Package ‘mcunit’

March 5, 2020

Title  Unit Tests for MC Methods
Version  0.3.1
Maintainer  Axel Gandy <a.gandy@imperial.ac.uk>
Description  Unit testing for Monte Carlo methods, particularly Markov Chain Monte Carlo (MCMC) methods, are implemented as extensions of the 'testthat' package. The MCMC methods check whether the MCMC chain has the correct invariant distribution. They do not check other properties of successful samplers such as whether the chain can reach all points, i.e. whether is recurrent. The tests require the ability to sample from the prior and to run steps of the MCMC chain. The methodology is described in Gandy and Scott (2020) <arXiv:2001.06465>.

URL  https://bitbucket.org/agandy/mcunit/
License  GPL-3
Encoding  UTF-8
LazyData  true
Depends  R (>= 3.1)
Imports  testthat (>= 2.3), stats, rlang, Rdpack (>= 0.7), methods, simctest (>= 2.6)
Suggests  knitr
VignetteBuilder  knitr
RoxygenNote  7.0.2
RdMacros  Rdpack
NeedsCompilation  no
Author  Axel Gandy [aut, cre], James Scott [aut]
Repository  CRAN
Date/Publication  2020-03-05 11:10:05 UTC
Description

Test if the success probability of a Bernoulli experiment lies within a desired 'bucket'. This used the sequential procedure described in Gandy et al. (2019).

Usage

expect_bernoulli(object, J, ok, epsilon = 0.001, ...)

Arguments

object: Function that performs one sampling step. Returns 0 or 1.
J: Buckets to use. A matrix with two rows, each column describes an interval bucket. Column names give names for the bucket(s).
ok: Name of bucket(s) that pass the Unit test.
epsilon: Error bound.
... Further parameters to be passed on to 'mctest'.

Value

The first argument, invisibly, to allow chaining of expectations.

References


Examples

J <- matrix(nrow=2,c(0,0.945, 0.94,0.96, 0.955,1))
colnames(J) <- c("low","ok","high")
gen <- function() as.numeric(runif(1)<0.95)
expect_bernoulli(gen,J=J,ok="ok")
expect_mcmc

Test of MCMC chain

Description
Test of MCMC steps having the correct stationary distribution without assuming reversibility of the chain. Details of this are in Gandy and Scott (2020); it uses ideas described in the appendix of Gandy and Veraart (2017).

Usage
```
expect_mcmc(object, control = NULL, thinning = NULL, joint = FALSE)
```

Arguments
- **object**: A list describing the MCMC sampler with the following elements:
  - `genprior`: A function with no arguments that generates a sample of the prior distribution. No default value.
  - `gendata`: A function that takes as input the parameter value (of the type generated by genprior) and returns the observed data as an arbitrary R object. No default value.
  - `stepMCMC`: A function that takes three arguments:
    - `theta`: the current position of the chain (of the same type as produced by the prior).
    - `dat`: the observed data (of the same type as produced by gendat)
    - `thinning`: the number of steps the chain should take. 1 corresponds to one step.
  - `test`: Function that takes either one or two arguments, and returns a vector with components which will be used for checking the MCMC sampler. The first argument is interpreted as a parameter value, and if a second argument exists, it is interpreted as a data value. An example is the identity function: `function(x) x`. Alternatively, if you have access to the model’s likelihood function, you could use: `function(x,y) likelihood(x,y)`.
- **control**: a list controlling the algorithm
  - `n`: number of samples to be taken in the first step. Default: 1e3
  - `maxseqsteps`: Number of sequential attempts to use. Default: 7.
  - `incn`: Factor by which to multiply n from the second sequential attempt onwards. Default: 4.
  - `level`: bound on the type I error, ie the probability of wrongly rejecting a sampler with the correct distribution. Default: 1e-5.
  - `debug`: If positive then debug information will be printed via ‘message()’. Default: 0.
- **thinning**: Amount of thinning for the MCMC chain. 1 corresponds to no thinning. Default: automatically computed to ensure an autocorrelation of at most 0.5 at lag 1.
If TRUE, then the MCMC uses systematic scan of both data and parameters, rather than just updating parameters with the sampler to be tested. Default: FALSE.

Value

The first argument, invisibly, to allow chaining of expectations.

