Metropolis-Hastings example

Florian Hartig

This example has been taken from this blog post.

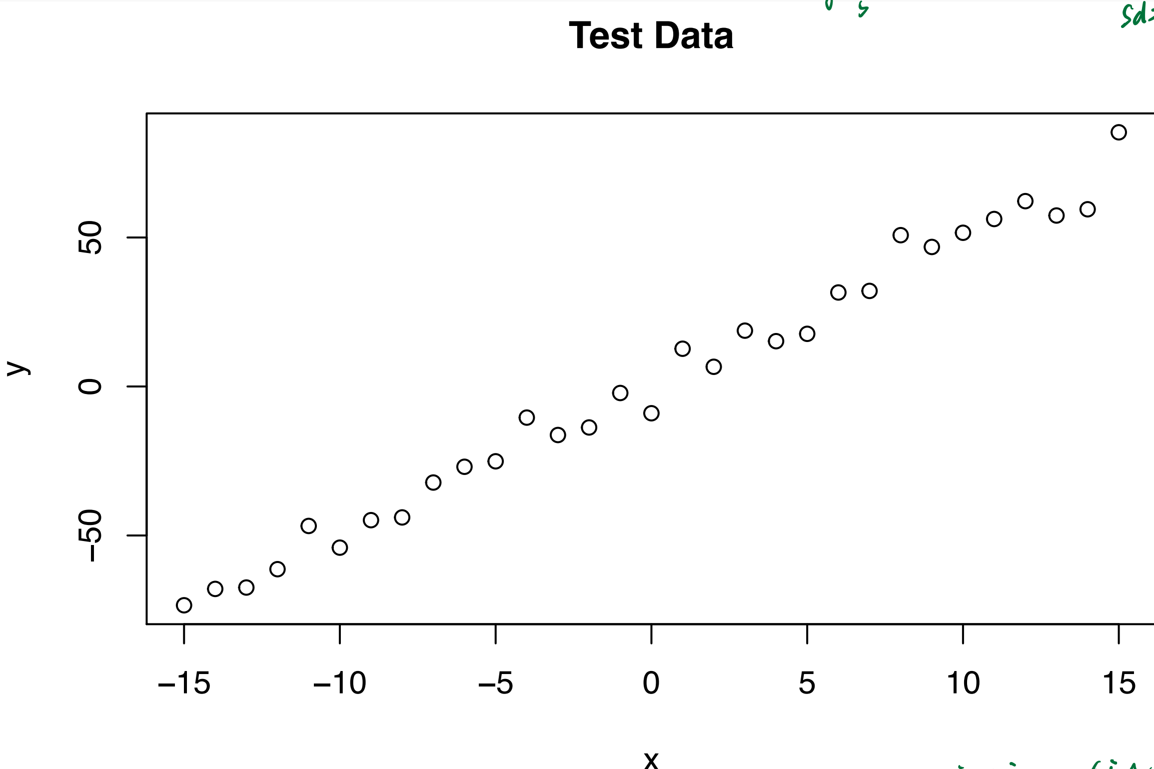
simulate test data

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

set.seed(1500)
# 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")
```

$x: -15:15$

$-15, -14, \cdots, 14, 15$

$y = \underbrace{b}_{0} + \underbrace{a}_{5}x + \varepsilon_i$ $\quad \varepsilon_i \sim N(0, \underbrace{sd^2}_{sd=5})$



$y_i = b + a x_i + \varepsilon_i \quad \varepsilon_i \sim N(0, sd^2).$

bayesian approach

prior : $a \sim U(0,10)$

$b \sim N(0, 5^2)$

$sd \sim U(0,10)$

want to find $P(a, b, sd \mid D)$

proposal density $q \Rightarrow \theta = (a, b, sd) \to \theta' = (a', b', sd')$

$a' \sim N(a, (0.1)^2)$ $\quad b' \sim N(b, (0.5)^2)$ $\quad sd' \sim N(sd, (0.3)^2)$.

$D = (D_1, D_2, \cdots D_{31})$ and $D_i = (x_i, y_i)$

efficiency of MH depend on proposal density q

Implementing MH algorithm

```r
######## MH algorithm ################
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)
}

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

# evaluate log posterior at given parameter values
posterior <- function(param){
   return (likelihood(param) + prior(param))
}

# evaluate log prior at given parameter values
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=10, log = T)
    return(aprior+bprior+sdprior)
}

