STAT 206 Final_Lihua Xu – Due Wednesday, December 13, 2017, 5:00 PM

General instructions: Final 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. **The final exam is open book/internet access, but absolutely no communicating with other humans.** Any questions you have must be directed to me.

Part I - Flour beetle population

The table below provides counts of a flour beetle population at various points in time. Beetles in all stages of development were counted, and the food supply was careful controlled.

Days (t_i)	0	8	28	41	63	79	97	117	135	154
Beetles ($N^{obs}(t_i)$)	2	47	192	256	768	896	1120	896	1184	1024

An elementary model for population growth is the logistic model given by

$$N(t) = \frac{2K}{2 + (K-2)\exp\{-rt\}}$$

where $N(t)$ is the population size at time t, r is a growth parameter and K is a parameter that represents the population carrying capacity of the environment. A popular method to estimate the parameters (K, r) is to minimize the objective function

$$\begin{aligned} g(K,r) &= \sum_{i=1}^{n} \left(N(t_i) - N^{obs}(t_i)\right)^2 \\ &= \sum_{i=1}^{n} \left(\frac{2K}{2 + (K-2)\exp\{-rt_i\}} - N^{obs}(t_i)\right)^2 \end{aligned}$$

with respect to K and r. Here n represents the sample size, and t_i take the values $0, 2, 8, 28, \ldots$ (see the table above).

1. Evaluate the function $g(K, r)$ over an appropriately chosen two-dimensional grid. Produce a surface plot using the function `persp()`.

- Solution:
- I use two sets of data for t_i and $(N^{obs}(t_i))$ to roughly estimate K and r using the logistic model $N(t)$ equation. When $t_i = 8$, $N^{obs}(8) = 47$. And when $t_i = 28$, $N^{obs}(28) = 192$. Bringing these vales into the equation $N(t)$:

$$47 = \frac{2K}{2 + (K-2)\exp\{-r * 8\}} 192 = \frac{2K}{2 + (K-2)\exp\{-r * 28\}}$$

- Then we can get a rough value for K and r.

$$K = 0.4283(around) r = 191.4893$$

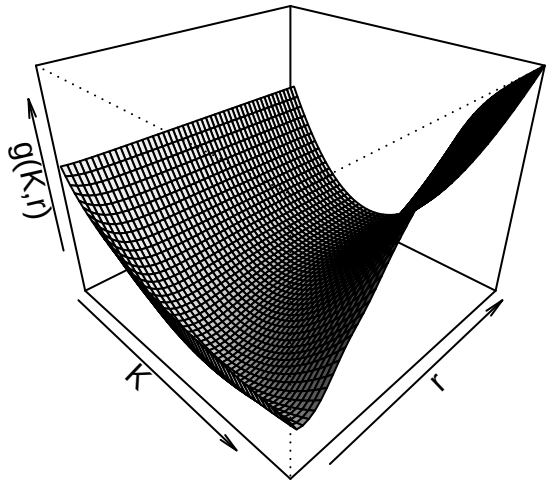
- The above values are just from two sets of data, as there are many dat sets. Based on these, I set the two-dimensional grid initially as following:

$$K - [0, 2000], r - [0, 1]$$

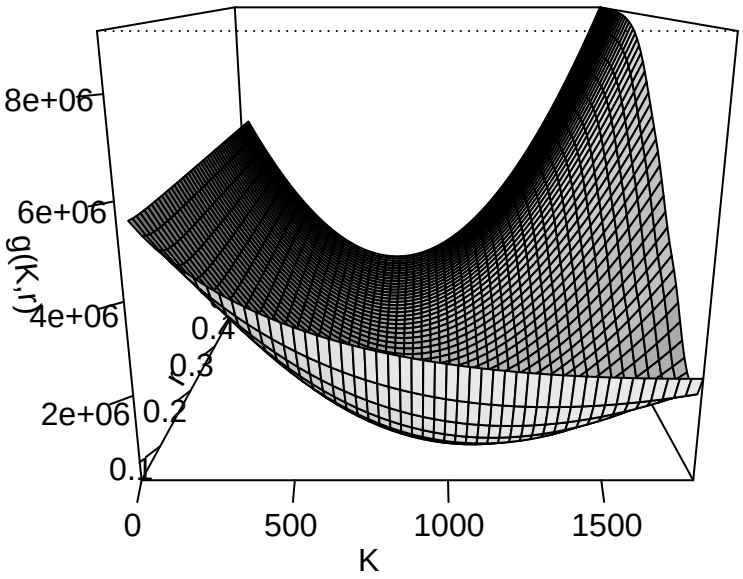
- After viewing the plot using the following codes and doing some adjustment for the two-dimensional grid (in order to make the plot more redable), I finally find the appropriate two-dimensional grid for me:

$$K - [0, 1800], r - [0.05, 0.5]$$

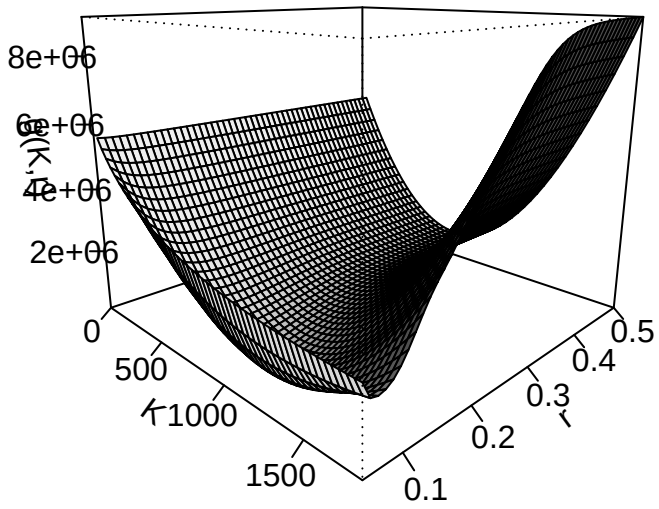
```
gf <- function(K,r){
  t <- c(0,8,28,41,63,79,97,117,135,154)
  N_obs <- c(2,47,192,256,768,896,1120,896,1184,1024)
  sum((2*K/(2+(K-2)*exp(-r*t))-N_obs)**2)}
K <- seq(0,1800,length.out=50)
r <- seq(0.05,0.5,length.out=50)
Z_new <- matrix(0,nrow=50,ncol=50)
for (i in 1:50){
  for (j in 1:50){
  Z_new[i,j] <- gf(K[i],r[j])
  }}
#We viw this plot different point so as to estimate the value for K abd r
#which minimize the function g(K,r)
persp(K,r,Z_new,theta=45,phi=30,expand=0.8,ltheta = 120,shade=0.5,
      ticktype="simple",xlab ="K",ylab ="r", zlab = "g(K,r)")
```



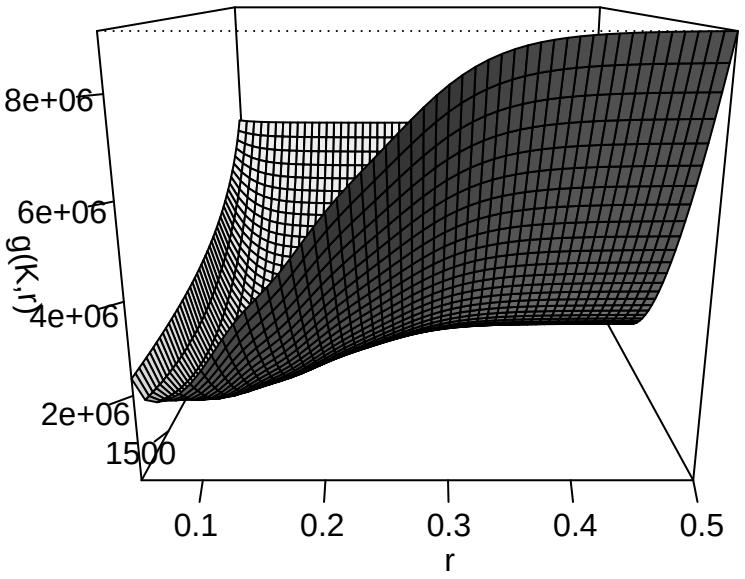

```
persp(K,r,Z_new,expand=0.8,shade=0.5,ticktype="detailed",xlab ="K",
      ylab ="r", zlab = "g(K,r)")
```




```
persp(K,r,Z_new,theta=45,expand=0.8,ltheta = 120,shade=0.5,
      ticktype="detailed",xlab ="K",ylab ="r", zlab = "g(K,r)")
```




```
persp(K,r,Z_new,theta=90,expand=0.8,ltheta = 120,shade=0.5,
      ticktype="detailed",xlab ="K",ylab ="r", zlab = "g(K,r)")
```



2. Based on the results of part (1), provide estimates $(\hat{K}, \hat{r})$ of the parameters (K, r), which will minimize the function g.

