# STAT 206 Homework 8_Lihua Xu

**Due Thursday, December 7, 5:00 PM**

***General instructions for homework***: Homework must be completed as an R Markdown file. 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 algorithm

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

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

```
ind.chain <- function(x, n, theta) {
  m <- length(x)
  x <- append(x, double(n))
  for(i in (m+1):length(x)){
    x.prime <- rgamma(1,shape=2, rate=theta)
    u <- exp(-theta*x[(i-1)]-x.prime+x[(i-1)]+theta*x.prime)
    if(runif(1) < u)
      x[i] <- x.prime
    else
      x[i] <- x[(i-1)]
  }
  return(x)
}
```

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

Solution: As $f \sim \Gamma(2,1)$:

$$f = \begin{cases} \frac{1}{\Gamma(2)} x e^{-x} & \text{x} > 0 \\ 0 \text{ otherwise} \end{cases}$$

and for $g \sim \Gamma(2,\theta)$:

$$g = \begin{cases} \frac{\theta^2}{\Gamma(2)} x e^{-\theta x} & \text{x} > 0 \\ 0 \text{ otherwise} \end{cases}$$

So we cat get $R(x_t, x^*)$ by the following equation:

$$R(x_t, x^*) = \frac{f(x^*)g(x_t|x^*)}{f(x_t)g(x^*|x_t)}$$

$$R(x_t, x^*) = \frac{e^{-x^*} * e^{-\theta x_t}}{e^{-x_t} * e^{-\theta x^*}}$$

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

```
trial0 <- ind.chain(1, 10000, 1)
trial1 <- ind.chain(1, 10000, 2)
trial2 <- ind.chain(1, 10000, 1/2)
#I will not show the result as it is too long.
```

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

```
rw.chain <- function(x, n, sigma) {
  m <- length(x)
  x <- append(x, double(n))
  for(i in (m+1):length(x)){
    x.prime <- x[(i-1)] + rnorm(1,mean=0,sd = sigma)
    u <- (x.prime*exp(-x.prime))/(x[(i-1)]*exp(-x[(i-1)]))
    if(runif(1) < u && x.prime > 0)
      {x[i] <- x.prime}
    else
      {x[i] <- x[(i-1)]}
  }
  return(x)
}
```

5. What is $R(x_t, X^*)$ for this sampler? Solution: The function for $h \sim N(0, \sigma^2)$:

$$h = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x_t^2}{2\sigma^2}}$$

$$R(x_t, x^*) = \frac{x^* e^{-x^*} * e^{\frac{(x_t)^2}{2\sigma^2}}}{x_t e^{-x_t} * e^{-\frac{(x^*)^2}{2\sigma^2}}}$$

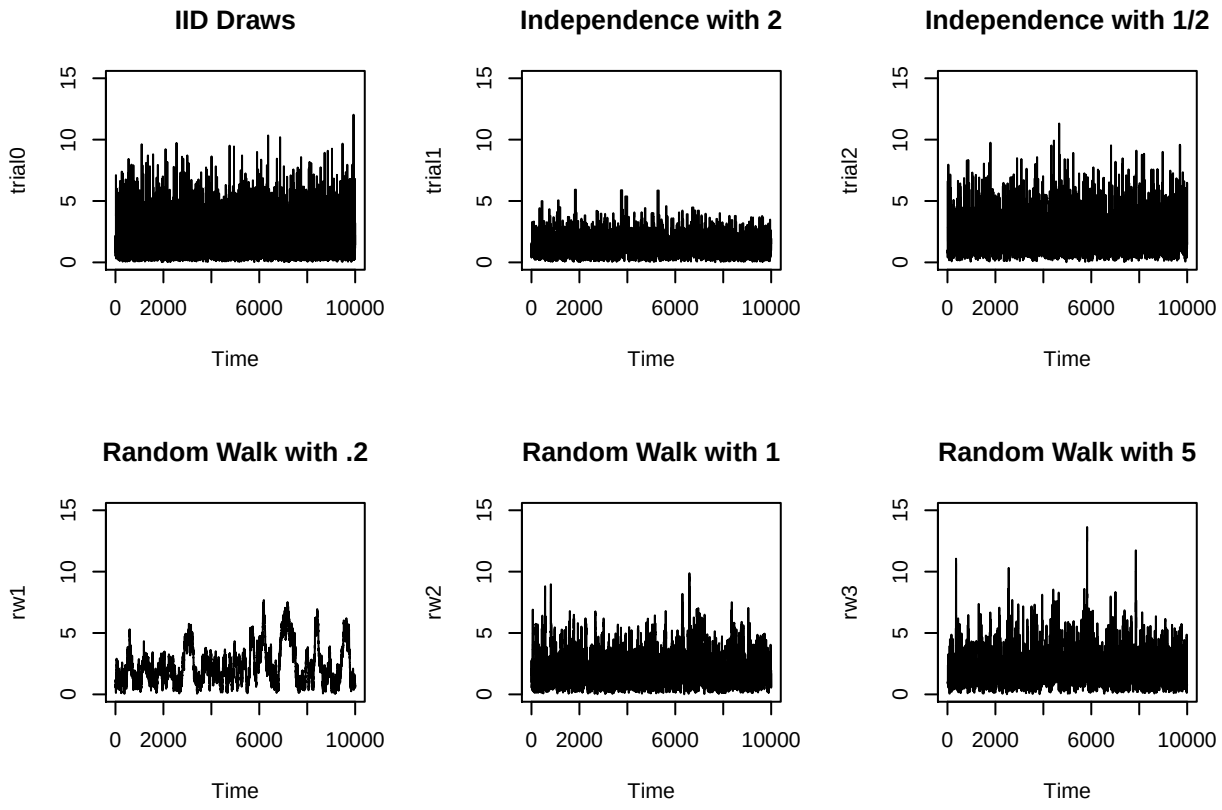
$$R(x_t, x^*) = \frac{x^* e^{-x^*}}{x_t e^{-x_t}} I(x^* > 0)$$

