Statistical Rethinking

Week 6: Markov Chain Monte Carlo

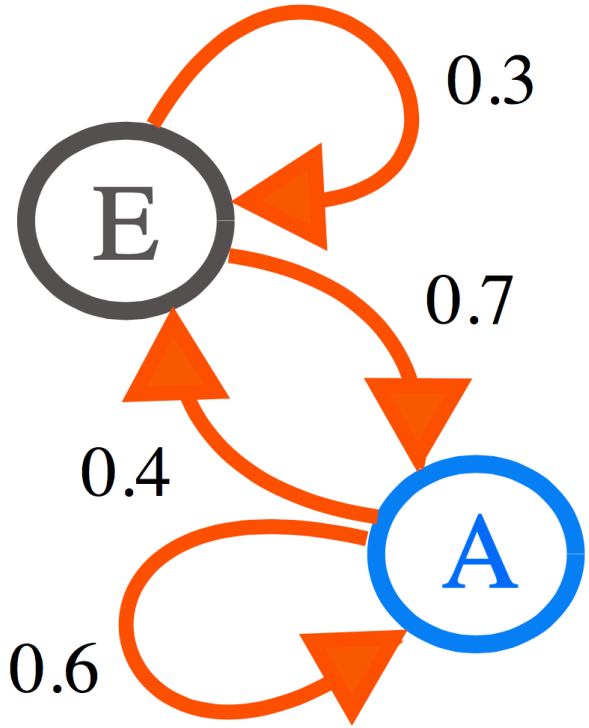
Richard McElreath

Table of random numbers 1 to 100 with no numbers repeated in each block of 25.

75 64 26 45 10	79 18 58 61 09	67 05 60 19 91	14 62 02 35 98	88 51 53 56 96
24 05 89 42 27	98 62 31 19 95	24 25 58 50 49	19 30 31 58 59	49 47 85 48 30
63 18 80 72 41	26 11 91 96 81	55 92 44 23 93	97 89 53 40 80	29 46 34 39 63
38 81 93 68 22	84 92 59 82 80	26 94 73 71 45	63 84 68 44 94	93 64 13 94 31
25 59 54 43 02	16 41 97 40 65	70 29 77 74 27	69 81 70 01 95	82 99 77 80 21
12 28 15 86 98	21 28 92 06 08	33 72 05 13 06	85 65 33 90 20	92 33 27 59 49
36 59 95 67 96	25 72 30 41 81	71 92 18 65 17	64 58 56 89 28	69 18 36 06 71
91 72 33 68 11	22 20 15 01 65	34 60 47 16 09	44 45 46 97 83	44 51 98 67 29
86 04 47 43 69	12 85 04 93 74	80 08 57 25 79	72 96 07 57 40	82 62 68 60 73
01 05 65 97 77	96 64 98 62 49	07 19 63 46 66	77 98 80 54 60	97 32 83 74 80
26 95 96 93 87	17 59 90 35 94	73 68 03 27 29	49 64 66 14 65	57 24 45 76 39
45 27 71 62 05	71 18 32 42 91	25 66 46 49 71	67 11 25 23 12	41 47 99 66 01
74 07 90 20 25	05 52 65 84 92	87 57 95 37 83	85 45 22 56 26	10 28 04 88 49
77 99 91 43 02	96 06 07 36 68	17 48 06 09 84	31 86 91 87 96	63 87 32 33 70
75 53 35 46 41	21 95 85 61 46	94 18 78 39 47	19 60 48 15 59	68 79 42 09 67
45 65 84 36 28	48 33 82 62 71	74 48 75 92 34	32 94 26 70 88	35 50 19 97 52
81 74 60 90 46	13 51 24 54 55	45 54 12 90 99	44 68 86 71 58	27 51 81 11 77
95 11 96 85 83	93 53 74 52 97	79 53 21 41 44	45 81 02 38 07	38 07 80 89 56
29 40 82 33 86	67 95 43 41 89	05 52 17 31 13	82 61 78 57 40	84 39 57 63 78
79 14 32 21 09	32 27 02 70 20	61 47 24 42 76	77 27 99 36 15	36 98 08 40 53
51 46 23 17 11	93 35 70 37 86	26 23 64 88 17	17 78 95 93 83	65 23 90 78 55
98 75 60 99 89	91 18 20 27 74	31 82 01 32 97	97 43 21 87 82	33 28 10 56 98
15 97 42 56 79	08 58 79 40 31	37 19 20 58 41	41 86 66 54 45	08 76 89 86 32
06 16 35 93 26	36 97 26 17 71	74 95 89 08 50	50 62 48 46 26	24 95 93 01 64
54 43 55 21 74	47 59 75 03 57	63 38 02 51 77	77 76 65 08 92	72 29 35 06 85
66 31 33 83 19	15 01 38 69 66	77 83 87 16 45	04 07 72 32 08	53 91 03 48 49
06 07 88 09 61	19 29 39 18 16	76 48 53 81 12	61 39 87 60 33	84 75 78 22 55
57 01 84 02 27	11 14 47 20 44	22 34 90 86 79	89 68 71 46 77	08 76 89 86 32
47 08 89 24 85	87 13 48 68 94	07 70 88 03 36	75 92 73 05 56	62 37 77 34 42
17 05 93 51 30	82 49 61 45 31	91 55 23 11 89	53 15 34 76 78	33 41 99 79 43
15 19 85 03 11	81 76 26 77 13	73 75 64 47 85	08 61 70 03 25	90 92 94 98 97
91 64 24 16 46	23 44 70 47 17	10 70 43 35 56	67 73 71 90 57	37 34 54 95 35
70 09 43 21 61	24 74 07 96 33	08 42 19 74 12	09 27 77 23 17	93 43 14 38 15
62 94 51 92 60	49 25 15 85 34	86 09 11 03 96	47 54 02 32 76	75 13 76 32 03
53 13 59 22 82	87 37 94 62 65	18 40 14 38 71	41 55 14 50 28	62 74 08 31 58
93 59 48 96 88	04 83 14 84 53	45 70 37 18 05	79 14 45 55 46	28 55 36 35 77
58 14 07 89 30	51 76 38 05 32	13 01 23 63 33	24 73 13 21 16	46 78 20 67 32
47 40 60 22 29	52 16 70 44 19	46 41 93 73 78	68 88 42 02 28	66 17 83 37 38
28 02 81 52 80	56 08 63 06 22	35 50 32 75 22	66 69 65 97 35	87 65 33 29 10
69 24 61 41 42	24 73 45 55 46	47 21 95 09 62	86 67 29 74 54	95 14 74 72 79

Goals this week

- Introduce Markov chain Monte Carlo
  - Understand the approach
  - Meet different algorithms
- Meet interface to MCMC, `map2stan`
- How to check chains
- How to fix common problems
- Meet maximum entropy & GLMs

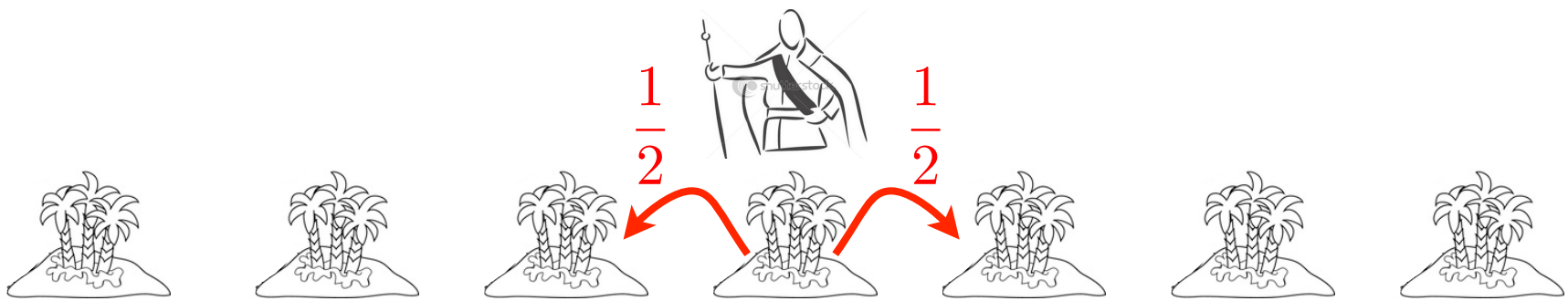


King Markov

The Metropolis Archipelago

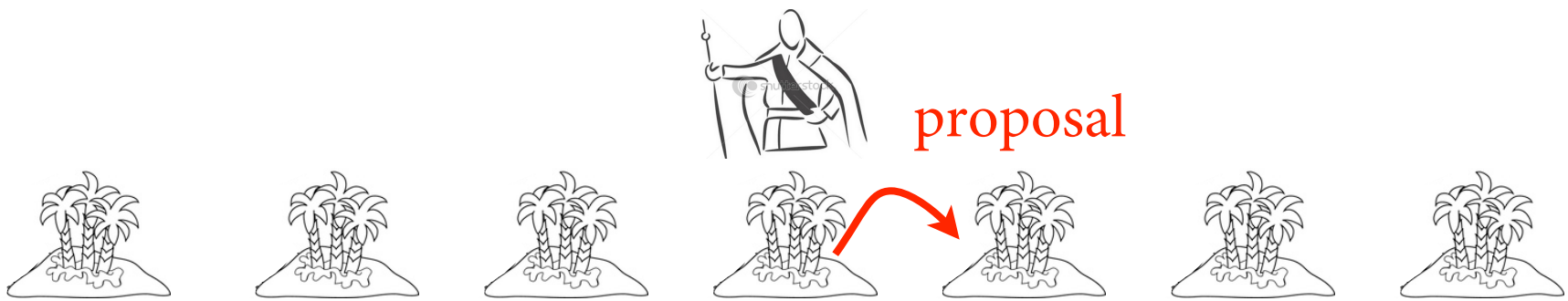
Contract: King Markov must visit each island in proportion to its population size.

Here's how he does it...



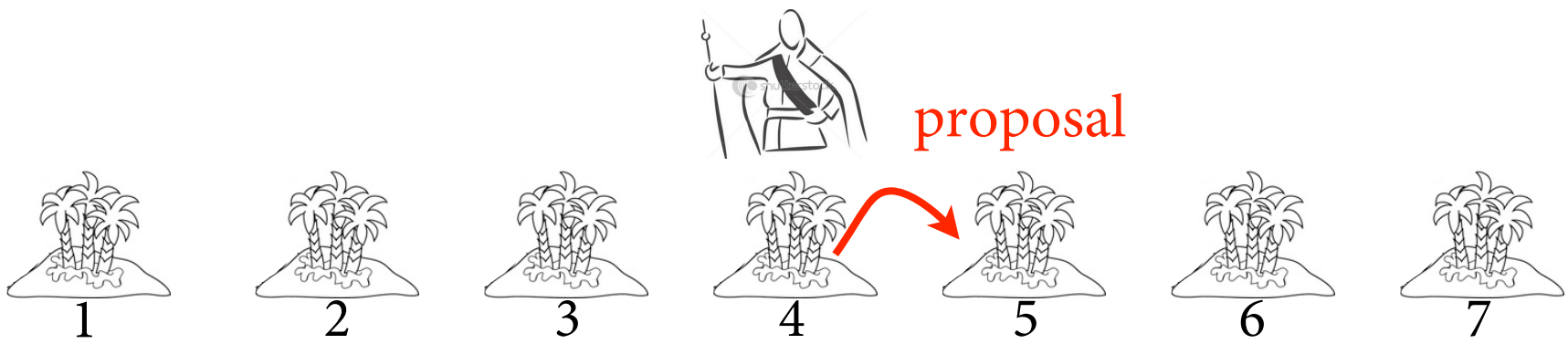
(1) Flip a coin to choose island on left or right.
Call it the “proposal” island.

(1) Flip a coin to choose island on left or right.
Call it the “proposal” island.



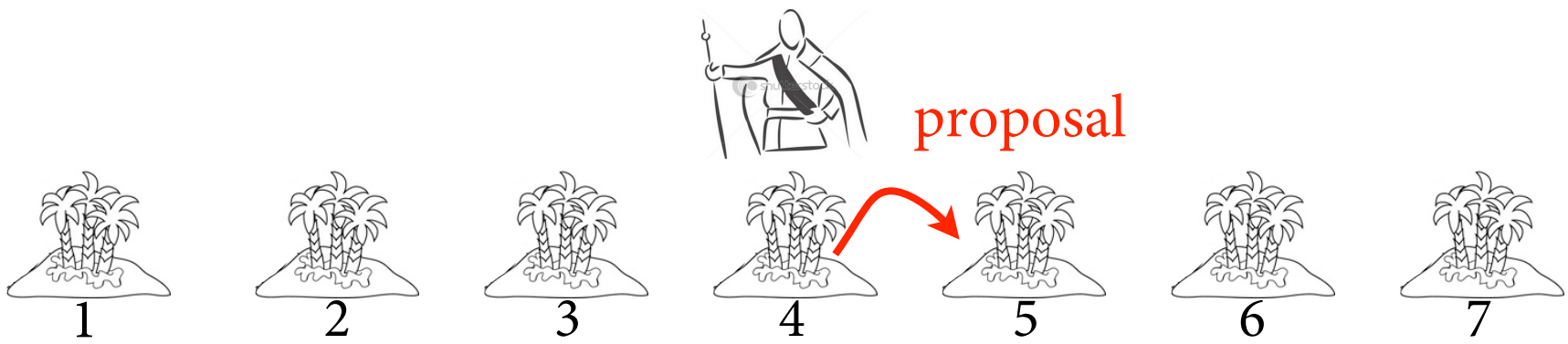
(2) Find population of proposal island.

(1) Flip a coin to choose island on left or right.
Call it the "proposal" island.



(2) Find population of proposal island.

(1) Flip a coin to choose island on left or right.
Call it the "proposal" island.

