Markov Chain Monte Carlo
Metropolis Hastings & Gibbs Sampler

Aymen Rumi

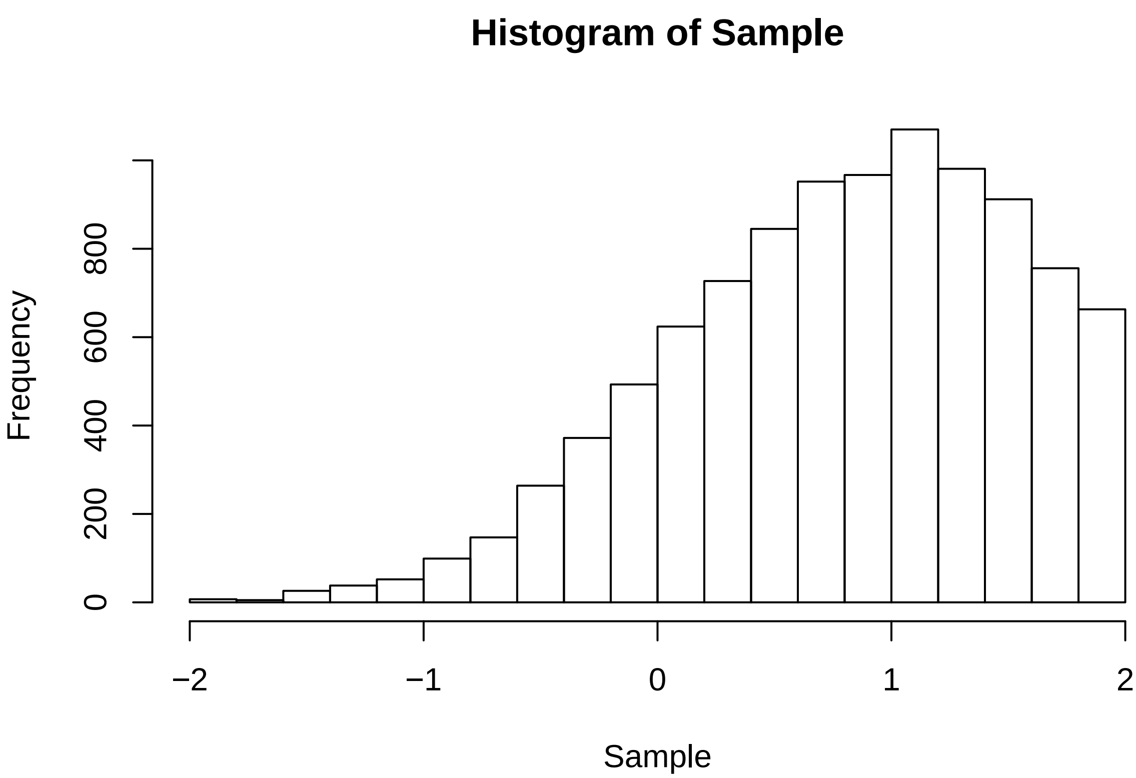
Function For Metropolis Hastings Algorithm & Target Distribution

```
# Metropolis Algorithm given target distribution

MetropolisAlgorithm<-function(target,lower_limit,upper_limit,sigma,n)
{
    X<-rep(0,n)
    X[1]<-runif(1, lower_limit, upper_limit)

    for (istep in 2:n)
    {
        x_t0<-X[istep-1]

        #sample from proposal density
        x_t1<-rnorm(1,x_t0,sigma)

        if(x_t1<lower_limit)
        {

            x_t1=abs(x_t1)
        }

        if(x_t1>upper_limit)
        {
            difference<-x_t1-upper_limit
            x_t1=upper_limit-difference
        }

        #calculate the acceptance probability

        al=min(1,(target(x_t1)/target(x_t0)))
        u<-runif(1)

        if(u<al)
        {
            X[istep]<-x_t1
        }
        else
        {
            X[istep]<-x_t0
```

```
        }
    }
    return (X)
}


# target distribution

target_distribution<-function(x)
{
    exp(-((x-1)^2)/2)-exp(-((x-4)^2)/2)
}
```

