

Beta Binomial Model

In this series we've been using the empirical Bayes method to estimate batting averages of baseball players. Empirical Bayes is useful here because when we don't have a lot of information about a batter, they're "shrunk" towards the average across all players, as a natural consequence of the beta prior.

But there's a complication with this approach. **When players are better, they are given more chances to bat!** (Hat tip to Hadley Wickham to pointing this complication out to me). That means there's a relationship between the number of at-bats (AB) and the true batting average. For reasons I explain below, this makes our estimates systematically inaccurate.

In this post, we change our model where all batters have the same prior to one where each batter has his own prior, using a method called **beta-binomial regression**. We show how this new model lets us adjust for the confounding factor while still relying on the empirical Bayes philosophy. We also note that this gives us a general framework for allowing a prior to depend on known information, which will become important in future posts.

Setup

As usual, I'll start with some code you can use to catch up if you want to follow along in R. If you want to understand what it does in more depth, check out the previous posts in this series. (As always, all the code in this post can be found [here](#)).

```
library(dplyr)
library(tidyr)
library(Lahman)
library(ggplot2)
theme_set(theme_bw())

# Grab career batting average of non-pitchers
# (allow players that have pitched <= 3 games, like Ty Cobb)
pitchers <- Pitching %>%
  group_by(playerID) %>%
  summarize(gamesPitched = sum(G)) %>%
  filter(gamesPitched > 3)

career <- Batting %>%
  filter(AB > 0) %>%
  anti_join(pitchers, by = "playerID") %>%
  group_by(playerID) %>%
  summarize(H = sum(H), AB = sum(AB)) %>%
  mutate(average = H / AB)

# Add player names
```

```

career <- Master %>%
  tbl_df() %>%
  dplyr::select(playerID, nameFirst, nameLast) %>%
  unite(name, nameFirst, nameLast, sep = " ") %>%
  inner_join(career, by = "playerID")

# Estimate hyperparameters alpha0 and beta0 for empirical Bayes
career_filtered <- career %>% filter(AB >= 500)
m <- MASS::fitdistr(career_filtered$average, dbeta,
  start = list(shape1 = 1, shape2 = 10))

alpha0 <- m$estimate[1]
beta0 <- m$estimate[2]
prior_mu <- alpha0 / (alpha0 + beta0)

# For each player, update the beta prior based on the evidence
# to get posterior parameters alpha1 and beta1
career_eb <- career %>%
  mutate(eb_estimate = (H + alpha0) / (AB + alpha0 + beta0)) %>%
  mutate(alpha1 = H + alpha0,
    beta1 = AB - H + beta0) %>%
  arrange(desc(eb_estimate))

```

Recall that the `eb_estimate` column gives us estimates about each player's batting average, estimated from a combination of each player's record with the beta prior parameters estimated from everyone (α_0, β_0). For example, a player with only a single at-bat and a single hit ($H=1; AB=1; H/AB=1$) will have an empirical Bayes estimate of

