerical stability, the log term is actually calculated with `weights+.Machine$double.eps`, which can cause the observed minimum to be less than 0.)

**Value**
a scalar indicating how uniform the weights are

**Author(s)**
Jarad Niemi

**References**

**See Also**
- `resample`
**Examples**

```r
ws = renormalize(runif(10))
ess.weights(ws)
cov.weights(ws)
ent.weights(ws)
```

---

**renormalize**  
*Renormalize weights*

**Description**

This function creates a discrete probability distribution from a set of unnormalized (log) weights.

**Usage**

```r
renormalize(weights, log=FALSE, engine="R")
```

**Arguments**

- `weights`: a vector of numbers (>0 if log=FALSE)
- `log`: a logical indicating whether the weights are given as the logarithm of the weights
- `engine`: run using "R" or "C" code

**Value**

a vector of probabilities that sum to 1

**Author(s)**

Jarad Niemi

**See Also**

*resampling*

**Examples**

```r
ws = runif(10)
renormalize(ws)

log.ws = dnorm(rnorm(10,1,0),log=TRUE)
renormalize(log.ws,log=TRUE)
```
Renormalize weights

Description

This function creates a discrete probability distribution from a set of unnormalized (log) weights.

Usage

renormalize.weights(weights, log=FALSE, engine="R")

Arguments

weights    a vector of numbers (>0 if log=FALSE)
log        a logical indicating whether the weights are given as the logarithm of the weights
engine     run using "R" or "C" code

Value

a vector of probabilities that sum to 1

Author(s)

Jarad Niemi

See Also

renormalize

Resample

Description

A set of resampling functions with unbiased number of replicates.

Usage

resample(weights, num.samples=length(weights),
         method = c("stratified","residual","multinomial","systematic","branching"),
         nonuniformity = c("none","ess","cov","entropy"), threshold = 0.5,
         rrf = "stratified", engine="R", log=TRUE, normalized=FALSE)
Arguments

weights a vector of weights, possibly logged, unnormalized, or both
num.samples a scalar indicating the number of samples to return (for 'branching.resample', 'num.samples' is the expected number of samples as the actual number is random)
method a character string indicating the resampling method to use. One of "multinomial" (default), "residual", "stratified", "systematic", or "branching", can be abbreviated.
nonuniformity a character string indicating which heuristic to use for measuring weight nonuniformity. One of "none" (default), "ess", "cov", or "entropy", can be abbreviated. If "none", resampling will always be performed.
threshold a scalar in [0,1] indicating when to resample. If 'nonuniformity="none"', resampling is always performed regardless of threshold. If 'nonuniformity="ess"' or "entropy", resampling is performed when 'nonuniformity/num.samples<threshold'. If 'nonuniformity="cov"', resampling is performed when 'nonuniformity/log2(num.samples)>threshold'.
rrf for residual resampling, the resampling function to use on the residual
engine run using "R" or "C" code
log if TRUE, the weights are assumed to be logged
normalized if TRUE, the weights are assumed to be normalized and no error checking is done

Value

weights the component weights
indices an integer vector containing the indices of resampled components. If no resampling was performed, then 'indices=1:length(weights)'.

Author(s)

Jarad Niemi

See Also

resampling

Examples

ws = dnorm(0:5, log=TRUE)
resample(ws)
resample(ws,13)
resample(ws, method="residual")
resample(ws, method="residual", rrf="stratified")
resample(ws,17,"stratified","ess",0.5)
Description

These functions are provided for compatibility with older versions of smcUtils and may be defunct as soon as the next release.

See Also

Deprecated.renormalize.weights-deprecated

Unbiased resampling  Resampling functions

Description

A set of resampling functions with unbiased number of replicates.

Usage

```r
multinomial.resample(weights, num.samples = length(weights), engine="R")
residual.resample( weights, num.samples = length(weights), engine="R", rrf = "stratified")
stratified.resample( weights, num.samples = length(weights), engine="R")
systematic.resample( weights, num.samples = length(weights), engine="R")
branching.resample( weights, num.samples = length(weights), engine="R")
```

Arguments

- `weights`: a vector of normalized weights
- `num.samples`: the number of samples to return (for `branching.resample`, `num.samples` is the expected number of samples as the actual number is random)
- `rrf`: for residual resampling, the resampling function to use on the residual
- `engine`: run using "R" or "C" code

Details

- `multinomial.resample` samples component i with probability `weights[i]`, repeats this sampling `num.samples` times, and returns indices for the sampled components.
- `residual.resample` deterministically copies `floor(weights)` number of each component and then performs ‘rrf’ on the remainder.
- `stratified.resample` draws `num.samples` uniform random variables on the ((i-1)/num.samples,i/num.samples) intervals of (0,1). It then uses the inverse.cdf.weights function to determine which components to sample.
Unbiased resampling

‘s systematic.resample’ draws 1 uniform random variable on (0,1/num.samples), builds a sequence of ‘num.samples’ numbers by sequentially adding ‘1/num.samples’, and then uses ‘inverse.cdf.weights’ to determine which components to sample.

‘branching.resample’ deterministically copies ‘floor(weights)’ number of components and then draws another component i with probability equal to the residual for that component. Note: the actual number of components after resampling is random.

Value

Returns a vector of length ‘num.samples’ with indices for sampled components.

Author(s)

Jarad Niemi

References


See Also

resample, renormalize,

Examples

ws = renormalize(runif(10))
m multinational.resample(ws)
residual.resample(ws, rrf="stratified")
stratified.resample(ws,15)
s systematic.resample(ws)
branching.resample(ws)
Index

*Topic htest
  ess.weights, 2
  renormalize, 3
  renormalize.weights-deprecated, 4
  resample, 4
  Unbiased resampling, 6

branching.resample (Unbiased resampling), 6

cov.weights (ess.weights), 2

Deprecated, 6

tent.weights (ess.weights), 2
ess.weights, 2

multinomial.resample (Unbiased resampling), 6

renormalize, 3, 4, 7
renormalize.weights (renormalize.weights-deprecated), 4
renormalize.weights-deprecated, 4
resample, 2, 4, 7
resampling, 3, 5
resampling (Unbiased resampling), 6
residual.resample (Unbiased resampling), 6

smcUtils-deprecated, 6
stratified.resample (Unbiased resampling), 6
systematic.resample (Unbiased resampling), 6

Unbiased resampling, 6