THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENT OF STATISTICS

PREDICTING THE OUTCOME OF THE FINAL SIXTEEN TEAMS IN COLLEGE BASKETBALL USING TIME SERIES ANALYSIS AND MARKOV CHAINS

CHRISTOPER KENNETH MILLER
SPRING 2023

A thesis
submitted in partial fulfillment
of the requirements
for baccalaureate degrees
in Statistics and Mathematics
with honors in Statistics

Reviewed and approved* by the following:

Andrew Wiesner
Associate Teaching Professor of Statistics
Thesis Supervisor

David R. Hunter
Professor of Statistics
Honors Adviser

* Electronic approvals are on file.

ABSTRACT

The culminating tournament for collegiate level basketball brings together a field of sixty-four teams to crown a champion in a spectacle called March Madness. There are many different approaches to ranking and predicting the outcome of the games within the tournament. In this paper, a new approach is taken using a time-series analysis approach to capture the effect of trends throughout the season. The underlying time-series model will be based on a team's efficiency rating, or put otherwise, a team's ability to score and defend well against other teams, adjusting for relative strength. Two types of time-series models are considered: a dynamic auto-regressive integrated moving average (ARIMA) model that is fit uniquely to every team, and a dynamic exponential smoothing (ETS) model that is also fit uniquely to every team. These models are used on nine different college basketball seasons from 2011 to 2019 using a probability Markov chain approach to determine winning probabilities for predicting only the first two rounds of the March Madness tournament. Over the nine-year period, either the ARIMA or ETS method predicted the same or more correct games over the first two rounds when compared to a baseline method which predicts the favorite seed for every matchup. The analysis also looks at the effectiveness of predicting upsets. The ARIMA method predicted upsets correctly 37.2% of the time while the ETS method registers at 44.2%. It is recommended that this type of modeling for March Madness predictions would be best used to quantify trending teams. The ETS model slightly outperforms the ARIMA model and both models are most effective when trying to determine closely seeded matchups.

TABLE OF CONTENTS

LIST OF FIGURES iii

LIST OF TABLES iv

ACKNOWLEDGEMENTS v

Chapter 1 Introduction 1

Motivation and Overview 2

Chapter 2 Efficiency Calculations 5

Chapter 3 Data Considerations and Setup 7

Chapter 4 Time-Series Models 9

ARIMA Model 9
ETS Model 13

Chapter 5 Markov Chain as Probability Model 17

Chapter 6 Analysis of Results 21

Upsets 25
2023 Results 27

Chapter 7 Conclusions 29

Limitations 29
Future Considerations 30
Conclusions and Recommendations 32

Appendix A R Code 34

BIBLIOGRAPHY 43

LIST OF FIGURES

Figure 1 – Penn State 2011 Time Series.....11

Figure 2 – Diagnostic Plots for Penn State 2011.....11

Figure 3 – ARIMA Prediction.....13

Figure 4 – All Considered Time-Series Models (Hyndman, 2002).....15

Figure 5 – ETS Prediction 16

Figure 6 – NCAA March Madness Region Seeding (Schwertman, 1991) 18

Figure 7 – Probability of Winning in Round 1.....21

LIST OF TABLES

Table 1 – Example of Different Methods Probabilities in East 2011 Region.....19

Table 2 – Overall Percent Games Correct by Year for Each Method and Round.....22

Table 3 – Overall Games Correct by Year for Each Method and Round.....23

Table 4 – Close Matchups Comparison.....24

Table 5 – Percent of Upsets Predicted.....25

Table 6 – Accuracy of Predicted Upsets.....26

Table 7 – 2023 Results.....27

ACKNOWLEDGEMENTS

I would like to thank Dr. Andrew Wiesner for guiding me through the entire process. Not only was he a great inspiration to all the ideas in this paper, but he was also a great resource throughout the process of the thesis to talk about sports, statistics, and all things in between.

I also want to thank my parents for supporting me throughout my whole undergraduate degree and to my father, who sentenced me to a lifelong obsession of sports since the day I was born.

Chapter 1

Introduction

National Collegiate Athletic Association (NCAA) Division I men's basketball is a widely publicized sport in the United States. Like many other sports, the use of statistics to make informed decisions has become relevant for the league, analysts, coaches and even the casual fan. The validity and usefulness of certain statistically backed reasoning can be questioned and explored but it is undeniable that data-backed decisions provide concreteness to any given conclusion. In the eyes of analysis, this sport provides an extra level of difficulty due to the collegiate aspect. In a traditional professional league, you can note some small material differences between teams, for example, payroll size, organization location, owner investment strategies, but overall, you can do analysis with the assumption that the professional teams are all on the same level of fairness. When we look at the collegiate level, there are two main levels of unfairness that we must consider when looking to do any sort of statistical analysis: access to funds and recruiting level.

