An Introduction to Estimating Monte Carlo Standard Errors with R Package `mcmcse`

Dootika Vats

March 5, 2016

Contents

1 Introduction

The R package `mcmcse` provides estimates of Monte Carlo standard errors for Markov chain Monte Carlo (MCMC) algorithms. This package is useful when estimating means and quantiles of functions of the MCMC output. In addition to MCMC output, the package can be used for time series and other correlated processes.

The package is predominantly useful after MCMC output has been obtained by the user. In addition to estimating the Monte Carlo standard errors, the package also provides basic graphical diagnostics and calculation of effective sample size. Various features in the package can be implemented using both multivariate and univariate methods.

2 An MCMC Example

To illustrate the use of our package, we present the following simple multivariate AR(1) process. The process is defined for $t = 1, 2, 3, \ldots$ is defined as,

$$y_t = w + Ay_{t-1} + \epsilon_t,$$

where w is a constant vector in $\mathbb{R}^p$, $y_t \in \mathbb{R}^p$, A is a $p \times p$ matrix and $\epsilon_t \sim N_p(0, C)$. In our example, we let A and C be diagonal matrices. The invariant distrbution for this process is $F = N_p(0, V)$ where V is a function of A and C.

The function `mAr.sim` in package `mAr` draws samples from the above model. We let $p = 3$.

```
library(mAr)
p <- 3
A <- diag(c(.1, .5, .8))
C <- diag(rep(2, 3))

set.seed(100)
chain <- mAr.sim(w = rep(2,p), A = A, C = C, N = 10000)
```

For using the `mcmcse` package the rows of the MCMC output should store each iteration of the algorithm. Thus the output should have n rows and p columns. We will denote each row i of the output as $(y_i^{(1)}, y_i^{(2)}, y_i^{(3)})$.

This vignette will discuss estimating two sets of features of interest of F.

- the expectation of y, $E_F y$

- the expectation of sum of the second moments of all components of y, $E_F(y^{(1)2} + y^{(2)2} + y^{(3)2})$.

Suppose first we are interested in estimating $\mu = E_F y$. Then the estimator for that is just the sample mean

$$\mu_n = \frac{1}{n} \sum_{t=1}^{n} y_t.$$

μ_n is obtained using the usual `colMeans` function.

```
colMeans(chain)
```

```
##       X1       X2       X3
## 2.213455 3.989894 9.990273
```

Due to a central limit theorem argument,

$$\sqrt{n}(\mu_n - \mu) \xrightarrow{d} N_p(0, \Sigma). \tag{1}$$

Alternatively, we could also be interested in estimating say the sum of the second moments of each component of y. In this case, we define the function $g : \mathbb{R}^3 \to \mathbb{R}$ as $g((x_1, x_2, x_3)) = x_1^2 + x_2^2 + x_3^2$. This is defined in `R` by creating a function that takes a vector argument.

```
g <- function(x)
{
        return(sum(x^2))
}
```

The Monte Carlo estimate for g is

$$\mu_{g,n} = \frac{1}{n} \sum_{t=1}^{n} g(y_t),$$

and a CLT of the following form may be available

$$\sqrt{n}(\mu_{g,n} - \mu) \xrightarrow{d} N_p(0, \Sigma_g). \tag{2}$$

Finding the estimate and the Monte Carlo standard errors for $E_F g$ are explained in the following section.

3   Estimating Monte Carlo Standard Error

Using the `mcmcse` package we can estimate Σ in (1) with the `mcse.multi` function.

```
library(mcmcse)
mcerror_bm <- mcse.multi(x = chain, method =  "bm",
        size = "sqroot", g = NULL, level = .95, large = FALSE)
mcerror_bart <- mcse.multi(x = chain, method =  "bartlett",
        size = "cuberoot", g = NULL, level = .95, large = FALSE)
mcerror_tuk <- mcse.multi(x = chain, method =  "tukey",
        size = "sqroot", g = NULL, level = .95, large = FALSE)
```

- `x` takes the $n \times p$ MCMC data. `x` can take only numeric entries in the form of a matrix or data frame. The rows of `x` are the iterations of the MCMC.

- `method = ``bm''`, ` ``bartlett''`, ` ``tukey''` calculates the estimate using the batch means method and spectral variance methods with the modified-Bartlett and Tukey-Hanning windows.

- `size` is the batch size for the `bm` method and the truncation point for `tukey` and `bartlett` methods. `size = ``sqroot''` sets the size as $\lfloor \sqrt{n} \rfloor$ and `size = ``cuberoot''` sets it at $\lfloor n^{1/3} \rfloor$. An integer value of `size` less than n is also valid.

