



# Monte Carlo Sampling Methods Using Markov Chains and Their Applications

Hastings, W. K., (1970)

Zhengxiao Wei

January 21 & 28, 2022

# Monte Carlo Method

**Monte Carlo methods** are a broad class of computational algorithms (method  $\neq$  algorithm) that rely on **repeated** random sampling to obtain numerical results, i.e., using **randomness** to solve problems that might be deterministic in principle.

The name “Monte Carlo” was chosen as a secret name for the nuclear weapons projects involving Nicholas Metropolis, Stanislaw Ulam, and John von Neumann at the Los Alamos National Laboratory in 1947. It was named after the Casino de Monte-Carlo in Monaco, where Ulam’s uncle used to gamble. Enrico Fermi also independently invented it amid the discovery of neutron-induced radioactivity in 1934.

**Exercise:** What are Las Vegas algorithm and Atlantic City algorithm?



N. Metropolis



S. Ulam



J. von Neumann



E. Fermi

Image: SlidePlayer

# Steps of Monte Carlo Methods

- 1 Define a domain of possible inputs.
- 2 Generate inputs randomly from a probability function over the domain.
- 3 Perform deterministic computation on the inputs,  
(one input - one output).
- 4 Aggregate (compile) the results.

Monte Carlo method depends on the **randomness** - more random inputs we take, better approximation we will get.

For example, illustrating the Monty Hall problem, computing a definite integral (the area of a graph), and approximating  $\pi$ .

# Example: Buffon's Needle Problem

[yihui.org/animation/example/buffon-needle/](http://yihui.org/animation/example/buffon-needle/)

Suppose we have a floor made of parallel strips of wood, each the same width  $t$ , and we drop a needle with a length  $l$  onto the floor. What is the probability  $p$  that the needle will lie across a line between two strips?

(Georges-Louis Leclerc, a.k.a. Comte de Buffon, 1777 since calculus has been well-defined)

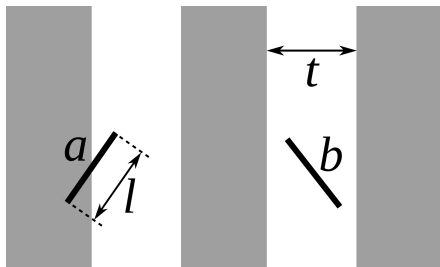


Image: Wikipedia

$$p = \frac{2l}{\pi t}$$

$$\hat{\pi} = \frac{2l}{\hat{p}t} = \frac{2l}{t} \frac{\text{number of trials}}{\text{number of hits}}$$



**Exercise:** Lazzerini (1901) claimed 1,808 hits out of 3,408 trials ( $\hat{\pi} \approx 3.1415929$ ). Is this possible?  $|\hat{\pi} - \pi| = O(1/\sqrt{n})$ .

# Monte Carlo Sampling: Inverse Transformation Method

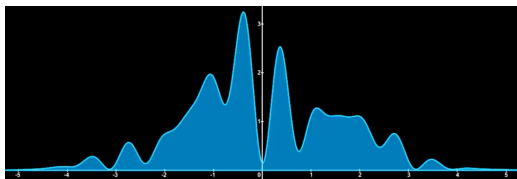
**Exercise:** Show the continuous  $F$  is the cumulative distribution function for  $F^{-1}(U)$ , where  $U \sim U[0, 1]$ .

Suppose we want to sample from  $f(x) = \frac{1}{2\sqrt{x}}e^{-\sqrt{x}}$ ,  $0 < x < \infty$ .

- 1 Derive the CDF,  $F(x) = 1 - e^{-\sqrt{x}}$ ,  $0 < x < \infty$ .
- 2 Derive the inverse function,  $F^{-1}(x) = (\ln(1 - x))^2$ .
- 3 Draw a value  $u$  from  $U[0, 1]$ .
- 4 Compute  $F^{-1}(u)$  as the desired sample.