It is known that the NCAA basketball tournament, also known as March Madness and the final tournament of the NCAA season, is regarded as the pinnacle of college basketball. It is a single elimination tournament that decides the overall champion of college basketball each year. The tournament is set up such that out of the 32 Division I conferences, the champion of each is guaranteed a spot in the tournament. Then 36 other teams are selected by the NCAA committee. This guarantee's representation of every conference, and rewards teams that play in tougher

conferences, which goes back to the unfairness factor in this sport. When the field of 64 teams is set, the NCAA committee decides seeding such that the best teams would play the worst teams on a path to the championship. This rewards the teams that did the best in the regular season. This seeding decision by the NCAA is, at least in part, statistically based, and by creating an order of teams, the NCAA is essentially making their own prediction of what teams they think are better than others (NCAA.com, 2023). If the NCAA's ranking of seeds were completely accurate to the predictiveness of the tournament games, then the better seed would always win with the best ranked number one seed winning the whole tournament. We know this is not true and the probability of the better seed winning every time has elements of randomness. Much of this randomness in predicting relative team performance has to do with the complexity of valuing how much the unfairness contributed to the team's performance.

This complexity has led to many analysts having different ideas than the NCAA. Other ranking systems focus on different metrics and methods like efficiency and relative performance. Some years these methods perform better than others and no one method is correct. Rather, other methods provide a reasonable prediction performance, and more importantly, a unique way to explore the relative strength of the teams in the tournament. This brings about the objective of this research paper; to provide a new and unique way to look at March Madness teams' relative strength while also having a reasonable prediction performance.

Motivation and Overview

One thing commonly heard when it comes to March Madness predictions is, "what teams are hot?". This prompt usually explores what teams are trending up recently and what teams are

trending down and how that all plays into overall predictions. If we look at popular prediction methods, we can see how they take this prompt into consideration.

The industry standard prediction, KenPom, is an adjusted efficiency margin rating that uses some adjusted offense and defense efficiency (this will be discussed in more detail later) and uses an additive model to create efficiencies for each game throughout the season then averages the two metrics to rank the teams (Pomeroy, 2016). Many other popular rankings will have the same general idea, calculate a rating for a team's performance in any given game weighted by strength of schedule (SOS), venue advantage, etc., then take the average rating to rank the teams. Most of these methodologies tackle recency in a similar way. As Ken Pomeroy states, “The adjusted game efficiencies are then averaged (with more weighting to recent games)” (Pomeroy, 2006). RoundTable ratings states, “More recent games are weighted heavier than games from the early season” (RoundTable, n.d.). Sagarin comes the closest to focusing purely on teams’ trends by stating, “The RECENT, is score-based and weights RECENT play more heavily than earlier games. Its effect will become more pronounced the longer a season goes if a given team happens to have an upward or downward trend.” (Sagarin, 2023) What is unclear is how Sagarin decides what a trend is and how much more pronounced the effect is. Also, important to note is that Sagarin uses this factor as one of three factors in his overall rankings.

The methodology used in this paper will completely focus on a time-based approach to the prediction of overall teams’ ratings at the time of the start of the tournament. An adjusted offensive and defensive efficiency will be calculated for each team’s game performance and those metrics will be combined for an overall rating. These game efficiencies will be used in two separate time-series model approaches to predict performance values for the field of 64 teams in

the tournament. Then a Markovian approach will be taken to calculate the probabilities for each team's probability of winning for the first two rounds of the tournament.

Chapter 2

Efficiency Calculations

The underlying data that we will use in the time-series model is a metric called adjusted game efficiencies. For efficiency calculations we will not stray from the industry standard. Using a standard calculation is preferred as the focus of the paper is on the time-series methods and not the intricacies of efficiency calculations. The efficiency formula used will be from Bart Torvik's calculations. (Torvik, n.d.) For each game the winner and loser are assigned an offensive and defensive efficiency. These efficiencies are calculated using points scored/allowed per 100 possessions (PPP). Possessions are defined as

$$Poessession = (FGA - OR) + TO + (Y * FTA)$$

(Pomeroy 2004) The first component, $(FGA - OR)$, is a shot by the offense that is discounted by continued possessions, which is denoted as offensive rebounds. The second component, TO, is an offensive turnover. The last, $(Y * FTA)$, is a random number from 0 to 1 multiplied by free throws attempted because it is not known how much a free throw should count as a possession. After calculating offensive and defensive PPP for a team (one team's PPP for offense will be the other team's PPP for defense and vice versa), we can adjust these PPPs to get our adjusted offensive and defensive efficiencies. There are many unique ways to calculate these efficiencies, but the reason for choosing the efficiencies from Bart Torvik is because of the accessibility to these game-by-game efficiencies at the exact time before the tournament. Access could not be found for data at the time before the tournament for game-by-game efficiencies anywhere else.

The PPPs are then adjusted by opposing team strength and venue location. The formula for this adjustment follows:

$$Game\ Adj.OE = PPPo/(\frac{Opp.Adj.Average\ DE * Venue\ Factor}{League\ Average\ PPP})$$

$$Game\ Adj.DE = PPPd/(\frac{Opp.Adj.Average\ OE * Venue\ Factor}{League\ Average\ PPP})$$

The venue factor is 1.4% so if the team is home the factor is equal to 98.6% and is 101.4% on the road. The calculation boils down to a performance metric to see how well a team scores/defends comparative to the relative strength of their opponent. It is important to note that a higher game adjusted OE is related to a strong offensive performance and a lower game adjusted DE is related to a strong defensive performance.

At this step, the method used in this paper will diverge from most predictive methods. Here, most predictive methods will use some algorithm combined with outside factors to create predictive ranks. For example, KenPom popularized using Bill James' Pythagorean expectation method for calculating actual rank. (Miller, 2007) For this paper, a game performance metric is calculated as:

$$Game\ Adj.OE - \ Game\ Adj.DE$$

The predictiveness of this method now comes with applying this metric to time-series forecasting methods. There could be a better way to combine the two efficiencies to prepare for the time-series model, but this method is chosen for simplicity.

