UiO : Matematisk institutt
Det matematisk-naturvitenskapelige fakultet

STK-4051/9051 Computational Statistics Spring 2024
Comments to exercise 9

Instructor: Odd Kolbjørnsen, oddkol@math.uio.no

6.5. Prove the following results.

a. If h_1 and h_2 are functions of m random variables $U_1, \ldots, U_m$, and if each function is monotone in each argument, then

$$\operatorname{cov}\{h_1(U_1, \ldots, U_m), h_2(1 - U_1, \ldots, 1 - U_m)\} \leq 0.$$

b. Let $\hat{\mu}_1(\mathbf{X})$ estimate a quantity of interest, μ, and let $\hat{\mu}_2(\mathbf{Y})$ be constructed from realizations $\mathbf{Y}_1, \ldots, \mathbf{Y}_n$ chosen to be antithetic to $\mathbf{X}_1, \ldots, \mathbf{X}_n$. Assume that both estimators are unbiased for μ and are negatively correlated. Find a control variate for $\hat{\mu}_1$, say Z, with mean zero, for which the control variate estimator $\hat{\mu}_{\mathrm{CV}} = \hat{\mu}_1(\mathbf{X}) + \lambda Z$ corresponds to the antithetic estimator based on $\hat{\mu}_1$ and $\hat{\mu}_2$ when the optimal λ is used. Include your derivation of the optimal λ.

$\mu_1(\boldsymbol{X})$: $E(\mu_1(\boldsymbol{X})) = \mu$, $Var(\mu_1(\boldsymbol{X})) = \sigma_1^2$
$\mu_2(\boldsymbol{Y})$: $E(\mu_2(\boldsymbol{Y})) = \mu$, $Var(\mu_1(\boldsymbol{X})) = \sigma_2^2$

$$Cov(\mu_1, \mu_2) = \sigma_1 \sigma_2 \rho$$

e.g. X-Normal and $\boldsymbol{Y} = -\boldsymbol{X}$,
$\mu_1(\boldsymbol{X})$, $\mu_2(\boldsymbol{Y}) = \mu_1(-\boldsymbol{X})$ then: $\sigma_1 = \sigma_2 = \sigma$

$$Z = \mu_1(\boldsymbol{X}) - \mu_2(\boldsymbol{Y})$$
$$E(Z) = 0$$

- $\mu_C = \mu_1 + \lambda Z$ (assume) $\sigma_1 = \sigma_2 = \sigma$

- $\mathrm{Var}(\mu_C) = Var(\mu_1) + 2\lambda Cov(\mu_1, \lambda\, Z) + \lambda^2 Var(Z)$
  - $Var(\mu_1) = \sigma^2$
  - $Var(Z) = \sigma^2 - 2\rho\sigma^2 + \sigma^2 = 2\sigma^2(1-\rho)$
  - $Cov(\mu_1, Z) = \sigma^2 - \sigma^2\rho = \sigma^2(1-\rho)$

$$\lambda = -\frac{Cov(\mu_1, Z)}{Var(Z)} = -\frac{1}{2}$$

$$\Rightarrow\ \mu_C = \mu_1 - \frac{1}{2}(\mu_1 - \mu_2) = \frac{1}{2}(\mu_1 + \mu_2)$$

Exercise 33 (Importance resampling)
We shall in this exercise test the importance resampling algorithm on the same example as in Exercise 22. The objective is now sampling of random variables rather than approximate calculation of expectations. The simulated, importance resampling (SIR) algorithm, due to Rubin, is in general form as follows. Suppose we want a sample from some awkward distribution with density f. We assume that f can be computed up to a constant. Choose a more convenient distribution with density g. Draw a sample $X_1, ..., X_M$ from g. This is the first step. The second start by computing

$$w_i = \frac{f(X_i)}{g(X_i)}, \qquad i = 1, 2, .., M,$$

and

$$q_i = \frac{w_i}{\sum_j w_j}, \qquad i = 1, 2, ..., M.$$

M = total number of samples from g
m = number of resampled

Let Y be a random variable, defined conditionally of $X_1, ..., X_M$ with distribution

$$P(Y = X_i | X_1, ..., X_M) = q_i, \qquad i = 1, 2, ..., M.$$

Draw m samples of Y. It can then be proved that as $M \to \infty$, a sample of m independent variables from f appears in the limit.