# Monte Carlo Sampling: Acceptance-Rejection Method

R Script



Source: YouTube FunInCode

Independently sample from a  
(unnormalized)

**target** probability density function

$$\tilde{f}(x) = e^{-\frac{x^2}{5}} (3 \cos^2 x \cdot \sin^2 4x + 2 \sin^2(6+x)),$$

$$x \in [-5, 5]$$

- 1 Draw a value  $a$  from the **proposal** distribution  $q(x) \sim U[-5, 5]$  instead.
- 2 Draw a value  $b$  from  $U[0, Cq(a)]$ , where  $C = 33$  satisfies the “envelope”  
 $Cq(x) \geq \tilde{f}(x)$  for all  $x$ .
- 3 If  $\tilde{f}(a) \geq b$ , then  $a$  is accepted; otherwise, rejected.


**Question:** How can we improve the **efficiency** regarding the acceptance rate?

The algorithm will take an average of  $C$  iterations to obtain a sample.

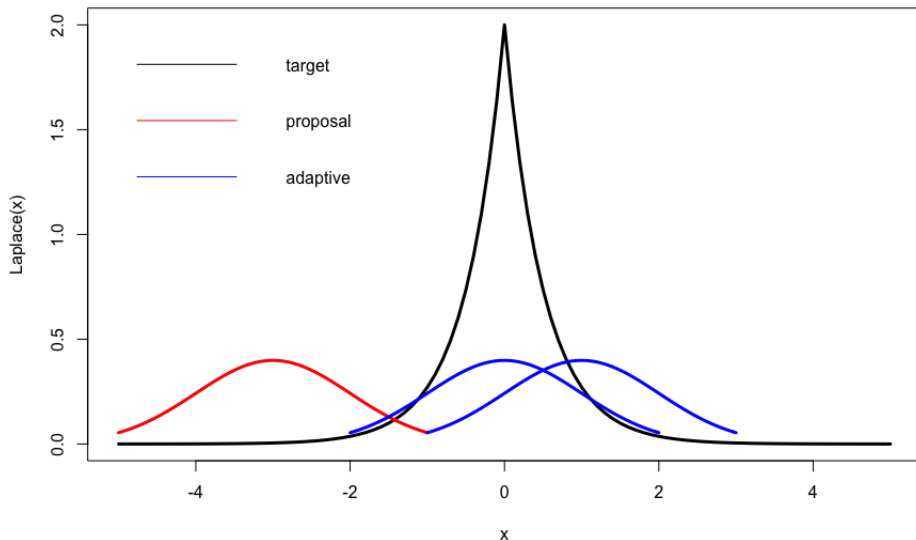
Try  $q(x) \sim \mathcal{N}(0, 1.6^2)$  and  $C = 14$ . The acceptance rate increases to 49% from 21%.

Promising?

# Objectives: Pseudo-Random Number Generator

- ▶ inverse transform sampling
  - needs the inverse of the cumulative distribution function (what if no analytical CDF?)  
polynomial approximation, e.g., the default method for `rnorm()` in 
  
- ▶ rejection sampling
  - introduces the proposal distribution (what if low efficiency?)
  
- ▽ adaptive rejection sampling (from a univariate log-concave PDF)
  - uses the Markov chain
  - Metropolis method (what if multivariate?)

# Adaptive Rejection Sampling





# Markov Chain

A **Markov chain** is a stochastic model describing a sequence of possible events in which the probability of each event depends **only** on the state attained in the previous event. It was named after Andrey Markov in 1877.

$$\mathbb{P}(X_{n+1} = x \mid X_n = x_n, \dots, X_1 = x_1) = \mathbb{P}(X_{n+1} = x \mid X_n = x_n)$$

Some fundamental concepts for the discrete-time Markov chain are random variable, state space, Markov property, the Chapman–Kolmogorov equation, class structure (transience, recurrence, and irreducibility), ergodicity, transition diagram and probability matrix, random walk, limiting probability and stationarity, **reversibility**, *etc.*

**Exercise:** Prove reversibility implies stationarity, but not the other way.

Detailed balance,  $\pi_i p_{ij} = \pi_j p_{ji}$  for all states  $i$  and  $j$ .



Image: Wikipedia

## Example: Google Page Rank

**PageRank** is an algorithm used by Google Search to rank web pages in their search engine results. It was developed by Larry Page and Sergey Brin in 1996.

The PageRank value of a page reflects the chance that the random surfer will land on that page by clicking on a link. It can be understood as a **non-reversible** Markov chain in which the states are pages, and the transitions are the links between pages – all of which are all equally probable.

