Mini Project 1

Samantha Goerger (sgoerger) & Mikako Inaba (minaba)

October 14, 2019

Problem 1

```
set.seed(08544)
N.p1 <- c(5000, 10000, 100000, 250000, 300000, 450000)
ps <- c()
allX <- c()
allY <- c()

for (i in 1:length(N.p1)){
  e <- rnorm(N.p1[i], 0, sd = 0.01)
  x <- rnorm(N.p1[i], 0, sd = 0.01)
  y <- 0.5 + 0.01*x + e
  allX <- c(allX, x)
  allY <- c(allY, y)
  summary(lm(y ~ x))
  ps[i] <- summary(lm(y ~ x))$coefficients[2,4]
  }
```

```
plot(allX[1:1000], allY[1:1000],
     main = "Y vs X",
     sub = "Figure 1: Correlation between X and Y is 0.01",
     xlab = "X",
     ylab = "Y")
```

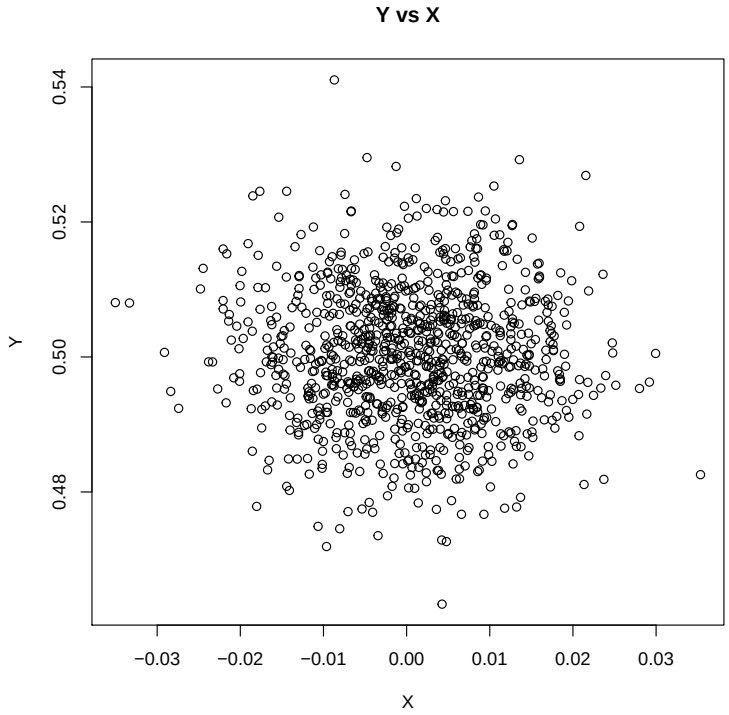


Figure 1: Correlation between X and Y is 0.01

```
plot(N.p1, ps,
     main = "P-value vs Number of Observations",
     sub = "Figure 2",
     ylab = "P-value",
     xlab = "Number of Observations"
)
abline(h = .05)
```

