Stats Bootcamp Day 7 - Empirical Bayes

Your Name

9/14/2021

Recap + Today's Material

Empirical Bayes - Baseball example

Example from the following blog post by David Robinson:
http://varianceexplained.org/r/empirical_bayes_baseball/
(http://varianceexplained.org/r/empirical_bayes_baseball/)

You may see a long red message when you run the code below, do not worry, if you are able to run the codes after this chunk properly you do not need to fix anything

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.0.2
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 4.0.2
```

```
library(Lahman)
```

```
## Warning: package 'Lahman' was built under R version 4.0.2
```

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

# use names along with the player IDs
career <- Master %>%
  tbl_df() %>%
  select(playerID, nameFirst, nameLast) %>%
  unite(name, nameFirst, nameLast, sep = " ") %>%
  inner_join(career, by = "playerID") %>%
  select(-playerID)
```

```
## Warning: `tbl_df()` was deprecated in dplyr 1.0.0.
## Please use `tibble::as_tibble()` instead.
```

```
career
```

name <chr>	H <int>	AB <int>	average <dbl>
Hank Aaron	3771	12364	0.30499838
Tommie Aaron	216	944	0.22881356
Andy Abad	2	21	0.09523810
John Abadie	11	49	0.22448980
Ed Abbaticchio	772	3044	0.25361367
Fred Abbott	107	513	0.20857700
Jeff Abbott	157	596	0.26342282
Kurt Abbott	523	2044	0.25587084
Ody Abbott	13	70	0.18571429
Frank Abercrombie	0	4	0.00000000

1-10 of 9,802 rows Previous **1** 2 3 4 5 6 … 981 Next

Above we can get a look at the entire data set, displaying batting statistics for various baseball players. Now, we are going to try sorting by batting average and take a look at who's at the top

```
head(career[order(career$average, decreasing = TRUE),])
```

name <chr>	H <int>	AB <int>	average <dbl>

name <chr>	H <int>	AB <int>	average <dbl>
Jeff Banister	1	1	1
Doc Bass	1	1	1
Steve Biras	2	2	1
C. B. Burns	1	1	1
Jackie Gallagher	1	1	1
Roy Gleason	1	1	1

6 rows

So clearly just sorting by “average” is not meaningful. People with small sample sizes are dominating the extremes.

NB: I am using the terms “average”, “batting average”, and “batting percentage” interchangeably.

An unprincipled approach could be to just exclude people with less than, say, 500 at bats from this.

```
filtered_career = career[career$AB>=500,]
head(filtered_career[order(filtered_career$average, decreasing = TRUE),])
```

name <chr>	H <int>	AB <int>	average <dbl>
Rogers Hornsby	2930	8173	0.3584975
Shoeless Joe Jackson	1772	4981	0.3557519
Ed Delahanty	2597	7510	0.3458056
Billy Hamilton	2164	6283	0.3444215
Harry Heilmann	2660	7787	0.3415950
Willie Keeler	2932	8591	0.3412874

6 rows

There is a more principled way to go about this though: Empirical Bayes.

Are some batting averages more likely than others? What is the probability that we will ever see a player who hits 0.800, given that no one with more than 500 at bats has batted above 0.350? I’d say very small! What is the probability that a batter’s true average is 0.100? Very small, given history!

What I did in the above paragraph is informing our “prior” belief – in a data-dependent way. In reality, our priors can come from many different avenues, including previous experiments. While it might be somewhat concerning to have our prior be solely informed by just the data we are trying to analyze, there are some mitigating circumstances here (e.g. using a parametric approximation of the prior). If we want to be extra rigorous, we could only fit a prior based on a hold-out set.

So, what do we make of Jeff Banister, who is batting 100% (1 hit on 1 try)? Is his "real" batting percentage 100%? "Real" here means "as the sample size goes to infinity what does his percentage go to." I'm sure his "real" batting percentage is smaller. But how much smaller?

