Bayes and R

4th Annual Bayesian Biostatistics Conference

Jim Albert
Department of Mathematics and Statistics
Bowling Green State University

Why R for Bayes?

Why R for Bayes?

- R has a large collection of functions for simulation, data analysis, and graphics.

Why R for Bayes?

- R has a large collection of functions for simulation, data analysis, and graphics.
- Easy to extend the R language

Why R for Bayes?

- R has a large collection of functions for simulation, data analysis, and graphics.
- Easy to extend the R language
  - by writing functions

Why R for Bayes?

- R has a large collection of functions for simulation, data analysis, and graphics.
- Easy to extend the R language
  - by writing functions
  - by creating packages

Why R for Bayes?

- R has a large collection of functions for simulation, data analysis, and graphics.
- Easy to extend the R language
  - by writing functions
  - by creating packages
- There are nice interfaces of R with more efficient MCMC computing engines like OpenBUGS

Three Stories

Three Stories

- Learning About Streakiness

Three Stories

- Learning About Streakiness
  - Using R simulation functions.

Three Stories

- Learning About Streakiness
  - Using R simulation functions.
  - Using functions from `LearnBayes` package.

Three Stories

- Learning About Streakiness
  - Using R simulation functions.
  - Using functions from `LearnBayes` package.
- Exploring Patterns in Website Counts

Three Stories

- Learning About Streakiness
  - Using R simulation functions.
  - Using functions from `LearnBayes` package.
- Exploring Patterns in Website Counts
  - Using R to program Gibbs sampling.

Three Stories

- Learning About Streakiness
  - Using R simulation functions.
  - Using functions from `LearnBayes` package.
- Exploring Patterns in Website Counts
  - Using R to program Gibbs sampling.
- Website Counts, Continuted

Three Stories

- Learning About Streakiness
  - Using R simulation functions.
  - Using functions from `LearnBayes` package.
- Exploring Patterns in Website Counts
  - Using R to program Gibbs sampling.
- Website Counts, Continuted
  - Using `LearnBayes`.

Three Stories

- Learning About Streakiness
  - Using R simulation functions.
  - Using functions from `LearnBayes` package.
- Exploring Patterns in Website Counts
  - Using R to program Gibbs sampling.
- Website Counts, Continuted
  - Using `LearnBayes`.
  - Using `MCMCpack`

Three Stories

- Learning About Streakiness
  - Using R simulation functions.
  - Using functions from `LearnBayes` package.
- Exploring Patterns in Website Counts
  - Using R to program Gibbs sampling.
- Website Counts, Continuted
  - Using `LearnBayes`.
  - Using `MCMCpack`
  - Using R interface to Openbugs

References

References

- Albert (2009), *Bayesian Computation with R*, 2nd edition, Springer.

References

- Albert (2009), *Bayesian Computation with R*, 2nd edition, Springer.
- `LearnBayes` and `MCMCpack` packages, available from CRAN (`http://cran.r-project.org/`)

References

- Albert (2009), *Bayesian Computation with R*, 2nd edition, Springer.
- LearnBayes and MCMCpack packages, available from CRAN (http://cran.r-project.org/)
- R code and bugs file for my examples available at http://bayes.bgsu.edu/bcwr/bayes.biostats.2011/

References

- Albert (2009), *Bayesian Computation with R*, 2nd edition, Springer.
- `LearnBayes` and `MCMCpack` packages, available from CRAN (`http://cran.r-project.org/`)
- R code and bugs file for my examples available at `http://bayes.bgsu.edu/bcwr/bayes.biostats.2011/`
- CRAN Task View: Bayesian Inference (`http://cran.r-project.org/`)

Learning About Streakiness

4th Annual Bayesian Biostatistics Conference

Jim Albert
Department of Mathematics and Statistics
Bowling Green State University

Was Dustin Pedroia streaky in the 2008 season?

Was Dustin Pedroia streaky in the 2008 season?

- Pedroia had 628 batting opportunities – for each opportunity, observe Hit (1) or Out (0).

Was Dustin Pedroia streaky in the 2008 season?

- Pedroia had 628 batting opportunities – for each opportunity, observe Hit (1) or Out (0).
- Here is some of the data:

```
 [1] 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1 0
[18] 0 0 0 0 1 0 1 1 0 0 0 0 1 1 0 0 0
[35] 0 0 0 0 1 1 0 0 0 0 1 0 1 0 0 0 0
```

. . .

Was Dustin Pedroia streaky in the 2008 season?

- Pedroia had 628 batting opportunities – for each opportunity, observe Hit (1) or Out (0).
- Here is some of the data:

```
  [1] 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1 0
 [18] 0 0 0 0 1 0 1 1 0 0 0 0 1 1 0 0 0
 [35] 0 0 0 0 1 1 0 0 0 0 1 0 1 0 0 0 0
...
```

- Focus on the spacings between successive hits that we put in a vector y:

```
> y
  [1]  0  1  8  3  5  1  0  4  0  7  0
 [12]  4  1  4  0  2  0  1  1  1  0  3
...
```

Plan to Investigate Streakiness

Plan to Investigate Streakiness

- Assume the spacings are independent with a geometric distribution with a constant hitting probability p

Plan to Investigate Streakiness

- Assume the spacings are independent with a geometric distribution with a constant hitting probability p
- Fit the model using some prior knowledge about p

Plan to Investigate Streakiness

- Assume the spacings are independent with a geometric distribution with a constant hitting probability p
- Fit the model using some prior knowledge about p
- Check the goodness of fit of the model

Plan to Investigate Streakiness

- Assume the spacings are independent with a geometric distribution with a constant hitting probability p
- Fit the model using some prior knowledge about p
- Check the goodness of fit of the model
- We'll find that the geometric model is inappropriate – fit a new model (beta/geometric) that allows for variability in the hitting probability p

Plan to Investigate Streakiness

- Assume the spacings are independent with a geometric distribution with a constant hitting probability p
- Fit the model using some prior knowledge about p
- Check the goodness of fit of the model
- We'll find that the geometric model is inappropriate – fit a new model (beta/geometric) that allows for variability in the hitting probability p
- We'll introduce some modern simulation methods for exploring a posterior distribution

A Geometric Sampling Model

A Geometric Sampling Model

- Assume the spacings $y_1, ..., y_n$ are independent where y_i is geometric with probability p.

$$f(y_i|p) = p(1-p)^y, \;\; y = 0, 1, 2, ...$$

A Geometric Sampling Model

- Assume the spacings $y_1, ..., y_n$ are independent where y_i is geometric with probability p.

$$f(y_i|p) = p(1-p)^y, \;\; y = 0, 1, 2, ...$$

- Likelihood function $L(p)$ is joint density of the spacings viewed as a function of the parameter p.

$$L(p) = \prod_{i=1}^{n} f(y_i|p) = p^n(1-p)^s,$$

where $s = \sum y$ is the sum of the observations and the n is the sample size.

The Prior

The Prior

- Next step in a Bayesian analysis is to assign a prior density $g(p)$.

The Prior

- Next step in a Bayesian analysis is to assign a prior density $g(p)$.
- A prior reflects one's beliefs about the location of Pedroia's batting probability before sampling.

The Prior

- Next step in a Bayesian analysis is to assign a prior density $g(p)$.
- A prior reflects one's beliefs about the location of Pedroia's batting probability before sampling.
- Here's my beliefs:
  $P(p < 0.310) = 0.5, P(p < 0.350) = 0.8.$

The Prior

- Next step in a Bayesian analysis is to assign a prior density $g(p)$.
- A prior reflects one's beliefs about the location of Pedroia's batting probability before sampling.
- Here's my beliefs:
  $P(p < 0.310) = 0.5$, $P(p < 0.350) = 0.8$.
- Then I find a beta prior density $g(p)$ that matches these beliefs.

```
> library(LearnBayes)
> beta.select(list(p=.5,x=.31), list(p=.8,x=.35))
[1] 30.89 68.35
```

The Posterior

The Posterior

- Observe the data. We use R to compute the sample size n and the sum of observations s.

  ```
  > n = length(y)
  > s = sum(y)
  ```

  We see $n = 213$ and $s = 438$.

The Posterior

- Observe the data. We use R to compute the sample size n and the sum of observations s.

  ```
  > n = length(y)
  > s = sum(y)
  ```

  We see $n = 213$ and $s = 438$.
- By Bayes' rule, posterior density of p is proportional to the product of the likelihood $L(p)$ and the beta prior $g(p)$.

The Posterior

- Observe the data. We use R to compute the sample size n and the sum of observations s.

  ```
  > n = length(y)
  > s = sum(y)
  ```

  We see $n = 213$ and $s = 438$.
- By Bayes' rule, posterior density of p is proportional to the product of the likelihood $L(p)$ and the beta prior $g(p)$.
- Applying this recipe

$$g(p|y) \propto p^n(1-p)^s \times p^{a-1}(1-p)^{b-1}.$$

- Substituting the observed values of $n = 213$ and $s = 438$, and the beta parameter values $a = 31, b = 68$, we obtain the posterior density

$$g(p|y) \propto p^{213+31-1}(1-p)^{438+68-1}$$

which we recognize as a beta density with shape parameters $a_1 = 244$ and $b_1 = 506$.

- Substituting the observed values of $n = 213$ and $s = 438$, and the beta parameter values $a = 31, b = 68$, we obtain the posterior density

$$g(p|y) \propto p^{213+31-1}(1-p)^{438+68-1}$$

which we recognize as a beta density with shape parameters $a_1 = 244$ and $b_1 = 506$.
- R package contains a set of functions for plotting a beta density, computing beta probabilities and quantiles, and for simulating beta variates and these are all helpful for summarizing the posterior density.

