HARMONY

Novel tools for test evaluation and disease prevalence estimation

ΤΜΗΜΑ ΔΗΜΟΣΙΑΣ ΚΑΙ ΕΝΙΑΙΑΣ ΥΓΕΙΑΣ
DEPARTMENT OF PUBLIC AND ONE HEALTH
ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ UNIVERSITY OF THESSALY

cost

EUROPEAN COOPERATION IN SCIENCE & TECHNOLOGY

MCMC explained

Polychronis Kostoulas

Bayesian modeling framework

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)}$$

$p(y|\theta)$: likelihood function
(the distribution of the data given θ)

$p(\theta)$: prior distribution of θ

$p(y)$: marginal distribution of y

$p(\theta|y)$: posterior distribution of θ

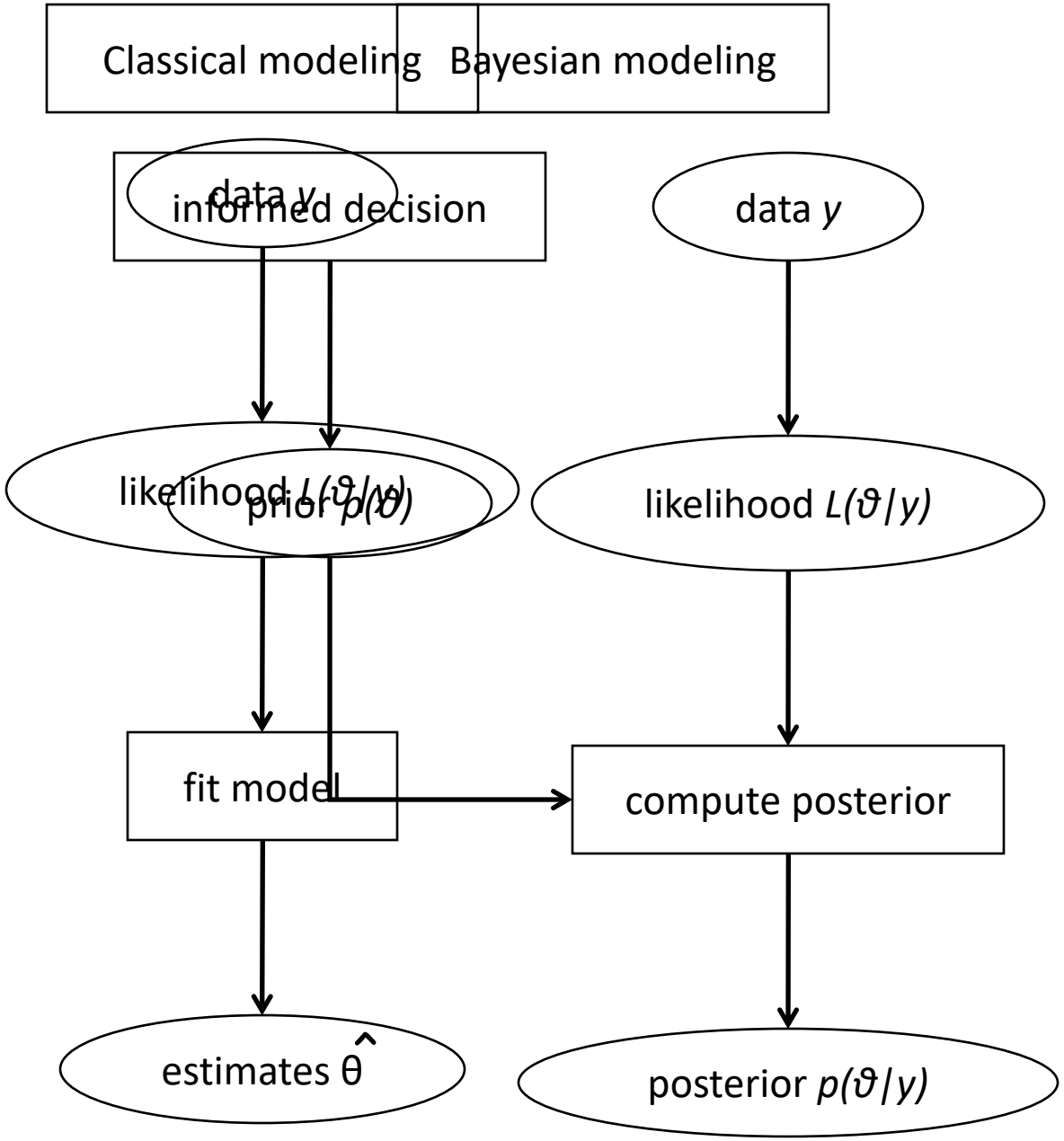
Bayesian inference

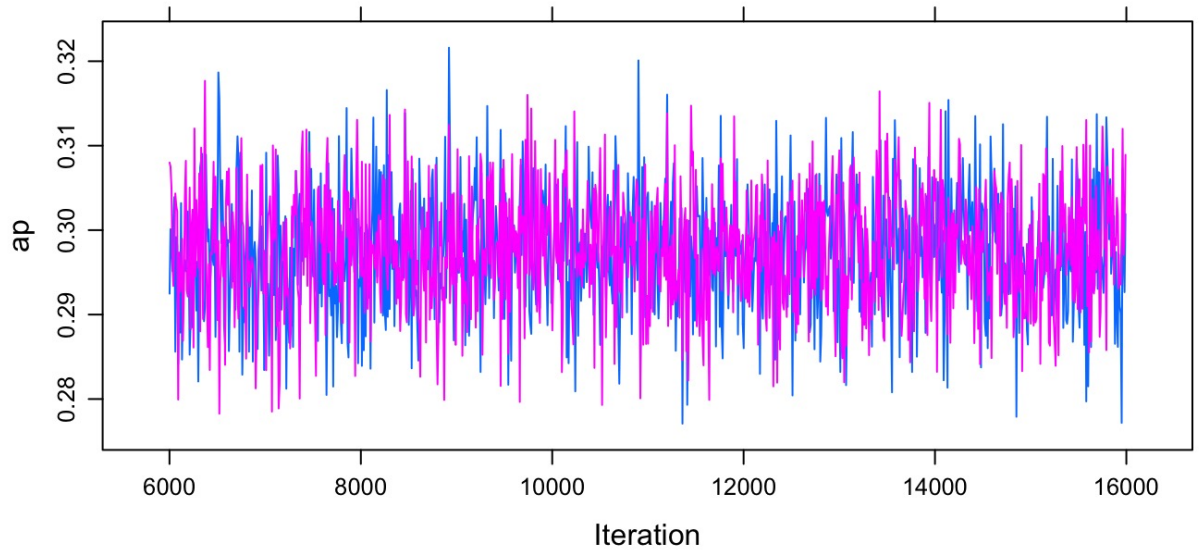
ignoring terms not involving θ:

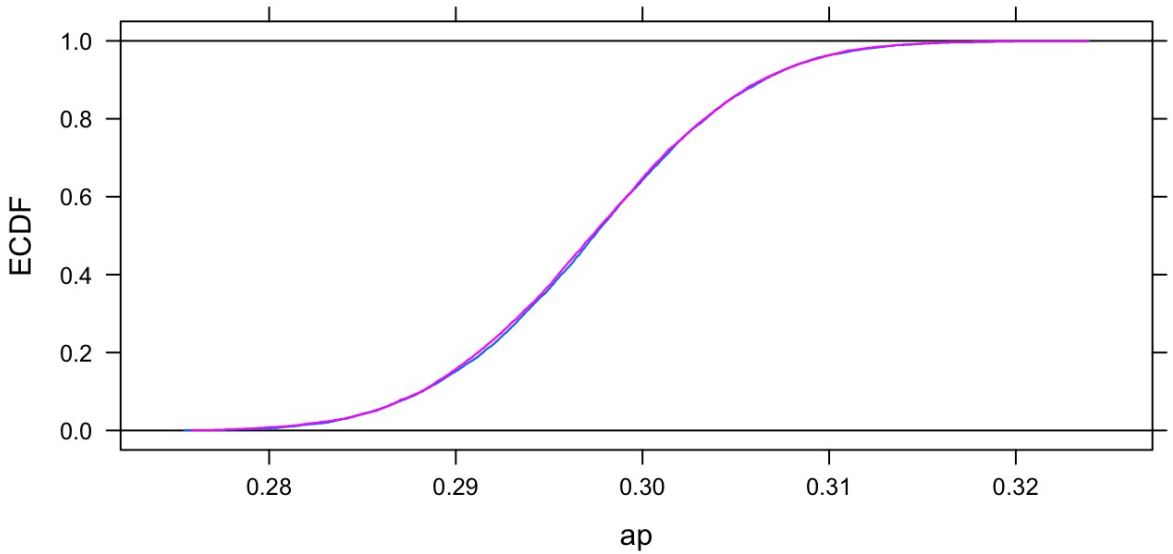
$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)}$$

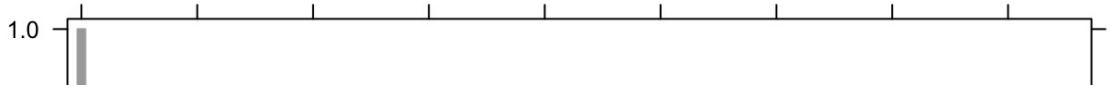
$$p(\theta|y) \propto p(y|\theta)p(\theta)$$

posterior $\propto$ likelihood $\times$ prior


Classical modeling
Bayesian modeling
data y
informed decision
data y
likelihood L(ϑ|y)
prior p(ϑ)
likelihood L(ϑ|y)
fit model
compute posterior
estimates θ̂
posterior p(ϑ|y)



