Hierarchical Bayesian Modeling with MCMC

Kenny Shirley

Data Mining, Columbia University

November 17, 2011

Outline

1. Model 1: A simple football example (20 minutes)
2. Model 2: A larger, more realistic football example (5 minutes)
3. MCMC basics: The Metropolis-Hastings Algorithm (30 minutes)
4. Output from Model 2 and post-processing MCMC results (10 minutes)
5. A prediction for tonight's football game!

1. A Simple Example: Football Odds

Question: What is the probability that each of these NFL football teams will win their home game this weekend?

Let's look at each of their *home* win-loss records for the past year (about 8-9 games per team):

game	team	Wins	Losses
1	DEN	2	5
2	SFN	6	2
3	MIA	1	7
4	DET	4	4
5	BAL	7	1
6	WAS	2	5
7	CLE	3	5
8	MIN	2	6
9	NYG	5	2
10	CHI	7	3
11	STL	2	5
12	GBP	8	0
13	ATL	4	4
14	NEP	7	2

1. Two solutions

Two solutions are:

1. Assume each team is independent, i.e. do no pooling.
   - Use a binomial model and compute the MLE for each team:

$$Y_t \sim \text{Binom}(n_t, p_t)$$

$$\hat{p}_t = \frac{Y_t}{n_t},$$

     for teams $t = 1, ..., 14$.
   - For example, P(DEN wins) $= 2/7 = 28\%$, because DEN has 2 win and 5 losses in the data.
2. Assume the teams are identical, i.e. do complete pooling.
   - In all 111 home games these teams have played in the past year, the home team has won 60 times and lost 51 times, for a 54% win probability.

1. No Pooling vs. Complete Pooling

Do we believe either solution?

game	team	W	L	no.pooling	complete.pooling
1	DEN	2	5	0.29	0.54
2	SFN	6	2	0.75	0.54
3	MIA	1	7	0.12	0.54
4	DET	4	4	0.50	0.54
5	BAL	7	1	0.88	0.54
6	WAS	2	5	0.29	0.54
7	CLE	3	5	0.38	0.54
8	MIN	2	6	0.25	0.54
9	NYG	5	2	0.71	0.54
10	CHI	7	3	0.70	0.54
11	STL	2	5	0.29	0.54
12	GBP	8	0	1.00	0.54
13	ATL	4	4	0.50	0.54
14	NEP	7	2	0.78	0.54

1. No Pooling vs. Complete Pooling

Do we believe either solution?

game	team	W	L	no.pooling	complete.pooling
1	DEN	2	5	0.29	0.54
2	SFN	6	2	0.75	0.54
3	MIA	1	7	0.12	0.54
4	DET	4	4	0.50	0.54
5	BAL	7	1	0.88	0.54
6	WAS	2	5	0.29	0.54
7	CLE	3	5	0.38	0.54
8	MIN	2	6	0.25	0.54
9	NYG	5	2	0.71	0.54
10	CHI	7	3	0.70	0.54
11	STL	2	5	0.29	0.54
12	GBP	8	0	1	0.54
13	ATL	4	4	0.50	0.54
14	NEP	7	2	0.78	0.54

1. A hierarchical model does *partial pooling*

- In the model $Y_t \sim \text{Binom}(n_t, p_t)$, a hierarchical model treats $(p_1, p_2, ..., p_t)$ as if they are random draws from a larger population.
- In the frequentist statistical framework, these are called random effects.
- In the Bayesian statistical framework, the distribution of the p's is called the prior distribution.
- The term hierarchical (or multilevel) refers to (1) the levels of the written model, and (2) the different levels of variability inherent in the data. [In a sense they are equivalent, since estimating different levels of variation is the purpose of the model!]

$$Y_t \sim \text{Binom}(n_t, p_t)$$

$$p_t \sim f(p_t)$$

1. Basic Bayes

- Basic idea of Bayesian statistics:
  - The likelihood is the model for the observed data: $\mathrm{p}(Y \mid \theta)$.
  - The prior is the probability model for the unknown parameters in the likelihood: $p(\theta)$.
  - We want to know the posterior distribution, the probability of the unknown parameters conditional on the observed data.

$$\begin{aligned} p(\theta \mid Y) &= \frac{p(Y \mid \theta)p(\theta)}{p(Y)} \\ &\propto p(Y \mid \theta)p(\theta), \end{aligned}$$

  by a simple application of Bayes rule.

- Not all hierarchical models are Bayesian, but today's lecture will focus on Bayesian models. Their use has exploded in the past 20 years due to advances in computational speed.

1. Back to football: priors for binomial proportions

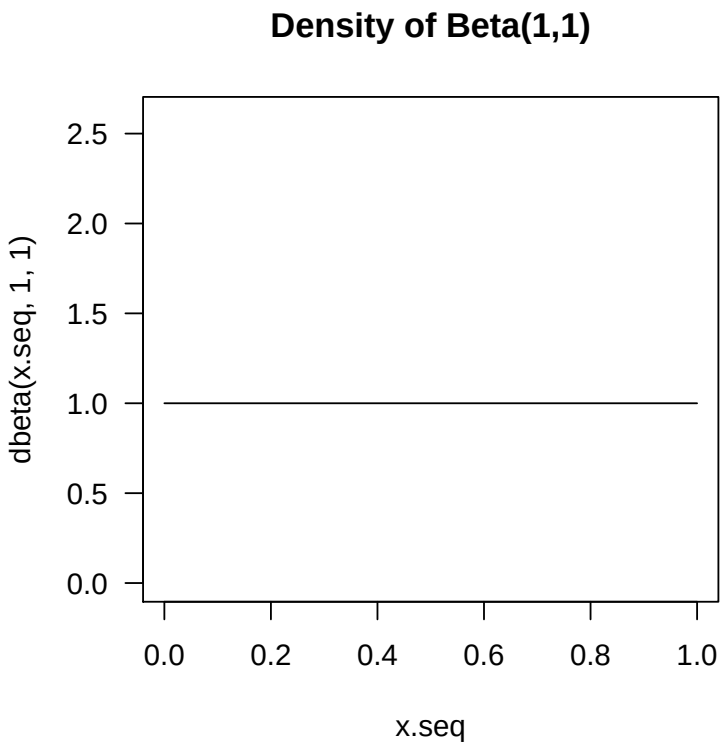
- Since p_t is a probability between 0 and 1, the Beta distribution is a good candidate for a prior for p_t.
- Another reason the Beta distribution is a good prior here is that it is the conjugate prior for a binomial probability.
- In the case of a binomial likelihood and a beta prior, the posterior distribution can be computed in closed form, and it is also a beta distribution:

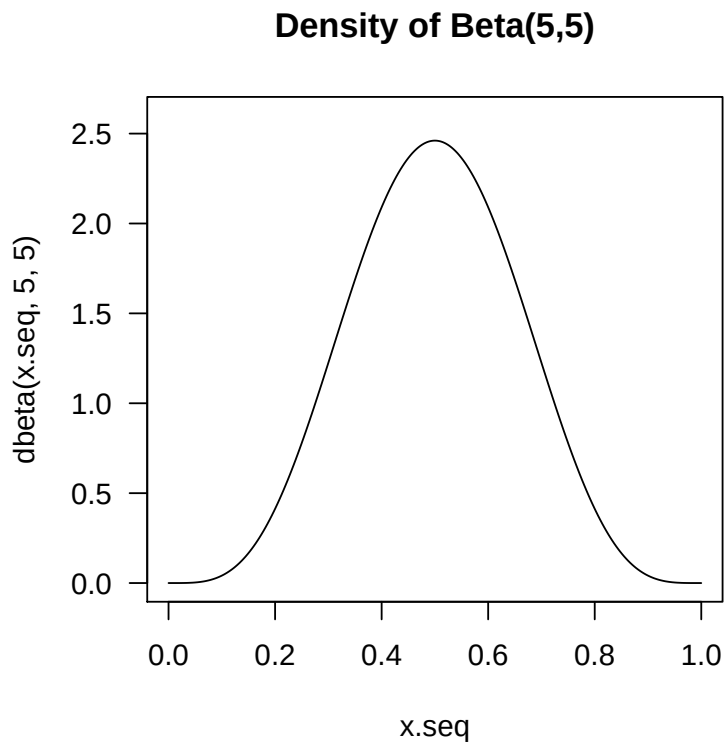
$$\begin{aligned} Y_t &\sim \text{Binom}(n_t, p_t) \\ p_t &\sim \text{Beta}(\alpha, \beta) \\ p_t \mid \boldsymbol{Y} &\sim \text{Beta}(\alpha + Y_t, \beta + (n - Y_t)). \end{aligned}$$

  (easy to show with a bit of algebra... try it)
- A conjugate pair is a (likelihood, prior) pairing such that the posterior distribution is of the same family as the prior.

1. Priors for binomial proportions

- Which Beta(α, β) prior should we use? In other words, what values of α and β should we choose? These variables are known as hyperparameters in a Bayesian model.



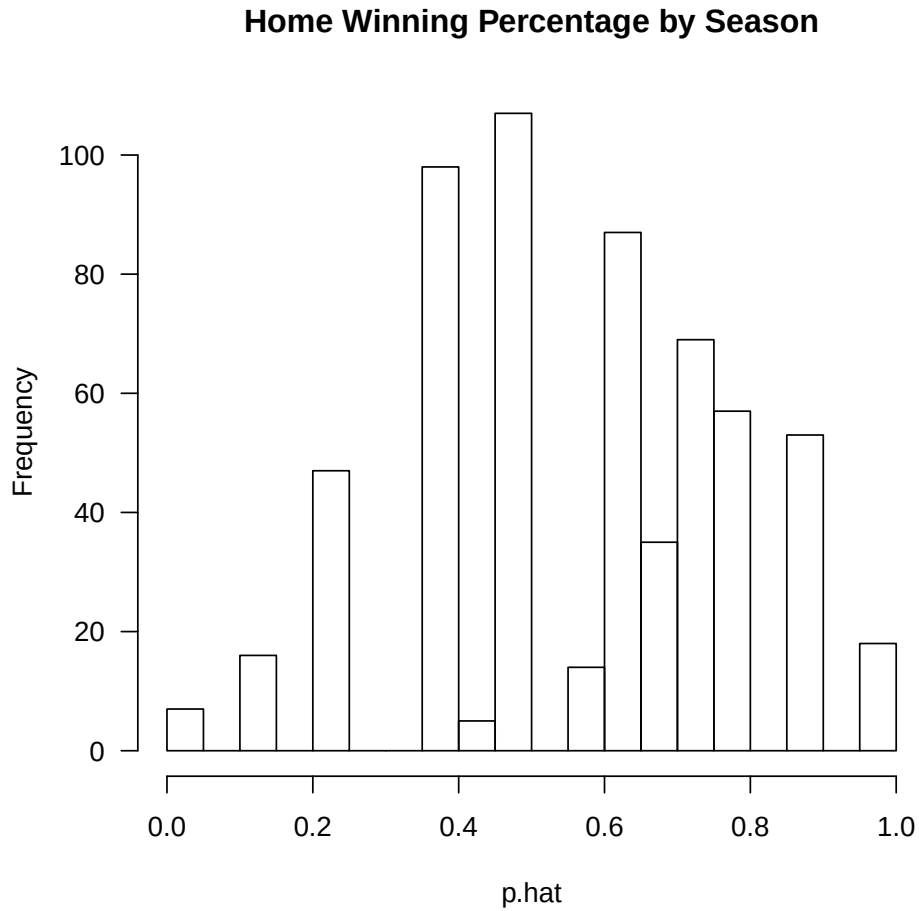


1. Priors for binomial proportions

- Beta(1,1) is the uniform distribution on (0,1), and is an example of a "non-informative" prior. (aka "flat" prior or "default" prior).
- Other similar possibilities are the Jeffreys prior and the reference prior – these are formal ways to define priors that are non-informative.
- Reference: "The Selection of Prior Distributions by Formal Rules" Kass and Wasserman, 1996, *JASA*, Vol 91, No. 435, pp. 1343-1370.
- Beta(c,c) is centered at 0.5, and its variance decreases as $c \to \infty$.

