Chance in Games

Sonya Eason and Sarah Ouda

Introduction

To determine how skill and chance play a role in team performance, and also predict who would win 2024 March Madness, we decided to take the following approach:

1) Identify a rational metric of team skill.
2) Model team wins as a function of skill, using Bayesian methods.
3) Measure how much wins can be explained by skill versus chance, and predict the likely tournament outcome.

> **Note**
>
> When teams win basketball games, is it because they're truly the best—or did luck play a role? Here, we try to measure skill in a fair way, use math to see how much skill predicts success, and then estimate how much "luck" is left over.

Data Used & Background Research

In addition to using the provided data from the 2024 season for all teams in the March Madness Tournament, we gathered historical game data for the 2022 and 2023 seasons as well (https://www.sports-reference.com/cbb/seasons/men/2022-school-stats.html). Our system for quanitifying skill was inspired by this article: https://www.siam.org/publications/siam-news/articles/feeling-lucky-the-relative-roles-of-skill-and-chance-in-sports/.

> **Note**
>
> Multiple years can help establish team skill over time, if such quantification is possible.

Identify a Rational Metric of Team Skill

Read in data

combining data

Df names: 2021-2022: combined_2021 2022-2023: combined_2022 2023-2024: combined_2023

Building a Skill Metric

skill task

To develop the skill metric, it was of utmost importance to determine whether a team performed well but also performed consistently. A good performance, could in theory appear to be based on skill, but it would be wrong to give skill points to a team if that appearingly skilled performance came out of a performance that was a high score based on luck. For this reason, we decided to utilize a skill metric that would give weight to consistently and performance. Consistency in this case can be considered to be a function of variance.

$$skill = consistency \ * performance$$

$$consistency = 1 - \frac{variance}{k}$$

Consider k to be a constant that normalizes so the consistency is relative across teams. We chose our normalizing constant to be the max variance a team had in performance metric.

There were two main metrics with which skill could be calculated: win fraction and point differentials. Win fraction refers to the amount of wins out of total games a team played in a given time interval. Point differentials refer to the value of the team's earned points minus their opponent's points per game. For example, if the team lead by 6 points, they would have a +6 point differential for a specific game but if they were behind by 4 points, they would have a -4 point differential.

$$skill = (1 - \frac{var(win \ fraction)}{max(var(winfraction))}) * mean(winfraction)$$

$$skill = (1 - \frac{var(point \ differential)}{max(var(point \ differential))}) * mean(point \ differential)$$

> **Note**
>
> Why consistency? A skilled team won't just win; they'll win often.

```
`summarise()` has grouped output by 'Team'. You can override using the
`.groups` argument.
```

Metropolis-Hastings Algorithm

We believe the probability of winning a game to be a function of skill. This is how we will predict who wins 2024. The other question we will answer is how much is skill explanatory of wins. We must start by building a model that quantifies skill in game wins and seeing what variability can not be explained by skill differential. For those other factors not accounted for are chance-based factors.

Since teams have different skill levels. It is also clear that the probability a team wins a game varies as a function of this skill level.

Consider the binary variable of a team winning a game. In this case, the probability a team wins a game, θ_t varies as a function of skill.

$$Y_i \sim binary(\theta_t)$$

We define this as below.

$$\theta_t = \frac{1}{1 + exp(\beta_0 + \beta_1 * x_i)}$$

x_i is the skill metric for team i.

y_i is the performance of team i, win 1, lose 1

β_1: a coefficient revealing how x_i impacts probability a team wins

$$p(y_1...y_n \mid x_1, ..., x_n, \beta_0, \beta_1) = \prod_{i=1}^{n} (\frac{1}{(1 + exp(\beta_0 + \beta_1 * x_i)})^{y_i} * (1 - \frac{1}{(1 + exp(\beta_0 + \beta_1 * x_i))})^{1-y_i}$$

We will define things on a log scale to avoid issues in numerical computation.

$$log(p(y_1...y_n \mid x_1, ..., x_n, \beta_0, \beta_1)) = \sum_{i=1}^{n} [\ y_i * log(\frac{1}{(1 + exp(\beta_0 + \beta_1 * x_i)}) + (1 - y_i)]$$

$$* \, log(1 - \frac{1}{(1 + exp(\beta_0 + \beta_1 * x_i))})$$

In order to build a distribution for $p(\beta_0, \beta_1 | \vec{y}, \vec{x})$, which is how we will land on the coefficients for our model, we need to set priors on the coefficients.