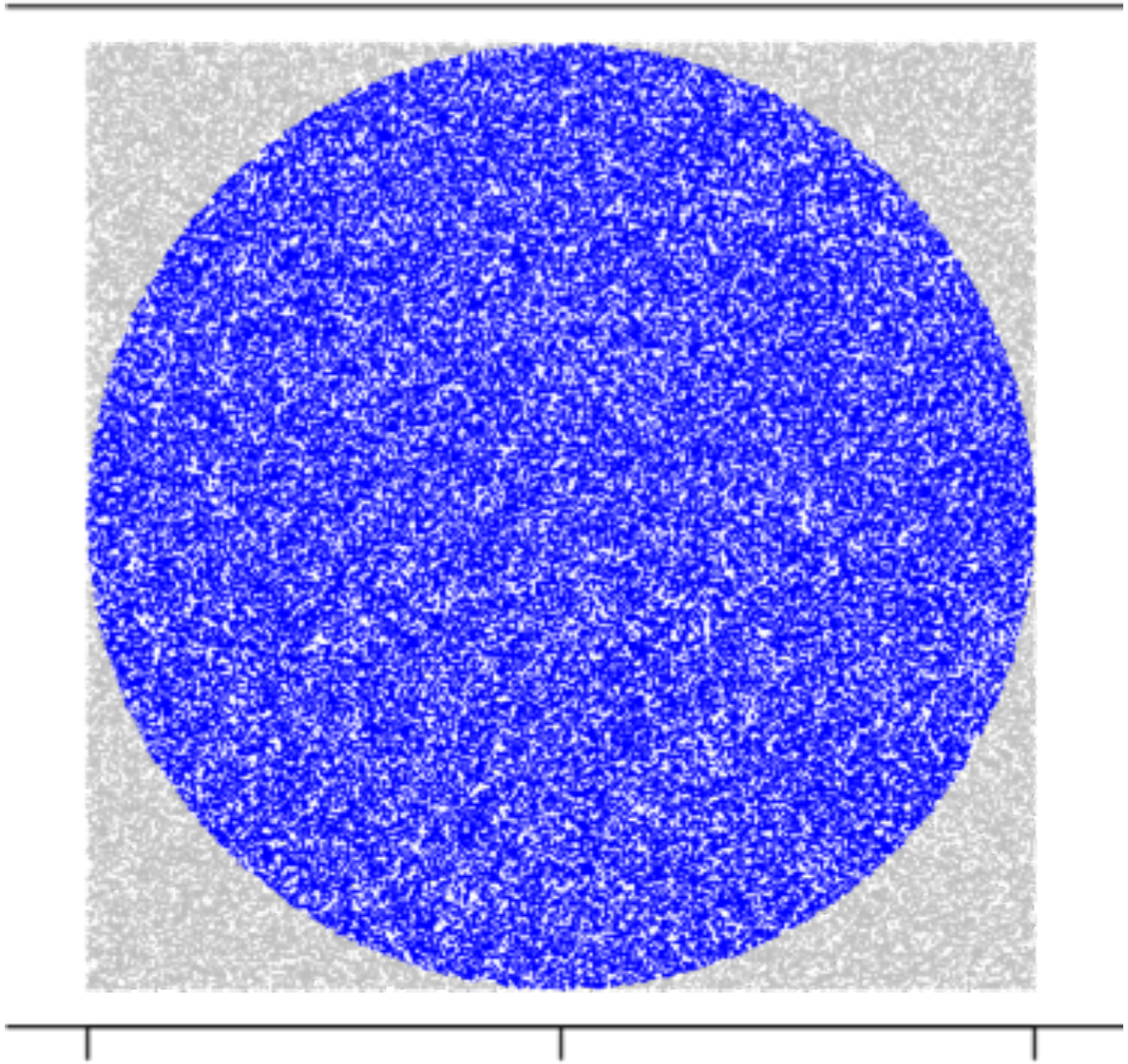


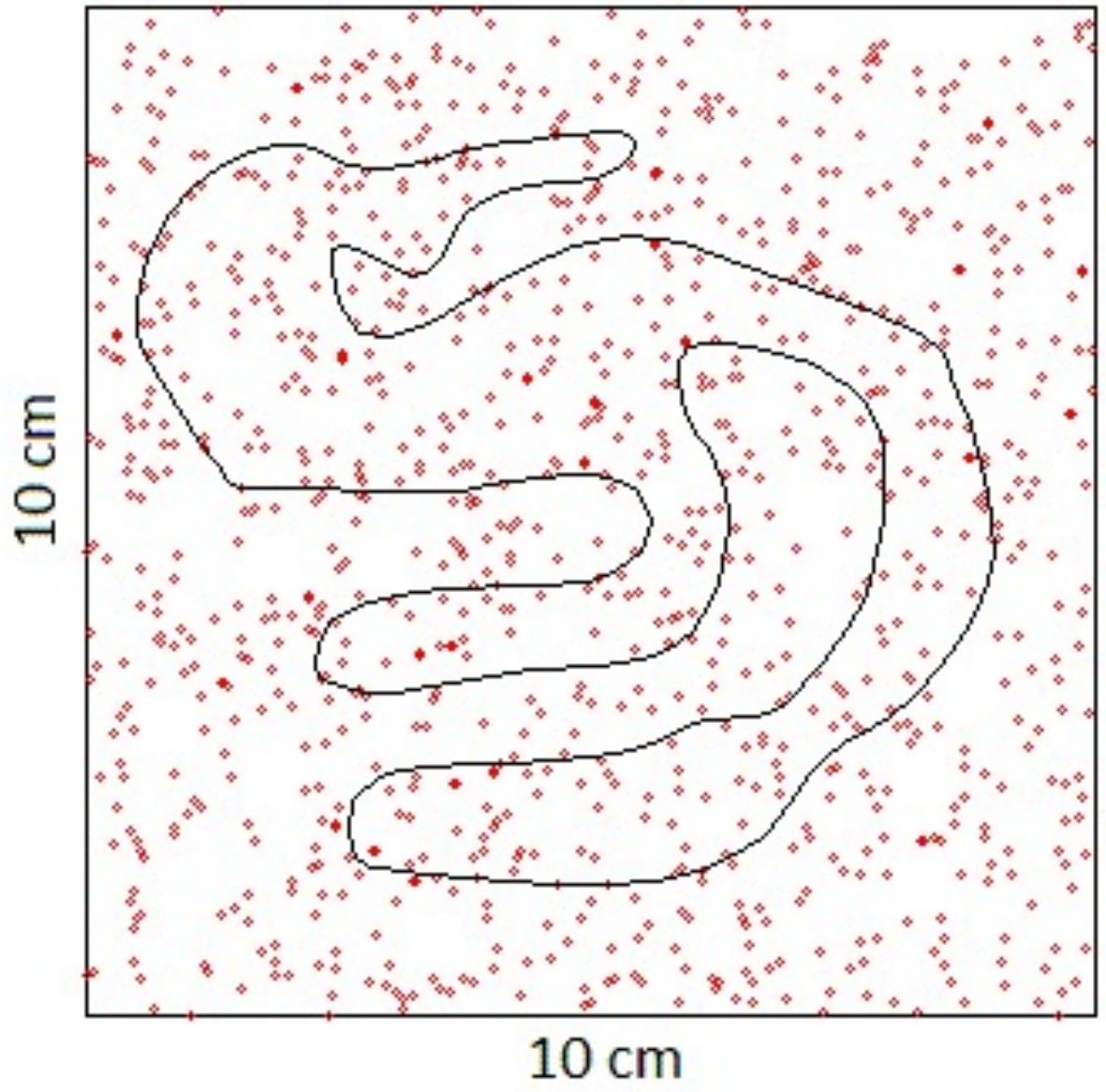


Markov Chain Monte Carlo

Monte Carlo

- What is the probability of rolling two dice?
  - there are 36 combinations
  - you can manually compute the probability of a particular outcome.
- Monte Carlo Simulation
  - simulate rolling the dice 100,000 times (or more)
  - calculate the probabilities from the simulations




10 cm
10 cm

Markov Chain

- Andrey Markov
- The future state of a stochastic variable is only dependent on its present state.
- Google predicts the next word in your sentence based on your previous word.

How does MCMC work in practice?

1. Define a function to calculate a posterior probability
2. Choose a ‘reasonable’ place to start looking for the parameter value
3. Sample another parameter value using a random walk
4. Keep the new value if it’s better than the old
5. Repeat from step 2.

Define a function to calculate a posterior probability

```
# Define a function that calculates a posterior:
log_posterior_fun <- function(parameter){
  # ... Some R code e.g. ...
  ll <- dbinom(data$Pos, data$N, parameter, log=TRUE)
  lp <- dbeta(parameter, 1, 1, log=TRUE)
  return(ll + lp)
}
```

Choose a ‘reasonable’ place to start looking for the parameter value

```
parameter[1] <- 0.25
log_post[1] <- log_posterior_fun(parameter[1])
```

