Workshop

Bayesian Thinking: Fundamentals, Computation, and Multilevel Modeling

Instructor: Jim Albert
Bowling Green State University
albert@bgsu.edu

OUTLINE

1. Why Bayes?
(some advantages of a Bayesian perspective)

2. Normal Inference
(introduction to the Bayesian paradigm and computation)

3. Overview of Bayesian Computation
(discussion of computational strategies and software)

4. Regression
(introduction to Bayesian regression)

5. Worship Data
(regression models for count data)

6. Attendance Data
(beta regression model for fraction response data)

7. Home Runs
(introduction to multilevel modeling)

8. Multilevel Modeling
(multilevel regression model)

Bayesian Thinking: Fundamentals, Computation, and Multilevel Modeling

Resources

Books:

• Albert, J. (2009) Bayesian Computation using R, 2nd edition, Springer.

• McElreath, R. (2015) Statistical Rethinking: A Bayesian Course with Examples in R and Stan, Chapman and Hall.

• Gelman, A. and Hill, J. (2007) Data Analysis Using Regression and Multilevel/Hierarchical Models, Cambridge.

R Packages:

• LearnBayes, version 2.15, available on CRAN.
• rethinking, version 1.59, available on github
• rstan, version 2.17.3, available on CRAN
• rstanarm, version 2.17.4, available on CRAN
• rjags, version 4-6, available on CRAN (also need jags software available on https://sourceforge.net/projects/mcmc-jags)

R Scripts:

All of the R code for the examples in the course (and additional examples) is available as a collection of Markdown files at

https://github.com/bayesball/BayesComputeR

Why Bayes?

Jim Albert

July 2018

Frequentist and Bayesian Paradigms

Traditional (frequentist) statistical inference

– evaluates methods on the frequency interpretation of probability

– sampling distributions

– 95% confidence interval means . . .

– P(Type I error) $= \alpha$ means . . .

– Don't have confidence in actual computation, but rather confidence in the method

Bayesian inference

– rests on the subjective notion of probability

– probability is a degree of belief about unknown quantities

– I'll describe 10 attractive features of Bayes

Reason 1 – One Recipe

- Bayesian model consists of a sampling model and a prior
- Bayes' rule – Posterior is proportional to the Likelihood times Prior
- Summarize posterior to perform inference
- Conceptually it is simple

Reason 2 – Conditional Inference

- Bayes inference is conditional on the observed data
- What do you learn about, say a proportion, based on data, say 10 successes in 30 trials?
- Easy to update one's posterior sequentially
- Contrast with frequentist sequential analysis

Reason 3 – Can Include Prior Opinion

- Often practitioners have opinions about parameters
- Represent this knowledge with a prior
- Knowledge can be vague – parameters are ordered, positively associated, positive

Reason 4 – Conclusions are Intuitive

- Bayesian probability interval:

Prob(p in (0.2, 0.42)) = 0.90

- Testing problem

P($p \leq 0.2$) = .10

- Folks often interpret frequentist p-values as Bayesian probability of hypotheses

Reason 5 – Two Types of Quantities in Bayes

- Quantities are either observed or unobserved
- Parameters, missing data, future observations are all the same (all unobserved)
- Interested in probabilities of unobserved given observed

Reason 6 – Easy to Learn About Derived Parameters

- Suppose we fit a regression model

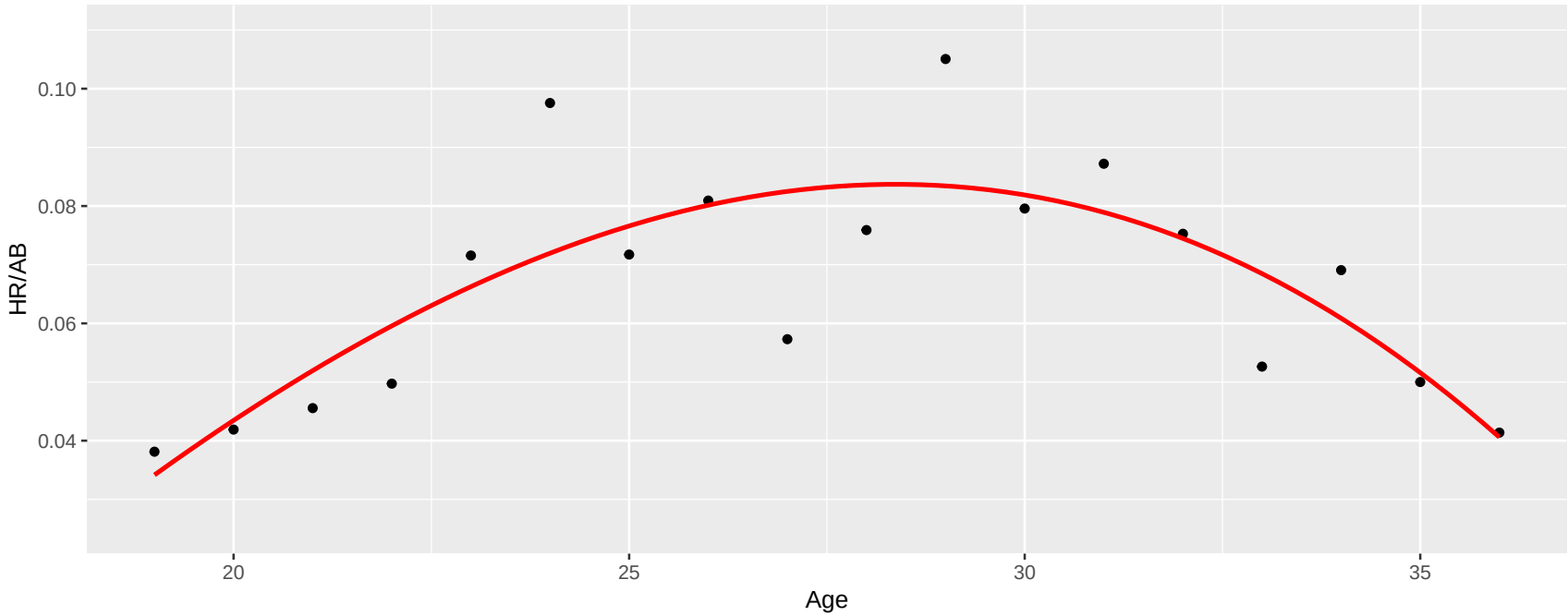
$$y_i = x_i\beta + \epsilon_i$$

- All inferences about β based on the posterior distribution $g(\beta|y)$
- Interested in learning about a function $h(\beta)$
- Just a tranformation of β – especially easy to obtain the posterior of $h(\beta)$ by simulation-based inference

Example: Learning about Career Trajectory

– Fit a quadratic regression model to Mickey Mantle's home run rates

– Interested in age $h(\beta_0, \beta_1, \beta_2)$ where the trajectory is peaked



Reason 7 – Can Handle Sparse Data

- "the zero problem" – want to learn about a proportion p when you observe $y = 0$ successes in a sample of n
- What is a reasonable estimate of p?
- Instead of an ad-hoc correction, can construct a prior which indicates that p is strictly between 0 and 1

Reason 8 – Can Move Away from Normal Distributions

- Many of the methods in an introductory statistics class are based on normal sampling
- What is so special about normal sampling?
- Bayes allows one to be more flexible in one's modeling (say, allow for outliers using a t distribution)

Reason 9 – Multilevel Modeling

- Common to fit the same regression model to several subgroups
- How to effectively combine the regressions?
- How does academic achievement in college depend on gender, major, high school grades, ACT score?
- Fit a regression model for several schools?
- Convenient to construct a multilevel model from a Bayesian perspective

Reason 10 – Advances in Bayesian Computation

- More people are doing Bayes in applied work due to ease of computation
- Markov Chain Monte Carlo algorithms are becoming more accessible
- Illustrate using Bayesian alternatives to workhouse R functions `lm`, `glm`, `lmer`, etc.

Exercises

Questions for discussion:

1. These "10 reasons" were not ranked. From your perspective what are the top 3 reasons to go Bayes?
2. David Moore (famous statistics educator) thought that we should not teach Bayes in introductory statistics since Bayes is not popular in applied statistics. Do you agree?
3. What are the challenges, in your mind, in performing a Bayes analysis?

Normal Inference

Jim Albert

July 2018

A Gaussian Model of Height (Chapter 4 from Rethinking Statistics)

- Data: partial census data for the Dobe area !Kung San
- Load from rethinking package
- Interested in modeling heights of individuals 18 and over

```
library(rethinking)
data(Howell1)
d2 <- Howell1[Howell1$age >= 18, ]
```

Bayesian Model

Normal Sampling

– Heights $h_i \sim \text{Normal}(\mu, \sigma)$

Prior

– $(\mu, \sigma) \sim g_1(\mu)g_2(\sigma)$

Questions about prior:

– What does it mean for μ and σ to be independent?

– How does one specify priors for μ, and for σ?

Specifying Prior

– Beliefs about μ are independent of beliefs about σ

– Author chooses vague prior for μ with mean equal to his height (cm), and large standard deviation (include a large range of plausible mean heights)

– Likewise choose vague prior for σ to allow for large or small spreads of individual heights

– $\mu \sim N(178, 20), \sigma \sim N(0, 50)$

This Prior Implies a Distribution for Heights

– Prior predictive density of heights

$$f(h) = \int f(h|\mu, \sigma) g(\mu, \sigma) d\mu d\sigma$$

– Sumulate heights from this distribution by . . .

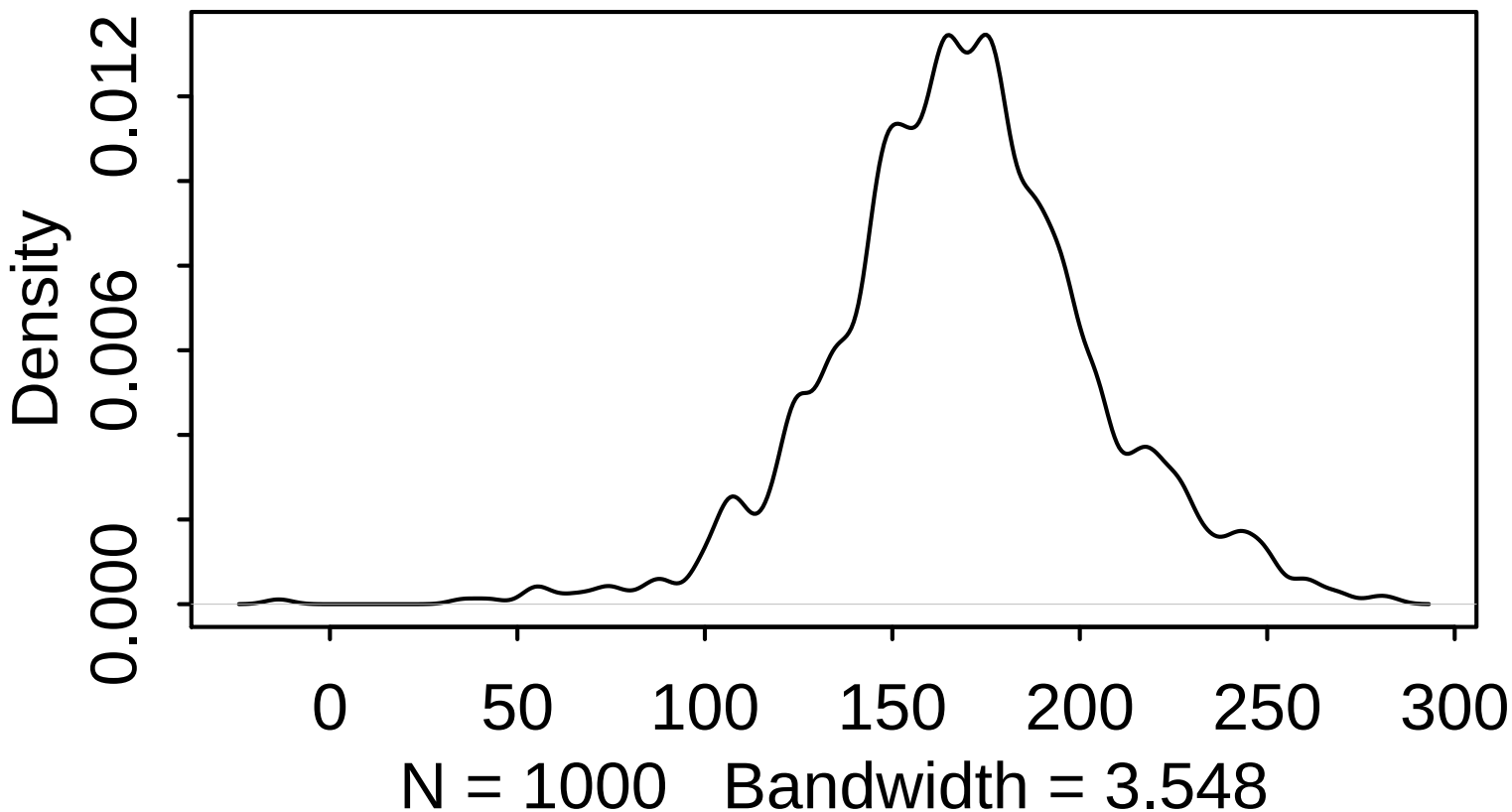
– (1) simulate 1000 values from prior of (μ, σ)

– (2) simulate 1000 heights using these random parameters

Simulation of Heights

- "expected distribution of heights, averaged over the prior"

```
sample_mu <- rnorm(1000, 170, 20)
sample_sigma <- runif(1000, 0, 50)
prior_h <- rnorm(1000, sample_mu, sample_sigma)
dens(prior_h)
```



Posterior

– Collect heights from 352 adults

– Posterior is given by "Prior times Likelihood" recipe

$$g(\mu, \sigma | data) \propto g(\mu, \sigma) L(\mu, \sigma)$$

– Likelihood is sampling density of heights, viewed as a function of the parameters

$$L(\mu, \sigma) = \prod N(y_i; \mu, \sigma)$$

Summarizing the Posterior – Some Basic Strategies

– **Grid Approximation** One computes values of the posterior over a large grid of values of μ and σ

– **Quadratic Approximation** Find the posterior mode and use this to approximate the posterior by a multivariate normal distribution

– **MCMC** Simulate from the posterior using a MCMC algorithm (Metropolis, Gibbs sampling, Stan)

Illustrate Grid Approximation

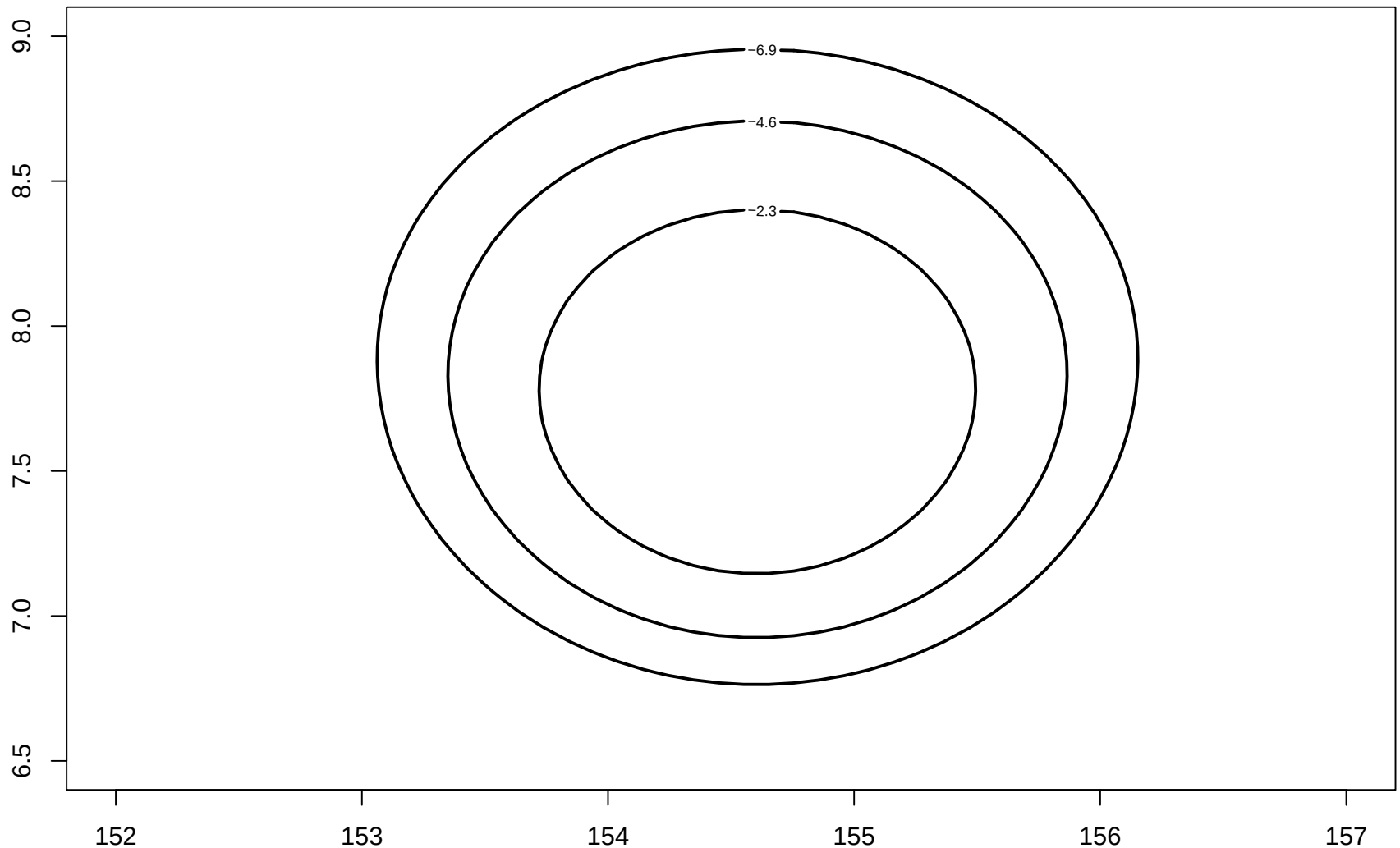
– In LearnBayes package, write function to compute the log posterior of $(\mu, \log \sigma)$

```
require(LearnBayes)
lpost <- function(parameters, y){
  mu <- parameters[1]
  sigma <- parameters[2]
  log_likelihood <- sum(dnorm(y, mu, sigma, log=TRUE))
  log_prior <- dnorm(mu, 178, 20, log=TRUE) +
               dunif(sigma, 0, 50, log=TRUE )
  log_likelihood + log_prior
}
```

Compute Posterior Over a Grid

– function `mycontour` produces contours over exact posterior density

```
mycontour(lpost, c(152, 157, 6.5, 9), d2$height)
```



Quadratic Approximation

– In `resampling` package, write some R code that defines the model

```
flist <- alist(
  height ~ dnorm( mu, sigma ) ,
  mu ~ dnorm( 178, 20 ) ,
  sigma ~ dunif( 0, 50)
)
```

- Fit model using `map` function

```
m4.1 <- map( flist, data = d2)
```

Normal Approximation

- Posterior of (μ, σ) is approximately bivariate normal
- Extract mean and standard deviations of each parameter by `precis` function

```
precis( m4.1 )
```

```
##             Mean StdDev   5.5%  94.5%
## mu     154.61   0.41 153.95 155.27
## sigma    7.73   0.29   7.27   8.20
```

Interpret

- We have Bayesian interval estimates for each parameter
- $P(153.95 < \mu < 155.27) \approx 0.89$
- $P(7.27 < \sigma < 8.20) \approx 0.89$
- Why 89 percent intervals?

