THE PERFECT BRACKET
A Mathematical Exploration of March Madness

November 17, 2024

Charles Tang, Curtis Ying, Jaeho Lee
Brown University

Problem Choice	2024 Brown MCM Summary Sheet	Team Number
March Madness		01

Summary

The perfect bracket in the annual March Madness tournament is highly sought after in the college basketball community. The March Madness modeling problem consists of two main tasks: (1) developing a model to predict the winning team of a March Madness match-up and (2) predicting the outcome of the March Madness tournament (i.e. the full bracket).

To model the winning team of any March Madness match-up, we developed a *Strength Index Model* which incorporates key statistics of a basketball team such as point differentials and historical wins and losses. We decided to use a probabilistic approach in modeling the strength indices of each team to introduce a component of randomness by treating each team's ability as a normally distributed random variable. Thus, we could model probability distributions of a team's performance in any given matchup.

After we assigned a Strength Index distribution to each team, we applied the *Bradley-Terry Model*, which allows for predictions of winners in two-team match-ups based on an ability score for each team. To model luck, we followed a method where samples were taken from strength distributions to model game outcomes. To simulate an entire March Madness tournament, we created an iterative method to update strength indices based on a team's performance in each round.

After we constructed our model, we ran a statistical analysis on the historical data of each basketball team in the 2024 March Madness tournament, which allowed us to calculate a strength index for each team. Our results show that a team's strength index is highly correlated with a team's NCAA-designated seed.

After running 100+ simulations of March Madness tournaments based on our iterative model, we converged to an expected solution that predicts 80% of all matches correctly, which is better than the baseline average of 66% by the average human and 77% for machine learning techniques in the literature. Our model successfully predicted several upset games, where a team of worse strength beats a team of projected greater strength. Furthermore, our model successfully projected both finalist teams, as well as the result (UConn winning over Purdue). We further analyzed the output variance across simulations, showing that teams with higher seeds and strength indices tend to place higher.

Lastly, to test the validity of our model, we employed a Monte Carlo sampling simulation ($\pm 10\%$ variation of parameters, 4,000 total simulations) and a sensitivity analysis of parameters such as team strength (skill) and variance (luck), which revealed that variance significantly impacts the number of projected upsets in a tournament.

Although there exist shortcomings primarily due to the lack of data and the naturally unpredictable nature of March Madness, our probabilistic model thoroughly predicts March Madness outcomes and is easily adaptable to small amounts of data.

Keywords: March Madness, statistical analysis, Bradley-Terry model, point differentials, variance, Monte Carlo simulation

Contents

1 Introduction

1.1 Background

The March Madness Tournament is the NCAA's annual Division 1 college men's basketball tournament. The tournament begins with 68 qualifying collegiate teams, which have been ranked, or seeded, based on their performance during the season leading up to the tournament [1]. The tournament itself uses a single-elimination format, meaning if a team loses in its bracket, it is automatically eliminated. Therefore, the national champion title is highly contested, as March Madness is one of the biggest annual sporting events in the USA.

It is notoriously difficult to predict March Madness brackets and outcomes for an upcoming year. When someone correctly predicts the outcome of every game in the entire tournament, it is deemed a "perfect bracket." It is so difficult to construct a perfect bracket ($\mathbb{P}$(Perfect Bracket) = 1 in 9.2 quintillion that Warren Buffet offered a $1 billion prize to any perfect bracket predictor [4]. See Figure 1 for an example bracket.

When predicting March Madness brackets, it is necessary to understand some basketball vernacular. In particular, sports analytics firms and enthusiasts predict March Madness brackets with historical statistics of college teams. Predicting brackets is so popular that the discipline coined a new term, "bracketology." Some of the most common statistics of these teams are:

- Point differential – total points scored minus points allowed [2].
- Historical win rate – the win rate of a team against historical teams over the past year during the regular season.
- Team seed – the starting "seed" of a school's team is determined by the NCAA committee based on a variety of factors, and greatly affects their starting matchup [16].

However, the tournament has a great reputation for unpredictability, with lower-seeded teams often upsetting higher-seeded teams. Using the NCAA's definition of an upset, in which the winning team was seeded at least five seed lines worse than the losing team, there have been an average of 8.5 upsets per year the tournament has been held [6, 15]. As such, predicting the outcomes of even an individual game is extremely challenging. Thus, understanding the impact of skill and luck on a team's performance is key to more accurately predicting future game and tournament outcomes.

1.2 Problem Restatement

The problem presented has the following requirements:

1. Construct a mathematical model that predicts the winning team based on a combination of luck and skill-based factors.
2. Analyze the historical margins of victory and evaluate whether it impacts model predictions of game outcomes.
3. Use the developed mathematical model to predict the outcome of the 2024 March Madness Tournament.

After developing and testing the model, we are asked to present our findings to two audiences:

1. Write a technical report to explain the model and findings to our newspaper's analytics team.
2. Write a one-page letter to the newspaper's chief editor, explaining our report's main results and findings that can be shared with basketball fans reading the newspaper.

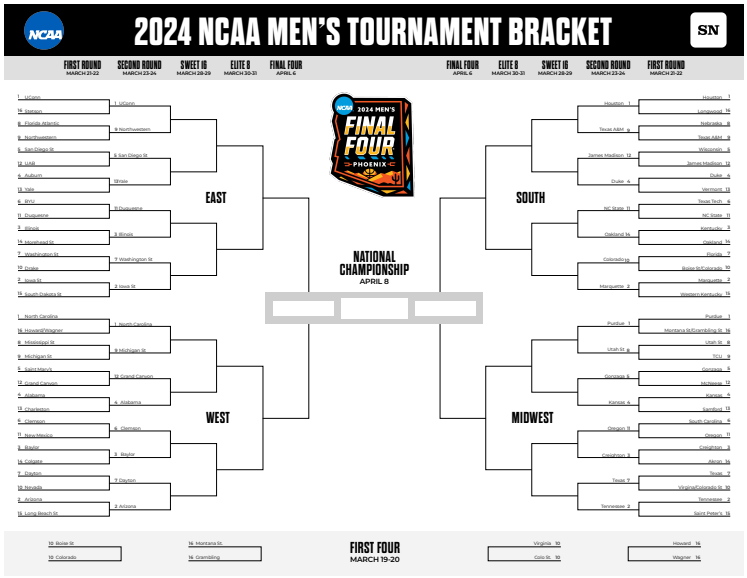


Figure 1: **March Madness bracket for 2024.** An example March Madness bracket.

2 Modeling March Madness Brackets

2.1 Problem Analysis

Part 1 of the problem tasks us with building a mathematical model that predicts the winning team from a basketball game of two teams in March Madness based on luck and skill-based factors. To accurately reflect the skill of a college basketball team in our model, we considered several major factors directly associated with game performance as described in the background section.

Since predicting winning teams aligns closely with estimating the probability of one team beating another, we use a probabilistic approach to model initial team strength before March Madness.

2.2 Assumptions

#	Assumption	Justification
1	Historical data is a good predictor of the future.	To predict the future performance of a team, we must use historical data to provide an estimate of a team's strength [17].
2	Luck is modeled by variance.	To account for luck, a team's strength (probability of winning) is modeled by a random variable.
3	A team's strength can be modeled as a random variable.	This allows us to simplify a team's ability to a probability distribution, which can therefore be more easily compared with other teams.
4	A March Madness bracket has 64 teams	This assumption helps simplify our simulation such that it is a perfect 64-team bracket.
5	A team's luck (variance) is round-independent.	To sample from a team's ability in a given round, we assume variance stays constant [6, 15].

2.3 Brief Overview

Below is a table of variables used throughout our model.

Variable	Symbol	Description
Round	t	Each round is either 1, 2, ... in the March Madness bracket.
Strength Index	$\theta_{i,t}$	The strength index random variable for team i at round t.

2.4 Strength Index

This section rests upon Assumption 1—that historical performance in basketball is a good predictor of future performance. We can use game data from the regular season to predict performance in the March Madness tournament. The problem statement also encouraged us to incorporate the margin of victory as a primary consideration in our model, especially when attempting to differentiate between skill and luck.

