STAT 206 Homework 8

Xin Feng(Vanessa)

11/30/2019

Due Tuesday, December 3, 5:00 PM

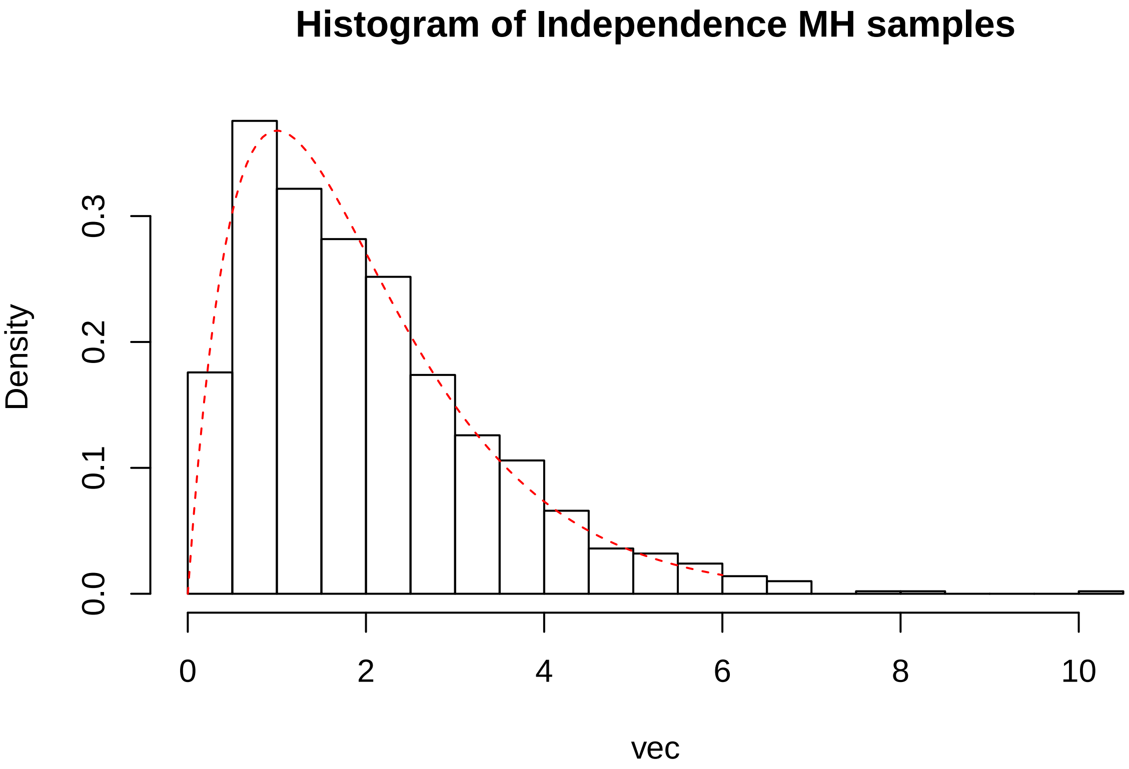
General instructions for homework: Homework must be submitted as pdf file, and be sure to include your name in the file. Give the commands to answer each question in its own code block, which will also produce plots that will be automatically embedded in the output file. Each answer must be supported by written statements as well as any code used. (Examining your various objects in the "Environment" section of RStudio is insufficient – you must use scripted commands.)

Part I - Metropolis-Hasting algorith

Suppose $f \sim \Gamma(2, 1)$.

1. Write an independence MH sampler with $g \sim \Gamma(2, \theta)$.

```
ind.chain <- function(x=1, n, a=2, b=1){
  # if theta=1, then this is an iid sampler.
  m<- length(x)
  x <- append(x, double(n))
  for(i in (m+1):length(x)){
    x.prime <- rgamma(1, a, b)
    Rt <- exp((x[i-1]-x.prime)*(1-b))
    x[i] <- ifelse(runif(1) < Rt, x[i] <- x.prime, x[i-1])
  }
  return(x)
}
vec <- ind.chain(n=1000)
hist(vec, probability = TRUE, 30, main="Histogram of Independence MH samples")
curve(dgamma(x, 2,1), from=0, to=6, add=T, lty=2, col=2)
```



2. What is $R(x_t, X^*)$ for this sampler?

$R(x_t, X^*) = \frac{f(x^*)g(x_t)}{f(x_t)g(x^*)}$ where $f \sim gamma(2,1), g \sim gamma(2,\theta)$.

