Bayesian inference
& Markov-Chain Monte Carlo

Everyone uses Bayesian Inference
– whether they know it or not

Person-of-interest A

(visits room)

Person-of-interest B

(visits room & knows wallet/purse has money)

Person-of-interest A

(visits room)

Person-of-interest B

(visits room & knows wallet/purse has money)

10%

90%

Which of the two hypotheses is most consistent with
your new data

Person-of-interest A

(visits room)

Person-of-interest B

(visits room & knows wallet/purse has money)

10%

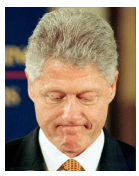
90%

Person-of-interest A

(visits room)

Person-of-interest B

(visits room & knows wallet/purse has money)



10 % (?)

90 % (?)

Given prior knowledge of each person's frequency of stealing,
which of the two hypotheses is more likely to be TRUE?

Under a Bayesian framework, an hypothesis is assessed on

(1) Fit with new data (conventional stats)
(2) Fit with the current world view ("priors")

That's how everyone operates …

We're more likely to accept ideas which (1) have new supporting evidence and (2) fit our existing world-view.

If something really contradicts our biases, then we will demand to see lots of compelling new evidence before we change our mind.

Darwin was a Bayesian.

Under a Bayesian framework, an hypothesis is assessed on

(1) Fit with new data (conventional stats)
(2) Fit with the current world view ("priors")

Priors are controversial (improve accuracy or reinforce delusions, e.g. creationists!)

- can test just how influential they are (e.g. vary them systematically)
- can make some/all very conservative (e.g. uninformative or "flat" priors)

Bayes' Theorem: 3 components

Posterior Probability of hypothesis

Prior Probability of hypothesis

Likelihood (how well hypothesis fits new data)

$$f(\theta \mid X) = \frac{f(\theta) f(X \mid \theta)}{\int f(\theta) f(X \mid \theta) d\theta}$$

Normalizing constant (total prob of data over all Param values)

Bayes' Theorem: 3 components

Posterior Probability of hypothesis

Prior Probability of hypothesis

Likelihood (how well hypothesis fits new data)

$$f(\theta \mid X) = \frac{f(\theta) f(X \mid \theta)}{\int f(\theta) f(X \mid \theta) d\theta}$$

All variables (priors and new data) are considered together

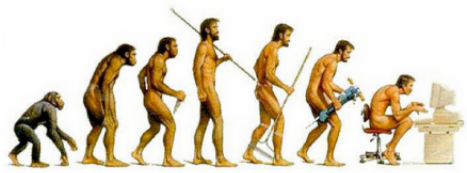
Everything is co-estimated, best global solution (phyl, dates, trait evol, biog)

Computationally very intensive (MCMC methods, not exact solutions)

Not feasible in many evolutionary problems until recent computational advances

The methods used in phylogenetics have closely tracked available computing power

But different disciplines have evolved at different rates

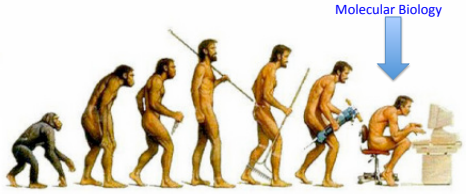


Distance e.g. UPGMA (1950-60s) | Parsimony (1970-1980s) | Likelihood (1990s) | Bayesian inference (2000s)

The methods used in phylogenetics have closely tracked available computing power

But different disciplines have evolved at different rates

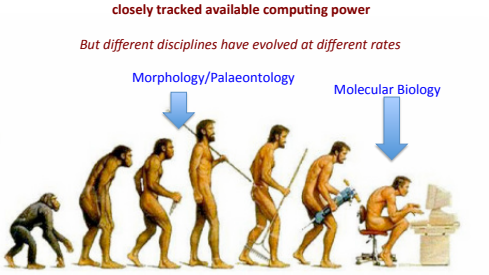
Molecular Biology



Clustering (phenetics) (1950-60s) | Parsimony (1970-1980s) | Likelihood (1990s) | Bayesian inference (2000s)

The methods used in phylogenetics have closely tracked available computing power

But different disciplines have evolved at different rates



Clustering (phenetics) (1950-60s) | Parsimony (1970-1980s) | Likelihood (1990s) | Bayesian inference (2000s)

Markov-Chain Monte Carlo

- **What is it?**
- How should it work in an ideal situation?
- How can you check that it is working (in your analysis)?