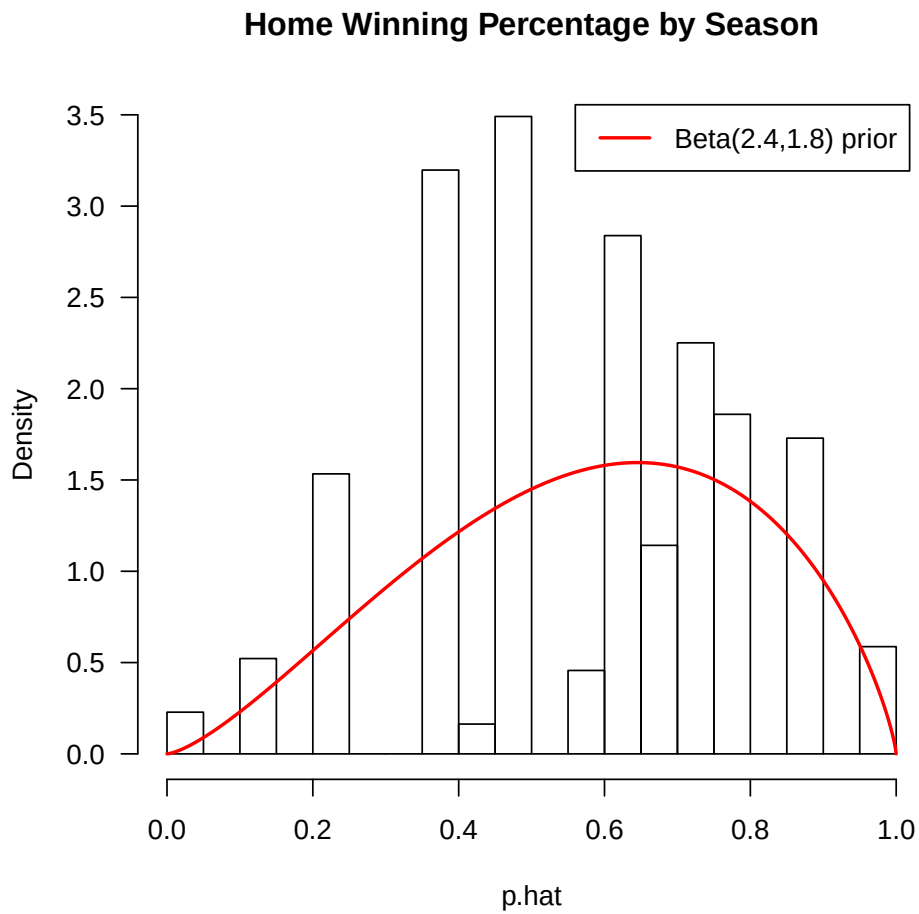
1. Empirical Bayes Estimate

- Let's do something smarter: estimate the hyperparameters using past data. Get the season-long home winning percentages for all teams for many years and estimate α and β using method-of-moments estimators.
- This is known as an empirical Bayes approach.



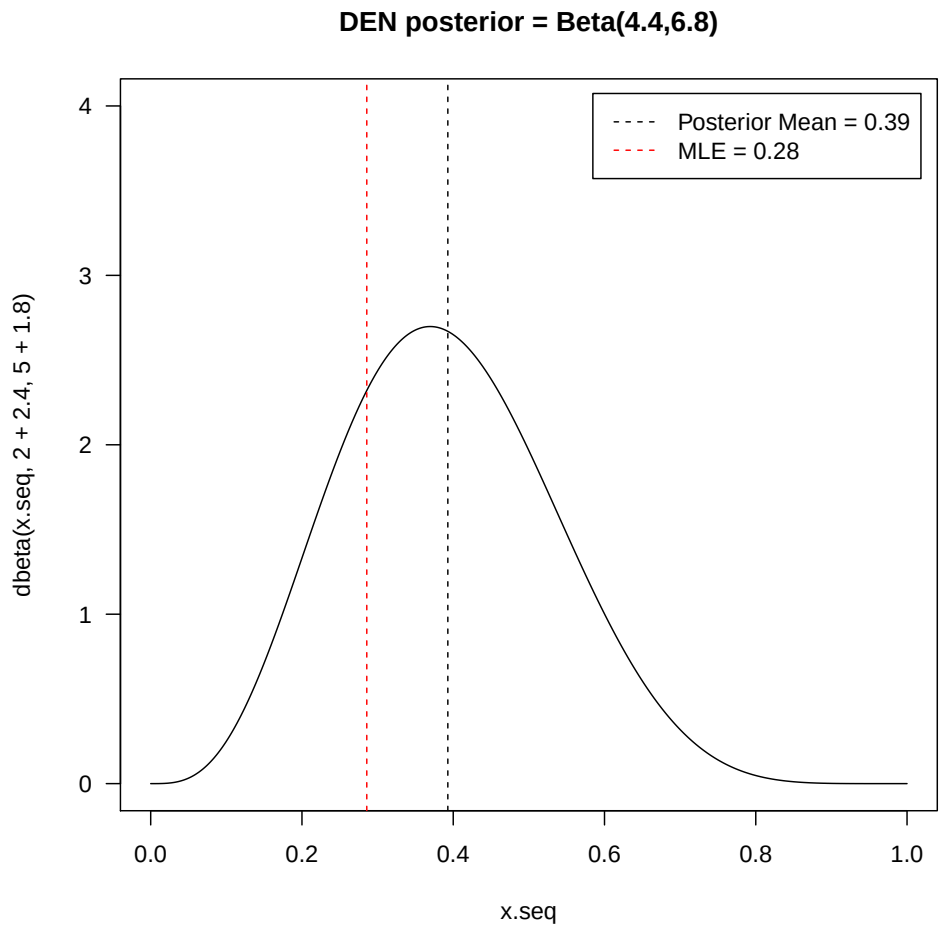
1. Empirical Bayes Estimate

- The mean and sd of the historical home winning percentages for the past 20 years are 0.57 and 0.21.
- Using method of moments estimates, we estimate $\hat{\alpha} = 2.4$ and $\hat{\beta} = 1.8$.



1. Solution

- Now DEN has a posterior distribution of Beta(2+2.4,5+1.8) = Beta(4.4,6.8):



1. Visualizing Partial Pooling Solutions

The nickname for this phenomenon is "borrowing strength". Also known as partial pooling, shrinkage, or in machine learning, "regularization". Upper points are "No Pooling", middle points are "Partial Pooling", and bottom point is "Complete Pooling".

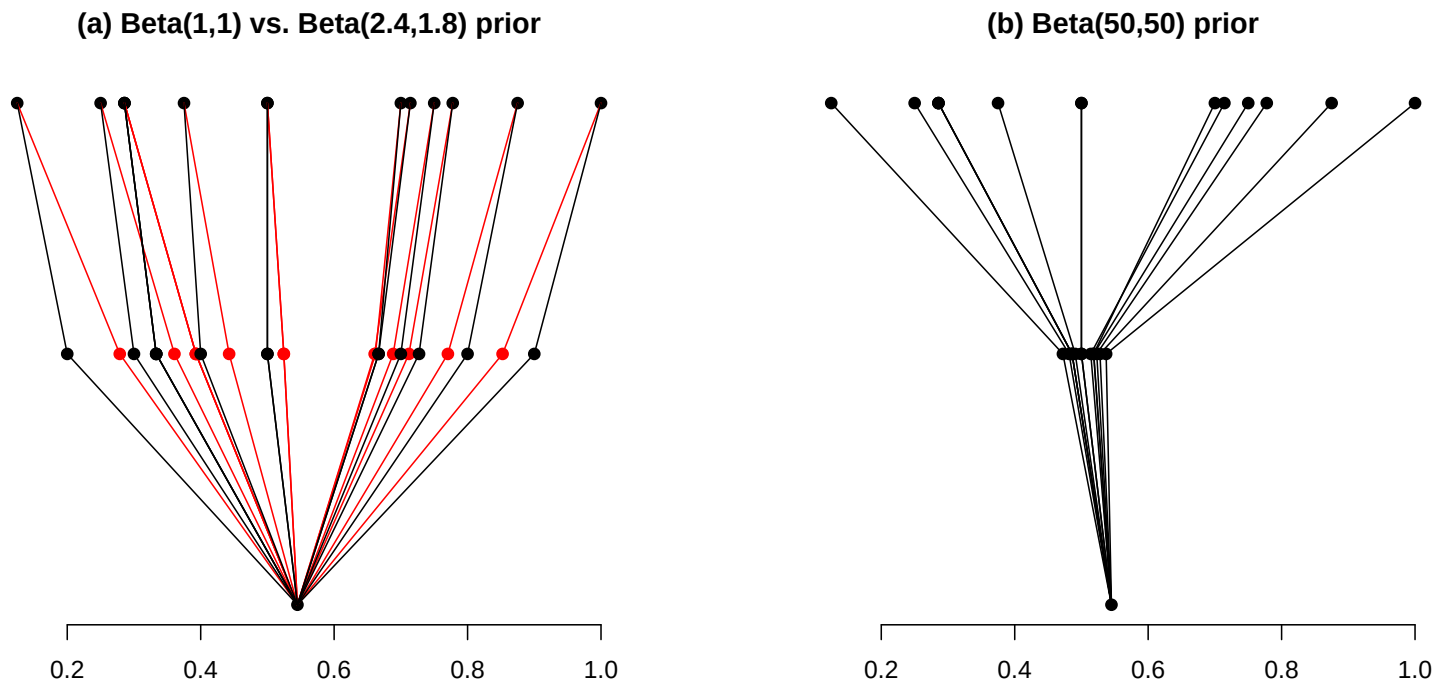


Figure: (a) In the left plot, the black lines show the Beta(1,1) prior, and the red lines show the Beta(2.4,1.8) prior.

1. Recap

- Hierarchical models assume that the unknown parameters of a model come from some probability distribution.
- This provides a compromise between "no pooling" and "complete pooling", such that inference about one group of data points is influenced by other groups of data points.
- A nice paper about this: "The Future of Indirect Evidence", by Brad Efron (Stanford Stats dept., tech. report, 2009).

Outline

1. Model 1: A simple football example
2. Model 2: A larger, more realistic football example
3. MCMC basics: The Metropolis-Hastings Algorithm
4. Output from Model 2 and post-processing MCMC results
5. A prediction for tonight's football game!

2. A more realistic football model

- The biggest missing factor in our football model was... the opponent!
- Also – a team's ability changes over time.
- Let Y_{ijt} be the margin of victory in points for home team i over away team j in week t.

$$
\begin{aligned}
Y_{ijt} &\sim \mathsf{N}(\alpha_{it} - \alpha_{jt} + \beta_i^{\text{home}}, \sigma^2) \\
\alpha_{it} &\sim \mathsf{N}(\mu_i + \phi(\alpha_{i(t-1)} - \mu_i), \sigma^2_{\text{team}}) \\
\sigma &\sim p(\sigma) \\
\beta_i^{\text{home}} &\sim \mathsf{N}(\mu_{\text{home}}, 2^2) \\
\mu_i &\sim \mathsf{N}(0, 10^2) \\
\phi &\sim \mathsf{U}(0, 1) \\
\sigma_{\text{team}} &\sim p(\sigma_{\text{team}}). \\
\mu_{\text{home}} &\sim \mathsf{N}(0, 10^2).
\end{aligned}
$$

2. A more realistic football model

- This is hard! The unknown parameters are:
  1. α, a 32 x T matrix, where T is the number of weeks in the data
  2. β_i^{home}, a vector of home field advantages (allowed to vary by team)
  3. σ, the sd of score differences
  4. μ, a 32-length vector of mean team abilities (across time).
  5. ϕ, the AR(1) parameter describing week-to-week dependence of team ability
  6. σ_{team}, the sd of team ability
  7. μ_{home}, the mean home field advantage across teams
- Give the standard deviation parameters flat priors, like an inverse gamma or half-t.
- Now we can predict the margin of victory (or defeat, if it's negative) for each of the 14 games this week.