Chapter 3

Data Considerations and Setup

This method is predictive and not retrodictive. When carrying out this methodology, the rankings are supposed to predict how well the teams will do going forward. It is not a model to see what factors best explain results that have already happened. The data range used for the time series analysis is only the regular season games for any given team. Regular season games are defined as a matchup between two Division I teams each year. The data for the time-series model is not carried across seasons because college programs can experience a lot of turnovers with transfers, graduates, and NBA draft declarations. Though, the previous season does have a small implication on a couple of the first games efficiency metrics. There needs to be a base ranking set to calculate the first games efficiency ranking, and the base ranking is in part based on preseason rankings. When saying that just regular season games are used, that also means that conference tournament games are not included in the data. This is because it can be argued that teams don't have a similar number of games when some teams play more conference tournament games than others. Also, any one-bid conference (a conference in which only the champion makes the tournament) will automatically have higher predictions than intended just because they succeeded in the tournament format.

The regular season data is collected from Bart Torvik's T-Rank site, https://barttorvik.com, which allows me to go back into previous seasons and collect season data at a specific date. The tournament seeding data is collected from the Kaggle March Machine Learning Mania 2023 tournament datasets. Combining these two datasets covers the seasons from 2011-2019. This allows for predictions for 9 tournaments, which equates to 432 predicted

games and 576 unique time-series models. It can be noted that there are no major rule changes over these nine seasons. (History of, n.d.) Having no major rule changes for the length of the dataset is important. Certain rule changes, for example shot clock reduction, could affect the calculation of efficiencies, and would not allow consistency over each year. In 2011 the NCAA committee added the First Four play-in games, but we will not be predicting those games as most brackets do not count those games. Over the nine seasons, there are 47460 regular season games, 603 March Madness games, and 352 unique teams.

Chapter 4

Time-Series Models

ARIMA Model

Given a set of adjusted offensive efficiencies and a set of adjusted defensive efficiencies and the combination of both, we can now apply a time-series analysis on the season data for the teams. The main problem is that we are dealing with many different teams in multiple years and the main goal is to fit a time-series that is right for each one individually to forecast future values. This can be done using the auto.arima function in R in the forecast package. (Hyndman, 2023; R Core, 2021)

The function uses an algorithm called the Hyndman-Khandakar algorithm which determines the best ARIMA (Autoregressive Integrated Moving Average) model for a given team each year. (Hyndman, 2008) We will consider three parameters for this model, p, which is the degree of the Autoregressive term, d which is the number of times the time-series data has to be differenced to create stationary data, and q, which is the degree of the Moving Average term. For the sake of our data, there will be an assumption made that there is no seasonal component to our data because we are only modeling within a singular year. The formula for this model is:

$$\phi(B)(1 - B^d)y_t = c + \theta(B)\varepsilon_t$$

Where B is a backshift operator and the phi and theta functions of B correspond to a degree polynomial of the order p and q, respectively. The ε_t is a white noise process with mean zero and variance σ^2. If $c = 0$ then there is an implied order of differencing of d.

This algorithm works by first finding the order of differencing needed in the time-series model. Approaches before have suggested minimizing Alkaline Information Criteria (AIC), an estimated prediction error, from different combinations of models, but this can lead to over-differencing, instead the algorithm uses successive Kwiatkowski–Phillips–Schmidt–Shin (KPSS) unit-root tests to determine the differencing coefficient. This is essentially testing a null hypothesis that a time-series is stationary against an alternative hypothesis of a non-stationary time-series to see if there needs to be differencing. Then p and q are found using a stepwise selection that minimizes the AIC. This is important in relation to working with team data as a time series. Many teams' data could already have a stabilized mean throughout the season, for example a team that has never seemed to go on a stretch of good or bad games, compared to a team that is streakier and would have not had a stabilized mean. The algorithm will take both of those scenarios into account and determine which is appropriate according to the team's trends. The last thing to note is that for this algorithm to work as intended there is an assumption of homoscedasticity that must be made about the time-series data beforehand. With team game data we will find that there are not enough games in a season to identify a trend in variance that would violate the homoscedasticity assumption. This is helpful in the fact that the ARIMA model can be used with this data, but it also begins to uncover a limitation with the time-series model and this team data in the fact that ARIMA models tend to work stronger with more data points. Here is an example of Penn State in 2011 and how they were fit to an ARIMA model. Figure 1 is the time-series graph with Penn State's performance metric on the y-axis and the games in chronological order on the x-axis.