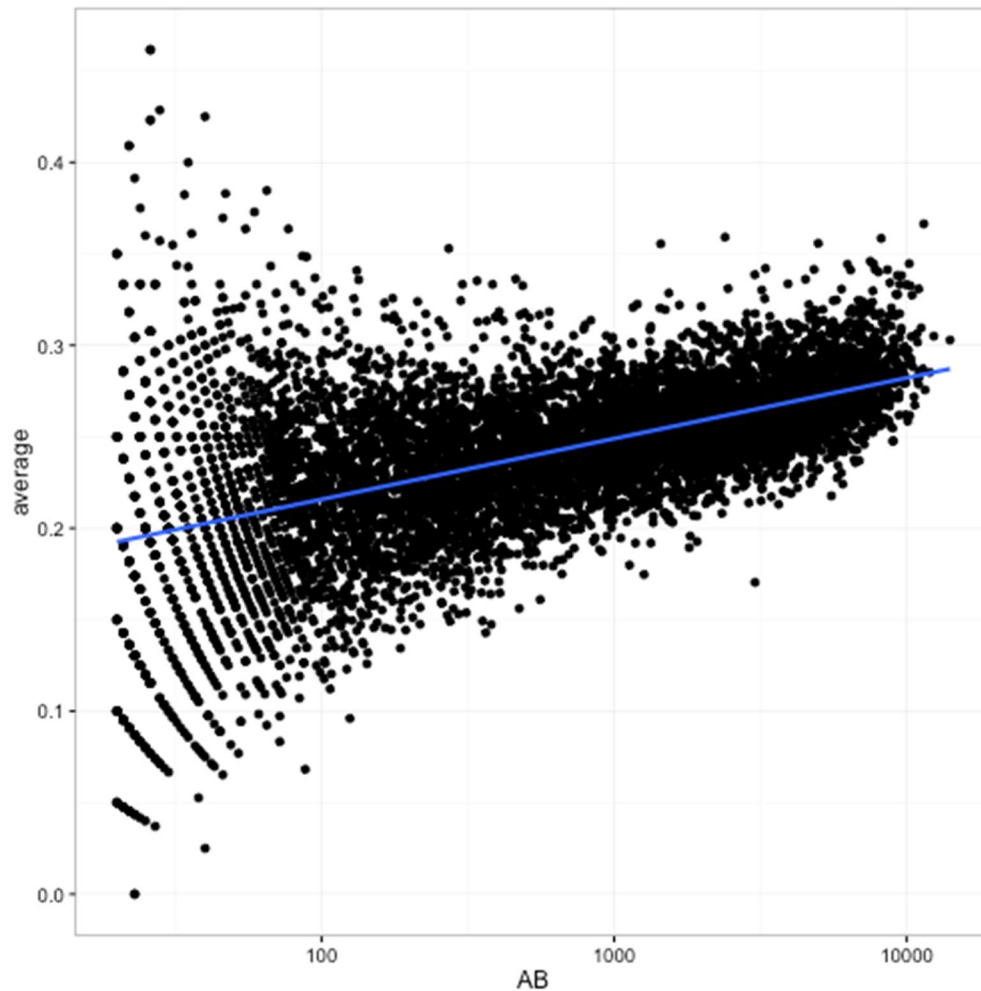
$$(H+\alpha_0)/(AB+\alpha_0+\beta_0)=(1+76.5)/(1+76.5+218.6)=0.262$$

Now, here's the complication. Let's compare at-bats (on a log scale) to the raw batting average:

```

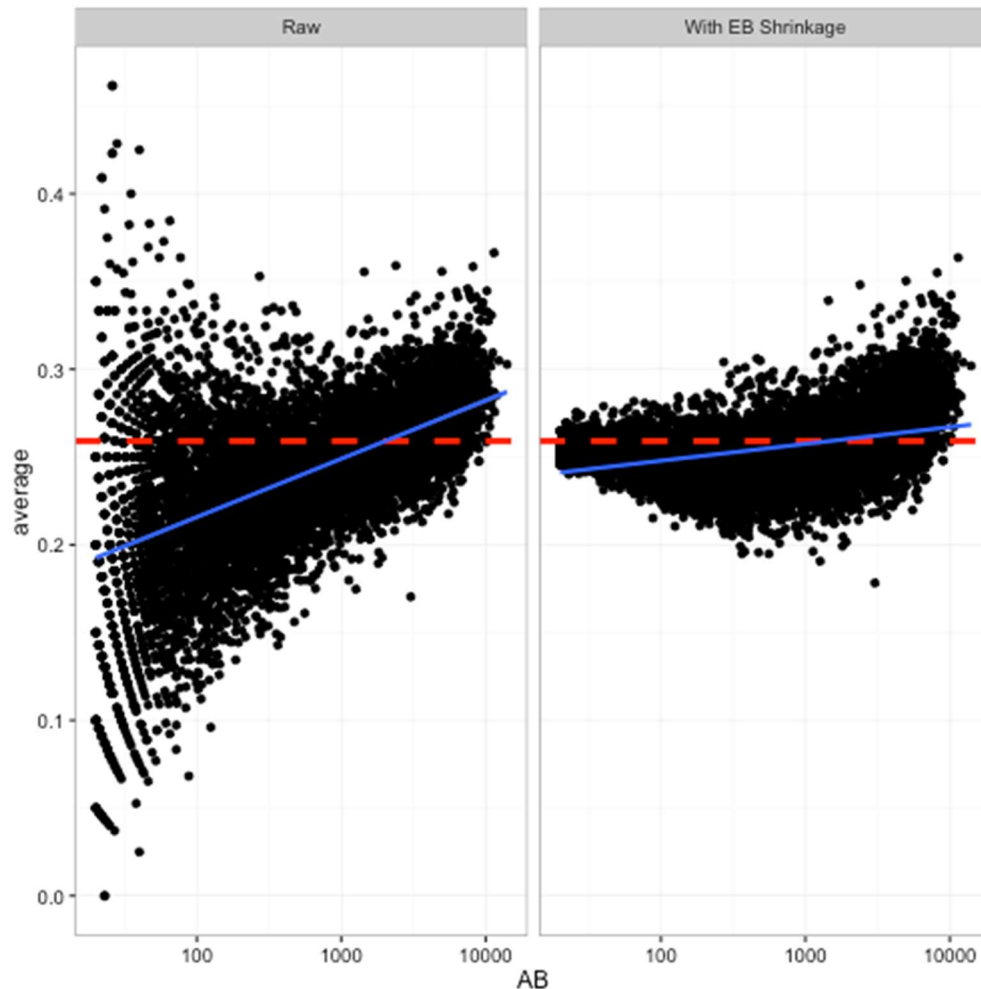
career %>%
  filter(AB >= 20) %>%
  ggplot(aes(AB, average)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  scale_x_log10()

