Hochschule Konstanz
Fakultät Informatik

Markov Chain Monte Carlo (MCMC)

The missing chapter ;-)

Brown Bag Seminar

Background: Practical Bayesian Statistics

- Bayes
  - $p(\theta|D) = \frac{p(D|\theta)p(\theta)}{\int p(D|\theta)p(\theta)d\theta}$
- Easy: unnormalized posterior (likelihood times prior)
  - $p(D|\theta)p(\theta)$

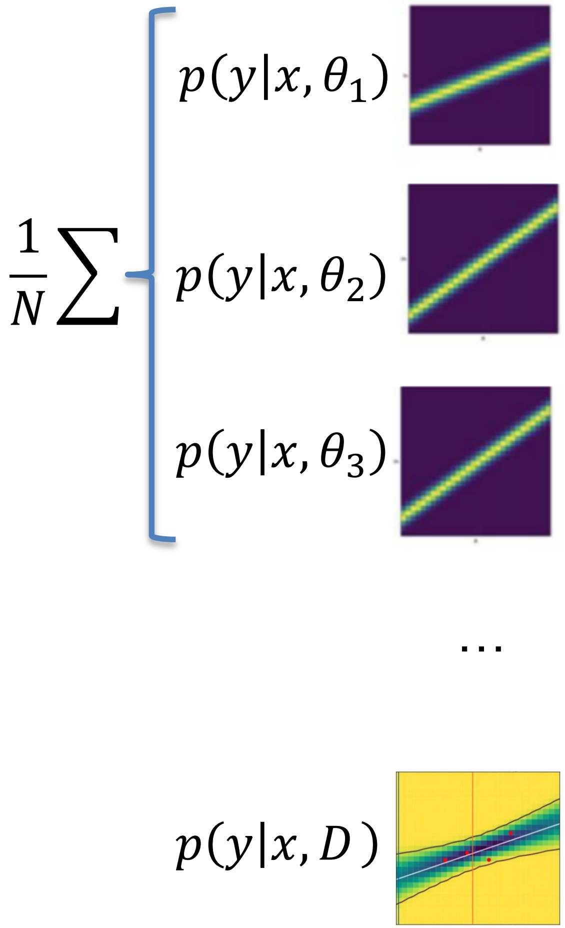
- Methods to access $p(\theta|D)$

- Analytical (often impossible)
- Grid Approximation (very intensive)
- Quadratic Approximation (only valid for small dimensions)
- Variational Inference
- Markov Chain Monte Carlo

Background: Monte-Carlo Integration in Bayes (CPD)
See Daniel's Talk

Bayesian Model averaging BMA:

- $p(y|x, D) = \int p(y|x, \theta) \cdot p(\theta|D) d\theta$
- $p(y|x, D) = E_{\theta \sim p(\theta|D)}[p(y|x, \theta)]$
- $p(y|x, D) \approx \frac{1}{N} \sum_{\theta_i \sim p(\theta|D)} p(y|x, \theta_i)$



Samples is all you need

- Samples from $\theta_i \sim p(\theta|D)$

Have (Unormalized Posterior)

- $p(\theta|D) \propto p(D|\theta) p(\theta)$
  - $p(D|\theta)$ Likelihood how prob. is D given θ
  - $p(\theta)$ Prior
  - Easy to specify in a few lines of code

Monte-Carlo Integration General

$$E_\theta[f(\theta)] = \int f(\theta)p(\theta)\,d\theta \approx \frac{1}{N}\sum_{\theta \sim p(\theta)} f(\theta)$$

Need:

- Samples from $\theta \sim p(\theta)$

Have:

- Probability density $p(\theta)$ (up to constant)

MCMC:

- Is a method to draw samples θ from the (unnormalized) probability density $p(\theta)$

Markov Chain Monte Carlo

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

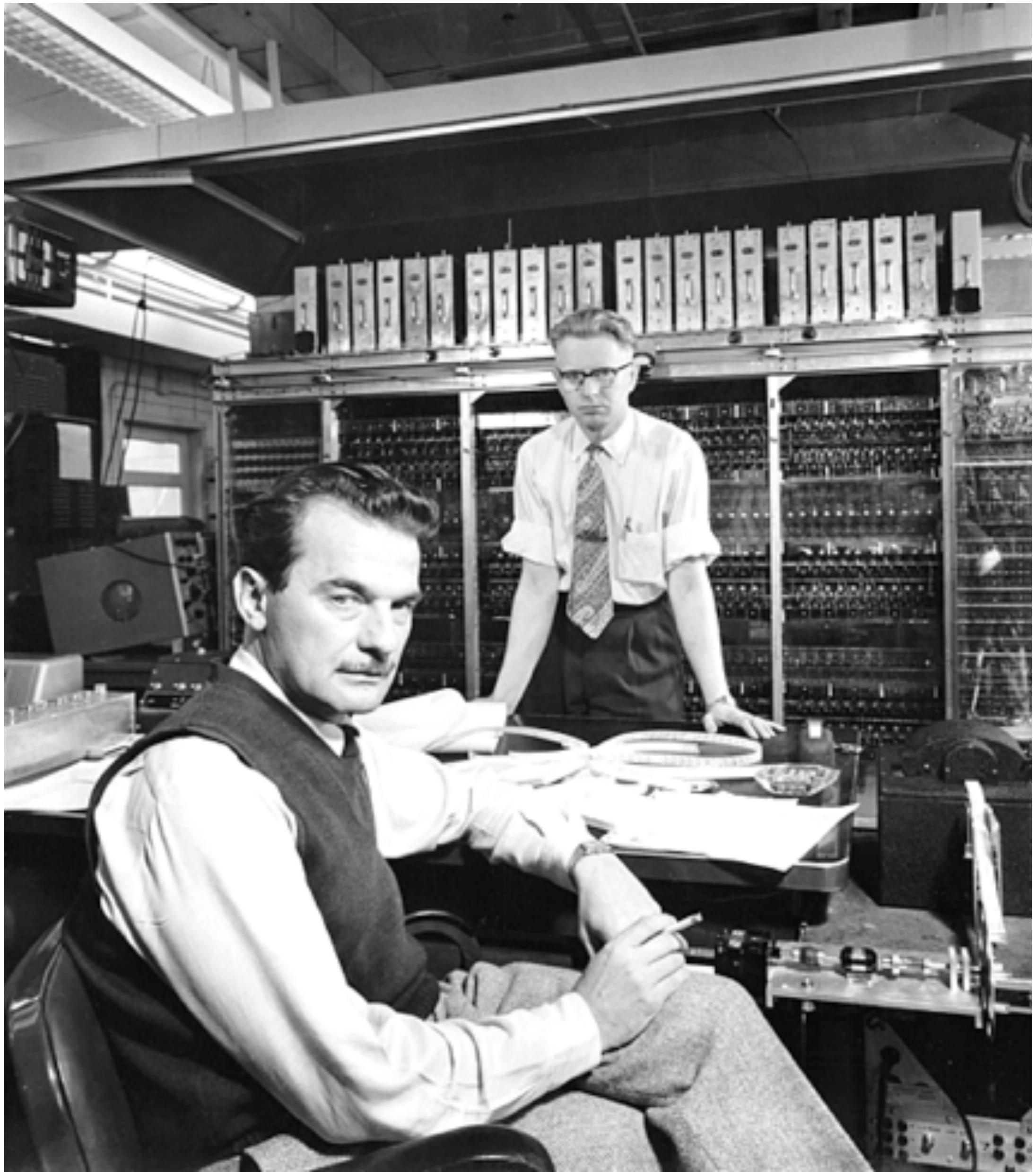
AND

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)

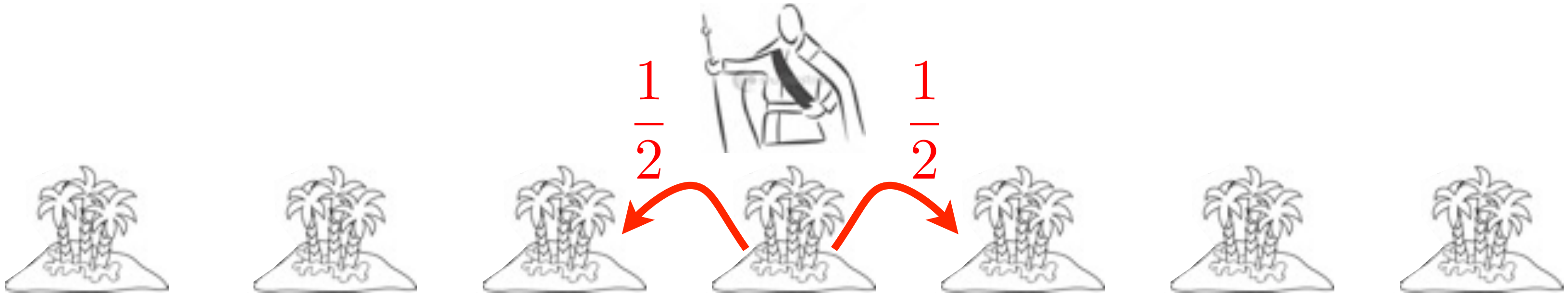
King Markov


Texts in Statistical Science
Statistical Rethinking
A Bayesian Course
with Examples in R and Stan
SECOND EDITION
Richard McElreath
CRC Press