2. How can we fit such a model?

Write down the posterior distribution first:

$$
\begin{aligned}
& p(\alpha, \beta^{\text{home}}, \sigma, \mu, \phi, \sigma_{\text{team}} \mid \boldsymbol{Y}) && \text{(Posterior)} \\
\propto\ & p(\boldsymbol{Y} \mid \alpha, \beta^{\text{home}}, \sigma) \times && \text{(Likelihood)} \\
& p(\alpha \mid \mu, \phi, \sigma_{\text{team}}) p(\beta^{\text{home}} \mid \mu_{\text{home}}) \\
& p(\sigma) p(\phi) p(\mu) p(\sigma_{\text{team}}) p(\mu_{\text{home}}) && \text{(Prior)}
\end{aligned}
$$

No conjugate prior or easy way to compute the joint posterior – it's too high-dimensional.

Solution: Use Markov Chain Monte Carlo (MCMC)

Outline

1. Model 1: A simple football example
2. Model 2: A larger, more realistic football example
3. MCMC basics: The Metropolis-Hastings Algorithm
4. Output from Model 2 and post-processing MCMC results
5. A prediction for tonight's football game!

3. MCMC basics

- MCMC is an algorithm for sampling from the joint posterior distribution of the parameters of a Bayesian model.
- Instead of estimating the distribution directly, we construct a sequence of samples from the distribution. The sequence is a Markov chain.
- We run the algorithm for many iterations (hundreds or thousands are typical), and monitor it to assess whether it has converged.
- After convergence, we run it longer to collect samples for inference – we can collect as many samples as we wish to estimate posterior distributions as accurately as desired.

3. The Metropolis-Hastings Algorithm

- One of the first discovered, easiest, and most general MCMC algorithms is the Metropolis-Hastings Algorithm. Started in the physics literature.
- Metropoolis, et al (1953)
- Hastings (1970)
- A great primer is Chib and Greenberg, (1995), "Understanding the Metropolis-Hastings Algorithm", *The American Statistician*, Vol 49 No. 4 pp. 327-335.

3. The Metropolis-Hastings Algorithm: How it works

- We want a Markov chain whose stationary distribution is the posterior distribution, $p(\theta \mid \boldsymbol{Y})$. Call this distribution $\pi(\theta)$.
- We can compute $\pi(\theta)$ up to a proportionality constant (but we can't sample directly from it).
- Start at a random point, θ_0.
- Sample a candidate for the next value of θ, denoted θ^* from $q(\theta, \theta^*)$. It only depends on current value of θ, so it's a Markov chain. Example: $\theta^* \sim \mathrm{N}(\theta, \tau^2)$.
- Key condition: If this sequence satisfies the detailed balance equation,

$$\pi(\theta)q(\theta, \theta^*) = \pi(\theta^*)q(\theta^*, \theta),$$

then $\pi(\cdot)$ is the stationary distribution of this Markov chain.

3. How can we select the right $q(\theta, \theta^*)$

- Here, we have a desired stationary distribution, $\pi(\cdot)$, and we're looking for the transition kernel, $q(\theta, \theta^*)$, that will give us the Markov chain with $\pi(\cdot)$ as its stationary distribution.
- Suppose, for example, that

$$\pi(\theta)q(\theta, \theta^*) > \pi(\theta^*)q(\theta^*, \theta).$$

  - Maybe the posterior mass is higher at θ than at θ^*.
  - Or maybe the jump from θ to θ^* is more likely than the reverse.

- This means without correction, we would spend too much time at θ^*, i.e. we'd make too many jumps from θ to θ^*.
- Million dollar idea: Correct this "imbalance" with a simple accept-reject step, scaled to satisfy detailed balance. Reject some of the jumps from θ to θ^*.

3. Reject some of the jumps from θ to θ^*

- Let $\alpha(\theta, \theta^*)$ be the probability of accepting a jump from θ to θ^*. [and $\alpha(\theta^*, \theta)$ is the probability of accepting the reverse jump].
- Now the detailed balance equation is:

$$\pi(\theta)q(\theta, \theta^*)\alpha(\theta, \theta^*) = \pi(\theta^*)q(\theta^*, \theta)\alpha(\theta^*, \theta).$$

- Set $\alpha(\theta^*, \theta)$ to 1, because we don't need to reject any of these jumps.
- Now, just solve for $\alpha(\theta, \theta^*)$, the proper acceptance probability:

$$\alpha(\theta, \theta^*) = \frac{\pi(\theta^*)q(\theta^*, \theta)}{\pi(\theta)q(\theta, \theta^*)}.$$

3. Metropolis-Hastings Intuition

* Accept every step that goes "uphill", and accept only some fraction of the steps that go "downhill".

Start with θ_0. For iterations $t = 1, ..., T$:

- Generate a candidate value θ^* from $q(\theta_{t-1}, \theta^*)$.
- Compute the acceptance probability

$$\alpha = \min\left[\frac{\pi(\theta^*)q(\theta^*, \theta_{t-1})}{\pi(\theta_{t-1})q(\theta_{t-1}, \theta^*)}, 1\right].$$

- Draw $u \sim \mathrm{U}(0, 1)$.
- If $u < \alpha$, set $\theta_t = \theta^*$.
- If $u > \alpha$, set $\theta_t = \theta_{t-1}$. [Keep another copy of the previous draw].

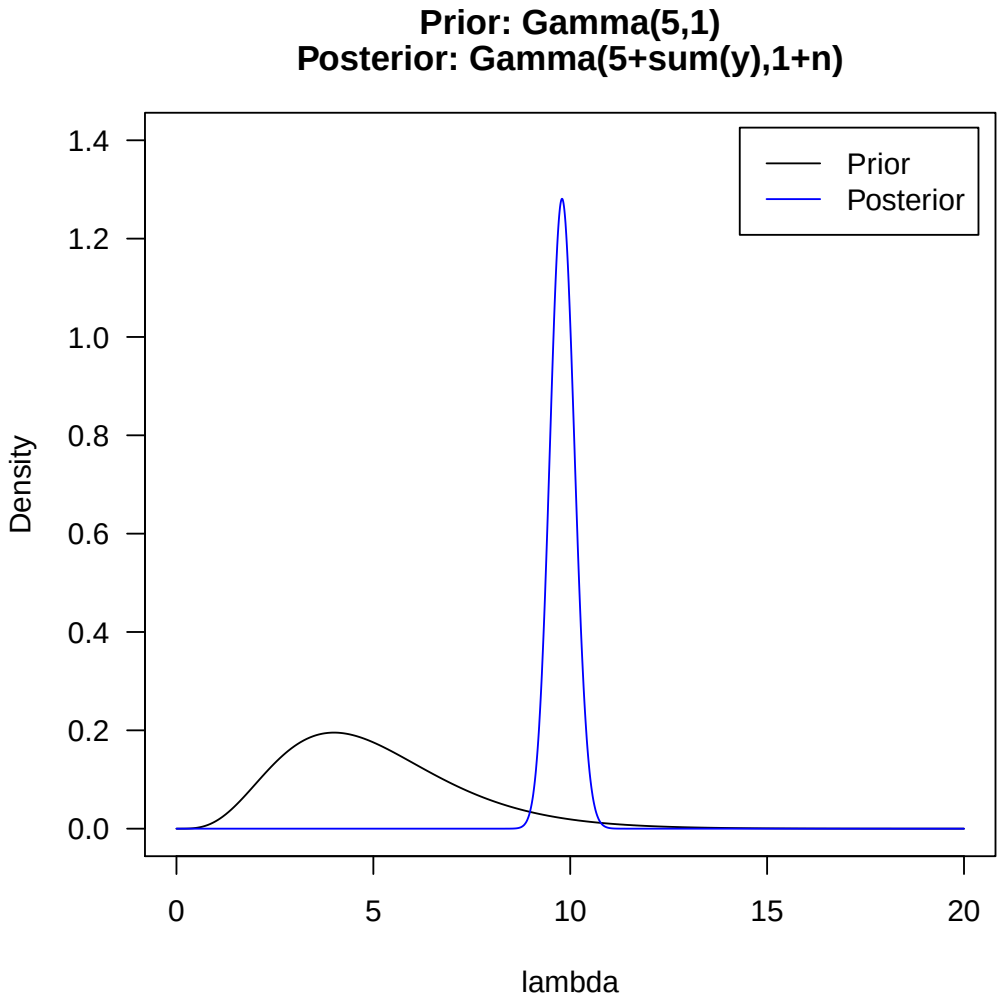
3. Metropolis-Hastings Note

- Interesting point: this is not a mode-finding algorithm (even though it is intuitive to think of algorithms trying to do this!)
- It samples from the whole posterior distribution.
- This is generally a core concept in Bayesian stats. People sometimes care a great deal about their prior distributions, for example, (exactly where does the mass belong? What is plausible?), and correspondingly, they care a lot about the whole posterior distribution, not just the mode or mean.

3. Simple M-H example: Gamma-Poisson conjugate pair

Generate data $Y_i \sim \text{Poisson}(\lambda = 10)$ for $i = 1, .., 100$.

Let the prior be $\lambda \sim \text{Gamma}(\text{shape} = 5, \text{rate} = 1)$.



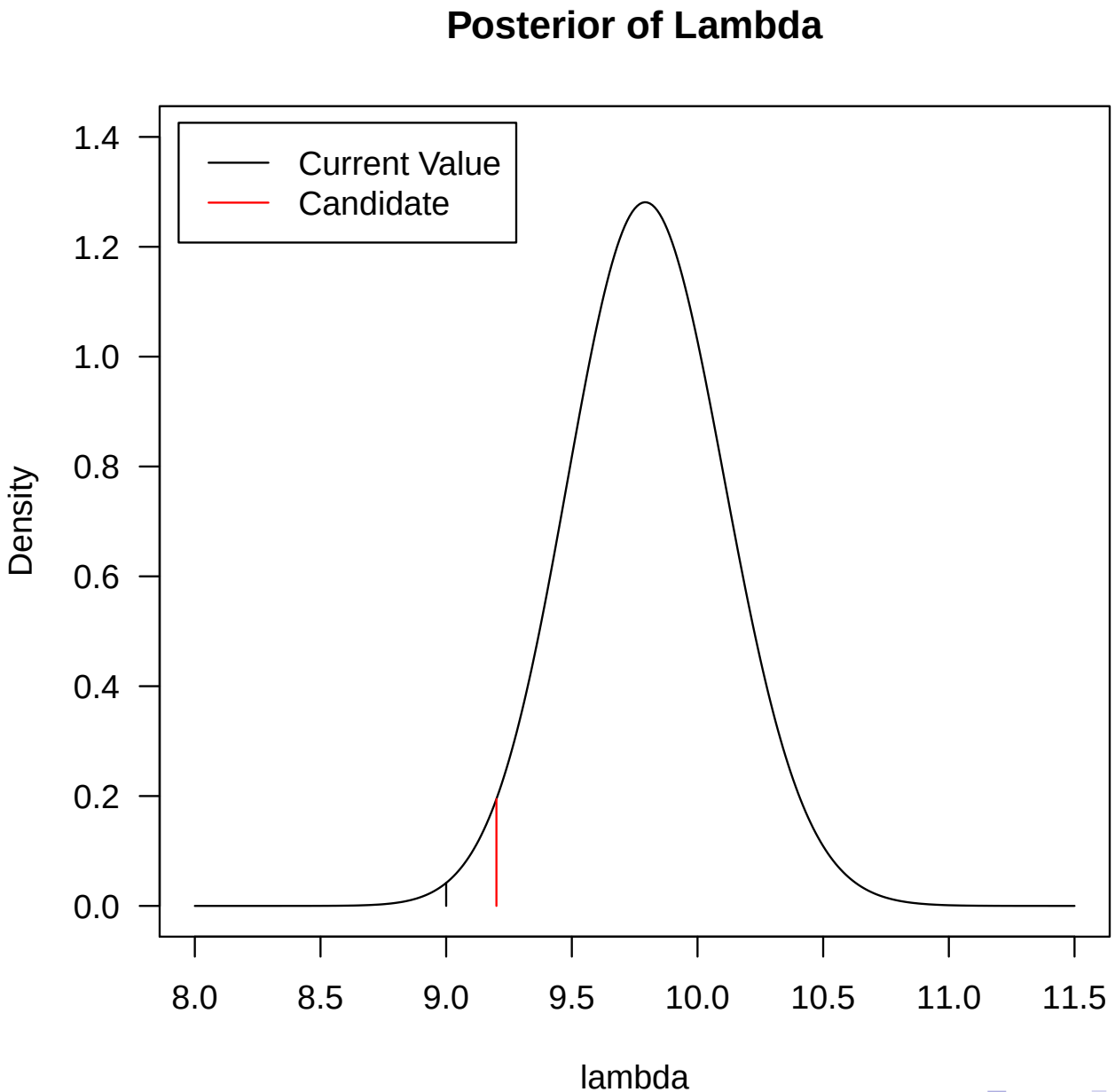
3. Let's pretend we don't know the posterior