It can be shown that the PageRank of a page is the probability of arriving at that page after a large number of clicks. This happens to be equal to the expectation of the number of clicks (or random jumps) required to get from the page back to itself.

# Markov Chain Integration

**Monte Carlo integration** is convenient when it is not possible to analytically compute a definite integral such as  $c \cdot \mathbb{E}[g(\theta)] = \int_{\Theta} g(\theta) \tilde{p}(\theta) d\theta$ .

**Exercise:** Show  $\int_{-\infty}^{\infty} x^3 e^{-\frac{x^2}{2}} dx = \sqrt{2\pi} \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n x_i^3$ . Hint: Simulate  $X_i \sim \mathcal{N}(0, 1)$ .

When it is not possible to directly sample from  $\tilde{p}(\cdot)$ ,

**Gibbs sampling** and **Metropolis-Hastings** stochastic algorithms can generate (Markovian dependent) samples  $\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(K)}$  from  $\tilde{p}(\cdot)$  that evaluated at  $g(\cdot)$  and averaged over the samples.

For a sufficiently large number of samples, the strong law of large numbers holds this average converges almost surely.

# Objectives: Markov Chain Monte Carlo (MCMC)

Consider a target PDF  $\pi$  and a state space  $x$ 's.

$$\underbrace{x^{(0)} \longrightarrow x^{(1)} \longrightarrow x^{(2)} \dots}_{\text{burn-in (warm-up) chain}} \longrightarrow x^{(n)} \longrightarrow x^{(n+1)} \longrightarrow x^{(n+2)} \dots$$

In general,  $\pi_i q_{ij} \neq \pi_j q_{ji}$ .

Find the acceptance rate  $\alpha$  such that  $\pi_i q_{ij} \alpha_{ij} = \pi_j q_{ji} \alpha_{ji}$ . Thus,  $\pi = \pi \mathbf{Q}'$ .

An easy solution is  $\alpha_{ij} = \pi_j q_{ji}$  and  $\alpha_{ji} = \pi_i q_{ij}$  (but  $\alpha$  may be low).

Let  $1 = C \cdot \alpha_{ji}$ . And,  $\tilde{f}_i$  is the unnormalized  $\pi_i$ .

- Metropolis algorithm gives a solution, when  $\mathbf{Q}'$  is symmetric,

$$\alpha_{ij} = \min \left\{ 1, \frac{\pi_j}{\pi_i} \right\} = \min \left\{ 1, \frac{\tilde{f}_j}{\tilde{f}_i} \right\}.$$

- Metropolis-Hastings algorithm gives a solution

$$\alpha_{ij} = \min \left\{ 1, \frac{\pi_j q_{ji}}{\pi_i q_{ij}} \right\} = \min \left\{ 1, \frac{\tilde{f}_j q_{ji}}{\tilde{f}_i q_{ij}} \right\}.$$

- Gibbs sampler works when  $\mathbf{Q}$  exists.

## Algorithm: Gibbs Sampling

Suppose that  $p(\boldsymbol{\theta} \mid \mathbf{y})$  is the joint posterior distribution (that has no closed-form expression) and its conditional distributions (that do) are

$$p(\theta_k \mid \theta_0, \dots, \theta_{k-1}, \theta_{k+1}, \dots, \theta_{d-1}, \mathbf{y}) \text{ for } k = 0, \dots, d-1.$$

Gibbs sampling is to sample from each conditional distribution of  $\theta_k$  in order to obtain samples from the joint posterior distribution. (Used to be a special case of the Metropolis–Hastings algorithm, 1984)

- 1 Initialize  $t = 1$  and define initial values  $\theta_0^{(0)}, \theta_1^{(0)}, \dots, \theta_{d-1}^{(0)}$  for the vector  $\boldsymbol{\theta} = (\theta_0, \theta_1, \dots, \theta_{d-1})^\top$ .

- 2 Sample from the conditional distributions:

$$\theta_0^{(t)} \sim p(\theta_0 \mid \theta_1^{(t-1)}, \theta_2^{(t-1)}, \theta_3^{(t-1)}, \dots, \theta_{d-1}^{(t-1)}, \mathbf{y}).$$

$$\theta_1^{(t)} \sim p(\theta_1 \mid \theta_0^{(t)}, \theta_2^{(t-1)}, \theta_3^{(t-1)}, \dots, \theta_{d-1}^{(t-1)}, \mathbf{y}).$$

$$\vdots$$

$$\theta_{d-1}^{(t)} \sim p(\theta_{d-1} \mid \theta_1^{(t)}, \theta_2^{(t)}, \theta_3^{(t)}, \dots, \theta_{d-2}^{(t)}, \mathbf{y}).$$

- 3 Take  $t = t + 1$  and return to Step 2 until the desired  $t$ .