Simulation Approach for Summarizing the Posterior

Simulation Approach for Summarizing the Posterior

- Simulate a large independent sample from the posterior density.

  ```
  > p=rbeta(1000, n+a, s+b)
  ```

Simulation Approach for Summarizing the Posterior

- Simulate a large independent sample from the posterior density.

  ```
  > p=rbeta(1000, n+a, s+b)
  ```

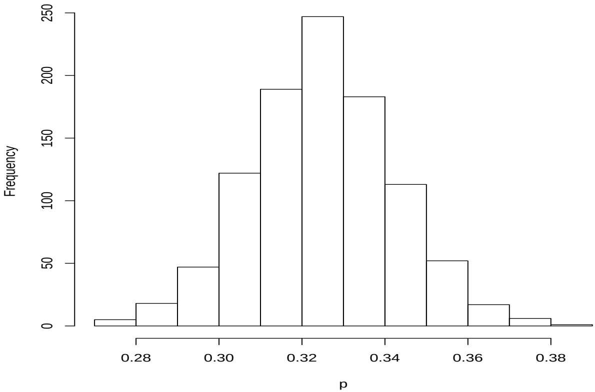
- Use data analysis graphs and summaries of the simulated sample to learn about the parameter.

Graph the Posterior

Graph the Posterior

- Plot the posterior density using the R functions `hist` (can use `density` for a smoother density estimate):

  `> hist(p,main="")`



- Summarize the posterior density by finding the .05, .5, .95 quantiles of the gamma density.

```
> quantile(p,c(0.05,0.5,0.95))
```

```
       5%       50%       95%
0.2974597 0.3260421 0.3552960
```

A point estimate at p is the posterior median 0.326. A 90% interval estimate for λ is found by the 5th and 95th percentiles (0.297, 0.355).

- Summarize the posterior density by finding the .05, .5, .95 quantiles of the gamma density.

  ```
  > quantile(p,c(0.05,0.5,0.95))
         5%       50%       95%
  0.2974597 0.3260421 0.3552960
  ```

  A point estimate at p is the posterior median 0.326. A 90% interval estimate for λ is found by the 5th and 95th percentiles (0.297, 0.355).
- Is it likely that Pedroia's true batting ability exceeds 0.340? Answer this by computing the probability that p is larger than 0.340.

  ```
  > mean(p > 0.340)
  [1] 0.212
  ```

The Predictive Distribution

The Predictive Distribution

- Two key distributions in a Bayesian analysis: the posterior distribution and the predictive distribution.

The Predictive Distribution

- Two key distributions in a Bayesian analysis: the posterior distribution and the predictive distribution.
- Given sampling density $f(y|p)$ and current beliefs about p are given by the distribution $g(p)$.

The Predictive Distribution

- Two key distributions in a Bayesian analysis: the posterior distribution and the predictive distribution.
- Given sampling density $f(y|p)$ and current beliefs about p are given by the distribution $g(p)$.
- The predictive density of y is given by

$$f(y) = \int f(y|p)g(p)dp.$$

The Predictive Distribution

- Two key distributions in a Bayesian analysis: the posterior distribution and the predictive distribution.
- Given sampling density $f(y|p)$ and current beliefs about p are given by the distribution $g(p)$.
- The predictive density of y is given by

$$f(y) = \int f(y|p)g(p)dp.$$

- The predictive density can be used to predict future observation values y.

Predicting Future Spacings

Predicting Future Spacings

- After data is observed, current beliefs about p are reflected in Beta(244, 506) distribution.

Predicting Future Spacings

- After data is observed, current beliefs about p are reflected in Beta(244, 506) distribution.
- Suppose we wish to predict spacings $y_1^*, \dots, y_{213}^*$ (we observed 213 spacings in the 2008 season).

Predicting Future Spacings

- After data is observed, current beliefs about p are reflected in Beta(244, 506) distribution.
- Suppose we wish to predict spacings $y_1^*, \ldots, y_{213}^*$ (we observed 213 spacings in the 2008 season).
- Density of these future counts is given by

$$f(y^*) = \int \prod Geom(y_i^*; p) Beta(p; 244, 506) dp$$

Predicting Future Spacings

- After data is observed, current beliefs about p are reflected in Beta(244, 506) distribution.
- Suppose we wish to predict spacings $y_1^*, \ldots, y_{213}^*$ (we observed 213 spacings in the 2008 season).
- Density of these future counts is given by

$$f(y^*) = \int \prod Geom(y_i^*; p) Beta(p; 244, 506) dp$$

- Easy to simulate values of y^* by simulation.

Predicting Future Spacings

- After data is observed, current beliefs about p are reflected in Beta(244, 506) distribution.
- Suppose we wish to predict spacings $y_1^*, ..., y_{213}^*$ (we observed 213 spacings in the 2008 season).
- Density of these future counts is given by

$$f(y^*) = \int \prod Geom(y_i^*; p) Beta(p; 244, 506) dp$$

- Easy to simulate values of y^* by simulation.
  1. Simulate p from Beta(244, 506)

Predicting Future Spacings

- After data is observed, current beliefs about p are reflected in Beta(244, 506) distribution.
- Suppose we wish to predict spacings $y_1^*, \ldots, y_{213}^*$ (we observed 213 spacings in the 2008 season).
- Density of these future counts is given by

$$f(y^*) = \int \prod Geom(y_i^*; p) Beta(p; 244, 506) dp$$

- Easy to simulate values of y^* by simulation.
  1. Simulate p from Beta(244, 506)
  2. Simulate $y_1^*, \ldots, y_{213}^*$ from Geometric(p)

Prediction on R

Prediction on R

- Write a short function to simulate one future sample of spacings.

```
> predict = function() {
+       p = rbeta(1, 244, 506)
+       rgeom(213, p)
+ }
```

Prediction on R

- Write a short function to simulate one future sample of spacings.

```
> predict = function() {
+       p = rbeta(1, 244, 506)
+       rgeom(213, p)
+ }
```

- Simulate one sample.

```
> predict()
```

```
 [1] 0 4 0 0 0 1 7 1 1 0 0 0 1 0 0 3 0
[18] 7 0 6 5 0 0 0 4 2 1 2 0 0 0 2 0 0
[35] 0 3 1 2 3 0 0 4 0 2 1 0 0 3 2 0 3
[52] 4 1 3 1 0 1 2 4 4 1 2 4 0 0 4 0 0
...
```

Model Checking

Model Checking

- Question: do the simulated predicted counts from the fitted model resemble the actual counts?

Model Checking

- Question: do the simulated predicted counts from the fitted model resemble the actual counts?
- Answer this tabulating the observed and predicting counts.

Model Checking

- Question: do the simulated predicted counts from the fitted model resemble the actual counts?
- Answer this tabulating the observed and predicting counts.
- We'll see that the actual counts look more spread out than the predicted counts.

Comparing Predicted and Actual Counts

Comparing Predicted and Actual Counts

- Use `table` to tabulate actual counts:

```
> table(y)

y
 0  1  2  3  4  5  6  7  8  9 12 16 18
72 56 24 13 22  8  3  4  3  3  3  1  1
```

Comparing Predicted and Actual Counts

- Use `table` to tabulate actual counts:

```
> table(y)

y
 0  1  2  3  4  5  6  7  8  9 12 16 18
72 56 24 13 22  8  3  4  3  3  3  1  1
```

- Use `predict` to simulate the counts and `tabulate`:

```
> ys=predict()
> table(ys)

ys
 0  1  2  3  4  5  6  7  8  9 11 13 16
58 42 38 24 11 10  6  8  6  7  1  1  1
```

Comparing Predicted and Actual Counts

- Use `table` to tabulate actual counts:

```
> table(y)

y
 0  1  2  3  4  5  6  7  8  9 12 16 18
72 56 24 13 22  8  3  4  3  3  3  1  1
```

- Use `predict` to simulate the counts and `tabulate`:

```
> ys=predict()
> table(ys)

ys
 0  1  2  3  4  5  6  7  8  9 11 13 16
58 42 38 24 11 10  6  8  6  7  1  1  1
```

- Do you see any differences between the predicted and actual counts?

Model Checking by the Predictive Distribution

Model Checking by the Predictive Distribution

- Simulate values from the (posterior) predictive distribution.

Model Checking by the Predictive Distribution

- Simulate values from the (posterior) predictive distribution.
- Compute value of a checking function $T(y^*)$ (here $T(y^*) = SD(y^*)$ would be a good choice).

Model Checking by the Predictive Distribution

- Simulate values from the (posterior) predictive distribution.
- Compute value of a checking function $T(y^*)$ (here $T(y^*) = SD(y^*)$ would be a good choice).
- Repeat simulation many times – get distribution of checking function $T(y^*)$.

Model Checking by the Predictive Distribution

- Simulate values from the (posterior) predictive distribution.
- Compute value of a checking function $T(y^*)$ (here $T(y^*) = SD(y^*)$ would be a good choice).
- Repeat simulation many times – get distribution of checking function $T(y^*)$.
- See if $T_{observed}$ is consistent with this distribution.

Model Checking by the Predictive Distribution

- Simulate values from the (posterior) predictive distribution.
- Compute value of a checking function $T(y^*)$ (here $T(y^*) = SD(y^*)$ would be a good choice).
- Repeat simulation many times – get distribution of checking function $T(y^*)$.
- See if $T_{observed}$ is consistent with this distribution.
- If $T_{observed}$ is "extreme", indicates model misfit.