```



We notice that batters with low ABs have more variance in our estimates- that's a familiar pattern because we have less information about them. But notice a second trend: as the number of at-bats increases, the batting average *also* increases. Unlike the variance, this is *not* an artifact of our measurement: it's a result of the choices of baseball managers! Better batters get played more: they're more likely to be in the starting lineup and to spend more years playing professionally.

Now, there are many other factors that are correlated with a player's batting average (year, position, team, etc). But this one is particularly important, because it confounds our ability to perform empirical Bayes estimation:



That horizontal red line shows the prior mean that we're "shrinking" towards ($\alpha_0\alpha_0+\beta_0=0.259$). Notice that it is too high for the low-AB players. For example, the median batting average for players with 5-20 at-bats is 0.167, and they get shrunk *way* towards the overall average! The high-AB crowd basically stays where they are, because each has a lot of evidence.

So since low-AB batters are getting overestimated, and high-AB batters are staying where they are, we're working with a biased estimate that is systematically *overestimating* batter ability. If we were working for a baseball manager (like in Moneyball), that's the kind of mistake we could get fired for!

Accounting for AB in the model

How can we fix our model? We'll need to have AB somehow influence our priors, particularly affecting the mean batting average. In particular, we want the typical batting average to be linearly affected by $\log(\text{AB})$.

First we should write out what our current model is, in the form of a **generative process**, in terms of how each of our variables is generated from particular distributions. Defining p_i to be the true probability of hitting for batter i (that is, the “true average” we’re trying to estimate), we’re assuming

$$p_i \sim \text{Beta}(\alpha_0, \beta_0) \\ H_i \sim \text{Binom}(AB_i, p_i)$$

(We’re letting the totals AB_i be fixed and known per player). We made up this model in one of the first posts in this series and have been using it since.

I’ll point out that there’s another way to write the p_i calculation, by **re-parameterizing** the beta distribution. Instead of parameters α_0 and β_0 , let’s write it in terms of μ_0 and σ_0 :

$$p_i \sim \text{Beta}(\mu_0/\sigma_0, (1-\mu_0)/\sigma_0)$$

Here, μ_0 represents the mean batting average, while σ_0 represents how spread out the distribution is (note that $\sigma_0 = 1/\alpha_0 + 1/\beta_0$). When σ_0 is high, the beta distribution is very wide (a less informative prior), and when σ_0 is low, it’s narrow (a more informative prior). Way back in my first post about the beta distribution, this is basically how I chose parameters: I wanted $\mu = .27$, and then I chose a σ_0 that would give the desired distribution that mostly lay between .210 and .350, our expected range of batting averages.