Accordingly, we used the **Point Differential** statistic in regular season games as a central consideration when determining our initial strength index. The formula for calculating Point Differential is shown in Equation 1.

$$\textbf{Point Differential (PD)} = \text{Points Scored (PS) - Points Allowed (PA)} \tag{1}$$

In essence, Point Differential (PD) captures the net scoring ability of a team, factoring in both offensive and defensive performance without the need for more granular data. A team with a high Point Differential has demonstrated both the ability to score and the capacity to prevent opponents from scoring, signaling superior overall performance. This measure is a stronger indicator of team strength than just win-loss records, as it accounts for the margin by which teams win or lose games, offering a clearer picture of a team's potential for success in the high-stakes environment of the March Madness tournament [7].

However, only calculating point differential as a total across the season would not be sufficient; it would fail to account for the fact that we are aiming to predict the outcome of a single basketball game. There is naturally some variance involved in each game's outcome, and a team's performance can fluctuate due to a variety of luck-based factors. To better capture this variance (see Assumption 2), we investigated the mean and standard deviation of both points scored per game (PS) and points allowed per game (PA) for each team i throughout the regular season.

$$\text{PA}_i \sim N(\mu_{\text{PA}}, \sigma^2_{\text{PA}})$$
$$\text{PS}_i \sim N(\mu_{\text{PS}}, \sigma^2_{\text{PS}})$$

Next, we assumed that PS and PA for each team follow a normal distribution based on the past year of data by the Law of Large Numbers. This assumption is reasonable provided that teams play a large number of games in the regular season (almost always above 30, for each team).

In a single game, by definition, some team i wins if their point differential is positive. This only happens when they score more points than they allow: that is, $PS_i > PA_i$, or $PS_i - PA_i > 0$. Therefore, we can rewrite this as

$$N(\mu_{\text{PS}}, \sigma^2_{\text{PS}}) - N(\mu_{\text{PA}}, \sigma^2_{\text{PA}}) > 0$$

In this case, our pooled standard deviation would be $\sqrt{\sigma^2_{PS} + \sigma^2_{PA}}$. Then, a team wins a game if

$$N(\mu_{PD}, \sigma^2_{PS} + \sigma^2_{PA}) > 0$$

Let $\theta_{i,0}$ be the initial Strength Index distribution for team i in round 0 of a March Madness tournament. Then, from our previous conclusions, we have

$$\theta_{i,0} = \mathbb{P}\left(N(\mu_{PD}, \sigma_{PS}^2 + \sigma_{PA}^2) > 0\right) \tag{2}$$

where θ_i is the calculated likelihood of team i winning any given game.

Due to the complexity of finding a confidence interval for θ_i analytically, we obtain a variance estimate by sampling. See examples of these strength distributions for different teams in Figure 2.

To solidify our claims that our strength index is a good measure of team ability (Equation 2), we obtain a Pearson Correlation coefficient of $\rho = -0.625$ when measuring the relationship between our determined strength index and NCAA's pre-determined seed-based ranking system. This proves that our initial strength estimations are closely in line with the industry standard measurement of a team's strength.

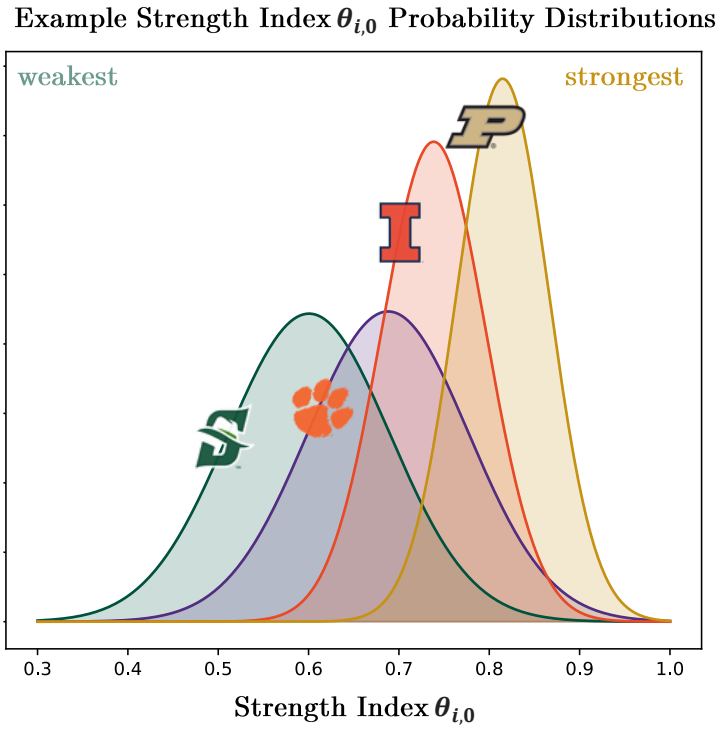


Figure 2: **Example initial strength index distributions.**

In summary, we use team statistics to approximate a probability distribution for each team's strength, as shown in Figure 3.

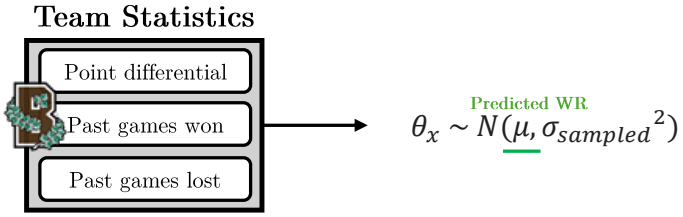


Figure 3: **Method to calculate initial team strengths pre-bracket.**

2.5 Bradley-Terry Model

Provided that we have just calculated the initial strength index distributions for each team at round 0 of March Madness, $\theta_{i,t=0}$, we want to measure two main processes:

a. How can the *strength index distributions* be used to predict a winning team?

b. How can we model the *change in strength* of a team over time?

Note that strength indices for each team are probability distributions, and we will sample from these distributions rather than comparing expected values to simulate the "luck" part of March Madness games. Therefore, teams with strength indices with greater variance tend to have a greater chance of good and bad luck in their games.

To address part (a), we want to identify a method to compare two strength indices, say $\theta_{x,t}$ and $\theta_{y,t}$ for teams x, y respectively on round t. By the Bradley-Terry model, which is most commonly used for predicting sports game outcomes and comparing the strength of two teams, we can express the following [9].

$$\mathbb{P}(x \text{ beats } y \text{ at round } t) := \frac{\exp(\theta_{x,t})}{\exp(\theta_{x,t}) + \exp(\theta_{y,t})} \tag{3}$$

In the original Bradley-Terry model, for rating systems with ELO scores (elo $\in [0, 400]$), it is defined that for players i, j, the probability that player i can beat player j is

$$\mathbb{P}(i > j) = \frac{\exp(\frac{\text{elo}_i}{400})}{\exp(\frac{\text{elo}_i}{400}) + \exp(\frac{\text{elo}_j}{400})}$$

Intuitively, the Bradley-Terry model is a ratio between the strength of a team x and the combined strength of teams in their match-up [3]. This is ideal for modeling projected win percentages because

$$\begin{aligned}\mathbb{P}(x \text{ beats } y \text{ at round } t) + \mathbb{P}(y \text{ beats } x \text{ at round } t) &= \frac{\exp(\theta_{x,t}) + \exp(\theta_{y,t})}{\exp(\theta_{x,t}) + \exp(\theta_{y,t})} \\ &= 1\end{aligned}$$

In deriving this model, we altered the original Bradley-Terry model (Equation 3) since our strength index is scaled such that $\theta_{i,t} \in [0, 1]$, so we do not have to scale our indices [10].

We use an exponential transformation of each strength index in the Bradley-Terry equation to allow for a greater probability of "upsets" in later rounds. See the model application and sensitivity analysis sections as follows for details [10].