Markov-Chain Monte Carlo

- Key tool used in Bayesian Inference
- An efficient technique for sampling from a target distribution you are trying to infer (e.g. relative probabilities of different tree topologies)
- Doesn't give you a single best solution, but a cloud of possible solutions weighted by their probabilities

Maximum Likelihood vs Bayesian Inference

- There's a common view that ML / BI methods differ in that
  - ML gives a single best tree (for those who want a single best solution)
  - BI gives a cloud of trees via MCMC (for those interested in exploring clouds of possibilities)
- This distinction is a bit of a fallacy.

Maximum Likelihood vs Bayesian Inference

- Can compute a cloud of possibilities in ML (e.g. examine all tree topologies and get their likelihoods, bootstrapping etc)
- You can find a single best Bayesian solution (by evaluating the exact probability of every possible solution).

From Wikipedia (observation: it rained on Monday calculate: probability of rain/no rain on Sunday)

The equation used is:

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)},$$

In our case, "Raining on Sunday" is event A, and "Raining on Monday" is event B.

- $P(B|A) = 0.10$ = Probability of it raining on Monday, if it rained on Sunday.
- $P(A) = 0.40$ = Probability of Raining on Sunday.
- $P(B) = 0.52$ = Probability of Raining on Monday.

So, to calculate the probability it rained on Sunday, given that it rained on Monday:

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)},$$

or:

$$P(A|B) = \frac{0.10 * 0.40}{0.52} = .0769$$

In other words, if it rained on Monday, there's a 7.69% chance it rained on Sunday.

Bayesian Inference & MCMC

- Use of MCMC approaches in BI isn't a philosophical decision ("*Bayesians are more interested in clouds of possibilities rather than single best solutions*" – not always true).
- It's a practical decision. Bayesian inference is always more complex than likelihood (you still do everything you do in likelihood, but add a swag of priors).

Posterior Probability of hypothesis | Prior Probability of hypothesis | Likelihood (how well hypothesis fits new data)

$$f(\theta \mid X) = \frac{f(\theta) f(X \mid \theta)}{\int f(\theta) f(X \mid \theta) d\theta}$$

Normalizing constant

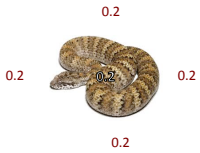
- For complex problems, you often **can't** evaluate every single solution and find the best one. MCMC approaches allow you to sample a bunch of solutions and see which types of solutions tend to be "better".

Bayesian Inference & MCMC

- MCMC might not ever sample the "best" solution (single best tree topology, branch lengths and rate patterns).
- But you could get a good estimate of what this best solution would look like, if you have enough nearby samples.
- Use of MCMC is a **practical** or **empirical** choice ("it works on our data"), not a **theoretical** or **philosophical** one ("we prefer clouds of solutions to single ones").

Markov Chain Monte Carlo - basics

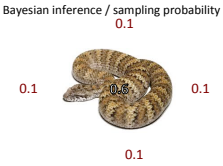
- A series of variables (integers, numbers, vectors, tree topologies) which is:
  - **Linear** (Chain)
  - **Randomly-generated** (Monte Carlo)
  - **Memory-less**, the next variable in the chain depends only on the current variable (Andrey Markov)



The series of positions (grid-coordinates) occupied by this death adder would be a Markov-Chain (1,2) (1,3) (1,3) (2,3) (2,2) etc…

Markov Chain Monte Carlo - basics

- BUT if the probability of staying/moving is dependent on how long the snake has **already** been stationary, we would **not** have a Markov Chain.
- The chain (snake) has **memory** beyond the last/current state
- Biologically realistic? Think of yourselves.
- But not useful for Bayesian inference / sampling probability space



Markov-Chain Monte Carlo

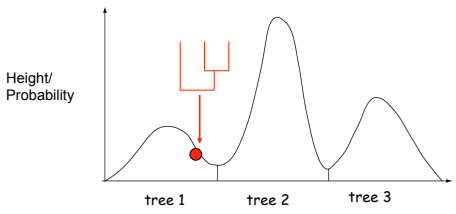
- What is it?
- **How does it work in phylogenetics?**
- How can you check that it is working (in your analysis)?

Bayesian Phylogenetis using MCMC

- Often used to generate the pool of plausible trees in Bayesian phylogenetics
- Each step in the chain is a tree and all associated parameters (topology, branch lengths, evolutionary rates, divergence dates, position and magnitude of rate changes, GTR matrix, Gamma parameter).
- Each new "step" in the chain is made by perturbing 1 (or 2) of these parameters and either accepting or rejecting this new value (proposal).

