Approximating posterior distribution and some samplers

Kaiden Liu

11/08/2021

Bayes' theorem

1) prior: p(parameter values) - $p(\theta)$
2) likelihood: p(data values | parameter values) - $p(D|\theta)$
3) posterior: p(parameter values | data values) - $p(\theta|D)$

Bayes' theorem Cont.

$$
\begin{aligned}
p(D,\theta) &= p(\theta)\cdot p(D|\theta) \\
&= p(D)\cdot p(\theta|D) \\
p(D)\cdot p(\theta|D) &= p(\theta)\cdot p(D|\theta)
\end{aligned}
$$

We can conclude that:

$$
\begin{aligned}
p(\theta|D) &= \frac{p(\theta)\cdot p(D|\theta)}{p(D)} \\
p(\theta|D) &\propto p(\theta)\cdot p(D|\theta) \\
(Shape\ of)posterior &= prior\cdot likelihood
\end{aligned}
$$

Grid approximation

1. Define a discrete grid of possible θ values.
2. Evaluate the prior pdf $f(\theta)$ and likelihood function $L(\theta|D)$ at each θ grid value.
3. Obtain a discrete approximation of posterior pdf $f(\theta|D)$ by:

- calculating the product $f(\theta)L(\theta|D)$ at each θ grid value
- normalizing the products so that they sum to 1 across all θ.

4. Randomly sample n θ grid values with respect to their corresponding normalized posterior probabilities.

Example

Figure 1: rainbow

2) Rejection Sampling

Sample data from a complicated distribution

- Target (distribution) function f(x) — The "difficult to sample from" distribution. Our distribution of interest!
- Proposal (distribution) function g(x) — The proxy distribution from which we can sample.

1)

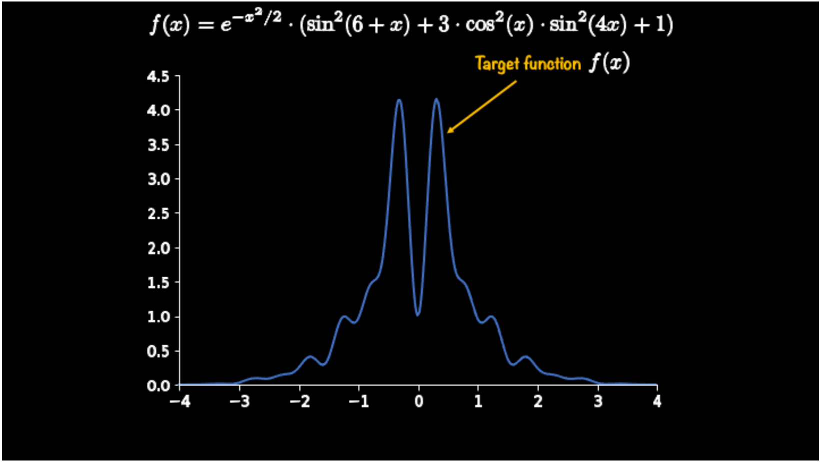


Figure 2: Target Function

2)

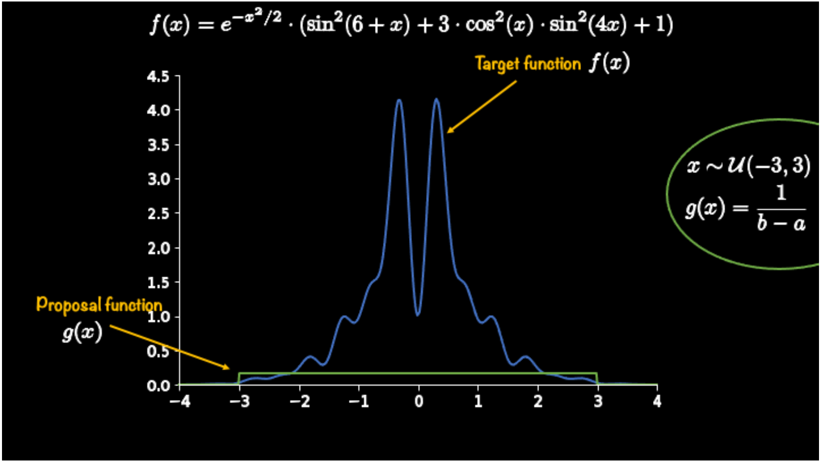


Figure 3: Proposal Function

3)