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

```
rw1 <- rw.chain(1, 10000, .2)
rw2 <- rw.chain(1, 10000, 1)
rw3 <- rw.chain(1, 10000, 5)
#I will not show the result as it is too long.
```

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

```
par(mfrow=c(2,3))
plot.ts(trial0, ylim=c(0,15), main="IID Draws")
plot.ts(trial1, ylim=c(0,15), main="Independence with 2")
plot.ts(trial2, ylim=c(0,15), main="Independence with 1/2")
plot.ts(rw1, ylim=c(0,15), main="Random Walk with .2")
plot.ts(rw2, ylim=c(0,15), main="Random Walk with 1")
plot.ts(rw3, ylim=c(0,15), main="Random Walk with 5")
```




```
par(mfrow=c(1,1))
# Except the dist one, which is the IID draws,
# the third figure named as "Independence with 1/2" is the best.
# For me, I will more prefer an independence chain with 1/2.
# It will not stuck to the large values and is more close to the plot of IID Draws.
```

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

```
ind.chain.implement <- function(x, n, theta, epsilon) {
  m <- length(x)
  x <- append(x, double(n))
  for(i in (m+1):length(x)){
    x.prime <- rgamma(1,shape=2, rate=theta)
    u <- exp(-theta*x[(i-1)]-x.prime+x[(i-1)]+theta*x.prime)
    if(runif(1) < u)
      x[i] <- x.prime
    else
      x[i] <- x[(i-1)]
    sigma_bar <- sd(x)
    if((1.96*sqrt(sigma_bar)/sqrt(i)) < epsilon)
      break
  }
  data <- list(x,i)
  return(data)
}
```

```
#when we assume the epsilon equal to 0.02
#("Relative fixed-width stopping rules for Markov chain Monte Carlo simulations--
#--James M. Flegal and Lei Gong"), then
trial3 <- ind.chain.implement(1, 100000, 1, 0.02)
#The number of times needed:
trial3[[2]]
```

```
## [1] 5495
```

```
trial4 <- ind.chain.implement(1, 100000, 2, 0.02)
#The number of times needed:
trial4[[2]]
```

```
## [1] 4275
```

```
trial5 <- ind.chain.implement(1, 100000, 1/2, 0.02)
#The number of times needed:
trial5[[2]]
```

```
## [1] 5183
```

# 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 `DSSlope`; 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 $i$th observation of length $k$ and $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_9)$ be the vector regression coefficients. For the $i$th 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

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

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.

```
#install.packages("dismo")
library("dismo")
```

```
## Warning: package 'dismo' was built under R version 3.4.3
```

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

```
## Warning: package 'raster' was built under R version 3.4.3
```

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

```
## Warning: package 'sp' was built under R version 3.4.3
```

```
#install.packages("MCMCpack")
library("MCMCpack")
```

```
## Warning: package 'MCMCpack' was built under R version 3.4.3
## Loading required package: coda
## Warning: package 'coda' was built under R version 3.4.3
## 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) 2003-2017 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)
## ##
```

