Computational Statistics - Suggested Solution for Exam

Frank Miller, IDA, Linköping University

2025-05-16

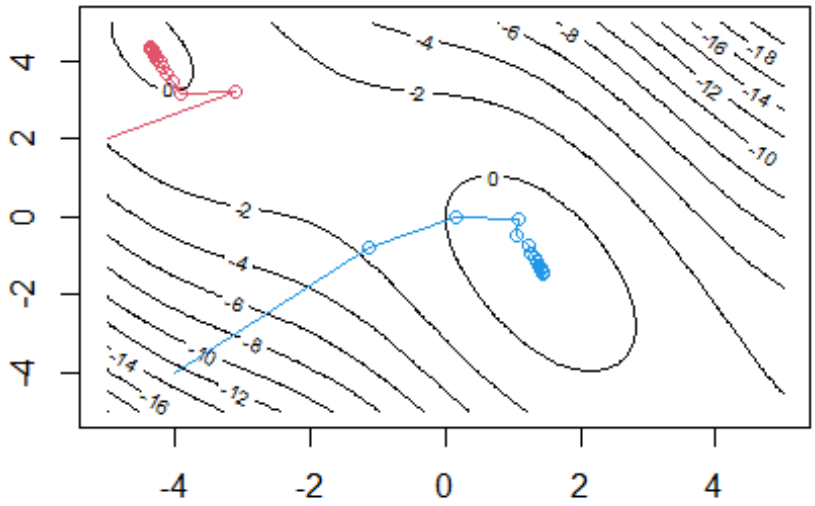
Question 1

Question 1a and 1b

The contour plot below was not necessary, but can be helpful. An alternative which still achieves full marks is to plot the iterations in an otherwise empty plot.

```
g <- function(x){
  sin(x[1])-(x[2]+x[1])^2/5-x[1]^2/25
}
glength <- 200
range   <- 5
x1grid  <- seq(-range, range, length=glength)
x2grid  <- seq(-range, range, length=glength)
grid    <- expand.grid(x1grid, x2grid)

ggrid   <- matrix(apply(grid, 1, g), nrow=glength)
contour(x1grid, x2grid, ggrid)

gradient <- function(x){
  gradx <- cos(x[1])-(x[1]+x[2])*2/5-x[1]*2/25
  grady <- -(x[1]+x[2])*2/5
  c(gradx, grady)
}

steepestascent <- function(start, alpha0=1, eps=1e-6, col=2){
  alpha  <- alpha0
  x      <- start
  change <- 999
  while(change>eps){
    xnew <- x + alpha*gradient(x)
    while (g(xnew)<g(x)){
      alpha <- alpha/2
      xnew <- x + alpha*gradient(x)
    }
    lines(rbind(x, xnew), col=col)
    points(t(xnew), col=col)
    change <- sum((x-xnew)^2)
    x <- xnew
  }
  x
```

```
}
res1 <- steepestascent(c(-5, 2))
res2 <- steepestascent(c(-4, -4), col=4)
```




```
g1 <- g(res1);  g1
## [1] 0.1781727
g2 <- g(res2);  g2
## [1] 0.9086218
if (g1 > g2) print(res1) else print(res2)
## [1]  1.453527 -1.452183
```

We see that the second run results in a higher value of g. Therefore, relying on the information of these two runs (according to instructions here), we conclude that the global maximum is achieved at $(1.45, -1.45)$.

Question 2

The user can specify starting values as parameters of the function, e.g., after looking at a histogram.

The stopping criterion is changed. The expected joint log likelihood (Q-function) is

$$Q = \sum_{j=1}^{n}\sum_{i=1}^{3} \hat{\pi}_{ij}\left\{\log(\hat{p}_i) + \log\left(\varphi(x_j;\hat{\mu}_i;\hat{\sigma}_i)\right)\right\} = \sum_{j=1}^{n}\left\{\sum_{i=1}^{3} \hat{\pi}_{ij}\log\left(\hat{p}_i \cdot \varphi(x_j;\hat{\mu}_i;\hat{\sigma}_i)\right)\right\}$$