# evaluate log likelihood at given parameter values
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)
}
```

a b sd

t=0 → initial value

t=1

⋮

t=M

↓ # iterations

θ' $\quad \theta$

$\frac{\pi(a', b', sd')}{\pi(a, b, sd)}$.

$\exp(\log \pi(a', b', sd') - \log(a, b, sd))$

AP = min{1, probab}

if probab > 1 ⇒ AP = 1 ⇒ θ'

if probab < 1 ⇒ AP = probab { wp probab θ'; wp 1-probab θ

① if probab > 1, then since $U \in (0,1)$ so probab > U

② if probab < 1 then w.p $P(U < \text{probab}) \Rightarrow \theta'$ otherwise ⇒ θ

⇒ simulate U $U \sim U(0,1)$.

U < probab ⇒ take proposal θ'.

U > probab ⇒ take θ'

independent prior

$\pi(a, b, sd) = p(a, b, sd)\, p(D \mid a, b, sd)$.

$\log \pi(a, b, sd) = \boxed{\log p(a) + \log p(b) + \log p(sd)}$

$+ \log \prod_{i=1}^{31} p(D_i \mid a, b, sd)$

$\boxed{\sum_{i=1}^{31} \log p(D_i \mid a, b, sd)}$

← log prior

Run MH algorithm

```
set.seed(1)
# initial value
startvalue = c(4,0,4)
# simulate 10000 samples
chain = run_metropolis_MCMC(startvalue, 10000)

# remove the first 5000 as burn-in
burnIn = 5000

# computing average acceptance probability
acceptance = 1-mean(duplicated(chain[-(1:burnIn),]))
acceptance
```

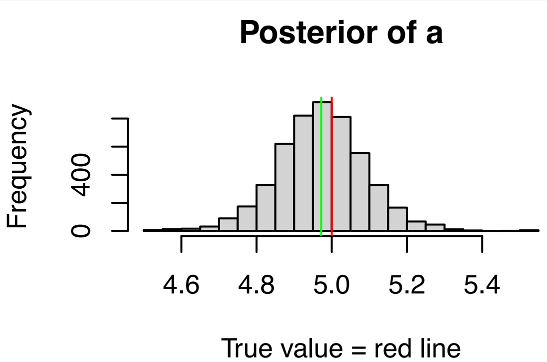
proportion of iteration value is equivalent to previous iteration

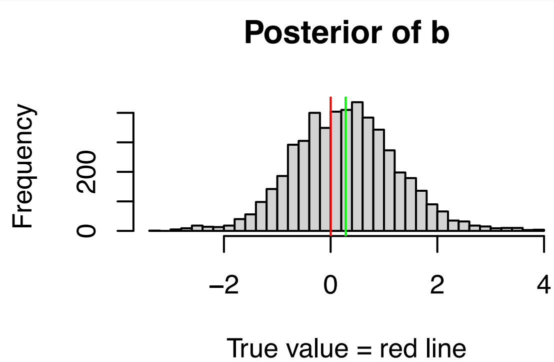
⇒ mean we take θ rather than θ'

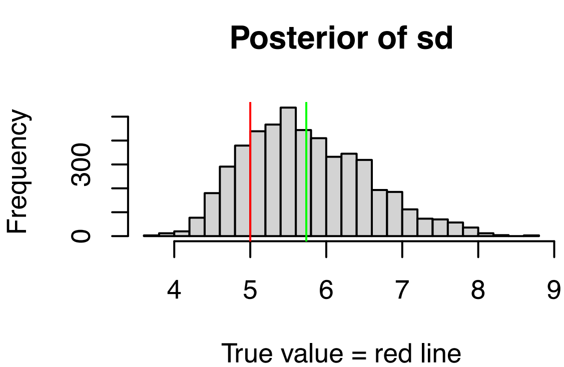
```
## [1] 0.6414717
```

> 0.2

```
par(mfrow = c(2,2))
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" )
```








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

frequency approach

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.3580  -3.8445   0.8254   2.4071  10.7585
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.2710     0.9989   0.271    0.788
## x             4.9678     0.1117  44.482   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.562 on 29 degrees of freedom
## Multiple R-squared:  0.9856, Adjusted R-squared:  0.9851
## F-statistic:  1979 on 1 and 29 DF,  p-value: < 2.2e-16
```