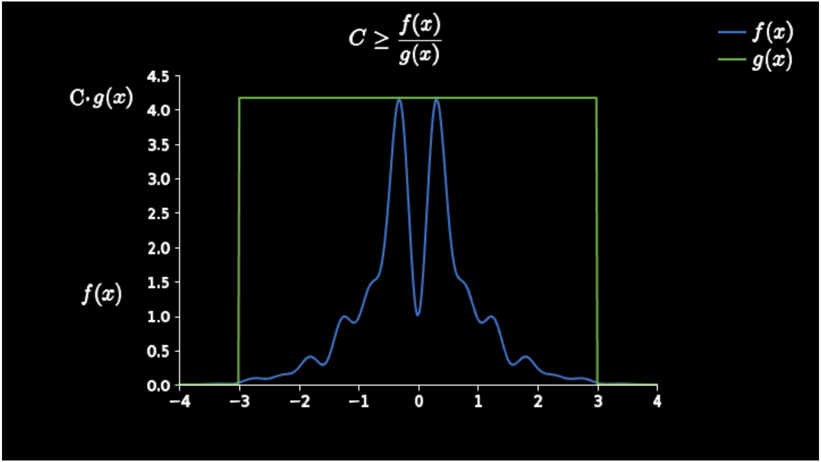
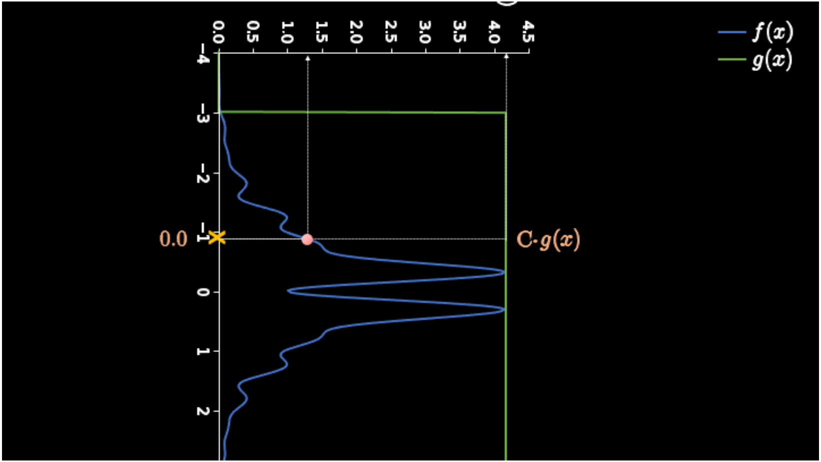


Figure 4: Constant

4)

Accept with probability $\frac{f(x)}{C \cdot g(x)}$



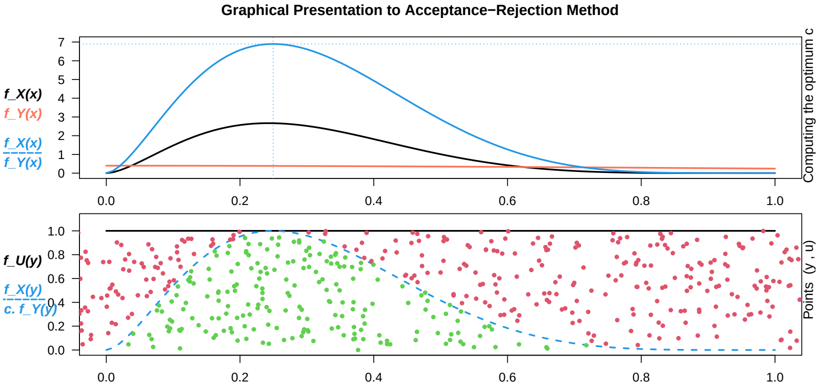
Example

blue is ratio

```r
library(AR)

simulation = AR.Sim( n = 200,
                f_X = function(y){dbeta(y,2.7,6.3)},
                Y.dist = "norm", Y.dist.par = c(0,1),
                Rej.Num = TRUE,
                Rej.Rate = TRUE,
                Acc.Rate = FALSE
)
```

```
## Optimal c = 6.898
```



Output

n

The number/length of data which must be generated/simulated from (f_X)(TARGET) density.

