Model 3 Neg Binom

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 3 implementation

$$
\begin{aligned}
y_{ikj} &\sim Binom(n_{ikj}, p_{ik}) \\
p_{ik} &\sim \phi \times Beta(5,5) \\
n_{ijk} &\sim NegBinom(r, \theta) \\
p(r,\theta) &\propto \sqrt{\frac{r_i}{\theta^2}(1-\theta_i)} \qquad \text{Jeffreys prior}
\end{aligned}
$$

Next we write out the full conditionals

$$
\begin{aligned}
y_{ikj}|... &\sim Bin(n_{ikj}, p_{ik}) \\
p_{ik}|... &\propto \phi p(y|p_{ik}, n_{ikj}) p(p_{ik}|\alpha,\beta) \\
&\propto \min(1, Beta(\sum(y)+\alpha, N-\sum(y)+\beta \\
n_{ikj}|... &\propto p(y_{ikj}|p_{ik}, n_{ikj}) p(n_{ikj}|r,\theta) \\
&\propto \ln\left(\prod_{j,k}(n_{ijk}+r_i-1)!\right) - \ln\left(\prod_{j,k}(n_{ijk}-y_{ijk})!\right) + \ln\left(\prod_{j,k}[(1-\theta_i)(1-p_{ik})]^{n_{ijk}}\right) \\
p(r_i|...) &\propto p(n_ijk|r_i,\theta_i) p(r_i,\theta_i) \\
&\propto \ln(\prod_{j,k}[(n_{ijk}+r_i-1)!] - \ln(\prod_{j,k}[(r_i-1)!]) + \ln(\prod_{j,k}[(\theta_i^{r_i})!]) + \ln(r_i^{\frac{1}{2}}) \\
p(\theta_i) &\propto p(n_ijk|r_i,\theta_i) p(r_i,\theta_i) \\
&\sim Beta(\sum_{j,k} n_{ijk} + \frac{1}{2}, r_i - 1)
\end{aligned}
$$

Bayes factor stuff (marginal likelihood)

$$
p(y \mid M_3) = \frac{\Gamma(a_i)\Gamma(b_i)}{\Gamma(a_i+b_i)} \sum_{n_i} \prod_j \binom{n_i}{y_{ijk}} \times \frac{\Gamma\left(\sum_j y_{ijk} + a_i - 1\right)\Gamma\left(\sum_j (n_i - y_{ijk}) + b_i - 1\right)}{\Gamma\left(\sum_j n_i + a_i + b_i - 2\right)} \times \frac{\sqrt{\pi 2^{1-n_{ijk}}}}{5} \left(\frac{3 - 2n_{ijk}\Gamma(n_{ijk}+1)}{\Gamma(n_{ijk}-1/2)}\right.
$$

So for this, we kinda have a full thingy set up?