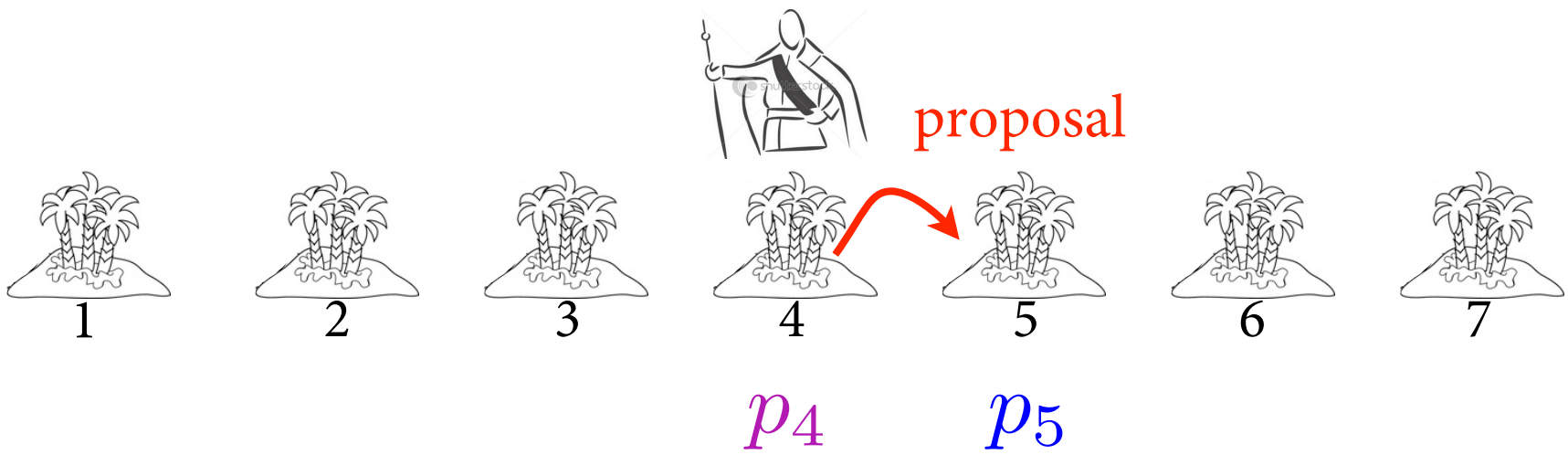


p_5

(2) Find population of proposal island.

(1) Flip a coin to choose island on left or right.
Call it the "proposal" island.

(2) Find population of proposal island.

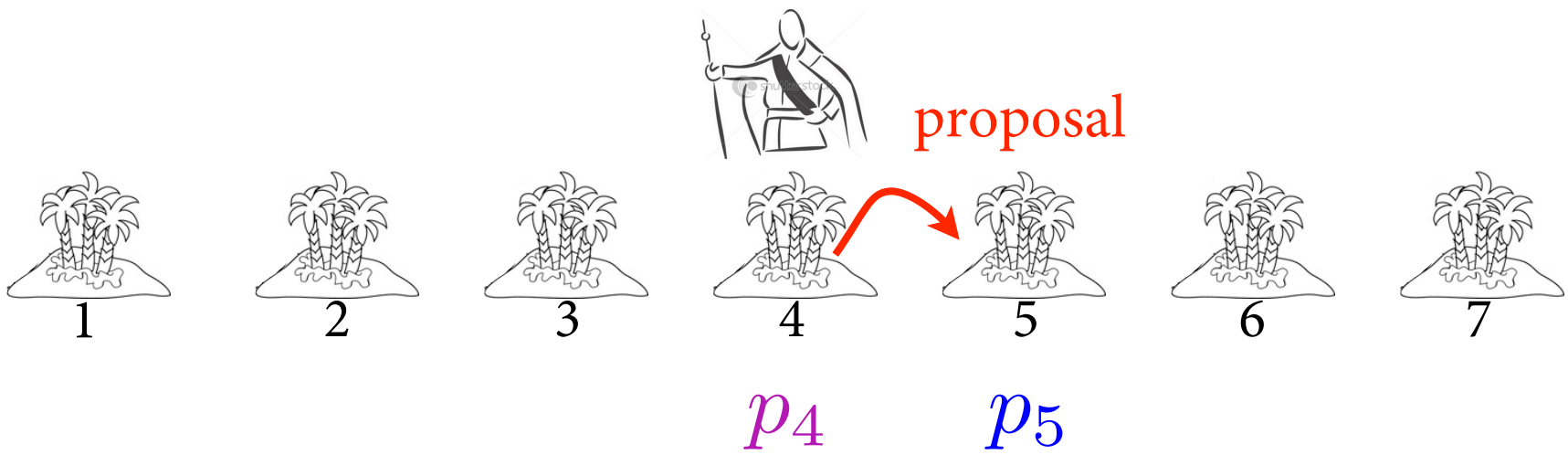


(3) Find population of current island.

(1) Flip a coin to choose island on left or right.
Call it the "proposal" island.

(2) Find population of proposal island.

(3) Find population of current island.



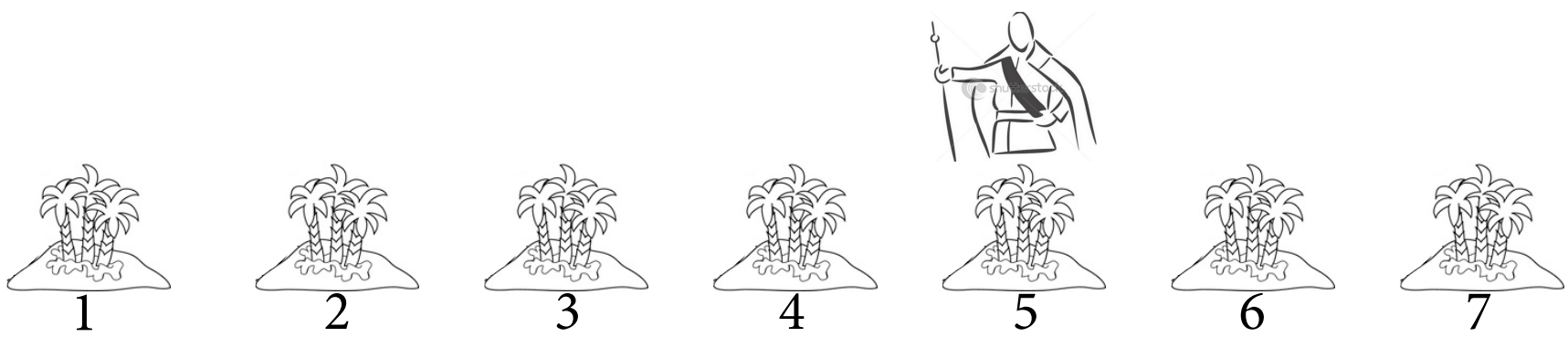
(4) Move to proposal, with probability = $\frac{p_5}{p_4}$

(1) Flip a coin to choose island on left or right.
Call it the "proposal" island.

(2) Find population of proposal island.

(3) Find population of current island.

(4) Move to proposal, with probability = $\frac{p_5}{p_4}$



(5) Repeat from (1)

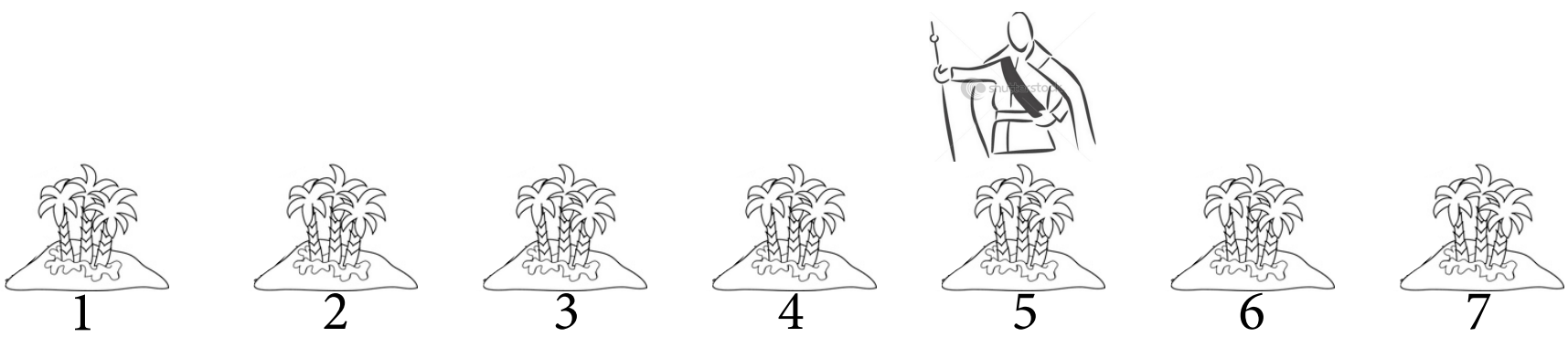
(1) Flip a coin to choose island on left or right.
Call it the "proposal" island.

(2) Find population of proposal island.

(3) Find population of current island.

(4) Move to proposal, with probability = $\frac{p_5}{p_4}$

(5) Repeat from (1)



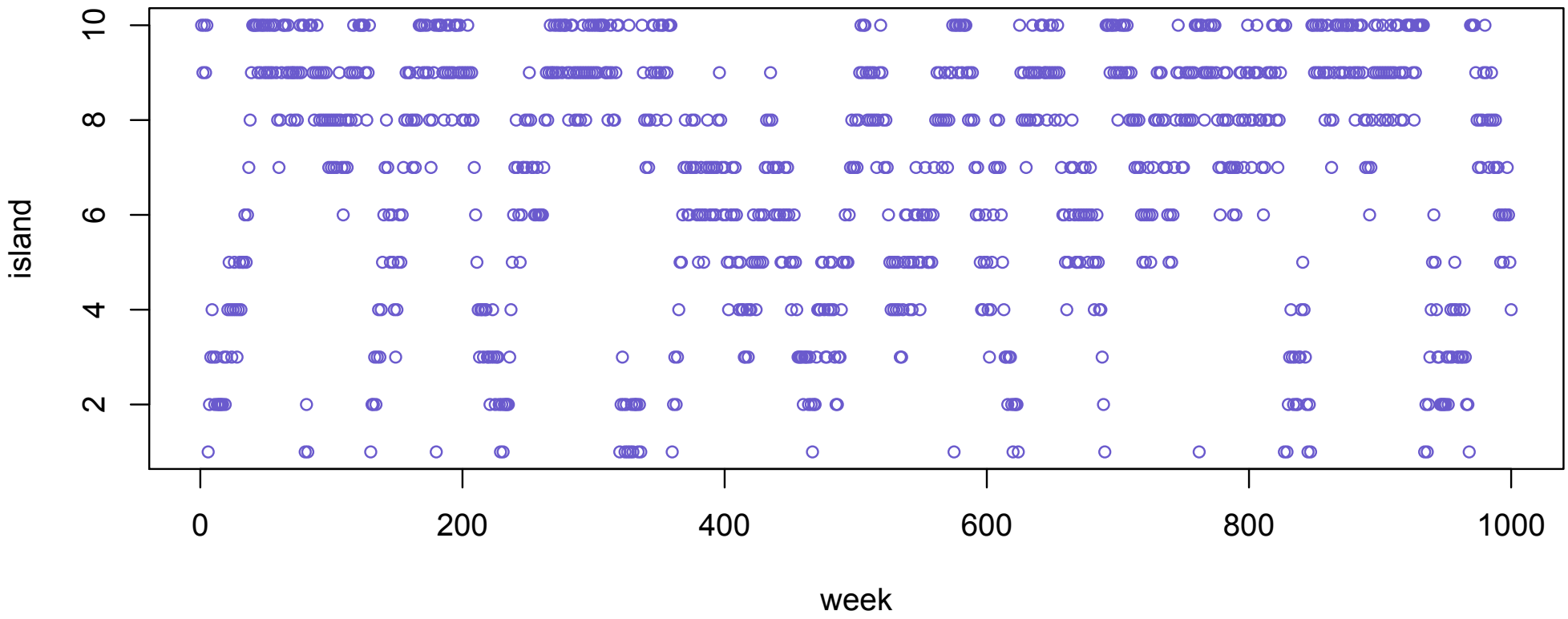
This procedure ensures visiting each island in proportion to its population, *in the long run.*