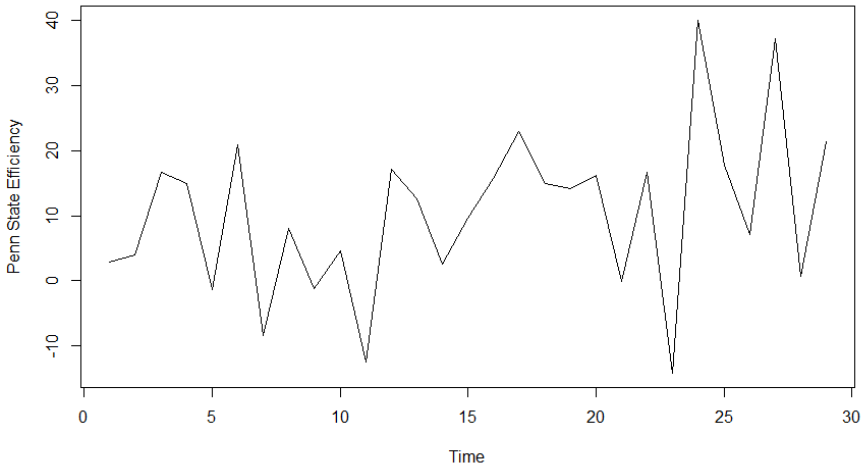

Figure 1 – Penn State 2011 Time Series

We can see that there is some high variability over the year for Penn State, therefore differencing the data to make it non-stationary before fitting the time-series model would be considered.

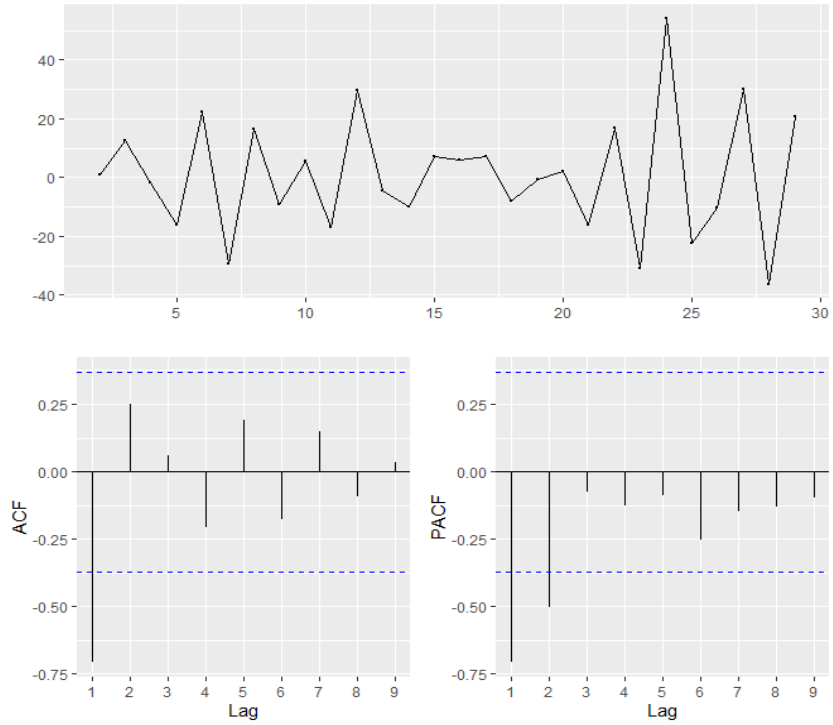

Figure 2 – Diagnostic Plots for Penn State 2011

From Figure 2 we can see that from differencing the time-series data we have stationary data. If the model was being fit manually, the Auto-Correlation Function (ACF) and Partial Auto-Correlation Function (PACF) plots would be considered to find the p and q parameters of the ARIMA model. ACF plots provide an auto-correlation value with the present value of the time series and the past game performances, for example, about -0.70 is the autocorrelation between the value of Penn State's performance at the time of the 2011 tournament and its first game of that season. The PACF plot looks at the correlation between a lag value and the residuals of the previous lag values, which were accounted for in the ACF plot. We will use both plots to determine the type of ARIMA model we have, if there is a gradual decrease in the ACF plot it is most likely only an AR model and if there is a gradual decrease in the PACF plot then it is only an MA process, and if neither have a gradual decrease ARIMA should be considered. To determine the lag, or parameters p and q, we will use the first instance of the ACF and PACF crossing the confidence interval line which is denoted as the blue dashed line in Figure 2. We can see that there is no gradual tail-off in the ACF and PACF plots, so models with p and q parameters greater than one should be considered, and looking at the plots specifically, models with a moving average component of 1 and an autoregressive component of 1 or 2. Now, if the auto.arima function is used it considers multiple possible combinations of these models and fits the model with the lowest AICc, which is a sample size adjusted version of the AIC that is useful for model selection when the sample size is smaller. For Penn State in 2011 this turned out to be a model of parameters $p = 1$ and $q = 1$, with one differencing of the time-series data, denoted as ARIMA(1,1,1). The fitted and forecasted points overlaid on the original time-series graph looks as in Figure 3, with the red line being the model and the black line is the original.

