Bayesian Thinking: Fundamentals, Regression and Multilevel Modeling

Jim Albert and Jingchen (Monika) Hu

November 2020

Regression Models for Count Data

- Response variable y is a count
- Traditional sampling model is Poisson, where the log means satisfy a linear regression model
- Data is typically overdispersed – see more variability in counts than predicted by Poisson
- We'll describe several ways to handle overdispersion

Famous Bayesian Study

- Mosteller and Wallace (1963)
- Authorship problem: 85 Federalist papers wrote to promote ratification of U.S. constitution
- Some were written by Alexander Hamilton and some were written by James Madison
- Who wrote the "unknown" Federalist papers – Madison or Hamilton?
- Illustrated Bayesian reasoning to determine authorship

Focus on the “Filler Words”

- Use of some words depend on the content of the essay
- Other words, so-called filler words, are less influenced by the essay content
- Focus on the use of the word “can” by Hamilton

Read Data

```
library(tidyverse)
library(ProbBayes)
d <- filter(federalist_word_study,
            Authorship == "Hamilton",
            word == "can") %>%
     select(Name, Total, N)
head(d)
```

```
##                      Name Total  N
## 65     Federalist No. 1  1622  3
## 1526 Federalist No. 11  2511  5
## 2437 Federalist No. 12  2171  2
## 3125 Federalist No. 13   970  4
## 4256 Federalist No. 15  3095 14
## 5530 Federalist No. 16  2047  1
```

Poisson Model

- Assume the number of occurrences of "can" in the jth document y_j is Poisson with mean $n_j\lambda/1000$.
- λ is true rate of "can" among 1000 words
- Poisson sampling density

$$f(y_j|\lambda) = \frac{(n_j\lambda/1000)^{y_j}\exp(-n_j\lambda/1000)}{y_j!}.$$

Log-Linear Model

- On log scale, the Poisson mean can be written

$$\log \lambda = \log(n_i/1000) + \beta$$

- A generalized linear model with Poisson sampling, log link, intercept model with an offset of $\log(n_i/1000)$.

Prior

- Assume know little about location of λ
- We complete this model by assigning the prior

$$\log \lambda \sim N(0, 2)$$

Fitting Model

- Use the `brm()` function with `family = poisson`, specifying the offset `N`, and specifying the prior by use of the "prior" argument.

```
library(brms)
fit <- brm(data = d, family = poisson,
    N ~ offset(log(Total / 1000)) + 1,
    prior = c(prior(normal(0, 2),
                    class = Intercept)),
              refresh = 0
    )
```

Saving Posterior Draws

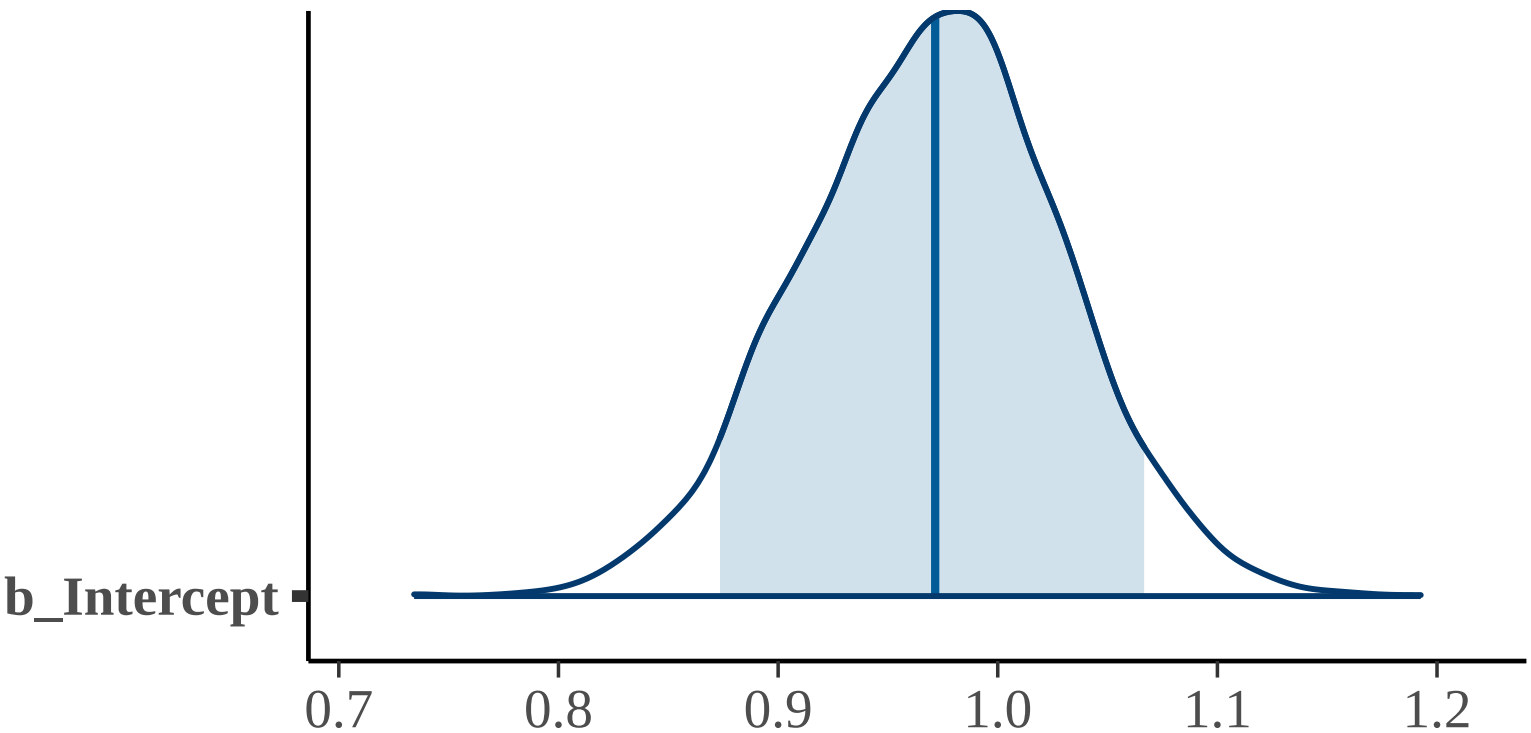
- Save `post` as a matrix of simulated draws.

```
post <- posterior_samples(fit)
```

