Model 2: Poisson

Ben Moolman, Craig Orman, Ethan Pross

Data Prep and Cleaning

```
# Load and clean data
original_tbl <- read.csv("./NBA-BoxScores-2023-2024.csv") |>
  mutate(
    START_POSITION = na_if(START_POSITION, "") |> factor(),
    COMMENT = na_if(COMMENT, "") |> factor(),
    MIN = na_if(MIN, ""),
    MIN = str_replace(MIN, "([0-9]+)\\.[0-9]+:", "\\1:")
  )

# Filter to starters only
starting_dat <- original_tbl |>
  filter(!is.na(START_POSITION))

# Calculate team points per game
team_points <- original_tbl |>
  filter(!is.na(PTS)) |>
  group_by(GAME_ID, TEAM_ID) |>
  summarize(TeamPoints = sum(PTS), .groups = "drop")

# Join with itself to get opponent points
team_vs_opponent <- team_points |>
  inner_join(team_points, by = "GAME_ID", suffix = c("", ".opp")) |>
  filter(TEAM_ID != TEAM_ID.opp) |>
  rename(OPP_TEAM_ID = TEAM_ID.opp, OpponentPoints = TeamPoints.opp)

# Compute average opponent points allowed per team (DRTG)
team_drtg <- team_vs_opponent |>
  group_by(TEAM_ID) |>
  summarize(DRTG_proxy = mean(OpponentPoints), n_games = n(), .groups = "drop")

# Build opponent_map from distinct team-game pairs
game_team_pairs <- original_tbl |>
  select(GAME_ID, TEAM_ID) |>
  distinct()

# Create mapping of TEAM_ID and OPP_TEAM_ID for each game
opponent_map <- game_team_pairs |>
  inner_join(game_team_pairs, by = "GAME_ID") |>
  filter(TEAM_ID.x != TEAM_ID.y) |>
  rename(TEAM_ID = TEAM_ID.x, OPP_TEAM_ID = TEAM_ID.y)
```

```r
# Join with defensive ratings (DRTG)
opponent_map <- opponent_map |>
  left_join(team_drtg |> rename(OPP_TEAM_ID = TEAM_ID, OPP_DRTG = DRTG_proxy), by = "OPP_TEAM_ID")


# Merge opponent info into starting dataset and center DRTG
mean_drtg <- mean(team_drtg$DRTG_proxy)

starting_dat <- starting_dat |>
  left_join(opponent_map, by = c("GAME_ID", "TEAM_ID")) |>
  mutate(centered_OPP_DRTG = OPP_DRTG - mean_drtg)
```