Suppose $f(x) = \phi(x - a)$, where $\phi(x) = (2\pi)^{-1/2}\exp(-x^2/2)$ is the standard normal density. We shall test the efficiency of the SIR algorithm when sampling from $g(x) = \phi(x)$. Use $m = 100$.

(*a*). First let $a = 1$. Vary M from 1000 and upwards. Try to find out how large M must be. You must compare the mean and the variance of the final sample to their known values. Q-Q plotting might be a good idea, and you must also worry about how large M must be in order to make the final sample an independent one. Try to think of a way to measure dependence.

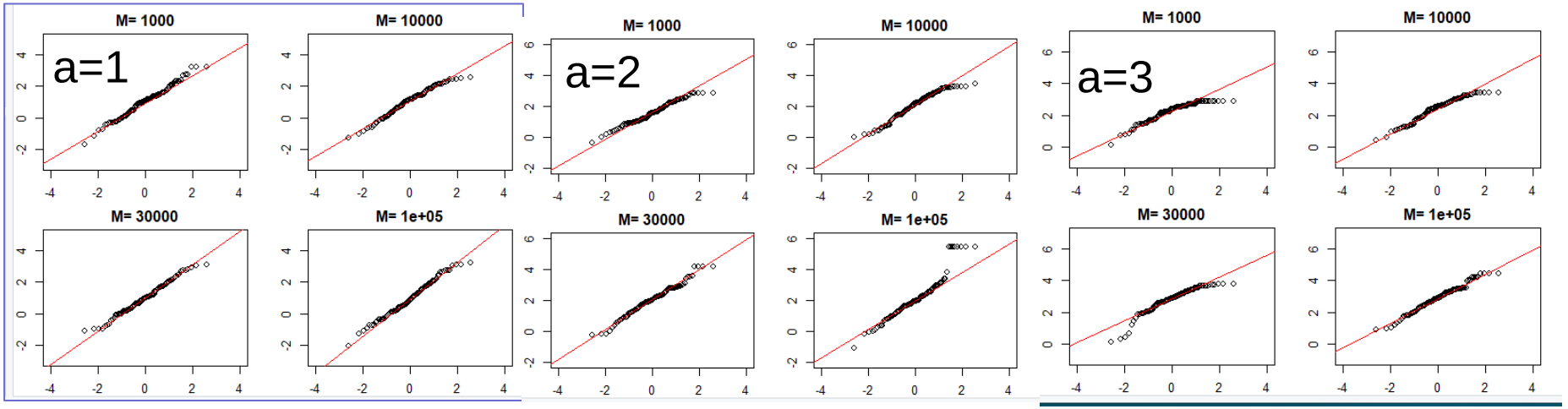
$$f(x) = \phi(x - a) \qquad g(x) = \phi(x)$$

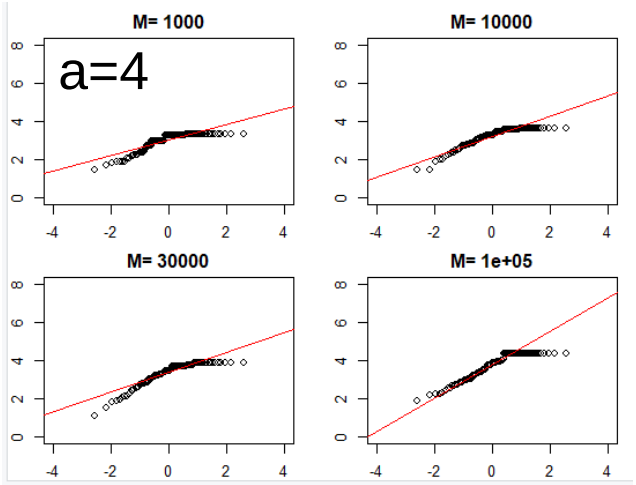
(*b*). Repeat (*a*) for $a = 2, 3, 4$.

(*c*). Formulate general conclusions.

(*d*). Why can we do without the normalization constant in f?