Since, we don't know much about skill's effect on performance, we will use diffuse priors. In this case, normal distributions with variance 1000.

$$p(\beta_0, \beta_1) = dnorm(\beta_0, 0, sqrt(1000)) * dnorm(\beta_1, 0, sqrt(1000))$$
$$log(p(\beta_0, \beta_1)) = log(dnorm(\beta_0, 0, sqrt(1000))) + log(dnorm(\beta_1, 0, sqrt(1000)))$$

We will chose a symmetric target distribution for sampling. If we choose a target that is irreducible, aperiodic, and recurrent, Ergodic theorem promises that our Markov chain will yield a stationary target distribution which is in this case a posterior defined by $p(\beta_0, \beta_1 | \vec{y}, \vec{x})$.

To determine if we accept a proposed value, we will utilize the following acceptance rate.

$$\frac{p(\beta_0^*, \beta_1^s |\ y_1, ..., y_n,\ x_1, ..., x_n)}{p(\beta_0^s, \beta_1^s |\ y_1, ..., y_n,\ x_1, ..., x_n)}$$

$$\frac{p(\beta_0^{s+1}, \beta_1^s |\ y_1, ..., y_n,\ x_1, ..., x_n)}{p(\beta_0^{s+1}, \beta_1^* |\ y_1, ..., y_n,\ x_1, ..., x_n)}$$

Since we do not have posteriors yet, we will weaponize the fact that this ratio is equivalent to the likelihood times posterior ratio.

$$\frac{posterior(new\ state)}{posterior(old\ state)} = \frac{prior(new\ state) * likelihood(new\ state)}{prior(old\ state) * likelihood(old\ state)}$$

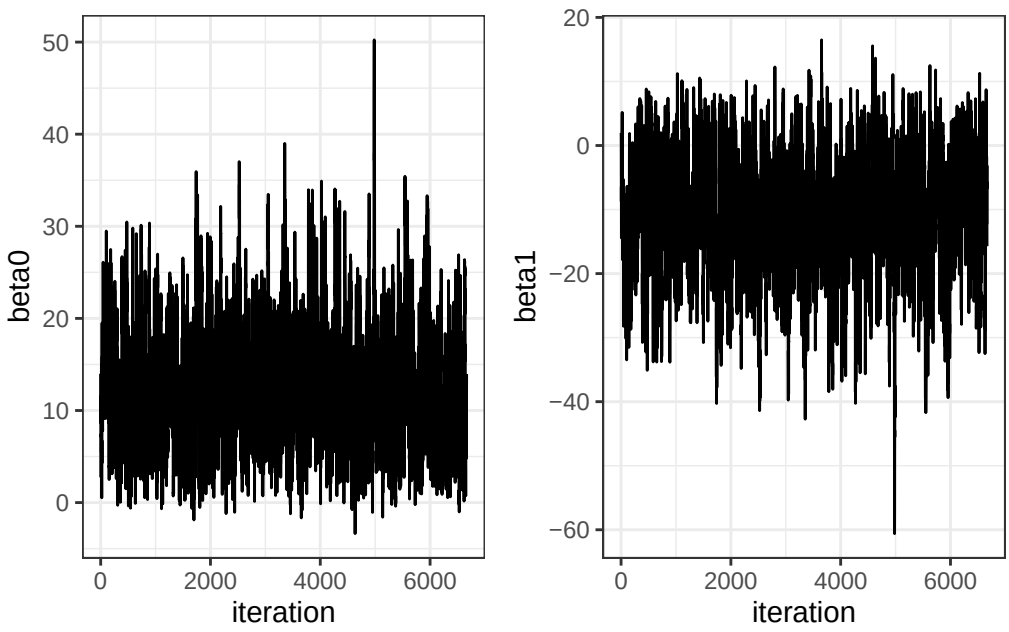
Below we do Metropolis Hastings to determine the posterior representing coefficients in the model that considers point differentials instead of win-fractions.

