Assignment 3 - Simulation Studies, Integration and MCMCs

Name: Amine Natik
Date: March 15, 2018

1. Consider two groups of observations $X_1, \ldots, X_n \sim \mathcal{N}(\mu_1, \sigma^2)$ and $Y_1, \ldots, Y_m \sim \mathcal{N}(\mu_2, \sigma^2)$. We want to test the null hypothesis $\mathcal{H}_0 : \mu_1 = \mu_2$ versus $\mathcal{H}_1 : \mu_1 \neq \mu_2$. Put,

$$T = \frac{\left(\bar{X} - \bar{Y}\right)}{SE\left(\bar{X} - \bar{Y}\right)} \tag{1}$$

where $SE\left(\bar{X} - \bar{Y}\right) = S_p\sqrt{\frac{1}{n} + \frac{1}{m}}$. Under $\mathcal{H}_0$ the random variable T has a Student's t-distribution with $n + m - 2$ degree of freedom. Set the significance level to $\alpha = 0.05$ (type I error). Let C_α be the corresponding quantile, that is :

$$\alpha = P\left[|T| > C_\alpha \mid \mu_1 = \mu_2\right] \tag{2}$$

finally the power of the test is defined as,

$$\mathfrak{p} = P\left[|T| > C_\alpha \mid \mu_1 \neq \mu_2\right] \tag{3}$$

Note that the power depends on the difference $\delta = \mu_2 - \mu_1$, moreover if the variance σ^2 is known then we can calculate the power using the following formula:

$$\mathfrak{p}_\delta = P\left[T > C_\alpha - \frac{\delta}{\sigma\sqrt{\frac{1}{n} + \frac{1}{m}}}\right] + P\left[T < -C_\alpha - \frac{\delta}{\sigma\sqrt{\frac{1}{n} + \frac{1}{m}}}\right] \tag{4}$$

Where $T \sim t_{n+m-2}$, otherwise if σ^2 is unknown we replace σ by the pooled sample standard deviation S_p.

This test is valid only under the normality and equality of variances assumptions. The goal of this question is to show the effects of violation of these assumptions on the type 1 error rate α, and the power $\mathfrak{p}$. For simulation purpose we are going to assume that the total number of observations $N = n + m$ is fixed, moreover we assume that N is relatively small, because if n and m are big then by the CLT the test holds even if the underlying distributions are not normal. We are going to use $N = 12$ for the simulations. Define first the following initial values :

```
# Type I error
alpha=0.05
# Total number of observations
N=12
# degree of freedom
df=N-2
# quantile
quant=qt(1-alpha/2,df)
```

The following function calculate the pooled standard error of a given two-samples (x, y),

```
# Standard error
SE=function(x,y){
  n=length(x)
  m=length(y)
  Sx=sd(x)
  Sy=sd(y)
  Sp=sqrt(((n-1)*Sx^2+(m-1)*Sy^2)/(n+m-2))
  SE=Sp*sqrt(1/n+1/m)
  return(SE)
}
```

The function below uses the test to return 1 if we reject $\mathcal{H}_0$ and 0 otherwise.

```
# Two Sample t-test
TwoTest=function(x,y){
  SE=SE(x,y)
  T=(mean(x)-mean(y))/SE
  return(abs(T)>quant)
}
```

In following sections we are going to show the effects of the assumptions on the results, in most cases we can not calculate α and/or $\mathfrak{p}$ using an analytical formula, therefore the idea is to use simulation to get approximations of them.

For example if $\mathcal{H}_0$ is true that is $\mu_1 = \mu_2$, then $\alpha = P(|T| > C_\alpha)$ where T is the random variable in (1), hence the idea is as follows,

```
Choose a large number of simulations Numsim
Set Numrej=0
for $i = 1, \ldots, Numsim$ do
    1. Generate random samples $(x, y)$ of size $(n, m)$ from the desired distribution.
    2. if we reject $\mathcal{H}_0$ then
        Numrej=Numrej+1
    end
end
Put $\hat{\alpha}$ = Numrej/Numsim
```

the number $\hat{\alpha}$ = Numrej/Numsim is going to be used as an estimate of the type I error $\alpha = 0.05$. Define the relative error to be

$$\epsilon(\hat{\alpha}) = \frac{|\Delta\hat{\alpha}|}{\alpha} = \frac{|\hat{\alpha} - \alpha|}{\alpha}. \tag{5}$$

Define first a function that sample from this distribution : (note that this function simulate also from the normal distribution)

```
# the function gives an (n,m) sample from the distribution.
# the distribution can be normal or exp-exp.
sample=function(n,dist,mean1,mean2,sd1,sd2){
  if(dist=="normal"){
    x=rnorm(n,mean=mean1,sd=sd1)
    y=rnorm(N-n,mean=mean2,sd=sd2)
  }
  if(dist=="expnormal"){# we are going to define
                         # this distribution later.
    x=exp(exp(rnorm(n)))
    y=exp(exp(rnorm(N-n)))
  }
  return(list(x=x,y=y))
}
```

as explained before we approximate the portion of rejection using a large number of simulation. For this purpose use the function:

```
RejPortion=function(Numsim,n,dist,mean1,mean2,sd1,sd2){
  Numrej=0
  for(i in 1:Numsim){
    Sample=sample(n,dist,mean1,mean2,sd1,sd2)
    x=Sample$x
    y=Sample$y
    # if we reject H0 then increase Numrej:
    Numrej=Numrej+TwoTest(x,y)
  }
  return(Numrej/Numsim) #This is an approximation to rejected portion.
}
```

The function RejPortion returns a single random value which can approximate α if $\mu_1 = \mu_2$ and $\mathfrak{p}_\delta$ if $\mu_2 - \mu_1 = \delta$ (we will use this function to approximate α only). However we can not get too much information using one single value. We define a function that returns a vector of approximations of the rejected portion.

```
# A function that returns a random vector of rejected portions
# we use equal sample sizes by default
PortionVector=function(length=100,Numsim=100,n=6,
                       dist="normal",mean1=0,mean2=0,sd1=1,sd2=1){
  vec=rep(NA,length)
  for(i in 1:length){
    vec[i]=RejPortion(Numsim,n,dist,mean1,mean2,sd1,sd2)
  }
  return(vec)
}
```

If the underlying distributions are normal then the analytical formula in (4) holds, and will be helpful to draw the graph of $\mathfrak{p}$ as a function of δ.

```
#a function that returns the power
power=function(delta,sd1=1,sd2=1){
  SE=sqrt(sd1^2/n+sd2^2/(N-n)) #This term is explained below
  #set the power according to the analytical equation:
  pow=1-pt(quant,df,ncp=delta/SE)+pt(-quant,df,ncp=delta/SE)
  return(pow)
}
```

Finally define a function that plots the graph of $\mathfrak{p}$ on the interval $[-5, 5]$

```
#The function draws the graph of the power
graphpower=function(n=6,sd1=1,sd2=1){
  div=seq(from=-5,to=5,length.out=100)
  pow=rep(NA,100)
  for(i in 1:100){
    pow[i]=power(delta=div[i],sd1=sd1,sd2=sd2)
  }
  plot(div,pow,type="l")
}
```

After initializing all these functions in R let's study the robustness of the two sample t-test in the following sections. Starting with:

♣ The influence of the underlying distribution on type I error α.

