David Giles

Bayesian Econometrics

7. Acceptance-Rejection Sampling

- Sometimes we can't use the inversion of the c.d.f. to get random values (as was done, essentially, in the Table Look-up Method).
- For these cases, use some indirect method.
- We generate a "candidate" random variable.
- Only accept it if it passes some "test".
- Used appropriately, this general approach allows us to simulate from almost any distribution.

- The so-called "Acceptance-Rejection" method of sampling will form basis later for the Metropolis-Hastings methodology (a generalization of G.S.)
- Only require the functional form of the kernel of the density, f, of interest.
- Terminology: f = "target density"; g = "candidate (enveloping) density".
- Useful if easy to simulate random variables from g, but not from f.
- Impose 2 constraints on the candidate density, g:

  (i) f and g have the same supports : *i.e.*, $g(x) > 0$ when $f(x) > 0$).

  (ii) There is a finite constant, M, such that $f(x) / g(x) \leq M$, for all x.

  (Clearly, $0 \leq [f(x) / g(x)]$.)
- We can then simulate values, x, of X from f as follows:

(i) Generate values of, y, of Y from g and, *independently*, generate a values u from U [0 , 1].

(ii) If $u \leq \frac{f(y)}{Mg(y)}$; then set $x = y$.

(iii) Otherwise discard that value of Y, and repeat.

- Note that:

(i) Pr.(Accept) = (1 / M).

(ii) Expected "Waiting Time" = M.

(iii) Computational efficiency will be achieved if M is chosen to be as small as possible.

- Why does this method work?

- Easy to show that

$$\Pr.(Y \leq x \mid \text{Accept}) = \Pr.(Y \leq x \mid u \leq f(y) / [M\, g(\mathrm{y})]) = \Pr.(X \leq x)$$

- Simulating from g, the output of this algorithm is exactly distributed from f.

- The Acceptance - Rejection method can be used no matter what the dimensionality of the random variables.

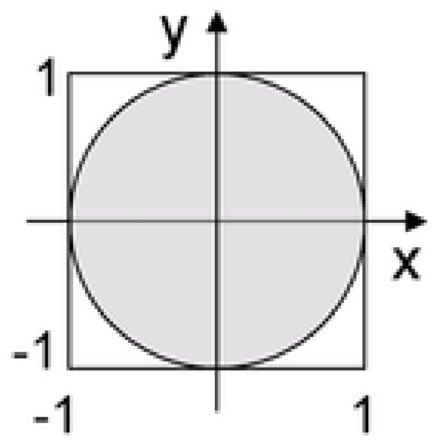
- Just need g to be a density over the same space as f.

- Only need to know (f / g), and hence $f(.)$, *up to a constant*

- Only need an *upper bound* on M.

A Geometric Motivation:

- Suppose we want to generate a random point within the *unit circle*.
- Generate a candidate point, (x, y) where x and y are independent uniformly distributed between -1 and 1.
- If it happens that $(x^2 + y^2) \leq 1$ then the point is within the unit circle and should be accepted.
- If not, then this point should be rejected and another candidate should be generated.



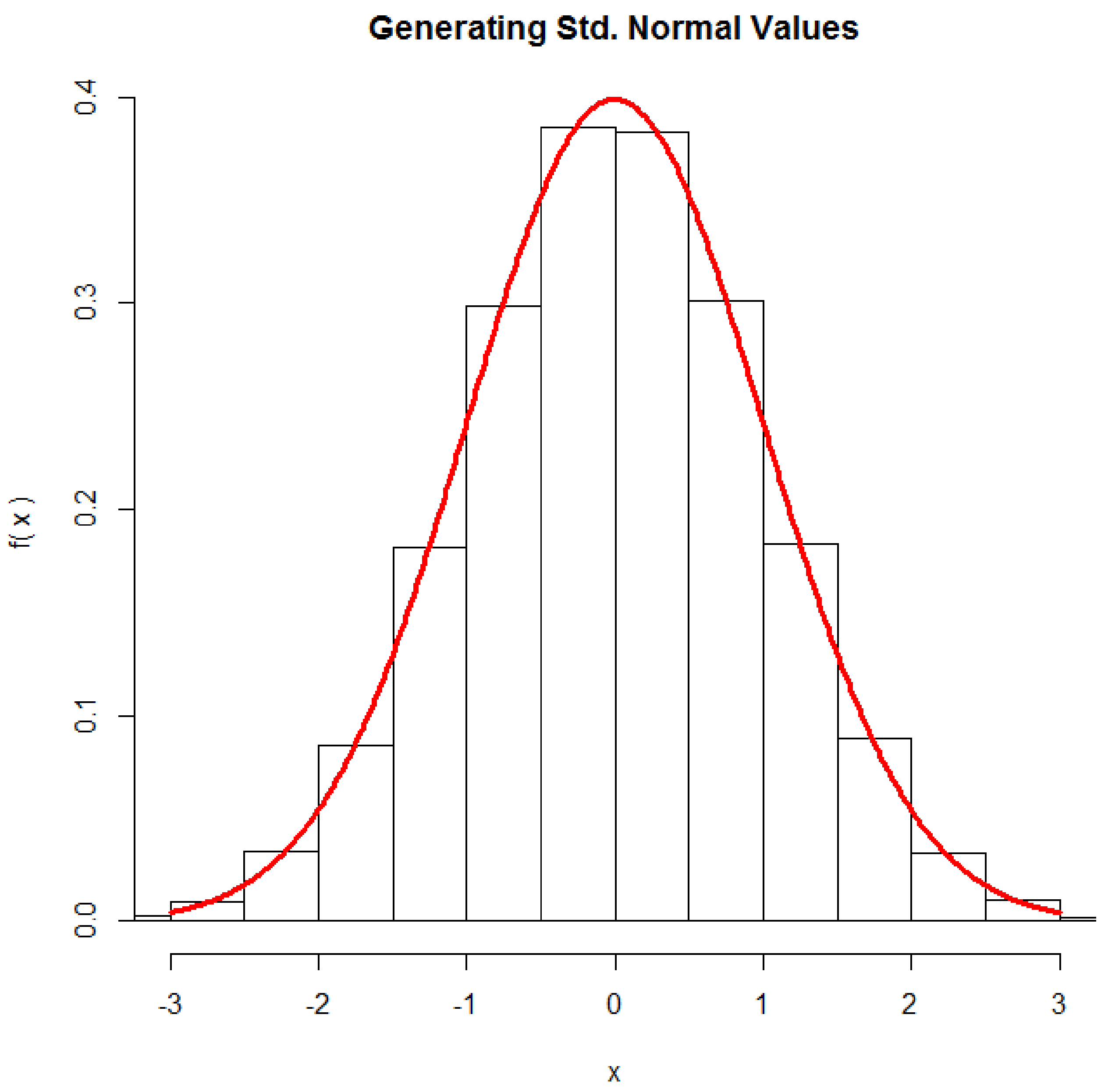
Example 1

- Want to generate Standard Normal values, using the Logistic distribution as the “envelope”.
- $f(\,.\,) \sim$ N [0 , 1] ; $g(\,.\,) \sim$ Logistic [0 , s].
- Choose scale parameter for Logistic so that $[f(\,.\,) / g(\,.\,)] < 1$.
- Modal height of N [0 , 1] = $(1 / \sqrt{2\pi})$;

  modal height of Logistic [0 , s] = $(1 / 4s)$.
- Heights will be equal if $s = 0.6267$
- Set $M = 1.1$, for instance.
- Consider the R code:

```
myrnorm = function(M){
  while(1) {
    u = runif(1); x = rlogis(1, scale = 0.627)
    if(u < dnorm(x)/M/dlogis(x, scale = 0.627))
        return(x)
  }
}
```

```
nrep<-100000

hist(replicate(nrep, myrnorm(1.1)), prob=TRUE, main = "Simulated Std. Normal Values",

xlab="x", ylab="f( x )", xlim=c(-3,3))

lines(seq(-3, 3, 0.01), dnorm(seq(-3, 3, 0.01)), col=2, lwd=3 )
```


