Metropolis Hasting Change Point

Deepak Bastola

May 23, 2018

```r
set.seed(1234)
library(mcmcse)
library(batchmeans)
library(parallel)

data <- read.table("/home/deepak/Desktop/Research/Codes/ChangePoint/changepoint.dat",
                   header = TRUE)
data <- as.matrix(data)
Y <- data[,2]

k.guess = 10

mhsampler <- function(Y, n.iterations, kfixed=FALSE){
    n <- length(Y)
    mchain <- matrix(NA, n.iterations, 5)
    acc <- 0
    # starting values
    if (kfixed) kinit <- k.guess # start the chain at the guess value
    else kinit <- floor(n/2) # midpoint
    mchain[1,] <- c(1,1,kinit,1,1)

    for (i in 2:n.iterations)
      {
        ## parameters at last iterations
        currtheta <- mchain[i-1,1]
        currlambda <- mchain[i-1,2]
        currk <- mchain[i-1,3]
        currb1 <- mchain[i-1,4]
        currb2 <- mchain[i-1,5]

        ## sample from full conditional distribution of theta (Gibbs update)
        currtheta <- rgamma(1,shape=sum(Y[1:currk])+0.5,
                           scale=currb1/(currk*currb1+1))

        ## sample from full conditional distribution of lambda (Gibbs update)
        currlambda <- rgamma(1,shape=sum(Y[(currk+1):n])+0.5,
                            scale=currb2/((n-currk)*currb2+1))

        ## sample from full conditional distribution of k (Metropolis-Hastings update)

        propk <- sample(x=seq(2,n-1), size=1) # draw one sample at random from uniform{2,..(n-1)}

        if (kfixed) {
          currk <- KGUESS
        } else {
          ## Metropolis accept-reject step (in log scale)
          logMHratio <- sum(Y[1:propk])*log(currtheta)+sum(Y[(propk+1):n])*
```

```r
                        log(currlambda)-propk*currtheta- (n-propk)*currlambda -
                            (sum(Y[1:currk])*log(currtheta)+sum(Y[(currk+1):n])*
                            log(currlambda)-currk*currtheta- (n-currk)*currlambda)
          logalpha <- min(0,logMHratio) # alpha = min(1,MHratio)
          if (log(runif(1))<logalpha) # accept if unif(0,1)<alpha, i.e. accept with
            {                         # probability alpha, else stay at current state
              acc <- acc + 1 # increment count of accepted proposals
              currk <- propk
            }
        }

        ## sample from full conditional distribution of b1 (Gibbs update): draw from Inverse Gamma
        currb1 <- 1/rgamma(1,shape=1.5, scale=1/(currtheta+1))

        ## sample from full conditional distribution of b2 (Gibbs update): draw from Inverse Gamma
        currb2 <- 1/rgamma(1,shape=1.5, scale=1/(currlambda+1))

        ## update chain with new values
        mchain[i,] <- c(currtheta,currlambda,currk,currb1,currb2)
      }

    #cat("Markov chain algorithm ran for", MCMCiterations, "iterations ")
   # if (!kfixed) cat("\n acc. rate for k: ", acc/(MCMCiterations-1))
    return(mchain)
  }

chain <- mhsampler(Y,10000)

pdf("tsplot.pdf")
ts.plot(Y,main="Time series plot of change point data", lty=3)
dev.off()
```

```
## pdf
##   2
```

```r
bm.est <- apply(chain, 2, function(i) bm(i)$est)
bm.est
```

```
## [1]  5.725579  8.697127 11.874400 13.179794 19.274803
```

```r
bm.se <- apply(chain, 2, function(i) bm(i)$se)
bm.se
```

```
## [1] 0.05600583 0.05983196 0.55320640 0.46053360 1.11419017
```

```r
#standard errors - cov mat
mcse.matrix <- mcse.mat(chain)
mcse.matrix
```

```
##            est         se
## [1,]  5.725579 0.05600583
## [2,]  8.697127 0.05983196
## [3,] 11.874400 0.55320640
## [4,] 13.179794 0.46053360
## [5,] 19.274803 1.11419017
```

```r
#effective sample size
ess <- lapply(1:5, function(i) ess(chain[,i]))
ess
```

```
## [[1]]
## [1] 505.1146
##
## [[2]]
## [1] 337.4047
##
## [[3]]
## [1] 232.9069
##
## [[4]]
## [1] 9789.233
##
## [[5]]
## [1] 10000
```

```r
#relative tolerance eps = 0.05
miness <- minESS(p=5, eps = 0.05, alpha = 0.05)
miness
```

```
## minESS
##   8605
```

```r
#multivariate sample size
multess <- multiESS(chain)
multess
```

```
## [1] 3138.157
```

```r
chain.new <- mhsampler(Y, 32000)

#recalculate ess
multess.final <- multiESS(chain.new)