The Metropolis Archipelago

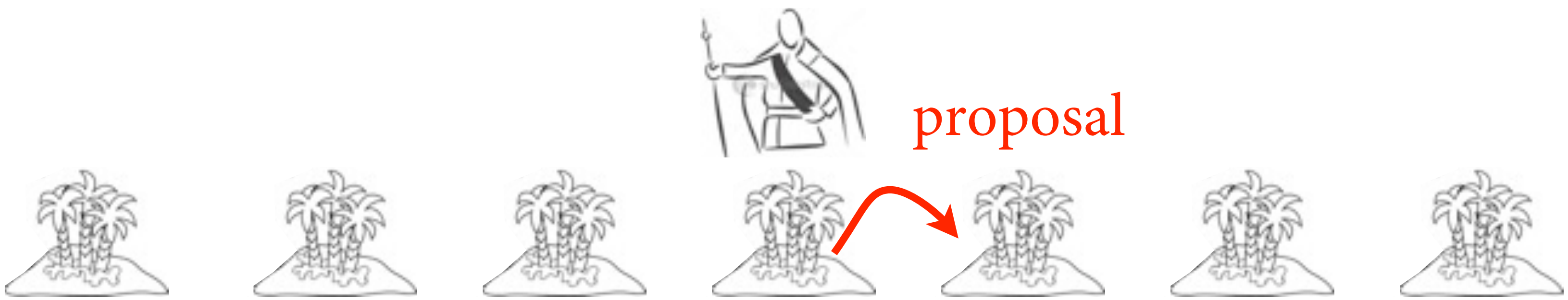
Contract: King Markov must visit each island in proportion to its population size.

Here's how he does it...



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

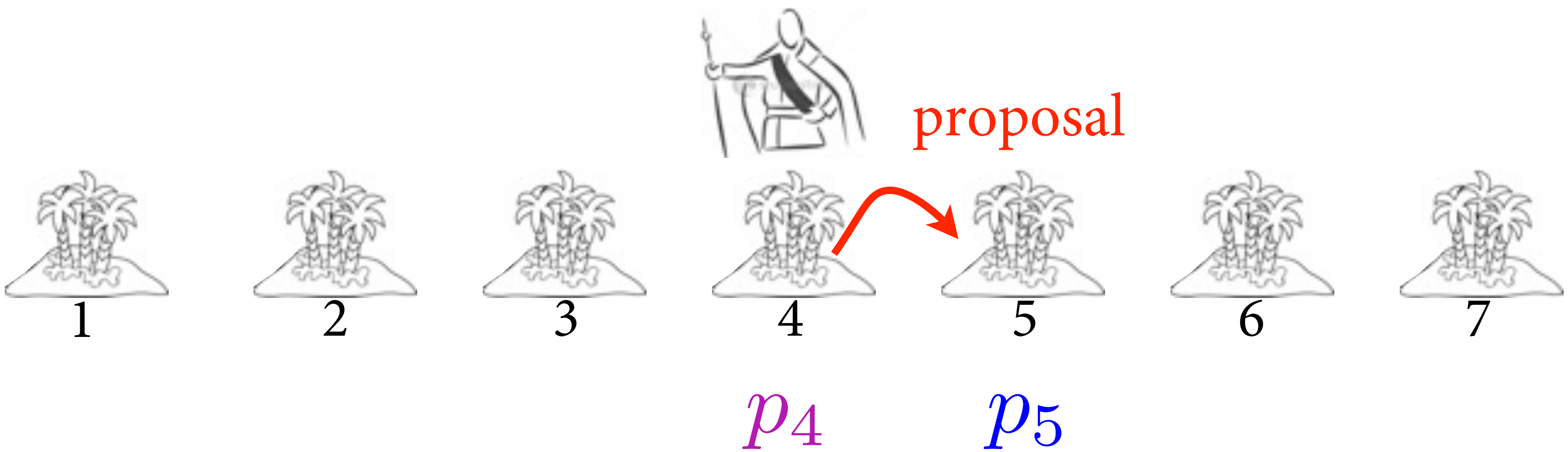
(1) Flip a coin to choose island on left or right.
Call it the "proposal" island.



(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.

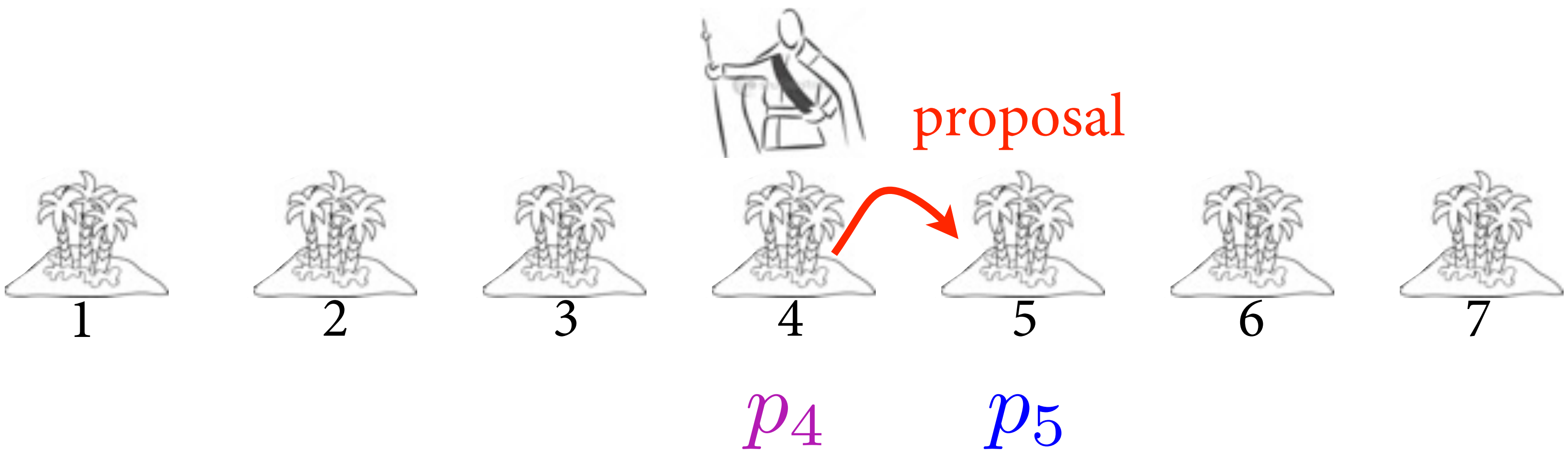


(3) Find population of current island.

(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.