# Algorithm: Metropolis-Hastings

Metropolis-Hastings may be more flexible than Gibbs sampling, since the former can generate samples from conditional distributions that have or have **not** closed-form expressions.

Metropolis-Hastings is based on a proposal distribution  $q(\boldsymbol{\theta}^* | \boldsymbol{\theta}^{(t-1)})$  (should have the appropriate support of  $\boldsymbol{\theta}$ ), which generates candidate values  $\boldsymbol{\theta}^*$  that are accepted as values from  $p(\boldsymbol{\theta} | \mathbf{y})$  with certain probability. Metropolis et al. (1953) proposed **symmetric**  $q(\boldsymbol{\theta}^* | \boldsymbol{\theta}^{(t-1)})$  as a normal distribution where the mean has a Markovian structure and the variance is constant. Hastings (1970) developed a **general framework** in terms of the covariance matrix.

(Cited more than 47,000 and 17,700 times respectively)

**New Step 2:** Sample candidate  $\boldsymbol{\theta}^*$  from  $q(\boldsymbol{\theta}^* | \boldsymbol{\theta}^{(t-1)})$ :

(i) Compute a ratio  $\alpha = \min \left\{ 1, \frac{p(\boldsymbol{\theta}^* | \mathbf{y})}{p(\boldsymbol{\theta}^{(t-1)} | \mathbf{y})} \frac{q(\boldsymbol{\theta}^{(t-1)} | \boldsymbol{\theta}^*)}{q(\boldsymbol{\theta}^* | \boldsymbol{\theta}^{(t-1)})} \right\}.$

(ii) Draw  $u \sim U[0, 1]$ .

If  $\alpha \geq u$ , then add  $\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^*$ ; otherwise, retain  $\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(t-1)}$ .

# Application in Bayesian Linear Regression

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \text{ where } \boldsymbol{\epsilon} \sim \mathcal{N}_n(\mathbf{0}, \sigma^2 \mathbf{I}) \quad (1)$$

The likelihood function is

$$f(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\beta}, \sigma^2) = (2\pi\sigma^2)^{-\frac{n}{2}} \exp \left\{ -\frac{1}{2\sigma^2} (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \right\} \quad (2)$$

The joint posterior distribution for  $\boldsymbol{\beta}$  and  $\sigma^2$  is

$$f(\boldsymbol{\beta}, \sigma^2 \mid \mathbf{y}, \mathbf{X}) \propto f(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\beta}, \sigma^2) \cdot p(\boldsymbol{\beta} \mid \sigma^2) \cdot p(\sigma^2) \quad (3)$$

## Procedures

- 1 Specify priors for the model parameters,  $p(\boldsymbol{\beta} \mid \sigma^2)$  and  $p(\sigma^2)$ .
- 2 Create a model mapping the training inputs to the training outputs.
- 3 Apply an MCMC algorithm to draw samples from the posterior distributions (that may have no closed-form expressions) for the parameters.

## Application in Bayesian Linear Regression (continued)

Consider  $\sigma$  known and a Laplace distribution as (non-conjugate) prior for each  $\beta_i$ , i.e.,

$$f(\beta_i | m_i, v_i) = \frac{1}{2v_i} \exp \left\{ -\frac{|\beta_i - m_i|}{v_i} \right\}, \quad v_i > 0. \quad (4)$$

The posterior distribution (that has no closed form of a known PDF) will be

$$\begin{aligned} f(\boldsymbol{\beta} | \mathbf{y}, \mathbf{X}, \sigma^2) &\propto f(\mathbf{y} | \mathbf{X}, \boldsymbol{\beta}) \prod_{i=1}^d f(\beta_i | m_i, v_i) \\ &\propto \exp \left\{ -\frac{1}{2\sigma^2} (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) - \sum_{i=1}^d \frac{|\beta_i - m_i|}{v_i} \right\}. \end{aligned} \quad (5)$$

**Question:** Which algorithm do we implement here?  
Gibbs sampling or Metropolis-Hastings?



# In Memoriam: Wilfred Keith Hastings

[probability.ca/hastings/](http://probability.ca/hastings/)



In 1971, Hastings joined the department of mathematics at the University of Victoria as an associate professor and was granted tenure there in 1974. He taught for 21 years and retired in 1992. He has passed away peacefully on May 13, 2016 at the age of 85 in Victoria.