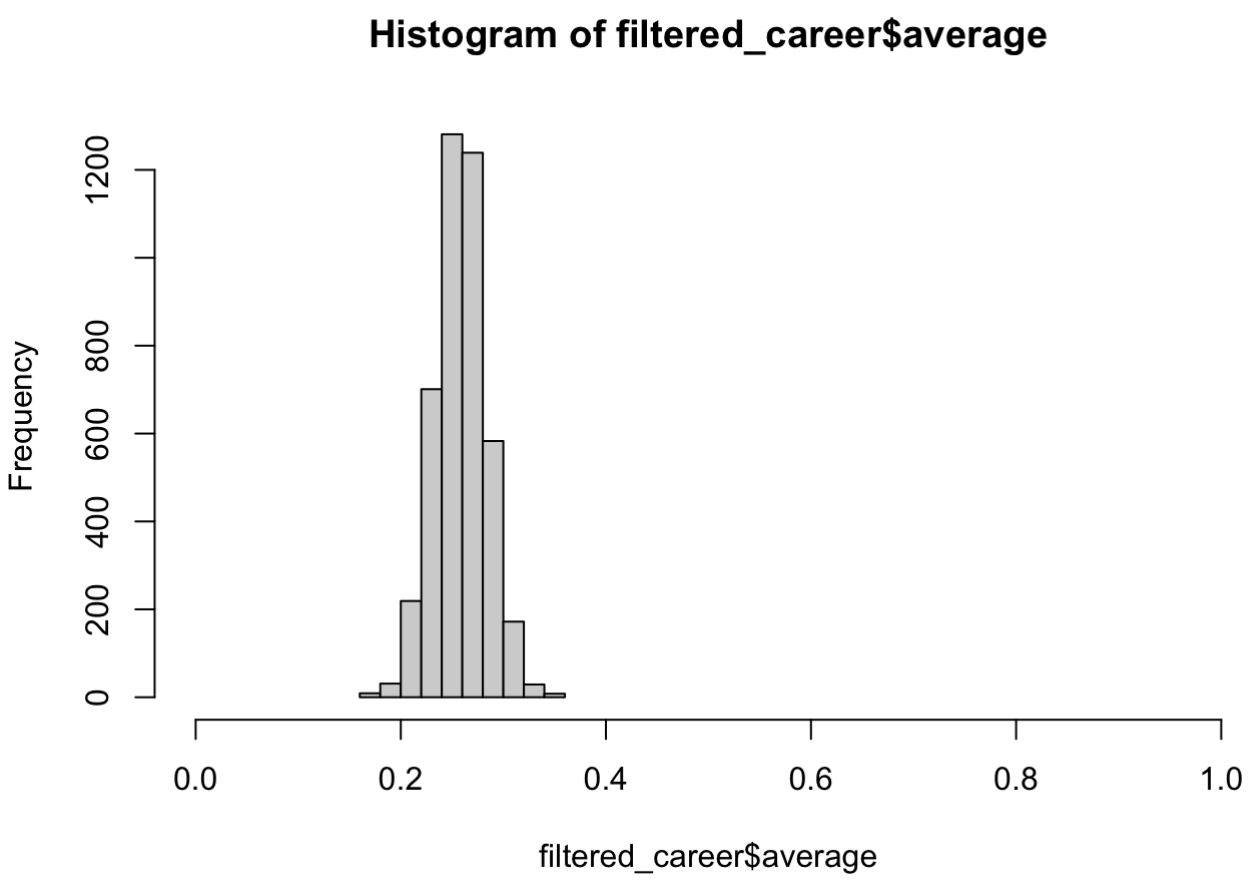
Our goal here is to "shrink" Jeff Banister's 100% to our best guess of his "real" batting average. We want to do this in a way that incorporates our prior beliefs of what the distribution of "real" batting averages looks like.

Bayesian framework is perfect for this!

$$P(real\_batting = 80\% \mid going\,4\,for\,5) = \frac{P(going\,4\,for\,5 \mid real\_batting = 80\%)P(real\_batting = 80\%)}{P(going\,4\,for\,5)}$$