Generating Std. Normal Values
f(x)
0.0
0.1
0.2
0.3
0.4
-3
-2
-1
0
1
2
3
x

Example 2

Courtesy of Patrick Lam (*Harvard*)

Want to sample from a Triangular distribution, whose density is

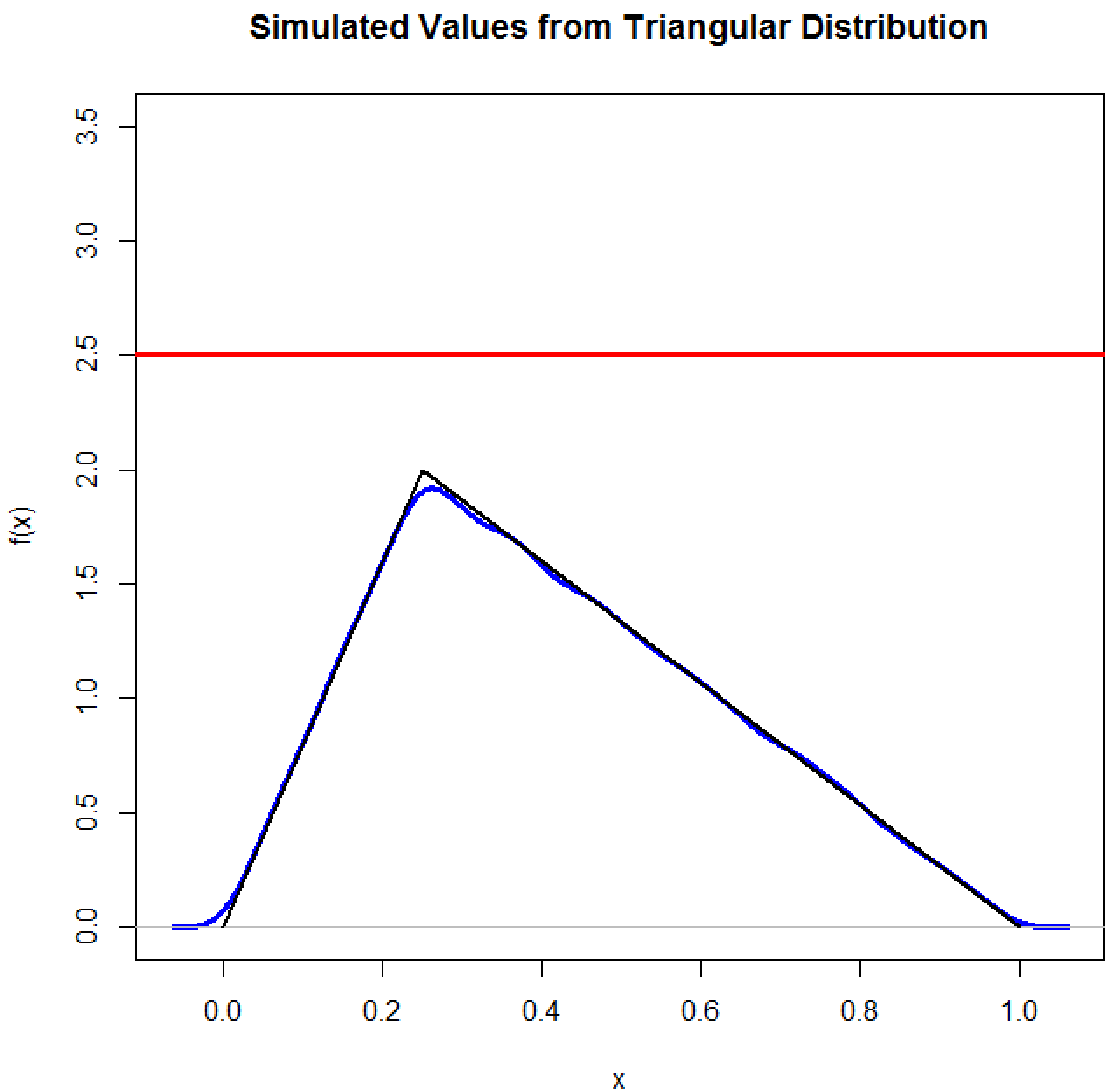
$$f(x) = 8x \quad ; \quad 0 \leq x \leq 0.25$$

$$= \left(\frac{8}{3}\right) - \left(\frac{8x}{3}\right) \quad ; \quad 0.25 < x \leq 1$$

- Use a Uniform distribution for $g(y)$.
- Maximum height of $f(x)$ is 2.0, so set $M = 2.5$, say.
- Use R code to simulate 50,000 draws from $f(.)$.

```
f <- function(x) {
if (x >= 0 && x < 0.25)
8 * x
else if (x >= 0.25 && x <= 1)
 8/3 - 8 * x/3
else 0
 }

g <- function(x) {
    if (x >= 0 && x <= 1)
    1
    else 0
 }
```

```
rep <- 50000
M<- 2.5
n.draws <- 0
draws <- c()
x.grid <- seq(0, 1, by = 0.01)
while (n.draws < rep) {
  y <- runif(1, 0, 1)
  accept.prob <- f(y)/(M * g(y))
  u <- runif(1, 0, 1)
    if (accept.prob >= u) {
  draws <- c(draws, y)
  n.draws <- n.draws + 1
 }
 }
```


Simulated Values from Triangular Distribution
f(x)
x
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0.0
0.2
0.4
0.6
0.8
1.0

Why did it work?

- The difference between $f(x)$ and $Mg(x)$ at places with higher density (*i.e.*, around $x = 0.25$) is smaller than at places with lower density (*i.e.*, around $x = 0.8$).
- So the acceptance probability at $x = 0.25$ is higher and more draws of $x = 0.25$ are accepted.
- There are an infinite number of candidate densities $g(x)$ and constants M that we can use.
- The only difference between them is computation time.
- If $g(x)$ is significantly different in shape than $f(x)$ or if $Mg(x)$ is significantly greater than $f(x)$, then more of our candidate draws will be rejected.
- If $f(x) = Mg(x)$, then all our draws will be accepted.