- Siolution:
- Based on the results from Problem 1, my initial estimation for K and r would be $K = 1000$, $r = 0.1$, respectively.

```
#I will use the optim function in R.
gf_minimize <- function(parameter){
  t <- c(0,8,28,41,63,79,97,117,135,154)
  N_obs <- c(2,47,192,256,768,896,1120,896,1184,1024)
  sum((2*parameter[1]/(2+(parameter[1]-2)*exp(-parameter[2]*t))-N_obs)**2)}
optim(c(1000,0.1),gf_minimize,method="Nelder-Mead",hessian = TRUE)
```

```
## $par
## [1] 1033.4751685    0.1179645
##
## $value
## [1] 83240.49
##
## $counts
## function gradient
##      119       NA
##
## $convergence
## [1] 0
##
```

```
## $message
## NULL
##
## $hessian
##              [,1]         [,2]
## [1,]    10.29355     23253.74
## [2,] 23253.74071 306984962.12
```

```
#Here I used "Nelder-Mead" algorithm which is often used in caculating the extreme value
#for multivariate function.
#The estimated value for K is 1033.5 and the estimated value for r is 0.118.
#In this case, the value for g(K,r) is 83240.49.
```

3. In many population modeling applications, an assumption of log-normality is adopted: $\log(N^{obs}(t))$ are independent and normally distributed with mean $\log(N(t))$ and variance $\sigma^2 = 1$. Design a Monte Carlo approach to estimate the sampling distribution of the estimates found in (2). Implement your approach and display histograms for the sampling distributions for $\hat{K}$ and $\hat{r}$.

- Solution: (We will use the random work Metropolis Hastings samplers to do the parameter sampling distribution.)
- We will seperate the two parameters:
- First, we will do MC for K:
- (1)Generating $K^* = K_t + \epsilon$ in which $\epsilon \sim Uniform(-1, 1)$
- (2)As the mean for $Uniform(-1, 1)$ is $(-1 + 1)/2 = 0$, the MH ratio would be:

$$R(K_t, K^*) = \frac{f(K^*)}{f(K_t)}$$

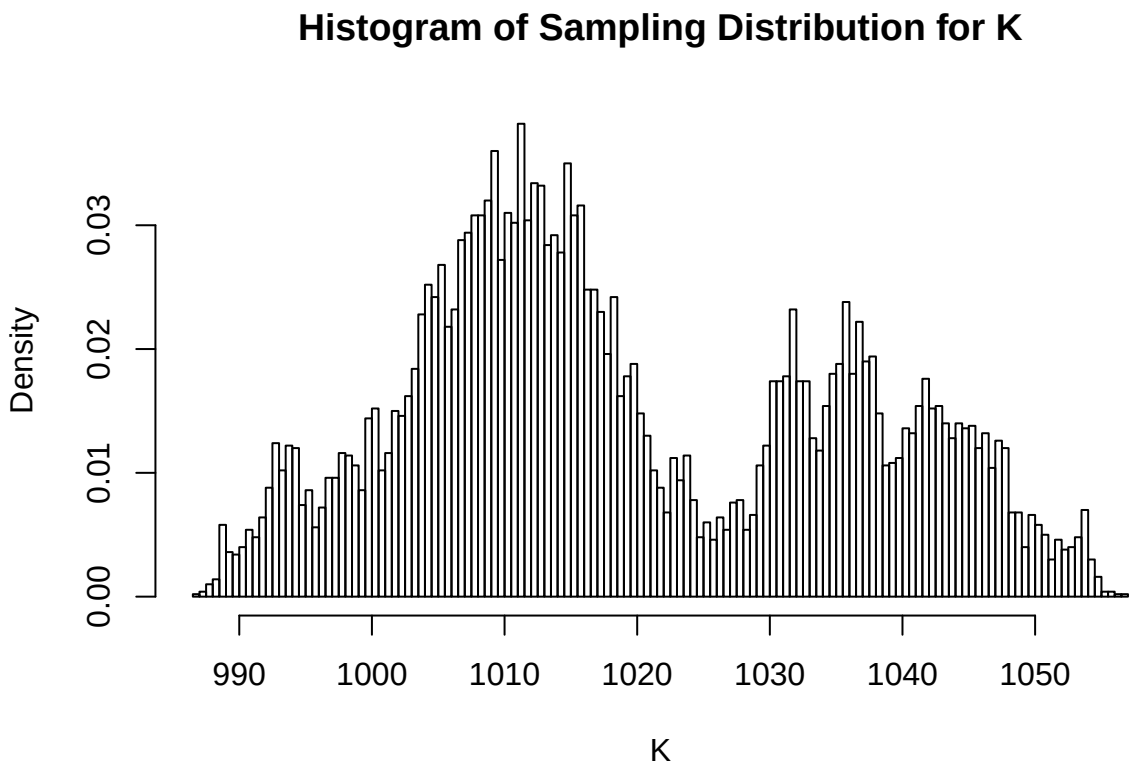
- (3)We can set the new X_{t+1}:

$$\alpha(K^*, K_t) = min(1, \frac{f(K^*)f(K_t)}{f(K_t)f(K^*)})$$

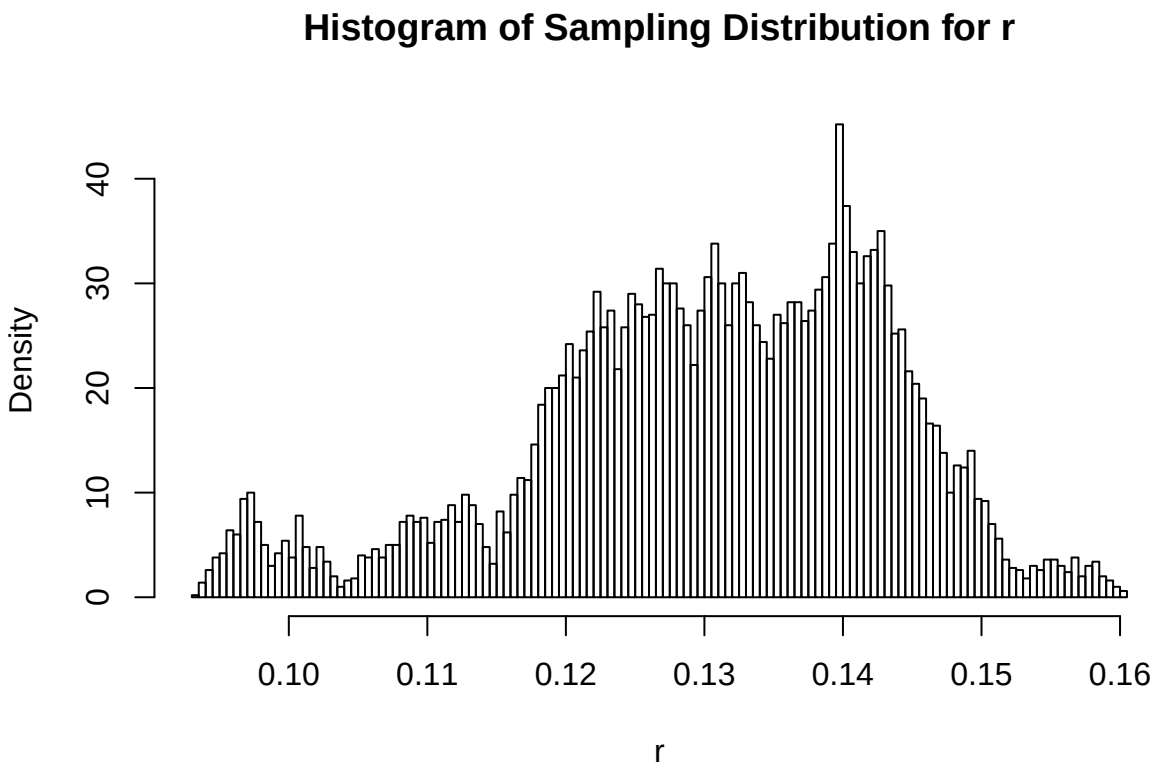
- Same for r parameter.

```
#The random walk MH function:
n <- 10000
t <- c(0,8,28,41,63,79,97,117,135,154)
K_t <- numeric(n)
K_true <- 1033.4751685
r_true <- 0.1179645
K_t[1] <- 1000
sigma_value <- 1
N_obs_initial <- log((2*K_true)/(2+(K_true-2)*exp(-r_true*t)))
log_N_obs <- rnorm(10, N_obs_initial,sigma_value)
#First I will estimate for the parameter K.
for (i in 2:n) {
K_estimate <- K_t[i-1]+runif(1,-1,1)
N_obs_initial_value <- log((2*K_t[i-1])/(2+(K_t[i-1]-2)*exp(-r_true*t)))
log_N_obs_K_value <- dnorm(log_N_obs, N_obs_initial_value,sigma_value)
N_obs_initial_prime <- log((2*K_estimate)/(2+(K_estimate-2)*exp(-r_true*t)))
log_N_obs_K.prime <- dnorm(log_N_obs, N_obs_initial_value,sigma_value)
u <- prod(log_N_obs_K.prime/log_N_obs_K_value)
if (runif(1) < u && K_estimate >=0)
  {K_t[i] <- K_estimate}
```

```
else{K_t[i] <- K_t[i-1]}}
hist(K_t,prob=TRUE,breaks=100,xlab="K",main="Histogram of Sampling Distribution for K")
```




```
#Secondly, I will estimate for the parameter r.
n <- 10000
t <- c(0,8,28,41,63,79,97,117,135,154)
r_t <- numeric(n)
K_true <- 1033.4751685
r_true <- 0.1179645
r_t[1] <- 0.1
sigma_value <- 1
for (i in 2:n) {
r_estimate <- r_t[i-1]+runif(1,-0.001,0.001)
N_obs_initial_value <- log((2*K_true)/(2+(K_true-2)*exp(-r_t[i-1]*t)))
log_N_obs_r_value <- dnorm(log_N_obs, N_obs_initial_value,sigma_value)
N_obs_initial_prime <- log((2*K_true)/(2+(K_true-2)*exp(-r_estimate*t)))
log_N_obs_r.prime <- dnorm(log_N_obs, N_obs_initial_value,sigma_value)
u <- prod(log_N_obs_r.prime/log_N_obs_r_value)
if (runif(1) < u && r_estimate >=0)
  {r_t[i] <- r_estimate}
else{r_t[i] <- r_t[i-1]}}
hist(r_t,prob=TRUE,breaks=100,xlab="r",main="Histogram of Sampling Distribution for r")
```

