Report

Margarita, Orlova, Kenneth Ruiter, Patrick Walker

November 3, 2018

Metropolis-Hastings

In this part of the report, we will use the Metro-polis Hastings algorithm to draw samples from a Beta distribution ϕ, with parameters $(6, 4)$. Of course, this is usually not necessary as most languages are already able to draw from a Beta distribution, but this way we are able to compare our results easily with the actual distribution. However, before we start, we will describe the algorithm itself.

Algorihm Description

The Metropolis-Hastings algorithm is a widely used statistical algorithm to generate samples from a distribution with a given probability density function (pdf). The algorithm uses a proposal distribution, that we are able to draw from, to obtain candidates for the actual sample draws. These candidates are then either accepted or rejected, according to a certain acceptance ratio. This ratio is calculated as follows:

$$r = \frac{p(\phi_{prop}|y)/J_+(\phi_{prop}|\phi_{old})}{p(\phi_{old}|y)/J_+(\phi_{old}|\phi_{prop})}$$

The value of r can actually be bigger than 1, but that just means the candidate will definitely be accepted. Here, $p(x|y)$ is the pdf of the distribution that we want to draw from, with the parameters y (possibly a vector). The function $J_+(x|y)$ is the pdf of the proposal (or jumping) distribution, given the parameters y. Finally, ϕ_{prop} is the candidate and ϕ_{old} is the value of ϕ used to find the candidate. For this algorithm we need a starting value, which we will randomly draw from a uniform distribution in between 0 and 1. We also need a proposal distribution, which in this case will be a Beta distribution with parameters $(c\phi_{old}, c(1-\phi_{old}))$, where c is a constant. The number of times a candidate is drawn is a formerly specified number of iterations. Both the constant c and the number of iterations are used as an input of the function. Every accepted value of ϕ is stored in a chain, which is the output of the algorithm. The code for the function, including comments is shown below.

```
Metropolis <- function(c,iteration){ #Setting constant c and number of iterations

  if(c <= 0){stop("Input c must be greater than 0")} #prevents nonsensical user inputs

  if(iteration <= 0){stop("Input iterations must be greater than 0")}
      #prevents nonsensical user inputs

  fi=runif(1,0,1) #starting value for "Phi"

  chain = array(dim=c(iteration+1,1)) #initialzing array

  chain[1] = fi #putting initial value into chain

  for (i in 1:iteration){ #for loop for each iteration of the chain

    proposal = rbeta(1,c*chain[i],c*(1-chain[i])) #selects a random draw from the beta
          #distribution using parameter from chain as the proposal value
```

```
    acceptance_ratio = (dbeta(proposal,6,4)/dbeta(proposal,c*chain[i],c*(1-chain[i])))/
      (dbeta(chain[i],6,4)/dbeta(chain[i],c*proposal,c*(1-proposal)))
       #the probability of acceptance, acceptance ratio has been corrected for
       #asymmetry of jumping function
    if (runif(1) < acceptance_ratio){
       #this combined if/else statment accepts the proposal if the acceptance ratio
       #is greater than a random draw from the uniform function.
      chain[i+1,] = proposal}

    else{

      chain[i+1,] = chain[i,]

      }
  }
  return(chain) #done
}
```

Results

Now that the algorithm itself has been discussed, we will run the function with $c = 1$ and $10,000$ total iterations. We will not discard any initial samples, which is sometimes done to prevent bias introduced by the starting value. The following plots show the results.