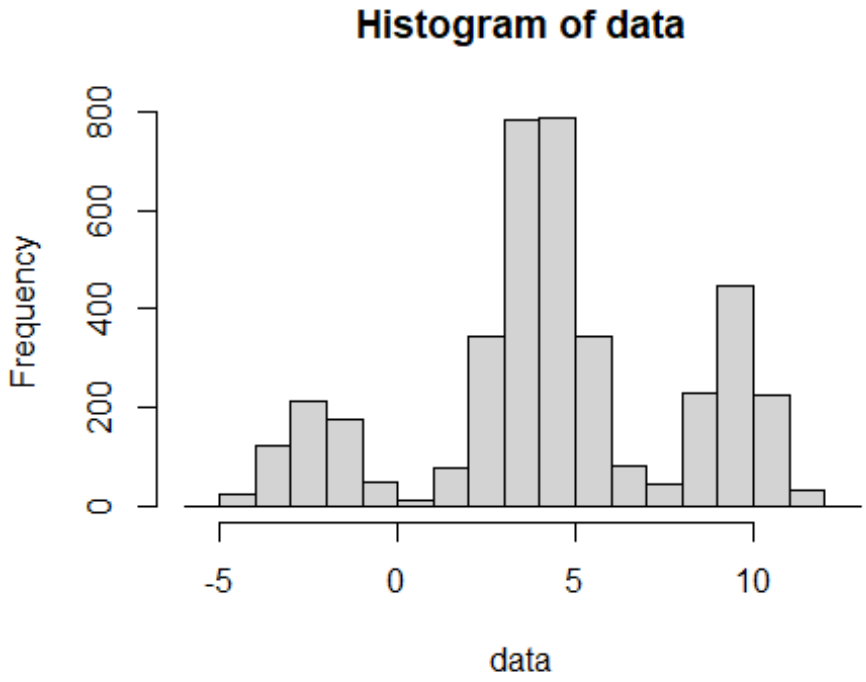
The expression in the last big {}-parentheses is calculated in the loop for the E-step (where we have already the relevant quantities), stored as qv[j], and the sum of the vector-elements qv[j] is then the Q-function.

```
set.seed(2025)
data <-
read.csv("C:\\Users\\frami01\\Dropbox\\Teaching\\CompStat\\Exams\\mixturedat.
csv")[, 1]

emalg <- function(dat, p1, p2, mu1, mu2, mu3, sigma1, sigma2, sigma3,
eps=0.000001){
  n      <- length(dat)
  pi1    <- rep(NA, n)  # init vector for prob. to belong to group 1
  pi2    <- rep(NA, n)  # init vector for prob. to belong to group 2
  qv     <- rep(NA, n)  # init vector for Q-function per observation

  pv     <- c(p1, p2, mu1, mu2, mu3, sigma1, sigma2, sigma3) # par. vec.

  cc     <- eps + 100   # init convergence criterion
  q      <- -1e9        # init Q-criterion to a bad value
  pviter <- NULL        # init the results for each iteration

  while (cc>eps){
    q1  <- q            # save previous q-criterion
    ### E step ###
    for (j in 1:n){
      pj1    <- p1*dnorm(dat[j], mean=mu1, sd=sigma1)
      pj2    <- p2*dnorm(dat[j], mean=mu2, sd=sigma2)
      pj3    <- (1-p1-p2)*dnorm(dat[j], mean=mu3, sd=sigma3)
      pi1[j] <- pj1/(pj1+pj2+pj3)
      pi2[j] <- pj2/(pj1+pj2+pj3)
      qv[j]  <- pi1[j]*log(pj1)+pi2[j]*log(pj2)+pj3/(pj1+pj2+pj3)*log(pj3)
    }
    ### M step ###
    p1     <- mean(pi1)
    p2     <- mean(pi2)
    mu1    <- sum(pi1*dat)/(p1*n)
    mu2    <- sum(pi2*dat)/(p2*n)
    mu3    <- sum((1-pi1-pi2)*dat)/((1-p1-p2)*n)
    sigma1 <- sqrt(sum(pi1*(dat-mu1)*(dat-mu1)/(p1*n)))
    sigma2 <- sqrt(sum(pi2*(dat-mu2)*(dat-mu2)/(p2*n)))
    sigma3 <- sqrt(sum((1-pi1-pi2)*(dat-mu3)*(dat-mu3)/((1-p1-p2)*n)))
    q      <- sum(qv)
    ######
    pv     <- c(p1, p2, mu1, mu2, mu3, sigma1, sigma2, sigma3, q)
```

```
    pviter <- rbind(pviter, pv)
    cc     <- abs(q-q1)  # convergence criterion, based on Q
  }
  list(pv=pv, pviter=pviter)
}
```