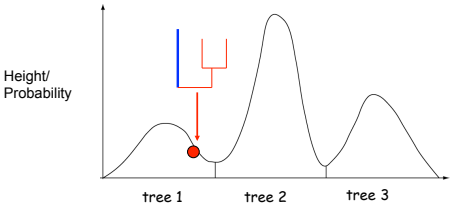
Bayesian Markov-Chain Monte-Carlo (based on FRs slides)

1. Start at an arbitrary point (random tree & random parameter values)
2. Make a small random move
3. Calculate height/probability ratio (*r*) of new state to old state:
   1. r > 1 -> new state accepted
   2. r < 1 -> new state accepted with probability *r*. If new state not accepted, stay in the old state
4. Return to step 2



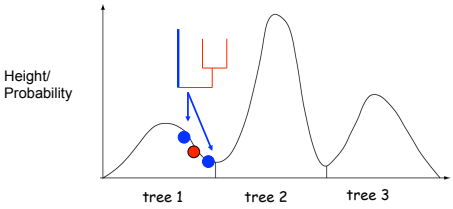
Bayesian Markov-Chain Monte Carlo

1. Start at an arbitrary point
2. Make a small random move (e.g. change a branch length)
3. Calculate relative likelihoods (*r*) of new state to old state:
   1. r > 1 -> new state accepted
   2. r < 1 -> new state accepted with probability *r*. If new state not accepted, stay in the old state
4. Return to step 2



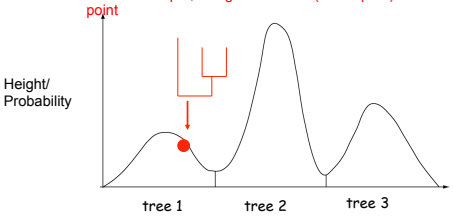
Bayesian Markov-Chain Monte Carlo

1. Start at an arbitrary point
2. Make a small random move (e.g. change a branch length)
3. Calculate relative likelihoods (*r*) of new state to old state:
   1. r > 1 -> new state accepted
   2. r < 1 -> new state accepted with probability *r*. If new state not accepted, stay in the old state
4. Return to step 2



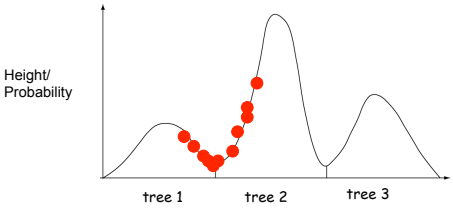
Bayesian Markov-Chain Monte Carlo

1. Start at an arbitrary point
2. Make a small random move (e.g. change a branch length)
3. Calculate relative likelihoods (*r*) of new state to old state:
   1. r > 1 -> new state accepted
   2. r < 1 -> new state accepted with probability *r*. If new state not accepted, stay in the old state
4. Return to step 2, using new values (if accepted) as starting point



Bayesian Markov-Chain Monte Carlo

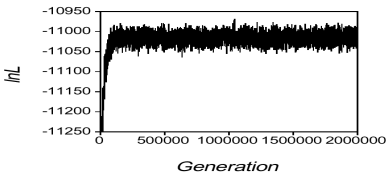
1. Start at an arbitrary point (random tree & random parameter values)
2. Make a small random move
3. Calculate height/probability ratio (*r*) of new state to old state:
   1. r > 1 -> new state accepted
   2. r < 1 -> new state accepted with probability *r*. If new state not accepted, stay in the old state
4. Return to step 2



Bayesian Markov-Chain Monte Carlo

Initially, your likelihoods/probabilities will increase rapidly (your random tree will have low likelihood, which can be improved upon with random moves)

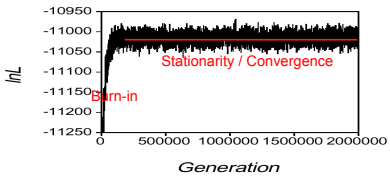
Eventually, your likelihoods/probabilities will hit a plateau (once sampled trees are very good, most changes will not lead to improved likelihoods/probability and will be rejected) –



Bayesian - Markov-Chain Monte Carlo

Initially, your likelihoods/probabilities will increase rapidly (your random tree will have low likelihood, which can be improved upon with random moves) - Burn-in

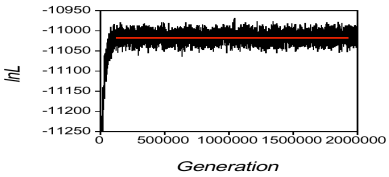
Eventually, your likelihoods/probabilities will hit a plateau (once sampled trees are very good, most changes will not lead to improved likelihoods/probability and will be rejected) – Stationarity / Convergence



Bayesian - Markov-Chain Monte Carlo

At stationarity, MCMC chain will sample trees in proportion to their true posterior probability (ie the target distribution)!

