Predicting Team Probabilities of Winning the 2023-2024 NBA Finals: Do Advanced Metrics Matter?

Will Singleton

I. Abstract

The use of data analytics has exponentially increased over the past two decades regardless of industry, and major league sports are no expectation to this. With this in mind, sports statistician Dean Oliver paved the way for the adoption of advanced metrics in the NBA with the publicantion his 2004 book, *Basketball on Paper*. Within this study, we analyze the 2023-2024 playoff teams in an attempt to predict each team's probabilities to win this year's NBA finals focusing on a variation of Oliver's Four Factor Rating proposed by Ryan Sullivan. To estimate these probabilities, we use a Bayesian logistic regression model with a Bernoulli sampling distribution and non-conjugate Normal prior. Overall, this study aims to highlight which team statistics and advanced metrics are the most important to win an NBA championship.

II. Introduction

This study aims to answer the question of "Can advanced metrics accurately predict the winner of the NBA finals?" We are interested in this question because the heavy adoption of these metrics in the past five years in sports to grade both players and teams. In nearly every broadcast within the three major US sports leagues, the MLB, NBA, and NFL, the announcers either mention one of the many advanced metrics, or they're displayed on the broadcast. These metrics seem useful, but there is often a gap between the viewer's understanding of the measurement. We look to bridge this gap of understanding through the application of these metrics to predict what teams have the best chance of winning the Finals and gain further insight into why teams are successful in the playoffs. With this rise in data analytics in the NBA, nearly all of the data from NBA games is publicly available online. Thus, we are able to explore this data to get a better understanding of today's teams and teams in the future. This study will focus on the combination of basic and advanced team metrics to estimate the probabilities of this years' teams to win an NBA championship.

III. Data Summary

The data used was collected from Sports Reference's Basketball Reference. Basketball Reference specializes in the collection and organization of NBA and WNBA for every season in their respective histories. The data is collected through

two main means, scorekeepers and player tracking systems. Each game as a team of four scorekeepers to manually track stats throughout the game. With the advancement of technology and machine learning within the past decade, the NBA adopted the use of player tracking in 2013 with the STATS SportVU system, and since 2017 the league relies on the use Second Spectrum. These systems use cameras present in the catwalks of every NBA arena to track and analyze the movements of every player on the court. For the study, we used the advanced stats for NBA teams from the regular season. These advanced stats were broken into two groups a training set, including data from all 48 teams that appeared in the Finals from 1999-2023, and a test set, including the data from all 16 teams in the 2024 NBA playoffs. It should be noted that the test set includes only the winners of the play-in games.

i. Response Variable

The response variable is log-odds of a team to win the NBA finals, thus becoming the league champion. The log-odds of winning the NBA finals is not the most useful in terms of discussion, hence this will be transformed into each teams probability of winning the Finals.

ii. Explanatory Variables

The NBA's base regular advanced stats include 26 different variables ranging from a team's total number of wins to their total attendance for the season. We narrowed down the number of variables to the 13 variables that we believe are the most important for being crowned the champion. These variables included the average player's ages on the team calculated on February 1st of the season, total wins, total losses, offensive rating, defensive rating, and Oliver's Four Factors for both offensive and defensive. Oliver's Four Factors are based on a team's offensive metrics, Offensive Four Factors, and their opponents' metrics against the team, Defensive Four Factors. These ratings are constructed from effective field goal percentage, turnover percentage, offensive rebound percentage, and free throw rate.

Ryan Sullivan, writer for the Sports Gambling Podcast who focuses on NBA analysis, proposed a variation for calculation for Oliver's team and opponent Four Factor Ratings.

$$Oliver's\ Team\ FFR = ((0.4 * eFG\%) - (0.25 * TOV\%) + (0.2 * OREB) + (0.15 * FTR))$$

$$Oliver's\ Opponent\ FFR = ((0.4 * eFG\%) - (0.25 * TOV\%) + (0.2 * OREB) + (0.15 * FTR))$$

$$Sullivan's\ Team\ FFR = ((0.5 * eFG\%) - (0.3 * TOV\%) + (0.15 * OREB) + (0.05 * FTR))$$

$$Sullivan's\ Opponent\ FFR = ((0.5 * eFG\%) - (0.3 * TOV\%) + (0.15 * OREB) + (0.05 * FTR))$$

Although, the Net Four Factor Rating calculation remains the same.

$$Net\ Four\ Factor\ Rating = Team\ FFR - Opponent\ FFR$$

Through Sullivan's adjustments of the weights for the individual elements of the Four Factor Rating, he was able to achieve stronger correlation to win percentage than Oliver's Four Factor Rating using data from the 1999-2000 season to the 2018-2019 season. Thus, we will use Sullivan's Four Factors to calculate the Net Four Factor Ratings for all teams in our datasets, where a higher Net Four Factor Rating is associated with a more successful team.

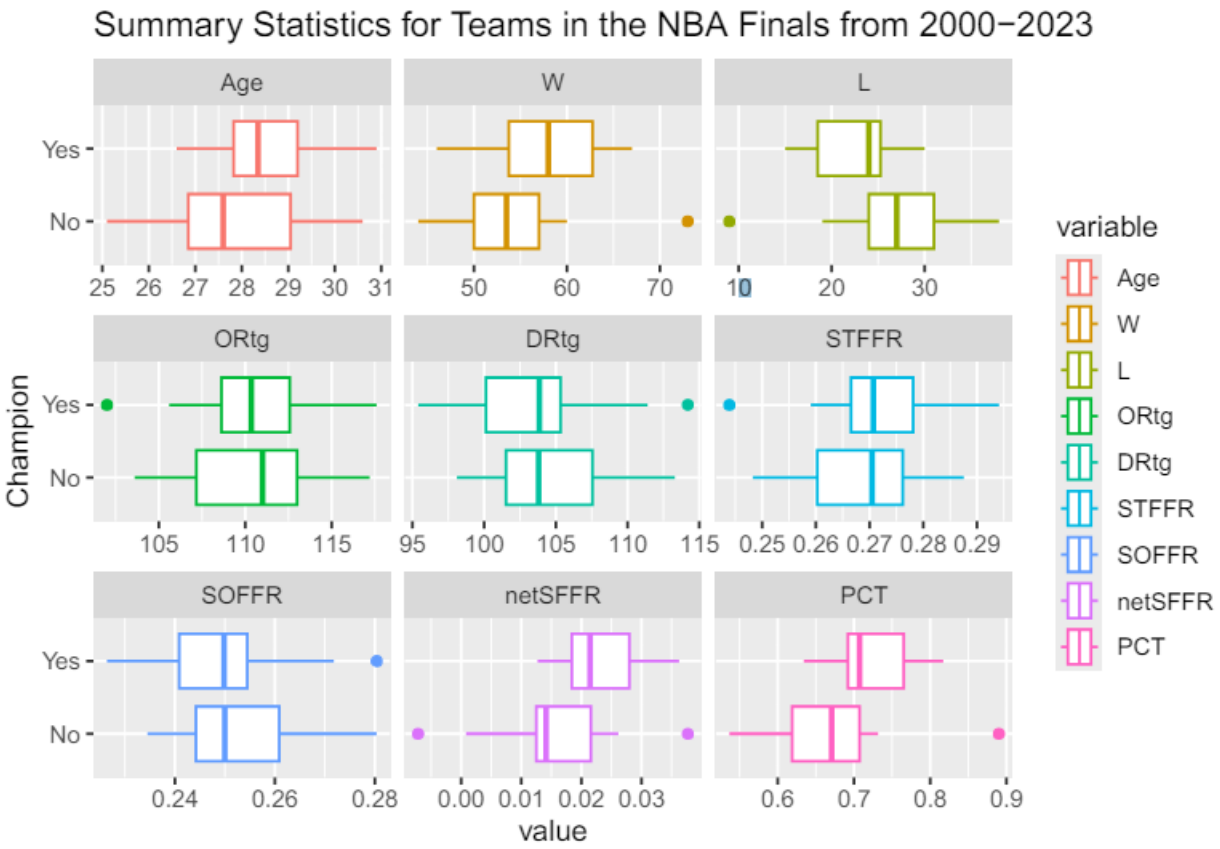
iii. Variable Calculations

Although, Basketball Reference's advanced stats includes both the offensive and defensive Four Factor measurements of effective field goal percentage, turnover percentage, free throw rate, the calculation of the rebounding percentages was different from the form we required. The advanced stats had both rebounding percentages in forms of per 100 rebounding opportunities, hence we simply divided them by 100. Now, the offensive rebounding percentage was in the necessary form. But, another issue arose, where instead of opponent offensive rebounding percentage, it includes team defensive rebounding percentage. Thus, we took this team defensive rebounding percentage and subtracted by 100 to achieve our correction of the rebounding percentages. We also had to adjust the turnover percentages by dividing by 100, as we ran into an issue when calculating the Net Four Factors Ratings, where all net ratings were negative. Lastly, we calculated each team's winning percentage in an attempt to capture the impact of both wins and losses.

IV. Analysis

i. Summary Statistics

We first began by visualizing our data of the teams who played the past years' Finals to determine the basic impact of our selected variables. From stacked box plots, we see that the means of only wins and Sullivan's Net Four Factor Rating seem to differ from champions and those who lost in the Finals, losers. Interestingly, the means of losses and winning percentage did not seem to differ, contrary to the means of wins.



ii. Model Selection

In order to estimate the probabilities of a team winning the NBA finals, we rely on the use of logistic regression. With our beliefs about the association of a teams total wins and Sullivan's Net Four Factor Rating with winning the Finals, we will include these variables in our model, and despite the lack of strong evidence for age being associated with becoming the NBA champion we will included as well. To achieve our goal of using a Bayesian approach, first, we will fit a Frequentist's logistic model predicting the log-odds of winning the Finals to build our sampling distribution. Although, the only statistically significant coefficient at level of 5% is Sullivan's Net Four Factor Rating, we will continue with our analysis due to the differences in Bayesian and Frequentist's approaches.

$$E(Y_i|\boldsymbol{X_i}) = p(Y_i = 1|\boldsymbol{X_i}) = \frac{\exp\{\boldsymbol{X_i^{T\beta}}\}}{1 + \exp\{\boldsymbol{X_i^{T\beta}}\}}$$

Where Y_i our response variable of winning the Finals or not.