- Start with $\lambda_0 = 9$.
- Generate a candidate $\lambda^* \sim (9, 0.5) = 9.2$. Here, we use a normal distribution as the jump distribution, where 0.5 is the jump sd. It is a tuning parameter that we'll talk about later.
- Compute acceptance probability:

$$\alpha = \frac{\text{dpois}(\boldsymbol{Y} \mid \lambda^*) \times \text{dgamma}(\lambda^* \mid \alpha, \beta) \times \text{dnorm}(\lambda^* \mid \lambda_0)}{\text{dpois}(\boldsymbol{Y} \mid \lambda_0) \times \text{dgamma}(\lambda_0 \mid \alpha, \beta) \times \text{dnorm}(\lambda_0 \mid \lambda^*)}.$$

  - This is just likelihood $\times$ prior $\times$ jump probability.
  - In other words, posterior $\times$ jump probability. $(\pi(\cdot) \times q(\cdot, \cdot))$
  - Here, $\alpha = 4.5$. It is greater than 1 (an uphill move).
- Accept the jump, and set $\lambda_1 = 9.2$.

3. Picture Iteration 2

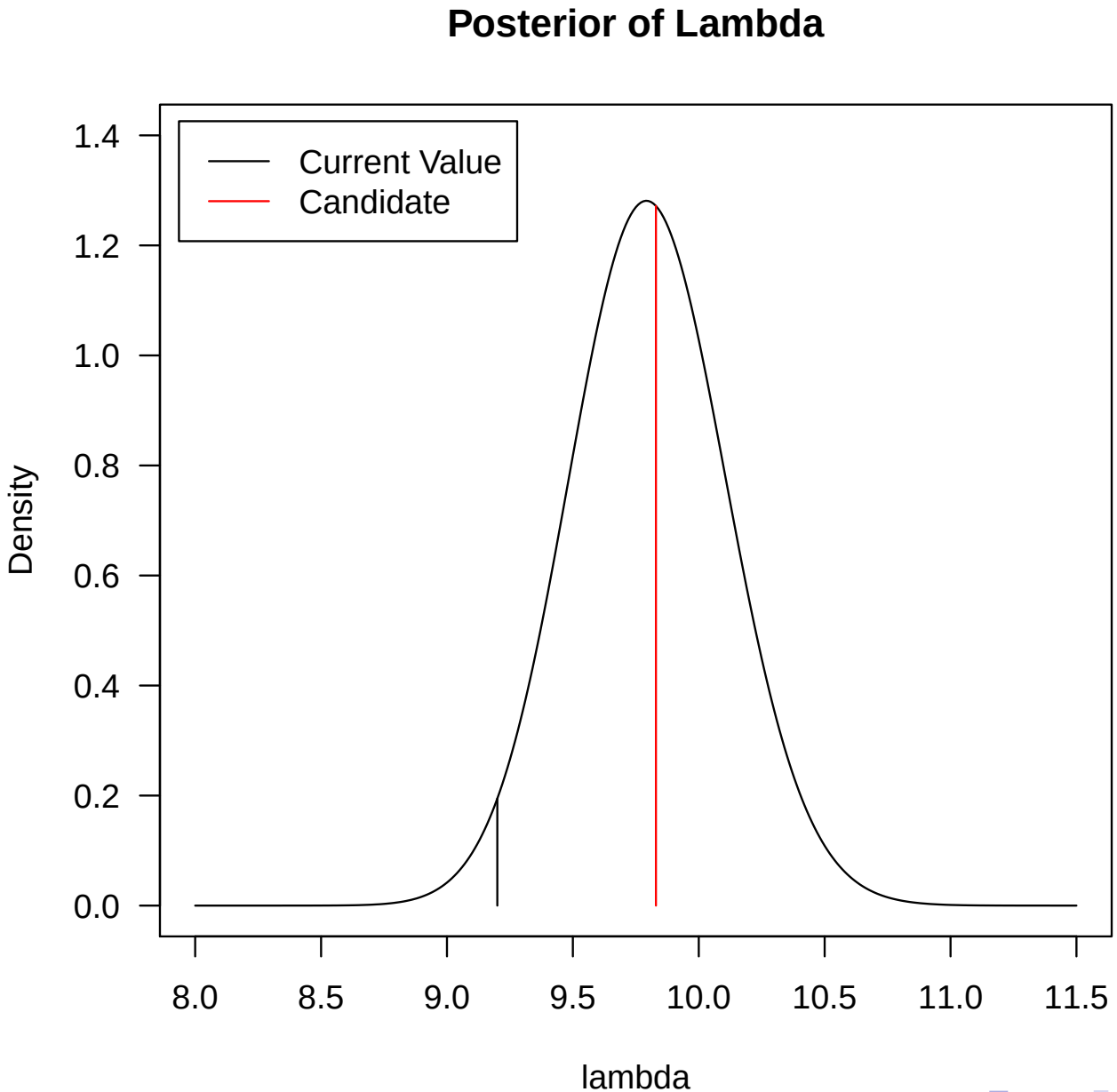


3. Table of Draws

λ^* is the candidate
α is the probability of accepting the candidate
"accept" is random – this is just one realization of the algorithm
λ is our final sequence of draws that we keep

Iteration	λ^*	α	accept?	λ
1	9	-	-	9
2	9.2	4.5	yes	9.2

3. Picture Iteration 3

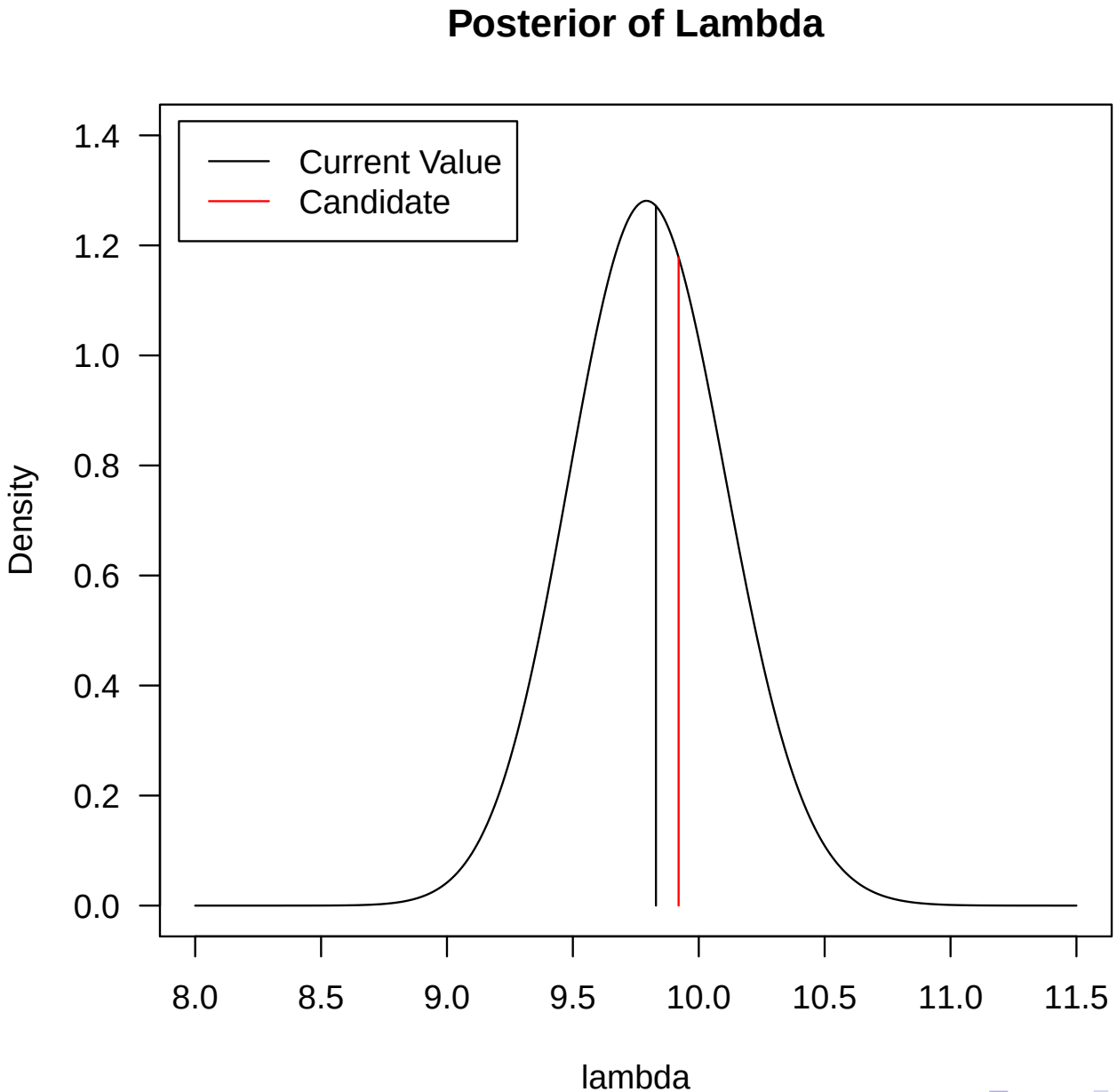


3. Table of Draws

λ^* is the candidate
α is the probability of accepting the candidate
"accept" is random – this is just one realization of the algorithm
λ is our final sequence of draws that we keep