To simulate an outcome of a game between x and y on round t, we must sample from $\theta_{x,t}$ and $\theta_{y,t}$ which would give us a single sample of the probability. In our Bradley-Terry model, we sample from the distributions rather than calculating the expected value to preserve the "luck" aspect of modeling game outcomes.

Once obtaining samples of $\theta_{x,t}$ and $\theta_{y,t}$, we determine the winner of a game as the team with a probability of winning greater than 0.5. Then, we iteratively update the strength distribution of the winner to be more relative to all teams in the previous round's bracket. Without loss of generality, assume team y won the bracket of x versus y. Then, we would recalculate the winner's strength index distribution's mean in round $t + 1$ as

$$\theta_{y,t+1} = \mathbb{E}\left[\mathbb{P}\left(i \text{ beats } j \text{ on round } t\right) \mid i = y\right] \tag{4}$$

where we assume variance of $\theta_{i,t}$ is time-independent according to Assumption 5. This recalculation of strength indices is applied to every winning team of match-ups in round t entering round $t + 1$. See Figure 4 for a visualization of this process for each single match-up. We perform this recalculation step because we hope that it can address some faults of other similar models, which are unable to adapt to new rounds and use inaccurate strength estimates. [3, 8].

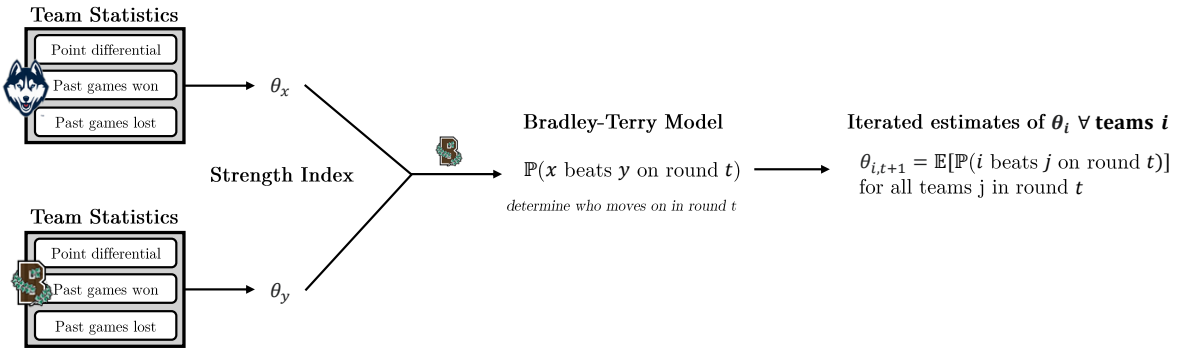
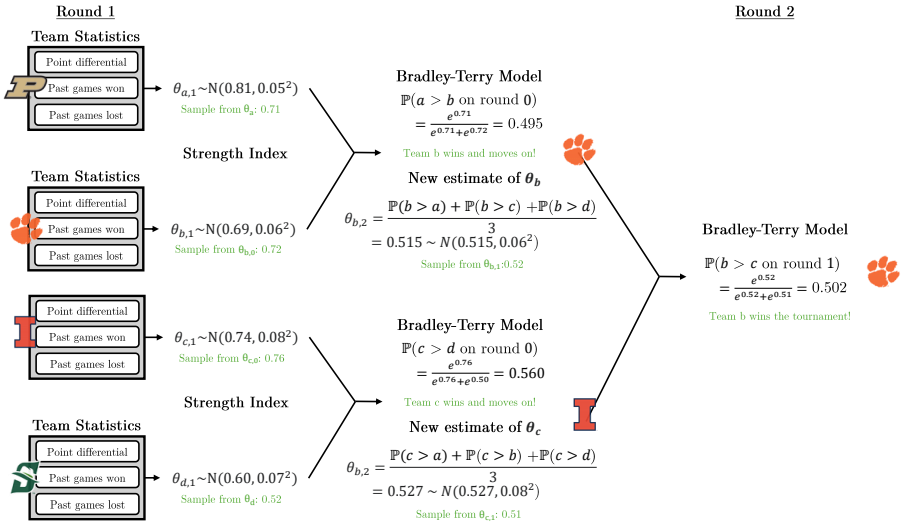


Figure 4: **Overview process of predicting a single March Madness bracket.**

2.6 Example Bracket

Example: Simple 4-Team Bracket

In this example, we annotate a 4-team bracket to illustrate how our model can iteratively update and project winning teams in a bracket system.



We approach this example in several steps:

1. Calculate initial strength indices for each team.
2. Sample from each strength index distribution, and use the sampled value in the Bradley-Terry model to calculate the probability of winning for each team. This accounts for the luck-based factor based on each distribution's unique variance.
3. Advance teams with the probability of winning greater than 0.5 (Clemson, UIUC). Recalculate strength indices for these teams.
4. Repeat steps 2-3 until the bracket is filled.

One key takeaway from this example scenario is that as rounds continue, the strength indices converge towards 0.5, modeling greater uncertainty in predicting matchups in higher rounds. For the actual model, we run the same process on 64 teams. Each run is a single simulation, and we run multiple simulations to measure the variance of placements and expected outcomes.

3 Application of March Madness Model

To apply our March Madness model to the 2024 bracket, we must first obtain each team's starting $\theta_{i,0}$ strength index distribution. Following the procedure as derived above, we arrive at $\theta_{i,0}$ values as shown below for each of the participating 64 teams in March Madness (see Assumption 4). The data provided on historical wins and losses was used to obtain these values.

3.1 2024 Bracket Prediction

Based on each of the strength indices, we pre-load the bracket with the initial 64 teams and their positions in the elimination bracket. See Figure 11 in the Appendix for the strength distribution calculations for the first round.

Since in every simulation, we sample only once from each strength index distribution in every match-up to determine the winner, we must simulate every match-up numerous times (or use the expected value) to extract the most common filled-out bracket for the 64-team elimination.

After 100 simulations, we arrived at the following full bracket as the converged prediction, provided that we only knew information for first-round match-ups.

Figure 5: **Converged prediction to 2024 March Madness.** The most common bracket after 100 simulation runs.

To answer the question of how our model performs on single match-ups, below is a distribution of how many "first-round" match-ups we correctly predicted over the 100 simulations. Our model boasts an 80.4% **overall accuracy** for the entire bracket (see Figure 6), counting correct positions as correct placements in each bracket. Compared with the NCAA's calculated average for bracket accuracy (66.7% [14]) based on people who submit brackets and machine learning techniques from the literature (77% [5]), our model outperforms other bracket-prediction methodologies. Furthermore, we can provide a richer analysis of our model's variance and ability to predict on a round-by-round basis as discussed below.

On the other hand, significant room exists for improvement (currently 70% accuracy) in predictions for first-round match-ups, which most bracketology experts claim have the least variability [6]. Intuitively, improving first-round match-up predictions can significantly improve future-round predictions, since incorrect answers will carry over into subsequent match predictions.