7.1 Hierarchical Bayes

- One difficulty with Bayesian inference, in practice, is the specification of the prior for the parameters.
- One way to proceed is to set up a "Hierarchical" set of priors.
- We specify a prior for the primary parameters of the model, say $p(\boldsymbol{\theta} \mid \boldsymbol{\omega})$.
- Rather than just assign values for the elements of "prior parameter vector", $\boldsymbol{a}$, we'll assign a further prior, $p(\boldsymbol{\omega})$.
- In fact, this additional prior may involve other unknown parameters - *e.g.*, $p(\boldsymbol{\omega}) = p(\boldsymbol{\omega} \mid \boldsymbol{\varphi})$
- Then we could assign a prior for the elements of $\boldsymbol{\varphi}$; *etc*.

- When would we stop?
- When we have information for the parameters of the "last" prior; or when we can reasonably put uniform or diffuse priors on parameters of the penultimate prior.
- We'll consider a simple example of this - and also use this to illustrate the use of the "Acceptance-Rejection" sampling procedure, within the context of the Gibbs Sampler.
- Return to the Consumption function example, but now with a different set of prior information.
- We'll avoid any integration this time by using the G.S.

Example

$$y_i = \beta x_i + \varepsilon_i \quad ; \quad \varepsilon_i \sim i.i.d.\,N[0\,,\sigma^2] \quad ; \quad i = 1, 2, 3, \ldots\ldots, n$$

(Deviations about sample means, so no intercept.)

- Prior information:

(i) $p(\sigma) \propto 1/\sigma \quad ; \quad 0 < \sigma < \infty$

(ii) $p(\beta\,|a, b) \propto \beta^{a-1}(1-\beta)^{b-1} \quad ; \quad 0 < \beta < 1$

Beta(a , b) ; $a, b > 0$

(iii) $p(a) \propto 1/a \quad ; \quad 0 < a < \infty$

(iv) $p(b) \propto 1/b \quad ; \quad 0 < b < \infty$

(*How could this be extended even further*?)

- Joint prior p.d.f.:

$$p(\beta, \sigma, a, b) = p(\beta\ |a, b)p(a, b)p(\sigma) = p(\beta\ |a, b)p(a)p(b)p(\sigma)$$

- So,

$$p(\beta, \sigma, a, b)\ \propto (ab\sigma)^{-1}\beta^{a-1}(1-\beta)^{b-1}$$

- The Likelihood function is

$$L(\beta, \sigma, a, b \mid \boldsymbol{y})\ \propto\ \sigma^{-n} exp\left[-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \beta x_i)^2\right]$$

- Now we'll apply Bayes Theorem:

$$p(\beta, \sigma, a, b \mid \boldsymbol{y}) \propto (ab)^{-1} \sigma^{-(n+1)} \beta^{a-1} (1-\beta)^{b-1}$$

$$\times exp\left[-\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \beta x_i)^2\right]$$

- We can marginalize this joint posterior with respect to σ *analytically*, as in previous examples:

$$p(\beta, a, b \mid \boldsymbol{y}) = \int_0^\infty p(\beta, \sigma, a, b \mid \boldsymbol{y}) d\sigma$$

$$\propto (ab)^{-1} \beta^{a-1} (1-\beta)^{b-1} \left[\sum_{i=1}^{n} (y_i - \beta x_i)^2\right]^{-n/2}$$

- Now let's think about the information that we need if we're going to apply the Gibbs Sampler to get the marginal posterior densities for the various parameters.

- We have to determine the various *conditional posterior densities*:

(i) $p(\beta \mid a, b, \boldsymbol{y}) \propto \beta^{a-1}(1-\beta)^{b-1}$

$$\times \left[\sum_{i=1}^{n}(y_i - \beta x_i)^2\right]^{-n/2}$$

(ii) $p(a \mid b, \beta, \boldsymbol{y}) \propto (1/a)\beta^{(a-1)}$

(iii) $p(b \mid a, \beta \mid \boldsymbol{y}) \propto \left(\frac{1}{b}\right)(1-\beta)^{(b-1)}$

- All of these distributions are *totally non-standard*.

- Hence the proposal that we use Acceptance-Rejections sampling.

- Recall, we don't need to know the *normalizing constants* for these densities

  - knowledge of the *kernels* is sufficient.

- R code:

- Functions used for "Acceptance-Rejection" sampling:

```
myfdenbeta = function(M,a,b,n,cons,inc) {
 while(1) {
   u = runif(1); x=rbeta(1,5,2)
if(u < (x^(a-1)*(1-x)^(b-1)*(sum(cons-x*inc)^2))^(-n/2)/M/dbeta(x, 5, 2))
      return(x)
 }
}
```

```
myfdena = function(M,beta) {
     while(1) {
     u = runif(1);  x=rgamma(1, scale=1, shape=1)
     if(u < ((1/x)*beta^(x-1))/M/dgamma(x,scale=1, shape=1 ))
      return(x)
  }
}

myfdenb = function(M,beta) {

  while(1) {
    u = runif(1); x=rgamma(1, scale=1, shape=3)
     if(u < ((1/x)*(1-beta)^(x-1))/M/dgamma(x, scale=1, shape=3 ))
      return(x)
  }
}
```

Rest of the R code for the Gibbs Sampler:

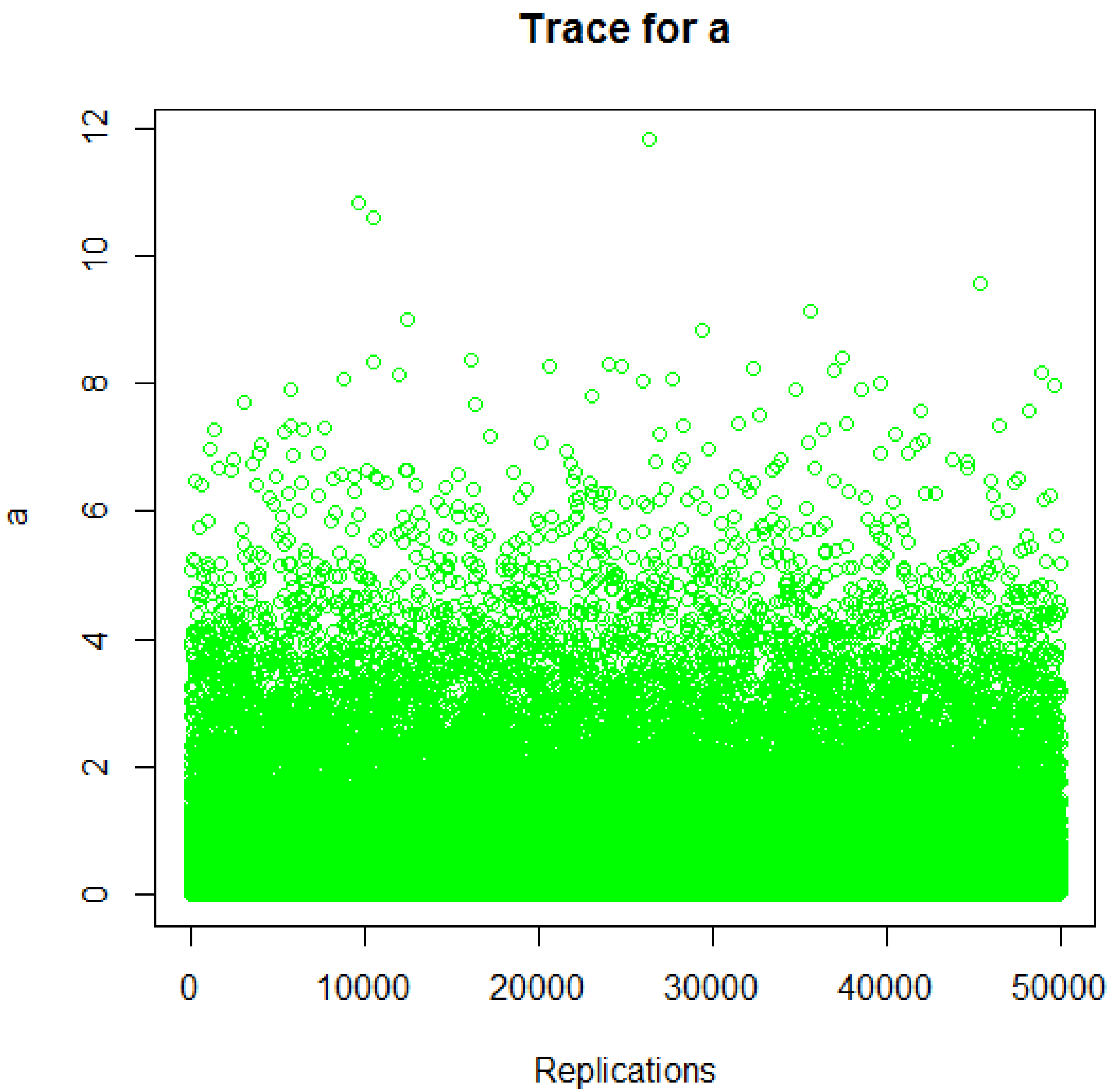
```
library(modeest)

set.seed(123)

nrep<- 52000

burnin<- 2000

margbeta<- vector(length=nrep)

marga<-vector(length=nrep)

margb<-vector(length=nrep)
```