What Impact Does the Prior Have?

- Try different choices for prior and see impact on the posterior intervals
- Author tries a N(178, 0.1) prior for μ – do you think this make a difference in the posterior?
- Here we have a lot of data, so ...

Different Prior

- Applying a precise prior on μ – does it make a difference?

```
m4.2 <- map(flist <- alist(
  height ~ dnorm( mu, sigma ) ,
  mu ~ dnorm( 178, 0.1 ) ,
  sigma ~ dunif( 0, 50)
  ), data=d2
)
precis(m4.2)
```

```
##           Mean StdDev   5.5%  94.5%
## mu     177.86   0.10 177.70 178.02
## sigma  24.52   0.93  23.03  26.00
```

Sampling from the Posterior

– If we use quadratic approximation, then posterior will be (approximately) multivariate normal with particular mean and var-cov matrix

```
vcov( m4.1 )
```

```
##                   mu         sigma
## mu    0.1697396253 0.0002182758
## sigma 0.0002182758 0.0849058410
```

– We sample from the posterior by simulating many values from this normal distribution

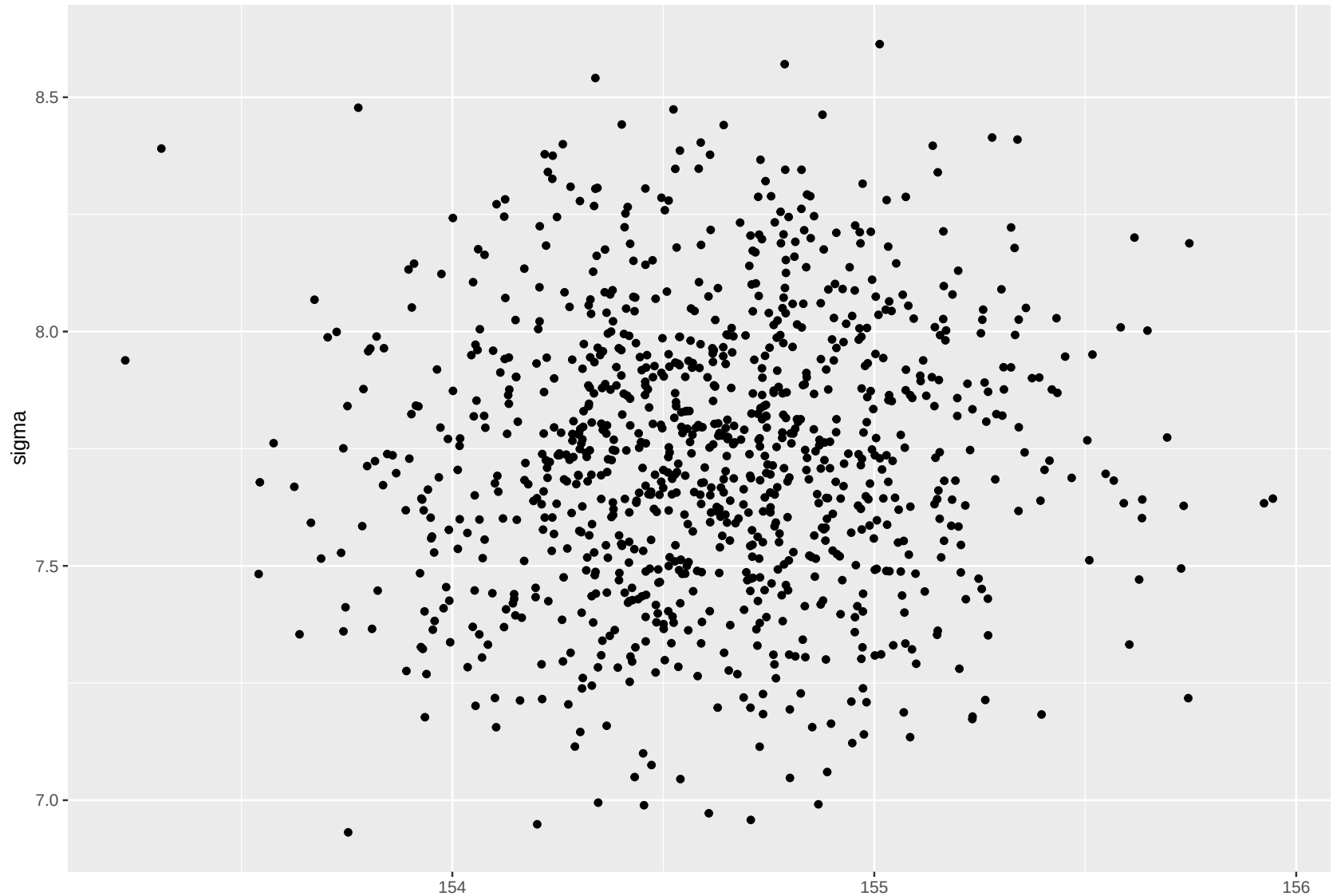
Using rethinking Package

```
post <- extract.samples( m4.1, 1000)
head(post)
```

```
##          mu    sigma
## 1 154.5127 8.279496
## 2 155.6352 7.641609
## 3 154.8807 7.578034
## 4 153.9321 7.618513
## 5 154.1646 7.389376
## 6 155.1924 7.681899
```

Graph of the Posterior

```
library(ggplot2)
ggplot(post, aes(mu, sigma)) + geom_point()
```



Can Perform Inference from Simulated Draws

Testing problem

- What is the probability *mu* is larger than 155 cm?

```
library(dplyr)
summarize(post, Prob=mean(mu > 155))
```

```
##    Prob
## 1 0.161
```

Inference from Simulating Draws

Estimation problem

- 80 percent probability interval for σ

```
summarize(post, LO = quantile(sigma, .1),
                HI = quantile(sigma, .9))
```

```
##         LO       HI
## 1 7.35395 8.132645
```

Can Learn About Functions of the Parameters

- Suppose you are interested in coefficient of variation

$$CV = \mu/\sigma$$

- Simulate from posterior distribution of CV by computing CV for each simulated draw of (μ, σ)

```
post <- mutate(post, CV = mu / sigma)
```

- 80 percent interval estimate for CV

```
summarize(post, LO = quantile(CV, .1),
                HI = quantile(CV, .9))
```

```
##          LO       HI
## 1 19.01912 21.00162
```

Prediction

- Suppose we are interested in predicting the maximum height of a future sample of size 10 - y_S
- Interested in the (posterior) predictive density

$$f(y_S) = \int f(y_S|\mu, \sigma) g(\mu, \sigma) d\mu d\sigma$$

where $g(\mu, \sigma)$ represents my current beliefs about the parameters

Simulating from the Predictive Density

- Can be difficult to directly compute $f(y_S)$
- Simulation from f is straightforward:

1. Simulate (μ, σ) from $g()$
2. Using these simulated draws, simulate y_S from $f(y_S, \mu, \sigma)$

(We have already done this earlier in this part)

Simulation of Predictive Density

- This simulates one future sample of 10 and computes the maximum height

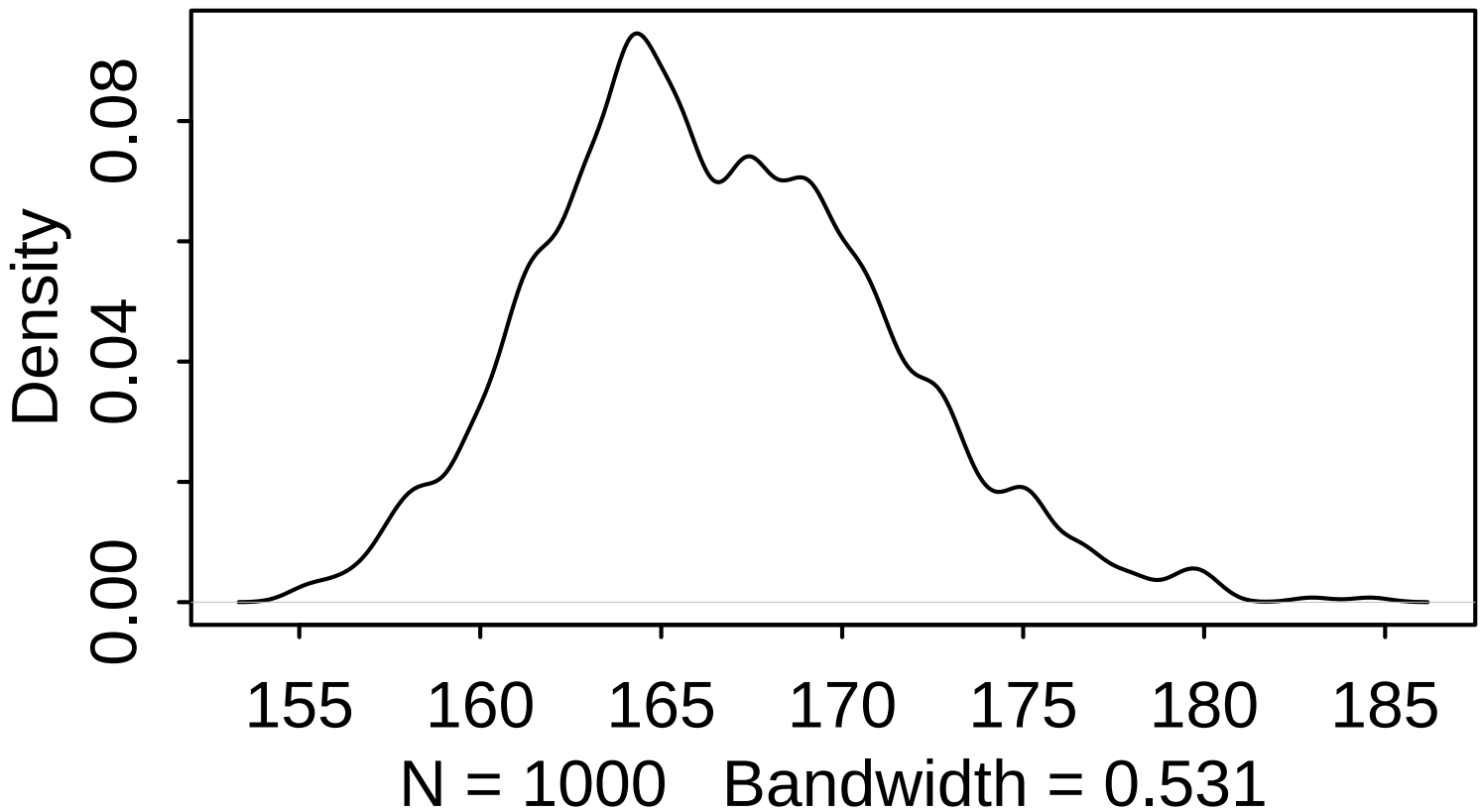
```
one_sample <- function(j){
  pars <- post[j, ]
  ys <- rnorm(10, pars$mu, pars$sigma)
  max(ys)
}
```

- Repeat this for all 1000 posterior draws

```
library(tidyverse)
MS <- map_dbl(1:1000, one_sample)
```

Posterior Predictive Density of Max

```
dens(MS)
```



Prediction Interval

- Construct a 80 percent prediction interval for M

```
quantile(MS, c(.10, .90))
```

```
##       10%       90%
## 160.8546 172.6148
```

- Interval incorporates two types of uncertainty

1. don't know values of parameters (inferential uncertainty)
2. don't know values of heights conditional on parameters (sampling uncertainty)

Exercises

Suppose you are interested in estimating the mean number of traffic accidents μ in your home town. We assume that the actual number of accidents y has a Poisson distribution with mean μ.

1. Make a best guess at the value of μ.
2. Suppose you are very sure about your guess in part 1. Construct a normal prior which reflects this belief.
3. Suppose instead that the "best guess" in part 1 was likely inaccurate. Construct a normal prior that reflects this belief.
4. Suppose you collect the numbers of traffic accidents for 10 days in your home town – call these numbers $y_1, ..., y_{10}$. Write down the Bayesian model including the sampling distribution and the prior distribution from part 1.
5. Write down the Bayesian model as a script using the `resampling` package language.

Exercises (continued)

6. Describe how you could summarize the posterior density for λ and simulate 1000 draws from the posterior.
7. Based on the simulated draws from the posterior, how can you construct a 90 percent interval estimate for λ?
8. How will your Bayesian interval in part 7. compare with a traditional 90% frequentist confidence interval?
9. Suppose you want to predict the number of traffic accidents y_N in a day next week. Explain how you would produce a simulated sample from the predictive distribution of y_N.
10. What is one advantage of a Bayesian approach over a frequentist approach for this particular problem?

Exercises (Continued)

11. I collected the numbers of traffic accidents in Lincoln, Nebraska for 10 summer days in 2016:

36, 26, 35, 31, 12, 14, 46, 19, 37, 28

– Assuming Poisson sampling with a weakly-informative prior, find a 90% interval estimate for λ.

– Find a 90% prediction interval for the number of traffic accidents in a future summer day in Lincoln.

– Explain what each line of code is doing on the next page.

Exercises (Continued)

```
library(rethinking)
d <- data.frame(y = c(36, 26, 35, 31, 12,
                      14, 46, 19, 37, 28))
bfit <- map(flist <- alist(
  y ~ dpois( lambda ) ,
  lambda ~ dnorm( 30,  10 )
), data=d
)
precis(bfit)
sim_draws <- extract.samples(bfit, 1000)
ynew <- rpois(1000, sim_draws$lambda)
quantile(ynew, c(.05, .95))
```

Overview of Bayesian Computation

Jim Albert

July 2018

Overview of Bayesian modeling

– A statistical model: y is distributed according to a sampling density $f(y|\theta)$

– θ is unknown – view it as random quantity

– Beliefs about the locations of θ described by a prior density $g(\theta)$

Observe data

– Learn about θ by computing the posterior density $g(\theta|y)$ (Bayes' theorem)

– All inferences (say interval estimates or decisions) based on the posterior density

Bayes computation problem

– An arbitrary Bayesian model: $y \sim f(y|\theta), \theta \sim g(\theta)$

– Interested in efficiently summarizing the posterior distribution

$$g(\theta|y) \propto L(\theta)g(\theta|y)$$

– Want to summarize marginal posteriors of functions $h(\theta)$

– Want to predict future replicates from the model – distribution $f(y_{rep}|y)$

Example: Item response modeling

– 200 Students take a 33 question multiple choice math exam

– Observed response of ith student to jth question is y_{ij} which is 1 (correct) or 0 (incorrect)

– Interested in modeling $P(y_{ij} = 1)$

– This probability depends on the student and also on the qualities of the item (question)

2-parameter Item Response Model

$$P(y_{ij} = 1) = \Phi\left(\alpha_j \theta_i - \gamma_j\right)$$

where

– θ_i is the ability of the ith student

– α_j, γ_j are characteristics of the jth item

– called discrimination and difficulty parameters

Some sample item response curves

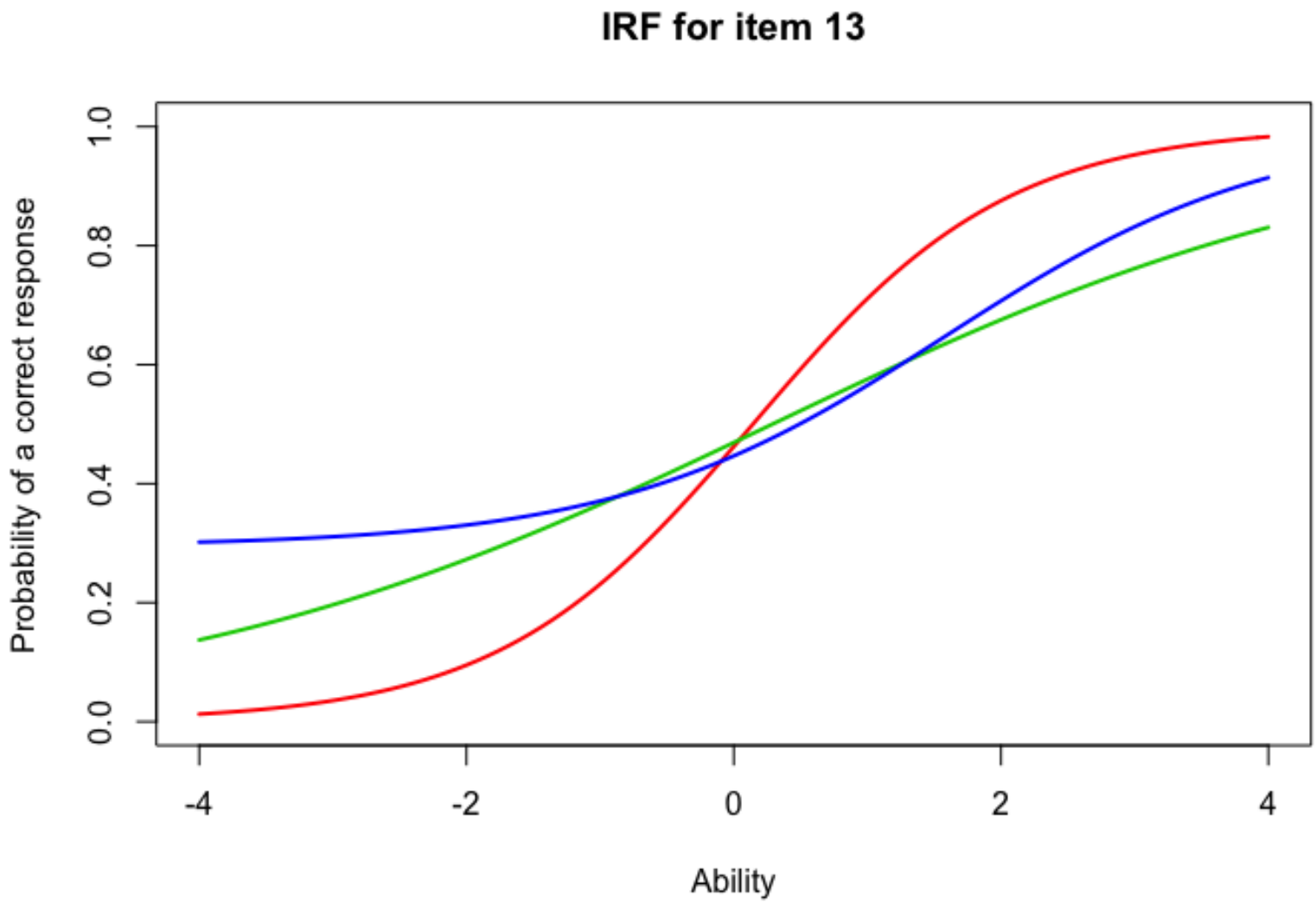


Figure 1:

Priors for IRT models

- Assume that abilities $\theta_1, ..., \theta_n \sim N(0, 1)$
- Believe discrimination parameters $\alpha_j > 0$
- Relationships between the item parameters?