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	-14.224	7.603	-1.871	0.061	-30.589	-0.122
Age	0.455	0.271	1.680	0.093	-0.055	1.026
W	-0.031	0.082	-0.384	0.701	-0.198	0.131
netSFFR	160.779	69.852	2.302	0.021	32.319	309.800

Thus, we arrive at the conclusion of:

$$Y_i | X_i \sim Bern(p_i)$$

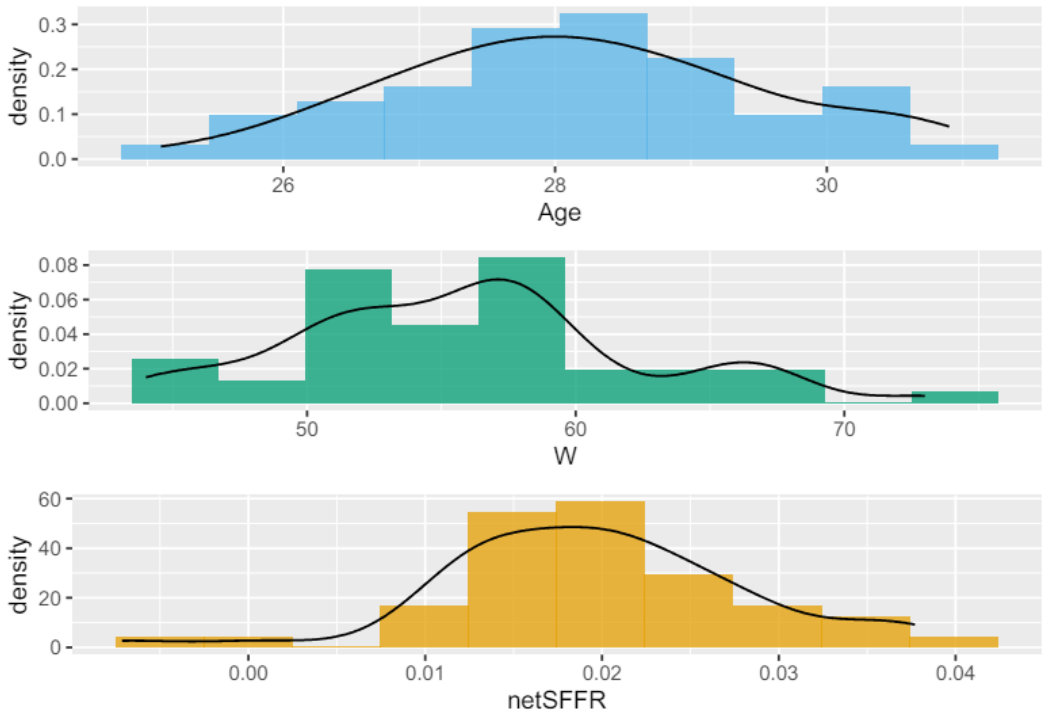
$Y_i, \ldots, Y_n$ are independent and:

$$p_i = p(Y_i = 1|\boldsymbol{X_i}) = \frac{\exp\{\boldsymbol{X_i^T\beta}\}}{1 + \exp\{\boldsymbol{X_i^T\beta}\}}$$

Due to the difficultly of establishing a conjugate prior for our β coefficients, we proceed with a non-conjugate independent Normal prior:

$$\beta_j \sim N(\beta_{j0}, \sigma_{j0}^2)$$

Using a Normal prior is applicable here due to the approximately Normal distribution of age, win totals, and Sullivan's Net Four Factor Rating.



Thus, our coefficients are independent with hyper-parameters $\beta_{j0}, \sigma_{j0}^2$ for *j = 1,…, p.* Next, we establish our posterior:

$$p(\beta|Y_{1:n}, \boldsymbol{X}_{1:n}) \propto L(\beta; Y_{1:n}, \boldsymbol{X}_{1:n})p(\beta)$$

$$\propto \frac{\exp\{\sum_{i=1}^{n} Y_i \boldsymbol{X}_i^T \beta\}}{\prod_{i=1}^{n}(1+\exp\{\boldsymbol{X}_i^T \beta\})} \cdot [\prod_{j=1}^{p} \texttt{dnorm}(\beta_j; \beta_{j0}, \sigma_{j0})]$$

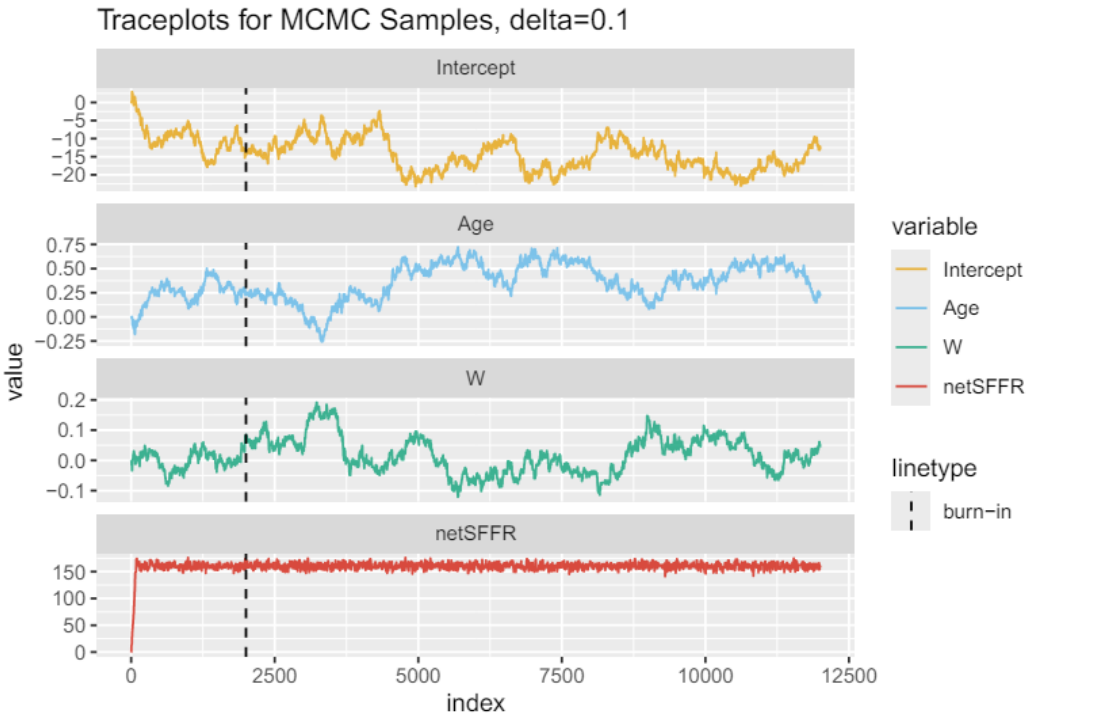
Now that the posterior is no longer a conjugate, we must rely on the use of Metropolis Algorithm to form a posterior distribution.

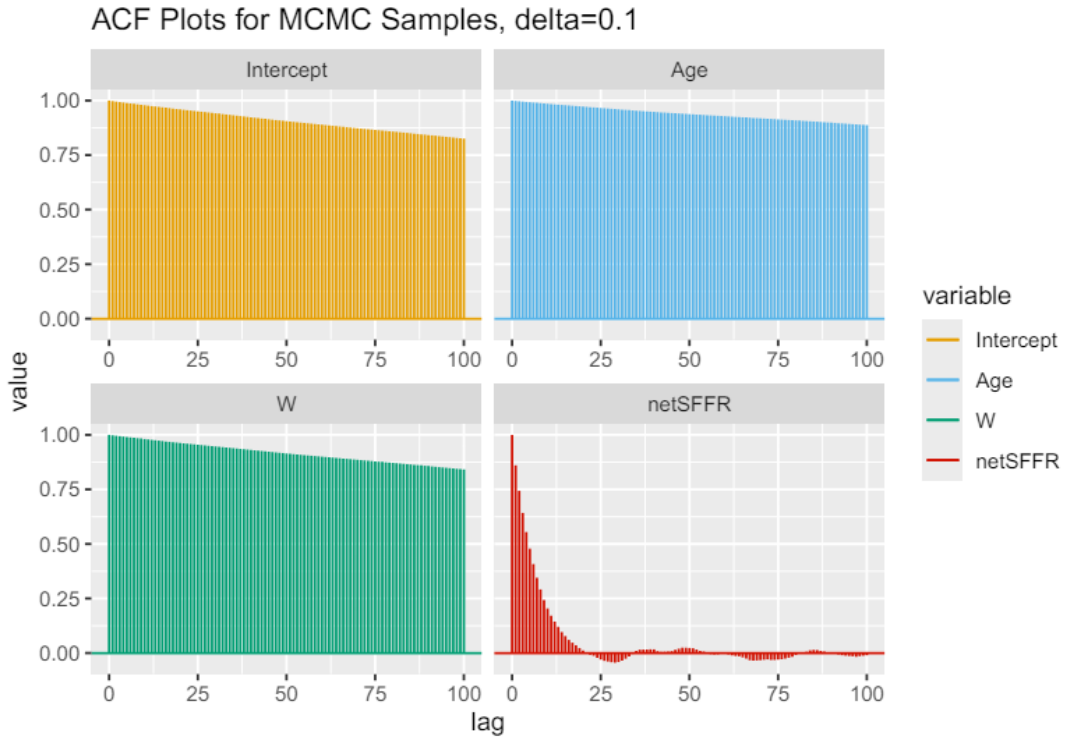
iii. Metropolis Algorithm

First, we will run the Metropolis Algorithm with a total of 10,000 Markov chain Monte Carlo samples after a burn-in of 2,000 using our training set based on teams' ages, win totals, and Sullivan's Net Four Factor Rating from the teams that appeared in the Finals from the past two decades. After varying the stepsizes, we proceeded with a stepsize of 0.1 to obtain a acceptance rate of approximately 24%.