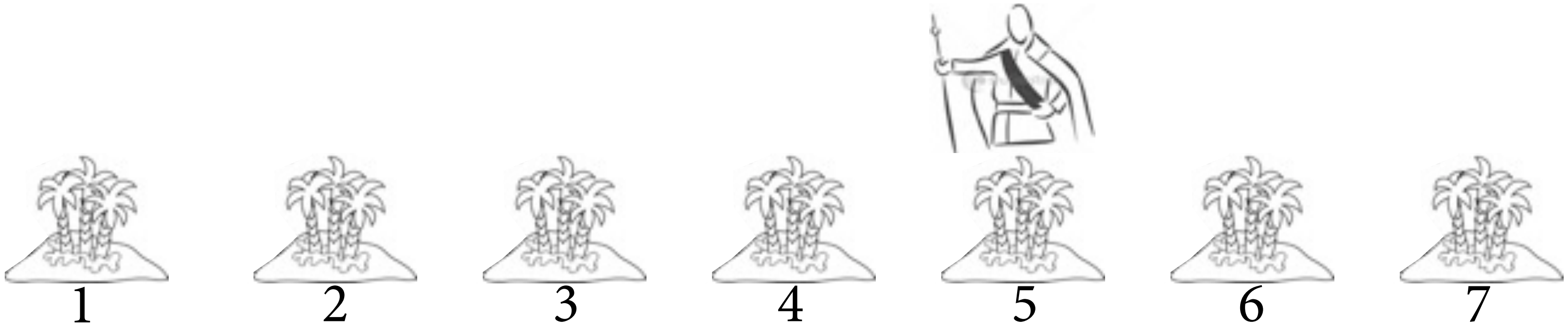
(4) Move to proposal, with probability = $\frac{p_5}{p_4}$

(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)

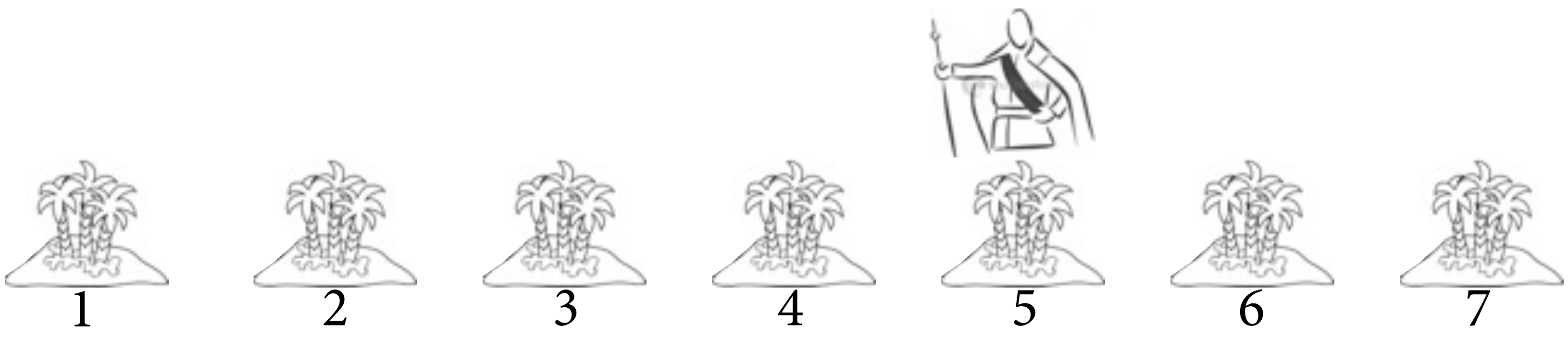
(1) Flip a coin to choose island on left or right.
Call it the "proposal" island.

(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)

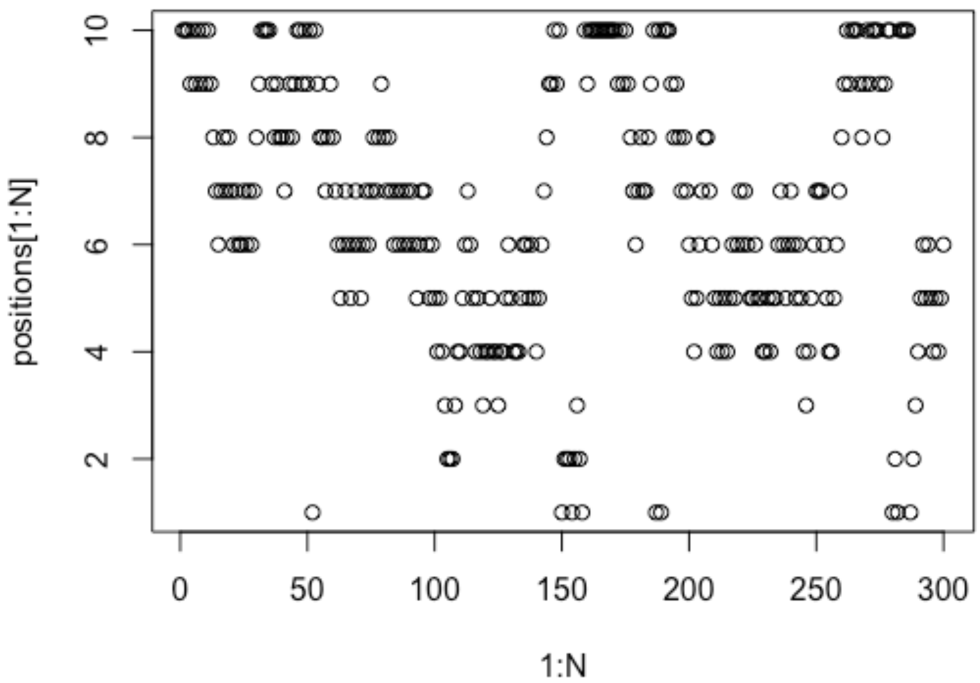


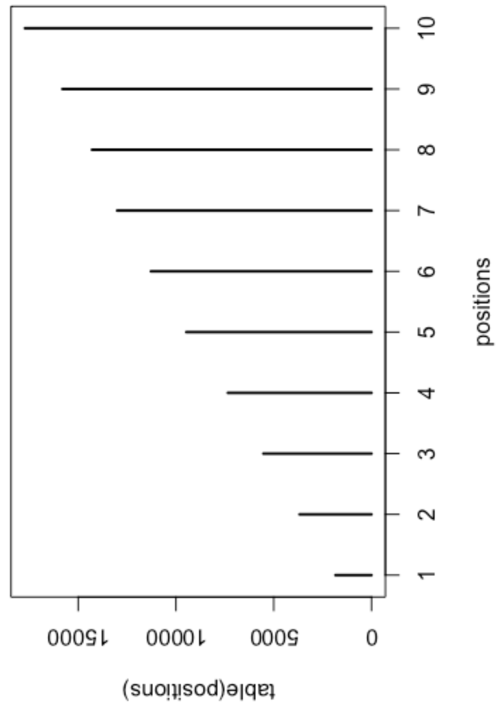
This procedure ensures visiting each island in proportion to its population, *in the long run.*

Metropolis algorithm

```
num_weeks <- 1e5
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 )
}
```

Hello World to a Markov-Chain





R code 9.2
N = 300
plot(1:N , positions[1:N])

R code 9.3
plot(table(positions))

Homework

* Change population of islands to abitrary distribution.

* Multiply the probabilities with a constant factor

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

Detailed Balance Condition (Why does MCMC Work)

- Split p_{ji} into $p_{ji} = T_{ji} \cdot p_{ji}^{a}$
  - T_{ji} probability for a (tried) move
  - p_{ji}^{a} of accepting the move

- Stuff (# Kings) which goes from i to j is
  - Consider N kings moving in one time step
  - Probability p_i to be in i in the first place
    - $N_i = N \cdot p_i$
  - Number of Kings moving from i to j
    - $N_{ji} = N_i \, T_{ji} \, p_{ji}^{a} = N \, p_i \, T_{ji} \, p_{ji}^{a}$
    - Probability T_{ji} of a jump from i to j
    - Probability p_{ji}^{a} that the jump is accepted

- Detailed Balance Condition (Einfluß=Ausfluß)
  - $N_{ji} = N_{ij}$
  - $p_i \, T_{ji} \, p_{ji}^{a} = p_j \, T_{ij} \, p_{ij}^{a}$
  - Valid if one king moves (ergodicity) over time

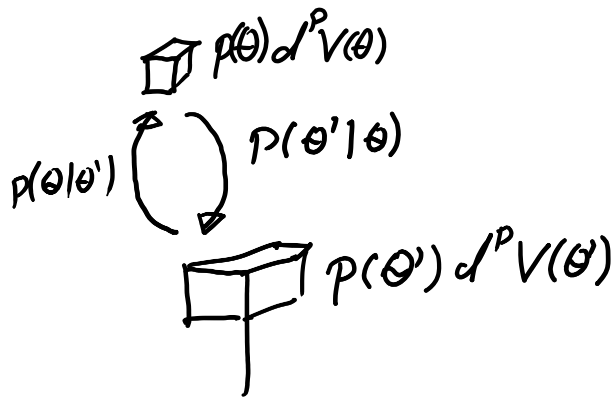
Metropolis Hasting acceptance criterium

- Detailed Balance reads
  - $p_i\, T_{ji}\, p_{ji}^a = p_j\, T_{ij}\, p_{ij}^a$
- We can choose T_{ji} and p_{ji}^a so that p_i and p_j matches the desired distribution
- Acceptance criterium
  - $p_{ji}^a = \min\left(1, \frac{p_j\, T_{ij}}{p_i\, T_{ji}}\right)$
  - $p_{ji}^a = \min\left(1, \frac{p_j}{p_i}\right)$ if $T_{ij} = T_{ji}$
  - Note just the ratio p_i/p_j enters