Question 2b

```
hist(data)
```



In the histogram, we see that starting mean values around -3, 4, and 10 are reasonable with variance of around 1 for each class and the middle class is larger

```
res <- emalg(data, p1=0.25, p2=0.5, mu1=-3, mu2=4, mu3=10, sigma1=1,
sigma2=2, sigma3=1)
res$pv
```

```
## [1]     0.1483142     0.6085053    -2.3015162     3.9926914     9.4906027
## [6]     0.9950604     1.0991093     0.8150432 -9420.5617595
```

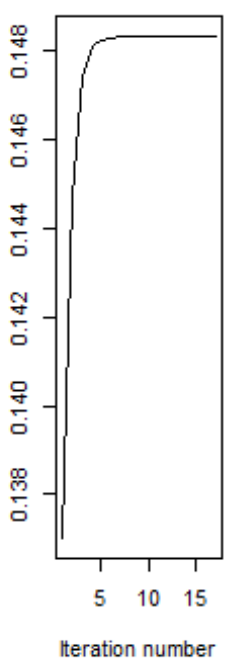
We get as estimates: $\hat{p}_1 = 0.15, \hat{p}_2 = 0.61, \hat{p}_3 = 1 - 0.15 - 0.61 = 0.24, \hat{\mu}_1 = -2.3, \hat{\mu}_2 = 4.0, \hat{\mu}_3 = 9.5, \hat{\sigma}_1 = 1.00, \hat{\sigma}_2 = 1.10, \hat{\sigma}_3 = 0.82$. (The expected log-likelihood is -9420.6.)

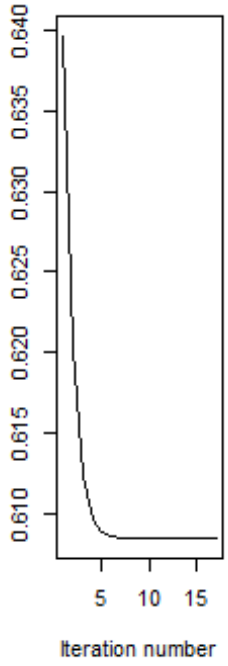
Question 2c

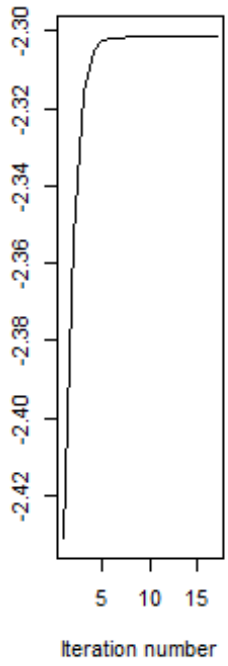
Convergence plots of the 8 parameters and of Q are below. They show that convergence is achieved (after 17 iterations).

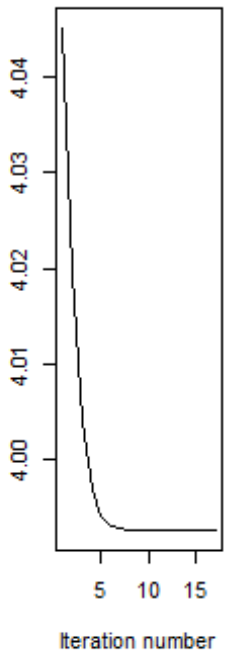
```
dim(res$pviter)[1]  # Number of iterations used
```

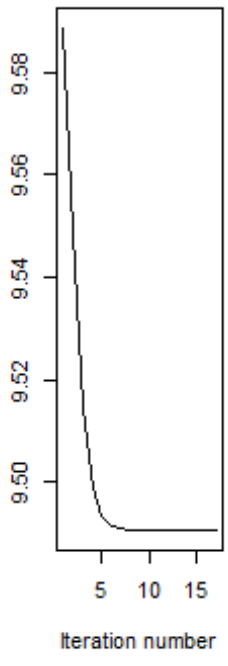
```
## [1] 17

oldpar <- par(mfrow=c(1, 3))
for (k in 1:9){
  plot(res$pviter[, k], type="l", xlab="Iteration number", ylab=" ")
}
```