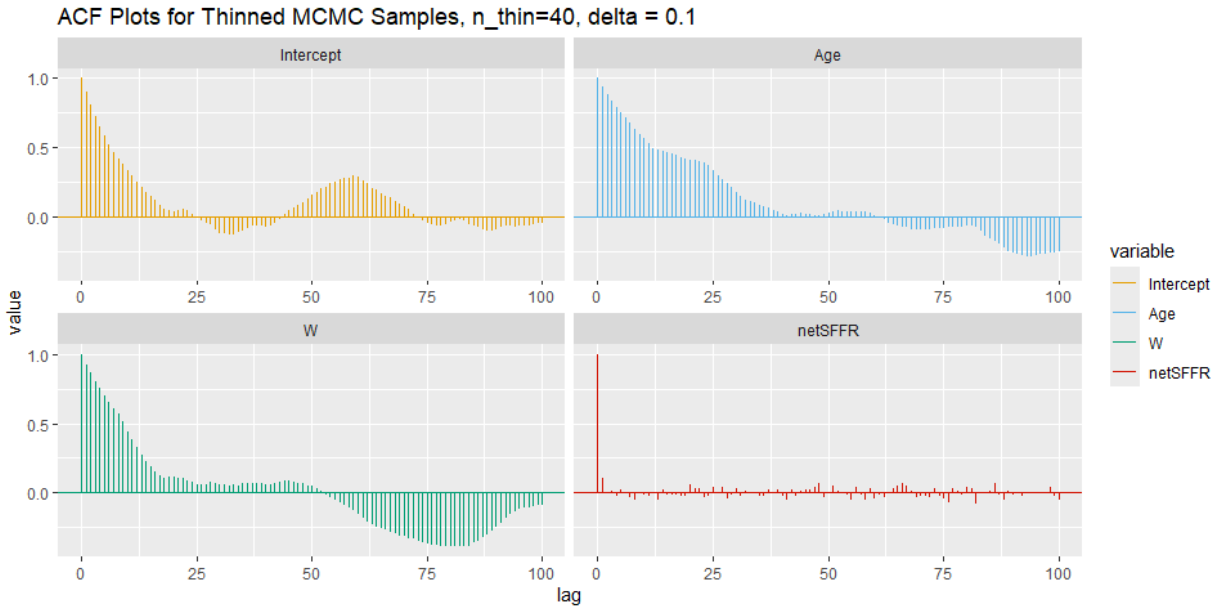
iv. Diagnostics

After running the Metropolis Algorithm, we checked for signs of low efficiency and autocorrelation with the use of trace plots and ACF plots. From the plots we realized that there was very low efficiency for intercept, age, and wins, since they did not converge within MCMC samples. As well, we saw that there was very high autocorrelation for intercept, age, and wins.


Traceplots for MCMC Samples, delta=0.1
Intercept
Age
W
netSFFR
value
index
variable
Intercept
Age
W
netSFFR
linetype
burn-in


ACF Plots for MCMC Samples, delta=0.1
Intercept
Age
W
netSFFR
value
lag
variable
Intercept
Age
W
netSFFR

To resolve these issues of high autocorrelation, we preformed MCMC thinning by a thinning rate of 40.



v. Posterior Predictions

With the issue of high autocorrelation addressed we are able to proceed to our posterior predictions using our thinned MCMC samples. Our coefficients and 95% confident intervals for our Bayesian logistic model follow:

Intercept	Age	Wins	netSFFR
-13.25161642	0.33244400	0.01406283	160.38423039

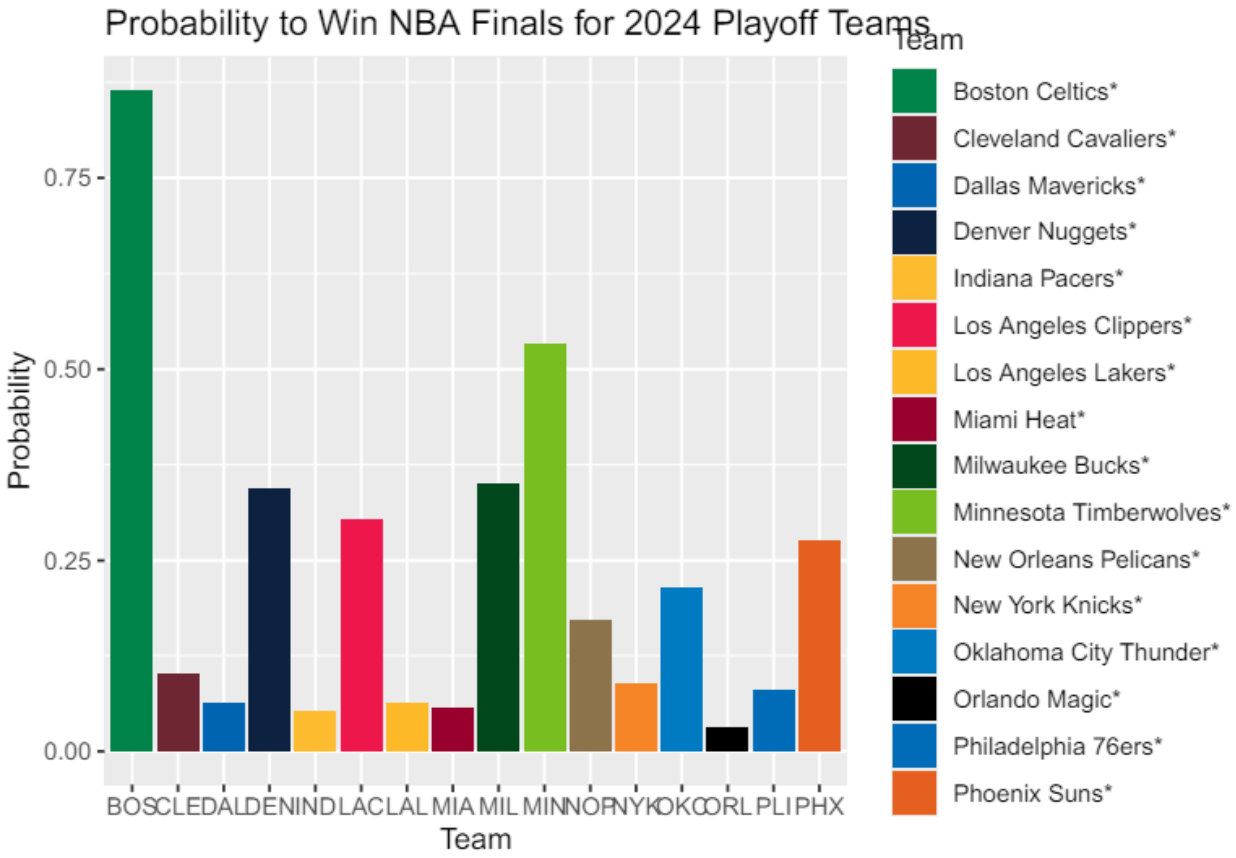
	Intercept	Age	Wins	netSFFR
2.5%	-21.512697	-0.08983975	-0.08084164	150.9064
97.5%	-5.281457	0.64137010	0.15578242	170.7474

In comparison to the Frequentist's GLM, our estimates are quite close, signifying that model is fit to estimate the probabilities for each team to win this year's NBA Finals. This is further assured through the use of in-sample estimations to measure our accuracy of predicting the winner of the past finals in our training set, where we had accuracy measurement of 73%.

vi. Conclusions

Finally, to estimate each current team's probability of winning this the Finals this year, we approximate the posterior distribution of $\tilde{p}_i | \tilde{X}_i, Y, X$, and apply the logistic function:

$$\mu_{\boldsymbol{X}} = E(Y|\boldsymbol{X}) = p(Y = 1|\boldsymbol{X}) = \frac{\exp\{\boldsymbol{X}^T\beta\}}{1 + \exp\{\boldsymbol{X}^T\beta\}}$$



From our results of Bayesian logistic regression, we are able to conclude that the Boston Celtics have the highest probability of winning this year's Finals based on the application of advanced stats. This result is not very surprising due the fact that the Celtics had the highest Net Four Factor Rating based on Sullivan's variation, as well as having the highest number of total wins and being one of the older teams in this year's playoffs.

VII. Appendix

Run this before: Sys.setenv("VROOM_CONNECTION_SIZE" = 131072 * 2)

Data Entry

```
games2024 <- game_logs(seasons = 2024, result_types = "team")
```

```
## Acquiring NBA basic team game logs for the 2023-24 Regular Season
```

```
## Warning: Supplying `...` without names was deprecated in tidyr 1.0.0.
## i Please specify a name for each selection.
## i Did you want `data = -c(slugLeague, typeResult, slugSeason, yearSeason)`?
## i The deprecated feature was likely used in the nbastatR package.
##   Please report the issue at <https://github.com/abresler/nbastatR/issues>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## Warning: `cols` is now required when using `unnest()`.
## i Please use `cols = c(dataTables)`.
```

```
adv <- read.xlsx("nba2024.xlsx",1)
#nba_train <- read.xlsx("nbatrain.xlsx",1)
nba_train <- read.xlsx("nbatrainfinals.xlsx",1)
```

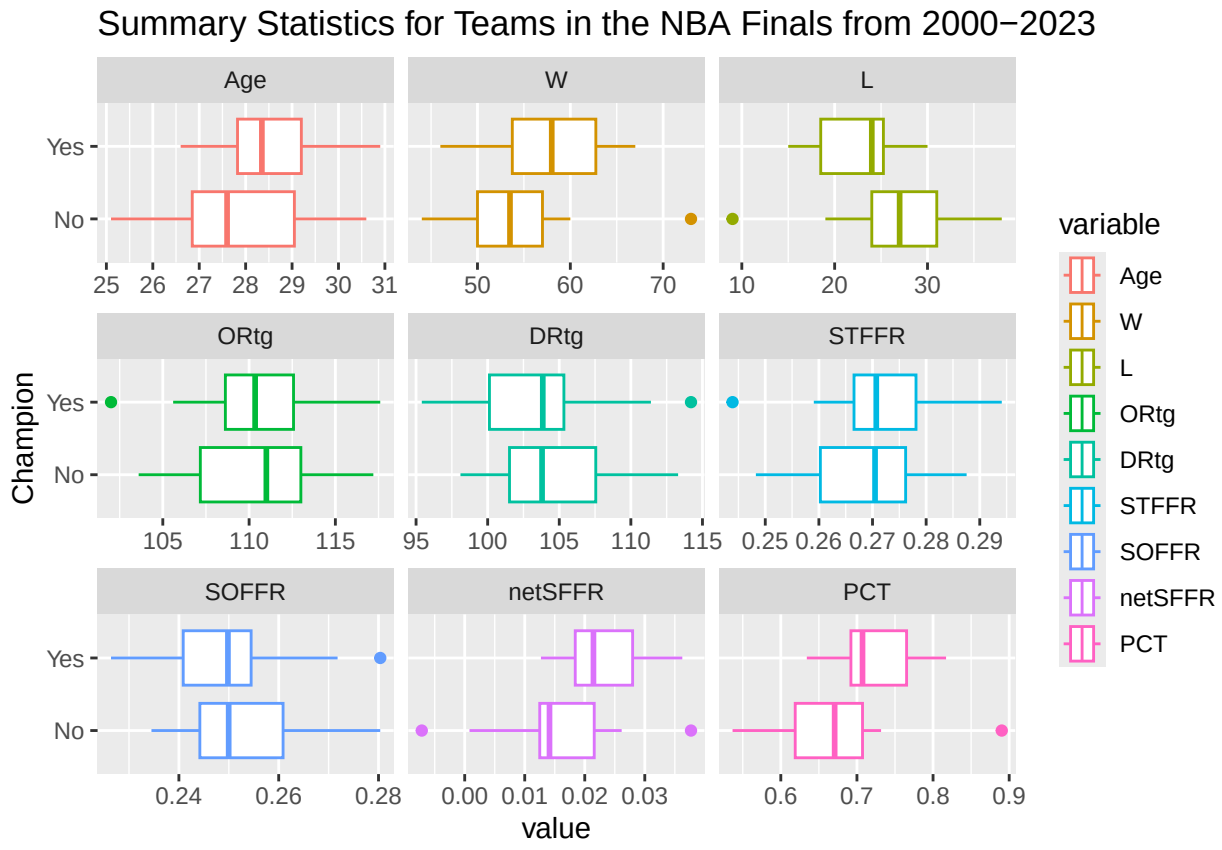
```
refine_train <- c("Team", "Age", "W", "L","ORtg","DRtg", "STFFR", "SOFFR", "netSFFR")
finals_train <- nba_train[,refine_train]

finals_train$PCT <- finals_train$W/(finals_train$W + finals_train$L)
finals_train$Champion <- c(1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,
                           1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0)
#finals_train

base <- c("Team","Age", "W", "L","ORtg","DRtg", "STFFR", "SOFFR", "netSFFR")
pres_nba <- adv[,base]
pres_nba$PCT <- pres_nba$W/(pres_nba$W + pres_nba$L)
#pres_nba
```