- `g` is a function that is applied to each row of `x` and represents the features of interest of the process. Since here we are interested in only means, `g` is `NULL`. `g` will be explained in later examples.

- `level` is the confidence level of the resulting confidence region. This is required to calculate the volume of the confidence region.

- `large` is a logical argument. If `large` is `TRUE` the volume of the confidence region is the large sample volume obtained using χ^2 critical values. By default, volume is calculated using F distribution critical values.

`mcse.multi` returns a list with multiple components. `cov` stores the estimate of Σ obtained using the method chosen, `vol` returns the volume to the pth root of the resulting confidence region, `est` stores the estimate of g applied on the Markov chain and `nsim`, `critical` and `size` store the arguments used to calculate Σ.

```
mcerror_bm$cov

##            [,1]       [,2]       [,3]
## [1,]   2.1818978 -0.2932679  0.8416831
## [2,]  -0.2932679  7.1329697  1.9953946
## [3,]   0.8416831  1.9953946 44.2180584

mcerror_bart$cov

##            [,1]       [,2]       [,3]
## [1,] 2.4769750  0.1504705  0.3108498
## [2,] 0.1504705  7.5311309 -0.2104305
## [3,] 0.3108498 -0.2104305 36.2779449

mcerror_tuk$cov

##            [,1]       [,2]      [,3]
## [1,] 2.5605087 0.1785598  1.291766
## [2,] 0.1785598 7.1818559  1.107386
## [3,] 1.2917665 1.1073863 46.506845

rbind(mcerror_bm$est, mcerror_bart$est, mcerror_tuk$est)

##             X1       X2       X3
## [1,] 2.213455 3.989894 9.990273
## [2,] 2.213455 3.989894 9.990273
## [3,] 2.213455 3.989894 9.990273

c(mcerror_bm$vol, mcerror_bart$vol, mcerror_tuk$vol)

## [1] 0.1370514 0.1335043 0.1384341
```

Note: The estimates are not affected by the choice of the method.

Note: The batch means estimators are significantly faster to calculate than the spectral variance estimators. The user is advised to use the default `method = ``bm''` for large input matrices.

Note: `cov` returns an estimate of Σ and not Σ/n.

If the diagonals of Σ are σ_{ii}^2, the function `mcse` and `mcse.mat` returns $\sigma_{ii}/\sqrt{n}$. `mcse` does it for one component and `mcse.mat` does it for all diagonals.

```
mcse(x = chain[,1], method = "bm", g = NULL)

## $est
## [1] 2.213455
##
## $se
## [1] 0.01477125

mcse.mat(x = chain, method = "bm", g = NULL)

##         est          se
## X1 2.213455 0.01477125
## X2 3.989894 0.02670762
## X3 9.990273 0.06649666
```

In order to estimate $\mu_{n,g}$ and Σ_g as in (2), we use the R function g we had defined before. Recall that g should be a funcation that takes vector inputs.

```
g

## function(x)
## {
##   return(sum(x^2))
## }

mcerror_g_bm <- mcse.multi(x = chain, g = g)

mcerror_g_bm$cov

##           [,1]
## [1,] 18247.05

mcerror_g_bm$est

## [1] 130.4437
```
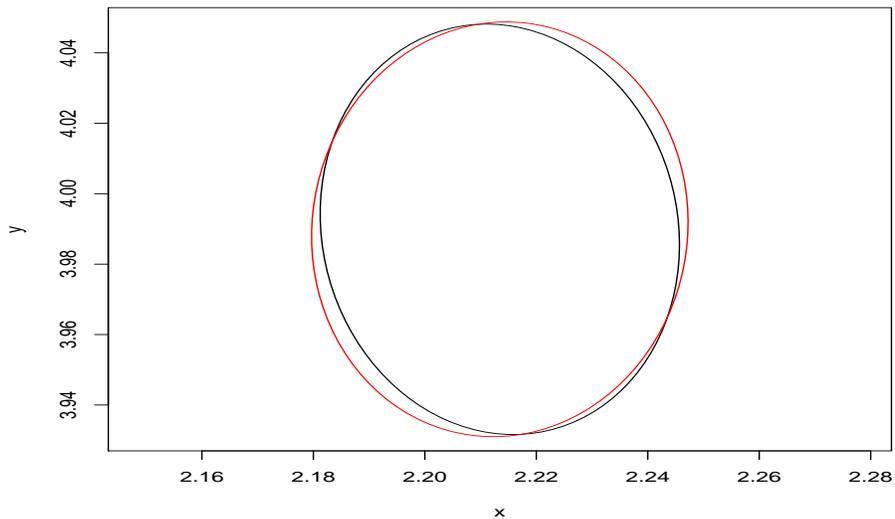
4   Confidence Regions

