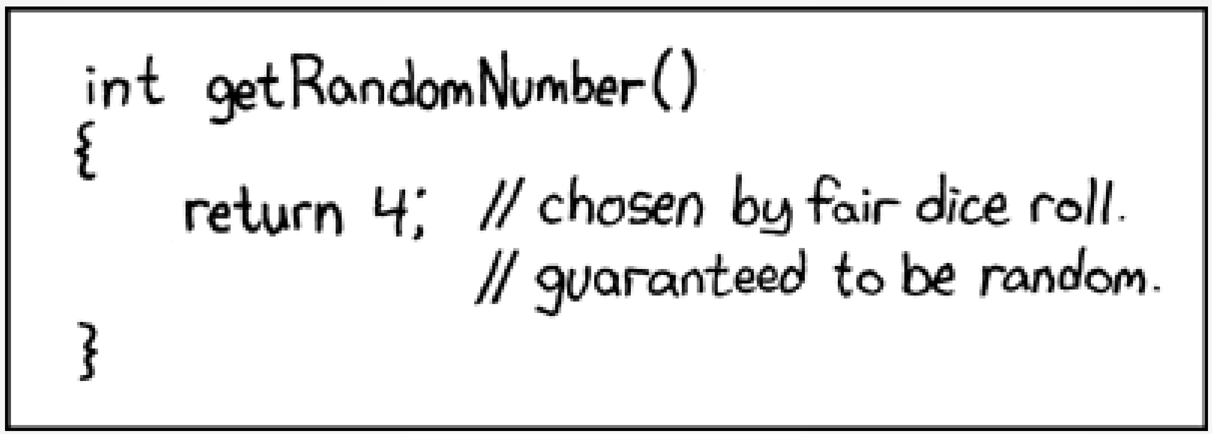
STATS331

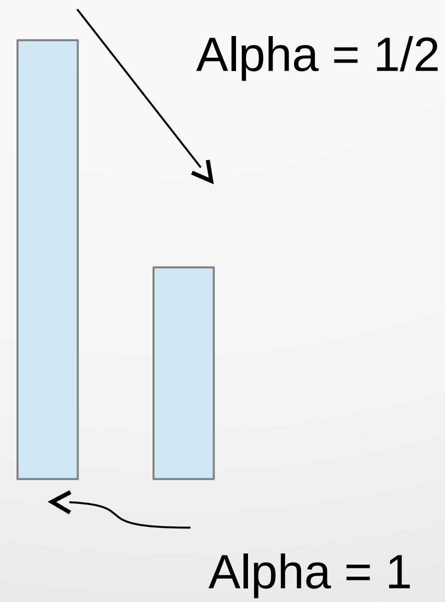


Credit: Randall Monroe (xkcd.com)
Again!

Introduction to Bayesian Statistics
Semester 2, 2016

Metropolis

- Proposal distribution *q(proposed state | current state)*
- If moving to a state of higher prob, accept
- If moving to a state of lower prob, accept with prob $\alpha = h_2/h_1$ *(This assumes symmetric proposal – we'll only use these in 331)*



Tactile MCMC

- In groups of 2-4, do 30 iterations of the Metropolis algorithm

State 1: h_1 = `prior*lik = 0.1`

State 2: h_2 = `prior*lik = 0.3`

Results

- What proportion of “1”s did you get in the final column?
- This is a (hopefully) accurate approximation to the posterior probability of state 1.

Steady State Distribution

- The *steady state distribution* (also called the *stationary distribution*) of a Markov Chain:

- is the probability distribution representing where the algorithm will be after a long time
- is the frequency distribution of results you (probably) get after running the chain for a long time (assuming it's possible to go everywhere)

Using Metropolis on real problems

- Metropolis is usually used on problems with continuous parameter spaces, and with “random walk proposals”

Election Poll Example with Metropolis

- The data: ten people we called and asked whether they support a certain political party

- Results: {0, 1, 1, 1, 0, 0, 1, 0, 1, 1} (six 'yes', four 'no')