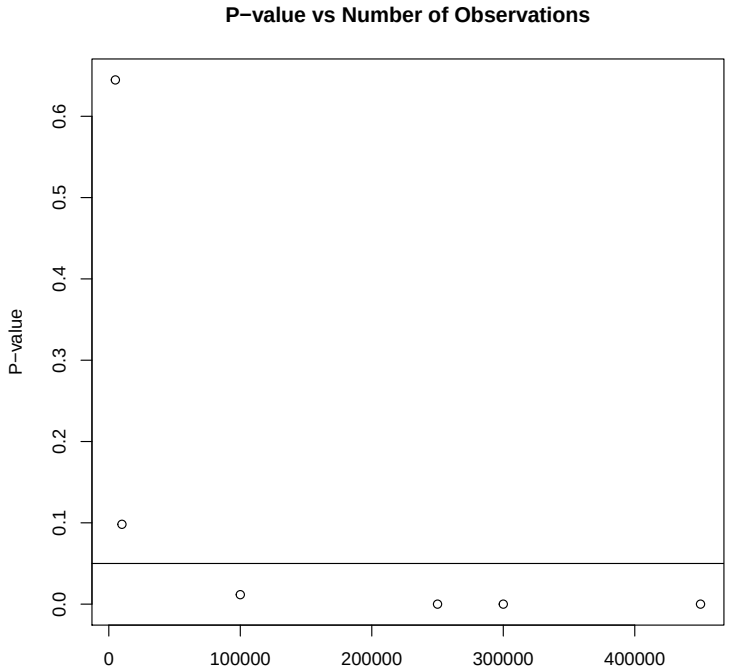


Figure 2

We modeled X and e such that both were normal with mean 0 and standard deviation 0.01. We set the parameters a_0 and a_1 to 0.5 and 0.01, respectively, such that $Y = 0.5 + 0.01 * X + e$. The relationship between X and Y is weak, as seen in Figure 1. The plotted values form a cloud, indicating the weak correlation of 0.01 (we only plotted the first 1000 Xs and Ys due to the large number of data generated). We generated datasets for samples of increasing size, from 5000 to 450000, and plotted the p-values of X for each of the 6 trials in Figure 2. A horizontal line was drawn at a p-value of 0.05, the common threshold for rejecting the null hypothesis. The relationship is consistently statistically significant after 100000 samples, which shows that as the sample size increases, even weak relationships will be shown to be significant. Therefore, low p-values do not necessarily imply an important assocation between two variables.

Problem 2

Part a

```
set.seed(08544)
N.p2 <- 1000

x1 <- rbinom(N.p2, size = 1, prob = .056)
x2 <- rnorm(N.p2, (5 - x1*3), 1.5)
y <- plogis(x2 - 6)
```

We completed Problem 2b prior to 2a, which is why the code is nearly identical. The creation of the dataset is described in 2b. The correlation between X_1 and Y is -0.252, while the coefficient on X_1 from the regression is 0.191, as required.

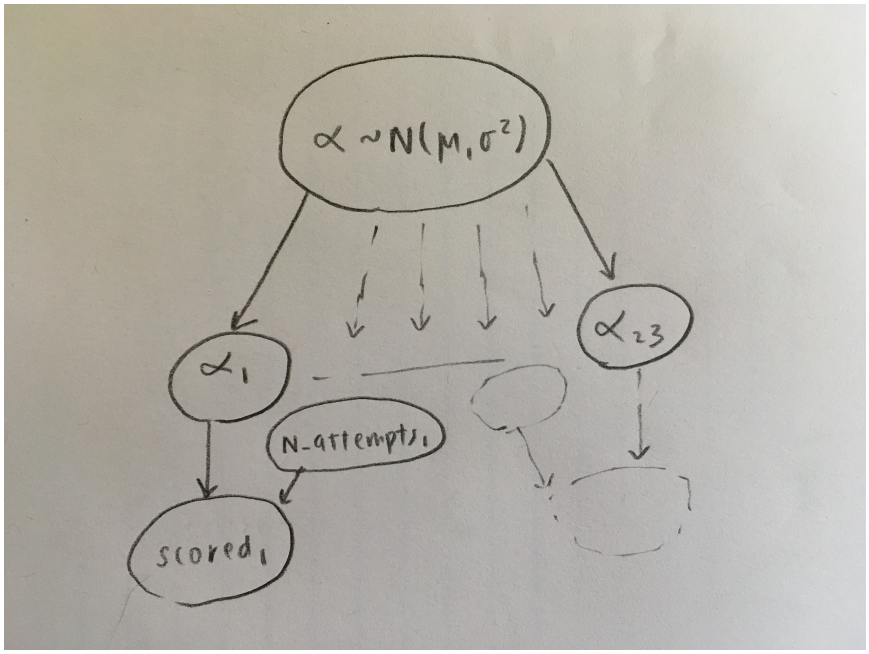
Part b

```
set.seed(08544)

asian <- rbinom(N.p2, size = 1, prob = .056)
personality <- rnorm(N.p2, (5 - asian*3), 1.5)
harvard <- plogis(personality - 6)
```