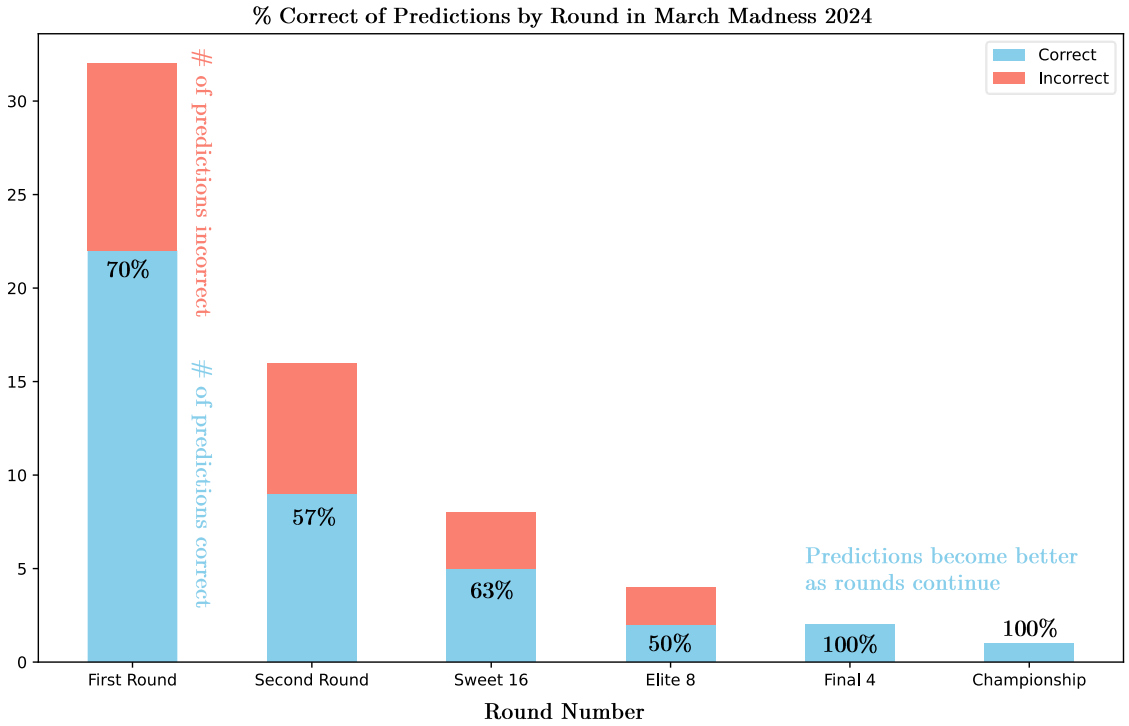


Figure 6: **Proportion correct for predictions in each round.**

3.2 Predicting "Upsets"

What is more interesting is simulating and identifying the most probable "upsets" in match-ups. In our model, we define an upset as a game won by a team with a lower strength distribution mean compared to another team, regardless of the strength we sampled from both distributions.

From the previous 100 simulations, below is a visualization of the most probable Round 1 upsets of the 2024 March Madness tournament. The following figure (Figure 7) shows the most probable upsets.

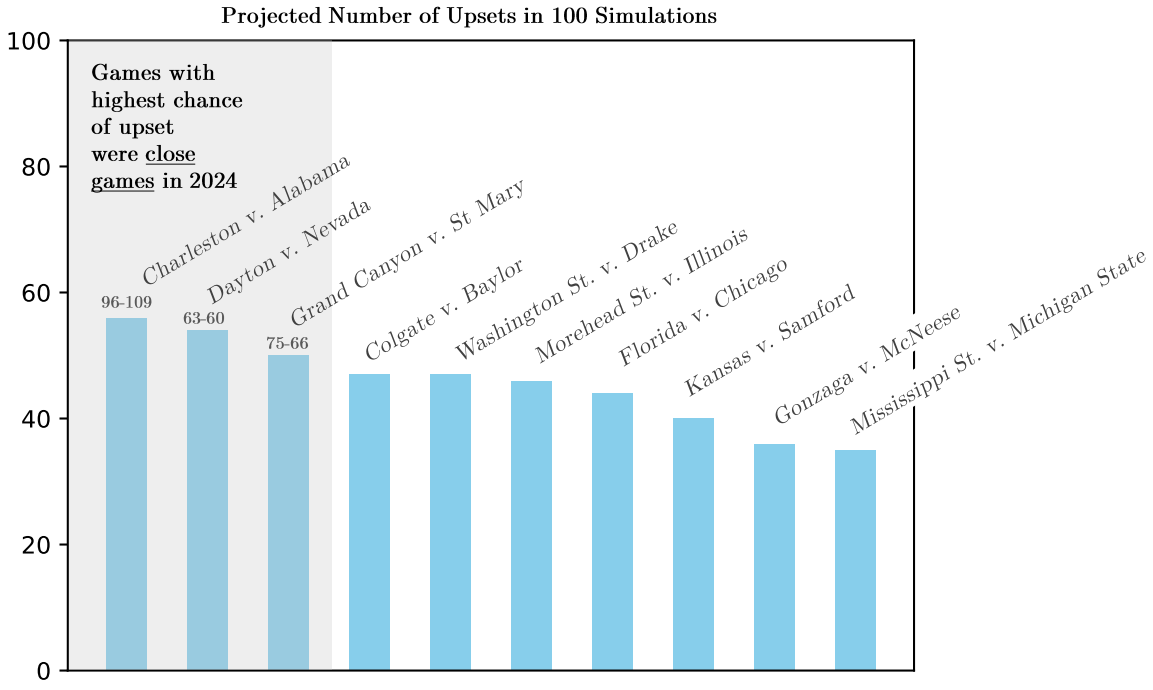


Figure 7: **Most common "upset" match-ups in simulation.** Of 100 simulations run, these most common upsets were found.

The most common upsets from our simulation reflect the actual outcomes of the matches from March Madness. The most frequent upsets in the simulation resulted from games where two teams had similar mean strength evaluations and sufficiently large standard deviations in their normal distributions, causing the lower-strength team to often perform at a higher level than the higher-strength team based on our definition of "luck."

The games that were projected to have the most probable upsets had very close scores in their actual games in 2024. In Figure 7, we list the scores of games in 2024 with the most probable upsets and the corresponding match-up.

We also want to examine the variability of the outcome placements of the top 10 seeded teams across our 100 simulations. We examine the top 10 seeded teams (Figure 8) because they are most commonly found in the highest placements by the end of the tournament. Note that we can also easily expand this examination to any top X seeds.

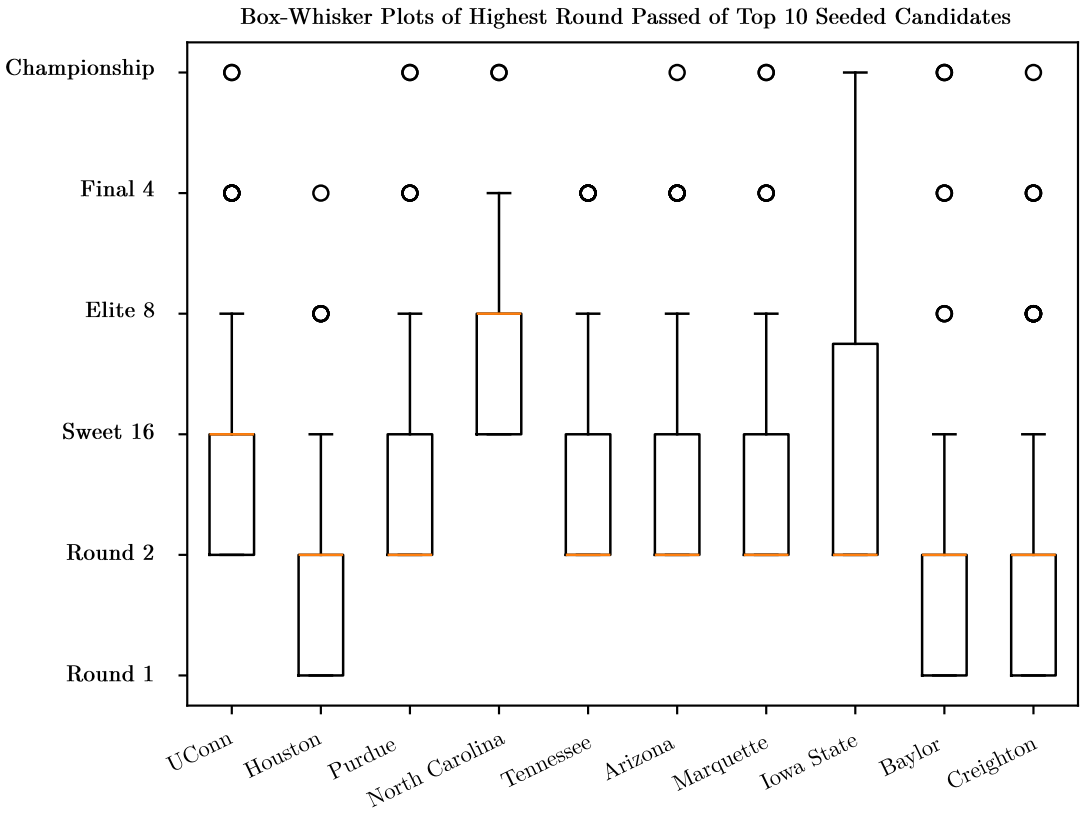


Figure 8: **Box-whisker plots of highest round reached by top 10 seeded teams.** Of 100 simulations run, these plots measure the variance in final placements (highest round passed).