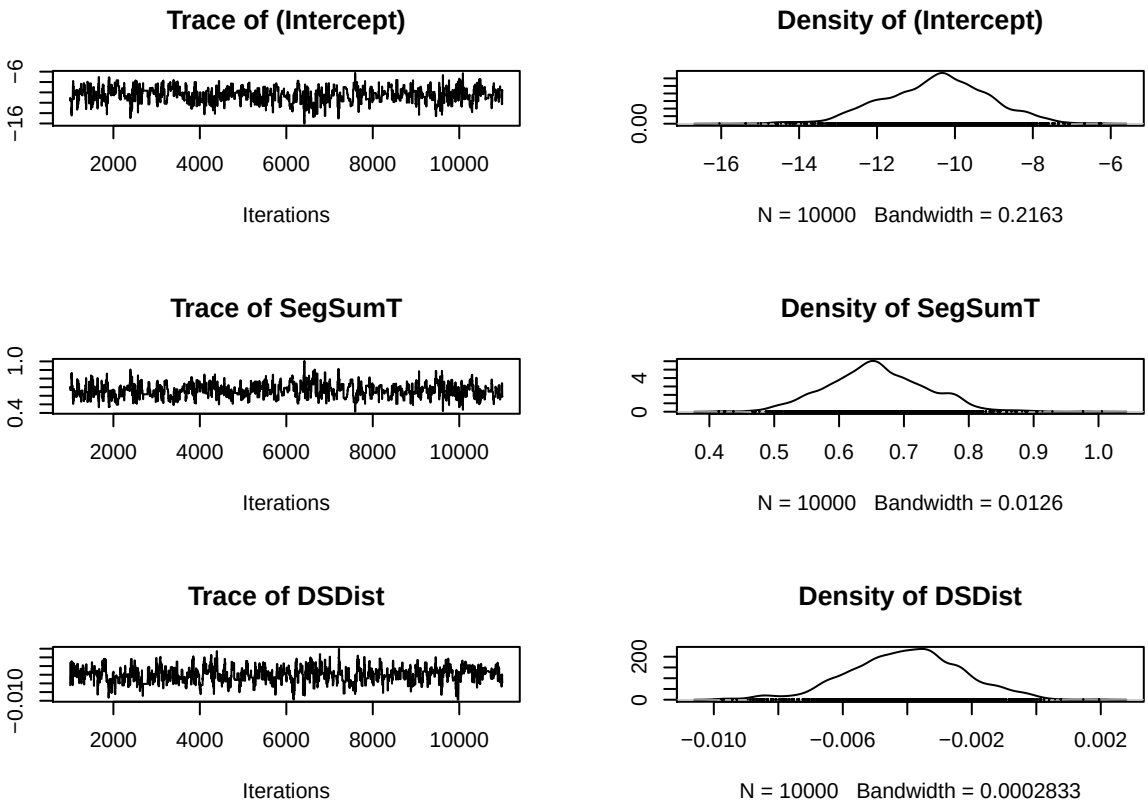
```
#install.packages("mcmcse")
library("mcmcse")
```

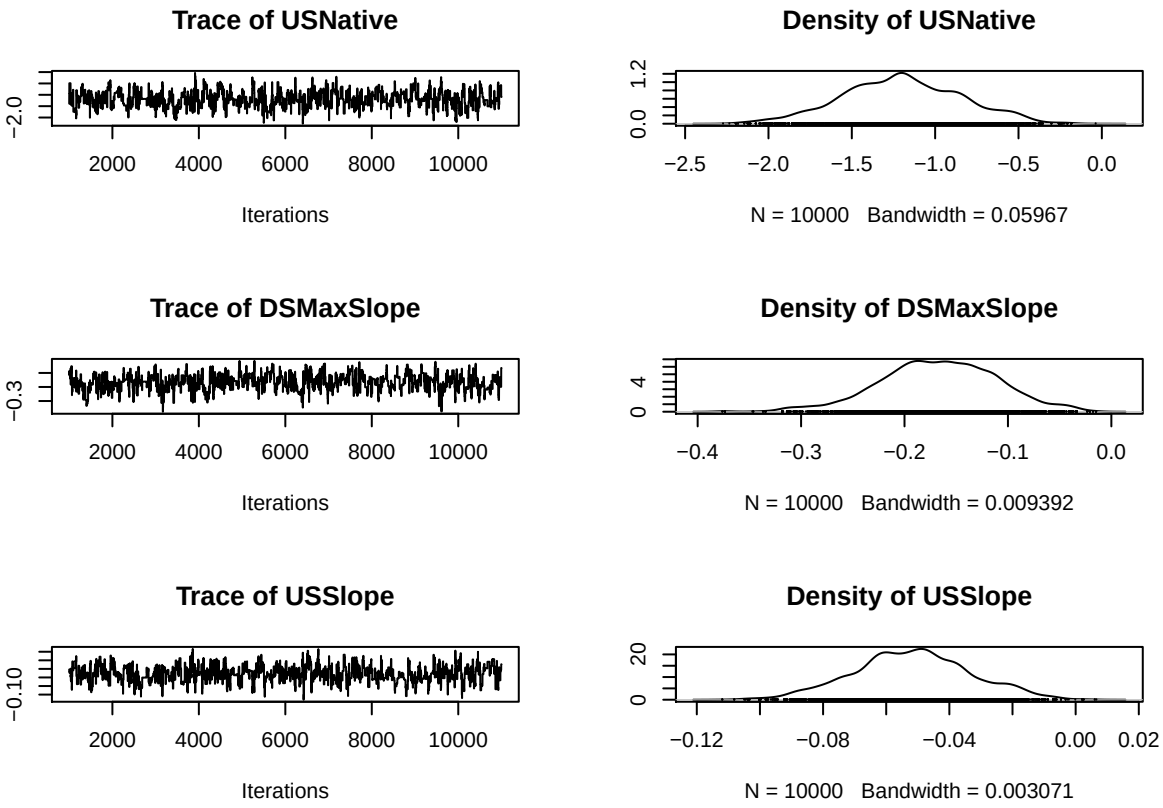
```
## Warning: package 'mcmcse' was built under R version 3.4.3
## mcmcse: Monte Carlo Standard Errors for MCMC
## Version 1.3-2 created on 2017-07-03.
## copyright (c) 2012, James M. Flegal, University of California, Riverside
##                     John Hughes, University of Colorado, Denver
##                     Dootika Vats, University of Warwick
##                     Ning Dai, University of Minnesota
##  For citation information, type citation("mcmcse").
##  Type help("mcmcse-package") to get started.
```