Ex 33 importance sampling





c) As a increase (1 to 4):

- The effective number of samples goes down
- The qq plot does not match the line
- The mean is too low
- The std is too low

d)

- The weigths are normalized prior to re-sampling

Dependency

Let Y be a random variable, defined conditionally of $X_1, ..., X_M$ with distribution

$$P(Y = X_i | X_1, ..., X_M) = q_i, \qquad i = 1, 2, ..., M.$$

Draw m samples of Y. It can then be proved that as $M \to \infty$, a sample of m independent variables from f appears in the limit.

and you must also worry about how large M must be in order to make the final sample an independent one. Try to think of a way to measure dependence.

- How would you know that the samples are not directly from f?
  - Repeated values
- Number of repeats in a sample of size m: multinomial

$$p(n_1, \dots, n_M) = \frac{m!}{n_1! \cdots n_M!} q_1^{n_1} \cdots q_M^{n_M}$$

- Number of repeats of sample k (among m samples): binomial

$$p(n_k) = \frac{m!}{n_k!(m-n_k)!} q_k^{n_k} (1 - q_k)^{m-n_k}$$

Exercise 35 (Sampling by Metropolis-Hastings)
Assume we are interested in simulating from the bivariate Gaussian distribution $N_2(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ where

$$\boldsymbol{\mu} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \quad \boldsymbol{\Sigma} = \begin{pmatrix} 1 & a \\ a & 1 \end{pmatrix}$$

This is of course a simple distribution for which we could perform simulation directly, but we will use it to illustrate the Metropolis-Hastings method.

(a). Implement a random walk Metropolis-Hastings algorithm were one component is changed at a time and the proposal distribution is Gaussian centered at the current value.

(b). Run the algorithm 1000 iterations for $a = 0$. Use simulations from the standard Gaussian distribution as starting points. Tune the standard deviations in the proposal distributions so that the acceptance rate become approximately 0.3.

Estimate $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ from your simulations. Use traceplots to see how many of the first samples you should discard.

Make a plot of your simulations in the two-dimensional space, drawing a line between each iteration.

Comment on the results.

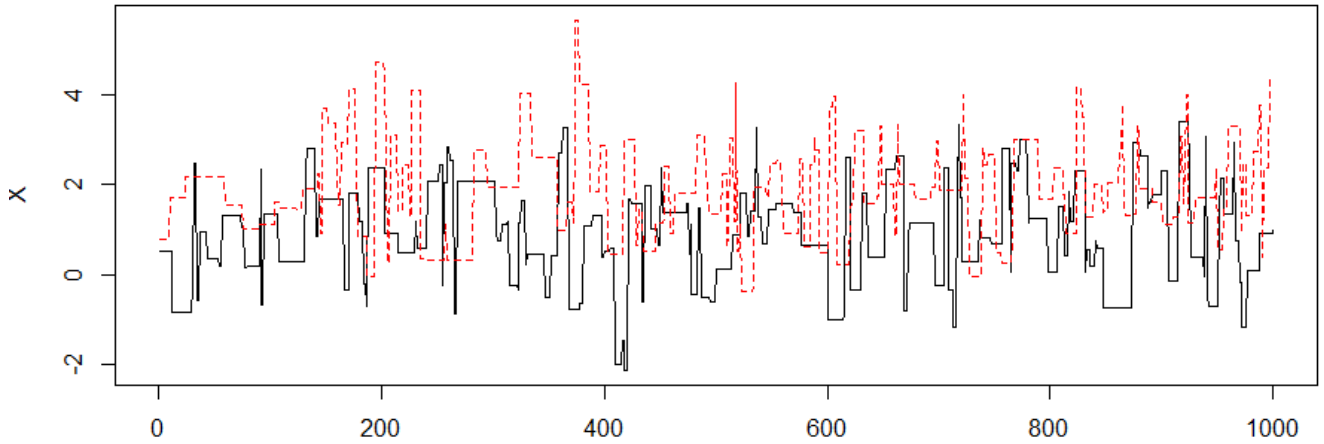
Ex 35

- Random Walk
- Use library mvtnorm
- B) We need more than 1000 samples to get good estimates, but we are in the ball park
- C) We are not even close with the 1000 samples. We are trapped in the valley.