Summary Plots

```
finals_train$Champion <- c(1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,
                           1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0)
finals_train$Champion <- ifelse(finals_train$Champion == 1, "Yes", "No"); #finals_train
df1 <- melt(finals_train); #df1
```

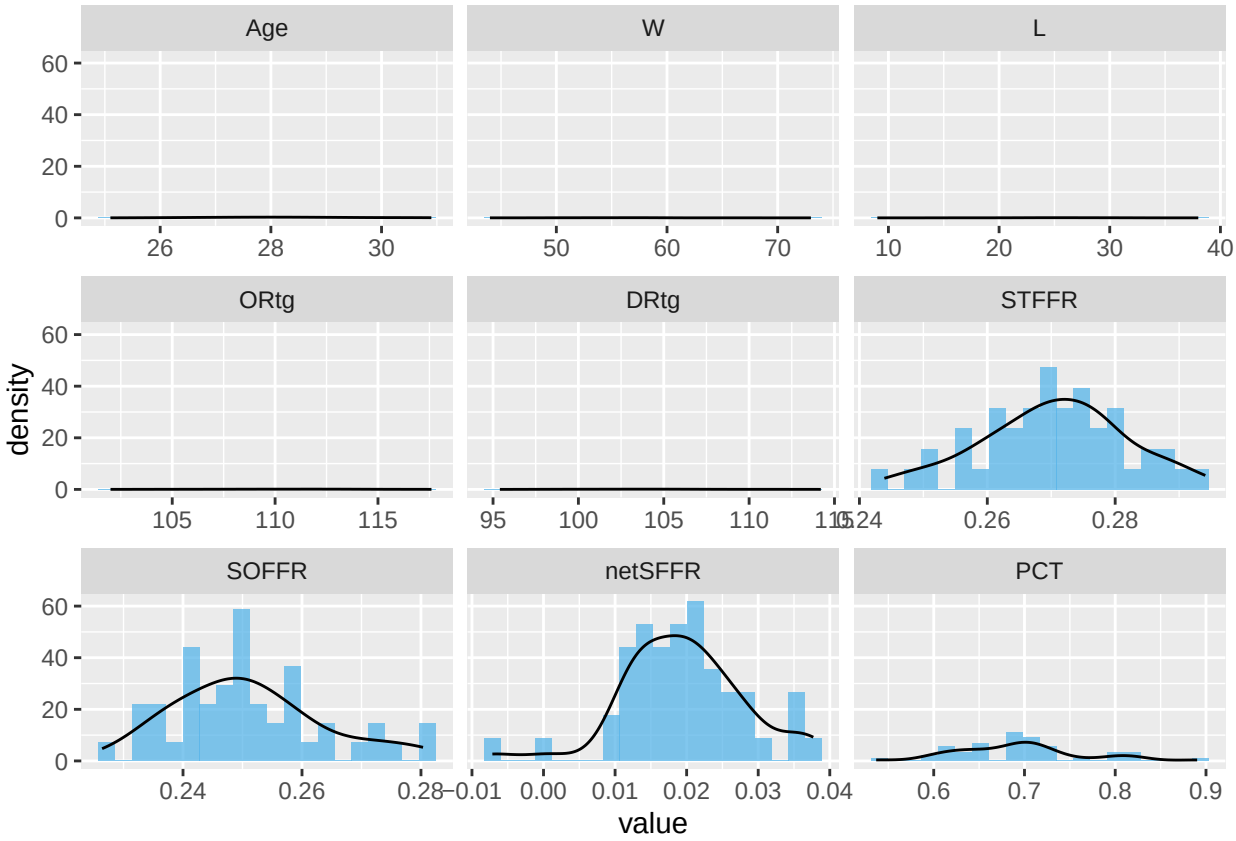
```
## Using Team, Champion as id variables
```

```
ggplot(data=df1, aes(x=value, y=Champion)) +
   geom_boxplot(aes(color=variable)) +
  facet_wrap(~variable, scales = "free_x", ncol=3) +
  ggtitle("Summary Statistics for Teams in the NBA Finals from 2000-2023")
```




```
ggplot(data = df1, aes(x=value, y=..density..)) + geom_histogram(fill = cbbPalette[3], bins=20, alpha=0
  geom_density(aes(y=..density..)) + facet_wrap(~variable, scales = "free_x", ncol=3)
```

```
## Warning: The dot-dot notation (‘..density..‘) was deprecated in ggplot2 3.4.0.
## i Please use ‘after_stat(density)‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```



Frequentist's Logit/GLM

```
finals_train$Champion <- c(1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,
                           1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0)
fit_glm <- glm(Champion ~ Age + W + netSFFR, family = binomial, data = finals_train)
summary(fit_glm)
```

```
##
## Call:
## glm(formula = Champion ~ Age + W + netSFFR, family = binomial,
##     data = finals_train)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -14.22399    7.60287  -1.871   0.0614 .
## Age           0.45529    0.27104   1.680   0.0930 .
## W            -0.03143    0.08185  -0.384   0.7010
## netSFFR     160.77856   69.85212   2.302   0.0214 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 66.542  on 47  degrees of freedom
## Residual deviance: 53.224  on 44  degrees of freedom
## AIC: 61.224
##
## Number of Fisher Scoring iterations: 4
```

```
beta_glm <- coef(fit_glm); beta_glm
```

```
##  (Intercept)          Age            W      netSFFR
## -14.22398628   0.45528708  -0.03143165 160.77855586
```

```
## The log odds model
tidy(fit_glm, vcov=vcovHC(fit_glm), conf.int=T) %>% kable(format="latex", digits=3)
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	-14.224	7.603	-1.871	0.061	-30.589	-0.122
Age	0.455	0.271	1.680	0.093	-0.055	1.026
W	-0.031	0.082	-0.384	0.701	-0.198	0.131
netSFFR	160.779	69.852	2.302	0.021	32.319	309.800

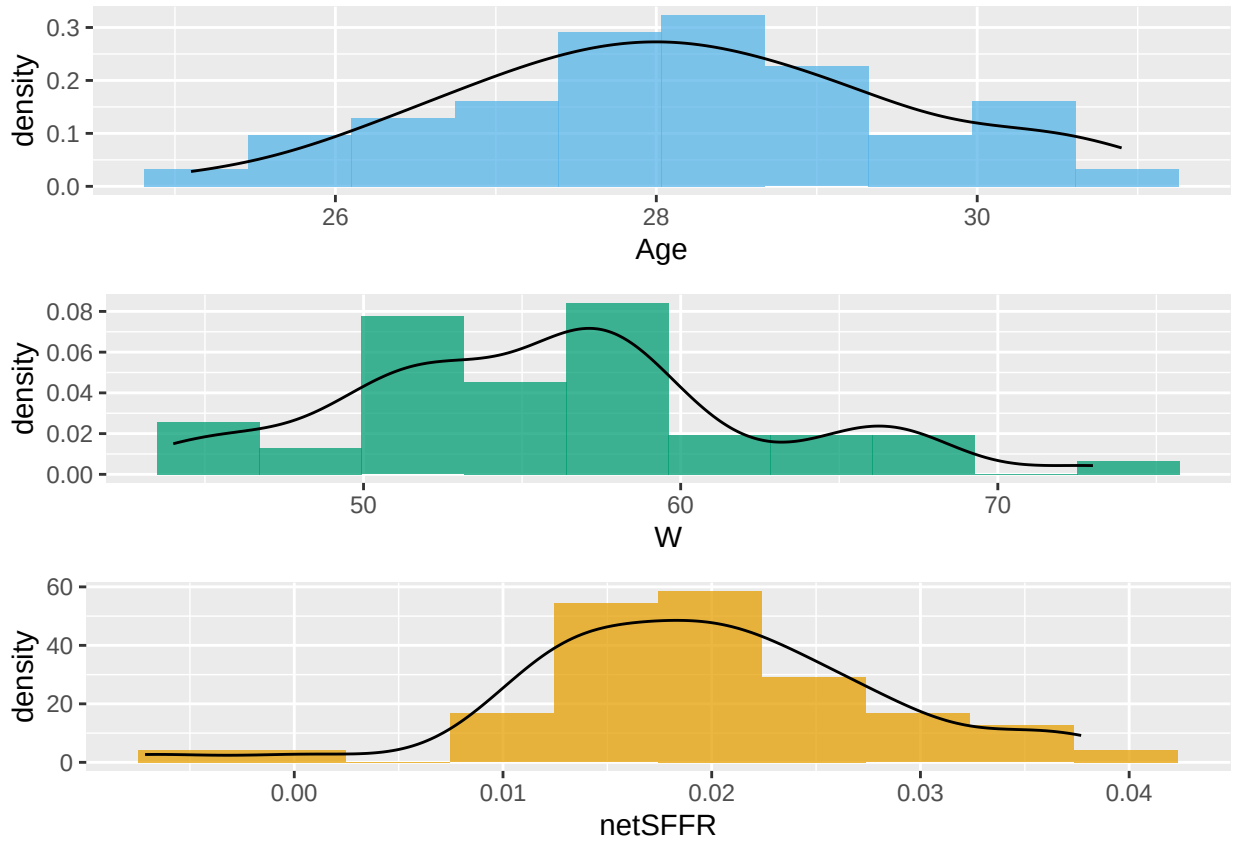
```
## The odds model
tidy(fit_glm, vcov=vcovHC(fit_glm), conf.int=T,
     exponentiate=T) %>% kable(format="latex", digits=3)
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	0.000000e+00	7.603	-1.871	0.061	0.000000e+00	8.850000e-01
Age	1.577000e+00	0.271	1.680	0.093	9.470000e-01	2.789000e+00
W	9.690000e-01	0.082	-0.384	0.701	8.200000e-01	1.140000e+00
netSFFR	6.687128e+69	69.852	2.302	0.021	1.085921e+14	3.503026e+134