Metropolis algorithm

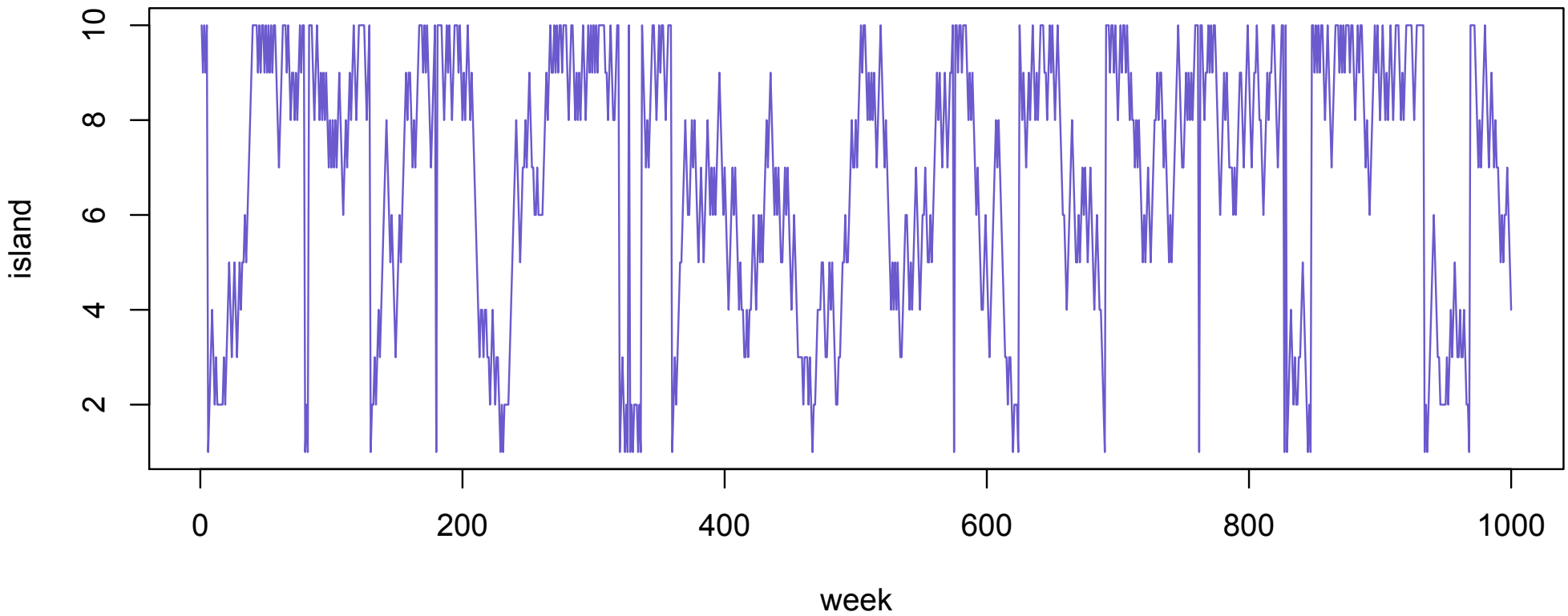
R code 8.1

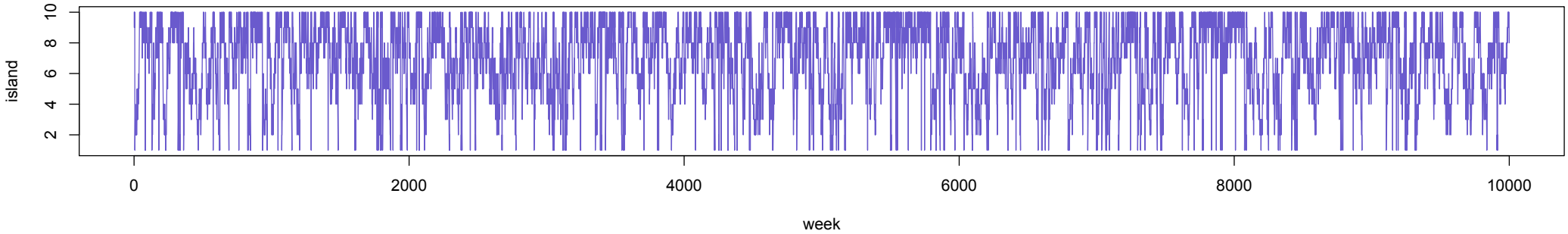
```
num.weeks <- 1e4
positions <- rep(0,num.weeks)
current <- 10
for ( i in 1:num.weeks ) {
    # record current position
    positions[i] <- current

    # flip coin to generate proposal
    proposal <- current + sample( c(-1,1) , size=1 )
    # now make sure he loops around the archipelago
    if ( proposal < 1 ) proposal <- 10
    if ( proposal > 10 ) proposal <- 1

    # move?
    prob.move <- proposal/current
    current <- ifelse( runif(1)<prob.move , proposal , current )
}
```

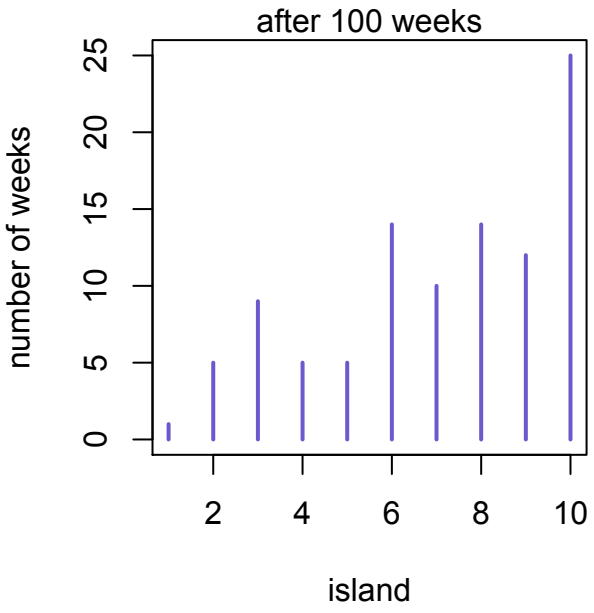
Markov's chain of visits

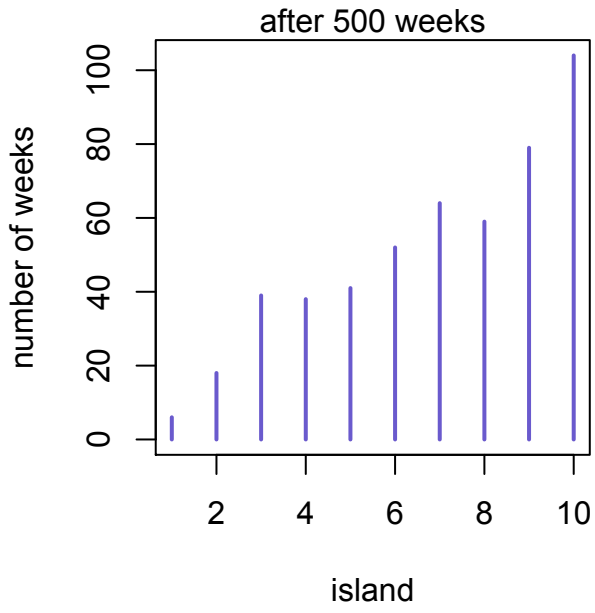


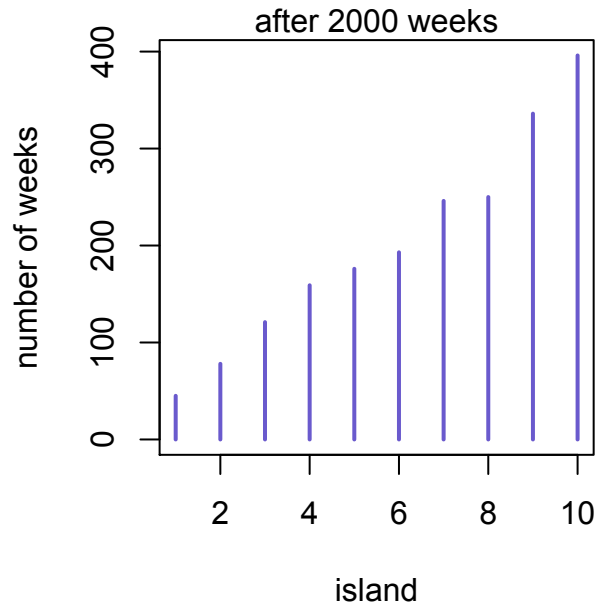
Markov's chain of visits




island
2
4
6
8
10
0
2000
4000
6000
8000
10000
week

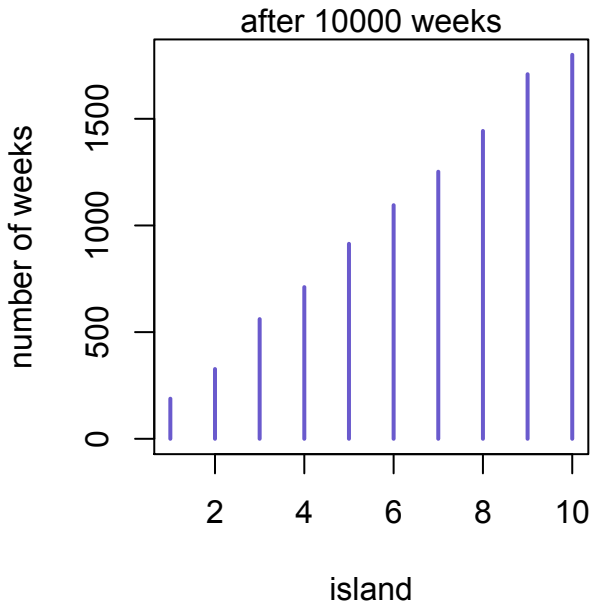

after 100 weeks
number of weeks
0
5
10
15
20
25
2
4
6
8
10
island


after 500 weeks
number of weeks
0
20
40
60
80
100
2
4
6
8
10
island


after 2000 weeks
number of weeks
0
100
200
300
400
2
4
6
8
10
island

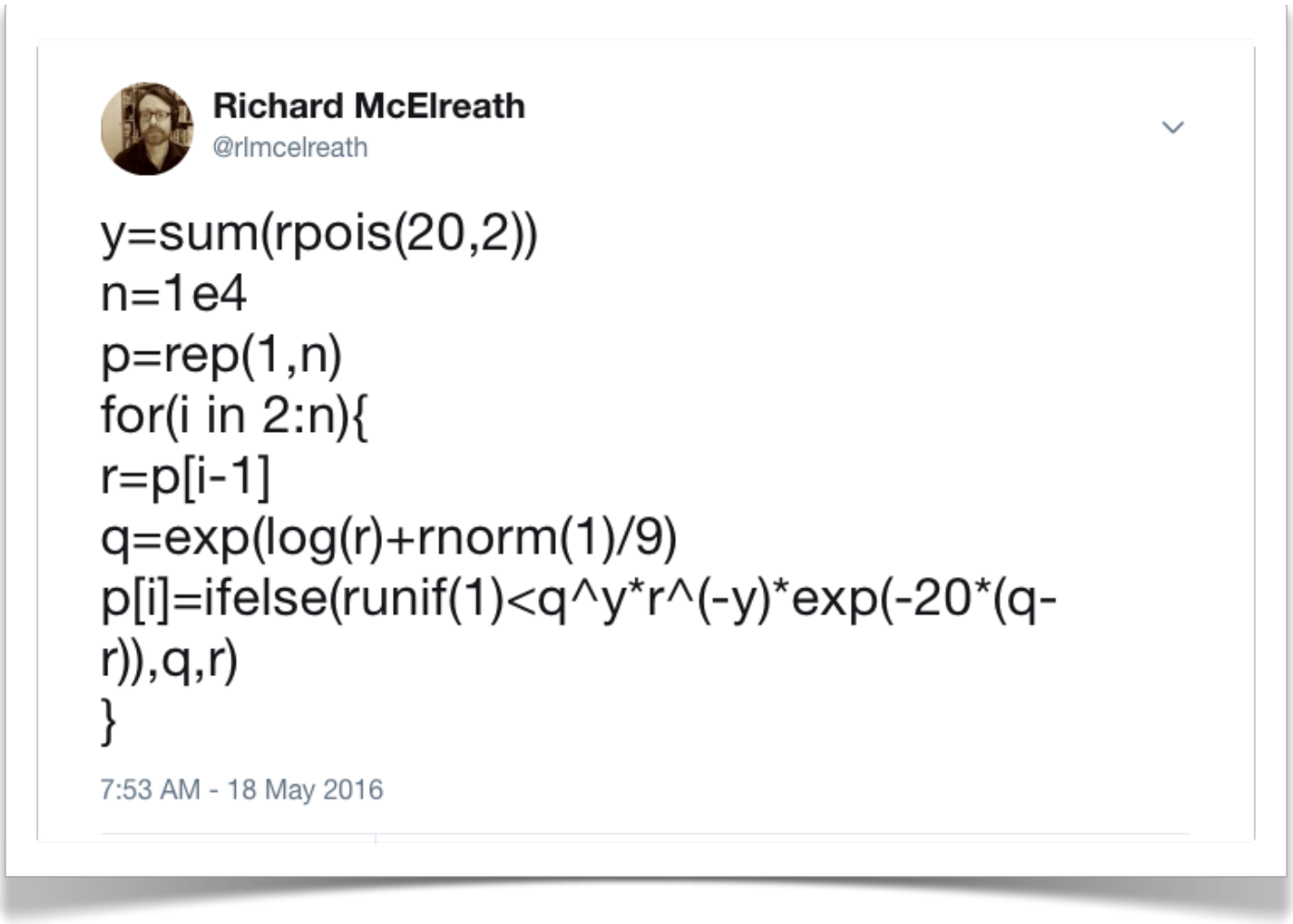
Markov's chain of visits

- Converges to correct proportions, in the long run
- No matter which island starts
- As long as proposals are *symmetric*
- Example of *Metropolis algorithm*



Metropolis and MCMC

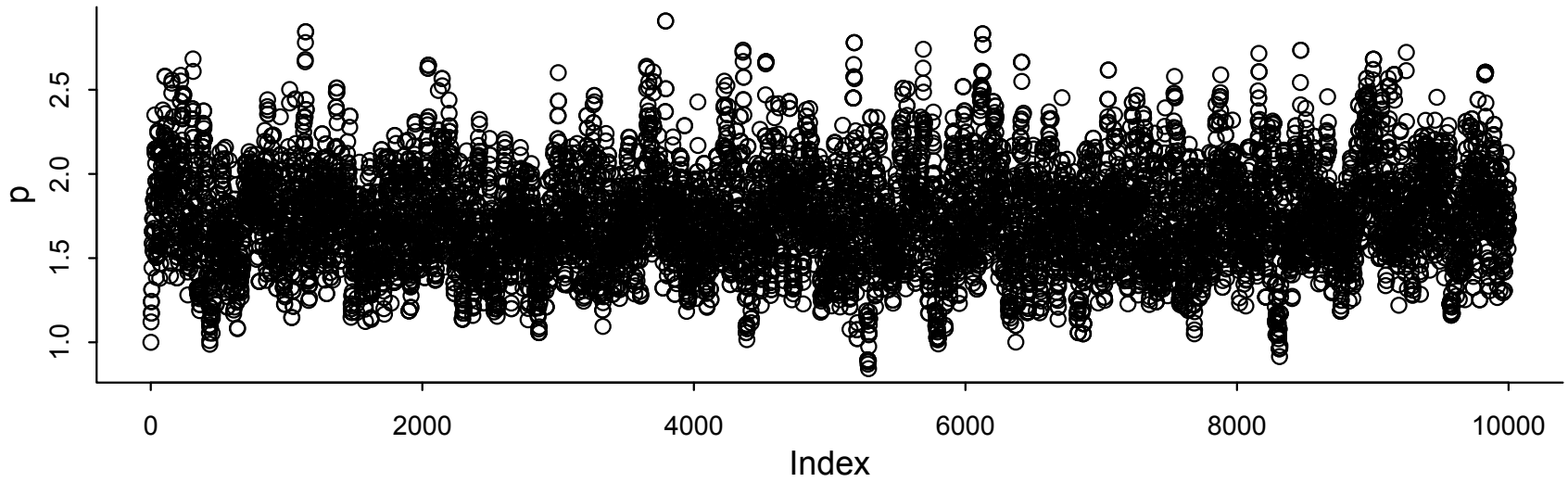
- Usual use is to draw samples from a posterior distribution
  - "Islands": parameter values
  - "Population size": proportional to posterior probability
- Works for any number of dimensions (parameters)
- Works for continuous as well as discrete parameters