Clustering effects

– Perhaps the math exam covers three subjects (fractions, trig, etc)

– Believe that item parameters within a particular subject are positively correlated

– Like to "borrow strength" to get better estimates of discrimination for a particular item

This is relatively large Bayesian problem

– Have 200 ability estimates $\theta_1, ..., \theta_n$

– Each item has two associated parameters (α_j, γ_j) (total of 66 parameters)

– May have additional parameters from the prior that need to be estimated

– Total of 266 + parameters - large multivariate posterior distribution

But model can get more complicated

– Give the same exam for several years, or at different testing environments (schools and students)

– 5 different years, have 5 x 266 parameters

– Additional parameters that model similarity of the parameters across years

In the old days (pre 1990)

– Effort by Adrian Smith at University of Nottingham to develop efficient quadrature algorithms for integrating (Bayes 4 software)

$$p(y) = \int g(\theta) d\theta$$

– General purpose, worked well for small number of parameters

– Computationally expensive for large models (number of posterior evaluations grows exponentially as function of number of parameters)

Normal approximation

– General result:

$$g(\theta|y) \approx N(\hat{\theta}, V)$$

where $\hat{\theta}$ is the posterior mode, V negative of the inverse of the second derivative matrix of the log posterior evaluated at the mode

– Analogous to the approximate normal sampling distribution of the MLE

Simulating Posterior Distributions

– Suppose one can simulate $\theta^{(1)}, ..., \theta^{(m)}$ from the posterior $g(\theta|y)$

– Then one can summarize the posterior by summarize the simulated draws $\{\theta_j\}$

– For example, the posterior mean of θ_1 is approximated by the sample mean of draws:

$$\hat{\theta}_1 = \sum \theta_1^{(j)} / m$$

– Assumes that one can simulate directly from the posterior

Conjugate Problems

– For basic distributions (normal, binomial, Poisso, etc) choose a "conjugate" prior

– Prior and posterior distributions are in the same family of distributions

– Can perform exact posterior calculations

– Can simulate from posterior using standard R functions (rnorm, rbeta, etc)

1990 Markov Chain Monte Carlo

– Gelfand and Smith (1990) introduced Gibbs sampling

– Set up a Markov Chain based on a set of conditional posterior distributions

Gibbs sampling: Missing data approach

– View inference problem as a missing data problem

– Have a probability distribution (posterior) in terms of the missing data and the parameters

– Example: regression with censored data

Censored data

– have regression model $y_i = N(\beta_0 + \beta_1 x_i, \sigma)$

– problem: some of the y_i are not observed – instead observe $w_i = \min(y_i, c)$, where c is a censoring variable

– the likelihood function of the observed data is complicated

Gibbs sampling approach

– Add the missing data y_i to the estimation problem

– want to learn about the distribution of $(y, \beta_0, \beta_1, \sigma)$

– can conveniently simulate from the (1) distribution of the missing data y conditional on the parameters $(\beta_0, \beta_1, \sigma)$ and (2) the parameters $(\beta_0, \beta_1, \sigma)$ conditional on the missing data

Metropolis Algorithm

– Provides a general way of setting up a Markov Chain

– Random walk algorithm – suppose the current value is $\theta = \theta^c$

1. Propose a value $\theta^p = \theta^c + \mathit{scale} \times Z$
2. Compute an acceptance probability P depending on $g(\theta^p)$ and $g(\theta^c)$
3. With probability P move to proposal value θ^p, otherwise stay at current value θ^c

BUGS software

– General-purpose MCMC simulation-based for summarizing posteriors

– Write a script defining the Bayesian model

– Sampling based on Gibbs sampling and Metropolis algorithtms

– Good for multilevel modeling (see BUGS examples)

– BUGS, then WinBUGS, openBUGS, JAGS (works for both Windows and Macintosh)

Cautions with any MCMC Bayesian Software

– Need to perform some MCMC diagnosistics

– Concerned about burn-in (how long to achieve convergence), autocorrelation of simulated draws, Monte Carlo standard errors of summarizes of posterior of interest

– We will see some examples of problematic MCMC sampling

R Packages

– Many packages provide efficient MCMC sampling for specific Bayesian models

– `MCMCpack` provides compiled code for sampling for a variety of regression models

– For example, function `MCMCoprobit` simulates from the posterior of an ordered regression model using data augmentation/Gibbs sampling

– `coda` package provides a suite of diagnostic and graphics routines for MCMC output

Current Status of Bayesian Computing

– Advances in more efficient MCMC algorithms

– Want to develop more attractive interfaces for Bayesian computing for popular regression models

– Include generalized linear models and the corresponding mixed models

– STAN and its related R packages

Introduction to STAN

- Interested in fitting multilevel generalized linear models
- General-purpose software like JAGS doesn't work well
- Need for a better sampler

Hamiltonian Monte Carlo (HMC)

- HMC accelerates convergence to the stationary distribution by using the gradient of the log probability function
- A parameter θ is viewed as a position of a fictional particle
- Each iteration generates a random momentum and simulates path of participle with potential energy determined by the log probability function
- Changes in position over time approximated using the leapfrog algorithm

Description of HMC in Doing Bayesian Data Analysis (DBDA)

- In Metropolis random walk, proposal is symmetric about the current position
- Inefficient method in the tails of the distribution
- In HMC, proposal depends on current position – proposal is "warped" in the direction where the posterior increases
- Graph in DBDA illustrates this point

Graph from DBDA

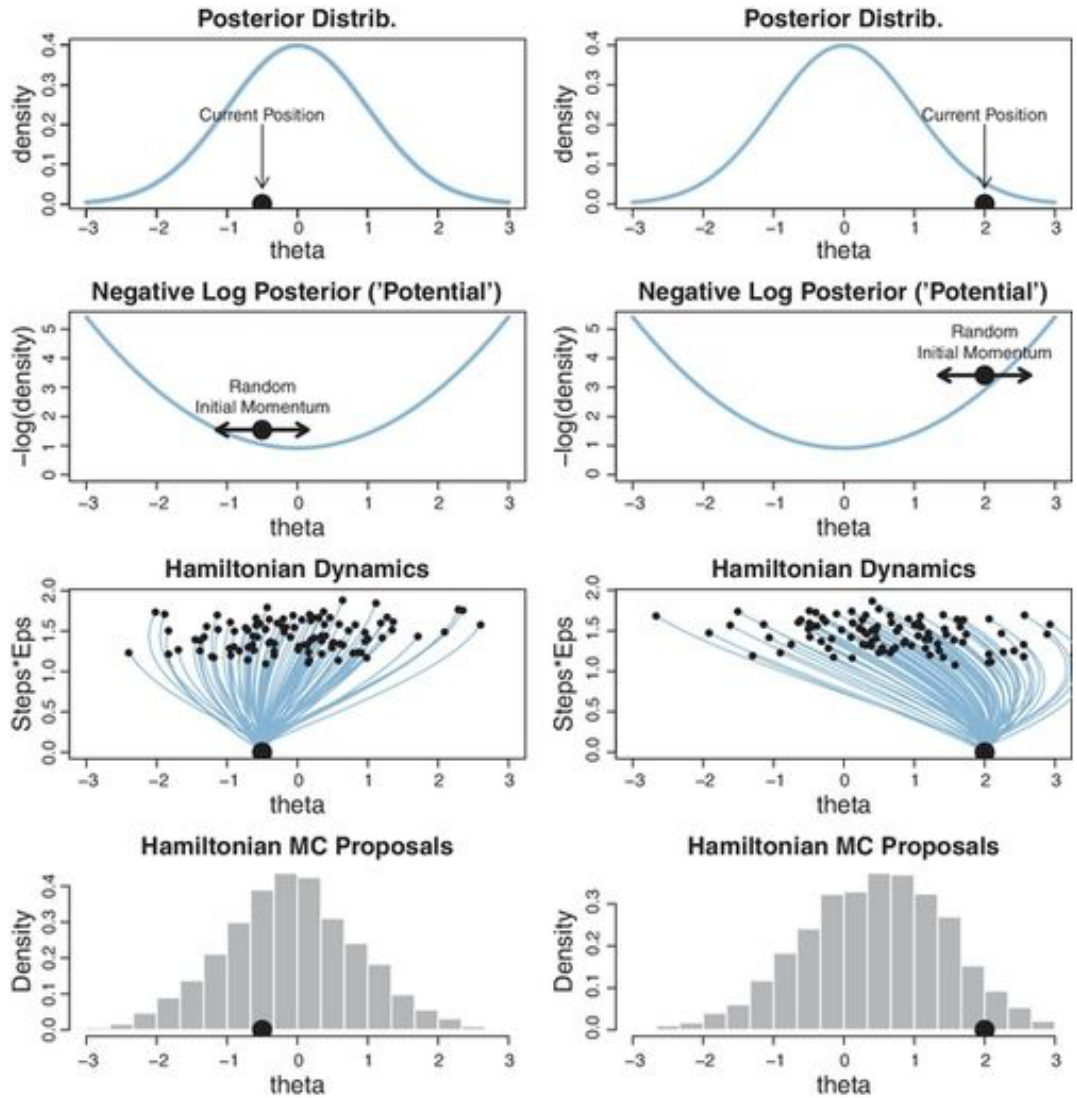


Figure 2:

Today's Focus

– normal approximations to posteriors (`LearnBayes` and `rethinking` packages)

– use of simulation to approximate posteriors and predictive distributions

– JAGS (Gibbs and Metropolis sampling)

– STAN (Hamiltonian sampling)

Regression

Jim Albert

July 2018

Birthweight regression study

– Data: collect gestational age, gender, and birthweight from 24 babies

– Data loaded from LearnBayes package

– Interested in relationship between gestational age and birthweight

– Want to predict birthweight given particular values of gestational age

Load data

```
library(LearnBayes)
head(birthweight)
```

```
##   age gender weight
## 1  40      0   2968
## 2  38      0   2795
## 3  40      0   3163
## 4  35      0   2925
## 5  36      0   2625
## 6  37      0   2847
```

Regression model

– (Sampling) Heights $y_1, ..., y_n$ are independent

$$y_j \sim N(\beta_0 + \beta_1 x_j, \sigma)$$

– (Prior)

$$(\beta_0, \beta_1, \sigma) \sim g()$$

– Think of weakly informative choice for g

Standardizing covariate and response

– Helpful to standardize both response and covariate

$$y_j^* = \frac{y_j - \bar{y}}{s_y}$$

$$x_j^* = \frac{x_j - \bar{x}}{s_x}$$

– Helps in thinking about prior and interpretion

– mean $E(y_j^*) = \beta_0 + \beta_1 x_j^*$

Choosing a prior

– What information would one have about regression?

– β_0 will be close to 0

– β_1 represents correlation between covariate and response

– certainly think $\beta_1 > 0$

Using LearnBayes package

– Write a function to compute the logarithm of the posterior

Arguments are

– parameter vector

$\theta = (\beta_0, \beta_1, \log \sigma)$

– data (data frame with variables `s_age` and `s_weight`)

Expression for posterior

- Likelihood is

$$L(\beta_0, \beta_1, \sigma) = \prod f_N(y_j; \beta_0 + \beta_1 x_j, \sigma)$$

- Posterior is proportional to

$$L(\beta_0, \beta_1, \sigma) g(\beta_0, \beta_1, \sigma)$$

where $g()$ is my prior

Write function to compute log posterior

– Here my prior on $(\beta_0, \beta_1, \log \sigma)$ is uniform (why?)

```
birthweight$s_age <- scale(birthweight$age)
birthweight$s_weight <- scale(birthweight$weight)
logpost <- function(theta, data){
  a <- theta[1]
  b <- theta[2]
  sigma <- exp(theta[3])
  sum(dnorm(data$s_weight,
            mean = a + b * data$s_age,
                       sd = sigma, log=TRUE))
}
```

Obtaining the posterior mode (LearnBayes)

- The `laplace` function in `LearnBayes` finds the posterior mode
- Inputs are `logpost`, starting guess at mode, and data

```
laplace(logpost, c(0, 0, 0), birthweight)$mode
```

```
## [1] -0.000116518  0.674379519 -0.324668140
```

Using rethinking package

– Write description of model by a short script

```
library(rethinking)

flist <- alist(
  s_weight ~ dnorm( mu, sigma ) ,
  mu <- a + b * s_age ,
  a ~ dnorm( 0, 10) ,
  b ~ dnorm(0, 10) ,
  sigma ~ dunif( 0, 20)
)
```

Find the normal approximation to posterior

– `map` function in `rethinking` package

– `precis` function finds the posterior means, standard deviations, and quantiles

```
m3 <- rethinking::map(flist, data=birthweight)
precis(m3)
```

```
##         Mean StdDev  5.5% 94.5%
## a       0.00   0.15 -0.24  0.24
## b       0.67   0.15  0.43  0.92
## sigma 0.72   0.10  0.56  0.89
```

Extract simulated draws from normal approximation

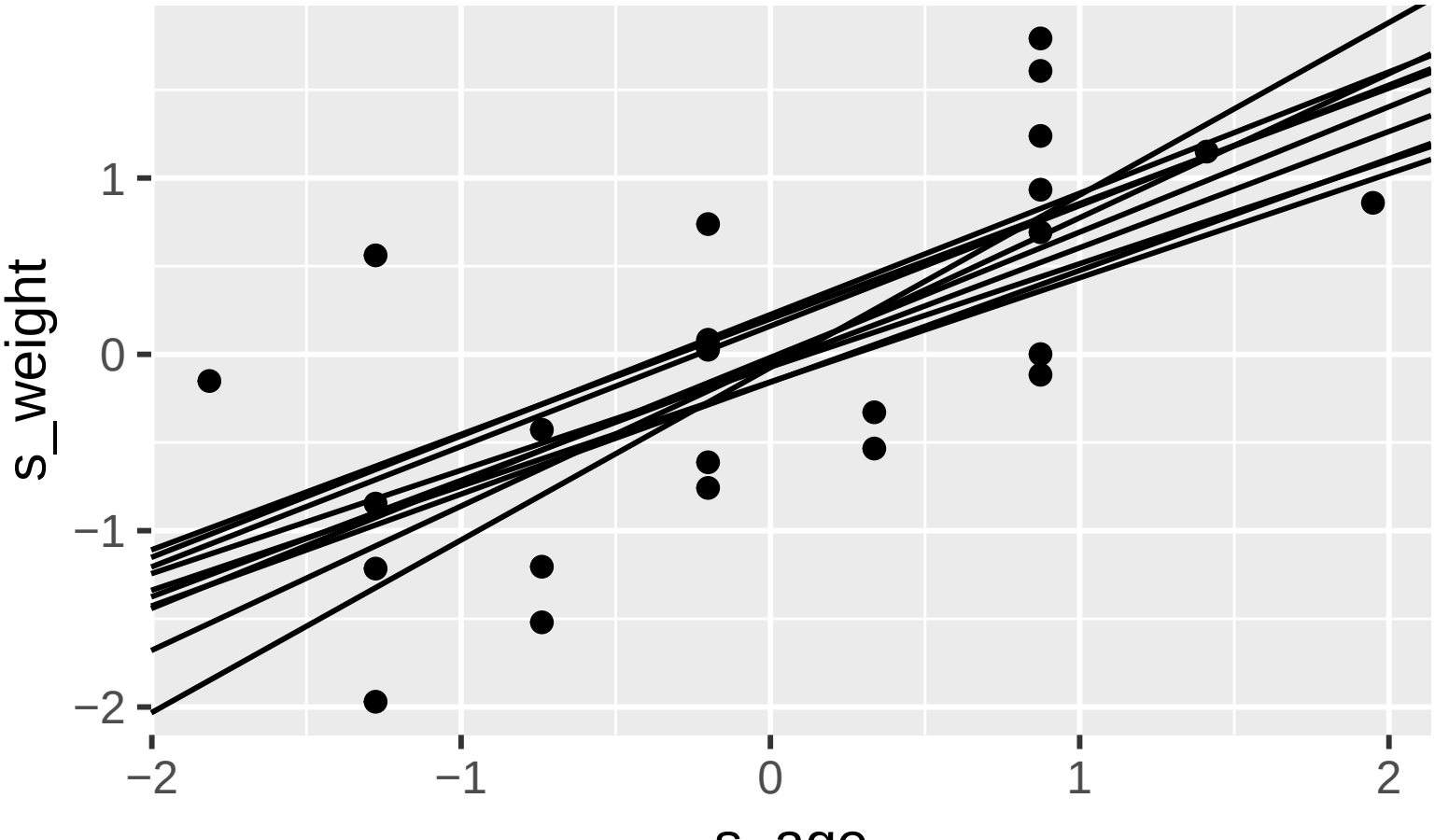
– `map` function in `rethinking` package

```
post <- extract.samples(m3)
head(post)
```

```
##              a          b     sigma
## 1  0.16083915 0.6827045 0.5888092
## 2 -0.01800199 0.7110118 0.7134538
## 3 -0.07470528 0.9780813 0.7074324
## 4 -0.15637663 0.5905935 0.6874017
## 5  0.20079573 0.6544093 0.6761567
## 6 -0.05493612 0.6595597 0.8279068
```