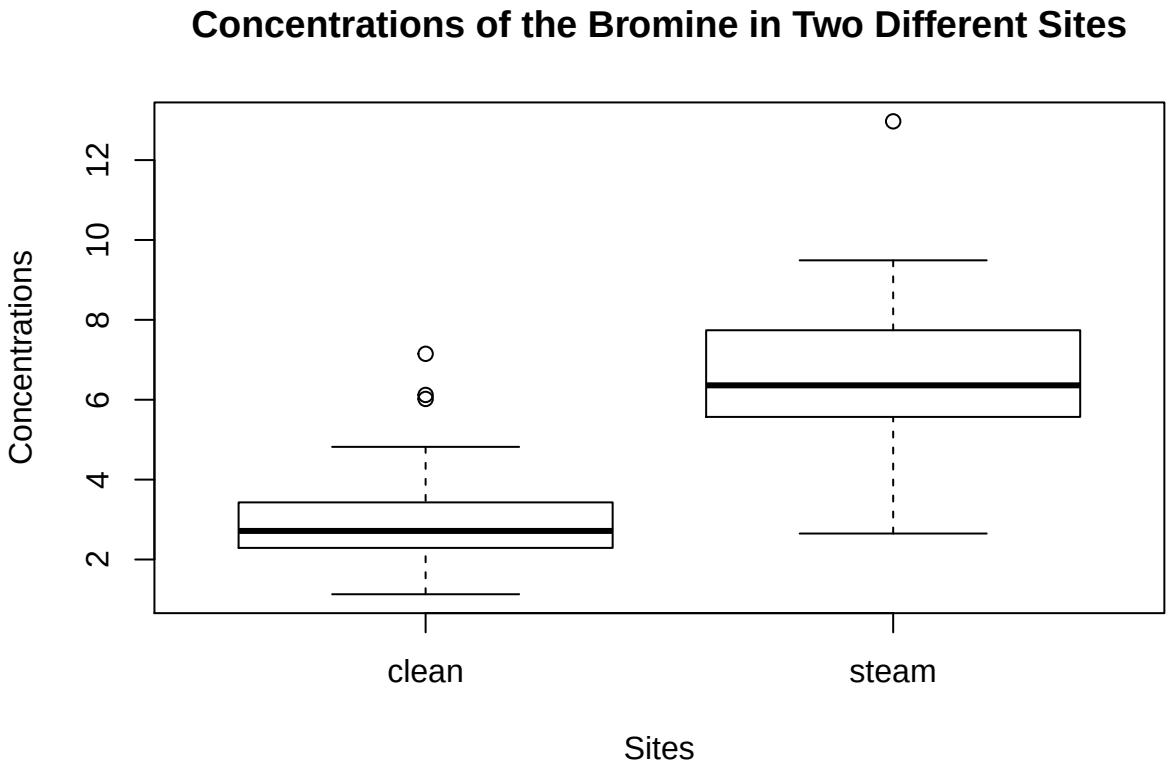


Part II - Pine needles

In the article "Pine needle as sensors of atmospheric pollution", the authors use neutron-activity analysis to determine pollution levels, by measuring the Bromine concentration in pine needles. The investigators collect 18 pine needles from a plant near an oil-fired steam plant and 22 near a cleaner site. The data can be found at [http://faculty.ucr.edu/~jflegal/206/pine_needles.txt].

4. Describe the data using plots and summary statistics. Show that the data are **not** normally distributed by drawing an appropriate graphical display for each sample.

```
url <- 'http://faculty.ucr.edu/~jflegal/206/pine_needles.txt'
Pine_needle_data <- read.table(url,head=TRUE)

#Showing the plot for each data and the summary of the data:
plot(Pine_needle_data,xlab="Sites",ylab="Concentrations",
     main="Concentrations of the Bromine in Two Different Sites")
```




```
clean_s <- list(clean=summary(Pine_needle_data[Pine_needle_data$site=="clean",]$concentrations))
clean_s
```

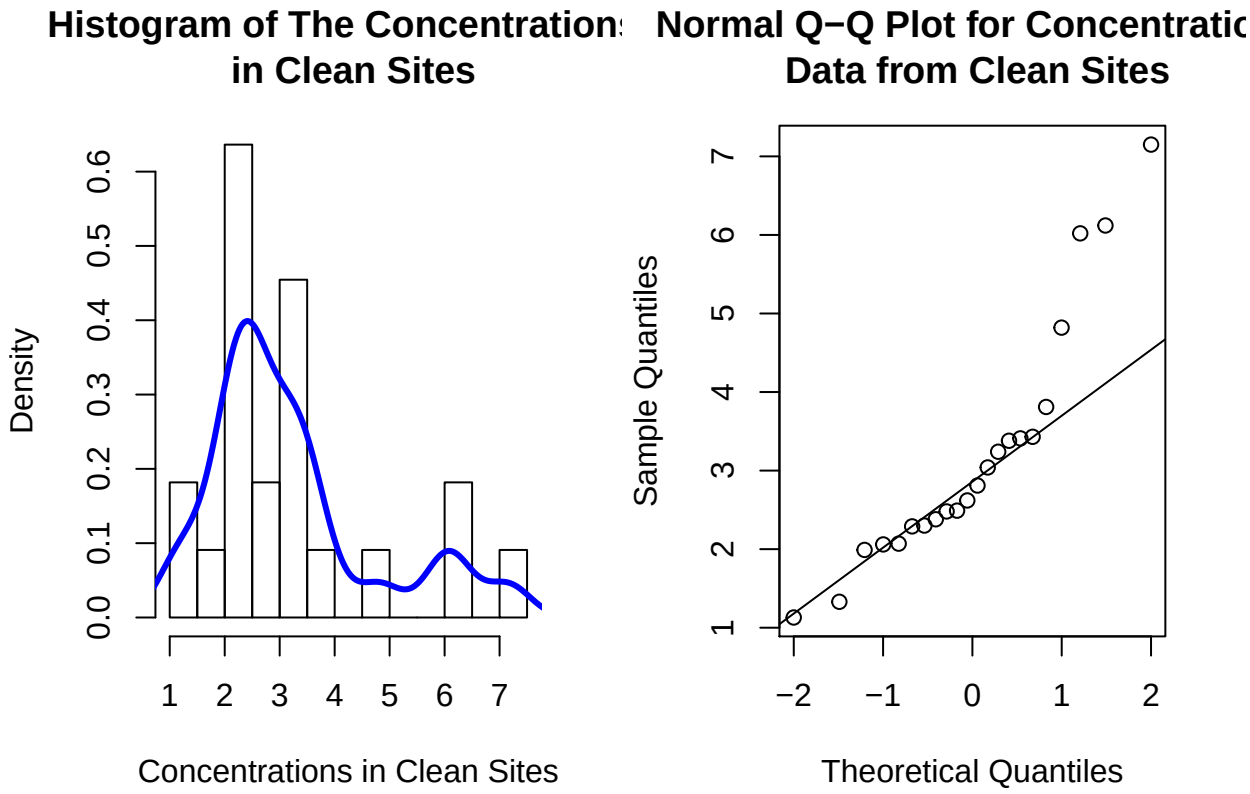
```
## $clean
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.130   2.292   2.715   3.199   3.425   7.150
```

```
steam_s <- list(steam=summary(Pine_needle_data[Pine_needle_data$site=="steam",]$concentrations))
steam_s
```

```
## $steam
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.650   5.607   6.360   6.602   7.692  12.970
```