Richard McElreath
@rlmcelreath

```
y=sum(rpois(20,2))
n=1e4
p=rep(1,n)
for(i in 2:n){
r=p[i-1]
q=exp(log(r)+rnorm(1)/9)
p[i]=ifelse(runif(1)<q^y*r^(-y)*exp(-20*(q-
r)),q,r)
}
```

7:53 AM - 18 May 2016



Metropolis and MCMC

- *Metropolis*: Simple version of *Markov chain Monte Carlo* (MCMC)
- Metropolis, Rosenbluth, Rosenbluth, Teller, and Teller (1953)

THE JOURNAL OF CHEMICAL PHYSICS VOLUME 21, NUMBER 6 JUNE, 1953

Equation of State Calculations by Fast Computing Machines

NICHOLAS METROPOLIS, ARIANNA W. ROSENBLUTH, MARSHALL N. ROSENBLUTH, AND AUGUSTA H. TELLER,
Los Alamos Scientific Laboratory, Los Alamos, New Mexico

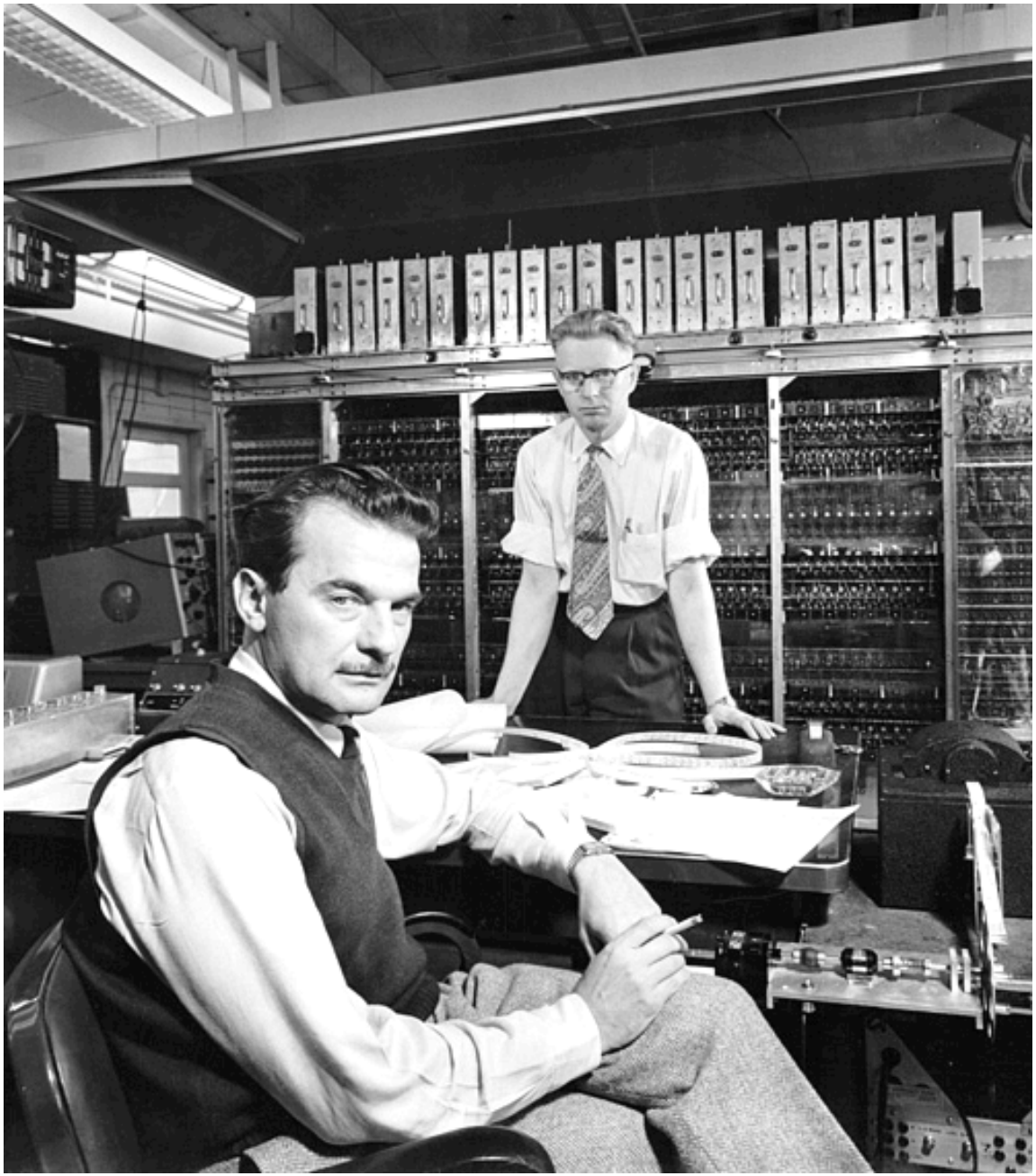
AND

EDWARD TELLER,* *Department of Physics, University of Chicago, Chicago, Illinois*

(Received March 6, 1953)

A general method, suitable for fast computing machines, for investigating such properties as equations of state for substances consisting of interacting individual molecules is described. The method consists of a modified Monte Carlo integration over configuration space. Results for the two-dimensional rigid-sphere system have been obtained on the Los Alamos MANIAC and are presented here. These results are compared to the free volume equation of state and to a four-term virial coefficient expansion.

MANIAC:
Mathematical Analyzer, Numerical Integrator, and Computer



MANIAC:
1000 pounds
5 kilobytes of memory
70k multiplications/sec

Your laptop:
4–7 pounds
2–8 million kilobytes
Billions of multiplications/sec

Metropolis and MCMC

- *Metropolis*: Simple version of *Markov chain Monte Carlo* (MCMC)
- *Chain*: Sequence of draws from distribution
- *Markov chain*: History doesn't matter, just where you are now
- *Monte Carlo*: Random simulation

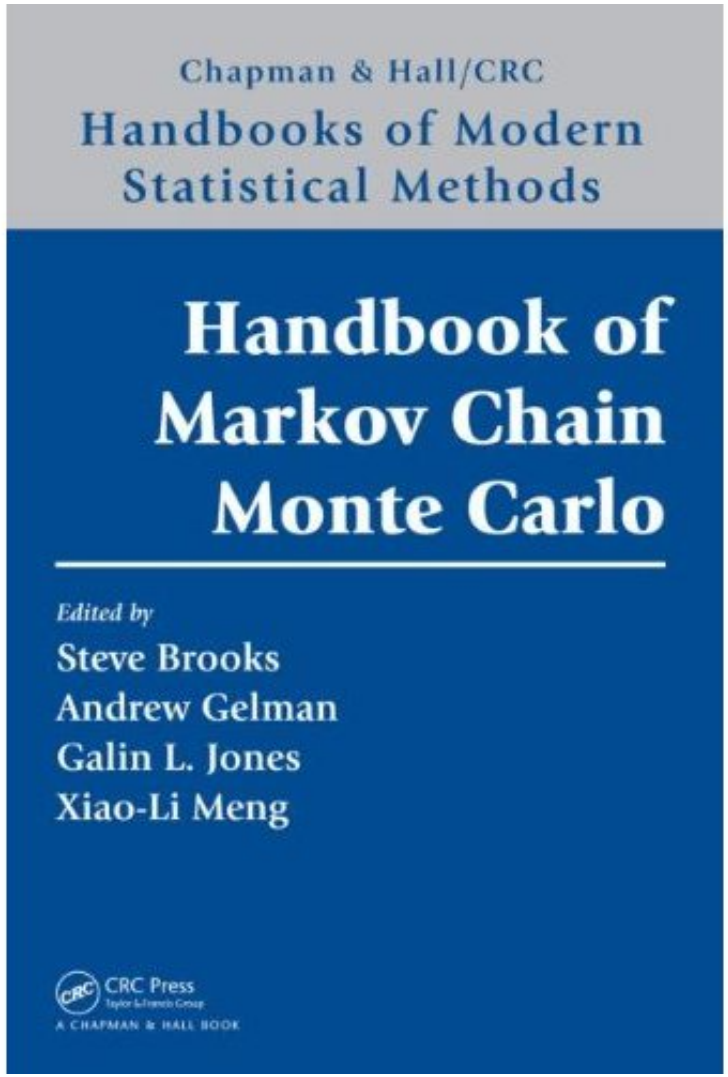
Andrei Andreyevich Markov
(Ма́рков)
(1856–1922)

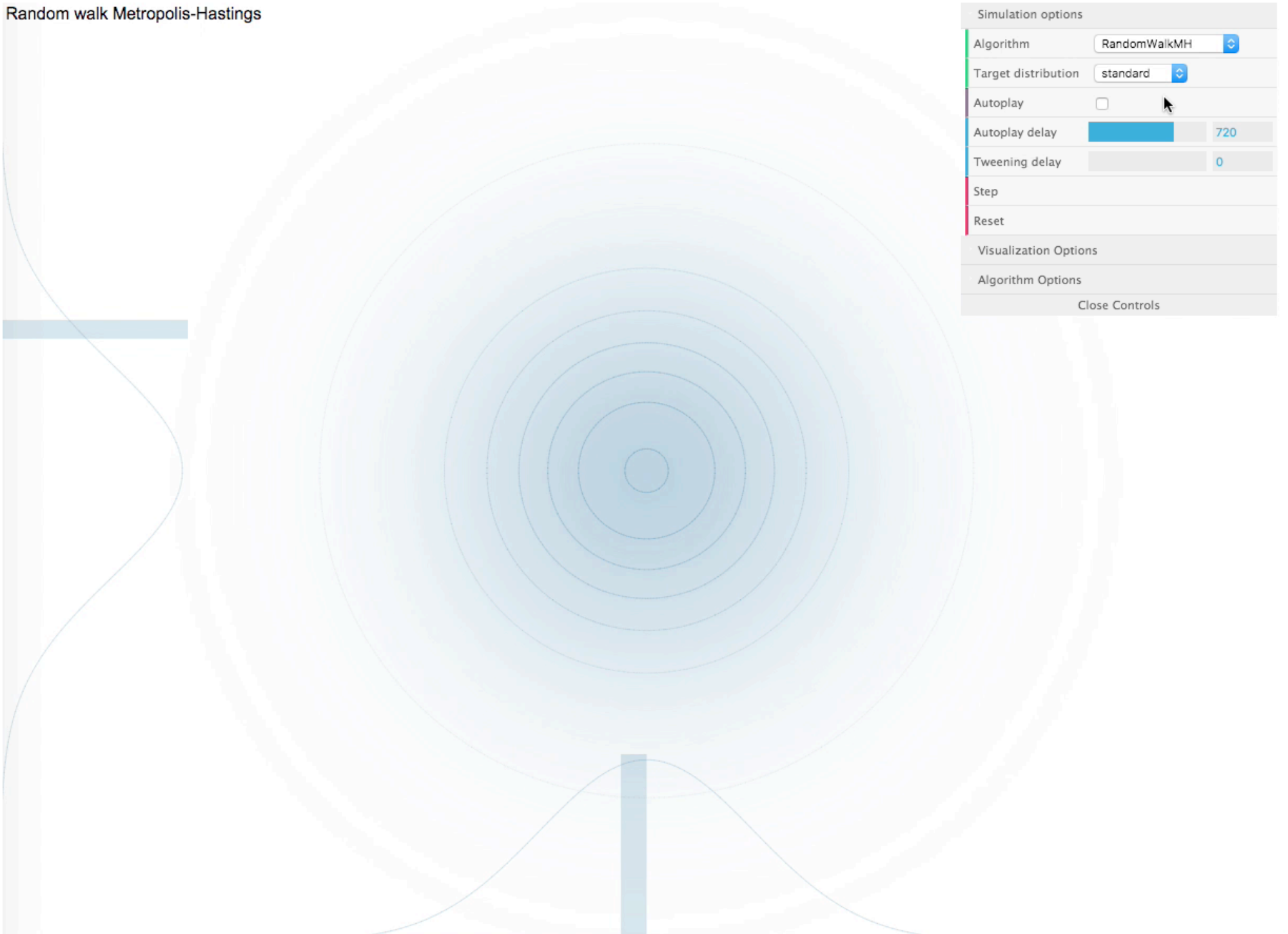
Why MCMC?

