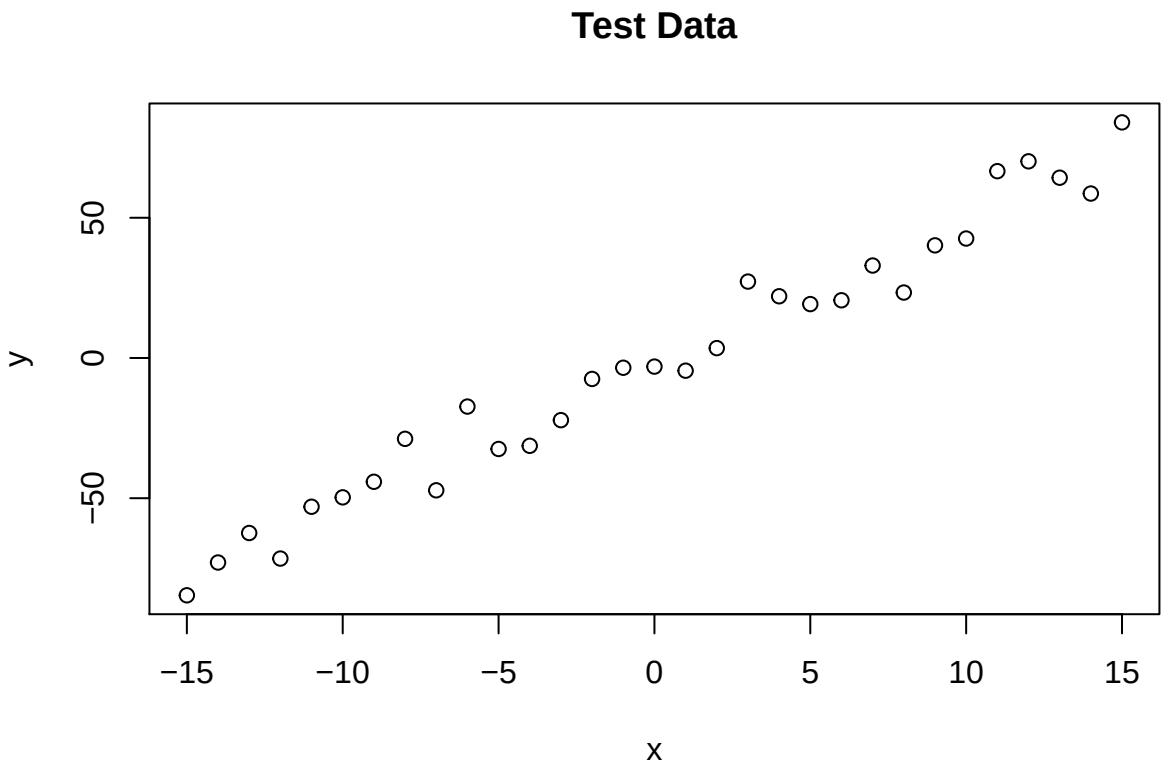
Random-walk Metropolis-Hastings example from A simple Metropolis-Hastings MCMC in R, Florian Hartig, September 17 2010, copied from http://theoreticalecology.wordpress.com/2010/09/17/metropolis-hastings-mcmc-in-r/

