Markov Decision Processes with Dynamic Transition Probabilities: An Analysis of Shooting Strategies in Basketball

A Simplified Walkthrough in R

Nathan Sandholtz and Luke Bornn

3/2/2020

Data

The folder `./data` contains one game of optical player tracking data from the 2012-2013 NBA regular season filtered to observations with tagged ball-events including dribbles, passes, turnovers, and shots. The game was between the Miami Heat and the Brooklyn Nets. Additionally, we filtered the data to plays in which no fouls occurred. For this simplified walkthrough, we have categorized players into three position groups: Guards (G), Forwards (F), and Centers (C).

```
# Load data
dat = read.csv("./data/2013_11_01_MIA_BRK_formatted.csv")
head(dat)
```

```
##         game   time quarter game_clock shot_clock entity team        x
## 1 2013110117  98398       1     675.56      24.00 172537  BRK 46.88853
## 2 2013110117  99198       1     674.76      23.20 172537  BRK 42.26108
## 3 2013110117  99358       1     674.60      23.04 172537  BRK 41.44996
## 4 2013110117 100278       1     673.68      22.12 172537  BRK 36.87506
## 5 2013110117 101318       1     672.64      21.08 172537  BRK 32.91451
## 6 2013110117 101998       1     671.96      20.40 172537  BRK 32.55346
##          y event_id       ndd change_poss play terminal_event location_id
## 1 11.02848       21 10.408324           1    2          FALSE       heave
## 2 11.64887       21  8.325748           0    2          FALSE       heave
## 3 11.88701       21  7.790360           0    2          FALSE       heave
## 4 13.75815       21  4.816156           0    2          FALSE       heave
## 5 16.13047       21  1.690733           0    2          FALSE       heave
## 6 17.08186       21  3.705388           0    2          FALSE       heave
##    def_pres points time_lapse firstname lastname position_simple home_away
## 1      open      0      -0.80     Deron Williams               G         h
## 2      open      0      -0.16     Deron Williams               G         h
## 3      open      0      -0.92     Deron Williams               G         h
## 4 contested      0      -1.04     Deron Williams               G         h
## 5 contested      0      -0.68     Deron Williams               G         h
## 6 contested      0      -0.24     Deron Williams               G         h
```

We also source some utility functions that will be used in the walkthrough. These functions include Algorithm 1 from the paper and functions used to get the initial states and shot clock times for each team's set of plays and the empirical distribution of time lapses between on-ball events.

```
# Source utils
source("./code/simulation_utils.R")
```

```
##
## 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
```

Models

In order to simulate plays, we require fits from the models of the latent MDP componets. Specifically, the player-specific parameters from the estimated shot policy model, state transition model, and reward model. These models are formally defined in Section 3 of the paper. We fit the models using Stan. The Stan model scripts are included in the folder `./code/stan_models` and the code to fit them are contained in files `./code/policy_fit.R`, `./code/transition_probability_fit.R`, and `./code/reward_fit.R` respectively. As these models take a considerable amount of time to fit, we have included 300 posterior draws from the player-specific parameters for each model.

```r
# Load posterior draws
n_draws = 300
lambda_MIA_draws = readRDS("./model_output/lambda_MIA_draws.rds")
lambda_BRK_draws = readRDS("./model_output/lambda_BRK_draws.rds")
mu_draws = readRDS("./model_output/mu_draws.rds")
theta_draws = readRDS("./model_output/theta_draws.rds")
xi_draws = readRDS("./model_output/xi_draws.rds")
```

Simulating plays

Before we can simulate plays, we require a few more in inputs. As noted in Section 4 of the paper, we require the starting states of all plays and the corresponding shot clock times at the start of each play. We also need the empirical distribution of time-lapses between events in order to take time off of the shot clock at each step of the MDP.

```r
# Get initial states and shot clock times for each team.
MIA_initial_states <- get_initial_states(dat, "MIA")
BRK_initial_states <- get_initial_states(dat, "BRK")

# Get empirical shot clock distribution
shot_clock_dist <- get_sc_dist(dat = dat, num_intervals = 3)
```

We can now simulate each team's plays in this game for a chosen number of simulations. We'll simulate the game 100 times.

