Quiz 3 Solution

April 1, 2019

1 1a)

In [1]:
```
generateSamples <- function(R, lambda) {

    Xstart <- 5.01
    burnIn <- ceiling(R / 10)
    Xs <- rep(NA, R)

    Xs[1] <- Xstart

    for (r in 2:(R+burnIn)) {
        # Generate proposal
        U1 <- (runif(1) < 0.5)
        sign <- (-1)^U1
        Y <- Xs[r-1] + sign * rexp(1, lambda)

        # Calculate acceptance probability.
        alpha <- (Y > 5) * exp(0.5 * (Xs[r-1]^2 - Y^2))

        # Transition with this probability
        U <- runif(1)
        if (U < alpha) {
            Xs[r] <- Y
        } else {
            Xs[r] <- Xs[r-1]
        }
    }

    return(Xs[-(1:burnIn)])
}
```

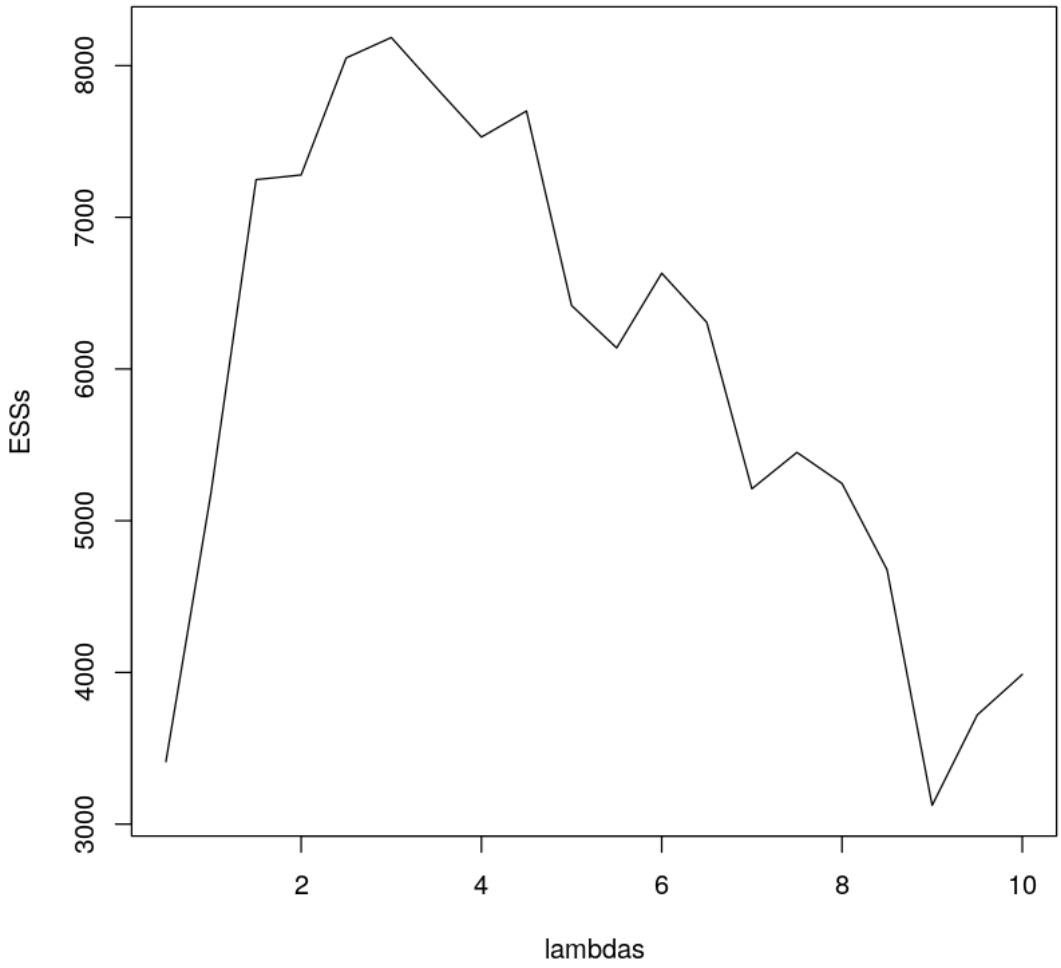
1.1 b)