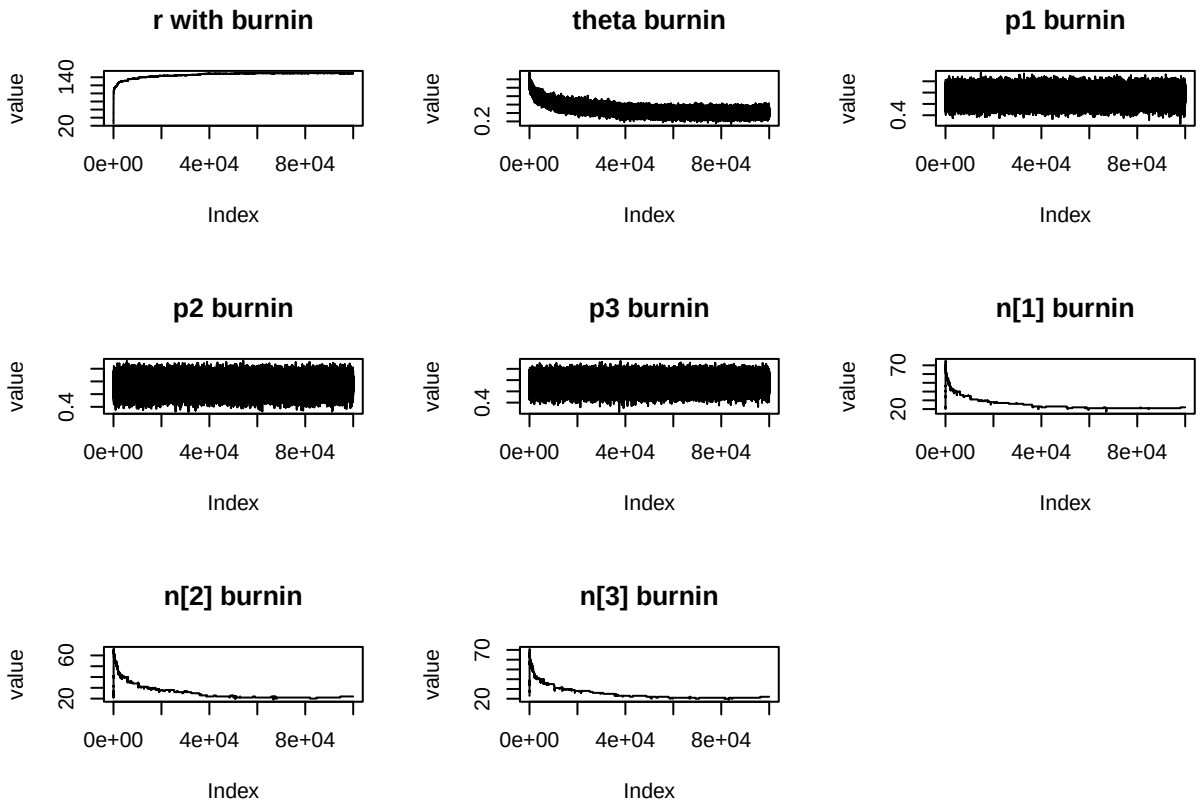
```r
### MODEL 3 ###
log_n_con = function(n, r, theta, y,p) {
  dummy<- sum(log((factorial(n+r-1)))) - sum(log((factorial(n-y)))) + log((1-theta)*((1-p)^sum(n)))
```

```
  if (all(is.nan(dummy))){
    return(rep(-Inf,length(n)))
  }else{
    return(dummy)
  }
}
log_r_con = function(n, r, theta, y) {
  dummy<- sum(log((factorial(n+r-1)))) - sum(log((factorial(r-1)))) + log(theta^sum(r)) + log(r^(length
  if (all(is.nan(dummy))){
    return(rep(-Inf,length(r)))
  }else{
    return(dummy)
  }
}
mcmc_model_3 = function(data, player_id, opp_team_id, gamma = 0.01,
                        n_iter=5000, init_r, init_theta, init_n, init_p,
                        prop_r_sd = 3.5, prop_n_sd = 3.5) {
  # 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]))

  # Start code
  big_N<- length(y)
  r<- init_r
  theta<- init_theta
  n<- rep(init_n,big_N)
  p<- init_p

  # setting up lists/matrices for returning
  r_list<- rep(NA, n_iter)
  theta_list<- rep(NA, n_iter)
  p_matrix<- matrix(NA, nrow=n_iter, ncol=big_N)
  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 theta
    theta<- rbeta(1,sum(n)+1/2,r-1)

    # sample r
    r_prop<- rnorm(1, r, prop_r_sd) # the third 1 is a tuning parameter
    logr<- log_r_con(n, r_prop, theta, y)-log_r_con(n, r, theta, y)
    if (is.finite(logr)) {
      if (log(runif(1))<logr) {
        r<- r_prop
      }
    }
```