Show 10 simulated fits of (β_0, β_1)

```
library(ggplot2)
ggplot(birthweight, aes(s_age, s_weight)) +
  geom_point() +
  geom_abline(data=post[1:10, ],
              aes(intercept=a, slope=b))
```



Posterior of E(y)

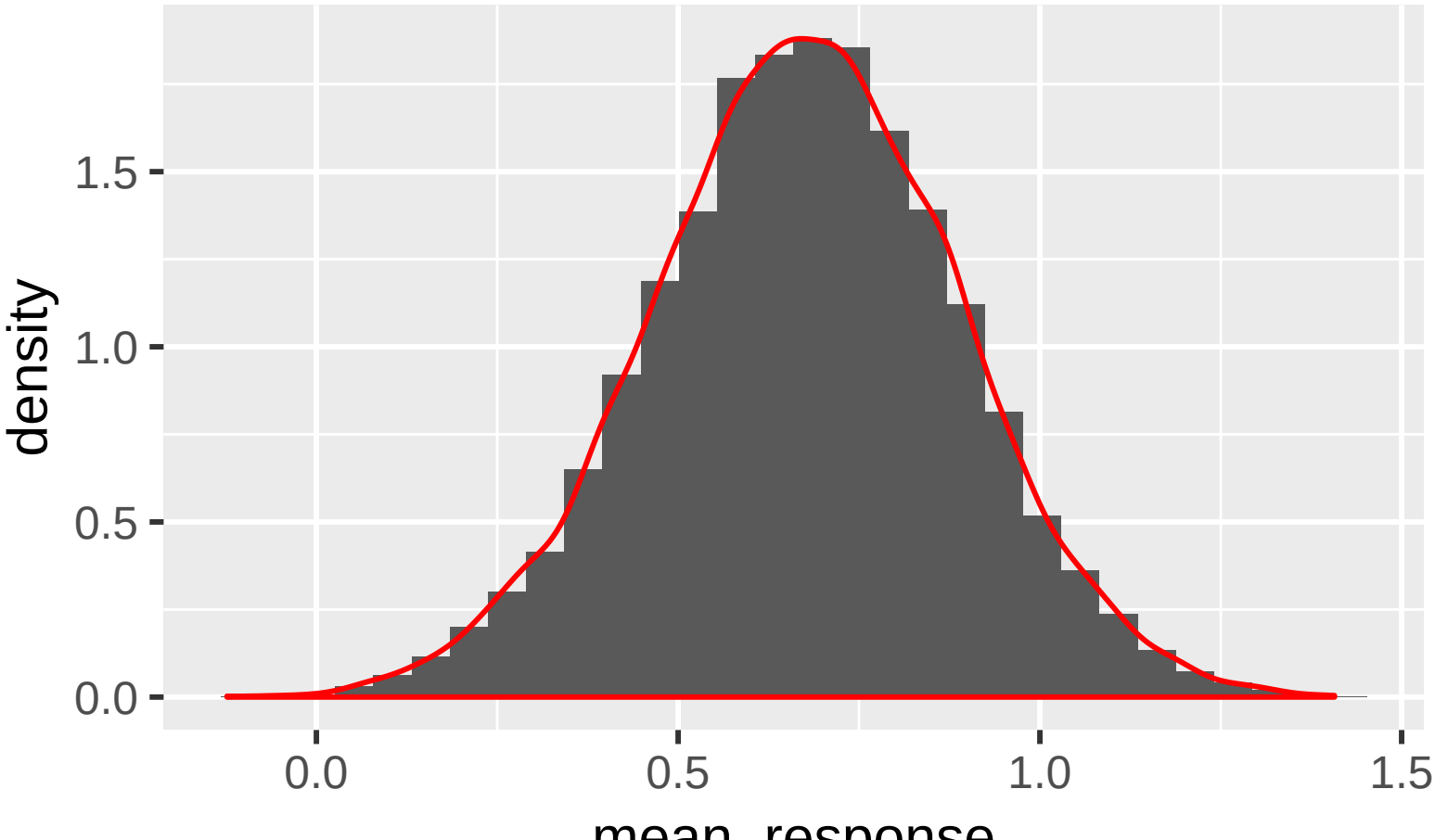
– Suppose you want the posterior for expected response for a specific covariate value

– Posterior of $h(\beta) = \beta_0 + \beta_1 s_{age}$ for specific value of `s_age`

– Just compute this function for each simulated vector from the posterior

Posterior of E(y)

```
s_age <- 1
mean_response <- post[, "a"] + s_age * post[, "b"]
ggplot(data.frame(mean_response), aes(mean_response)) +
  geom_histogram(aes(y=..density..)) +
  geom_density(color="red")
```



Prediction

– Interested in learning about future weight of babies of particular gestational age

– Relevant distribution $f(y_{new}|y)$

– Simulate by (1) simulating values of β_0, β_1, σ from posterior and (2) simulating value of y_{new} from (normal) sampling model

– In `rethinking` package, use function `sim`

Illustrate prediction

- Define a data frame of values of `s_age`
- Use `sim` function with inputs model fit and data frame

```
data_new <- data.frame(s_age = c(-1, 0, 1))
pred <- sim(m3, data_new)
```

```
## [ 100 / 1000 ]
[ 200 / 1000 ]
[ 300 / 1000 ]
[ 400 / 1000 ]
[ 500 / 1000 ]
[ 600 / 1000 ]
[ 700 / 1000 ]
[ 800 / 1000 ]
[ 900 / 1000 ]
[ 1000 / 1000 ]
```

Prediction intervals

- Output of `sim` is matrix of simulated predictions, each column corresponds to one covariate value
- Summarize this matrix using `apply` and `quantile` to obtain 80% prediction intervals

```
apply(pred, 2, quantile, c(.1, .9))
```

```
##            [,1]       [,2]       [,3]
## 10% -1.5955363 -0.9406548 -0.3182419
## 90%  0.2554911  0.9425076  1.5903297
```

Exercises

Suppose you collect the weight and mileage for a group of 2016 model cars.

Interested in fitting the model

$$y_i \sim N(\beta_0 + \beta_1 x_i, \sigma)$$

where x_i and y_i are standardized weights and mileages

1. Construct a reasonable weakly informative prior on $(\beta_0, \beta_1, \sigma)$
2. Suppose you are concerned that your prior is too informative in that the posterior is heavily influenced by the prior. What could you try to address this concern?
3. Describe a general strategy for simulating from the posterior of all parameters

Exercises (continued)

4. Suppose one is interested in two problems: (P1) estimating the mean mileage of cars of average weight and (P2) learning about the actual mileage of a car of average weight. What are the relevant distributions for addressing problems (P1) and (P2)?
5. For some reason, suppose you are interested in obtaining a 90% interval estimate for the standardized slope β_1/σ. How would you do this based on simulating a sample from the joing posterior?

Attendance Data

Jim Albert

July 2018

Attendance study

– collected Cleveland Indians (baseball) attendance for 81 home games in 2016 season

– response is fraction of capacity

– input: game is Weekend/Weekday

– input: period of season (first, second, third)

```
library(readr)
d <- read_csv("tribe2016.csv")
```

Beta regression model

– Suitable for modeling response data which are rates or proportions

– $y_i \sim Beta$ with shape parameters a and b

– $a = \mu\phi,\ b = (1-\mu)\phi$

(μ is the mean, ϕ is a precision parameter)

– $logit\mu = \alpha + x\beta$

(logistic model on the means)

Traditional beta regression

- using function betareg in betareg package

```
library(betareg)
fit <- betareg(fraction ~ Weekend + Period, data=d,
               link="logit")
```

Output from betareg

```
summary(fit)
```

```
##
## Call:
## betareg(formula = fraction ~ Weekend + Period, data = d
##
## Standardized weighted residuals 2:
##     Min       1Q  Median      3Q     Max
## -1.6924 -0.5986 -0.1458  0.2937  5.9356
##
## Coefficients (mean model with logit link):
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -0.6531     0.1573  -4.151 3.31e-05 ***
## Weekendyes     0.9521     0.1612   5.907 3.48e-09 ***
## PeriodSecond   1.0852     0.1982   5.474 4.39e-08 ***
## PeriodThird    0.4797     0.1918   2.501   0.0124 *
##
## Phi coefficients (precision model with identity link):
```

Using STAN in rstanarm package

- Function `stan_betareg` implements MCMC sampling of a Bayesian beta regression model
- Same model syntax as `betareg`
- Can specify a variety of priors (we'll use default one here)

```
library(rstanarm)
fit2 <- stan_betareg(fraction ~ Weekend + Period, data=d,
                     link="logit")
```

```
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
##
## Gradient evaluation took 0.00193 seconds
## 1000 transitions using 10 leapfrog steps per transition
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
```

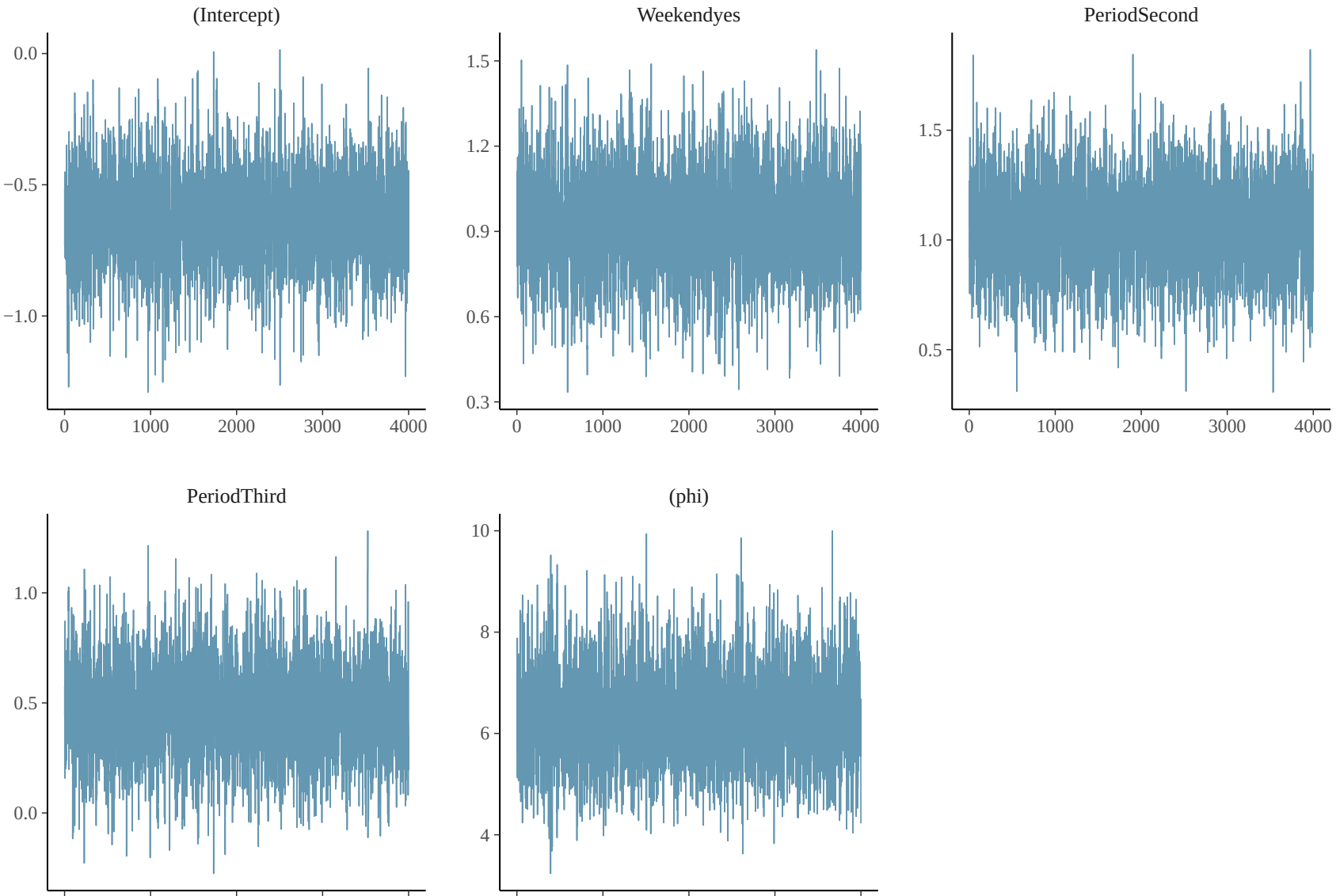
What priors are used?

```
prior_summary(fit2)
```

```
## Priors for model 'fit2'
## ------
## Intercept (after predictors centered)
##  ~ normal(location = 0, scale = 10)
##
## Coefficients
##  ~ normal(location = [0,0,0], scale = [2.5,2.5,2.5])
##
## Auxiliary (phi)
##  ~ exponential(rate = 1)
## ------
## See help('prior_summary.stanreg') for more details
```

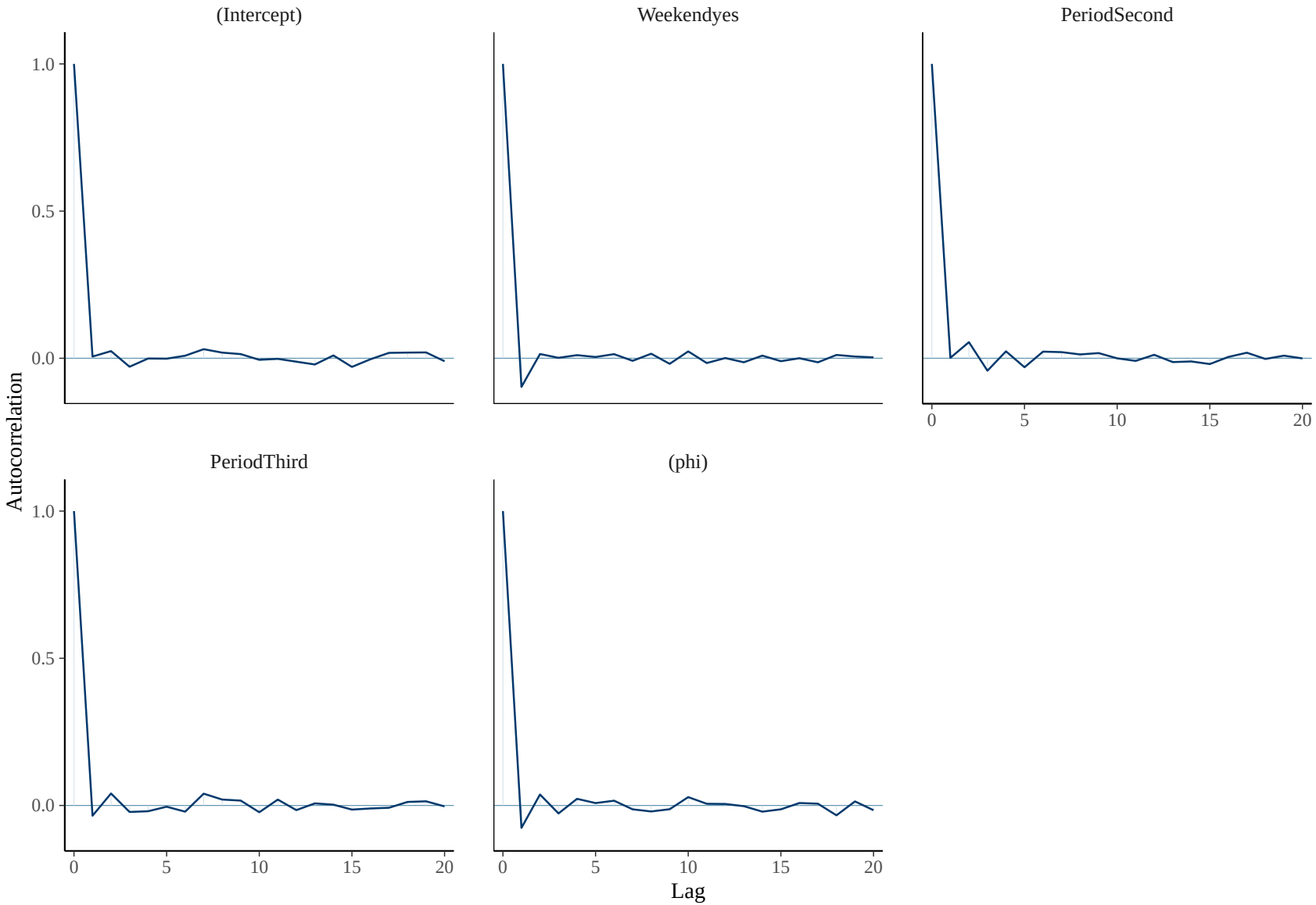
MCMC diagnostics – trace plots

```
library(bayesplot)
mcmc_trace(as.matrix(fit2))
```



