Data Analysis Report

Advanced Bayesian Modeling, Spring 2022, Ron Hong

5/8/2022

Introduction

In basketball, a field goal is any basket scored during regular game play.[1] This report will analyze the field goals successfully made and attempted by the 2021–22 University of Illinois men's basketball season, which was ranked 1st in the Big Ten and 19th overall in the NCAA Division I, with 23 wins and 10 losses.[2] It will include 15 players and 31 games.

Data

The data for this report is provided in `IlliniMensBB_FG_21_22.csv`. It consists of four columns:

- **Game**: date of the basketball game
- **Player**: surname of the player
- **FGM**: number of field goals successfully made by the player during the game
- **FGA**: total number of field goals attempted by the player during the game

Each row represents a different combination of players and games in which at least one field goal was attempted.

The table below is created by aggregating the data for each player over all games.

Player	total number of games with at least one field goal attempt	total number of field goals made	total number of field goal attempts	apparent field goal success rate
Bosmans-Verdonk	12	17	36	0.4722222
Cockburn	26	214	358	0.5977654
Curbelo	17	48	138	0.3478261
Frazier	29	119	292	0.4075342
Goode	21	16	48	0.3333333
Grandison	28	102	224	0.4553571
Hawkins	27	62	141	0.4397163
Hutcherson	3	3	8	0.3750000
Lieb	2	4	5	0.8000000
Melendez	18	25	46	0.5434783
Payne	18	20	33	0.6060606

[1] MasterClass staff, "How Basketball Scoring Works: Inside the 3 Ways to Score," in *MasterClass* (24 May 2021), `https://www.masterclass.com/articles/basketball-scoring-guide#3-ways-to-score-in-basketball`

[2] "Illinois Fighting Illini," in *ESPN*, `https://www.espn.com/mens-college-basketball/team/_/id/356`

Player	total number of games with at least one field goal attempt	total number of field goals made	total number of field goal attempts	apparent field goal success rate
Plummer	31	147	341	0.4310850
Podziemski	8	8	19	0.4210526
Serven	1	0	1	0.0000000
Williams	30	42	136	0.3088235

From this table, Cockburn has the most total field goal attempts, while Serven has the least. Lieb has the highest apparent field goal success rate, while Serven has the lowest.

First Model

The first model treats the response, which is the number of field goals successfully made by a player during a game, FGM, as being distributed from a binomial, with its size being the number of field goals attempted by the player during the game, FGA. Its probability, p, is found using logistic regression, and depends on an intercept parameter, β_0, and indicator parameters for each player, β^{player}. Here, i ranges over the indices for the combinations of players and games in which at least one field goal was attempted.

$$FGM_i|\beta, X_i \sim \text{Bin}(FGA_i, p_i)$$
$$\text{logit}(p_i) = \beta_0 + \beta^{player}_{player_i}$$

The intercept parameter, β_0, has a Cauchy prior distribution to prevent an improper posterior in the case of a complete separation. The indicator parameters, β^{player}, have a normal prior distribution with the variance being a hyperparameter, σ_{player}. Here, j ranges over the indices for the players.

$$\beta_0 \sim t_1(0, 10^2)$$
$$\beta^{player}_j|\sigma_{player} \sim \text{N}(0, \sigma^2_{player})$$

The hyperparameter, σ_{player}, has a uniform hyperprior distribution to approximate an improper distribution.

$$\sigma_{player} \sim \text{U}(0, 10)$$

Since the β^{player} have a normal prior distribution with 0 mean, they are treated as random effects.

The JAGS code for this model is listed below. It includes the variable FGM^{rep} to test for overdispersion.

```
model {

  for (i in 1:length(FGM)) {
    FGM[i] ~ dbin(prob[i], FGA[i])
    logit(prob[i]) <- beta0 + betaplayer[player[i]]

    FGMrep[i] ~ dbin(prob[i], FGA[i])
  }

  beta0 ~ dt(0, 0.01, 1)

  for (j in 1:max(player)) {
    betaplayer[j] ~ dnorm(0, 1 / sigmaplayer ^ 2)
  }

  sigmaplayer ~ dunif(0, 10)

}
```

The Markov Chain Monte Carlo runs four chains with overdispersed starting values. The chains are run for 2000 iterations of burn-in, and 20000 iterations for the posterior sample, to ensure that the effective sample size of each top-level parameter is at least 2000. Convergence is accessed using the Gelman-Rubin statistic. The effective sample sizes of the top-level parameters are listed below.

parameter	n_{eff}
β_0	3451.253
σ_{player}	10788.357

The chi-square discrepancy is used to check the model for overdispersion.