Posterior Plot

- Function `mcmc_areas()` displays a density estimate of the simulated draws and shows the location of a 90% probability interval.

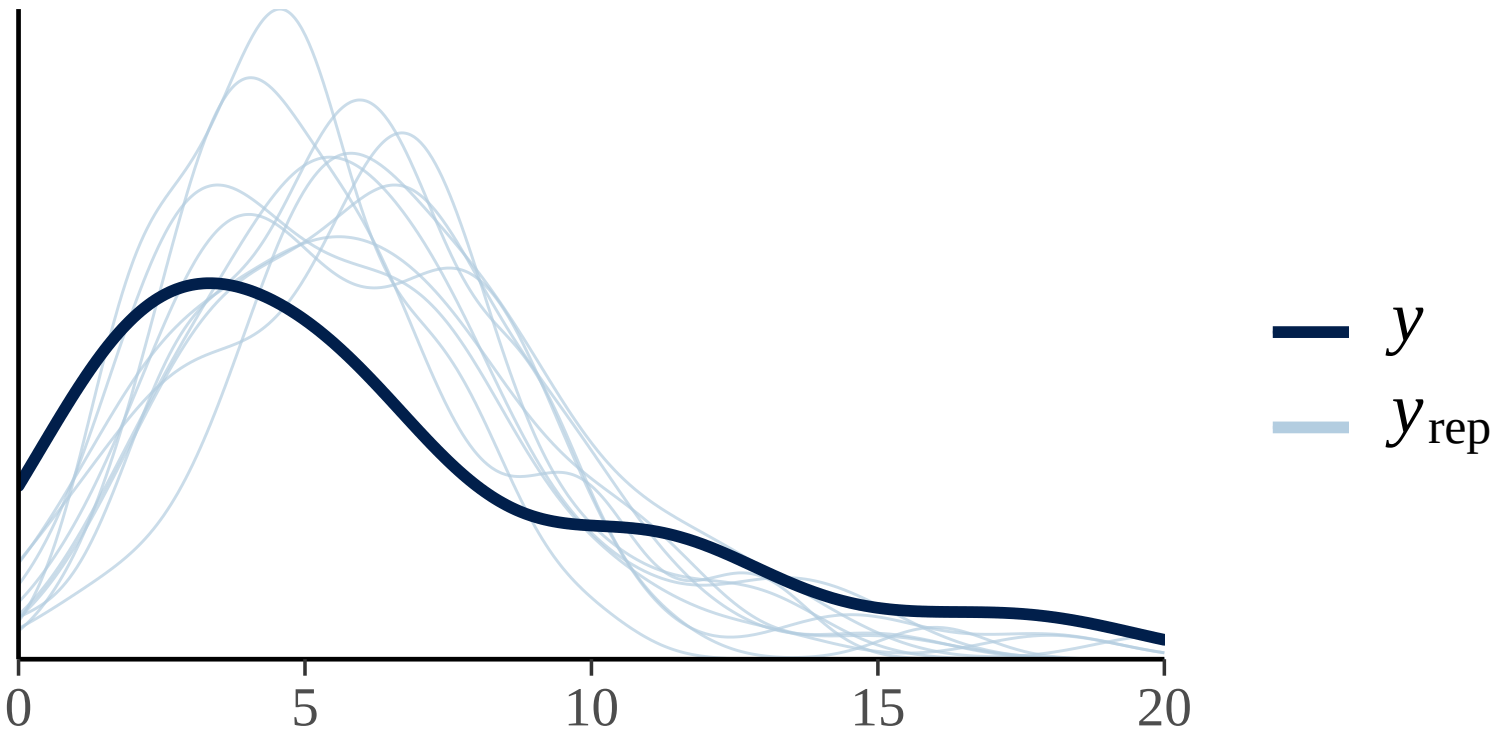
```
library(bayesplot)
mcmc_areas(post, pars = "b_Intercept",
           prob = 0.90)
```



Model Checking

- To check if the Poisson sampling model is appropriate we illustrate several posterior predictive (PP) checks.
- Plot density estimates for 10 replicated samples from the PP distribution of y and overlay the observed count distribution.