Posterior Probability ≈ Prior Probability x Likelihood

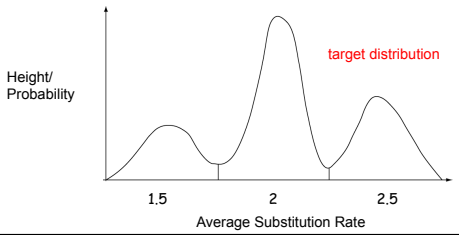


Markov-Chain Monte Carlo

- What is it?
- How does it work in phylogenetics?
- **How can you check that it is working (in your analysis)?**

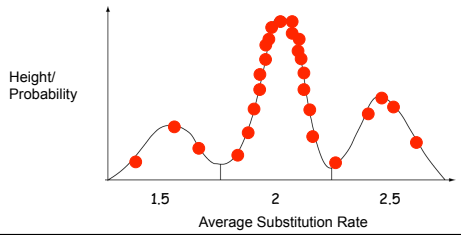
Achieving Stationarity (Convergence)

- At stationarity, MCMC chain will sample trees/parameters in proportion to their true posterior probability (ie the target distribution)



Achieving Stationarity (Convergence)

- At stationarity, MCMC chain will sample trees/parameters in proportion to their true posterior probability (ie the target distribution)

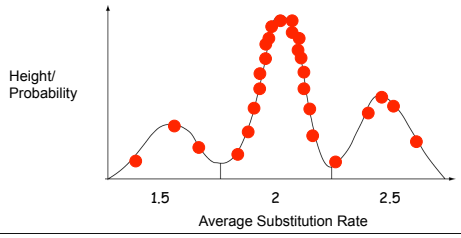


Achieving Stationarity (Convergence)

- Correct proposal moves for each parameter
- Correct weights for different parameter

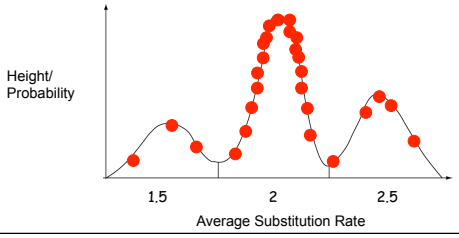
Achieving Stationarity (Convergence)

- The MCMC proposals for each parameter should be "just right"
- Not too timid (e.g. +/- 0.000001).
- Not too bold (e.g. window +/- 1000)
- In this case, window of +/- 0.3 would be good



Achieving Stationarity (Convergence)

- Can adjust the proposal boldness (BEAST / MrBayes / RevBayes)
- Autotuning – the program will try various proposal "sizes" and select what it thinks is good.



Achieving Stationarity (Convergence)

- Correct proposal moves for each parameter
- Correct weights for different parameter

Achieving Stationarity (Convergence)

- The MCMC needs to adequately sample ALL parameters (e.g. tree topology, branch lengths, overall rates, branch-specific rates, substitution matrix, invariant sites, gamma parameter).
- Every time a new state is proposed, only 1 parameter (or 2 related parameters) is altered.

```
<upDownOperator scaleFactor="0.75" weight="3">
    <up>
        <compoundParameter idref="allRates"/>
    </up>
    <down>
        <parameter idref="treeModel.allInternalNodeHeights"/>
    </down>
</upDownOperator>
```

- The MCMC chain should spend more time exploring "difficult" parameters (often topology etc), and less time exploring "easy" parameters (often Ti/Tv ratio etc).