The Example

The Example

- Write function to simulate sample from y^* and compute $SD(y^*)$.

```
> model.check=function()
+ {
+ p=rbeta(1,244,506)
+ ys=rgeom(n, p)
+ sd(ys)
+ }
```

The Example

- Write function to simulate sample from y^* and compute $SD(y^*)$.

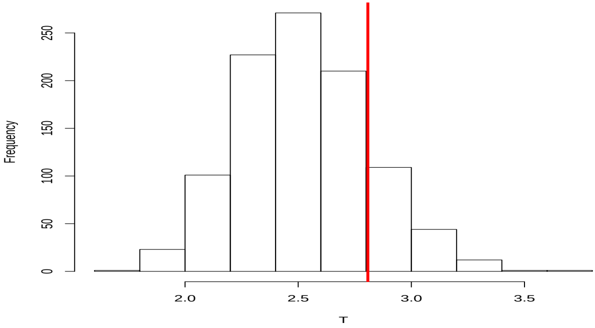
```
> model.check=function()
+ {
+ p=rbeta(1,244,506)
+ ys=rgeom(n, p)
+ sd(ys)
+ }
```

- Repeat this simulation 1000 times and collect values of T.

```
> T = replicate(1000, model.check())
```

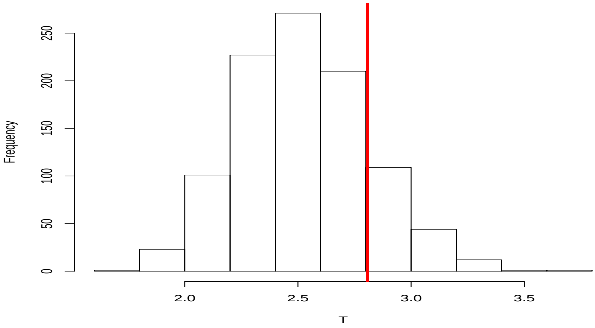
- Construct histogram of T and show value of $T_{observed}$ by a vertical line.

```
> hist(T)
> abline(v = sd(y), lwd = 3, col = "red")
```



- Construct histogram of T and show value of $T_{observed}$ by a vertical line.

```
> hist(T)
> abline(v = sd(y), lwd = 3, col = "red")
```



- There is more dispersion in the data than predicted from the geometric model.

A Overdispersion Model: Beta/Geometric

A Overdispersion Model: Beta/Geometric

- Assume y_i is Geometric with parameter $p_i, i = 1, ..., n$.

A Overdispersion Model: Beta/Geometric

- Assume y_i is Geometric with parameter $p_i, i = 1, ..., n$.
- p_i distributed Beta(K, η)

A Overdispersion Model: Beta/Geometric

- Assume y_i is Geometric with parameter $p_i, i = 1, ..., n$.
- p_i distributed Beta(K, η)
- (K, η) have (weakly informative) prior

$$g(K, \eta) = \frac{1}{\eta(1-\eta)} \frac{1}{(1+K)^2}.$$

A Overdispersion Model: Beta/Geometric

- Assume y_i is Geometric with parameter $p_i, i = 1, ..., n$.
- p_i distributed Beta(K, η)
- (K, η) have (weakly informative) prior

$$g(K, \eta) = \frac{1}{\eta(1-\eta)} \frac{1}{(1+K)^2}.$$

- Want to learn about K – indicates degree of overdispersion in data.

A Overdispersion Model: Beta/Geometric

- Assume y_i is Geometric with parameter $p_i, i = 1, ..., n$.
- p_i distributed Beta(K, η)
- (K, η) have (weakly informative) prior

$$g(K, \eta) = \frac{1}{\eta(1-\eta)} \frac{1}{(1+K)^2}.$$

- Want to learn about K – indicates degree of overdispersion in data.
- As K approaches ∞, model approaches Geometric(p).

Posterior of 2nd Stage Parameters

Posterior of 2nd Stage Parameters

- One can show that the marginal posterior density of (K, η) is

$$g(K, \eta | y) \propto g(K, \eta) \times \prod_{i=1}^{n} f(y_i | K, \eta),$$

where

$$f(y_i | K, \eta) = \frac{B(K\eta + 1, K(1 - \eta) + y_j)}{B(K\eta, K(1 - \eta))}$$

Posterior of 2nd Stage Parameters

- One can show that the marginal posterior density of (K, η) is

$$g(K, \eta | y) \propto g(K, \eta) \times \prod_{i=1}^{n} f(y_i | K, \eta),$$

  where

$$f(y_i | K, \eta) = \frac{B(K\eta + 1, K(1 - \eta) + y_j)}{B(K\eta, K(1 - \eta))}$$

- Want to summarize this distribution for inference.

Posterior of 2nd Stage Parameters

- One can show that the marginal posterior density of (K, η) is

$$g(K, \eta | y) \propto g(K, \eta) \times \prod_{i=1}^{n} f(y_i | K, \eta),$$

where

$$f(y_i | K, \eta) = \frac{B(K\eta + 1, K(1-\eta) + y_j)}{B(K\eta, K(1-\eta))}$$

- Want to summarize this distribution for inference.
- Problem: we can't use direct simulation since this is not a familiar functional form.

Summarization of the Posterior Using R

Summarization of the Posterior Using R

- Write a R function that computes the logarithm of the posterior density.

Summarization of the Posterior Using R

- Write a R function that computes the logarithm of the posterior density.
- Write log posterior as

$$\log g(K, \eta | y) = \log g(K, \eta) + \sum_{i=1}^{n} \log f(y_i | K, \eta),$$

where

$$\begin{aligned} \log f(y_i | K, \eta) &= \log B(K\eta + 1, K(1-\eta) + y_j) \\ &- \log B(K\eta, K(1-\eta)) \end{aligned}$$

Summarization of the Posterior Using R

- Write a R function that computes the logarithm of the posterior density.
- Write log posterior as

$$\log g(K, \eta | y) = \log g(K, \eta) + \sum_{i=1}^{n} \log f(y_i | K, \eta),$$

where

$$\begin{aligned} \log f(y_i | K, \eta) &= \log B(K\eta + 1, K(1-\eta) + y_j) \\ &- \log B(K\eta, K(1-\eta)) \end{aligned}$$

- Helpful to transform posterior to $(\theta_1, \theta_2) = (logit\eta, \log K)$. (Why?)

Why are We Transforming the Parameters?

Why are We Transforming the Parameters?

- Posterior is very right skewed in K.

Why are We Transforming the Parameters?

- Posterior is very right skewed in K.
- Accuracy of normal approximation will be poor.

Why are We Transforming the Parameters?

- Posterior is very right skewed in K.
- Accuracy of normal approximation will be poor.
- Can improve things by a suitable reexpression so that each parameter is real-valued.

Why are We Transforming the Parameters?

- Posterior is very right skewed in K.
- Accuracy of normal approximation will be poor.
- Can improve things by a suitable reexpression so that each parameter is real-valued.
- Reexpress to $(\theta_1, \theta_2) = (logit \eta, \log K)$.

Why are We Transforming the Parameters?

- Posterior is very right skewed in K.
- Accuracy of normal approximation will be poor.
- Can improve things by a suitable reexpression so that each parameter is real-valued.
- Reexpress to $(\theta_1, \theta_2) = (logit \eta, \log K)$.
- Write the log posterior that is a function of (θ_1, θ_2).

Why are We Transforming the Parameters?

- Posterior is very right skewed in K.
- Accuracy of normal approximation will be poor.
- Can improve things by a suitable reexpression so that each parameter is real-valued.
- Reexpress to $(\theta_1, \theta_2) = (logit \eta, \log K)$.
- Write the log posterior that is a function of (θ_1, θ_2).
- Don't forget the Jacobian!

R Function to Compute Log Posterior

R Function to Compute Log Posterior

- `theta` is vector of parameters $(\text{logit}\eta, \log K)$

R Function to Compute Log Posterior

- `theta` is vector of parameters $(\text{logit}\eta, \log K)$
- `y` is vector of counts

R Function to Compute Log Posterior

- `theta` is vector of parameters ($\text{logit}\eta, \log K$)
- `y` is vector of counts
- output is the value of the log posterior evaluated at `theta`

```
beta.geom=function(theta,y)
{
eta = exp(theta[1])/(1 + exp(theta[1]))
K = exp(theta[2])
N=length(y)
a=K*eta
b=K*(1-eta)
sum(lbeta(a+1,y+b))-N*lbeta(a,b) +
   theta[2] - 2 * log(1 + exp(theta[2]))
}
```

Summarizing the Posterior by the Mode

Summarizing the Posterior by the Mode

- Use an algorithm such as Newton's method to find the posterior model.

Summarizing the Posterior by the Mode

- Use an algorithm such as Newton's method to find the posterior model.
- This algorithm is implemented in the function `laplace` in the `LearnBayes` package.

Summarizing the Posterior by the Mode

- Use an algorithm such as Newton's method to find the posterior model.
- This algorithm is implemented in the function `laplace` in the `LearnBayes` package.
- Inputs to `laplace` are

Summarizing the Posterior by the Mode

- Use an algorithm such as Newton's method to find the posterior model.
- This algorithm is implemented in the function `laplace` in the `LearnBayes` package.
- Inputs to `laplace` are
  - the function defining the log posterior

Summarizing the Posterior by the Mode

- Use an algorithm such as Newton's method to find the posterior model.
- This algorithm is implemented in the function `laplace` in the `LearnBayes` package.
- Inputs to `laplace` are
  - the function defining the log posterior
  - guess at the posterior mode

Summarizing the Posterior by the Mode

- Use an algorithm such as Newton's method to find the posterior model.
- This algorithm is implemented in the function `laplace` in the `LearnBayes` package.
- Inputs to `laplace` are
  - the function defining the log posterior
  - guess at the posterior mode
  - any data used in the log posterior

Summarizing the Posterior by the Mode

- Use an algorithm such as Newton's method to find the posterior model.
- This algorithm is implemented in the function `laplace` in the `LearnBayes` package.
- Inputs to `laplace` are
  - the function defining the log posterior
  - guess at the posterior mode
  - any data used in the log posterior
- Output of `laplace` is posterior mode $\hat{\theta}$ and estimate V at variance-covariance matrix.

