General Metropolis Algorithm

Peter Büchel

HSLU W

SA: Week 12

Markov Chain Monte Carlo (MCMC)

- Present methods to generate good approximations of Bayes' posterior distributions
- Class of methods: *Markov Chain Monte Carlo* (MCMC)
- MCMC algorithms: Perform Bayesian data analysis for realistic applications that would have been virtually impossible 30 years ago
- Explain some of essential ideas of MCMC methods to understand sophisticated software of MCMC methods
- Proceed to derive probability θ that coin will show head from a series of observed tosses

- Seen: Grid approximation
- Seen: Exact solution for posterior
- Most of the time: Not practical
- Approximate posterior distribution numerically
- Markov Chain Monte Carlo (MCMC) algorithms
- Method described starts from two assumptions:
  - Prior distribution: Function that can be easily evaluated by a *computer*
  - That is, for θ the value of $P(\theta)$ should be easy to determine
  - Likelihood function: Function that can be easily evaluated by a *computer*

- If conditions are satisfied, calculate numerator of Bayes theorem

$$P(D \mid \theta) \cdot P(\theta)$$

- Posterior distribution is proportional to numerator of Bayes theorem:

$$P(\theta \mid D) \propto P(D \mid \theta) \cdot P(\theta)$$

- Approximation of posterior distribution with expression on the right except for multiplicative constant

- To get probability distribution, divide expression on the right by

$$P(D) = \sum_{\theta} P(D \mid \theta) \cdot P(\theta)$$

- Sum of probabilities of posterior distribution then becomes 1

- Method produces *approximation* of posterior probability $p(\theta \mid D)$ in terms of a large number of θ values sampled from this distribution
- Method does not require evaluation of difficult integral in denominator of Bayes' theorem to be computed in the case of a continuous random variable θ
- Set of representative θ values: Used to estimate central tendency, for example, mode, posterior distribution and interval of highest densities (HDI), and so on
- Posterior distribution is estimated by randomly generating a set of θ values from it
- This approach is called Monte Carlo method by analogy with random events in games in a casino

Approximation of a Distribution Mean Large Samples

- Concept of representing a distribution using a large sample of representative θ values is fundamental to approach to Bayesian analysis of complex models
- Idea is intuitively and routinely applied in everyday life and in science
- For example, polls and surveys are based on this concept: By randomly selecting subset of people from a population, can estimate underlying trends in population as a whole
- The larger the sample, the better the estimate

Example

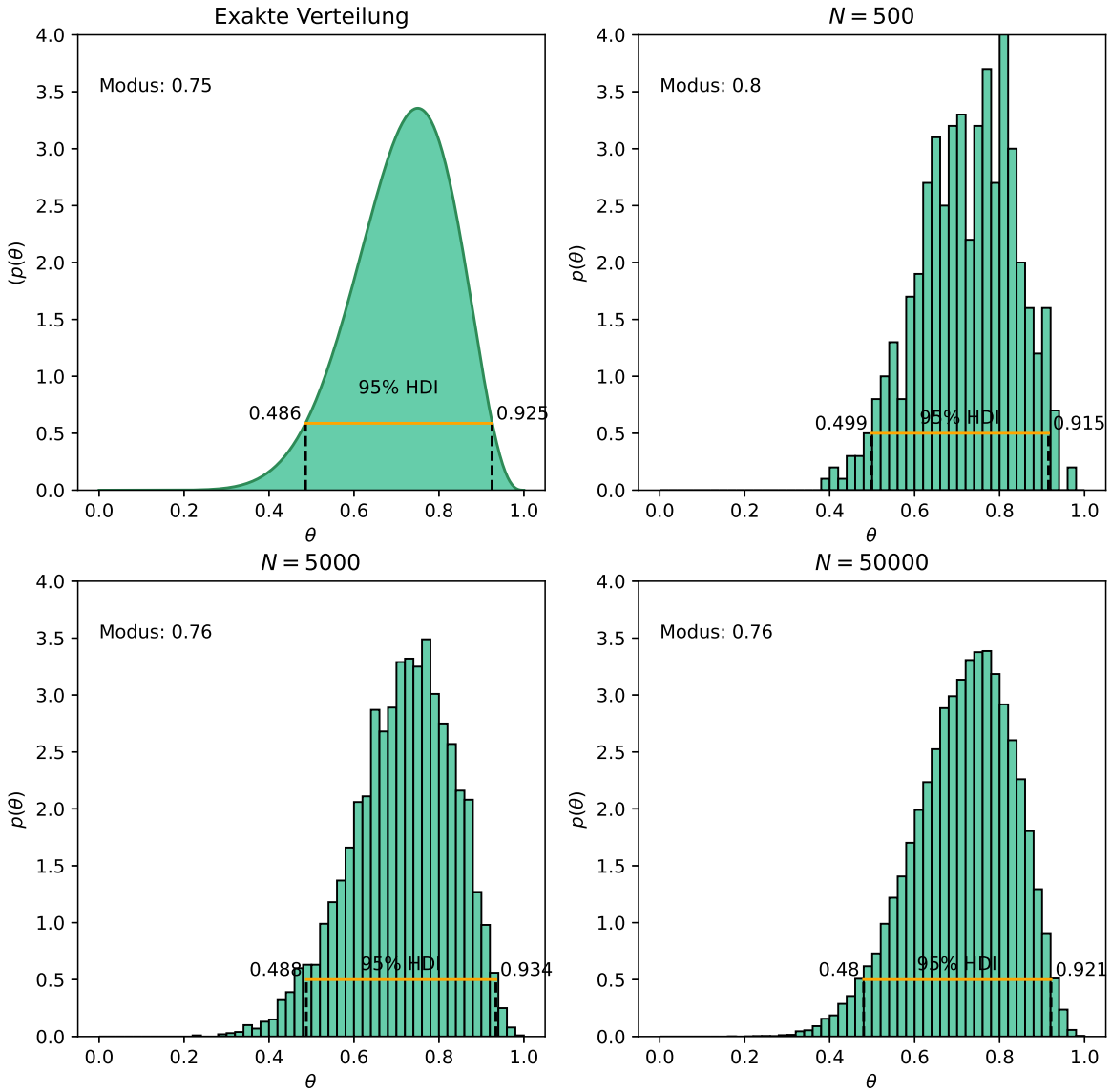
- Election polls ask about 1000 randomly selected people
- If, for example, 60 % of these 1000 people are in favour of party A and 40 % are in favour of party B
- This should also be true for total population

- Novel aspect to present application: Sampling from a mathematically defined distribution, such as a posterior distribution

- Chosen distribution: Beta distribution to show MCMC on already known posterior distribution

Example

- Figure:



- Figure: Approximations of an exact mathematical distribution by large random samples of representative θ values
- Top left: Exact beta distribution Beta($\theta \mid 10, 4$)
- Could be posterior distribution for estimating underlying probability of tossing head of a coin
- Exact mode of distribution and 95 %-HDI are also shown
- Exact values from mathematical formula of beta distribution

- Approximate exact values by randomly selecting a large number of representative θ values from distribution
- Top right: Histogram of 500 random representative θ values
- Only 500 values: Histogram is not smooth
- Estimated mode and 95 %-HDI close to true values
- But unstable: Different random sample of 500 representative θ values would give noticeably different estimates
- Bottom left: Approximation with a larger sample of 5000 representative θ-values
- Bottom right: An even larger sample of 50 000 representative θ-values

- Histogram becomes smoother with larger sample sizes and estimated values will be closer (on average) to true values
- Computer generates pseudo-random values: Looks random but are not

Introductory Example for Metropolis Algorithm

- Goal in Bayesian inference: Obtain as accurate a representation of posterior distribution as possible
- One way to achieve this is to randomly draw a large number of representative θ values from posterior
- Question: *How* can a large number of representative θ values be drawn from a distribution?
- To find an answer, let's ask a politician

A Politician Stumbles over Metropolis Algorithm

- Suppose an elected politician lives on long chain of islands
- Is constantly travelling from island to island to stay in public eye
- At the end of day, he has to decide whether to
  - (i) stay on his current island
  - (ii) move on to neighbouring island west
  - (iii) move on to neighbouring island east
- His aim: Visit all islands with probability proportional to their relative population
- Spends most of his time on most populated islands and *proportionally* less time on less populated islands

- He has no idea what total population of island chain is
- He does not even know exactly how many islands there are
- However, his advisors are able to gather some minimal information
- They can ask mayor of island they are on how many people live on that island
- If politician wants to visit a neighbouring island, they can ask mayor of neighbouring island how many people live on that island

- Politician has a simple heuristic to decide whether he *may* travel to neighbouring island (so-called proposed island)
- First: Tosses (fair) coin to decide whether to visit neighbouring island to east or west
- If proposed island has larger population than current island, then he definitely goes to proposed island
- If proposed island has smaller population than current island, then he randomly goes to proposed island with probability proportional to population of current island
- If population of proposed island is half that of current island, then probability of going there is only 50 %

- If neighbouring island has three quarters of population of current island, he changes with a probability of 75 %
- This ensures that he does not always stay on largest island once he arrives there
- Denote population of proposed island as B_{proposed} and population of current island as B_{current}
- Then he moves to less populated island with probability