Optimal c = 6.898

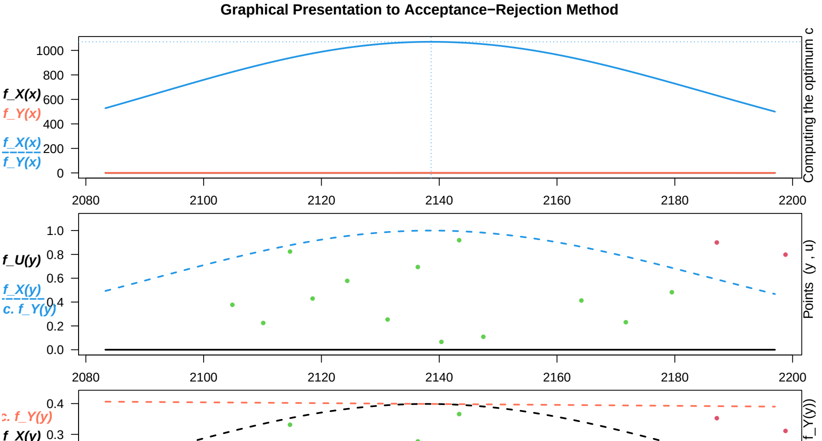
The numbers of Rejections = 1295

Ratio of Rejections = 0.866

Example_DNA

```
start.time<-Sys.time()
lr_rs = AR.Sim( n = 20,
               f_X = function(y){(posterior(y,as.numeric(gammaPrior_Cont[[sam]]$kij[taxa]), al_c,be_c))},
               Y.dist = "gamma", Y.dist.par = c((k/mean(lc))*al_c,be_c), xlim=c(k-2*mean(lc),k),
               Rej.Num = TRUE,
               Rej.Rate = TRUE,
               Acc.Rate = FALSE
)
```

```
## Optimal c = 1069.962
```



Limitation

- Selecting the appropriate proposal function & finding its scaling constant
- Requires that the PDF of the target function is known
- Generally inefficient especially in higher dimensions

2.5) Adaptive Rejection Sampling

Define our proposal distribution in log space

1

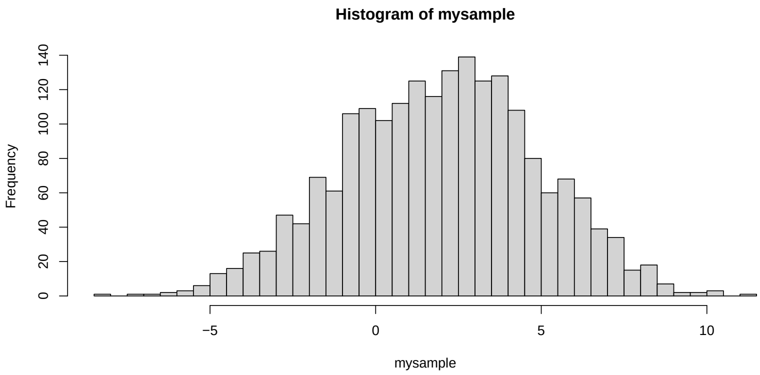
[1] Gilks, W. R., & Wild, P. (1992). Adaptive rejection sampling for Gibbs sampling. Journal of the Royal Statistical Society: Series C (Applied Statistics), 41(2), 337-348.

Example

sample 2000 values from the normal distribution N(2,3)

```
library("ars")

f<-function(x,mu=0,sigma=1){-1/(2*sigma^2)*(x-mu)^2}
fprima<-function(x,mu=0,sigma=1){-1/sigma^2*(x-mu)}
mysample<-ars(2000,f,fprima,mu=2,sigma=3)
hist(mysample, breaks=30)
```