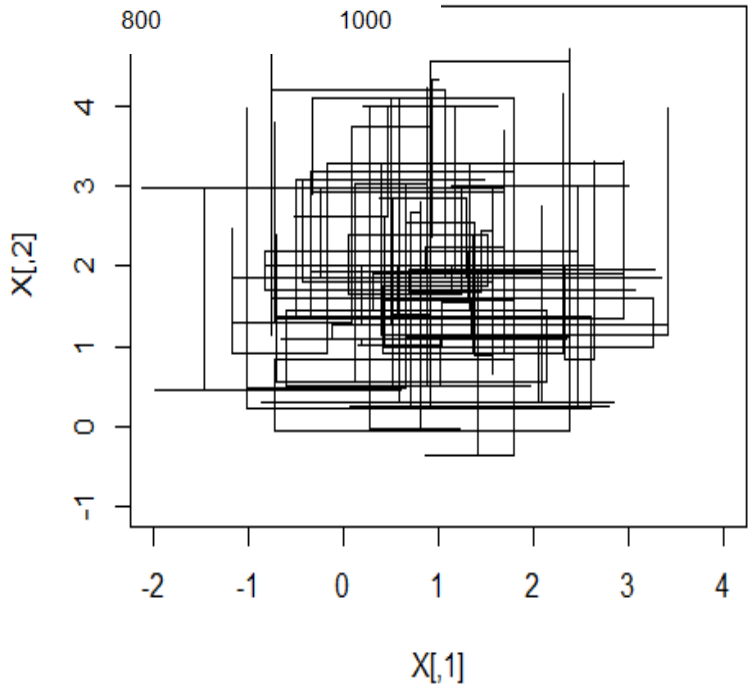
$$R(\mathbf{x}, \mathbf{x}^*) = \frac{f(\mathbf{x}^*)g(\mathbf{x}|\mathbf{x}^*)}{f(\mathbf{x})g(\mathbf{x}^*|\mathbf{x})} = \frac{f(\mathbf{x}^*)}{f(\mathbf{x})}$$

```
acc = 0
for(i in 2:N)
{
  j = sample(1:2,1)
  y = X[i-1,]
  y[j] = rnorm(1,X[i-1,j],sig.prop)
  R = dmvnorm(y,mu,Sigma)/dmvnorm(X[i-1,],mu,Sigma)
  if(runif(1)<R)
  {
    X[i,] = y
    acc = acc +1
  }
  else
    X[i,] = X[i-1,]
}
```

