Model 3 Neg Binom

Ben Moolman, Craig Orman, Ethan Pross

Beginning Data preparing

```
## Ben's DRTG code!
# Calculate total points per team per game
# here, datatest2 is the entire data frame that is not filtered for starters
# filtering dataset to remove NAs which arise a player doesnt record any minutes in the game (sitting o
team_points <- na.omit(original_tbl) %>%
  group_by(GAME_ID, TEAM_ID) %>%
  summarize(TeamPoints = sum(PTS), .groups = "drop")

#
team_points_opponent <- team_points %>%
  rename(OPP_TEAM_ID = TEAM_ID, OpponentPoints = TeamPoints)

# join and filter
team_vs_opponent <- team_points %>%
  inner_join(team_points_opponent, by = "GAME_ID") %>%
  filter(TEAM_ID != OPP_TEAM_ID)

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

range(team_drtg$DRTG_proxy)
```

```
## [1] 106.5244 123.0366
```

```
mean(team_drtg$DRTG_proxy)
```

```
## [1] 114.2114
```

DRTG data being joined to the starting data

```
## NOW ADDING DRTG vars to original_tbl


# Each team plays one opponent per game, so we pair them like this:
game_team_pairs <- original_tbl %>%
  select(GAME_ID, TEAM_ID) %>%
  distinct()
```

```r
# Join to get each game twice: once for each team, with their opponent
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)

# Add opponent DRTG to the mapping
opponent_map <- opponent_map %>%
  left_join(team_drtg %>% rename(OPP_TEAM_ID = TEAM_ID, OPP_DRTG = DRTG_proxy),
            by = "OPP_TEAM_ID")

# Join to add opponent DRTG column to the full data
starting_dat <- starting_dat %>%
  left_join(opponent_map, by = c("GAME_ID", "TEAM_ID"))

mean_drtg <- mean(team_drtg$DRTG_proxy)
starting_dat <- starting_dat %>%
  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

$$
\begin{aligned}
y_{ikj}|... &\sim Bin(n_{ikj}, p_{ik}) \\
p_{ik}|... &\propto \phi p(y|p_{ik}, n_{ikj}) p(p_{ik}|a,b) \\
&\propto \min(1, Beta(\sum(y)+5, N-\sum(y)+5 \\
n_{ikj}|... &\propto p(y_{ikj}|p_{ik}, n_{ikj}) p(n_{ikj}|r,\theta) \\
&\propto \ln\left(\prod_{j,k}(n_{ijk}+r_i-1)!\right) - \ln\left(\prod_{j,k}(n_{ijk}-y_{ijk})!\right) + \ln\left(\prod_{j,k}[(1-\theta_i)(1-p_{ik})]^{n_{ijk}}\right) \\
p(r_i|...) &\propto p(n_ijk|r_i,\theta_i) p(r_i,\theta_i) \\
&\propto \ln(\prod_{j,k}[(n_{ijk}+r_i-1)!] - \ln(\prod_{j,k}[(r_i-1)!]) + \ln(\prod_{j,k}[(\theta_i^{r_i})!]) + \ln(r_i^{\frac{1}{2}}) \\
p(\theta_i) &\propto p(n_ijk|r_i,\theta_i) p(r_i,\theta_i) \\
&\sim Beta(\sum_{j,k} n_{ijk} + \frac{1}{2}, r_i - 1)
\end{aligned}
$$

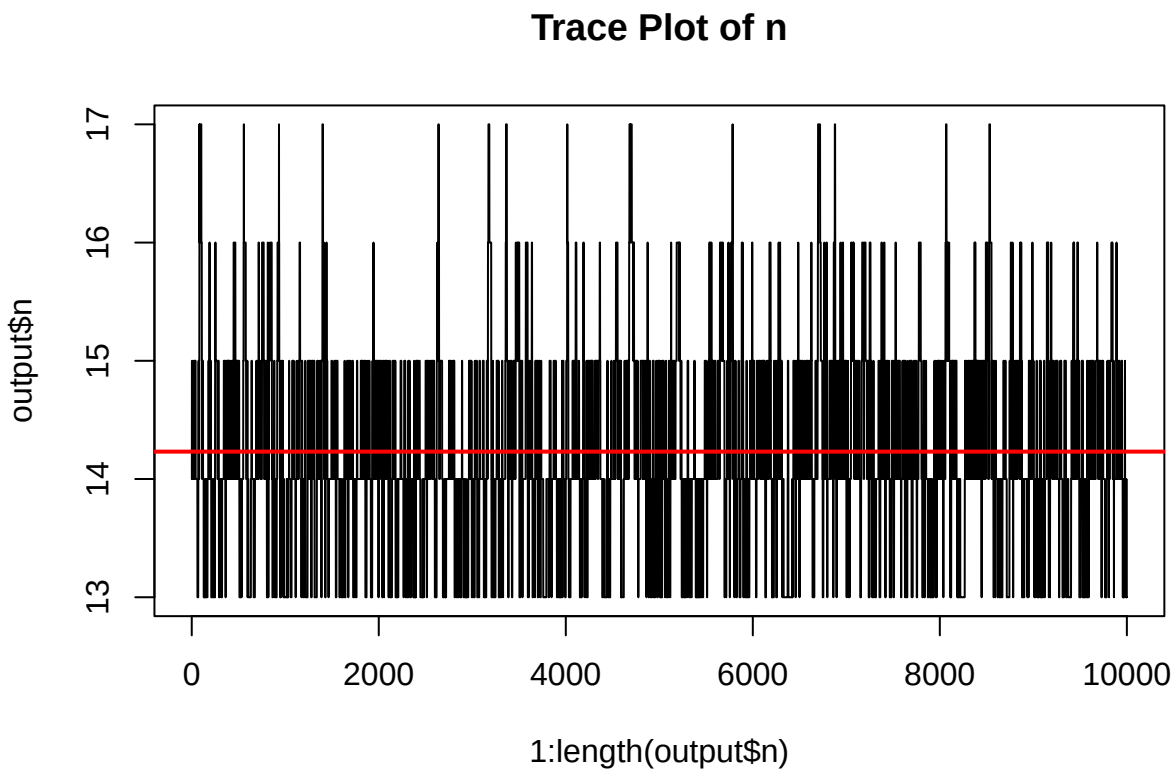
So for this, we kinda have a full thingy set up?