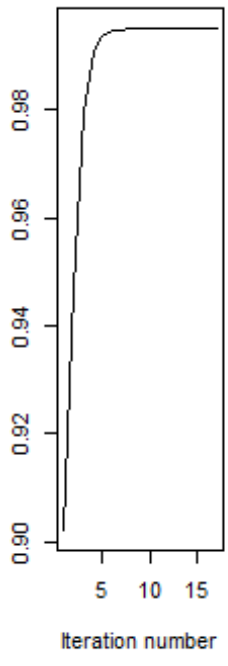

0.138
0.140
0.142
0.144
0.146
0.148
5
10
15
Iteration number

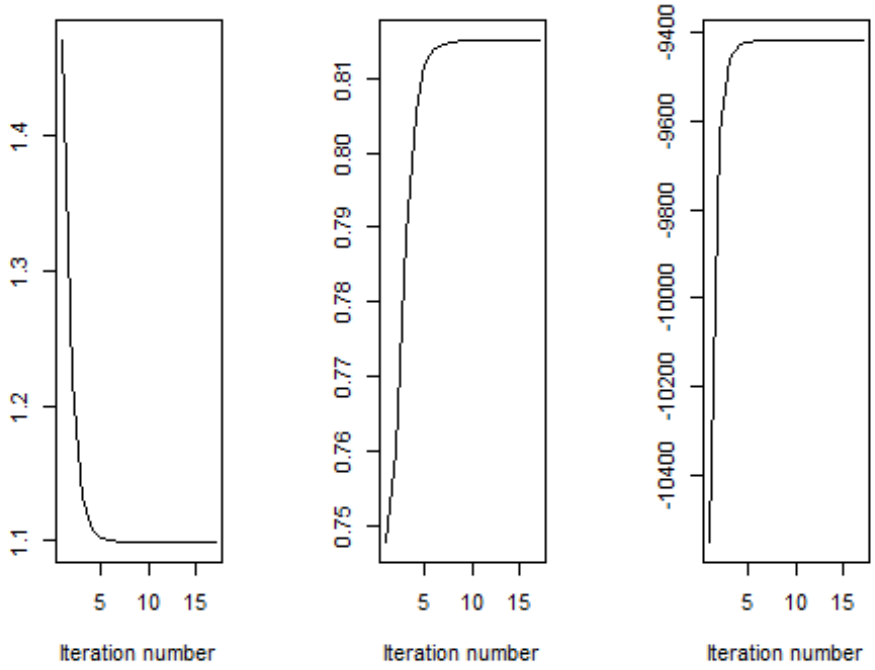

0.610
0.615
0.620
0.625
0.630
0.635
0.640
5
10
15
Iteration number


-2.42
-2.40
-2.38
-2.36
-2.34
-2.32
-2.30
5
10
15
Iteration number


4.00
4.01
4.02
4.03
4.04
5
10
15
Iteration number


9.50
9.52
9.54
9.56
9.58
5
10
15
Iteration number


0.90
0.92
0.94
0.96
0.98
5
10
15
Iteration number



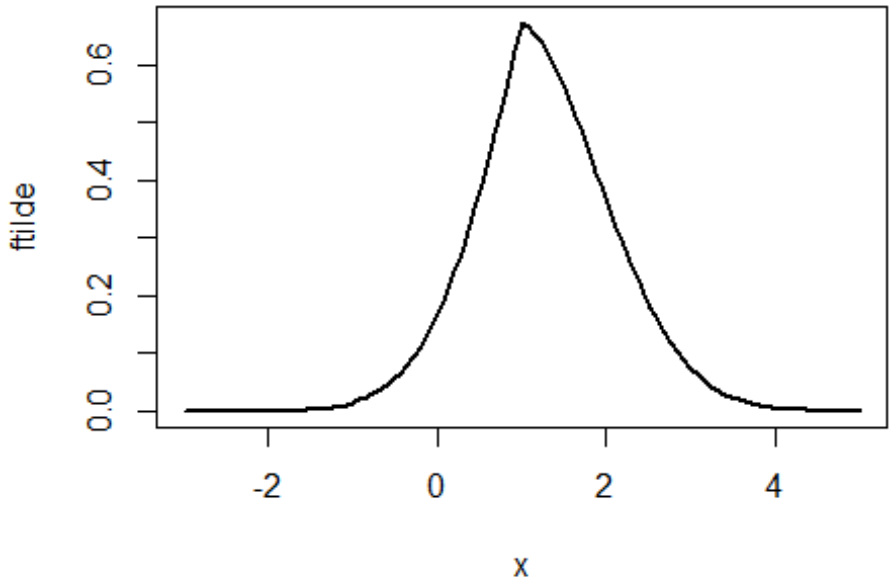

```
par(oldpar)
```

Question 3

Question 3a

We set $c = 1$ for the plot.

```
g <- function(x){
  (abs(x)>=1) * exp(-abs(x)+1) + (abs(x)<1)
}
ftilde <- function(x){
  g(x) * exp(-(x-1.89)^2/2)
}
plot(ftilde, xlim=c(-3, 5), lwd=2)
```