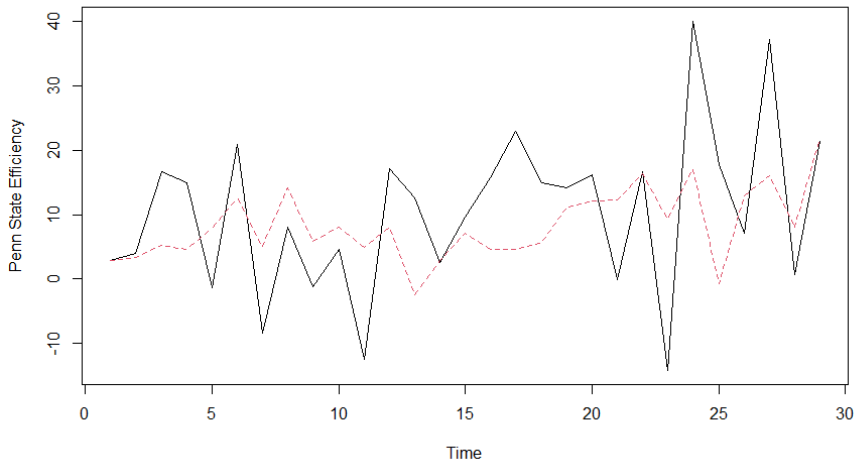

Figure 3 – ARIMA Prediction

ETS Model

We will also explore a second method for forecasting the performance values of the tournament teams. For this method, we will model the time-series data and apply an automatic exponential smoothing method to the data. Exponential time smoothing is a time-series analysis method that predicts future values based on a weighted linear sum of lags, giving exponentially more weight toward current observations. This is a function in the same R forecast package, called ETS (exponential time smoothing). In the automatic ARIMA method, we assume homoscedasticity, or constant variance, but this is a weak assumption in the scope that we don't truly know if this holds up for every team. This automatic exponential smoothing method provides more flexibility in that it does not need the homoscedasticity assumption and it is not as strict as a model parameter-wise when compared to ARIMA.

With this automatic ETS method, also created by Hyndman, we will consider four types of ETS models, no trend and no seasonal component (NN), additive trend and no seasonal

component (AN), multiplicative trend and no seasonal component (MN), or damped trend and no seasonal component (DN). (Hyndman, 2002) The NN model can be explained as seeing no trend throughout the season, the AN model would be seeing a linear trend, either upward or downward, throughout the season, the MN model would be seeing an exponential trend, either upward or downward, throughout the season, and the DN model would see a trend that accounts for trailing off for forecasts multiple points in the future. The DN model would not matter too much for this data because we are only forecasting one point into the future and the DN model, which can either be an additive or multiplicative damped, will have a similar one-point forecast to either the AN or MN model. These four model's equations can be written as follows:

$$\ell_t = \alpha P_t + (1 - \alpha) Q_t$$

$$b_t = \beta R_t + (\phi - \beta) b_{t-1}$$

Where ℓ_t is the series level at game t, and b_t represents the slope at game t. P_t, Q_t, R_t varies based on the type of model and α, β, and ϕ are all constants. Figure 4 shows how the P_t, Q_t, R_t, and the forecast, F_{t+h} will be calculated.

Trend component	**N** (none)
N (none)	$P_t = Y_t$ $Q_t = \ell_{t-1}$ $\phi = 1$ $F_{t+h} = \ell_t$
A (additive)	$P_t = Y_t$ $Q_t = \ell_{t-1} + b_{t-1}$ $R_t = \ell_t - \ell_{t-1}$ $\phi = 1$ $F_{t+h} = \ell_t + hb_t$
M (multiplicative)	$P_t = Y_t$ $Q_t = \ell_{t-1}b_{t-1}$ $R_t = \ell_t/\ell_{t-1}$ $\phi = 1$ $F_{t+h} = \ell_t b_t^h$
D (damped)	$P_t = Y_t$ $Q_t = \ell_{t-1} + b_{t-1}$ $R_t = \ell_t - \ell_{t-1}$ $\beta < \phi < 1$ $F_{t+h} = \ell_t +$ $(1 + \phi + \cdots + \phi^{h-1})b_t$

Figure 4 – All Considered Time-Series Models (Hyndman, 2002)

Now seeing the four types of models used, we can see that if two teams had the same type of model, for example, AN, then the only uniqueness between the model of the two teams is in the coefficients. To determine the coefficients, we will minimize the equation of twice the negative logarithm of the conditional likelihood function in OKS, an error calculation by Hyndman expanded from Ord, Koehler, and Snyder (Ord, 1977), with constant terms eliminated. After the coefficient estimates are optimized, the automatic algorithm decides the best model based on AIC. We can go back to the Penn State in 2011 example and can see in Figure 5, the smoothing model fit to the time-series data. This model is an AN model with optimized α = 0.0001,

essentially smoothing the time-series data into a flat line, which is fair considering it does not look like there is any trend in the data.