Now that we’ve written our model in terms of μ and σ , it becomes easier to see how a model could take AB into consideration. We simply define μ_i so that it includes $\log(AB_i)$ as a linear term:

$$\mu_i = \mu_0 + \mu_{AB} \cdot \log(AB_i) \\ \alpha_{0,i} = \mu_i / \sigma_0 \\ \beta_{0,i} = (1 - \mu_i) / \sigma_0$$

Then we define the batting average p_i and the observed H_i just like before:

$$p_i \sim \text{Beta}(\alpha_{0,i}, \beta_{0,i}) \\ H_i \sim \text{Binom}(AB_i, p_i)$$

This particular model is called **beta-binomial regression**. We already had each player represented with a binomial whose parameter was drawn from a beta, but now we're allowing the expected value of the beta to be influenced.

Step 1: Fit the model across all players

Going back to the basics of empirical Bayes, our first step is to fit these prior parameters: μ_0 , μ_{AB} , σ_0 . When doing so, it's ok to momentarily "forget" we're Bayesians- we picked our α_0 and β_0 using maximum likelihood, so it's OK to fit these using a maximum likelihood approach as well. You can use the gamlss package for fitting beta-binomial regression using maximum likelihood.

```
library(gamlss)

fit <- gamlss(cbind(H, AB - H) ~ log(AB),
              data = career_eb,
              family = BB(mu.link = "identity"))
```

We can pull out the coefficients with the broom package (see `?gamlss_tidiers`):

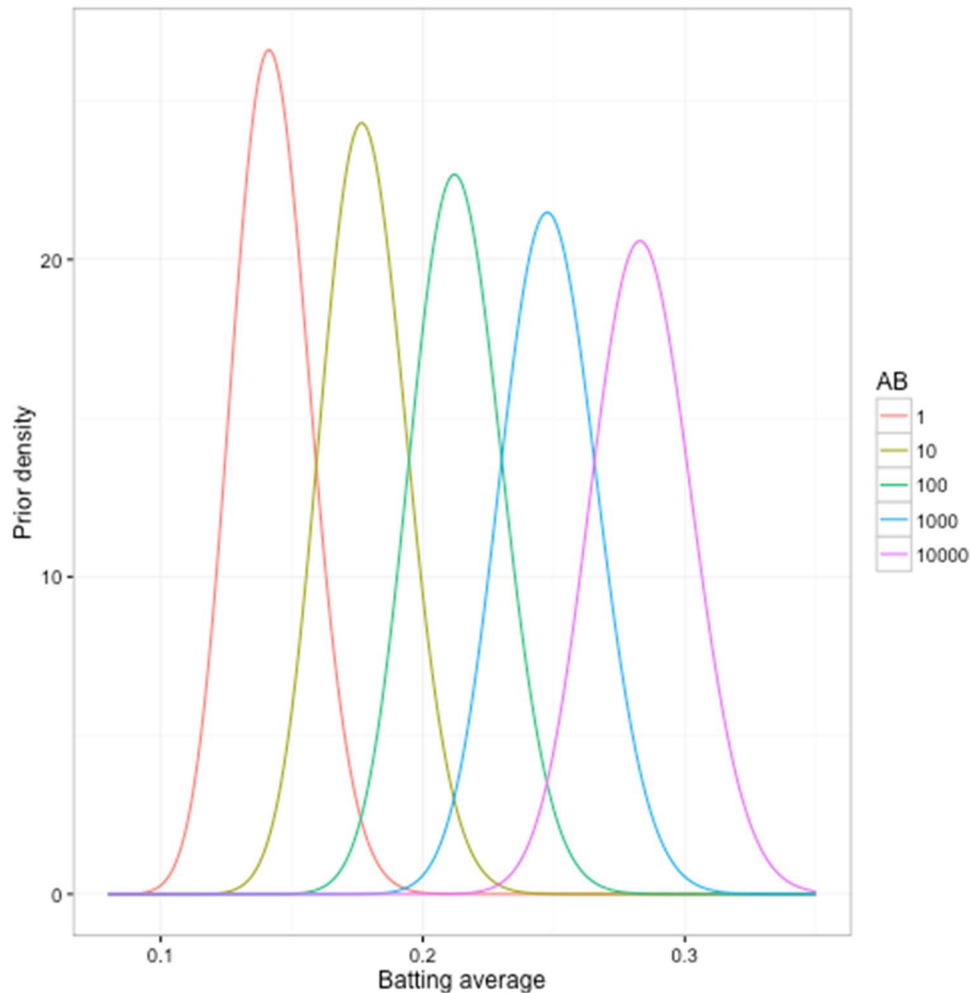
```
library(broom)

td <- tidy(fit)
td
```

| | parameter | term | estimate | std.error | statistic | p.value |
|------|-----------|-------------|----------|-----------|-----------|---------|
| ## 1 | mu | (Intercept) | 0.1426 | 0.001577 | 90.4 | 0 |
| ## 2 | mu | log(AB) | 0.0153 | 0.000215 | 71.2 | 0 |
| ## 3 | sigma | (Intercept) | -6.2935 | 0.023052 | -273.0 | 0 |

