Stan:
Probabilistic Modeling & Bayesian Inference

Development Team

Andrew Gelman, **Bob Carpenter**, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Allen Riddell, Marco Inacio, Jeffrey Arnold, **Mitzi Morris**, Rob Trangucci, Rob Goedman, Brian Lau, Jonah Sol Gabry, Robert L. Grant, Krzysztof Sakrejda, Aki Vehtari, Rayleigh Lei, Sebastian Weber, Charles Margossian, Vincent Picaud, Imad Ali, **Sean Talts**, Ben Bales, Ari Hartikainen, Matthijs Vàkàr, Andrew Johnson, Dan Simpson

Stan 2.17 (November 2017) http://mc-stan.org

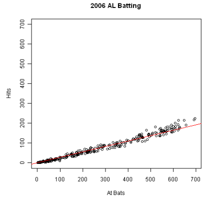
Hierarchical Models

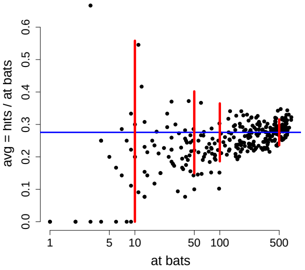
Baseball At-Bats

- For example, consider baseball batting ability.
  - Baseball is sort of like cricket, but with round bats, a one-way field, stationary “bowlers”, four bases, short games, and no draws
- Batters have a number of “at-bats” in a season, out of which they get a number of “hits” (hits are a good thing)
- Nobody with higher than 40% success rate since 1950s.
- No player (excluding “bowlers”) bats much less than 20%.
- Same approach applies to hospital pediatric surgery complications (a BUGS example), reviews on Yelp, test scores in multiple classrooms, . . .

Baseball Data

- Hits vs. at bats for 2006 AL (no bowlers); line at average
- not much variation
- variation related to number of trials
- success rate increases with number of trials
  - at bats predictively in nuanced models
  - blue is pooled average, red is +/- 2 binomial std devs





Pooling Data

- How do we estimate the ability of a player who we observe getting 6 hits in 10 at-bats? Or 0 hits in 5 at-bats? Estimates of 60% or 0% are absurd!
- Same logic applies to players with 40 hits in 131 at bats or 152 hits in 537.
- *No pooling*: estimate each player separately
- *Complete pooling*: estimate all players together (assume no difference in abilities)
- *Partial pooling*: somewhere in the middle
  - use information about other players (i.e., the population) to estimate a player's ability

Complete Pooling Model in Stan

- Assume players all have same ability
- Assume uniform prior on abilities

```
data {
  int<lower=0> N;                  // items
  int<lower=0> K[N];               // trials
  int<lower=0> y[N];               // successes
}
parameters {
  real<lower=0, upper=1> phi;   // chance of success
}
model {
  y ~ binomial(K, phi);          // vectorized likelihood
}
```

No Pooling Model in Stan

- Assume each player has independent ability
- Assume uniform priors on abilities

```
data {
  int<lower=0> N;
  int<lower=0> K[N];
  int<lower=0> y[N];
}
parameters {
  real<lower=0, upper=1> phi[N];
}
model {
  y ~ binomial(K, phi);  // now y[n] matches phi[n]
}
```

Hierarchical Models

- Hierarchical models are principled way of determining how much pooling to apply.
- Pull estimates toward the population mean based on amount of variation in population
  - low variance population: more pooling
  - high variance population: less pooling
- In limit
  - as variance goes to 0, get complete pooling
  - as variance goes to ∞, get no pooling

Hierarchical Batting Ability

- Instead of fixed priors, estimate priors along with other parameters
- Still only uses data once for a single model fit
- Data: y_n, K_n: hits, at-bats for player n
- Parameters: ϕ_n: ability for player n
- Hyperparameters: α, β: population mean and variance
- Hyperpriors: fixed priors on α and β (hardcoded)

Hierarchical Batting Model (cont.)

$$\begin{aligned}
\theta &\sim \text{Uniform}(0, 1) \\
\kappa &\sim \text{Pareto}(1.5) \\
\phi_n &\sim \text{Beta}(\kappa\,\theta,\ \kappa\,(1-\theta)) \\
y_n &\sim \text{Binomial}(K_n, \phi_n)
\end{aligned}$$

- Pareto provides power law distro on prior count:

$$\text{Pareto}(u \mid \alpha) \propto \frac{\alpha}{u^{\alpha+1}}$$

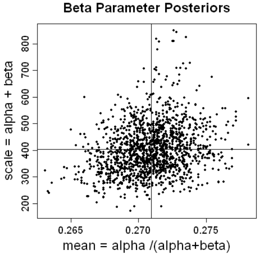
- θ is prior mean; κ is prior count (plus 2).
- Should use more informative prior on θ.