```r
# Data and prior
# 1610612746 is steph currys team maybe
# 2544 is Lebron's player id
```

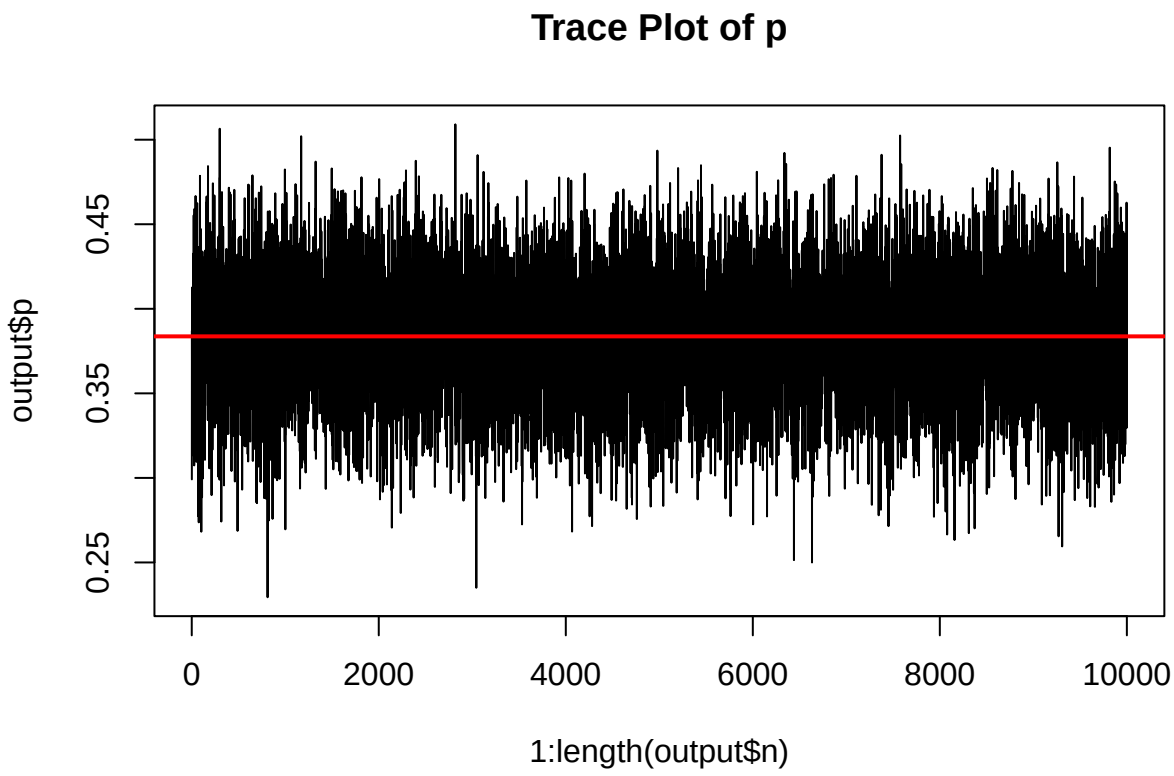
```
log_n_func = function(n, r, y, theta, p){
  sum(log(factorial(n+r-1)))- sum(log(factorial(n-y))) + sum(n * log((1-theta)*(1-p)))
}