Iteration	λ^*	α	accept?	λ
1	9	-	-	9
2	9.2	4.5	yes	9.2
3	9.83	6.7	yes	9.83

3. Picture Iteration 4

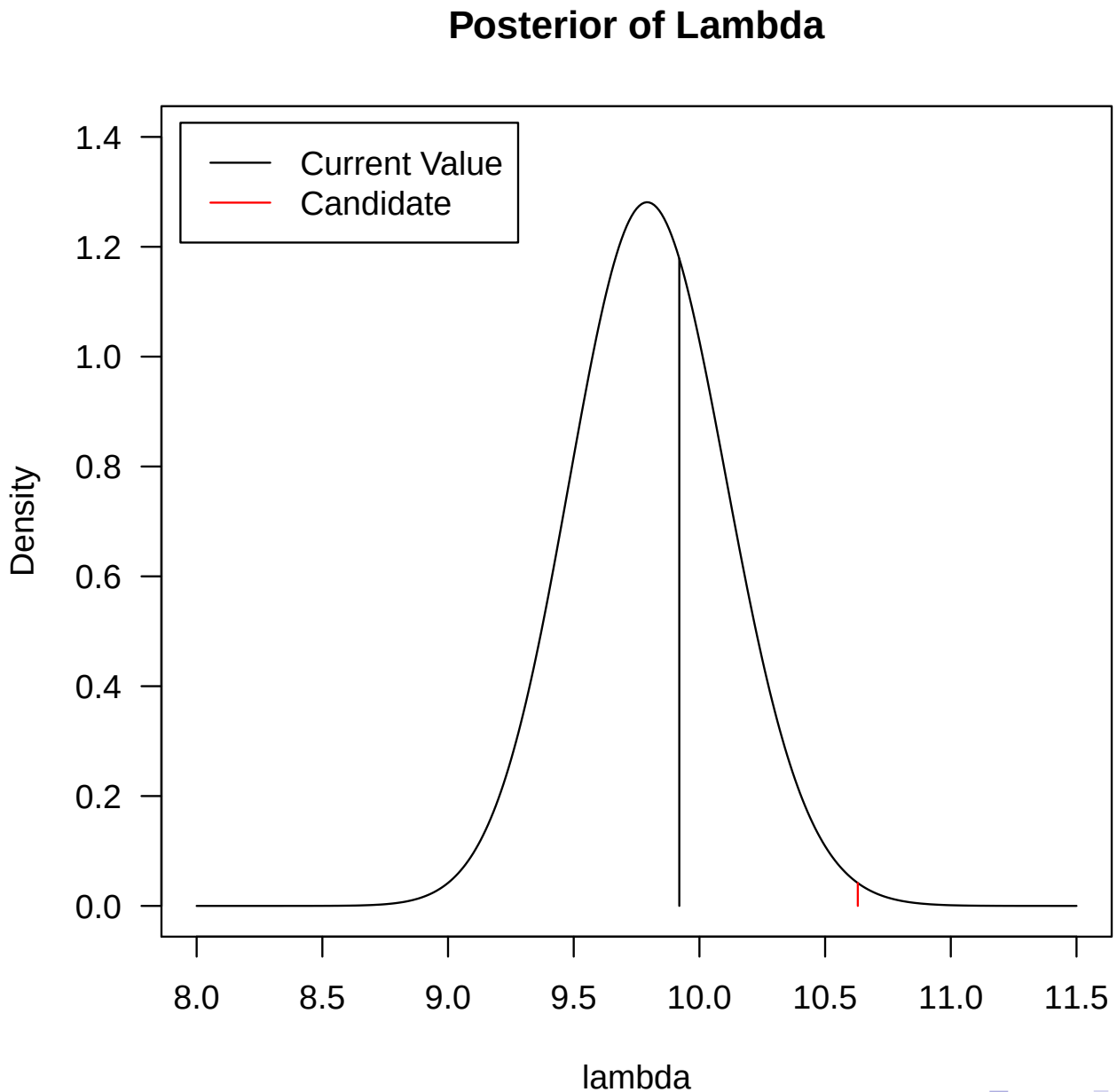


3. Table of Draws

λ^* is the candidate
α is the probability of accepting the candidate
"accept" is random – this is just one realization of the algorithm
λ is our final sequence of draws that we keep

Iteration	λ^*	α	accept?	λ
1	9	-	-	9
2	9.2	4.5	yes	9.2
3	9.83	6.7	yes	9.83
4	9.92	0.93	yes	9.92

3. Picture Iteration 4



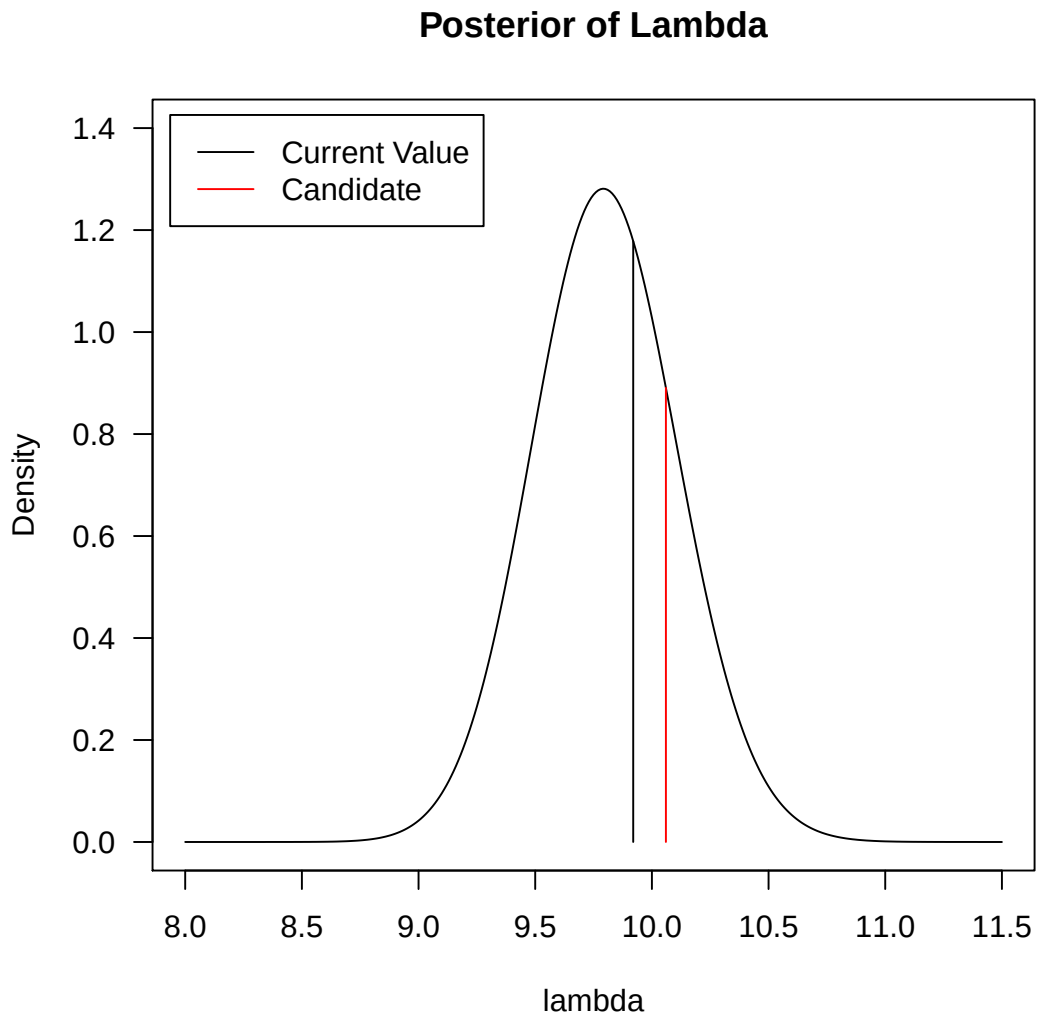
3. Table of Draws

λ^* is the candidate
α is the probability of accepting the candidate
"accept" is random – this is just one realization of the algorithm
λ is our final sequence of draws that we keep

Iteration	λ^*	α	accept?	λ
1	9	-	-	9
2	9.2	4.5	yes	9.2
3	9.83	6.7	yes	9.83
4	9.92	0.93	yes	9.92
5	10.63	0.03	no	9.92

3. Picture Iteration 5

Remember: we stayed at $\lambda = 9.92$ last time.



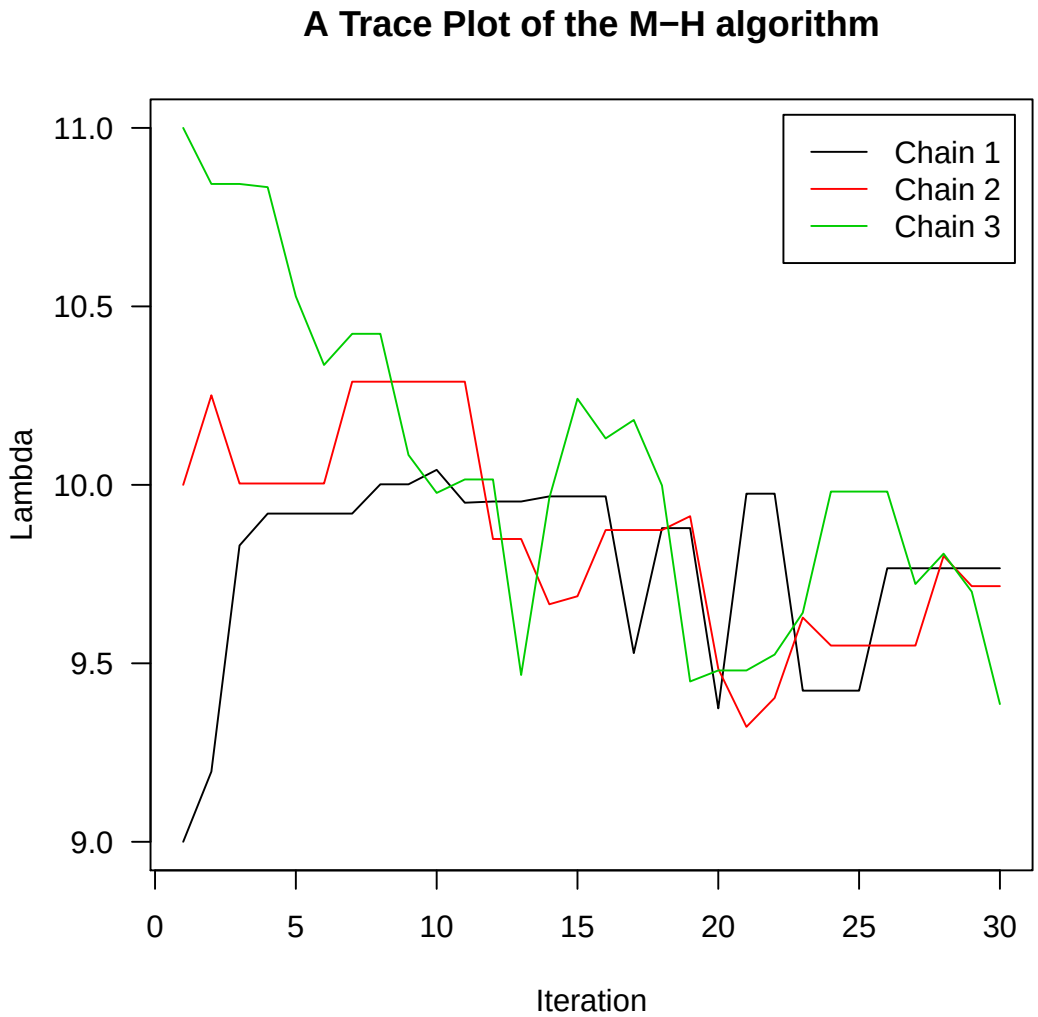
3. Table of Draws

λ^* is the candidate
α is the probability of accepting the candidate
"accept" is random – this is just one realization of the algorithm
λ is our final sequence of draws that we keep

Iteration	λ^*	α	accept?	λ
1	9	-	-	9
2	9.2	4.5	yes	9.2
3	9.83	6.7	yes	9.83
4	9.92	0.93	yes	9.92
5	10.63	0.03	no	9.92
6	10.06	0.76	no	9.92