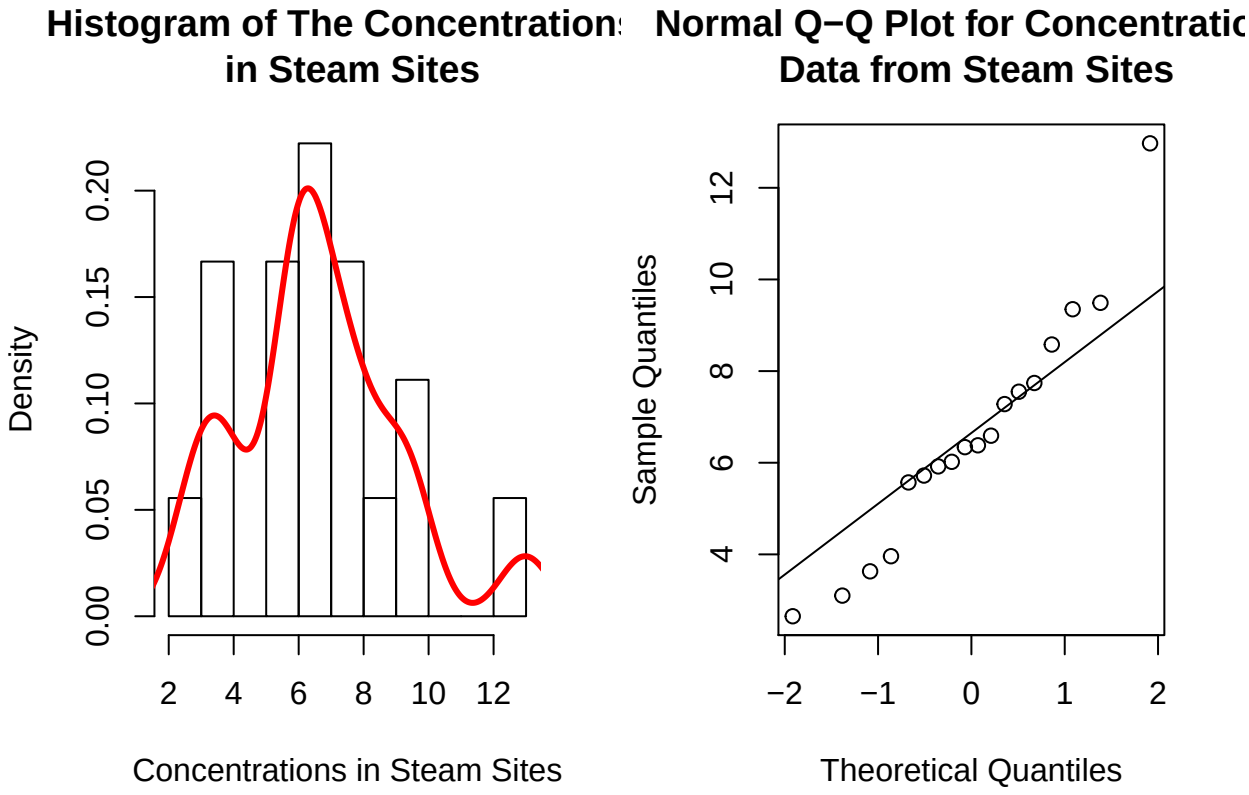
```
#From both the plots and the summary, we find that the concentration of the Bromine
#in the clean sites are always less than that oil-fired steam plant.
#The minimum, median, mean and maximum concentration near the clean sites
#are 1.130, 2.715, 3.199,7.150, respectively.
#The minimum, median, mean and maximum concentration near the steam sites
#are 2.650, 6.360, 6.602, 12.970, respectively.

#Showing the data are not normally distributed.
#Clean site:
par(mfrow=c(1,2))
Clean_data <- Pine_needle_data[Pine_needle_data$site=="clean",]$concentrations
hist(Clean_data,breaks=13,main = "Histogram of The Concentrations \n in Clean Sites",
     freq=FALSE,xlab="Concentrations in Clean Sites")
lines(density(Clean_data),col="Blue",lwd=3)
```

```r
qqnorm(Clean_data,main="Normal Q-Q Plot for Concentration \n Data from Clean Sites")
qqline(Clean_data)
```




```r
#After using the density command to fit the data, the blue curve shows the trend.
#There are two peaks, one is small and one is big.
#The curve doesn't mirror the shape for the normal distribution.
#Also, I plot the Q-Q plot, the points at higher sample quantiles deviate
#from the straight line more compared with the lower sample quantiles.
#Thus, the data doesn't follow the normal distribution.
```

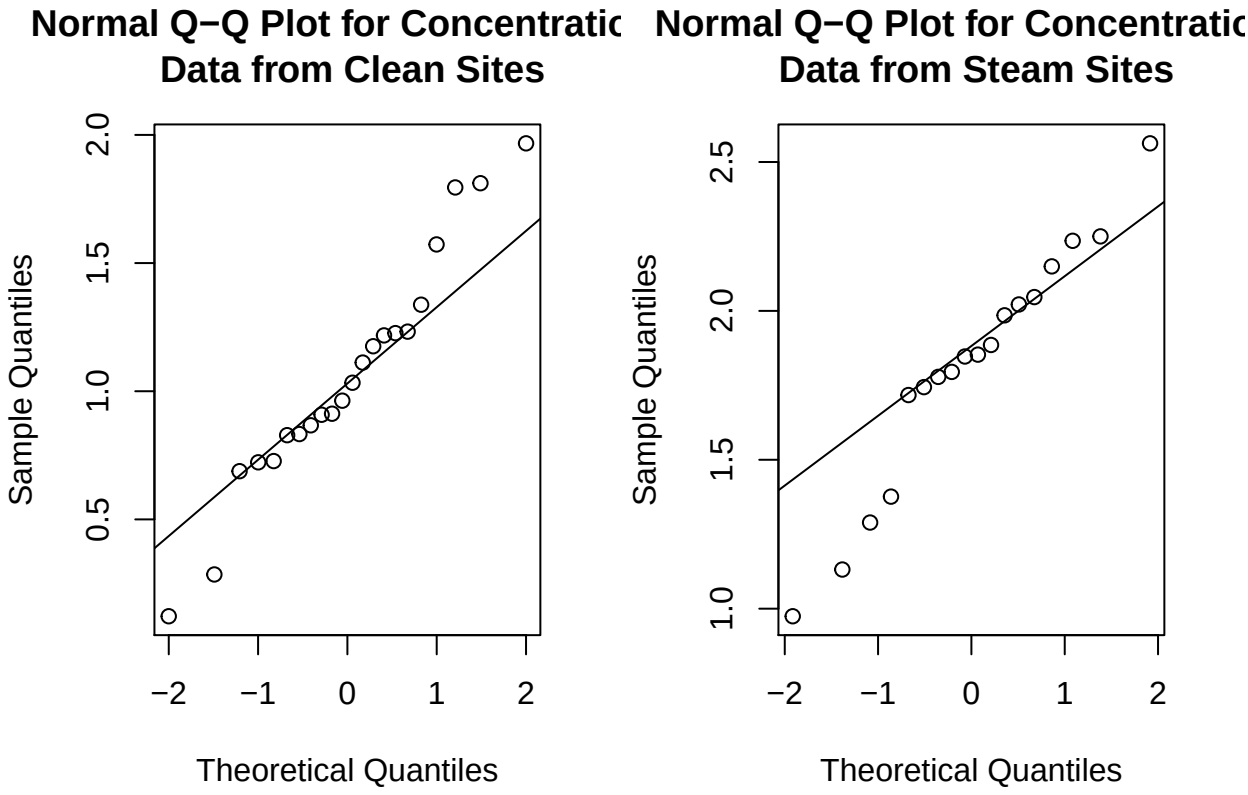
```r
#Steam sites:
par(mfrow=c(1,2))
Steam_data <- Pine_needle_data[Pine_needle_data$site=="steam",]$concentrations
hist(Steam_data,breaks=11,main = "Histogram of The Concentrations \n in Steam Sites",freq=FALSE,xlab="C
lines(density(Steam_data),col="red",lwd=3)
qqnorm(Steam_data,main="Normal Q-Q Plot for Concentration \n Data from Steam Sites")
qqline(Steam_data)
```




```
#After using the density command to fit the data, the red curve shows the trend.
#There are three peaks, two are small and one is big.
#The curve doesn't mirror the shape for the normal distribution.
#Also, I plot the Q-Q plot, the points at higher and lower sample
#quantiles don't follow the straight line very well.
#So the data doesn't follow the normal distribution.
```