Sample another parameter value using a random walk

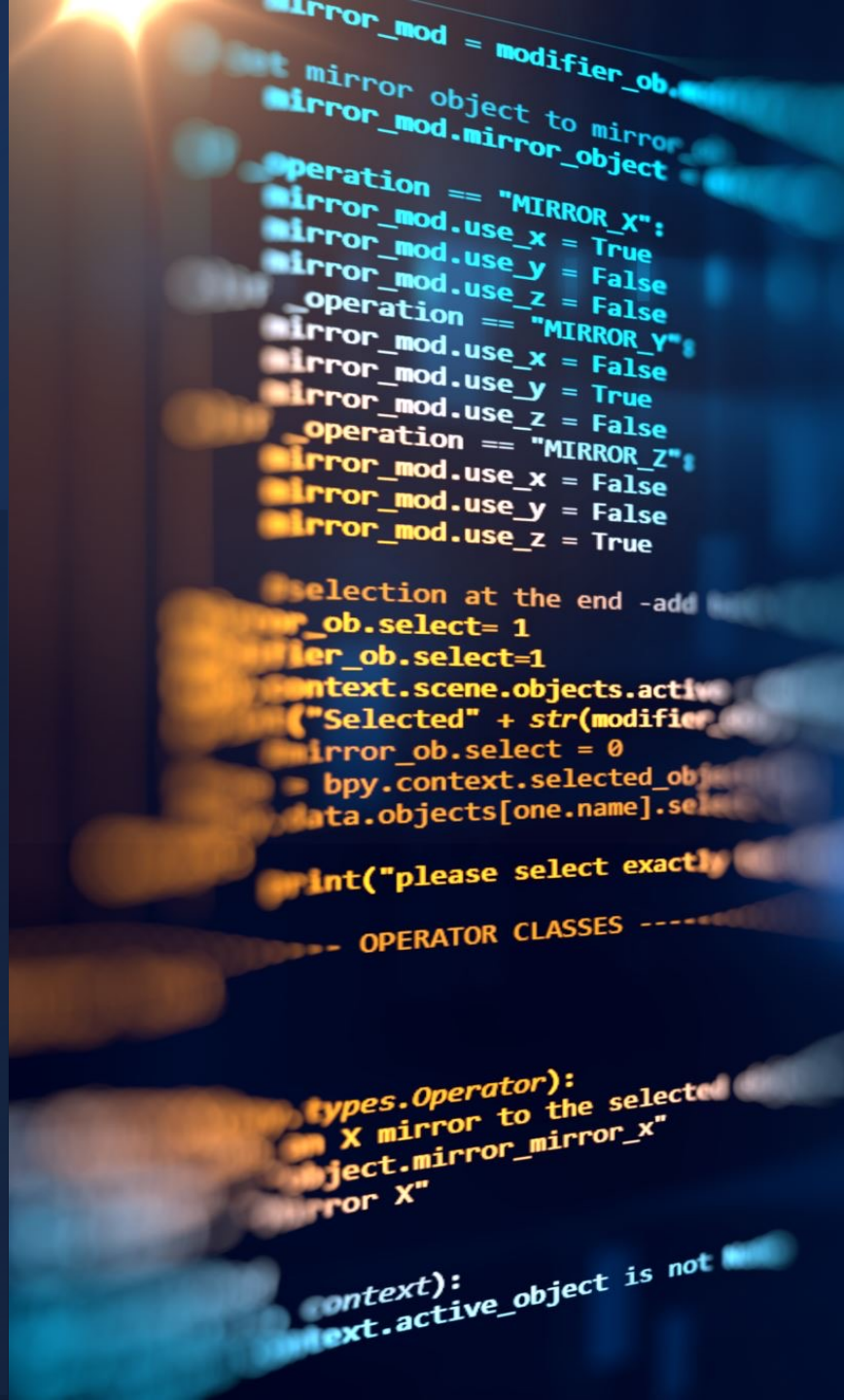
```
for(i in 2:iters){

  # Sample a new parameter value:
  new_par <- rnorm(1, mean=parameter[i-1], sd=sigma)
```