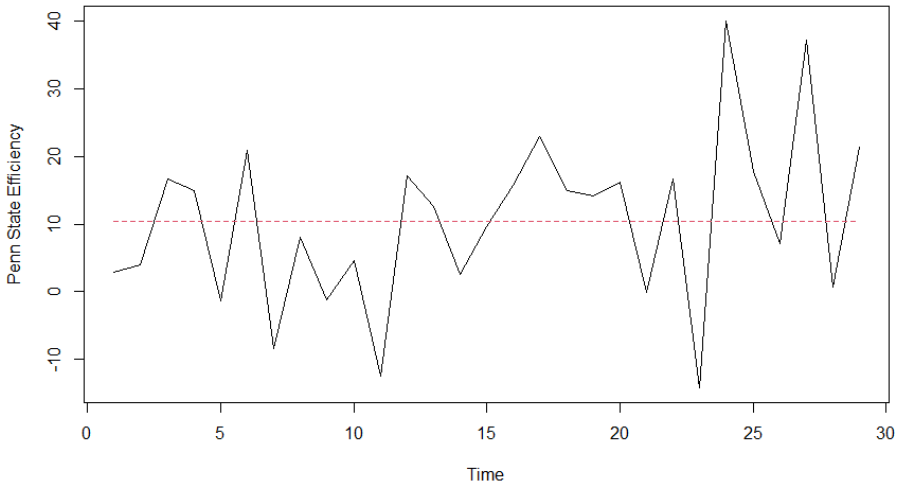

Figure 5 – ETS Prediction

Chapter 5

Markov Chain as Probability Model

For every year, we have a set of numbers defining the teams to participate in the final NCAA tournament. In the previous section, it was shown how a time-series analysis is used to populate a performance metric, now the following Markov chain method will be used to get expected probabilities of teams making it to the first and second rounds. It should be noted that this Markov chain can be applied to any set of numbers as long as those sets of numbers are positive. This idea of a probabilistic model to create expected results will follow the work of Schwertman. (Schwertman, 1991)

There are 64 teams in the tournament and these teams are split into four different regions. For any given year we are predicting the last four remaining teams in each region, therefore overall predicting 16 teams. This stochastic method takes the set of any pair of four connected teams in any given region, for example, seeds (1,16,8,9) or (2,15,7,10), and finds the probability of making it past the first round and the probability of making it past the second round.

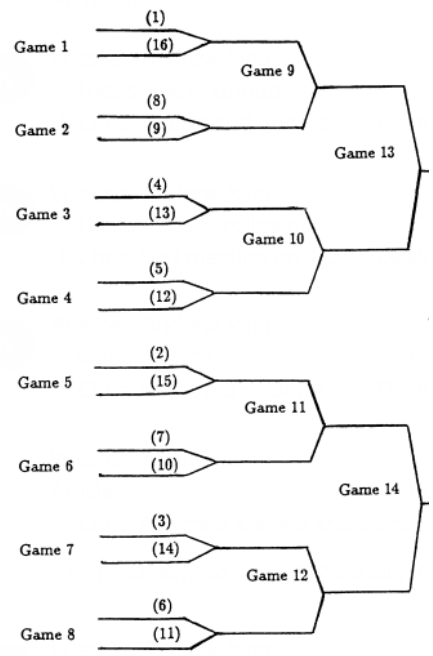

Figure 6 – NCAA March Madness Region Seeding (Schwertman, 1991)

We can define $P_k(i,j)$ as the probability of a given team with seed i beating seed j in the k^{th} game of the region. Then we can define $P(i,j) = \frac{u(i)}{u(i)+u(j)}$ where, $i \neq j$ and,

$$u(x) = Performance\ Metric\ of\ x^{th}\ seed\ in\ region \text{ and } u(x) > 0$$

$$Therefore, 0 < P(i,j) < 1$$

We can now see that if, for example, we wanted the probability that the third seed wins their first game, we can find it using $P_7(3,14) = P(3,14) = \frac{u(3)}{u(3)+u(14)}$. A Markov chain matrix can now be populated with $P(i,j)$ as the cells and the rows of this matrix will be seeds one through sixteen and the columns will be the same. Now if we wanted the probability that the third seed made it to sweet sixteen (second round), we can find it using,

$$P_{12}(3,j) = [P_7(3,14) * P_8(6,11) * P(3,6)] + [P_7(3,14) * P_8(11,6) * P(3,11)]$$

As an example, we will take the East region in the year 2011 and calculate $u(x)$ for each team to create the probabilities of winning in round one and two.

Probabilites of Making Round 1 and 2 using Markov Chain

Team	Seed	Seed Method R1	Seed Method R2	ARIMA Method R1	ARIMA Method R2	ETS Method R1	ETS Method R2
Ohio St.	1	0.94	0.84	0.86	0.54	0.93	0.59
North Carolina	2	0.88	0.71	0.78	0.46	0.83	0.43
Syracuse	3	0.82	0.58	0.72	0.35	0.76	0.37
Kentucky	4	0.76	0.47	0.72	0.46	0.73	0.45
West Virginia	5	0.71	0.36	0.62	0.29	0.65	0.32
Xavier	6	0.65	0.26	0.74	0.48	0.76	0.49
Washington	7	0.59	0.17	0.47	0.21	0.36	0.16
George Mason	8	0.53	0.08	0.46	0.19	0.47	0.18
Villanova	9	0.47	0.06	0.54	0.24	0.53	0.22
Georgia	10	0.41	0.09	0.53	0.26	0.64	0.37
Marquette	11	0.35	0.1	0.26	0.1	0.24	0.09
Clemson	12	0.29	0.09	0.38	0.14	0.35	0.13
Princeton	13	0.24	0.08	0.28	0.11	0.27	0.1
Indiana St.	14	0.18	0.06	0.28	0.08	0.24	0.06
LIU Brooklyn	15	0.12	0.04	0.22	0.06	0.17	0.03
UTSA	16	0.06	0.02	0.14	0.03	0.07	0.01

Table 1 – Example of Different Methods Probabilities in East 2011 Region

We can see the differences between the three methods in Table 1. It is important to note that for any region the seed method will always have the same probabilities because the nature of $u(x)$ for that method is not dependent on the actual characteristics of the team. For this region and this year, we can see that the ARIMA method only predicts one upset, although it does predict a close first round game between 5 seed West Virginia and 12 seed Clemson. The ETS method predicts the same upset as the ARIMA method but is much more favorable to the performance of the lower seeds in that region. This dynamic will change depending on the region

and the year and this can be explored along with the performance of each method's ability to predict what happened.