We chose to comment on the Harvard Circut Court decision on admission of Asian American applicants. Here, X_1 is Bernoulli variable that flags a candidate as Asian or not (5.6% of the U.S. population is Asian). X_2 is the 'personality score' as described in the brief. X_2 is normally distributed with mean 5 and standard deviation 1.5 if a candidate is non-Asian. If the candidate is Asian, the 'personality score' is lower, with X_2 normally distributed with mean 2 and standard deviation 1.5. Y is the probability of acceptance to Harvard based on the personality score (which is lower for Asian applicants). We can see that the 'personality score' is a lurking variable. The correlation between Asian and Harvard is -0.252. However, for a constant 'personality score,' the coefficient on Asian is 0.191.

Problem 3

Part a



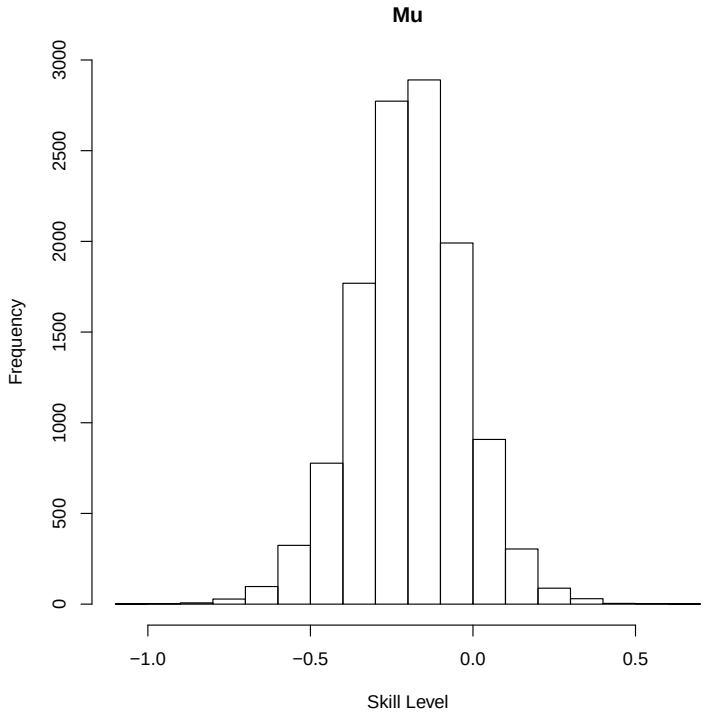
Shown in the diagram above, the partial-pooling hierarchical model would make sense if Shaq had a different probablity of scoring on different days. We assumed skill level is normally distributed (α), and each game Shaq has a different skill level from this distribution (α_i). Each α_i influences the probability of scoring for that game in addition to the number of attempts.