log_r_func = function(r, n, y, theta, p) {
  sum(log(factorial(n+r-1)))-sum(log(factorial(r-1)))+sum(r*log(theta))+log(r)/2
}

#### RESETING COMMANDS START

data = starting_dat
inits=list(p = 0.5,
           n = 20,
           r = 20,
           theta = 0.3)
prop_sd = list(n = 4, r = 3, theta = 5)
psi = 0.01
player_id = 2544
opp_team_id = 1610612746
n_iter = 100

### RESETTING COMMANDS END

gibby = function(data, inits, prop_sd, psi, player_id, opp_team_id, n_iter=5000, n_burnin = 1) {
  # Gather true data
  player_dat = data[data$PLAYER_ID == player_id, ]
  player_dat = player_dat[player_dat$Opposing_Team_Name_ID == opp_team_id, ]
  true_y = player_dat$FGM
  true_n = player_dat$FGA
  DRTG_var = data$centered_OPP_DRTG[1]

  # initial value setup
  p = inits$p
  n = inits$n
  r = inits$r
  theta = inits$theta

  # Save structure (REVIEW)
  n_keep = numeric(n_iter)
  p_keep = numeric(n_iter)
  r_keep = numeric(n_iter)
  theta_keep = numeric(n_iter)

  # Proposal variances (TODO)
  prop_n_sd = prop_sd$n
  prop_r_sd = prop_sd$r

  # Gibbs sampler main
  for (i in 1:n_iter) {
    # Generate p
    pi = rbeta(1,5+sum(true_y), 5+sum(true_n - true_y))
    p = min(1, pi * exp(psi * DRTG_var))
```

```
    # Generate n
    prop_n = round(rnorm(1, mean=n, sd = prop_n_sd)) # round to ensure its integer, using r as thats wh
    logn = log_n_func(n=prop_n, r, true_y, theta, p) - log_n_func(n=n, r, true_y, theta, p)
    if (is.finite(logn) && log(runif(1)) < logn) {
      n <- prop_n
    }

    # generate r
    prop_r = round(rnorm(1, mean=r, sd = prop_r_sd)) # round to ensure its integer, using r as thats wh
    logr = log_r_func(prop_r, n, true_y, theta, p) - log_r_func(r, n, true_y, theta, p)
    if (is.finite(logr) && log(runif(1)) < logr) {
      r <- prop_r
    }
    # generate theta
    theta = rbeta(1, (sum(true_n)+ (1/2)), (r - 1))

    # Save values

    n_keep[i] =n
    p_keep[i] = p
    r_keep[i] = r
    theta_keep[i] = theta
  }
  output = list(n = n_keep[n_burnin:n_iter], p = p_keep[n_burnin:n_iter],
                r = r_keep[n_burnin:n_iter], theta = theta_keep[n_burnin:n_iter])
  return(output)
}