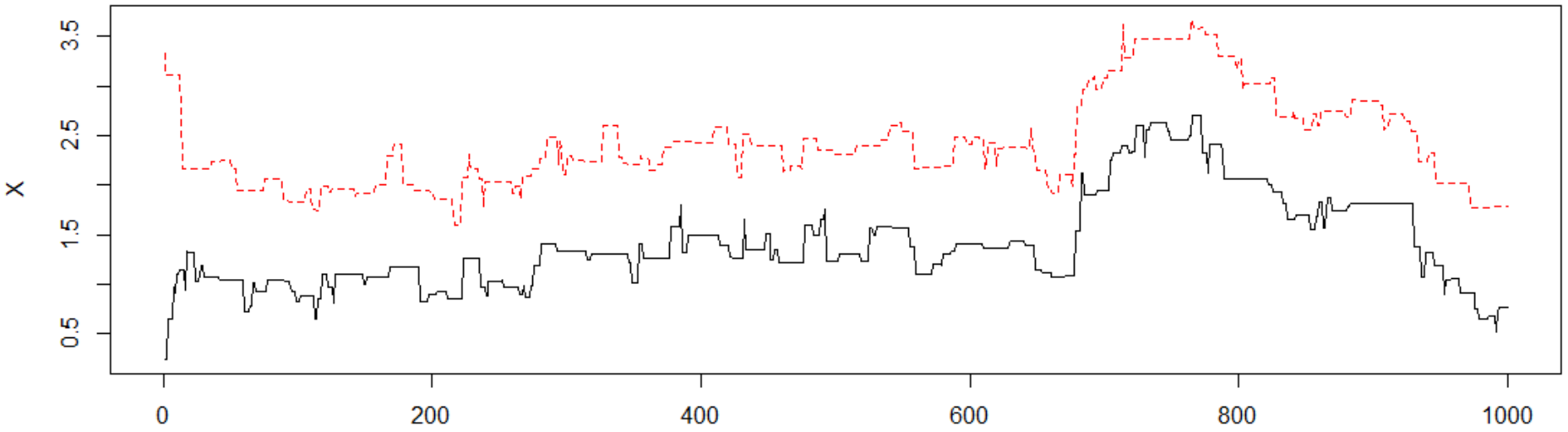
Correlation = 0 in target distribution



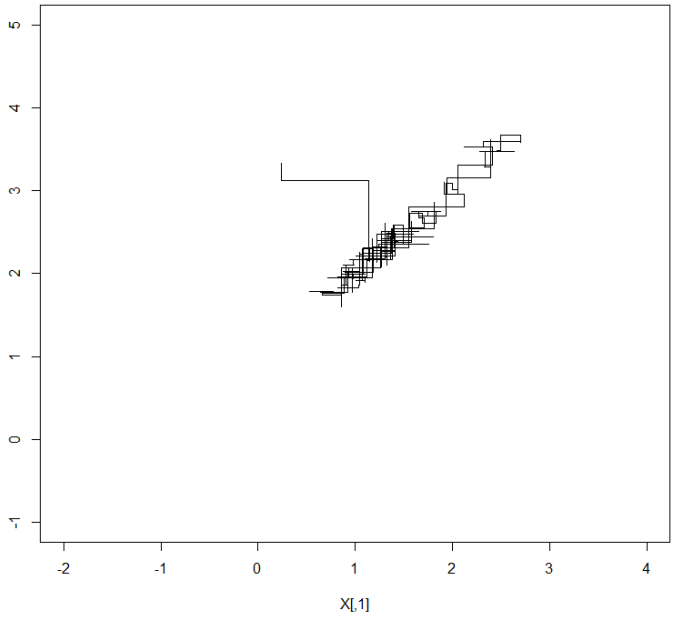

```
> show(colMeans(X))
[1] 0.9082993 1.8454524
> show(var(X))
             [,1]        [,2]
[1,]  1.22264549 -0.02540639
[2,] -0.02540639  1.17549567
> par(mfrow=c(1,1))
> plot(X,type="l",xlim=c(-2,4),ylim=c(-1,5))
> cat("Acceptance rate=",acc/(N-1),"\n")
Acceptance rate= 0.3063063
```



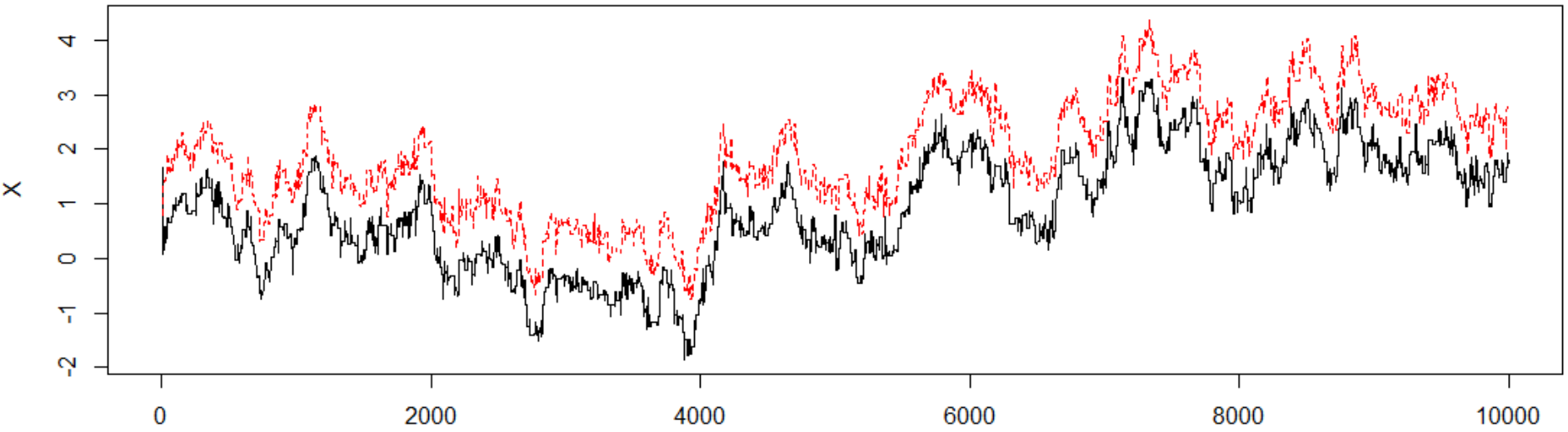
Correlation = 0.99 in target distribution