This gives us our three parameters: $\mu_0=0.143$, $\mu_{AB}=0.015$, and (since `sigma` has a log-link) $\sigma_0=\exp(-6.294)=0.002$.

This means that our new prior beta distribution for a player *depends on* the value of AB. For example, here are our prior distributions for several values:



Notice that there is still uncertainty in our prior- a player with 10,000 at-bats could have a batting average ranging from about .22 to .35. But the range of that uncertainty changes greatly depending on the number of at-bats- any player with $AB = 10,000$ is almost certainly better than one with $AB = 10$.

Step 2: Estimate each player's average using this prior

Now that we've fit our overall model, we repeat our second step of the empirical Bayes method. Instead of using a single α_0 and β_0 values as the prior, we choose the prior for each player based on their AB. We then update using their HH and AB just like before.

Here, all we need to calculate are the μ (that is, $\mu = \mu_0 + \mu_0 \log(AB)$) and σ (σ) parameters for each person. (Here, σ will be the same for everyone, but that may not be true in more complex models). This can be done using the `fitted` method on the `gamlss` object (see [here](#)):

```

mu <- fitted(fit, parameter = "mu")
sigma <- fitted(fit, parameter = "sigma")

head(mu)
##      1      2      3      4      5      6
## 0.286 0.281 0.273 0.262 0.279 0.284
head(sigma)
##      1      2      3      4      5      6
## 0.00185 0.00185 0.00185 0.00185 0.00185 0.00185

```

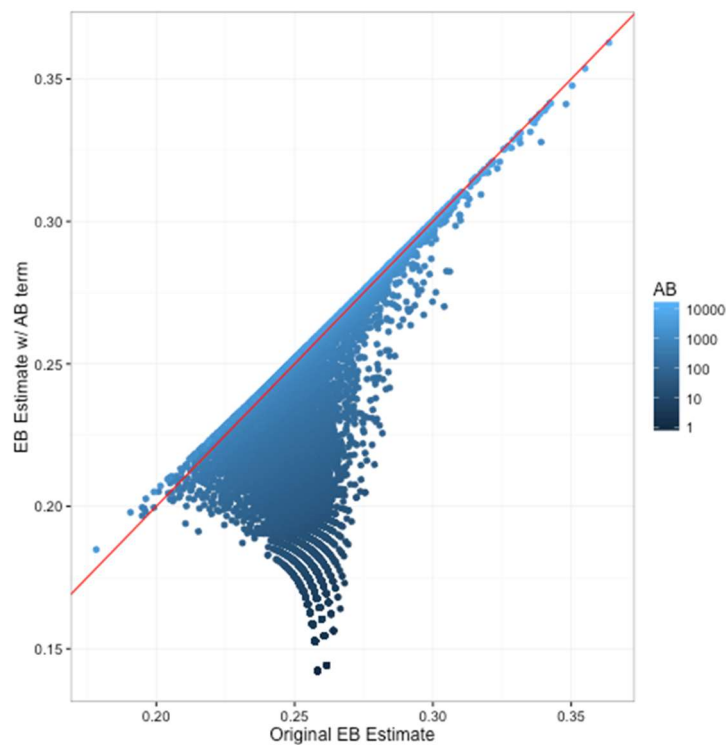
Now we can calculate α_0 and β_0 parameters for each player, according to $\alpha_{0,i} = \mu_i / \sigma_0$ and $\beta_{0,i} = (1 - \mu_i) / \sigma_0$. From that, we can update based on HH and ABAB to calculate new $\alpha_{1,i}$ and $\beta_{1,i}$ for each player.

```

career_eb_wAB <- career_eb %>%
  dplyr::select(name, H, AB, original_eb = eb_estimate) %>%
  mutate(mu = mu,
         alpha0 = mu / sigma,
         beta0 = (1 - mu) / sigma,
         alpha1 = alpha0 + H,
         beta1 = beta0 + AB - H,
         new_eb = alpha1 / (alpha1 + beta1))

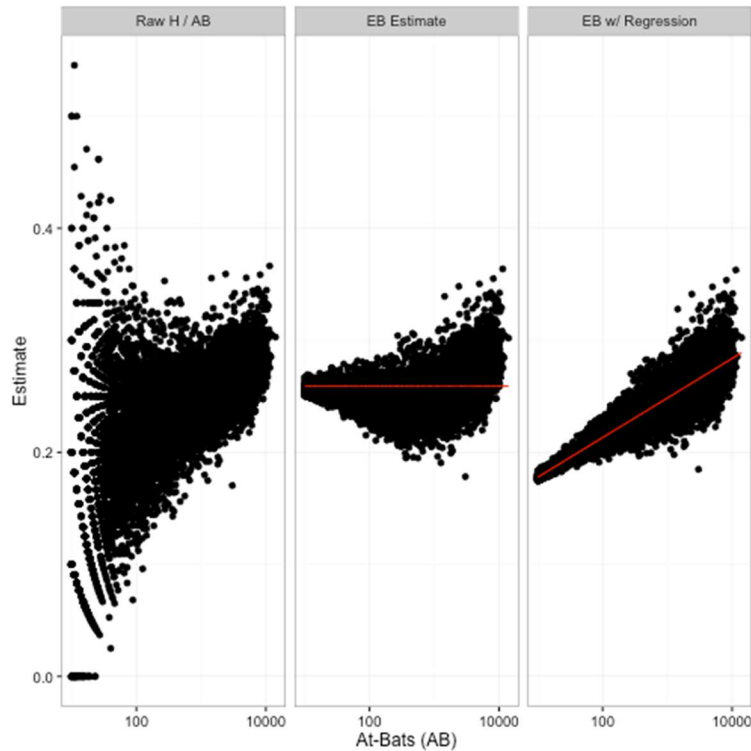
```