5. Take a log transformation of the values in each sample. Does it seems reasonable that the transformed samples are each drawn from a normal distribution? Test this formally using an appropriate test (of your choosing).

```
log_clean_data <- log(Clean_data)
log_steam_data <- log(Steam_data)
#Q-Q Plots:
par(mfrow=c(1,2))
qqnorm(log_clean_data,main="Normal Q-Q Plot for Concentration \n Data from Clean Sites")
qqline(log_clean_data)
qqnorm(log_steam_data,main="Normal Q-Q Plot for Concentration \n Data from Steam Sites")
qqline(log_steam_data)
```




```
#The deviation of the point from the straight line become smaller,
#and all the points are more close to the straight line.
#It seems reasonable that the transformed samples are each drawn from a normal distribution.

#Test:(Kolmogorov-Smirnov test)
ks_clean <- ks.test(log_clean_data,"pnorm",mean=mean(log_clean_data),sd=sqrt(var(log_clean_data)))
list(ks.test.clean=ks_clean)
```

```
## $ks.test.clean
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  log_clean_data
## D = 0.12666, p-value = 0.8291
## alternative hypothesis: two-sided
```

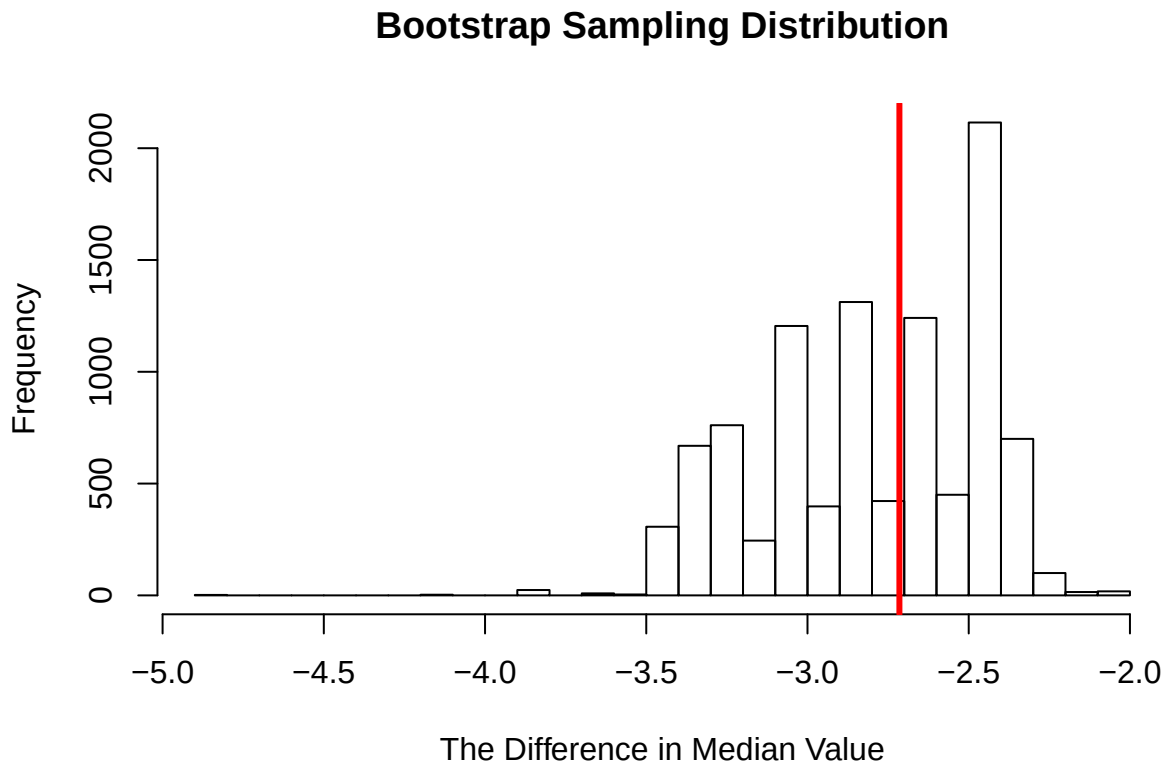
```
ks_steam <- ks.test(log_steam_data,"pnorm",mean=mean(log_steam_data),sd=sqrt(var(log_steam_data)))
list(ks.test.steam=ks_steam)
```

```
## $ks.test.steam
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  log_steam_data
## D = 0.18467, p-value = 0.5127
## alternative hypothesis: two-sided
```

```
#As in both case, P-value is larger than D value, both the data are normally distributed.
```

6. Now suppose that the authors of this study want to calculate an interval for the difference between the median concentrations at the two sites, on the original measurement scale. Write code to calculate a 95% bootstrap interval for the difference in the medians between the two samples. Summarize your conclusion in words.

```
bootstrap.resample <- function (object) sample(object, length(object), replace=TRUE)
diff.medians <- function(Pine_needle_data) {
  median(Pine_needle_data[Pine_needle_data$site=="steam",]$concentrations)
  - median(Pine_needle_data[Pine_needle_data$site=="clean",]$concentrations)}
#Resample 10000 times
resample.madian.diffs <- replicate(10000,
    diff.medians(Pine_needle_data[bootstrap.resample(1:nrow(Pine_needle_data)),]))
hist(resample.madian.diffs,breaks=20,main="Bootstrap Sampling Distribution",
     xlab="The Difference in Median Value")
abline(v=diff.medians(Pine_needle_data), col="red", lwd=3)
```




```
#95% bootstrap interval:
quantile(resample.madian.diffs, c(0.025,0.975))
```

```
##  2.5% 97.5%
## -3.41 -2.30
```

```
#The Plot is the distribution of the Bootstrap resampling data
#(median value between two sets of data), and the read line is the true median
#value difference between two set of data.
#The estimated confidence interval with level 0.95 is
```

```
#between the above two values.
```

Part III - Metropolis Hastings

Suppose we have observed data $y_1, y_2, \ldots, y_{200}$ sampled independently and identically distributed from the mixture distribution

$$\delta N(7, 0.5^2) + (1 - \delta)N(10, 0.5^2).$$

7. Simulate 200 realizations from the mixture distribution above with $\delta = 0.7$.

```
n <- 200
mu_values <- c(7,10)
sd_values <- 0.5
distribution_parts <- sample(1:2,prob=c(0.7,0.3),size=n,replace=TRUE)
realizations <- rnorm(n,mean=mu_values[distribution_parts],sd=sd_values)
#The 200 realizations from the mixture distribution with delta equal to 0.7 would be
realizations
```

```
##   [1]  7.334270  6.969122  9.385685 10.221707  7.247931  6.809790 10.636375
##   [8]  7.502622  9.691597  6.724820  6.214960  7.281888  6.674500  6.146899
##  [15]  6.986811 10.107488  7.550630  7.582592  6.671169 10.153342  7.140264
##  [22] 10.506625  7.006771  7.241683  7.441120  6.343533  7.015339 10.141343
##  [29] 10.237327 10.663194  7.619762  5.744516  6.657076  7.270257  7.060568
##  [36]  6.135545  7.106645  9.284209 10.565027  9.870784  7.204705  6.199927
##  [43]  9.200756  7.397383  5.876238  9.297838 10.091258  7.510689  7.743076
##  [50] 10.788690  5.532920  6.618653  6.671499  7.038464  6.533770  9.876539
##  [57]  9.099823  8.108222  6.975746  7.329292  9.728940  9.833275  7.000290
##  [64]  7.850102  6.931016  7.164077  6.463716  7.209082  6.052455  9.723377
##  [71]  6.955888 10.009763  7.462411 10.063007  7.155772 10.272660  7.041590
##  [78]  7.626556  7.481992  7.531186  7.442491  9.892342  6.878171  6.252729
##  [85]  6.895599  7.741869  6.785795 11.328084 10.482521  9.421931 10.582157
##  [92]  7.606234  6.607081  6.721997  7.468202  7.027392  9.268905  9.799215
##  [99]  7.727544  7.232015  5.904309  9.294022  7.431971  9.378191 10.155264
## [106]  8.400773 10.616448  6.092092  6.612865  6.596828  7.075832  6.082403
## [113]  7.133593  7.157411  7.058412 10.036507  9.628687  9.443072  6.924974
## [120]  7.005562 10.133211  7.219348 10.454523  6.833648  6.903818  6.497532
## [127] 10.353101  7.754806  7.677377  6.971747  7.372691 10.561805  7.113402
## [134]  6.197961  7.090020 10.345260  7.314659  7.393419  6.417362  7.128616
## [141]  6.924223 10.639436  7.509047 10.055638  6.982601  6.885104  7.226934
## [148]  7.772650  6.949621  7.655752  7.769145  7.081444  6.876756  6.724215
## [155]  9.279393  9.890806 10.107736  7.637763  6.949446  7.129618 10.109626
## [162]  6.916369  6.871674  6.696686  7.453881 10.001591  7.310207 10.380257
## [169] 10.950501  6.697197  7.916077  6.674881  8.851141  7.182730  6.490643
## [176]  6.137721  6.856137 10.238323 11.011103  9.785247 10.343887 10.763797
## [183]  7.134152  5.585180  9.696406  8.345876  7.224170 10.260188  6.821410
## [190] 10.739993  9.844090  6.655272 10.087273 10.073474  6.294637  7.322089
## [197]  7.258161  6.842058 10.451620  9.572902
```

8. Draw a histogram of the data that also includes the true density. How close is the histogram to the true density?

- Solution:

- We know that the Probability density for the normal distribution is :

$$y = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- For $ N (7, 0.5^2)$ and $ N (10, 0.5^2)$, the probability distribution equation would be :