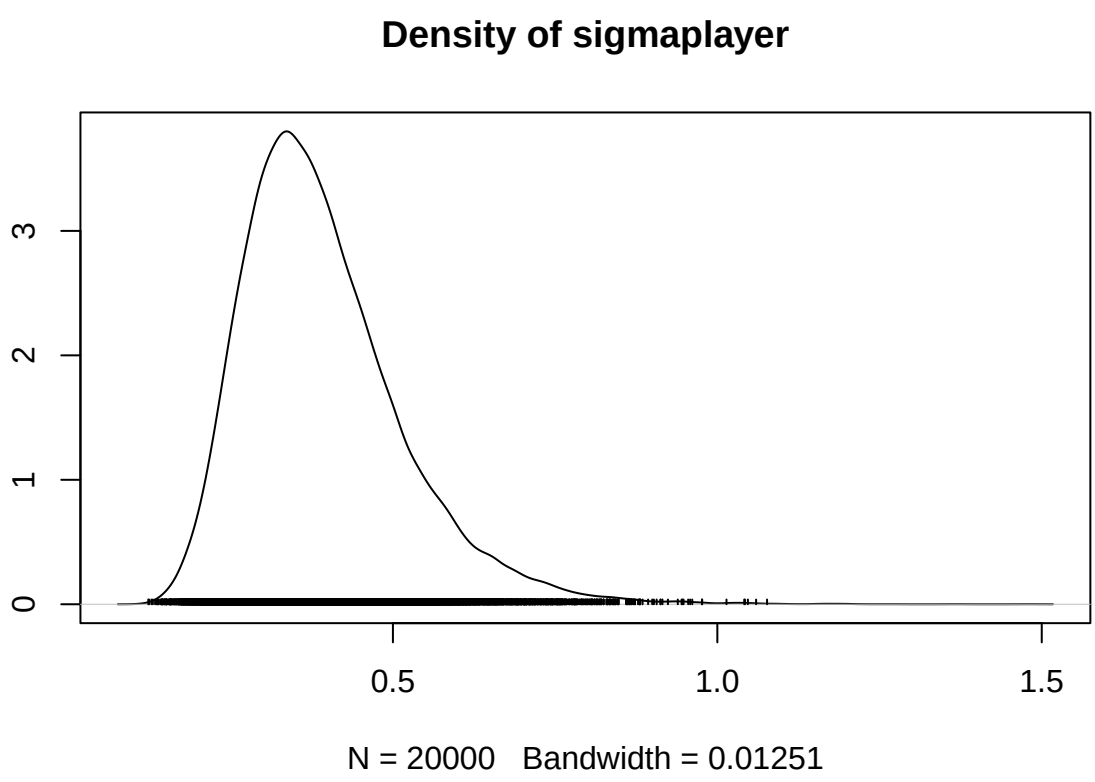
$$T(FGM, \beta, X) = \sum_{i=0}^{n} \frac{(FGM_i - FGA_i p_i)^2}{FGA_i p_i (1 - p_i)}$$

The posterior predictive p-value is,

$$\Pr(T(FGM^{\text{rep}}, \beta, X) \geq T(FGM, \beta, X) | FGM)$$

$p = 0.454975$, so there is no evidence of overdispersion.

The approximate posterior density of σ_{player} is graphed below. Since it does not appear to include 0, actual differences in field goal success rates among the players are suggested.



The modeled probability that Cockburn successfully makes an attempted field goal has the approximate 95% central posterior interval of (0.5341396, 0.6349736).

The posterior probability that Cockburn has the highest probability of success during a new field goal attempt is 0.5464625 according to the model. If all players have equal prior probability of having the highest

probability of success, then Cockburn's prior probability is $\frac{1}{15}$, since there are 15 players, and the approximate Bayes factor in favor of Cockburn having the highest probability of success is 16.86845. Therefore, the data evidence that Cockburn has the highest probability of success is positive.