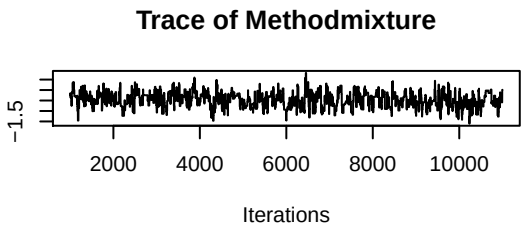
```
data(Anguilla_train)
posterior <- MCMClogit(Angaus~SegSumT+DSDist+USNative+DSMaxSlope+USSlope+Method,
                       data=Anguilla_train,b0=0,B0=0.01)
```

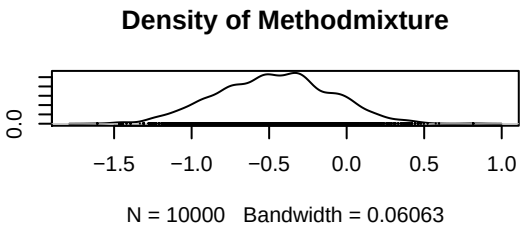
10. Comment on the mixing properties for your sampler. Include at least one plot in support of your comments.

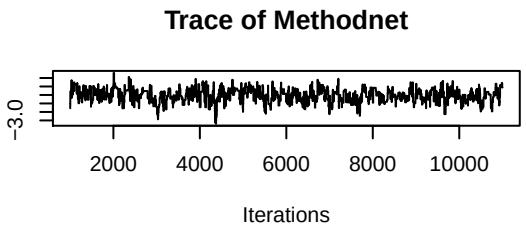
```
plot(posterior)
```

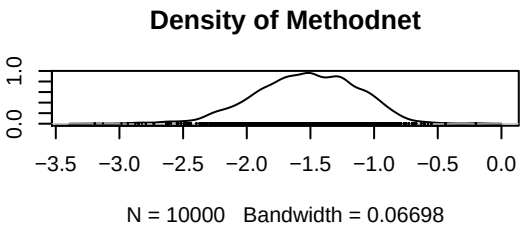

Trace of (Intercept)
−16
−6
2000
4000
6000
8000
10000
Iterations
Density of (Intercept)
0.00
−16
−14
−12
−10
−8
−6
N = 10000 Bandwidth = 0.2163
Trace of SegSumT
0.4
1.0
2000
4000
6000
8000
10000
Iterations
Density of SegSumT
0
4
0.4
0.5
0.6
0.7
0.8
0.9
1.0
N = 10000 Bandwidth = 0.0126
Trace of DSDist
−0.010
2000
4000
6000
8000
10000
Iterations
Density of DSDist
0
200
−0.010
−0.006
−0.002
0.002
N = 10000 Bandwidth = 0.0002833