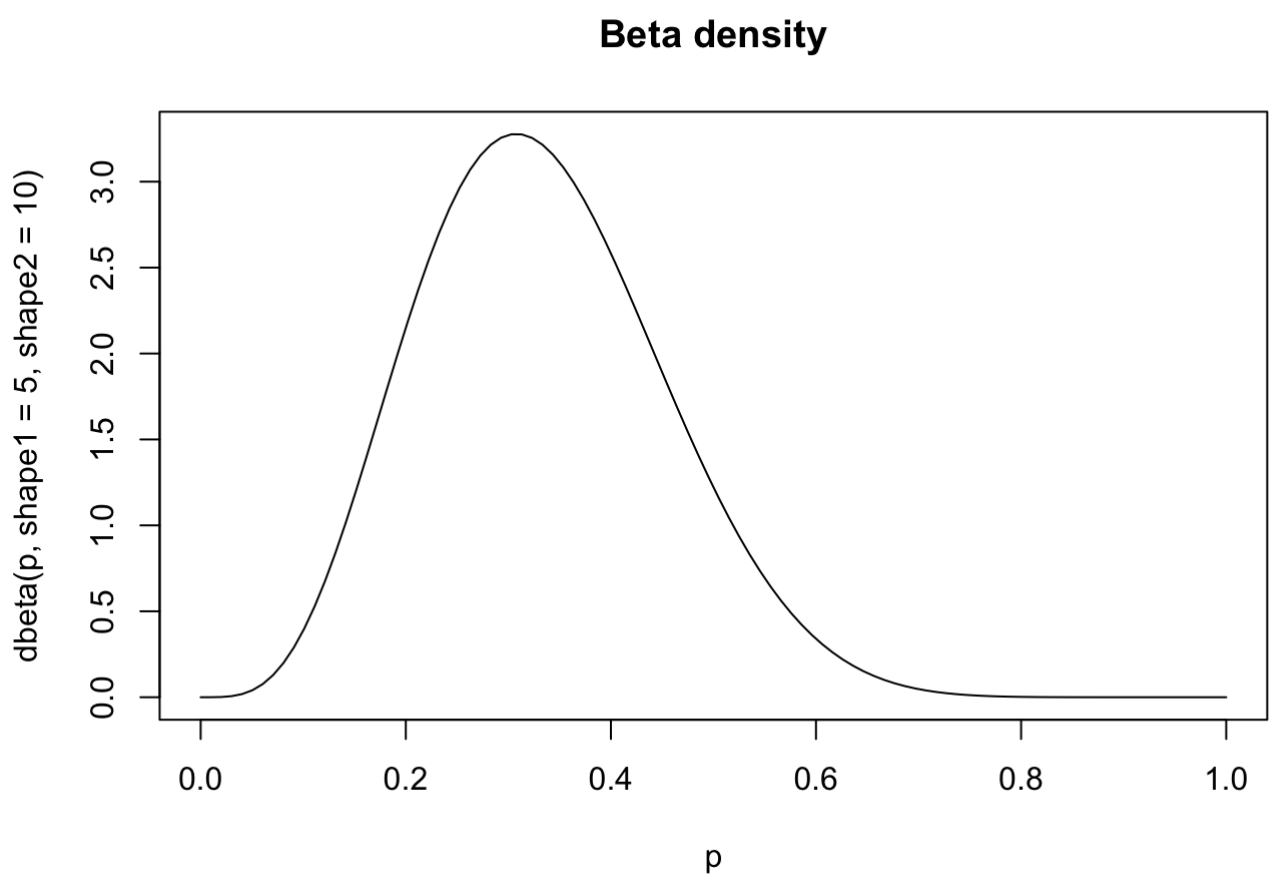
Let's see what the distribution of "real" batting averages looks like:

```
hist(filtered_career$average, xlim=c(0,1))
```



Since this is between 0 and 1, there likely is a "beta" distribution here. You can see here what the beta distribution looks like below. The only constraint on shape1 and shape2 is that they be >0.