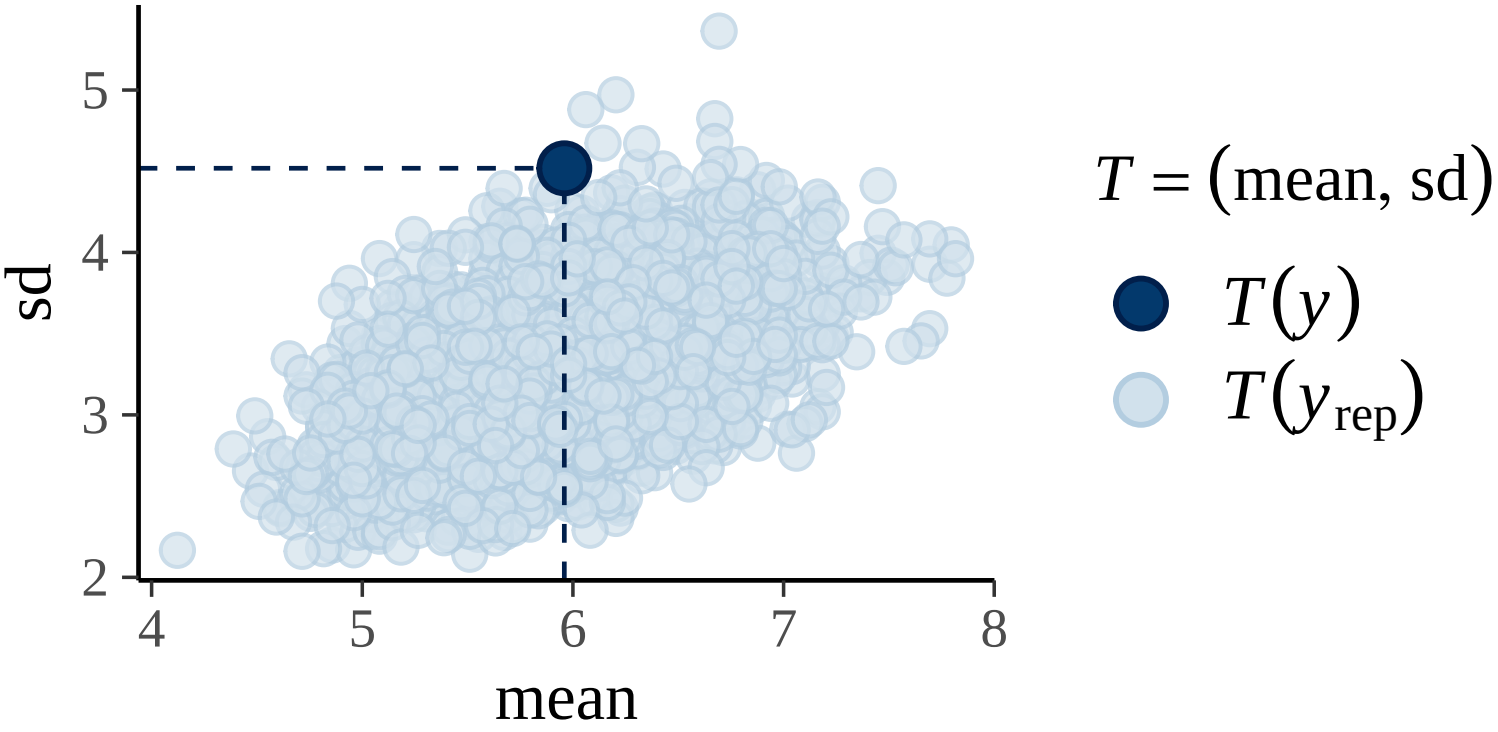
```
pp_check(fit)
```



Overdispersion?