- Sometimes can't write an integrated posterior
- Even when can, often cannot use it
- Many problems are like this:
  - Many multilevel models
  - Networks/phylogenies
  - Some spatial models
- Maximization not a good strategy in high dimensions — must have full distribution

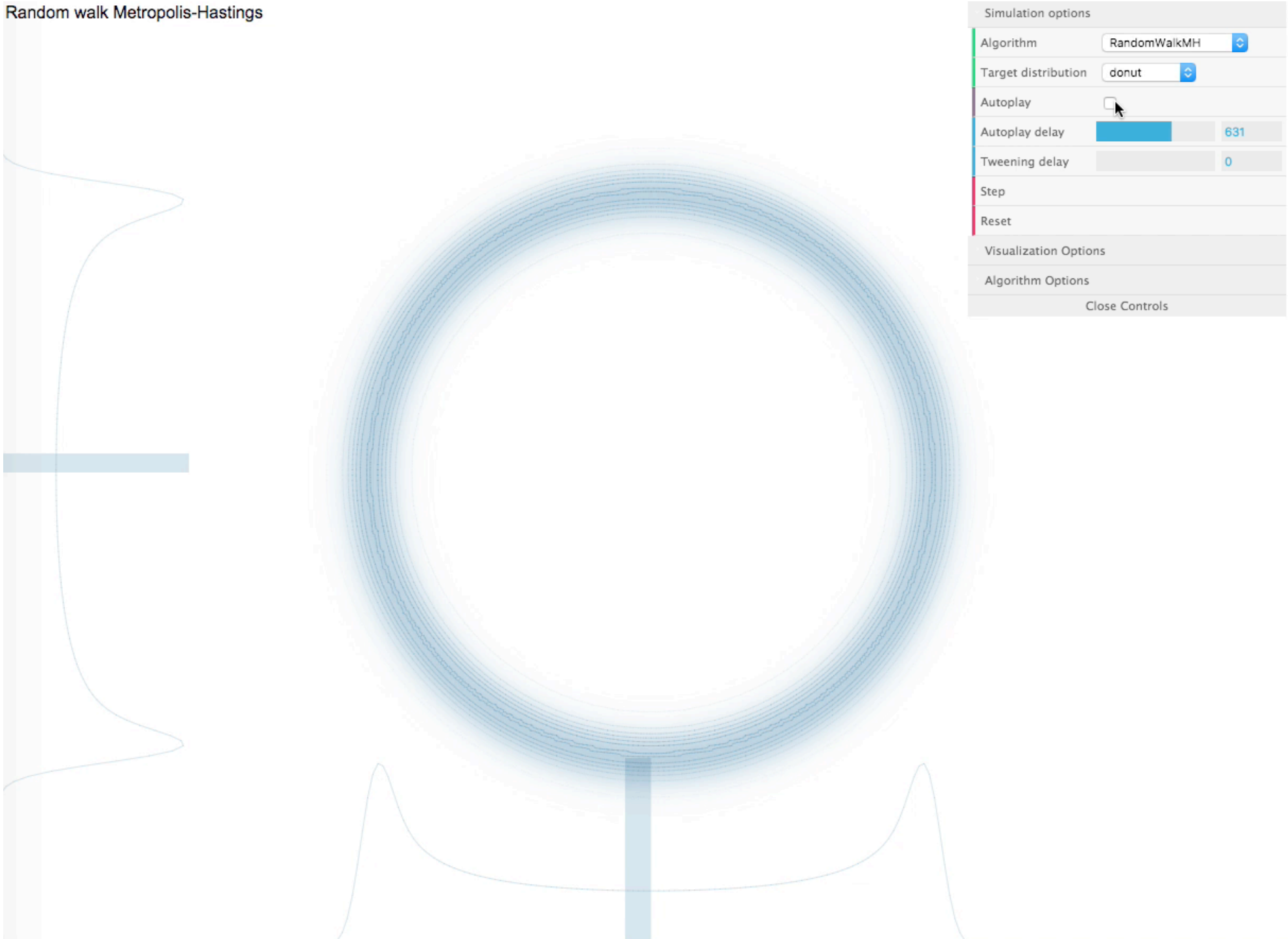
MCMC strategies

- Metropolis: Granddaddy of them all
- Metropolis-Hastings (MH): More general
- Gibbs sampling (GS): Efficient version of MH
- Hamiltonian Monte Carlo (HMC)
- Newer algorithms much better
- New methods being developed





https://chi-feng.github.io/mcmc-demo/



https://chi-feng.github.io/mcmc-demo/

Hamiltonian Monte Carlo

- Problem with Gibbs sampling (GS)
  - Models with many parameters usually have lots of highly correlated parameters
  - GS gets stuck, degenerates towards random walk
  - At best, inefficient because re-explores
- Hamiltonian dynamics to the rescue
  - represent parameter state as particle in n-dimensional space
  - flick it around frictionless log-posterior
  - record positions