Model 2 implementation

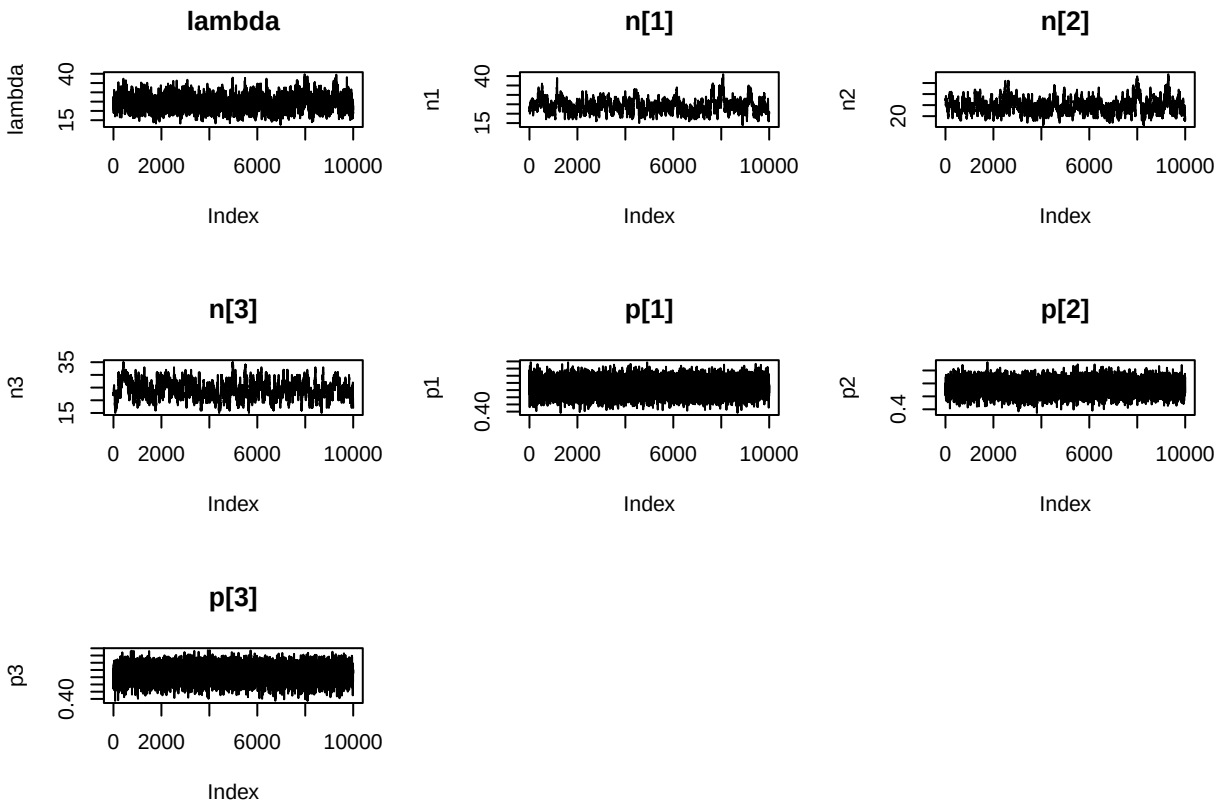
```r
### MODEL 2 ###
log_n_con = function(p, lambda, n, y) {
  if (all(is.nan(log( ((1-p)*lambda)^sum(n) ) - sum(log((factorial(n-y)))) ))){
    return(rep(-Inf,length(y)))
  }else{
    log( ((1-p)*lambda)^sum(n) ) - sum(log((factorial(n-y))))
  }
}
mcmc_model_2 = function(data, player_id, opp_team_id, n_iter=5000, init_lambda = c(), init_n = c(), gamm
  # Gather true data
  player_dat = data[data$PLAYER_ID == player_id, ]
  player_dat = player_dat[player_dat$OPP_TEAM_ID == opp_team_id, ]
  y = player_dat$FGM
  true_n = player_dat$FGA
  def_factor<- exp(gamma*(data$centered_OPP_DRTG[1]))
  if(length(init_lambda) ==0) {
    init_lambda = mean(player_dat$FGA)
  }
  if(length(init_n) ==0) {
    init_n = mean(player_dat$FGA)
  }


  big_N<- length(y)
  lambda<- init_lambda
  n<- rep(init_n,big_N)

  # setting up lists/matrices for returning
  p_matrix<- matrix(NA, nrow=n_iter, ncol=big_N)
  lambda_list<- rep(lambda, n_iter)
  n_matrix<- matrix(NA, nrow=n_iter, ncol=big_N)

  for (i in 1:n_iter) {
    # sample p
    p_unscaled<- rbeta(big_N, 5 + sum(y), 5 + sum(true_n-y))
    p<- p_unscaled*def_factor
```

```
    # sample lambda
    lambda<- rgamma(1,shape=sum(n)-1/2,rate=big_N)

    # sample n
    n_prop<- rnorm(big_N, n, 1) # the third 1 is a tuning parameter
    logr<- log_n_con(p, lambda, n_prop, y)-log_n_con(p, lambda, n, y)
    for (j in 1:length(logr)) {
      if (is.finite(logr[j]) && log(runif(1))<logr[j]) {
        n[j]<- n_prop[j]
      }
    }
    # save values
    p_matrix[i,] = p
    n_matrix[i,] = n
    lambda_list[i] = lambda
  }
  return(data.frame(iteration=1:n_iter,
                    parameter=rep(c(paste("n[",1:big_N,"]", sep=""), "lambda", paste("p[",1:big_N,"]",
                    value=c(as.numeric(n_matrix),lambda_list, as.numeric(p_matrix))))
}
# running mcmc
n_iter<- 10000
MCMC_model_2 = mcmc_model_2(data = starting_dat, player_id = 2544, #LeBron
                            opp_team_id = 1610612744, #GSW
                            n_iter=10000,
                            gamma=0.01) #Previously found to be good value
```