*He had a great sense of humour and wrote jokes and limericks for people. He would be laughing with whomever he shared his jokes, which would make you laugh even more.*

—— Rachel, Daughter

# In Memoriam: Hastings

[mccallgardens.com/obituaries/wilfred-keith-hastings](http://mccallgardens.com/obituaries/wilfred-keith-hastings)

*It wasn't until much later, with the advancement of computing power, that the importance of the method was fully appreciated. It has become a cornerstone in Bayesian methodology, where numerical computations of properties of posterior distributions are required. It is probably no exaggeration to say that the Metropolis-Hastings algorithm, and related methods, led to the current day revival and success of Bayesian methods.*

—— William J. Reed, Ph.D., UVic Professor Emeritus

## Computational Cost and Parallel Computing

*To estimate parameters using Monte Carlo methods, we have to generate a lot of random numbers, so it takes a considerable amount of time. It might seem alien to young people nowadays, but back then computer CPUs were still at the Intel i386 stage, meaning that they struggled to perform calculations with any speed. Fortunately, Chris (Nobel Prize winner 2011) allowed me to use the lab next to his own, which was equipped with an i486 33MHz computer, the fastest available back then. Even so, estimating SV (stochastic volatility) models still took the best part of a week. With Chris' help however, I managed to finish my dissertation and successfully completed my PhD. I felt like giving up on Monte Carlo methods after that, because it just took too long. As computers have become significantly faster however (the time taken to estimate SV models has gone **from nearly a week to around 15 minutes**), I still use Monte Carlo methods to this day. I don't know what sort of research I would be working on if I hadn't met Chris. He helped to make me the researcher I am today.*

—— Toshiaki Watanabe, Ph.D., Professor & Econometrician

# Summary

A generalization of the sampling method introduced by Metropolis et al. (1953) is presented along with an exposition of the relevant theory, techniques of application and methods and difficulties of assessing the error in Monte Carlo estimates. Examples of the methods, including the generation of random orthogonal matrices and potential applications of the methods to numerical problems arising in statistics, are discussed.

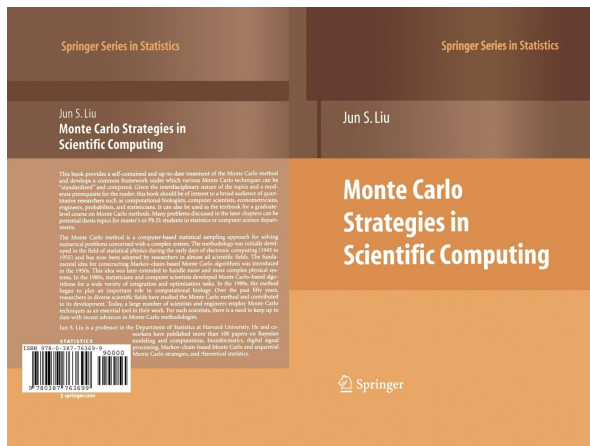
## Further Reading

- Convergence diagnostics using the traceplots (“hairy caterpillar”) and  $\hat{R}$  (1.05).
- Efficiency diagnostics using the effective sample size (ESS) and maximum tree-depth.
- Software such as JAGS and JASP;  
R packages such as *brms*, *BayesFactor*, *rstan*, and *rjags*.
- Other newly developed algorithms such as Hamiltonian Monte Carlo (1987) and No-U-Turn Sampler (2011), which are implemented in Stan.

# Recommended Book

ISBN: 978-0-387-76371-2

Jun S. Liu (刘军) received the COPSS Presidents' Award (known as the "Nobel Prize in Statistics") in 2002. He is a professor in the Department of Statistics at Harvard University and has written many research papers and a book about MCMC algorithms, including their applications in biology (Gibbs sampling for finding functional fragments with high similarity in DNA).