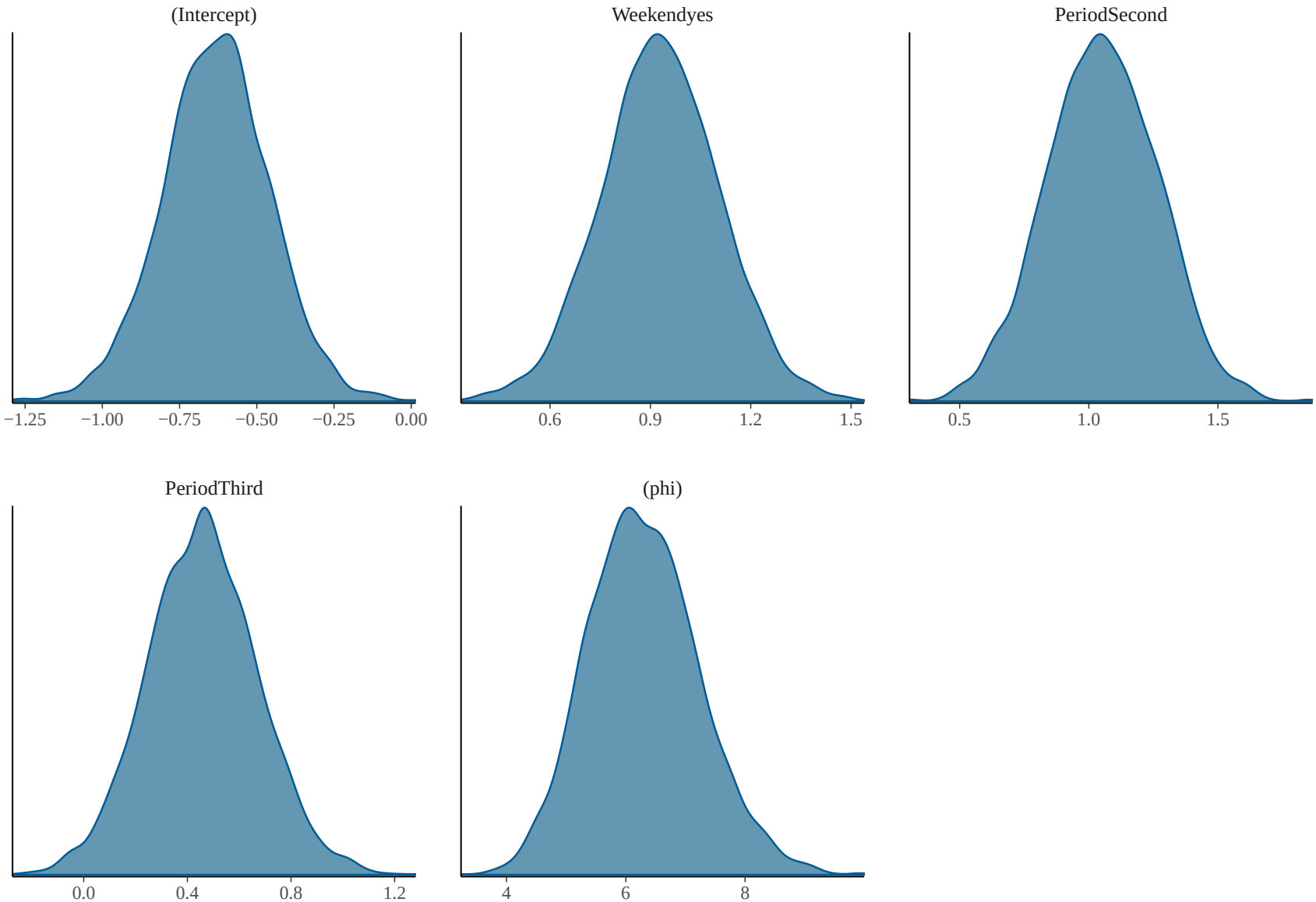
Autocorrelation plots

```
mcmc_acf(as.matrix(fit2))
```



Density plots for all parameters

```
mcmc_dens(as.matrix(fit2))
```



Posterior interval estimates

```
posterior_interval(fit2)
```

```
##                        5%        95%
## (Intercept)  -0.9201864 -0.3483581
## Weekendyes    0.6432198  1.2209528
## PeriodSecond  0.6976143  1.3991345
## PeriodThird   0.1135333  0.8071830
## (phi)         4.8578388  7.8803825
```

Matrix of simulated draws from posterior

```
posterior_sims <- as.matrix(fit2)
head(posterior_sims)
```

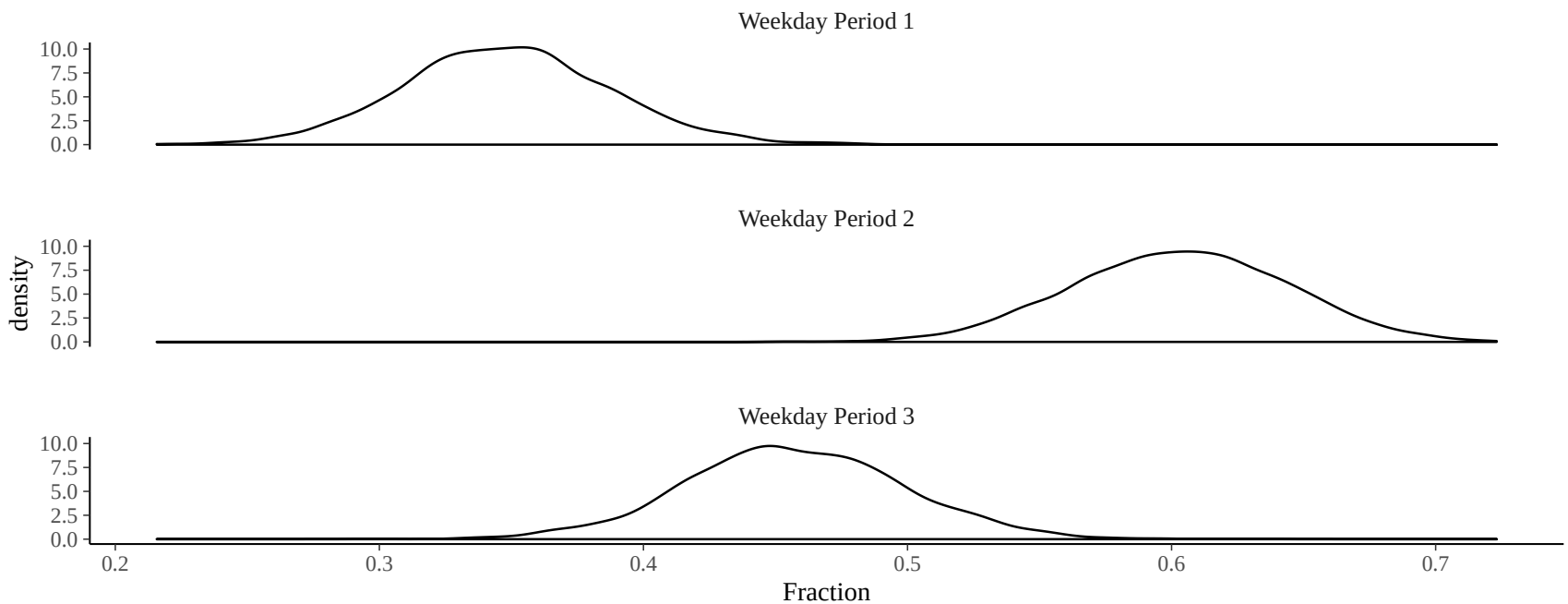
```
##           parameters
## iterations (Intercept) Weekendyes PeriodSecond PeriodTh
##       [1,]  -0.7335286  0.8846116    1.1226532   0.5771
##       [2,]  -0.5095653  1.0137855    1.0647633   0.4390
##       [3,]  -0.7369211  1.1109959    1.1243881   0.5241
##       [4,]  -0.7162241  1.0109144    1.2689236   0.6402
##       [5,]  -0.4509849  1.0507150    0.7555692   0.1578
##       [6,]  -0.7595283  0.7757580    1.1355195   0.8715
```

Interested in expected attendance, weekdays, each period

```
library(arm)
d1 <- data.frame(Label="Weekday Period 1",
      Fraction=invlogit(posterior_sims[, "(Intercept)"]))
d2 <- data.frame(Label="Weekday Period 2",
      Fraction=invlogit(posterior_sims[, "(Intercept)"] +
                        posterior_sims[, "PeriodSecond"]))
d3 <- data.frame(Label="Weekday Period 3",
      Fraction=invlogit(posterior_sims[, "(Intercept)"] +
                        posterior_sims[, "PeriodThird"]))
```

Posteriors of expected attendance, weekday, each period

```
library(ggplot2)
ggplot(rbind(d1, d2, d3), aes(Fraction)) +
  geom_density() + facet_wrap(~ Label, ncol=1)
```



Nice graphical interface

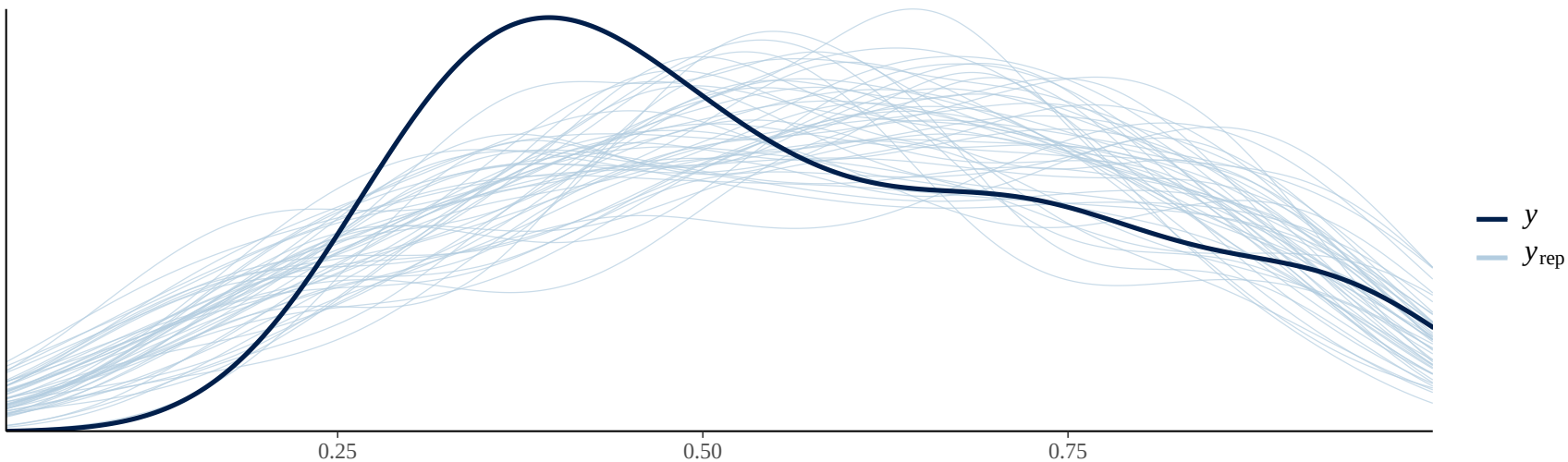
– Launches graphical interface for diagnostics/summaries

```
launch_shinystan(fit2)
```

Commands for posterior predictive checking

– shows density plot of response and some replicated posterior predictive data

```
pp_check(fit2)
```



Obtain replicated simulations from posterior predictive distribution

```
ynew <- posterior_predict(fit2)
```

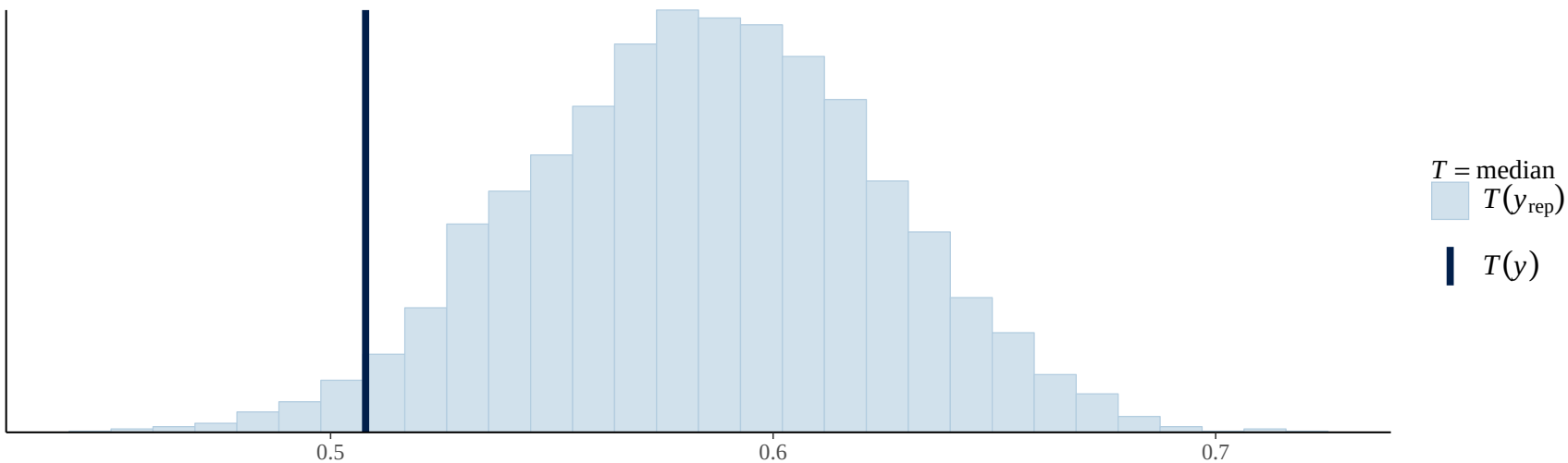
– need `ynew` to implement the following posterior predictive checks

– compare $T(y_{rep})$ with T_{obs} using specific checking function T

Posterior predictive check

- using “median” as the test statistic

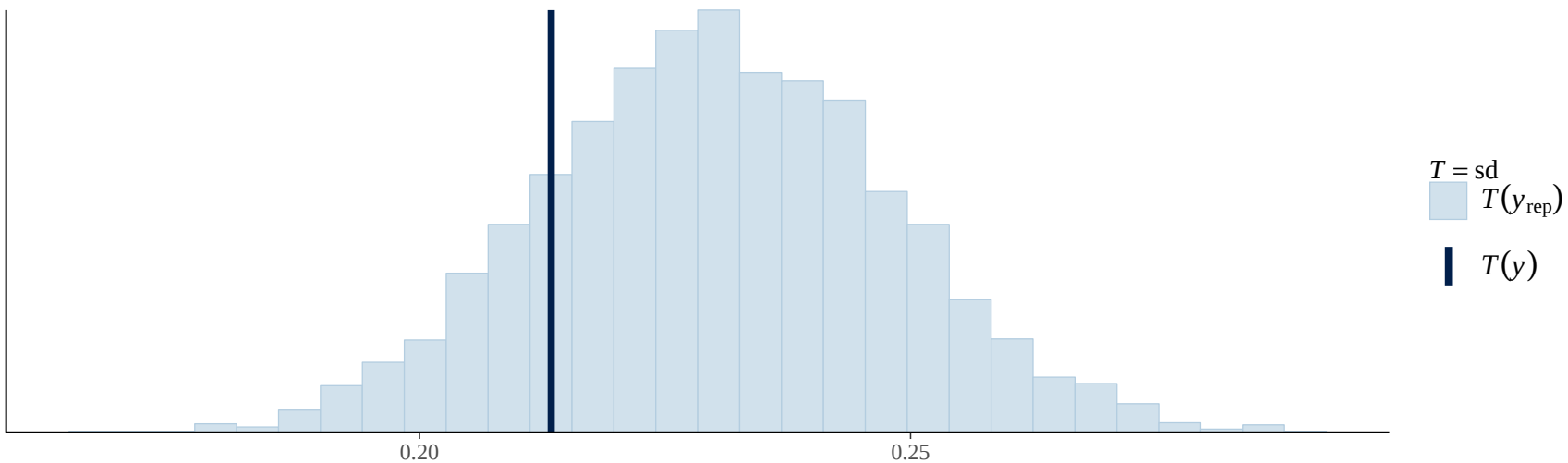
```
ppc_stat(d$fraction, ynew, stat="median")
```



Posterior predictive check

- using “sd” as the test statistic

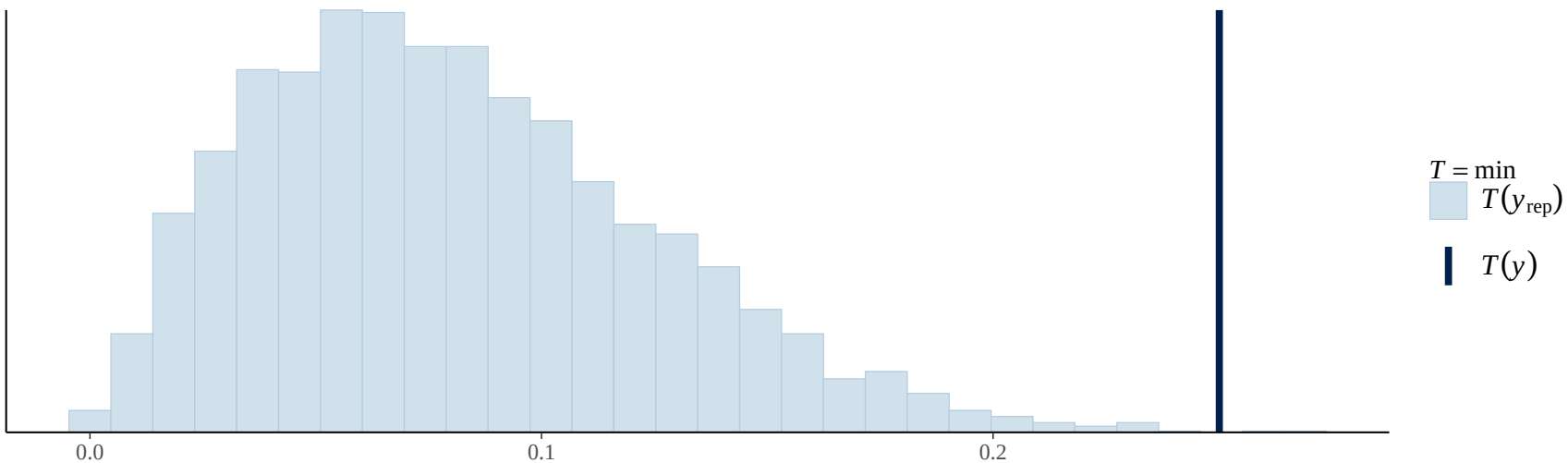
```
ppc_stat(d$fraction, ynew, stat="sd")
```



Posterior predictive check

- using “min” as the test statistic

```
ppc_stat(d$fraction, ynew, stat="min")
```



Exercises

1. Is the beta regression a reasonable model for this attendance data?
2. Assuming the answer to question 1 is "no", what models might you try next?
3. What is the advantage of a Bayesian fit compared to a traditional (ML) fit in this setting?
4. What alternative priors might you try?

Exercises (continued)

5. (Logistic Modeling)

Suppose you are interested in looking at the relationship between insecticide dose and kill (1 or 0) for 10 insects. Here is R code setting up the data and implementing a traditional logistic fit.

```
dose <- 1:10
kill <- c(rep(0, 6), rep(1, 4))
df <- data.frame(dose, kill)
fit <- glm(kill ~ dose,
    family=binomial, data=df)
```

– This code will produce a warning? What is going on? (Look at the parameter estimates and standard errors.)

– Run this model using the `stan_glm` function. Compare the parameter estimates for the two fits. Why are they so different?