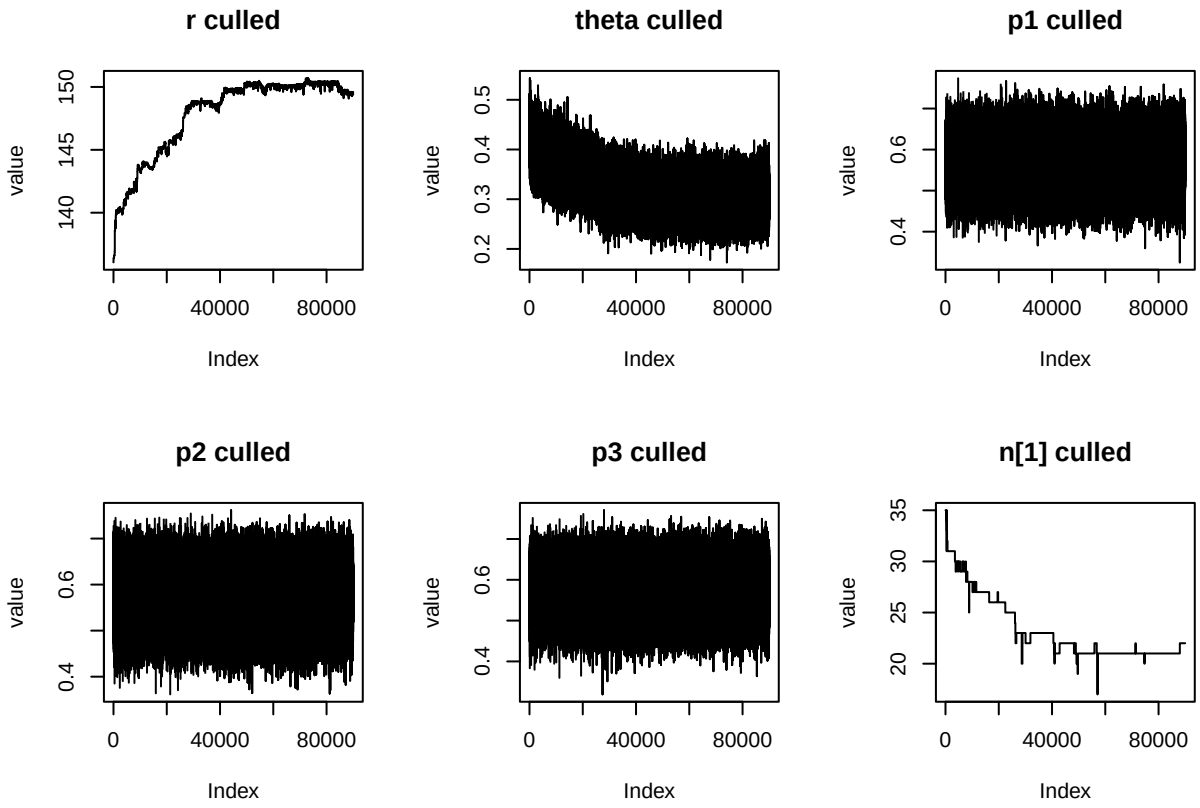
```
    # sample n
    n_prop<- rnorm(big_N, n, prop_n_sd) # the third 1 is a tuning parameter
    logr<- log_n_con(n_prop, r, theta, y,p)-log_n_con(n, r, theta, y,p)
    for (j in 1:length(logr)) {
      if (is.finite(logr[j])) {
        if (log(runif(1))<logr[j]) {
          n[j]<- n_prop[j]
        }
      }
    }

    # save values
    r_list[i] = r
    theta_list[i] = theta
    p_matrix[i,] = p
    n_matrix[i,] = n
  }
  # return(data.frame(iteration=1:n_iter,
  #                   r = r_list,
  #                   theta = theta_list,
  #                   p1 = p_matrix[,1],
  #                   p2 = p_matrix[,2],
  #                   p3 = p_matrix[,3],
  #                   n1 = n_matrix[,1],
  #                   n2 = n_matrix[,2],
  #                   n3 = n_matrix[,3]))
  return(data.frame(iteration=1:n_iter,
                    parameter=rep(c("r","theta",paste("p[",1:big_N,"]", sep=""),paste("n[",1:big_N,"]",
                    value=c(r_list,theta_list,as.numeric(p_matrix),as.numeric(n_matrix))))
}
# running mcmc
lebron_GSW_n<- c(25,23,22)
player_id = 2544
opp_team_id = 1610612744
data = starting_dat
lebron_mean_n<- 18
lebron_median_n<- 18
lebron_max_n<- 27
lebron_GSW_p<- c(0.560,0.652,0.636)
lebron_GSW_y<- c(14,15,14)
GSW_DRTG<- 115.1585
mean_DRTG<- 114.2114
gamma<-0.01
factor_gsw<- exp(gamma*(GSW_DRTG-mean_DRTG))
n_iter<- 100000
init_r<- max(lebron_GSW_n) # maybe tune?
init_theta<- mean(lebron_GSW_p) # maybe tune ?
init_p<- mean(lebron_GSW_p) # maybe tune ?
init_n<- round(mean(lebron_GSW_n)) #maybe tune ?


MCMC_model_3 = suppressWarnings(mcmc_model_3(starting_dat, player_id = 2544, opp_team_id = 1610612744,
                           n_iter=n_iter, init_r,init_theta,init_n,init_p,
```

```
                                prop_r_sd=3.7, prop_n_sd = 3.7))
# MCMC_model_3
```