Using the function confRegion in the package, the user can create joint confidence regions for two parameters. The input for this function is the

6

output list from the `mcse.multi` function. The function uses the attributes `critical`, `est` and `nsim` from the `mcse.multi` output list.

```
plot(confRegion(mcerror_bm, which = c(1,2), level = .90), type = 'l', asp = 1)
lines(confRegion(mcerror_bart, which = c(1,2), level = .90), col = "red")
```


- `which` should be a vector of size 2 that indicates the two components for which the confidence ellipse is to be constructed.

- `level` is the confidence level of the confidence region. The default is .95

NOTE: The argument `confRegion` calls on the function `ellipse` in package `ellipse` to draw the ellipse.

NOTE: Since the confidence region is created for two parameters only, the size of the ellipse is determined by setting $p = 2$ irrespective of the original dimension of the problem.

To determine the effect of the confidence level, we draw two regions with difference confidence levels.

```
plot(confRegion(mcerror_bm, which = c(1,2), level = .95), type = 'l', asp = 1)
lines(confRegion(mcerror_bm, which = c(1,2), level = .90), col = "red")
```
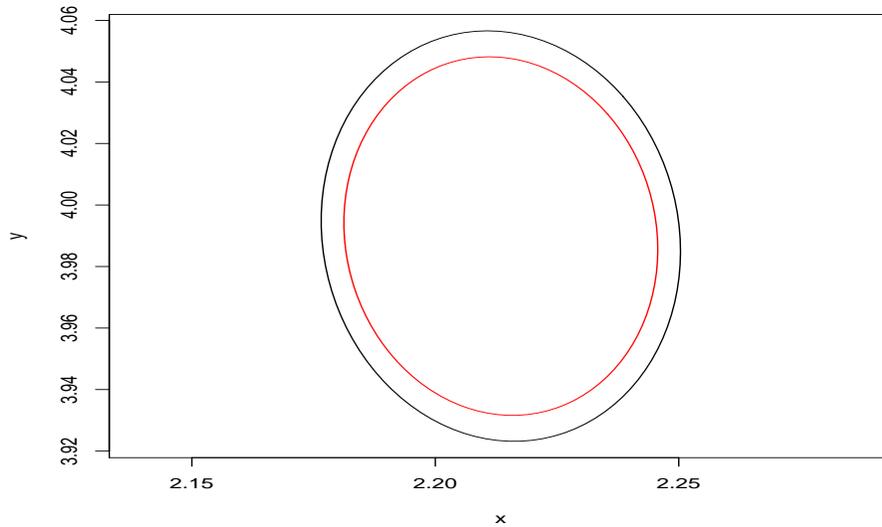

5 Effective Sample Size

Before sampling the Markov chain, the user is advised to used the function
`minESS` to figure out how many minimum effective samples are needed for
analysis. See Vats et al. (2015) for details. Since we have two quantities, μ
and μ_g, we will find the minimum samples required for each analysis.

```
# For mu
minESS(p = 3, alpha = .05, eps = .05)

## [1] 8123

#For mu_g
minESS(p = 1, alpha = .05, eps = .05)

## [1] 6146
```

- `p` is the dimension of the estimation problem.

- `alpha` is the confidence level

8

- **eps** is the tolerance level. Default is .05. Reasonable levels are anywhere from .01 to .05. The smaller the tolerance, the larger the minimum effective samples.

`minESS` is independent of the Markov chain or process, and is only a function of the p, α, and ϵ. The user should find `minESS` and then sample their process until the required minimum samples are achieved.

`multiESS` and `ess` are two functions that calculate the effective sample size of a correlated sample. `ess` calculations are based on Gong and Flegal (2016) and is component-wise, and `multiESS` utilizes the multivariate nature of the problem.

```
ess(chain)
```

```
##       X1        X2        X3
## 9381.155 3670.521 1165.908
```

Since `ess` produces a different estimate for each component, conservative practice dictates choosing the smallest of the values. `multiESS` returns one estimate of the effective sample size based on the whole sample. The function calls `mcse.multi` function to obtain a batch means estimate of Σ. The user can provide another estimate of Σ using the `covmat` argument.

```
multiESS(chain)
```

```
## [1] 3455.318
```

```
multiESS(chain, covmat = mcerror_bart$cov)
```

```
## [1] 3446.062
```

Since the effective sample size is less than the minimum effective samples, we should simulate more. Looking at the ratio of the Monte Carlo samples size of 10^4 and multiESS, we might need around $28,000$ Monte Carlo samples.

```
set.seed(100)
chain <- mAr.sim(w = rep(2,p), A = A, C = C, N = 28000)
multiESS(chain)
```

```
## [1] 8832.498
```

We see that the effective samples is now larger than the minimum effective samples, so we know 28,000 are enough.

