An analysis on the Metropolis-Hastings algorithm for various iterations

Kenneth Ruiter (12206909)

October 19, 2018

Contents

1 Introduction

In statistics, being able to draw from any distribution you like is a very useful tool for analysing your data. Aside from just the 'usual' distributions, some user-defined ones can be hard to sample from directly, when their density is somewhat odd. A well-known way of solving this problem, is by using the Metropolis-Hastings algorithm. This algorithm is a so-called MCMC (Markov Chain Monte Carlo) method to get random samples from such a difficult distribution. In this report we will focus on a particular example of the Metropolis-Hastinds algorithm. We will use the algorithm to estimate the parameters of a simple Ordinary Least Squares regression model, as to easily compare them with the analyitically computed values of the estimates. We will focus on the accuracy of the algorithm by comparing the outputs for a different number of iterations used to compute our estimate.

1.1 Overview of the report

In Section 2, we will give more elaborate discription of the Metropolis-Hastings algorithm. We will not explain why it works, however we will go through the algorithm step-by-step. Afterwards, we will randomly generate the data and define some parameters in Section 3. Then the results of the algorithm will be shown and discussed in Section 4. Finally, a short conclusion will be given in Section 5.

2 Algorithm Description

In this section, the steps involved in the Metropolis-Hastings algorithm will be discussed, using pseudo-code. For this, let $f(x)$ be the density of the distribution that we want to sample from, also known as the target distribution. We will need an initial draw from our sample, so we arbitrarily take x_0. We also need an arbitrary second distribution, that we can easily sample from, with density $g(x)$. This distribution will suggest a new candidate for the following draws. We usually take a symmetric distribution, like the Normal distribution, because it makes it easy to compute the acceptance rate of the candidates. This distribution is ususally called the jumping distribution, or sometimes the proposal distribution. The actual steps of the algorithm are shown now.

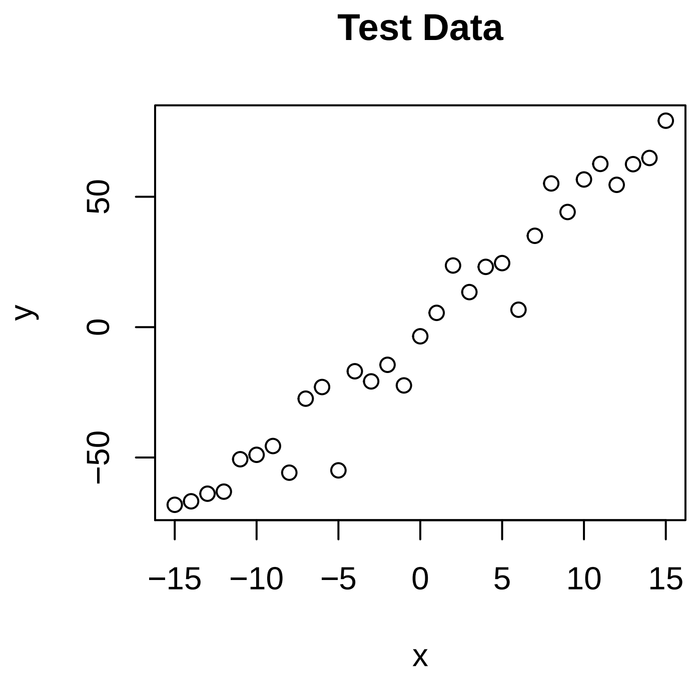
Algorithm 1: The Metropolis-Hastings algorithm