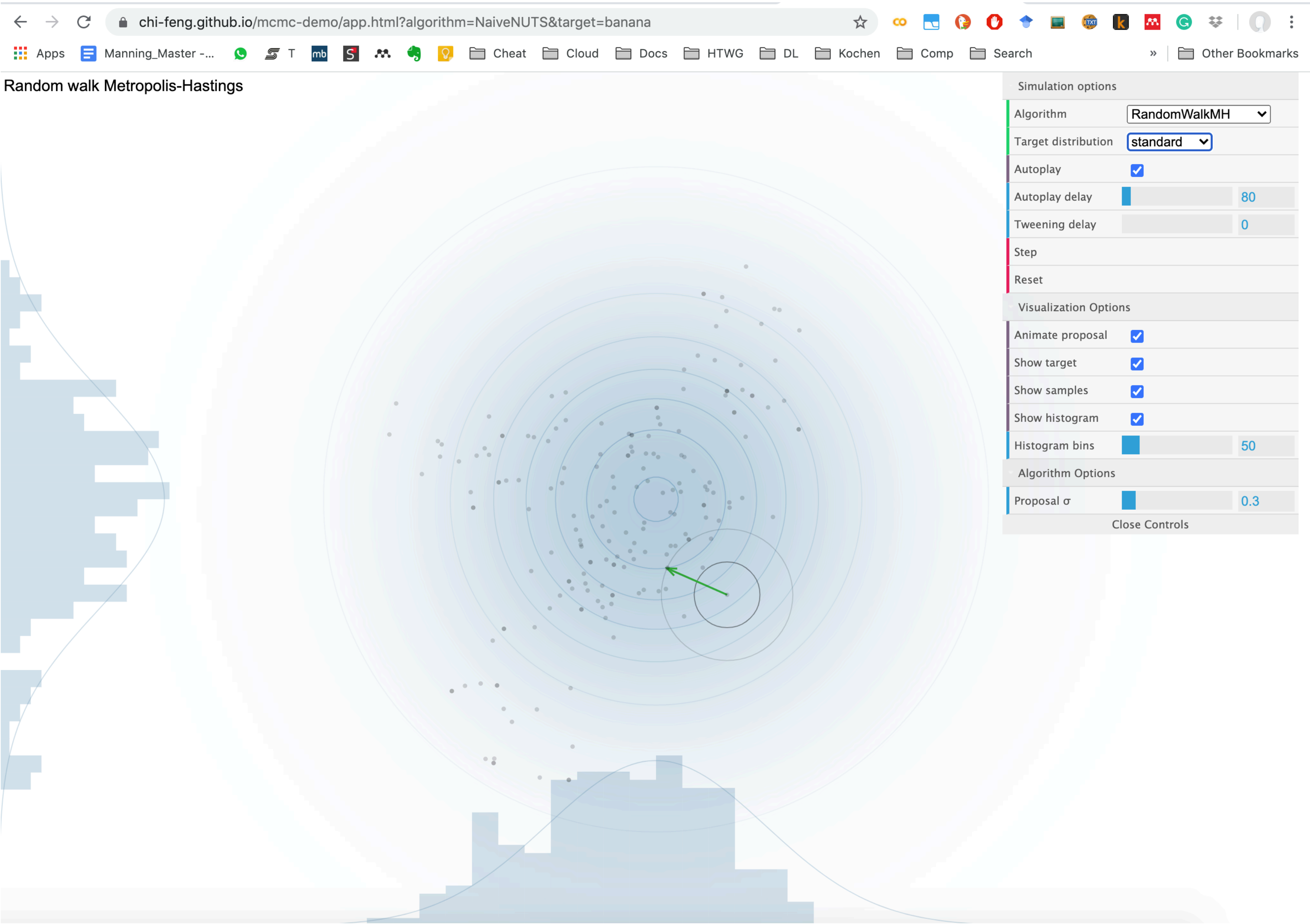
  - Proof
    - Let $p_i\, T_{ji}$ > $p_j\, T_{ij}$ w.l.o.g. (can be repeated other way around) both terms are not zero
    - $p_{ji}^a = \left(\frac{p_j\, T_{ij}}{p_i\, T_{ji}}\right)$ *and* $p_{ij}^a = 1$
    - *Insert into (*)*
    - $p_i\, T_{ji}\, \frac{p_j\, T_{ij}}{p_i\, T_{ji}} = p_j\, T_{ij}$ *fits*

MCMC for continuous

- See blackboard (E.g. sampling from Gaussian)



Metropolis Hastings at Work

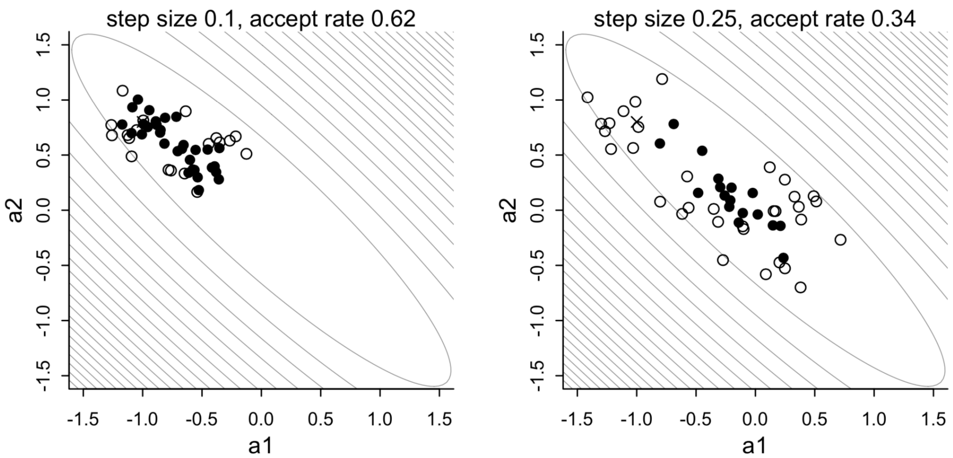


https://chi-feng.github.io/mcmc-demo https://chi-feng.github.io/mcmc-demo/app.html?algorithm=RandomWalkMH&target=standard

Desired Properties of MCMC

- Desired:
  - High Acceptance Rate
  - Fast Exploration of Probability Landscape

Metropolis gets stuck



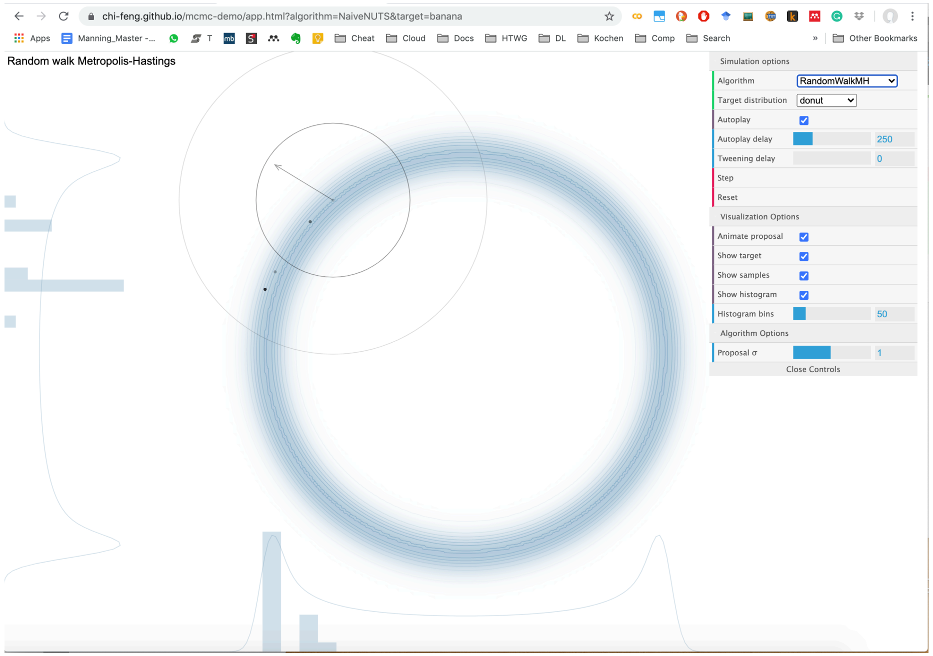
small steps, slow walk

bigs steps, low accept rate

- Problem get worse in high dimensions

MCMC Algorithms has problems for complex distributions

- Metropolis Hastings at work
  - https://chi-feng.github.io/mcmc-demo/



HMC

One component of the HMC

https://elevanth.org/blog/2017/11/28/build-a-better-markov-chain/

Hybrid / Hamilton Monte Carlo (Algorithm, idea)