Achieving Stationarity (Convergence)

- The MCMC needs to adequately sample ALL parameters (e.g. tree topology, branch lengths, overall rates, branch-specific rates, substitution matrix, invariant sites, gamma parameter).
- Every time a new state is proposed, only 1 parameter is altered.
- The MCMC chain should spend more time exploring "difficult" parameters (often topology etc), and less time exploring "easy" parameters (often Ti/Tv ratio etc).
- There is NO magic algorithm which allows programs to weight MCMC searches correctly – you often need to do this yourself.
- BEAST/MrBayes/RevBayes permits you to adjust relative weights of MCMC searches

Achieving Stationarity (Convergence)

- Need correct proposal "boldness" of each parameter
- Need the right amount of time on each parameter


```
<upDownOperator scaleFactor="0.75" weight="3">
    <up>
        <compoundParameter idref="allRates"/>
    </up>
    <down>
        <parameter idref="treeModel.allInternalNodeHeights"/>
    </down>
</upDownOperator>
```


Achieving Stationarity (Convergence)

- Need correct proposal "boldness" of each parameter
- Need the right amount of time on each parameter

```
25 -- Move       = Multiplier(Alpha{1})
      Type       = Multiplier
      Parameter  = Alpha{1} [param. 14] (Shape of scaled gamma distribution of site rates)
      Tuningparam = lambda (Multiplier tuning parameter)
          lambda = 0.500
      Targetrate  = 0.250
      Rel. prob.  = 10.0

26 -- Move       = Multiplier(Alpha{2})
      Type       = Multiplier
      Parameter  = Alpha{2} [param. 15] (Shape of scaled gamma distribution of site rates)
      Tuningparam = lambda (Multiplier tuning parameter)
          lambda = 0.811
      Targetrate  = 0.250
      Rel. prob.  = 1.0

27 -- Move       = Multiplier(Alpha{3})
      Type       = Multiplier
      Parameter  = Alpha{3} [param. 16] (Shape of scaled gamma distribution of site rates)
      Tuningparam = lambda (Multiplier tuning parameter)
          lambda = 0.811
      Targetrate  = 0.250
      Rel. prob.  = 1.0

28 -- Move       = Multiplier(Alpha{4})
      Type       = Multiplier
      Parameter  = Alpha{4} [param. 17] (Shape of scaled gamma distribution of site rates)
      Tuningparam = lambda (Multiplier tuning parameter)
          lambda = 0.811
      Targetrate  = 0.250
      Rel. prob.  = 1.0
```

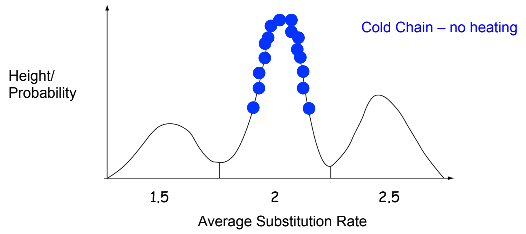
showmoves

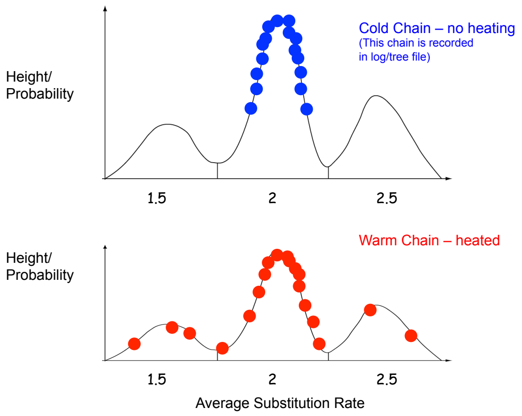
```
The MCMC sampler will use the following moves:
   With prob.  Chain will use move
      0.47 %   Dirichlet(Revmat{2})
      0.47 %   Slider(Revmat{2})
      0.47 %   Dirichlet(Revmat{3})
      0.47 %   Slider(Revmat{3})
      0.47 %   Dirichlet(Revmat{4})
      0.47 %   Slider(Revmat{4})
      0.47 %   Dirichlet(Revmat{5})
      0.47 %   Slider(Revmat{5})
      0.47 %   Dirichlet(Revmat{6})
      0.47 %   Slider(Revmat{6})
      0.47 %   Dirichlet(Revmat{7})
      0.47 %   Slider(Revmat{7})
      0.47 %   Dirichlet(Pi{2})
      0.47 %   Slider(Pi{2})
      0.47 %   Dirichlet(Pi{3})
      0.47 %   Slider(Pi{3})
      0.47 %   Dirichlet(Pi{4})
      0.47 %   Slider(Pi{4})
      0.47 %   Dirichlet(Pi{5})
      0.47 %   Slider(Pi{5})
      0.47 %   Dirichlet(Pi{6})
      0.47 %   Slider(Pi{6})
      0.47 %   Dirichlet(Pi{7})
      0.47 %   Slider(Pi{7})
      9.48 %   Multiplier(Alpha{1})
      0.95 %   Multiplier(Alpha{2})
      0.95 %   Multiplier(Alpha{3})
      0.95 %   Multiplier(Alpha{4})
      0.95 %   Multiplier(Alpha{5})
      0.95 %   Multiplier(Alpha{6})
      0.95 %   Multiplier(Alpha{7})
      0.95 %   Slider(Pinvar{2})
      0.95 %   Slider(Pinvar{3})
      0.95 %   Slider(Pinvar{4})
      0.95 %   Slider(Pinvar{5})
      0.95 %   Slider(Pinvar{6})
      0.95 %   Slider(Pinvar{7})
      0.71 %   Dirichlet(Ratemultiplier{all})
      0.71 %   Slider(Ratemultiplier{all})
     11.37 %   ExtSPR(Tau{all},V{all})
     11.37 %   ExtTBR(Tau{all},V{all})
     11.37 %   NNI(Tau{all},V{all})
     11.37 %   ParsSPR(Tau{all},V{all})
     14.22 %   Multiplier(V{all})
      4.74 %   Nodeslider(V{all})
      1.90 %   TLMultiplier(V{all})
```