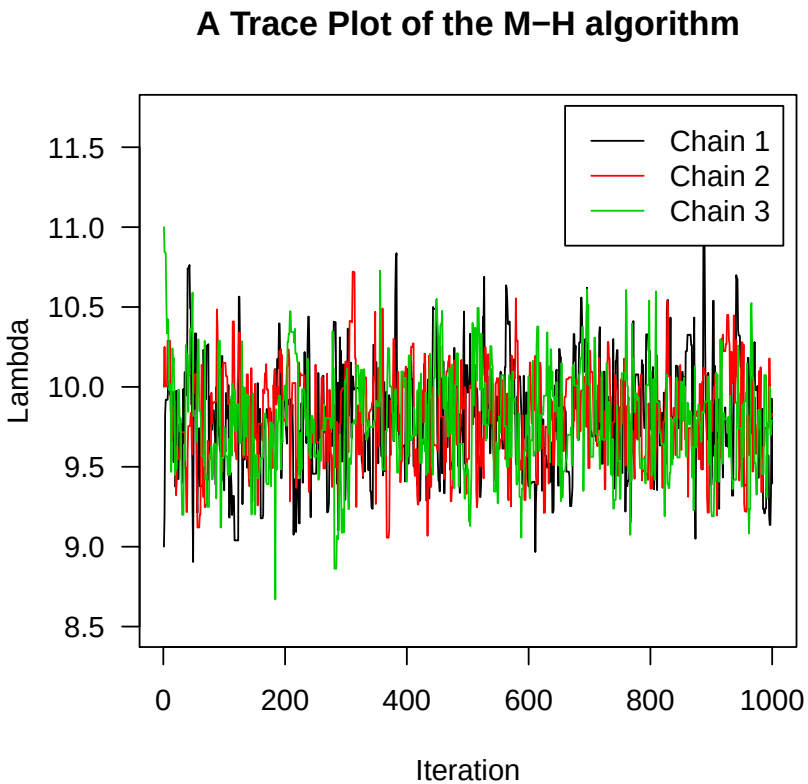
3. MCMC Output: Trace Plots

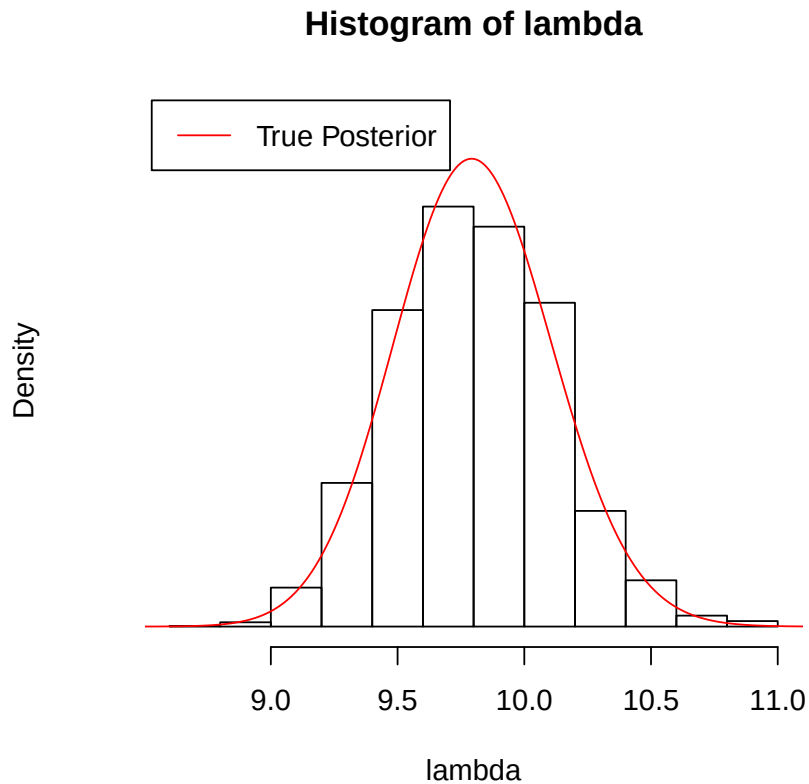
The most basic plot of MCMC output is the trace plot, where iterations are plotted on the x-axis, and the value of θ is plotted on the y-axis.



3. MCMC: Pretty close to Magic

All we needed was to be able to compute the densities of the likelihood (to a proportionality constant) and the prior.





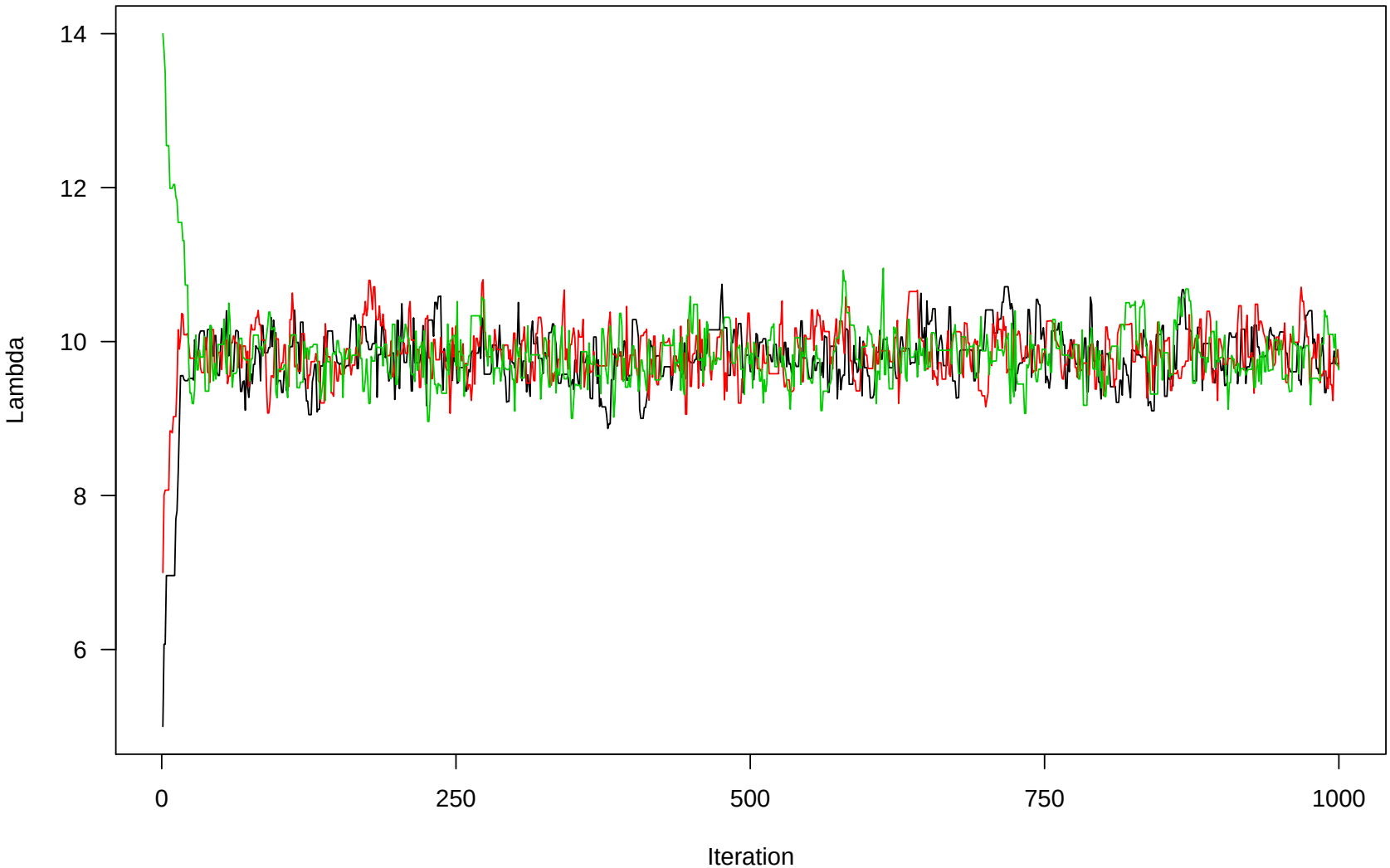
Pause for Questions

3. Two practical issues

1. Convergence

- All we've done is constructed a Markov chain with the right stationary distribution. How do we know the algorithm has converged, and our samples are actually from the stationary distribution?
- Or does our initial starting value still have an effect on the samples we're using for inference?
- "You only know where you've been". Solution: run multiple chains.
- Key idea: Use overdispersed starting point to get a better picture of whether you've explored the whole posterior mass.
- Compute $\hat{R}$ from Gelman and Rubin (1992), the most common convergence diagnostic.
- Throw away the first half of iterations, and use only the second half, if $\hat{R} < 1.1$. Otherwise, run the chain longer.

3. I'm much more inclined to believe it's converged now

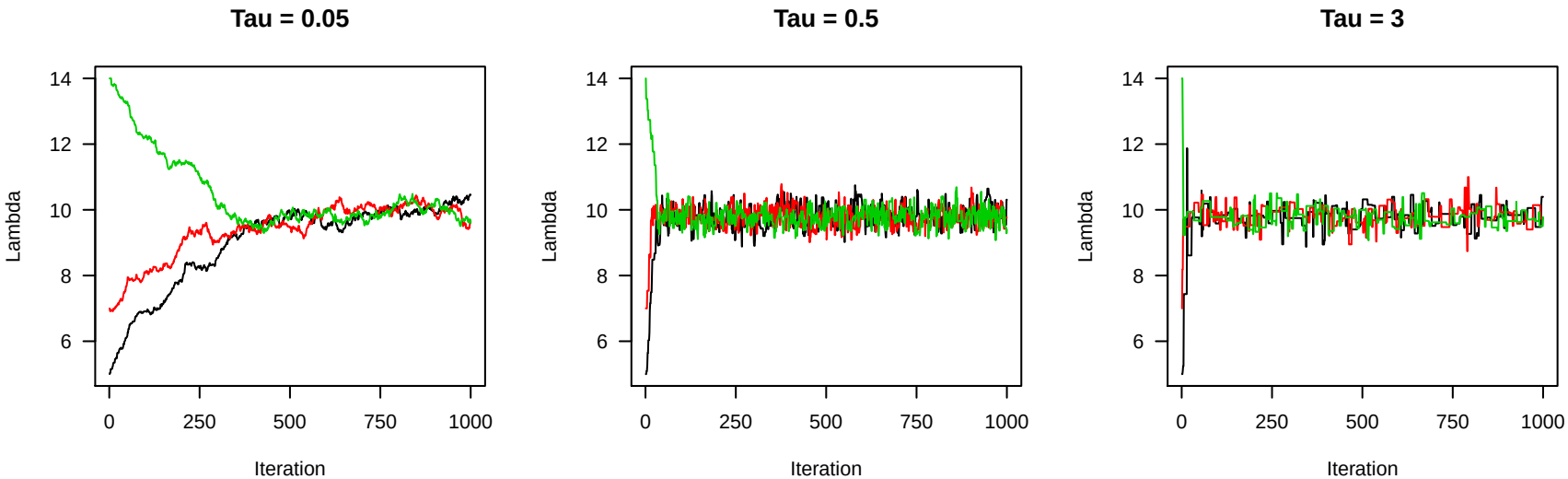


3. Two practical issues

1. Tuning Parameters

- How do select the size of the sd in the jump distribution for a random walk?
- If τ is too small, we'll explore the space very slowly, with a high acceptance rate. Maybe we'll even get stuck in a local mode forever.
- If τ is too big, we'll skip over lots of posterior mass and will accept very few steps.
- Solution: Tune τ for the first 100 iterations, increasing it to accept fewer draws, or shrinking it to accept more draws. You typically can't use these iterations for inference, but after you fix τ, then you can keep them.
- Gelman et al (1995) Bayesian Data Analysis textbook describes how a 45% acceptance rate is optimal for certain settings.

3. Different values of jump sd



- $\tau = 0.05$. Very slow exploration of the space. High autocorrelation. High (86%) acceptance rate.
- $\tau = 0.5$. Good mixing. 56% acceptance rate.
- $\tau = 3$. Low acceptance rate (13%).

3. MCMC bigger picture

- About this algorithm: If the jump distribution is symmetric, this is a special case of the Metropolis-Hastings algorithm called the Metropolis Algorithm.
- Two common kinds of Metropolis algorithms
  - Random-walk (the jump is a random walk centered at the current value of θ.)
  - Independence (the jump does not depend on the current value of θ.)
- Other kinds of MCMC include:
  - Gibbs Sampling. Very popular. Sample parameters from their full conidtional distributions (they must be available in closed form). Reference: "Gibbs for kids" Casella and George (1992) "Explaining the Gibbs Sampler" *The American Statistician* Vol. 46 pp. 167-174.
  - Hybrid, or Hamiltonian Monte Carlo. Lower autocorrelation. Use derivatives to explore the posterior space within each iteration, and the result is the iterations depend less on each other.

Outline

1. Model 1: A simple football example
2. Model 2: A larger, more realistic football example
3. MCMC basics: The Metropolis-Hastings Algorithm
4. Output from Model 2 and post-processing MCMC results
5. A prediction for tonight's football game!

