# Homework 1 in Computer Intensive Statistics & Data Mining

*Alari Varmann*

## Theoretical Background

### Null-hypothesis and P-value

When there is a lack of difference between two samples, we refer to this a the null hypothesis, we often want to analyze the difference between samples to determine if the case of study has a true effect over a population or data. It is interesting to note that the null hypothesis is never accepted or proved. Wikipedia: "Rejecting or disproving the null hypothesis—and thus concluding that there are grounds for believing that there is a relationship between two phenomena (e.g. that a potential treatment has a measurable effect)—is a central task in the modern practice of science, and gives a precise sense in which a claim can be proven false." \ \ The p-value gives you the probability of how likely your data holds the null hypothesis or not. This is calculated by assuming that the null-hypothesis is true, the difference in the sample is due to random chance. If a p-value is low the interpretation is that the sample is unlikely to accept the null hypothesis but there can be the case where the null hypothesis is true and the data handled was unusual in some sense.

### Chi-squared test

The Chi-squared test is often used to measure the relation between two features of the same population. This method is appropriate when simple random sampling is used as the sampling method. The formula for the method is:

$$\chi^2 = \sum \frac{(O-E)^2}{E}$$

E= Expected values, O= Observed values.

### Kolmogorov-Smirnov test

When you want to compare densities of one dimensional, continuous probability distributions, in the case of a one sample K-S test, the comparison is made with a reference probability distribution, in the case of a two sample K-S, you compare the difference between the two samples. It is considered to be one of the best and most useful non-parametric methods of comparing the shape and location of two distribution functions.

### Congruential Generator Algorithm

A mixed congruential generator or a linear congruential generator is a pseudorandom number generator from a uniform distribution, it involves three parameters a,b,M and a seed variable. If the parameters are selected correctly this method produces a sequences of integers including all between 0 and M-1. When divided by M we obtain the random generation of numbers between 0 and 1. In particular, well suited valued for a and b should be inside ${0,1…,M-1 } $ and by choosing a floating point number as our seed, the sequence of numbers are not restricted to integers.

## Rejection Sampling

The Rejection sampling is a method to generate samples for a specific density function by utilizing an instrumental probability density function, from which we generate the sample candidates that are kept if they fall within our density of interest. The idea is that with this method we can sample from the area under any curve.

## Monte Carlo Methods

Monte Carlo methods are a very powerful tool to approximate the values of complicated integrals using probabilistic techniques. We are interested in finding the value of

$$\mu = \int f(x)dx < \infty$$

We first have to find a suitable factorization of $f(x) = h(x)p(x)$ where $p(x)$ is a known probability density function. We can see that $\mu$ is interpreted as an Expected value.

$$Eh(x) = \int p(x)h(x)dx$$

After generating a sample $x_1, x_2, \ldots, x_n$ from the density function we calculate

$$\tilde{\mu} = \frac{1}{n}\sum_{i}^{n} h(x_i)$$

From $\sigma^2 = Var(h(x)) = \int (h(x)-\mu)^2 p(x)dx < \infty$ The law of large numbers and the center limit theorem we have that $\tilde{\mu} \to \mu$.

## Hit or miss Monte Carlo

In this method random samples are produced from $U \sim$ U(0, 1) and also from $V \sim$ V(0, 1) to later make the estimation following:

$$\tilde{\mu} = \frac{\#\{i : cV_i < f(a + U_i(b-a))\}}{n}$$

## Importance sampling

The idea of importance sampling is precisely to focus on regions of importance, this is done by obtaining a set of weights that will help to correct the bias of the sample from the trial distribution $g(x)$. Again our goal is to approximate the integral $\mu = \int f(x)dx$ and we proceed to make the factorization

$$\mu = \int p(x)h(x)dx$$

obtaining the $p(x)$ from which we produce our sample $x^{(1)}, x^{(2)}, \ldots, x^{(n)}$ to then calculate the weights as follows:

$$w(x^{(j)}) = \frac{p(x^{(j)})}{g(x^{(j)})} \qquad , j = 1, \ldots, n$$

and finally approximate $\mu$ by:

$$\tilde{\mu} = \frac{h(x^{(1)})w(x^{(1)}) + h(x^{(2)})w(x^{(2)}) + \cdots + h(x^{(n)})w(x^{(n)})}{w(x^{(1)}) + \cdots + w(x^{(n)})}$$

It can be easily shown, using the law of large numbers, that importance sampling can be used to approximate $\mu$ by calculating $\tilde{\mu}$.

## Problems - Part 1

### Hardy-Weinberg model with 6 genotypes.

Let $\theta_1$, $\theta_2$, $\theta_3$ be the probabilities of alleles S,I,F. We know that 1 genotype consists of two alleles. To find the $2n = 6$ genotypes, we have the system $(\theta_1 + \theta_2 + \theta_3)^2$. We are interested in finding out if the resulting distribution specified by $\vec{\theta}$ has a chi-squared distribution. Let's find out the degree of freedom (DOF): We have 6 possible outcomes. Since $\theta_3$ is dependent on $\theta_1 + \theta_2$, that reduces DOF by 1 as well. So DOF is 4. Compare Chi-square statistic for 5 degrees of freedom to the observed one. Use significance level $\alpha = 5\%$

**Hypothesis testing:**

- $H_0$ hypothesis – 5 of the sample variables (different genotypes) are independent of each other (conversely: how probable is that the resulting sample is due to chance) – the sample is $\chi^2_5$-distributed. We have 4 of the 6 genotypes because of the 2 constraints on the system $\sum \theta_i = 1$ and $\sum_{i,j} \theta_i\theta_j = 1$.
- $H_1$ hypothesis – 5 of the sample variables are somehow mutually dependent – the sample is not $\chi^2_5$-distributed

We know that if $p > \alpha$ then the probability of making type 1 error is higher than the significance level, meaning that we would need to stay with $H_0$ (we don't prove $H_0$ but we always assume that $H_0$ is true unless proven otherwise).

$\chi^2$-distribution is given by $X^2 = \sum_{i=1}^{100} \frac{(O_i - E_i)^2}{E_i}$, where $i_{\max}$ is the sample size.

- The classic 3-allele Hardy-Weinberg Model assumes the following:

```
* No Natural Selection: neither allele confers a selective advantage or disadvantage
* No migration: No one enters or leaves the population
* No mutation: Alleles cannot transform into each other
* Infinite Population Size: No genetic drift
* Random Mating
```

Source: Chi-Square Tutorial

- a. *Code:*

```
require(stats)
set.seed(123)
theta <- runif(100)
specifycount <- function(theta){
        count_theta = rep(NA,6)

        a = c(1/9,2/9,3/9,5/9,7/9,1)
        for (i in 1:length(a)){
                if (i == 1){
                         count_theta[i] <- length(which(theta <a[i]))
                } else {
                count_theta[i] <- length(which(theta <a[i] & theta>=a[i-1]))
                }

        }
        return(count_theta)
}
specifycount(theta)
```

```
## [1]  9 13 11 24 19 24
```

- *b. Chi-squared test:*

```
genotype_prob <- c(rep(1/9,3), 2*rep(1/9,3))
chisq.test(specifycount(theta), p = genotype_prob)
```

```
##
##  Chi-squared test for given probabilities
##
## data:  specifycount(theta)
## X-squared = 1.475, df = 5, p-value = 0.9159
```

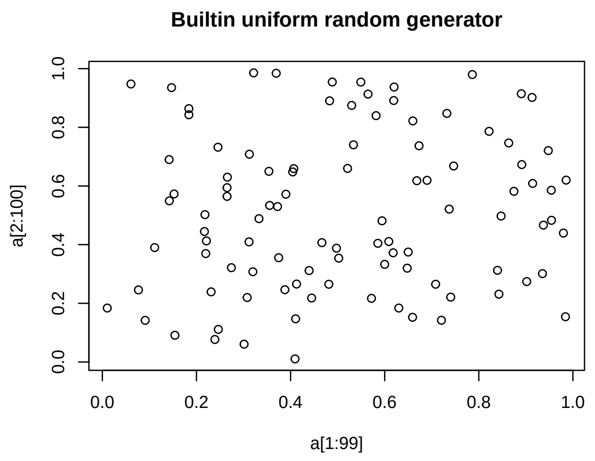
Since $p > \alpha$, the probability of making type 1 (mistakenly claiming that distributions of the samples are not independent of each other – meaning that claiming that they have some mutual dependency, e.g. correlation or regressive dependence) is a lot higher than the significance level (upper limit for type 1 error), we have to stay with the null hypothesis that 5 of the genotypes are independent – this means that they do not have the same distribution, the resulting sample is due to chance (again: we don't prove the null hypothesis but we always assume that it is true unless proven otherwise).