The DIC for this model is approximately 817.6 and its effective number of parameters is 9.753. The nonredundant parameters in this model include β_0, β^{player}, and σ_{player} for a total of 17 parameters, so the hierarchical structure of this model allows the effective number of parameters to be lower, due to partial pooling.

Second Model

The second model extends from the first by investigating if the game changes the probability that a field goal is successfully made. A random effect parameter, representing the games, is added to the logistic regression.

$$\text{logit}(p_i) = \beta_0 + \beta^{player}_{player_i} + \beta^{game}_{game_i}$$

The game indicator parameters, β^{game}, and their hyperparameter, σ_{game}, are defined similarly to β^{player} and σ_{player}. Here, k ranges over the indices for the games.

$$\beta^{game}_k | \sigma_{game} \sim \text{N}(0, \sigma^2_{game})$$
$$\sigma_{game} \sim \text{U}(0, 10)$$

All other aspects of the previous model are unchanged.

The JAGS code for this model is listed below.

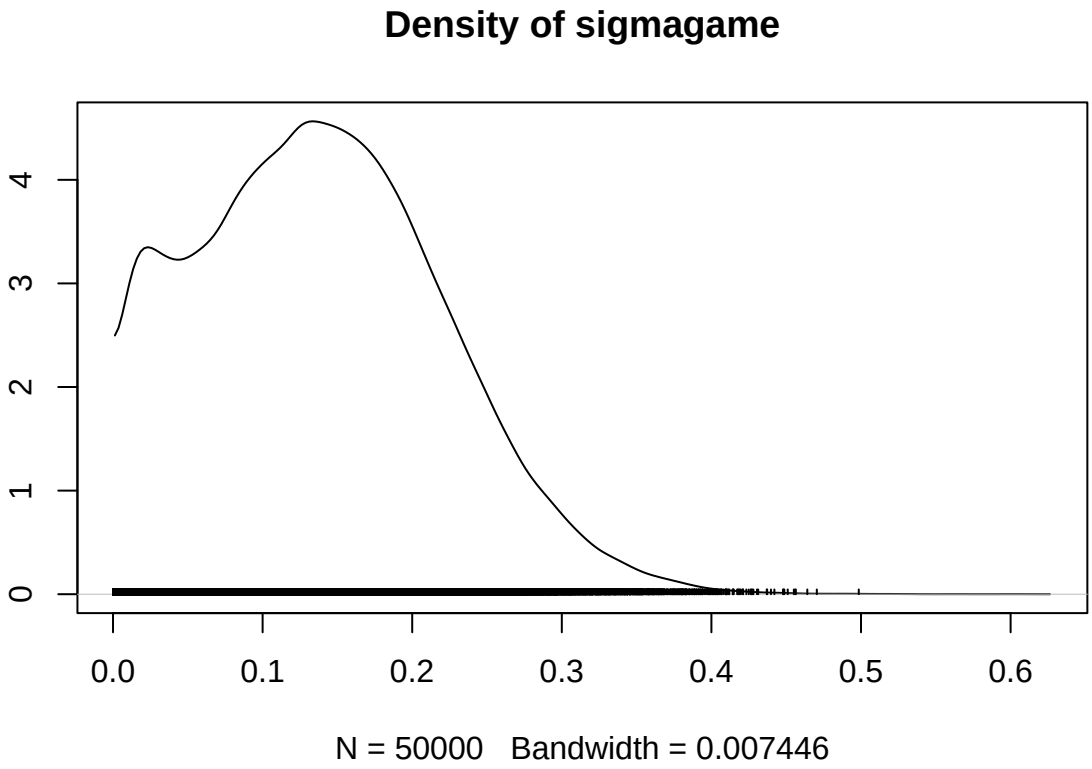
```
model {

  for (i in 1:length(FGM)) {
    FGM[i] ~ dbin(prob[i], FGA[i])
    logit(prob[i]) <- beta0 + betaplayer[player[i]] + betagame[game[i]]

    FGMrep[i] ~ dbin(prob[i], FGA[i])
  }

  beta0 ~ dt(0, 0.01, 1)

  for (j in 1:max(player)) {
    betaplayer[j] ~ dnorm(0, 1 / sigmaplayer ^ 2)
  }

  sigmaplayer ~ dunif(0, 10)

  for (k in 1:max(game)) {
    betagame[k] ~ dnorm(0, 1 / sigmagame ^ 2)
  }

  sigmagame ~ dunif(0, 10)

}
```