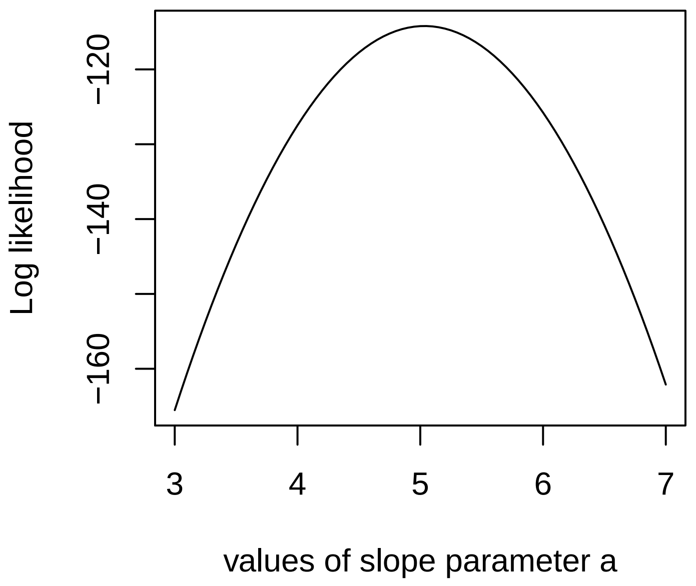
```
1  function run_metropolis_MCMC ($f, g, x_0, iterations$);
   Input  : Density function of target distribution $f$, density function of jumping
            distribution $g$, initial draw $x_0$, number of iterations $iterations$.
   Output: A vector of the ordered accepted draws, representing draws from the target
            distribution.
2  if $intdist$ is $exponential$ then
3      Set number of random interarrival times $N$ to $T * intarrpar1$;
4      Randomly generate $N$ interarrival times using an exponential distribution with
         parameter $intarrpar1$;
5  else if $intdist$ is $gamma$ then
6      Set number of random interarrival times $N$ to $T * intarrpar2/intarrpar1$;
7      Randomly generate $N$ interarrival times using a gamma distribution with parameters
         $intarrpar1$ and $intarrpar2$;
8  if $cldist$ is $uniform$ then
9      Generate $N$ claim sizes using a continuous uniform distribution with parameters
         $clsizepar1$ and $clsizepar2$;
10 else if $cldist$ is $exponential$ then
11     Generate $N$ claim sizes using an exponential distribution with parameter $clsizepar1$;
12 else if $cldist$ is $gamma$ then
13     Generate $N$ claim sizes using a gamma distribution with parameters $clsizepar1$ and
         $clsizepar2$;
14 Set arrival times to the cumulative sum of the interarrival times;
15 Set levels to the cumulative sum of the claim sizes;
16 Set the premium to $u0 + c * arrivalTimes$;
17 Set the boolean $ruin$ to $min(premium - levels) < 0$;
18 Create a list $ret$ of the arrival times, levels and $ruin$;
19 Return $ret$;
```

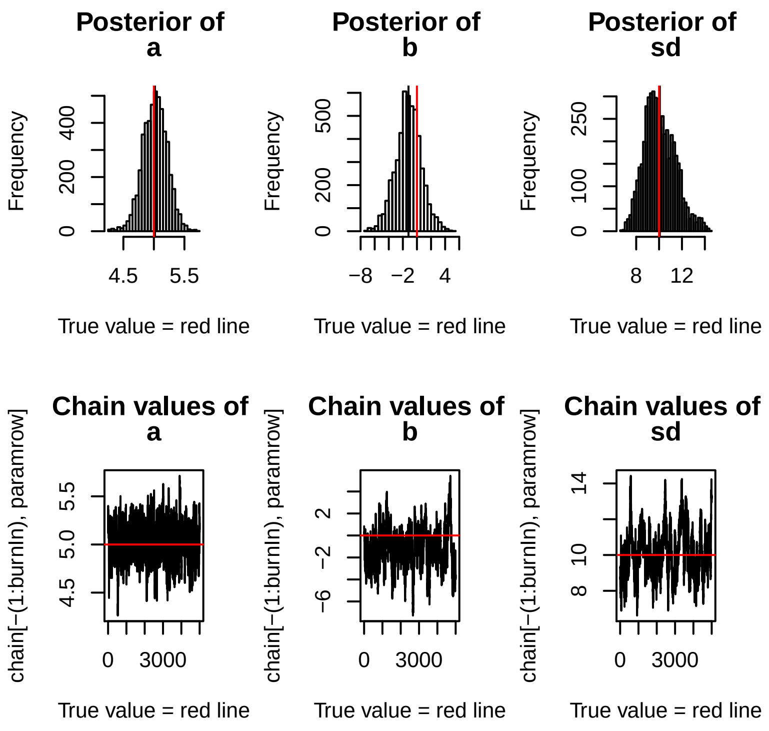
3 Setup

4 Results

5 Conclusion


Test Data
y
50
0
−50
−15
−10
−5
0
5
10
15
x


Log likelihood
−120
−140
−160
3
4
5
6
7
values of slope parameter a



```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -28.638  -2.827   0.917   5.319  15.967
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.1176     1.7739   -0.63    0.534
## x             5.0344     0.1983   25.38   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.877 on 29 degrees of freedom
## Multiple R-squared:  0.9569,Adjusted R-squared:  0.9554
## F-statistic: 644.3 on 1 and 29 DF,  p-value: < 2.2e-16
```