Part b

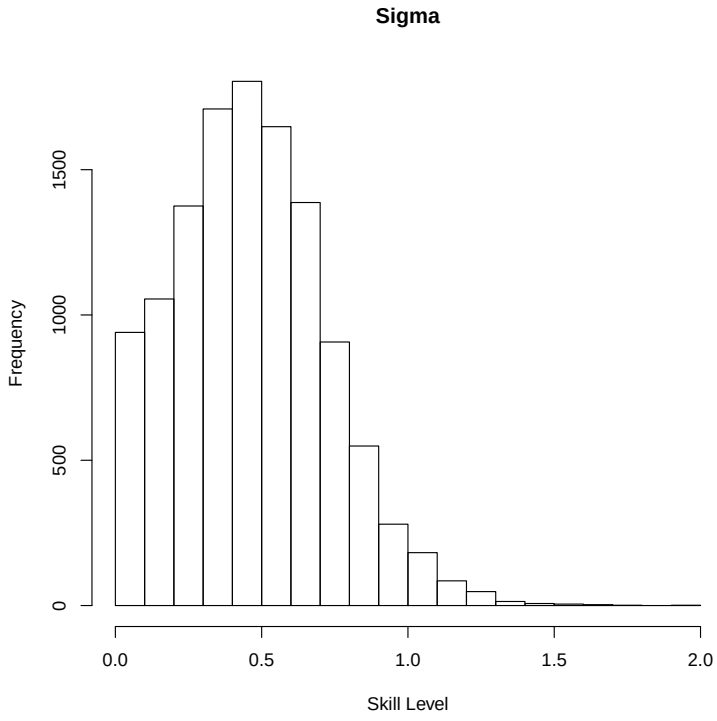
```
lines <-
"Game   Scored  N.Attempts
1   4   5
2   5   11
3   5   14
4   5   12
5   2   7
6   7   10
7   6   14
8   9   15
9   4   12
10  1   4
11  13  27
12  5   17
13  6   12
14  9   9
15  7   12
16  3   10
17  8   12
18  1   6
19  18  39
20  3   13
21  10  17
22  1   6
23  3   12"
con <- textConnection(lines)
shaq <- read.csv(con, sep="")
shaq

shaq_model_stanPartial <- "
data{
  // data supplied
  int<lower=0> N; // game number, must be positive
  int scored[N]; // array of scores indexed by game
  int attempted[N]; // array of number of attempts indexed by game
}

parameters{
  real mu; // overall average skill level
  real<lower=0> sigma; // overall stdev of skill level, lower bounded by 0
  vector[N] alpha_norm; // array of `zscores' / number of stdev away from mean
                        // indexed by game
}

transformed parameters{
  real alpha[N]; // array of average skill level, produced using alpha_norm
                 // indexed by game

  // loop populates alpha from ~N(mu, sigma^2)
  for(n in 1:N)
    alpha[n] = mu + sigma * alpha_norm[n];
```

```
}

model{
  alpha_norm ~ normal(0, 1);
  scored ~ binomial(attempted, inv_logit(alpha));
  // scored is the sum of n Bernoulli RVs, each with a probability determined by alpha
  // n determined by the number of attempted shots
}"

adaptSteps = 1000             # Number of steps to "tune" the samplers.
burnInSteps = 5000            # Number of steps to "burn-in" the samplers.
nChains = 3                   # Number of chains to run.
numSavedSteps=12000           # Total number of steps in chains to save.
thinSteps=10                  # Number of steps to "thin" (1=keep every step).

shaq_model_stanPartial <- stan_model(model_code = shaq_model_stanPartial,
                                     model_name = "shaq_model_stanPartial")
shaqPartial_fit <- sampling(object=shaq_model_stanPartial,
                      data = list(N=nrow(shaq), scored=shaq$Scored,
                                  attempted = shaq$N.Attempts),
                      chains = nChains,
                      iter = (ceiling(numSavedSteps/nChains)*thinSteps
                              + burnInSteps),
                      warmup = burnInSteps,
                      thin = thinSteps,
                      init = "random")

samplesPartial <- extract(shaqPartial_fit)
good <- plogis(mean(samplesPartial$mu) + 2 * mean(samplesPartial$sigma))
bad <- plogis(mean(samplesPartial$mu) - 2 * mean(samplesPartial$sigma))
```

```
hist(samplesPartial$mu, main = "Mu", xlab = "Skill Level")
```




```
hist(samplesPartial$sigma, main = "Sigma", xlab = "Skill Level")
```