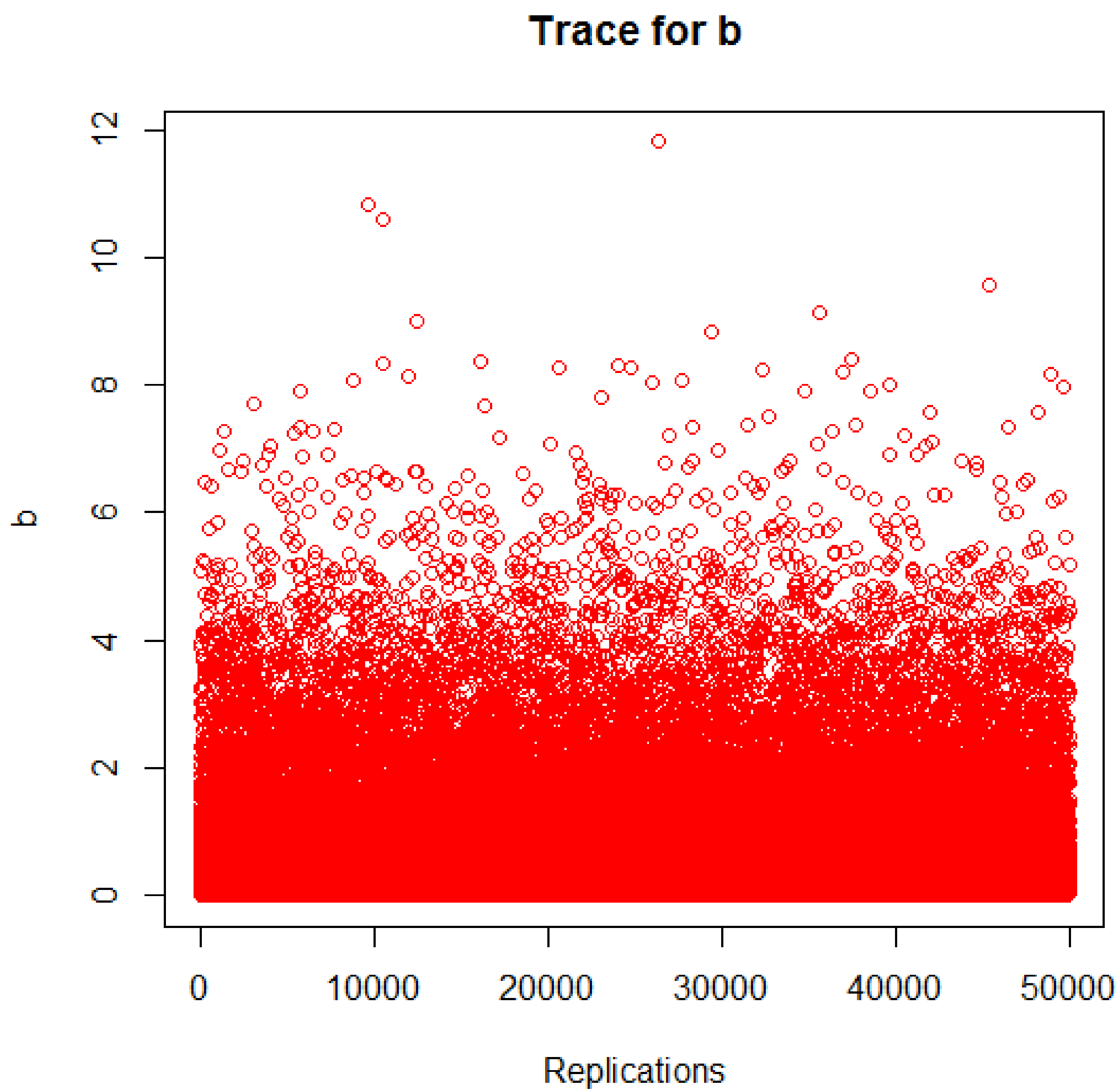
```
# Read the data:

cons.df<-
read.table("http://web.uvic.ca/~dgiles/blog/consump.dat",header=TRUE)

# TAKE DEVIATIONS ABOUT MEANS

consump<- (cons.df$CONS-mean(cons.df$CONS))

income<-  (cons.df$Y-mean(cons.df$Y))

mle<- lm(consump~income -1)

# ASSUMING NORMAL ERRORS, WE NOW HAVE THE MLE OF Beta

summary(mle)

beta<- as.numeric(mle[1])
```