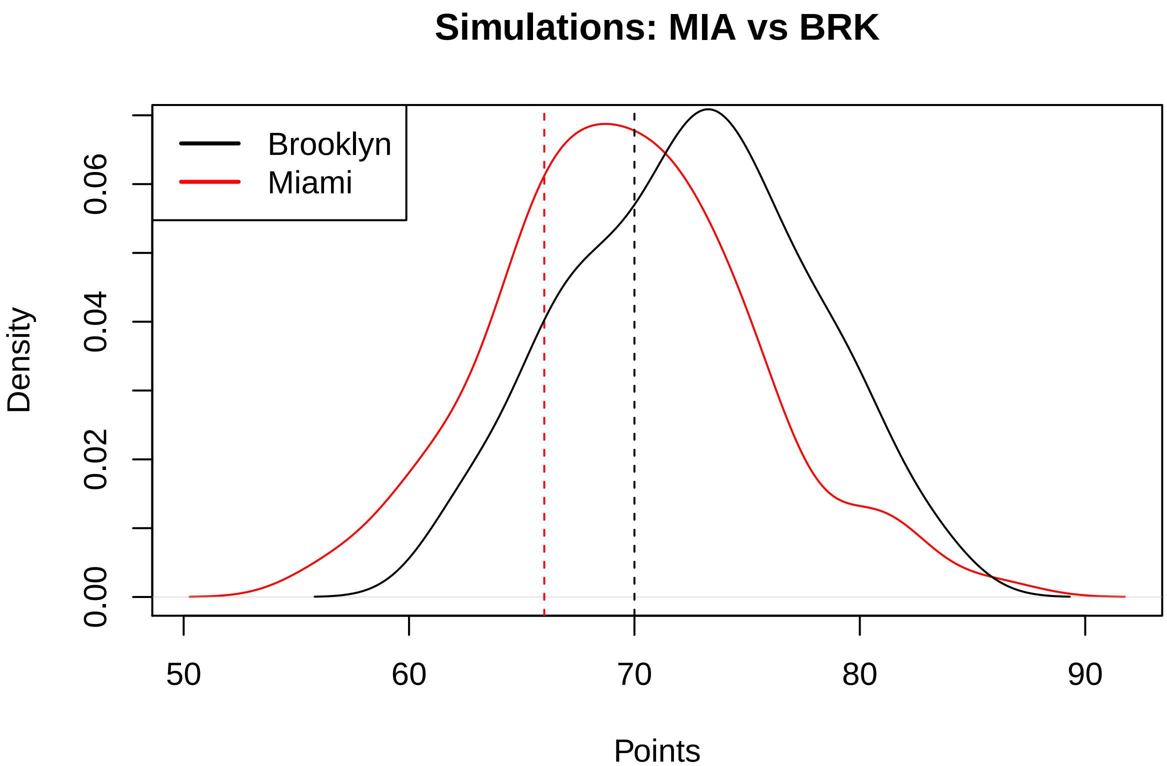
```r
n_sim = 100

# MIAMI SIMULATIONS
MIA_points = NA
for(iter in 1:n_sim){
  for(play in 1:nrow(MIA_initial_states)) {
    if (play == 1) {
      game_moments_MIA = algorithm_1(
        s_0 = MIA_initial_states[play, "state"],
        c_0 = MIA_initial_states[play, "shot_clock"],
        theta_draws = theta_draws,
```

```
      mu_draws = mu_draws,
      xi_draws = xi_draws,
      lambda_draws = lambda_MIA_draws,
      L_dist = shot_clock_dist,
      num_mcmc = n_draws
    )
  } else {
    game_moments_MIA = rbind(
      game_moments_MIA,
      algorithm_1(
        s_0 = MIA_initial_states[play, "state"],
        c_0 = MIA_initial_states[play, "shot_clock"],
        theta_draws = theta_draws,
        mu_draws = mu_draws,
        xi_draws = xi_draws,
        lambda_draws = lambda_MIA_draws,
        L_dist = shot_clock_dist,
        num_mcmc = n_draws
      )
    )
  }
}
MIA_points[iter] = sum(game_moments_MIA$reward)
}

# BROOKLYN SIMULATIONS
BRK_points = NA
for(iter in 1:n_sim){
  for(play in 1:nrow(BRK_initial_states)) {
    if (play == 1) {
      game_moments_BRK = algorithm_1(
        s_0 = BRK_initial_states[play, "state"],
        c_0 = BRK_initial_states[play, "shot_clock"],
        theta_draws = theta_draws,
        mu_draws = mu_draws,
        xi_draws = xi_draws,
        lambda_draws = lambda_BRK_draws,
        L_dist = shot_clock_dist,
        num_mcmc = n_draws
      )
    } else {
      game_moments_BRK = rbind(
        game_moments_BRK,
        algorithm_1(
          s_0 = BRK_initial_states[play, "state"],
          c_0 = BRK_initial_states[play, "shot_clock"],
          theta_draws = theta_draws,
          mu_draws = mu_draws,
          xi_draws = xi_draws,
          lambda_draws = lambda_BRK_draws,
          L_dist = shot_clock_dist,
          num_mcmc = n_draws
        )
```

```
      )
    }
  }
  BRK_points[iter] = sum(game_moments_BRK$reward)
}
```