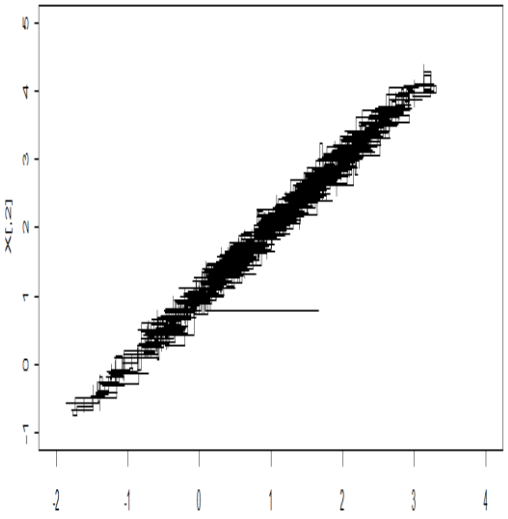

```
> show(colMeans(X))
[1] 1.404351 2.420031
> show(var(X))
          [,1]      [,2]
[1,] 0.214458 0.1920640
[2,] 0.192064 0.2116643
> par(mfrow=c(1,1))
> plot(X,type="l",xlim=c(-2,4),ylim=c(-1,5))
> cat("Acceptance rate=",acc/(N-1),"\n")
Acceptance rate= 0.3073073
```



Correlation = 0.99 in target distribution




```
> show(colMeans(X))
[1] 1.404351 2.420031
> show(var(X))
          [,1]      [,2]
[1,] 0.214458 0.1920640
[2,] 0.192064 0.2116643
> par(mfrow=c(1,1))
> plot(X,type="l",xlim=c(-2,4),ylim=c(-1,5))
> cat("Acceptance rate=",acc/(N-1),"\n")
Acceptance rate= 0.3073073
```



Exercise 36 (Sampling random variables by Metropolis)
Let X be a random variable with density $f(x) = ch(x)$, where h is a computable expression. We shall in this exercise discuss sampling of X through the Metropolis algorithm. The aim is to demonstrate how the algorithm works, not to produce the most efficient sampling scheme conceivable.

Let $\varepsilon_1, \varepsilon_2, ..$ be a sequence of independently and identically distributed random variables. You may choose any distribution symmetrical about the origin. Suggested choice: The uniform over $(-a, a)$, where $a > 0$ has to be adapted for each application. In addition, let $U_1, U_2, ...$ be a sequence of uniforms over $(0, 1)$. Consider the recursion

$$X_{t+1} = X_t + \varepsilon_t I_t$$

where

$$I_t = 1, \qquad \text{if } U_t \leq \frac{h(X_t + \varepsilon_t)}{h(X_n)}$$

and $= 0$ otherwise.

(a). Show that this recursion is a special case of the Metropolis algorithm. Explain, in particular, the relevance of demanding ε_t to be a symmetric distribution.

(a). Show that this recursion is a special case of the Metropolis algorithm. Explain, in particular, the relevance of demanding ε_t to be a symmetric distribution.

Solution to exercise 36. (a). We have that the proposal is $X_t^* = X_t + \varepsilon_t$. Since the proposal distribution is symmetric, the Metropolis-Hastings ratio becomes

$$R_t = \frac{h(X_t^*)}{h(X_t)} = \frac{h(X_t + \varepsilon_t)}{h(X_t)}.$$

If we generate $U_t \sim \text{Uniform}[0,1]$, then we can accept if $U_t \leq \min\{1, R_t\}$ which is equivalent to accept if $U_t < R_t$. This means that

$$\begin{aligned} X_{t+1} &= \begin{cases} X_t^* & \text{if } U_t < R_t \\ X_t & \text{otherwise} \end{cases} \\ &= X_t + I_t \varepsilon_t \end{aligned}$$

given the definition of I_t.