Summarizing the Posterior by the Mode

- Use an algorithm such as Newton's method to find the posterior model.
- This algorithm is implemented in the function `laplace` in the `LearnBayes` package.
- Inputs to `laplace` are
  - the function defining the log posterior
  - guess at the posterior mode
  - any data used in the log posterior
- Output of `laplace` is posterior mode $\hat{\theta}$ and estimate V at variance-covariance matrix.
- Approximately, posterior of θ is $\mathrm{N}(\hat{\theta}, V)$.

R Code to Find the Mode

```
> library(LearnBayes)
> # read in beta.geom function
> guess.at.mode=c(1,1)
> fit=laplace(beta.geom,guess.at.mode,y)
> fit

$mode
[1] -0.5668859  2.9022052
$var
            [,1]        [,2]
[1,]  0.01206898 -0.03539901
[2,] -0.03539901  0.28599773
$int
[1] -414.655
$converge
[1] TRUE
```

Posterior Normal Approximation

Posterior Normal Approximation

- Marginal posterior of $\log(K)$ is approximately $\mathrm{N}(2.90, \sqrt{0.285})$.

Posterior Normal Approximation

- Marginal posterior of $\log(K)$ is approximately N(2.90, $\sqrt{0.285}$).
- 90% interval estimate for $\log(\alpha)$ is

$$(2.90 - 1.645 \times \sqrt{0.285}, 2.90 + 1.645 \times \sqrt{0.285})$$

Posterior Normal Approximation

- Marginal posterior of $\log(K)$ is approximately N(2.90, $\sqrt{0.285}$).
- 90% interval estimate for $\log(\alpha)$ is

$$(2.90 - 1.645 \times \sqrt{0.285}, 2.90 + 1.645 \times \sqrt{0.285})$$

- This approximation is a good starting point in the simulation-based methods of fitting the model.

Posterior Normal Approximation

- Marginal posterior of $\log(K)$ is approximately N(2.90, $\sqrt{0.285}$).
- 90% interval estimate for $\log(\alpha)$ is

$$(2.90 - 1.645 \times \sqrt{0.285}, 2.90 + 1.645 \times \sqrt{0.285})$$

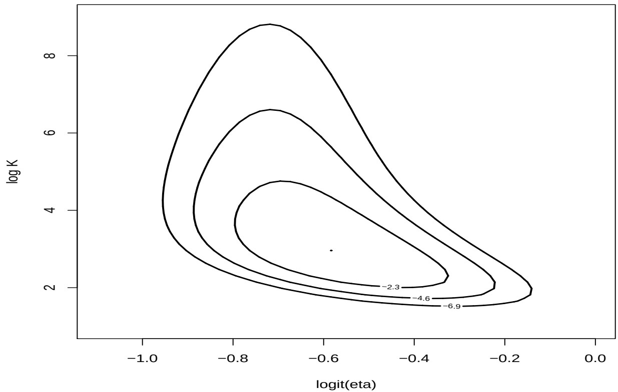
- This approximation is a good starting point in the simulation-based methods of fitting the model.
- How accurate is this approximation?

Plot the Posterior

Plot the Posterior

- Choose a viewing interval of (-1.1, 0) for logit η and (1, 9) for log K.

```
> mycontour(beta.geom,c(-1.1,0,1,9),y,
+    xlab="logit(eta)",ylab="log K")
```



Markov Chain Monte Carlo Fitting

Markov Chain Monte Carlo Fitting

- Set up a Markov Chain to explore the posterior distribution.

Markov Chain Monte Carlo Fitting

- Set up a Markov Chain to explore the posterior distribution.
- Under general conditions, the limiting distribution of the chain will be the posterior density of interest.

Markov Chain Monte Carlo Fitting

- Set up a Markov Chain to explore the posterior distribution.
- Under general conditions, the limiting distribution of the chain will be the posterior density of interest.
- There are general-purpose Markov Chain algorithms that work for many problems.

Random Walk Metropolis-Hastings Algorithm

Random Walk Metropolis-Hastings Algorithm

- Given that the chain is at a current value $\theta^{(i)}$, choose a proposal value $\theta^{(p)}$ that is in a neighborhood of the current value.

Random Walk Metropolis-Hastings Algorithm

- Given that the chain is at a current value $\theta^{(i)}$, choose a proposal value $\theta^{(p)}$ that is in a neighborhood of the current value.
- Compute a probability P of moving to the proposal value.

Random Walk Metropolis-Hastings Algorithm

- Given that the chain is at a current value $\theta^{(i)}$, choose a proposal value $\theta^{(p)}$ that is in a neighborhood of the current value.
- Compute a probability P of moving to the proposal value.
- With probability P, the next value in the chain is the proposal value; other the next value is the current value.

Random Walk Metropolis-Hastings Algorithm

- Given that the chain is at a current value $\theta^{(i)}$, choose a proposal value $\theta^{(p)}$ that is in a neighborhood of the current value.
- Compute a probability P of moving to the proposal value.
- With probability P, the next value in the chain is the proposal value; other the next value is the current value.
- Main task is to choose a reasonable neighborhood by the selection of cV, where V is an approximate variance-covariance matrix and c is a scale factor.

Random Walk Metropolis-Hastings Algorithm

- Given that the chain is at a current value $\theta^{(i)}$, choose a proposal value $\theta^{(p)}$ that is in a neighborhood of the current value.
- Compute a probability P of moving to the proposal value.
- With probability P, the next value in the chain is the proposal value; other the next value is the current value.
- Main task is to choose a reasonable neighborhood by the selection of cV, where V is an approximate variance-covariance matrix and c is a scale factor.
- Want acceptance rate of the algorithm to be approximately 20% - 40%.

R function `rwmetrop`

```
rwmetrop(logpost,proposal,start,m,par)
```

R function `rwmetrop`

```
rwmetrop(logpost,proposal,start,m,par)
```

- `logpost`: function defining the log posterior density

R function `rwmetrop`

```
rwmetrop(logpost,proposal,start,m,par)
```

- `logpost`: function defining the log posterior density
- `proposal`: a list containing var, an estimated variance-covariance matrix, and scale, the Metropolis scale factor

R function `rwmetrop`

```
rwmetrop(logpost,proposal,start,m,par)
```

- `logpost`: function defining the log posterior density
- `proposal`: a list containing var, an estimated variance-covariance matrix, and scale, the Metropolis scale factor
- `start`: vector containing the starting value of the parameter

R function `rwmetrop`

```
rwmetrop(logpost,proposal,start,m,par)
```

- `logpost:` function defining the log posterior density
- `proposal:` a list containing var, an estimated variance-covariance matrix, and scale, the Metropolis scale factor
- `start:` vector containing the starting value of the parameter
- `m:` the number of iterations of the chain

R function `rwmetrop`

```
rwmetrop(logpost,proposal,start,m,par)
```

- `logpost:` function defining the log posterior density
- `proposal:` a list containing var, an estimated variance-covariance matrix, and scale, the Metropolis scale factor
- `start:` vector containing the starting value of the parameter
- `m:` the number of iterations of the chain
- `par:` data that is used in the function logpost

MCMC for the example

MCMC for the example

- `beta.geom` contains the definition of the log posterior

MCMC for the example

- `beta.geom` contains the definition of the log posterior
- Have already stored the output of `laplace` in the variable `fit`.

MCMC for the example

- `beta.geom` contains the definition of the log posterior
- Have already stored the output of `laplace` in the variable `fit`.
- Have estimate at var-cov matrix in `fit$var`.

MCMC for the example

- `beta.geom` contains the definition of the log posterior
- Have already stored the output of `laplace` in the variable `fit`.
- Have estimate at var-cov matrix in `fit$var`.
- Starting value can be `fit$mode`

MCMC for the example

- `beta.geom` contains the definition of the log posterior
- Have already stored the output of `laplace` in the variable `fit`.
- Have estimate at var-cov matrix in `fit$var`.
- Starting value can be `fit$mode`
- Use a large number of iterates, say 10,000.

```
> mcmc.fit=rwmetrop(beta.geom,
+     list(var=fit$var,scale=2),
+     fit$mode,10000,y)
```

MCMC Diagnostics

MCMC Diagnostics

- Is the MCMC sample a reasonable approximation to the posterior distribution?

MCMC Diagnostics

- Is the MCMC sample a reasonable approximation to the posterior distribution?
- **Burn-in?** How many iterations does it take the chain to "reach" the posterior?

MCMC Diagnostics

- Is the MCMC sample a reasonable approximation to the posterior distribution?
- **Burn-in?** How many iterations does it take the chain to "reach" the posterior?
- **Good mixing?** Is the chain moving well across the posterior? Is there significant autocorrelation in the chain?

MCMC Diagnostics

- Is the MCMC sample a reasonable approximation to the posterior distribution?
- **Burn-in?** How many iterations does it take the chain to "reach" the posterior?
- **Good mixing?** Is the chain moving well across the posterior? Is there significant autocorrelation in the chain?
- **How many?** How many iterations should be collected?

Package CODA

Package CODA

- Create a MCMC object from the matrix of simulated draws by the `mcmc` function.