We can plot density estimates of our simulations and compare these to the empirical total points from these plays in the data. Dotted vertical lines represent each team's observed total points from the filtered plays from this game.

```
# Compare simulations to empirical
plot(density(MIA_points), col = "red",
     main = "Simulations: MIA vs BRK",
     xlab = "Points")
dat %>% filter(team == "MIA") %>% with(abline(v = sum(points),
                                              col = "red",
                                              lty = 2))

lines(density(BRK_points))
dat %>% filter(team == "BRK") %>% with(abline(v = sum(points),
                                              lty = 2))
legend("topleft", c("Brooklyn", "Miami"),
       col = c("black", "red"),
       lwd = 2, bg = NA)
```

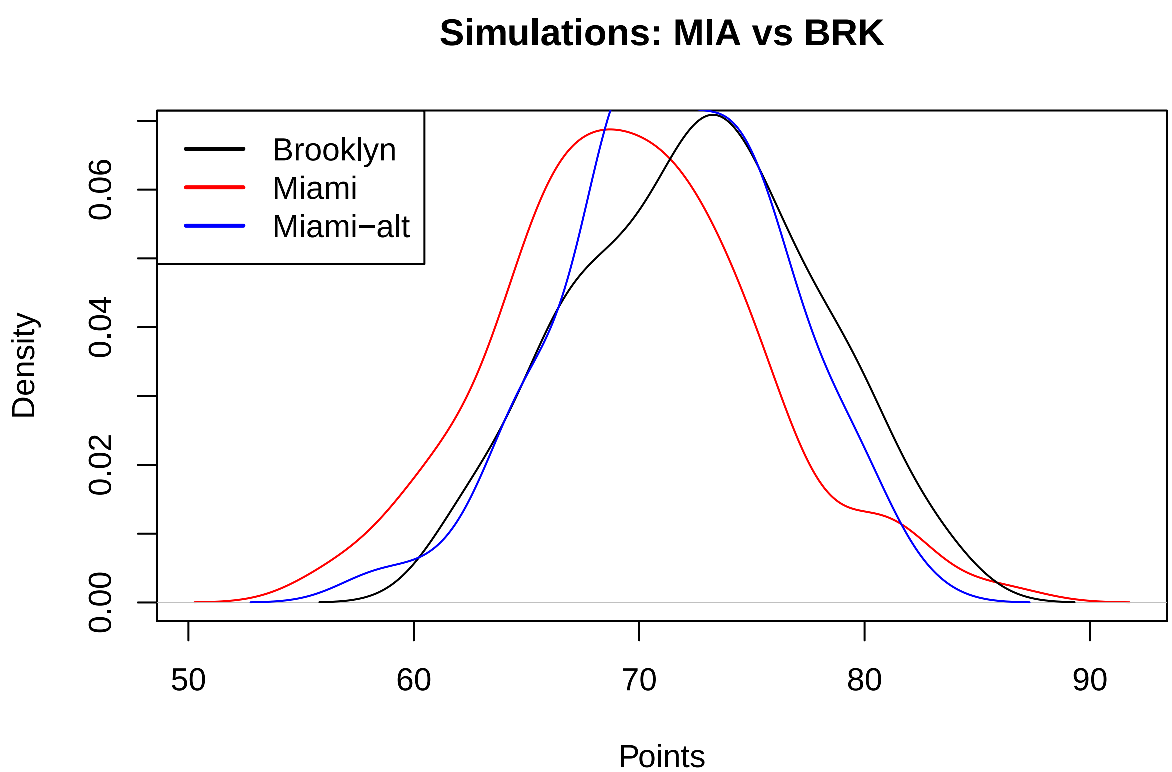


Altering Policies

We now will explore an alteration to Miami's shot policy. We will decrease each player's midrange shot policy by 20% (except late in shot clock) and increase each player's three point policy by 20% regardless of time on clock. The function `alter_theta` contained in the simulation utilities script alters the posterior draws of theta according to our desired changes.

```
# POLICY ALTERATION
# Decrease midrange shot policy by 20% (except late in shot clock) and
# increase three point policy by 20% (regardless of time on clock)

# Identify MIA players
MIA_players = dat %>%
  filter(team == "MIA") %>%
  distinct(entity) %>%
  pull(entity)

# Identify states to alter
# 1) ALL Midrange shots
to_alter_1 = c(paste(MIA_players, "long2_contested", sep = "_"),
             paste(MIA_players, "long2_open", sep = "_"))
# 2) ALL three point shots
to_alter_2 = c(paste(MIA_players, "three_contested", sep = "_"),
              paste(MIA_players, "three_open", sep = "_"))