```
expmodel <- tidy(fit_glm, vcov=vcovHC(fit_glm), conf.int=T,
     exponentiate=T) %>% mutate(across(where(is.numeric), ~format(.x, scientific = FALSE, digits = 3))) %>%
  kable(format="latex", digits=3)
```

Plots Showing that selected variables are approximately normal

```
## Density Plots
# finals_train

ap <- ggplot(data = finals_train, aes(x=Age, y=..density..)) + geom_histogram(fill = cbbPalette[3], bins
  geom_density(aes(y=..density..))

pp <- ggplot(data = finals_train, aes(x=W, y=..density..)) + geom_histogram(fill = cbbPalette[4], bins=
  geom_density(aes(y=..density..))

np <- ggplot(data = finals_train, aes(x=netSFFR, y=..density..)) + geom_histogram(fill = cbbPalette[2],
  geom_density(aes(y=..density..))

grid.arrange(ap, pp, np)
```



Metropolis for Logistic Regression

```
## final variable selection
dat_train <- c("Age", "W","netSFFR", "Champion")
past_train <- finals_train[,dat_train]

dat_test <- c("Age", "W","netSFFR")
pres_test <- pres_nba[,dat_test]
```

```
## Metropolis
## Data prep
X <- past_train[,-4]
Xi <- as.matrix(cbind(1, X))
Y <- as.numeric(past_train$Champion)

##Prior
beta0 <- as.matrix(coef(fit_glm))
sigma2_beta_glm <- summary(fit_glm)$coefficients[,2]^2
sigma2_0 <- 5^2

##dimension and size
n <- length(Y)
p <- dim(Xi)[2]
```

```r
n_mcmc <- 10000
burn <- 2000

##MCMC prep
## storage
beta_all <- matrix(nrow=(n_mcmc + burn), ncol=p)
## stepsize
delta <- 0.5
```

```r
##loglikelihood for logistic regression
loglik_lr <- function(x = Xi, y = Y, beta){
  loglik <- sum((x %*% beta) * y) - sum(log(1 + exp(x %*% beta)))
  return(loglik)
}
```

```r
##initialize
beta_init <- c(0,0,0,0)
beta_curr <- beta_init

## keep track of the number of accepted proposals
n_acc <- 0

set.seed(465)
for(s in 1:(burn+n_mcmc)){
  ##current
  loglik_curr <- loglik_lr(beta=beta_curr)
  logprior_curr <- sum(dnorm(beta_curr, mean=beta0, sd=sqrt(sigma2_0), log = T))
  logpost_curr <- loglik_curr + logprior_curr

  ##proposal
  beta_prop <- rnorm(n = p, mean = beta_curr, sd = sqrt(sigma2_beta_glm) * delta)
  loglik_prop <- loglik_lr(beta = beta_prop)
  logprior_prop <- sum(dnorm(beta_prop, mean = beta0, sd = sqrt(sigma2_0),log = T))
  logpost_prop <- loglik_prop + logprior_prop

  ##accept/reject
  u <- log(runif(1))
  if(u < (logpost_prop -logpost_curr)){ ##accept
    beta_curr <- beta_prop
    n_acc <- n_acc + 1
  }
  beta_all[s, ] <- beta_curr
}
```

```r
###burn-in
beta_mcmc <- beta_all[-c(1:burn),]

##Acceptance Rate
rate_acc <- n_acc/(n_mcmc + burn)
rate_acc
```

```
## [1] 0.01691667
```

```r
## posterior mean
beta_pm_mcmc <- apply(beta_mcmc, 2, mean)
beta_pm_mcmc
```

```
## [1] -13.60523809   0.41910276  -0.02423574 160.75256905
```

```r
## 95% credible interval
apply(beta_mcmc, 2, function(x){quantile(x, c(0.025, 0.975))})
```

```
##             [,1]      [,2]        [,3]     [,4]
## 2.5%  -21.427603 0.1142568 -0.12884064 150.9974
## 97.5%  -6.286607 0.7046748  0.08346527 170.7632
```

```r
apply(beta_mcmc, 2, function(x){quantile(x, c(0.05, 0.95))})
```

```
##           [,1]      [,2]        [,3]     [,4]
## 5%  -19.826709 0.1415427 -0.11647261 152.3431
## 95%  -6.426148 0.6747802  0.06200218 168.6182
```

Metropolis: Varying Stepsizes

```r
beta_all <- matrix(nrow=(n_mcmc + burn), ncol=p)
delta_all <- c(0.5, 0.1, 0.07)
rate_acc_all <- numeric(length(delta_all))
list_mcmc_all <- list()

for(id in 1:length(delta_all)){
  delta <- delta_all[id]
  ##initialize
  beta_curr <- beta_init

  ## keep track of the number of accepted proposals
  n_acc <- 0

  set.seed(465)
  for(s in 1:(burn+n_mcmc)){
    ##current
    loglik_curr <- loglik_lr(beta=beta_curr)
    logprior_curr <- sum(dnorm(beta_curr, mean=beta0, sd=sqrt(sigma2_0), log = T))
    logpost_curr <- loglik_curr + logprior_curr

    ##proposal
    beta_prop <- rnorm(n = p, mean = beta_curr, sd = sqrt(sigma2_beta_glm) * delta)
    loglik_prop <- loglik_lr(beta = beta_prop)
    logprior_prop <- sum(dnorm(beta_prop, mean = beta0, sd = sqrt(sigma2_0),log = T))
    logpost_prop <- loglik_prop + logprior_prop

    ##accept/reject
    u <- log(runif(1))
```

```
    if(u < (logpost_prop -logpost_curr)){ ##accept
      beta_curr <- beta_prop
      n_acc <- n_acc + 1
    }
    beta_all[s, ] <- beta_curr
  }
  ###burn-in
  #beta_mcmc <- beta_all[-c(1:burn),]
  list_mcmc_all[[id]] <- beta_all
  ##Acceptance Rate
  rate_acc <- n_acc/(n_mcmc + burn)
  rate_acc_all[id] <- rate_acc
}
```

```
## acceptance rates
names(rate_acc_all) <- delta_all
rate_acc_all
```

```
##        0.5        0.1       0.07
## 0.01691667 0.23908333 0.35775000
```

Estiamtes for stepsize $\delta = 0.1$

```
## burn-in
list_mcmc_rmburn <- lapply(list_mcmc_all, function(x){x[-(1:burn), ]})

## estimates for delta = 0.1
beta_mcmc_delta <- list_mcmc_rmburn[[2]]
#beta_mcmc_delta <- list_mcmc_all[[2]]
## posterior mean
beta_pm_mcmc <- apply(beta_mcmc_delta, 2, mean)
beta_pm_mcmc
```

```
## [1] -14.7878035   0.3763367   0.0189702 160.6324844
```

```
## 95% credible interval
apply(beta_mcmc_delta, 2, function(x){quantile(x, c(0.025, 0.975))})
```

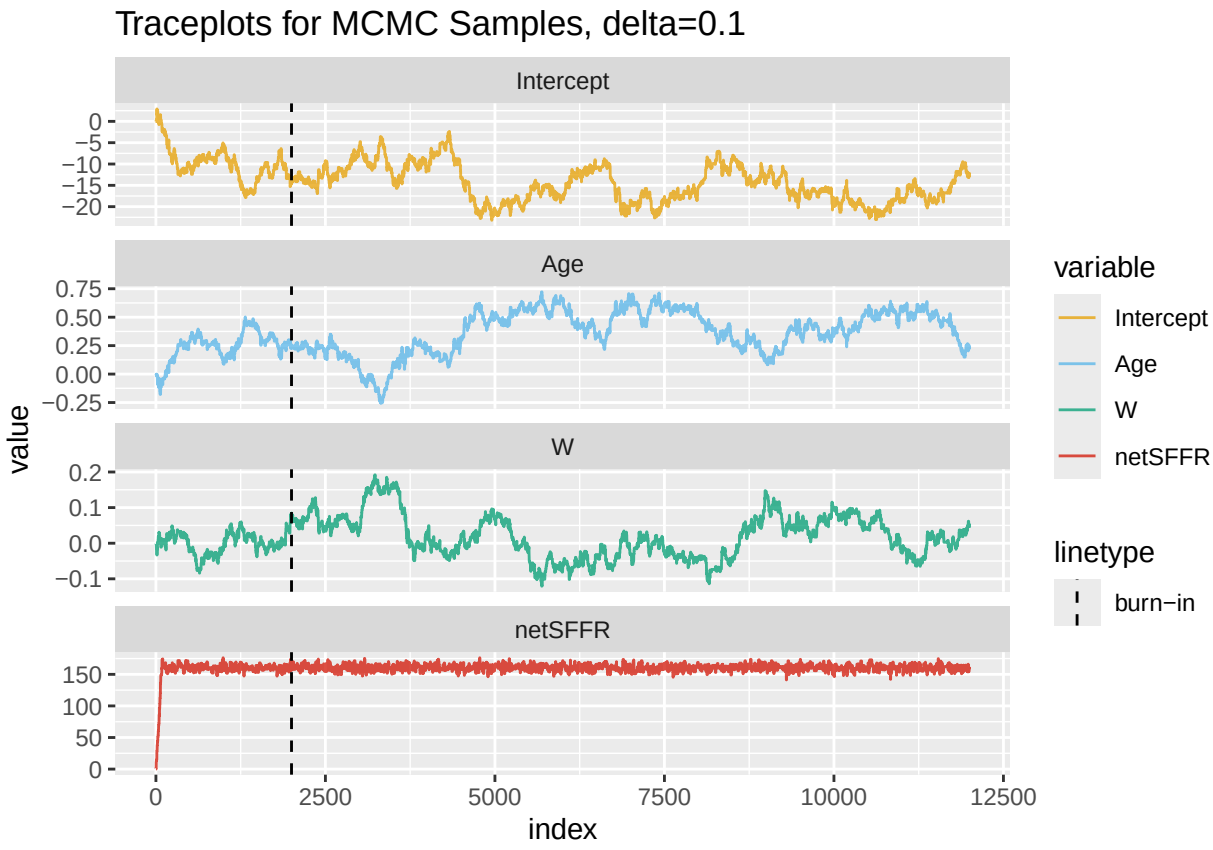
```
##             [,1]        [,2]        [,3]     [,4]
## 2.5%  -21.787138 -0.05664605 -0.08191674 151.0552
## 97.5%  -6.082182  0.64523914  0.15414194 170.1665
```