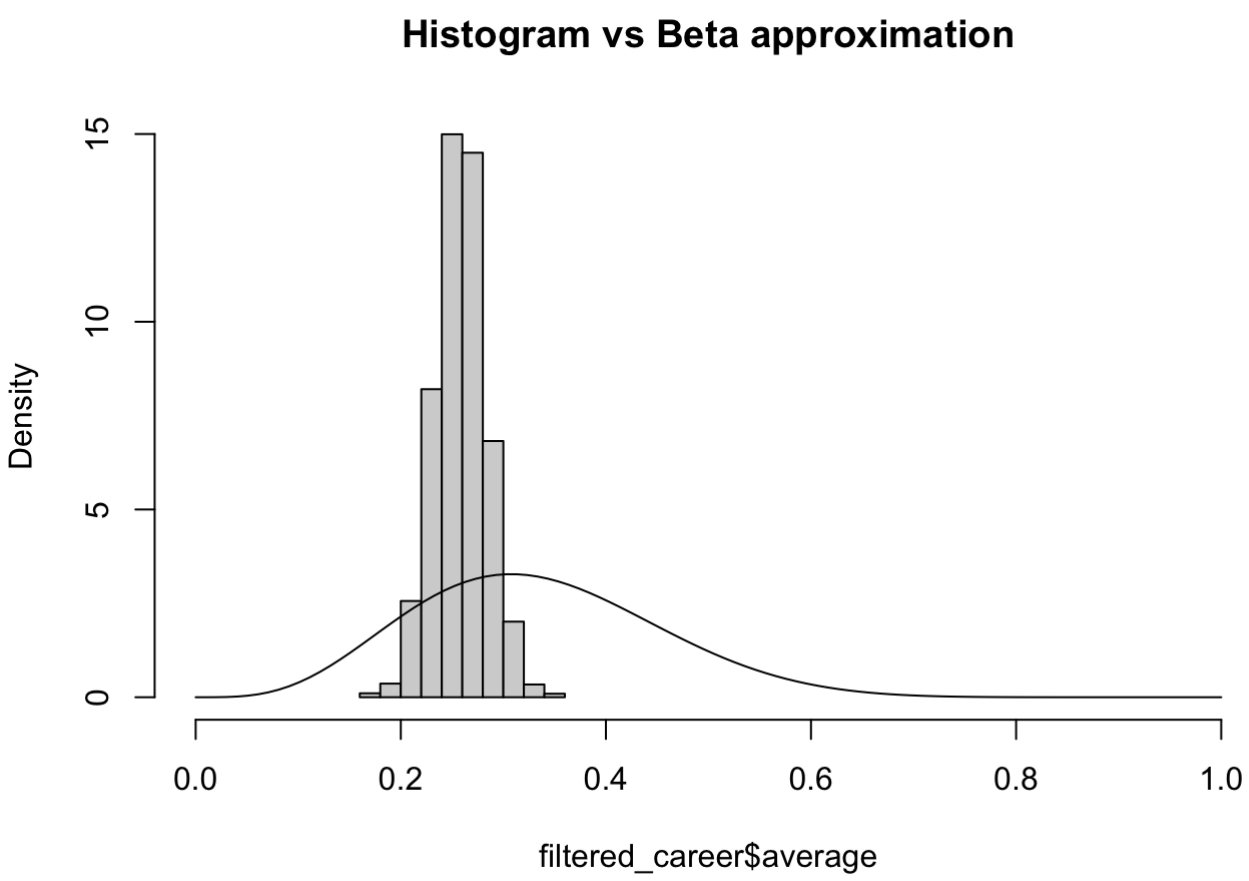
```
p = seq(0,1, length=100)
plot(p, dbeta(p, shape1=5, shape2=10), type="l",
     main="Beta density")
```



There are a couple key facts about the beta distribution (shape1 = α and shape2 = β): - the mean is shape1/(shape1+shape2) - the variance is shape1 * shape2 / [(shape1+shape2)^2 * (shape1 + shape2 + 1)]

Let's overlay this density on our observed data of "real" batting averages:

```
hist(filtered_career$average, xlim=c(0,1), probability = TRUE,
     main="Histogram vs Beta approximation")
lines(p, dbeta(p, shape1=5, shape2=10), type="l")
```



Check-in 1:

Play around with the beta distribution parameters until you think you got a good approximation for the distribution of “real” batting averages"

```
#Add in your answer here!
```

Check-in 1 Solution

There are two main ways to go about this: - Method of Moments (MoM) - Maximim Likelihood Estimator (MLE)

Method of moments:

Let's choose shape1,shape2 such that the mean and variance of our beta distribution will equal our empirical mean and variance.