$\mathrm{q} = \theta$ Position (e.g. in half-pipe)

$\mathrm{r} = m \cdot v$ Momentum (additional variable)

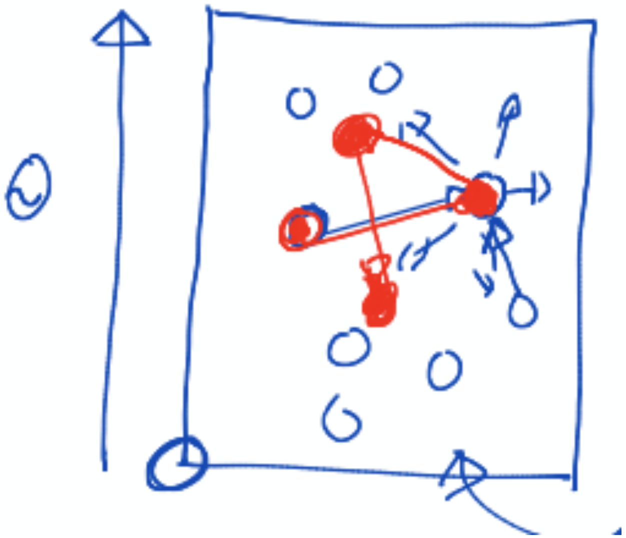
- Start at random position q_i
- Give the skater a random kick (momentum)
- Let her skate for some time t
- stop her at position q_{i+1}

q_i are samples from probability $\mathrm{p}(\theta)$

if potential energy (height of the ramp) is $U(\theta) = -\ln(p(\theta))$

WTF Why should this be samples from p(θ)?

A real world HMC simulation currently going on



An air particle (red) flies around in the air, like the skater it obeys the laws of classical physics.

After some time, get hit by another air molecule. It gets random momentum.

The momentum/velocity is independent of height (assume temperature is constant)

That's the HMC algorithm!

What does the samples look like, should be distributed as ($U(\theta) = m \cdot g \cdot h$)

$$U(\theta) \propto -\ln\big(p(\theta)\big) \Rightarrow p(\theta) \propto e^{-U(\theta)} = e^{-m \cdot g \cdot \theta}$$

That's correct: The number of partials or the pressure is distributed like:

$$\text{pressure}(\theta) = \text{pressure}(0)\ e^{-\frac{1}{k_b T}\ m \cdot g \cdot \theta}$$

Hamilton Monte Carlo (Algorithm, idea)

$\mathrm{q} = \theta$ Position (e.g. in half-pipe)

$\mathrm{r} = m \cdot v$ Momentum (additional variable)

- Start at random position q_i
- Give the skater a random impulse
- Let her skate for some time t (Equation of motion)
- stop her at position q_{i+1}

q_{i+1} samples from $p(\theta)$ if potential energy is $U(\theta) = -\ln(p(\theta))$

Equation of motion (during the scatting phase)

- $\frac{dq}{dt} = v = \frac{r}{m}$
- $\frac{dr}{dt} = -\frac{dU(\theta)}{d\theta} = \nabla U(\theta)$ Newton's 3rd law (Force = mass * acceleration)

Numerics bites

Algorithm 1: Hamiltonian Monte Carlo

Input: Starting position $\theta^{(1)}$ and step size ϵ
for $t = 1, 2 \cdots$ **do**
  Resample momentum r
  $r^{(t)} \sim \mathcal{N}(0, M)$
  $(\theta_0, r_0) = (\theta^{(t)}, r^{(t)})$
  Simulate discretization of Hamiltonian dynamics in Eq. (4):
  $r_0 \leftarrow r_0 - \frac{\epsilon}{2} \nabla U(\theta_0)$
  for $i = 1$ ***to*** m **do**
    $\theta_i \leftarrow \theta_{i-1} + \epsilon M^{-1} r_{i-1}$
    $r_i \leftarrow r_{i-1} - \epsilon \nabla U(\theta_i)$
  end
  $r_m \leftarrow r_m - \frac{\epsilon}{2} \nabla U(\theta_m)$
  $(\hat{\theta}, \hat{r}) = (\theta_m, r_m)$
  Metropolis-Hastings correction:
  $u \sim \mathrm{Uniform}[0, 1]$
  $\rho = e^{H(\hat{\theta}, \hat{r}) - H(\theta^{(t)}, r^{(t)})}$
  if $u < \min(1, \rho)$, **then** $\theta^{(t+1)} = \hat{\theta}$
end

Integration / discretization can cause numerical problems.

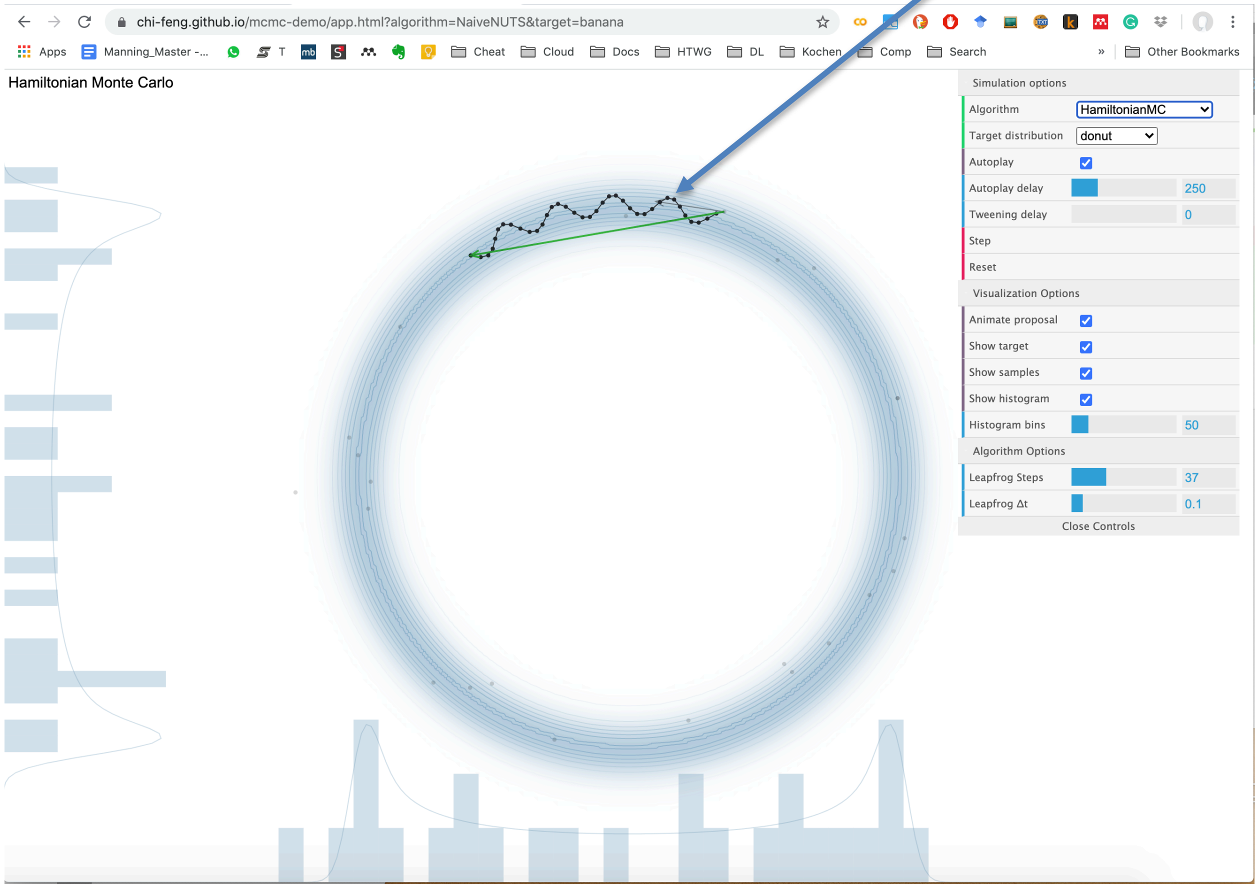
Fix Detailed Balance
(Still high acceptance rate)