How does this change our estimates?



Notice that relative to the previous empirical Bayes estimate, this one is lower for batters with low AB and about the same for high-AB batters. This fits with our earlier description- we've been systematically over-estimating batting averages.

Here's another way of comparing the estimation methods:



Notice that we used to shrink batters towards the overall average (red line), but now we are shrinking them towards the overall *trend*- that red slope.²


Don't forget that this change in the posteriors won't just affect shrunken estimates. It will affect all the ways we've used posterior distributions in this series: credible intervals, posterior error probabilities, and A/B comparisons. Improving the model by taking AB into account will help all these results more accurately reflect reality.

What's Next: Bayesian hierarchical modeling

This problem is in fact a simple and specific form of a Bayesian hierarchical model, where the parameters of one distribution (like α_0 and β_0) are generated based on other distributions and parameters. While these models are often approached using more precise Bayesian methods (such as Markov chain Monte Carlo), we've seen that empirical Bayes can be a powerful and practical approach that helped us deal with our confounding factor.

Beta-binomial regression, and the `gamlss` package in particular, offers a way to fit parameters to predict “success / total” data. In this post, we’ve used a very simple model- μ linearly predicted by AB. But there’s no reason we can’t include other information that we expect to influence batting average. For example, left-handed batters tend to have a slight advantage over right-handed batters- can we include that information in our model? In the next post, we’ll bring in additional information to build a more sophisticated hierarchical model. We’ll also consider some of the limitations of empirical Bayes for these situations.

Footnotes

1. If you have some experience with regressions, you might notice a problem with this model: μ can theoretically go below 0 or above 1, which is impossible for a β distribution (and will lead to illegal α and β parameters). Thus in a real model we would use a “link function”, such as the logistic function, to keep μ between 0 and 1. I used a linear model (and `mu.link = "identity"` in the `gamlss` call) to make the math in this introduction simpler, and because for this particular data it leads to almost exactly the same answer (try it). In our next post we’ll include the logistic link. 
2. If you work in in my old field of gene expression, you may be interested to know that empirical Bayes shrinkage towards a trend is exactly what some differential expression packages such as edgeR do with per-gene dispersion estimates. It’s a powerful concept that allows a balance between individual observations and overall expectations. 