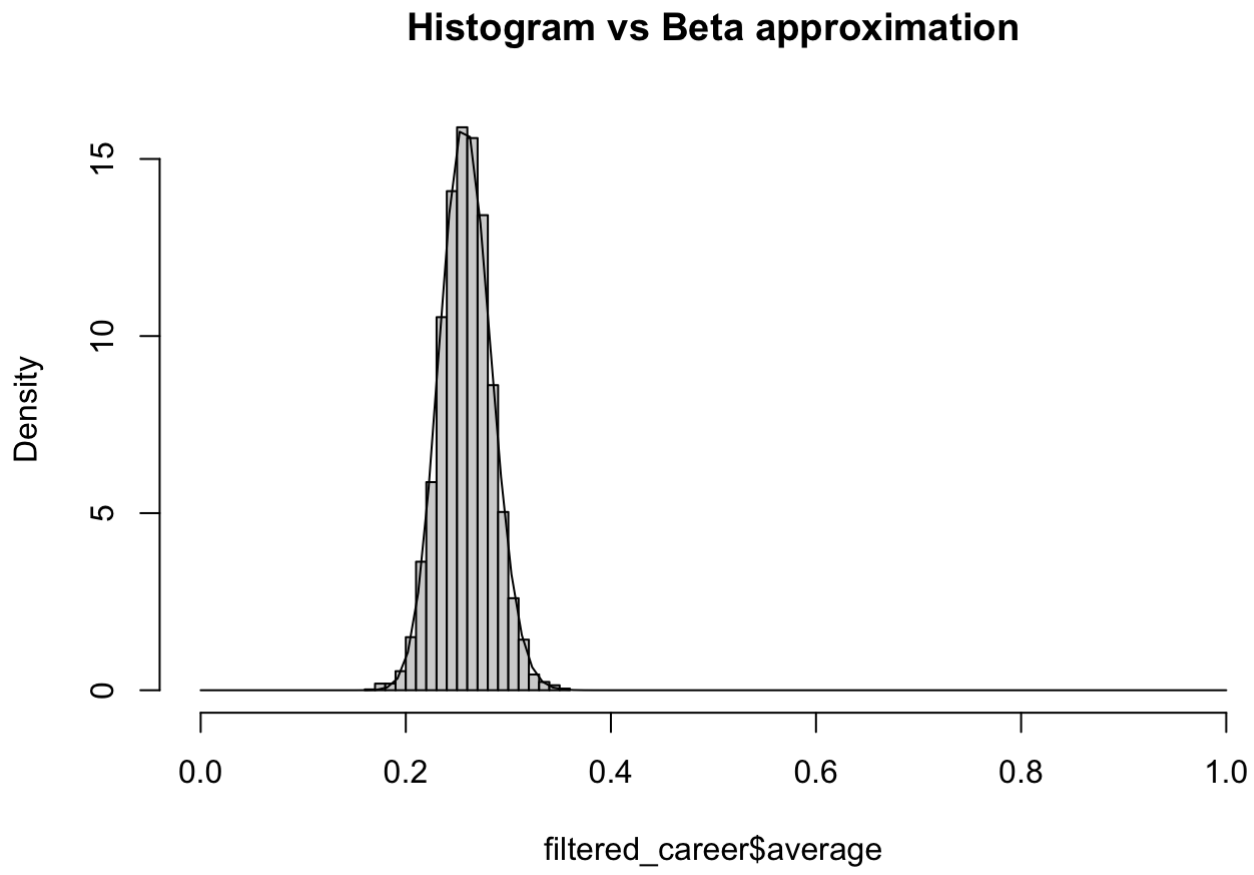
```
emp_mean = mean(filtered_career$average)
emp_var = var(filtered_career$average)
c(emp_mean, emp_var)
```

```
## [1] 0.2585556196 0.0006192446
```

Plugging into Wolfram-Alpha, I get shape1 = 79.8159 and shape2 = 228.884.

Try this out:

```
hist(filtered_career$average, xlim=c(0,1), breaks=20, probability = TRUE,
     main="Histogram vs Beta approximation")
lines(p, dbeta(p, shape1=79.8159, shape2=228.884), type="l")
```



So our beta approximation looks very good here!

Maximum Likelihood Estimation (MLE)

The MLE finds shape1, shape2 that maximizes the probability of observing the data, assuming the data is distributed that way.

I.e.

$$max_{\alpha,\beta} \prod_{i=1}^{n} P(batting\_average_i|\alpha,\beta)$$

$$= max_{\alpha,\beta} \sum_{i=1}^{n} log(P(batting\_average_i|\alpha,\beta))$$

$$= min_{\alpha,\beta} - \sum_{i=1}^{n} log(P(batting\_average_i|\alpha,\beta))$$

We minimize negative log likelihood (instead of maximizing log likelihood) since a lot of optimization literature usually minimizes quantities (even though maximization is equivalent)

For starters, let's plug in random values for shape1, shape2 and see the NLL.

```
sum(-dbeta(filtered_career$average, shape1 = 500, shape2 = 500, log=TRUE))
```

```
## [1] 563390.9
```

Using the method of moments parameters:

```
sum(-dbeta(filtered_career$average, shape1=79.8159, shape2=228.884, log=TRUE))
```

```
## [1] -9714.071
```

These are much lower, and hence much better!