4. Back to the hierarchical football model

- Let Y_{ijt} be the margin of victory in points for home team i over away team j in week t.

$$
\begin{aligned}
Y_{ijt} &\sim \mathsf{N}(\alpha_{it} - \alpha_{jt} + \beta_i^{\text{home}}, \sigma^2) \\
\alpha_{it} &\sim \mathsf{N}(\mu_i + \phi(\alpha_{i(t-1)} - \mu_i), \sigma^2_{\text{team}}) \\
\sigma &\sim p(\sigma) \\
\beta_i^{\text{home}} &\sim \mathsf{N}(\mu_{\text{home}}, 2^2) \\
\mu_i &\sim \mathsf{N}(0, 10^2) \\
\phi &\sim \mathsf{U}(0, 1) \\
\sigma_{\text{team}} &\sim p(\sigma_{\text{team}}). \\
\mu_{\text{home}} &\sim \mathsf{N}(0, 10^2).
\end{aligned}
$$

4. MCMC option 1: Code by hand

Write down the posterior distribution first:

$$
\begin{aligned}
& p(\alpha, \beta^{\text{home}}, \sigma, \mu, \phi, \sigma_{\text{team}} \mid \boldsymbol{Y}) && \text{(Posterior)} \\
\propto\ & p(\boldsymbol{Y} \mid \alpha, \beta^{\text{home}}, \sigma) \times && \text{(Likelihood)} \\
& p(\alpha \mid \mu, \phi, \sigma_{\text{team}}) p(\beta^{\text{home}} \mid \mu_{\text{home}}) && \\
& p(\sigma) p(\phi) p(\mu) p(\sigma_{\text{team}}) p(\mu_{\text{home}}) && \text{(Prior)}
\end{aligned}
$$

Set up the MCMC by sampling each parameter or block of parameters conditional on the other parameters that they depend on:

1. sample $\alpha | \boldsymbol{Y}, \beta^{\text{home}}, \sigma, \mu, \phi, \sigma_{\text{team}}$
2. sample $\beta^{\text{home}} \mid \boldsymbol{Y}, \alpha, \sigma, \mu_{\text{home}}$ + hyperparameters
3. sample $\sigma \mid \boldsymbol{Y}, \alpha, \beta^{\text{home}}$ + hyperparameters
4. sample $\mu \mid \alpha, \phi, \sigma_{\text{team}}$ + hyperparameters
5. sample $\phi \mid \alpha, \mu, \sigma_{\text{team}}$ + hyperparameters
6. sample $\sigma_{\text{team}} \mid \alpha, \mu, \phi$ + hyperparameters
7. sample $\mu_{\text{home}} \mid \beta_{\text{home}}$ + hyperparameters

4. MCMC option 1: Code by hand

1. sample $\alpha | \boldsymbol{Y}, \beta^{\text{home}}, \sigma, \mu, \phi, \sigma_{\text{team}}$
2. sample $\beta^{\text{home}} \mid \boldsymbol{Y}, \alpha, \sigma, \mu_{\text{home}}$ + hyperparameters
3. sample $\sigma \mid \boldsymbol{Y}, \alpha, \beta^{\text{home}}$ + hyperparameters
4. sample $\mu \mid \alpha, \phi, \sigma_{\text{team}}$ + hyperparameters
5. sample $\phi \mid \alpha, \mu, \sigma_{\text{team}}$ + hyperparameters
6. sample $\sigma_{\text{team}} \mid \alpha, \mu, \phi$ + hyperparameters
7. sample $\mu_{\text{home}} \mid \beta_{\text{home}}$ + hyperparameters

- Note: μ, for example, doesn't depend on the data, conditional on α being known.
- We pretend we know the value of α when we're sampling μ. Same goes for every batch.
- For each batch of parameters, you can either compute the full conditional distribution (if possible), and then use a Gibbs step, or just use something like Metropolis-Hastings.
- These 6 steps form one iteration of the MCMC algorithm.

4. MCMC option 1: Code by hand

- If your data is huge, or you need to fit the same model many times (i.e. a production-level system to model a daily feed of data), you might want to code your own MCMC in C++ or something like that.
- Otherwise, you're probably best off using a package...

4. MCMC option 2: Use software like JAGS

- JAGS stands for "Just Another Gibbs Sampler", and is an off-the-shelf, open-source software package for doing MCMC.
- http://mcmc-jags.sourceforge.net/
- It uses slice sampling, another general purpose sampler, and an alternative to Metropolis-Hastings.
- You write code as if you're writing the mathematical equations of the model. Very Easy.

4. Typical JAGS code

- Typical code:

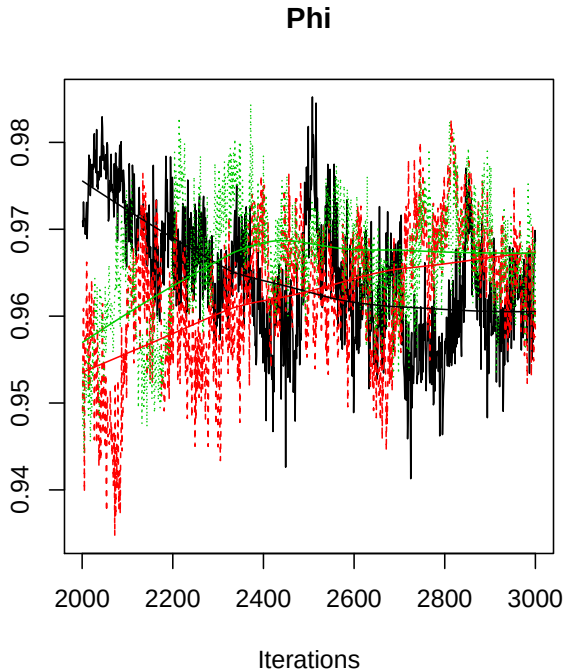
```
model{
  for (i in 1:N){
    y[i] ~ dnorm(z[i],1/(sigma*sigma))
    z[i] <- theta.team[X.home[i],X.week[i]] - theta.team[X.away[i],X.week[i]] + alpha.home
  }
  phi ~ dunif(0,1)

  for (t in 1:Teams){
    for (w in 1:Weeks){
        theta.team[t,w] ~ dnorm(mu[t] + phi*(theta.team[t,w-1]-mu[t]),1/
(sigma.team*sigma.team))
      }
    }
  }
}
```

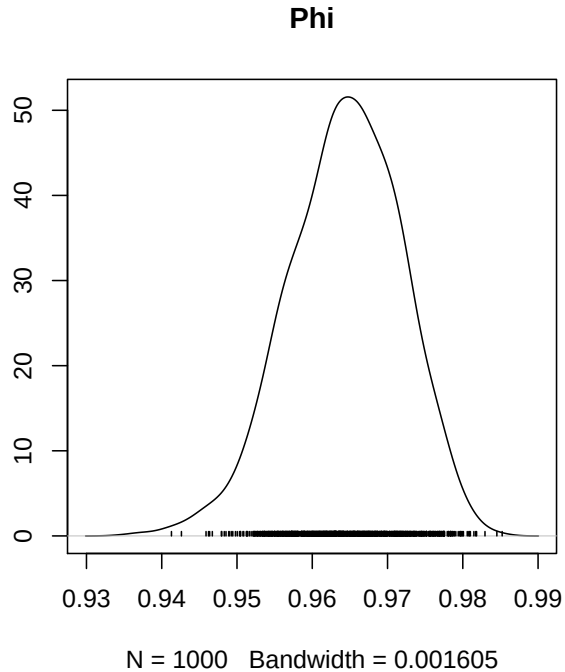
- JAGS has a nice interface with R for data setup and reading in output.

4. Some Football Output

Let's look at the week to week dependence, ϕ:

- We adapted tuning parameters for 1000 iterations, then burn-in was 1000 iterations, and these are the last 1000 iterations. Took about 5 minutes in JAGS for 3 chains.
- A bit messy, but $\hat{R} = 1.06$, so it seems to have converged.
- Very high dependence from week to week. In other words, one big win or loss doesn't mean a team is suddenly good or bad.

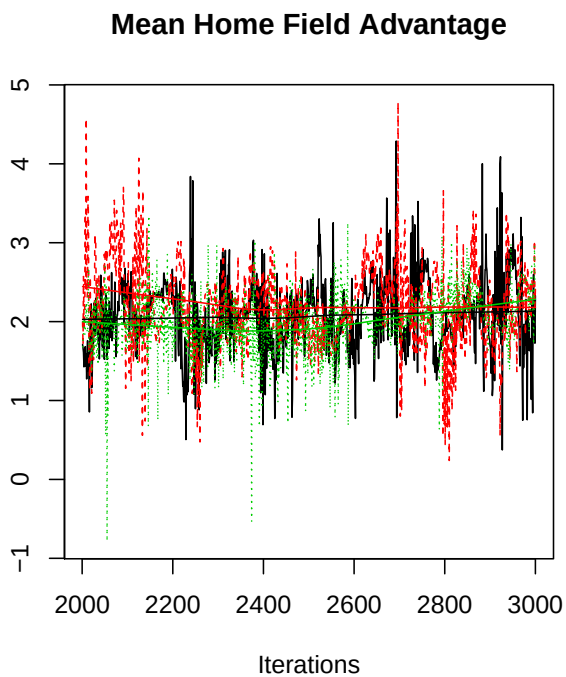


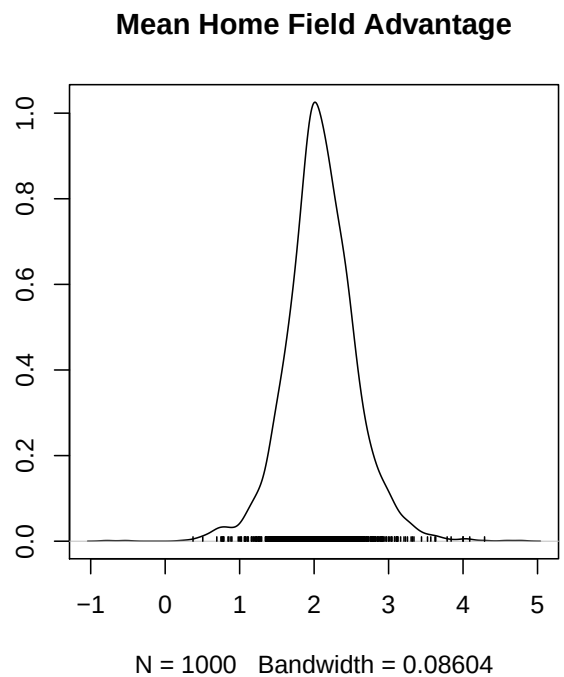


4. Some Football Output

Let's look at home-field advantage, μ_{home}:

- $\hat{R} = 1.05$, so it seems to have converged.
- Here, this is the mean of a group of parameters, so occasionally the group mean veers off course, and the individual members correct for it by adjusting in the opposite direction.
- This happens from a lack of identifiability.

