Computational Statistics - Suggested Solution for Exam

Frank Miller, IDA, Linköping University

2024-01-09

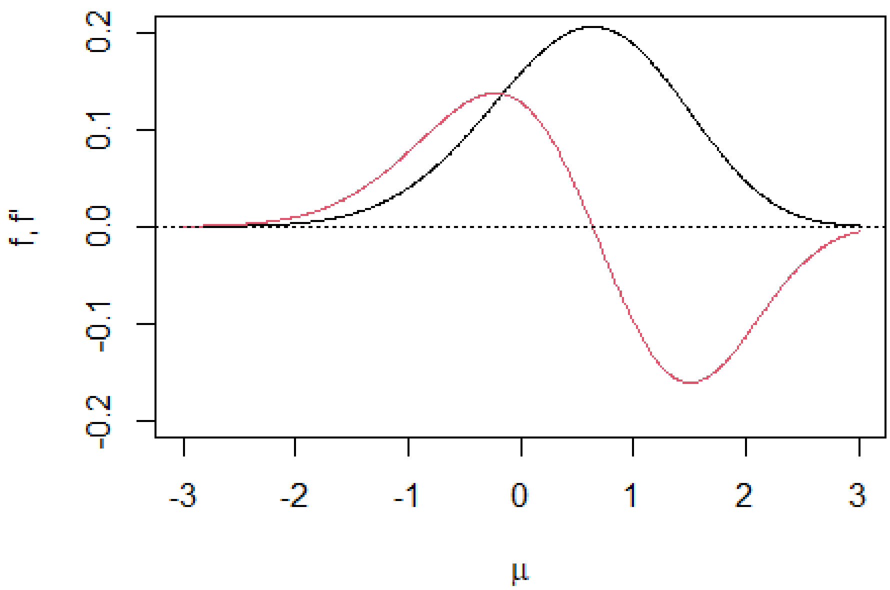
Assignment 1

Question 1.1

```
xobs <- 1.65
f <- function(mu){
  x <- xobs
  exp(-x+mu-exp(-x+mu))*exp(-mu^2/2)
}
vf <- Vectorize(f)
mu <- -300:300/100
optimize(f, c(-3, 3), maximum=TRUE)
```

```
## $maximum
## [1] 0.6369141
##
## $objective
## [1] 0.2061786
```

```
df <- function(mu){
  x  <- xobs
  e1 <- exp(-x+mu-exp(-x+mu))
  e2 <- exp(-mu^2/2)
  e1*e2*(-mu) + e1*e2*(1-exp(-x+mu))
}
vdf <- Vectorize(df)
plot(c(min(mu), max(mu)), c(-0.2, 0.2), type="n", xlab=expression(mu),
ylab="f, f'")
lines(mu, vf(mu), col=1)
lines(mu, vdf(mu), col=2)
abline(h=0, lty=3)
```



Question 1.2

```
secant <- function(mu1, mu2, eps=1e-6, printiter=0){
  count <- 0
  while (abs(mu1-mu2)>eps){
    mustar <- mu1 - df(mu1)*(mu1-mu2)/(df(mu1)-df(mu2))
    mu2    <- mu1
    mu1    <- mustar
    if (count<printiter) print(mustar)  # print approximations for the first
iterations
    count  <- count+1
  }
  mustar
}
sol1 <- secant(1, 1.2, printiter=10)

## [1] 0.5013346
## [1] 0.6378854
## [1] 0.6368766
## [1] 0.6369061
## [1] 0.6369061

print(paste("Solution 1:", round(sol1, 6)))

## [1] "Solution 1: 0.636906"

sol2 <- secant(1, 1.8, printiter=10)
```

```
## [1] -0.8214697
## [1] 0.08055599
## [1] -4.493835
## [1] -4.493853
## [1] -4.682196
## [1] -4.788694
## [1] -4.920555
## [1] -5.038775
## [1] -5.158472
## [1] -5.274191
```

```r
print(paste("Solution 2:", round(sol2, 6)))
```

```
## [1] "Solution 2: -37.580087"
```

In the unsuccessful second case, the secant through f'(1) and f'(1.8) at the low value -0.82; while the next secant still has a negative slope, the following will have a positive slope while f'>0. The algorithm will then look for a minimum and diverges slowly to -infinity. The stopping criterion is then fulfilled for mu = -37.6.