Package CODA

- Create a MCMC object from the matrix of simulated draws by the `mcmc` function.
- `plot` function produces trace plots and density plots of each parameter.

Package CODA

- Create a MCMC object from the matrix of simulated draws by the `mcmc` function.
- `plot` function produces trace plots and density plots of each parameter.
- `autocorr.plot` function produces autocorrelation plots.

Package CODA

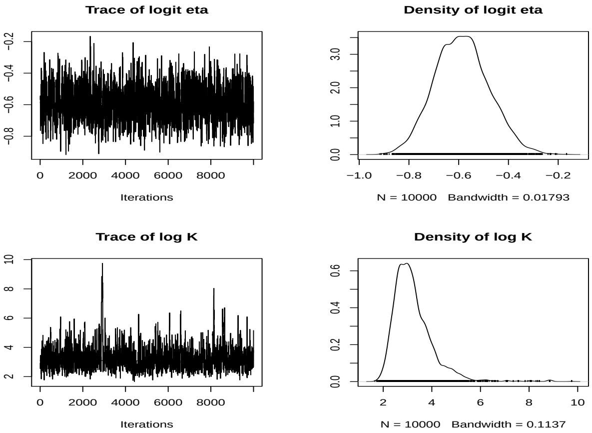
- Create a MCMC object from the matrix of simulated draws by the `mcmc` function.
- `plot` function produces trace plots and density plots of each parameter.
- `autocorr.plot` function produces autocorrelation plots.
- `summary` function produces summaries and correct standard errors for posterior means.

Load in coda package and create MCMC object

```
>  library(coda)
> dimnames(mcmc.fit$par)[[2]]=c("logit eta","log K")
> sim.draws=mcmc(mcmc.fit$par)
```

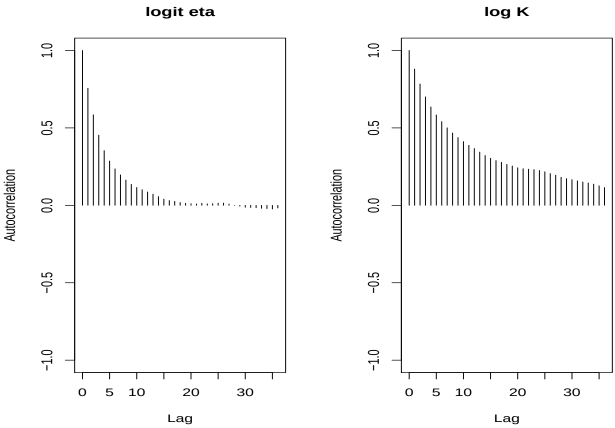
Trace and density plots

> plot(sim.draws)



Autocorrelation plots

```
> autocorr.plot(sim.draws)
```

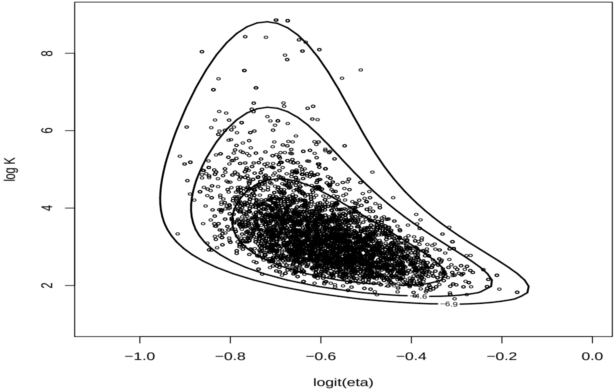


Summaries of output with correct standard errors

```
> summary(sim.draws)

1. Empirical mean and standard deviation for each varia
   plus standard error of the mean:

               Mean     SD Naive SE Time-series SE
logit eta -0.5873 0.1086 0.001086       0.003166
log K      3.2345 0.8462 0.008462       0.052512

2. Quantiles for each variable:

              2.5%     25%     50%     75%   97.5%
logit eta -0.788 -0.6613 -0.5897 -0.5183 -0.3674
log K      2.173  2.6858  3.0681  3.5924  5.1845
```

Demonstration that MCMC sample appears to have found posterior.

```
> mycontour(beta.geom,c(-1.1,0,1,9),y,
+   xlab="logit(eta)",ylab="log K")
> points(mcmc.fit$par,cex=.5)
```



Introduction to Gibbs Sampling

4th Annual Bayesian Biostatistics Conference

Jim Albert
Department of Mathematics and Statistics
Bowling Green State University

Web Counts

Web Counts

- Website for my book at `bayes.bgsu.edu/bcwr`.

Web Counts

- Website for my book at `bayes.bgsu.edu/bcwr`.
- Record the number of visits to my website for each day for a month.

Web Counts

- Website for my book at `bayes.bgsu.edu/bcwr`.
- Record the number of visits to my website for each day for a month.
- Text file `may.june.webhits.txt` contains the day of the week and the visit count for the 31 days between May 23 and June 22 in 2009.

Web Counts

- Website for my book at `bayes.bgsu.edu/bcwr`.
- Record the number of visits to my website for each day for a month.
- Text file `may.june.webhits.txt` contains the day of the week and the visit count for the 31 days between May 23 and June 22 in 2009.
- I read this dataset into R and store it in the variable `data` by the read.table function.

  ```
  > data = read.table("may.june.webhits.txt",
  + header = TRUE, skip = 1)
  ```

Weekday/weekend effect?

Weekday/weekend effect?

- If one looks at the web site counts, one would think there would be a day effect.

Weekday/weekend effect?

- If one looks at the web site counts, one would think there would be a day effect.
- One would think there are fewer visits during the weekend (Saturday and Sunday).

Weekday/weekend effect?

- If one looks at the web site counts, one would think there would be a day effect.
- One would think there are fewer visits during the weekend (Saturday and Sunday).
- Interested in estimating the ratio of the average number of visits during weekdays versus weekends.

Weekday/weekend effect?

- If one looks at the web site counts, one would think there would be a day effect.
- One would think there are fewer visits during the weekend (Saturday and Sunday).
- Interested in estimating the ratio of the average number of visits during weekdays versus weekends.
- Can do this through a simple Poisson model

Two Sample Model

Two Sample Model

- Observe counts $\{y_{Ai}\}$ from a Poisson distribution with mean λ_A, counts $\{y_{Bi}\}$ from a Poisson distribution with mean λ_B.

Two Sample Model

- Observe counts $\{y_{Ai}\}$ from a Poisson distribution with mean λ_A, counts $\{y_{Bi}\}$ from a Poisson distribution with mean λ_B.
- Let n_A and s_A denote number of counts and sum of counts for the first sample; similarly define n_B and s_B.

Two Sample Model

- Observe counts $\{y_{Ai}\}$ from a Poisson distribution with mean λ_A, counts $\{y_{Bi}\}$ from a Poisson distribution with mean λ_B.
- Let n_A and s_A denote number of counts and sum of counts for the first sample; similarly define n_B and s_B.
- Likelihood function is given by

$$L(\lambda_A, \lambda_B) = \exp(-n_A\lambda_A)\lambda_A^{s_A} \exp(-n_B\lambda_B)\lambda_A^{s_B}.$$

Reparameterize

Reparameterize

- Since we are interested in comparison, reparameterize (λ_A, λ_B) to

$$\theta = \lambda_A, \ \gamma = \frac{\lambda_B}{\lambda_A}.$$

Reparameterize

- Since we are interested in comparison, reparameterize (λ_A, λ_B) to

$$\theta = \lambda_A, \ \gamma = \frac{\lambda_B}{\lambda_A}.$$

- Likelihood function in terms of the new parameters is given by

$$L(\theta, \gamma) = \exp(-n_A\theta)\theta^{s_A} \exp\left(-n_B(\theta\gamma)\right)(\theta\gamma)^{s_B}.$$

Reparameterize

- Since we are interested in comparison, reparameterize (λ_A, λ_B) to
$$\theta = \lambda_A, \ \gamma = \frac{\lambda_B}{\lambda_A}.$$
- Likelihood function in terms of the new parameters is given by
$$L(\theta, \gamma) = \exp(-n_A\theta)\theta^{s_A} \exp\left(-n_B(\theta\gamma)\right)(\theta\gamma)^{s_B}.$$
- Focus is on learning about the risk ratio γ.

The Prior

The Prior

- Assume that our beliefs about θ and γ are independent. (Is this a reasonable assumption?)

The Prior

- Assume that our beliefs about θ and γ are independent. (Is this a reasonable assumption?)
- Assume θ is Gamma(a_0, b_0), γ is Gamma(a_g, b_g).

The Prior

- Assume that our beliefs about θ and γ are independent. (Is this a reasonable assumption?)
- Assume θ is Gamma(a_0, b_0), γ is Gamma(a_g, b_g).
- Posterior is the product of the likelihood and prior:

$$g(\theta, \gamma) \propto L(\theta, \gamma) \exp(-b_0\theta)\theta^{a_0-1} \exp(-b_g\gamma)\gamma^{b_g-1}.$$

The Prior

- Assume that our beliefs about θ and γ are independent. (Is this a reasonable assumption?)
- Assume θ is Gamma(a_0, b_0), γ is Gamma(a_g, b_g).
- Posterior is the product of the likelihood and prior:

$$g(\theta, \gamma) \propto L(\theta, \gamma) \exp(-b_0\theta)\theta^{a_0-1} \exp(-b_g\gamma)\gamma^{b_g-1}.$$

- Want to simulate from joint posterior of θ, γ.

How to Simulate from Posterior?

How to Simulate from Posterior?