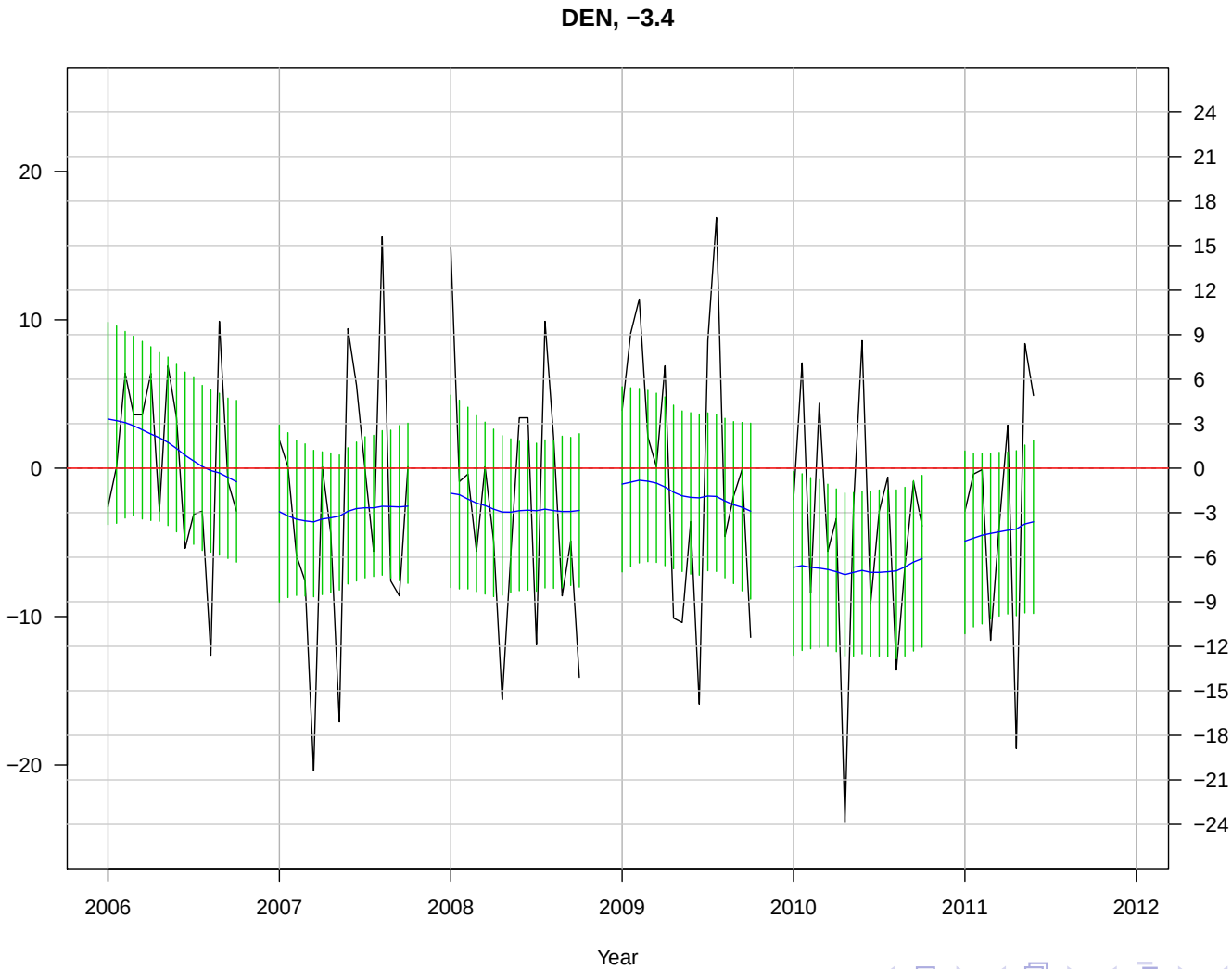




4. Some Football Output

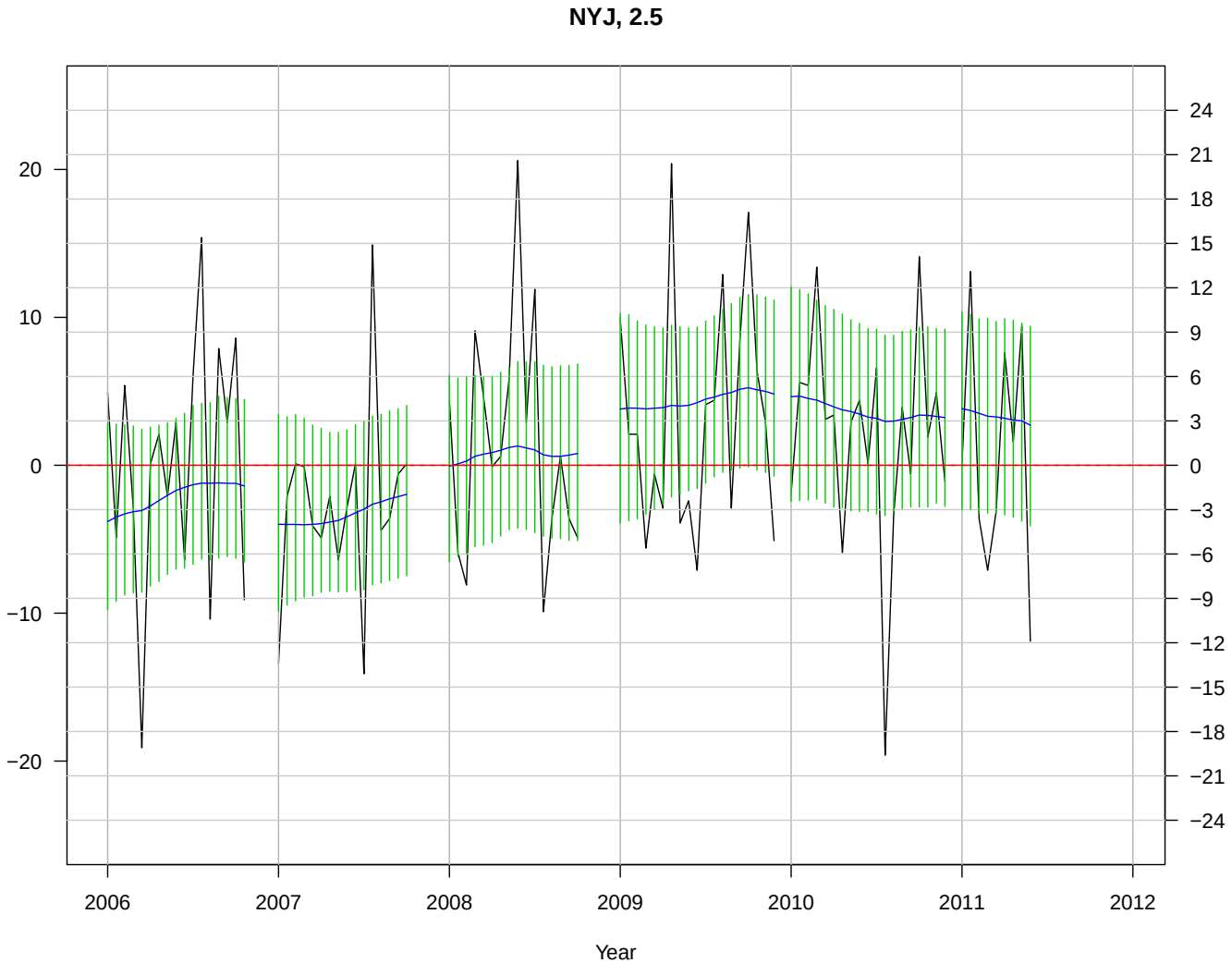
A plot of how good Denver has been over time. On average, they are 3.4 points worse than average.

- The black lines are the observed margins of victory for Denver (divided by two and adjusted for home field advantage)
- The blue line is our posterior mean of their weekly strength parameter, $\alpha_{\text{DEN},t}$.
- The green lines are 95% intervals for the weekly strength.



4. Some Football Output

Same plot for the New York Jets:



5. Last Main Slide: Tonight's Prediction

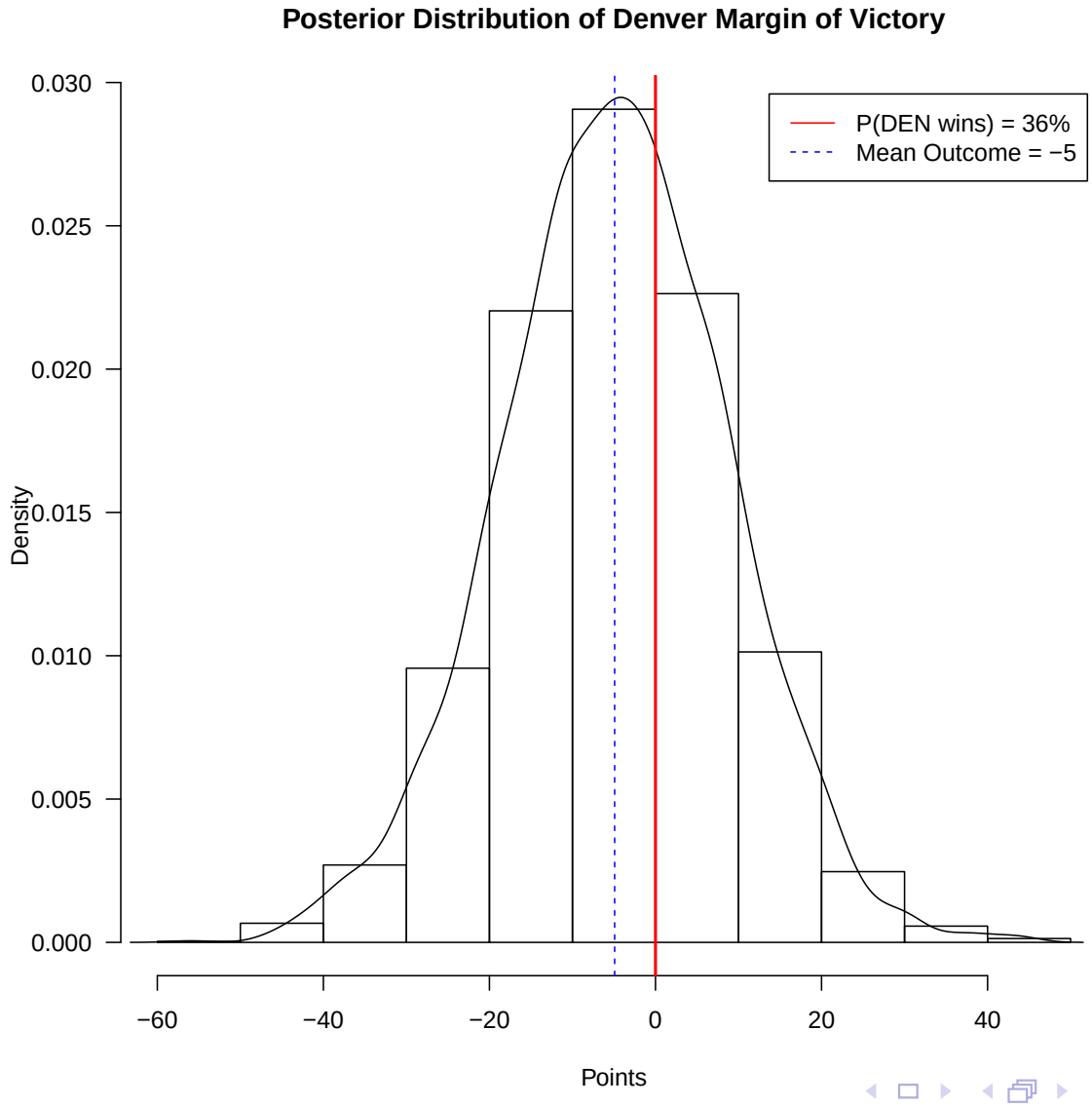
What will be the outcome of tonight's New York Jets at Denver Broncos game?
We can just simulate from the posterior distribution of this quantity of interest

$$Y^{\text{pred}} \sim \text{N}(\alpha_{\text{DEN},11} - \alpha_{\text{NYJ},11} + \beta^{\text{home}}, \sigma^2),$$

and get 3000 simulated games. Look at the distribution of these simulated outcomes...

5. Last Main Slide: Tonight's Prediction

Earlier estimates were 28% (MLE), 57% (complete pooling), 39% (partial pooling) Now our estimate is about 36%.



Recap

- Hierarchical models accomplish partial pooling, which is usually a better way to model data than the alternatives.
- Often it's necessary to use a Bayesian model and MCMC to estimate the parameters of these models.
- Metropolis-Hastings is a very general algorithm for sampling from the posterior distribution of an unknown parameter; all you need is to be able to compute the likelihood and prior density.

References

- Football model: Glickman and Stern, 1998, "A State Space Model for National Football League Scores", *JASA*, Vol. 93, pp. 25-35.
- A nice textbook about Bayesian stats with some MCMC and programming tips: Gelman, Carlin, Stern, Rubin, *Bayesian Data Analysis*, 2003, 2nd edition.
- The coolest MCMC example you'll ever see (and can now code up, because it's just a random walk metropolis): Diaconis, P. (2009). "The Markov Chain Monte Carlo Revolution", Bulletin of the American Mathematical Society, Vol. 46, No. 2, April 2009, pp. 179-205.

  A simple example of how MCMC was used to decode a cipher that was being used by prisoners to communicate via passed notes containing strange symbols. Fascinating stuff.