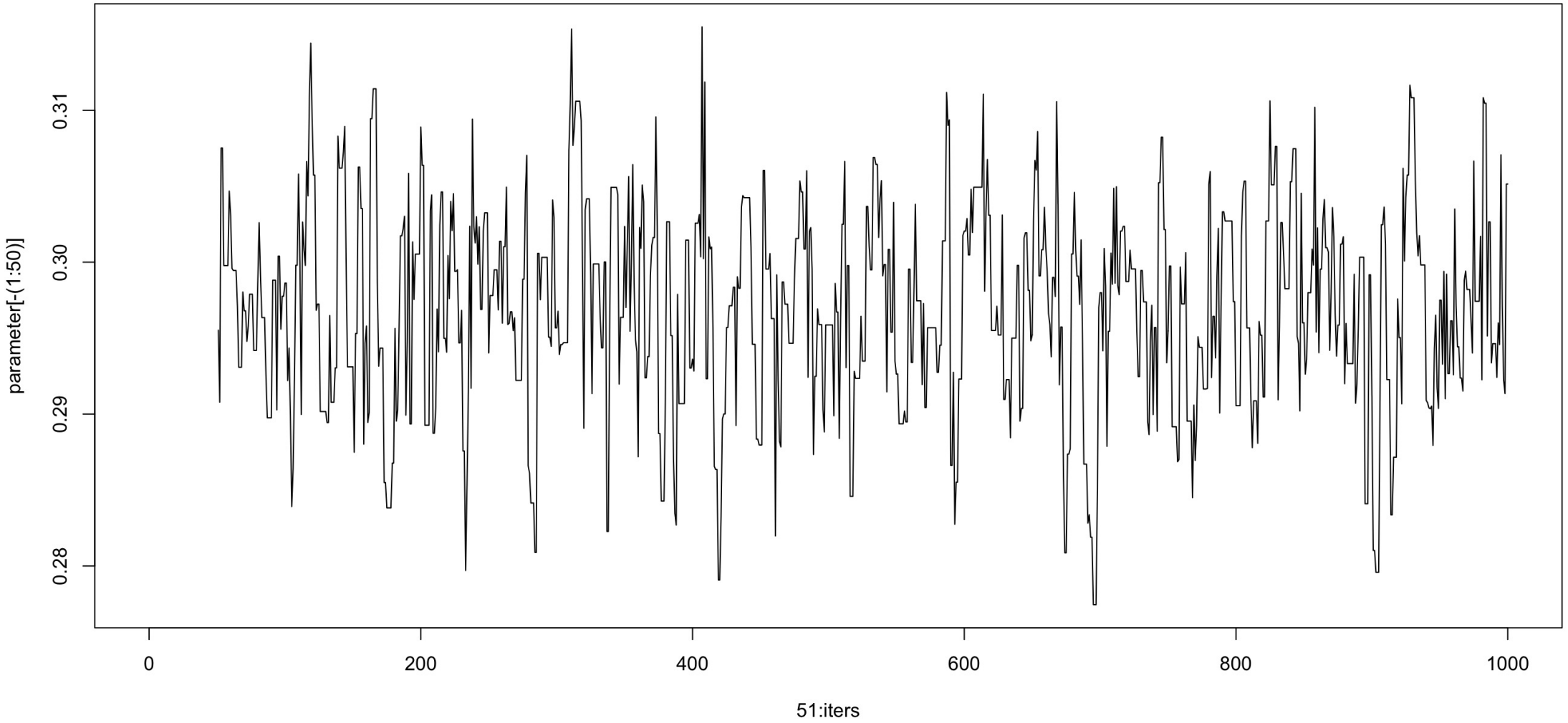
Keep the new value if it’s better than the old

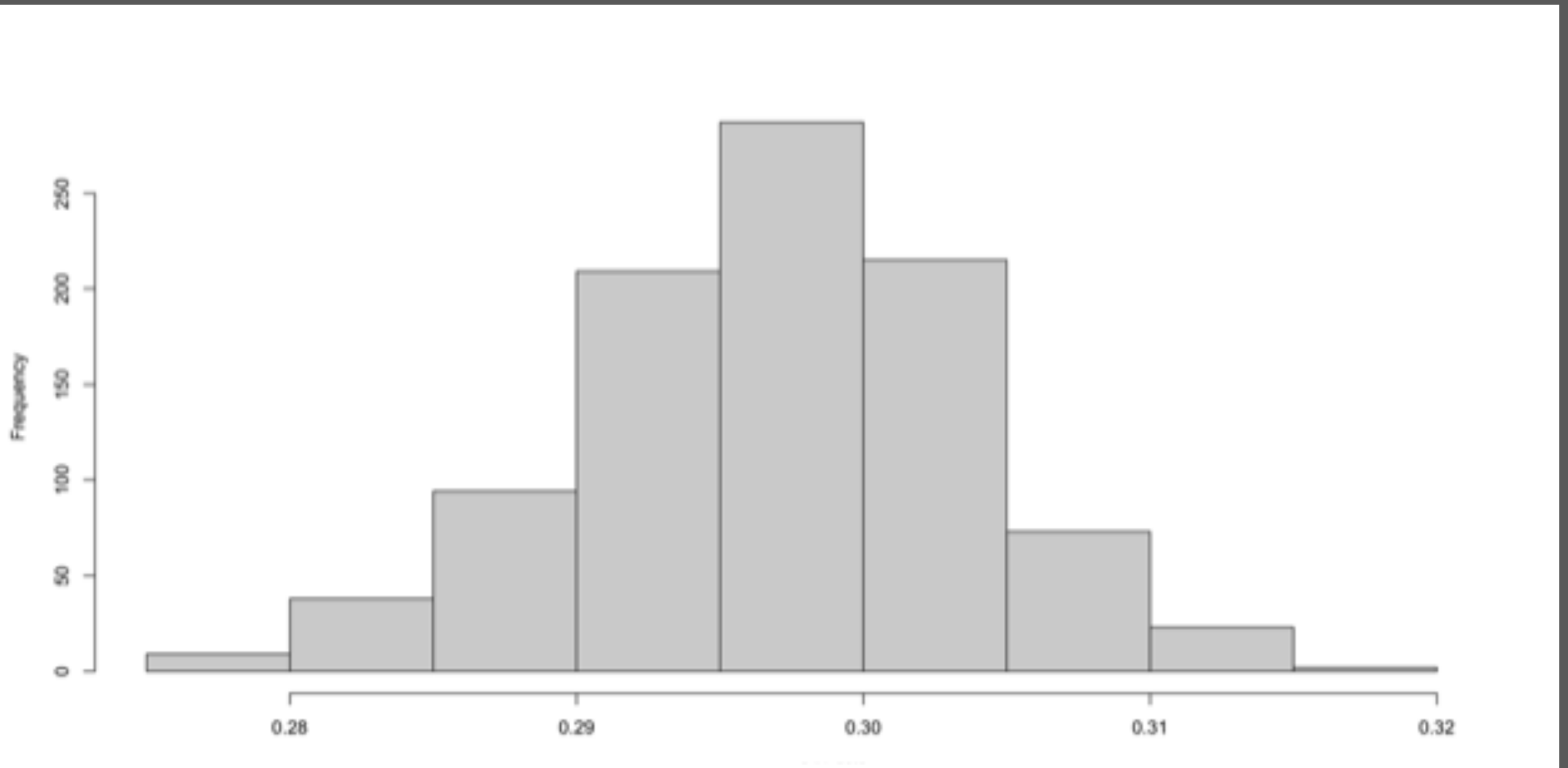
```
new_lpost <- log_posterior_fun(new_par)
   if(new_lpost > log_post[i-1]){
     # If this is an improvement always accept:
     accept <- 1
```