$f(x,2,1) = \frac{1}{\Gamma(1)} xe^{-x}$ $g(x,2,\theta) = \frac{\theta^2}{\Gamma(2)} xe^{-\theta x}$

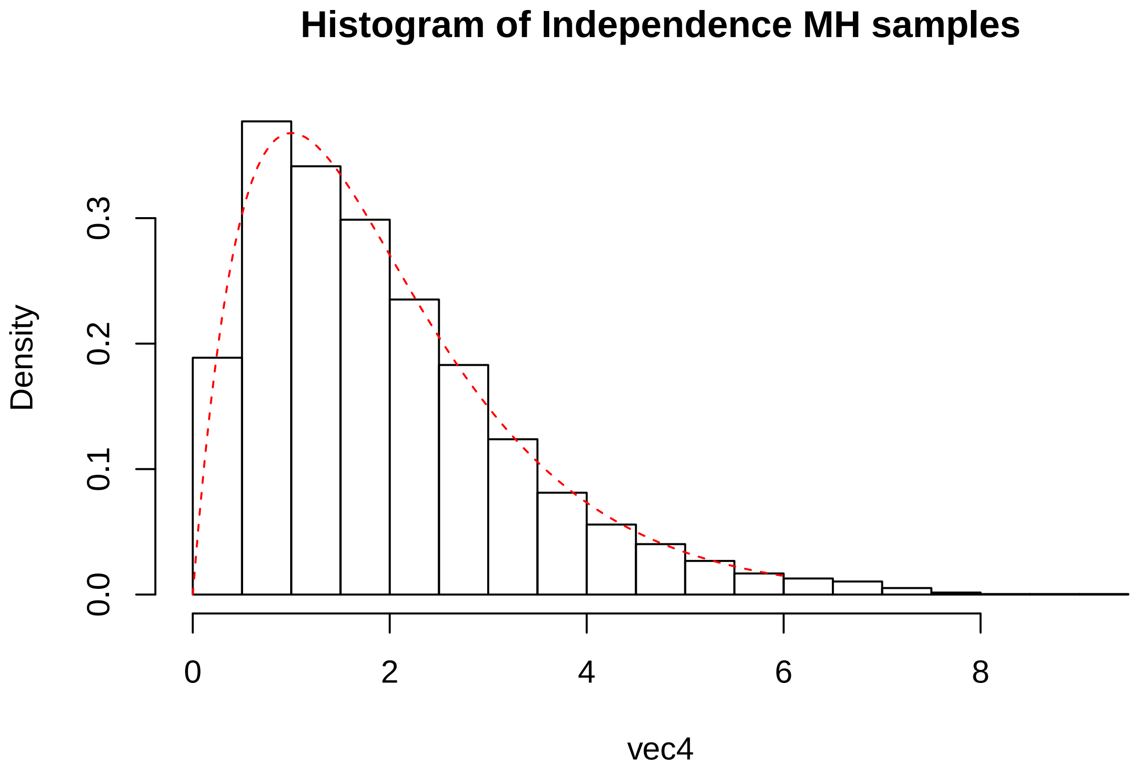
Thus,$R(x_t, X^*) = \frac{f(x^*,2,1)*g(x_t,2,\theta)}{f(x_t,2,1)*g(x^*,2,\theta)} = e^{(x_t - x^*)(1-\theta)}$

3. Generate 10000 draws from f with $\theta \in \{1/2, 1, 2\}$.

```
N <- 1e+4
vec1 <- ind.chain(n=N, b=1/2)
vec2 <- ind.chain(n=N, b=1)
vec3 <- ind.chain(n=N, b=2)
```

4. Write a random walk MH sampler with $h \sim N(0, \sigma^2)$.

```
set.seed(1)
rw.chain <- function(x=1, n, a=2, b=1) {
  m <- length(x)
  x <- append(x, double(n))
  for(i in (m+1):length(x)){
    x.prime <- x[i-1]+rnorm(1, sd=b)
    u <- exp(x[i-1]-x.prime)*(x.prime/x[i-1])
    x[i] <- ifelse(runif(1) < u && x.prime > 0, x.prime, x[i-1])
  }
  return(x)
}
vec4 <- rw.chain(n=10000, b=1)
hist(vec4, probability = TRUE, 30, main="Histogram of Independence MH samples")
curve(dgamma(x, 2,1), from=0, to=6, add=T, lty=2, col=2)
```



5. What is $R(x_t, X^*)$ for this sampler?

Because $x^* \sim Normal(0, \sigma^2)$ is symmetric zero mean random variables, $R(x_t, X^*) = \frac{f(x^*)}{f(x_t)}$ where $f \sim gamma(2, 1)$.

$f(x, 2, 1) = \frac{1}{\Gamma(1)} x e^{-x}$

Thus,$R(x_t, X^*) = \frac{f(x^*,2,1)}{f(x_t,2,1)} = (\frac{x^*}{x_t}) e^{(x_t - x^*)}$