In order to get interesting results we must simulate from a highly skewed distribution, for example we could use the *exp-exp distribution*, defined as follows if X is a normal random variable then $Y = e^{e^X}$ is an exp-exp distribution.

Let's see first the results for a standard normal distribution, with equal sample sizes that is $X_1, \dots, X_6 \sim \mathcal{N}(0,1)$ and $Y_1, \dots, Y_6 \sim \mathcal{N}(0,1)$.

```
#Take a vector of approximations
>normalvec=PortionVector(Numsim=1000)
># a bigger Numsim give much more precision.
># by default the function PortionVector takes a normal distribution.
>mean(normalvec)
 [1] 0.04962 # Very close to alpha
>relerr=abs(normalvec-alpha)/alpha #Calculate the relative error vector
>mean(relerr) # take the mean of the relative
              # error vector to have an estimation
 >[1] 0.112 # a reasonable relative error
```

If the underlying distributions are both standard normal then the approximation of the rejected portion matches with the significance level $\alpha = 0.05$, we get an approximation $\hat{\alpha}_{\text{normal}} \approx 0.496$ with a relative error $\epsilon(\hat{\alpha}_{\text{normal}}) \approx 0.1$ which is quite small. Now let's use the distribution exp-exp,

```
#Take a vector of approximations
>expvector=PortionVector(dist="expnormal",Numsim=1000)
>mean(expvector)
 [1] 0.00757 # Very far from alpha!
>relerr=abs(expvector-alpha)/alpha
>mean(relerr)
> [1] 0.8486 # the relative error is very large.
```

In this case, $\hat{\alpha}_{\text{exp}} \approx 0.007$ with a relative error $\epsilon(\hat{\alpha}_{\text{exp}}) \approx 0.8$, which shows the affect of the chosen distribution in the results! The Figure 1 contains the distributions of the $\hat{\alpha}$ in each case, in the normal case the graph is centered around $\alpha = 0.05$ with a small spread. However, in the exp-exp case the graph is centered around 0.007 which is very far from the true value of α.

```
>#Plot the two distributions.
>plot(density(normalvec),main="Normal␣distribution",
      xlab=TeX('$\\alpha$'),ylab="")
>plot(density(expvector),main="exp-exp␣distribution",
      xlab=TeX('$\\alpha$'),ylab="")
```

♣ The influence of σ on type I error α.

Assume now that the normality assumption holds but the two groups of observations does not have the same variance that is, $X_1, \dots, X_n \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $Y_1, \dots, Y_m \sim \mathcal{N}(\mu_2, \sigma_2^2)$ with $\sigma_1 \neq \sigma_2$. How does this affect the type I error α? To answer this question, we are going to follow the same approach used in the last section. For simulation assume that $X_1, \dots, X_6 \sim N(0,1)$ but $Y_1, \dots, Y_6 \sim \mathcal{N}(0, k^2)$ where $k \in \{1, 2, \dots, 10\}$. The Figure 2 shows as plot of the error of $\hat{\alpha}$ as a function of k. We can see that if $k = 1$ then the relative error is very small which is normal since both groups have the same mean and variance, but as the variance of the Y group grows, the relative error grows as well. We can conclude that the non-equality of the variance have a huge influence on the type I error.

```
>error=rep(NA,10) #empty vector that will contain the relative errors
>sd=1:10 # different values of the standard deviation
>for (i in 1:10){
  #create a vector vec of approximations of alpha,
  #the first group X is from a standard normal
  #the second group Y is from a normal with mean 0
  #and standard deviation k
  vec=PortionVector(sd2=i,Numsim=1000)
  #calculate the mean of relative errors of vec
  error[i]=mean(abs(vec-alpha)/alpha)
}
#Then plot the relative errors obtained
>plot(sd,error,type="l",main="",
      xlab=TeX('$\\k$'),ylab=TeX('$\\epsilon(\\alpha)$'))
```

♣ **The influence of σ on the power $\mathfrak{p}$.**

Note that if $X_1,\dots,X_n \sim \mathcal{N}(\mu_1,\sigma_1^2)$ and $Y_1,\dots,Y_m \sim \mathcal{N}(\mu_2,\sigma_2^2)$ with $\sigma_1 \neq \sigma_2$, and if σ_1 and σ_2 are known then the test in equation (1) is going to change, we instead set:

$$T = \frac{\left(\bar{X}-\bar{Y}\right)-(\mu_1-\mu_2)}{\sqrt{\frac{\sigma_1^2}{n}+\frac{\sigma_2^2}{m}}} \tag{6}$$

The denominator in (6) explains the term in the first line of the body of the function power defined on R previously, this function will be useful for simulation in this section.

As in the previous section, let $X_1,\dots,X_6 \sim \mathcal{N}(0,1)$ and $Y_1,\dots,Y_6 \sim \mathcal{N}(0,k^2)$ where in this case $k \in \{1,2,\dots,5\}$. In the Figure 3 we plot the power as a function of δ on the interval $[-5,5]$ in different cases $k=1,\dots,5$ (using the function graphpower). We notice that as long as k is large, the power become smaller. By a symmetric argument we can conclude that as long as $\sigma_1/\sigma_2 \neq 1$ the power function $\mathfrak{p}$ decreases. We could summarize this in the following way:

$$\mathfrak{p}^{(\sigma_1,\sigma_1)} \geq \max_{\sigma_2\neq\sigma_1} \mathfrak{p}^{(\sigma_2,\sigma_1)} \tag{7}$$

The non-equality of the variances can violate the results, since the power decreases.

♣ **The influence of the sample sizes n and m on the power $\mathfrak{p}$.** Assume that all the assumptions hold, more specifically let $X_1,\dots,X_n \sim \mathcal{N}(\mu_1,\sigma^2)$ and $Y_1,\dots,Y_m \sim \mathcal{N}(\mu_2,\sigma^2)$ and the total number of observations $N=n+m$ is fixed. In this case the formula in (4) holds, and we can see that the power is large as long as the term $\frac{\delta}{\sigma\sqrt{\frac{1}{n}+\frac{1}{m}}}$ is large, where $\delta=\mu_2-\mu_1$ is fixed, then $\sqrt{\frac{1}{n}+\frac{1}{m}}$ should be as small as possible to get a larger power, which is possible if $n=m=\frac{N}{2}$. Therefore, if $n \neq m$ we would expect a smaller power. The Figure 4 gives the graphs of the power for different values of $n=1,\dots,6$. Note that using symmetry between n and m the graphs for $n=7,\dots,11$ are the same as for $n=5,\dots,1$ respectively. We can see that as long as n is far from 6 the power becomes smaller. We conclude that :

$$\mathfrak{p}^{(6)} \geq \max_{k\neq 6} \mathfrak{p}^{(k)} \tag{8}$$

that is when we have equal sample sizes $n=m=6$, we get the highest power.