References


See Also

expect_mcmc_reversible

Examples

object <- list(genprior=function() rnorm(1),
               gendata=function(theta) rnorm(5,theta),
               stepMCMC=function(theta,data,thinning){
                 f <- function(x) prod(dnorm(data,x))*dnorm(x)
                 for (i in 1:thinning){
                   thetanew = rnorm(1,mean=theta,sd=1)
                   if (runif(1)<f(thetanew)/f(theta))
                     theta <- thetanew
                 }
                 theta
               })
expect_mcmc(object)

expect_mcmc_reversible

Test of MCMC chain assuming reversibility of the chain

Description

Test of MCMC steps having the correct stationary distribution assuming reversibility of the chain. Uses ideas from Besag and Clifford (1989) as extended to possible ties in Gandy and Scott (2020).
Usage

expect_mcmc_reversible(
  object,
  control = NULL,
  thinning = NULL,
  nsteps = 10,
  p = 1,
  tolerance = 1e-08
)

Arguments

object A list describing the MCMC sampler with the following elements:
  • genprior: A function with no arguments that generates a sample of the prior
distribution. No default value.
  • gendata: A function that takes as input the parameter value (of the type gen-
erated by genprior) and returns the observed data as an arbitrary R object.
    No default value.
  • stepMCMC: A function that takes three arguments:
    – theta: the current position of the chain (of the same type as produced
      by the prior),
    – dat: the observed data (of the same type as produced by gendat)
    – thinning: the number of steps the chain should take. 1 corresponds to
      one step.
  • test: Function that takes either one or two arguments, and returns a vector
    with components which will be used for checking the MCMC sampler. The
    first argument is interpreted as a parameter value, and if a second argument
    exists, it is interpreted as a data value. An example is the identity function:
    function(x) x. Alternatively, if you have access to the model’s likelihood
    function, you could use: function(x,y) likelihood(x,y).

control a list controlling the algorithm
  • n number of samples to be taken in the first step. Default: 1e3
  • maxseqsteps: Number of sequential attempts to use. Default: 7.
  • incn: Factor by which to multiply n from the second sequential attempt
  • level: bound on the type I error, ie the probability of wrongly rejecting a
    sampler with the correct distribution. Default: 1e-5.
  • debug: If positive then debug information will be printed via ‘message()’.
    Default: 0.

thinning Amount of thinning for the MCMC chain. 1 corresponds to no thinning. Default:
automatically computed to ensure an autocorrelation of at most 0.5 at lag 1.

nsteps Number of steps of the chain to use. Default: 10.

p The probability with which the MCMC updates the parameter as opposed to the
data in a given step. If less than 1.0, then the MCMC is a random scan on both
data and parameters. Default: 1.0.

tolerance Absolute error where value of the samplers are treated as equal. Default: 1e-8.
**Value**

The first argument, invisibly, to allow chaining of expectations.

**References**


**See Also**

*expect_mcmc*

**Examples**

```r
object <- list(genprior=function() rnorm(1),
               gndata=function(theta) rnorm(5,theta),
               stepMCMC=function(theta,data,thinning){
                 f <- function(x) prod(dnorm(data,x))*dnorm(x)
                 for (i in 1:thinning){
                   thetanew = rnorm(1,mean=theta,sd=1)
                   if (runif(1)<f(thetanew)/f(theta))
                     theta <- thetanew
                   }
                 theta
               },
               test=function(x) x
             )
expect_mcmc_reversible(object)
```

---

**expect_mc_iid_chisq**  
*Test iid samples for correct cdf using chisq test*

**Description**

Test if samples are behaving like an iid sample from a given distribution via the chisq test and a sequential approach. Only works for discrete distributions taking finitely many values.

**Usage**

```r
expect_mc_iid_chisq(object, prob, control = NULL)
```
expect_mc_iid_ks

Arguments
  object  A function taking one argument - that generates n univariate iid samples.
  prob   A vector of probabilities for finitely many consecutive integers from 0 onwards.
  control  a list controlling the algorithm
    • n number of samples to be taken in the first step. Default: 1e3
    • maxseqsteps: Number of sequential attempts to use. Default: 7.
    • incn: Factor by which to multiply n from the second sequential attempt onwards. Default: 4.
    • level: bound on the type I error, i.e. the probability of wrongly rejecting a sampler with the correct distribution. Default: 1e-5.
    • debug: If positive then debug information will be printed via 'message()'. Default: 0.