In [2]:
```
# ADAPTED FROM: https://github.com/tpapp/MCMCDiagnostics.jl/blob/master/src/MCMCDiagnostics.jl
autocorrelation <- function(x, k, v) {
    R <- length(x)
    x1 <- (x[1:(R-k)])
    x2 <- (x[(1+k):R])
    V <- sum((x1 - x2)^2) / length(x1)
    return(1 - V / (2*v))
}

ess_estimate <- function(x)  {
```

```
    v <- var(x)
    R <- length(x)
    tau_inv <- 1 + 2 * autocorrelation(x, 1, v)
    K <- 2
    while (K < R - 2) {
        delta <- autocorrelation(x, K, v) + autocorrelation(x, K + 1, v)
        if (delta < 0)
            break

        tau_inv <- tau_inv + 2*delta
        K <- K + 2
    }

    return( R * min(1 / tau_inv, 1) )
}
```

In [3]:
```
R <- 10^5
lambdas <- seq(0.5,10,0.5)
ESSs <- rep(NA, length(lambdas))
fracRejected <- rep(NA, length(lambdas))

for (i in 1:length(lambdas)) {
    X <- generateSamples(R, lambdas[i])
    fracRejected[i] <- mean(diff(X)==0)
    ESSs[i] <- ess_estimate(X)
}

plot(lambdas, ESSs, type="l")

lambdaStarInd = which.max(ESSs)
print(lambdas[lambdaStarInd])
```

```
[1] 3
```