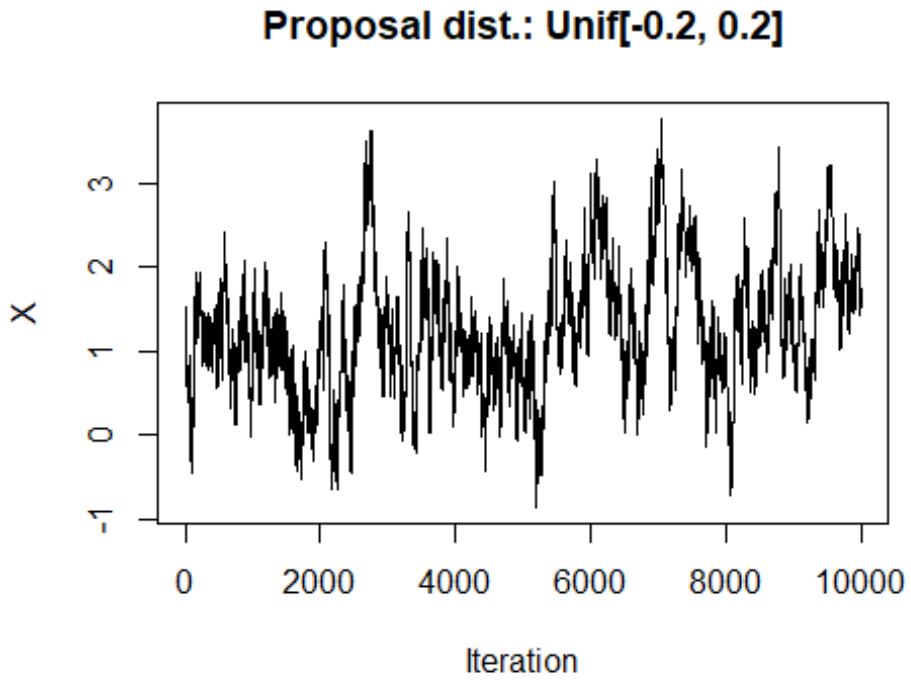
Question 3b

We use the uniform distribution as proposal distribution. Important: We use uniform distributions around 0 (i.e., on the interval $[-r, r]$) to ensure a symmetric proposal which simplifies the Metropolis-Hastings ratio $\tilde{f}(x^*)/\tilde{f}(x)$. A common mistake is here to use a non-symmetric proposal and to forget then that the Metropolis-Hastings ratio is (slightly) more complicated.