- Likelihood = $\theta^6(1-\theta)^4$

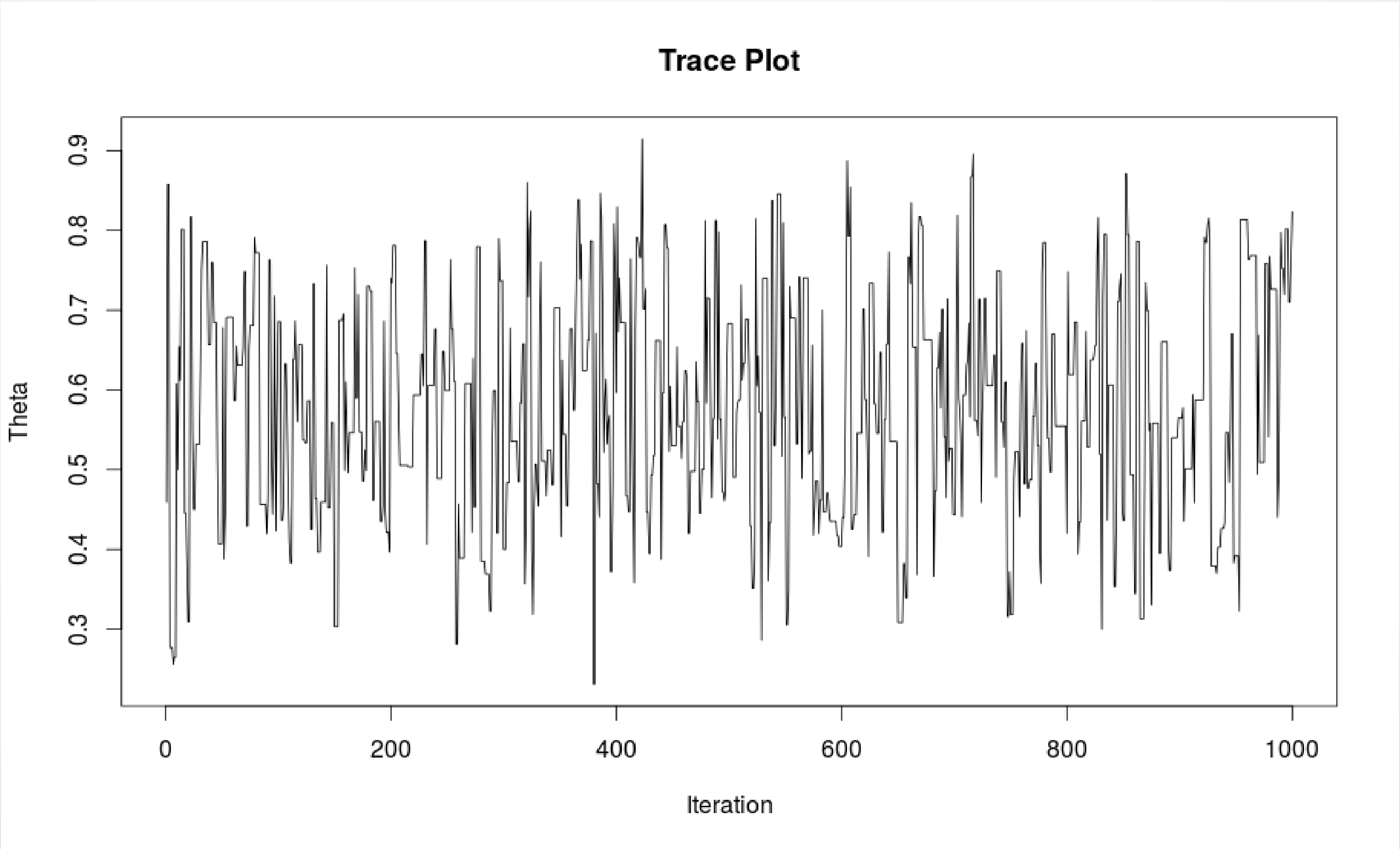
R Code for Metropolis

- This code is on Canvas (`simple_metropolis.R`)
- There is also an industrial strength version

  (`metropolis.R` and `model.R`)

Trace Plots

- These are plots of the parameter(s) moving around over time. Here's a healthy one:



Tuning the proposal

- If the proposal distribution is too wide or too narrow, the Metropolis algorithm can be inefficient
- Let's try various values for the size of the proposal 'jumps'

Tuning the proposal

- Having the proposal width too small is inefficient (accepted steps are very small)
- Having the proposal width too wide is inefficient

  (too many rejected steps)

Image courtesy of the Global Legal Post

An alternative to tuning, for lazy people

- In the ‘industrial strength’ code, I randomise the step size. Steps will be good *sometimes*.