Making the distribution of ε_t to symmetric simplifies the MH-ratio in that the proposal densities disappear.

(b). Suppose f is the standard normal density. Show that the ratio defining I_t becomes $\exp(-X_t\varepsilon_t - 0.5\varepsilon_t^2)$. I_t is more likely to become 1 if the signs of X_t and ε_t are alternate. Explain the meaning of this.

(c). Will you need all U_t to run the algorithm?

(b). In this case

$$R_t = \frac{\exp(-0.5(X_t + \varepsilon_t)^2)}{\exp(-0.5X_t^2)} = \exp(-X_t\varepsilon_t - 0.5\varepsilon_t^2)$$

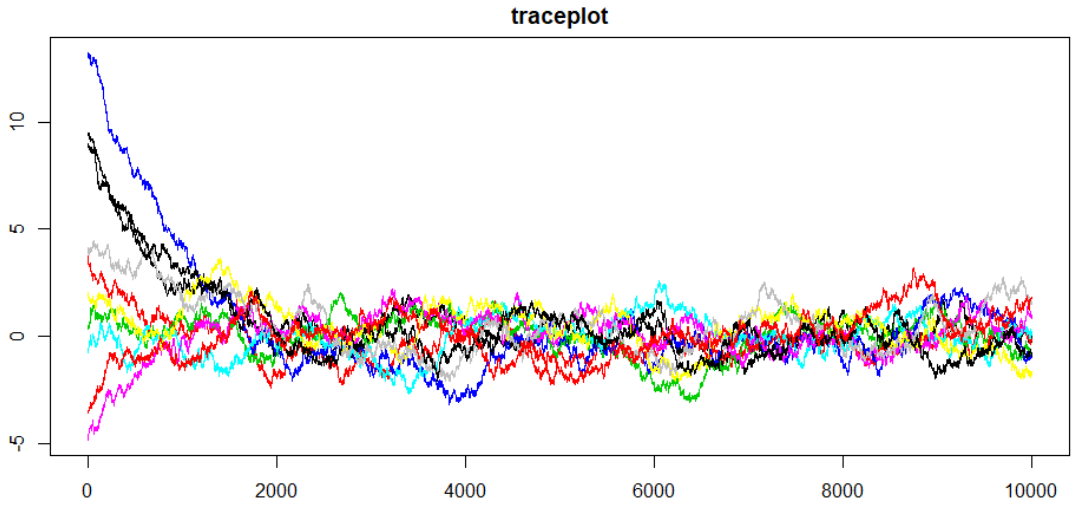
Since we have a distribution centered at zero, we would like to mostly move towards zero, which means making ε_t negative if X_t is positive and vice verse.

(c). No, if $R_t \geq 1$, you do not need to generate U_t.

(*d*). Implement m parallel runs of the algorithm when f is the standard normal. Use, for example, $m = 100$. [Hint: Using the above description of the algorithm, you can implement this in vectorized form in R.]

(*e*). Discuss ways of finding out how long you have to let the recursion last. Here are some possibilities: Plot the m parallel runs in a joint plot against the iteration number. Compute means and variances for the m simulations at iteration t. You may plot against t. You may also compare against the known mean and variance of the standard normal.

```
a = 5
acc = 0
for(i in 2:N)
{
  eps = runif(m,-a,a)
  R = dnorm(X[i-1,]+eps)/dnorm(X[i-1,])
  I = as.numeric(runif(m)<R)
  X[i,] = X[i-1,]+I*eps
  acc = acc + sum(I)
}
```



Just do it…

(f). Discuss strategies for starting the iteration. Possibilities are start at a fixed point and random start.

f) use multiple start points, (extreme values or random)

(g). Run your program. Start at some fixed points . Experiment with the choice of a. Find out when you can stop the iteration. Is there an optimal region for a?

(h). Repeat (g) for various values of the starting point. Find out whether good choices for a changes with the starting point. (The experiment is highly relevant. In complicated situations it may be unknown where to start.).

g,h) Do it

(i). It there scope for changing a during the iteration? Does that invalidate the algorithm?

i) Changing *a* you do not have detailed balance, since the reverse probability is different This must be accounted for