#function(data, inits, prop_sd, psi, player_id, opp_team_id, n_iter=5000)
inits=list(p = 0.5,
           n = 20,
           r = 20,
           theta = 0.5)
prop_sd = list(n = 0.5, r = 2)
n_iter = 11000
n_burnin = 1000
output = suppressWarnings(gibby(starting_dat, inits, prop_sd = prop_sd,
                                psi=0.1, player_id = 2544, opp_team_id = 1610612746,
                                n_iter=n_iter, n_burnin = n_burnin))
```

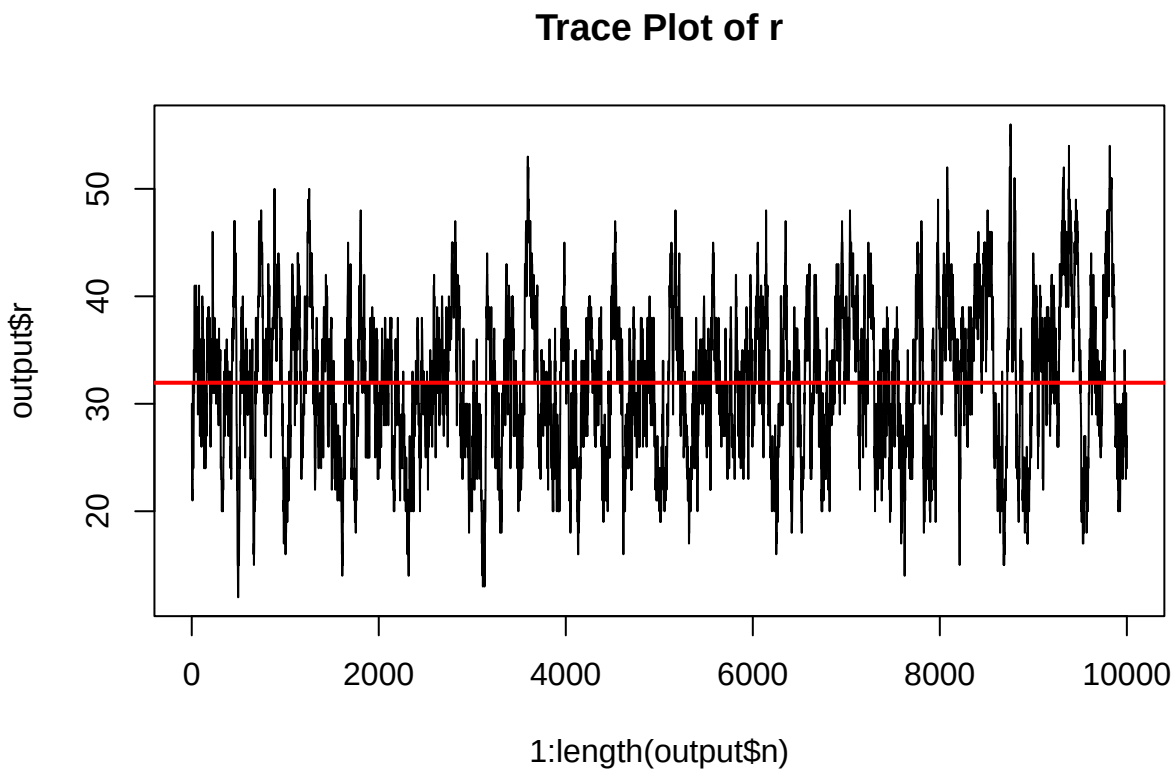
```
# Diagnostics
plot(y=output$n, x = 1:length(output$n), type = 'l', main="Trace Plot of n")
# plot(as.mcmc(output$n), main="n")
abline(h = mean(output$n), col='red', lwd=2)
```



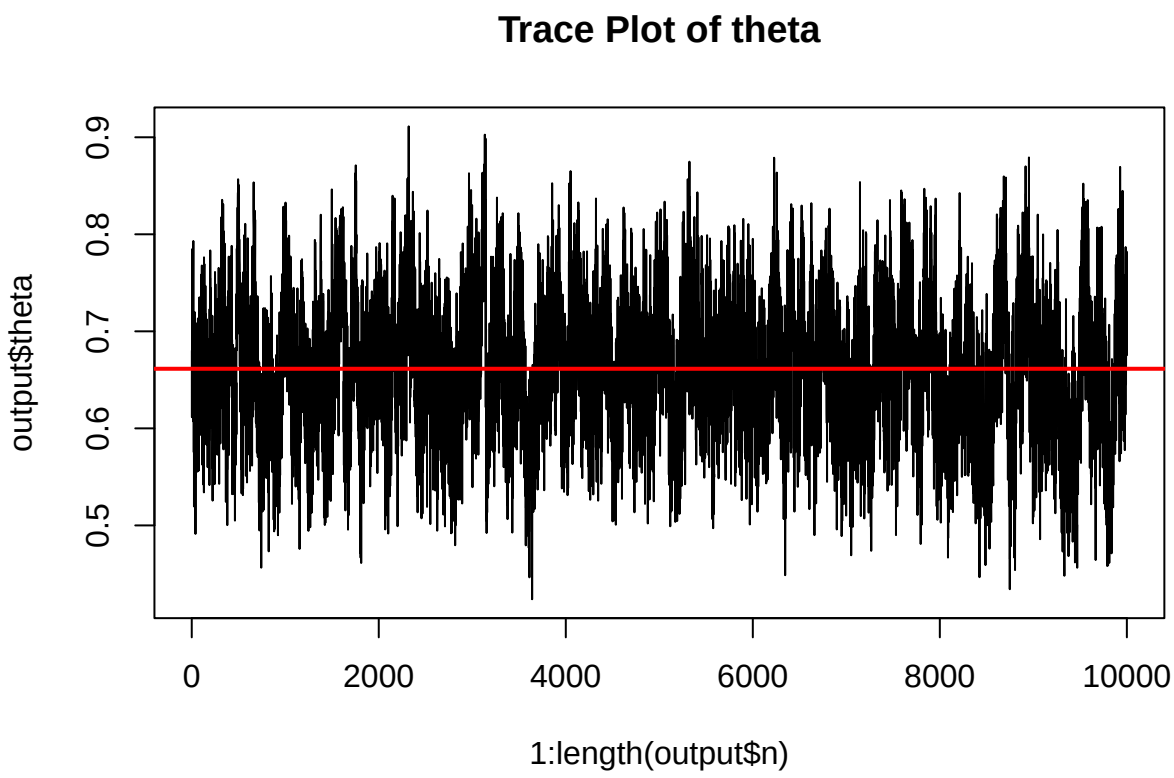