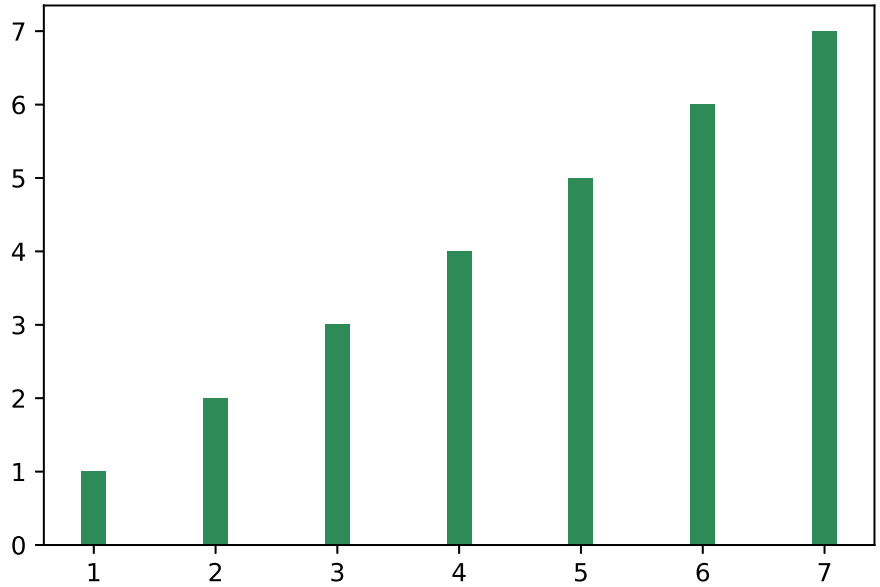
$$P_{\text{change}} = \frac{B_{\text{proposed}}}{B_{\text{current}}}$$

- Politician does this by spinning a fair spinner, which is marked on rim with uniform values from zero to one
- If value shown is between zero and P_{change}, then he changes
- If population B_{proposed} is very large, i.e., close to B_{current}, then he changes island with high probability
- If B_{proposed} is very small compared to B_{current}, he is very unlikely to change island
- With this heuristic: Politician visits big islands more than small ones
- He does not get stuck on largest island: Changes islands with some probability, even if neighbouring islands have smaller populations

- The amazing thing about this heuristic is that it works
- In the long run, probability that politician is on one of islands is same as relative population of each island, even though population size is unknown to politician

Example

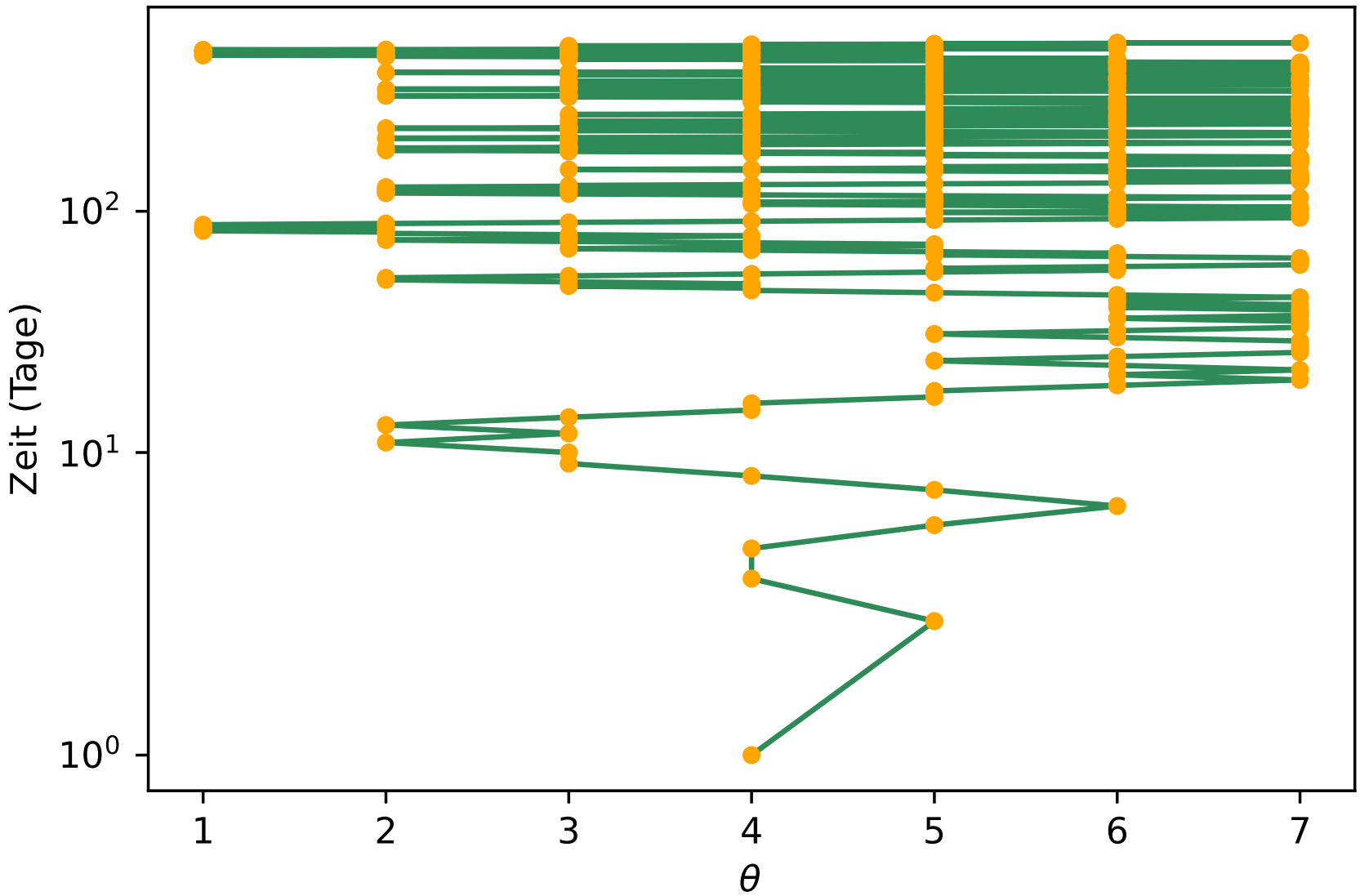
- Figure:



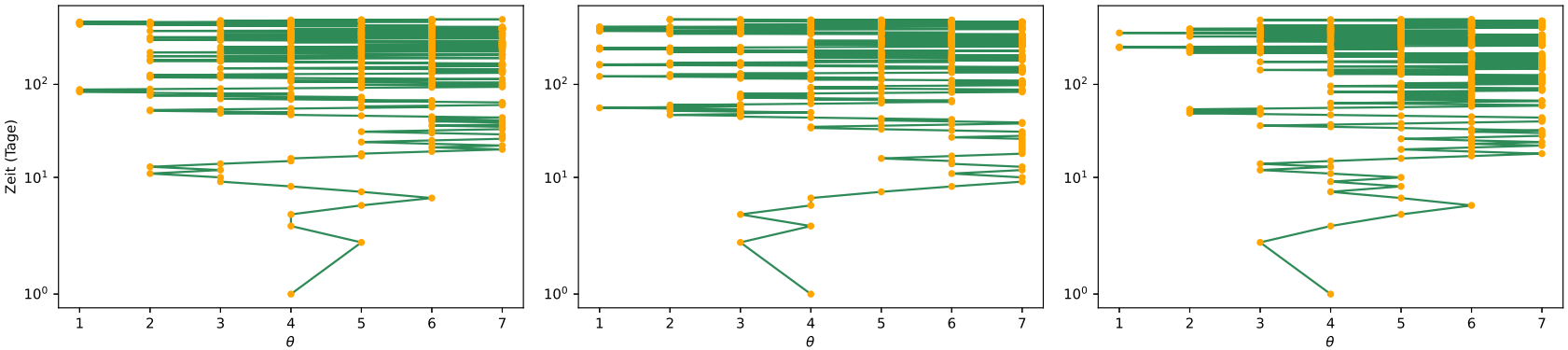
- Suppose: Seven islands in chain, with relative populations as in Figure
- Islands are indexed by θ and denote outermost western island by $\theta = 1$ and outermost eastern island by $\theta = 7$
- Important: *We* know distribution of population, politician does *not*

- Relative populations increase linearly so that $B(\theta) = \theta$
- Capital letter $B()$: Denotes *relative* population of island and not its absolute
- Thus, the population of island $\theta = 7$ is seven times larger than population of island $\theta = 7$
- To complete the model: Islands to left of 1 and to right of 7 have population 0
- Politician can propose to jump to these islands, but proposal will always be rejected because population is 0
- Probability of moving to these islands is 0

- Figure:



- Figure:



- Figures: Three possible trajectories of politician
- Consider the left trajectory
- Each tag: Unique increment indicated on vertical axis
- Plot: On first day ($t = 1$), politician on middle island in chain, i.e., $\theta_{\text{current}} = 4$

- To decide where to go on second day: Tosses coin and suggests going either one position to left or one position to right
- In this case: Coin suggests going to right, so $\theta_{\text{proposed}} = 5$
- Since population at proposed position is greater than population at current position (i.e., $B(5) > B(4)$), proposed movement is accepted
- In trajectory: This movement is evident because for $t = 2$ is $\theta = 5$

- For $t = 3$: Coin toss suggests a move to left, so $\theta_{\text{proposal}} = 4$
- Probability of accepting this suggestion is:

$$P_{\text{change}} = \frac{B(\theta_{\text{proposed}})}{B(\theta_{\text{current}})} = \frac{4}{5} = 0.80$$

- Politician spins fair spinner, which has marks between 0 and 1 on its edge
- Spinner happens to show value less than 0.80, say:

```
runif(1, 0, 1)
## [1] 0.2655087
```