> **Note**
>
> What's Metropolis-Hastings? Imagine we pick random shirts until we find the shirt you wear the most. It's a process with mathematical guarantees that we use here.

```r
set.seed(4)

logLikelihood = function(beta0, beta1) {
  l = 1 + exp(beta0 + (beta1 * x))
sum(y*log(1/l)+(1-y)*log(1-1/l))
}

logPrior = function(beta0, beta1) {
  dnorm(beta0, 0, sqrt(1000), log = TRUE) +
    dnorm(beta1, 0, sqrt(1000), log = TRUE)
}

logPosterior = function(beta0, beta1) {
  logLikelihood(beta0, beta1) + logPrior(beta0, beta1)
}


BETA0 = NULL
BETA1 = NULL


accept1 = 0
accept2 = 0

y <- rep(0, 68)  # Create a vector of zeros
y[59] <- 1  # Set the 59th element to 1
x = skill
S = 1000000

beta0_s = 0
beta1_s = 0
for (s in 1:S) {

  ## propose and update beta0
  beta0_proposal = rnorm(1, mean = beta0_s, 1)
   log.r = logPosterior(beta0_proposal, beta1_s) -
     logPosterior(beta0_s, beta1_s)

   if(log(runif(1)) < log.r)  {
    beta0_s = beta0_proposal
    accept1 = accept1 + 1
```

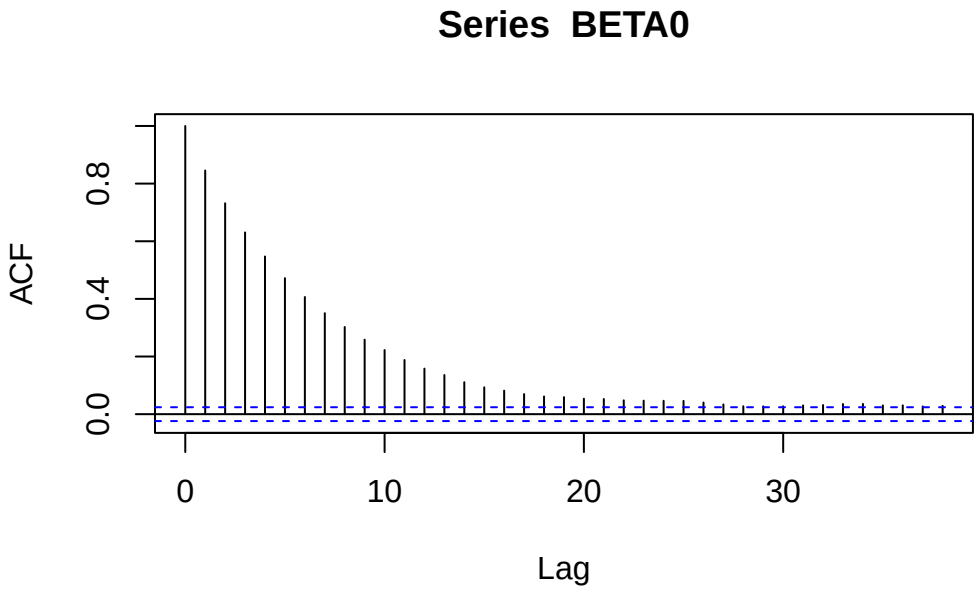
```
  }

      if (s%%150 == 0)
   {
        BETA0 = c(BETA0, beta0_s)
   }
  ## propose and update beta1
   beta1_proposal = rnorm(1, mean = beta1_s, .5)
  log.r = logPosterior(beta0_s, beta1_proposal) -
    logPosterior(beta0_s, beta1_s)

  if(log(runif(1)) < log.r)  {
   beta1_s = beta1_proposal
   accept2 = accept2 + 1

  }
      if (s%%150 == 0)
   {
        BETA1 = c(BETA1, beta1_s)
   }



}
```

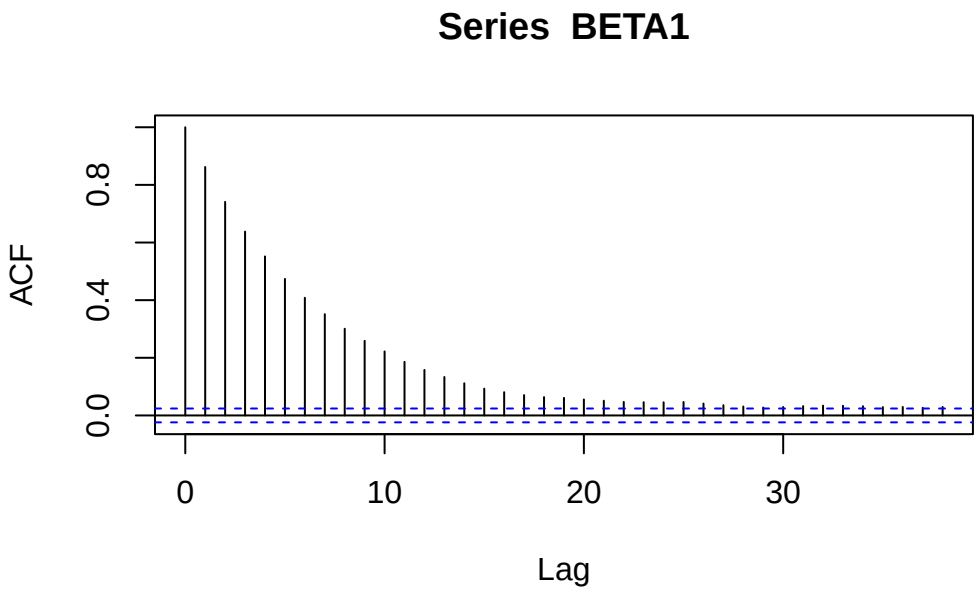
```
   BETA0    BETA1
495.4416 493.5512
```