```
apply(beta_mcmc_delta, 2, function(x){quantile(x, c(0.05, 0.95))})
```

```
##           [,1]      [,2]        [,3]     [,4]
## 5%  -21.238683 0.0469895 -0.07271184 152.2977
## 95%  -7.656427 0.6214132  0.13797245 168.6674
```

Diagnostics

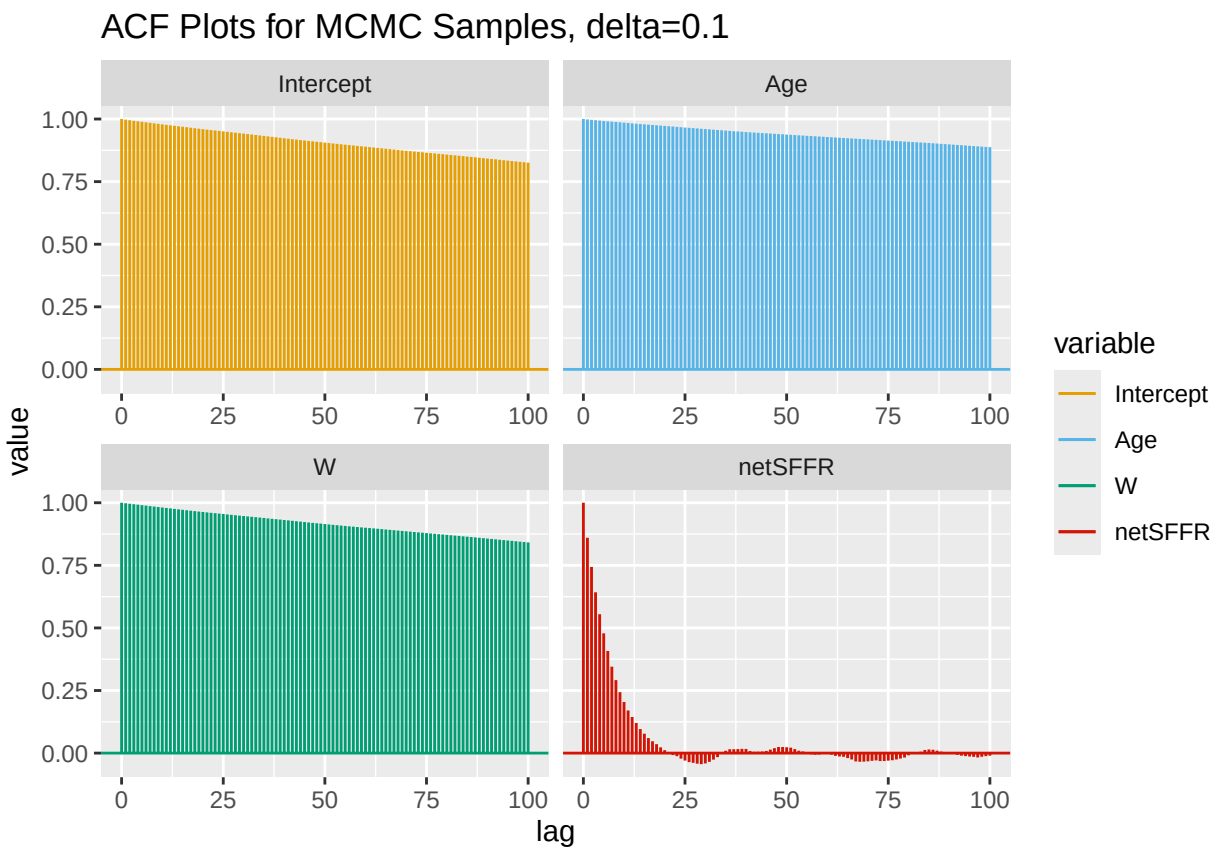
```
##Traceplot

data.beta <- data.frame(list_mcmc_all[[2]])
colnames(data.beta) <- colnames(Xi)
colnames(data.beta)[1] <- "Intercept"
data.beta <- data.frame(data.beta)
data.beta$index <- 1:(n_mcmc+burn)
data.beta.melt <- melt(data.beta, id.vars = "index")

g.trace.beta <- ggplot(data.beta.melt, aes(index, value, color=variable)) +
  facet_wrap(~variable, scales = "free_y", ncol=1) +
  geom_line(alpha=0.75) +
  geom_vline(aes(xintercept=burn, linetype = "burn-in")) +
   scale_color_manual(values = cbbPalette[2:5]) +
  scale_linetype_manual(values = 2)
g.trace.beta + ggtitle("Traceplots for MCMC Samples, delta=0.1 ")
```



Stepsize potentially too low → we have low efficiency for intercept, Age, and Wins

```
beta.acfs <- apply(list_mcmc_rmburn[[2]], 2, function(x){acf(x, lag.max=100,plot = F)$acf})
data.beta.acf <- data.frame(beta.acfs)
colnames(data.beta.acf) <- colnames(data.beta[1:4])
data.beta.acf$lag <- 0:100
```

```
data.beta.acf <- melt(data.beta.acf, id.vars = "lag")

ggplot(data = data.beta.acf, mapping = aes(x = lag, y = value)) +
  geom_hline(aes(yintercept = 0, color=variable)) +
  geom_segment(mapping = aes(xend = lag, yend = 0, color=variable)) +
  facet_wrap(~variable, scales = "free_x", ncol=2) +
   scale_color_manual(values = cbbPalette[2:5]) +
  ggtitle("ACF Plots for MCMC Samples, delta=0.1")
```



ACF Plots → very high auto-correlation for intercept, Age, and Wins

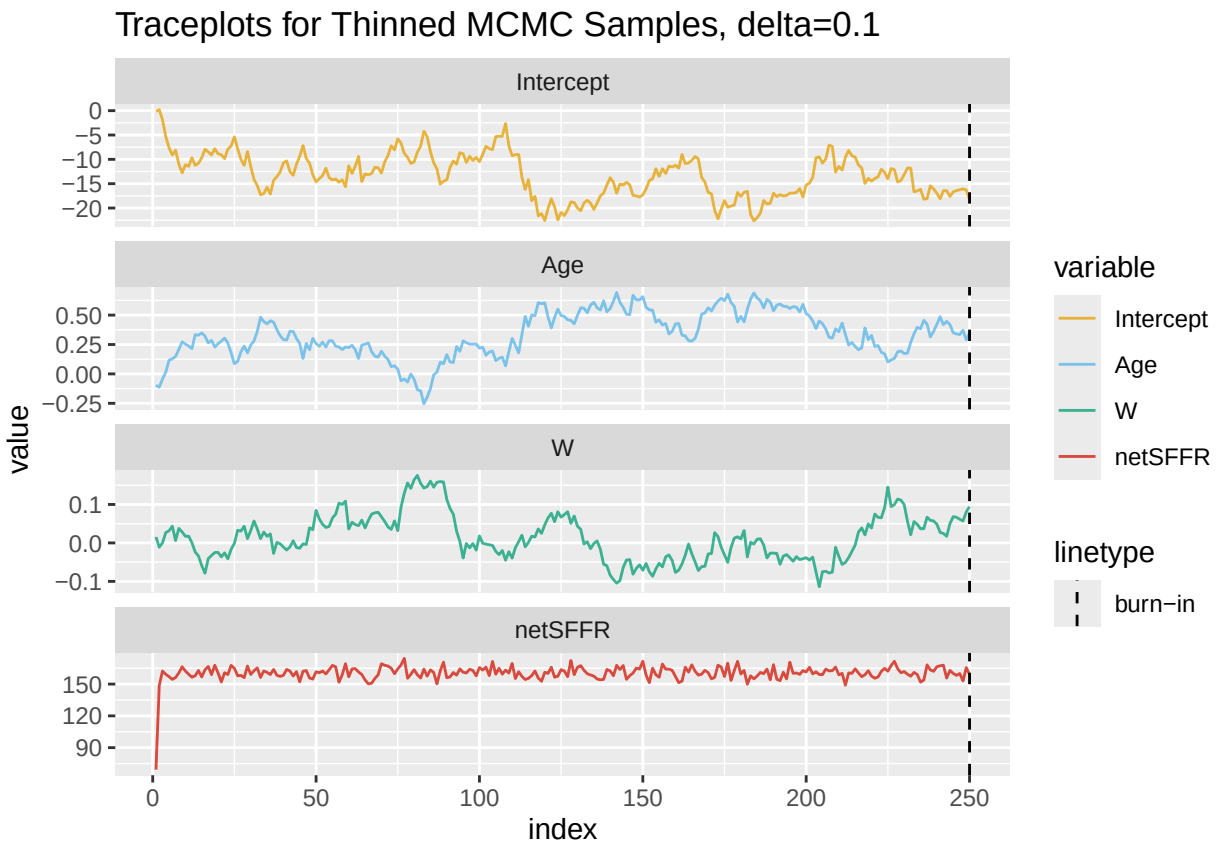
```
colnames(list_mcmc_rmburn[[2]]) <- c("Intercept",colnames(X))
effectiveSize(list_mcmc_rmburn[[2]])
```

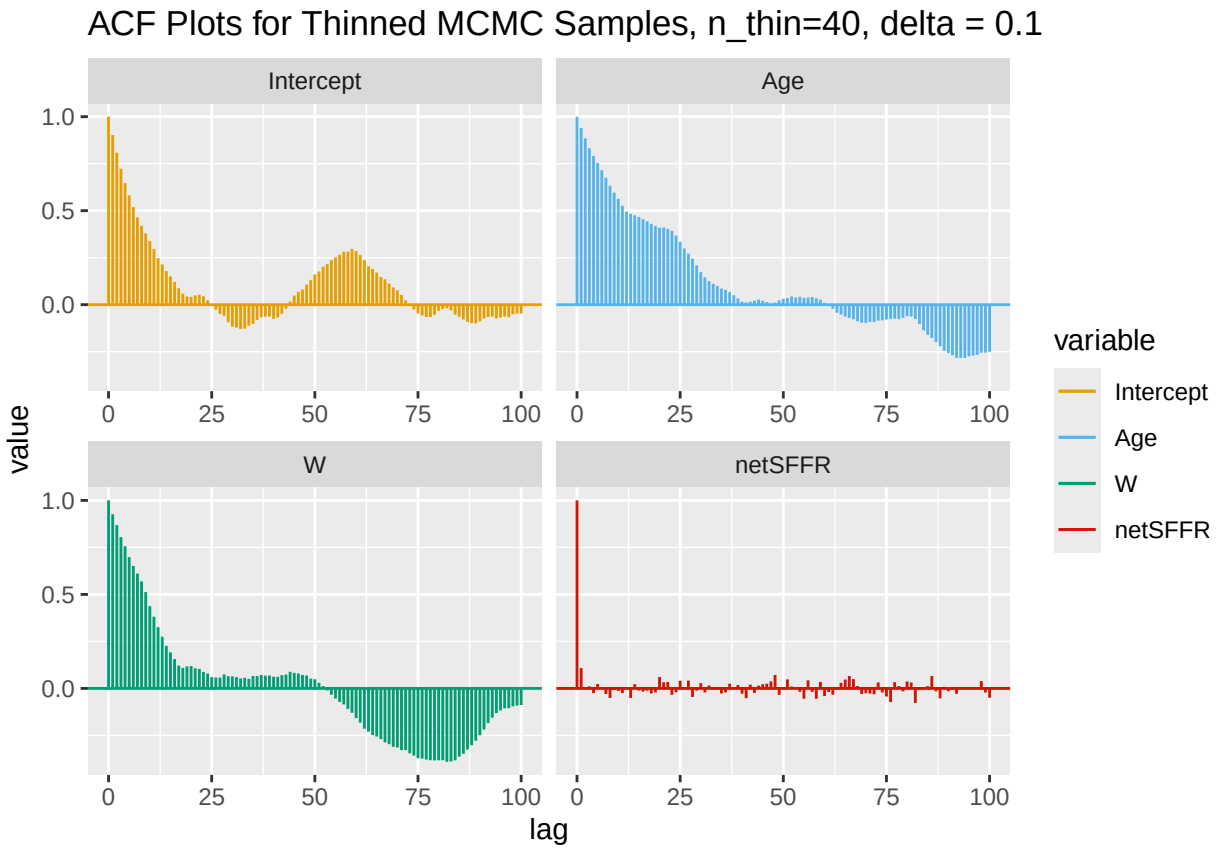
```
##  Intercept        Age          W    netSFFR
##  10.617457   7.268437   9.906979 751.429540
```