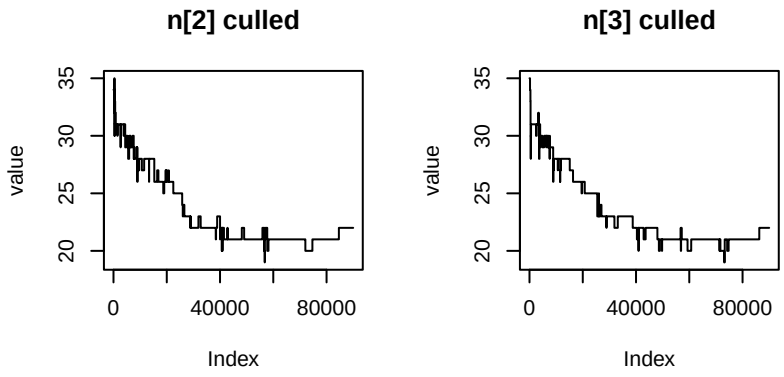
```
### Traceplots

# plots of the raw samples without removing burnin
par(mfrow=c(3,3))
plot(MCMC_model_3$value[which(MCMC_model_3$parameter=="r")],type="l",main="r with burnin",ylab="value")
plot(MCMC_model_3$value[which(MCMC_model_3$parameter=="theta")],type="l",main="theta burnin",ylab="valu
plot(MCMC_model_3$value[which(MCMC_model_3$parameter=="p[1]")],type="l",main="p1 burnin",ylab="value")
plot(MCMC_model_3$value[which(MCMC_model_3$parameter=="p[2]")],type="l",main="p2 burnin",ylab="value")
plot(MCMC_model_3$value[which(MCMC_model_3$parameter=="p[3]")],type="l",main="p3 burnin",ylab="value")
plot(round(MCMC_model_3$value[which(MCMC_model_3$parameter=="n[1]")]),type="l",main="n[1] burnin",ylab=
plot(round(MCMC_model_3$value[which(MCMC_model_3$parameter=="n[2]")]),type="l",main="n[2] burnin",ylab=
plot(round(MCMC_model_3$value[which(MCMC_model_3$parameter=="n[3]")]),type="l",main="n[3] burnin",ylab=

#burnin removal (I know I probably made this more complicated than necessary)
burnin<-10000
r_culled<- MCMC_model_3$value[which(MCMC_model_3$parameter=="r")][burnin:n_iter]
theta_culled<- MCMC_model_3$value[which(MCMC_model_3$parameter=="theta")][burnin:n_iter]
p1_culled<- MCMC_model_3$value[which(MCMC_model_3$parameter=="p[1]")][burnin:n_iter]
p2_culled<- MCMC_model_3$value[which(MCMC_model_3$parameter=="p[2]")][burnin:n_iter]
p3_culled<- MCMC_model_3$value[which(MCMC_model_3$parameter=="p[3]")][burnin:n_iter]
n1_culled<- MCMC_model_3$value[which(MCMC_model_3$parameter=="n[1]")][burnin:n_iter]
n2_culled<- MCMC_model_3$value[which(MCMC_model_3$parameter=="n[2]")][burnin:n_iter]
n3_culled<- MCMC_model_3$value[which(MCMC_model_3$parameter=="n[3]")][burnin:n_iter]
iter_index<- burnin:n_iter
parameter_index<- c("r","theta","p[1]","p[2]","p[3]","n[1]","n[2]","n[3]")
MCMC_model_3_culled<- data.frame(iteration = rep(iter_index,length(parameter_index)),
                                 parameter = rep(parameter_index,each=(n_iter-burnin+1)),
                                 value = c(r_culled,theta_culled,p1_culled,p2_culled,p3_culled,n1_culled


# plots of the samples with burnin removed
par(mfrow=c(2,3))
```