Radford Neal https://arxiv.org/abs/1701.02434

HMC at work

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

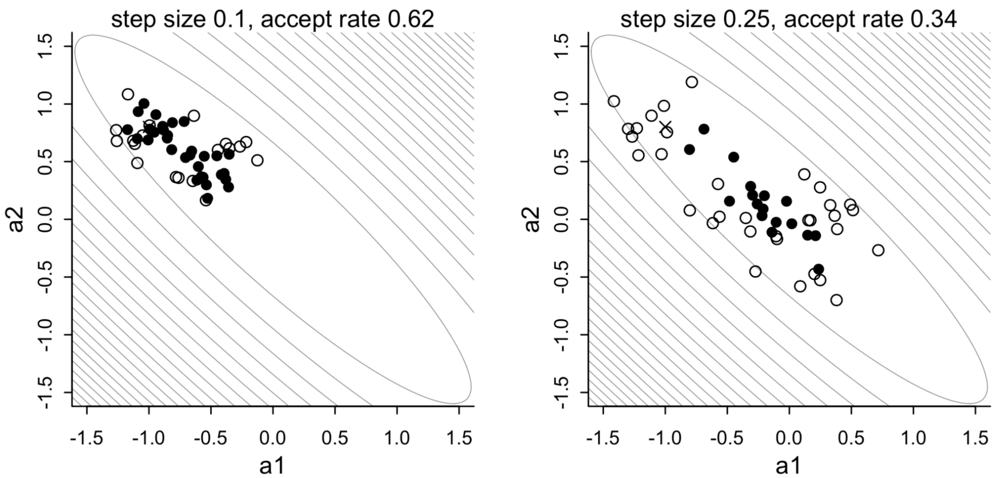
The skater in the pool
(between two kicks)



MCMC in Action

Metropolis gets stuck

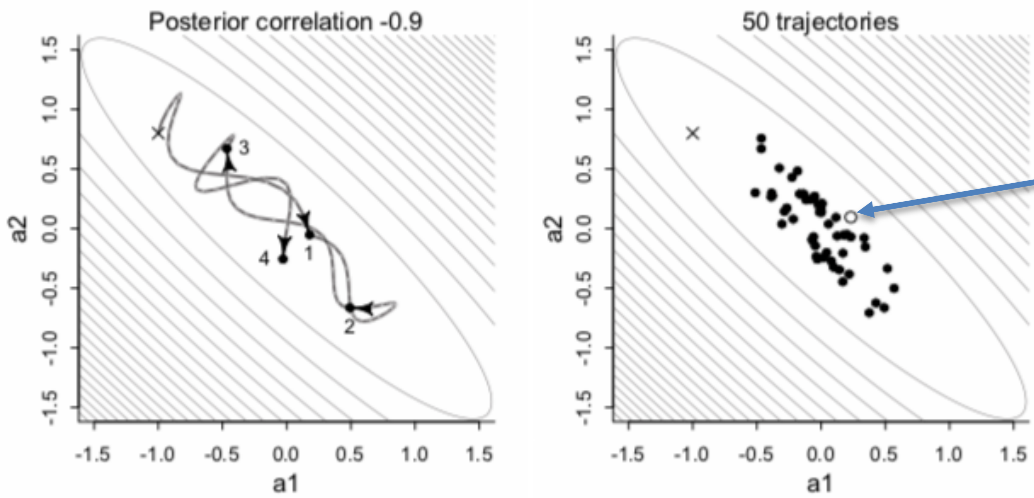
Metropolis



small steps, slow walk

bigs steps, low accept rate

HMC



Only one rejection

Facts about HMC

- Replace random moves with directed moves
  - Makes a number of leapfrog steps, typically around 40
  - This steps correspond to the movement of a physical particle
- Moves quickly in the space
- It fulfills detailed balance
- Needs gradient information

In Practice (see lr_mcmc.ipynb) it's just another sampler

```
def unnormalized_posterior(θ,D):
    …
    return unnormalized_post

#metropolis=tfp.mcmc.RandomWalkMetropolis(unnormalized_posterior,seed=42)
#metropolis=tfp.mcmc.NoUTurnSampler(unnormalized_posterior, step_size=100)
metropolis=tfp.mcmc.HamiltonianMonteCarlo(unnormalized_posterior,
                                          step_size=0.1,
                                          num_leapfrog_steps=40)
```

Code at: https://github.com/oduerr/ml-playground/blob/master/python/TFP_Bayes/lr_mcmc.ipynb

HMC esp. for people who had physics back a quarter of a century ago…

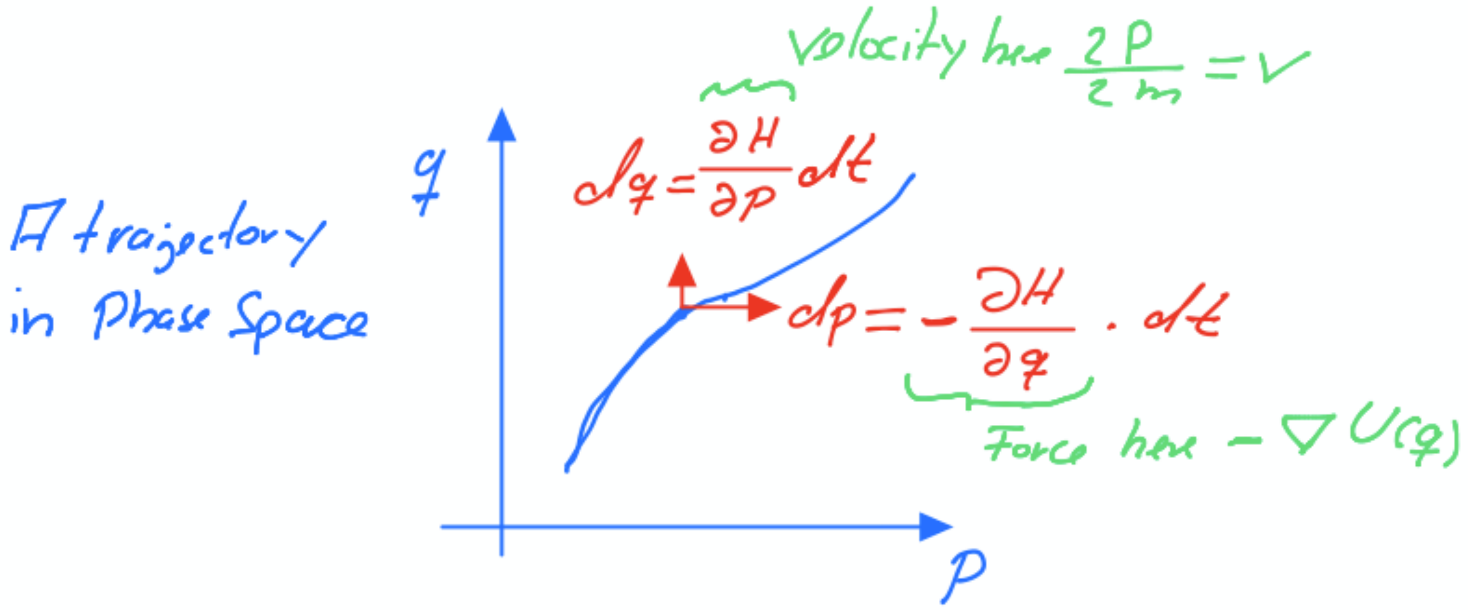
https://www.youtube.com/watch?v=DanoR0MXNrM

Recap (;-) Hamilton Mechanic

- Hamiltonian equation of motion (physics rulez p is momentum)

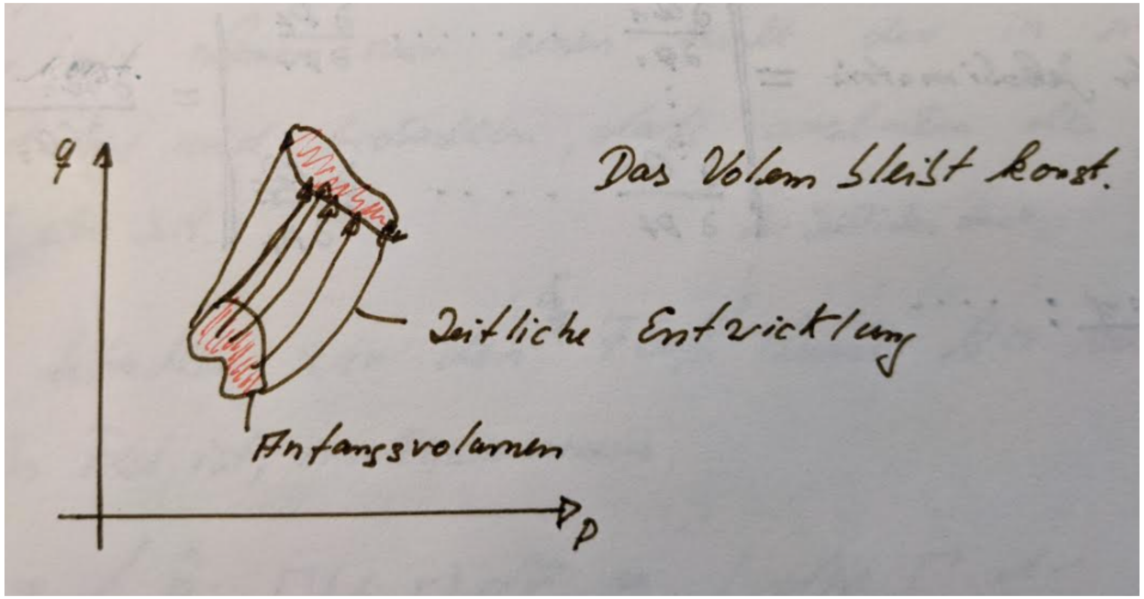
$$\frac{dq_i}{dt} = \frac{\partial H}{\partial p_i}$$
$$\frac{dp_i}{dt} = -\frac{\partial H}{\partial q_i}$$