2. Let $X \sim \mathcal{U}(1,a)$ where $a > 1$, consider $Y = f(X) = \frac{a-1}{X}$.

   (a) we know that the distribution of X is given by

$$\phi(u) = \frac{1}{a-1}\mathbb{I}_{(1,a)}(u) \tag{9}$$

   thus

$$E\left[f(X)\right] = \int_1^a \frac{a-1}{x}\frac{1}{a-1}dx = \ln(x)\Big|_{x=1}^{x=a} = \ln(a) \tag{10}$$

   (b) We define first in R the function $x \mapsto \frac{1}{x}$

```
func=function(x){1/x}
```

   Then calculate approximations of $E[Y] = E[f(X)]$ using the function integrate,

```
a=c(1.1, 1.5, 2.5, 4) # given values of a
approx=c() #vector of approximations
error=c() #vector of errors
for(i in 1:length(a)){#for each value of a
# the function integrate returns the value and the error.
  approx[i]=integrate(func,lower=1,upper=a[i])$value
  error[i]=integrate(func,lower=1,upper=a[i])$abs.error
}

>approx
[1] 0.09531018 0.40546511 0.91629073 1.38629436
>error
[1] 1.058156e-15 4.501567e-15 1.017287e-14 5.813999e-11
#The true values using part (a)
> log(a)
[1] 0.09531018 0.40546511 0.91629073 1.38629436
```

   We can summarize the results in the Table 1

a	The true value	The approximation	The error
1.1	0.09531018	0.09531018	$1.058156 \cdot 10^{-15}$
1.5	0.40546511	0.40546511	$4.501567 \cdot 10^{-15}$
2.5	0.91629073	0.91629073	$1.017287 \cdot 10^{-14}$
4	1.38629436	1.38629436	$5.813999 \cdot 10^{-11}$

Table 1: The true values of $E[Y]$, the approximation and the errors for each value of a.

   (c) In order to estimate $E[Y]$ we could use methods of approximating integrals such as the Trapezoidal rule and the Simpson rule. Since we are integrating from 1 to a, we consider $\{x_0, \dots, x_n\}$ a regular subdivision of $n+1$ points of the interval $[1,a]$, that is $x_k = (a-1)k/n + 1$ for all $k = 0, 1, \dots, n$.
   ♣ **Trapeziodal rule.** We can approximate $E[Y]$ using the following formula,

$$E[Y] = \int_1^a \frac{1}{x}dx \simeq \sum_{i=0}^{n-1}\frac{1}{2}(x_{i+1} - x_i)\left(g(x_i) + g(x_{i+1})\right) \tag{11}$$

   Therefore the function in R is defined as follows

```
trapezoid=function(n,a){
  sub=seq(from=1,to=a,length.out=n)#the subdivision
  approx=0 #approx will approximate the expected value
  for(i in 1:(n-1)){
    #approx is calculated according to equation (11)
    approx=approx+(sub[i+1]-sub[i])*(func(sub[i+1])+func(sub[i]))/2
  }
  return(approx)
}
```

♣ **Simpson's rule.** We can approximate $E[Y]$ using the following formula,

$$E[Y] = \int_1^a \frac{1}{x}dx \simeq \sum_{i=0}^{n-1} \frac{1}{6}(x_{i+1} - x_i)\left(g(x_i) + 4g\left(\frac{x_i + x_{i+1}}{2}\right) + g(x_{i+1})\right) \tag{12}$$

Therefore the function in R is defined as follows

```
simpson=function(n,a){
  sub=seq(from=1,to=a,length.out=n)#the subdivision
  approx=0 #approx will approximate the expected value
  for(i in 1:(n-1)){
    #approx is calculated according to equation (12)
    approx=approx+(sub[i+1]-sub[i])*(func(sub[i])
    +4*func((sub[i]+sub[i+1])/2)+func(sub[i+1]))/6
  }
  return(approx)
}
```

♣ **Simulations.** We are going to run both functions for different values of n, and for the same values of a as part (b). We choose $n = 100,\ 1000$ and 10000 for simulation. We define a function Simulation(Nvec,method) that takes a vector Nvec of values of n and the used method (Trapezoid or Simpson) and return the corresponding approximations $E[Y]$ and absolute errors.

```
simulation=function(Nvec,method){
  #define empty matrices to contain
  #the approximations and errors.
  approx=matrix(nrow=length(Nvec),ncol=length(a))
  error=matrix(nrow=length(Nvec),ncol=length(a))
  if(method=="trapezoid"){
    for(i in 1:length(Nvec)){
      for(j in 1:length(a)){
        approx[i,j]=trapezoid(Nvec[i],a[j])
        error[i,j]=abs(approx[i,j]-log(a[j]))
      }
    }
  }
  if(method=="simpson"){
    for(i in 1:length(Nvec)){
      for(j in 1:length(a)){
        approx[i,j]=simpson(Nvec[i],a[j])
        error[i,j]=abs(approx[i,j]-log(a[j]))
      }
    }
  }
  return(list(approx=approx, error=error))
}
```

And now we run the function Simulation for both methods,

```
> a=c(1.1,  1.5,  2.5,  4)
> Nvec=c(100,1000,10000)
> simulation(Nvec,"trapezoid")
$approx
          [,1]      [,2]      [,3]     [,4]
[1,] 0.09531019 0.4054663 0.9163068 1.386366
[2,] 0.09531018 0.4054651 0.9162909 1.386295
[3,] 0.09531018 0.4054651 0.9162907 1.386294

$error
             [,1]         [,2]         [,3]         [,4]
[1,] 1.475646e-08 1.180903e-06 1.606936e-05 7.173313e-05
[2,] 1.449178e-10 1.159726e-08 1.578154e-07 7.045327e-07
[3,] 1.445871e-12 1.157645e-10 1.575313e-09 7.032651e-09

> simulation(Nvec,"simpson")
$approx
          [,1]      [,2]      [,3]     [,4]
[1,] 0.09531018 0.4054651 0.9162907 1.386294
[2,] 0.09531018 0.4054651 0.9162907 1.386294
[3,] 0.09531018 0.4054651 0.9162907 1.386294

$error
             [,1]         [,2]         [,3]         [,4]
[1,] 6.938894e-16 1.087908e-12 1.069693e-10 1.748901e-09
[2,] 6.938894e-17 4.440892e-16 1.021405e-14 1.689759e-13
[3,] 3.469447e-16 1.165734e-15 1.110223e-15 1.554312e-15
```