multess.final
```

```
## [1] 9098.187
```

```r
pdf("acfplots.pdf")
par(mfrow = c(3,2))
acf(chain.new[1,],main="acf plot for theta")
acf(chain.new[2,],main="acf plot for lambda")
acf(chain.new[3,],main="acf plot for k")
acf(chain.new[4,],main="acf plot for b1")
acf(chain.new[5,],main="acf plot for b2")
par(mfrow=c(1,1))
dev.off()
```

```
## pdf
##   2
```

```r
#estimates with their standard errors
mcse.matrix <- mcse.mat(chain.new)
mcse.matrix
```

```
##              est          se
## [1,]  5.646364 0.02481501
## [2,]  8.788484 0.02744105
## [3,] 11.006281 0.23909290
## [4,] 12.694284 0.24910216
## [5,] 19.813735 0.66366031
```

```
#output analysis
sigma.bm <- mcse.multi(chain.new)
sigma.bm[[1]]
```

```
##             [,1]       [,2]      [,3]       [,4]         [,5]
## [1,]  19.705114  -19.14527  178.9332   29.82303     7.540342
## [2,] -19.145270   24.09635 -197.3670  -40.77194    52.434593
## [3,] 178.933192 -197.36698 1829.2932  305.31402  -201.592952
## [4,]  29.823035  -40.77194  305.3140 1985.66037  -723.005274
## [5,]   7.540342   52.43459 -201.5930 -723.00527 14094.240425
```

```
corr.bm <- cov2cor(sigma.bm[[1]])
corr.bm
```

```
##             [,1]        [,2]        [,3]       [,4]        [,5]
## [1,]  1.00000000 -0.87861046  0.94245316  0.1507682  0.01430806
## [2,] -0.87861046  1.00000000 -0.94006349 -0.1863946  0.08997491
## [3,]  0.94245316 -0.94006349  1.00000000  0.1601963 -0.03970204
## [4,]  0.15076824 -0.18639456  0.16019634  1.0000000 -0.13666841
## [5,]  0.01430806  0.08997491 -0.03970204 -0.1366684  1.00000000
```

```
#Principal Component Analysis
ncores <- detectCores()
pc.trial <- prcomp(corr.bm)

# components and explained variances
var.explained <- cumsum(pc.trial$sdev^2/sum(pc.trial$sdev^2))

#scree plot
pdf("scree_var.pdf")
par(mfrow = c(1,2))
plot(pc.trial, type = "l", main = "Scree Plot")
plot(var.explained, type = "l", main = "Percent Variance Explained")
par(mfrow = c(1,1))
dev.off()
```

```
## pdf
##   2
```

```
#num.PC <- num of PCs

pca <- function(num.PC, corr.bm){
pc <- prcomp(corr.bm , rank. = num.PC)
PC <- mclapply(1:num.PC, function(i) pc[[2]][,i], mc.cores = ncores)
g <- mclapply(1:num.PC, function(i) function(x) return(PC[[i]]%*%x),
              mc.cores = ncores)

chain.scratch <- mclapply(1:num.PC, function(i) apply(chain.new,1,g[[i]]),mc.cores = ncores)
chain.final <-matrix(unlist(chain.scratch), ncol = num.PC, byrow = FALSE)
return(chain.final)
}
```

```r
chain.PCA <- mclapply(1:5, function(i) pca(i, corr.bm), mc.cores = ncores)

multess <- mclapply(1:5, function(i) multiESS(chain.PCA[[i]]), mc.cores = ncores)
miness <- mclapply(1:5, function(i) minESS(i, 0.05, 0.05))

out <- do.call(cbind, list(multess, miness))
colnames(out) <- c("MultiESS", "MinESS")
out
```

```
##      MultiESS MinESS
## [1,] 5366.525 6146
## [2,] 4683.735 7529
## [3,] 9991.679 8123
## [4,] 9901.215 8431
## [5,] 9098.187 8605
```

```r
#3 PC components is appropriate
#output analysis

pdf("acfplots_pca.pdf")
par(mfrow = c(2,3))
acf(chain.PCA[[3]][,1],main="acf plot for PC1")
acf(chain.PCA[[3]][,2],main="acf plot for PC2")
acf(chain.PCA[[3]][,3],main="acf plot for PC3")
estvssamp(chain.PCA[[3]][,1],main = "PC1 vs sample size")
estvssamp(chain.PCA[[3]][,2],main = "PC2 vs sample size")
estvssamp(chain.PCA[[3]][,3],main = "PC3 vs sample size")
par(mfrow=c(1,1))
dev.off()
```

```
## pdf
##   2
```

```r
#sequential stopping rule
# k variable

y <- mhsampler(Y,5000)[,3]
est <- mean(y)
mcse <- sd(y)/sqrt(length(y))
interval <- est + c(-1,1)*1.96*mcse

eps <- 0.05
len <- diff(interval)
out <- c(est, interval)

while (len > eps){
  y.new <- mhsampler(Y,5000)[,3]
  y <- cbind(y, y.new)
  est <- mean(y)
  mcse <- sd(y)/sqrt(length(y))
  interval <- est + c(-1,1)*1.96*mcse
  len <- diff(interval)
  out <- rbind(out, c(est,interval))

}
```

```r
temp <- seq(5000, length(y), 5000)

pdf("stoppingrule.pdf")
plot(temp, out[,1], type = "l")
points(temp, out[,2], type = "l", col = "red")
points(temp, out[,3], type = "l", col = "red")
dev.off()
```

```
## pdf
##   2
```