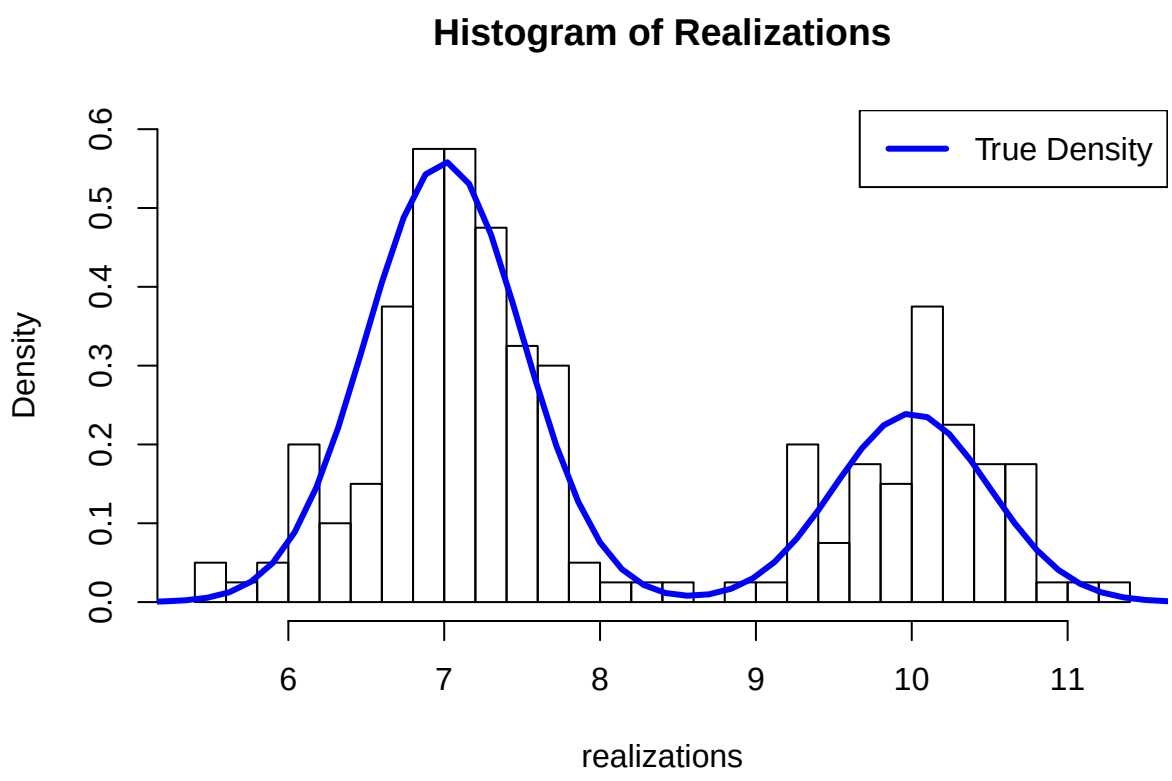
$$y_{N(7,0.5^2)} = \frac{1}{\sqrt{0.5\pi}} e^{-\frac{(x-7)^2}{0.5}}$$

$$y_{N(10,0.5^2)} = \frac{1}{\sqrt{0.5\pi}} e^{-\frac{(x-10)^2}{0.5}}$$

- The true density distribution (Probability density distribution) in this problem would be:

$$y_{true} = 0.7 * \frac{1}{\sqrt{0.5\pi}} e^{-\frac{(x-7)^2}{0.5}} + 0.3 * \frac{1}{\sqrt{0.5\pi}} e^{-\frac{(x-10)^2}{0.5}}$$

```
#The histogram of all the data:
hist(realizations,freq=FALSE,breaks=30,main="Histogram of Realizations",ylim=c(0,0.6))
#True density curve:
curve(0.7*(1/sqrt(0.5*pi)*exp(-(x-7)**2/(0.5)))+0.3*(1/sqrt(0.5*pi)*exp(-(x-10)**2/(0.5))),
      1,15,add=TRUE,col="blue",lw=3,xlab="Realizations")
legend("topright",c("True Density"),lwd=3,col="blue")
```




```
#From the plot, the true density curve matches the data well.
#But if we increase the number of the sampe realizations (increase the amount of data),
#the histogram would be more close to the true density.
```

9. Construct kernel density estimates of your 200 realizations using the Gaussian and Epanechnikov kernels. How do these compare to the true density?

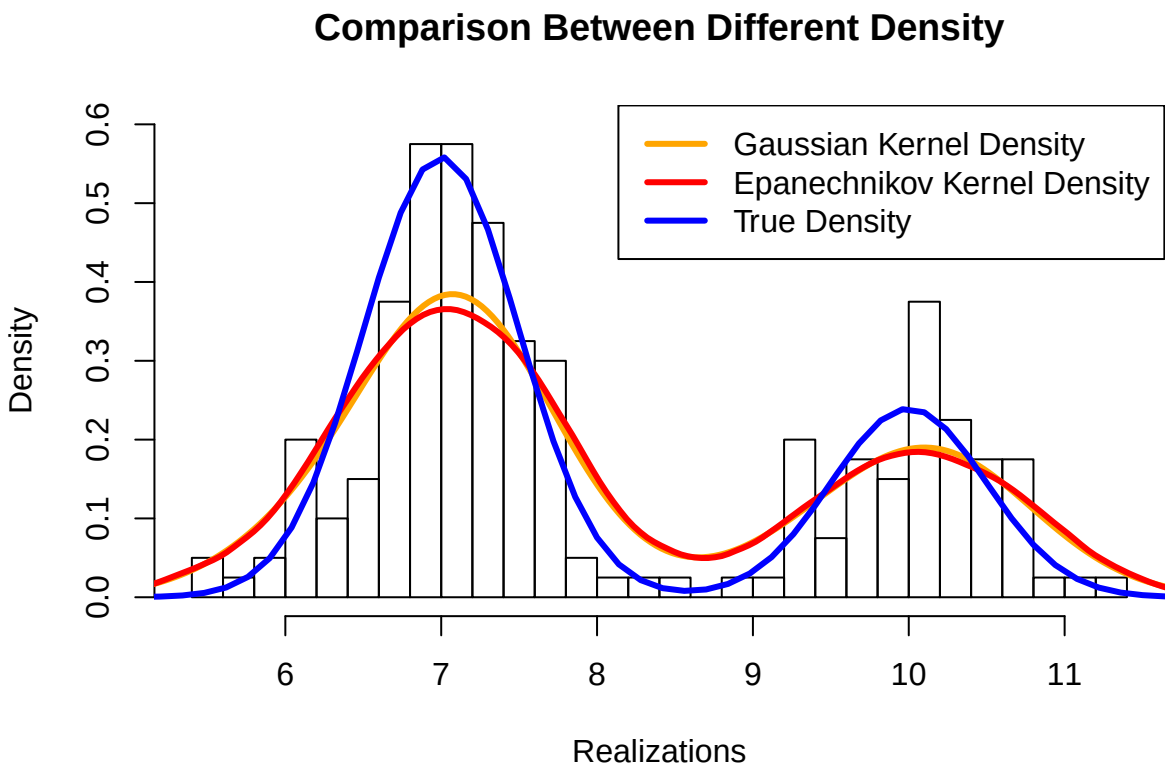
```r
#Kernel density estimates using "Gaussian" kernels
gaussian_kernel <- density(realizations,bw ="nrd0",adjust=1,kernel="gaussian",weights=NULL,
                           window=kernel,width=1,give.Rkern=FALSE,n=200,cut=1, na.rm=FALSE)
gaussian_kernel
```

```
##
## Call:
##  density.default(x = realizations, bw = "nrd0", adjust = 1, kernel = "gaussian",     weights = NULL,
##
## Data: realizations (200 obs.);   Bandwidth 'bw' = 0.4779
##
##        x                y
##  Min.   : 5.055   Min.   :0.00764
##  1st Qu.: 6.743   1st Qu.:0.05940
##  Median : 8.431   Median :0.12396
##  Mean   : 8.431   Mean   :0.14688
##  3rd Qu.:10.118   3rd Qu.:0.18962
##  Max.   :11.806   Max.   :0.38452
```

```r
#Kernel density estimates using "Epanechnikov" kernels
gEpanechnikov_kernel <- density(realizations,bw ="nrd0",adjust=1,kernel="epanechnikov",n=200)
gEpanechnikov_kernel
```

```
##
## Call:
##  density.default(x = realizations, bw = "nrd0", adjust = 1, kernel = "epanechnikov",     n = 200)
##
## Data: realizations (200 obs.);   Bandwidth 'bw' = 0.4779
##
##        x                y
##  Min.   : 4.099   Min.   :0.00000
##  1st Qu.: 6.265   1st Qu.:0.01351
##  Median : 8.431   Median :0.08286
##  Mean   : 8.431   Mean   :0.11498
##  3rd Qu.:10.596   3rd Qu.:0.17593
##  Max.   :12.762   Max.   :0.36555
```

```r
#Compare these density to the true density:
hist(realizations,freq=FALSE,breaks=30,main="Comparison Between Different Density",
     xlab="Realizations",ylim=c(0,0.6))
lines(gaussian_kernel,col="orange",lwd=3)
lines(gEpanechnikov_kernel,col="red",lwd=3)
curve(0.7*(1/sqrt(0.5*pi)*exp(-(x-7)**2/(0.5)))+0.3*(1/sqrt(0.5*pi)*exp(-(x-10)**2/(0.5))),
      1,15,add=TRUE,col="blue",lw=3)
legend("topright",c("Gaussian Kernel Density","Epanechnikov Kernel Density","True Density"),
       lwd=3,col=c("orange","red","blue"))
```




```
#View from the plot, the Epanechnikov Kernel Density cuve and Gaussian Kernel Density curve
#are very similar to each other.
#These two kinds of Kenel density curve are estimated based on the data.
#And the True density cuve resulted from the True equation.
#Thus, there are some difference between the Kernel density curve and the Normal density curve.
#But the trends for the Kernel and Nomal density curves are the same.
```

10. Now assume δ is unknown with a Uniform(0,1) prior distribution for δ. Implement an independence Metropolis Hastings sampler with a Uniform(0,1) proposal.