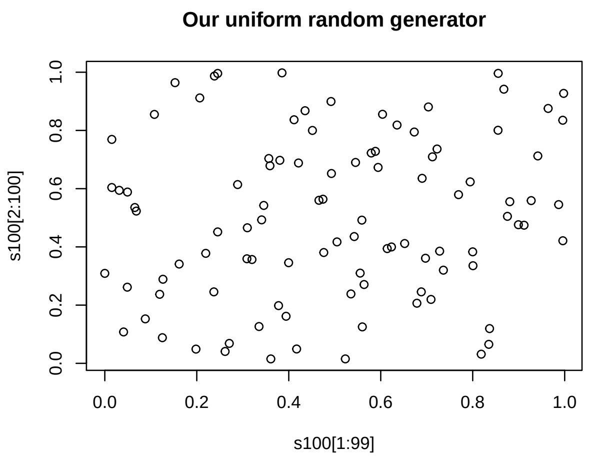
## Task 2.2

*a) and c) Find a suitable combination of a,b,M*

```
random <- function(seed = 55,n=100,M=2^20, a=12,c= 3)
{
        rand <- rep(NA,n);
        rand[1] <- seed;
        for(i in 2:n){
                rand[i] <- (a*rand[i-1]+c)%%M;
        }
        return(rand/M)
}


par(mfrow=c(2,2))
s10 <- random(0.5,10,2^32,69069,1327217885)
```

```r
a <- runif(100)
s100 <- random(0.5,100,2^32,69069,1327217885)
plot(a[1:99],a[2:100], main = "Builtin uniform random generator")
plot(s100[1:99],s100[2:100], main = "Our uniform random generator")
```





> *b) Apply the Kolmogorov Smirnov test to a generated sample of size 10 and 100*

```r
ks.test(s100, "punif")
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  s100
## D = 0.053751, p-value = 0.9348
## alternative hypothesis: two-sided
```

```r
ks.test(s10,"punif")
```

```
##
##  One-sample Kolmogorov-Smirnov test
```

```
##
## data:  s10
## D = 0.24083, p-value = 0.5311
## alternative hypothesis: two-sided
```

From the results we see that the combination without _my ending is better, so a reasonable combination of parameters for the pseudorandom generator is $seed = 0.5, M = 2^{32}, a = 69069, c = 1327217885$, so we'll use these values. Pick significance value $\alpha = 5\%$. > Hypothesis testing:

- $H_0$ hypothesis – the random sample is uniformly distributed
- $H_1$ hypothesis – reject $H_0$

  for 10-sample: Since p is higher than 60% in both cases, $H_0$ cannot be rejected on a significance level $\alpha = 5\%$. – We can see that increasing the sample size makes the probability of making type 1 error higher. p-value in that case is around 44-99%.

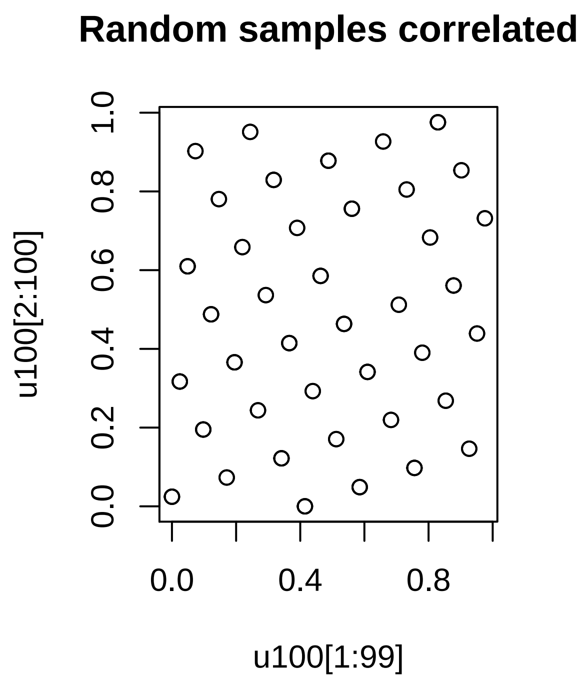
  - d) Find an unreasonable combination of a,b,M that still passes the Kolmogorov Smirnov test*

  One-sample Kolmogorov-Smirnov test using inbuilt parameter 'punif' – Pick significance value $\alpha = 5\%$. $H_0$-hypothesis: the random samples have the same distribution. $H_1$-hypothesis: the random samples do not have the same distribution

```r
u100 <- random(66,100,123,12,3)
cor(u100[1:99],u100[2:100])
```

```
## [1] 0.1434807
```

```r
#lots of correlation, short period
plot(u100[1:99],u100[2:100], main = "Random samples correlated")
```



```r
#OK result
ks.test(u100,"punif")
```

```
## Warning in ks.test(u100, "punif"): ties should not be present for the
## Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  u100
## D = 0.070244, p-value = 0.7072
## alternative hypothesis: two-sided
```

Although we see that there is considerable amount of correlation between the variables (Pearson correlation coefficient 0.1435), Kolmogorov-Smirnov test is still passed (p-value 0.7072). This shows that a deterministic algorithm can still pass the Kolmogorov Smirnov test even in case of correlation between the variables.

## Task 2.3

(a) Write an R-code for the rejection algorithm with normal distribution as instrumental distribution g(x).

(b) Determine the parameters of the normal distribution and the con- stant M for optimizing the algorithm..
(c) Sign a picture of p(x), Mg(x).
(d) Apply an ‘usual’ test on the generated sample.

**Take the smallest M, such that p(x) < M*g(x)**

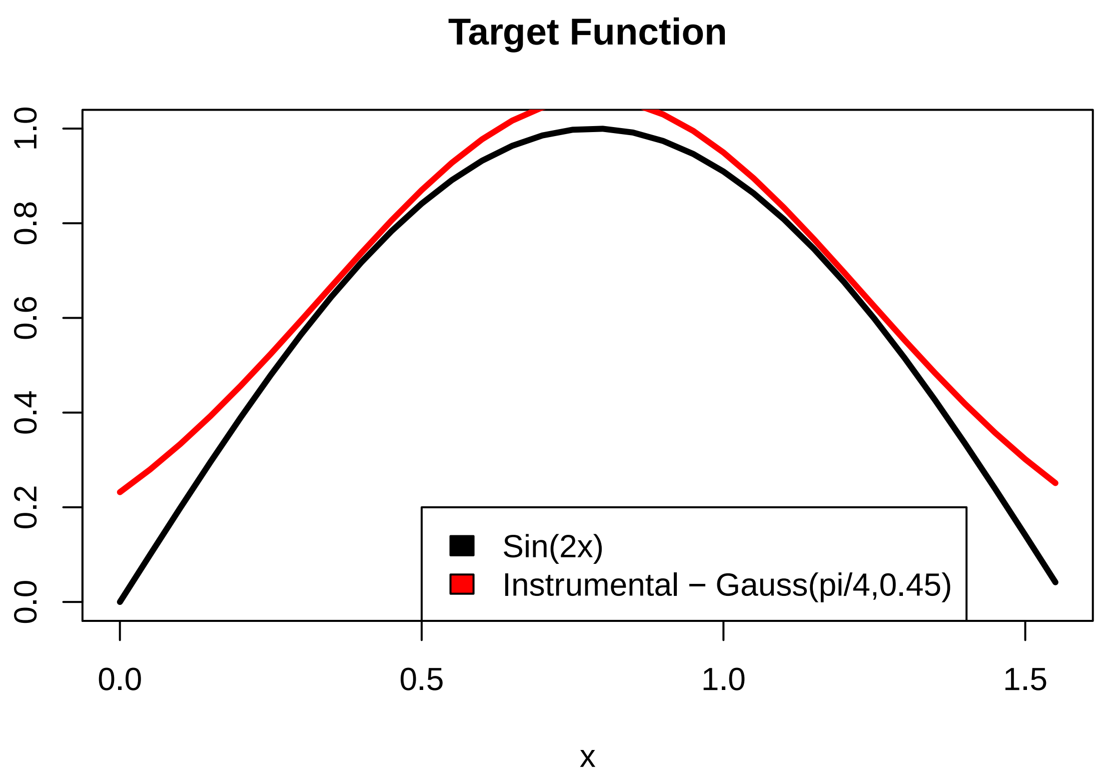
**Rejection sampling**

Wanted an i.i.d. sample of length 100 from the distribution with density $p(x) = sin(2x), 0 < x < \pi/2$

