Homework 5

PSTAT 115, Winter 2024

Due on March 17, 2024 at 11:59 pm

```
options(tinytex.verbose = TRUE)
options(buildtools.check = function(action) TRUE )
knitr::opts_chunk$set(echo = TRUE, eval=FALSE)
suppressPackageStartupMessages(library(tidyverse))
suppressPackageStartupMessages(library(cmdstanr))
suppressPackageStartupMessages(library(testthat))
library(coda)
```

Problem 1. Estimating Skill In Baseball

In baseball, the batting average is defined as the fraction of base hits (successes) divided by "at bats" (attempts). We can conceptualize a player's "true" batting skill as $p_i = \lim_{n_i \to \infty} \frac{y_i}{n_i}$. In other words, if each at bat was independent (a simplifying assumption), p_i describes the total fraction of success for player i as the number of attempts gets very large. Our goal is to estimate the true skill of all players as best as possible using only a limited amount of data. As usual, for independent counts of success/fail data it is reasonable to assume that $Y_i \sim \text{Bin}(n_i, p_i)$. The file "lad.csv" includes the number of hits, `y` and the number of attempts `n` for $J = 13$ players on the Los Angeles Dodgers after the first month of the 2023 baseball season. The variable `val` includes the end-of-season batting average and will be used to validate the quality of various estimates. If you are interested, at the end of the assignment we have included the code that was used to scrape the data.

```
baseball_data <- read_csv("lad_2023.csv", col_types=cols())
baseball_data

## observed hits in the first month
y <- baseball_data$y

## observed at bats in the first month
n <- baseball_data$n

## observed batting average in the first month (same as MLE)
theta_mle <- y/n

## number of players
J <- nrow(baseball_data)

## end of the year batting average, used to evaluate estimates
val <- baseball_data$val
```

1a. Compute the standard deviation of the empirical batting average, y/n and then compute the sd of the "true skill", (the `val` variable representing the end of season batting average). Which is smaller? Why does this make sense? *Hint:* What sources of variation are present in the empirical batting average?

Type your answer here, replacing this text.

```
empirical_sd <- NULL # YOUR CODE HERE
true_sd <- NULL # YOUR CODE HERE
print(empirical_sd)
print(true_sd)
```

```
. = ottr::check("tests/q1a.R")
```

1b. Consider two estimates for the true skill of player i, p_i: 1) $\hat{p}_i^{(\text{mle})} = \frac{y_i}{n_i}$ and 2) $\hat{p}_i^{(\text{comp})} = \frac{\sum_j y_j}{\sum_j n_j}$. Estimator 1) is the MLE for each player and ignores any commonalities between the observations. This is sometimes termed the "no pooling" estimator since each parameter is estimating separately without "pooling" information between them. Estimator 2) assumes all players have identical skill and is sometimes called the "complete pooling" estimator, because the data from each problem is completely "pooled" into one common set. In this problem, we'll treat the end-of-season batting average as a proxy for true skill, p_i. Compute the root mean squared error (RMSE), $\sqrt{\frac{1}{J}\sum_i (\hat{p}_i - p_i)^2}$ for the "no pooling" and "complete pooling" estimators using the variable `val` as a stand-in for the true p_i. Does "no pooling" or "complete pooling" give you a better estimate of the end-of-year batting averages in this specific case?

Type your answer here, replacing this text.

```
# Maximum likelihood estimate
phat_mle <- NULL # YOUR CODE HERE

# Pooled estimate
phat_pooled <- NULL # YOUR CODE HERE

rmse_complete_pooling <- NULL # YOUR CODE HERE
rmse_no_pooling <- NULL # YOUR CODE HERE

print(sprintf("MLE: %f", rmse_no_pooling))
print(sprintf("Complete Pooling: %f", rmse_complete_pooling))
```

```
. = ottr::check("tests/q1b.R")
```

The no pooling and complete pooling estimators are at opposite ends of a spectrum. There is a more reasonable compromise: "partial pooling" of information between players. Although we assume the number of hits follow a binomial distribution. To complete this specification, we assume $\text{logit}(p_i) \sim N(\mu, \tau^2)$ for each player i. μ is the "global mean" (on the logit scale), $\exp(\mu)/(1+\exp(\mu)$ is the overall average batting average across all players. τ describes how much variability there is in the true skill of players. If $\tau = 0$ then all players are identical and the only difference in the observed hits is presumed to be due to chance. If τ^2 is very large then the true skill differences between players is assumed to be large and our estimates will be close to the "no pooling" estimator. How large should τ be? We don't know but we can put a prior distribution over the parameter and sample it along with the p_i's! Assume the following model:

$$
\begin{aligned}
y_i &\sim Bin(n_i, p_i) \\
\theta_i &= logit(p_i) \\
\theta &\sim N(\mu, \tau^2) \\
p(\mu) &\propto \text{const} \\
p(\tau) &\propto \text{Cauchy}(0,1)^+, \text{ (the Half-cauchy distribution, see part d.)}
\end{aligned}
$$

1c. State the correct answer in each case: as $\tau \to \infty$, the posterior mean estimate of p_i in this model will approach the (complete pooling / no pooling) estimator and as $\tau \to 0$ the posterior mean estimate of p_i will approach the (complete pooling / no pooling) estimator. Give a brief justification for your answer.