- Therefore: Politician accepts proposal and moves to proposed island
- Trajectory thus jumps to $\theta = 4$ at time $t = 3$
- For $t = 4$: Coin toss again suggests a switch to left
- Probability of accepting this suggestion:

$$P_{\text{change}} = \frac{B(\theta_{\text{proposed}})}{B(\theta_{\text{current}})} = \frac{3}{4} = 0.75$$

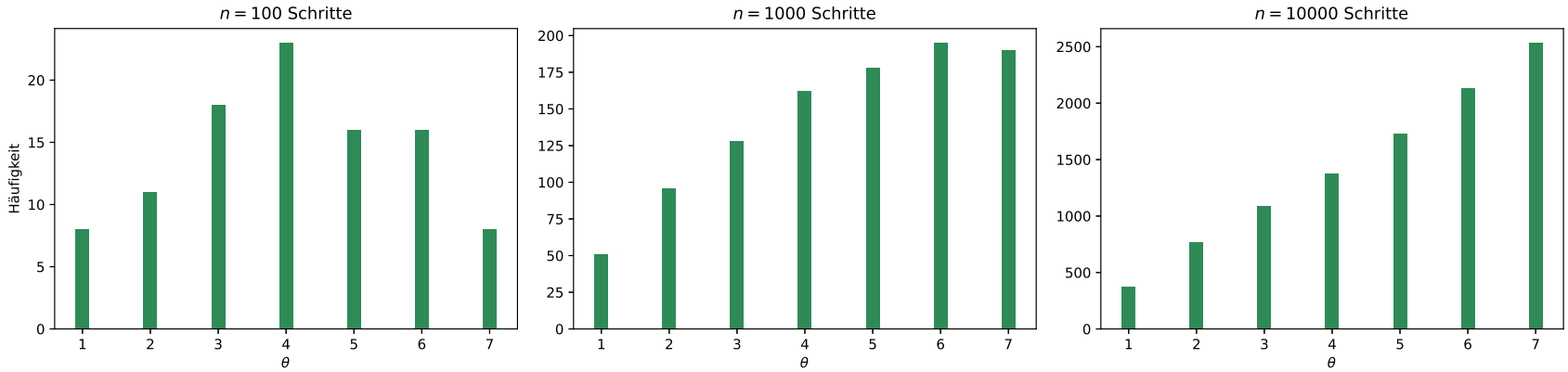
- Politician then spins fair spinner again, which happens to show a value greater than 0.75, say:

```
runif(1, 0, 1)
## [1] 0.9889093
```

- Therefore: Politician rejects proposed island change and stays on current island
- Thus, trajectory shows that θ is still 4 for $t = 4$
- Figure shows trajectories for first 500 steps in this *random walk* across islands
- Scale of time step is plotted logarithmically: See details of early steps, but also trend of later steps

- Since randomness is involved here, trajectories also look different
- See roughly: Politician visits populous islands more often than sparsely populated ones
- However: True distribution is not yet quite obvious

- Figure:



- For $n = 100$ steps: Histogram shows days stayed on each island
- Does not yet look at all like original distribution
- But as number of steps increases, histograms become more similar to original distribution
- For $n = 10000$: Histogram no longer distinguishable from shape of original distribution

- Crucial point: *Exact* distribution *not* known to politician
- Can only look a little to left and right and determine population of current island and proposed island
- Using procedure described: Sample original distribution and obtain representative θ values of this distribution
- Important: After politician have moved to new island or have stayed, he forgets what populations of current and neighbouring islands are
- So they have to start anew every day
- Such processes: Called *Markov Chain* (first MC of MCMC)
- Second MC of MCMC, as already mentioned, refers to *Monte Carlo*

Metropolis Algorithm in General

- Procedure described before: Only special case of more general procedure known as *Metropolis algorithm*
- Named after first author of famous article (Metropolis, Rosenbluth & Rosenbluth, Teller & Teller, 1953)
- Before simple case of:
  - (i) discrete positions
  - (ii) in *one* dimension
  - (iii) with suggestions for steps that change only by *one* position to left or right
- Example of politician: Target distribution $B(\theta)$ was population size on each island θ

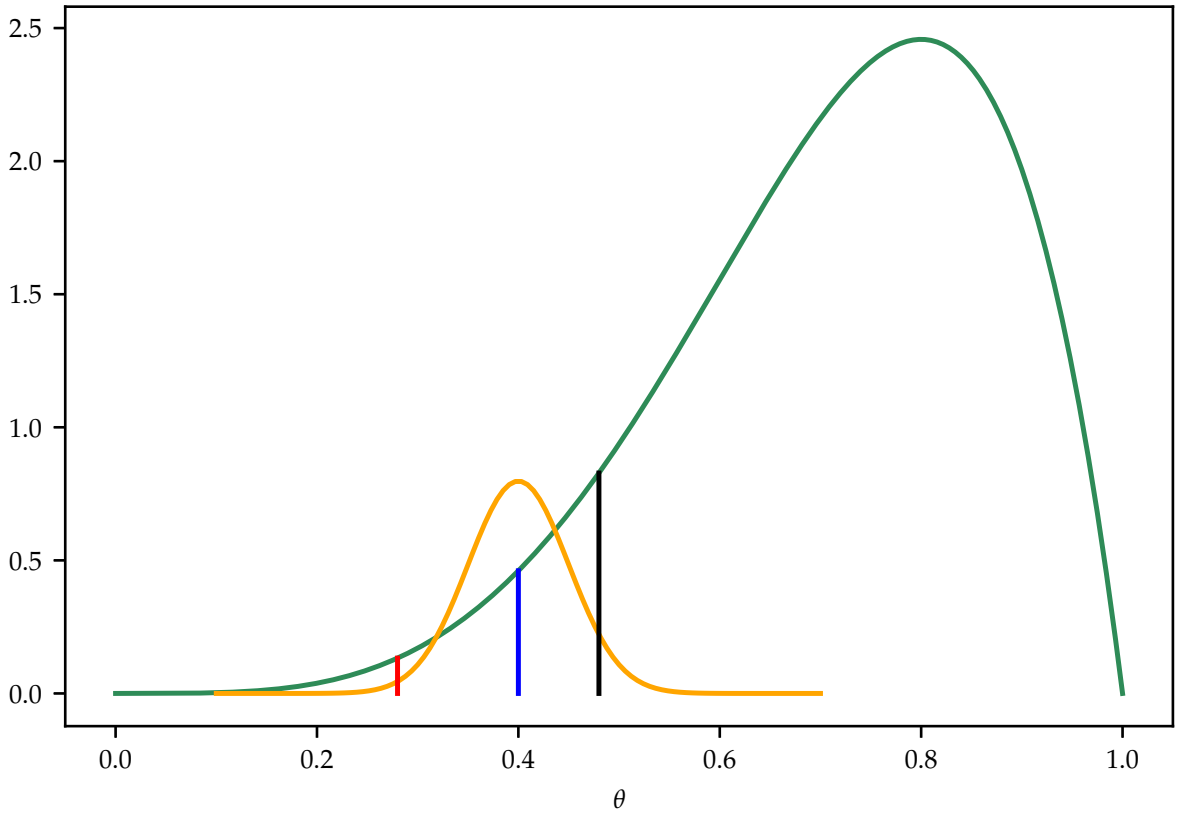
- General Metropolis algorithm applies to
  - (i) continuous step sizes
  - (ii) in any number of dimensions
  - (iii) with more general proposal distributions
- Goal: Approximation of continuous posterior distribution
- Specifically: Metropolis algorithm generates many representative θ values whose histogram approximates posterior distribution given a sufficiently large number of sample θ values
- Show this for beta distribution
- Know what beta distribution looks like and can show that Metropolis algorithm produces what it should

- Principle of general Metropolis method is identical to Metropolis algorithm for discrete models
- Target distribution $B(\theta)$ over a multidimensional continuous parameter space from which we wish to generate representative sample values of θ
- Need to be able to calculate value of $B(\theta)$ for each possible value of θ
- However: Distribution $B(\theta)$ does not have to be normalised
- It only needs to be non-negative
- In typical applications of Bayesian inference: $B(\theta)$ is non-normalised posterior distribution for θ
- I.e.: Product of likelihood function and prior distribution

- Sample values from target distribution by a random walk through parameter space:
  - Walk starts at any point specified by user which is non-zero
  - Random walk advances at each time step by proposing move to new position in parameter space and then deciding whether or not to accept proposed move
  - Proposal distributions can take many different forms: Goal to use proposal distribution that *efficiently* explores regions of parameter space where $B(\theta)$ has largest part of its distribution
  - Use proposal distribution: Random values for can efficiently generated
  - For our purposes: Proposal distribution normally distributed and centred at current position
  - Idea behind using a normal distribution: Suggested step is typically close to current position, with probability of suggesting a more distant position falling off according to normal curve
  - Computer languages such as `R` or `Python` have built-in functions for generating pseudorandom values from a normal distribution

Example

- Figure:



- Beta distribution Beta$(\theta \mid 5, 2)$: Curve in green
- Target distribution $B(\theta)$ is given directly by Beta$(\theta \mid 5, 2)$

- If at point $\theta_{\text{current}} = 0.4$: Not only to decide whether to go left or right
- But also *how far* to go left or right
- Decision is made randomly, according to orange normal distribution curve
- Proposed values for θ_{proposed} near 0.4 are more likely than those far away
- For example: Generate proposal for next step from normal distribution with mean 0.4 and standard deviation (SD) 0.2
- Determine new position with R:

```
rnorm(n = 1, mean = 0.4, sd = 0.2)
## [1] 0.4873186
```