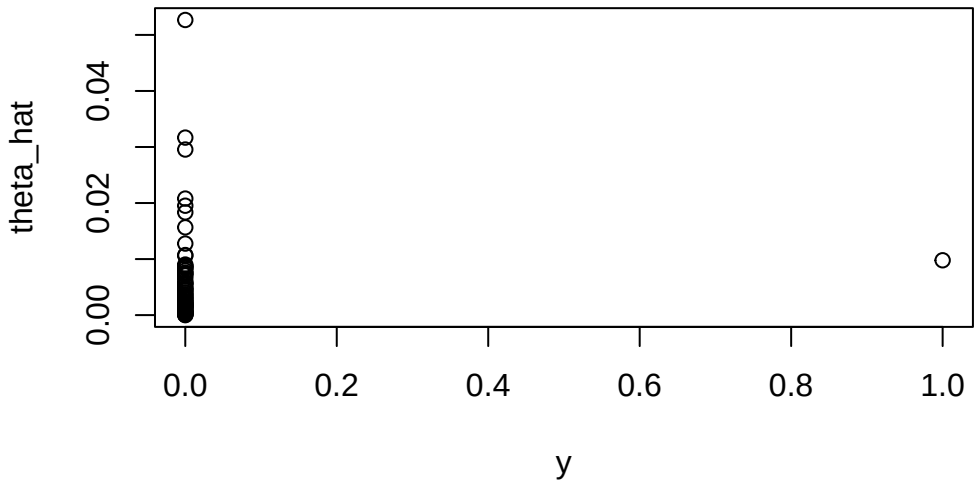



```
[1] 12.27676
```

```
[1] -10.90221
```