Monte Carlo

Relies on repeated random sampling to obtain numerical result

Ex) $\theta_t \sim$ Normal $(0.5, \sigma)$

Monte Carlo Trace Plot

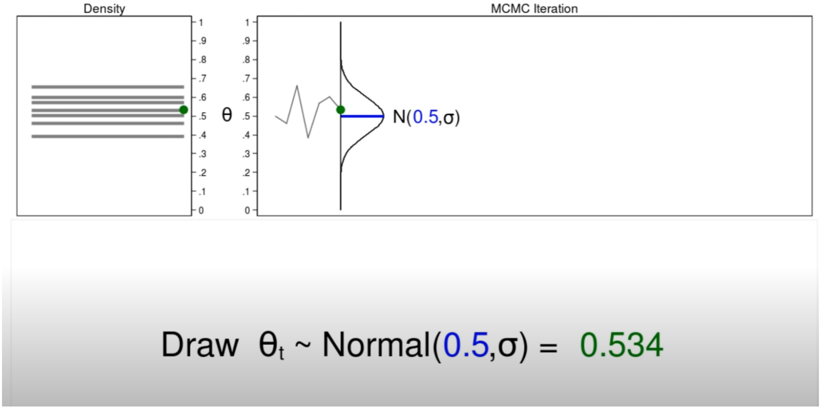


Figure 5: Trace Plot

50000 iteratons

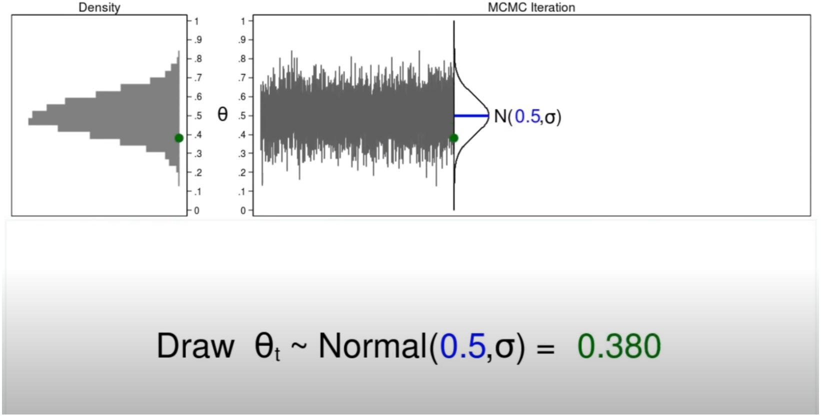


Figure 6: 50000

Markov property

Given the present, the future does not depend on the past.

$$P(X_n = x_n \mid X_{n-1} = x_{n-1}, \ldots, X_0 = x_0) = P(X_n = x_n \mid X_{n-1} = x_{n-1}).$$

Figure 7: markovproperty

Example

Ex) $\theta_t \sim$ Normal (θ_{t-1}, σ)

Depends on the previous number on a sequence

Trace Plot

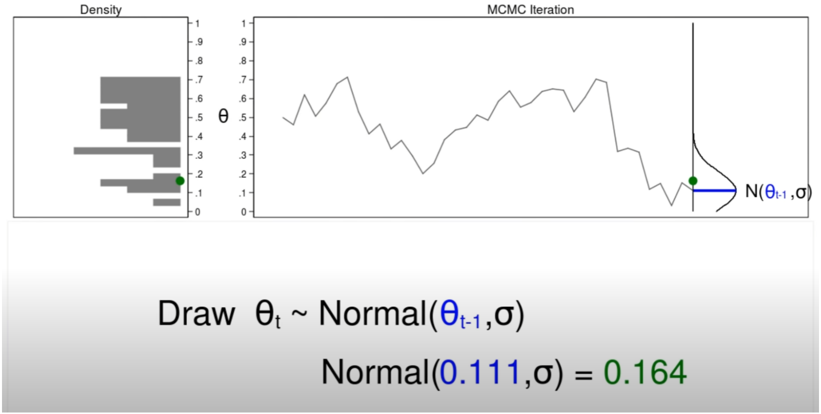
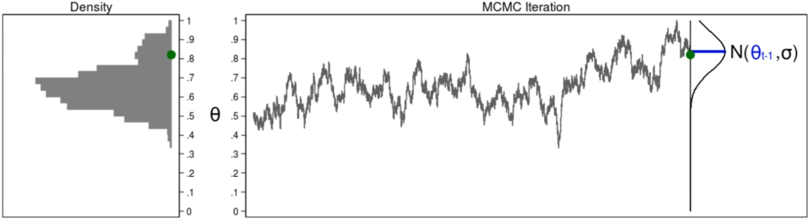


Figure 8: Trace Plot

50000 entries



$$\text{Draw} \quad \theta_t \sim \text{Normal}(\theta_{t-1}, \sigma)$$

$$\text{Normal}(0.836, \sigma) = 0.820$$

3) Metropolis Hasting

The Metropolis–Hastings algorithm can draw samples from any probability distribution f(x), provided that we know a function q(x) proportional to the density of f and the values of q(x) can be calculated. The requirement that q(x) must only be proportional to the density