The Markov Chain Monte Carlo runs four chains with overdispersed starting values. The chains are run for 2000 iterations of burn-in, and 50000 iterations for the posterior sample, to ensure that the effective sample

size of each top-level parameter is at least 2000. Convergence is accessed using the Gelman-Rubin statistic. The effective sample sizes of the top-level parameters are listed below.

parameter	n_{eff}
β_0	7922.203
σ_{player}	21820.970
σ_{game}	2024.484

The approximate posterior density of σ_{game} is graphed below. Since it appears to reach 0, actual differences in field goal success rates among the games could be insignificant.



The DIC for this model is approximately 817.7 and its effective number of parameters is 17.27. Since the DIC for the first model and the second model are approximately the same, neither is strongly preferred, but the first model should be chosen as the simpler model.

Conclusion

Based on a comparison of the two models, the probability that a field goal is successfully made appears to depend on the player but not on the game. The preferred model suggests that Cockburn has the highest probability of success during a new field goal attempt, with that probability having a 95% chance of being within (0.5341396, 0.6349736).

Appendix

The R code used in the analysis is listed below. First, the data is loaded.

```
library(rjags)

IlliniMensBB = read.csv("IlliniMensBB_FG_21_22.csv")
IlliniMensBB$Player = factor(IlliniMensBB$Player)
```

Data

The table is constructed here, and used to answer questions about total attempts and apparent field goal success rates.

```
d2 = data.frame(Player = levels(IlliniMensBB$Player))
for (j in 1:nrow(d2)) {
  d2$FGG[j] = sum(IlliniMensBB$Player == d2$Player[j])
  d2$FGM[j] = sum(IlliniMensBB$FGM[IlliniMensBB$Player == d2$Player[j]])
  d2$FGA[j] = sum(IlliniMensBB$FGA[IlliniMensBB$Player == d2$Player[j]])
}
d2$FGP = d2$FGM / d2$FGA

d2$Player[which.max(d2$FGA)]
```

```
## [1] "Cockburn"
```

```
d2$Player[which.min(d2$FGA)]
```

```
## [1] "Serven"
```

```
d2$Player[which.max(d2$FGP)]
```

```
## [1] "Lieb"
```

```
d2$Player[which.min(d2$FGP)]
```

```
## [1] "Serven"
```

First Model

The first model requires the `FGM` and `FGA` columns, and the `Player` column enumerated.

```
d = IlliniMensBB[, c(3, 4)]
d$player = unclass(IlliniMensBB$Player)
```

Overdispersed starting values are chosen.

```
inits1 = list(
  list(
    beta0 = -10,
    sigmaplayer = 0.01,
    ".RNG.name" = "base::Wichmann-Hill",
    ".RNG.seed" = 1
  ),
  list(
    beta0 = -10,
    sigmaplayer = 9,
    ".RNG.name" = "base::Wichmann-Hill",
    ".RNG.seed" = 2
  ),
  list(
    beta0 = 10,
    sigmaplayer = 0.01,
    ".RNG.name" = "base::Wichmann-Hill",
    ".RNG.seed" = 3
  ),
  list(
    beta0 = 10,
    sigmaplayer = 9,
    ".RNG.name" = "base::Wichmann-Hill",
    ".RNG.seed" = 4
  )
)
m1 = jags.model("firstmodel.bug", d, inits1, 4)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 271
##    Unobserved stochastic nodes: 288
##    Total graph size: 1138
##
## Initializing model
```

2000 iterations of burn-in are run.

```
update(m1, 2000)
```

20000 iterations are used for the posterior samples.

```
x1 = coda.samples(m1,
                  c("beta0", "sigmaplayer", "betaplayer", "prob", "FGMrep"),
                  20000)
```

Convergence is accessed using the Gelman-Rubin statistic.

```
gelman.diag(x1[, c("beta0", "sigmaplayer")])
```

```
## Potential scale reduction factors:
##
##             Point est. Upper C.I.
## beta0                1          1
## sigmaplayer          1          1
##
## Multivariate psrf
##
## 1
```