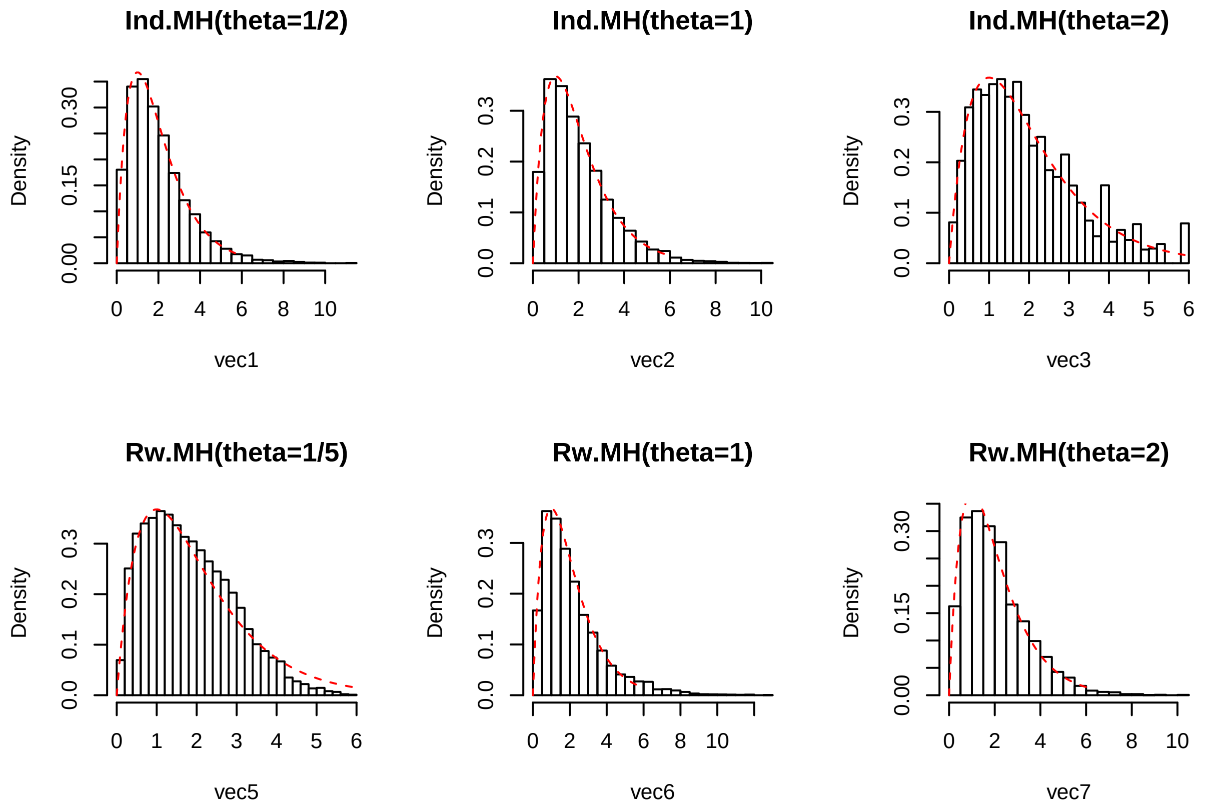
6. Generate 10000 draws from f with $\sigma \in \{.2, 1, 5\}$.

```
vec5 <- rw.chain(n=10000, b=1/5)
vec6 <- rw.chain(n=N, b=1)
vec7 <- rw.chain(n=N, b=5)
```

7. In general, do you prefer an independence chain or a random walk MH sampler? Why?

```
par(mfrow=c(2,3))
hist(vec1, probability = TRUE, 30, main="Ind.MH(theta=1/2)")
curve(dgamma(x, 2,1), from=0, to=6, col=2, add=T, lty=2)
hist(vec2, probability = TRUE, 30, main="Ind.MH(theta=1)")
```

```r
curve(dgamma(x, 2,1), from=0, to=6, col=2, add=T, lty=2)
hist(vec3, probability = TRUE, 30, main="Ind.MH(theta=2)")
curve(dgamma(x, 2,1), from=0, to=6, col=2, add=T, lty=2)
hist(vec5, probability = TRUE, 30, main="Rw.MH(theta=1/5)")
curve(dgamma(x, 2,1), from=0, to=6, col=2, add=T, lty=2)
hist(vec6, probability = TRUE, 30, main="Rw.MH(theta=1)")
curve(dgamma(x, 2,1), from=0, to=6, col=2, add=T, lty=2)
hist(vec7, probability = TRUE, 30, main="Rw.MH(theta=2)")
curve(dgamma(x, 2,1), from=0, to=6, col=2, add=T, lty=2)
```



```r
# According to the plots, I consider a random walk MH sampler is better.
```