## Determine the parameters of the normal distribution and the constant M for optimizing the algorithm

Since $Mg(x) = p(x)$, *at the optimum:* $M_{\text{opt}} = p(x_{\text{opt}})/g_{(}x_{\text{opt}})$ Since at the optimum the densities have to be equal, one obvious choice would be to take $\mu = \{x|p(x) = \max\sin(2x)\}, x \in ]0, \pi/2[$. and since that value is $\pi/4$, we will take $\mu = \pi/4$

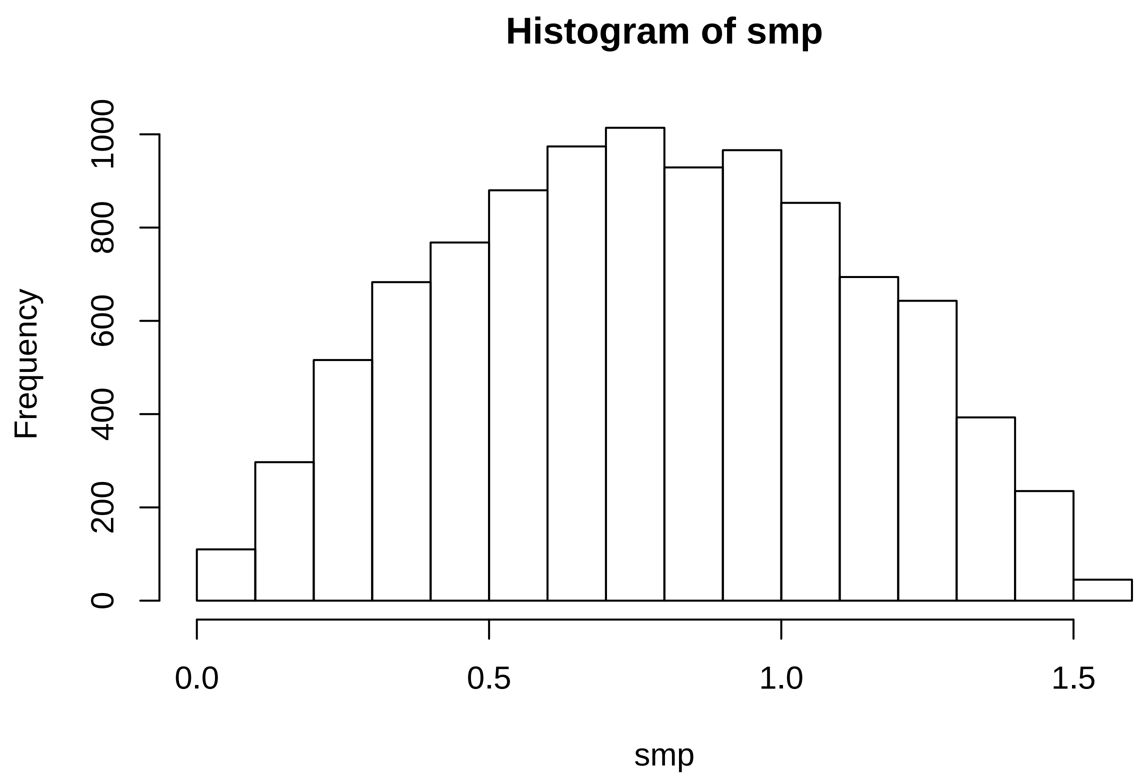
```r
x<-seq(0,pi/2,0.05)

m<-1.2

plot(x,sin(2*x),"l",main="Target Function",xlab="x",ylab="",col=1,lwd=3)
lines(x,m*dnorm(x,pi/4,0.45),lwd=3,col=2)
legend(0.5, 0.2, c("Sin(2x)", "Instrumental - Gauss(pi/4,0.45)"), fill = c("black","red"))
```



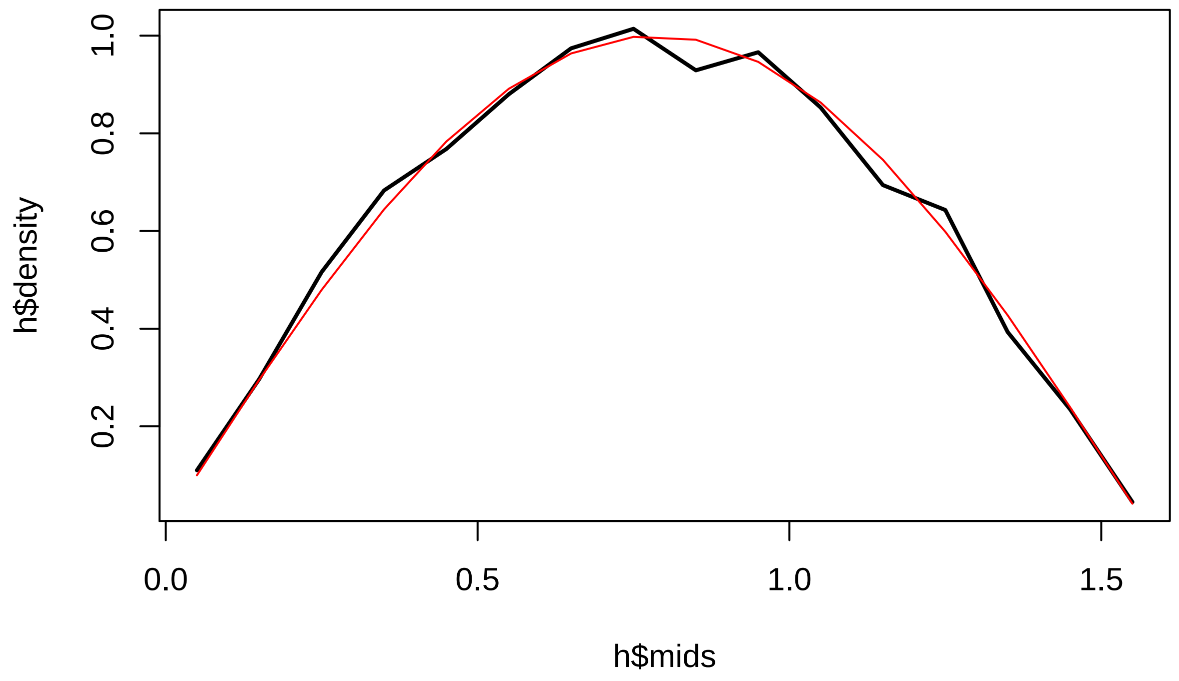
```r
rand.reject<-function(N,M) # define your rejection sampling here
{
  rand<-rep(NA,N);
  for(i in 1:N)
  {
    L<-TRUE;
    while(L)
    {
      rand[i]<-rnorm(1,pi/4,0.45);
      r<-sin(2*rand[i])/(M*dnorm(rand[i],pi/4,0.45));
      if(runif(1)<r)
      {
        L<-FALSE
      }
    }
  };
  return(rand)
}

smp <- rand.reject(10000,m)
length(smp)
```

```
## [1] 10000
```

```
h <- hist(smp)
```



```r
plot(h$mids,h$density,"l", col=1,lwd=2)
lines(h$mids,sin(2*h$mids),col=2)
```



```r
ks.test(sin(2*h$mids),h$density)
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  sin(2 * h$mids) and h$density
## D = 0.0625, p-value = 1
## alternative hypothesis: two-sided
```

The best set of parameters seems to be $M = 1.2$, $\mu = \pi/4$ , $\sigma = 0.45$. Since $\sin(2x)$ is a true density in $x \in ]0, \pi/2[$, one can compare its value at the midpoints of the 16 columns compared to the density of points obtained from our accept-rejection sampling and run the Kolmogorov Smirnov Test on the densities to see if they have the same distribution.

On a significance level $\alpha = 5\%$ we can say that since $p > \alpha$, thus the probability of mistakenly accepting $H_1$ is over the critical value, we can not reject $H_0$ that the samples have the same distribution.

We see that for sample size 100, we have often **p-value equal to 1 or 0.9999**. Intererestingly, mostly all the points in the accept-reject method get accepted – the p-value is 1. *Does this mean that our normal distribution estimation is as good as it can get since no points are excluded?* It seems so to us right now . . . .