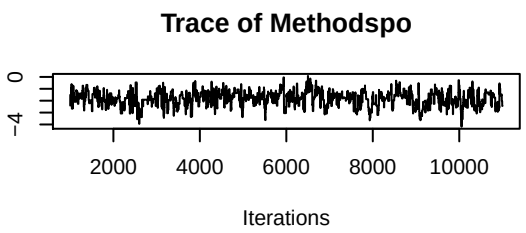

Trace of USNative
−2.0
2000
4000
6000
8000
10000
Iterations
Density of USNative
0.0
1.2
−2.5
−2.0
−1.5
−1.0
−0.5
0.0
N = 10000 Bandwidth = 0.05967
Trace of DSMaxSlope
−0.3
2000
4000
6000
8000
10000
Iterations
Density of DSMaxSlope
0
4
−0.4
−0.3
−0.2
−0.1
0.0
N = 10000 Bandwidth = 0.009392
Trace of USSlope
−0.10
2000
4000
6000
8000
10000
Iterations
Density of USSlope
0
20
−0.12
−0.08
−0.04
0.00
0.02
N = 10000 Bandwidth = 0.003071

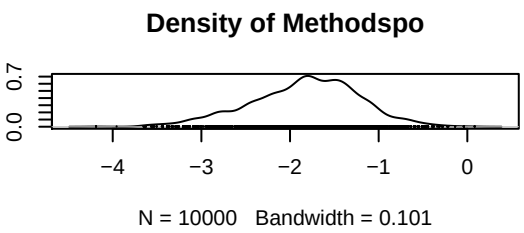

Trace of Methodmixture
−1.5
2000
4000
6000
8000
10000
Iterations

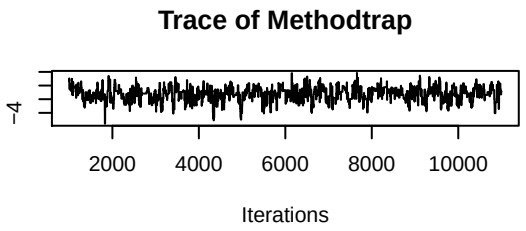

Density of Methodmixture
0.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
N = 10000 Bandwidth = 0.06063

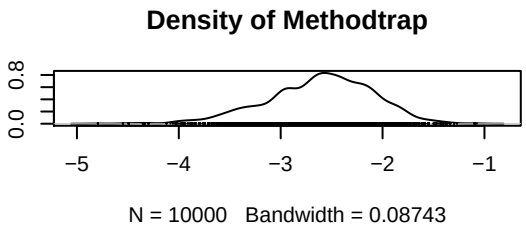

Trace of Methodnet
−3.0
2000
4000
6000
8000
10000
Iterations


Density of Methodnet
0.0
1.0
−3.5
−3.0
−2.5
−2.0
−1.5
−1.0
−0.5
0.0
N = 10000 Bandwidth = 0.06698


Trace of Methodspo
−4
0
2000
4000
6000
8000
10000
Iterations


Density of Methodspo
0.0
0.7
−4
−3
−2
−1
0
N = 10000 Bandwidth = 0.101




```
summary(posterior)
```

```
##
## Iterations = 1001:11000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                     Mean       SD  Naive SE Time-series SE
## (Intercept)   -10.47720 1.344483 1.344e-02      0.0797443
## SegSumT         0.65884 0.075855 7.585e-04      0.0044851
## DSDist         -0.00404 0.001746 1.746e-05      0.0001091
## USNative       -1.19975 0.355155 3.552e-03      0.0217863
## DSMaxSlope     -0.16712 0.056552 5.655e-04      0.0034618
## USSlope        -0.05176 0.018452 1.845e-04      0.0010866
## Methodmixture  -0.46730 0.360869 3.609e-03      0.0215716
## Methodnet      -1.53036 0.398669 3.987e-03      0.0263901
## Methodspo      -1.83139 0.613023 6.130e-03      0.0394532
## Methodtrap     -2.61342 0.520430 5.204e-03      0.0317179
##
## 2. Quantiles for each variable:
##
##                    2.5%        25%        50%       75%     97.5%
```

```
## (Intercept)   -13.08495 -11.343939 -10.391855 -9.618494 -7.991135
## SegSumT         0.51649   0.608240   0.656600  0.708767  0.805260
## DSDist         -0.00776  -0.005172  -0.003991 -0.002912 -0.000607
## USNative       -1.90731  -1.444627  -1.207530 -0.950169 -0.498107
## DSMaxSlope     -0.28874  -0.202801  -0.165107 -0.127886 -0.055226
## USSlope        -0.08743  -0.063798  -0.051776 -0.039306 -0.015960
## Methodmixture  -1.17193  -0.724461  -0.463801 -0.226057  0.209949
## Methodnet      -2.32850  -1.792387  -1.511816 -1.242883 -0.846636
## Methodspo      -3.13204  -2.212332  -1.795904 -1.406867 -0.688692
## Methodtrap     -3.71155  -2.953262  -2.583058 -2.238212 -1.731241
```

```
#When parameters are highly correlated with each other.
#Poor mixing means that the Markov chain slowly traverses the parameter space
#(Trace plots above) and the chain has high dependence.
#The trace tells if the chain has not converged to its stationary distribution
#and also tells if it needs a longer period.
#The trace plot can also tell whether the chain is mixing well.
#If the distribution for the points is not changing along the chain,
#this chain might reach to stationarity situation.
```