We can find the parameters than minimize NLL via R's "optim" function.

First, we have to make a negative log likelihood function that we can pass to the "optim" function, where the only input is a vector of parameters

```
nll = function(params){sum(-dbeta(filtered_career$average, shape1=params[1], shape2=para
ms[2], log=TRUE))}

nll(c(79.8159, 228.884))
```

```
## [1] -9714.071
```

We can now run the optim function. We only have to provide it some initial values to search from (this shouldn't matter much here). We also have to specify that the parameters have to be strictly positive (we have to use the L-BFGS-B method to put bounds).

```
optim(c(1,1), nll, method="L-BFGS-B", lower=0)
```

```
## $par
## [1]  79.4633 227.8750
##
## $value
## [1] -9714.092
##
## $counts
## function gradient
##       20       20
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

Notice that our NLL with these new parameters is *slightly* smaller and therefore slightly better than the MoM ones. This goes to show that MoM and MLE can sometimes return similar results.

Let's store our MLE parameters:

```
mle_params = optim(c(1,1), nll, method="L-BFGS-B", lower=0)$par
mle_params
```

```
## [1]  79.4633 227.8750
```

Using a built-in function for MLE

You may see a red message saying "NaNs produced", do not worry it will not affect your code

```
MASS::fitdistr(filtered_career$average, "beta",
                    start = list(shape1 = 1, shape2 = 10))
```

```
## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced

## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced

## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced
```

```
##      shape1       shape2
##    79.462866   227.874304
##  (  1.715888) (  4.930722)
```

Why MLE versus MoM?

MLE has some nice statistical properties:

- Invariance under MLE. if $\hat{\theta}$ is the MLE for θ, then $f(\hat{\theta})$ is *also* the MLE for $f(\theta)$ (assuming f is a one-to-one function)
- Asymptotic normality of MLE. for many distributions the MLE will follow a normal distribution asymptotically (similar to the CLT). So, we can do confidence intervals on the MLE.
- MLE is consistent. As sample size goes to infinity, MLE will get arbitrarily close to the true parameter value.

Using our new prior

Bayesian review (shape1 = α and shape2 = β):

$$\begin{aligned} P(real\_batting = 80\% \mid going\,4\,for\,5) &= \frac{P(going\,4\,for\,5 \mid real\_batting = 80\%)P(real\_batting = 80\%)}{P(going\,4\,for\,5)} \\ &\propto P(going\,4\,for\,5 \mid real\_batting = 80\%)P(real\_batting = 80\%) \\ &= \binom{5}{4} 0.8^4(1-0.8)^1 \cdot \frac{0.8^{\alpha-1}(1-0.8)^{\beta-1}}{B(\alpha,\beta)} \\ &\propto 0.8^{4+\alpha-1}(1-0.8)^{1+\beta-1} \\ &\sim Beta(4+\alpha, 1+\beta) \end{aligned}$$

Generalizing:

$$P(real\_batting = p\%|H, AB) \sim Beta(H+\alpha, AB-H+\beta)$$

Recall that the mean of the beta distribution is shape1/(shape1+shape2) So:

$$\begin{aligned} E[real\_batting|H, AB] &= \frac{H+\alpha}{AB+\alpha+\beta} \\ &= \frac{H}{AB+\alpha+\beta} + \frac{\alpha}{AB+\alpha+\beta} \\ &= \frac{AB}{AB+\alpha+\beta} * \frac{H}{AB} + \frac{\alpha+\beta}{AB+\alpha+\beta} * \frac{\alpha}{\alpha+\beta} \\ &= \frac{AB}{AB+\alpha+\beta} * \frac{H}{AB} + (1 - \frac{AB}{AB+\alpha+\beta}) * \frac{\alpha}{\alpha+\beta} \end{aligned}$$

So! The posterior mean is a weighted average between the prior mean and our naive batting average.

We can adjust the weights of this average by adjusting α, β while maintaining $\alpha/(\alpha+\beta) = 0.259$. Making $\alpha+\beta$ big will put more weight on our prior.