# Problems - Part 2 : MC and MCMC

## Calculate an approximative value with help of Monte Carlo methods for the following integral:

$$\mu = \int_{-2}^{0} \exp(-(x-1)^2)dx$$

- 1. *Reformulate the integral as an expected value of a normal distribution. Find out the value with R, use pnorm.*
- 2. *Approximate u by independent MC basing on the normal distribution. Generate normal distributed random variables of a suitable size and estimate the related expected value.*
- 3. *Approximate u by a hit and miss procedure. Sign a picture, where the area is colored and the random points are included.*
- 4. *Approximate u by importance sampling, use a reasonable trial distribution.*

## 1. PNORM and Integrand Factorization

**Find the integral and formulate it in terms of a normally distributed random variable sampled expectation value:**

$$\mu = \int_{-2}^{0} \exp\left(-(x-1)^2\right)dx = \int_{-2}^{0} \exp\left(\frac{-(x-1)^2}{2}\right)\frac{-(x-1)^2}{2}dx = \sqrt{2\pi}\int_{-2}^{0} \exp\left(\frac{-(x-1)^2}{2}\frac{1}{\sqrt{2\pi}}\frac{-(x-1)^2}{2}dx,\right.$$

We see that if $X \sim \mathcal{N}(1,1)$, then $h(x) = \sqrt{2\pi}\exp\left(\frac{-(x-1)^2}{2}\right.$

So we can approximate the initial integral as an expectation value of $\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} h(x_i)$, where $x_i$ are sampled from $X \sim \mathcal{N}(1,1)$

**Using Laplace function we can re-formulate the integral in another way:**

$$P(\alpha \le X \le \beta) = \frac{\operatorname{erf}\left(\frac{\beta-\mu}{\sigma\sqrt{2}}\right) - \operatorname{erf}\left(\frac{\alpha-\mu}{\sigma\sqrt{2}}\right)}{\beta-\alpha}$$

We see that if $X \sim \mathcal{N}(1, \frac{1}{\sqrt{2}})$, then $f(x) = \frac{1}{\sqrt{\pi}}\exp\big(-(x-1)^2\big)$ and

$$\mu = \sqrt{\pi}P(-2 \le X \le 0) = \frac{\sqrt{\pi}}{2}\big(\operatorname{erf}(3) - \operatorname{erf}(1)\big) \approx 0.1394$$

We used the fact that the Laplace error function is an odd function.

**Using PNORM**

*pnorm is a CDF i.e. antidrivative*

```
sqrt(pi)*pnorm(0, 1, 1/(sqrt(2)))-pnorm(-2, 1, 1/(sqrt(2)))
```

```
## [1] 0.1393917
```

## 2. Independent Monte Carlo

Sample from $X \sim \mathcal{N}(1,1)$, then we can use the $h(x)p(x)$ factorization indicated above.

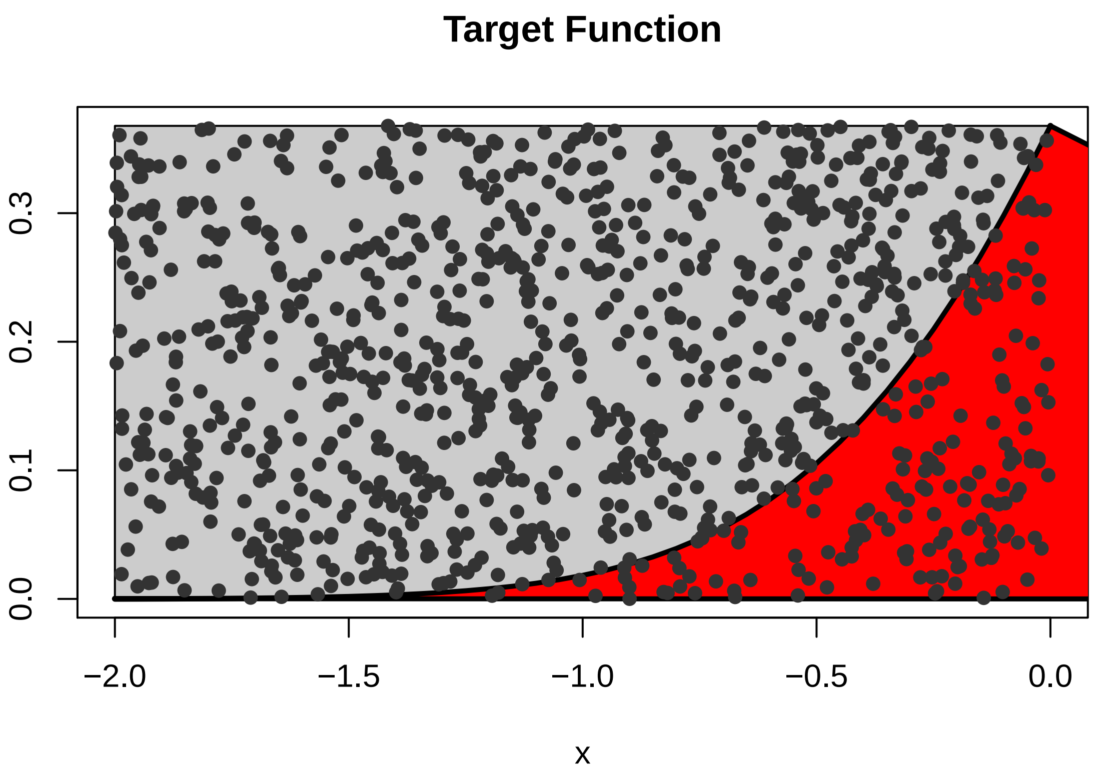
```r
independent <- function(N){
xx <- rnorm(N,1,1)

 # always use & to get VECTORIZED OPERATION
zz <- xx[ xx < 0 & xx >-2]
hzz <-sqrt(2*pi)*exp(-1/2*(zz-1)^2) # indicator function
return (sum(hzz)/N)
}
independent(10000)
```

```
## [1] 0.1433551
```

## 3. Hit and Miss Monte Carlo

**The integral value is obtained by first finding the probability of a uniformly distributed random variable falling into the given exp-function subtended area by counting the points there, and then the integral (the area of subtended by exp-function) can be found by multiplying the rectangle area by the probability found by uniform sampling.**

```r
x<-seq(-2,0,0.05)
plot(x,exp(-(x-1)^2),"l",main="Target Function",xlab="x",ylab="",col=1,lwd=3, xlim=c(-2,0))

m <- max(exp(-(x-1)^2))

rect(-2,0,0,m,col=gray(0.8))
lines(x,exp(-(x-1)^2),"l",lwd=2)

#sample the rectangle
xr<-runif(1000,-2,0)
yr<-runif(1000,0,m)

#our integral is the fraction of the dots times the rectangle area
integral_val<-(length(which(yr < exp(-(xr-1)^2)))/length(yr))*(m*2)


xp <- c(x,-x)
yp <- c(exp(-(x-1)^2),rep(0,length(x)))
polygon(xp,yp,col="red",lwd=2.5)

points(xr,yr,pch=16,lwd=0.5,col=gray(0.2))
```



```
hitandmiss <- function(N){
    #max at 0
    m <- exp(-1)
    xr<-runif(N,-2,0)
    yr<-runif(N,0,m)
    area<-(length(which(yr < exp(-(xr-1)^2)))/length(yr))*(m*2)
    return(area)
}
```

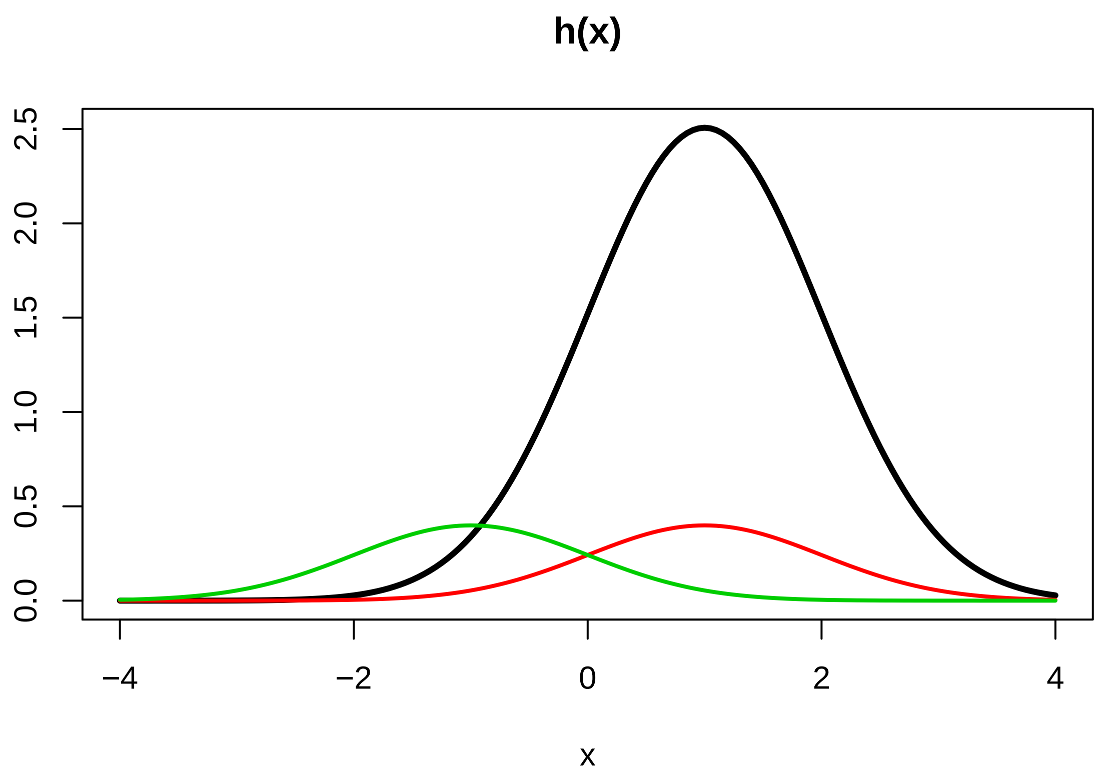
## 4. Importance sampling:

$h(x) = \sqrt{2\pi}\exp\left(-\frac{(x-1)^2}{2}\right)$, $p(x) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{(x-1)^2}{2}\right)$ $g(x) \sim \mathcal{N}(-1,1)$ normal distribution centered in the middle of the interval We have:

```
importance <- function(N=1000){
  gx<-rnorm(N,-1,1)
  #weight
  wx<-exp(2*gx)
  hx<-sqrt(2*pi)*exp(-0.5*(gx-1)^2)[gx<0 & gx>-2]
  wwx<-wx[gx<0 & gx > -2]
  return(sum(hx*wwx)/sum(wx))
}

importance(100000)
```

```
## [1] 0.1393919
```

```
x<-seq(-4,4,0.05)
plot(x,sqrt(2*pi)*exp(-1/2*(x-1)^2 ),"l",main="h(x)",xlab="x",ylab="",col=1,lwd=3)
#px
lines(x,dnorm(x,1,1),lwd=2,col=2)
#gx
lines(x,dnorm(x,-1,1),lwd=2,col=3)
```



```
M1 <- rep(NA,200)
M2 <- rep(NA,200)
M3 <- rep(NA,200)
M4 <- rep(NA,200)
for (i in 1:200)
{
    M1[i]<-importance(i*10);M2[i]<-hitandmiss(i*10);
}
```

## 5. MCMC problems

(a) Write down the steps of a random walk Metropolis algorithm for N(1,0.5).

b) What is the proposal distribution T(x,y)? Is T(x,y) = T(y,x)? What is the actual transition function A(x,y) in this example?

(c) How should the scaling parameter be chosen taking into account the acceptance probability and the autocorrelation.
(d) Does the convergence depend on the starting point? Experiment with different starting values.
(e) Generate a Markov chain with stationary distribution N(1,0.5) by this algorithm. (Give an argument for the scaling parameter you have chosen).
(f) Compute the approximate value of the integral above by MCMC. Don't forget to analyze the samples drawn after the burn-in period.

### 5.a) Metropolis-Random walk

> We know that kernel density estimation (KDE) is a non-parametric way to estimate the probability density function of a random variable.

In MCMC case, we are constructing a Markov Chain and using the stationary distribution of this, $\pi(x)$, we want to calculate the integral $\mu \approx \hat{\mu} = \int h(x)\pi(x)dx$. in our case, $\pi(x) = \mathcal{N}(1, 1/2)$.

```
> (a)  Write down the steps of a random walk Metropolis algorithm for N(1,0.5).
```

**Random walk Metrolopolis algorithm:**

- 1) Draw $\varepsilon \sim g_\sigma$, where $\sigma$ is the scaling parameter. If $\varepsilon$ has a uniform distribution, then $\sigma$ ($a$ in this case), determines the range of it, since that is the only parameter the uniform distribution has. Update the state $y = x^{(t)} + \varepsilon$.
- 2) Draw a random number $u \sim U[0, 1]$.
- 3) Calculate the ratio $R(x^{(t)}, y) = \frac{\pi(y)}{\pi(x^{(t)})}$
- 4) Update the state according to

$$x^{(t+1)} = \begin{cases} y, & \text{if } u \leq \min(1, R(x^{(t)}, y)) \\ x^{(t)}, & \text{otherwise} \end{cases}$$

### 5.b) Metropolis-Random walk

```
b) What is the proposal distribution T(x,y)? Is T(x,y) = T(y,x)?
What is the actual transition function A(x,y) in this example?
```

T – proposed y distribution. R – y acceptance distribution. The proposal distribution is $T(x, y) = x^{(t)} + \varepsilon$. In our random walk case, the distribution is symetrical, centered around current state and $T(x, y) = T(y, x)$ since the proposal window is symmetrical. Transition probabilities $A(x, y) = T(x, y) * r(x, y)$ where $r(x, y) = min(1, R(x, y))$

### 5.c) Metropolis-Random walk

```
(c) How should the scaling parameter be chosen taking into account the
acceptance probability and the autocorrelation.
```

Autocorrelation in Wikipedia:

“In statistics, the autocorrelation of a random process describes the correlation between values of the process at different times, as a function of the two times or of the time lag. Let X be some repeatable process, and i be some point in time after the start of that process. Then Xi is the value (or realization) produced by a given run of the process at time i. Suppose that the process is further known to have defined values for mean ??i and variance ??_i-squared for all times i. Then the definition of the autocorrelation between times s and t is

$$R(s,t) = \frac{E[(X_t - \mu_t)(X_s - \mu_s)]}{\sigma_t \sigma_s}$$

- High $a$ – lower acceptance probability (since we may go out of our distribution), high coverage of the wanted distribution, lower autocorrelation between states at different times. The chain spends most of it’s time in the same state, occasionally moving.
- Low $a$ – higher acceptance probability, but we may not cover the whole distribution (get stuck in a local maximum), since we have little variation, states at times $t$ and $s$ more highly autocorrelated since the spread is smaller. Most of proposals get accepted, however the chain will converge slowly

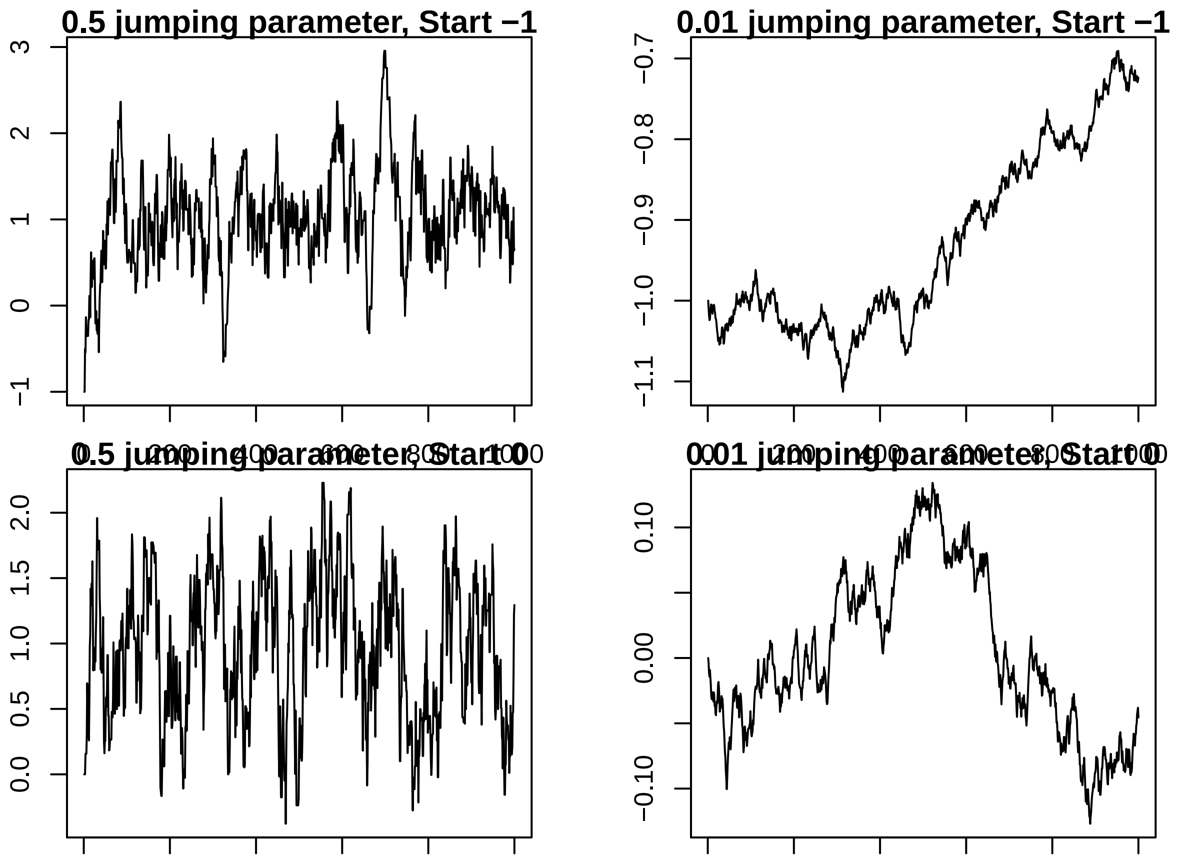
### 5.d) Metropolis-Random walk

```
**5.(d) Does the convergence depend on the starting point? Experiment with different starting values.*
```

