Solution Lab 22

June 2, 2016

Exercise 8

```r
# n : number of iterations
# p : (1+r)*(m+1) matrix where p[i,j]=P_i(j)
# x0 : initial state

run.Hasting <- function(n,p,x0){

  r <- dim(p)[1]-1
  m <- dim(p)[2]-1

  if(length(x0) != m+1 | sum(x0<0) > 0 | sum(x0) != r){
    print('Wrong initial position')
    break}

# Initialization
  Xn <- x0
  counts <- 1

# Computation of the Markov Chain with transition probability q
  while(counts <= n){
    X <- Xn
    rn1 <- ceiling(length(X[X>0])*runif(1))
    X[X>0][rn1] <- X[X>0][rn1] - 1
    rn2 <- ceiling(length(X)*runif(1))
    X[rn2] <- X[rn2] + 1

# Computation of the stationary probabilities at Xn and X
    piX <- 1
    piXn <- 1
    for(j in 1:(m+1)){
      piX <- piX*p[X[j]+1,j]
      piXn <- piXn*p[Xn[j]+1,j]
    }

# Changing or not the state Xn
    U <- runif(1)
    test <- ( piX/((m+1)*sum(X>0)) )/(piXn/((m+1)*sum(Xn>0)))
    if(U<test){
      NS <- X
    }else{
      NS <- Xn
    }
  counts <- counts+1
  Xn <- NS
  }
  return(Xn)
}
```

Exercise 9

a)

We use Example 12d in the textbook. Fitting our problem we have that $(\lambda_1, \lambda_2, \lambda_3) = (1, 1/2, 1/3)$.

```r
# n : number of iterations
# c : value whom the summation of the state vector is larger
# lambda : vector of rate parameters for the exponential variables
# x0 : intial state

run.expovector <- function(n,c,lambda,x0){

  if(length(x0) != 3 | sum(x0)<c){
    print('Wrong initial position')
    break}

  Xn <- x0
  counts <- 1
  while(counts <= n){
    i <- ceiling(3*runif(1))
    Xn[i] <- -log(runif(1))/lambda[i]+max(0,c-sum(Xn[1:3!=i]))
    counts <- counts+1
  }
  return(Xn)
}
```

b)

We cannot use the previous algorithme anymore since the conditionnal event is different. Thus we use the Gibbs Sampler method.

```r
# n : number of iterations
# c : value whom the summation of the state vector is lower
# lambda : vector of rate parameters for the exponential variables
# x0 : intial state

run.Gibbs <- function(n,c,lambda,x0){
  if(length(x0) != 3 | sum(x0)>c){
    print('Wrong initial position')
    break}

  Xn <- x0
  counts <- 1
  while(counts <= n){
    i <- ceiling(3*runif(1))

# Generate the exponential variable Xn[i] given that Xn[i] < c-sum(X_n[1:3!=i])
# by inverse method
    Xn[i] <- qexp(runif(1)*(c-sum(X_n[1:3!=i])),lambda[i])
    counts <- counts+1
  }
  return(Xn)
}
```

Assuming that we can generate the random vector $(X_1, 2X_2, 3X_3)$ given $\{X_1 + 2X_2 + 3X_3 > (<)c\}$, it becomes clear how to estimate $\mathbb{E}\left[X_1 + 2X_2 + 3X_3 \middle| X_1 + 2X_2 + 3X_3 > (<)c\right]$.

Exercise 10

We use Example 12b.

```r
# n : number of iterations
# N : r sized vector with the colors repartition
# m : number of balls withdrawn
# x0 : initial state
run.urn <- function(n,N,m,x0){
  r <- length(N)

  if(length(x0) != r | sum(x0 == 0 ) > 0 | sum(x0)!=m ){
    print('Wrong initial position')
    break}
  Xn <- x0
  counts <- 1
  while(counts <= n){
    X <- Xn

# Replace one ball inside the urn uniformly chosen among the selection
    rn1 <- ceiling(r*runif(1))
    X[rn1] <- X[rn1] - 1

# We add a ball of color i in the selection with probability p_i
    p <- (N-X)/(sum(N)-m+1)
    fp <- cumsum(p)
    rn2 <- which.min(runif(1) > fp)
    X[rn2] <- X[rn2] + 1

# Xn jumps to the next state if all color type are represented in the new one
    if(sum(X == 0) == 0){
      Xn <- X
    }
    counts <- counts + 1
  }
  return(Xn)
}
```

Exercise 12

We use Example 12g.

```r
# n : number of iterations
# alpha,beta,lambda : parameters
# x0 : initial state
run.step <- function(n,alpha,beta,lambda,x0){
  if(length(x0) != 3 | sum(x0 == 0 ) > 0 | x0[1] > x0[3] ){
    print('Wrong initial position')
    break}

  Xn <- x0
  counts <- 1
  while(counts <= n){

# Conditionally on (Xn[2],Xn[3]), Xn[1] admits the probability mass
# function/vector pX on 0,...,Xn[3]
    pX <- 1/(factorial(Xn[3]-0:Xn[3])*factorial(0:Xn[3]))
    fpX <- cumsum(pX)
    Xn[1] <- which.min(runif(1) > fpX)

# Conditionally on (Xn[1],Xn[3]), Xn[2] follows a beta distribution with parameters
# alpha+Xn[1] and Xn[3]*Xn[1]+beta
    Xn[2] <- rbeta(1,alpha+Xn[1],Xn[3]*Xn[1]+beta)

# Conditionally on (Xn[1],Xn[2]), Xn[3]-Xn[1] follows a Poisson distribution with
# parameters lambda
    Xn[3] <- rpois(1,lambda) + Xn[1]

    counts <- counts + 1
    }
  return(Xn)
  }
```