Type your answer here, replacing this text.

1d. Implement the hierarchical binomial model in Stan. As a starting point for your Stan file modify the `eight_schools.stan` file we have provided and save it as `baseball.stan`. To write the hierarchical binomial model, we need the following modifications to the normal hierarchical model: - Since we are fitting a hierarchical binomial model, not a normal distribution, we no longer need sampling variance σ_i^2. Remove this from the data block. - The outcomes `y` are now integers. Change `y` to an array of integer types in the data block. - We need to include the number of at bats for each player (this is part of the binomial likelihood). Add an array of integers, `n` of length J to the data block. - Replace the sampling model for y with the binomial-logit: `binomial_logit(n, theta)`. This is equivalent to `binomial(n, inv_logit(theta))`. - The model line for `eta` makes $\theta_i \sim N(\mu, \tau^2)$. Leave this in the model. - Add a half-cauchy prior distribution for τ: `tau ~ cauchy(0, 1);`. The half-cauchy has been suggested as a good default prior distribution for group-level standard deviations in hierarchical models. See http://www.stat.columbia.edu/~gelman/research/published/taumain.pdf.

Find the posterior means for each of the players batting averages by looking at the samples for `inv_logit(theta_samples)`. Report the RMSE for hierarchical estimator. How does this compare to the RMSE of the complete pooling and no pooling estimators? Which estimator had the lowest error?

```
# Run Stan and compute the posterior mean

stan_data <- list(J=J, n=n, y=y)
baseball_model <- cmdstan_model("baseball.stan")
baseball_results <- baseball_model$sample(data=stan_data, iter_sampling=2000, refresh=0)
baseball_draws <- baseball_results$draws(format="df")

# Get the matrix of Theta samples for all players
# Should be a matrix of size num_draws * num_players
# where num_players is 10
# Note: theats are on logit scale, need to convert back to prob
theta_samples <- NULL # YOUR CODE HERE

# Get batting averages by inverting with this function
inv_logit <- function(x) {
  # YOUR CODE HERE
}
# and compute the posterior mean for each theta
pm <- NULL # YOUR CODE HERE


# RMSE From Stan posterior means
rmse_partial_pooling <- NULL # YOUR CODE HERE

print(c(rmse_complete_pooling, rmse_no_pooling, rmse_partial_pooling))
```

```
. = ottr::check("tests/q1d.R")
```

1e. Use the `shrinkage_plot` function provided below to show how the posterior means shrink the empirical batting averages. Pass in `y/n` and the posterior means of p_i as arguments.

Type your answer here, replacing this text.

```
shrinkage_plot <- function(empirical, posterior_mean,
                           shrink_point=mean(posterior_mean)) {

  tibble(y=empirical, pm=posterior_mean) %>%
    ggplot() +
    geom_segment(aes(x=y, xend=pm, y=1, yend=0), linetype="dashed") +
```

```
    geom_point(aes(x=y, y=1)) +
    geom_point(aes(x=pm, y=0)) +
    theme_bw(base_size=16) +
    geom_vline(xintercept=shrink_point, color="blue", size=1.2) +
    ylab("") + xlab("Estimate") +

    scale_y_continuous(breaks=c(0, 1),
                       labels=c("Posterior Mean", "MLE"),
                       limits=c(0,1))

}

# YOUR CODE HERE
```

```
# YOUR CODE HERE
```

Appendix: Code for scraping Dodgers baseball data

http://billpetti.github.io/baseballr/

```
## Install the baseballr package
devtools::install_github("BillPetti/baseballr")

library(baseballr)
library(tidyverse)

## Download data from the chosen year
year <- 2023

one_month <- daily_batter_bref(t1 = sprintf("%i-04-01", year), t2 = sprintf("%i-05-01", year))
one_year <- daily_batter_bref(t1 = sprintf("%i-04-01", year), t2 = sprintf("%i-10-01", year))

## filter to only include players who hat at least 10 at bats in the first month
one_month <- one_month %>% filter(AB > 10)
one_year <- one_year %>% filter(Name %in% one_month$Name)

one_month <- one_month %>% arrange(Name)
one_year <- one_year %>% arrange(Name)

## Look at only the Dodgers
LAD <- one_year %>% filter(Team == "Los Angeles" & Level == "Maj-NL") %>% .$Name

lad_month <- one_month %>% filter(Name %in% LAD)
lad_year <- one_year %>% filter(Name %in% LAD)

write_csv(tibble(name=lad_month$Name,
                 y=lad_month$H,
                 n=lad_month$AB,
                 val=lad_year$BA),
          file="lad_2023.csv")
```