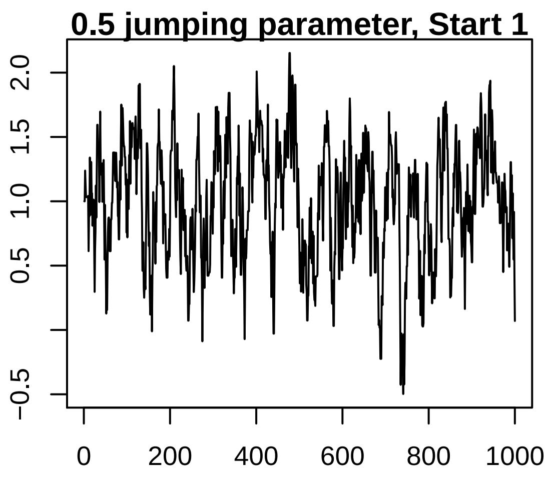
The convergence does not depend on the starting value if the scaling parameter was chosen properly. Too small scaling factor might lead to a lack of convergence.

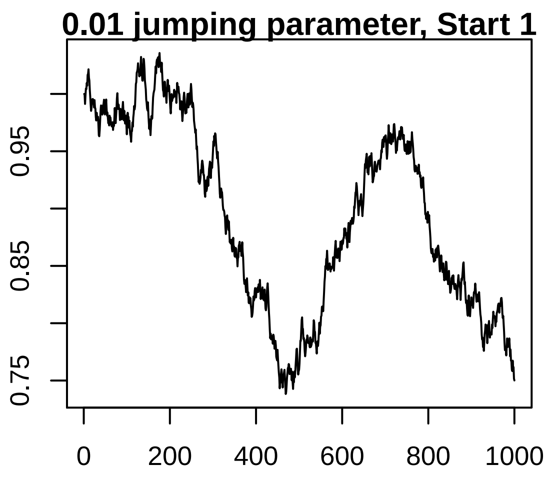
```r
MCMC <- function(a,seed,N)
{
  rand<-rep(NA,N);
  rand[1]<-seed;
  for(i in 2:N)
  {
    #random walk 0,a left or right
    rand[i]<-seed+a*runif(1,-1,1);
    r<-min(1,exp(2*((seed-1)^2-(rand[i]-1)^2)));
    if (runif(1)<r)
    {
      seed<-rand[i]
    }
    else
    {
      rand[i]<-seed
    }
  };
  return(rand)
}

ts1 <- MCMC(0.5,-1,1000)
ts2 <- MCMC(0.5,0,1000)
ts3 <- MCMC(0.5,1,1000)

ts4 <- MCMC(.01,-1,1000)
```

```r
ts5 <- MCMC(.01,0,1000)
ts6 <- MCMC(.01,1,1000)

par(mfrow=c(2,2),mar=c(1, 4, 1, 1))
plot.ts(ts1,ylab="", main = "0.5 jumping parameter, Start -1")
plot.ts(ts4,ylab="", main = "0.01 jumping parameter, Start -1")

plot.ts(ts2,ylab="", main = "0.5 jumping parameter, Start 0")
plot.ts(ts5,ylab="", main = "0.01 jumping parameter, Start 0")
```



```r
plot.ts(ts3,ylab="", main = "0.5 jumping parameter, Start 1")
plot.ts(ts6,ylab="", main = "0.01 jumping parameter, Start 1")
```





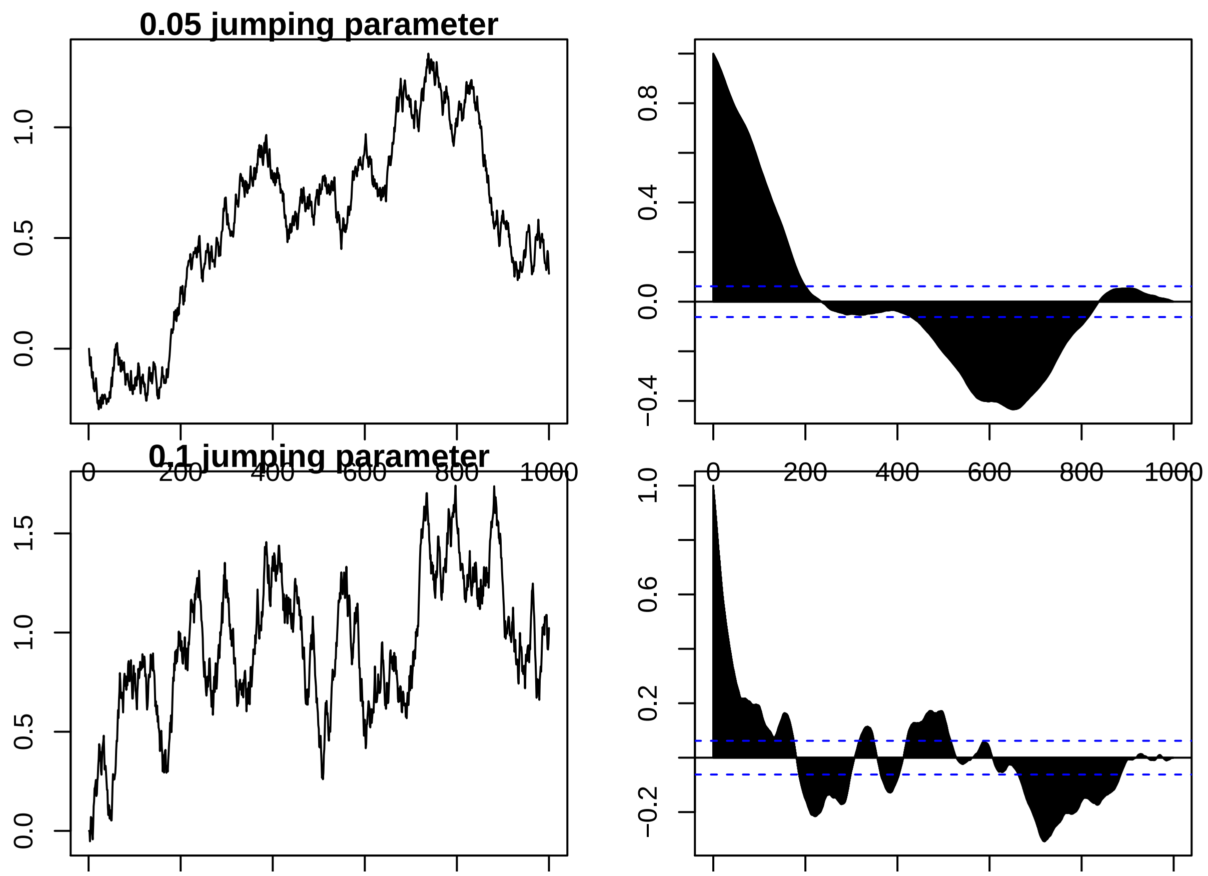
**5.e) Metropolis-Random walk**

```
(e) Generate a Markov chain with stationary distribution N(1,0.5) by
this algorithm. (Give an argument for the scaling parameter you
have chosen).
```