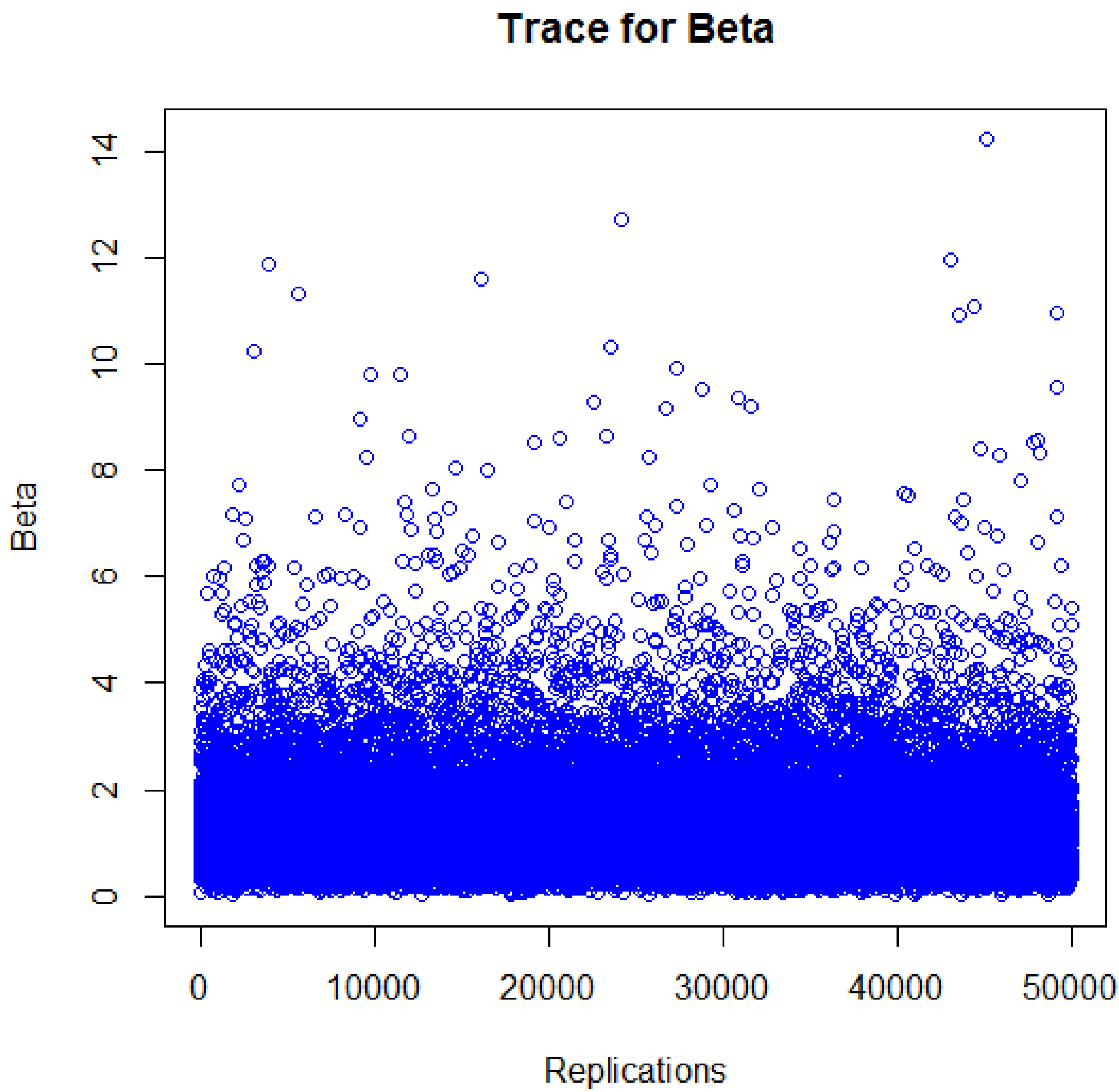
```
# START OF GIBBS SAMPLER

for (ii in 1:nrep)  {

marga[ii]<- a<- myfdena(5, beta)

margb[ii]<-b<- myfdenb(5, beta)

margbeta[ii]<- beta<- myfdenbeta(5, a, b,length(consump),consump, income)

}

# END OF GIBBS SAMPLER
```