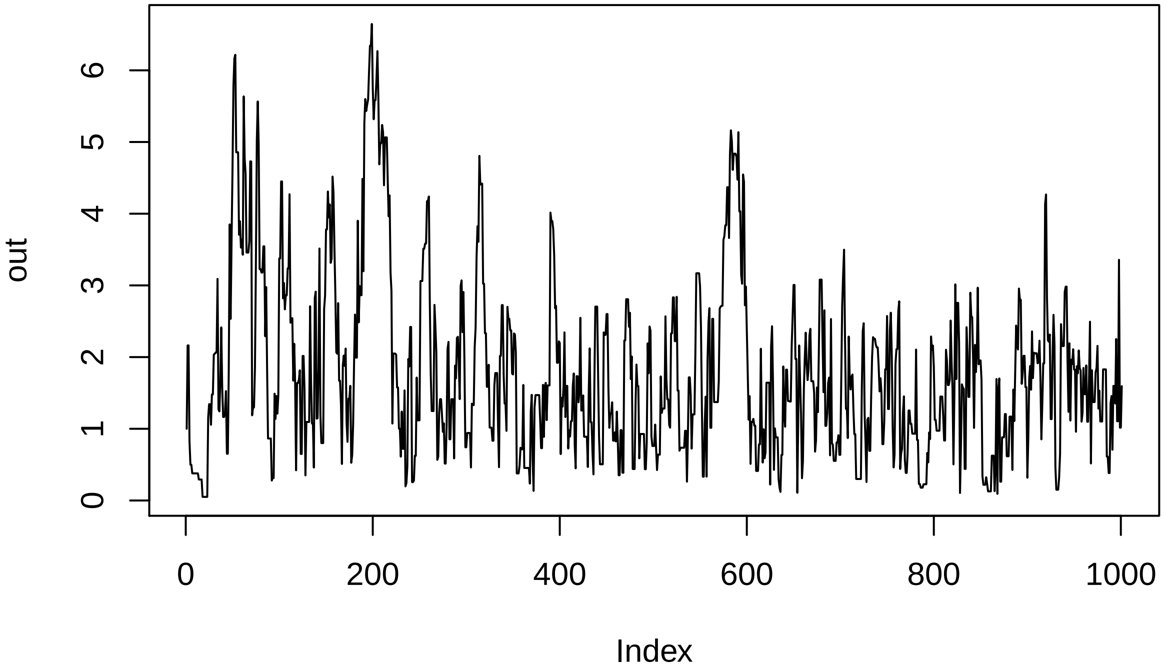
8. Implement the fixed-width stopping rule for you preferred chain.

```r
set.seed(20)
library(mcmcse)
```

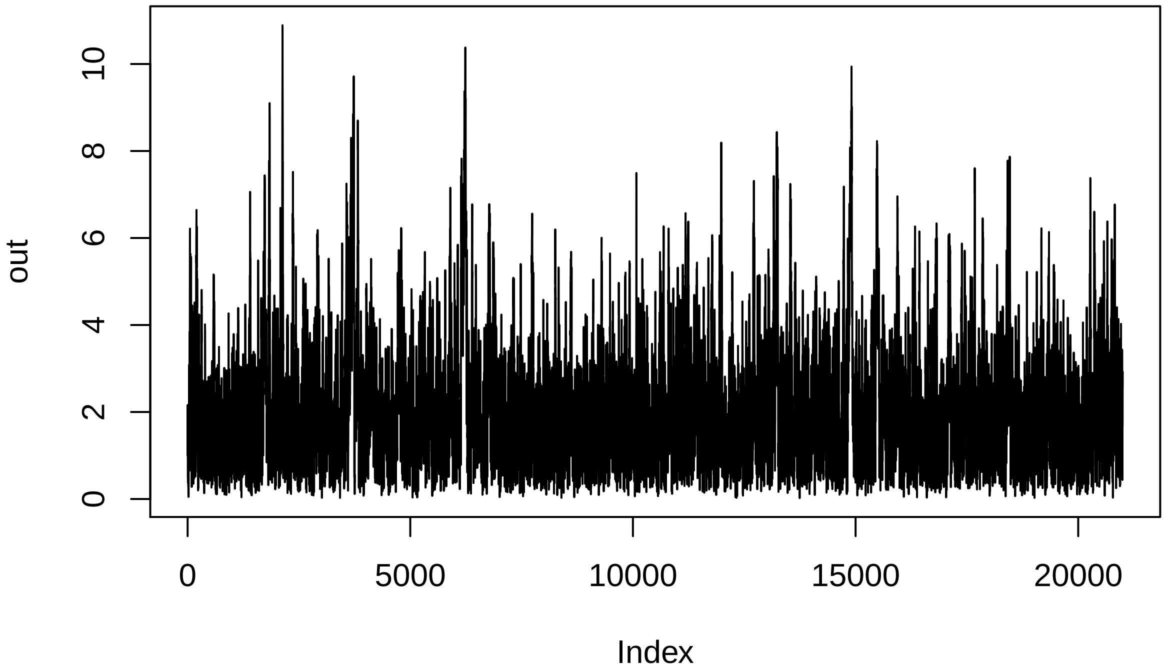
```
## mcmcse: Monte Carlo Standard Errors for MCMC
## Version 1.3-2 created on 2017-07-03.
## copyright (c) 2012, James M. Flegal, University of California, Riverside
##                     John Hughes, University of Colorado, Denver
##                     Dootika Vats, University of Warwick
##                     Ning Dai, University of Minnesota
##  For citation information, type citation("mcmcse").
##  Type help("mcmcse-package") to get started.
```

```
sigma <- 1
out <- 1
eps <- 0.1
start <- 1000
r <- 1000

out <- rw.chain(x=out, n=start, b=sigma)
MCSE <- mcse(out)$se
l <- length(out)
t <- qt(0.975, (floor(sqrt(l) - 1)))
muhat <- mean(out)
check <- MCSE * t
plot(out, type="l")
```



```
while(eps < check) {
  out <- rw.chain(x=out, n=r, b=sigma)
  MCSE <- append(MCSE, mcse(out)$se)
  l <- length(out)
  t <- qt(0.975, (floor(sqrt(l) - 1)))
  muhat <- append(muhat, mean(out))
  check <- MCSE[length(MCSE)] * t
}
plot(out, type="l")
```