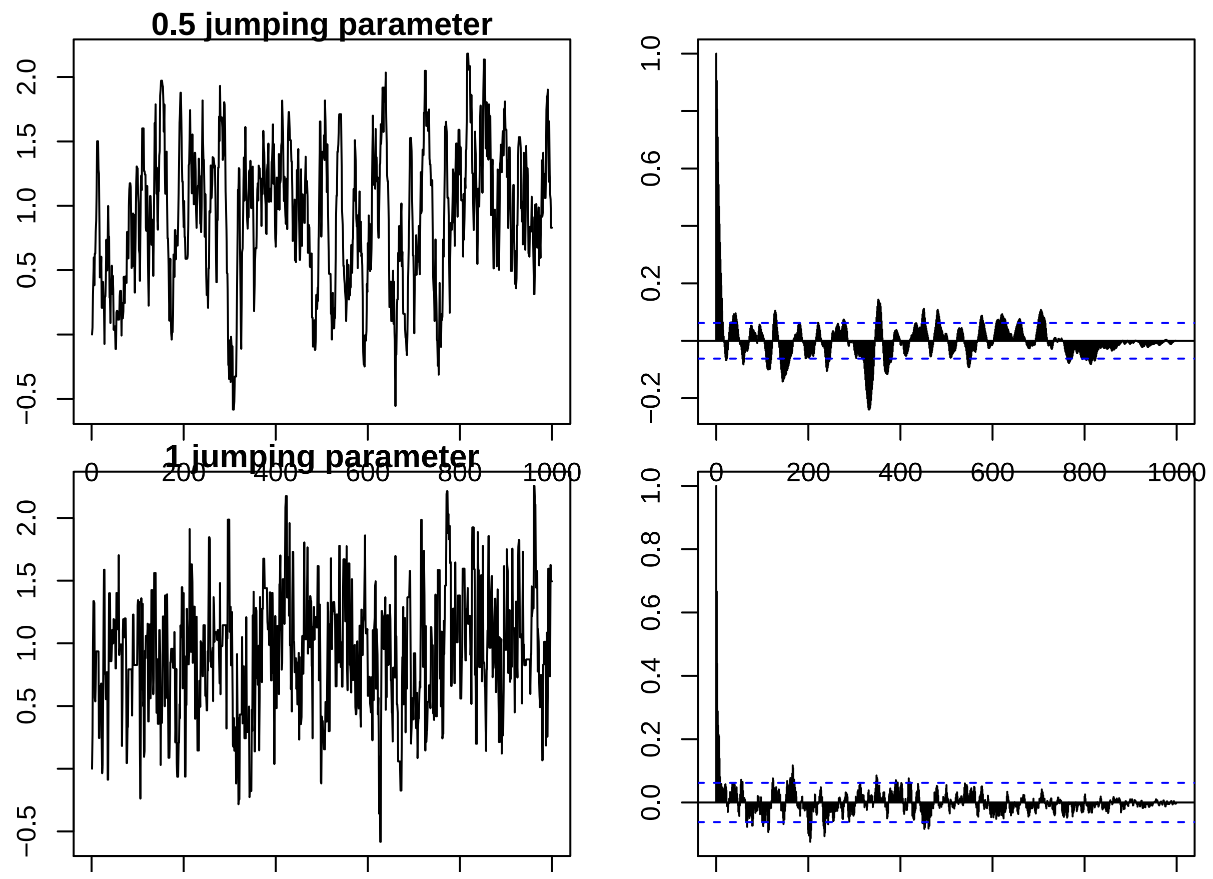
We test a set of scaling factors and conclude that a scaling factor between 0.5 and 1 gives the best results :

```r
ts1 <- MCMC(0.05,0,1000)
ts2 <- MCMC(0.1,0,1000)
ts3 <- MCMC(0.5,0,1000)
ts4 <- MCMC(1,0,1000)
ts5 <- MCMC(5,0,1000)
```

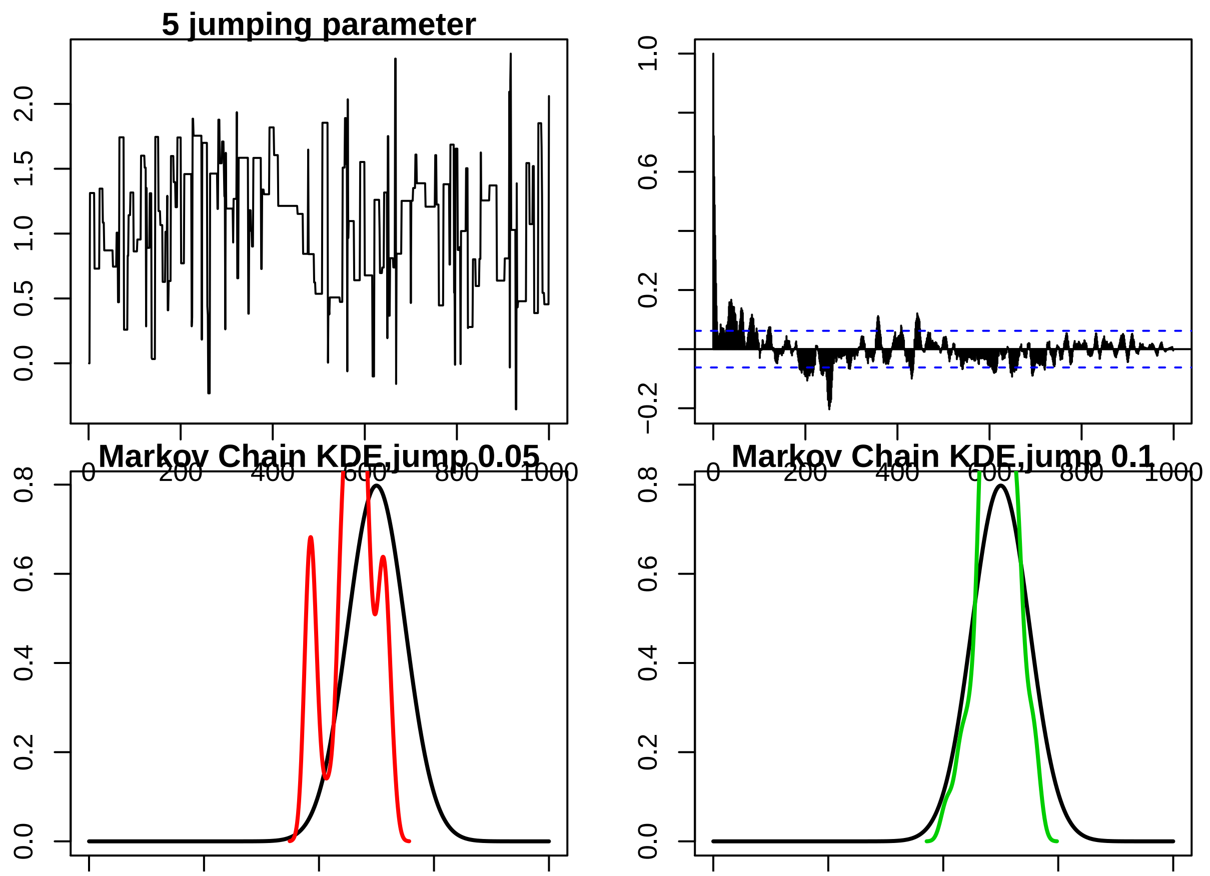
```r
par(mfrow=c(2,2),mar=c(0.5, 3, 1, 1) )
plot.ts(ts1,ylab="", main = "0.05 jumping parameter")
acf(ts1, lag.max=1000)

plot.ts(ts2,ylab="", main = "0.1 jumping parameter")
acf(ts2, lag.max=1000)
```