Figure 8 follows our expectations of performance, where UConn and North Carolina (UNC) have the greatest mean of the highest round passed and Iowa State has the greatest variance in performance [12, 13, 11]. Given that UConn and UNC were both seeded first in their respective regions and often had decisive matches (where their strengths were significantly greater than their opponents') for the first few rounds, it is expected that they would often reach the later rounds of the tournament.

4 Model Discussion

4.1 Model Parameters

Below is a table of parameters we conducted a sensitivity analysis on.

Parameter	Symbol	Description	Default Values
Strength Mean	μ_i	Average of team i's initial strength distribution	Provided data.
Strength Standard Dev	σ_i	Standard deviation of team i's initial strength distribution	Provided data

4.2 Sensitivity Analysis

To analyze the effect of altering initial conditions on potential match-up predictions and bracket results, we employed a Monte Carlo simulation to test different initial strength indices and measure the change in potential placements for the overall top 10 seeded teams. The top 10 seeds (out of 64 teams) generally place highest and have some of the highest strength ratings. As such, we seek to examine how variations in "skill" and "luck" levels could contribute to a team's overall performance, with results being clearest for teams that often dominate the tournament. Note that the methods we employ below can be extended to the rest of the teams.

To precisely determine how variations in a team's skill and luck affect their performance in the tournament brackets, for each Top 10 seeded team, we simulated brackets while only changing either the mean or standard deviation of *one* specific team. This would allow our results to clearly reflect the change in performance, which would be confounded if the other 9 teams also had varied performances in the same simulation. Thus, for each of the Top 10 teams, each of the following variations was implemented and the distribution of placements over 100 simulations was recorded: up to 10% increase in mean strength, up to 10% decrease in mean strength, up to 50% increase in strength variance, and up to 50% decrease in strength variance. Since variance typically has less of an impact on final-round placements, we test a greater change.

In essence, the goal of this sensitivity analysis is to uncover how a team's initial strength and luck-based variance affect its ability to make it past later rounds.

Since we are running a Monte Carlo simulation for each of the top 10 seeded teams, with four possible scenarios (increase mean, decrease mean, increase variance, decrease variance), we run a total of **4000** simulations to examine the sensitivity of our model.

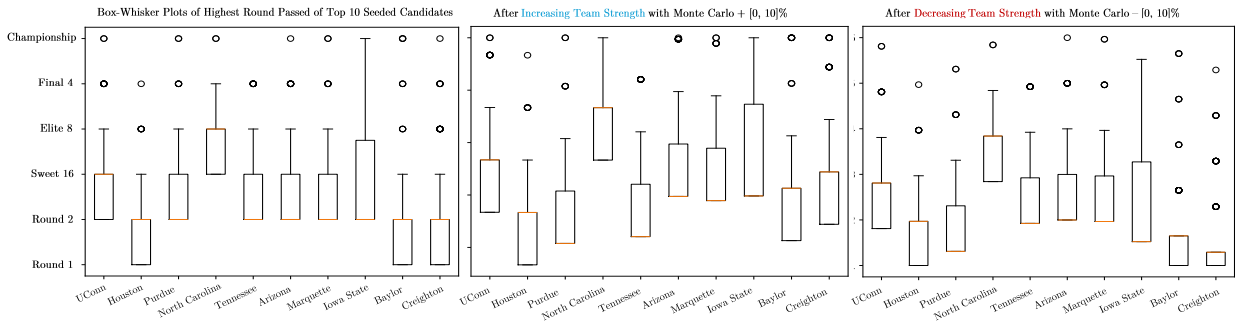


Figure 9: **Sensitivity of highest round reached by top 10 seeded teams based on changes in mean.** Monte Carlo simulation run by varying strength index mean of each of top 10 teams by up to $\pm[0, 10]\%$.

As shown in Figure 9, a relatively small increase in mean initial strength for each of the Top 10 teams led to an overall increase in how often they reached the later rounds of the tournament. Conversely, a decrease in initial strength, even if by a small margin, led to overall worse performances and less consistency in reaching the final rounds. This aligns with our expectations that a team's strength should correlate with its final placement.

As expected in Figure 10, an increase in initial strength variance for each of the top 10 seeded

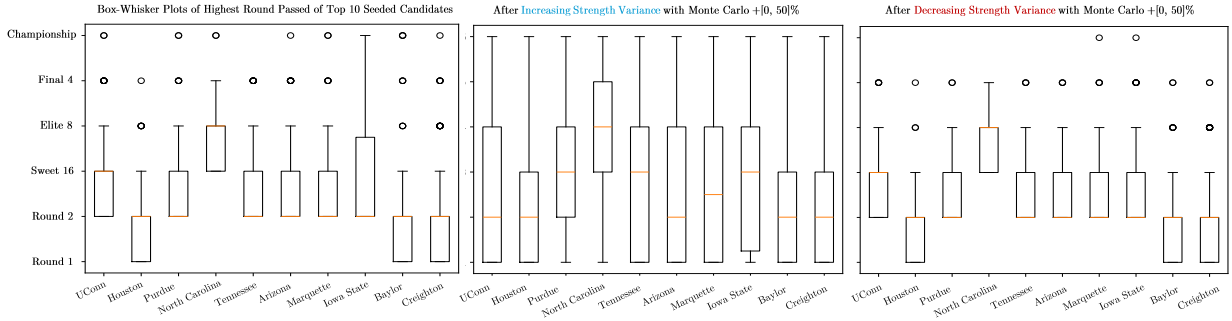


Figure 10: **Sensitivity of highest round reached by top 10 seeded teams based on changes in variance.** Monte Carlo simulation run by varying strength index variance of each of top 10 teams by up to $\pm[0, 50]\%$.

teams increased the variance of final-round placements. The opposite holds when the variance of initial strength estimates decreases. This means that when teams are considered to be more "lucky", they have greater chances of getting knocked out early or staying in later rounds.

4.3 Strengths

1. Our model's main strength is its ability to model the random nature of luck.

   By treating team strengths as random variables, the model acknowledges that a team's performance in a given game is not solely determined by its average ability but also by stochastic factors such as injuries, refereeing decisions, and unexpected hot or cold streaks. This stochastic nature is incorporated into the model by sampling from the strength index distributions for each team before applying the Bradley-Terry model to predict the outcome. As a result, the model can better account for the possibility of upsets and unexpected outcomes, which are common occurrences in March Madness.

2. Our model allows a team's strength to vary over time.

   This allows for a higher probability of upsets in later rounds, as teams that have exceeded expectations in earlier stages can have their strengths reevaluated to better align with their current performance. Furthermore, it aligns more closely with the inherent randomness and unpredictability of sports like basketball.

3. Our model only requires a few key statistics and features about each team's performance to produce game predictions.

   This allows our model to represent a complex scenario with relatively few requirements in available data, making it more accessible to the user and easily applicable to other March Madness and basketball tournaments. We mainly utilize the points won and points lost data to generate our strength index.

4.4 Limitations

1. A single strength index distribution may not capture all nuances of a team's ability and luck.

   While the model effectively captures the randomness and luck inherent in sports, a potential limitation lies in the use of a single strength index distribution for each team. For instance, a team's strength might vary depending on specific match-ups, home-court advantage, recent momentum, or public opinion. Additionally,

normal distributions may not accurately represent the potential for extreme outliers or unexpected upsets well, which can significantly impact tournament outcomes.