- Solution (Algorithm):
- (1)Setting the initial value of $\delta_0 = X_0 = x_0$. As δ is the probability, the value should be in the range [0,1].The independence Metropolis Hastings algorithm generates X_{t+1} given $X_t = x_t$
- (2)Sample a candidate value $X^* \sim g(x_t)$ which is the proposed distribution $g(x_t)$. This proposal does not depend on the current state.
- (3)Then we need to compute the Metropolis Hastings ratio $R(x_t, X^*)$ defined as the following expression:

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

- (4)We can set the new X_{t+1}:

$$\alpha(x^*, x_t) = min(1, \frac{f(x^*)f(x_t)}{f(x_t)f(x^*)})$$

In this problem:

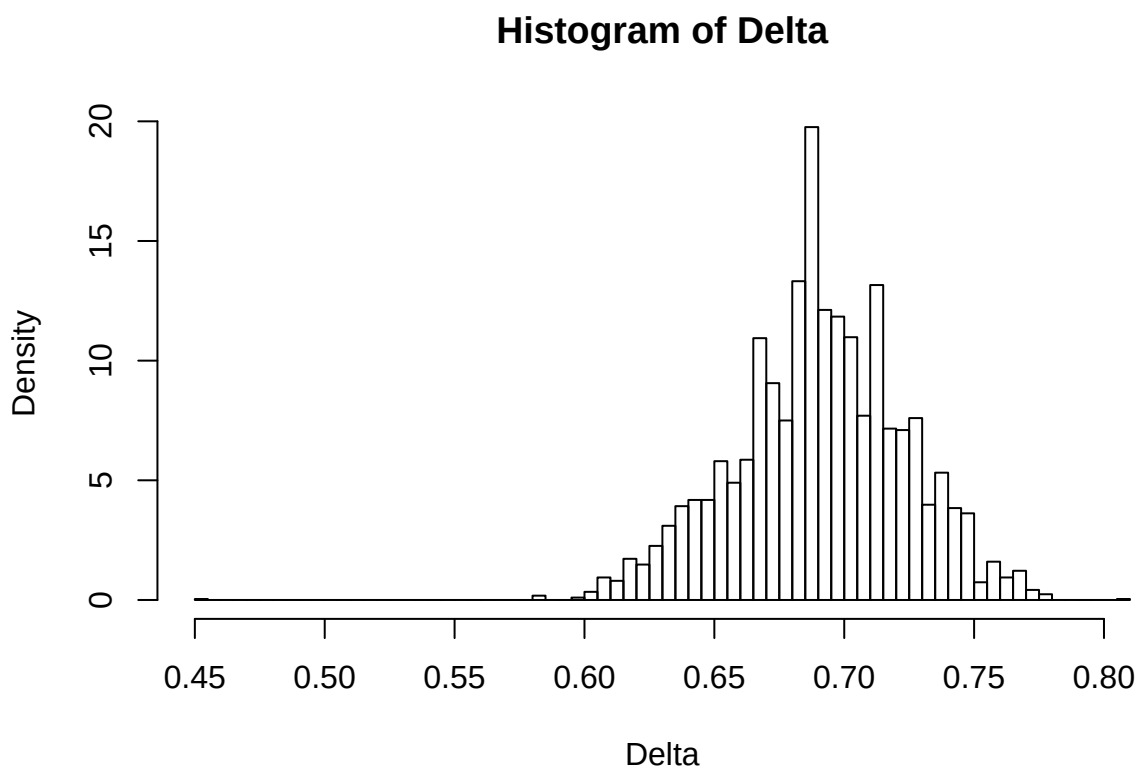
$$f(x) \sim x * N(7, 0.5^2) + (1 - x) * N(10, 0.5^2)$$

$$g(x) \sim Uniform(0, 1)$$

Thus we can get the ratio as :

$$R(x_t, X^*) = \frac{f(x^*)g(x_t)}{f(x_t)g(x^*)} = \frac{1 * \prod(x^* * N(7, 0.5^2) + (1 - x^*) * N(10, 0.5^2))}{\prod(x_t * N(7, 0.5^2) + (1 - x_t) * N(10, 0.5^2)) * 1}$$

```
#The independent MH function:
ind.chain.MH <- function(x_t,n){
x_t <- append(x_t, double(n-1))
mu_values <- c(7, 10)
sigma_values <- 0.5
distribution_parts <- sample(1:2, size=200, replace=TRUE, prob=c(0.7,0.3))
x <- rnorm(200, mu_values[distribution_parts],sigma_values)
for (i in 2:n) {
delta <- runif(1)
x.prime <- delta*dnorm(x,mu_values[1],sigma_values)+(1-delta)*dnorm(x, mu_values[2], sigma_values)
u <- prod(x.prime/(x_t[i-1]*dnorm(x,mu_values[1],sigma_values)+(1-x_t[i-1])
                   *dnorm(x, mu_values[2],sigma_values)))
if (runif(1) < u)
  {x_t[i] <- delta}
else{x_t[i] <- x_t[i-1]}}
return(x_t)}

#The initial value I set for x_t is 0.45, and I will generate 10000 draws.
trial <- ind.chain.MH(0.45,10000)
hist(trial,prob=TRUE,breaks=100,xlab="Delta",main="Histogram of Delta")
```



```
#There are some bars showing in the range [0,0.55].
#That may resulted from the first several times of calculations (burn-in part).
#This histgram shows that the area with more possibilities,
#the delta value will be more close to the target value.
```

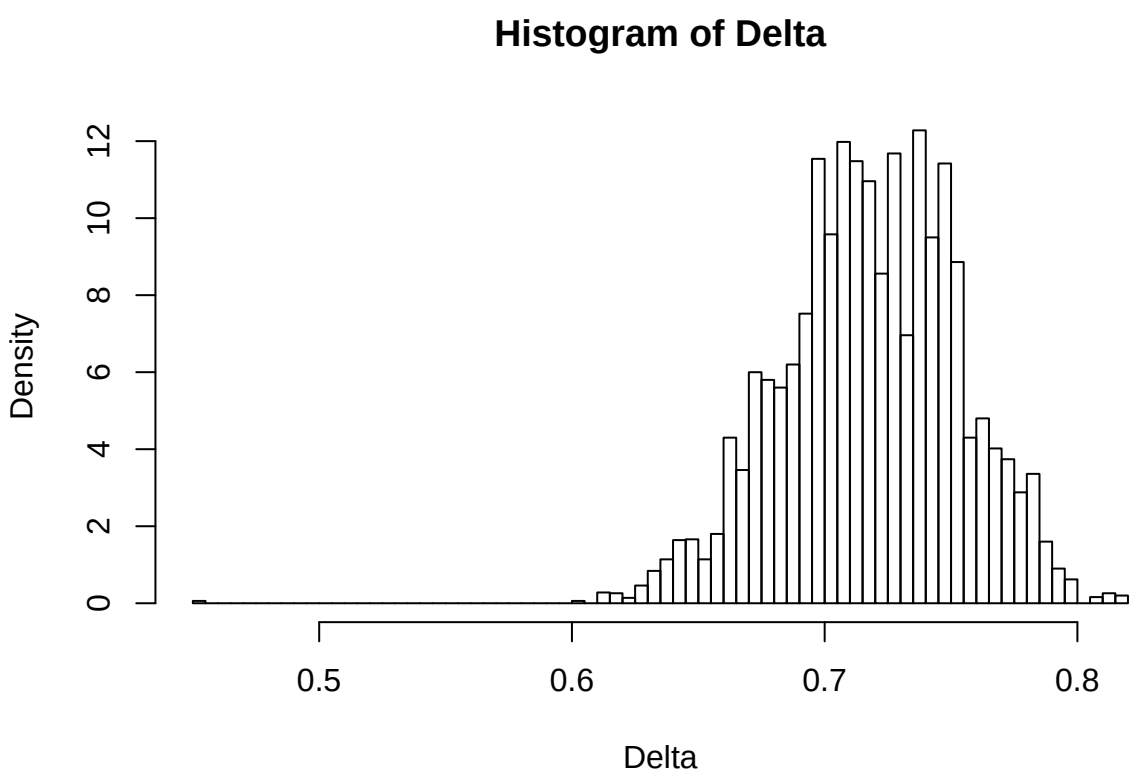
11. Implement a random walk Metropolis Hastings sampler where the proposal $\delta* = \delta^{(t)} + \epsilon$ with $\epsilon \sim$ Uniform$(-1, 1)$.