The effective sample sizes of the top level parameters are found.

```
effectiveSize(x1[, c("beta0", "sigmaplayer")])
```

```
##       beta0 sigmaplayer
##    3451.253   10788.357
```

The p-value from the chi-square discrepancy is found.

```
prob = as.matrix(x1)[, paste("prob[", 1:nrow(d), "]", sep = "")]
FGMrep = as.matrix(x1)[, paste("FGMrep[", 1:nrow(d), "]", sep = "")]
Tchi = colSums((d$FGM - d$FGA * t(prob)) ^ 2 / (d$FGA * t(prob) * (1 - t(prob))))
Tchirep = colSums((t(FGMrep) - d$FGA * t(prob)) ^ 2 / (d$FGA * t(prob) * (1 - t(prob))))
mean(Tchirep >= Tchi)
```

```
## [1] 0.454975
```

The posterior density of σ_{player} is graphed.

```
densplot(x1[, "sigmaplayer"])
```

The 95% central posterior interval for Cockburn successfully making an attempted field goal is found.

```
beta0 = as.matrix(x1)[, "beta0"]
betaplayer = as.matrix(x1)[, paste("betaplayer[", 1:max(d$player), "]", sep = "")]
quantile(
  boot::inv.logit(beta0 + betaplayer[, which(levels(IlliniMensBB$Player) == "Cockburn")]),
  c(0.025, 0.975)
)
```

```
##      2.5%     97.5%
## 0.5341396 0.6349736
```

The posterior probability that Cockburn has the highest probability of success during a new field goal attempt is found, along with its Bayes factor.

```
(p = mean(
  apply(betaplayer, 1, which.max) == which(levels(IlliniMensBB$Player) == "Cockburn")
))
```

```
## [1] 0.5464625
```

```
p / (1 - p) / (1 / (max(d$player) - 1))
```

```
## [1] 16.86845
```

DIC and the effective number of parameters are computed.

```
dic.samples(m1, 100000)
```

```
## Mean deviance:  807.8
## penalty 9.753
## Penalized deviance: 817.6
```

Second Model

The second model requires the `Game` column enumerated.

```
d$game = unclass(factor(IlliniMensBB$Game))
```

Overdispersed starting values are chosen.

```
inits2 = list(
  list(
    beta0 = -10,
    sigmaplayer = 0.01,
    sigmagame = 0.01,
    ".RNG.name" = "base::Wichmann-Hill",
    ".RNG.seed" = 1
  ),
  list(
    beta0 = -10,
    sigmaplayer = 9,
    sigmagame = 9,
    ".RNG.name" = "base::Wichmann-Hill",
    ".RNG.seed" = 2
  ),
  list(
    beta0 = 10,
    sigmaplayer = 0.01,
    sigmagame = 9,
    ".RNG.name" = "base::Wichmann-Hill",
    ".RNG.seed" = 3
  ),
  list(
    beta0 = 10,
```

```
    sigmaplayer = 9,
    sigmagame = 0.01,
    ".RNG.name" = "base::Wichmann-Hill",
    ".RNG.seed" = 4
  )
)
m2 = jags.model("secondmodel.bug", d, inits2, 4)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 271
##    Unobserved stochastic nodes: 320
##    Total graph size: 1955
##
## Initializing model
```

2000 iterations of burn-in are run.

```
update(m2, 2000)
```

50000 iterations are used for the posterior samples.

```
x2 = coda.samples(m2, c("beta0", "sigmaplayer", "sigmagame"), 50000)
```

Convergence is accessed using the Gelman-Rubin statistic.

```
gelman.diag(x2)
```

```
## Potential scale reduction factors:
##
##             Point est. Upper C.I.
## beta0                1          1
## sigmagame            1          1
## sigmaplayer          1          1
##
## Multivariate psrf
##
## 1
```

The effective sample sizes of the top level parameters are found.

```
effectiveSize(x2)
```

```
##       beta0   sigmagame sigmaplayer
##    7922.203    2024.484   21820.970
```

The posterior density of σ_{player} is graphed.

```
densplot(x2[, "sigmagame"])
```

DIC and the effective number of parameters are computed.

```
dic.samples(m2, 100000)
```

```
## Mean deviance:  800.4
## penalty 17.27
## Penalized deviance: 817.7
```