Value
  The first argument, invisibly, to allow chaining of expectations.

Examples
  sampler <- function(n) rbinom(n, prob = 0.6, size = 5)
  expect_mc_iid_chisq(sampler, dbinom(0:5, prob = 0.6, size = 5))
  testthat::expect_error(expect_mc_iid_chisq(sampler, dbinom(0:5, prob = 0.63, size = 5)))

expect_mc_iid_ks  Test iid samples for correct cdf using KS test

Description
  Test if samples are behaving like an iid sample from a given CDF via the KS test and a sequential approach. Only works for continuous CDFs. Will report a warning if values are discrete.

Usage
  expect_mc_iid_ks(object, cdf, control = NULL)

Arguments
  object  A function taking one argument - that generates n univariate iid samples.
  cdf    A univariate cumulative distribution function, taking exactly one argument.
  control  a list controlling the algorithm
    • n number of samples to be taken in the first step. Default: 1e3
    • maxseqsteps: Number of sequential attempts to use. Default: 7.
    • incn: Factor by which to multiply n from the second sequential attempt onwards. Default: 4.
expect_mc_iid_mean

- **level**: bound on the type I error, i.e., the probability of wrongly rejecting a sampler with the correct distribution. Default: 1e-5.
- **debug**: If positive then debug information will be printed via `message()`. Default: 0.

**Value**

The first argument, invisibly, to allow chaining of expectations.

**Examples**

```r
sampler <- function(n) rnorm(n)
expect_mc_iid_ks(sampler, pnorm)
```

---

**Description**

Test if samples are coming from a specific mean. Not guaranteed to be exact, as it estimates the standard error from the sample.

**Usage**

```r
expect_mc_iid_mean(object, mean, control = NULL)
```

**Arguments**

- **object**: A function taking one argument - that generates n univariate iid samples.
- **mean**: The expected mean of the samples returned from object.
- **control**: a list controlling the algorithm
  - n number of samples to be taken in the first step. Default: 1e3
  - maxseqsteps: Number of sequential attempts to use. Default: 7.
  - incn: Factor by which to multiply n from the second sequential attempt onwards. Default: 4.
  - level: bound on the type I error, i.e., the probability of wrongly rejecting a sampler with the correct distribution. Default: 1e-5.
  - debug: If positive then debug information will be printed via `message()`. Default: 0.

**Value**

The first argument, invisibly, to allow chaining of expectations.
**expect_mc_test**  
Test if p-values are coming from the null using a sequential approach.

**Description**

Requires as input a generic test that for a given sample size provides a vector of p-values. Aims to reject if these are not from the null. Guarantees a bound on the type I error rate.

**Usage**

```r
expect_mc_test(object, control = NULL, npval = 1)
```

**Arguments**

- **object**: A function taking one argument \( n \) - that generates p-values based on a sample size \( n \).
- **control**: a list controlling the algorithm
  - \( n \): number of samples to be taken in the first step. Default: 1e3
  - maxseqsteps: Number of sequential attempts to use. Default: 7.
  - incn: Factor by which to multiply \( n \) from the second sequential attempt onwards. Default: 4.
  - level: bound on the type I error, ie the probability of wrongly rejecting a sampler with the correct distribution. Default: 1e-5.
  - debug: If positive then debug information will be printed via `message()`. Default: 0.
- **npval**: number of p-values returned by the test. A Bonferroni correction is applied if \( >1 \). Default: 1.

**Value**

The first argument, invisibly, to allow chaining of expectations.

**Examples**

```r
pvalsampler <- function(n){
  x <- sample.int(11, size=n, replace=TRUE)-1;
  chisq.test(tabulate(x+1,nbins=11), p=rep(1/11,11))$p.value
}
expect_mc_test(pvalsampler)
```
Index

expect_bernoulli. 2
expect_mc iid chisq. 6
expect_mc iid ks. 7
expect_mc iid mean. 8
expect_mc_test. 9
expect_mcmc. 3, 6
expect_mcmc_reversible. 4, 4