We can summarize these results in the Table 2

			Trapezoid		Simpson	
n	a	True Value	approximation	error	approximation	error
100	1.1	0.09531018	0.09531019	$1.475646 \cdot 10^{-8}$	0.09531018	$6.938894 \cdot 10^{-16}$
	1.5	0.40546511	0.4054663	$1.180903 \cdot 10^{-6}$	0.4054651	$1.087908 \cdot 10^{-12}$
	2.5	0.91629073	0.9163068	$1.606936 \cdot 10^{-5}$	0.9162907	$1.069693 \cdot 10^{-10}$
	4	1.38629436	1.386366	$7.173313 \cdot 10^{-5}$	1.386294	$1.748901 \cdot 10^{-9}$
1000	1.1	0.09531018	0.09531018	$1.449178 \cdot 10^{-10}$	0.09531018	$6.938894 \cdot 10^{-17}$
	1.5	0.40546511	0.4054651	$1.159726 \cdot 10^{-8}$	0.4054651	$4.440892 \cdot 10^{-16}$
	2.5	0.91629073	0.9162909	$1.578154 \cdot 10^{-7}$	0.9162907	$1.021405 \cdot 10^{-14}$
	4	1.38629436	1.386295	$7.045327 \cdot 10^{-7}$	1.386294	$1.689759 \cdot 10^{-13}$
10000	1.1	0.09531018	0.09531018	$1.445871 \cdot 10^{-12}$	0.09531018	$3.469447 \cdot 10^{-16}$
	1.5	0.40546511	0.4054651	$1.157645 \cdot 10^{-10}$	0.4054651	$1.165734 \cdot 10^{-15}$
	2.5	0.91629073	0.9162907	$1.575313 \cdot 10^{-9}$	0.9162907	$1.110223 \cdot 10^{-15}$
	4	1.38629436	1.386294	$7.032651 \cdot 10^{-9}$	1.386294	$1.554312 \cdot 10^{-15}$

Table 2: Simulations using Trapezoid and Simpson's methods

♣ **Comparison of the two methods.** Clearly the errors for the Simpson's method are much smaller than those of the Trapezoid method (see Table 2) which means that the Simpson's method is more precise. On the other hand the running times for both methods are :

```
> system.time(simulation(Nvec,"trapezoid"))
   user  system elapsed
  0.054   0.000   0.055
> system.time(simulation(Nvec,"simpson"))
   user  system elapsed
  0.070   0.000   0.071
```

Which means that the Trapezoid method is much faster than Simpson's method.

3. In this question we are interested in $\rho = E\left[X^2\right]$ where the random variable X has density f such that $f(x) \propto g(x) = \exp\left(-\frac{|x|^3}{3}\right)$, let C be the normalizing constant, therefore :

$$f(x) = Cg(x) \tag{13}$$

We choose as a proposal distribution the standard normal for both questions. Set,

$$\pi(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) \tag{14}$$

Note that both f and π have the same support $\mathbb{R} = (-\infty, \infty)$

(a) Note that the rejection sampling can be used even if we do not know the constant C (see e.g. question 4 on the midterm). In fact,

$$\frac{f(x)}{\pi(x)} \leq M \Leftrightarrow \frac{g(x)}{\pi(x)} \leq M' = \frac{M}{C} \tag{15}$$

and we are able to find an explicit optimal constant M' that satisfies the last inequality, take

$$M' = \sup_{x\in\mathbb{R}} \frac{g(x)}{\pi(x)} = \sup_{x\in\mathbb{R}} \left(\sqrt{2\pi} \exp\left(\frac{x^2}{2} - \frac{|x|^3}{3}\right)\right) = \sqrt{2\pi e^{\frac{1}{3}}} \tag{16}$$

therefore $g(x)/\pi(x) \leq \sqrt{2\pi e^{\frac{1}{3}}}$ for all $x \in \mathbb{R}$ with equality if and only if $x = 1$ or $x = -1$. In order to use the accept-reject sampling, we have to generate a random uniform value $u \sim \mathcal{U}[0,1]$ and $x \sim \mathcal{N}(0,1)$, we accept x if

$$u \leq \frac{g(x)}{M'\pi(x)} = \exp\left(\frac{x^2}{2} - \frac{|x|^3}{3} - \frac{1}{6}\right) \tag{17}$$

Using this procedure we propose a function to sample from the distribution f :

```
rejSample=function(n){#The function returns an n-sample from f
  v=c()
  for(i in 1:n){
    accept=F
    while(accept==F){
      u=runif(1)
      x=rnorm(1)
      if(u<=exp(x^2/2-abs(x^3)/3-1/6)){#The test in equation (17)
        accept=T
        v=c(v,x)
      }
    }
  }
  return (v)
}
```

Since we can get a random sample from the distribution f, we could use the naive Monte Carlo approach to estimate ρ. The idea is to generate $X_1, \ldots, X_n \sim f$ for n very large using rejection sampling and then estimate ρ by

$$\hat{\rho} = \frac{1}{n}\sum_{i=1}^{n} X_i^2 \tag{18}$$

```
>n=1000
>sample=rejSample(n)
>rhohat=mean(sample^2)#according to equation (18)
>rhohat
 >[1] 0.7689561
```

Therefore $E\left[X^2\right] \approx 0.7$

4. As said before, we are going to use the standard normal distribution as proposal distribution. Define the function w to be the weight function, $w(x) = \frac{g(x)}{\pi(x)}$. Let $(x_1, \ldots, x_n)$ be a random sample of observations drawn from the standard normal distribution, then we have :

$$\begin{aligned}\rho &= \int_{-\infty}^{\infty} x^2 f(x)dx \\ &= C\int_{-\infty}^{\infty} x^2 g(x)dx \\ &= C\int_{-\infty}^{\infty} x^2 w(x)\pi(x)dx \\ &\simeq \frac{C}{n}\sum_{i=1}^{n} x_i^2 w(x_i)\end{aligned}$$

The normalizing constant can readily be obtained,

$$\begin{aligned}\frac{1}{C} &= \int_{-\infty}^{\infty} g(x)dx \\ &= \int_{-\infty}^{\infty} w(x)\pi(x)dx \\ &\simeq \frac{1}{n}\sum_{i=1}^{n} w(x_i)\end{aligned}$$

combining this two equations we finally obtain

$$\rho = E\left[X^2\right] \simeq \frac{\sum_{i=1}^{n} x_i^2 w(x_i)}{\sum_{i=1}^{n} w(x_i)} \tag{19}$$

therefore, the idea is to simulate a random sample $(x_1, \ldots, x_n)$ from $\mathcal{N}(0,1)$ and calculate an estimation of ρ according to (19).

first define a function to calculate the weights

```
weight=function(x){
  return(sqrt(2*pi)*exp(x^2/2-abs(x^3)/3))
}
```

Then define a function that returns an approximation using equation (19)

```
ImpSampling=function(n){
  x=rnorm(n)
  w=weight(x)
  approx=sum(x^2*w)/sum(w) # according to equation (19)
  return(approx)
}
```

Let's try this function for $n = 1000$,

```
> n=1000
> ImpSampling(n)
 [1] 0.7672867
```

Hence $E\left[X^2\right] \approx 0.7$ note that the result in part (b) matches with part (a).