- Solution:
- (1)Generating $X^* = X_t + \epsilon$ in which $\epsilon \sim Uniform(-1, 1)$
- (2)As the mean for $Uniform(-1, 1)$ is $(-1 + 1)/2 = 0$, the MH ratio would be:

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

- (3)Apply the above X^* and MH ratio in to the algorithm in Prob 10, we can reach to our results.

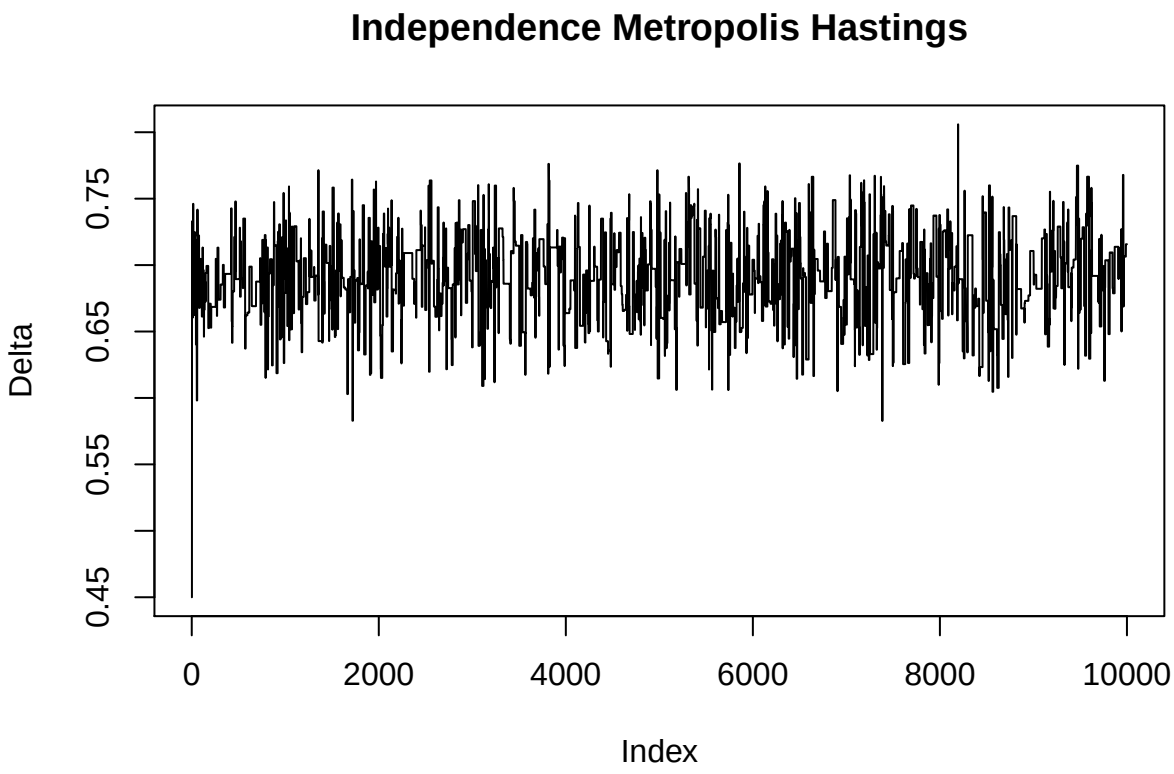
```
#The random walk MH function:
ind.chain.RWHS <- function(x_t,n){
x_t <- append(x_t, double(n-1))
mu_values <- c(7, 10)
sigma_values <- 0.5
distribution_parts <- sample(1:2, size=200, replace=TRUE, prob=c(0.7,0.3))
x <- rnorm(200, mu_values[distribution_parts],sigma_values)
for (i in 2:n) {
delta <- x_t[i-1]+runif(1,-1,1)
x.prime <- delta * dnorm(x,mu_values[1],sigma_values)+(1-delta)*dnorm(x, mu_values[2], sigma_values)
u <- prod(x.prime/(x_t[i-1]*dnorm(x,mu_values[1],sigma_values)+(1-x_t[i-1])
                   *dnorm(x, mu_values[2],sigma_values)))
if (runif(1) < u)
  {x_t[i] <- delta}
else{x_t[i] <- x_t[i-1]}}
return(x_t)}

#The initial value I set for x_t is 0.45, and I will generate 10000 draws.
RW <- ind.chain.MH(0.45,10000)
hist(RW,prob=TRUE,breaks=100,xlab="Delta",main="Histogram of Delta")
```



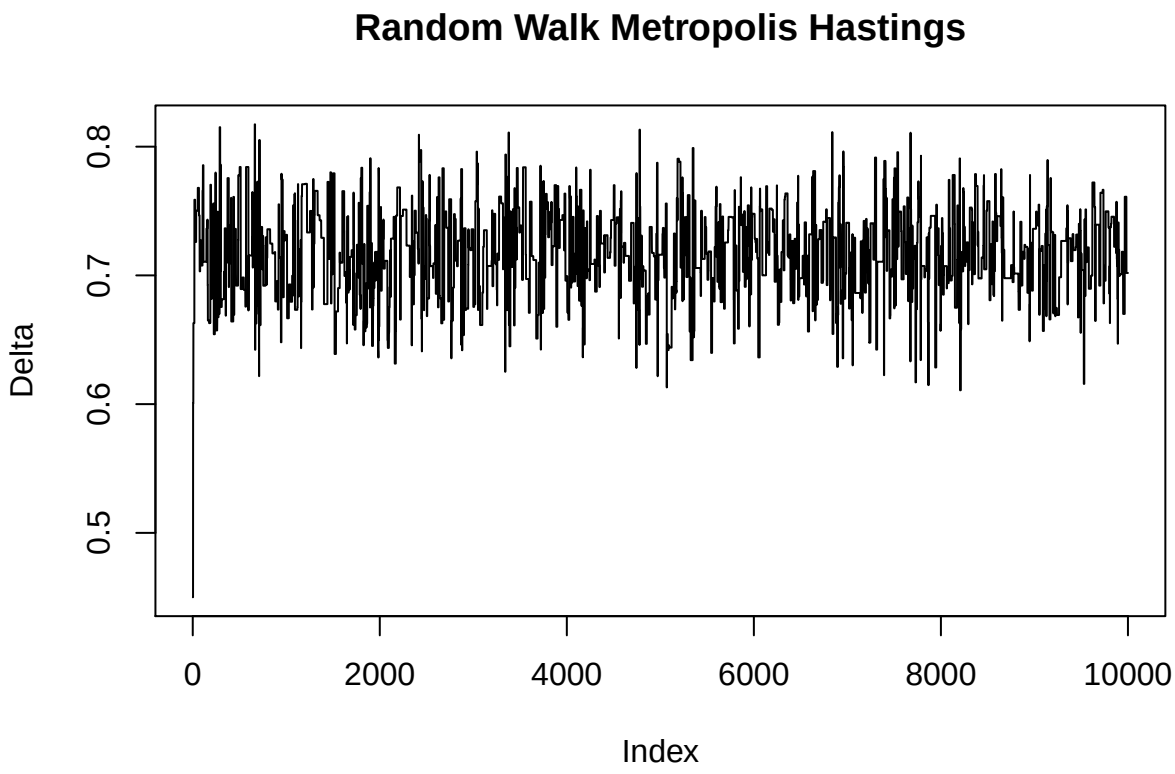

```
#There are some bars showing in the range [0,0.55].
#That may resulted from the first several times of calculations (burn-in part).
#This histgram shows that the area with larger density,
#the delta value will be more close to the target value.
```

12. Comment on the performance of the independence and random walk Metropolis Hastings samplers including at least one relevant plot.

```
#Trace plot for independent Metropolis Hastings:
plot.ts(trial,ylab="Delta",xlab="Index",main="Independence Metropolis Hastings")
```




```
plot.ts(RW,ylab="Delta",xlab="Index",main="Random Walk Metropolis Hastings")
```




```
#The independence and random walk Metropolis Hastings samplers gives the same results.
#They both can be adopted.
#As the proposal in independence MH samplers is Uniform distributed,
#the MH ratio equation in independence MH is same as that in random walk MH in reality.
#The first 1000(maybe) times of both MH is the burn-in part.
```