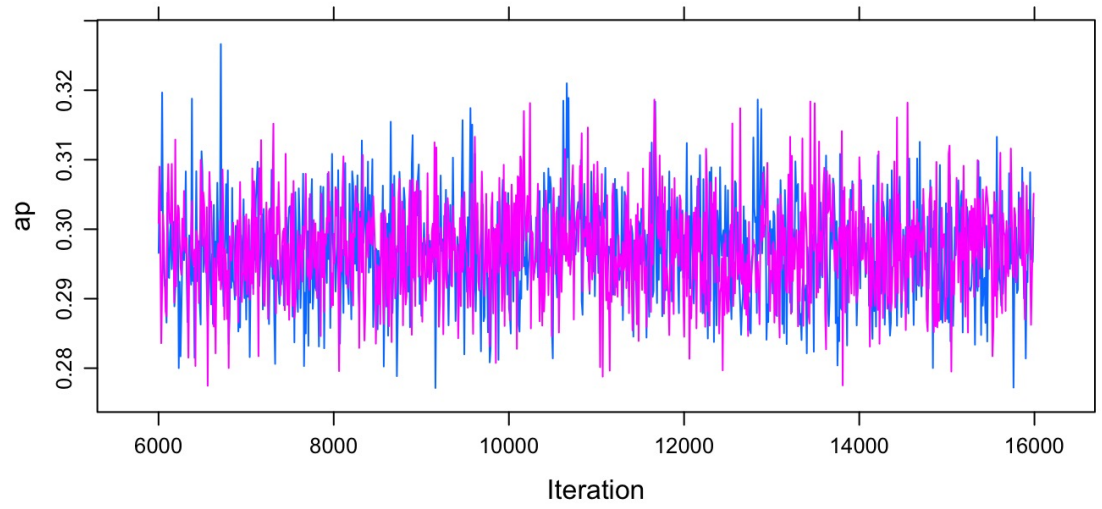
Prepared and Shared by Matt Denwood

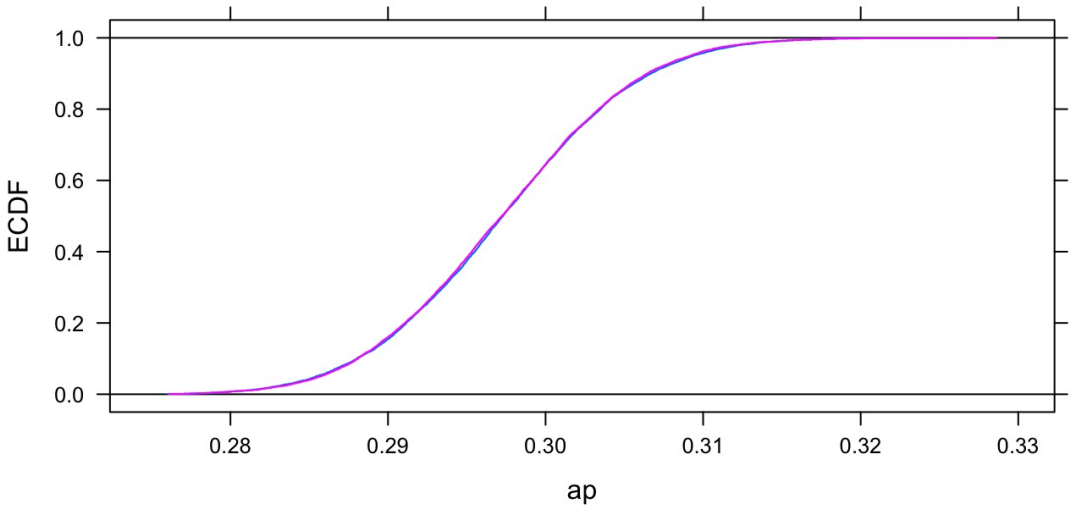
R-code

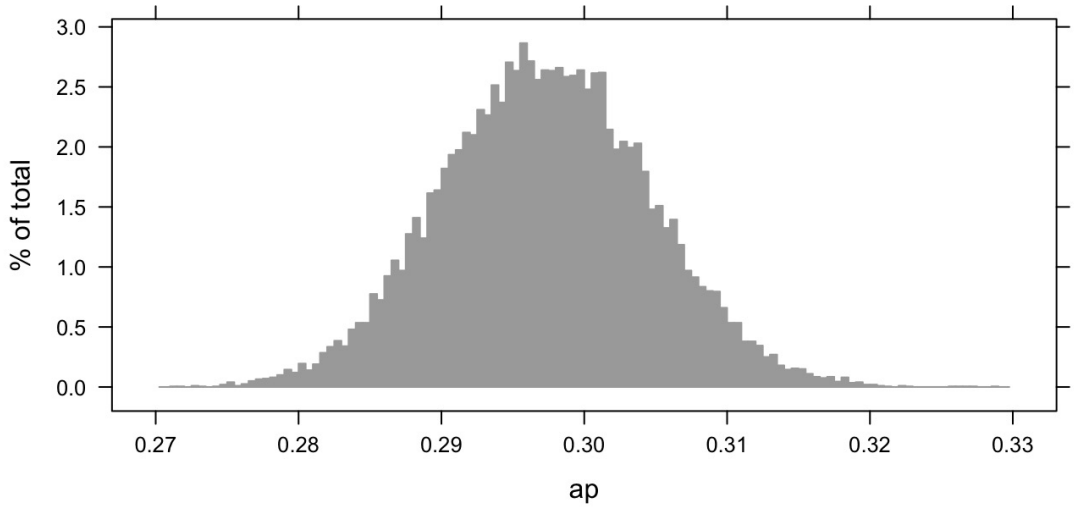


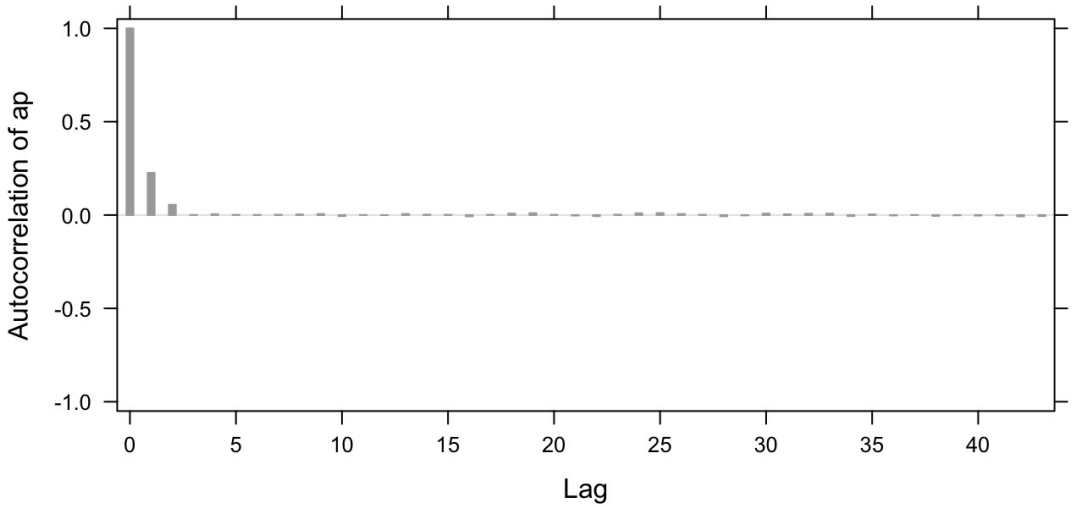

parameter[-(1:50)]
0.28
0.29
0.30
0.31
0
200
400
600
800
1000
51:iters


Frequency
0
50
100
150
200
250
0.28
0.29
0.30
0.31
0.32


ap
0.28
0.29
0.30
0.31
0.32
6000
8000
10000
12000
14000
16000
Iteration


ECDF
0.0
0.2
0.4
0.6
0.8
1.0
0.28
0.29
0.30
0.31
0.32
0.33
ap


% of total
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0.27
0.28
0.29
0.30
0.31
0.32
0.33
ap


Autocorrelation of ap
-1.0
-0.5
0.0
0.5
1.0
0
5
10
15
20
25
30
35
40
Lag

HARMONY

Novel tools for test evaluation and disease prevalence estimation