5. (a) ok
   (b) Using the given summary, the idea of the Gibbs sampling algorithm is as follows,

```
Set initial points $\left(\theta^{(0)}, \tau^{(0)}\right)$
for $i = 1, 2, \ldots, n$ do
    1. Sample $\theta^{(i)} \sim f\left(\theta \mid \tau^{(i-1)}, \boldsymbol{T}, \delta\right) = \mathcal{G}\text{amma}\left(a + \sum_{i=1}^{n} \delta_i + 1, c + d\tau^{(i-1)} + \tau^{(i-1)} \sum_{i\in\mathcal{H}} t_i + \sum_{i\in\mathcal{C}} t_i\right)$
    2. Sample $\tau^{(i)} \sim f\left(\tau \mid \theta^{(i)}, \boldsymbol{T}, \delta\right) = \mathcal{G}\text{amma}\left(b + \sum_{i\in\mathcal{H}} \delta_i + 1, d\theta^{(i)} + \theta^{(i)} \sum_{i\in\mathcal{H}} t_i\right)$
end
The procedure will produce an $n$-sample of $(\theta, \tau)$.
```

We start by reading data in R and defining some variables,

```
#read data
data=read.table("Brreastcancer.txt",sep="",
                fill=FALSE, strip.white=TRUE)

time=data[,1]
treatment=data[,2]
censored=data[,3]
#since the censored given data is different from the summary
#we switch the ones and the zeros by:
delta=1-censored
#Nh is the cardinal of the treatment group
Nh=sum(treatment)
#N is the total number of patients
N=length(treatment)
#Define the given parameters
a=3
b=1
c=60
d=120
#The following sums will be used to simplify
# the expressions of the conditional distributions.
TimeSumH=sum(time[1:Nh])
TimeSumC=sum(time[(Nh+1):N])
```

```
DeltaSum=sum(delta)
DeltaSumH=sum(delta[1:Nh])
```

Now we define our function :

```
#n will be the desired sample size
#initial is a vector that contain the starting point.
Gibbs=function(n,initial){
  #the matrix gibbs contains two colums,
  #the column 1 will contain a random sample from theta
  #while column 2 will contain a random sample from tau
  gibbs=matrix(nrow=n,ncol=2)
  gibbs[1,]=initial #set initial point
  for(i in 2:n){
    #sample from the full conditional for theta
    gibbs[i,1]=rgamma(n=1,shape=a+DeltaSum+1,
                rate=c+gibbs[(i-1),2]*(d+TimeSumH)+TimeSumC)
    #sample from the full conditional for tau
    gibbs[i,2]=rgamma(n=1,shape=b+DeltaSumH+1,
                rate=(d+TimeSumH)*gibbs[i,1])
  }
  require(coda)
  gibbs=mcmc(gibbs)
  return(gibbs)
}
```

(c) To evaluate convergence and mixing of the Gibbs sampler, we use the package coda in R, we run the function Gibbs for different initial points,

```
>n=1000 #sample size
>results=list(NULL) #empty list
>results[[1]]=Gibbs(n,initial=c(0.2,0.2)) #first initial point
>results[[2]]=Gibbs(n,initial=c(5,10)) #second initial point
>results[[3]]=Gibbs(n,initial=c(0.5,2)) #third initial point
>results=mcmc.list(results) #transform as an mcmc chain
>plot(results) (see Figure 5)
>gelman.plot(results) (see Figure 6)
```

We can conclude from the two Figures 5 and 6 that the Markov chain behave in a good way, for all the initial points the plots looks very similar, and there is no reason to say that sampler does not converge or the generated Markov chain is mixing.

(d) In order to get estimations of $E[\theta]$, $E[\tau]$, $Var[\theta]$ and $Var[\tau]$ we run the function Gibbs for a large value of n then calculate the means and variances returned samples.

```
> n=10000
> initial=c(0.2,0.2)
> Sample=Gibbs(n,initial)
> mean(Sample[,1])
[1] 0.009226887
> var(Sample[,1])
[1] 1.069223e-05
> mean(Sample[,2])
[1] 1.22346
> var(Sample[,2])
[1] 0.2350367
```

Therefore $E\left[\theta\right] \approx 0.009$, $E\left[\tau\right] \approx 1.2$, $Var\left[\theta\right] \approx 1.06 \cdot 10^{-5}$ and $Var\left[\tau\right] \approx 0.2$

6. The recurrence time T_i^* of breast cancer follows an exponential distribution with mean $1/\tau\theta$ and $1/\theta$ for the hormone and control group respectively, using our previous study a reasonable estimate of τ is its expected value $\tau \approx E[\tau] \approx 1.2$, we might assume that $\tau > 1$ therefore $1/\tau\theta < 1/\theta$. That is, the mean for the treated hormone group is less than the mean of the control group. We would advice the drug company to assign most of women to the control group since the time to recurrence will be much bigger.