Question 1.3

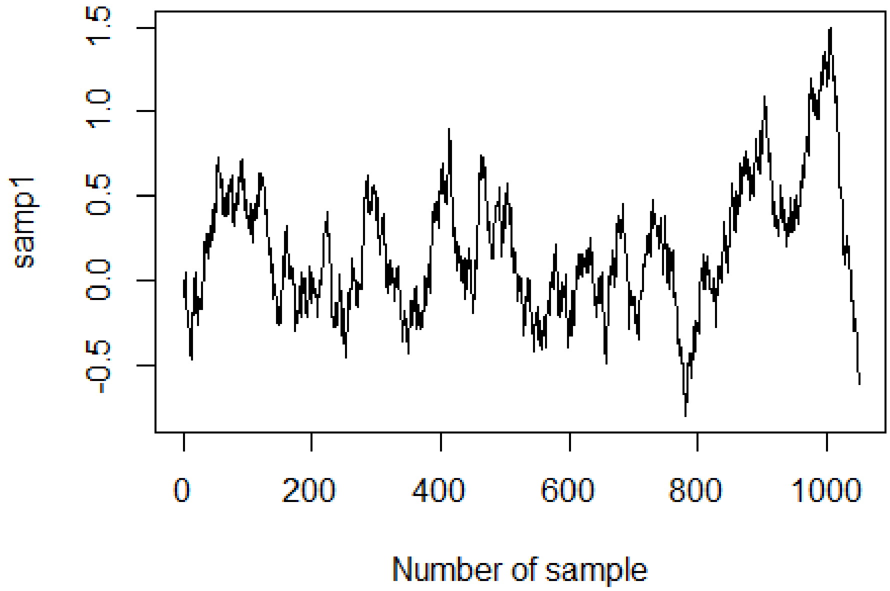
```r
mcmc <- function(xstart, n, xobs, sigma=1)
{
  x <- xstart
  sample <- x
  acc <- 0
  for (i in 1:n)
  {
    xcand <- x + sigma*runif(1, -1, 1)
    if (f(xcand)/f(x) > runif(1, 0, 1)){
      x <- xcand
      acc <- acc+1
    }
    sample <- c(sample, x)
  }
  print("Acceptance rate:")
  print(acc/n)
  sample
}
```

We will later remove a run-in period of xstart and the 50 initially generated observations, therefore we generate 1050 now:

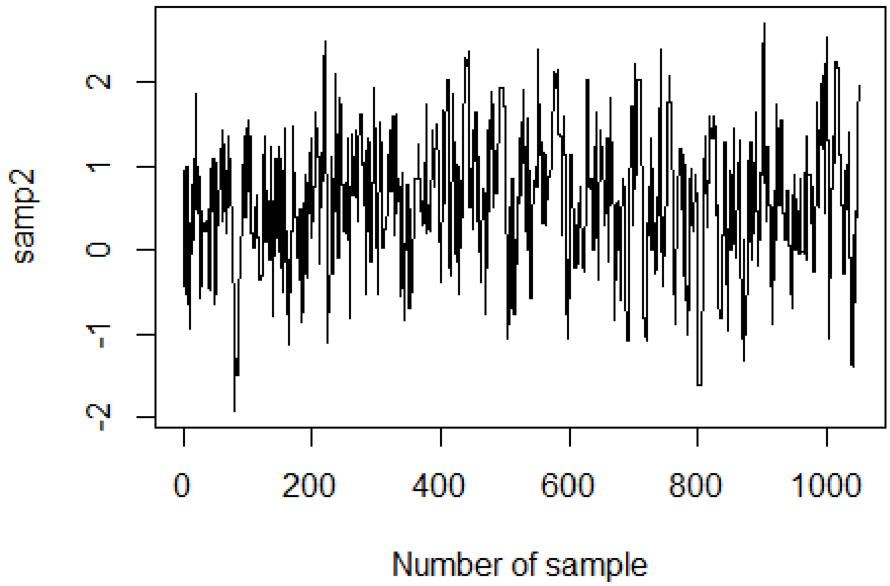
```r
samp1 <- mcmc(xstart=0, n=1050, sigma=0.15)
```

```
## [1] "Acceptance rate:"
## [1] 0.9819048
```

```r
plot(0:1050, samp1, type="l", xlab="Number of sample")
```