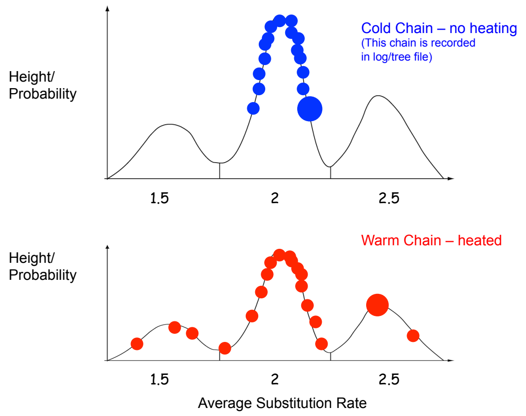
In MrBayes Standard Output

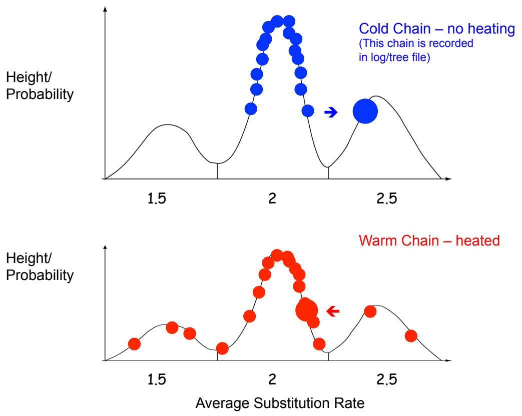
Achieving Stationarity (Convergence): Heating

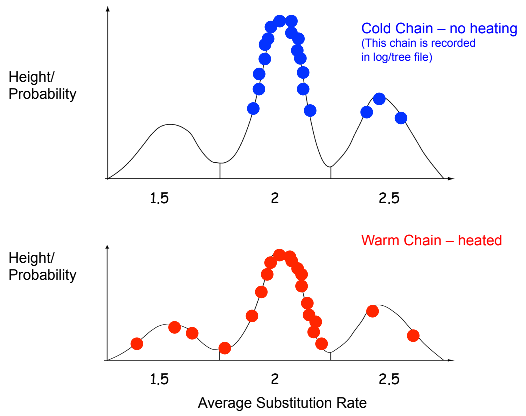
- Use of multiple MCMC chains simultaneously increases the speed at which a Bayesian analysis samples the target distribution. Metropolis-Coupled MC, or MC3.
- The probability surface of some chains is "flattened" (heating).