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.

```
posterior_100000 <- MCMClogit(Angaus~SegSumT+DSDist+USNative+DSMaxSlope+USSlope+Method,
                              data=Anguilla_train,
                              b0=0,B0=0.01,mcmc=100000)
mcse.q.mat(posterior_100000,0.1)
```

```
##                         est           se
## (Intercept)   -12.242628259 4.192379e-02
## SegSumT         0.555750839 1.825021e-03
## DSDist         -0.006261017 5.244791e-05
## USNative       -1.633926558 1.034104e-02
## DSMaxSlope     -0.244644299 1.596839e-03
## USSlope        -0.075867298 4.826678e-04
## Methodmixture  -0.905797524 8.907305e-03
## Methodnet      -2.022658541 1.037065e-02
## Methodspo      -2.633742039 1.949114e-02
## Methodtrap     -3.293057460 1.690163e-02
```

```
mcse.q.mat(posterior_100000,0.9)
```

```
##                        est           se
## (Intercept)   -8.666619444 3.447760e-02
## SegSumT        0.759131203 2.372978e-03
## DSDist        -0.001938343 4.105681e-05
## USNative      -0.712806084 1.058273e-02
## DSMaxSlope    -0.101159386 1.350279e-03
## USSlope       -0.028016374 4.810628e-04
## Methodmixture -0.025534406 9.492872e-03
## Methodnet     -1.031878909 9.739311e-03
## Methodspo     -1.073924660 1.490585e-02
## Methodtrap    -1.933341815 1.280765e-02
```

```
#The value should be between these two different quantiles.
```

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

```
fit_glm <- glm(Angaus~., data=Anguilla_train)
summary(fit_glm)$coefficient
```

```
##                     Estimate   Std. Error     t value     Pr(>|t|)
## (Intercept)   -7.244757e-01 1.967101e-01 -3.68296205 2.459016e-04
## Site          -5.727670e-05 4.256823e-05 -1.34552701 1.788330e-01
## SegSumT        7.481135e-02 8.595260e-03  8.70379219 1.789767e-17
## SegTSeas       4.008325e-02 1.371415e-02  2.92276542 3.566327e-03
## SegLowFlow     5.973291e-02 7.831727e-02  0.76270420 4.458627e-01
## DSDist        -5.114056e-05 2.212579e-04 -0.23113552 8.172681e-01
## DSMaxSlope    -4.427727e-03 4.711642e-03 -0.93974178 3.476312e-01
## USAvgT         2.094043e-02 1.584343e-02  1.32171063 1.866391e-01
## USRainDays    -6.935122e-02 1.605956e-02 -4.31837622 1.767452e-05
## USSlope       -2.745952e-03 2.489550e-03 -1.10299102 2.703599e-01
## USNative      -6.395745e-02 4.627135e-02 -1.38222579 1.672848e-01
## DSDam         -1.334500e-02 4.401995e-02 -0.30315794 7.618476e-01
## Methodmixture  6.493041e-05 6.050110e-02  0.00107321 9.991440e-01
## Methodnet     -1.511368e-01 6.083730e-02 -2.48427911 1.318321e-02
## Methodspo     -1.966807e-01 6.061897e-02 -3.24454140 1.224830e-03
## Methodtrap    -3.318698e-01 5.853498e-02 -5.66959900 1.993155e-08
## LocSed        -3.227897e-02 1.097999e-02 -2.93979834 3.378015e-03
```

```
#Compared this two method, Bayesian estimates is more accurate than maximum likelihood estimation.
#Bayesian estimation fully calculates the posterior distribution.
#theta was treated as a random variable in Bayesian inference.
#We put in and get out PDF in Bayesian estimation, rather than a single point as in MLE.
```

