Financial Econometrics

R Commands Used in Lecture 12

Prof. Hamed Ghoddusi

2019

MCMC Codes

The codes in the next slides are obtained from the following sources:

- Gibbs Sampler in R
- A simple Metropolis-Hastings MCMC in R
- Another source

Gibbs Sampler

```
# summary statistics of sample
n <- 30
ybar <- 15
s2 <- 3
# sample from the joint posterior (mu, tau | data)
mu <- tau <- rep(NA, 11000)
T <- 1000    # burnin
tau[1] <- 1  # initialisation
for(i in 2:11000)
{
   mu[i] <- rnorm(n = 1, mean = ybar, sd = sqrt(1 / (n * tau[i - 1])))
   tau[i] <- rgamma(n = 1, shape = n / 2, scale = 2 / ((n - 1) * s2 + n * (mu[i] - ybar)^2))
}
mu <- mu[-(1:T)]   # remove burnin
hist(mu)
hist(tau)
```

Metropolis-Hastings (MH): Create a Sample Data

```
trueA <- 5
trueB <- 0
trueSd <- 10
sampleSize <- 31
# create independent x-values
x <- (-(sampleSize-1)/2):((sampleSize-1)/2)
# create dependent values according to ax + b + N(0,sd)
y <-  trueA * x + trueB + rnorm(n=sampleSize,mean=0,sd=trueSd)
plot(x,y, main="Test Data")
```

Plot the Likelihood Function

```
likelihood <- function(param){
  a = param[1]
  b = param[2]
  sd = param[3]
  pred = a*x + b
  singlelikelihoods = dnorm(y, mean = pred, sd = sd, log = T)
  sumll = sum(singlelikelihoods)
  return(sumll)
}
# Example: plot the likelihood profile of the slope a
slopevalues <- function(x){return(likelihood(c(x, trueB, trueSd)))}
slopelikelihoods <- lapply(seq(3, 7, by=.05), slopevalues )
plot (seq(3, 7, by=.05), slopelikelihoods , type="l", xlab = "values of slope parameter a",
ylab = "Log likelihood")
```

Define Prior

```
# Prior distribution
prior <- function(param){
    a = param[1]
    b = param[2]
    sd = param[3]
    aprior = dunif(a, min=0, max=10, log = T)
    bprior = dnorm(b, sd = 5, log = T)
    sdprior = dunif(sd, min=0, max=30, log = T)
    return(aprior+bprior+sdprior)
}
```

Generate Posterior

```
posterior <- function(param){
  return (likelihood(param) + prior(param))
}
```

Metropolis algorithm

```
proposalfunction <- function(param){
   return(rnorm(3,mean = param, sd= c(0.1,0.5,0.3)))
}
 run_metropolis_MCMC <- function(startvalue, iterations){
   chain = array(dim = c(iterations+1,3))
   chain[1,] = startvalue
   for (i in 1:iterations){
      proposal = proposalfunction(chain[i,])
       probab = exp(posterior(proposal) - posterior(chain[i,]))
      if (runif(1) < probab){
         chain[i+1,] = proposal
      }else{
         chain[i+1,] = chain[i,]
      }
   }
   return(chain)
}
```

Metropolis algorithm

```
startvalue = c(4,0,10)
chain = run_metropolis_MCMC(startvalue, 10000)

burnIn = 5000
acceptance = 1-mean(duplicated(chain[-(1:burnIn),]))
```

Plot the Summary

```
par(mfrow = c(2,3))
hist(chain[-(1:burnIn),1],nclass=30, , main="Posterior of a", xlab="True value = red line" )
abline(v = mean(chain[-(1:burnIn),1]))
abline(v = trueA, col="red" )
hist(chain[-(1:burnIn),2],nclass=30, main="Posterior of b", xlab="True value = red line")
abline(v = mean(chain[-(1:burnIn),2]))
abline(v = trueB, col="red" )
hist(chain[-(1:burnIn),3],nclass=30, main="Posterior of sd", xlab="True value = red line")
abline(v = mean(chain[-(1:burnIn),3]) )
```

Plot the Summary

```
abline(v = trueSd, col="red" )

plot(chain[-(1:burnIn),1], type = "l", xlab="True value = red line" , main = "Chain values of a", )

abline(h = trueA, col="red" )

plot(chain[-(1:burnIn),2], type = "l", xlab="True value = red line" , main = "Chain values of b", )

abline(h = trueB, col="red" )

plot(chain[-(1:burnIn),3], type = "l", xlab="True value = red line" , main = "Chain values of sd", )

abline(h = trueSd, col="red" )

# for comparison:

summary(lm(y~x))
```

MCMC in R

- You can also use MCMC package to save direct coding
- library(mcmc)
- Example for Metropolis:

MCMC in R

- > library(mcmc)
- > data(logit)
- > out <- glm(y ~ x1 + x2 + x3 + x4, data = logit, family = binomial(), x = TRUE)
- > summary(out)

MCMC in R

- > x <- out$x
- > y <- out$y
- > lupost <- function(beta, x, y) { +   eta <- as.numeric(x %*% beta)

```
+   logp <- ifelse(eta < 0, eta - log1p(exp(eta)), - log1p(exp(- eta)))
+   logq <- ifelse(eta < 0, - log1p(exp(eta)), - eta - log1p(exp(- eta)))
+   logl <- sum(logp[y == 1]) + sum(logq[y == 0])
+  return(logl - sum(beta^2) / 8) }
```

MCMC in R

- > set.seed(42)   # to get reproducible results
- > beta.init <- as.numeric(coefficients(out))
- > out <- metrop(lupost, beta.init, 1e3, x = x, y = y)
- > names(out)