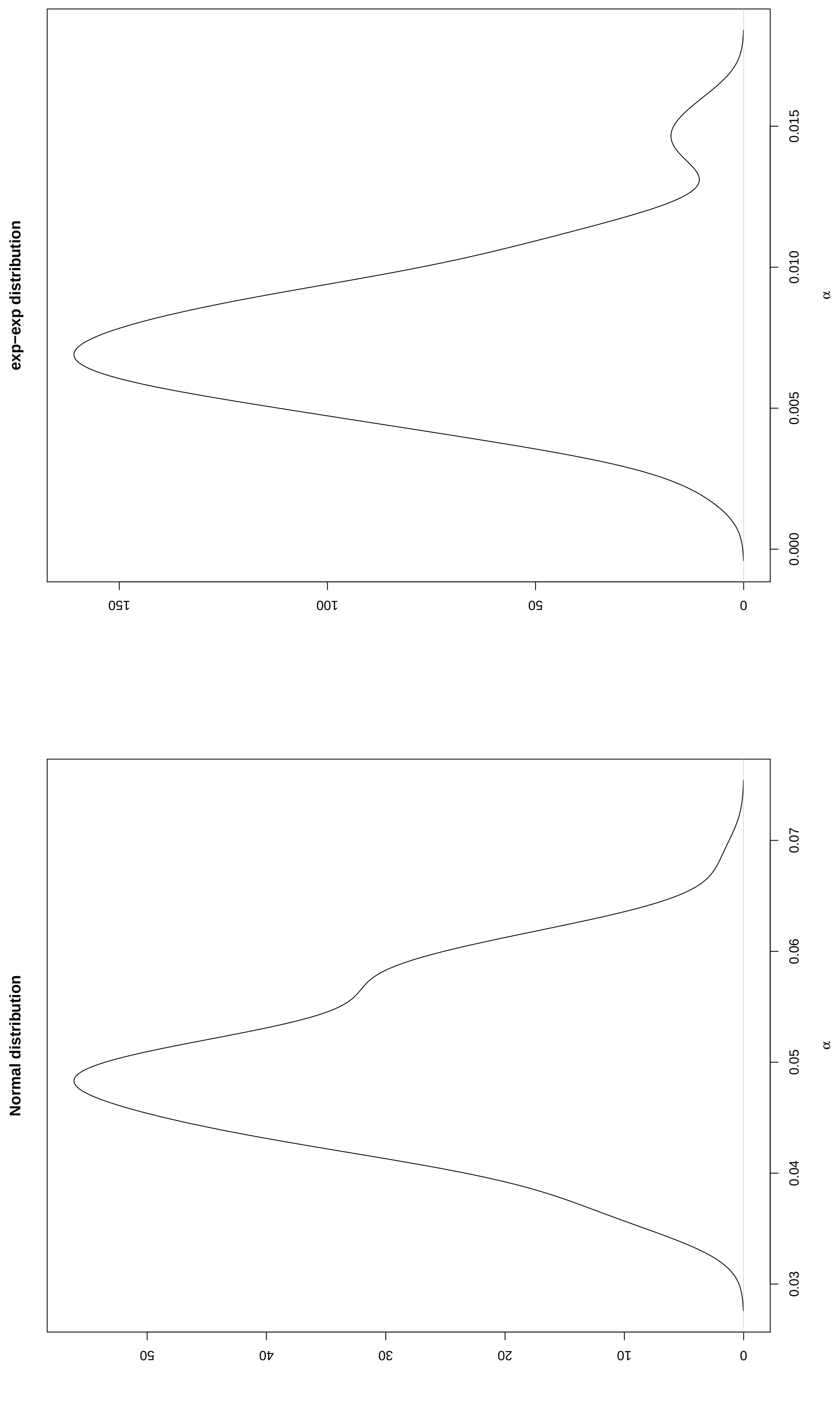

Figure 1: Distributions of $\hat{\alpha}_{\text{normal}}$ and $\hat{\alpha}_{\text{exp}}$