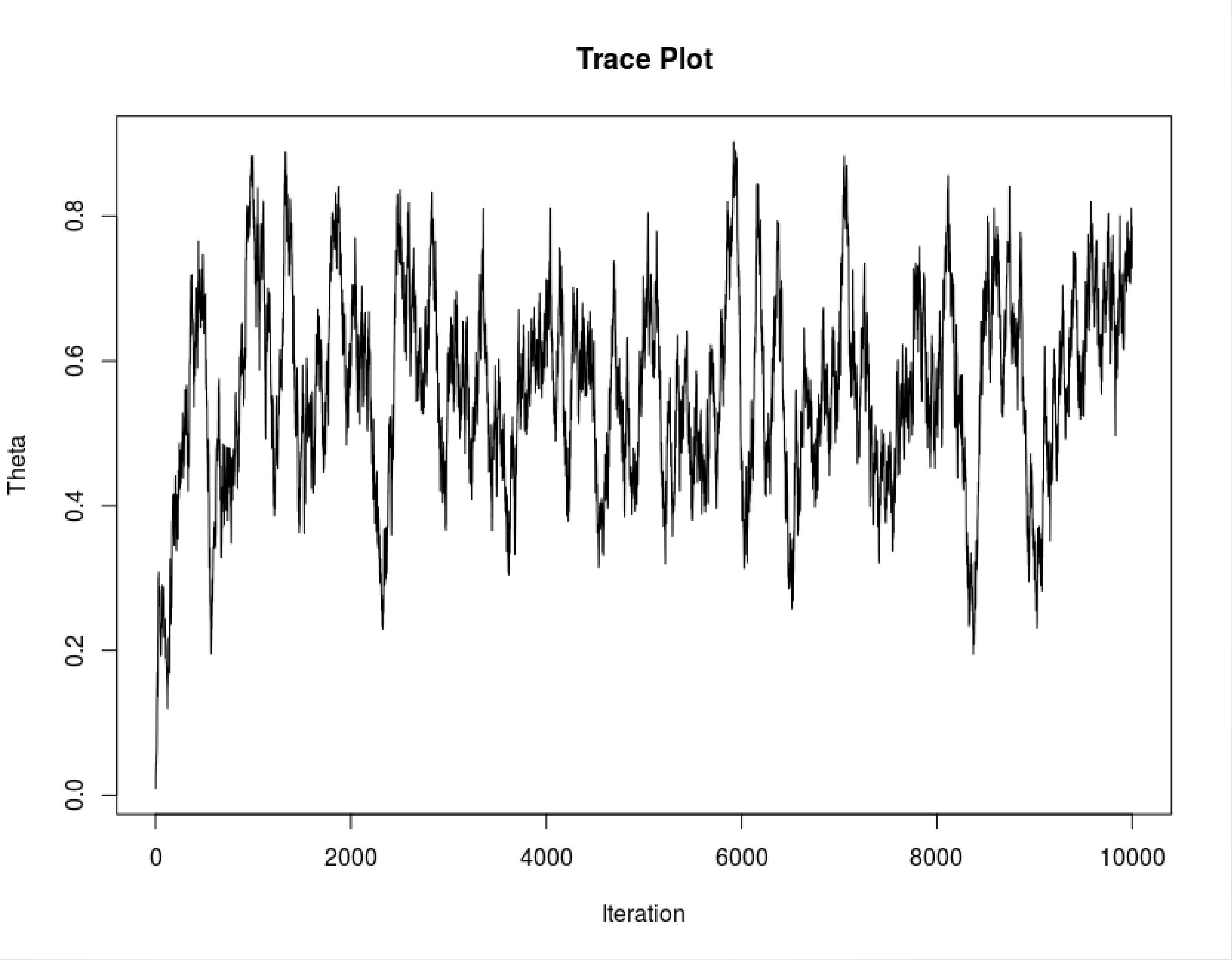
```
# Prior widths for each parameter
widths = c(1)

# Proposal distribution
proposal = function(params)
{
    # Copy the parameters
    params2 = params

    # Which parameter to change?
    i = sample(1:length(params), 1)

    # Step size - Brendon's favourite magic
    step_size = widths[i]*10^(1.5 - 3*abs(rt(1, df=3)))

    params2[i] = params2[i] + step_size*rnorm(1)
    return(params2)
}
```

“Burn In”

- What happens if we start the MCMC in a very low probability region?

- Let’s try it

- A just-working trace plot, with visible burn-in



MCMC in Practice

- While it is important to have a basic understanding of how MCMC works, in practice it's easier to use a software package such as JAGS.
- We will start with JAGS next week.