- Hamiltonian used here $H(q,p) = U(q) + K(p) = U(q) + \frac{p^2}{2 \cdot m}$



Recap (;-) Hamilton Mechanic

- Facts:
  - Hamilton Dynamics is reversible
    - For this Hamiltonian just change p→-p (flying backwards)
  - Energy stays constant
    - If p and q follows equation of motion $H(p,q)$ is constant
  - Volume stays constant (Liouville Theorem)



Recap (;-) Statistical Mechanics

- Need connection of $H(q,p)$ and probability* density $P(q,p)$

- Canonical Ensemble**
  - System has constant temperature T
  - $P(q,p) = \frac{1}{Z} e^{-H(q,p)/T} = \frac{1}{Z} e^{-U(q)/T} \cdot e^{-\frac{\frac{p^2}{2m}}{T}} = \frac{1}{Z} P(q) \cdot P(p)$

* Sorry to the stats guys for capital P **We set $k_B = 1$

Does Algorithm sample from $P(q)$?

- Step 1 (Random Momentum, kick the skate boarder)
  - Does not change distribution in q

- Step 2 (Hamilton Equation / skating)
  - Moving from $(q, p) \to (q^*, p^*)$
  - New proposal state $(q^*, -p^*)$

  Hamilton Mechanic is reversible and leaves volume in phase space constant. Detailed Balance:

$$P^A_{(q,p)\to(q^*,-p^*)} = \min\left(1, \frac{P(q^*, -p^*)\cancel{T(q^*, -p^*|q, p)}}{P(q, p)\cancel{T(q, -p|q^*, p^*)}}\right)$$

$$P^A_{(q,p)\to(q^*,-p^*)} = \min\left(1, e^{H(q^*,-p^*)-H(q,p)}\right)$$

ensures that

$(q, p) \sim P(q, p)$

and so

$q \sim e^{U(q)}$

The U-Turn Problem

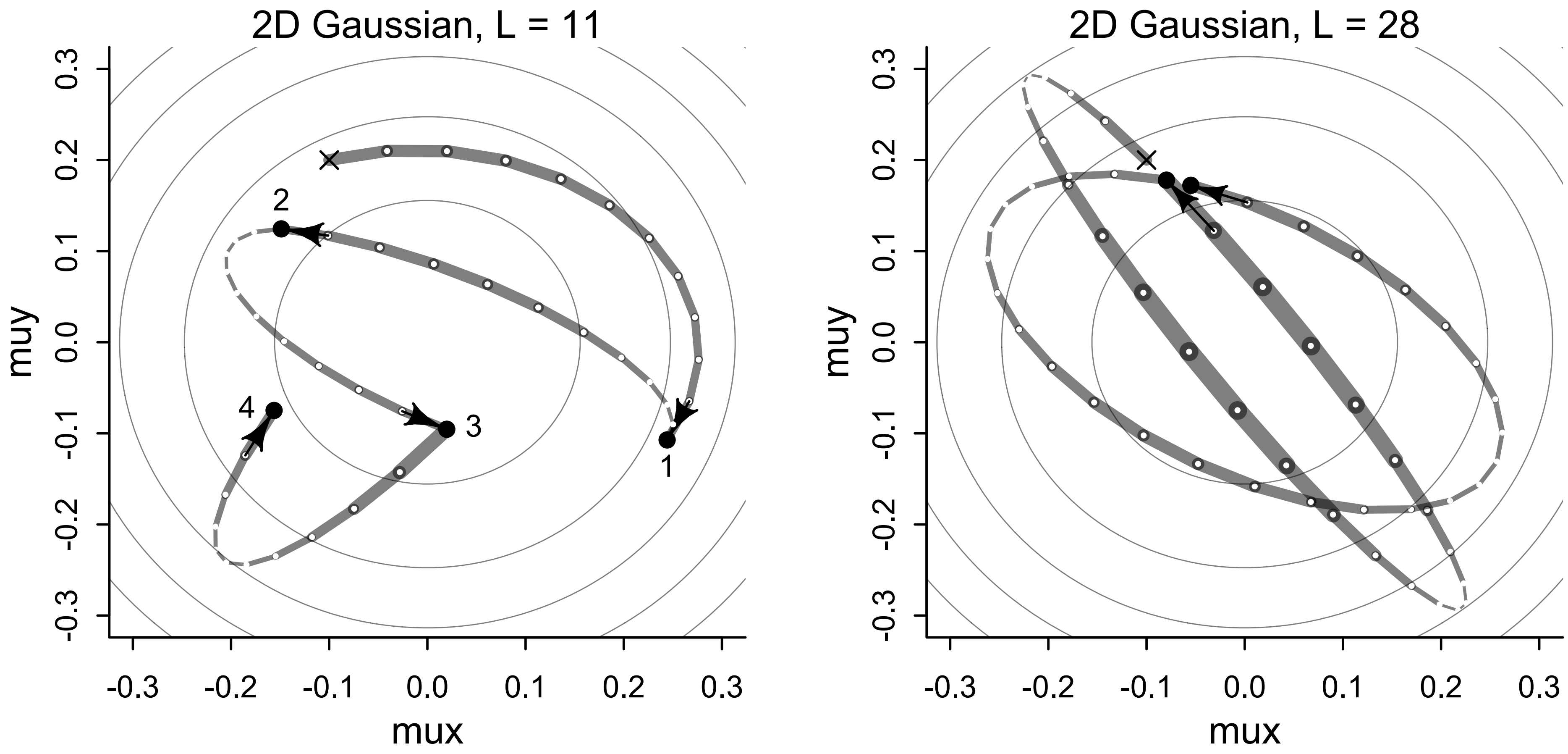
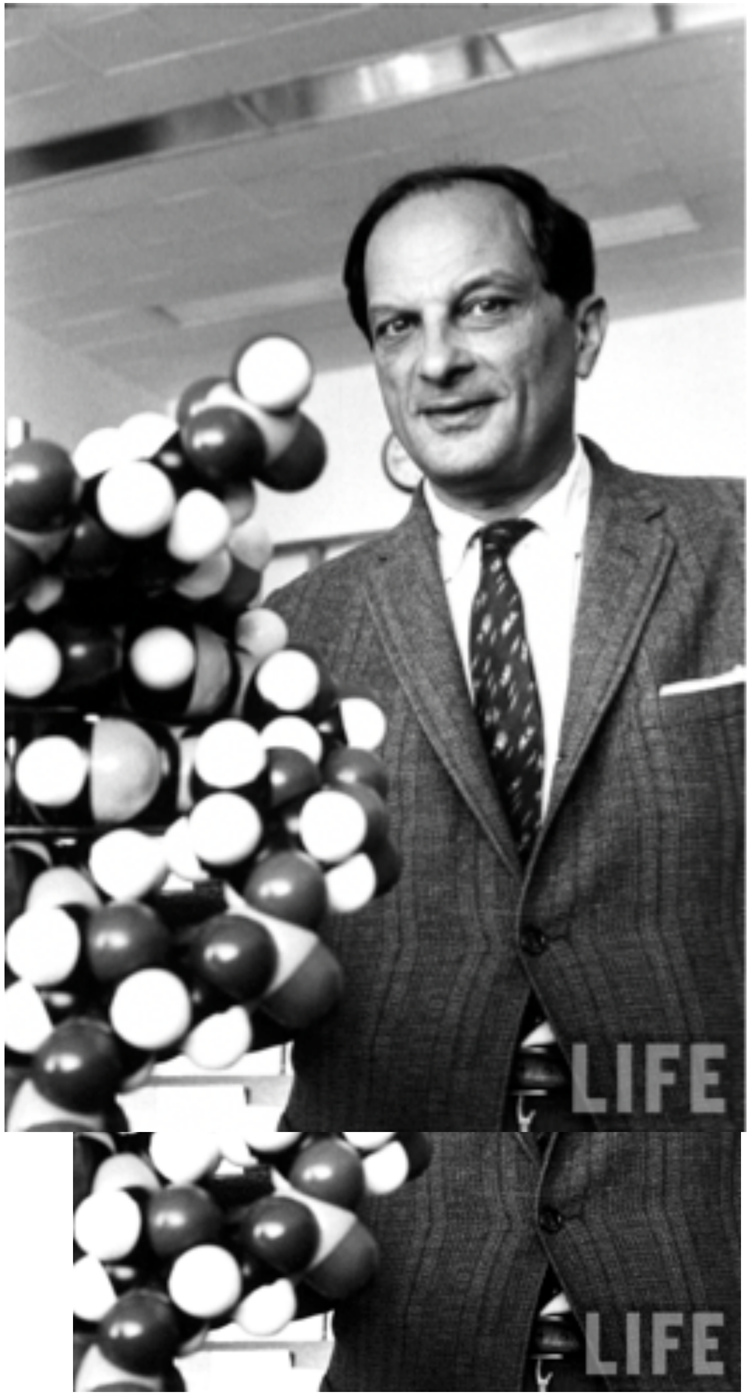