Sampling from Target Dsitribution & Plotting Distribution

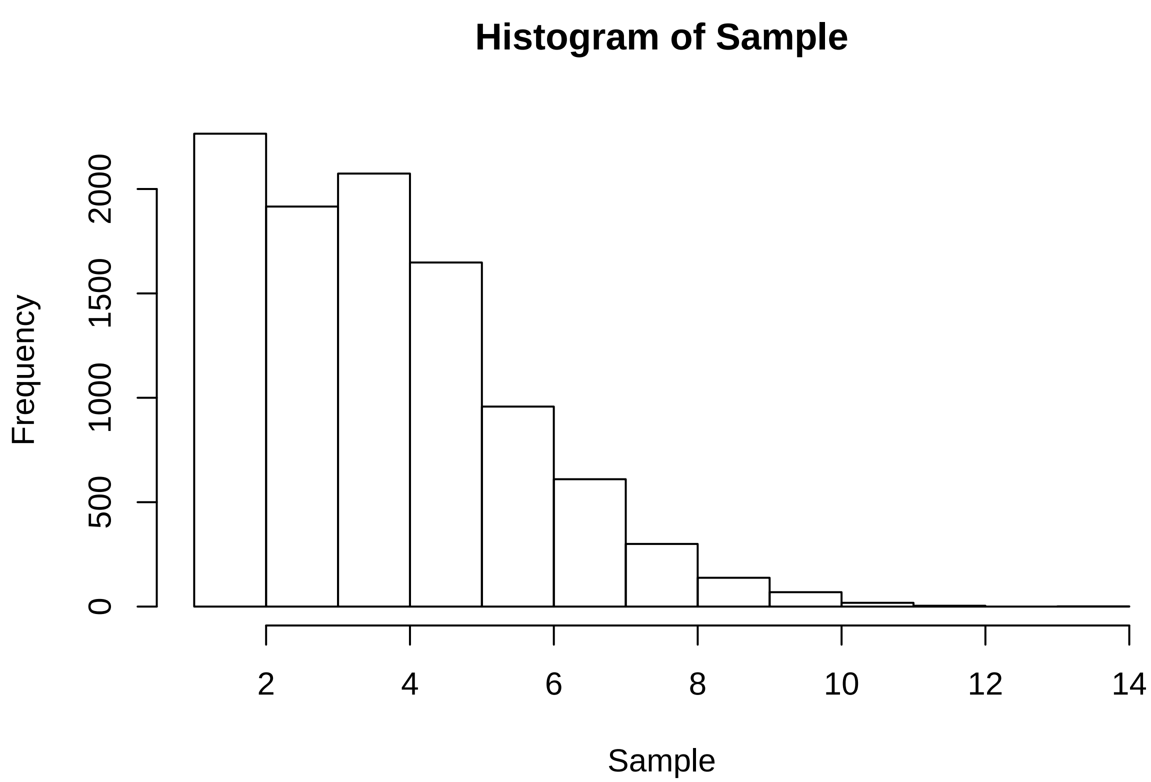
```
Sample<-MetropolisAlgorithm(target=target_distribution,lower_limit = -2,upper_limit = 2,sigma = 1,n=100

hist(Sample)
```



Metropolis Hastings for Sampling from Geometric Distribution

```
Metropolis_Hastings<-function(N,lambda,p)
{
```

```
    xt_0 = rgeom(1,1/3) + 1
    X = c(xt_0,rep(NA,N))

    for(i in 2:(N+1)){
        xt_1 = rgeom(1,1/3)+1
        log_accept = lfactorial(xt_0) - lfactorial(xt_1) +
        (xt_1-xt_0)*log(lambda) + (xt_0-xt_1)*log(1-p)

        if(runif(1) < exp(log_accept)){
            xt_0 <- xt_1
        }

        X[i] = xt_0


    }

    return (X)

}

# sampling with specified parameters
Sample<-Metropolis_Hastings(10000,3,0.5)

hist(Sample)
```



Gibbs Sampling: Sampling from Multidimentional Distribution

```r
# Conditional Distributions
x_cond_y<-function(y,u_x,u_y,sigma_x,sigma_y,rho)
{
    conditional_mean<-u_x+(rho*(sigma_x/sigma_y)*(y-u_y))
    conditional_var<-(sigma_x^2)*((1-rho)^2)
    rnorm(1,conditional_mean,conditional_var)
}



y_cond_x<-function(x,u_x,u_y,sigma_x,sigma_y,rho)
{
    conditional_mean<-u_y+(rho*(sigma_y/sigma_x)*(x-u_x))
    conditional_var<-(sigma_y^2)*((1-rho)^2)
    rnorm(1,conditional_mean,conditional_var)
}


# Gibbs Sampling for Bivariate Normal Distribution

GibbsSampler_BivariateNormal<-function(u_x,u_y,sigma_x,sigma_y,rho,N)
{
    mat <- matrix(ncol = 2, nrow = N)
    x <- u_x
    y <- u_y

    mat[1, ] <- c(x, y)

    for(i in 2:N)
    {
        x<-x_cond_y(y,u_x,u_y,sigma_x,sigma_y,rho)
        y<-y_cond_x(x,u_x,u_y,sigma_x,sigma_y,rho)

        mat[i,]<-c(x,y)
    }

    return (mat)
}


# Plotting Samples for multidimentional distribution
Plot_Samples<-function(samples)
{
    par(mfrow=c(3,2))
    plot(samples,col=1:length(samples))
    plot(samples,type="l")
    plot(ts(samples[,1]))
    plot(ts(samples[,2]))
    hist(samples[,1],40)
    hist(samples[,2],40)
    par(mfrow=c(1,1))
```

```
}


Sample<-GibbsSampler_BivariateNormal(0,3,2,0.5,0.5,10000)
Plot_Samples(Sample)
```