```
career
```

name <chr>	H <int>	AB <int>	average <dbl>
Hank Aaron	3771	12364	0.30499838
Tommie Aaron	216	944	0.22881356
Andy Abad	2	21	0.09523810

name <chr>	H <int>	AB <int>	average <dbl>
John Abadie	11	49	0.22448980
Ed Abbaticchio	772	3044	0.25361367
Fred Abbott	107	513	0.20857700
Jeff Abbott	157	596	0.26342282
Kurt Abbott	523	2044	0.25587084
Ody Abbott	13	70	0.18571429
Frank Abercrombie	0	4	0.00000000

1-10 of 9,802 rows Previous **1** 2 3 4 5 6 … 981 Next

```
posterior_mean = function(x){(x[1]+mle_params[1])/(x[2]+mle_params[1]+mle_params[2])}


career$posterior_mean = apply(career[,c(2,3)], 1, posterior_mean)
career
```

name <chr>	H <int>	AB <int>	average <dbl>	posterior_mean <dbl>
Hank Aaron	3771	12364	0.30499838	0.3038719
Tommie Aaron	216	944	0.22881356	0.2361178
Andy Abad	2	21	0.09523810	0.2481078
John Abadie	11	49	0.22448980	0.2538692
Ed Abbaticchio	772	3044	0.25361367	0.2540667
Fred Abbott	107	513	0.20857700	0.2273005
Jeff Abbott	157	596	0.26342282	0.2617661
Kurt Abbott	523	2044	0.25587084	0.2562214
Ody Abbott	13	70	0.18571429	0.2450409
Frank Abercrombie	0	4	0.00000000	0.2552314

1-10 of 9,802 rows Previous **1** 2 3 4 5 6 … 981 Next

```
career[order(career$posterior_mean, decreasing = FALSE),]
```

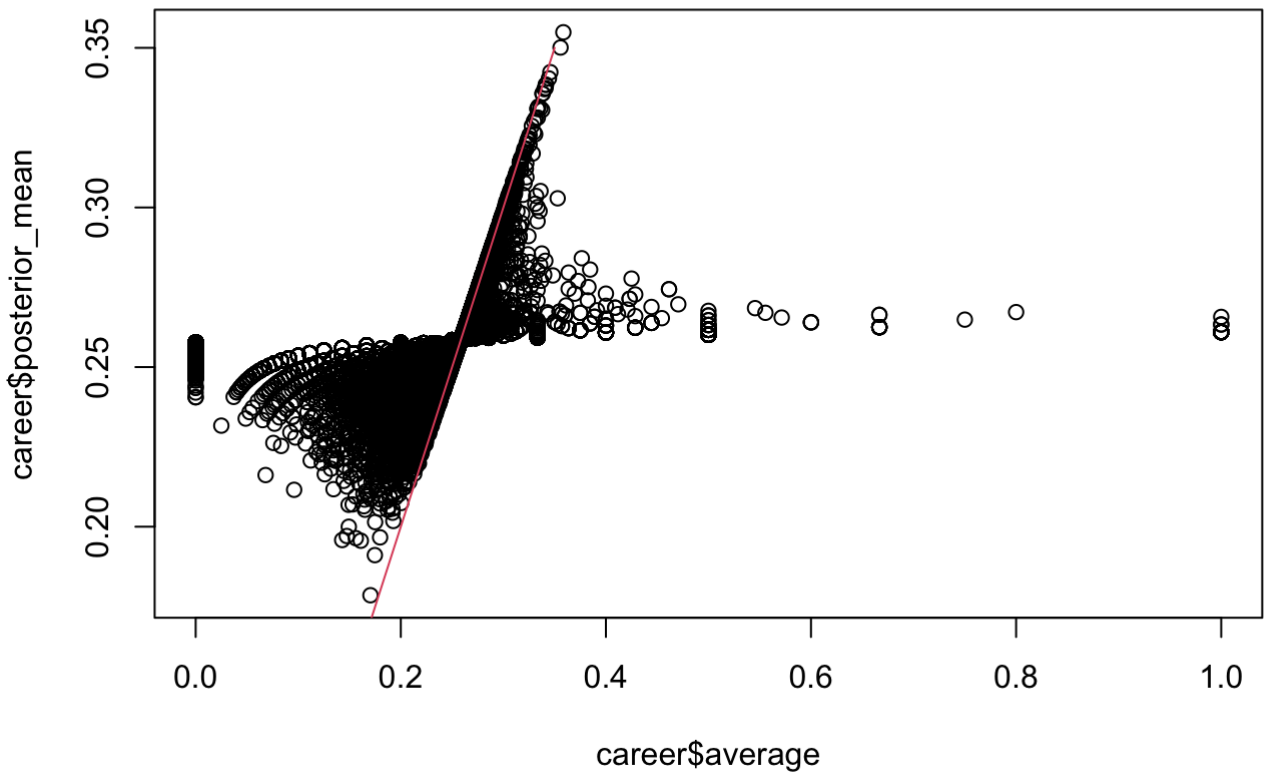
name <chr>	H <int>	AB <int>	average <dbl>	posterior_mean <dbl>
Bill Bergen	516	3028	0.17040951	0.1785316

name <chr>	H <int>	AB <int>	average <dbl>	posterior_mean <dbl>
Ray Oyler	221	1265	0.17470356	0.1910933
John Vukovich	90	559	0.16100179	0.1956087
John Humphries	52	364	0.14285714	0.1958227
George Baker	74	474	0.15611814	0.1964108
Henry Easterday	203	1129	0.17980514	0.1966551
Buck Gladmon	56	380	0.14736842	0.1970839
Charlie Armbruster	53	355	0.14929577	0.1999934
Bill Traffley	116	663	0.17496229	0.2014383
Mike Ryan	370	1920	0.19270833	0.2017939

1-10 of 9,802 rows Previous **1** 2 3 4 5 6 … 981 Next

Let's visualize how this operates as a shrinkage estimator:

```
plot(career$average, career$posterior_mean)
lines(seq(0,0.35, length.out=100),seq(0,0.35, length.out=100), col=2)
```



```
# plot(seq(0,0.35, length.out=100),seq(0,0.35, length.out=100), type="l", col="red")
```

"Bayesian" Confidence intervals / Posterior Credible Intervals

Let's say we want to get a 95% "confidence interval" of what Hank Aaron's true batting average is.

```
career[1,]
```

name <chr>	H <int>	AB <int>	average <dbl>	posterior_mean <dbl>
Hank Aaron	3771	12364	0.3049984	0.3038719

1 row

He has a lot of at bats, so we'd expect the interval to be tight.

We saw earlier that the posterior distribution of any player's batting average followed a beta distribution: $Beta(H+\alpha, AB-H+\beta)$.

We can figure out what the 2.5% quantile (or "percentile") is of this distribution and similarly the 97.5% of this distribution.

Let's do this for Hank Aaron:

```
qbeta(c(0.025, 0.975), shape1=3771+mle_params[1], shape2=12364-3771+mle_params[2])
```

```
## [1] 0.2958936 0.3119088
```

So, there is a less than 2.5% chance that Hank Aaron's true batting average is less than 29.6%, given that he batted 30.5% in 12,364 at bats.

Similarly, there is a less than 2.5% chance that his true batting average is above 31.2%, given his history.

Check-in 2

What is the bayesian confidence interval for the real batting average of a player who had 0 hits in 1 at bat?

```
#Add in your answer here!
```

Check-in 2 solution

A player who had just one at bat would have a much wider bayesian confidence interval:

```
qbeta(c(0.025, 0.975), shape1=0+mle_params[1], shape2=1-0+mle_params[2])
```

```
## [1] 0.2105133 0.3078927
```

Shortcomings of this analysis

We approximated the prior only with players who have more than 500 at bats. Since the players with less than 500 at bats might be worse than the players with 500 at bats (e.g. they didn't have a longer career because they weren't good enough to stay in professional baseball), our prior is likely "biased" for players with <500 at bats.

We can't simply include the players with fewer than 500 at bats in our beta fitting procedure, since it will treat the players that bat the 40% in 10 at bats the same as a player batter 40% in 1,000 at bats.

There is a principled approach to solving this, and that is the beta-binomial distribution. The blog post from the beginning includes a section on it in its appendix.

Different eras may have different prior distributions. Perhaps before 1920 the distribution looks different than the post-2000 distribution.

A player's "real" batting average may change over the course of their career (players usually get better, and then decline in old age)

Why did we look at posterior mean to estimate the "true" batting average? Why not posterior mode? Or posterior median?

Posterior mean corresponds to optimizing squared error, posterior median to mean absolute error, and posterior mode to 0-1 loss.