1.2 c)

```
In [4]: print(1-fracRejected[lambdaStarInd])
```

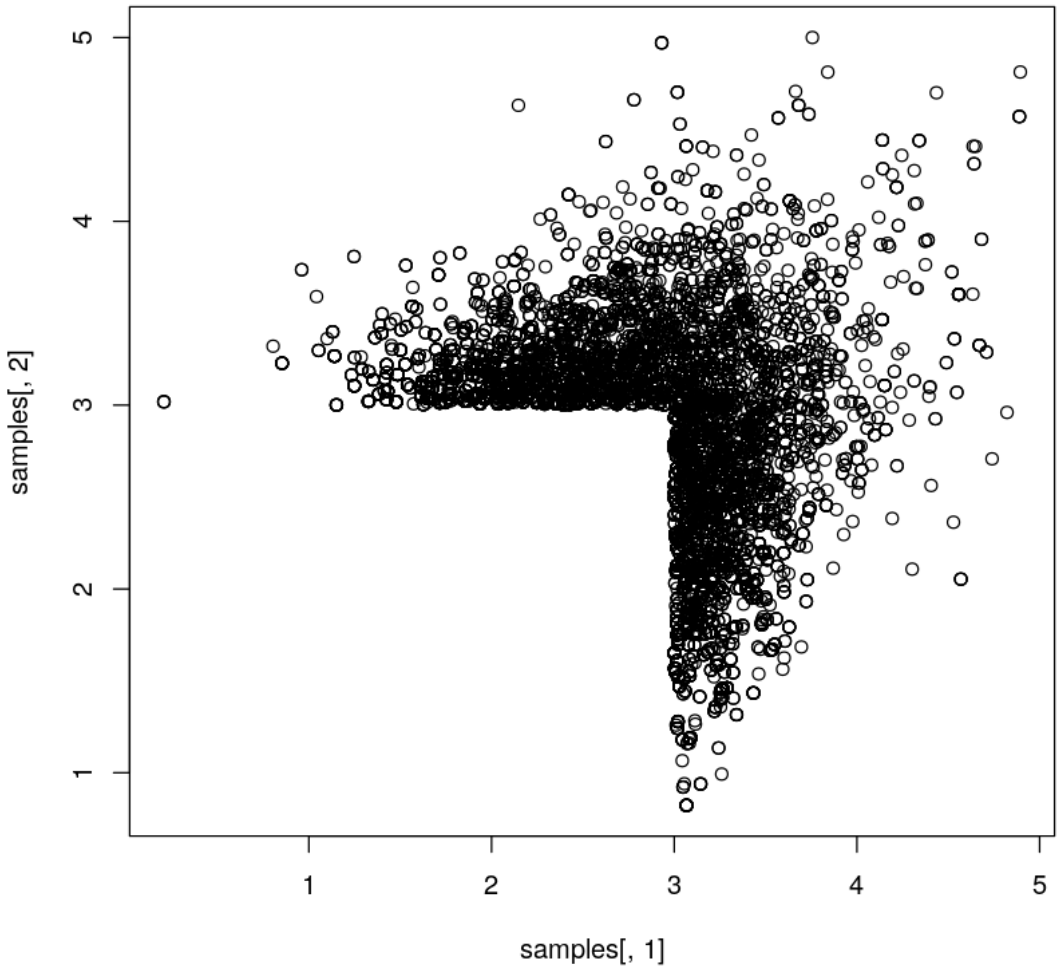
```
[1] 0.3616736
```

1.3 2a)

```
In [5]: library(mvtnorm)

        Zeros <- c(0, 0)
        Sigma_M <- matrix(c(1, 0.8, 0.8, 1), 2)

        generateSamples <- function(R, sigma2) {
```

```
    Xstart <- c(3.1, 3.1)
    burnIn <- 10^3

    Sigma_J <- matrix(c(sigma2, 0, 0, sigma2), 2)

    Xs <- matrix(rep(NA, 2*(R+burnIn)), ncol=2)
    Xs[1,] <- Xstart

    for (r in 2:(R+burnIn)) {
        # Generate proposal
        Y <- rmvnorm(1, Xs[r-1,], Sigma_J)

        # FOOLISH PAT! The second term cancels, so why calculate it.
        ## Calculate acceptance probability.
        #alphaNumer <- (max(Y)>3)*dmvnorm(Y, Zeros, Sigma_M) * dmvnorm(Xs[r-1,], Y, Sigma_J)
        #alphaDenom <- dmvnorm(Xs[r-1,], Zeros, Sigma_M) * dmvnorm(Y, Xs[r-1,], Sigma_J)
        #alpha <- min(alphaNumer/alphaDenom, 1)

        # Calculate acceptance probability.
        alphaNumer <- (max(Y)>3)*dmvnorm(Y, Zeros, Sigma_M)
        alphaDenom <- dmvnorm(Xs[r-1,], Zeros, Sigma_M)
        alpha <- min(alphaNumer/alphaDenom, 1)

        # Transition with this probability
        U <- runif(1)
        if (U < alpha) {
            Xs[r,] <- Y
        } else {
            Xs[r,] <- Xs[r-1,]
        }
    }

    return(Xs[-(1:burnIn),])
}
```

In [6]:
```
samples <- generateSamples(10^4, 0.5)
plot(samples[,1], samples[,2])
```



1.4 b)

```
In [7]: xs <- generateSamples(10^5, 0.5)
        print("Done MCMC sampling")
```

```
[1] "Done MCMC sampling"
```

```
In [8]: mu <- c(0, 0)
        muCE <- rep(mean(xs), 2)
        sigma <- matrix(c(1, 0.8, 0.8, 1), 2)

        R <- 10^6
        xs <- rmvnorm(R, muCE, sigma)
```

```
origPDF <- dmvnorm(xs, mu, sigma)
newPDF <- dmvnorm(xs, muCE, sigma)
LRs <- origPDF / newPDF

ms <- apply(xs, 1, max)
ests <- (ms > 3) * LRs

estMean <- mean(ests)
estVars <- var(ests)
```

In [9]:
```
estMean
```

0.00232072043617748

In [10]:
```
muCE
```

1. 2.88894653713672 2. 2.88894653713672