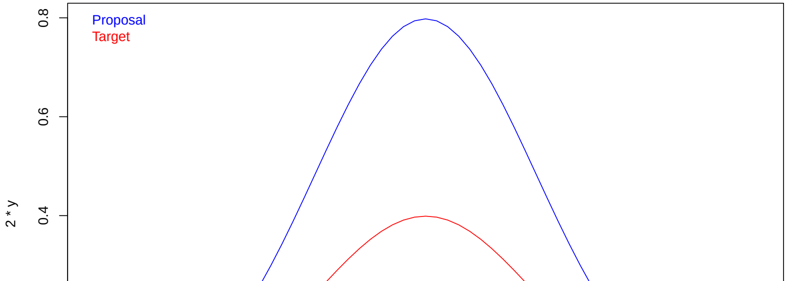
1. Compute $\rho(x,y) = min\left\{\frac{f(y)}{f(x)} \times \frac{q(x|y)}{q(y|x)}, 1\right\}$
2. If $\rho(x,y) < 1$ then generate $U_t \sim \text{Uniform}(0,1)$.
3. Set $X^{(t+1)} = \begin{cases} y & \text{if } \rho(x,y) = 1 \text{ or } U_t < \rho(x,y) \\ x & \text{if } U_t \geq \rho(x, Y_t) \end{cases}$

Figure 9: mh

Intuition

$\alpha = min\{\frac{f(b)}{f(a)}, 1\}$

```
x= seq(-3,3, by=0.1)
y=dnorm(x)
plot(x,2*y, type="l", col="blue")
lines(x,y, col="red")
legend("topleft", legend = c("Proposal", "Target"),
text.col = c('blue','red'), bty = "n")
```



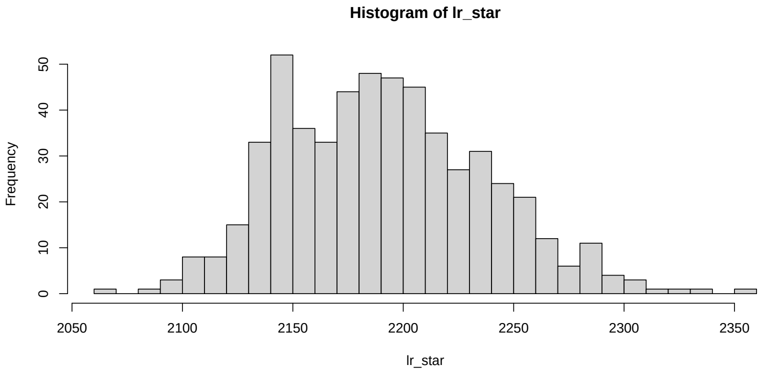
Limitation

- Dependence on starting value
  - Burn-in period
- Autocorrelation due to the Markov Chain properties

Example

```
start.time<-Sys.time()
chain <- MH_MCMC(itera = 600,
                            k = as.numeric(gammaPrior_Cont[[sam]]$kij[taxa]),
                            al_c = gammaPrior_Cont[[sam]]$alpha_ij_c[taxa],
                            be_c = gammaPrior_Cont[[sam]]$beta_ij_c[taxa],
                            startvalue_lamda_r = 0)
end.time<-Sys.time()
time.taken_mh <- end.time-start.time

lr_star <- chain[50:length(chain)]
hist(lr_star, breaks=30)
```



