MCMC I Methods

Vladimir Minin, Kari Auranen, M. Elizabeth Halloran
Summer Institute in Statistics and Modeling in Infectious Diseases, July 2016

1 Introduction to R and Bayes programming

1.1 Simple Beta posterior distribution

The goal is here to learn simple R programming commands relevant to introductory Bayesian methods. In this first exercise, we compute the posterior distribution of the transmission probability. The sampling distribution is binomial, the prior distribution is Beta, so the posterior distribution is Beta. You can use the command `help(dbeta)` in R to learn more about this function.

Let's see how the posterior distribution of the transmission probability depends on the amount of data given a uniform prior distribution (Sample mean $y/n = 0.40$).

n, number exposed	y, number infected
5	2
20	8
50	20
1000	400

```
##Simple Beta posterior distribution of the transmission probability

## R program to compute the posterior distribution of the transmission probability
## Beta prior distribution of the binomial likelihood

## We want to evaluate the density of the posterior of p along the interval [0,1]
## To start, generate a sequence from 0 to 1 in increments of .01 that will supply
## the values where we will evaluate the posterior density

x = seq(0,1, by = .01)
x

## Observed data
## Generate a vector of the observed number of trials in the four experiments
n=c(5,20,50,1000)
n

## Generate a vector of the number of successes (infections) in the four experiments
```

```
y=c(2,8,20,400)
y
##Set up noninformative Beta prior distributions
my.alpha = 1
my.beta = 1
my.alpha
my.beta

##Set up a matrix with 4 rows and the number of columns that is the length of the
## x vector where the values of the posterior densities  will be evaluated. This
## matrix will hold the values for the four posterior densities. The value
## 0 is a place holder. Other entries could be used.

posta = matrix(0, nrow=4, ncol = length(x))

##plot the four posterior densities using different amounts of data from
## the four experiments

## open pdf (or ps) file graphics device
#pdf(file="/Users/betz/Documents/Bayesintro/betaunif1.pdf", height=6.5, width = 8.9)
## set up to have 4 plots in one figure with 2 by 2

par(mfrow=c(2,2))
# loop through the for graphs. Use a for loop

for (i in 1:4){
  posta[i,] = dbeta(x,my.alpha+y[i],my.beta+n[i]-y[i])
  plot(x,posta[i,], type = "l", ylab ="Posterior Density", xlab="p")
}
## close graphics device if using pdf (or ps)
#dev.off()
#return to 1 plot if need be
par(mfrow=c(1,1))
```

1.2 Summaries of the posterior distribution

After obtaining the posterior distribution, we might be interested in certain summary measures such as the posterior mean or posterior median. We might want the 95% posterior interval, equitailed, or any quantiles of interest. We could also ask what is the posterior probability that $p > 0.5$.

```
## Posterior summaries using closed form distributions
#prior mean
priormean = my.alpha/(my.alpha + my.beta)
priormean
[1] 0.5
# posterior mean
postmean= (my.alpha+y)/(my.alpha+my.beta+n)
postmean
[1] 0.4285714 0.4090909 0.4038462 0.4001996

round(postmean,4)
[1] 0.4286 0.4091 0.4038 0.4002

#posterior median
# use qbeta to get the values at the given quantiles
postmedian=qbeta(0.5, my.alpha+y, my.beta+n-y)
round(postmedian,4)

[1] 0.4214 0.4063 0.4026 0.4001

#median, 95%  equitailed posterior or credible interval
# use qbeta to get the values at the given quantiles
# set up matrix
post95int=matrix(0,4,3)
for (i in 1:4){
  post95int[i,] = qbeta(c(0.5, 0.025,0.975), my.alpha+y[i], my.beta+n[i]-y[i])
}
round(post95int,3)

[,1]  [,2]  [,3]
[1,] 0.421 0.118 0.777
[2,] 0.406 0.218 0.616
[3,] 0.403 0.276 0.539
[4,] 0.400 0.370 0.431
```

1.3 Random sampling from the posterior distribution

The function `rbeta()` is used to generate random draws from a given beta distribution. Here we draw 5000 random samples from the four posterior distributions in the first exercise based on different amounts of data and a uniform Beta prior.

```
# Drawing random samples from the posterior distributions to imitate results
# of an MCMC output; use rbeta() command for random samples from a beta distribution

#Set the number of samples
nsamp=5000
#Set up matrix
post=matrix(0,4,nsamp)

##pdf(file="/Users/betz/Documents/TexWork/MCMC/betz/Bayesintro/betaunif1r.pdf", height=6.5, wi

par(mfrow=c(2,2))
for (i in 1:4){
  post[i,]=rbeta(nsamp,my.alpha+y[i],my.beta+n[i]-y[i])
  hist(post[i,], xlim=c(0,1), xlab = "p ", ylab = "Frequency")
}
#dev.off()
par(mfrow=c(1,1))
```