Part II - Anguilla eel data

Consider the **Anguilla** eel data provided in the `dismo` R package. The data consists of 1,000 observations from a New Zealand survey of site-level presence or absence for the short-finned eel (Anguilla australis). We will use six out of twelve covariates. Five are continuous variables: `SegSumT`, `DSDist`, `USNative`, `DSMaxSlope` and `DSMaxSlope`; one is a categorical variable: `Method`, with five levels `Electric`, `Spo`, `Trap`, `Net` and `Mixture`.

Let x_i be the regression vector of covariates for the ith observation of length k and $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_9)$ be the vector regression coefficients. For the ith observation, suppose $Y_i = 1$ denotes presence and $Y_i = 0$ denotes absence of Anguilla australis. Then the Bayesian logistic regression model is given by

$$\begin{aligned} Y_i &\sim Bernoulli(p_i)\ , \\ p_i &\sim \frac{\exp(x_i^T \boldsymbol{\beta})}{1+\exp(x_i^T \boldsymbol{\beta})} \ \text{ and,} \\ \boldsymbol{\beta} &\sim N(\mathbf{0}, \sigma_\beta^2 \mathbf{I}_k)\ , \end{aligned}$$

where $\mathbf{I}_k$ is the $k \times k$ identity matrix. For the analysis, $\sigma_\beta^2 = 100$ was chosen to represent a diffuse prior distribution on $\boldsymbol{\beta}$.

9. Implement an MCMC sampler for the target distribution using the `MCMClogit` function in the `MCMCpack` package.

```
library(dismo)
```

```
## Loading required package: raster
```

```
## Loading required package: sp
```

```
# View(Anguilla_train)
data("Anguilla_train")
At_data <- subset(Anguilla_train, select=c("Angaus", "SegSumT", "DSDist",
              "USNative", "DSMaxSlope", "USSlope", "Method"))

library(MCMCpack)
```

```
## Loading required package: coda

## Loading required package: MASS

##
## Attaching package: 'MASS'

## The following objects are masked from 'package:raster':
##
##     area, select

## ##
## ## Markov Chain Monte Carlo Package (MCMCpack)

## ## Copyright (C) 2003-2019 Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park

## ##
## ## Support provided by the U.S. National Science Foundation

## ## (Grants SES-0350646 and SES-0350613)
## ##
```

```
post_sample <- MCMClogit(Angaus~SegSumT+DSDist+USNative+DSMaxSlope+USSlope
                         +as.factor(Method), b0=0, B0=0.01, data=At_data)
```

```
## Warning in model.matrix.default(mt, mf, contrasts): non-list contrasts
## argument ignored
```

```
# as.factor(At_data$Method)
# levels(At_data$Method)
```

10. Comment on the mixing properties for your sampler. Include at least one plot in support of your comments.

```
# plot(post_sample)

# According to the following plot,
# the left plots show the randomicity,
# and the right plots similarly follow normal distribution,
# which show that the samples of each variance mix well.
```

11. Run your sampler for 100,000 iterations. Estimate the posterior mean along with an 80% Bayesian credible interval for each regression coefficient in the model. Be sure to include uncertainty estimates.