```
plot(y=output$p, x = 1:length(output$n), type = 'l', main="Trace Plot of p")
abline(h = mean(output$p), col='red', lwd=2)
```




```
plot(y=output$r, x = 1:length(output$n), type = 'l', main="Trace Plot of r")
abline(h = mean(output$r), col='red', lwd=2)
```




```r
plot(y=output$theta, x = 1:length(output$n), type = 'l', main="Trace Plot of theta")
abline(h = mean(output$theta), col='red', lwd=2)
```




```
# Testing and monitoring code, please ignore
# keep_n = matrix(nrow = 100, ncol = 10)
# prop_n_sd = 1:10
# for (j in 1:length(prop_n_sd)) {
#   n = 20
#   for (i in 1:100) {
#     # Generate n
#     prop_n = round(rnorm(1, mean=n, sd = prop_n_sd[j])) # round to ensure its integer, using r as tha
#     logn = log_n_func(n=prop_n, r, true_y, theta, p) - log_n_func(n=n, r, true_y, theta, p)
#     if (is.finite(logn) && log(runif(1)) < logn) {
#       n <- prop_n
#     }
#     keep_n[i,j] = n
#   }
# }
#
#
# plot(as.mcmc(keep_n), main="n")
# # cat("starting is:", base_r, " proposed is:", prop_r, "final is:", r)
```