Worship Data

Jim Albert

July 2018

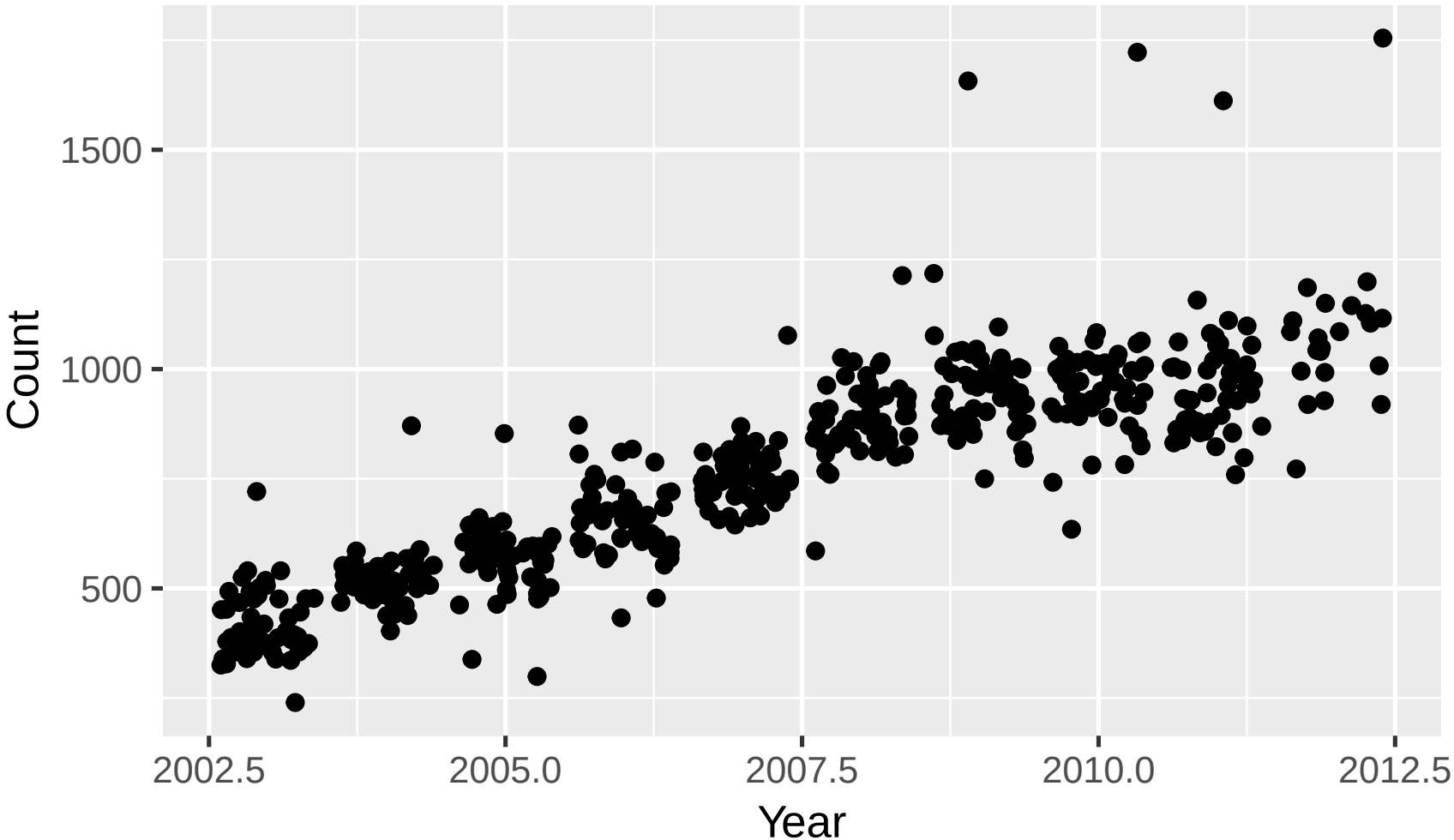
Worship Data Example

– Collect weekly attendance data from a local church for many years

– Interested in understanding pattern of growth and predict future attendance

Read the data

```
d <- read.csv("http://personal.bgsu.edu/~albert/data/gatewa
library(tidyverse)
ggplot(d, aes(Year, Count)) + geom_jitter()
```

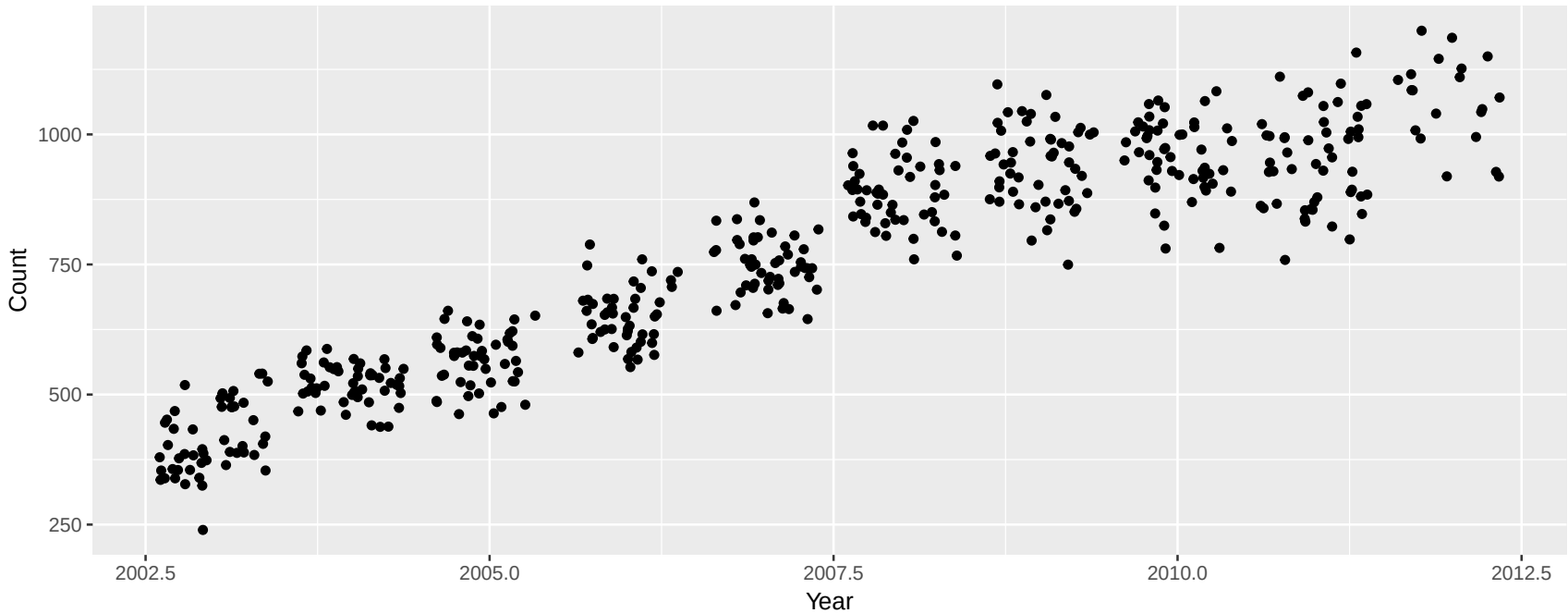


Remove the outliers (Easter, etc)

```
S <- summarize(group_by(d, Year),
               M=median(Count),
               QL=quantile(Count, .25),
               QU=quantile(Count, .75),
               Step=1.5 * (QU - QL),
               Fence_Lo= QL - Step,
               Fence_Hi= QU + Step)
d <- inner_join(d, S)
d2 <- filter(d, Count > Fence_Lo, Count < Fence_Hi)
```

New plot with outliers removed

```
ggplot(d2, aes(Year, Count)) +
  geom_jitter()
```



Bayesian model:

1. Worship counts y are Poisson(λ) where means satisfy log-linear model

$$\log \lambda = a + b \times year$$

2. Place weakly-informative prior on (a, b)

$$a \sim N(0, 100), b \sim N(0, 100)$$

Define a new variable

- `year_number` is number of years after 2002

```
d2$year_number <- d2$Year - 2002
```

Normal approximation to posterior

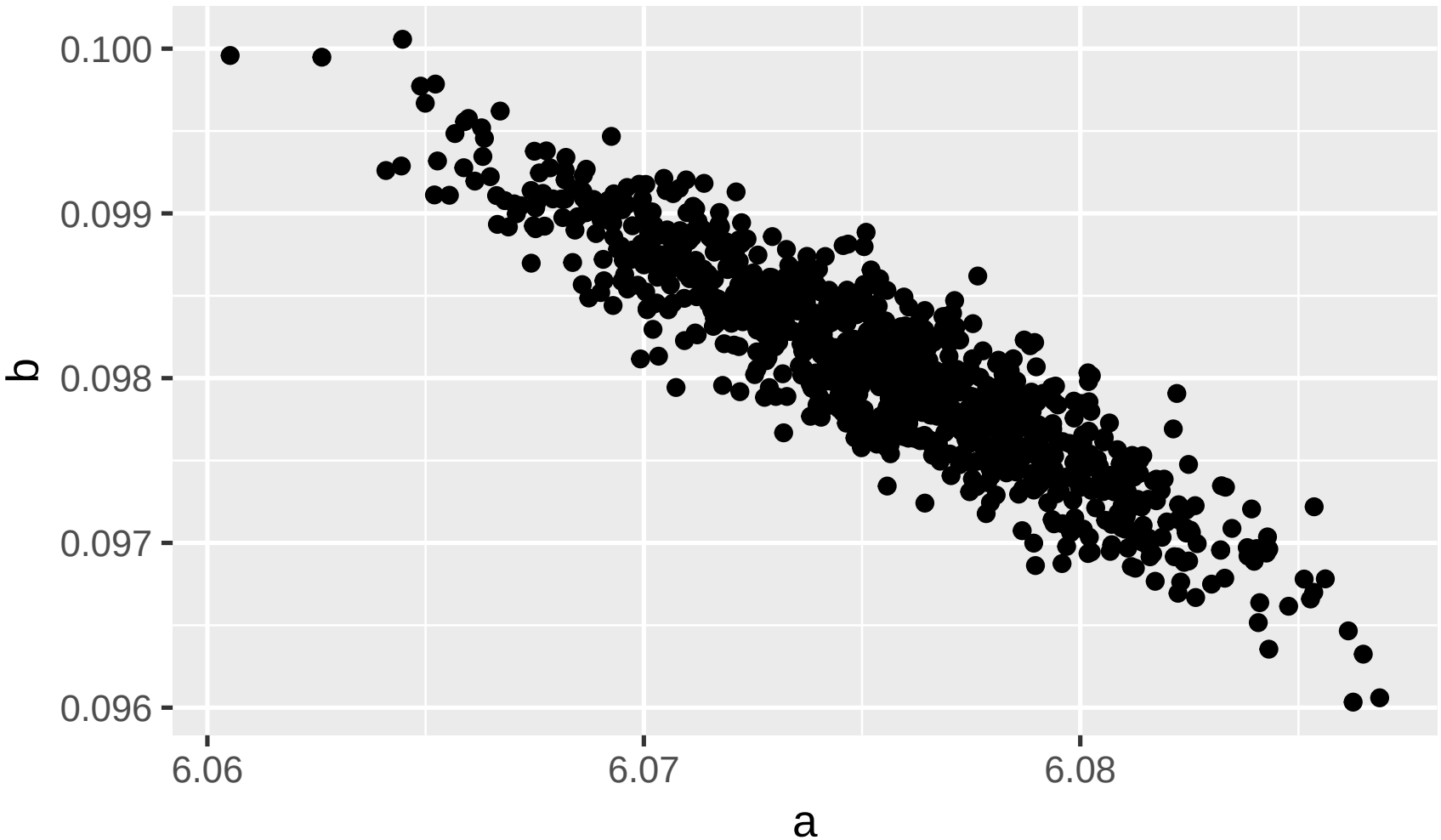
– Use `rethinking` package

– define this Bayesian model and find a normal approximation to the posterior.

```
library(rethinking)
m1 <- map(
  alist(
    Count ~ dpois( lambda ),
    log(lambda) <- a + b * year_number,
    a ~ dnorm(0, 100),
    b ~ dnorm(0, 100)
  ), data=d2, start=list(a=6, b=0.1)
)
```

Simulate and plot 1000 draws from the posterior

```
sim_m1 <- extract.samples(m1, n = 1000)
ggplot(sim_m1, aes(a, b)) + geom_point()
```



Posterior summaries of each parameter

```
precis(m1, digits=4)
```

```
##      Mean StdDev  5.5%  94.5%
## a 6.0757 0.0041 6.069 6.0823
## b 0.0980 0.0006 0.097 0.0990
```

Summarizing worship growth

– For a particular year number, interested in posterior distribution of expected count:

$$E(Y) = \exp(a + b\ year)$$

– Wish to summarize the posterior of $E(y)$ for several values of year

– Summarize simulated draws of $\exp(a + b\ year)$

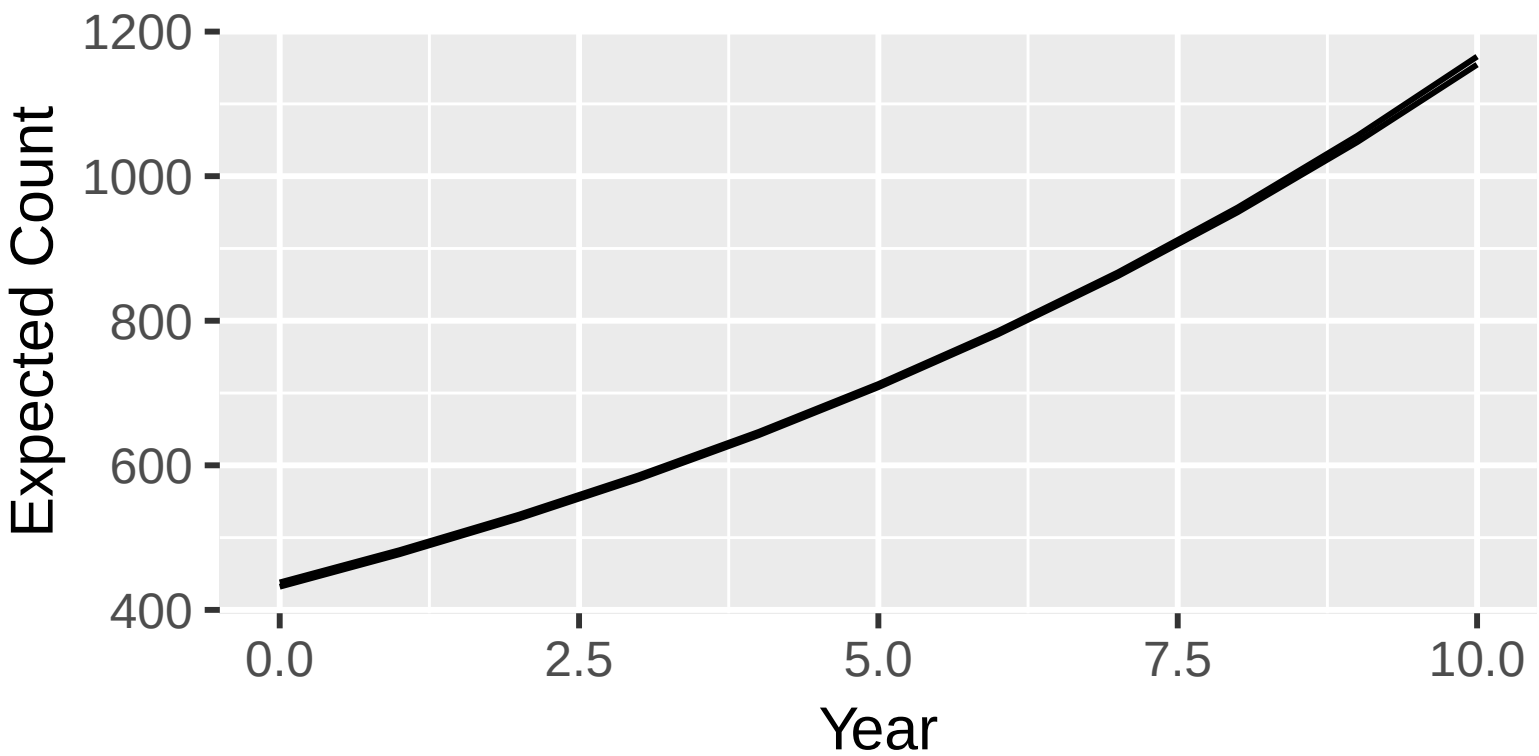
Posterior of expected worship count for each year

```
post_lambda <- function(year_no){
  lp <- sim_m1[, "a"] + year_no * sim_m1[, "b"]
  Q <- quantile(exp(lp), c(0.05, 0.95))
  data.frame(Year = year_no, L_05 = Q[1], L_95 = Q[2])
}
```

Graph the summaries of expected count

```
OUT <- do.call("rbind",
                lapply(0:10, post_lambda))
```

```
ggplot(OUT, aes(Year, L_05)) +
  geom_line() +
  geom_line(data=OUT, aes(Year, L_95)) +
  ylab("Expected Count")
```



Model checking

- Idea: Does the observed data resemble "replicated data" predicted from the model?
- Simulate data from the model (posterior predictive distribution)
- Use some checking function $T(y_{rep})$ (here we use the standard deviation as our function)
- Plot predictive distribution of $T(y_{rep})$ – how does $T(y_{obs})$ compare?