2. Converging winning probabilities to 0.5 in later rounds makes it difficult to estimate far-in-the-future games.

   As teams advance through the tournament and their strengths are updated, the winning probabilities between any two teams tend to converge towards 0.5. This means that as the tournament progresses, the model becomes less confident in predicting the outcomes of future games. This limitation arises because of the model's updating mechanism. While effective in capturing short-term fluctuations in team strength, the model may not adequately account for the long-term implications of early-round upsets or dominant performances. As a result, the model's predictive power diminishes for games that are further into the future.

4.5 Future Work

Beyond the scope of this paper, we intend to create an improved model by simulating not just the 64 teams in the bracket, but also the First Four games determining the 16th seed spots in each region. Furthermore, exploring more features, such as individual player and team statistics, as measures of the strength of a team may improve the accuracy of our predictions. Lastly, a potential improvement to our model is to measure luck across different rounds, as variance estimates may change over time for a team's strength.

5 Conclusion

In this paper, we began by constructing a mathematical model to predict March Madness game outcomes based on skill and luck. To incorporate skill, our model began with a calculated strength factor based on each team's historical performance, including statistics such as point differentials, past games won, and past games lost. To reflect luck in the actual games and the tournament's frequent upsets, we simulated random samples from each team's strength distribution. Using each team's calculated strength grades, the outcome of a game was decided by whose sampled strength, representing the team's performance level, was higher.

After developing the model, we tested its performance on the 2024 March Madness bracket. By employing a Monte Carlo simulation of the model's projections of each game in the tournament, we found its overall average accuracy to be around 80%, with a success rate of around 70% for the first round. Thus, the model's ability to account for upsets demonstrates its robustness and adaptability.

6 References

[1] NCAA division i men's basketball tournament. Page Version ID: 1252159934.

[2] T. R. . NCAA basketball stats - NCAA BB team average scoring margin.

[3] Y. . Bradley-terry model.

[4] J. Camenker. Warren buffett's march madness bracket challenge, explained | sporting news.

[5] R. Cedzo. Improving march madness bracket performance through machine learning.

[6] B. Gibbons. March madness upset picks, predictions, first round probabilities.

[7] G. Jeon and J. Park. Characterizing patterns of scoring and ties in competitive sports. 565:125544.

[8] I. G. Ludden, A. Khatibi, D. M. King, and S. H. Jacobson. Models for generating NCAA men's basketball tournament bracket pools. 16(1):1–15. Publisher: De Gruyter.

[9] I. McHale and A. Morton. A bradley-terry type model for forecasting tennis match results. 27(2):619–630.

[10] J. Morrison. Bradley-terry rankings: Introduction to logistic regression.

[11] J. Palm. Bracketology: Iowa state jumps north carolina for fourth no. 1, UConn takes overall top seed from purdue.

[12] L. Reyes, J. Mendoza, V. Hernandez, N. Armour, and L. Schnell. March madness highlights: Iowa, UConn earn final four spots as field is set.

[13] Sensei. Game analysis - game 3 - UConn vs north carolina.

[14] D. Wilco. The absurd odds of a perfect NCAA bracket | NCAA.com.

[15] A. Wittry. Here's how to pick march madness men's upsets, according to the data | NCAA.com.

[16] L. Zatzman. The factors that could make march madness mad: Upset seeds, deep 3s and post d.

[17] F. Zhu. A method to the madness: Predicting NCAA tournament success – the harvard sports analysis collective.