William Rowan Hamilton (1805–1865) Commemorated on Irish Euro coin

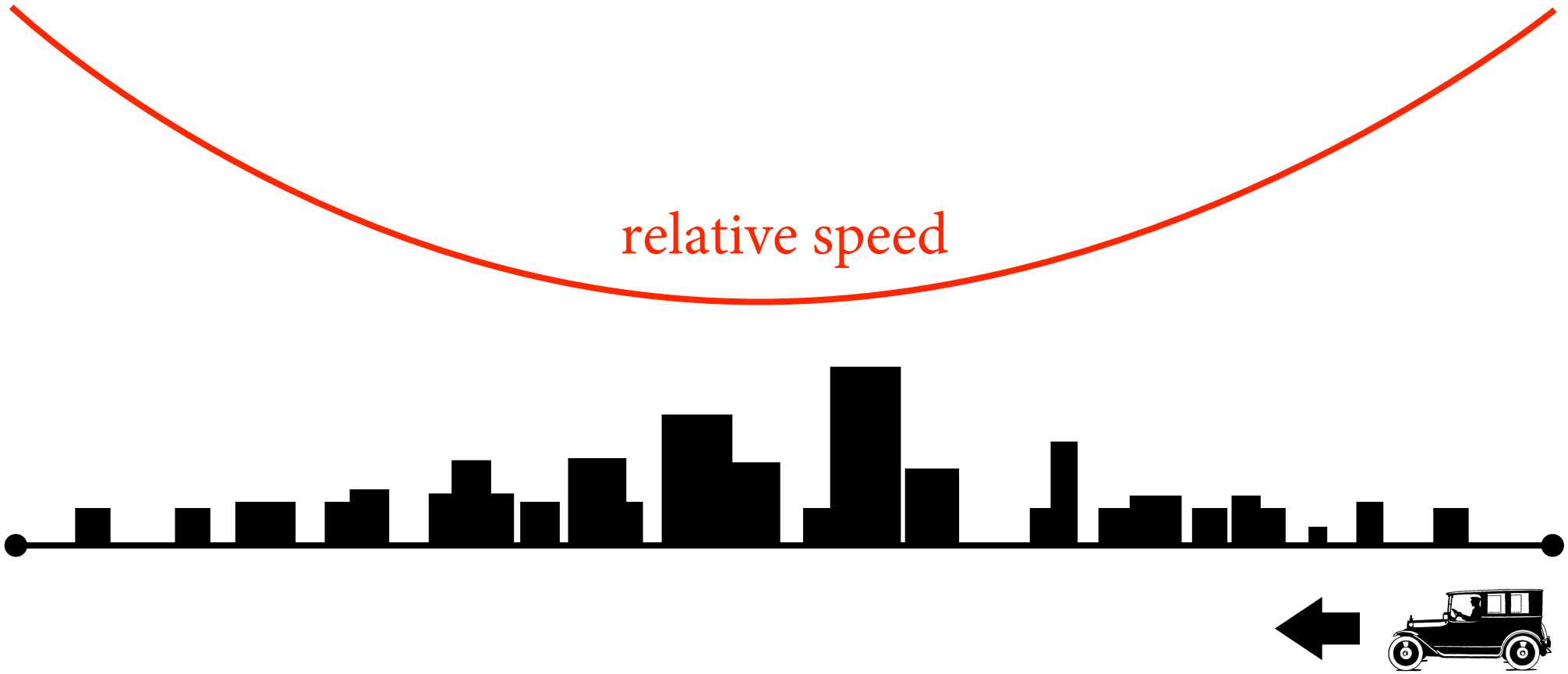
King Monty's Kingdom

King Monty's Kingdom

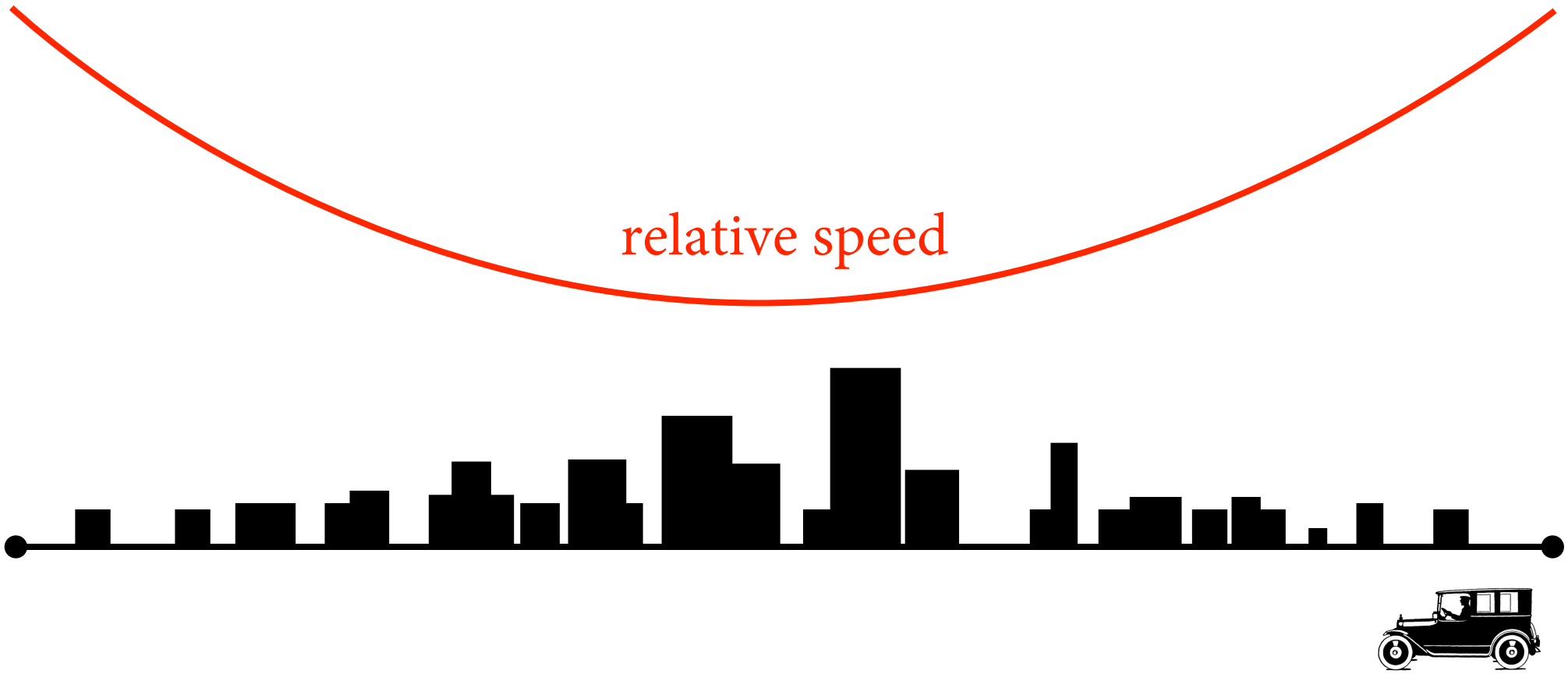
King Monty's Kingdom

King Monty's Kingdom

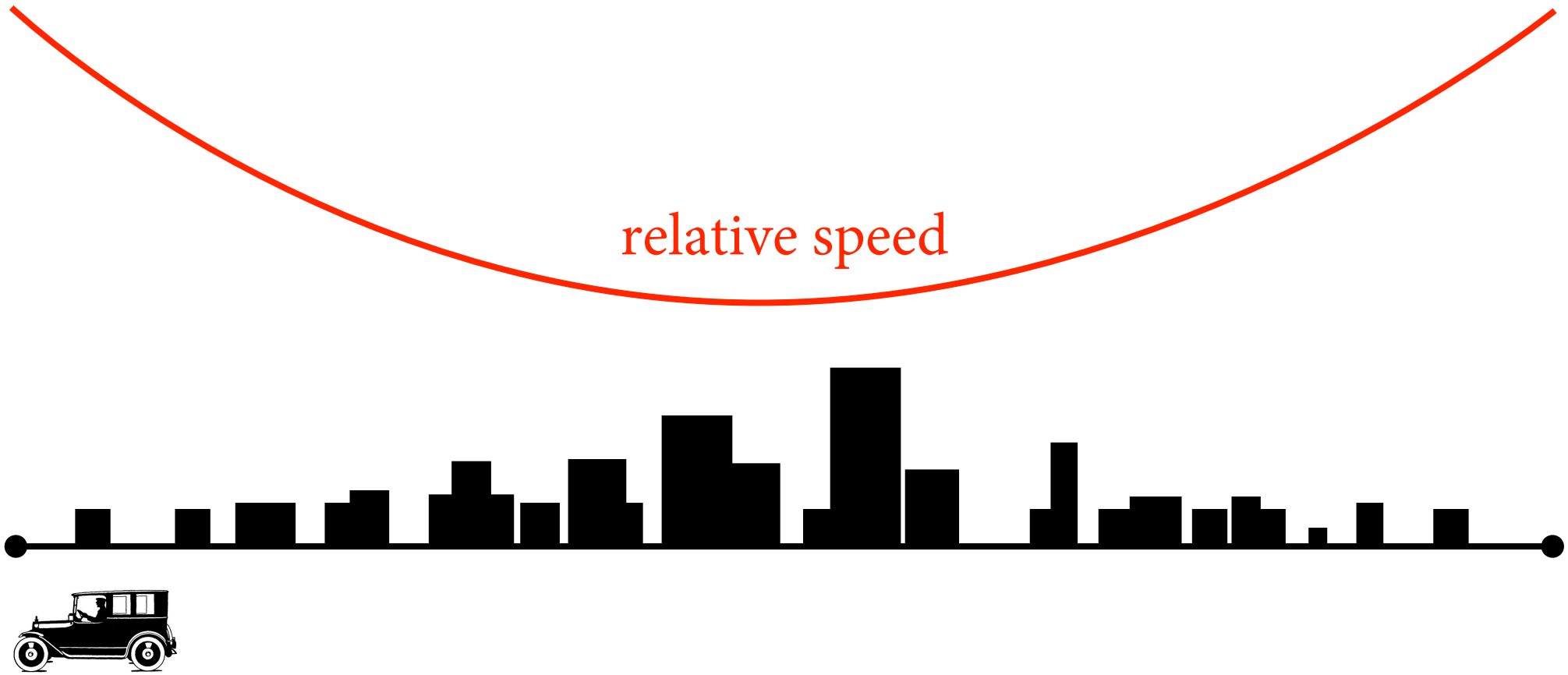
King Monty's Kingdom



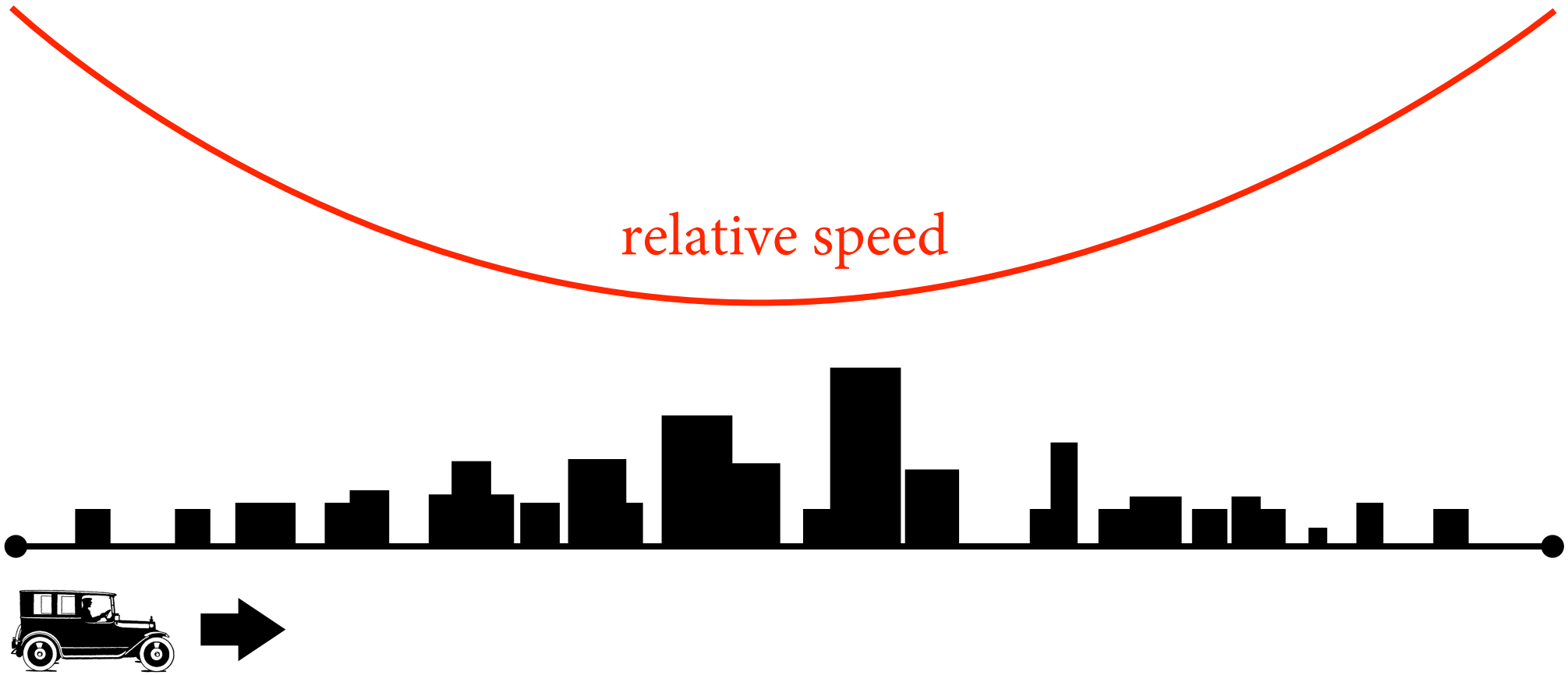
King Monty's Kingdom



King Monty's Kingdom

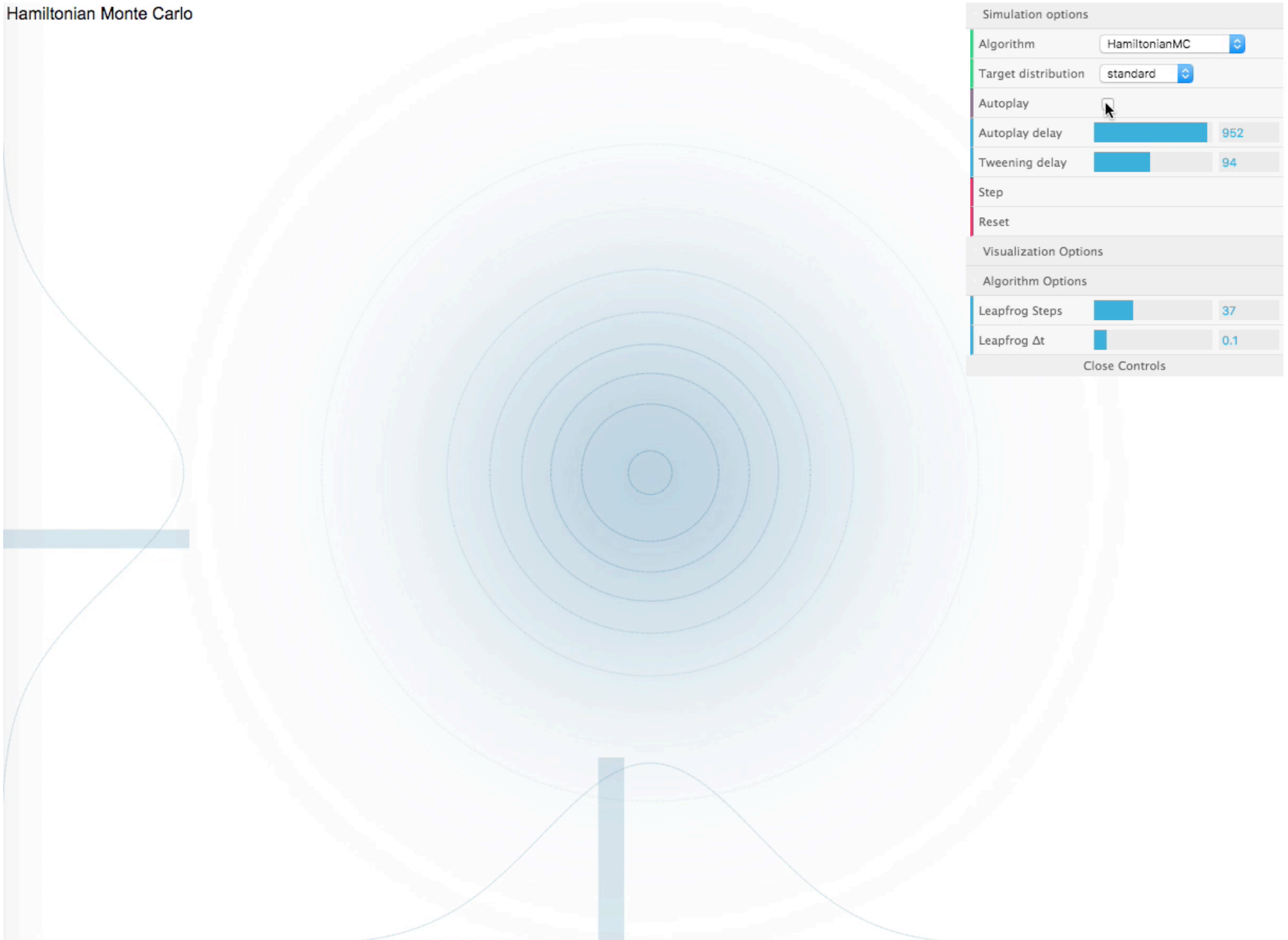


King Monty's Kingdom

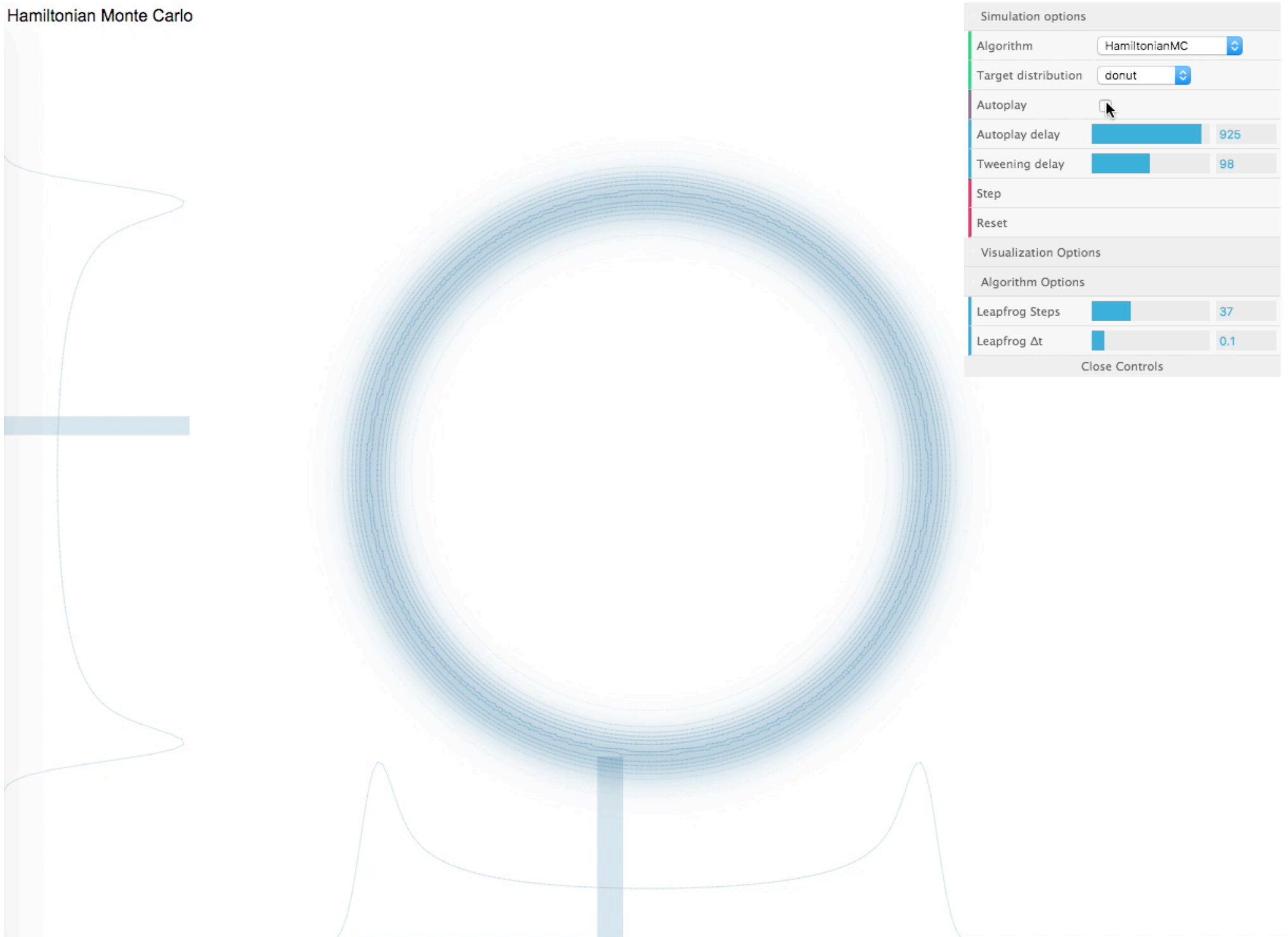


Hamiltonian Monte Carlo

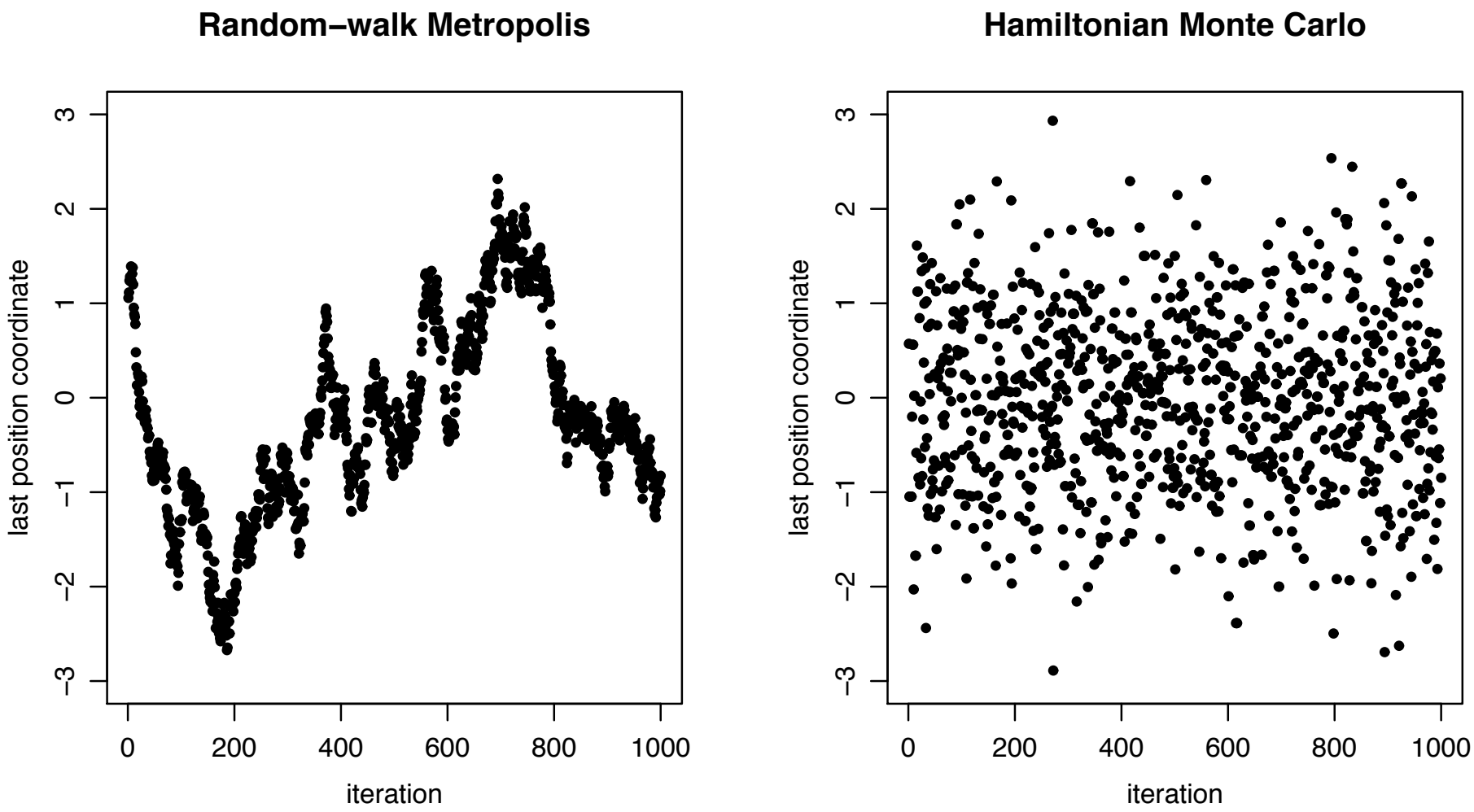
- Population density curve: log-posterior
- Position of car: parameter vector
- Speed of car: momentum of parameter values
  - Go fast when high
  - Go slow when low
  - Samples of position are samples from posterior