Posterior predictive checking

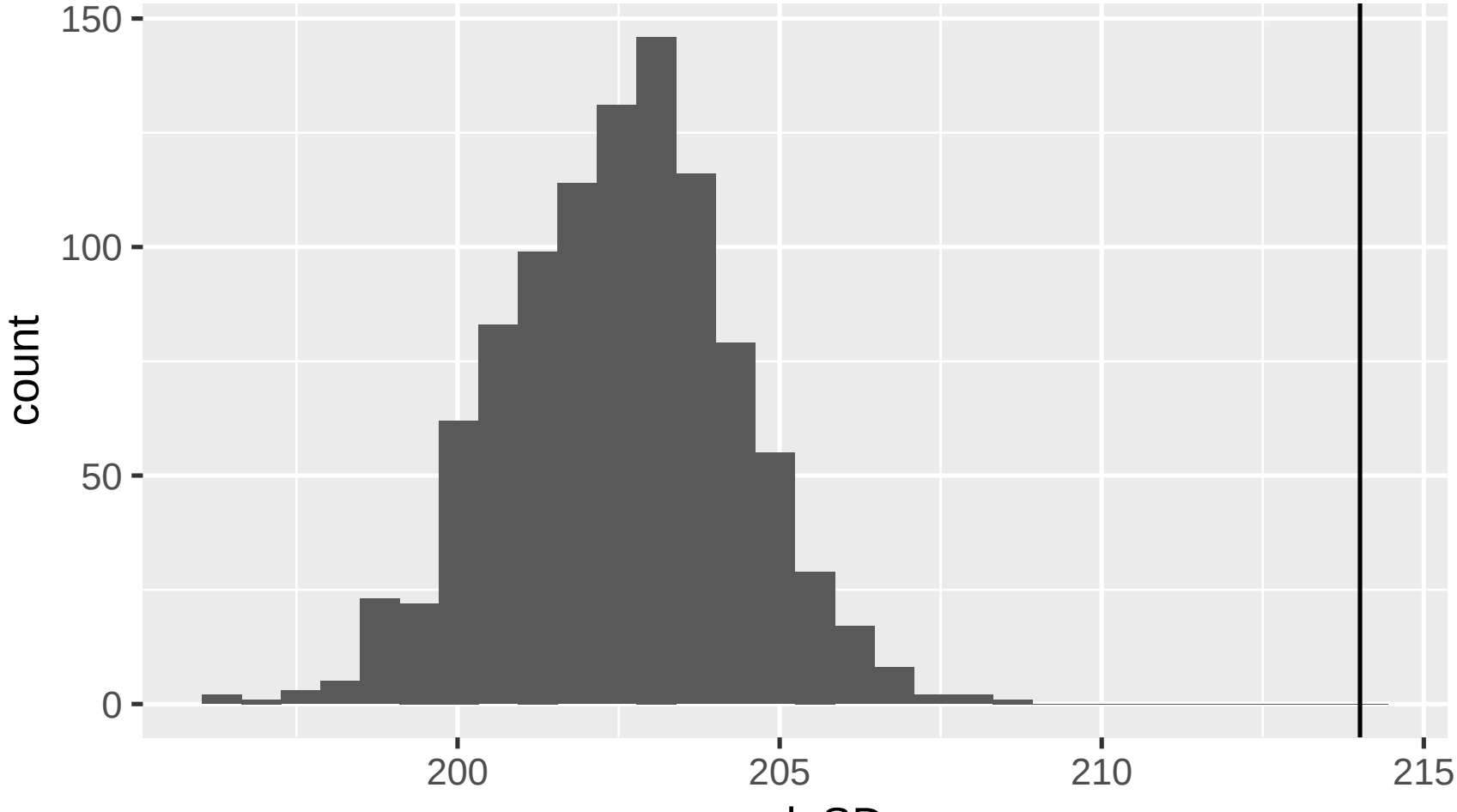
- Simulate vector of λ_j and then values of y
- Compute standard deviation of each sample

```
replicated_data <- function(j){
  lambda <- sim_m1[j, "a"] + sim_m1[j, "b"] *
         d2$year_number
  ys <- rpois(length(lambda), exp(lambda))
  sd(ys)
}
pred_SD <- map_dbl(1:1000, replicated_data)
```

Compare observed SD with predictive distribution.

What do we conclude?

```
ggplot(data.frame(pred_SD), aes(pred_SD)) +
  geom_histogram() +
  geom_vline(xintercept = sd(d2$Count))
```



Different Sampling Model

– Data is overdispersed

– Use another count distribution that can accomodate the extra-variation

– Try a negative-binomial(p, r)

– Parametrize in terms of the mean λ and a overdispersion parameter

Negative binomial regression

– count response $y \sim NB(p, r)$

– mean $\mu = \frac{(1-p)r}{p}$

– variance $\mu + \mu^2/r$

(r is overdispersion parameter)

– log-linear model $\log \mu = a + byear$

– prior on (a, b, r)

Use JAGS

– Write a script defining the Bayesian model

– Vague priors on β and overdispersion parameter r

– Inputs to JAGS are (1) model script, (2) data, (3) initial values for MCMC sampling

Use JAGS to fit a negative binomial model.

```
modelString = "
model{
for(i in 1:n){
mu[i] <- beta[1] +  beta[2] * year[i]
lambda[i] <- exp(mu[i])
p[i] <- r / (r + lambda[i])
y[i] ~ dnegbin(p[i], r)
}
beta[1:2] ~ dmnorm(b0[1:2], B0[ , ])
r ~ dunif(0, 200)
}"
writeLines(modelString, con="negbin1.bug")
```

JAGS Inputs

```
forJags <- list(n=dim(d2)[1],
                year = d2$year_number,
                y = d$Count,
                b0 = rep(0, 2),
                B0 = diag(.0001, 2))
```

```
inits <- list(list(beta=rep(0, 2),
                   r=1))
```

Running JAGS (MCMC Warmup)

```
require(rjags)
foo <- jags.model(file="negbin1.bug",
                  data=forJags,
                  inits=inits,
                  n.adapt = 5000)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 461
##    Unobserved stochastic nodes: 2
##    Total graph size: 1010
##
## Initializing model
```

Running JAGS (More warmup and collect draws)

```
update(foo,5000)
out <- coda.samples(foo,
                    variable.names=c("beta", "r"),
                    n.iter=5000)
```

Some Posterior Summarizes

```
summary(out)
```

```
##
## Iterations = 10001:15000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 5000
##
## 1. Empirical mean and standard deviation for each variab
##    plus standard error of the mean:
##
##              Mean       SD  Naive SE Time-series SE
## beta[1]   6.0435 0.013918 1.968e-04      0.0020931
## beta[2]   0.1017 0.002424 3.429e-05      0.0003019
## r        46.5681 3.231921 4.571e-02      0.0579909
##
## 2. Quantiles for each variable:
##
```

Exercises

1. How do the Poisson and Negative-Binomial Fits compare?
2. Suppose we want to predict a single worship attendance next year? How would we do this?
3. For Negative-Binomial fit, what type of posterior predictive checks should we try?
4. How would Negative-Binomial fit differ from a frequentist fit?
5. What are other covariates we could use to help explain variation in worship attendance?

Home Runs

Jim Albert

July 2018

Home Run Study

- Interested in learning about home run rates of baseball players
- Collect data for part of the 2017 season

Read in Baseball Data

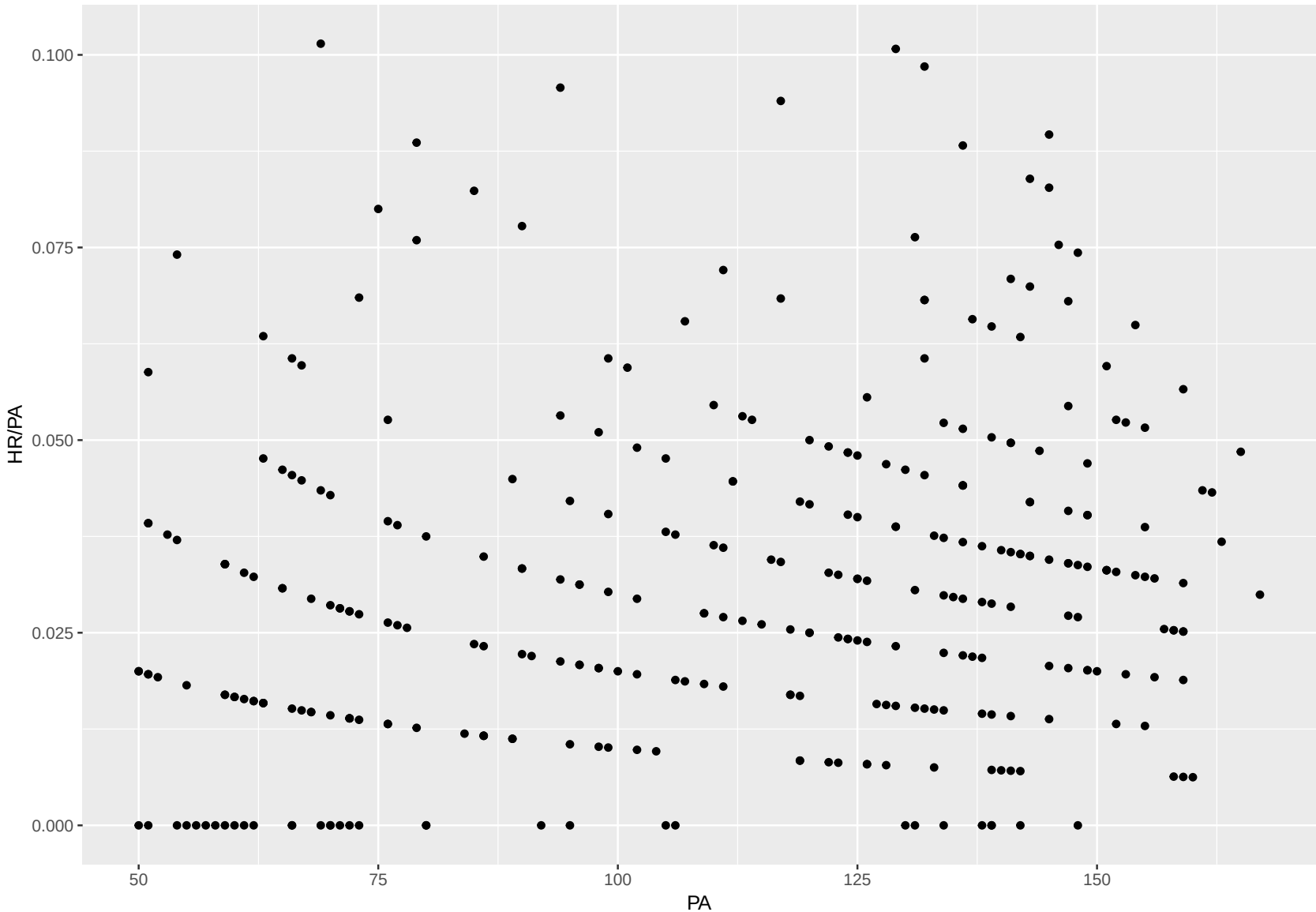
– From Fangraphs, read in batting data for players with at least 10 plate appearances

– Filter to only include batters with at least 50 PA

```
library(tidyverse)
d2017 <- filter(read_csv("may13.csv"), PA >= 50)
```

Graph Home Run Rates Against Plate Appearances

```
ggplot(d2017, aes(PA, HR / PA)) + geom_point()
```



Basic Sampling Model

– Home run count y_j is Poisson($PA_j \lambda_j$) for 323 hitters

– Want to estimate $\lambda_1, ..., \lambda_{323}$

No-Pool Model

- No relationship between the rates
- Estimate λ_j using the individual counts
- $\hat{\lambda}_j = y_j / PA_j$

Pooled Model

- Assume that $\lambda_1 = ... = \lambda_{323}$
- Estimate common rate by pooling the data
- $\hat{\lambda}_j = \sum y_j / \sum PA_j$

Both Models are Unsatisfactory

– Individual count estimates can be poor, especially with small sample sizes and sparse data

– Pooled estimates ignore the differences between the true rates $\lambda_1, ..., \lambda_{323}$

– Need a compromise model

Partial Pooling Model

– multilevel model

– assume that the λ_j have a common prior $g()$ with unknown parameters θ

– place a weakly-informative prior on θ

– estimate parameters from the data

Express as Poisson log-linear Models

- $y_j \sim Poisson(\lambda_j)$
- $\log \lambda_j = \log PA_j + \beta_0 + \gamma_j$
- Here $\log PA_j$ is an offset.

Fit the No-Pool Model

- Use `glm` with `Player` as a covariate
- Remember estimates are on log scale – exponentiate to get estimates at λ_j

```
fit_individual <- glm(HR ~ Player + offset(log(PA)),
                      family=poisson,
                      data=d2017)
d2017$rate_estimates <- exp(fit_individual$linear.predictor
                      d2017$PA
```

Fit the Pooled Model using glm

- Use `glm` with only a constant term

```
fit_pool <- glm(HR ~ 1 + offset(log(PA)),
                family=poisson,
                data=d2017)
exp(coef(fit_pool))
```

```
## (Intercept)
##  0.03247647
```

Partial Pooling Model

- $y_j \sim Poisson(\lambda_j)$
- $\log \lambda_j = \log PA_j + \beta_0 + \gamma_j$
- $\gamma_1, ..., \gamma_N \sim N(0, \sigma)$

Quick Fit of Partial Pooling Model

```
library(lme4)
fit_pp <- glmer(HR ~ (1 | Player) +
                  offset(log(PA)),
                    family=poisson,
                    data=d2017)
```

Quick Fit of Partial Pooling Model

```
fit_pp
```

```
## Generalized linear mixed model fit by maximum likelihood
##   Approximation) [glmerMod]
##  Family: poisson  ( log )
## Formula: HR ~ (1 | Player) + offset(log(PA))
##    Data: d2017
##       AIC       BIC    logLik  deviance  df.resid
## 1399.0705 1406.6258 -697.5353 1395.0705       321
## Random effects:
##  Groups Name        Std.Dev.
##  Player (Intercept) 0.4112
## Number of obs: 323, groups:  Player, 323
## Fixed Effects:
## (Intercept)
##      -3.523
```

Use STAN to Fit the Partial Pool Model

```
library(rstanarm)
fit_partialpool <- stan_glmer(HR ~ (1 | Player) +
                                offset(log(PA)),
                              family=poisson,
                              data=d2017)
```

```
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 1).
##
## Gradient evaluation took 0.000703 seconds
## 1000 transitions using 10 leapfrog steps per transition
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
```

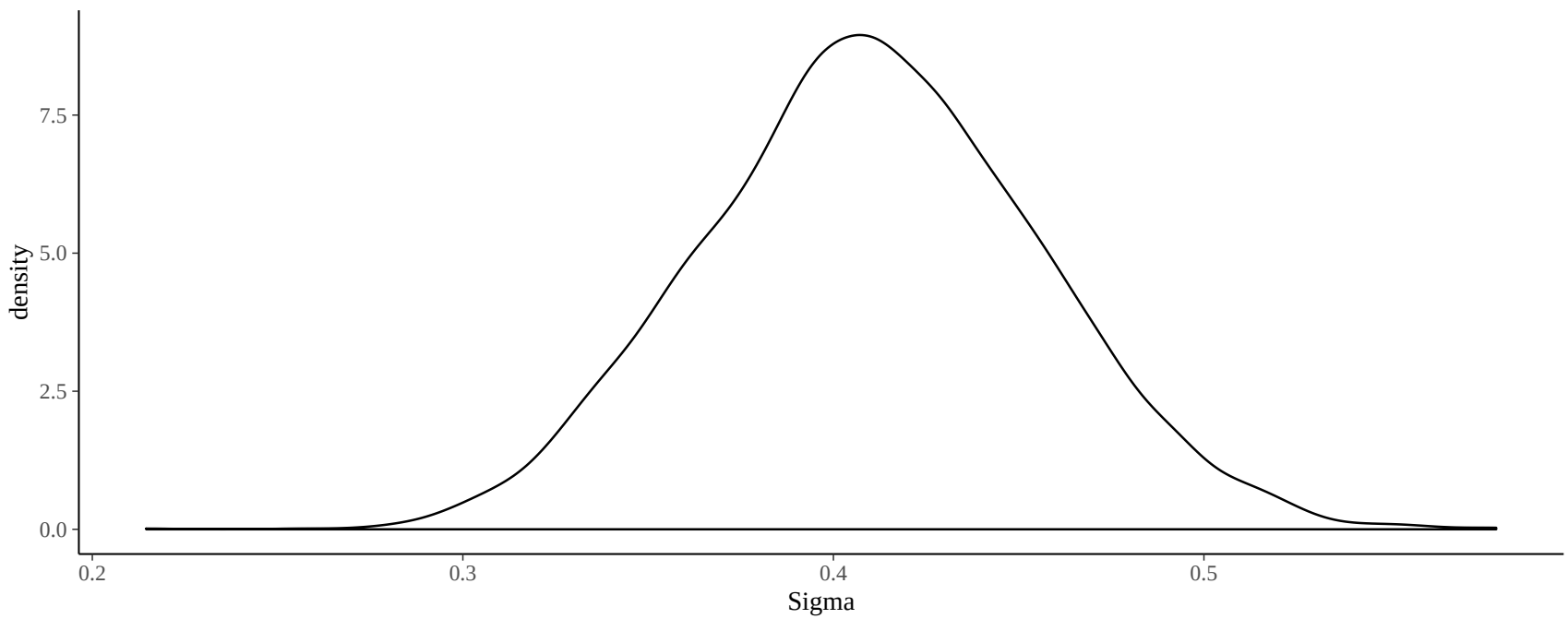
Priors?

```
prior_summary(fit_partialpool)
```

```
## Priors for model 'fit_partialpool'
## ------
## Intercept (after predictors centered)
##  ~ normal(location = 0, scale = 10)
##
## Covariance
##  ~ decov(reg. = 1, conc. = 1, shape = 1, scale = 1)
## ------
## See help('prior_summary.stanreg') for more details
```

Learn about random effects standard deviation

```
posterior <- as.matrix(fit_partialpool)
ggplot(data.frame(Sigma=sqrt(posterior[, 325])),
       aes(Sigma)) +
      geom_density()
```



Obtain Rate Estimates

Exercises

Efron and Morris (1975) demonstrated multilevel modeling using baseball data for 18 players after 45 at-bats. The data is contained in the dataset `bball1970` in the `rstanarm` package. Suppose $p_1, ..., p_{18}$ represent the probabilities of success for these 18 players.

1. Describe the "no-pooling" model for this example.
2. Describe the "pooling" model for this example.
3. What is the intent of a "partial pooling" model for this example?

Exercises (continued)

Here is a multilevel model for this baseball data

$$\log(p_j/(1-p_j)) = \beta_0 + \gamma_j$$

$$\gamma_j \sim N(0, \sigma)$$

Try out the basic fit of this model using the `lme4` package.

```
library(lme4)
library(rstanarm)
fit <- glmer(cbind(Hits, AB - Hits) ~ (1 | Player),
      family=binomial, data=bball1970)
```

Contrast this fit with the corresponding Bayesian fit using the `stan_glmer` function.