1.4 Posterior summary using samples of posterior

Now we can get the posterior means using the samples of the posterior distributions and compare them with the analytic posterior means. We can also use the function summary() to get posterior summaries.

```
#Posterior summaries using random samples
# posterior mean
postmeanr =  apply(post,1,mean)
postmeanr

#Compare with analytic posterior mean
postmean
```

```
# Get summary of simulated posterior distributions row by row
summary(post[1,])
summary(post[2,])
summary(post[3,])
summary(post[4,])

# Get all summaries of all four rows at the same time
apply(post, 1, summary)
```

1.5 Using informative conjugate priors

Let's assume we have more prior information, or stronger prior beliefs about the transmission probability p. See how it affects posterior inference about one data set, $n1 = 50$, $y1 = 20$ (sample mean = 0.40).

Prior mean	Prior sum		
$\frac{\alpha}{\alpha+\beta}$	$\alpha+\beta$	α	β
0.50	2	1	1
0.50	4	2	2
0.50	100	50	50
0.60	5	3	2
0.60	20	12	8
0.60	100	60	40
0.80	5	4	1
0.80	20	16	4

```
## Now use different informative conjugate priors

alpha1 = c(1,2,50,3,12,60,4,16)
beta1 = c(1,2,50,2,8,40,1,4)

priorsum = alpha1+beta1
priormean = alpha1/(alpha1+beta1)
priorsum
priormean

n1 = 50
y1 = 20

post95int2 = matrix(0,length(alpha1),3)
```

```
for (i in 1:length(alpha1)){
 post95int2[i,]  = qbeta(c(0.5,0.025,0.975), alpha1[i]+y1, beta1[i]+n1-y1)
}
round(post95int2,3)
      [,1]       [,2]     [,3]
[1,] 0.403 0.276 0.539
[2,] 0.406 0.281 0.540
[3,] 0.467 0.388 0.547
[4,] 0.417 0.292 0.550
[5,] 0.457 0.343 0.574
[6,] 0.533 0.453 0.612
[7,] 0.436 0.309 0.568
[8,] 0.514 0.398 0.630
```

2 Writing a function in R

Suppose we want to be able to write our own function that can be called repeatedly so that we do not need to write the code every time. This is the standard approach in R programming. Here we will write a simple function to compute the posterior distribution of the transmission probability as an illustration. The inputs to the function will be n, the number of trials, y, the number of infections, and α and β from the Beta prior distribution. The function plots the posterior, and returns the posterior median and 95% posterior interval.

```
## Here is a simple function to compute the posterior distribution of the transmission
## probability. It plots the posterior distribution and returns the posterior median and
## 95% equitailed credible interval
## n = number of observations,
## y = number of successes (infections)
## alpha, beta, parameters of the prior Beta distribution

### Return the values at the end.
mybeta2=function(n,y,alpha,beta){
          x=seq(0,1,by=.01)
          postbeta=dbeta(x,alpha+y,beta+n-y)
          plot(x,postbeta, type="l", ylab="Posterior Density ", xlab="p ")
          mybeta=qbeta(c(0.5, 0.025,0.975), alpha+y, beta+n-y)
          return(mybeta)
}
```

```
test2=mybeta2(50,20,1,1)
test2

### Alternatively, you can return a list at the end.

mybeta3=function(n,y,alpha,beta){
          x=seq(0,1,by=.01)
          postbeta=dbeta(x,alpha+y,beta+n-y)
          plot(x,postbeta, type="l", ylab="Posterior Density ", xlab="p ")
          mybeta=list(answer=qbeta(c(0.5, 0.025,0.975), alpha+y, beta+n-y))
}

test3=mybeta3(50,20,1,1)
test3
test3$answer

### Alternatively, you can return a list with more than one object at the end.

mybeta4=function(n,y,alpha,beta){
  x=seq(0,1,by =.01)
  postbeta=dbeta(x,alpha+y,beta+n-y)
  plot(x,postbeta, type="l", ylab="Posterior Density ", xlab="p ")
  answer1=qbeta(.5,alpha+y, beta+n-y)
  answer2=qbeta(.025,alpha+y, beta+n-y)
  answer3=qbeta(.975,alpha+y, beta+n-y)
  mybeta=list(answer1,answer2,answer3))
}

test4=mybeta4(50,20,1,1)
print(test4)
##extract the first element in the list test4
test4[[1]]
```