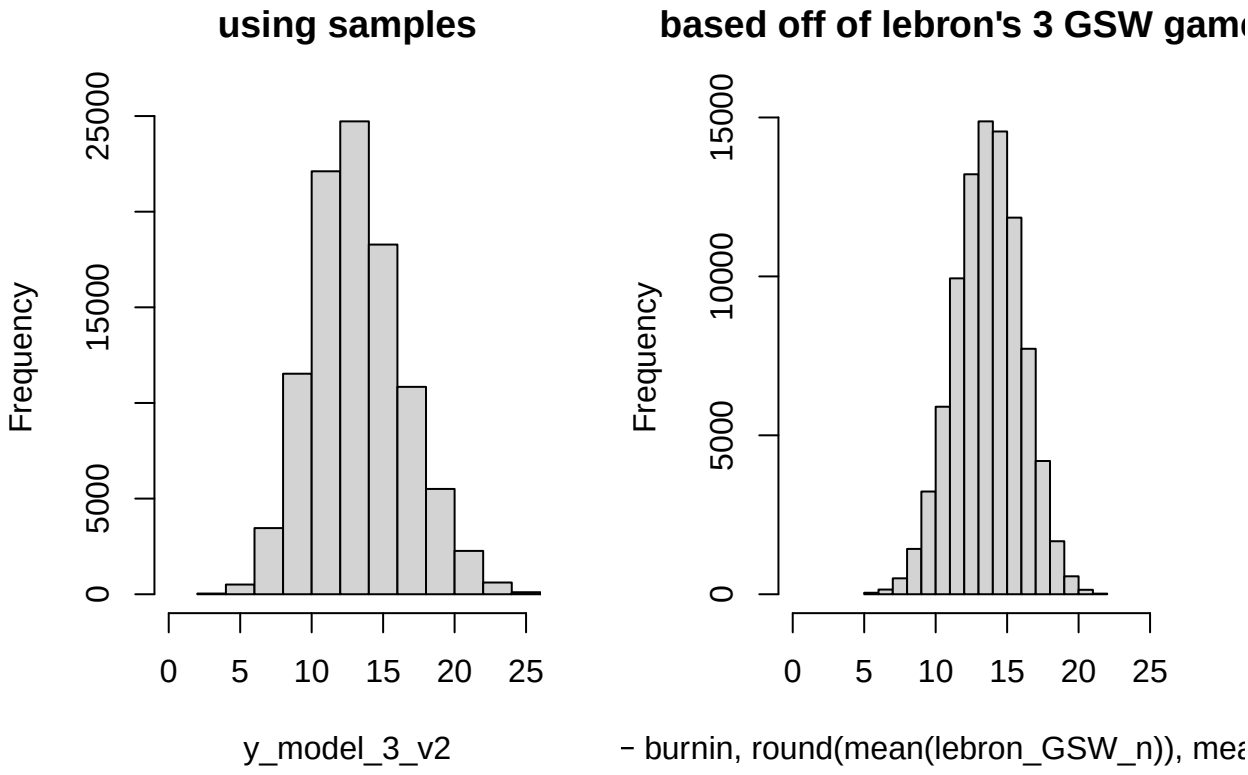

```
plot(MCMC_model_3_culled$value[which(MCMC_model_3_culled$parameter=="r")],type="l",main="r culled",ylab=
plot(MCMC_model_3_culled$value[which(MCMC_model_3_culled$parameter=="theta")],type="l",main="theta culle
plot(MCMC_model_3_culled$value[which(MCMC_model_3_culled$parameter=="p[1]")],type="l",main="p1 culled",y
plot(MCMC_model_3_culled$value[which(MCMC_model_3_culled$parameter=="p[2]")],type="l",main="p2 culled",y
plot(MCMC_model_3_culled$value[which(MCMC_model_3_culled$parameter=="p[3]")],type="l",main="p3 culled",y
plot(round(MCMC_model_3_culled$value[which(MCMC_model_3_culled$parameter=="n[1]")]),type="l",main="n[1]
```




```
plot(round(MCMC_model_3_culled$value[which(MCMC_model_3_culled$parameter=="n[2]")]),type="l",main="n[2]
plot(round(MCMC_model_3_culled$value[which(MCMC_model_3_culled$parameter=="n[3]")]),type="l",main="n[3]


### I don't think this is actually the Predictive Posterior

# getting the 10000 p's for sampling
posterior_mean_ps<- apply(data.frame(p1_culled,p2_culled,p3_culled),1,mean)
# getting the 10000 n's for sampling
posterior_mean_ns<- round(apply(data.frame(n1_culled,n2_culled,n3_culled),1,mean))
# making the y's using our posterior sample
y_model_3_v2<- rbinom(n_iter,posterior_mean_ns,posterior_mean_ps)
par(mfrow=c(1,2))
```




```
#hist of distribution based off our samples
hist(y_model_3_v2,xlim=c(0,25),main="using samples")
# abline(v=mean(lebron_GSW_y), col='red', lwd=2, "Sampled FGM")
#compare to a hist of lebron's games against GSW using observed values
hist(rbinom(n_iter-burnin,round(mean(lebron_GSW_n)),mean(lebron_GSW_p)),xlim=c(0,25), main="based off o
```




```
# abline(v=mean(lebron_GSW_y), col='red', lwd=2, xlab="FGM")
```