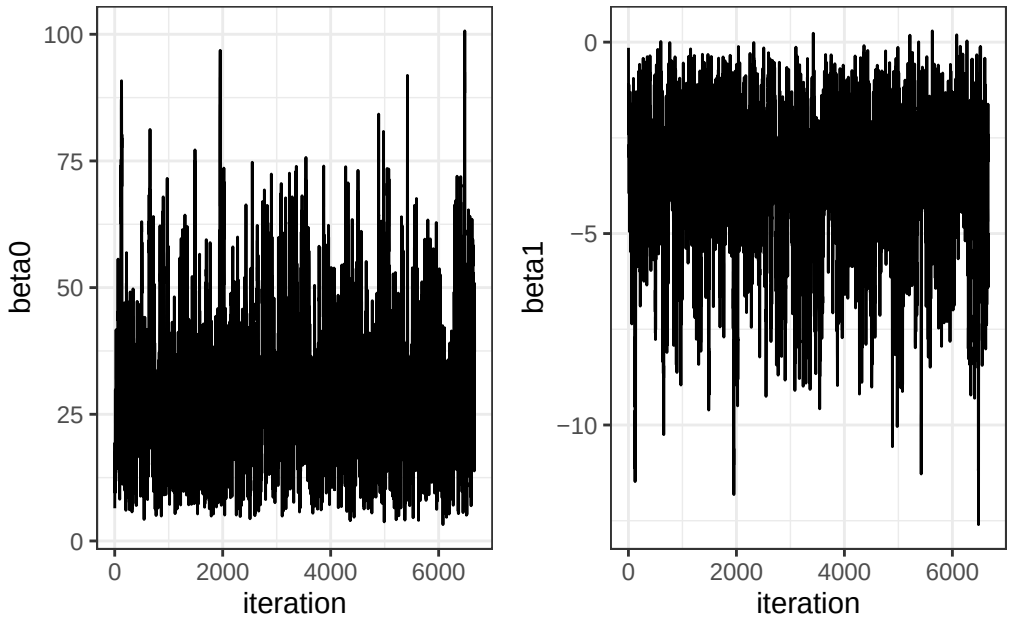


When the skill metric that's based on win fraction instead of point differential is used in our model, we found that many other teams had higher probabilities of winning March Madness 2024 that the true winner UConn. The results appear better for the point differential based skill metric that's detailed below.

```r
set.seed(4)

logLikelihood = function(beta0, beta1) {
  l = 1 + exp(beta0 + (beta1 * x))
sum(y*log(1/l)+(1-y)*log(1-1/l))
}

logPrior = function(beta0, beta1) {
  dnorm(beta0, 0, sqrt(1000), log = TRUE) +
    dnorm(beta1, 0, sqrt(1000), log = TRUE)
}

logPosterior = function(beta0, beta1) {
  logLikelihood(beta0, beta1) + logPrior(beta0, beta1)
}


BETA0 = NULL
BETA1 = NULL


accept1 = 0
accept2 = 0

y <- rep(0, 68)  # Create a vector of zeros
y[59] <- 1  # Set the 59th element to 1
x = skill_pt
S = 1000000

beta0_s = 0
beta1_s = 0
for (s in 1:S) {

  ## propose and update beta0
  beta0_proposal = rnorm(1, mean = beta0_s, 1)
   log.r = logPosterior(beta0_proposal, beta1_s) -
     logPosterior(beta0_s, beta1_s)

   if(log(runif(1)) < log.r)  {
    beta0_s = beta0_proposal
    accept1 = accept1 + 1
```

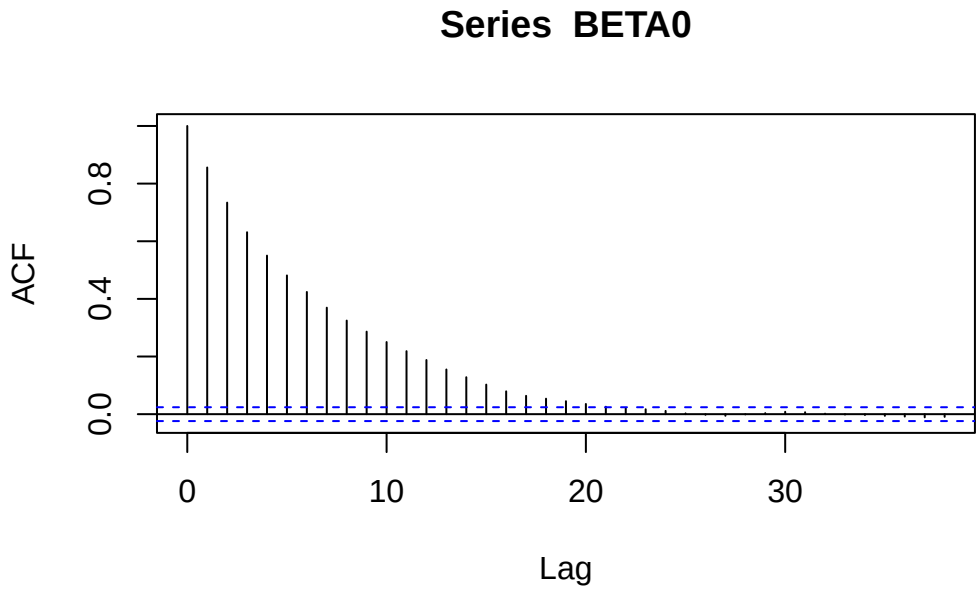
```
  }

      if (s%%150 == 0)
   {
        BETA0 = c(BETA0, beta0_s)
   }
  ## propose and update beta1
   beta1_proposal = rnorm(1, mean = beta1_s, .5)
  log.r = logPosterior(beta0_s, beta1_proposal) -
    logPosterior(beta0_s, beta1_s)

  if(log(runif(1)) < log.r)  {
   beta1_s = beta1_proposal
   accept2 = accept2 + 1

  }
      if (s%%150 == 0)
   {
        BETA1 = c(BETA1, beta1_s)
   }



}
```