- Use $(\bar{y}, s_y)$ as a checking function. The scatterplot represents values of $(\bar{y}, s_y)$ from the PP distribution of replicated data, and the dot is the observed value of $(\bar{y}, s_y)$.

```
pp_check(fit, type = "stat_2d")
```



- The observed data shows more variability than predicted from the

Consider Negative Binomial sampling

- Assume y_j is Negative Binomial (NB) with parameters p_j and α
- Reparametrize p_j to β

$$p_j = \frac{\beta}{\beta + n_j/1000}.$$

$$f(y_j|\alpha, \beta) = \frac{\Gamma(y_j + \alpha)}{\Gamma(\alpha)} p_j^{\alpha}(1 - p_j)^{y_j}$$

NB is Generalization of Poisson

- Mean of y_j is

$$E(y_j) = \mu_j = \frac{n_j}{1000}\frac{\alpha}{\beta}$$

- Variance of y_j is

$$Var(y_j) = \mu_j \left(1 + \frac{n_j}{1000\beta}\right).$$

- Parameter $\mu = \alpha/\beta$ is true rate per 1000 words
- β is overdispersion parameter

Negative Binomial Sampling

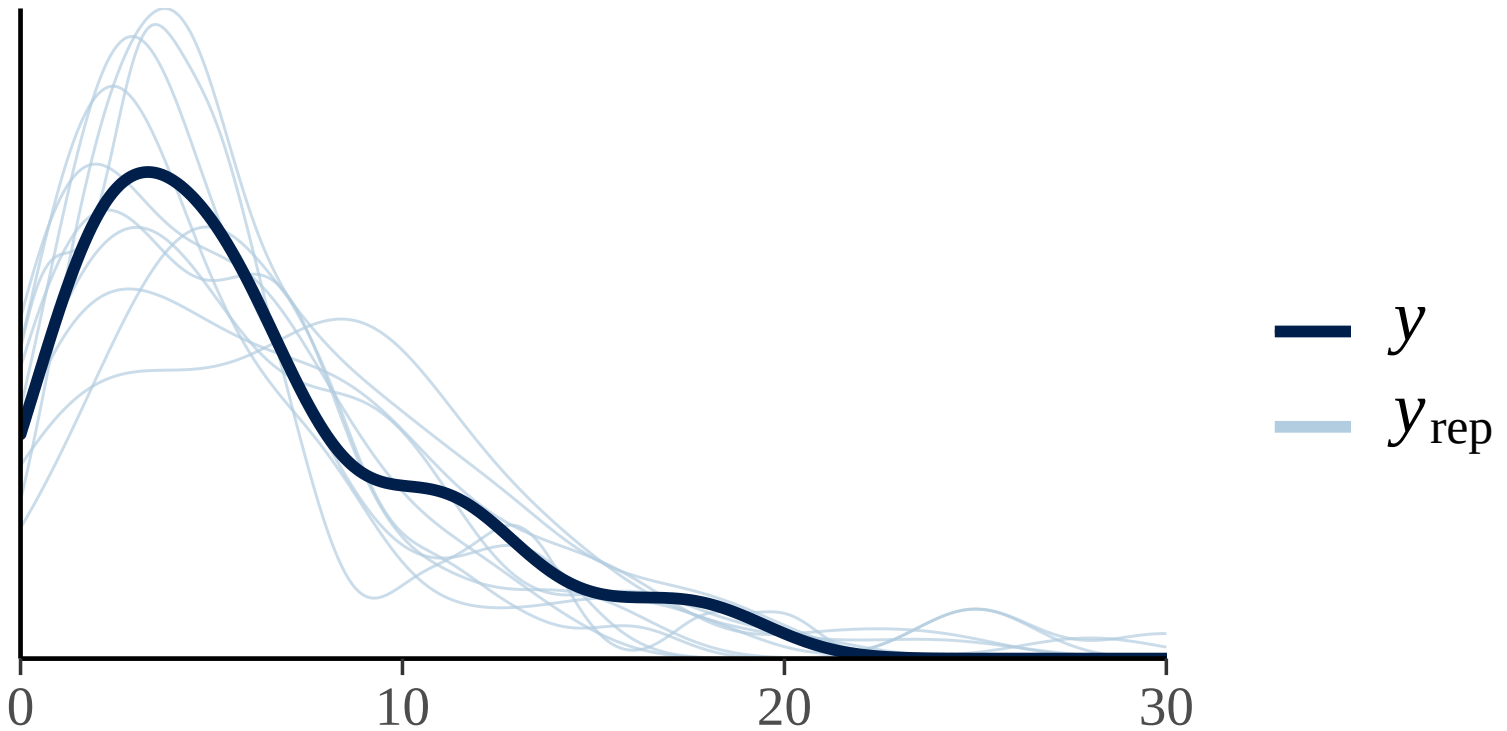
- Fit the negative binomial model with the `brm()` function with the "family = negbinomial" option.

```
fit_nb <- brm(data = d, family = negbinomial,
              N ~ offset(log(Total / 1000)) + 1,
              refresh = 0)
```