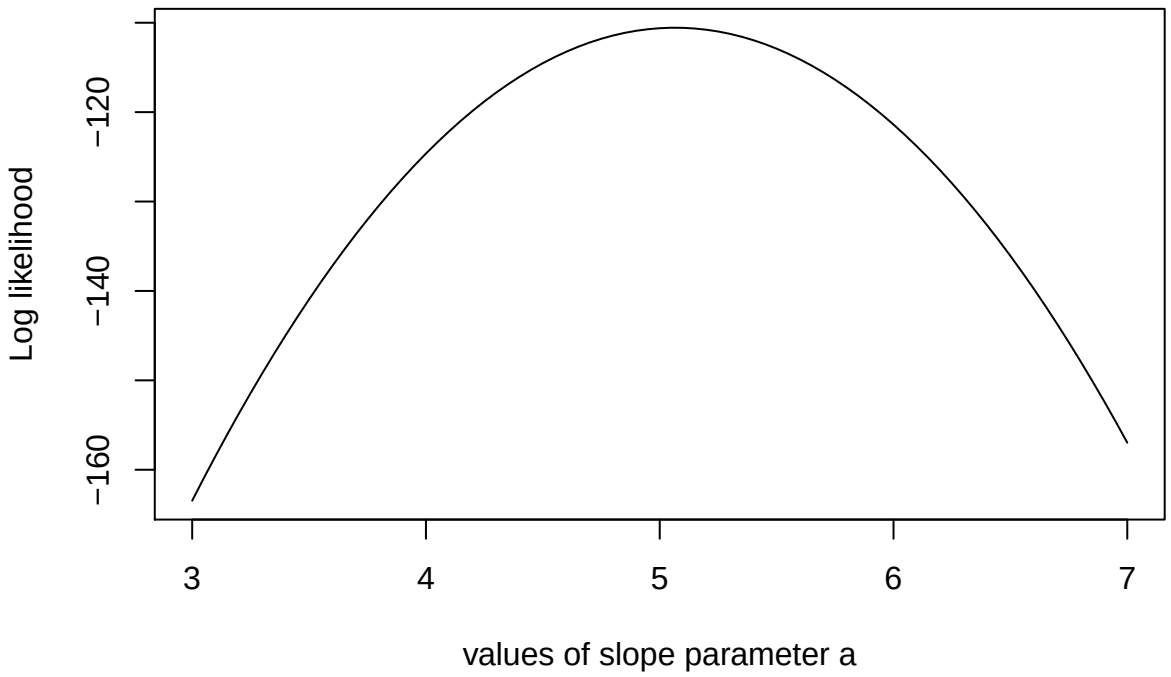
Creating test data

```
trueA <- 5
trueB <- 0
trueSd <- 10
sampleSize <- 31

set.seed(3)
# create independent x-values
x <- (-(sampleSize-1)/2):((sampleSize-1)/2)
# create dependent values according to ax + b + N(0,sd)
y <-  trueA * x + trueB + rnorm(n=sampleSize,mean=0,sd=trueSd)

plot(x,y, main="Test Data")
```



Defining the statistical model and Derive the likelihood function from the model

```
likelihood <- function(param){
    a = param[1]
    b = param[2]
    sd = param[3]

    pred = a*x + b
    singlelikelihoods = dnorm(y, mean = pred, sd = sd, log = T)
    sumll = sum(singlelikelihoods)
    return(sumll)
}

# Example: plot the likelihood profile of the slope a
slopevalues <- function(x){return(likelihood(c(x, trueB, trueSd)))}
slopelikelihoods <- lapply(seq(3, 7, by=.05), slopevalues )
plot (seq(3, 7, by=.05), slopelikelihoods , type="l", xlab = "values of slope parameter a", ylab = "Log
```



Defining the prior

```
# Prior distribution
prior <- function(param){
    a = param[1]
    b = param[2]
    sd = param[3]
    aprior = dunif(a, min=0, max=10, log = T)
```

```
    bprior = dnorm(b, sd = 5, log = T)
    sdprior = dunif(sd, min=0, max=30, log = T)
    return(aprior+bprior+sdprior)
}
```

The posterior

```
posterior <- function(param){
   return (likelihood(param) + prior(param))
}
```