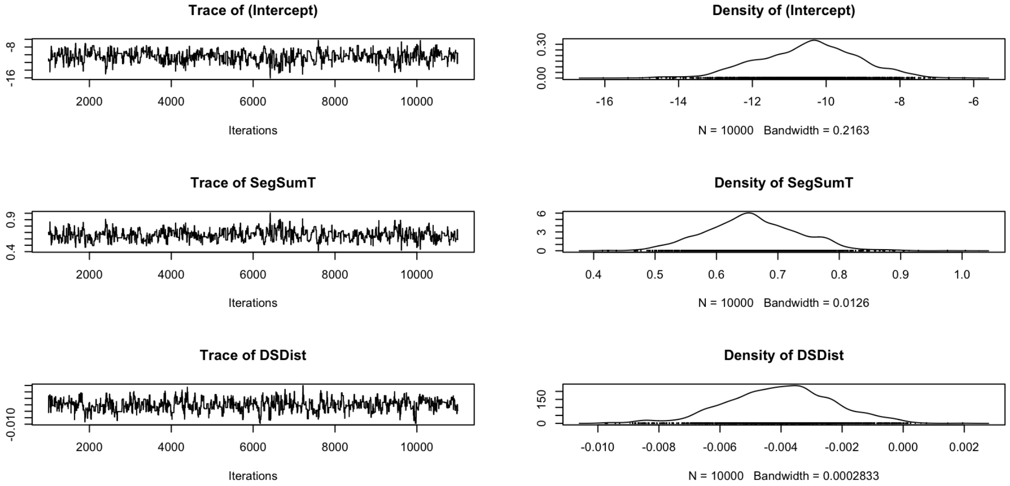


Figure 1: Question-10

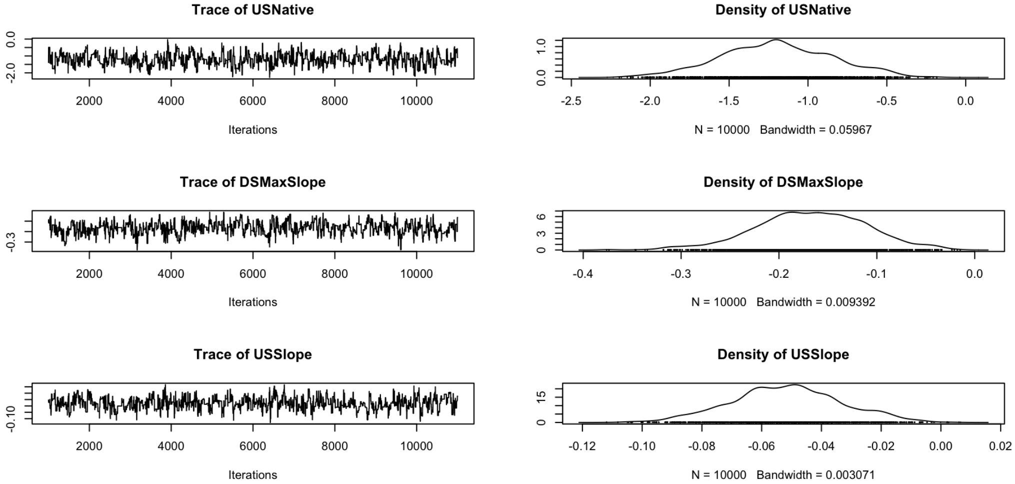


Figure 2: Question-10

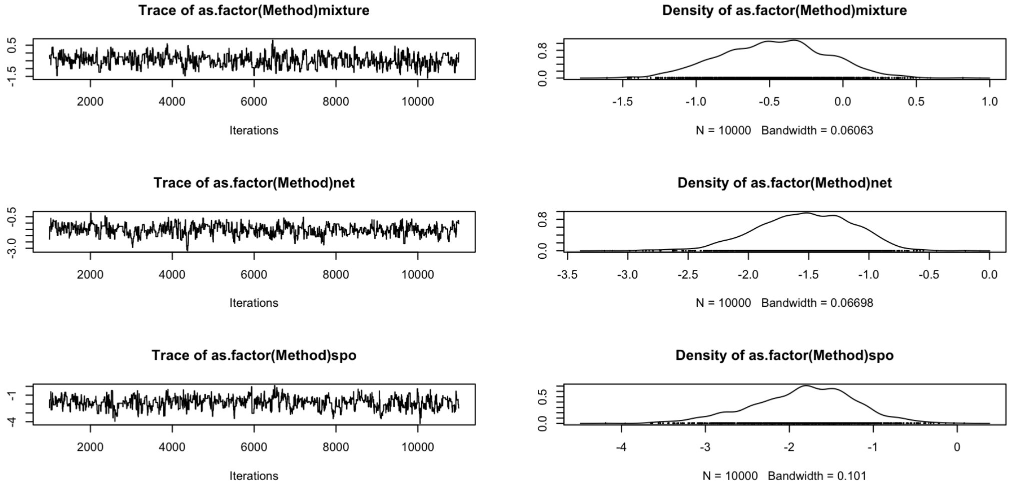


Figure 3: Question-10

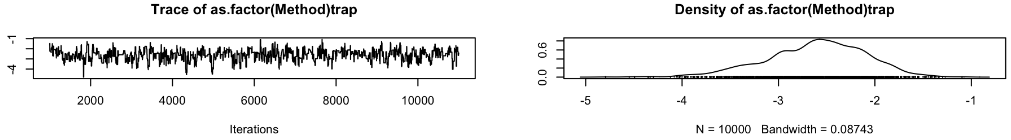


Figure 4: Question-10

```
library(mcmcse)
post_sample1 <- MCMClogit(Angaus~SegSumT+DSDist+USNative+DSMaxSlope+USSlope
                  +as.factor(Method), b0=0, B0=.01, data=At_data, mcmc=1e+5)
```

```
## Warning in model.matrix.default(mt, mf, contrasts): non-list contrasts
## argument ignored
```

```
for(i in 1:10){
  min.est<-mcse.q(post_sample[,i], 0.1)$est
  min.se<-mcse.q(post_sample[,i], 0.1)$se
  max.est<-mcse.q(post_sample[,i], 0.9)$est
  max.se<-mcse.q(post_sample[,i], 0.9)$se
  cat("(", min.est, "+", min.se,",",max.est,"+",max.se,")\n")
}
```

```
## ( -12.25238 + 0.1007808 , -8.840307 + 0.09610242 )
## ( 0.5596832 + 0.005264131 , 0.7636917 + 0.006084284 )
## ( -0.006278542 + 0.0001456728 , -0.001829195 + 0.0001347932 )
## ( -1.641218 + 0.03029746 , -0.7404843 + 0.03356553 )
## ( -0.2377378 + 0.004737119 , -0.09614401 + 0.004222614 )
## ( -0.07526085 + 0.001445666 , -0.02731731 + 0.001719858 )
## ( -0.9379482 + 0.02686542 , 0.002783441 + 0.02375967 )
## ( -2.040471 + 0.0382914 , -1.032479 + 0.02487854 )
## ( -2.691049 + 0.07158907 , -1.105186 + 0.03683184 )
## ( -3.32501 + 0.04932758 , -1.980706 + 0.03191623 )
```

12. Compare your Bayesian estimates to those obtained via maximum likelihood estimation.

```
# Bayesian estimates
Bs_est <- c()
for(i in 1:10){
  Bs_est[i] <- mcse(post_sample[,i])$est
}

# maximum likelihood estimates
# At_data$electric <- ifelse(At_data$Method=="electric", 1, 0)
At_data$spo <- ifelse(At_data$Method=="spo", 1, 0)
At_data$trap <- ifelse(At_data$Method=="trap", 1, 0)
At_data$net <- ifelse(At_data$Method=="net", 1, 0)
At_data$mixture <- ifelse(At_data$Method=="mixture", 1, 0)

At_data <- At_data[,-7]
l <- nrow(At_data)
dataX <- as.matrix(cbind(c(rep(1, times=l)), At_data[,-1]))
```

```
library(Rlab)
```

```
## Rlab 2.15.1 attached.
```

```