- Can write posterior density of (γ, θ) in the form

$$POST = \exp\{-(b_0 + n_A + n_B\gamma)\theta\}\theta^{a_0+s_A+s_B-1}$$

$$\times \exp\{-b_g\gamma\}\gamma^{a_g+s_B-1}$$

How to Simulate from Posterior?

- Can write posterior density of (γ, θ) in the form

$$POST = \exp\{-(b_0 + n_A + n_B\gamma)\theta\}\theta^{a_0+s_A+s_B-1}$$

$$\times \exp\{-b_g\gamma\}\gamma^{a_g+s_B-1}$$

- If we knew value of θ, then

$$[\gamma|\theta] \sim Gamma(a_g + s_B, b_g + n_B\theta)$$

How to Simulate from Posterior?

- Can write posterior density of (γ, θ) in the form

$$POST = \exp\{-(b_0 + n_A + n_B\gamma)\theta\}\theta^{a_0+s_A+s_B-1}$$

$$\times \exp\{-b_g\gamma\}\gamma^{a_g+s_B-1}$$

- If we knew value of θ, then

$$[\gamma|\theta] \sim Gamma(a_g + s_B, b_g + n_B\theta)$$

- If we knew value of γ, then

$$[\theta|\gamma] \sim Gamma(a_0 + s_A + s_B, n_A + n_B\gamma + b_0)$$

This suggests a simple way of simulating from the joint posterior

This suggests a simple way of simulating from the joint posterior

- Suppose current simulated draw of θ is $\theta^{(k-1)}$.

This suggests a simple way of simulating from the joint posterior

- Suppose current simulated draw of θ is $\theta^{(k-1)}$.
- Then one cycle of Gibbs sampling consists of

This suggests a simple way of simulating from the joint posterior

- Suppose current simulated draw of θ is $\theta^{(k-1)}$.
- Then one cycle of Gibbs sampling consists of
  - Simulate $\gamma^{(k)}$ from $Gamma(a_g + s_B, b_g + n_B\theta^{(k-1)})$.

This suggests a simple way of simulating from the joint posterior

- Suppose current simulated draw of θ is $\theta^{(k-1)}$.
- Then one cycle of Gibbs sampling consists of
  - Simulate $\gamma^{(k)}$ from $Gamma(a_g + s_B, b_g + n_B\theta^{(k-1)})$.
  - Simulate $\theta^{(k)}$ from
    $Gamma(a_0 + s_A + s_B, n_A + n_B\gamma^{(k)} + b_0)$

This suggests a simple way of simulating from the joint posterior

- Suppose current simulated draw of θ is $\theta^{(k-1)}$.
- Then one cycle of Gibbs sampling consists of
  - Simulate $\gamma^{(k)}$ from $Gamma(a_g + s_B, b_g + n_B\theta^{(k-1)})$.
  - Simulate $\theta^{(k)}$ from $Gamma(a_0 + s_A + s_B, n_A + n_B\gamma^{(k)} + b_0)$
- This constitutes one cycle of GS, producing $(\theta^{(k)}, \gamma^{(k)})$.

Write a function to perform Gibbs Sampling.

Write a function to perform Gibbs Sampling.

- The list S contains the data `yA`, `yB` and the prior hyperparameters `a0`, `b0`, `ag`, `bg`.

Write a function to perform Gibbs Sampling.

- The list S contains the data `yA, yB` and the prior hyperparameters `a0, b0, ag, bg`.
- Compute the sample sizes `nA, nB` and the sums `yA, yB`.

Write a function to perform Gibbs Sampling.

- The list S contains the data `yA, yB` and the prior hyperparameters `a0, b0, ag, bg`.
- Compute the sample sizes `nA, nB` and the sums `yA, yB`.
- Here is the two R commands that implement one Gibbs cycle:

```
gamma=rgamma(1, shape=S$ag+s.B,
                rate=S$bg+n.B*theta)
theta=rgamma(1, shape=S$a0+s.A+s.B,
                rate=S$b0+n.A+n.B*gamma)
```

Add a reasonable starting value and loop for iter iterations.

```
theta=s.A/n.A

for (i in 1:iter)
{
 gamma=rgamma(1, shape=S$ag+s.B,
                 rate=S$bg+n.B*theta)
 theta=rgamma(1, shape=S$a0+s.A+s.B,
                 rate=S$b0+n.A+n.B*gamma)
}
```

Set up storage matrix and store the simulated draws.

```
R=matrix(0,iter,2)
theta=s.A/n.A

for (i in 1:iter)
{ gamma=rgamma(1, shape=S$ag+s.B, rate=S$bg+n.B*theta)
  theta=rgamma(1, shape=S$a0+s.A+s.B,
                  rate=S$b0+n.A+n.B*gamma)
  R[i,]=c(theta,gamma)}
return(R)
```

Add initial statements finding the sample sizes and sums.

```
n.A=length(S$yA); s.A=sum(S$yA)
n.B=length(S$yB); s.B=sum(S$yB)
R=matrix(0,iter,2)
theta=s.A/n.A
for (i in 1:iter)
{ gamma=rgamma(1, shape=S$ag+s.B, rate=S$bg+n.B*theta)
  theta=rgamma(1, shape=S$a0+s.A+s.B,
                  rate=S$b0+n.A+n.B*gamma)
  R[i,]=c(theta,gamma)}
return(R)
```

Finish function poisson.gibbs – inputs are S and iter.

```
poisson.gibbs=function(S,iter)
{ n.A=length(S$yA); s.A=sum(S$yA)
  n.B=length(S$yB); s.B=sum(S$yB)
  R=matrix(0,iter,2)
  theta=s.A/n.A
for (i in 1:iter)
{ gamma=rgamma(1, shape=S$ag+s.B, rate=S$bg+n.B*theta)
  theta=rgamma(1, shape=S$a0+s.A+s.B,
                  rate=S$b0+n.A+n.B*gamma)
  R[i,]=c(theta,gamma)}
return(R)
}
```

The Plan

The Plan

- Save R function in file `poisson.gibbs.R`.

The Plan

- Save R function in file `poisson.gibbs.R`.
- Read in dataset `may.june.webhits.txt`.

The Plan

- Save R function in file `poisson.gibbs.R`.
- Read in dataset `may.june.webhits.txt`.
- Create two datasets.

The Plan

- Save R function in file `poisson.gibbs.R`.
- Read in dataset `may.june.webhits.txt`.
- Create two datasets.
- Create list with data and parameters.

The Plan

- Save R function in file `poisson.gibbs.R`.
- Read in dataset `may.june.webhits.txt`.
- Create two datasets.
- Create list with data and parameters.
- Source in R function.

The Plan

- Save R function in file `poisson.gibbs.R`.
- Read in dataset `may.june.webhits.txt`.
- Create two datasets.
- Create list with data and parameters.
- Source in R function.
- Run `poisson.gibbs`.

The Plan

- Save R function in file `poisson.gibbs.R`.
- Read in dataset `may.june.webhits.txt`.
- Create two datasets.
- Create list with data and parameters.
- Source in R function.
- Run `poisson.gibbs`.
- coda the MCMC output.

Data setup

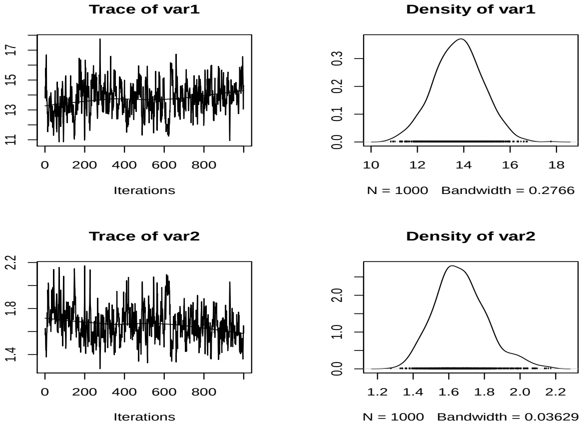
```
> options(width = 35)
> data = read.table("may.june.webhits.txt",
+     header = TRUE, skip = 1)
> weekend.counts = data$Count[data$Day ==
+     "Sat" | data$Day == "Sun"]
> weekday.counts = data$Count[data$Day ==
+     "Mon" | data$Day == "Tues" |
+     data$Day == "Wed" | data$Day ==
+     "Thur" | data$Day == "Fri"]
> stuff = list(yA = weekend.counts,
+     yB = weekday.counts, a0 = 2,
+     b0 = 1, ag = 8, bg = 8)
```

Source and run function

```
> source("poisson.gibbs.R")
> R = poisson.gibbs(stuff, 1000)
```

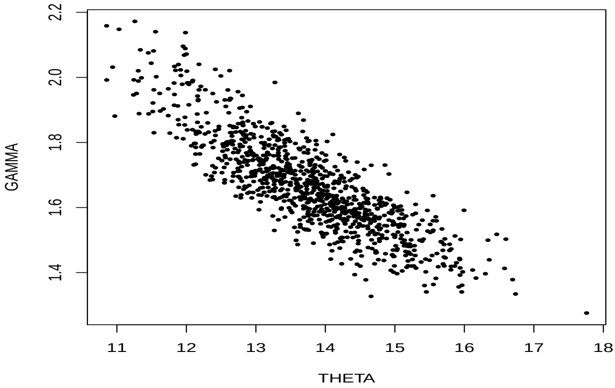
Some MCMC Diagnostics

```
> library(coda)

> plot(mcmc(R))
```



Posterior of (θ, γ)

```
> plot(R, pch = 19, cex = 0.5,
+      xlab = "THETA", ylab = "GAMMA")
```



What have we learned about the risk ratio γ?