- Proposed value: $\theta_{\text{proposed}} = 0.487$
- Now $B(\theta_{\text{proposed}})$ corresponds to black line in Figure
- $B(\theta_{\text{current}})$ corresponds to blue line
- Thus:

$$\frac{B(\theta_{\text{proposed}})}{B(\theta_{\text{current}})} > 1$$

- Accept proposal
- However, if proposal $\theta_{\text{proposed}} = 0.277$:

```
rnorm(n = 1, mean = 0.4, sd = 0.2)
## [1] 0.2768375
```

- $B(\theta_{\text{proposed}})$ would correspond to red line and:

$$P_{\text{change}} = \frac{B(\theta_{\text{proposed}})}{B(\theta_{\text{current}})} < 1$$

- Uniformly distributed random number in interval $[0, 1]$:

```
runif(n = 1, min = 0, max = 1)
## [1] 0.1848823
```

- If random number is between 0 and P_{change}, then the proposal is accepted and the step

- Process repeats and in the long run positions visited by random walk converge to target distribution

- As it is in this case

- Procedure: Same as in discrete case, except for different proposal distribution

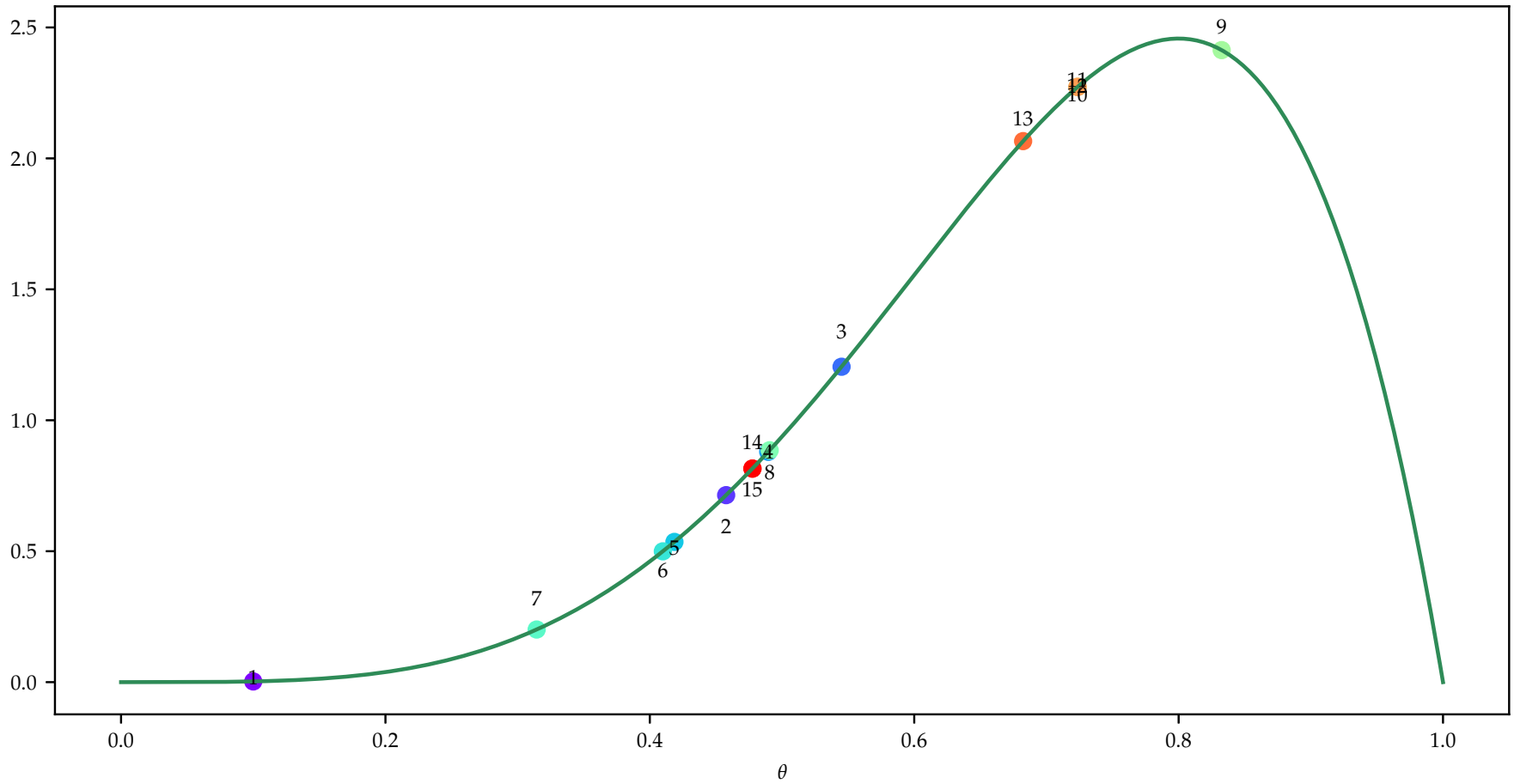
Example

- Beta distribution $\text{Beta}(\theta \mid 1, 1)$ as prior distribution with data $N = 5$ and $z = 4$
- Posterior distribution: $\text{Beta}(\theta \mid 5, 2)$
- Target distribution for $B(\theta)$: Choose $P(D \mid \theta)p(\theta)$
- In this example:

$$\begin{aligned} B(\theta) &= P(D \mid \theta)p(\theta) \\ &= \text{Bernoulli}(4, 5 \mid \theta)\text{Beta}(\theta \mid 1, 1) \\ &\propto \text{Beta}(\theta \mid 5, 2) \end{aligned}$$

- Here (normalised) posterior distribution $\text{Beta}(\theta \mid 5, 2)$ for simplicity
- Want to know: Approximates Metropolis algorithm as described in posterior distribution

- Figure:



- First 15 steps of random walk for Beta($\theta \mid 5, 2$) plotted

1. Start point $\theta_1 = 0.1$: Step is normally distributed around mean θ and standard deviation 0.2

2. Propose step to right to location $\theta_2 \approx 0.45$: Proposal accepted since $B(\theta_2) > B(\theta_1)$

3. Propose step to right to $\theta_3 \approx 0.55$: Proposal accepted as in 2.

4. From 3. follows proposal $\theta_4 \approx 0.5$
   - $B(\theta_4) < B(\theta_3)$: Randomly draw a number between 0 and 1
   - If number smaller than $\frac{B(\theta_4)}{B(\theta_3)}$: Accept proposal, which happened in this case

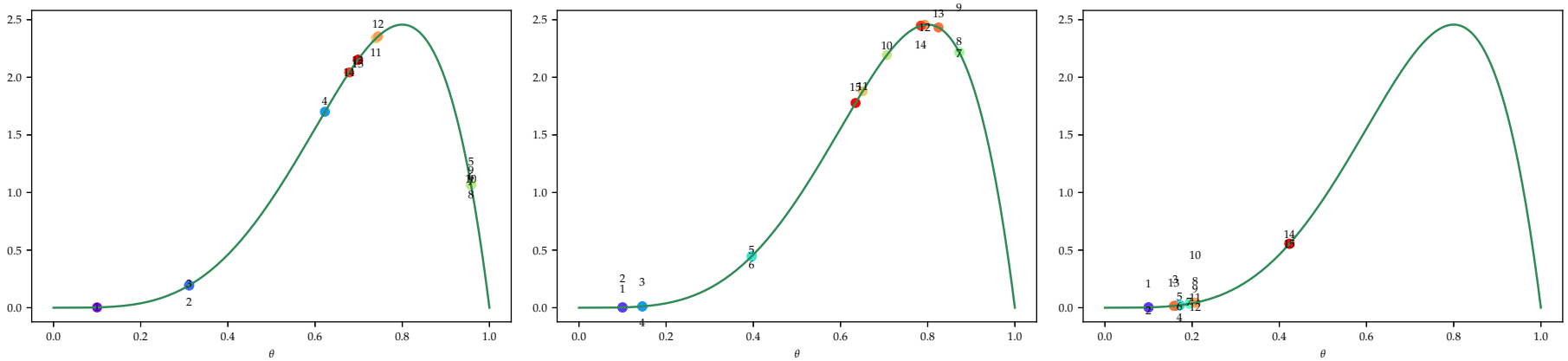
9. At θ_8: Proposal $\theta_9 \approx 0.83$: Proposal accepted because $B(\theta_9) > B(\theta_8)$

10. Propose $\theta_{10} \approx 0.72$:
    - Here $B(\theta_{10}) < B(\theta_9)$: Randomly select number between 0 and 1
    - If number is less than $\frac{B(\theta_{10})}{B(\theta_9)}$: Accept proposal, which happened in this case

11. Proposal to move either to left with $\theta_{11} \lessapprox 0.72$ or to right with $\theta_{11} \gtrapprox 0.85$.
    - In both cases, $B(\theta_{11}) < B(\theta_{10})$, and randomly choose a number between 0 and 1
    - If number is less than $\frac{B(\theta_{11})}{B(\theta_{10})}$: Accept proposal, which in this case *did not* happen
    - So: $\theta_{11} = \theta_{10}$