7 Appendices

7.1 Appendix 1: Simulation code

./marchmadness.py

```python
import numpy as np
import math
import random
import warnings
import copy
import json

warnings.filterwarnings('ignore')

class TeamWR:
    """
    A class that represents a team's average and stddev in win rate, or "strength"
    and simulating its performance in a given game.
    """
    name = ""
    seed = 1
    regional_seed = 1
    mean = 0.5
    stddev = 0.1

    # Instance attribute
    def __init__(self, name, seed, regional_seed, mean, stddev):
        self.name = name
        self.seed = seed
        self.regional_seed = regional_seed
        self.mean = mean
        self.stddev = stddev

    def update_mean(self, mean):
        self.mean = mean

    def update_stddev(self, stddev):
        self.stddev = stddev

    def sample_wr(self):
        return np.random.normal(loc=self.mean, scale=self.stddev, size=1)

    def equals(self, other_team):
        if (self.name == other_team.name) and (self.seed == other_team.seed) and (self.mean == other_team.mean) and (self.stddev == other_team.stddev):
            return True
        else:
            return False

def update_wr(team_1_wr, other_wr):
    """Given a team's win rate and the win rates of the other teams, returns the new average win rate for the team."""
    avg_wr = 0
    for team_wr in other_wr:
        avg_wr += (math.exp(team_1_wr) / (math.exp(team_1_wr) + math.exp(team_wr)))
    return avg_wr / len(other_wr)

def simulate_round(teams_list):
    """Given a list of teams (TeamWR objects), updates each team's mean win rate
    based on the Bradley Terry model. Returns a dictionary of the winning teams and upsets."""

    # Sampled strength
    sim_wrs = []
    for team in teams_list:
        sim_wrs.append(team.sample_wr())
    #print(sim_wrs)
    # Simulate bracket for winners and upsets
    bracket_results = winners_list(teams_list, sim_wrs)
    for i in range(len(teams_list)):
        other_teams = []
        # Gets sample strengths of other teams
        for j in range(len(teams_list)):
            if j != i:
                other_teams.append(sim_wrs[j])
        teams_list[i].update_mean(update_wr(sim_wrs[i], other_teams))
    return bracket_results

def winners_list(teams_list, sim_wrs):
    """Returns a dictionary of winners and upsets for a list of teams playing each other"""
    results = {
        "winners": [],
        "upsets": []
    }
    # To second-to-last index to avoid out-of-bounds
    for i in range(len(teams_list) - 1):
        # First team in each pair matchup, assuming first team in list plays second team, etc.
        curr_team = teams_list[i]
        opp_team = teams_list[i + 1]
        if i % 2 == 0:
            # 1 and 2 always win against 15 and 16 in region
            if (int(curr_team.regional_seed / 100) == int(opp_team.regional_seed / 100)) and ((int(curr_team.regional_seed % 100) == 1 and int(opp_team.regional_seed % 100) == 16) or (int(curr_team.regional_seed % 100) == 2 and int(opp_team.regional_seed % 100) == 15)):
                results["winners"].append(curr_team)
            else:
                win_chance = math.exp(sim_wrs[i]) / (math.exp(sim_wrs[i]) + math.exp(sim_wrs[i + 1]))
                # If win chance is over 50%, the current team wins, otherwise the opposing team wins
                if win_chance > 0.5:
                    # Upsets happen if a team's mean strength is less than its opponent's
                    if curr_team.mean < opp_team.mean:
                        results["upsets"].append(curr_team.name + "(W) vs. " + opp_team.name)
                    results["winners"].append(curr_team)
                else:
                    if curr_team.mean > opp_team.mean:
                        results["upsets"].append(opp_team.name + "(W) vs. " + curr_team.name)
                    results["winners"].append(opp_team)
    return results

def simulate_100_rounds(teams_list):
    """Simulates 100 rounds, tracking the number of upsets"""
    upsets = 0
    upsets_dict = {}

    for i in range(100):
        sim_wrs = []
        for team in teams_list:
            sim_wrs.append(team.sample_wr())
        sim_result = winners_list(teams_list, sim_wrs)
        #(sim_result["upsets"])
        upsets += len(sim_result["upsets"])
        for upset in sim_result["upsets"]:
            if upset in upsets_dict:
                upsets_dict[upset] += 1
            else:
                upsets_dict[upset] = 1
    return [upsets, upsets_dict]

def num_rounds(teams_list):
    """Calculates the number of rounds to determine a winner in a bracket"""
    # Check if length is a power of 2 so there are a perfect integer number of rounds

    if (len(teams_list) & (len(teams_list) - 1)) == 0:
        return int(math.log2(len(teams_list)))
    else:
        raise Exception("The number of teams should be a power of 2.")

def simulate_bracket(teams_list, rounds_left):
    if rounds_left > 0:
        #print("\nRound " + str(rounds_left))
        # for i in teams_list:
        #     print(i.name)
        round_result = simulate_round(teams_list)
        return [round_result] + simulate_bracket(round_result["winners"], rounds_left - 1)
    else:
        # print("\nRound 0")
        # print(teams_list[0].name)
        return []

# EAST-WEST-SOUTH-MIDWEST ORDER
east_bracket = [TeamWR("UConn", 1, 101, 0.877, 0.064), TeamWR("Stetson", 64, 116, 0.601, 0.090), TeamWR("Fla Atlantic", 31, 108, 0.699, 0.064), TeamWR("Northwestern", 36, 109, 0.612, 0.084), TeamWR("San Diego St", 18, 105, 0.733, 0.096), TeamWR("UAB", 50, 112, 0.542, 0.089), TeamWR("Auburn", 15, 104, 0.821, 0.071), TeamWR("Yale", 52, 113, 0.717, 0.116), TeamWR("BYU", 17, 106, 0.768, 0.058), TeamWR("Duquesne", 46, 111, 0.620, 0.097), TeamWR("Illinois", 12, 103, 0.738, 0.058), TeamWR("Morehead St", 57, 114, 0.733, 0.091), TeamWR("Washington St", 26, 107, 0.709, 0.070), TeamWR("Drake", 40, 110, 0.731, 0.090), TeamWR("Iowa St", 8, 102, 0.774, 0.090), TeamWR("South Dakota St", 61, 115, 0.641, 0.083)]

west_bracket = [TeamWR("North Carolina", 4, 201, 0.762, 0.052), TeamWR("Wagner", 68, 216, 0.525, 0.115), TeamWR("Mississippi St", 32, 208, 0.632, 0.045), TeamWR("Michigan St", 33, 209, 0.685, 0.110),
```

```
    TeamWR("Saint Mary's", 20, 205, 0.799, 0.058), TeamWR("Grand Canyon", 47, 212, 0.787, 0.058), TeamWR("Alabama", 16, 204, 0.7032, 0.058), TeamWR("Charleston", 54, 213, 0.7028, 0.083),
    TeamWR("Clemson", 22, 206, 0.688, 0.089), TeamWR("New Mexico", 42, 211, 0.768, 0.096), TeamWR("Baylor", 9, 203, 0.701, 0.090), TeamWR("Colgate", 58, 214, 0.680, 0.070),
    TeamWR("Dayton", 28, 207, 0.718, 0.070), TeamWR("Nevada", 37, 210, 0.731, 0.103), TeamWR("Arizona", 6, 202, 0.825, 0.070), TeamWR("Long Beach St", 59, 215, 0.541, 0.083)]

    south_bracket = [TeamWR("Houston", 2, 301, 0.871, 0.065), TeamWR("Longwood", 63, 316, 0.674, 0.103), TeamWR("Nebraska", 29, 308, 0.678, 0.089), TeamWR("Texas A&M", 34, 309, 0.591, 0.103),
    TeamWR("Wisconsin", 19, 305, 0.640, 0.064), TeamWR("James Madison", 48, 312, 0.796, 0.077), TeamWR("Duke", 13, 304, 0.835, 0.051), TeamWR("Vermont", 51, 313, 0.752, 0.096),
    TeamWR("Texas Tech", 23, 306, 0.660, 0.070), TeamWR("North Carolina St", 45, 311, 0.570, 0.071), TeamWR("Kentucky", 11, 303, 0.703, 0.090), TeamWR("Oakland", 55, 314, 0.571, 0.104),
    TeamWR("Florida", 25, 307, 0.670, 0.103), TeamWR("Colorado", 39, 310, 0.695, 0.077), TeamWR("Marquette", 7, 302, 0.740, 0.077), TeamWR("Western Kentucky", 60, 315, 0.645, 0.070)]

    midwest_bracket = [TeamWR("Purdue", 3, 401, 0.815, 0.051), TeamWR("Grambling", 66, 416, 0.450, 0.096), TeamWR("Utah St", 30, 408, 0.725, 0.064), TeamWR("TCU", 35, 409, 0.681, 0.070),
    TeamWR("Gonzaga", 21, 405, 0.832, 0.070), TeamWR("McNeese", 49, 412, 0.860, 0.064), TeamWR("Kansas", 14, 404, 0.683, 0.096), TeamWR("Samford", 53, 413, 0.719, 0.070),
    TeamWR("South Carolina", 24, 406, 0.662, 0.052), TeamWR("Oregon", 43, 411, 0.581, 0.097), TeamWR("Creighton", 10, 403, 0.729, 0.070), TeamWR("Akron", 56, 414, 0.700, 0.083),
    TeamWR("Texas", 27, 407, 0.671, 0.115), TeamWR("Colorado St", 44, 410, 0.716, 0.083), TeamWR("Tennessee", 5, 402, 0.769, 0.058), TeamWR("Saint Peter's", 62, 415, 0.553, 0.097)]

    entire_bracket = east_bracket + west_bracket + south_bracket + midwest_bracket

    # TEST ORDER
    def test_brackets(east_bracket, actuals):
        successes = []
        for i in range(100):
            sim_wrs = []
            for team in east_bracket:
                sim_wrs.append(team.sample_wr())
            num_successful = 0
            result = winners_list(east_bracket, sim_wrs)
            for j in range(8):
                #successes.append(result["winners"][j].name)
                if actuals[j] == result["winners"][j].name:
                    num_successful += 1
            successes.append(num_successful)
        return sum(successes) / len(successes)

    # FULL PREDICTION FOR 2024
    simulate_bracket(entire_bracket, num_rounds(entire_bracket))

    # Common Upsets for 1st Round
    common_upsets = simulate_100_rounds(entire_bracket)[1]
    sorted_upsets = dict(sorted(common_upsets.items(), key=lambda item: item[1]))

    # Top Seed Performance
    def top_seed_performances(teams_list):
        """
        Simulates 100 brackets and tracks how many times each top-seeded team reached specific rounds.
        """
        # Get the top 10 seeded teams
        top_10_teams = sorted(teams_list, key=lambda team: team.seed)[:10]

        # Initialize result dictionary
        rounds = [f"Round of {2 ** i}" for i in range(num_rounds(teams_list))]
        top_10_rounds = {team.name: {round_name: 0 for round_name in rounds} for team in top_10_teams}

        # Run 100 simulations
        for _ in range(100):  # Adjust the number of simulations if needed
            simulation_teams = teams_list.copy()
            bracket_result = simulate_bracket(simulation_teams, num_rounds(simulation_teams))

            for team in top_10_teams:
                # Track the rounds the team participated in
                for round_index, bracket_round in enumerate(bracket_result):
                    if team in bracket_round["winners"]:
                        round_name = f"Round of {len(bracket_round['winners'])}"
                        top_10_rounds[team.name][round_name] += 1
                    else:
                        break  # Team is eliminated

        return top_10_rounds

    top_performances = top_seed_performances(entire_bracket)

    def top_seed_sensitivity(teams_list):
        """
        Simulates 100 brackets and tracks how many times each top-seeded team reached specific rounds when
        varying mean and stddev of their strength metrics
        """
        # Get the top 10 seeded teams
        top_10_teams = sorted(teams_list, key=lambda team: team.seed)[:10]
        rounds = [f"Round {i + 1}" for i in range(num_rounds(teams_list))]
        sensitivity_results = {"Mean Inc": [], "Mean Dec": [], "StdDev Inc": [], "StdDev Dec": []}

        for top_team in top_10_teams:
            # Initialize result dictionary
            top_10_rounds = [
                {top_team.name: {round_name: 0 for round_name in rounds}}
                for _ in range(4)
            ]

            for _ in range(100):  # Simulate 100 times
                variations = {
                    "Mean Inc": copy.deepcopy(teams_list),
                    "Mean Dec": copy.deepcopy(teams_list),
                    "StdDev Inc": copy.deepcopy(teams_list),
                    "StdDev Dec": copy.deepcopy(teams_list),
                }

                # Apply variations to the top team
                for index, team in enumerate(teams_list):
                    if team.equals(top_team):
                        variations["Mean Inc"][index].update_mean(team.mean * (1 + 0.1 * random.random()))
                        variations["Mean Dec"][index].update_mean(team.mean * (1 - 0.1 * random.random()))
                        variations["StdDev Inc"][index].update_stddev(team.stddev * (1 + 0.5 * random.random()))
                        variations["StdDev Dec"][index].update_stddev(team.stddev * (1 - 0.5 * random.random()))

                # Simulate brackets for each variation
                results = {
                    "Mean Inc": simulate_bracket(variations["Mean Inc"], num_rounds(variations["Mean Inc"])),
                    "Mean Dec": simulate_bracket(variations["Mean Dec"], num_rounds(variations["Mean Dec"])),
                    "StdDev Inc": simulate_bracket(variations["StdDev Inc"], num_rounds(variations["StdDev Inc"])),
                    "StdDev Dec": simulate_bracket(variations["StdDev Dec"], num_rounds(variations["StdDev Dec"])),
                }

                # Track the maximum round reached by the team for each simulation
                for variation_index, (key, result) in enumerate(results.items()):
                    max_round = 0
                    for round_index, bracket_round in enumerate(result):
                        if any(winner.name == top_team.name for winner in bracket_round["winners"]):
                            max_round = round_index
                        else:
                            break
                    round_name = f"Round {max_round + 1}"

                    top_10_rounds[variation_index][top_team.name][round_name] += 1

            # Append results for this team
            sensitivity_results["Mean Inc"].append(top_10_rounds[0])
            sensitivity_results["Mean Dec"].append(top_10_rounds[1])
            sensitivity_results["StdDev Inc"].append(top_10_rounds[2])
            sensitivity_results["StdDev Dec"].append(top_10_rounds[3])

        return sensitivity_results

    bracket_sensitivity = top_seed_sensitivity(entire_bracket)
```