Posterior Predictive Checks

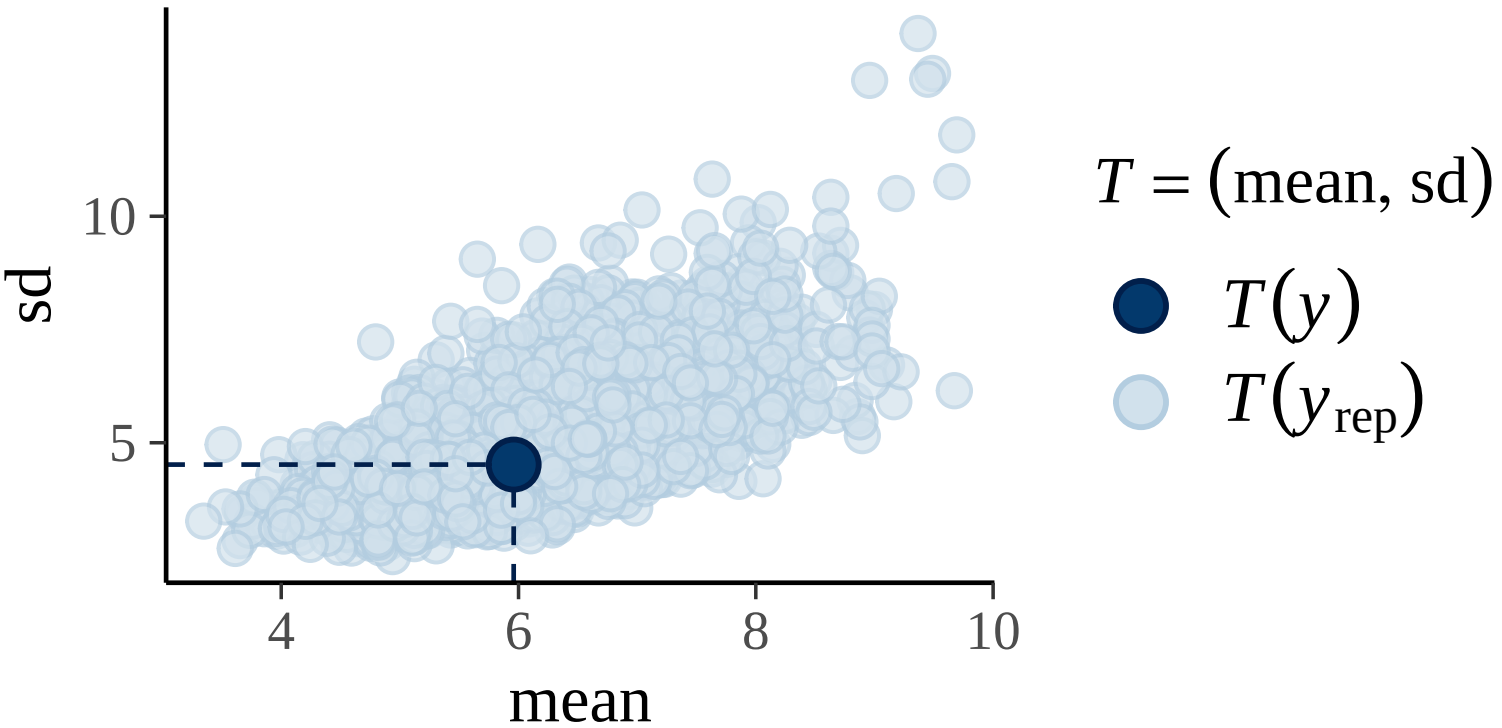
- Try the same posterior predictive checks as before. The message is that the negative binomial sampling model is a better fit to these data.

```
pp_check(fit_nb)
```



Posterior Predictive Checks

```
pp_check(fit_nb, type = "stat_2d")
```



Compare Authors's Use of a Word

- Compare Madison and Hamilton use of the word "can". The data frame d2 contains only the word data for the essays that were known to be written by Hamilton or Madison.

```
federalist_word_study %>%
  filter(word == "can",
         Authorship %in% c("Hamilton", "Madison")) -> d2
```

Model - Two Author Comparison

- Fit a regression model for the mean use of "can", where the one predictor is the categorical variable "Authorship".

```
fit_nb <- brm(data = d2, family = negbinomial,
              N ~ offset(log(Total / 1000)) +
             Authorship ,
              refresh = 0)
```

Comparing Authors

- By summarizing the fit, we can see if the two authors differ in their use of the word "can" in their writings.

```
summary(fit_nb)
```

```
##  Family: negbinomial
##   Links: mu = log; shape = identity
## Formula: N ~ offset(log(Total/1000)) + Authorship
##    Data: d2 (Number of observations: 74)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; th
##          total post-warmup samples = 4000
##
## Population-Level Effects:
##                   Estimate Est.Error l-95% CI u-95% CI Rhat
## Intercept             1.00      0.09     0.82     1.19 1.00
## AuthorshipMadison    -0.08      0.16    -0.40     0.24 1.00
##
## Family Specific Parameters:
```

Takeaways

- Hamilton more likely to use words “upon”, “to”, “this”, “there”, “any”, and “an”
- Madison more likely to use “on”, “by”, and “also”
- Inconclusive for the remaining words (may, his, from, can, and also)

Baseball Prediction Problem

- In baseball, much of the run scoring is due to home runs.
- In the 2020 World Series, I am interested in predicting the total number of home runs hit.