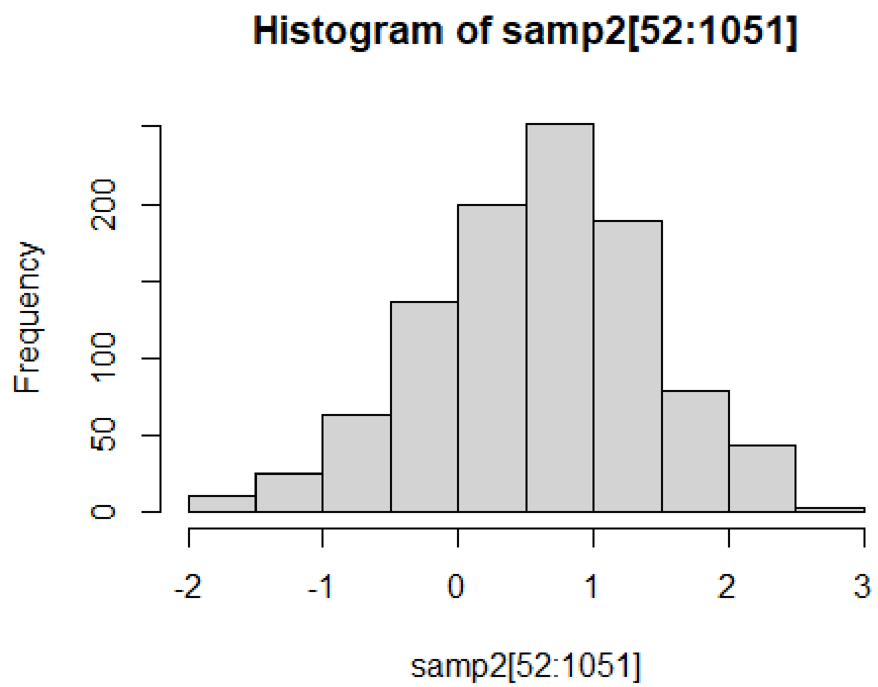

```
samp2 <- mcmc(xstart=0, n=1050, sigma=1.5)

## [1] "Acceptance rate:"
## [1] 0.6495238

plot(0:1050, samp2, type="l", xlab="Number of sample")
```



In the first case, there is too much structure suggesting that there is not yet a convergence to the target distribution; the second is much better. Further, the acceptance rate in the first case is too high (> 90%), while in the second case the acceptance rate is closer to the guidance of 44% valid for unidimensional, unimodel distributions. We choose therefore x+Unif[-1.5, 1.5] as proposal and remove the starting value and the first 50 (run-in).

Question 1.4

```
hist(samp2[52:1051])
```




```r
mean(samp2[52:1051]>=0)

## [1] 0.765
```

The above value is estimated probability $P(\mu \geq 0)$. Note that 1000 simulations is a small number for this purpose and the above estimate is still quite uncertain. Usually, one should draw a larger sample.

```r
sortsamp <- sort(samp2[52:1051])
q <- (sortsamp[50]+sortsamp[51])/2
q

## [1] -0.8271181
```

The lowest 5% of the generated sample are number 1 to 50 in the sorted sample. An estimated 5%-quantile of the posterior is therefore between the 50th and 51st of the ordered sample. Also for this estimate, one should usually draw a larger sample.

Assignment 2

There is no solution available for this assignment at the moment.