7.2 Appendix 2: Initial Strength Distributions ($\theta_{i,0}$)

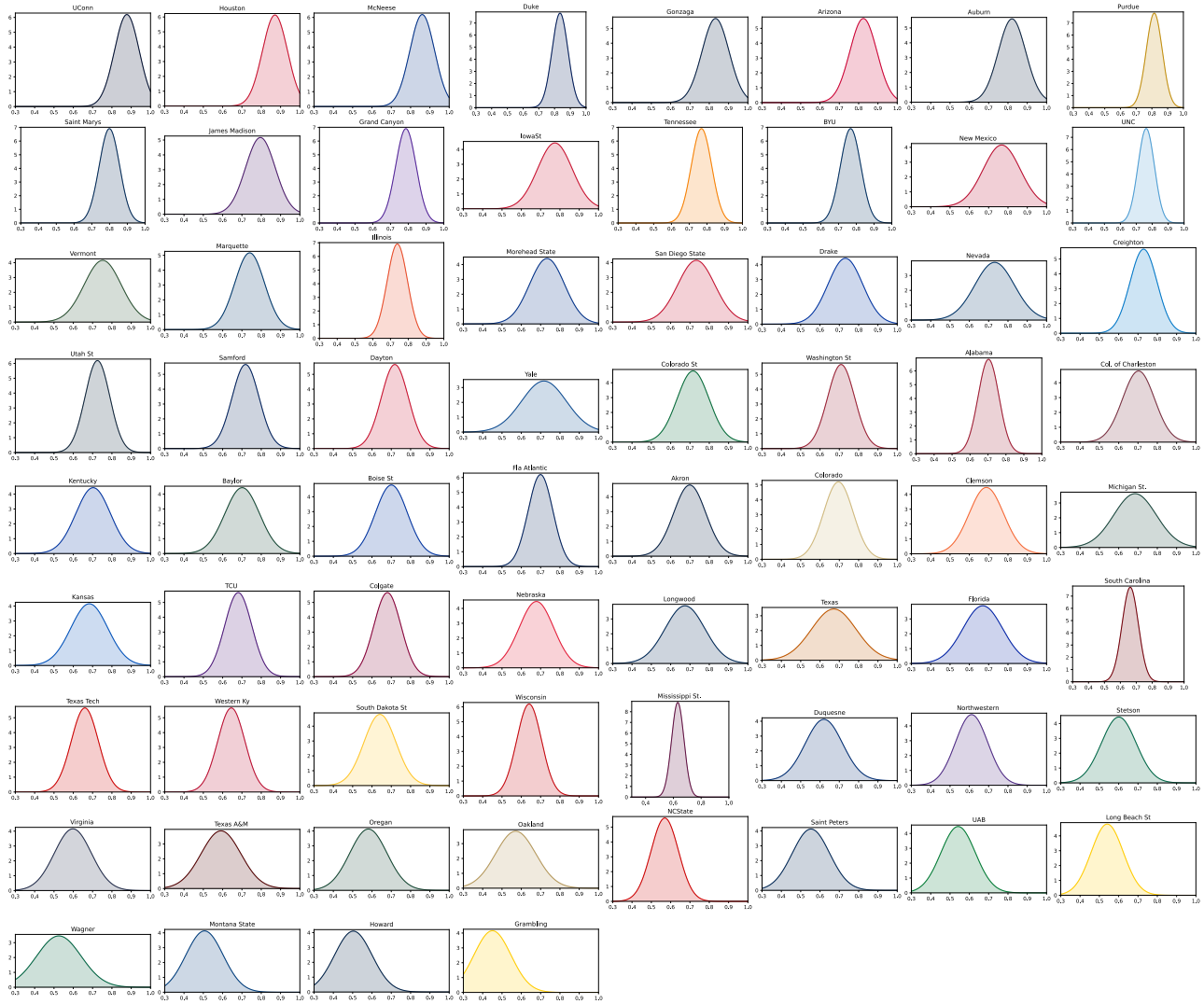


Figure 11: **Distributions for each strength index for every team.**

7.3 Appendix 3: One-page letter to newspaper chief editor

BMCM Team 01 Consultancy

November 17, 2024
Chief Editor
The Daily Hoopster
3-Pointer Plaza, Suite 333
Free Throw Lane, Dunkville, BB 90210

Dear Chief Editor,

We are thrilled to share the latest findings from our model of the 2024 NCAA March Madness tournament. Though we've sent the full technical report to your analytics team, we wanted to give you a direct summary of our model and share some insights that your readers might be interested in. Some key features of our paper up first:

1. Point Differential is All You Need – especially in the age of artificial intelligence, some models are complex and computationally expensive, requiring millions of data points and days' worth of computation. With this model, we prove that we can predict game outcomes with data as simple as "points scored" and "points allowed"

2. "Strength Index", a Novel Measure – A central feature of our approach is the "Strength Index", a metric based primarily on the point differential. However, we also take into account luck-based factors, acknowledging that basketball is a game of both luck and skill.

3. Bracket Accuracy – Our model becomes more and more accurate over time, using new information to update our odds after each round. In the final four and championship game, we reached 100% accuracy!

As for interesting findings for a wider audience, we think your readers might be most interested in our "common upsets" section. Here, we found that we were consistently able to forecast the closest games as being a tossup—our 3 most common upsets were all games where the score was neck-and-neck until the end. You'll have to read the report to find out which games, and we hope your readers will find it interesting enough in the format of an article!

Aside from that, your readers might also take some interest in using our model for their bracket predictions. We know that, for example, NC State went on an unprecedented run as an 11th seed, tying the lowest seed to make it to the Final Four in March Madness so far. Our model accounts for the fact that when teams achieve an upset, there's a substantial chance that it isn't a fluke. When a team achieves an unlikely win, in our model, it is rewarded in the following round. We hope the superfans in your reader base will feel validated in that way.

Thank you for this wonderful opportunity! We hope you enjoyed this summary of our technical report and look forward to hearing back from your analytics team.

Warm regards,
BMCM Team 01