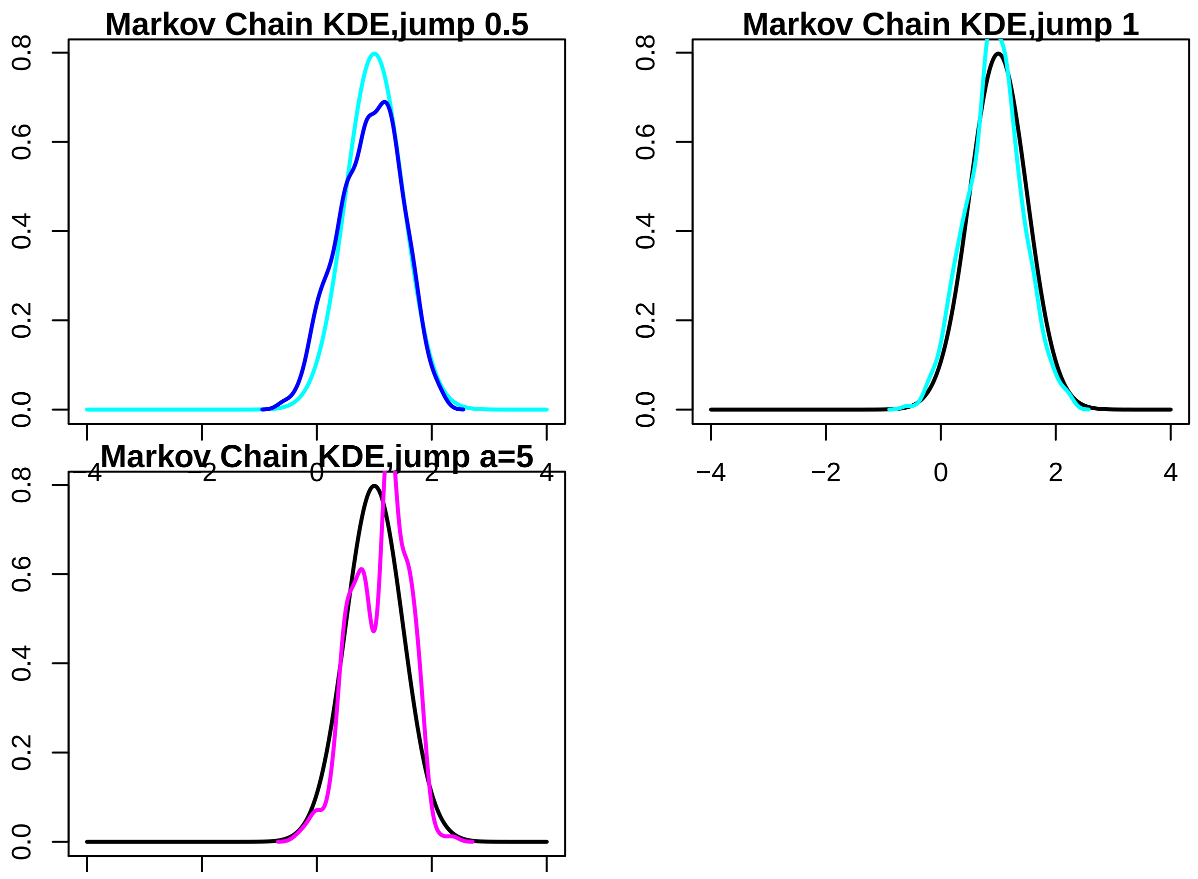
```
plot.ts(ts3,ylab="", main = "0.5 jumping parameter")
acf(ts3, lag.max=1000)

plot.ts(ts4,ylab="", main = "1 jumping parameter")
acf(ts4, lag.max=1000)
```



```
plot.ts(ts5,ylab="", main = "5 jumping parameter")
acf(ts5, lag.max=1000)

x<-seq(-4,4,0.01)
plot(x,dnorm(x,1,0.5),"l", lwd=2,col=1, main = "Markov Chain KDE,jump 0.05")
lines(density(ts1),lwd=2,col=2)


plot(x,dnorm(x,1,0.5),"l", lwd=2,col=1, main = "Markov Chain KDE,jump 0.1")
lines(density(ts2),lwd=2,col=3)
```



```r
plot(x,dnorm(x,1,0.5),"l", lwd=2,col=5, main = "Markov Chain KDE,jump 0.5")
lines(density(ts3),lwd=2,col=4)

plot(x,dnorm(x,1,0.5),"l", lwd=2,col=1, main = "Markov Chain KDE,jump 1")
lines(density(ts4),lwd=2,col=5)

plot(x,dnorm(x,1,0.5),"l", lwd=2,col=1, main = "Markov Chain KDE,jump a=5")
lines(density(ts5),lwd=2,col=6)
```



**The results:**

By looking at convergence, autocorrelation and density plots we propose value of \$ a \$ of 0.75

### 5.f) Metropolis-Random walk

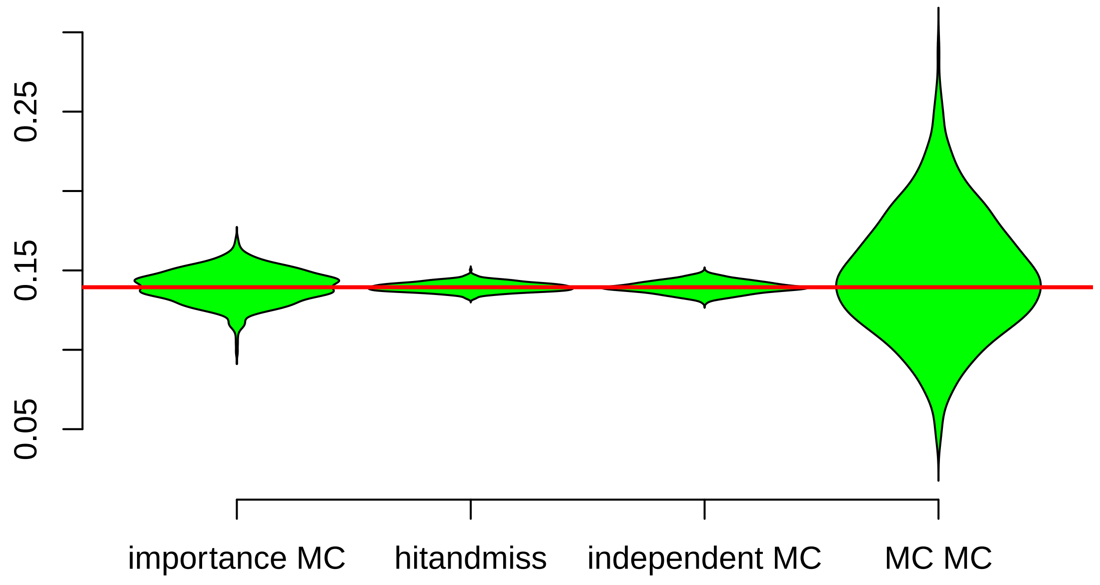
```
(f) Compute the approximate value of the integral from 5.
```

```
  MCMC_int <- function(a,seed,N)
{
  rand<-rep(NA,N);
  rand[1]<-seed;
  for(i in 2:N)
  {
    #random walk 0,a left or right
    rand[i]<-seed+a*runif(1,-1,1);
    r<-min(1,exp(2*((seed-1)^2-(rand[i]-1)^2)));
    if (runif(1)<r)
    {
      seed<-rand[i]
    }
    else
    {
      rand[i]<-seed
    }
  };
  return(sum(rand)/N)
}

MCMC_int(0.75, 1, 10000)
```

```
## [1] 1.006682
```

## 6. Comparison of methods

```
require(UsingR)

mcmc_exp <- function(a,seed,N)
{
    #exp(-(x-1)^2) [-2,0]
    rand<-rep(NA,N);
    rand[1]<-seed;
    for(i in 2:N)
    {
      #random walk 0,a left or right
      rand[i]<-seed+a*runif(1,-1,1);
      r<-min(1,dnorm(rand[i],1,1)/dnorm(seed,1,1))
      if (runif(1)<r)
      {
        seed<-rand[i]
      }
      else
      {
        rand[i]<-seed
      }
    };
    zz <- rand[rand<0]
    zz <- zz[zz>-2]
    hzz <-sqrt(2*pi)*exp(-1/2*(zz-1)^2)
    return(sum(hzz)/N)
}

#1000 replicates
K<-500
#5k samples
S<-10000
#run 4 methods
M1 <- rep(NA,K)
M2 <- rep(NA,K)
M3 <- rep(NA,K)
M4 <- rep(NA,K)

for (i in 1:K)
{
    M1[i]<-importance(S)
    M2[i]<-hitandmiss(S)
    M3[i]<-independent(S)
    M4[i]<-mcmc_exp(0.2,-1,S)
}

Mcomb<-c(M1,M2,M3,M4)
comb<-c(rep(1,K),rep(2,K),rep(3,K),rep(4,K))
violinplot(Mcomb~comb,col="green",names=c("importance MC","hitandmiss", "independent MC", "MC MC"))
abline(a=0.1393917,b=0,col="red",lw=2)
```



*A violin plot is a combination of a box plot and a kernel density plot. Specifically, it starts with a box plot. It then adds a rotated kernel density plot to each side of the box plot.*

**Overall Considerations: One should always choose the unbiased method with the lowest variance.The best methods variance-wise are independent MC and Hit and miss.Both have similar standard deviations, however independent MC seems to be biased towards slightly lower value. This is likely due to the fact that our integral lies in the tail of the sampling distribution – we sample $X \sim \mathcal{N}(1,1)$, but calculate the integral from -2 to 0 .Hit and miss is the most accurate as it samples uniformly across the integration range. The bounding rectangle is small enough to allow for accurate mean and low variance.**