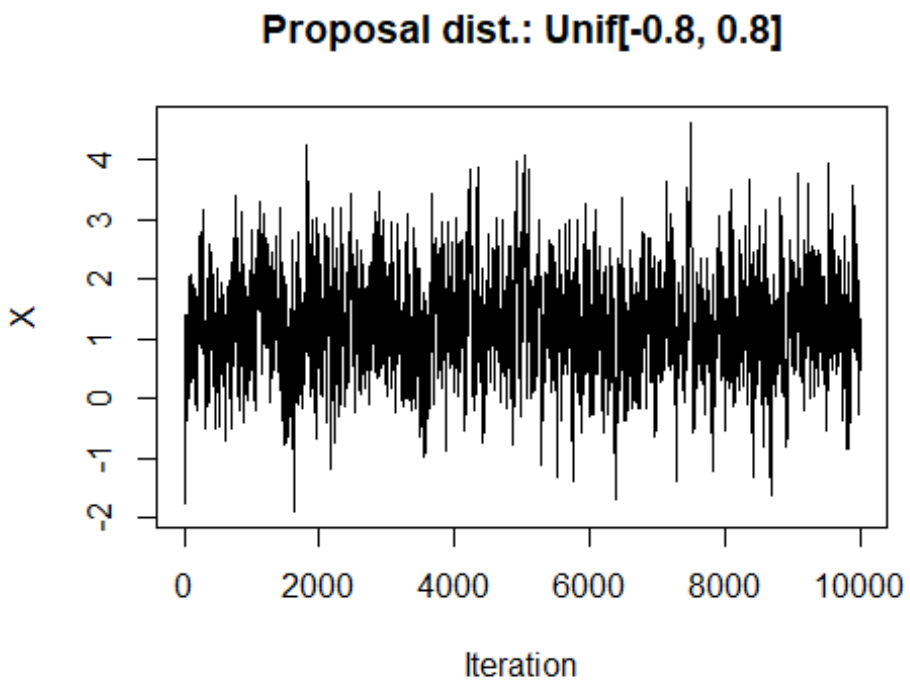
Here, we chose range $r = 0.2$ (used as default in the function below) and $r = 0.8$.

```
set.seed(2025)
metropolis <- function(x, s=10000, rang=0.2){
  samp <- c(x)
  for (i in 1:(s+50))
  {
    u     <- runif(1, min=-rang, max=rang)
    xcand <- x + u
    R     <- ftilde(xcand)/ftilde(x)
    ap    <- runif(1)
    if (ap<R) { x <- xcand }
    samp  <- rbind(samp, x)
  }
  #discart starting value samp[1] and burn-in of initial 50 simulations
samp[2:51]
  samp[52:(s+51)]
}
samp1 <- metropolis(0)
```

```r
plot(samp1, type="l", xlab="Iteration", ylab="X")
title("Proposal dist.: Unif[-0.2, 0.2]")
```



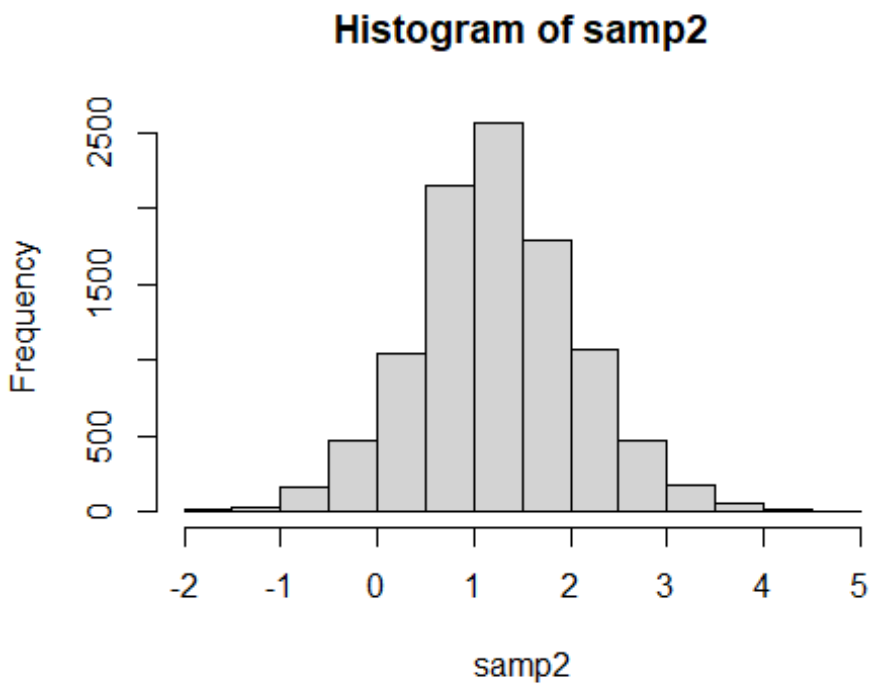

```r
samp2 <- metropolis(0, rang=0.8)
plot(samp2, type="l", xlab="Iteration", ylab="X")
title("Proposal dist.: Unif[-0.8, 0.8]")
```



The convergence plot for the second proposal distribution shows a better exploration of the distribution since we can see the simulation path in the first picture too clearly. We prefer therefore samp2 for further use. Higher ranges for the uniform proposal distribution might generate too many rejected samples; the convergence plot for $r = 0.8$ looks already good.

Question 3c

```
hist(samp2, breaks=20)
```




```
mean(samp2>1.89)
```

```
## [1] 0.2101
```

```
mean(samp2)
```

```
## [1] 1.24789
```