##
## Attaching package: 'Rlab'
## The following object is masked from 'package:MASS':
##
##     michelson
```

```
## The following objects are masked from 'package:stats':
##
##     dexp, dgamma, dweibull, pexp, pgamma, pweibull, qexp, qgamma,
##     qweibull, rexp, rgamma, rweibull
```

```
## The following object is masked from 'package:datasets':
##
##     precip
```

```
LogSum <- function(p=c(rep(0,times=10))){
  sum <- 0
  p <- as.matrix(p)
  bnl <- exp(dataX %*% p)
  p <- bnl/(1+bnl)
  for(i in 1:l){
    sum <- sum+dbern(At_data[,1][i], p[i], log=TRUE)
  }
  return(-sum)
}
mle_est <- nlm(LogSum, p=Bs_est)
```

```
## Warning in nlm(LogSum, p = Bs_est): NA/Inf replaced by maximum positive
## value

## Warning in nlm(LogSum, p = Bs_est): NA/Inf replaced by maximum positive
## value

## Warning in nlm(LogSum, p = Bs_est): NA/Inf replaced by maximum positive
## value

## Warning in nlm(LogSum, p = Bs_est): NA/Inf replaced by maximum positive
## value
```

```
Bs_est
```

```
##  [1] -10.477200616   0.658843844  -0.004040462  -1.199752804  -0.167122876
##  [6]  -0.051757410  -0.467299149  -1.530359206  -1.831387562  -2.613424774
```

```
mle_est$estimate
```

```
##  [1] -10.533155054   0.660163377  -0.003873045  -1.149261204  -0.163425460
##  [6]  -0.051482461  -1.728695013  -2.502861359  -1.489220206  -0.466428380
```

```
# The estimates of the two methods is closed.
```

Part III - Permutation tests

The Cram'er von Mises statistic estimates the integrated square distance between distributions. It can be computed using the following formula

$$W = \frac{mn}{(m+n)^2}\left[\sum_{i=1}^{n}(F_n(x_i) - G_m(x_i))^2 + \sum_{j=1}^{m}(F_n(y_j) - G_m(y_j))^2\right]$$

where F_n and G_m are the corresponding empirical cdfs.

13. Implement the two sample Cram'er von Mises test for equal distributions as a permutation test. Apply it to the `chickwts` data comparing the `soybean` and `linseed` diets.

```
library(RVAideMemoire)
```

```
## *** Package RVAideMemoire v 0.9-73 ***
```

```
##
## Attaching package: 'RVAideMemoire'
```

```
## The following object is masked from 'package:raster':
##
##     cv
```