Start with a Poisson Model

- Let y_{ij} be the number of home runs hit by the ith team in the jth game during the 2020 season.
- Let n_{ij} denote the number of opportunities (balls in play)
- Assume $y_{ij} \sim Poisson(n_{ij}\lambda_{ij})$
- Teams differ on their home run ability.
- There is a clear effect of the ballpark.

Random Effects Model

- Log-linear model

$$\log \lambda_{ij} = \log n_{ij} + \beta_0 + Team_i + Park_j$$

- Assume team effects $Team_1, ..., Team_{30}$ are $N(0, \sigma_T)$
- Assume park effects $Park_1, ..., Park_{30}$ are $N(0, \sigma_P)$.
- Assign prior to $(\beta_0, \sigma_T, \sigma_P)$.

Data

- Available at http://bayesball.github.io/baseball/2020homeruns.csv
- Contains number of home runs hit by each team for each game of 2020 season
- Variables `HR`, `N` (number of balls in play), `BAT_TEAM`, `venue_name`

```
S2 <- read_csv("http://bayesball.github.io/baseball/2020homeru
```

Fit Model Using Stan

```
bfit2 <- brm(HR ~ offset(log(N)) +
                  (1 | BAT_TEAM) +
                  (1 | venue_name),
             data = S2,
             family = poisson,
             refresh = 0)
```

Priors?

```
prior_summary(bfit2)
```

```
##                   prior     class       coef       group resp c
##  student_t(3, 0, 2.5) Intercept
##  student_t(3, 0, 2.5)        sd
##  student_t(3, 0, 2.5)        sd             BAT_TEAM
##  student_t(3, 0, 2.5)        sd Intercept   BAT_TEAM
##  student_t(3, 0, 2.5)        sd            venue_name
##  student_t(3, 0, 2.5)        sd Intercept venue_name
##        source
##       default
##       default
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
```

Summary of posterior fit

```
bfit2
```

```
##  Family: poisson
##   Links: mu = log
## Formula: HR ~ offset(log(N)) + (1 | BAT_TEAM) + (1 | venue_
##    Data: S2 (Number of observations: 1796)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; th
##          total post-warmup samples = 4000
##
## Group-Level Effects:
## ~BAT_TEAM (Number of levels: 30)
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bul
## sd(Intercept)     0.14      0.04     0.08     0.22 1.00
##
## ~venue_name (Number of levels: 30)
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bul
## sd(Intercept)     0.13      0.04     0.07     0.21 1.00
```

Collect posterior draws

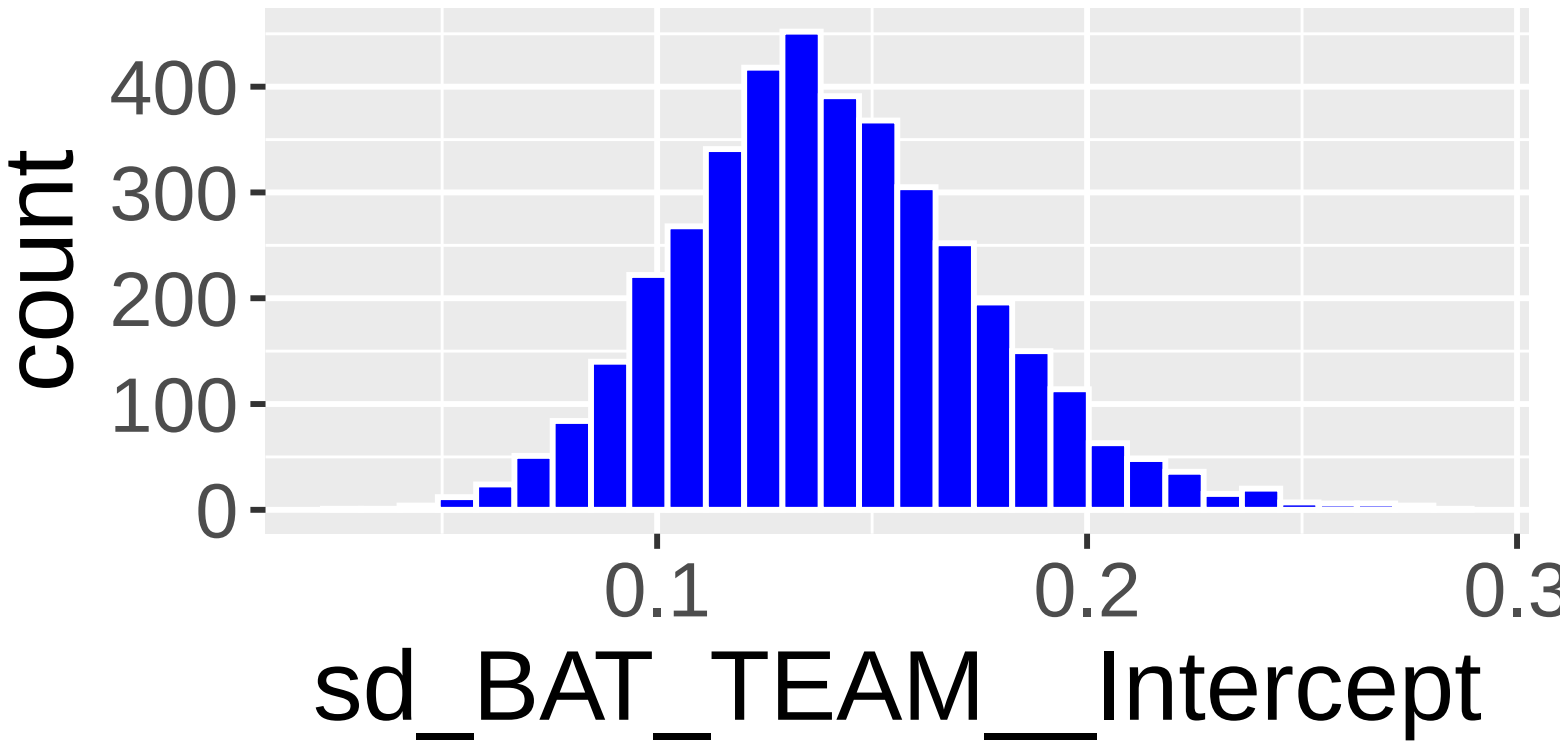
```
draws <- data.frame(bfit2)
head(draws)
```

```
##   b_Intercept sd_BAT_TEAM__Intercept sd_venue_name__Interce
## 1   -3.031669                0.1863260                0.11767
## 2   -2.978556                0.1476703                0.14145
## 3   -2.997755                0.1307075                0.14062
## 4   -2.988339                0.1064862                0.09957
## 5   -2.924203                0.1835403                0.14778
## 6   -2.888154                0.2074746                0.16721
##   r_BAT_TEAM.ARI.Intercept. r_BAT_TEAM.ATL.Intercept. r_BA
## 1               -0.31360042                 0.1241989
## 2               -0.19446204                 0.2628169
## 3               -0.08211085                 0.2327165
## 4               -0.17817782                 0.1254740
## 5               -0.21980544                 0.1988852
## 6               -0.19526046                 0.1801995
```