policy_change <- list(list(who_where = to_alter_1,
                                        when = 2:3,
                                        how_much = .8),
                                   list(who_where = to_alter_2,
                                        when = 1:3,
                                        how_much = 1.2)
                                   )

# Alter the posterior draws of theta
altered_theta_draws = alter_theta(theta_draws,
                                  altered_policy_rules = policy_change)

# MIAMI ALTERED SIMULATIONS
MIA_points_alt = NA
for(iter in 1:n_sim){
  for(play in 1:nrow(MIA_initial_states)) {
    if (play == 1) {
      game_moments_MIA = algorithm_1(
        s_0 = MIA_initial_states[play, "state"],
        c_0 = MIA_initial_states[play, "shot_clock"],
        theta_draws = altered_theta_draws,
        mu_draws = mu_draws,
        xi_draws = xi_draws,
        lambda_draws = lambda_MIA_draws,
        L_dist = shot_clock_dist,
        num_mcmc = n_draws
      )
    } else {
      game_moments_MIA = rbind(
        game_moments_MIA,
        algorithm_1(
          s_0 = MIA_initial_states[play, "state"],
```

```
          c_0 = MIA_initial_states[play, "shot_clock"],
          theta_draws = altered_theta_draws,
          mu_draws = mu_draws,
          xi_draws = xi_draws,
          lambda_draws = lambda_MIA_draws,
          L_dist = shot_clock_dist,
          num_mcmc = n_draws
        )
      )
    }
  }
  MIA_points_alt[iter] = sum(game_moments_MIA$reward)
}

plot(density(MIA_points), col = "red",
     main = "Simulations: MIA vs BRK",
     xlab = "Points")
lines(density(BRK_points))
lines(density(MIA_points_alt), col = "blue")
legend("topleft", c("Brooklyn", "Miami", "Miami-alt"),
       col = c("black", "red", "blue"),
       lwd = 2)
```



Miami's projected distribution of possible scores increases.