https://chi-feng.github.io/mcmc-demo/

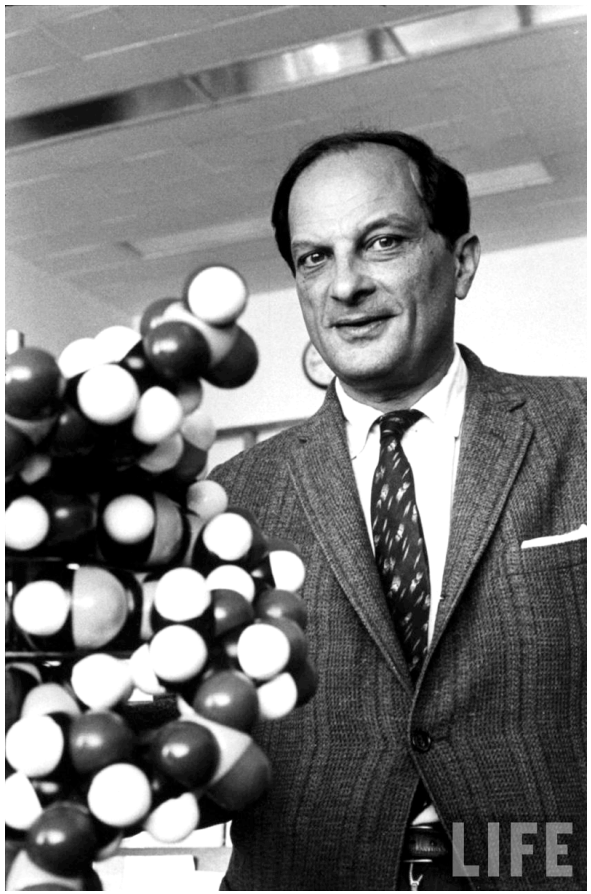


https://chi-feng.github.io/mcmc-demo/



Neal. 2011. MCMC from Hamiltonian dynamics

- mc-stan.org
- Install RStan
  1. Get C++ compiler
  2. ???
  3. Profit



Stanislaw Ulam (1909–1984)

Interfaces

ways to run Stan

Stan Interfaces

The Stan modeling language and statistical algorithms are exposed through interfaces into many popular computing environments.

- RStan (R)
- PyStan (Python)
- CmdStan (shell, command-line terminal)
- MatlabStan (MATLAB)
- Stan.jl (Julia)
- StataStan (Stata)
- MathematicaStan (Mathematica)

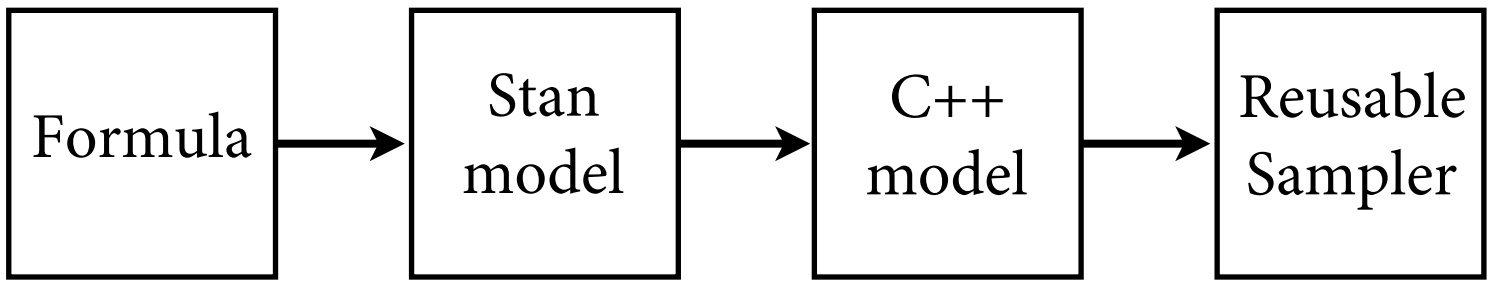
Programs written in the Stan modeling language are portable across interfaces.

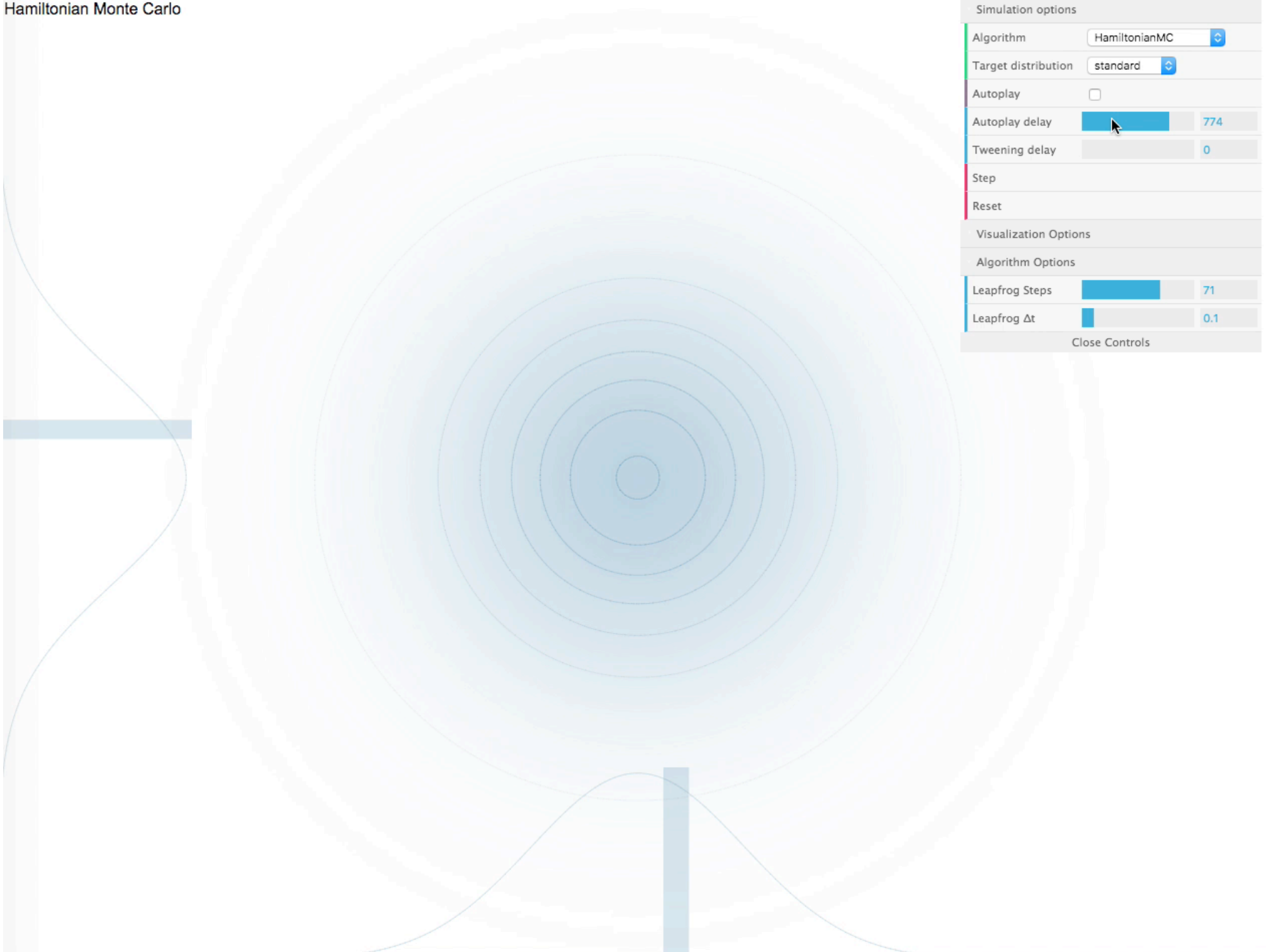
Stan is NUTS

No U-Turn Sampler

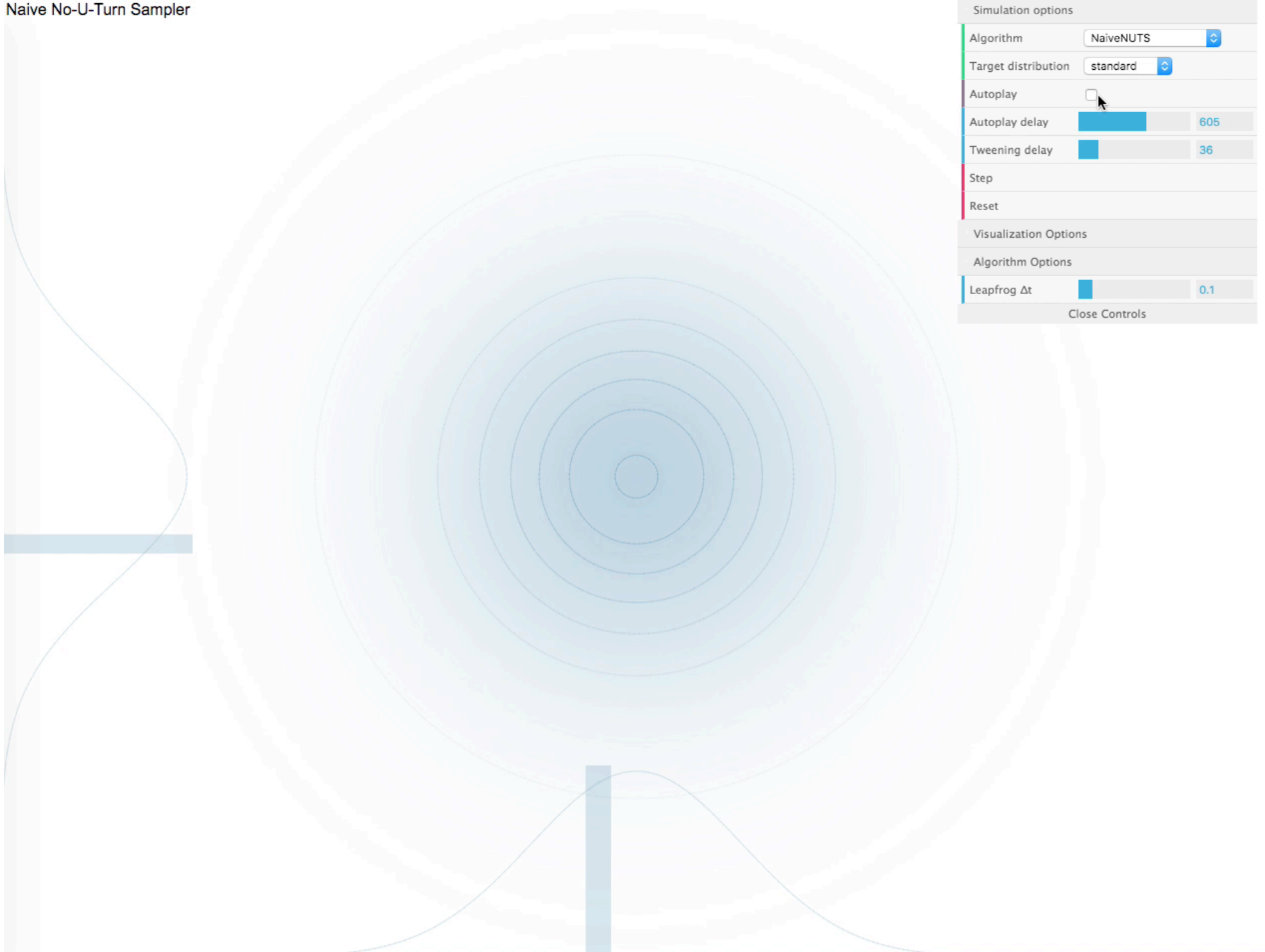
Automatic Step Size and Number Adaptation

- No U-Turn Sampler (NUTS2): Adaptive Hamiltonian Monte Carlo
- Implemented in Stan (rstan: mc-stan.org)
- We'll use `map2stan` for now