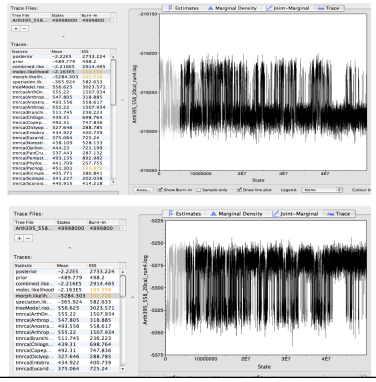




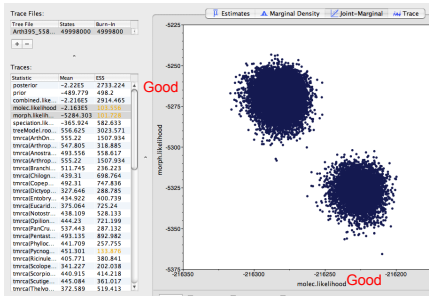




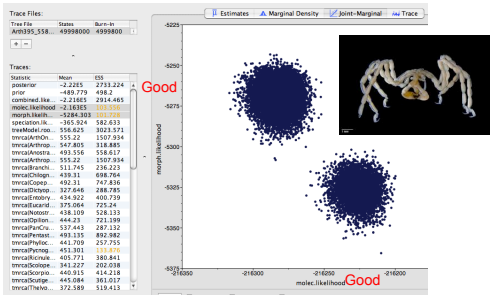
Very useful when you have complex datasets
With lots of isolated optima (e.g. morphology + molecules)



Very useful when you have complex datasets
With lots of isolated optima (e.g. morphology + molecules)

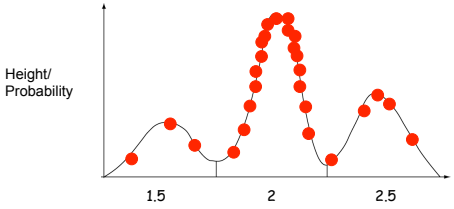


Very useful when you have complex datasets
With lots of isolated optima (e.g. morphology + molecules)



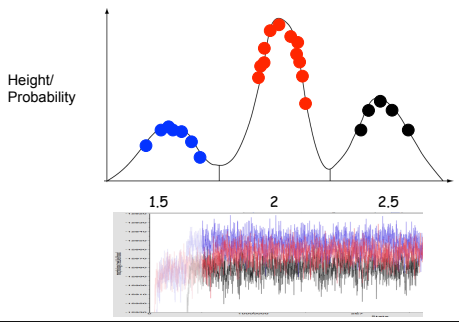
Checking for Stationarity (Convergence)

- Bayesian analyses need to sample the entire target distribution
- Every run should be sampling all the same optima at the same frequencies



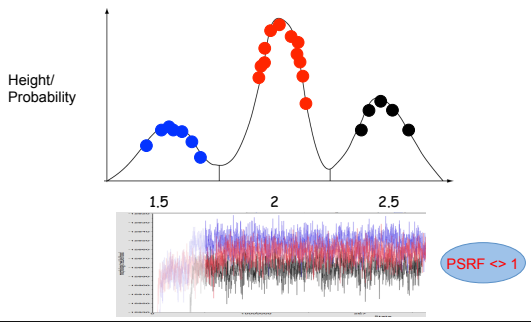
Checking for Stationarity (Convergence)

- But if the MCMC chains are not adequate, different runs can sample different optima



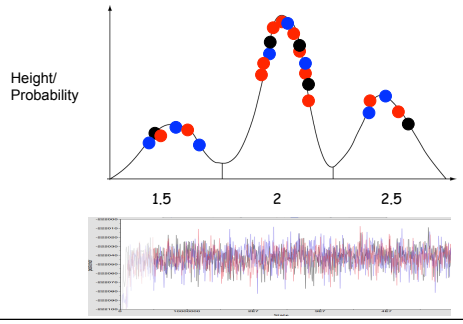
Checking for Stationarity (Convergence)

- But if the MCMC chains are not adequate, different runs can sample different optima



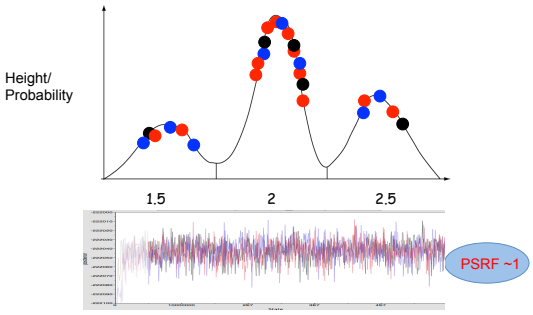
Checking for Stationarity (Convergence)

- But if the MCMC chains are not adequate, different runs can sample different optima



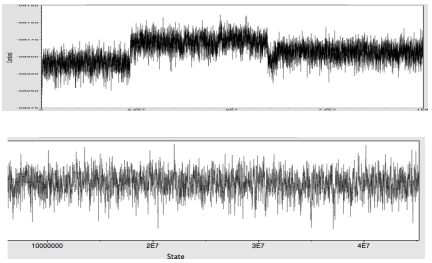
Checking for Stationarity (Convergence)