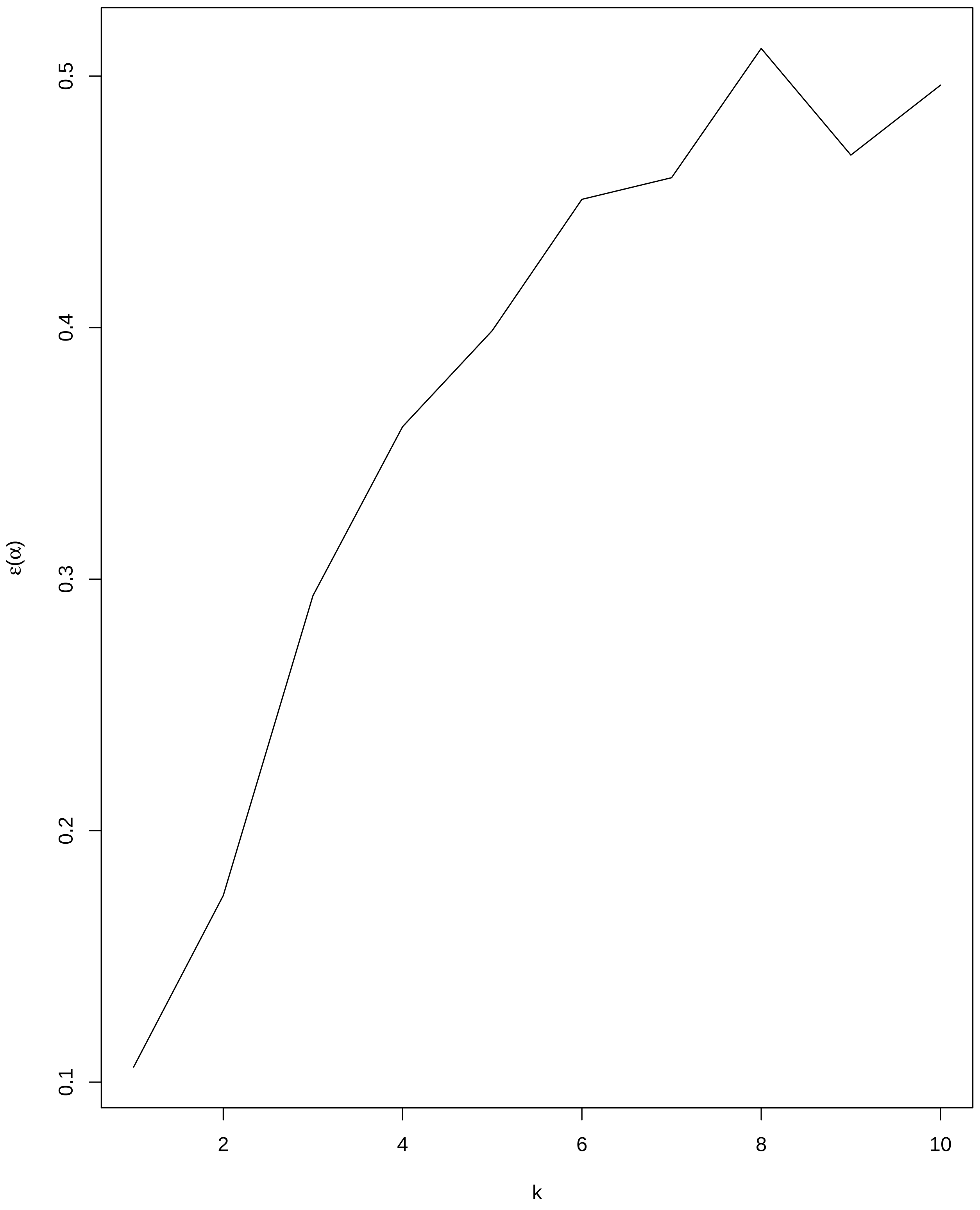

Figure 2: The relative error of type I error as a function of k

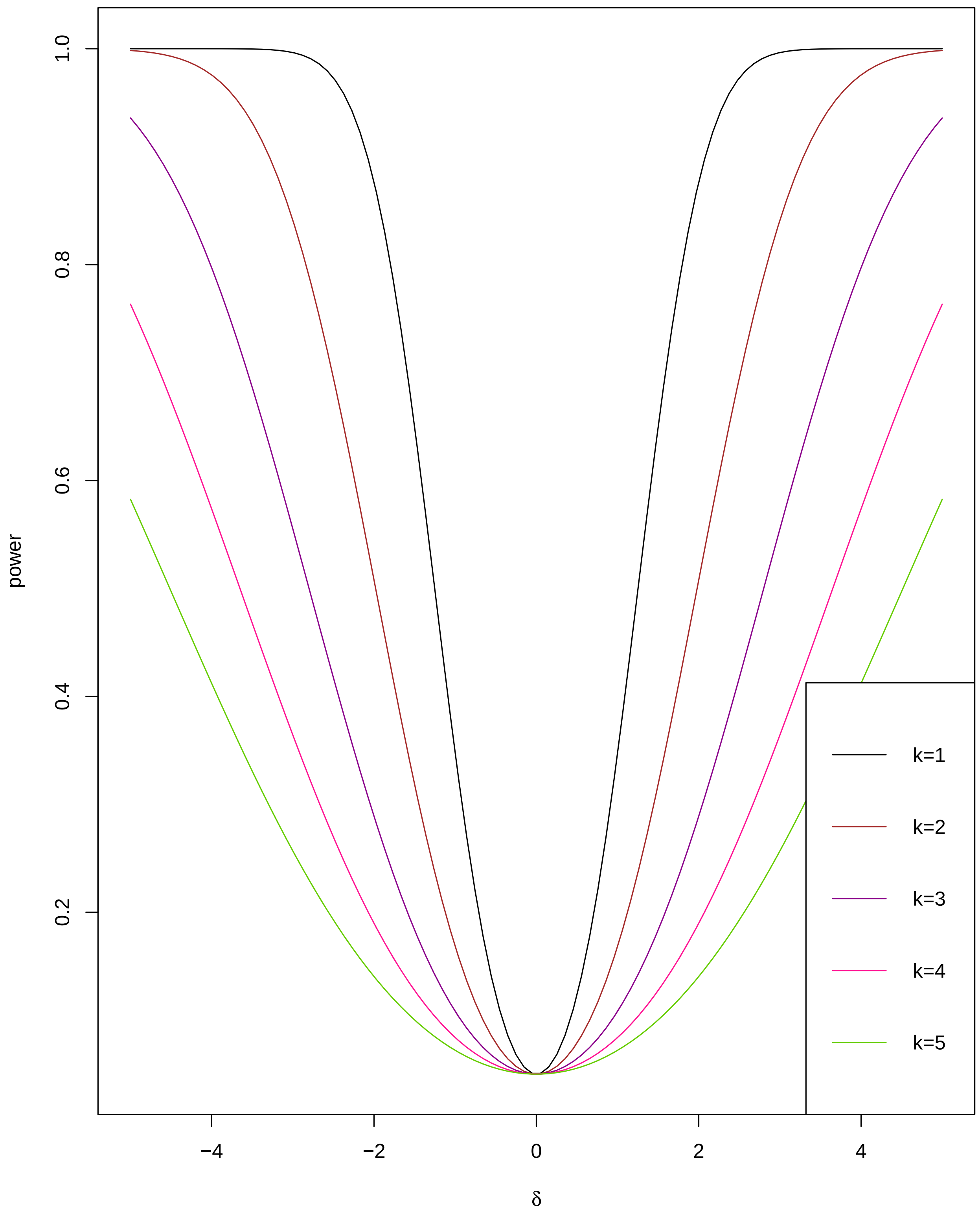

Figure 3: Different values of the standard deviation of the group Y.