Figure 9.6

MCMC with Stan



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.

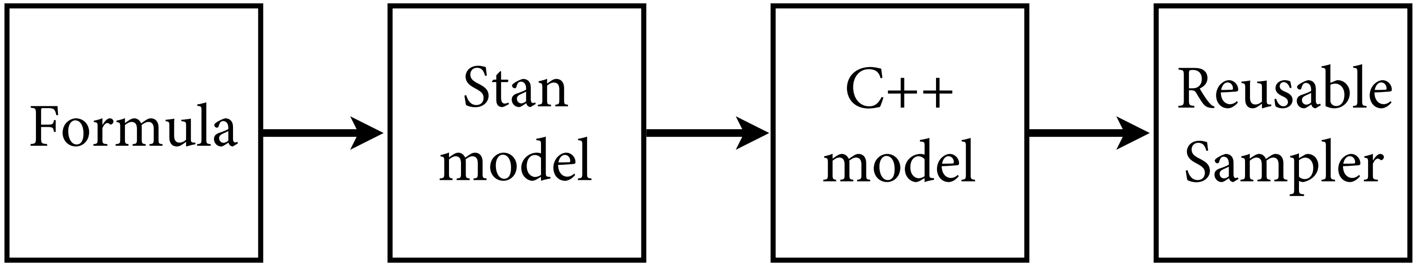
Stan is NUTS

No U-Turn Sampler

Automatic Step Size and Number Adaptation

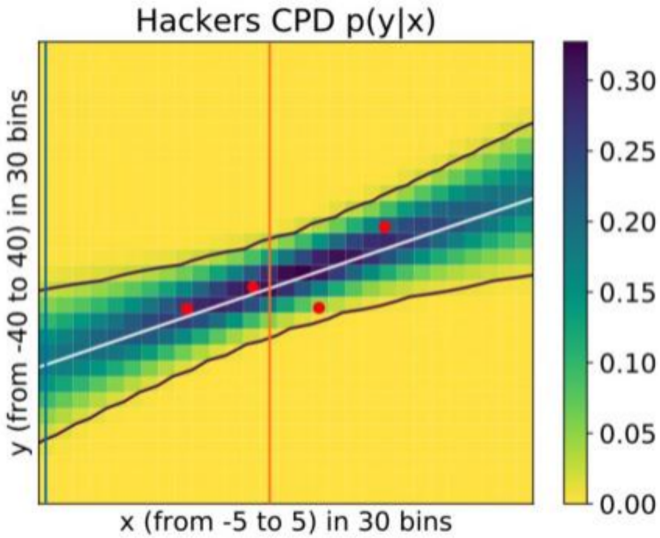
Rocks!

- No U-Turn Sampler (NUTS2): Adaptive Hamiltonian Monte Carlo
- Implemented in Stan (rstan: mc-stan.org)
- Stan figures out gradient for you
    - autodiff, back-propagation



Simple Linear Regression

From Daniel's talk



Linear regression

$$p(y|x) = N(y, a \cdot x + b, \sigma^2 = 1)$$

Priors (diagonal)

$$a \sim N(0,1)$$
$$b \sim N(0,1)$$

Stan (describing the model)

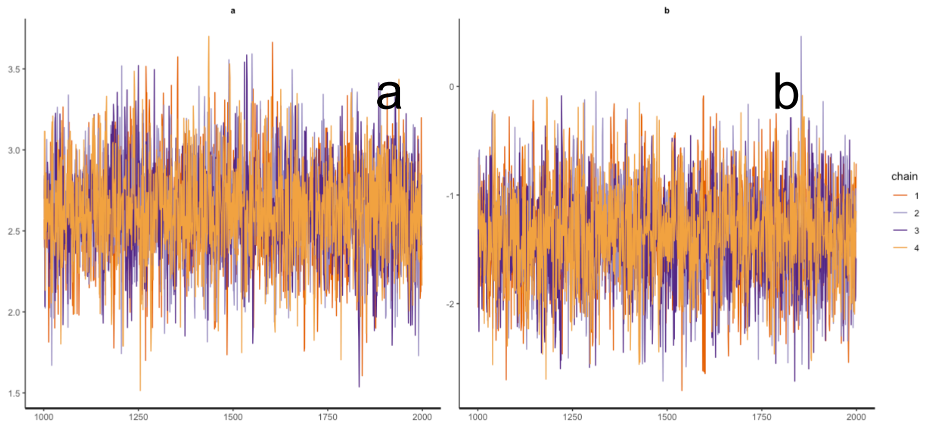
LR.stan × | LR.R* × | res × | compilearbeitsblaetter.R

Check on Save

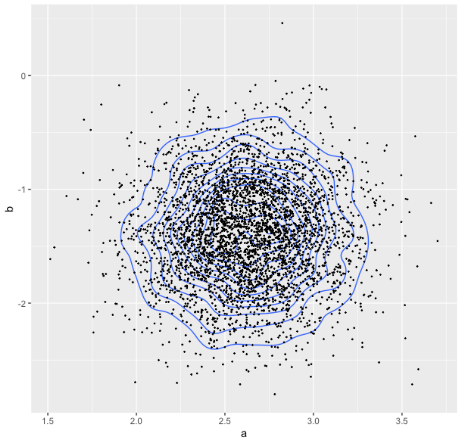
```
data{
    int<lower=0> N;
    vector[N] y;
    vector[N] x;
}

parameters{
    real a;
    #Here we have the possibility of constrains
    #real<lower=0> a;
    real b;
}

model{
  y ~ normal(a * x + b, 1);
  a ~ normal(0, 1); #Prior for a
  b ~ normal(0, 1); #Prior for b
}
```

Stan (samples of $p(\boldsymbol{\theta}|\boldsymbol{D})$)

```
samples = sampling(model, data=list(N=N,x=x, y=y))
samples
        mean se_mean   sd   2.5%    25%    50%    75%  97.5% n_eff Rhat
a       2.62    0.00 0.31   2.02   2.42   2.62   2.83   3.22  4117    1
b      -1.39    0.01 0.44  -2.26  -1.69  -1.39  -1.09  -0.50  3988    1
lp__  -34.59    0.02 0.97 -37.28 -34.95 -34.27 -33.91 -33.66  1831    1

traceplot(samples)
```



We see 4 chains which mixes well.
"Hairy Caterpillar"



Samples from the chains

We can also sample from posterior predictive, using the samples of a and b or via stan.

Further Reading

- Animations https://chi-feng.github.io/mcmc-demo/app.html

- HMC Overview
  - Statistical Rethinking Chapter 8

- HMC Advanced
  - Betancourt https://arxiv.org/abs/1701.02434
    - Focus more on intuition (some things not so clear, in the second look)
    - Talks:
      - https://www.youtube.com/watch?v=jUSZboSq1zg
      - https://www.youtube.com/watch?v=pHsuIaPbNbY
  - Radford Neal https://arxiv.org/abs/1701.02434
- Stochastic Gradient Hamiltonian Monte Carlo
  - https://arxiv.org/abs/1402.4102