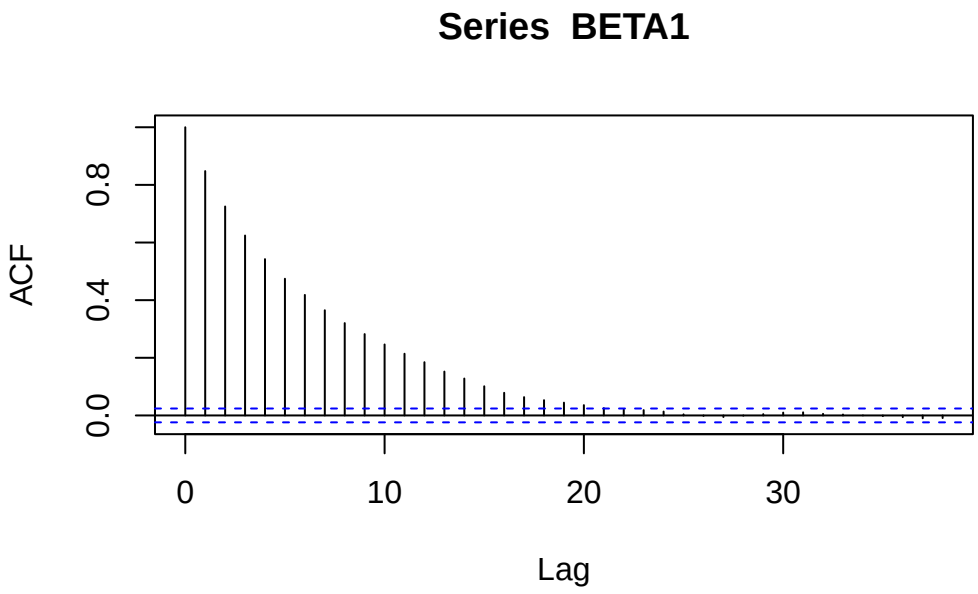
```
   BETA0    BETA1
517.1135 525.0017
```

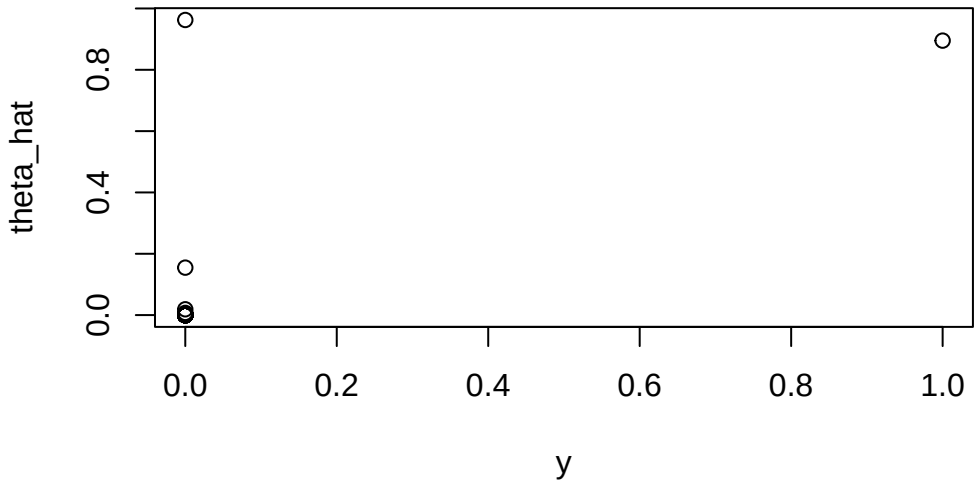



```
[1] 37.25832
```

```
[1] -4.9934
```




Series BETA1
ACF
0.0
0.4
0.8
0
10
20
30
Lag


theta_hat
0.0
0.4
0.8
0.0
0.2
0.4
0.6
0.8
1.0
y

Model Results

> **Note**
>
> We fit our model twice: Once with win-fraction-based skill metric. Once withthe point-differential based metric. We found that skill measured by point differential better matched who actually won March Madness 2024 (UConn was out model's #2 pick; they won).

We assessed the accuracy of our model by determining if the coefficients yielded a probability of a team's success that matched the data we observed, this model shows that UConn had the second highest probability of winning according to our skill-based model. In real life, UConn placed first as the winner of the 2024 March Madness tournament.

Our final model

To model whether a team won.

$$Y_i \sim binary(\theta_t)$$

$$\hat{\theta_t} = \frac{1}{1 + exp(37.258 - 4.9934 * x_i)}$$

where x_i is the skill metric based on point differential.

> **Note**
>
> Here, we've effectively quantified what magnitude our skill metric relates to team wins.