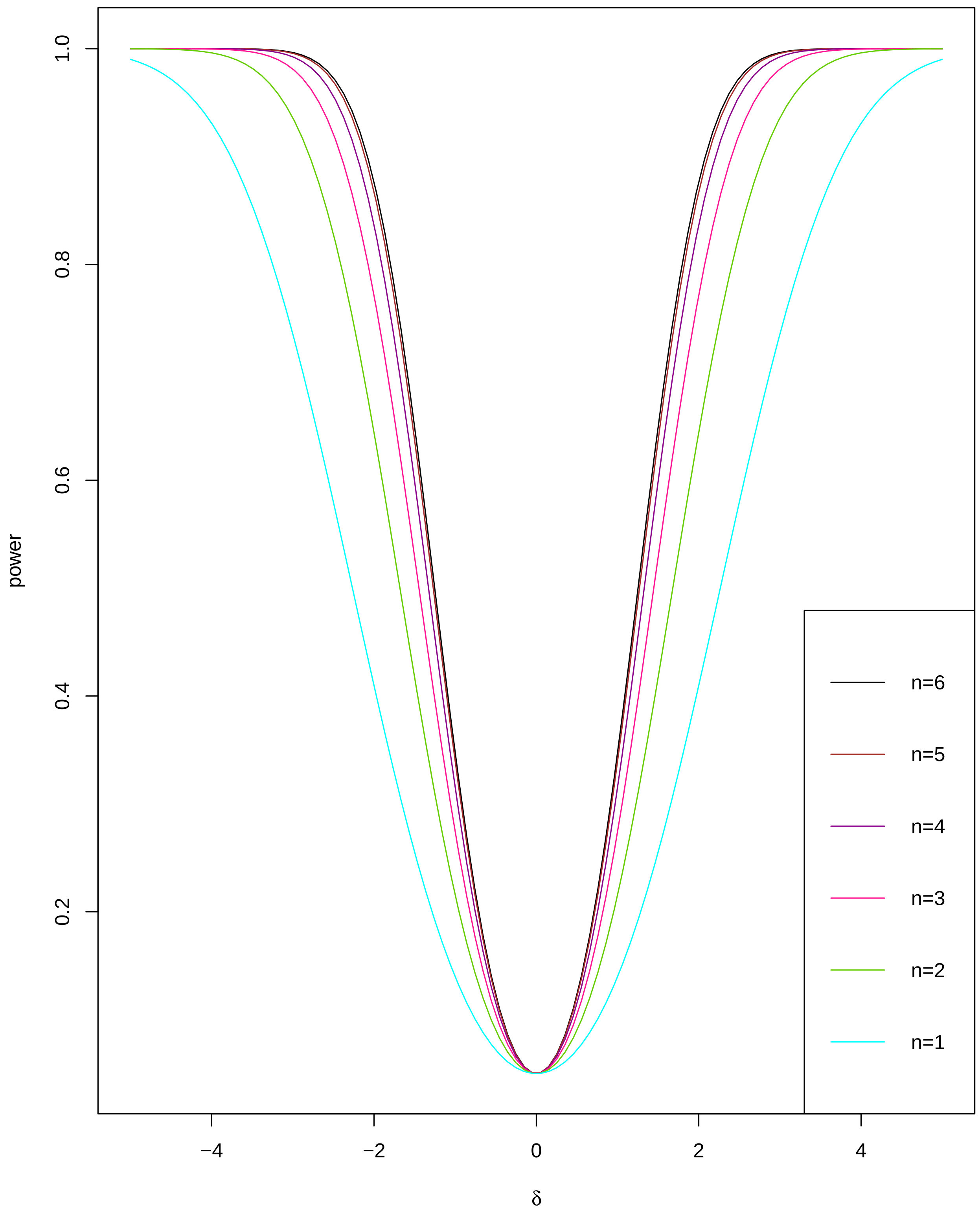

Figure 4: Different values of the sample size n

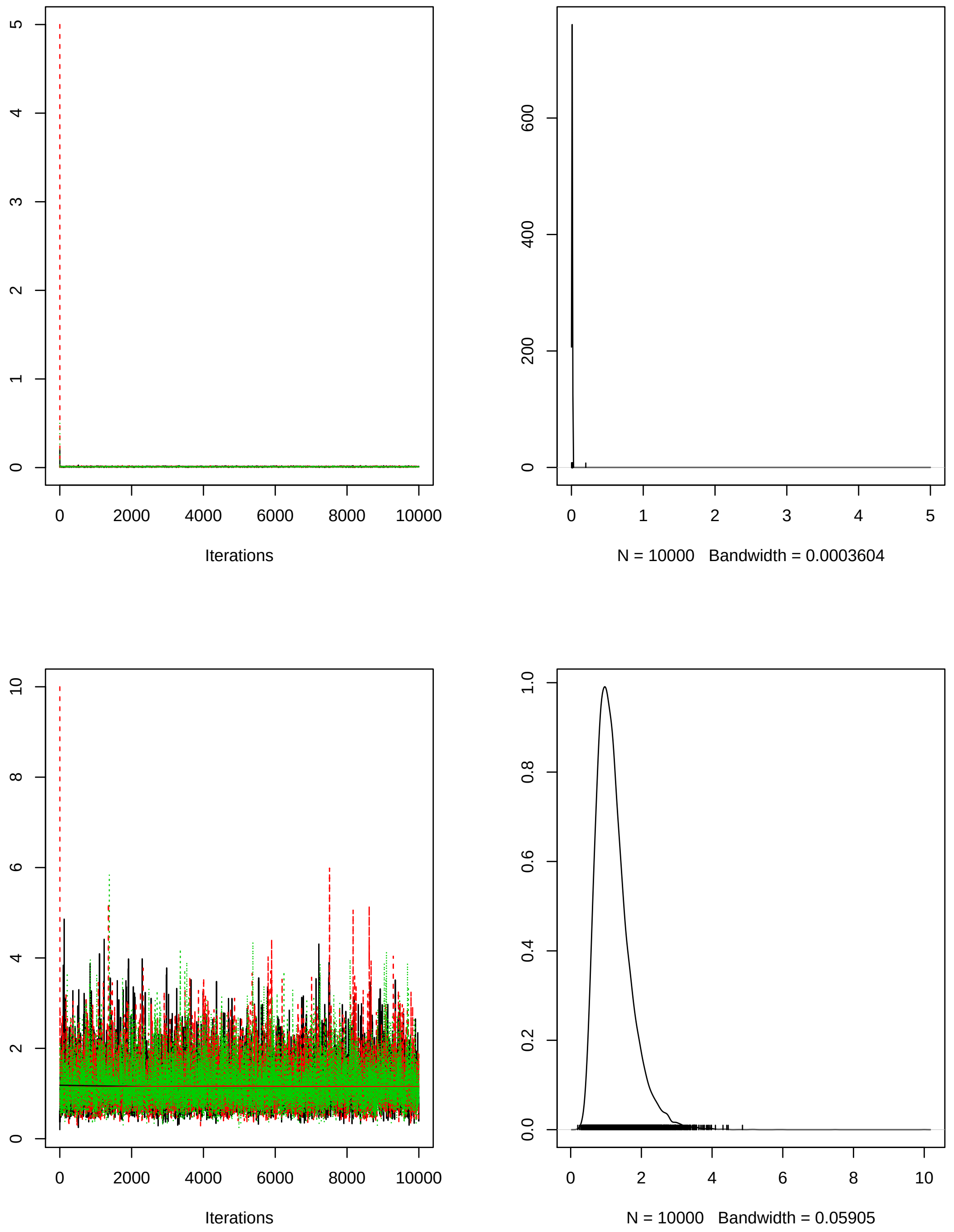

Figure 5: Gibbs sampler diagnostics for three different initial points

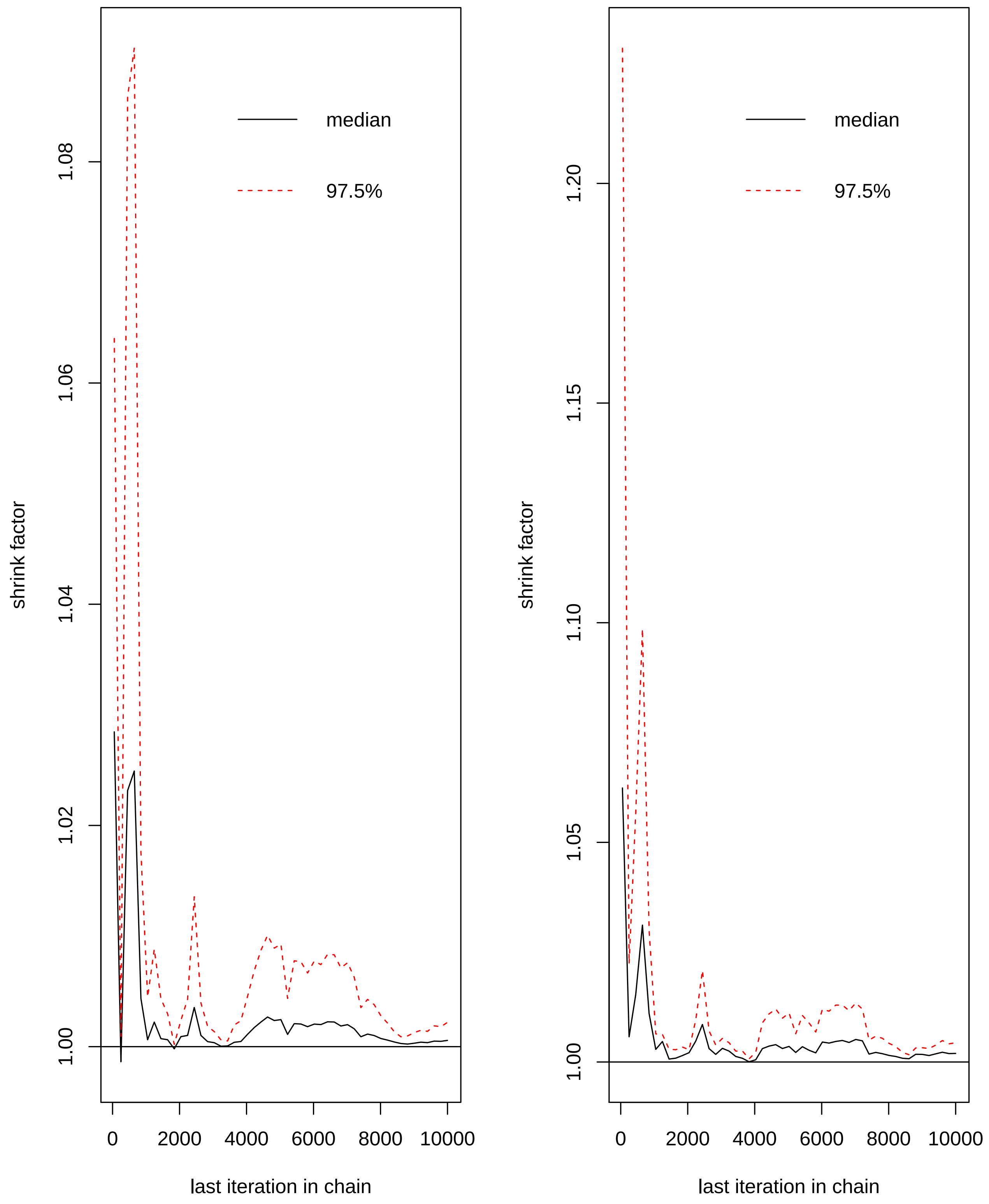

Figure 6: Gelman and Rubin diagnostic