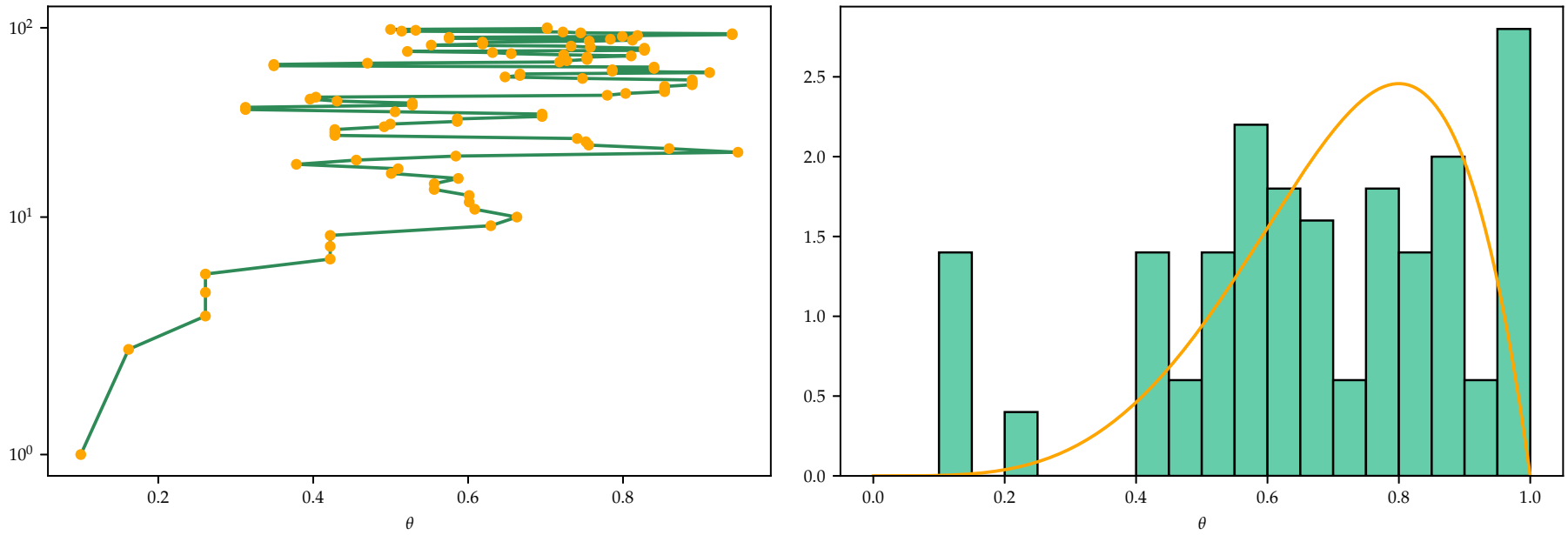
12. Same as in 11. also happens for θ_{12}

- Start over again: Random walk will certainly look different
- In Figure: Three such random walks:



- Beginning of random walk: Do not know whether it samples distribution of θ *representatively*
- Random walks on left and in middle of indicate this, but plot on the right
- Obviously need to take many more steps

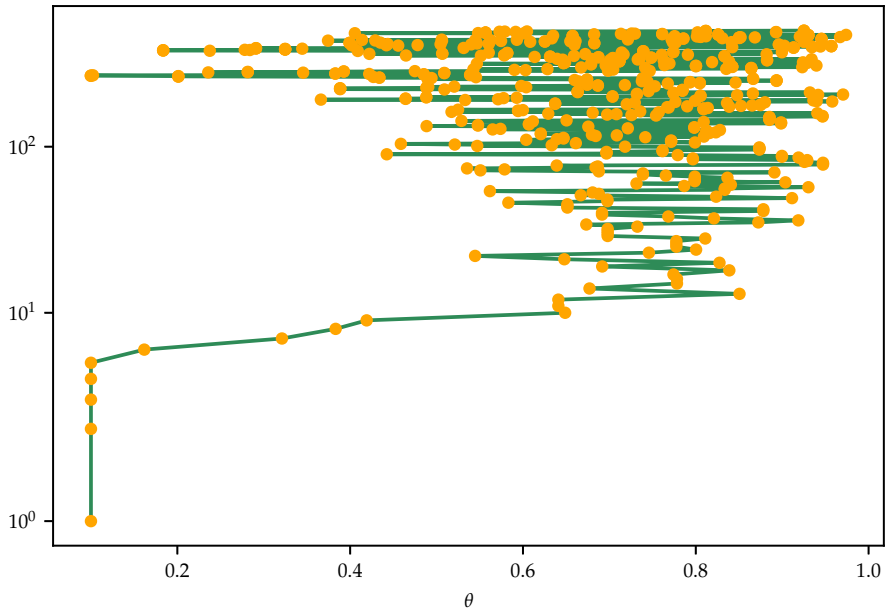
- Figure:

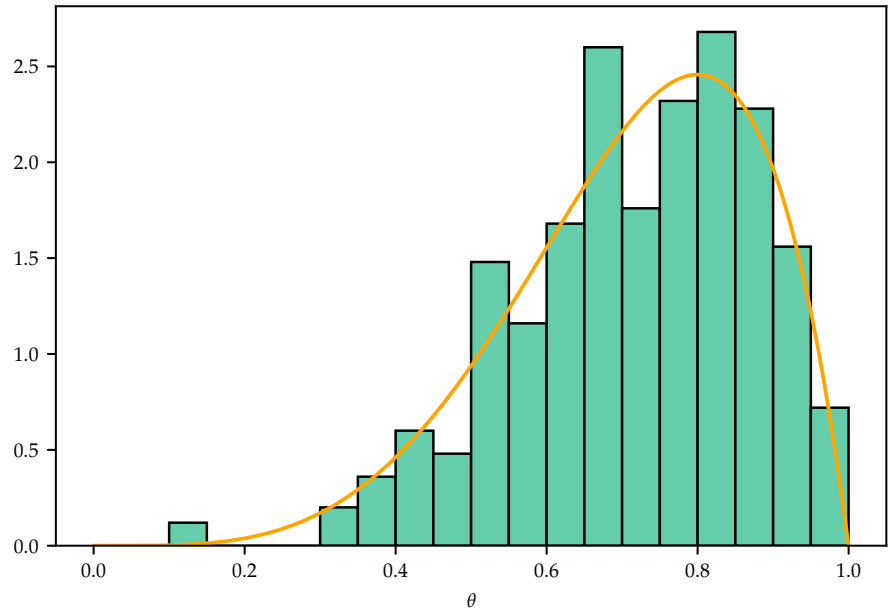


- 100 steps

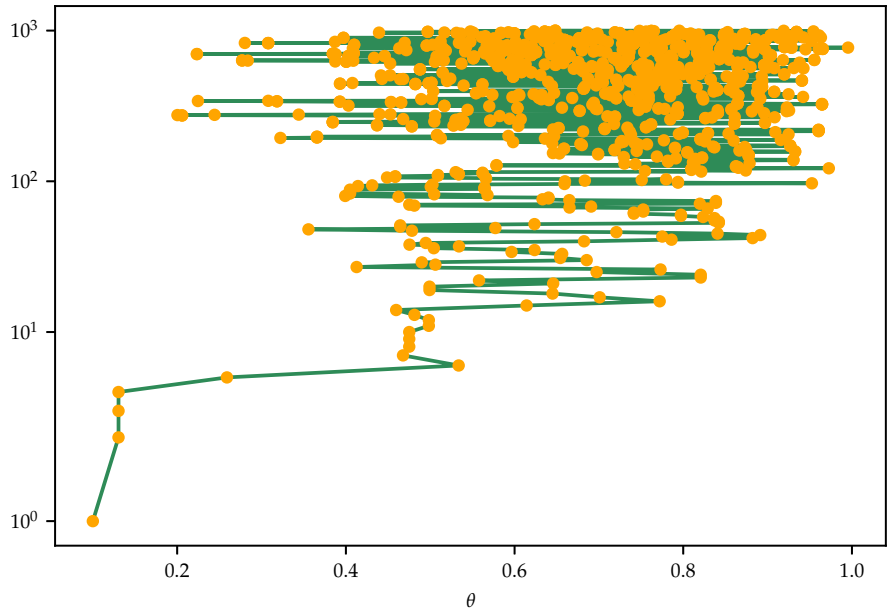
- Histogram on the right: Rudimentary approximation of histogram to density function, but it is not yet very clear

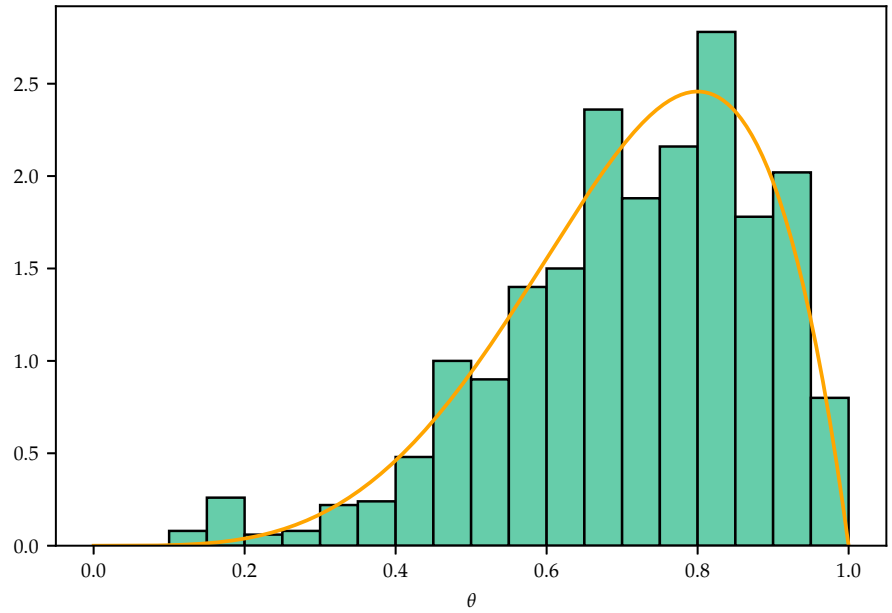
- 500 steps:


10^2
10^1
10^0
0.2
0.4
0.6
0.8
1.0
θ

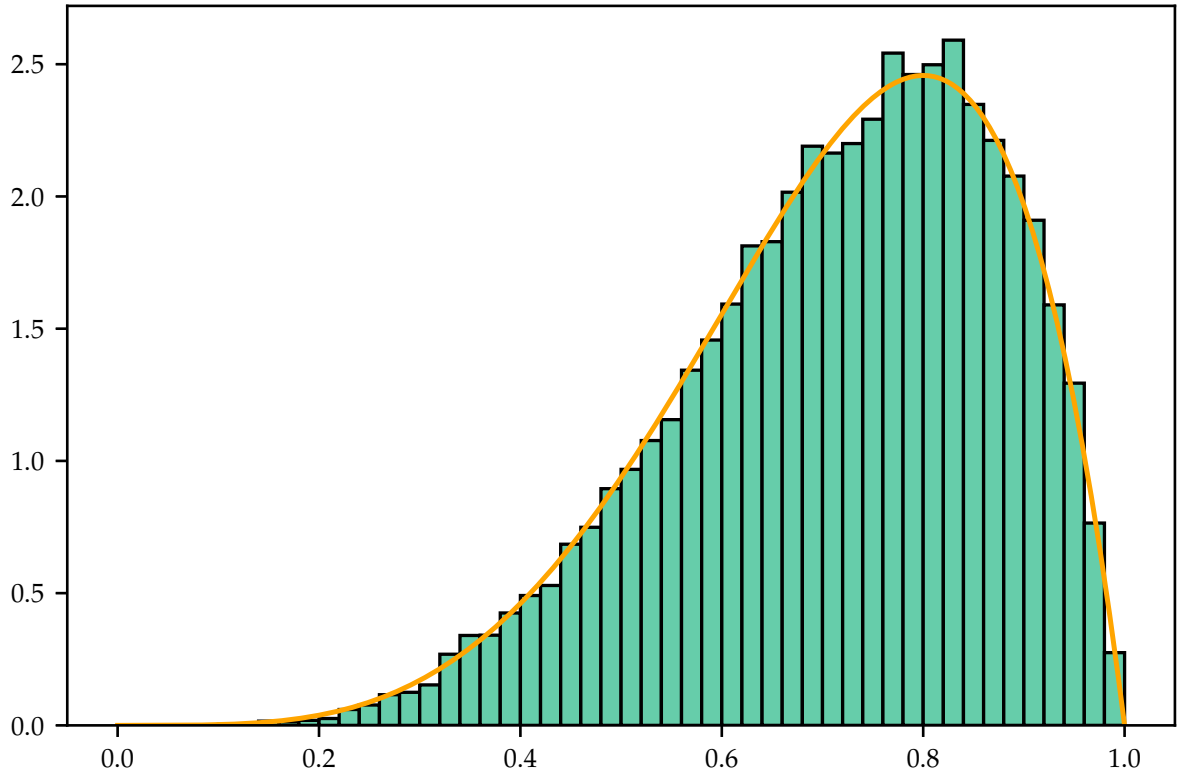

2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
θ

- 1000 steps:


10^3
10^2
10^1
10^0
0.2
0.4
0.6
0.8
1.0
θ


2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
θ

- 50 000 steps:



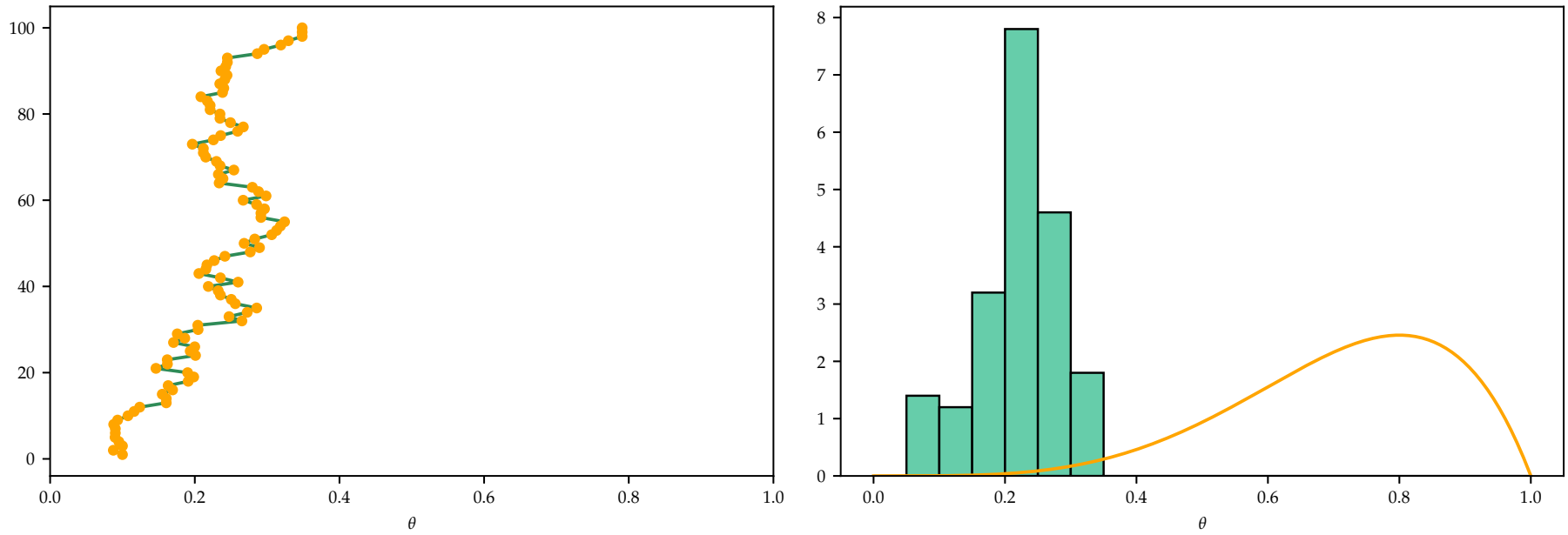
- Emphasise again: Metropolis algorithm does not "know" what distribution looks like, it just samples it

Questions

- First: How to choose initial value of θ?
  - Only requirement: $B(\theta)$ positive
  - In examples: Choice was not clever, since $\theta = 0.1$ far from maximum
  - However: Generally do not know where distribution has its maximum
  - Software: Analyses input and cleverly choose initial value of θ
- First steps random walk often not representative: Omitted
- Another problem: Choosing step size of standard deviation

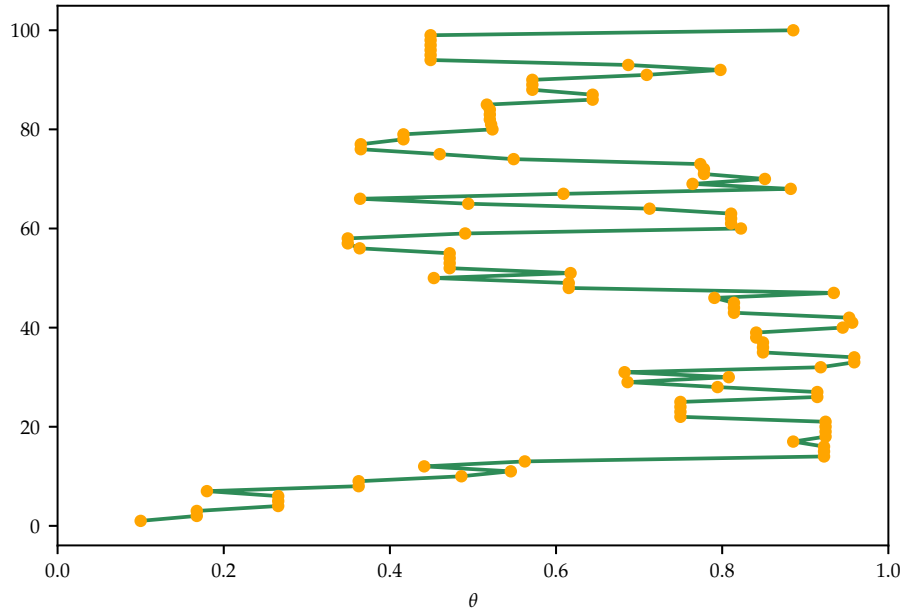
Example

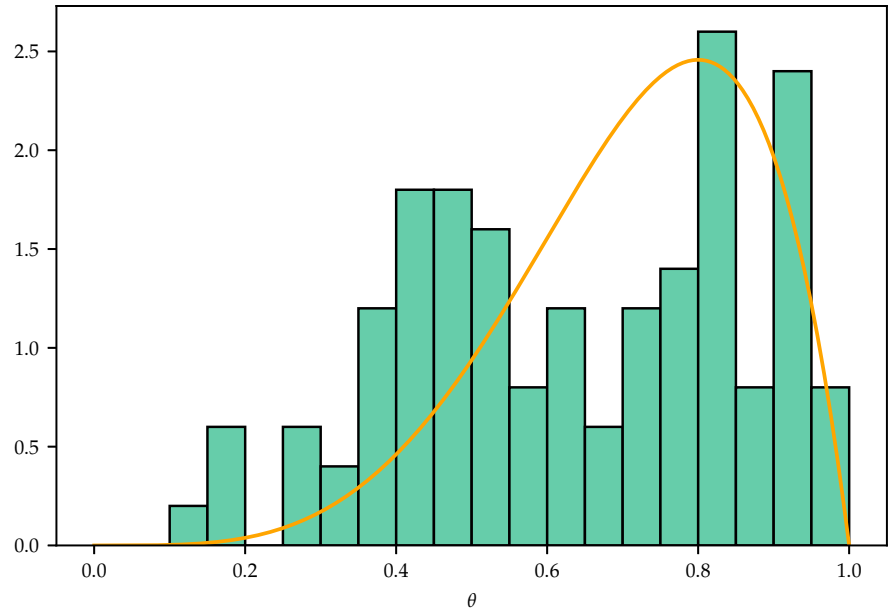
- Standard deviation 0.02:



- Small standard deviation: Random walk takes small steps and doesn't really get off the ground
- And even later, when it has already taken many steps, it moves slowly
- Distribution is not sampled very efficiently

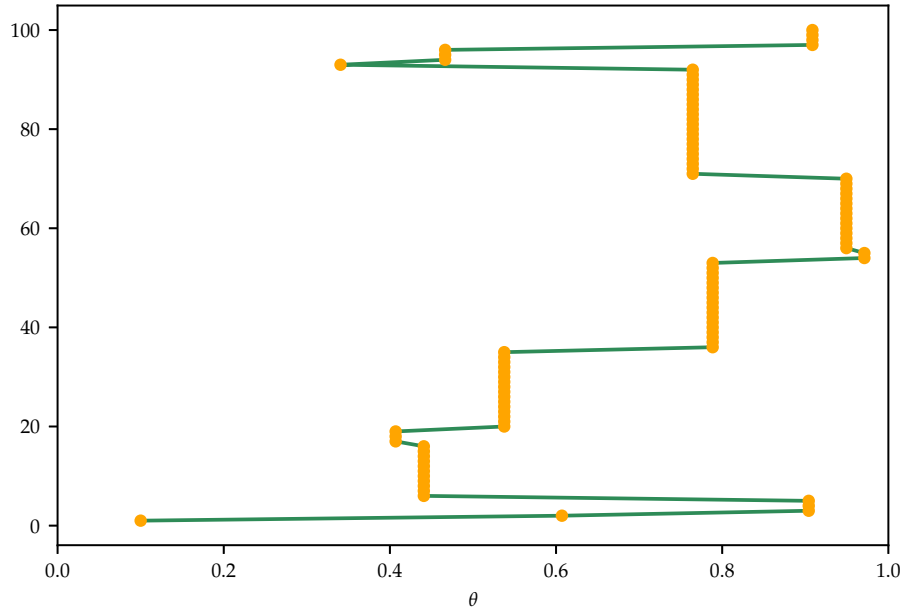
- Standard deviation 0.2:

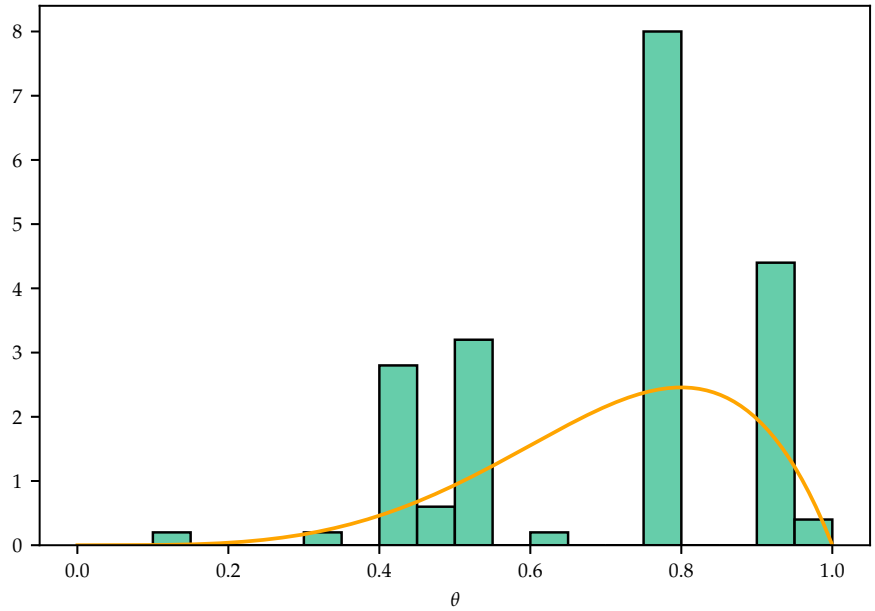




- Distribution is fairly well sampled even in first 100 steps

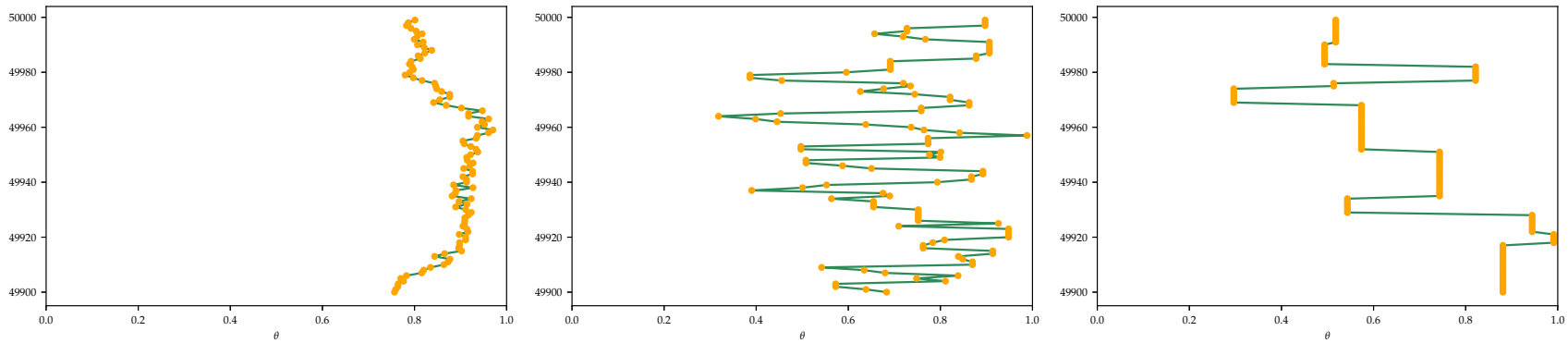
- Standard deviation 2:





- Increase standard deviation to 2: Situation worsens
- Random walk jumps wildly back and forth or stays in same place as suggestions are very often rejected
- Distribution is not well sampled in this case

- 50 000 steps:



- However: All three random walks approximate distribution

- Obviously: Middle random walk with standard deviation 0.2 samples distribution most efficiently

- Question: How to choose value of standard deviation?
- Software packages such as `Stan` analyses input and automatically choose different values for parameters

Other MCMC algorithms

- Metropolis algorithm described is granddaddy of all MCMC algorithms
- More efficient and faster MCMC algorithms have been developed
- Metropolis algorithm, for example, requires that proposal distribution be symmetric (normal distribution in our case)
- W. K. Hastings: Further developed Metropolis algorithm so that proposal distribution no longer needs to be symmetric
- This is Metropolis-Hastings-Algorithmus
- Other improvements: Gibbs algorithm or No-U-Turn algorithm
- More sophisticated than Metropolis-algorithmus, but same principle

Hypothesis Testing in Bayesian Statistics

- Have collected some data and want to answer questions:
  - Is coin fair or not?
  - What is uncertainty for probability for tossing head?
- Bayesian approach to answer these questions
- Coin tossing: What is probability of tossing head?
- Is coin fair?: Is probability 0.5 of tossing head credible?
- One way to ask question: Falls value of interest ($\theta = 0.5$) within range of most credible values of posterior?

- Used Bayes' inference to derive posterior distribution over parameter
- Use posterior distribution to determine most credible values of parameter using the HDI
- If null value is far from most credible values: Discard null value as not credible
- If all credible values are practically equivalent null value: Accept null value
- Formalise now this intuitive decision procedure

Region of Practical Equivalence (ROPE)

- *Region of practical equivalence* (ROPE): Specifies a small range of parameter values that is considered practically equivalent to null value for purposes of a particular application

Example

- Coin is used to determine which team can kick-off in a football match
- Is coin fair?
- Underlying probability of coin being tossed: Reasonably close to 0.5 (*null value*)
- It does not matter whether probability of flip is is 0.473 or 0.528, since these values are practically equivalent to 0.5 for our application
- ROPE for probability of tossing head could be between 0.45 and 0.55

Example

- When evaluating effectiveness of a drug compared to a placebo, only consider using drug if it improves probability of cure by at least 5 percentage points
- Define ROPE as differences in cure probabilities
- ROPE for difference in cure probabilities could have limits of ± 0.05