Summarize by the 5th, 50th, and 95th percentiles of the simulated draws of γ.

```
> quantile(R[, 2], c(0.05, 0.5,
+     0.95))
      5%      50%      95%
1.441356 1.656326 1.944656
```

MCMC Packages/MCMC Interfaces on R

4th Annual Bayesian Biostatistics Conference

Jim Albert
Department of Mathematics and Statistics
Bowling Green State University

Website Hits

Website Hits

- Using google.com/analytics, I collect the number of visits to my book's website for every day for 8 Wks (2010 data).

	Wk1	Wk2	Wk3	Wk4	Wk5	Wk6	Wk7	Wk8
Sun	11	10	12	15	8	24	14	11
Mon	23	28	27	22	23	18	28	20
Tues	18	21	27	23	27	28	25	19
Wed	51	45	32	19	15	37	34	32
Thurs	31	26	15	17	23	19	21	22
Fri	21	18	13	20	17	20	13	11
Sat	6	12	10	14	7	10	17	5

Website Hits

- Using google.com/analytics, I collect the number of visits to my book's website for every day for 8 Wks (2010 data).

	Wk1	Wk2	Wk3	Wk4	Wk5	Wk6	Wk7	Wk8
Sun	11	10	12	15	8	24	14	11
Mon	23	28	27	22	23	18	28	20
Tues	18	21	27	23	27	28	25	19
Wed	51	45	32	19	15	37	34	32
Thurs	31	26	15	17	23	19	21	22
Fri	21	18	13	20	17	20	13	11
Sat	6	12	10	14	7	10	17	5

- Use a Poisson log-linear model with an additive fit (Day + Week) to explore patterns.

Bayesian Poisson log linear model

Bayesian Poisson log linear model

- Observe counts y_j independent Poisson(λ_j)

Bayesian Poisson log linear model

- Observe counts y_j independent Poisson(λ_j)
- Poisson means satisfy log-linear model

$$\log \lambda_j = \delta + \alpha_{week[j]} + \gamma_{day[j]}$$

Bayesian Poisson log linear model

- Observe counts y_j independent Poisson(λ_j)
- Poisson means satisfy log-linear model

$$\log \lambda_j = \delta + \alpha_{week[j]} + \gamma_{day[j]}$$

- Let $\beta = (\delta, \alpha_2, ..., \alpha_8, \gamma_2, ...\gamma_7)$ be regression coefficient (note I'm using corner-point constraints)

Bayesian Poisson log linear model

- Observe counts y_j independent Poisson(λ_j)
- Poisson means satisfy log-linear model

$$\log \lambda_j = \delta + \alpha_{week[j]} + \gamma_{day[j]}$$

- Let $\beta = (\delta, \alpha_2, ..., \alpha_8, \gamma_2, ...\gamma_7)$ be regression coefficient (note I'm using corner-point constraints)
- Assign β a multivariate normal prior with mean b and precision matrix P.

Read in data matrix

```
> w=read.table("webhits2010.txt",
+   row.names=1,header=TRUE,sep="\t")
> w
      Week1 Week2 Week3 Week4 Week5 Week6 Week7 Week8
Sun      11    10    12    15     8    24    14    11
Mon      23    28    27    22    23    18    28    20
Tues     18    21    27    23    27    28    25    19
Wed      51    45    32    19    15    37    34    32
Thurs    31    26    15    17    23    19    21    22
Fri      21    18    13    20    17    20    13    11
Sat       6    12    10    14     7    10    17     5
```

Get data in ANOVA format

```
> day=factor(rep(1:7,8),labels=row.names(w))
> week=factor(rep(1:8,each=7),labels=dimnames(w)[[2]]
> count=stack(w)$values
> head(cbind(count,day,week))
```

```
     count day week
[1,]    11   1    1
[2,]    23   2    1
[3,]    18   3    1
[4,]    51   4    1
[5,]    31   5    1
[6,]    21   6    1
```

Standard glm fit

```
> fit=glm(count~day+week,family=poisson)
> fit


Coefficients:
(Intercept)       dayMon      dayTues       dayWed
    2.70098      0.58779      0.58248      0.92577
     daySat    weekWeek2    weekWeek3    weekWeek4    w
   -0.25951     -0.00623     -0.16875     -0.21387
  weekWeek7    weekWeek8
   -0.05752     -0.29391

Degrees of Freedom: 55 Total (i.e. Null);  42 Residual
Null Deviance:      220.8
Residual Deviance: 63.58       AIC: 357.7
```

Using LearnBayes

Using LearnBayes

- Write a function to compute the log posterior (uniform prior)

```
loglinearpost=function (beta, d)
{
    X=model.matrix(~day+week,data=d)
    beta=as.vector(beta)
    sum(dpois(d$count,lambda=exp(X%*%beta),log=TRUE
}
```

Using LearnBayes

- Write a function to compute the log posterior (uniform prior)

```
loglinearpost=function (beta, d)
{
     X=model.matrix(~day+week,data=d)
     beta=as.vector(beta)
     sum(dpois(d$count,lambda=exp(X%*%beta),log=TRUE
}
```

- Use the functions `laplace` and `rwmetrop` to summarize the posterior.

R code for LearnBayes

```
> library(LearnBayes)
> d=list(day=day, week=week, count=count)
> fit=laplace(loglinearpost,c(2.7,rep(0,13)),d)
> proposal=list(var=fit$var,scale=0.7)
> start=fit$mode
> my.mcmc.fit=rwmetrop(loglinearpost,proposal,
+   start,10000,d)
```

Takes 57.88 seconds for 10,000 iterations of MCMC.

R Package MCMCpack

R Package MCMCpack

- `MCMCpack` contains functions to perform simulation-based Bayesian fitting for a number of statistical models.

R Package MCMCpack

- `MCMCpack` contains functions to perform simulation-based Bayesian fitting for a number of statistical models.
- Models include linear regression, probit and logistic models, item response models, factor analysis models, multinomial logit, and general-purpose Metropolis fitting. (See package website `http://mcmcpack.wustl.edu/`.)

R Package MCMCpack

- `MCMCpack` contains functions to perform simulation-based Bayesian fitting for a number of statistical models.
- Models include linear regression, probit and logistic models, item response models, factor analysis models, multinomial logit, and general-purpose Metropolis fitting. (See package website `http://mcmcpack.wustl.edu/`.)
- Functions are programmed in compiled C++ to maximize computational efficiency.

R Package MCMCpack

- `MCMCpack` contains functions to perform simulation-based Bayesian fitting for a number of statistical models.
- Models include linear regression, probit and logistic models, item response models, factor analysis models, multinomial logit, and general-purpose Metropolis fitting. (See package website `http://mcmcpack.wustl.edu/`.)
- Functions are programmed in compiled C++ to maximize computational efficiency.
- My experience with `MCMCpack` is generally positive, but there are limitations in the models.

MCMCpack function MCMCpoisson

MCMCpack function MCMCpoisson

- Fits a Bayesian Poisson log-linear model by a MCMC random-walk algorithm.

MCMCpack function MCMCpoisson

- Fits a Bayesian Poisson log-linear model by a MCMC random-walk algorithm.
- Assigns a multivariate normal prior on β.

MCMCpack function MCMCpoisson

- Fits a Bayesian Poisson log-linear model by a MCMC random-walk algorithm.
- Assigns a multivariate normal prior on β.
- The function returns a `mcmc` object that can be analyzed using the `coda` package.

MCMCpack function MCMCpoisson

- Fits a Bayesian Poisson log-linear model by a MCMC random-walk algorithm.
- Assigns a multivariate normal prior on β.
- The function returns a `mcmc` object that can be analyzed using the `coda` package.
- As an option, it returns estimate at the marginal likelihood, that can be used to compare models by Bayes factors.

Our example using MCMCpoisson

Our example using MCMCpoisson

- Assuming a constant prior for β, so `b0 = 0`, `B0 = 0`

Our example using MCMCpoisson

- Assuming a constant prior for β, so `b0 = 0`, `B0 = 0`
- Play with `tune` input (equivalent to `scale` in `rwmetrop`) to get reasonable acceptance rate.

```
> mcmc.fit=MCMCpoisson(count~day+week,
+     burnin = 1000, mcmc = 10000,
+     thin = 1, tune = 0.7, verbose = 0,
+     seed = NA,  beta.start = NA,
+     b0 = 0, B0 = 0, marginal.likelihood = "none")
```

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The Metropolis acceptance rate for beta was 0.20727
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

Comments about MCMCpack

Comments about MCMCpack

- Seems like a nice package for fitting a wide variety of regression models.

Comments about MCMCpack

- Seems like a nice package for fitting a wide variety of regression models.
- It is fast – 10,000 MCMC iterations of Poisson log-linear regression took 0.39 seconds (contrast with 57.88 seconds using `LearnBayes`)

Comments about MCMCpack

- Seems like a nice package for fitting a wide variety of regression models.
- It is fast – 10,000 MCMC iterations of Poisson log-linear regression took 0.39 seconds (contrast with 57.88 seconds using `LearnBayes`)
- Outputs the simulation matrix of β. So it is easy to find the marginal posterior of any function $h(\beta)$ of interest.

Comments about MCMCpack

- Seems like a nice package for fitting a wide variety of regression models.
- It is fast – 10,000 MCMC iterations of Poisson log-linear regression took 0.39 seconds (contrast with 57.88 seconds using `LearnBayes`)
- Outputs the simulation matrix of β. So it is easy to find the marginal posterior of any function $h(\beta)$ of interest.
- But one sacrifices some flexibility of the Bayesian approach – why should we be limited to using a normal prior?