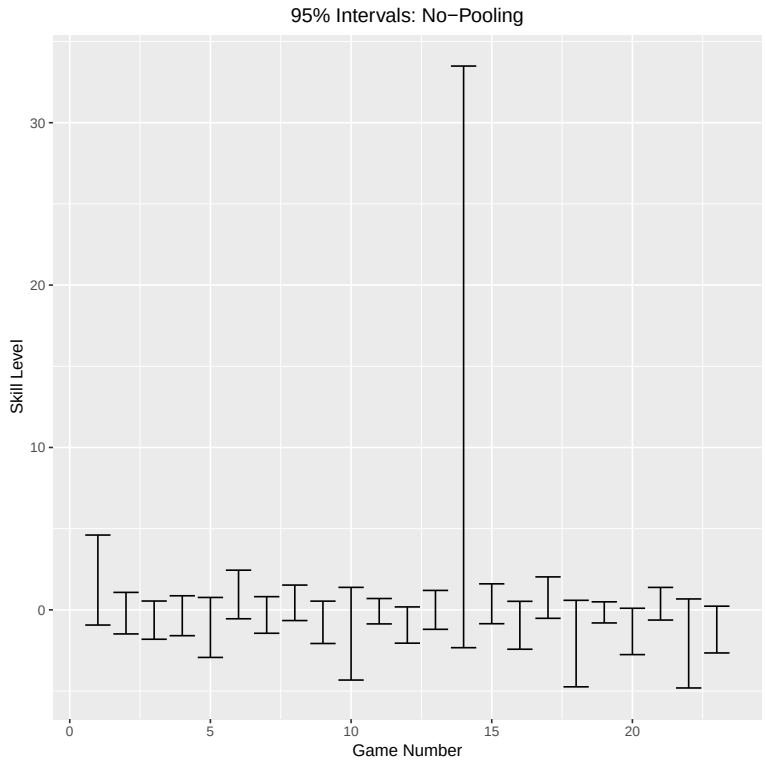
The posterior distribution for mu and sigma were displayed. Given that 95% of the data lies within 2 standard deviations from the mean, we found the probability of scoring on a particularly bad day (2 standard

deviations lower than the mean) and on an especially good day (2 standard deviations above the mean). On a bad day, Shaq's probability of making a free throw is 0.245, and on a good day, Shaq's probability of making a free throw is 0.676. This is a difference of 0.43, which means that on a good day, Shaq has a 43.04 percentage point increase in scoring probability. This difference could alter the course of a basketball game, so we decided this was large enough to conclude that Shaq has good and bad days. Each line of Stan code was commented above.