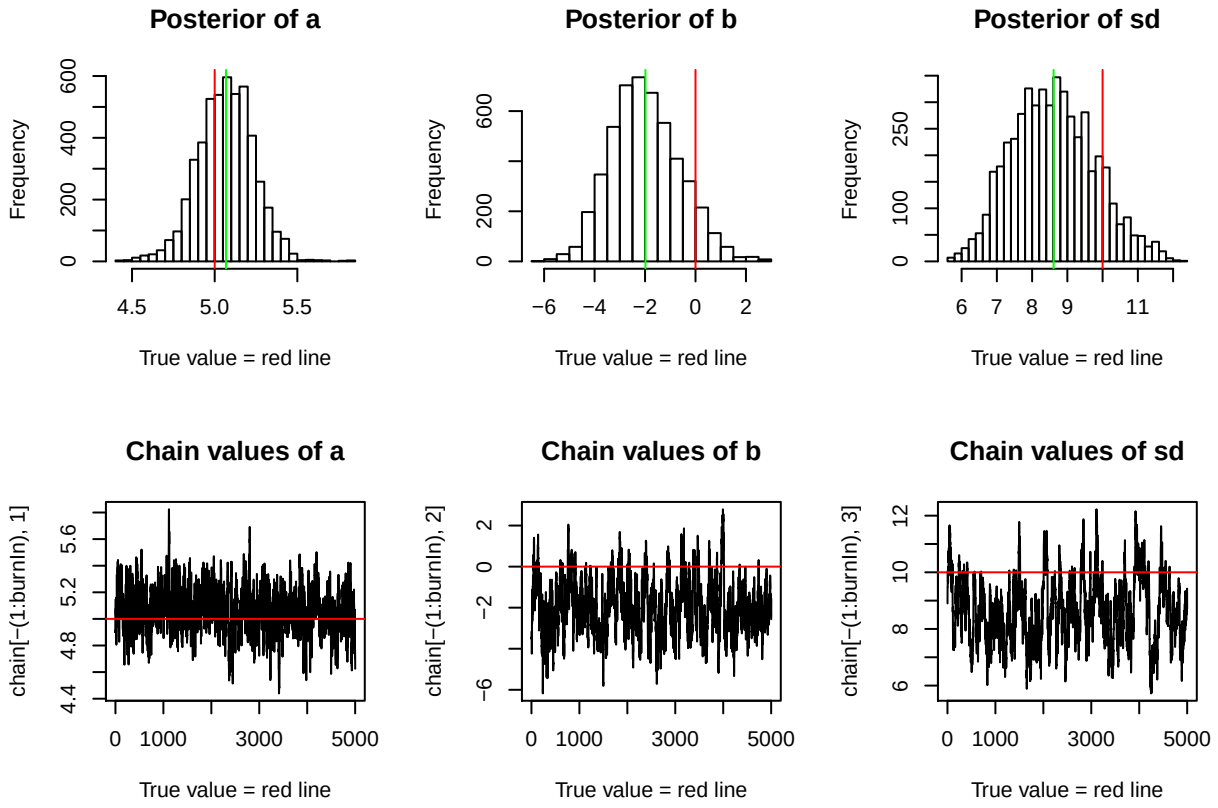
The MCMC

```
######## Metropolis algorithm #################

proposalfunction <- function(param){
    return(rnorm(3,mean = param, sd= c(0.1,0.5,0.3)))
}

run_metropolis_MCMC <- function(startvalue, iterations){
    chain = array(dim = c(iterations+1,3))
    chain[1,] = startvalue
    for (i in 1:iterations){
        proposal = proposalfunction(chain[i,])

        probab = exp(posterior(proposal) - posterior(chain[i,]))
        if (runif(1) < probab){
            chain[i+1,] = proposal
        }else{
            chain[i+1,] = chain[i,]
        }
    }
    return(chain)
}

set.seed(1)
startvalue = c(4,0,10)
chain = run_metropolis_MCMC(startvalue, 10000)

burnIn = 5000
acceptance = 1-mean(duplicated(chain[-(1:burnIn),]))
acceptance
```

```
## [1] 0.7546491
```

```
### Summary: ########################
```

```
par(mfrow = c(2,3))
hist(chain[-(1:burnIn),1],nclass=30, , main="Posterior of a", xlab="True value = red line" )
abline(v = mean(chain[-(1:burnIn),1]), col="green")
abline(v = trueA, col="red" )
hist(chain[-(1:burnIn),2],nclass=30, main="Posterior of b", xlab="True value = red line")
abline(v = mean(chain[-(1:burnIn),2]), col="green")
abline(v = trueB, col="red" )
hist(chain[-(1:burnIn),3],nclass=30, main="Posterior of sd", xlab="True value = red line")
abline(v = mean(chain[-(1:burnIn),3]), col="green" )
abline(v = trueSd, col="red" )
plot(chain[-(1:burnIn),1], type = "l", xlab="True value = red line" , main = "Chain values of a", )
abline(h = trueA, col="red" )
plot(chain[-(1:burnIn),2], type = "l", xlab="True value = red line" , main = "Chain values of b", )
abline(h = trueB, col="red" )
plot(chain[-(1:burnIn),3], type = "l", xlab="True value = red line" , main = "Chain values of sd", )
abline(h = trueSd, col="red" )
```




```
# for comparison:
summary(lm(y~x))
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.2214  -6.3852  -0.5285   4.6336  15.0324
```

```
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.9664     1.4985  -1.312      0.2
## x             5.0654     0.1675  30.234   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.343 on 29 degrees of freedom
## Multiple R-squared:  0.9693, Adjusted R-squared:  0.9682
## F-statistic: 914.1 on 1 and 29 DF,  p-value: < 2.2e-16
```