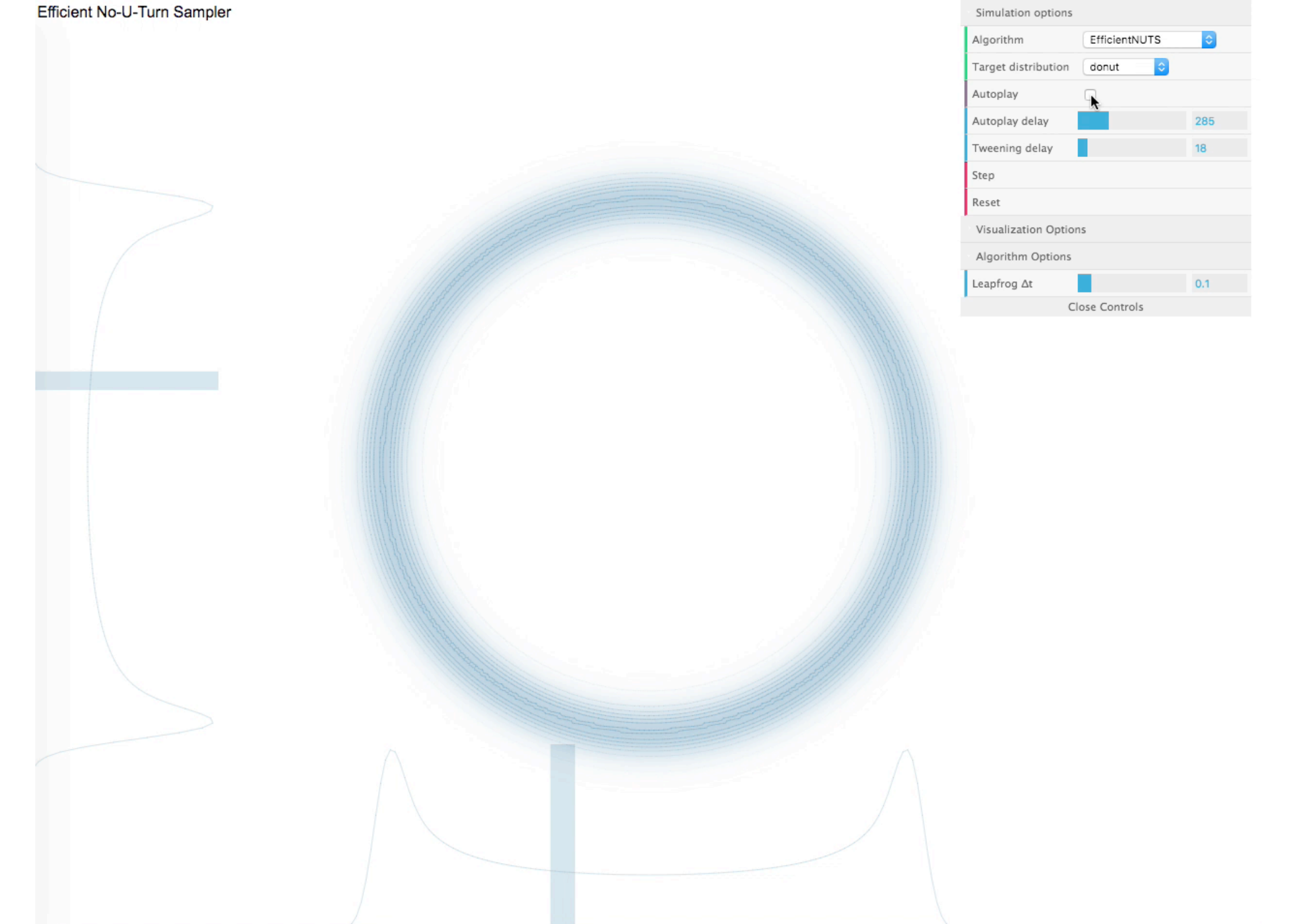




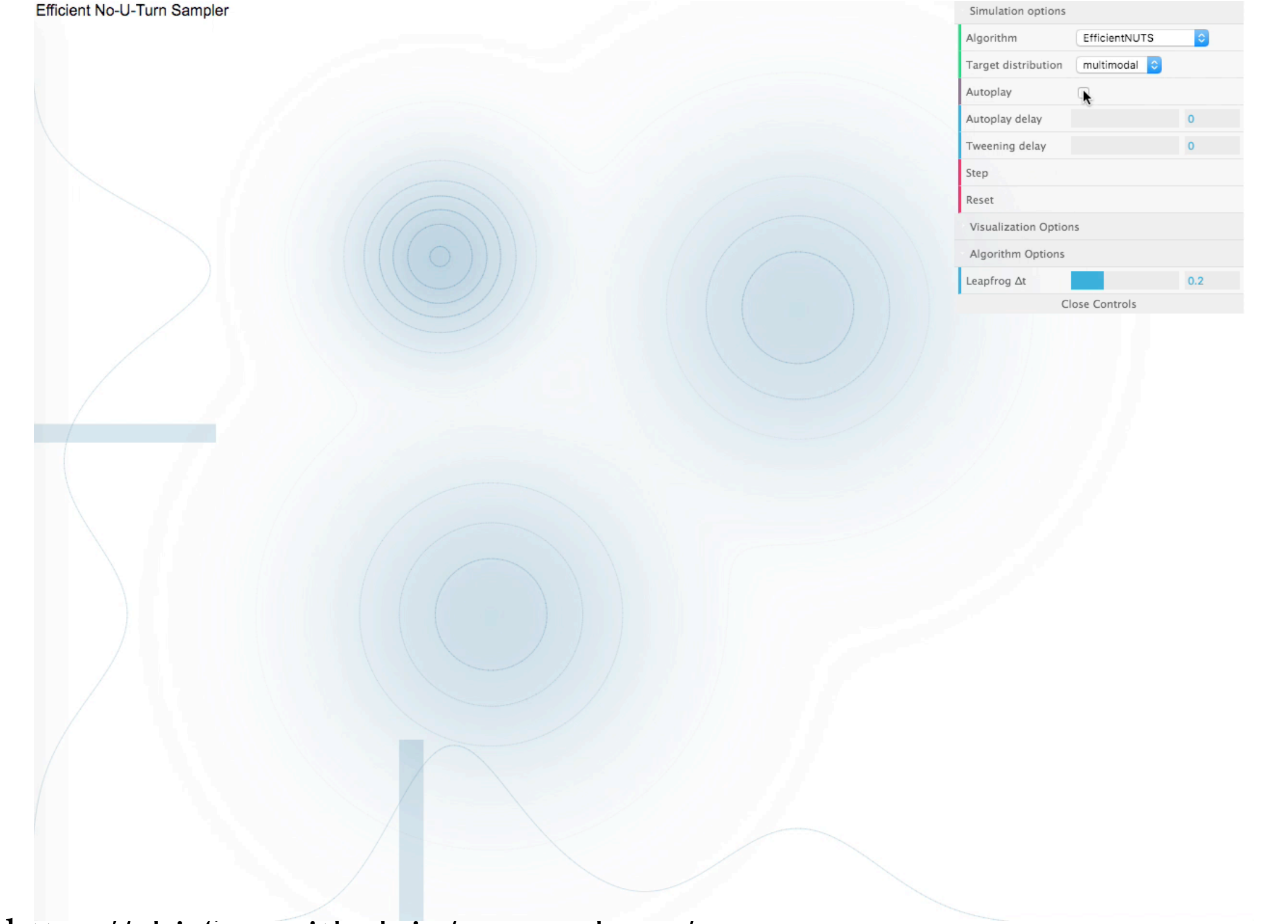
https://chi-feng.github.io/mcmc-demo/



https://chi-feng.github.io/mcmc-demo/



https://chi-feng.github.io/mcmc-demo/



https://chi-feng.github.io/mcmc-demo/

HMC example

- Back to terrain ruggedness, data(rugged)
- Re-estimate using Stan
- See how it works
- See what to expect from healthy Markov chains

MAP estimates

R code 8.3

```
m8.1 <- map(
    alist(
        log_gdp ~ dnorm( mu , sigma ) ,
        mu <- a + bR*rugged + bA*cont_africa + bAR*rugged*cont_africa ,
        a ~ dnorm(0,100),
        bR ~ dnorm(0,10),
        bA ~ dnorm(0,10),
        bAR ~ dnorm(0,10),
        sigma ~ dunif(0,10)
    ) ,
    data=dd )
precis(m8.1)
```

```
       Mean StdDev  5.5% 94.5%
a      9.22   0.14  9.00  9.44
bR    -0.20   0.08 -0.32 -0.08
bA    -1.95   0.22 -2.31 -1.59
bAR    0.39   0.13  0.19  0.60
sigma  0.93   0.05  0.85  1.01
```

HMC estimates

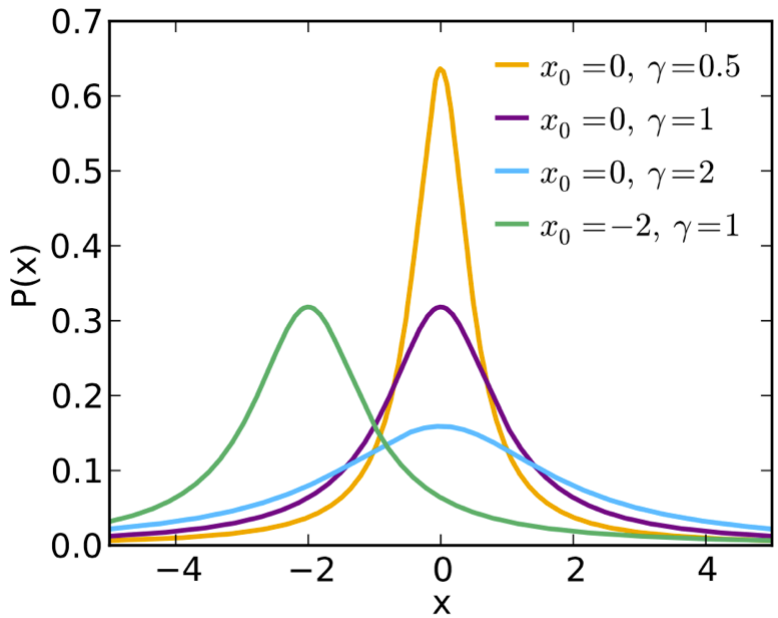
R code 8.5

```
m8.1stan <- map2stan(
    alist(
        log_gdp ~ dnorm( mu , sigma ) ,
        mu <- a + bR*rugged + bA*cont_africa + bAR*rugged*cont_africa ,
        a ~ dnorm(0,100),
        bR ~ dnorm(0,10),
        bA ~ dnorm(0,10),
        bAR ~ dnorm(0,10),
        sigma ~ dcauchy(0,2)
    ) ,
    data=dd.trim )
```

What is “Cauchy”?

- One of many things named after Augustin-Louis Cauchy (KO-shee)
  - Ratio of two Gaussian samples
  - Useful distribution with thick tails
  - Parameters: location of mode, scale
  - Mean and variance undefined
  - Related to Lévy flights

Baron Augustin-Louis Cauchy (1789–1857)



Extract samples

- Once you have samples, they are just samples

R code 8.8

```
post <- extract.samples( m8.1stan )
str(post)
```

```
List of 5
 $ a    : num [1:1000(1d)] 9.3 9.34 9.21 9.43 9.28 ...
 $ br   : num [1:1000(1d)] -0.133 -0.214 -0.215 -0.229 -0.209 ...
 $ bA   : num [1:1000(1d)] -1.91 -2.25 -2.26 -1.98 -1.99 ...
 $ brA  : num [1:1000(1d)] 0.133 0.367 0.533 0.254 0.468 ...
 $ sigma: num [1:1000(1d)] 0.988 0.949 0.904 0.976 0.934 ...
```

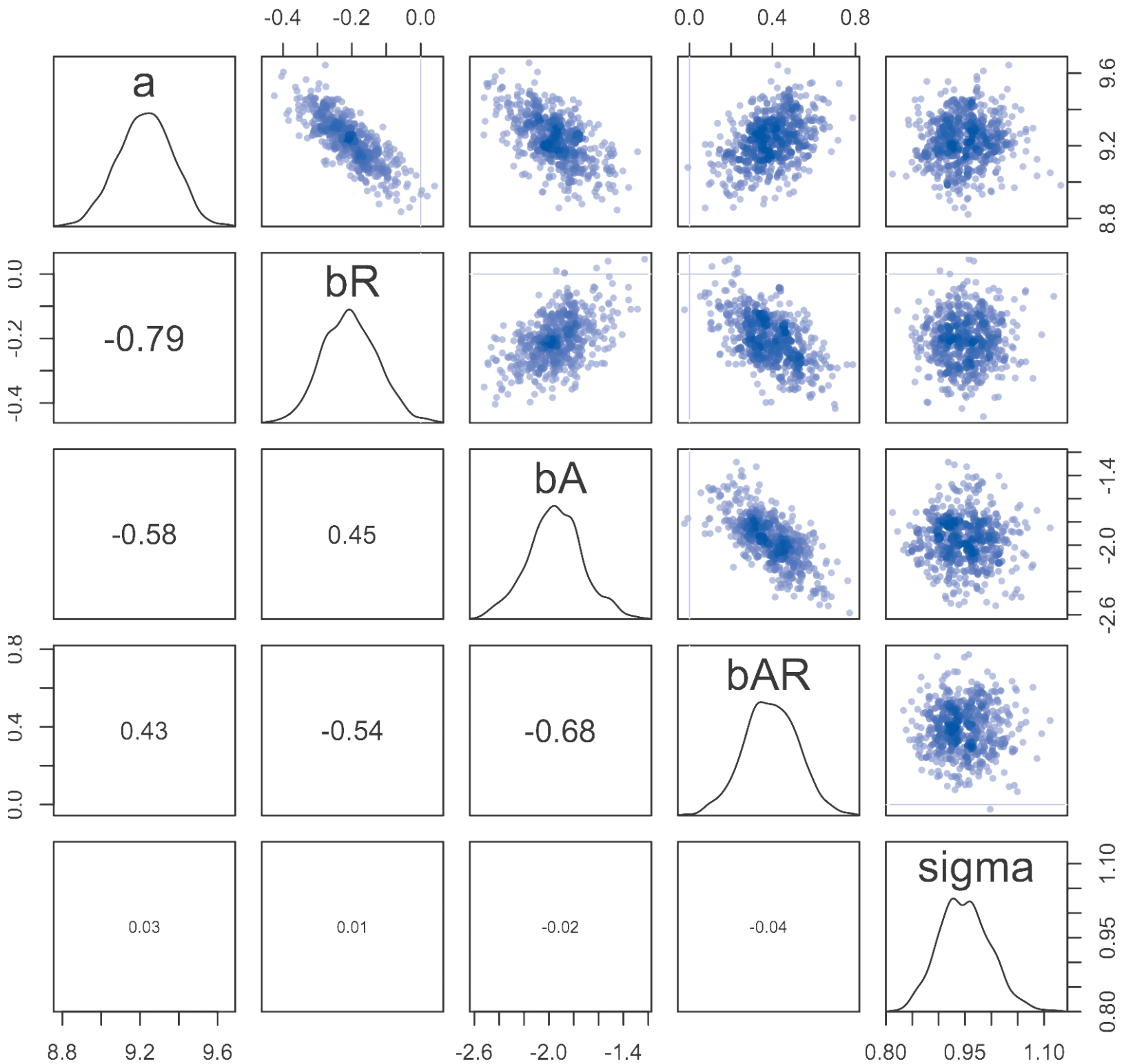

a
bR
bA
bAR
sigma
-0.79
-0.58
0.45
0.43
-0.54
-0.68
0.03
0.01
-0.02
-0.04
-0.4
-0.2
0.0
0.4
0.8
8.8
9.2
9.6
-2.6
-2.0
-1.4
0.80
0.95
1.10

Figure 8.3

Check the chain

- Sometimes it doesn't work
- First and most important check: trace plot

```
plot(m8.1stan)
```