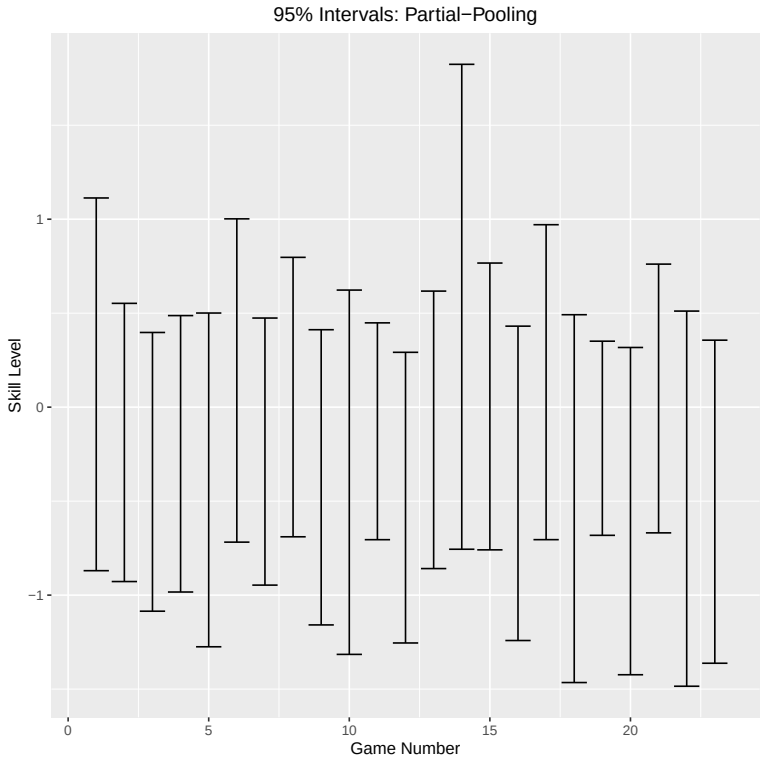
Part c

```
shaq_model_stanNo <- "
data{
  int<lower=0> N;
  int scored[N];
  int attempted[N];
}

parameters{
  real mu;
  vector[N] alpha_norm;
}

transformed parameters{
  real alpha[N];
  for(n in 1:N)
    alpha[n] = mu + 20 * alpha_norm[n];
}

model{
  alpha_norm ~ normal(0, 1);
  scored ~ binomial(attempted, inv_logit(alpha));
}"

adaptSteps = 1000             # Number of steps to "tune" the samplers.
burnInSteps = 5000            # Number of steps to "burn-in" the samplers.
nChains = 3                   # Number of chains to run.
numSavedSteps=12000           # Total number of steps in chains to save.
thinSteps=10                  # Number of steps to "thin" (1=keep every step).

shaq_model_stanNo <- stan_model(model_code = shaq_model_stanNo,
                                model_name = "shaq_model_stanNo")
shaqNo_fit <- sampling(object=shaq_model_stanNo,
                       data = list(N=nrow(shaq), scored=shaq$Scored,
                                   attempted = shaq$N.Attempts),
                       chains = nChains ,
                       iter = ( ceiling(numSavedSteps/nChains)*thinSteps
                                +burnInSteps ) ,
                       warmup = burnInSteps ,
                       thin = thinSteps ,
                       init = "random" )

samplesNo <- extract(shaqNo_fit)
```

```r
N.p3 <- 23
conf_dataNo <- data.frame(games = 1:N.p3, mean = NA, upper = NA, lower = NA)

for (i in 1:N.p3){
  conf_dataNo[i, "mean"] = mean(samplesNo$alpha[, i])
  conf_dataNo[i, "upper"] = mean(samplesNo$alpha[, i]) + 2 * sd(samplesNo$alpha[, i])
  conf_dataNo[i, "lower"] = mean(samplesNo$alpha[, i]) - 2 * sd(samplesNo$alpha[, i])
}

ggplot(conf_dataNo, aes(y = mean, x = games)) +
  geom_errorbar(aes(ymin=lower, ymax=upper)) + ggtitle("95% Intervals: No-Pooling") +
  theme(plot.title = element_text(hjust = 0.5)) +
  ylab("Skill Level") + xlab("Game Number")
```