```
## thinning by every 40 sample
beta_mcmc <- list_mcmc_all[[2]]
n_thin <- 40
beta_mcmc_thin <- beta_mcmc[(1:floor(n_mcmc/n_thin))*n_thin,]
```

```
thinned <- 250

data.beta <- data.frame(beta_mcmc_thin)
```

```r
colnames(data.beta) <- colnames(Xi)
colnames(data.beta)[1] <- "Intercept"
data.beta <- data.frame(data.beta)
data.beta$index <- 1:(250)
data.beta.melt <- melt(data.beta, id.vars = "index")

g.trace.beta <- ggplot(data.beta.melt, aes(index, value, color=variable)) +
  facet_wrap(~variable, scales = "free_y", ncol=1) +
  geom_line(alpha=0.75) +
  geom_vline(aes(xintercept=thinned, linetype = "burn-in")) +
   scale_color_manual(values = cbbPalette[2:5]) +
  scale_linetype_manual(values = 2)
g.trace.beta + ggtitle("Traceplots for Thinned MCMC Samples, delta=0.1 ")
```





Thinned MCMC Estimates

```
#beta_mcmc_thin

## posterior mean
beta_pm_mcmc <- apply(beta_mcmc_thin, 2, mean)
beta_pm_mcmc
```

```
## [1] -13.25161642   0.33244400   0.01406283 160.38423039
```

```
## 95% credible interval
apply(beta_mcmc_thin, 2, function(x){quantile(x, c(0.025, 0.975))})
```

```
##              [,1]        [,2]        [,3]     [,4]
## 2.5%  -21.512697 -0.08983975 -0.08084164 150.9064
## 97.5%  -5.281457  0.64137010  0.15578242 170.7474
```

```
apply(beta_mcmc_thin, 2, function(x){quantile(x, c(0.05, 0.95))})
```

```
##           [,1]         [,2]        [,3]     [,4]
## 5%  -20.423471 -0.005208949 -0.07499721 152.3372
## 95%  -6.848696  0.621721556  0.13623664 168.6606
```

In-Sample Estimations

```
## posterior of log-odds
logodd_mcmc <- Xi %*% t(beta_mcmc_thin)
prob_mcmc <- exp(logodd_mcmc)/(1 +  exp(logodd_mcmc))
prob_champ_pm <- apply(prob_mcmc, 1, mean)

## in-sample accuracy (with 0.5 as cutoff)
cutoff <- 0.5
champ_fit_pm <- (prob_champ_pm > cutoff)
mean(champ_fit_pm == Y)
```

```
## [1] 0.7291667
```

```
## plug-in estimate of prob champ
logodd_plug <- Xi %*% beta_pm_mcmc
prob_champ_plug <- exp(logodd_plug)/(1 +  exp(logodd_plug))

## in-sample accuracy (with 0.5 as cutoff)
cutoff <- 0.5
champ_fit_plug <- (prob_champ_plug > cutoff)
mean(champ_fit_plug == Y)
```

```
## [1] 0.7291667
```

Out-of-Sample Estimations → Generating Championship Probabilities

Can't compare to Current Sample because we don't have the winner yet

For a point prediction, we could use the posterior predictive probability $E(\tilde{Y}_i|\tilde{\boldsymbol{X}}_i, \boldsymbol{Y}, X) = E(\tilde{p}_i|\tilde{\boldsymbol{X}}_i, \boldsymbol{Y}, X)$. Based on MCMC samples, we could approximate the posterior distribution of $\tilde{p}_i|\tilde{X}_i, Y, X$.

```
## Data prep
X_tilde <- pres_test
Xi_tilde <- as.matrix(cbind(1, X_tilde))
#Y_tilde <- as.numeric(dat_test1$legendary)

## posterior of log-odds
logodd_pred_mcmc <- Xi_tilde %*% t(beta_mcmc)
logodd_champ_pred_pm <- apply(logodd_pred_mcmc, 1, mean)

odds_champ_pred_mcmc <- exp(logodd_pred_mcmc)
odds_champ_pred_pm <- apply(odds_champ_pred_mcmc, 1, mean)

prob_champ_pred_mcmc <- exp(logodd_pred_mcmc)/(1 +  exp(logodd_pred_mcmc))
prob_champ_pred_pm <- apply(prob_champ_pred_mcmc, 1, mean)


logodd_champ_pred_pm
```

```
##  [1]  1.9930110 -1.5484139  0.1348476 -0.6770062 -2.4369397 -1.6662740
##  [7] -0.9150924 -2.6159848 -1.0406977 -3.0746286 -0.6976569 -2.2915316
## [13] -2.8256529 -3.7435313 -2.9897539 -2.8538132
```

```
odds_champ_pred_pm
```

```
##  [1] 8.99089308 0.38123857 1.24386183 0.56272942 0.10095254 0.21818330
##  [7] 0.49952916 0.09007404 0.42348232 0.05961402 0.63903447 0.11912062
## [13] 0.06939705 0.03609527 0.06355695 0.07064403
```

```
length(prob_champ_pred_pm)
```

```
## [1] 16
```

```
prob_champ_pred_pm
```

```
##  [1] 0.86509564 0.21364725 0.53222392 0.34307684 0.08768291 0.17014174
##  [7] 0.30443733 0.07889678 0.27698926 0.05238148 0.35043460 0.10128411
## [13] 0.06181883 0.03111812 0.05683552 0.06299218
```

```
curr_vars <- c("Team", "Age", "W","netSFFR")
champ_probs_final <- pres_nba[,curr_vars]

champ_probs_final$Championship_LogOdds <- logodd_champ_pred_pm
champ_probs_final$Championship_Odds <- odds_champ_pred_pm
champ_probs_final$Championship_Probability <- prob_champ_pred_pm
champ_probs_final
```