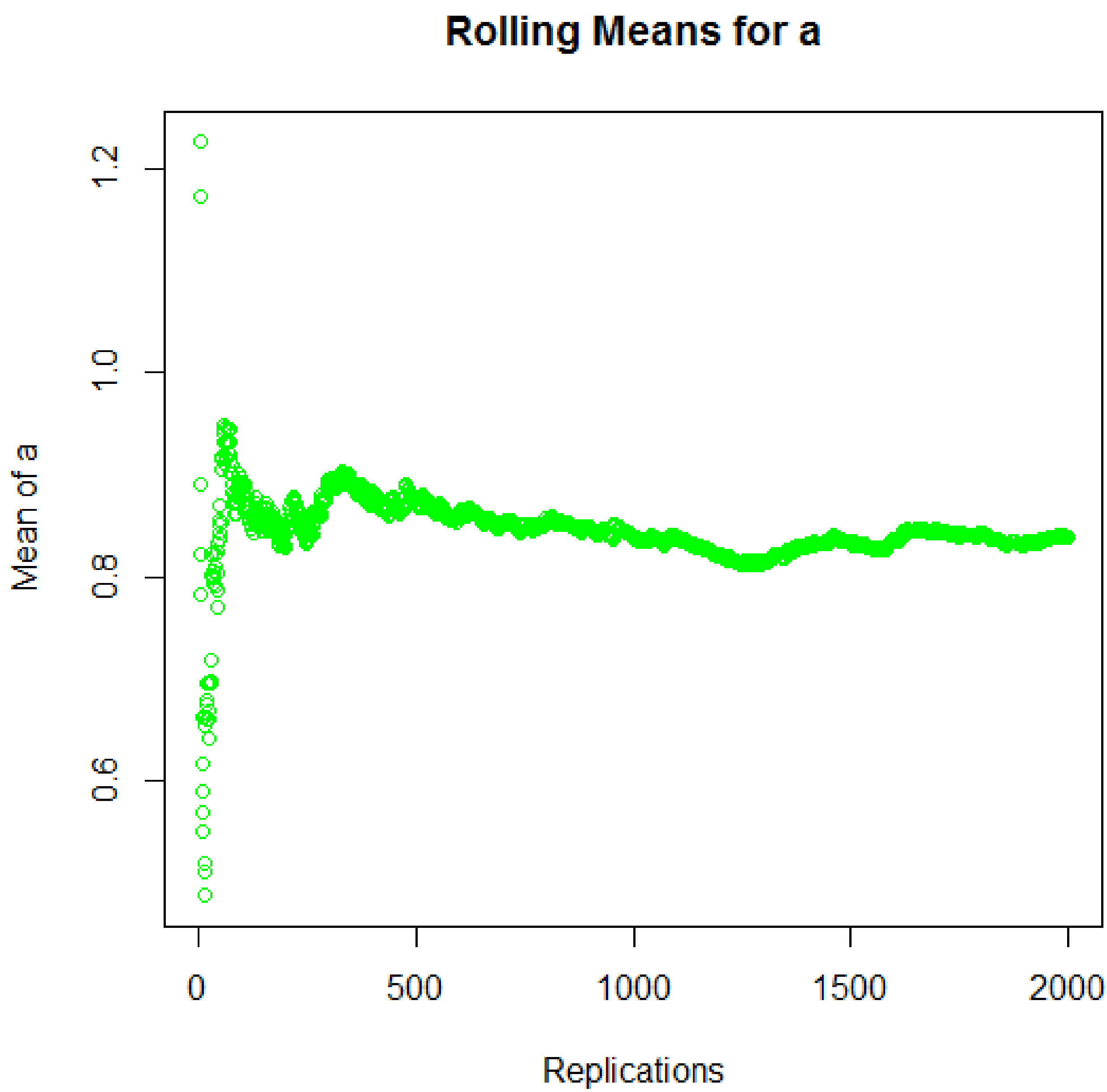
Maximum Likelihood Results

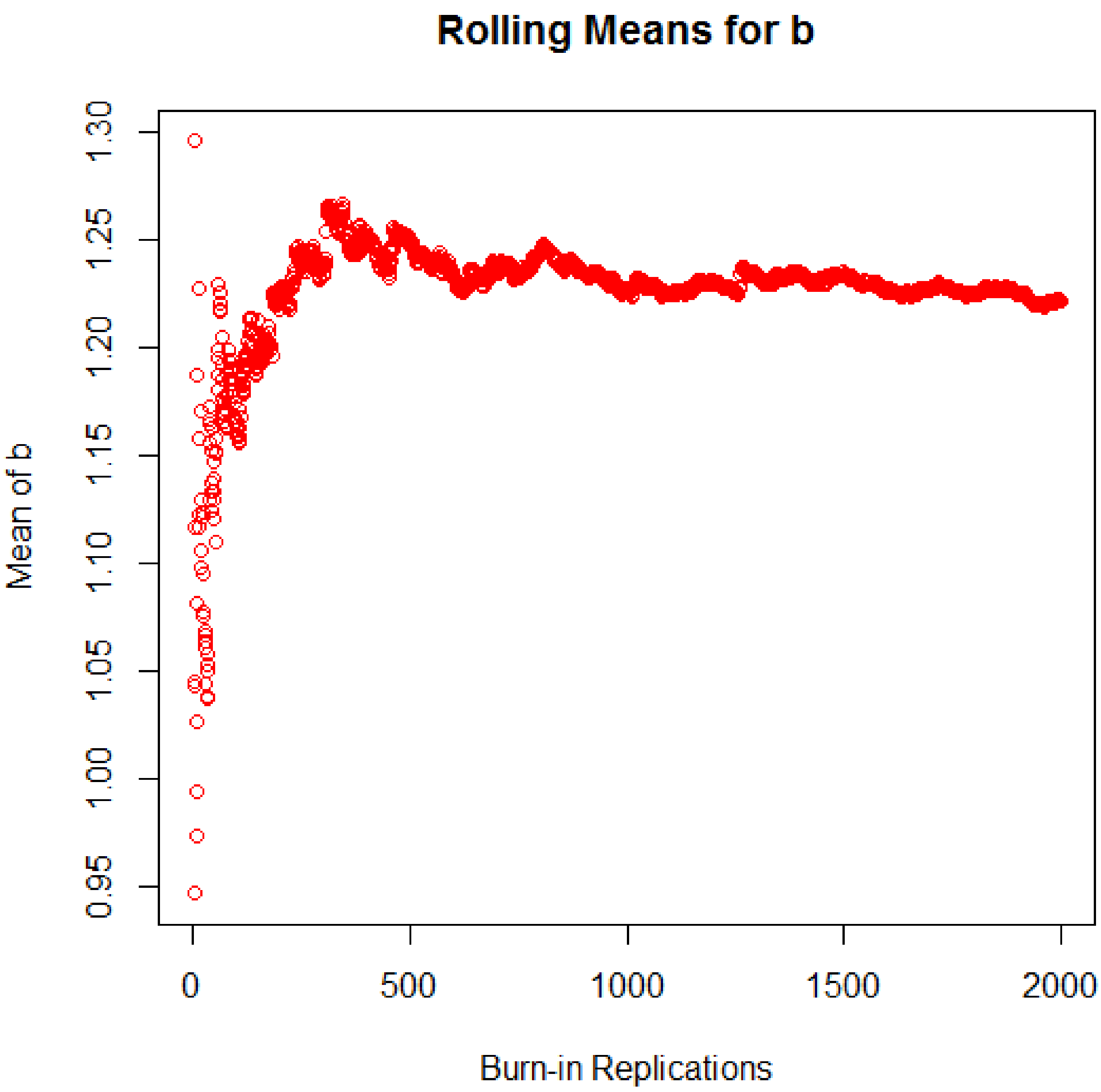
```
Residuals:
    Min      1Q  Median      3Q     Max
-43.304  -2.994   1.686   8.586  47.164

Coefficients:
       Estimate Std. Error t value Pr(>|t|)
income  0.89848    0.00581   154.7   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 18.69 on 35 degrees of freedom
Multiple R-squared:  0.9985,    Adjusted R-squared:  0.9985
F-statistic: 2.392e+04 on 1 and 35 DF,  p-value: < 2.2e-16
```