# Part II - Permutation tests

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

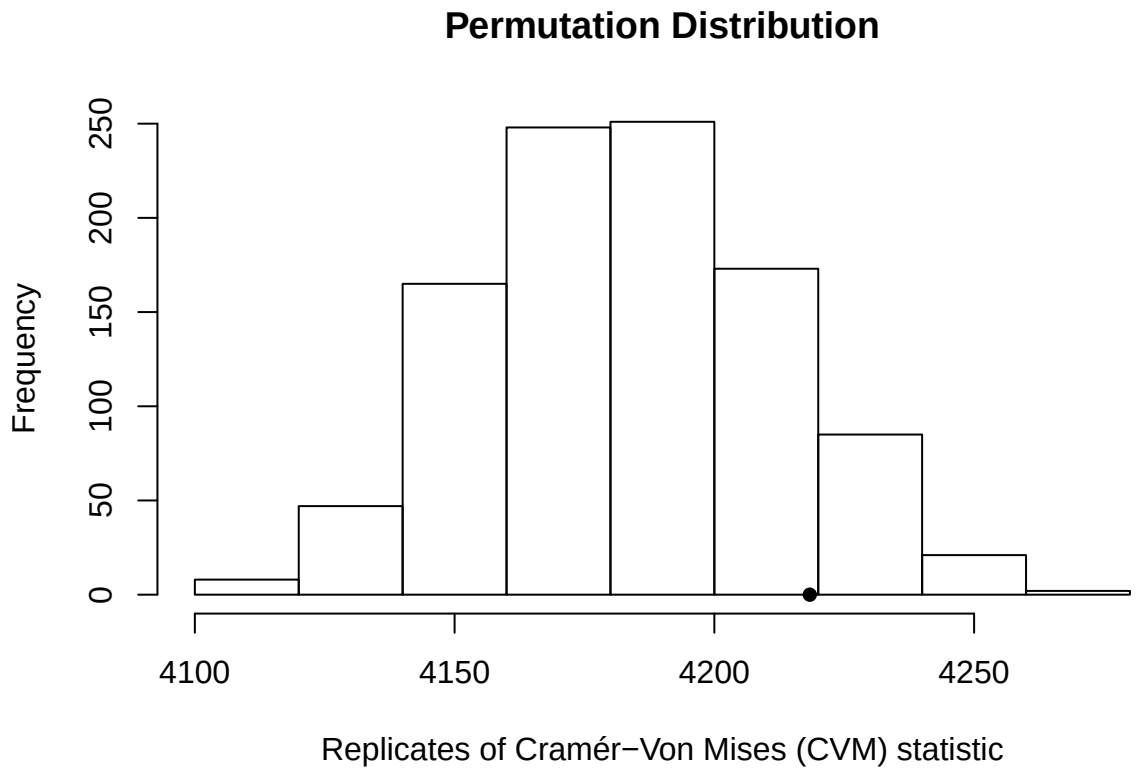
$$W = \frac{mn}{(m+n)^2}\left[\sum_{i=1}^{n}(F_n(x_i) - G_m(x_i))^2 + \sum_{j=1}^{m}(F_n(y_j) - G_m(y_j))^2\right]$$

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.

```
data("chickwts")
## soybean and linseed
x <- with(chickwts, sort(as.vector(weight[feed == "soybean"])))
y <- with(chickwts, sort(as.vector(weight[feed == "linseed"])))
r <- 1000
value <- vector(mode="numeric", length=r)
n <- length(x);m <- length(y)
n_0 <- vector(mode="numeric", length=n);n_1 <- vector(mode="numeric", length=n)
m_0 <- vector(mode="numeric", length=m);m_1 <- vector(mode="numeric", length=m)
z <- c(x, y)
N <- length(z)
for(i in 1:n){n_0[i] <- (x[i]-i)**2}
for(j in 1:m){m_0[j] <- (y[j]-j)**2}
```

```
# statistic
value_0 <- ((n*sum(n_0)+m*sum(m_0))/(m*n*N))-(4*m*n-1)/(6*N)
# Permautation
for (k in 1:r) {
  w <- sample(N, size = n, replace = FALSE)
  x1 <- sort(z[w])
  y1 <- sort(z[-w])
  for (i in 1:n){n_1[i] <- (x1[i]-i)**2}
  for (j in 1:m){m_1[j] <- (y1[j]-j)**2}
  value[k] <- ((n*sum(n_1)+m*sum(m_1))/(m*n*N))-(4*m*n-1)/(6*N)}
p <- mean(c(value_0, value)>=value_0)
hist(value,xlab = "Replicates of Cramér-Von Mises (CVM) statistic",
     main="Permutation Distribution")
points(value_0,0, cex=1, pch=16)
```




```
#P value is:
p
```

```
## [1] 0.1128871
```