- But if the MCMC chains are not adequate, different runs can sample different optima



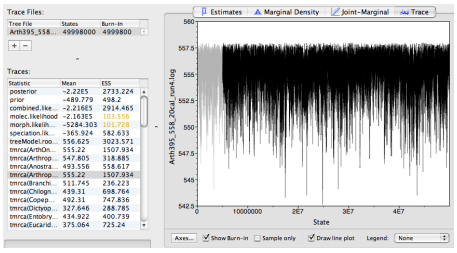
Checking for Stationarity (Convergence)

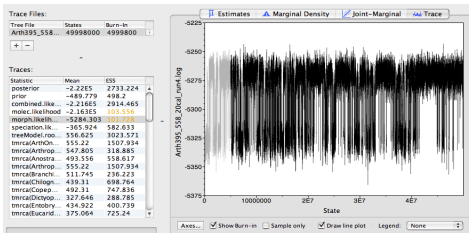
- Effective Sample Size or ESS (Tracer) is how *many independent samples* you have in the concatenated runs
- Chain length / "autocorrelation interval"
- **Should be >100**



3 runs
(not converged)
ESS ~3

3 runs
(converged)
ESS >500



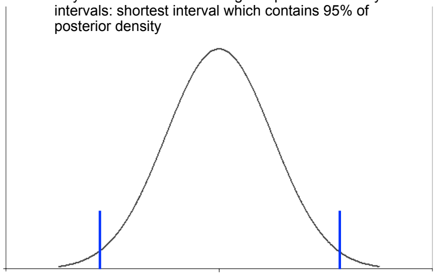


Checking for Stationarity (Convergence)

- Need to check for convergence for numerical parameters (Tracer, PSRF)
- Need to check for convergence of topological parameters (clades, branch lengths etc). Split freqs/AWTY. Simon T.
- **Absolutely vital in Bayesian analyses.**
  - Should do >4 independent runs of every analysis and make sure all parameters have converged
  - My guess is that >50% of published Bayesian analyses have not properly ensured or tested for convergence

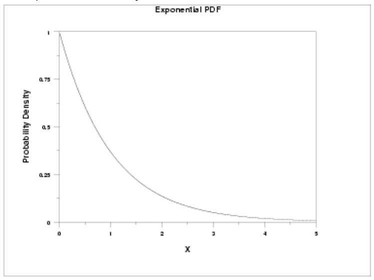
Confidence intervals on Bayesian Posterior Samples

- Frequentists like 95% confidence intervals: range between the lower 2.5 percentile and the upper 2.5 percentile.
- Bayesians often use 95% highest posterior density intervals: shortest interval which contains 95% of posterior density



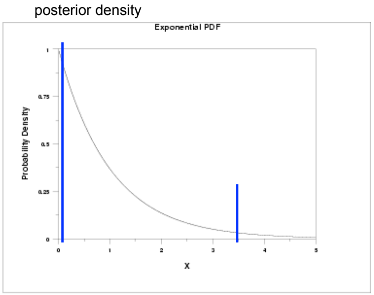
Confidence intervals on Bayesian Posterior Samples

- Frequentists like 95% confidence intervals: range between the lower 2.5 percentile and the upper 2.5 percentile.
- Bayesians often use 95% highest posterior density intervals: shortest interval which contains 95% of posterior density



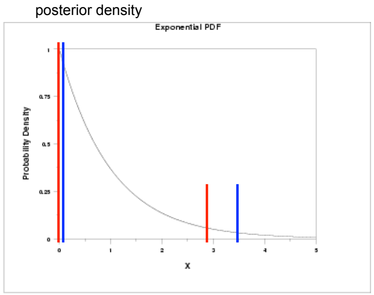
Confidence intervals on Bayesian Posterior Samples

- Frequentists like 95% confidence intervals: range between the lower 2.5 percentile and the upper 2.5 percentile.
- Bayesians often use 95% highest posterior density intervals: shortest interval which contains 95% of posterior density



Confidence intervals on Bayesian Posterior Samples

- Frequentists like 95% confidence intervals: range between the lower 2.5 percentile and the upper 2.5 percentile.
- Bayesians often use 95% highest posterior density intervals: shortest interval which contains 95% of posterior density



Confidence intervals on Bayesian Posterior Samples

- Confidence intervals are NOT the same as highest posterior density intervals
- Differences are greatest in skewed distributions