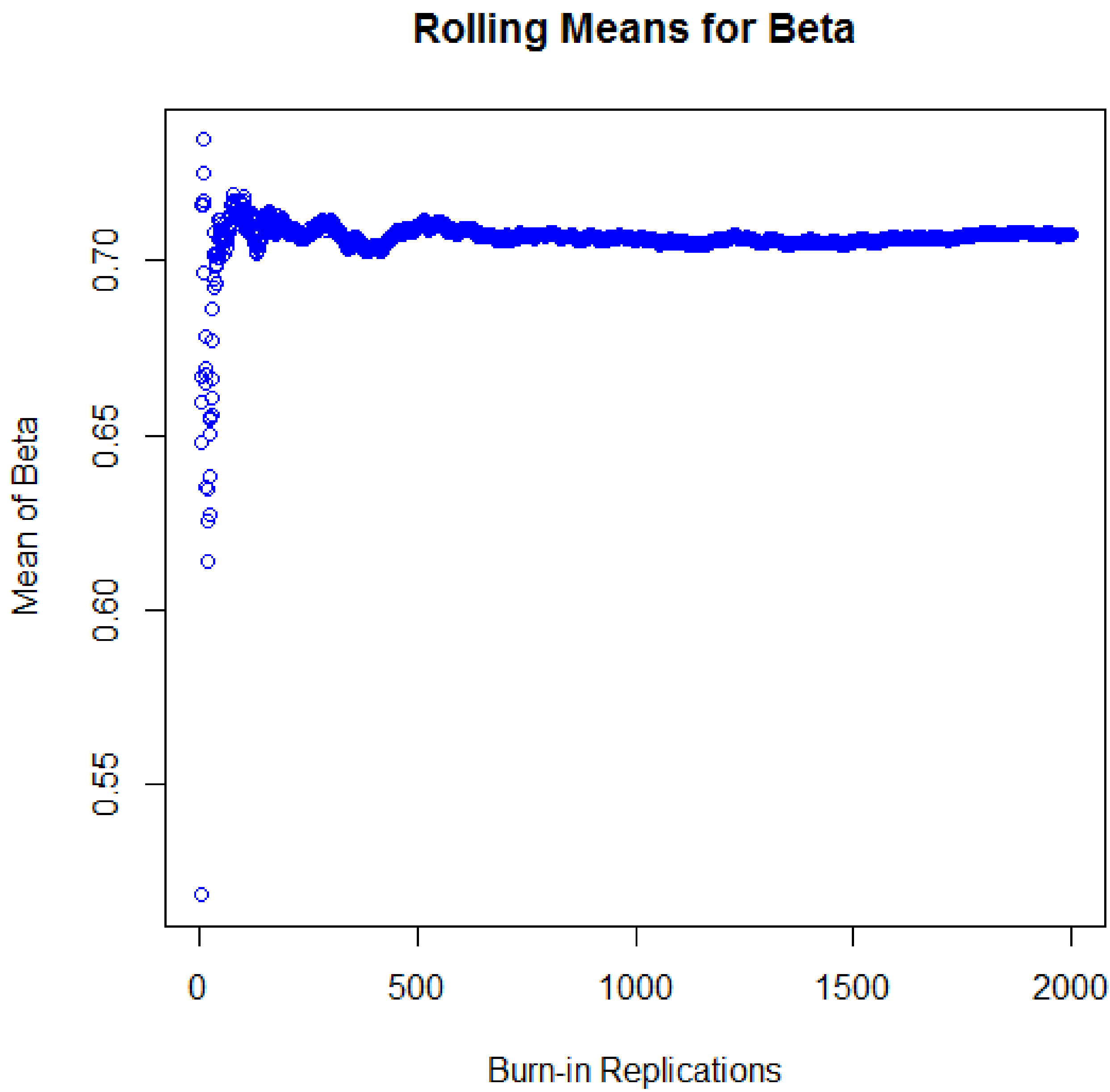

Trace for a
a
0
2
4
6
8
10
12
0
10000
20000
30000
40000
50000
Replications

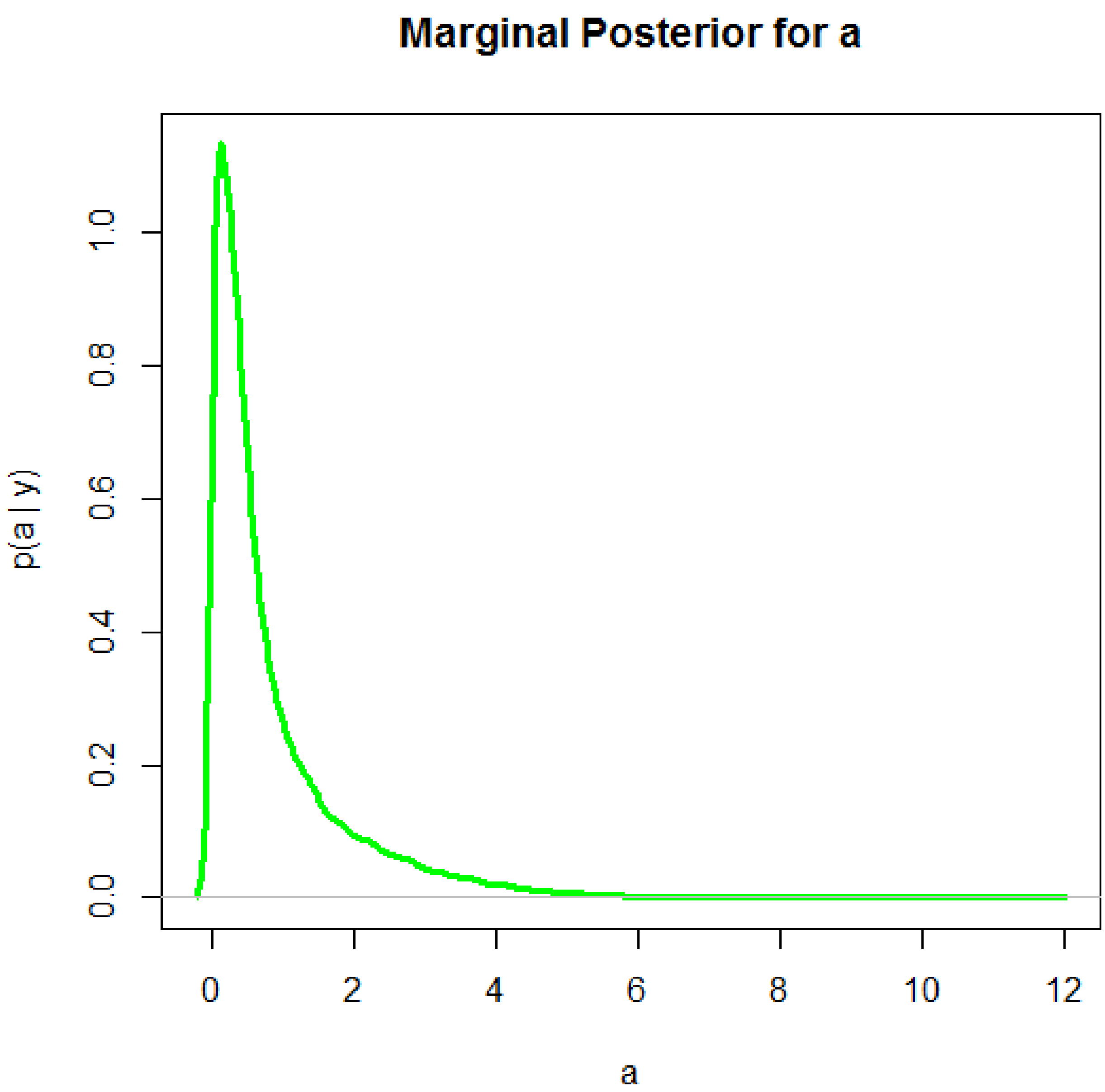

Trace for b
b
0
2
4
6
8
10
12
0
10000
20000
30000
40000
50000
Replications


Trace for Beta
Beta
Replications
0
2
4
6
8
10
12
14
0
10000
20000
30000
40000
50000