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.

```
data("iris")
#setosa
```

```r
data_1 <- as.matrix(iris[1:50, 1:4])
x <- (data_1[ ,1:2])
y <- (data_1[ ,3:4])
cor(x,y, method = "spearman")
```

```
##              Petal.Length Petal.Width
## Sepal.Length    0.2788849   0.2994989
## Sepal.Width     0.1799110   0.2865359
```

```r
cor.test(x, y, method = "spearman", exact = FALSE)$estimate
```

```
##       rho
## 0.8292811
```

```r
#versicolor
data_2 <- as.matrix(iris[51:100, 1:4])
x <- (data_2[ ,1:2])
y <- (data_2[ ,3:4])
cor(x,y, method = "spearman")
```

```
##              Petal.Length Petal.Width
## Sepal.Length    0.7366251   0.5486791
## Sepal.Width     0.5747272   0.6599826
```

```r
cor.test(x, y, method = "spearman", exact = FALSE)$estimate
```

```
##       rho
## 0.9254379
```

```r
#virginica
data_3 <- as.matrix(iris[101:150, 1:4])
x <- (data_3[ ,1:2])
y <- (data_3[ ,3:4])
cor(x,y, method = "spearman")
```

```
##              Petal.Length Petal.Width
## Sepal.Length    0.8243234   0.3157721
## Sepal.Width     0.3873587   0.5443098
```

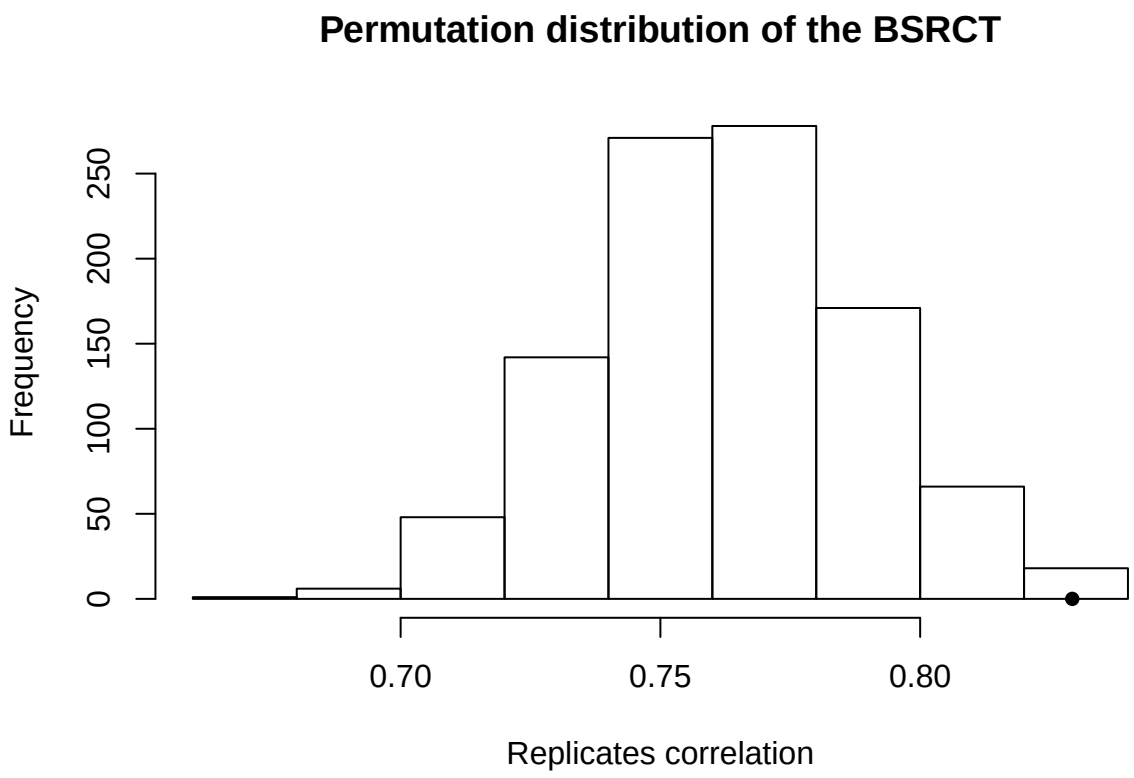
```r
cor.test(x, y, method = "spearman", exact = FALSE)$estimate
```

```
##       rho
## 0.9221177
```

```r
library(boot)
#Using data_1 (setosa) as an example
z <- data_1
rho <- function(z,i){
  x <- z[ ,1:2];y <- z[i,3:4]
  return(cor.test(x, y, method = "spearman", exact = FALSE)$estimate)}
#significance level: 1000 permutation samples
permutation_test <- boot(data=z,statistic=rho,sim ="permutation",R=1000)
#p-value
p <- with(permutation_test, mean(c(t,t0)>=t0))
p
```

```
## [1] 0.006993007
```

```r
# BSRCT means Permutation distribution of the bivariate Spearman rank correlation test
hist(with(permutation_test, c(t, t0)),xlab = "Replicates correlation",
     main="Permutation distribution of the BSRCT")
points(permutation_test$t0,0, cex=1, pch=16)
```




```r
#The P value from the permutation test is smaller than
#that reported by `cor.test` on the same samples
```