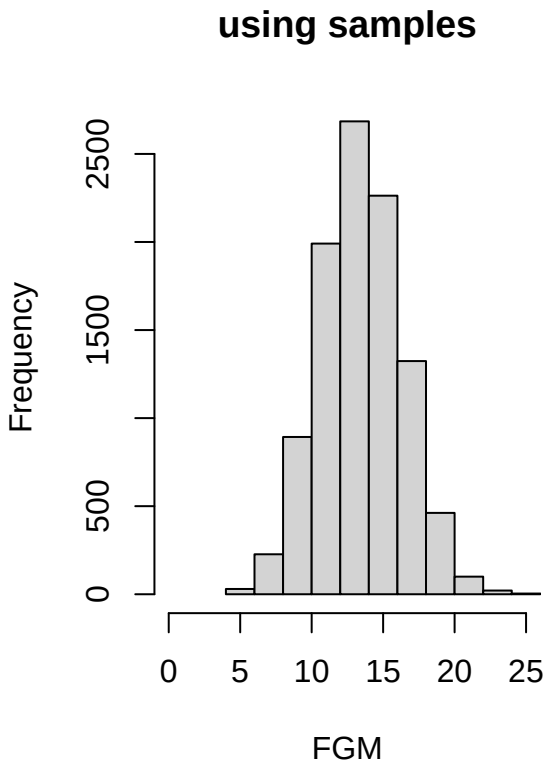
Model Diagnostics

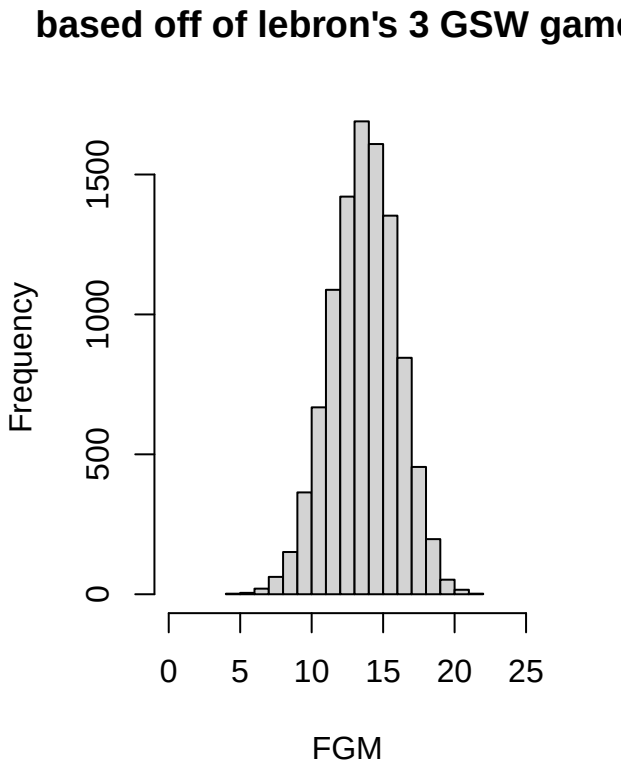
```
player_dat = starting_dat[starting_dat$PLAYER_ID == 2544, ]
player_dat = player_dat[player_dat$OPP_TEAM_ID == 1610612744, ]

# setup / data
lebron_GSW_n<- player_dat$FGA
lebron_mean_n<- mean(lebron_GSW_n)
lebron_median_n<- median(lebron_GSW_n)
lebron_max_n<- max(lebron_GSW_n)
lebron_GSW_p<- player_dat$FG_PCT
lebron_GSW_y<- player_dat$FGM



### Traceplots
lambda<- MCMC_model_2$value[which(MCMC_model_2$parameter=="lambda")]
n1<- round(MCMC_model_2$value[which(MCMC_model_2$parameter=="n[1]")])
n2<- round(MCMC_model_2$value[which(MCMC_model_2$parameter=="n[2]")])
n3<- round(MCMC_model_2$value[which(MCMC_model_2$parameter=="n[3]")])
p1<- MCMC_model_2$value[which(MCMC_model_2$parameter=="p[1]")]
p2<- MCMC_model_2$value[which(MCMC_model_2$parameter=="p[2]")]
p3<- MCMC_model_2$value[which(MCMC_model_2$parameter=="p[3]")]
```

```r
par(mfrow=c(3,3))
plot(lambda,type="l",main="lambda")
plot(n1,type="l",main="n[1]")
plot(n2,type="l",main="n[2]")
plot(n3,type="l",main="n[3]")
plot(p1,type="l",main="p[1]")
plot(p2,type="l",main="p[2]")
plot(p3,type="l",main="p[3]")


### I don't think this is actually the Predictive Posterior
posterior_mean_ps<- apply(data.frame(p1,p2,p3),1,mean)
posterior_mean_ns<- round(apply(data.frame(n1,n2,n3),1,mean))
# making the y's using our posterior sample
y_model_2<- rbinom(n_iter,posterior_mean_ns,posterior_mean_ps)
par(mfrow=c(1,2))
```




```r
#hist of distribution based off our samples
hist(y_model_2,xlim=c(0,25),main="using samples", xlab="FGM")
#compare to a hist of lebron's games against GSW using observed values
hist(rbinom(10000,round(mean(lebron_GSW_n)),mean(lebron_GSW_p)),xlim=c(0,25), main="based off of lebron
```


using samples
Frequency
0
500
1500
2500
0
5
10
15
20
25
FGM


based off of lebron's 3 GSW gam
Frequency
0
500
1000
1500
0
5
10
15
20
25
FGM