```
##                       Team  Age  W netSFFR Championship_LogOdds
## 1          Boston Celtics* 28.2 64 0.03105            1.9930110
## 2   Oklahoma City Thunder* 23.4 57 0.02025           -1.5484139
## 3  Minnesota Timberwolves* 27.2 56 0.02245            0.1348476
## 4          Denver Nuggets* 27.1 57 0.01750           -0.6770062
## 5         New York Knicks* 26.4 50 0.00875           -2.4369397
## 6    New Orleans Pelicans* 26.0 49 0.01455           -1.6662740
## 7    Los Angeles Clippers* 30.4 51 0.00930           -0.9150924
## 8      Philadelphia 76ers* 28.4 47 0.00350           -2.6159848
## 9            Phoenix Suns* 29.3 49 0.01115           -1.0406977
## 10         Indiana Pacers* 25.3 47 0.00750           -3.0746286
## 11        Milwaukee Bucks* 30.2 49 0.01130           -0.6976569
## 12    Cleveland Cavaliers* 26.2 48 0.01030           -2.2915316
## 13       Dallas Mavericks* 26.5 50 0.00610           -2.8256529
## 14          Orlando Magic* 24.0 47 0.00620           -3.7435313
## 15             Miami Heat* 28.0 46 0.00215           -2.9897539
## 16     Los Angeles Lakers* 28.0 47 0.00290           -2.8538132
##    Championship_Odds Championship_Probability
## 1         8.99089308               0.86509564
## 2         0.38123857               0.21364725
## 3         1.24386183               0.53222392
## 4         0.56272942               0.34307684
## 5         0.10095254               0.08768291
```

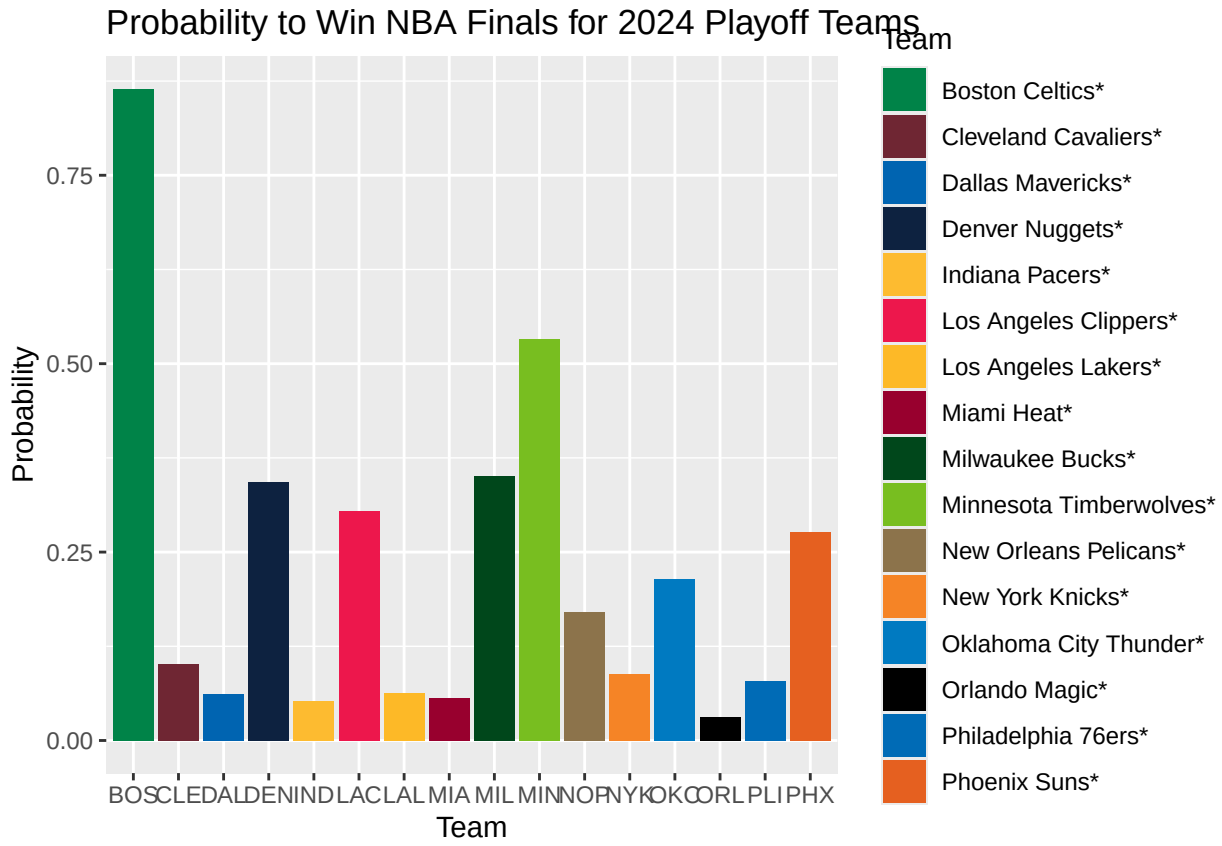
```
## 6         0.21818330              0.17014174
## 7         0.49952916              0.30443733
## 8         0.09007404              0.07889678
## 9         0.42348232              0.27698926
## 10        0.05961402              0.05238148
## 11        0.63903447              0.35043460
## 12        0.11912062              0.10128411
## 13        0.06939705              0.06181883
## 14        0.03609527              0.03111812
## 15        0.06355695              0.05683552
## 16        0.07064403              0.06299218
```

```
final_vars <- c("Team", "Championship_LogOdds", "Championship_Odds","Championship_Probability")
final_champ_table <- champ_probs_final[,final_vars]
final_champ_table
```

```
##                       Team Championship_LogOdds Championship_Odds
## 1          Boston Celtics*            1.9930110        8.99089308
## 2   Oklahoma City Thunder*           -1.5484139        0.38123857
## 3  Minnesota Timberwolves*            0.1348476        1.24386183
## 4          Denver Nuggets*           -0.6770062        0.56272942
## 5         New York Knicks*           -2.4369397        0.10095254
## 6    New Orleans Pelicans*           -1.6662740        0.21818330
## 7    Los Angeles Clippers*           -0.9150924        0.49952916
## 8      Philadelphia 76ers*           -2.6159848        0.09007404
## 9            Phoenix Suns*           -1.0406977        0.42348232
## 10         Indiana Pacers*           -3.0746286        0.05961402
## 11        Milwaukee Bucks*           -0.6976569        0.63903447
## 12    Cleveland Cavaliers*           -2.2915316        0.11912062
## 13       Dallas Mavericks*           -2.8256529        0.06939705
## 14          Orlando Magic*           -3.7435313        0.03609527
## 15             Miami Heat*           -2.9897539        0.06355695
## 16     Los Angeles Lakers*           -2.8538132        0.07064403
##    Championship_Probability
## 1                0.86509564
## 2                0.21364725
## 3                0.53222392
## 4                0.34307684
## 5                0.08768291
## 6                0.17014174
## 7                0.30443733
## 8                0.07889678
## 9                0.27698926
## 10               0.05238148
## 11               0.35043460
## 12               0.10128411
## 13               0.06181883
## 14               0.03111812
## 15               0.05683552
## 16               0.06299218
```

Summary Tables + Plots

```
plot_probs <- ggplot(data=champ_probs_final, aes(x=Team, y=Championship_Probability, fill=Team)) + geom_

labels <- c("BOS", "CLE", "DAL", "DEN", "IND", "LAC", "LAL", "MIA", "MIL",
                                        "MIN","NOP", "NYK", "OKC", "ORL", "PLI", "PHX")

plot_probs +scale_fill_manual(values=c("#008348", "#6F2633", "#0064B1", "#0D2240", "#FDBB30", "#ED174C",
                                       "#78BE20","#8C734B", "#F58426", "#007AC1", "#000000", "#006BB6",
  labs(x = "Team", y = "Probability") + scale_x_discrete(label = labels) +
  ggtitle("Probability to Win NBA Finals for 2024 Playoff Teams")
```



Metropolis-Hastings (Failed)

```
## M-H: independent proposal
##Prior
beta0 <- as.matrix(coef(fit_glm))
sigma2_beta_glm <- summary(fit_glm)$coefficients[,2]^2
sigma2_0 <- 5^2

##size
```

```r
n_mcmc <- 10000
burn <- 2000

##MCMC prep
beta_mh_all <- matrix(nrow=(n_mcmc+burn), ncol=p)
delta <- 1

##initialize
beta_init <- c(0,0,0,0)
beta_curr <- beta_init
```

```r
beta_curr <- beta_init
set.seed(465)
n_acc <- 0

for(s in 1:(burn+n_mcmc)){
  ##current
  loglik_curr <- loglik_lr(beta=beta_curr)
  logprior_curr <- sum(dnorm(beta_curr, mean=beta0, sd=sqrt(sigma2_0), log = T))
  logpost_curr <- loglik_curr + logprior_curr

  ##proposal
  beta_prop <- rnorm(n = p, mean=beta0, sd=sqrt(sigma2_beta_glm) * delta)
  loglik_prop <- loglik_lr(beta=beta_prop)
  logprior_prop <- sum(dnorm(beta_prop, mean=beta0, sd=sqrt(sigma2_0), log = T))
  logpost_prop <- loglik_prop + logprior_prop


  log_trans_ratio <- sum(dnorm(beta_curr, mean=beta0,
                               sd=sqrt(sigma2_beta_glm)*delta, log=T)) - sum(dnorm(beta_prop, mean=beta0
              sd=sqrt(sigma2_beta_glm)*delta, log=T))

  ##accept/reject
  u <- log(runif(1))
  if(u < (logpost_prop - logpost_curr + log_trans_ratio)){ ##accept
    beta_curr <- beta_prop
    n_acc <- n_acc + 1
  }
  beta_mh_all[s, ] <- beta_curr
}

##burn-in
beta_mh <- beta_mh_all[-c(1:burn),]
```

```r
rate_acc <- n_acc/(n_mcmc + burn); rate_acc
```

```
## [1] 0.004583333
```

```r
beta_all <- matrix(nrow=(n_mcmc + burn), ncol=p)
delta_all <- c(0.15, 0.11, 0.10)
rate_acc_all <- numeric(length(delta_all))
list_mcmc_all <- list()
```

```
for(id in 1:length(delta_all)){
  delta <- delta_all[id]
  ##initialize
  beta_curr <- beta_init

  ## keep track of the number of accepted proposals
  n_acc <- 0

  set.seed(465)
  for(s in 1:(burn+n_mcmc)){
    ##current
    loglik_curr <- loglik_lr(beta=beta_curr)
    logprior_curr <- sum(dnorm(beta_curr, mean=beta0, sd=sqrt(sigma2_0), log = T))
    logpost_curr <- loglik_curr + logprior_curr

    ##proposal
    beta_prop <- rnorm(n = p, mean=beta0, sd=sqrt(sigma2_beta_glm) * delta)
    loglik_prop <- loglik_lr(beta=beta_prop)
    logprior_prop <- sum(dnorm(beta_prop, mean=beta0, sd=sqrt(sigma2_0), log = T))
    logpost_prop <- loglik_prop + logprior_prop

    log_trans_ratio <- sum(dnorm(beta_curr, mean=beta0,
                                 sd=sqrt(sigma2_beta_glm)*delta, log=T)) - sum(dnorm(beta_prop, mean=beta0
              sd=sqrt(sigma2_beta_glm)*delta, log=T))

    ##accept/reject
    u <- log(runif(1))
    if(u < (logpost_prop -logpost_curr+log_trans_ratio)){ ##accept
      beta_curr <- beta_prop
      n_acc <- n_acc + 1
    }
    beta_all[s, ] <- beta_curr
  }
  ###burn-in
  #beta_mcmc <- beta_all[-c(1:burn),]
  list_mcmc_all[[id]] <- beta_all
  ##Acceptance Rate
  rate_acc <- n_acc/(n_mcmc + burn)
  rate_acc_all[id] <- rate_acc
}
```

```
## acceptance rates
names(rate_acc_all) <- delta_all
rate_acc_all
```

```
##    0.15    0.11     0.1
## 0.02425 0.04000 0.00000
```

M-H was attempted but we were unable to achieve an acceptance rate larger than 0.04.