Partial Pooling Model in Stan

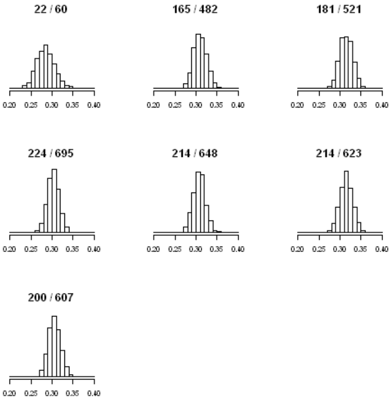
```
data {
  int<lower=0> N;
  int<lower=0> K[N];
  int<lower=0> y[N];
}
parameters {
  real<lower=0, upper=1> theta;
  real<lower=1> kappa;
  vector<lower=0, upper=1>[N] phi;
}
model {
  kappa ~ pareto(1, 1.5);                 // hyperprior
  theta ~ beta(kappa * theta,             // prior
               kappa * (1 - theta));
  y ~ binomial(K, theta);                 // likelihood
}
```

Posterior for Hyperpriors

- Scatterplot of draws
- Crosshairs at mean
- $\kappa = \alpha + \beta$ and $\theta = \frac{\alpha}{\alpha+\beta}$
- Prior mean est: $\hat{\theta} = 0.271$
- Prior count est: $\hat{\kappa} = 400$
- Together yield prior std dev of only 0.022



Posterior Ability (High Avg Players)



Who's the Best?

- Posterior probability that player n has highest ability:

$$\text{Pr}[\phi_n \geq \max(\phi) \mid y]$$

- Code up with indicator variable in Stan

```
generated quantities {
  int<lower=0, upper=1> is_best[N];
  for (n in 1:N)
    is_best[n] = (phi[n] >= max(phi));
}
```

Multiple Comparisons

- Hierarchical model **adjusts for multiple comparisons** by pulling all estimates toward population mean

Results for 2006 AL Season

Player	*Average*	*At-Bats*	Pr[best]
Mauer	.347	521	0.12
Jeter	.343	623	0.11
	.342	482	0.08
	.330	648	0.04
	.330	607	0.04
	.367	60	0.02
	.322	695	0.02

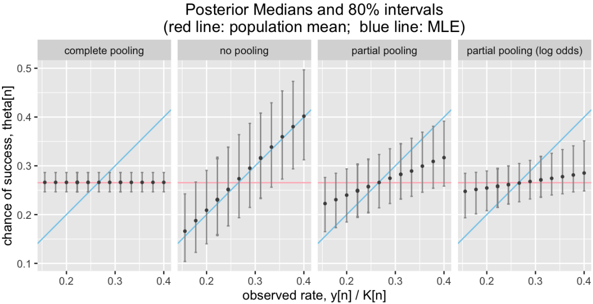
- Posterior probabilities reflect uncertainty in data
- In last game (of 162), Mauer (Minnesota) edged out Jeter (NY)

Efron & Morris (1975) Data

- From their classic analysis for shrinkage/empirical Bayes
- Picked batters with 45 at bats on a given day (artificial!)

	FirstName	LastName	Hits	At.Bats	Rest.At.Bats	Rest.Hits
1	Roberto	Clemente	18	45	367	127
2	Frank	Robinson	17	45	426	127
3	Frank	Howard	16	45	521	144
4	Jay	Johnstone	15	45	275	61
5	Ken	Berry	14	45	418	114
6	Jim	Spencer	14	45	466	126
7	Don	Kessinger	13	45	586	155
8	Luis	Alvarado	12	45	138	29
9	Ron	Santo	11	45	510	137
10	Ron	Swaboda	11	45	200	46
11	Rico	Petrocelli	10	45	538	142

Pooling vs. No-Pooling Estimates

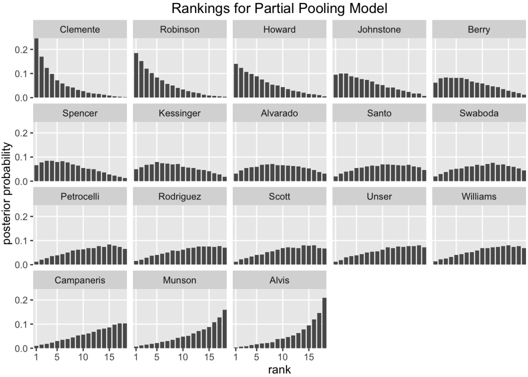


- complete pooling, no pooling, partial pooling, (log odds)