NOTE: Ideally, we do not want to get more samples using the last iteration of the previous Markov chain. However, `mAr.sim` does not allow user specified starting values, so to demonstrate the use of `minESS` and `multiESS`, we get a new sample altogether. When making `R` packages that simulate a Markov chain, it is often very useful to allow user specific starting values for this reason.

6 Graphical Diagnostics

The function `estvssamp` plots the Monte Carlo estimates versus the sample size for a component of the MCMC output. This plot indicates whether the Monte Carlo estimate has stabilized.

```
estvssamp(chain[,1])
```
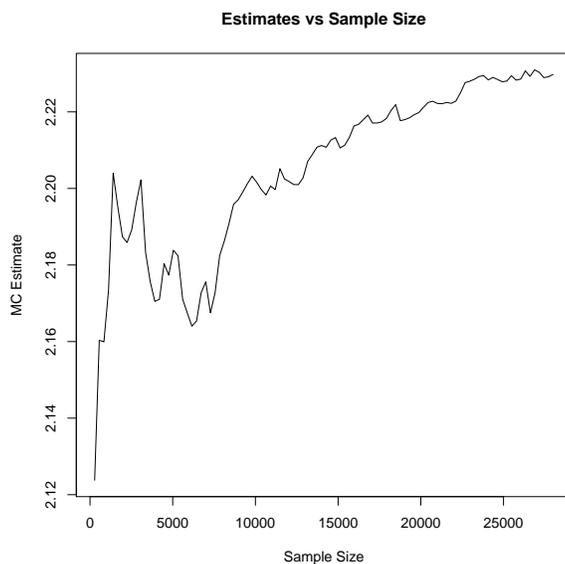

Additionally, if p is not too small, due to the central limit theorem in (1) and an estimate of Σ using the `mcse.multi` function, a QQ plot of the standardized estimates gives an idea of whether asymptopia has been achieved. We generate a new Markov chain with $p = 50$.

```
p <- 50
A <- diag(seq(.1, .9, length = p))
```

```
C <- diag(rep(2, p))

set.seed(100)
chain <- mAr.sim(w = rep(2,p), A = A, C = C, N = 10000)
```
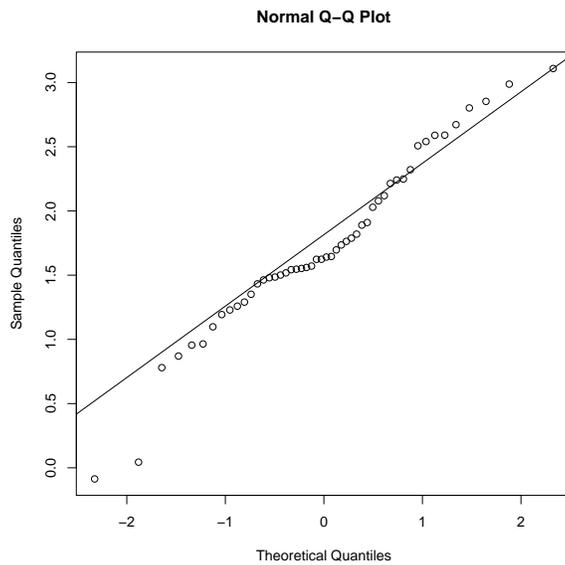
For this new Markov chain, we find an estimate of Σ to use for the `qqTest` function.

```
mcerror_bm <- mcse.multi(chain, method = "bm")
qqTest(x = chain, covmat = mcerror_bm$cov)
```

Normal Q–Q Plot

Thus, we see here that the chain has not quite reached asymptopia.

References

Gong, L. and Flegal, J. M. (2016). A practical sequential stopping rule for high-dimensional markov chain monte carlo. *Journal of Computational and Graphical Statistics*, (to appear).

Vats, D., Flegal, J. M., and Jones, G. L. (2015). Multivariate output analysis for markov chain monte carlo. *arXiv preprint arXiv:1512.07713*.

11