# References

- Ali, A., Inglis, A., Prado, E., & Wundervald, B. (2020). Bayesian linear regression.  
<https://brunaw.com/phd/bayes-regression/report.pdf>
- Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2013).  
Bayesian data analysis (3rd ed.). London, UK: Chapman and Hall. doi: 10.1201/b16018.
- Geman, S., Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian  
restoration of images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 6, 721–741.  
doi:10.1109/TPAMI.1984.4767596.
- Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their  
applications. *Biometrika*, 57, 97-109. doi:10.2307/2334940.
- Lazzerini, M. (1901). Un applicazione del calcolo della probabilita alla ricerca di un valore  
approssimato di. *Periodico di Matematiche*, 4, 140-143.
- Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953).  
Equation of state calculations by fast computing machines. *J. Chem. Phys.*, 21,  
1087-1092. doi:10.1063/1.1699114.
- Watanabe, T. (2011). Professor Christopher Sims wins Nobel Prize for Economics. *Hi-Stat  
Vox*, 21, <https://voxeu.org/article/why-christopher-sims-won-nobel-prize>.

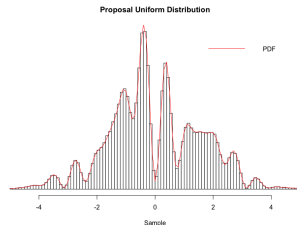
Thank you.

Contact: [zhengxiao@uvic.ca](mailto:zhengxiao@uvic.ca)

## R Script

BACK

```
f <- function(x) { #unnormalized target PDF
  exp(-x^2/5) * (3*cos(x)^2*sin(4*x)^2 + 2*sin(6+x)^2)
}
NSim <- 1e6; x <- rep(-10, NSim); set.seed(277)
for (i in 1:NSim) {
  a <- runif(1, -5, 5)
  b <- runif(1, 0, 3.3)
  if (f(a)>=b) {x[i]=a}
}
(rate <- sum(x>-10) / NSim) #0.2096
hist(x[x>-10], breaks=100, prob=T, col="white",
     yaxt="n", ylab="", main="Proposal Uniform Distribution", xlab="Sample")
curve(f(x)/(33*rate), from=-5, to=5, add=T, col="red")
legend("topright", "PDF", col="red", lty=1, bty="n")
```





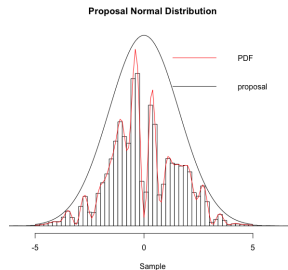
## R Script

BACK

```

y <- rep(-10, NSim); set.seed(277)
for (i in 1:NSim) {
  a <- rnorm(1, 0, 1.6)
  b <- runif(1, 0, 14*dnorm(a, 0, 1.6))
  if (f(a)>=b) {y[i]=a}
}
(rate2 <- sum(y>-10) / NSim) #0.494906
hist(y[y>-10], breaks=100, prob=T, col="white",
     yaxt="n", ylab="", main="Proposal Normal Distribution", xlab="Sample", ylim=c(0, 0.5))
curve(f(x)/(33*rate), from=-5, to=5, add=T, col="red")
curve(dnorm(x, 0, 1.6)*14/(33*rate), from=-5, to=5, add=T)
legend("topright", c("PDF", "proposal"), col=c("red", "black"), lty=c(1,1), bty="n")

```



## R Script

BACK

```

q <- function(x) {
  value <- rnorm(1, x, 0.2) #tuning sd
  if (value > 5) {value = 5} else if (value < -5) {value = -5}
  value
}
z <- numeric(1e5); z[1] <- 1; set.seed(277)
for (i in 1:99999) {
  zj <- q(z[i])
  A <- min(1, f(zj)/f(z[i]))
  if (A > runif(1)) {
    z[i+1] <- zj
  } else {
    z[i+1] <- z[i]
  }
}
hist(z, breaks=100, prob=T, col="white",
     yaxt="n", ylab="", main="Metropolis Method", xlab="Sample", ylim=c(0, 0.46))
curve(f(x)/(33*rate), from=-5, to=5, add=T, col="red")
legend("topright", "PDF", col="red", lty=1, bty="n")

```