R Interfaces with WinBUGS/OpenBUGS

R Interfaces with WinBUGS/OpenBUGS

- WinBUGS is a general Windows program for implementing MCMC calculations. (Many Bayesian courses are entirely based on using WinBUGS.)

R Interfaces with WinBUGS/OpenBUGS

- WinBUGS is a general Windows program for implementing MCMC calculations. (Many Bayesian courses are entirely based on using WinBUGS.)
- OpenBUGS is the open-source version of WinBUGS.

R Interfaces with WinBUGS/OpenBUGS

- WinBUGS is a general Windows program for implementing MCMC calculations. (Many Bayesian courses are entirely based on using WinBUGS.)
- OpenBUGS is the open-source version of WinBUGS.
- There are several R interfaces with WinBUGS and OpenBUGS. These are attractive since one can use R to get the data ready and summarize and display MCMC output.

R Interfaces with WinBUGS/OpenBUGS

- WinBUGS is a general Windows program for implementing MCMC calculations. (Many Bayesian courses are entirely based on using WinBUGS.)
- OpenBUGS is the open-source version of WinBUGS.
- There are several R interfaces with WinBUGS and OpenBUGS. These are attractive since one can use R to get the data ready and summarize and display MCMC output.
- I'll illustrate one R package BRugs that installs OpenBUGS and the R interface.

Basic Steps to Use R Interface with OpenBUGS

Basic Steps to Use R Interface with OpenBUGS

- Write a short script defining the Bayesian model. This script is a file in the current working directory.

Basic Steps to Use R Interface with OpenBUGS

- Write a short script defining the Bayesian model. This script is a file in the current working directory.
- Specify the data that will be used, the parameters of the model, and initial values of the parameters in the MCMC simulation.

Basic Steps to Use R Interface with OpenBUGS

- Write a short script defining the Bayesian model. This script is a file in the current working directory.
- Specify the data that will be used, the parameters of the model, and initial values of the parameters in the MCMC simulation.
- A special R function `bugs` is used to perform the MCMC simulation (using the OpenBUGS engine)

Basic Steps to Use R Interface with OpenBUGS

- Write a short script defining the Bayesian model. This script is a file in the current working directory.
- Specify the data that will be used, the parameters of the model, and initial values of the parameters in the MCMC simulation.
- A special R function `bugs` is used to perform the MCMC simulation (using the OpenBUGS engine)
- One obtains a matrix of simulated draws that one can summarize using the `coda` package.

The OpenBUGS model file “webhits.bug”

```
model
{
for (i in 1:n)
  { count[i] ~ dpois(lam[i])
     log(lam[i]) <- beta[1] + X[i,2]*beta[2] + X[i,3]*b
     X[i,4]*beta[4] + X[i,5]*beta[5] + X[i,6]*beta[6] +
     X[i,7]*beta[7] + X[i,8]*beta[8] + X[i,9]*beta[9] +
     X[i,10]*beta[10] + X[i,11]*beta[11] + X[i,12]*beta
     X[i,13]*beta[13] + X[i,14]*beta[14]
  }
for (j in 1:14)
  {  beta[j] ~ dnorm(0, 0.001)}
}
```

The R code

Specify the data, the parameters, and the initial parameter values:

```
> library(arm)
> library(BRugs)

# have already defined count and X

> n=56
> web.data = list("n","count","X")
> web.parameters = c("beta")
> web.inits = function(){
+   list(beta=fit$coef)}
```

R Code to run the MCMC

```
> web.fit = bugs(web.data, web.inits, web.parameters,
+   "webhits.bug", n.chains=1, n.iter=10000, debug=TRU
+    n.burnin=1000, program = "openbugs")
```

Now we can use coda to summarize the MCMC output:

```
> summary(mcmc(web.fit$sims.matrix))
```

Comments about WinBUGS/OpenBUGS

Comments about WinBUGS/OpenBUGS

- Works well for a large class of problems (such as multilevel models).

Comments about WinBUGS/OpenBUGS

- Works well for a large class of problems (such as multilevel models).
- Speed? For the example problem, it took 32 seconds for 10,000 iterations. (Between speed of `rwmetrop` and `MCMCpack`.)

Comments about WinBUGS/OpenBUGS

- Works well for a large class of problems (such as multilevel models).
- Speed? For the example problem, it took 32 seconds for 10,000 iterations. (Between speed of `rwmetrop` and `MCMCpack`.)
- As with any MCMC simulation, have to perform diagnostics on the output.

Comments about WinBUGS/OpenBUGS

- Works well for a large class of problems (such as multilevel models).
- Speed? For the example problem, it took 32 seconds for 10,000 iterations. (Between speed of `rwmetrop` and `MCMCpack`.)
- As with any MCMC simulation, have to perform diagnostics on the output.
- There are some programming tricks for special problems.

Comments about WinBUGS/OpenBUGS

- Works well for a large class of problems (such as multilevel models).
- Speed? For the example problem, it took 32 seconds for 10,000 iterations. (Between speed of `rwmetrop` and `MCMCpack`.)
- As with any MCMC simulation, have to perform diagnostics on the output.
- There are some programming tricks for special problems.
- For my introductory Bayes class, I prefer to use R procedures (like the ones in the `LearnBayes` package), since the MCMC algorithms are more transparent.

The Rest of the Webcounts Story

The Rest of the Webcounts Story

- I was really interested if the Poisson log-linear model was a suitable fit to the counts.

The Rest of the Webcounts Story

- I was really interested if the Poisson log-linear model was a suitable fit to the counts.
- Is there overdispersion?

The Rest of the Webcounts Story

- I was really interested if the Poisson log-linear model was a suitable fit to the counts.
- Is there overdispersion?
- Use posterior predictive checking – use the checking "chi-square" function

$$T(y, \beta) = \sum_j \frac{(y_j - \lambda_j)^2}{\lambda_j},$$

where $\lambda_j = x_j \beta$.

The Rest of the Webcounts Story

- I was really interested if the Poisson log-linear model was a suitable fit to the counts.
- Is there overdispersion?
- Use posterior predictive checking – use the checking "chi-square" function

$$T(y, \beta) = \sum_j \frac{(y_j - \lambda_j)^2}{\lambda_j},$$

  where $\lambda_j = x_j \beta$.
- Want to compare the distributions $T(y^*, \beta)$ with $T(y, \beta)$

Easy to do model checking

Easy to do model checking

- The variable `mcmc.fit` contains the matrix of simulated draws from the posterior of β.

Easy to do model checking

- The variable `mcmc.fit` contains the matrix of simulated draws from the posterior of β.
- The function `T.predicted` simulates a value of $T(y^*, \beta)$ from posterior predictive distribution.

```
> X=model.matrix(~day+week)
> T.predicted=function(j)
+ {
+ lambda=exp(X%*%mcmc.fit[j,])
+ ys=rpois(length(lambda),lambda)
+ sum((ys-lambda)^2/lambda)
+ }
```

Easy to do model checking

- The variable `mcmc.fit` contains the matrix of simulated draws from the posterior of β.
- The function `T.predicted` simulates a value of $T(y^*, \beta)$ from posterior predictive distribution.

```
> X=model.matrix(~day+week)
> T.predicted=function(j)
+ {
+ lambda=exp(X%*%mcmc.fit[j,])
+ ys=rpois(length(lambda),lambda)
+ sum((ys-lambda)^2/lambda)
+ }
```

- By using `sapply`, simulate 10,000 draws of $T(y^*, \beta)$

```
> TP=sapply(1:10000,T.predicted)
```

Model checking, continued

Model checking, continued

- The function `T.observed` simulates a value of $T(y, \beta)$ from posterior distribution.

```
> T.observed=function(j)
+ {
+ lambda=exp(X%*%mcmc.fit[j,])
+ sum((count-lambda)^2/lambda)
+ }
```

Model checking, continued

- The function `T.observed` simulates a value of $T(y, \beta)$ from posterior distribution.

```
> T.observed=function(j)
+ {
+ lambda=exp(X%*%mcmc.fit[j,])
+ sum((count-lambda)^2/lambda)
+ }
```

- By using `sapply`, simulate 10,000 draws of $T(y, \beta)$

```
> Tobs=sapply(1:10000,T.observed)
```

Model checking, continued

- The function `T.observed` simulates a value of $T(y, \beta)$ from posterior distribution.

```
> T.observed=function(j)
+ {
+ lambda=exp(X%*%mcmc.fit[j,])
+ sum((count-lambda)^2/lambda)
+ }
```

- By using `sapply`, simulate 10,000 draws of $T(y, \beta)$

```
> Tobs=sapply(1:10000,T.observed)
```

- Compute probability $P(T(y, \beta) > T(y^*, \beta))$.

```
> mean(Tobs>TP)
[1] 0.953
```

Model checking, continued

- The function `T.observed` simulates a value of $T(y, \beta)$ from posterior distribution.

```
> T.observed=function(j)
+ {
+ lambda=exp(X%*%mcmc.fit[j,])
+ sum((count-lambda)^2/lambda)
+ }
```

- By using `sapply`, simulate 10,000 draws of $T(y, \beta)$

```
> Tobs=sapply(1:10000,T.observed)
```

- Compute probability $P(T(y, \beta) > T(y^*, \beta))$.

```
> mean(Tobs>TP)
[1] 0.953
```

- Conclusion: the observed data displays more variability than predicted from the Poisson model.

What next?

What next?

- Use some alternative model such as a negative binomial regression.

What next?

- Use some alternative model such as a negative binomial regression.
- Can estimate overdispersion by the negative binomial parameter.

What next?

- Use some alternative model such as a negative binomial regression.
- Can estimate overdispersion by the negative binomial parameter.
- Obtain better estimates at the week and day effects (more realistic standard errors).