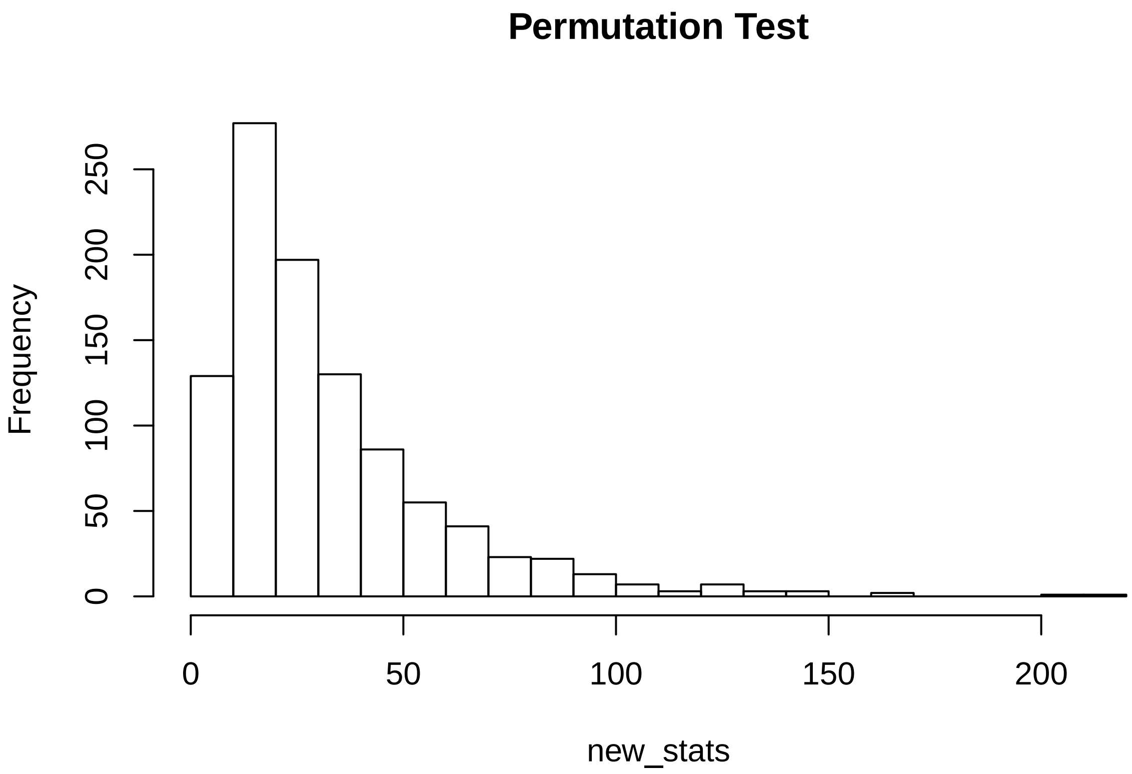
```
data("chickwts")
# View(chickwts)
S <- chickwts$weight[which(chickwts$feed=="soybean")]
L <- chickwts$weight[which(chickwts$feed=="linseed")]
n1 <- length(S)
n2 <- length(L)
Z <- c(S, L)
N <- length(Z)
B <- 1000
new_stats <- numeric(B)
CvM.test(S,L)
```

```
##
##  Two-sample Cramér-von Mises test
##
## data:  S and L
## T = 29.851, p-value = 0.3526
## alternative hypothesis: two.sided
```

```
obs_stat <- CvM.test(S,L)$statistic
for(i in 1:B){
  idx <- sample(1:N, size=n1, replace=F)
  newS <- Z[idx]
  newL <- Z[-idx]
  new_stats[i] <- CvM.test(newS, newL)$statistic
}
pvalue <- mean(c(obs_stat, new_stats)>=obs_stat)
cat("pvalue = ", pvalue, "\n")
```

```
## pvalue =  0.4015984
```

```
hist(new_stats, 30, main="Permutation Test")
```



14. How would you implement the bivariate Spearman rank correlation test for independence as a permutation test? The Spearman rank correlation test statistic can be obtained from the function `cor` with `method="spearman"`. Compare the achieved significance level of the permutation test with the p-value reported by `cor.test` on the same samples.

```
library(dismo)
# View(Anguilla_train)
data("Anguilla_train")
c_cor <- cor(Anguilla_train$SegSumT, Anguilla_train$SegTSeas, method="spearman")
N <- 1000
new_stats <- numeric(N)
for(i in 1:N){
  B <- Anguilla_train$SegSumT
  H <- sample(Anguilla_train$SegTSeas, length(Anguilla_train$SegTSeas), replace=F)
  new_stats[i] <- cor(B,H, method="spearman")
}
n <- length(new_stats[new_stats>=c_cor])
prob <- n/N
cat("\n p-value of Permutation test: ", prob, "\n")
```

```
##
##  p-value of Permutation test:  0
```

```
c_test <- cor.test(Anguilla_train$SegSumT, Anguilla_train$SegTSeas, method="spearman")
```

```
## Warning in cor.test.default(Anguilla_train$SegSumT,
## Anguilla_train$SegTSeas, : Cannot compute exact p-value with ties
```

```
c_test
```

```
##
##  Spearman's rank correlation rho
##
## data:  Anguilla_train$SegSumT and Anguilla_train$SegTSeas
## S = 123140619, p-value < 2.2e-16
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## 0.2611555
```

```
cat("p-value of Correlation test: ", c_test$p.value, "\n")
```

```
## p-value of Correlation test:  4.689618e-17
```

```
# p-value of Permutation test is a bit smaller than p-value of Correlation test.
```