Ranking

```
generated quantities {
  int<lower=1, upper=N> rnk[N];      // rank of player n
  {
    int dsc[N];
    dsc = sort_indices_desc(theta);
    for (n in 1:N)
      rnk[dsc[n]] = n;
  }
```

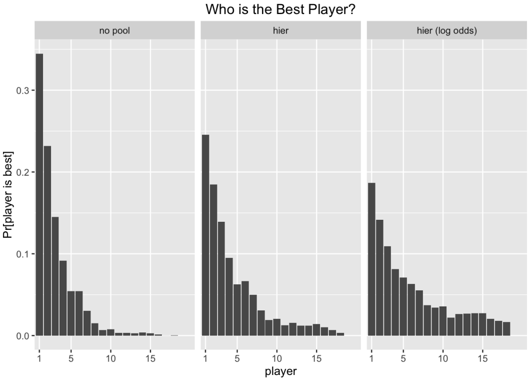
Posterior Ranks



Who is Best? better Stan code

```
generated quantities {
  ...
  int<lower=0, upper=1> is_best[N];
  ...
  for (n in 1:N)
    is_best[n] = (rnk[n] == 1);  // more efficient
  ...
```

Who is Best? Posterior



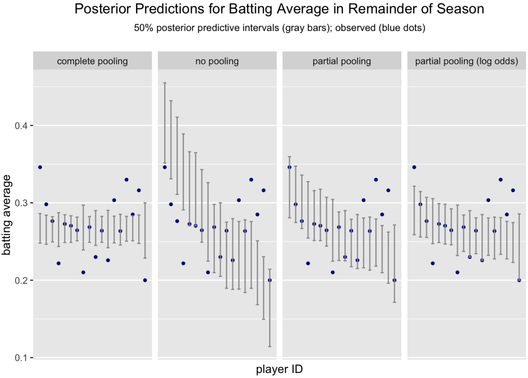
Posterior Predictive Inference

- How do we predict new outcomes (e.g., rest of season)?

```
data {
  int<lower=0> K_new[N];       // new trials
  int<lower=0> y_new[N];       // new outcomes
  ...
generated quantities {
  int<lower=0> z[N];  // posterior prediction
  for (n in 1:N)
    z[n] = binomial_rng(K_new[n], theta[n]);
```

- Full Bayes accounts for two sources of uncertainty
  - estimation uncertainty (built into posterior)
  - sampling uncertainty (explicit RNG function)

Posterior Predictions

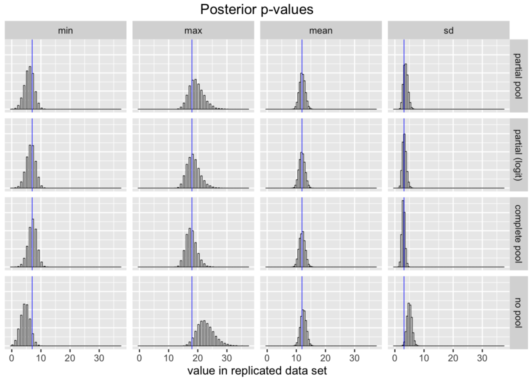


Posterior Predictive Check

- Replicate data from paraemters

```
generated quantities {
  ...
  for (n in 1:N)
    y_rep[n] = binomial_rng(K[n], theta[n]);
  for (n in 1:N)
    y_pop_rep[n] = binomial_rng(K[n],
                                beta_rng(phi * kappa,
                                         (1 - phi) * kappa));
  min_y_rep = min(y_rep);
  sd_y_rep = sd(to_vector(y_rep));
  p_min = (min_y_rep >= min_y);
  p_sd = (sd_y_rep >= sd_y);
}
```

Posterior p-Values



Calibration and Sharpness

- **Calibration**: A model is calibrated if the 50% intervals contain roughly 50% of the true intervals
  - technically, we expect Binomial$(N, 0.5)$ of N parameters to fall in their 50% intervals
  - we can evaluate with held-out data using cross-validation
- **Sharpness**: One posterior is sharper than another if it concentrates more posterior mass around the true value
  - e.g., central posterior intervals of interest are narrower
  - see: Gneiting, Balabdaoui, and Raftery (2007) Probabilistic forecasts, calibration and sharpness. *JRSS B*.

More in the Case Study

- This talk roughly followed my Stan case study:
  - Hierarchical Partial Pooling for Repeated Binary Trials
- Available under case studies at
  - `http://mc-stan.org/documentation`.
- Contribute case studies in knitr or Jupyter
  - Chris Fonnesbeck (of PyMC3 fame) wrote a great PyStan case study on hierarchical modeling for continuous data as a Python Jupyter notebook (follow above link)
  - Many more case studies, including new ones by Michael Betancourt on core Stan computational issues