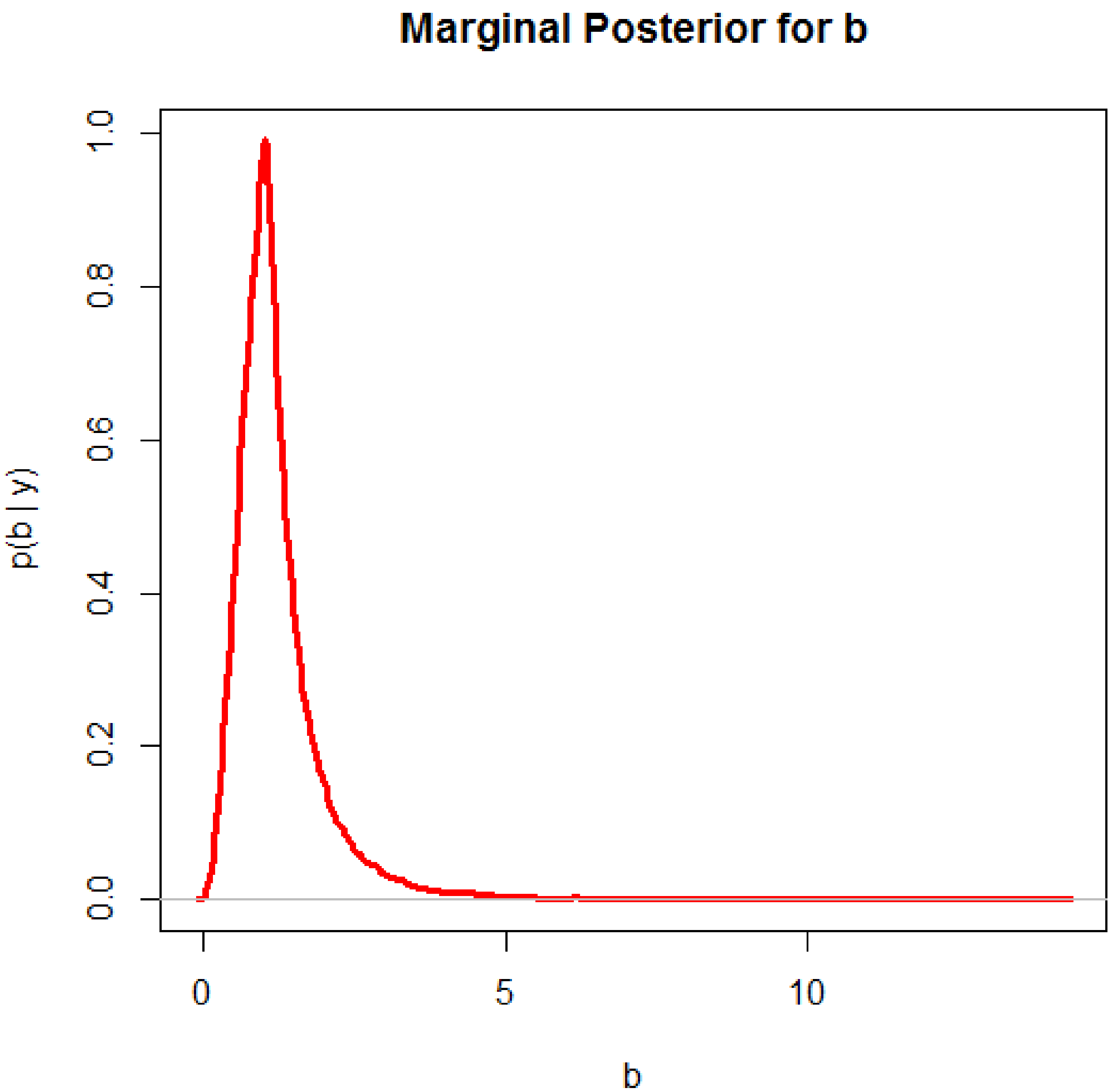

Rolling Means for a
Mean of a
0.6
0.8
1.0
1.2
0
500
1000
1500
2000
Replications

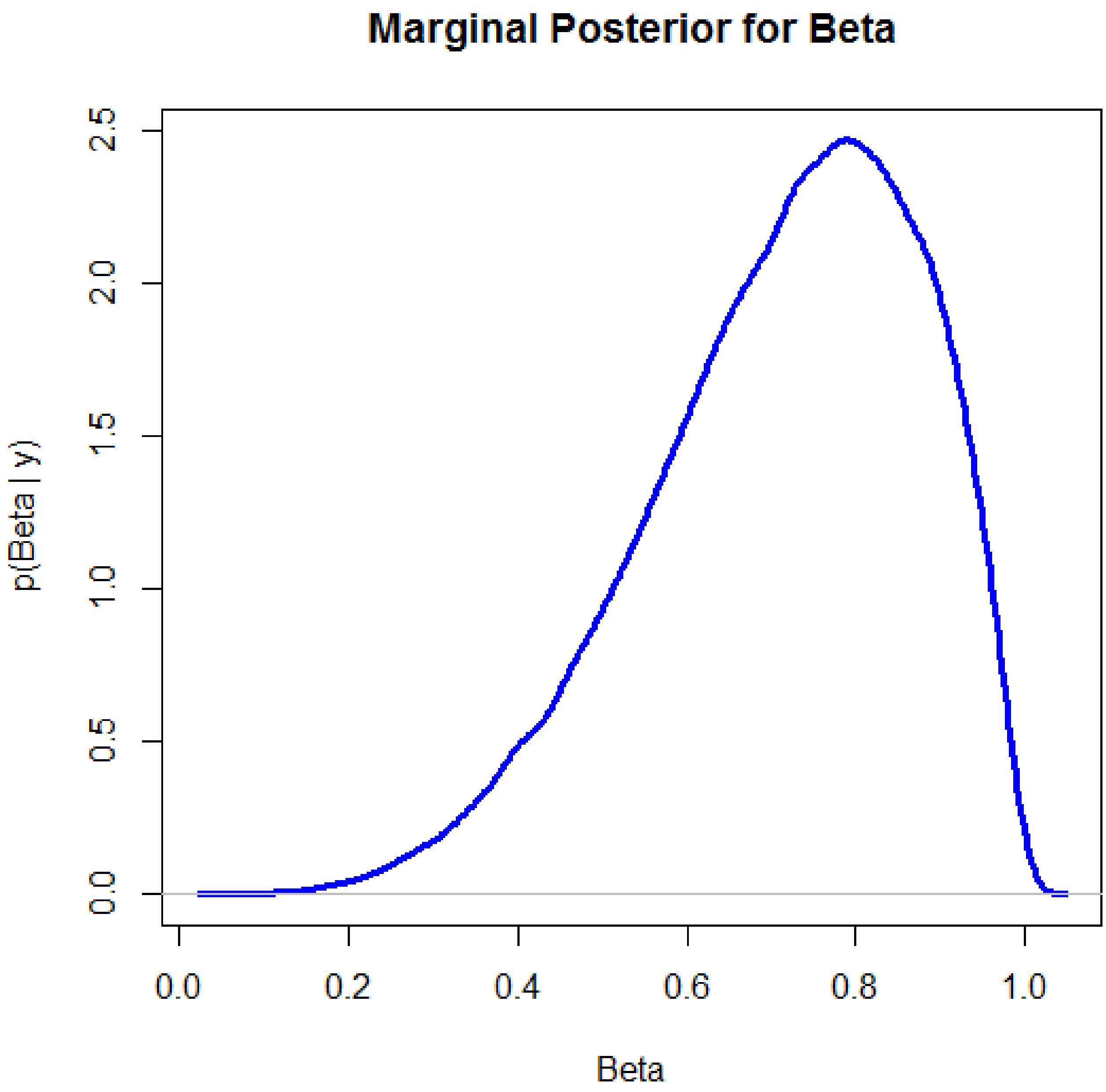

Rolling Means for b
Mean of b
0.95
1.00
1.05
1.10
1.15
1.20
1.25
1.30
0
500
1000
1500
2000
Burn-in Replications


Rolling Means for Beta
Mean of Beta
0.55
0.60
0.65
0.70
0
500
1000
1500
2000
Burn-in Replications



Marginal Posterior for a
p(a | y)
0.0
0.2
0.4
0.6
0.8
1.0
0
2
4
6
8
10
12
a


Marginal Posterior for b
p(b | y)
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
b


Marginal Posterior for Beta
p(Beta | y)
0.0
0.5
1.0
1.5
2.0
2.5
0.0
0.2
0.4
0.6
0.8
1.0
Beta

Summary of Marginal Posterior Distributions

```
> summary(marga[(burnin+1):nrep])
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
 0.000029  0.204400  0.476200  0.861000  1.118000 11.800000
> summary(margb[(burnin+1):nrep])
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
 0.01466  0.77670  1.04900  1.21900  1.42800 14.21000
> summary(margbeta[(burnin+1):nrep])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.07212 0.61100 0.73560 0.71420 0.83850 0.99870
```

a

b

Beta

```
> variances
[1] 1.04177646 0.59605127 0.02561821
```

a **b** **Beta**

```
> modes
[1] 0.2138555 0.9435237 0.7896216
```

Recall: MLE for beta was 0.89848