Multilevel Modeling

Jim Albert

July 2018

Statistical Rethinking

– *Statistical Rethinking: A Bayesian Course with Examples in R and Stan* by Richard McElreath

– Nice introduction to Bayesian ideas

– Associated R package (rethinking) that provides interface to Stan software

– Use a "coffee shop" illustration of multilevel modeling

How Long Do You Wait at a Coffee Shop?

– One is interested in learning about the pattern of waiting times at a particular coffee shop.

– Suppose the waiting time y is normally distributed

– You believe waiting times are different between the morning the afternoon

– Motivates the model

$$y \sim N(\alpha + \beta * PM, \sigma)$$

– Given a sample of waiting times $\{y_i\}$ can fit model

Several Coffee Shops

- Visit several coffee shops
- Observe waiting times for each shop
- For the jth coffee shop, have model

$$y \sim N(\alpha_j + \beta_j * PM, \sigma)$$

How to Estimate Regressions for J Coffee Shops?

– Separate estimates? (What if you don't have many measurements at one coffee shop?)

– Combined estimates? (Assume that waitings times from the J shops satisfy the same linear model.)

– Estimate by multilevel model (compromise between separate estimates and combined estimates)

Varying-Intercepts, Varying Slopes Model

- Sampling:

$$y_i \sim N(\alpha_{j[i]} + \beta_{j[i]} PM_i, \sigma)$$

- Prior:

Stage 1. $(\alpha_j, \beta_j) \sim N((\mu_\alpha, \mu_\beta), \Sigma)$

where

$$\Sigma = \begin{pmatrix} \sigma_\alpha^2 & \rho\sigma_\alpha\sigma_\beta \\ \rho\sigma_\alpha\sigma_\beta & \sigma_\beta^2 \end{pmatrix}$$

Stage 2. $(\mu_\alpha, \mu_\beta, \sigma_\alpha, \sigma_\beta, \rho)$ assigned weakly informative prior $g()$

Fake Data Simulation

- Simulate waiting times from the two-stage multilevel model

1. Fix values of 2nd-stage prior parameters
2. Simulate (true) regression coefficients for the J shops
3. Simulate waiting times from the regression models

- Fit model to simulated data
- The parameter estimates should be close to the values of the parameters in the simulated data

Simulating 2nd Stage Parameters

We set up the second-stage parameters for the coffee shop example. The average waiting time across all shops is $\mu_a = 3.5$ minutes and the afternoon wait time tends to be one minute shorter, so $\mu_b = -1$. The intercepts vary according to $\sigma_a = 1$ and the slopes vary by $\sigma_b = 0.5$. The true correlation between the population intercepts and slopes is $\rho = -.7$.

```
a <- 3.5                # average morning wait time
b <- (-1)               # average difference afternoon wait ti
sigma_a <- 1            # std dev in intercepts
sigma_b <- 0.5          # std dev in slopes
rho <- (-0.7)           # correlation between intercepts and s
```

Setting up 2nd Stage Multivariate Distribution

Sets up the parameters for the multivariate distribution for the coffeeshop-specific parameters (α_j, β_j).

```
Mu <- c( a , b )
cov_ab <- sigma_a * sigma_b * rho
Sigma <- matrix( c(sigma_a^2, cov_ab, cov_ab, sigma_b^2),
                 ncol=2 )
```

Simulate Varying Effects

Simulate the varying effects for the coffee shops.

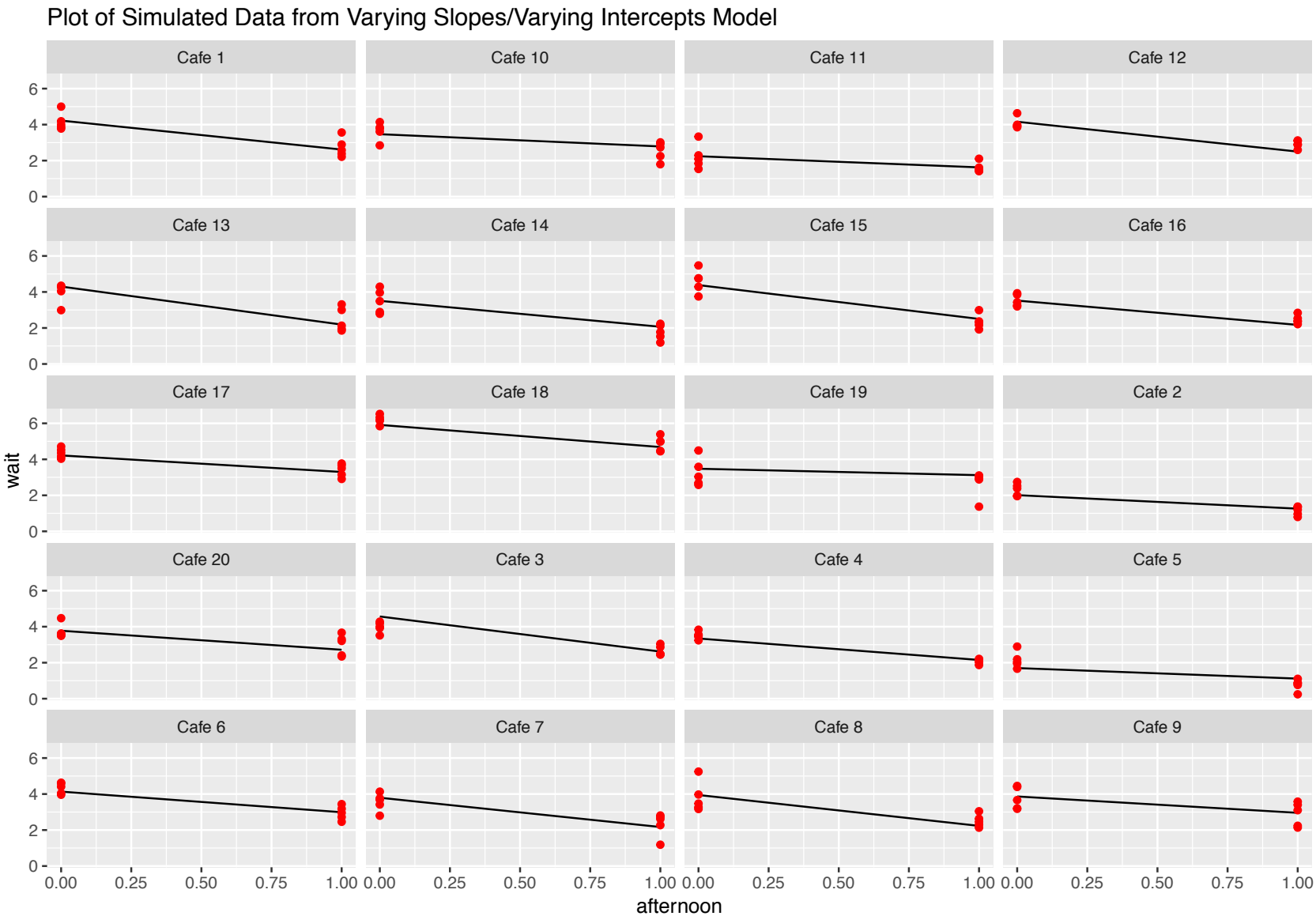
```
N_cafes <- 20
library(MASS)
set.seed(5) # used to replicate example
vary_effects <- mvrnorm( N_cafes , Mu , Sigma )
a_cafe <- vary_effects[, 1]
b_cafe <- vary_effects[, 2]
```

Simulate Observed Waiting Times

Simulate the actual waiting times (we are assuming that the sampling standard deviation is $\sigma_y = 0.5$).

```
N_cafes <- 20
N_visits <- 10
afternoon <- rep(0:1, N_visits * N_cafes / 2)
cafe_id <- rep( 1:N_cafes , each=N_visits )
mu <- a_cafe[cafe_id] + b_cafe[cafe_id] * afternoon
sigma <- 0.5  # std dev within cafes
wait <- rnorm( N_visits * N_cafes , mu , sigma )
d <- data.frame( cafe=cafe_id ,
                 afternoon=afternoon , wait=wait )
```

Simulated Data



Using STAN

- Write a Stan script defining the model (similar to a JAGS model script)
- Bring your data into R, define all data used in model description
- Use the function stan from the rstan package to fit the model using Stan
- Even easier using the rethinking package
- Extract samples and summarize/plot using the coda package

Using STAN to fit the model

Okay, we are ready to use STAN to fit the model to the simulated data.

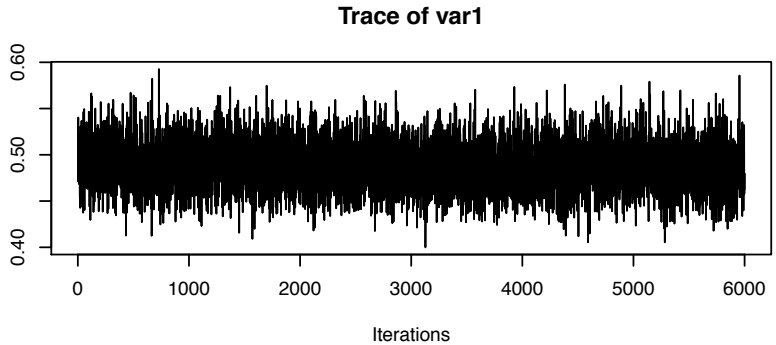
```
library(rethinking)
m13.1 <- map2stan(
  alist(
    wait ~ dnorm( mu , sigma ),
    mu <- a_cafe[cafe] + b_cafe[cafe] * afternoon,
    c(a_cafe, b_cafe)[cafe] ~ dmvnorm2(c(a, b), sigma_cafe,
    a ~ dnorm(0, 10),
    b ~ dnorm(0, 10),
    sigma_cafe ~ dcauchy(0, 2),
    sigma ~ dcauchy(0, 2),
    Rho ~ dlkjcorr(2)
  ) ,
  data=d ,
  iter=5000 , warmup=2000 , chains=2 )
```

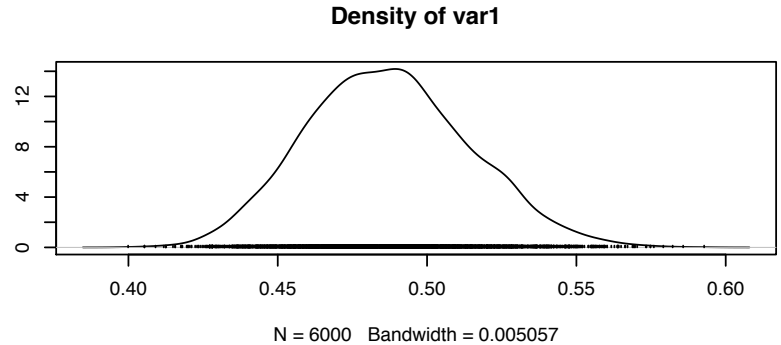
Posterior Summaries

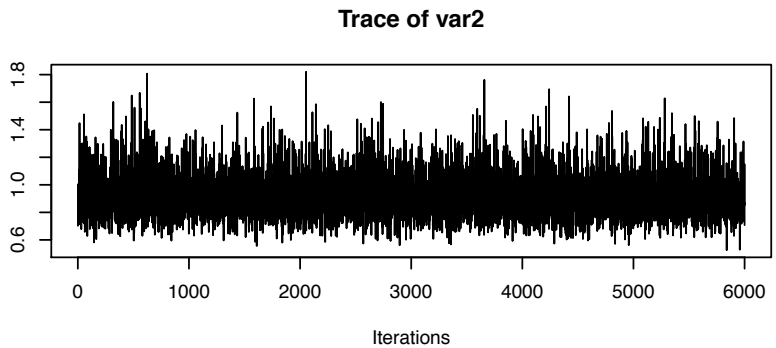
Extract simulated values by the `extract.samples` function. Note that post is a list where each component is a matrix of simulated draws of different parameters. For example, `sigma_cafe` is a matrix with 6000 rows and two columns where the columns correspond to σ_a and σ_b.

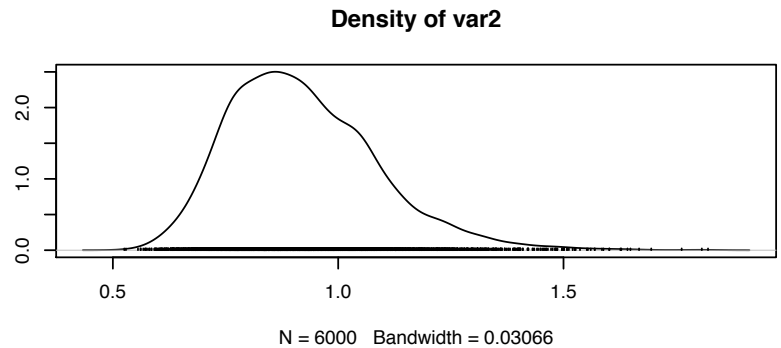
```
## [1] "b_cafe"     "a_cafe"     "a"          "b"
## [6] "sigma"      "Rho"
```

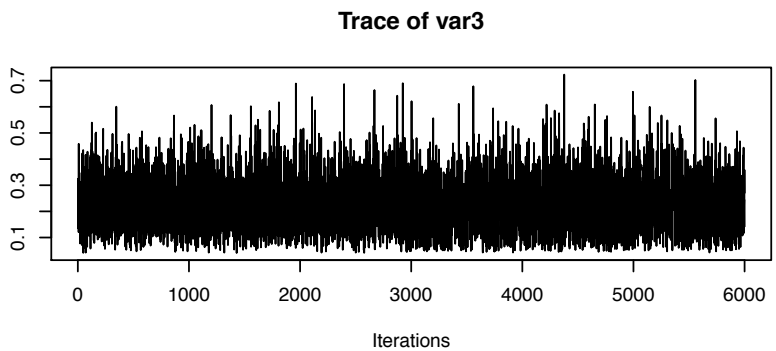
Summary of Posterior of Random Effects $(\sigma, \sigma_a, \sigma_b)$

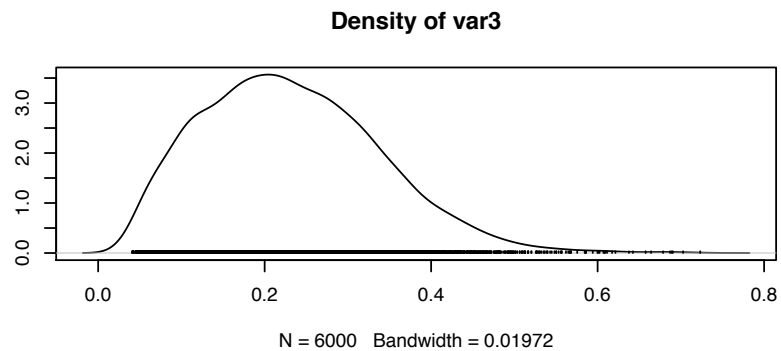












Summary of Posterior of Random Effects ($\sigma, \sigma_a, \sigma_b$)

```
##
## Iterations = 1:6000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 6000
##
## 1. Empirical mean and standard deviation for each variab
##    plus standard error of the mean:
##
##          Mean      SD  Naive SE Time-series SE
## [1,] 0.4870 0.02718 0.0003509      0.0003509
## [2,] 0.9223 0.16477 0.0021272      0.0021272
## [3,] 0.2319 0.10596 0.0013679      0.0013679
##
## 2. Quantiles for each variable:
##
##          2.5%    25%    50%    75%  97.5%
## var1 0.43712 0.4679 0.4860 0.5050 0.5417
```

Check

- In our simulation of the fake data, we assumed

$$\sigma = 0.5, \sigma_a = 1, \sigma_b = 0.5$$

- Posterior estimates of these parameters are close to these values
- Gives some reassurance that the MCMC algorithm is programmed correctly

Questions: Waiting Times at a Coffee Shop

Suppose one focuses on the morning waiting times of the J coffee shops. One considers the "varying intercepts" model

$$y_i \sim N(\mu_{j[i]}, \sigma^2), i = 1, ..., N$$

where the intercepts $\mu_1, ..., \mu_J$ follow the multilevel model

- $\mu_1, ..., \mu_J \sim N(\theta, \tau^2)$
- $(\theta, \tau^2) \sim g(\theta, \tau^2) = 1$

(We assume the sample standard deviation σ is known.)

1. First simulate data from this model. Assume that $\theta = 5, \tau = 1$, there are $J = 20$ coffee shops, and you will have a sample of $n = 5$ waiting times for each shop (so $N = 100$). Assume that the sampling standard deviation is $\sigma = .75$.
2. Explore the following computation strategies to estimate the second-stage parameters θ and τ^2.

Questions: Strategy One (LearnBayes)

Let $\bar{y}_j$ denote the sample mean of the jth group. One can show that the marginal posterior distribution of $(\theta, \log \tau^2)$ is given by

$$g(\theta, \log \tau^2) \propto \prod_{j=1}^{J} \phi\left(\bar{y}_j, \theta, \frac{\sigma^2}{n} + \tau^2\right) \tau^2$$

Here's a function to compute the log posterior:

```
logpost <- function(theta_vector, data){
   theta <- theta_vector[1]
   tausq <- exp(theta_vector[2])
   ybar <- data[, 1]
   sigmasq <- data[, 2]
   sum(dnorm(ybar, theta, sqrt(sigmasq + tausq),
             log=TRUE)) + log(tausq)
}
```

Questions: Strategy One (LearnBayes)

- Use the function laplace in the LearnBayes package to find the posterior mean and standard deviation of θ and $\log \tau^2$.
- Take a sample of size 1000 from the posterior distribution of $(\theta, \log \tau^2)$
- Use the simulated sample to find 90 percent interval estimates for θ and τ.

Questions: Strategy Two (JAGS)

The following JAGS script defines the varying intercepts model. The variable prec.y is the reciprocal of the sampling variance of $\bar{y}_j$ and prec.mu is the reciprocal of τ^2.

```
modelString = "
model {
for (i in 1:J){
y[i] ~ dnorm (mu[i], prec.y)
mu[i] ~ dnorm(theta, prec.mu)
}
prec.mu <- pow(tau2, -1)
tau2 ~ dunif(0, 100)
theta ~ dunif(0, 100)
}"
writeLines(modelString, con="normexch.bug")
```

Questions: Strategy Two (JAGS)

- Use JAGS and this model script to simulate 5000 values from the posterior distribution, collecting values of θ and τ^2.
- Construct trace plots of the simulated draws of θ and τ^2 to check convergence of the MCMC chain.
- Use the simulated draws to find 90 percent interval estimates for θ and τ.
- Compare your results with the results from Strategy One.