Model Fits

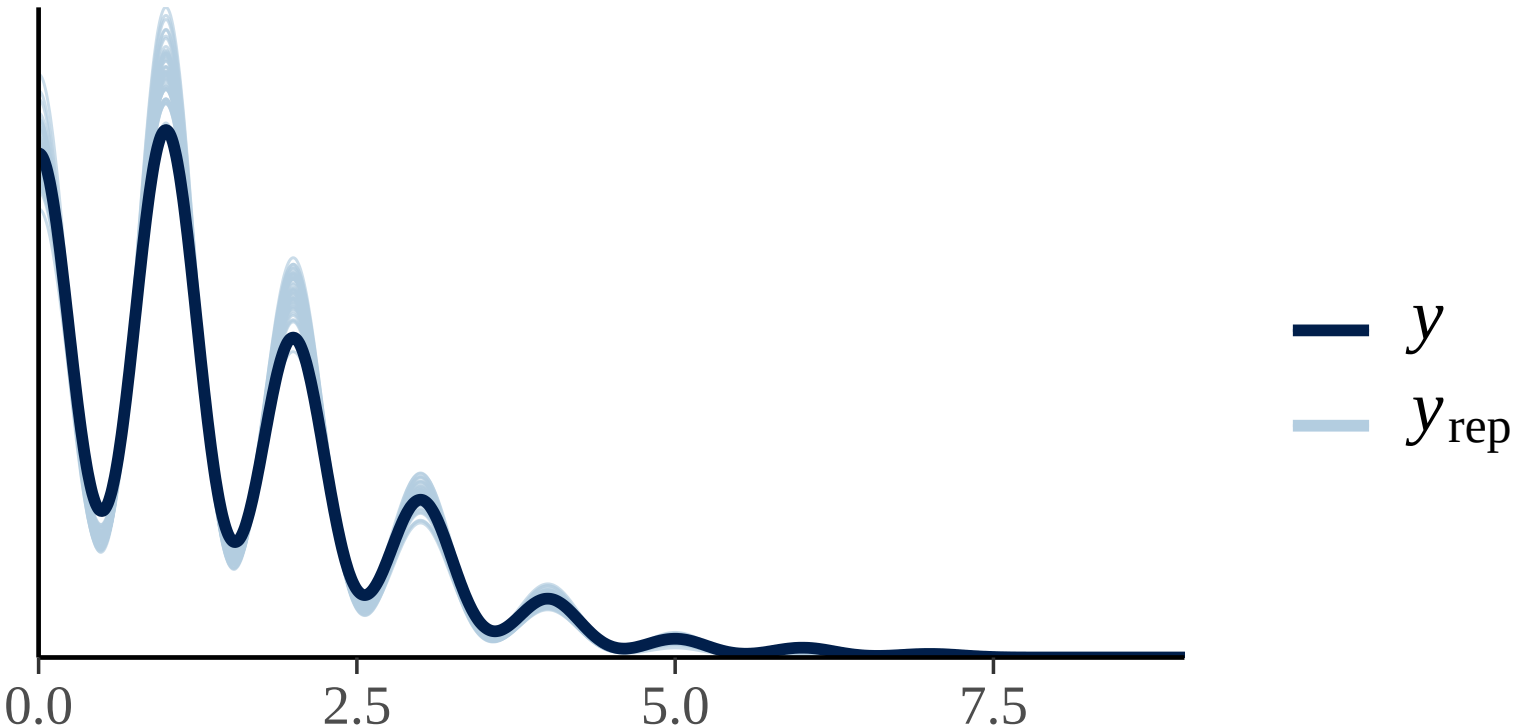
Draws MCMC diagnostics for intercept and standard deviations

```
ggplot(draws, aes(sd_BAT_TEAM__Intercept)) +
  geom_histogram(color = "white",
                 fill = "blue") +
  increasefont()
```