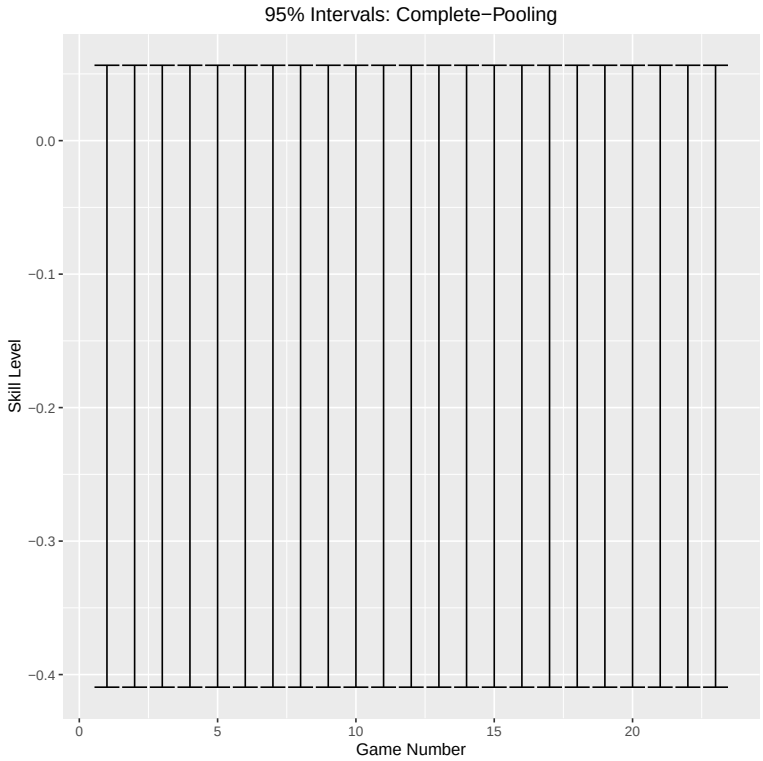

```r
conf_dataPartial <- data.frame(games = 1:N.p3, mean = NA, upper = NA, lower = NA)

for (i in 1:N.p3){
  conf_dataPartial[i, "mean"] = mean(samplesPartial$alpha[, i])
  conf_dataPartial[i, "upper"] = mean(samplesPartial$alpha[, i]) +
                                 2 * sd(samplesPartial$alpha[, i])
  conf_dataPartial[i, "lower"] = mean(samplesPartial$alpha[, i]) -
                                 2 * sd(samplesPartial$alpha[, i])
}

ggplot(conf_dataPartial, aes(x = games, y = mean)) +
  geom_errorbar(aes(ymin=lower, ymax=upper)) +
  ggtitle("95% Intervals: Partial-Pooling") +
  theme(plot.title = element_text(hjust = 0.5)) +
  ylab("Skill Level") + xlab("Game Number")
```




```
shaq_model_stanComplete <- "
data{
  int<lower=0> N;
  int scored[N];
  int attempted[N];
}

parameters{
  real mu;
  vector[N] alpha_norm;
}

transformed parameters{
  real alpha[N];
  for(n in 1:N)
    alpha[n] = mu + 0 * alpha_norm[n];
}

model{
  alpha_norm ~ normal(0, 1);
  scored ~ binomial(attempted, inv_logit(alpha));
}"

shaq_model_stanComplete <- stan_model(model_code = shaq_model_stanComplete,
                                      model_name = "shaq_model_stanComplete")
shaqComplete_fit <- sampling(object=shaq_model_stanComplete,
                      data = list(N=nrow(shaq), scored=shaq$Scored,
```

```
                                attempted = shaq$N.Attempts),
                        chains = nChains ,
                        iter = ( ceiling(numSavedSteps/nChains)*thinSteps
                                 +burnInSteps ) ,
                        warmup = burnInSteps ,
                        thin = thinSteps ,
                        init = "random" )

samplesComplete <- extract(shaqComplete_fit)
```

```
conf_dataComplete <- data.frame(games = 1:N.p3, mean = NA, upper = NA, lower = NA)

for (i in 1:N.p3){
  conf_dataComplete[i, "mean"] = mean(samplesComplete$alpha[, i])
  conf_dataComplete[i, "upper"] = mean(samplesComplete$alpha[, i]) +
                                  2 * sd(samplesComplete$alpha[, i])
  conf_dataComplete[i, "lower"] = mean(samplesComplete$alpha[, i]) -
                                  2 * sd(samplesComplete$alpha[, i])
}

ggplot(conf_dataComplete, aes(x = games, y = mean)) +
  geom_errorbar(aes(ymin=lower, ymax=upper)) +
  ggtitle("95% Intervals: Complete-Pooling") +
  theme(plot.title = element_text(hjust = 0.5)) +
  ylab("Skill Level") + xlab("Game Number")
```



The 95% predictive interval for each day was plotted for each of the three models, no-pooling, partial-pooling, and complete-pooling. The x-axis is the game number and the y-axis is the skill level. No-pooling assumes

there is no connection between Shaq's skill level at each game. This translates to a very large sigma (in this case, 20). Partial-pooling assumes that Shaq's skill level differs at each game, dictated by the overall sigma (more specifically explained in 3a). Complete-pooling pools all the games into one, assuming that each game has the same skill level. This translates to a sigma of 0.

*Side note with respect to reproducibility - the image is in a folder on my Desktop, but with that commented out, everything else should be reproducibile.