Gibbs Sampling

initialize Y^0, X^0

for j = 1, 2, 3,… do

  sample $X^j \sim p(X|Y^{j-1})$

  sample $Y^j \sim p(Y|X^j)$

end for

Figure 10: Gibbs

Limitation

	0	1
1	0	$\frac{1}{2}$
0	$\frac{1}{2}$	0

Limitation

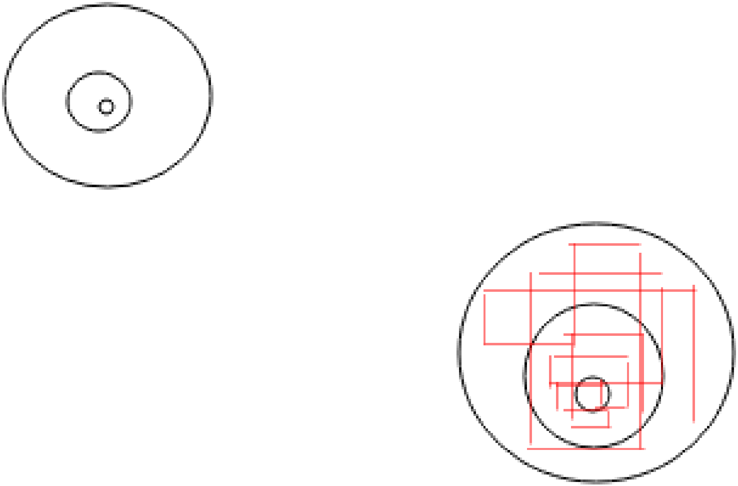
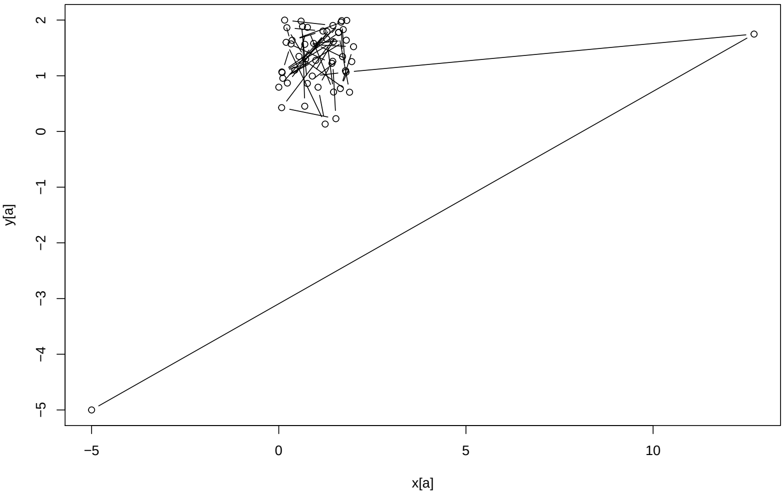


Figure 11: Limitation

Example



Example

```
#library(plotly)
#plot_ly(x=den$x, y=den$y, z=den$z) %>% add_surface
```

Goodness of fit

```
goodness_of_fit <- function(lc,lr){
  kij_c_star <- rpois(length(lc),mean(lc))
  kij_r_star <- rpois(length(lc),mean(lr))
  kij_star <- kij_r_star+kij_c_star
  hist(kij_star,
       main=paste("Goodness of fit for",deparse(substitute(lr))) , breaks =30, xlab="counts")
  abline(v=as.numeric(gammaPrior_Cont[[sam]]$kij[taxa]))
}
```

Grid Approximation

```r
#goodness_of_fit(lc,post_lr)

# actual observed counts
k
```

```
## [1] 2197
```

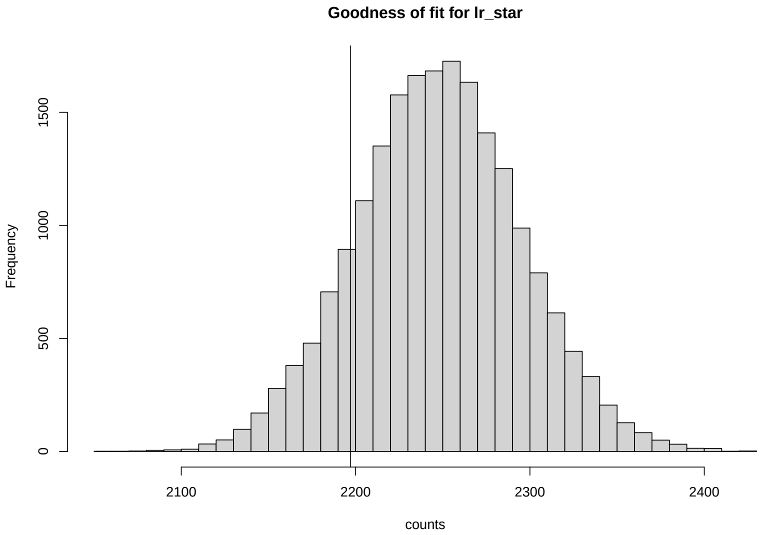
```r
# point estimation

mean(lc) + lr_ga
```

```
## [1] 2192.639
```

Metropolis-Hasting

```
goodness_of_fit(lc,lr_star)
```



Rejection Sampling

```
goodness_of_fit(lc,lr_rs)
```