Predictive checks

```
pp_check(bfit2, nsamples = 50)
```



Prediction

- Predict the number of home runs in the playoffs
- Inputs are the two teams, the ballpark, and the number of balls in play for each team

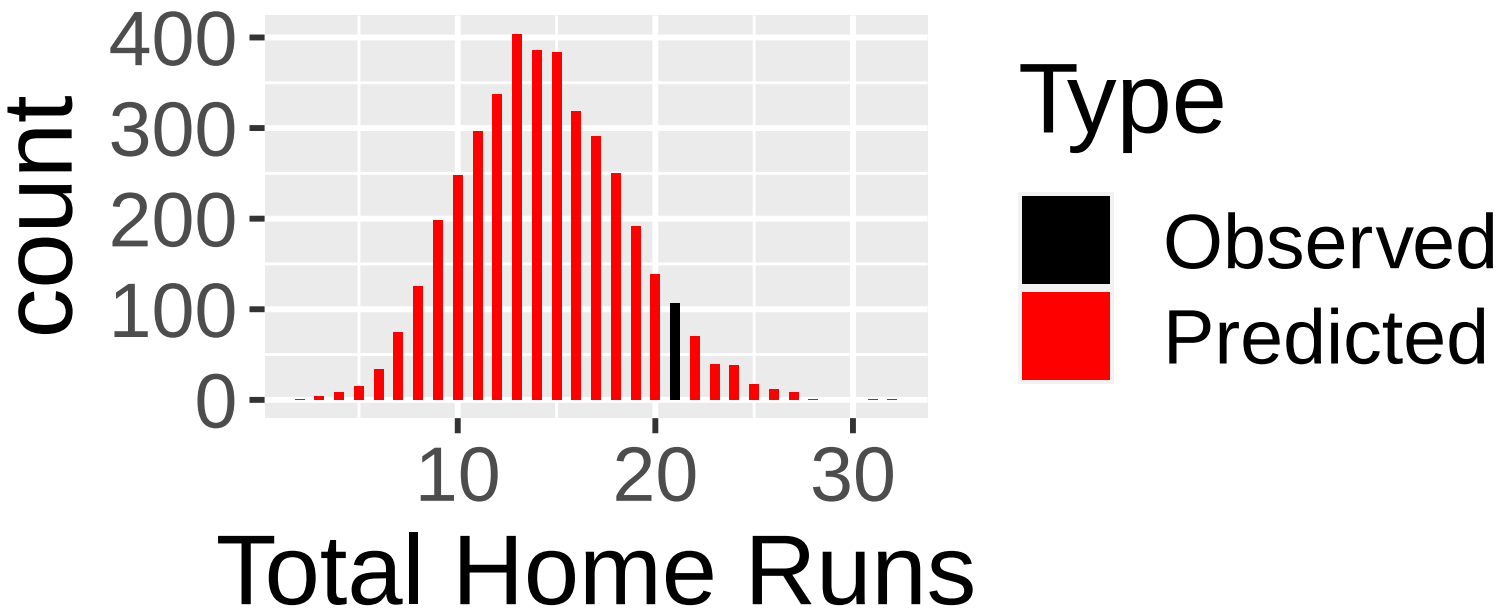
Simulate from Posterior Predictive

- First, simulate values of the random effects $Team_i$, $Team_j$, and $Park_j$ from the posterior distribution.
- Using the balls-in-play, have simulated values of the rates λ
- Simulate home run rates from the Poisson sampling distribution

Illustrate with a best-of-five series

```
predict_hr(draws,
           "NYY", "TB", "Petco.Park", 120, 114,
           10, 11)
```

YY vs TB: 90% Interval: (8, 21)



Summing Up

- Although Poisson is the canonical distribution for count data, typically data is overdispersed.
- One way of handling overdispersion is through another sampling model such as negative binomial.
- Another way is to introduce random effects that can soak up the extra variability.
- Illustrated both Bayesian inference and prediction.