- Discuss how ROPE limits are set in more detail later

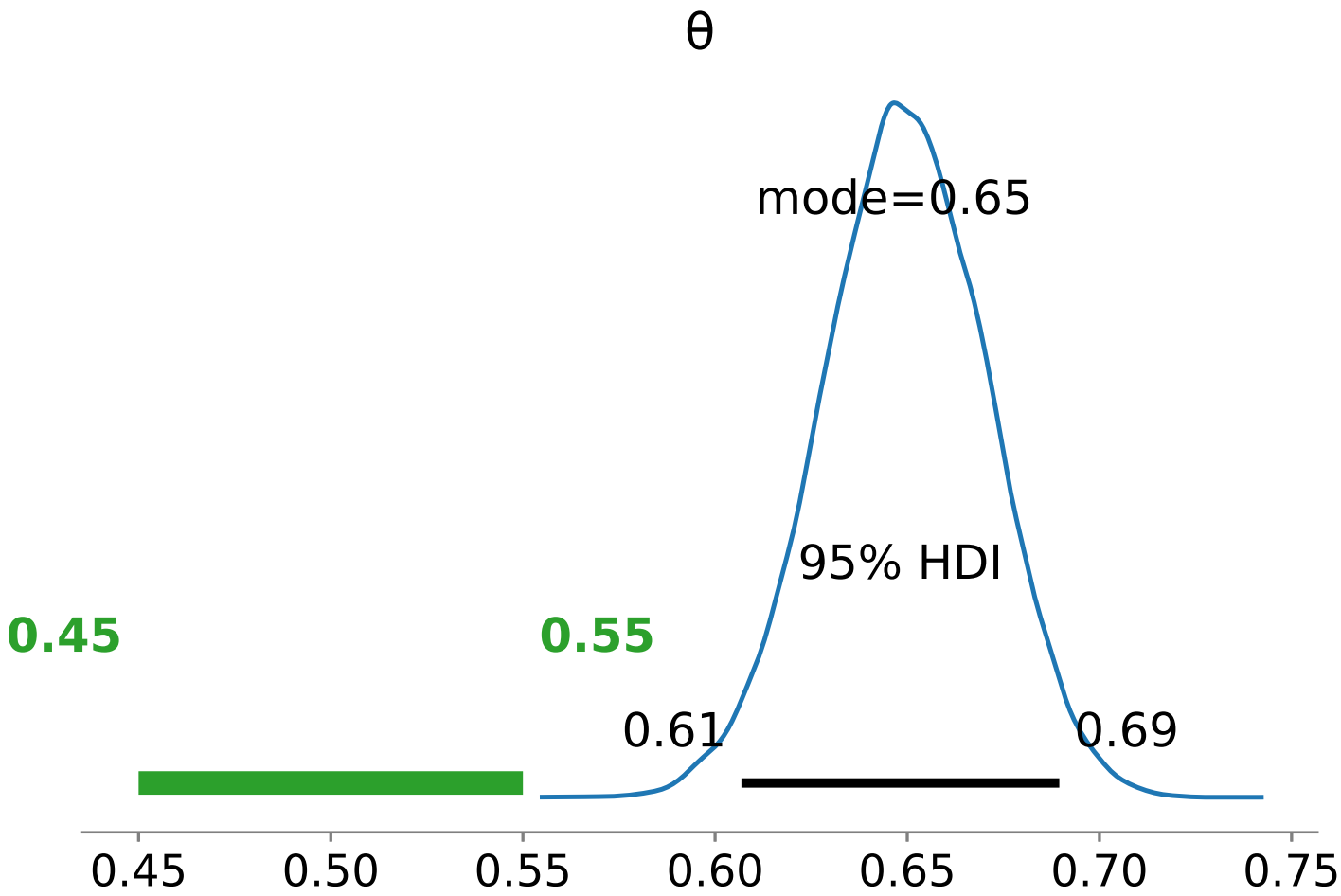
- Once a ROPE is set: Make decision to reject zero value according to following rule:

  A parameter value is declared non-credible or rejected if its total ROPE lies outside 95 %*-density interval (HDI) of posterior distribution of that parameter*

Example

- Want to know if a coin is fair and set ROPE which ranges from 0.45 to 0.55
- Flip coin 500 times and observe head 325 times
- If prior is uniformly distributed: Posterior distribution has 95 %-HDI of 0.608 to 0.691, which is completely outside ROPE (see Figure next slide)
- Therefore: Conclude that null value of 0.5 is discarded for practical purposes

- Figure:



- Idea: Null value with its ROPE only covers a very small percentage of distribution
- Note: In this case it does not matter very much how exactly width of ROPE is chosen
- Could have chosen ROPE from 0.4 to 0.6

- Note: Do *not* reject all values within ROPE when HDI excludes ROPE
- Only the null value is rejected in this case
- ROPE and HDI can overlap in different ways: Different decisions can be made
- In particular: Decision to *accept* a null value:

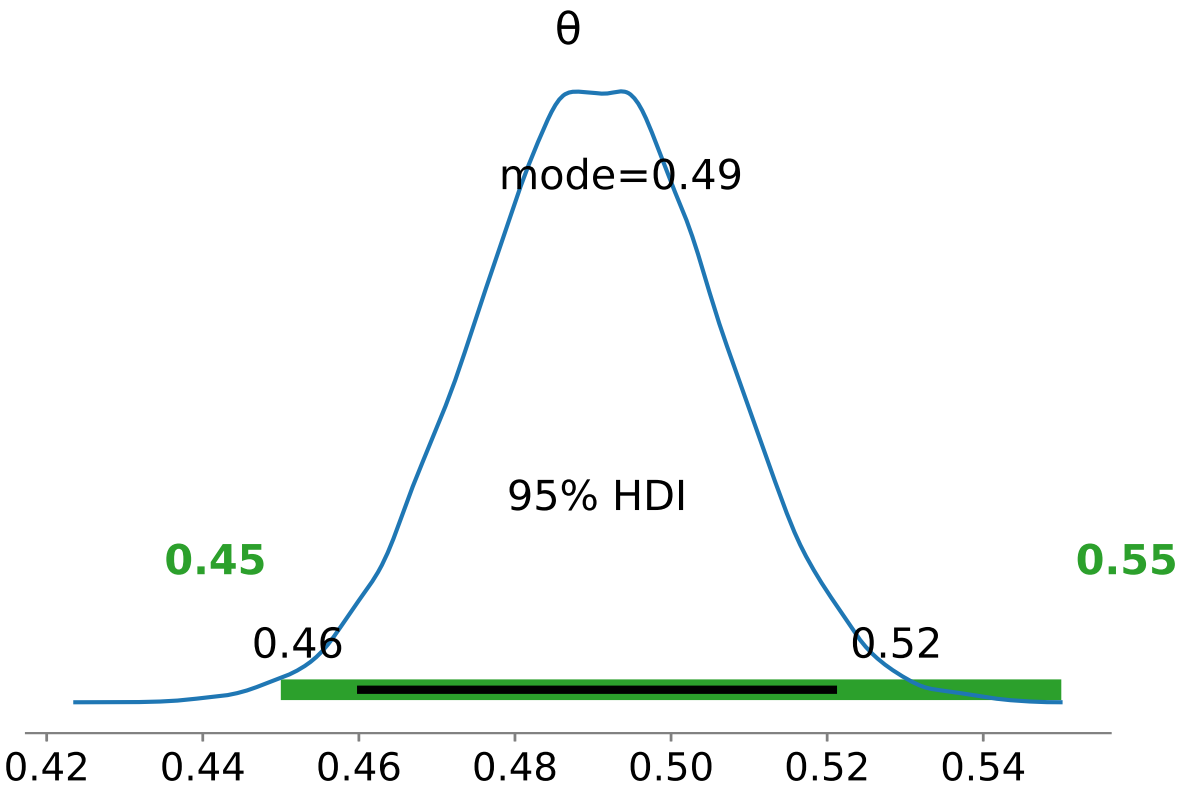
  A parameter value is declared to be accepted for practical purposes if the ROPE of that value is completely contains the 95 %*-HDI of the posterior of that parameter*

- With this decision rule: Zero value of a parameter can only be accepted if precision of parameter estimate is sufficiently large

Example

- Assume a coin is fair and set ROPE from 0.45 to 0.55
- Flip coin 1000 times and observe head 490 times
- If prior is uniformly distributed: Posterior has a 95 %-HDI of 0.459 to 0.521, which falls entirely within ROPE (see Figure next slide)
- Conclude that null value of 0.5 is confirmed for practical purposes, as most credible values are practically equal to null value

- Figure:



- Underlying idea: ROPE contains more than 95 % of most likely values
- In this case, null value (or a practically equivalent value) is also probability tossing head

- How is width of ROPE determined?
- In some fields, such as medicine, experts may be consulted
- These expert opinions can be synthesised into a reasonable consensus on how large an effect needs to be to be useful or important for the application
- However, setting ROPE boundaries is difficult
- Width of ROPE depends not only on state of the art in theory, but also on state of the art in technology, especially available measuring devices

- ROPE boundaries cannot be uniquely "correct" by definition, but are determined by practical goals
- Wider ROPE leads to more acceptance decisions of null value and fewer decisions to reject null value
- If HDI is far from ROPE: Exact ROPE is irrelevant as null value would be rejected for any reasonable ROPE
- If HDI is very narrow and overlaps with null value: HDI can again fall within any reasonable ROPE, again making exact ROPE irrelevant

- However, if HDI is only moderately narrow and close to the null value, the analysis can indicate how much of the posterior value falls within the ROPE
- Important to realise: Any discrete decision to reject or accept a null value does not exhaustively capture our knowledge of parameter value
- Knowledge of parameter value is described by full posterior distribution
- For binary decision: Packed all this rich detail into a single bit of information
- Overall goal of Bayesian analysis is to convey an informative summary of posterior, and where the value of interest lies within that posterior

- Specifying boundaries of HDI is more informative than specifying a rejection/acceptance decision
- By specifying HDI and other summary information about posterior, different readers can apply different ROPEs to decide for themselves whether parameter is practically equivalent to null value
- Decision procedure is separate from Bayesian inference
- Bayesian part of analysis is derivation of posterior distribution
- Decision procedure uses posterior distribution but does not itself use Bayes' rule

Stan

- Stan is one software to perform MCMC
- R package rstan falls back on Stan

```
install.packages("rstan", repos = "https://cloud.r-project.org/", dependencies = TRUE)
```

  (takes a while)
- Additionally:

```
install.packages("bayestestR", "latex2exp")
```