Chapter 6

Analysis of Results

With the methods set and the Markov chain creating winning probabilities, an R code now runs a loop for every tournament team in the nine years of study to simulate predictions for the first two rounds of each of those years' tournaments. First, we can look at the differences in expected win probabilities between the methods over the nine years.

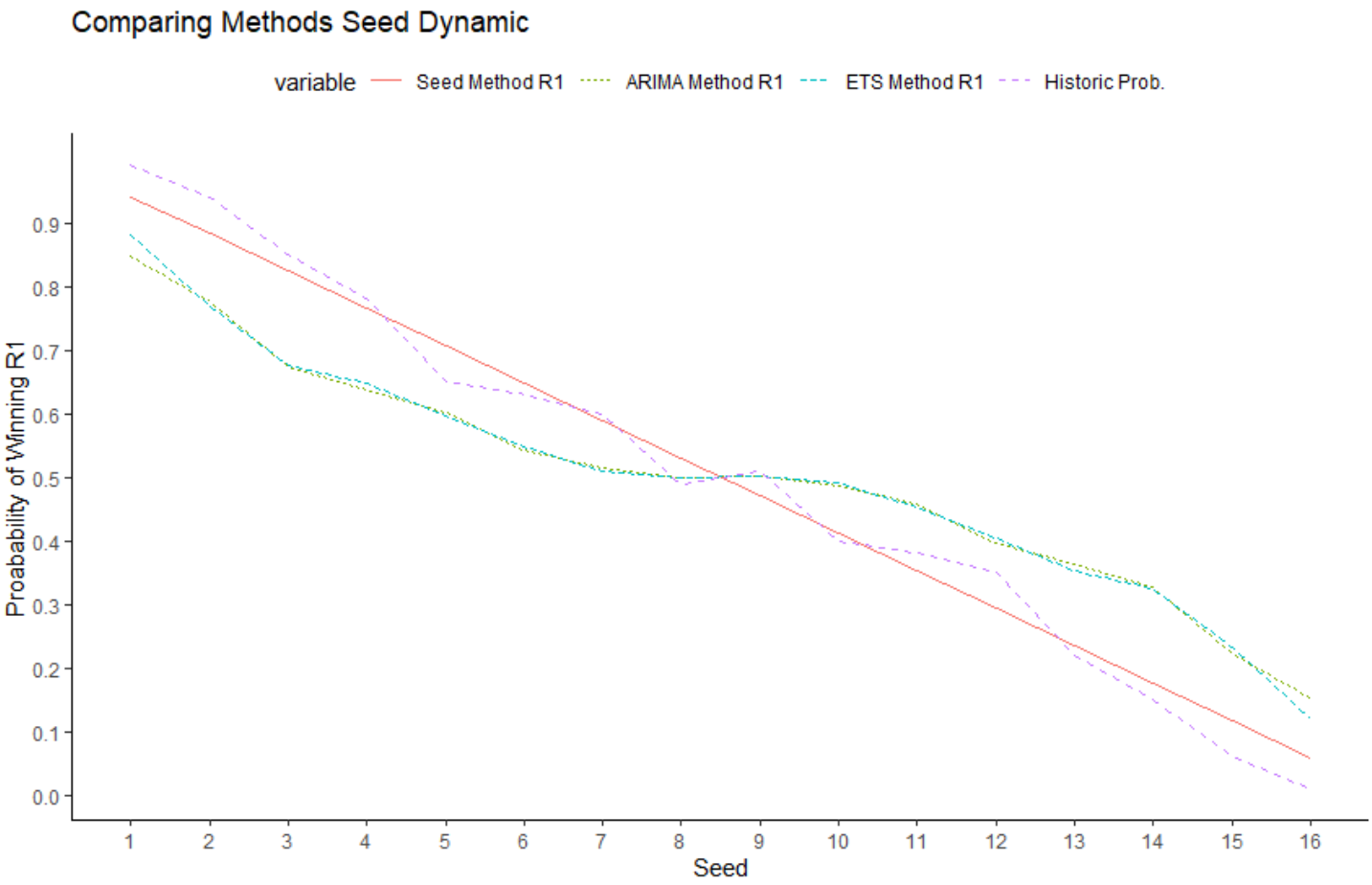

Figure 7 – Probability of Winning in Round 1

Overall, we can see that both the ARIMA and ETS methods act similarly on average when it comes to deciding the relative strengths of teams. Both methods give much more relative strength to lower seeds than the seed method, as expected. Historical winning probabilities, as seen with the blue dashed line, is closer to the ARIMA and ETS curves for the seeds 4 through 13 than it is for the high and low ends of the seeds. This is important to note because it provides some evidence to the fact that the ARIMA and ETS methods could be more accurate in closer seed matchups. On the contrary, there can be some worry that the method could be overvaluing lower seeds. Tables 2 and 3 show the overall predictiveness of the three methods:

Percent Correct in R1 and R2

Year	Seed Method R1 % Correct	ARIMA Method R1 % Correct	ETS Method R1 % Correct	Seed Method R2 % Correct	ARIMA Method R2 % Correct	ETS Method R2 % Correct
2011	78.12%	78.12%	78.12%	62.50%	62.50%	56.25%
2012	68.75%	65.62%	65.62%	68.75%	56.25%	50.00%
2013	68.75%	59.38%	68.75%	68.75%	62.50%	62.50%
2014	75.00%	78.12%	68.75%	62.50%	56.25%	50.00%
2015	84.38%	78.12%	78.12%	56.25%	56.25%	68.75%
2016	59.38%	78.12%	75.00%	62.50%	43.75%	43.75%
2017	81.25%	68.75%	75.00%	75.00%	31.25%	56.25%
2018	71.88%	71.88%	71.88%	43.75%	37.50%	43.75%
2019	62.50%	68.75%	68.75%	87.50%	87.50%	93.75%

Table 2 – Overall Percent Games Correct by Year for Each Method and Round

Number of Games Correctly Predicted

Year	Seed Method Games Correct	ARIMA Method Games Correct	ETS Method Games Correct
2011	35	35	34
2012	33	30	29
2013	33	29	32
2014	34	34	30
2015	36	34	36
2016	29	32	31
2017	38	27	33
2018	30	29	30
2019	34	36	37

Table 3 – Overall Games Correct by Year for Each Method and Round

In only three of the years, 2012, 2017, and 2018, picking the seed method is the best option. Between the ARIMA method and ETS method, they tend to perform slightly differently every year, with ARIMA predicting more games correctly in four years, ETS method predicting more in four other years, and the methods tying in 2019. Between the ARIMA and the ETS method, the ETS method holds a slight advantage, predicting 0.667 more games correctly. It seems to be that the ARIMA method is less consistent than the ETS method with a standard deviation of 3.15 compared to 2.79. This is consistent with the two time-series analysis methods as the smoothing method is going to give a more conservative estimate of a team's performance tending towards the team's overall mean performance whereas the ARIMA method can provide a larger range of predictions depending on the model fit. The seed method outperforms the ARIMA and ETS methods on average by 0.11200 and 0.00071 games respectively in round one, and by 1.667 and 1.111 games in round two. This can be indicative of the fact that the time-series methods have no regard to the underlying assumption of strength by seed position and

might predict more upsets than needed. It could also be seen from Figure 2 that the time-series methods were overpredicting the value of seeds above 13, as we can see from the difference between the estimated win probabilities for the time-series methods and the historical win probabilities. If we assume that we will lean heavily towards favorites when the difference in seeding is large, we can then also look at the predictiveness on just the closer matchups, or any first round matchup where the difference between the higher seed and lower seed is less than 7.

Close Matchup Games Correct

Year	Seed Close Matchup Games Correct	ARIMA Close Matchup Games Correct	ETS Close Matchup Games Correct
2011	10	10	10
2012	9	10	10
2013	9	6	9
2014	9	10	8
2015	13	11	11
2016	6	11	11
2017	10	7	8
2018	10	10	10
2019	5	7	7

Table 4 – Close Matchups Comparison

We can see in Table 4, that the difference of predictiveness between baseline seed method and the time-series methods is much tighter when looking at these close matchups only. Overall, the seed method predicts an average of 9 close matchup games correctly, while the ARIMA and ETS method predict an average of 9.11 and 9.33 close matchup games correctly. This provides some evidence that the time-series methods are more effective at predictions with closer seeded matchups compared to farther seeded matchups.

Upsets

Upsets are one thing that are hard to predict in a March Madness tournament and any method that has insight into an upset can be valuable. The seed method does not have the ability to predict these upsets, but we can see how effective the time-series methods are in predicting these matchups. An upset in the first round will be defined as a higher value seed beating a lower value seed, for example a 12 seed beating a 5 seed. In the second round, an upset will be defined as the winning team being any seed that was not expected at the start of the tournament; therefore, this implies an upset will be if a seed that has a value of 5 or higher wins, regardless of their opponent.

Percent of Upset Games Correctly Predicted

Year	ARIMA Method % Correct	ETS Method % Correct
2011	30.77%	23.08%
2012	20.00%	26.67%
2013	6.67%	20.00%
2014	35.71%	21.43%
2015	33.33%	41.67%
2016	36.84%	31.58%
2017	10.00%	20.00%
2018	16.67%	22.22%
2019	28.57%	28.57%

Table 5 – Percent of Upsets Predicted

On average the ARIMA method predicted 24.3% of all upsets in the first two rounds of the March Madness tournament over these nine years, while the ETS method predicted 26.1% of upsets. These numbers can be useful, but more importantly, we would want to know the number

of upsets correctly guessed divided by the total amount of predicted upsets by the method. This would be more useful because if you were deciding to use this model, it could be good to know how accurate the upset picks of the model are.

Percent of Predicted Upsets Correct

Year	ARIMA Method % Correct	ETS Method % Correct
2011	44.44%	42.86%
2012	30.00%	30.77%
2013	14.29%	37.50%
2014	50.00%	30.00%
2015	40.00%	50.00%
2016	53.85%	60.00%
2017	7.69%	22.22%
2018	37.50%	44.44%
2019	57.14%	80.00%

Table 6 – Accuracy of Predicted Upsets

Overall, the ARIMA method is correct about its upsets 37.2% of the time while the ETS method is correct 44.2% of the time. The standard deviation of the ARIMA method is 0.176 while the ETS is at 0.171. In certain situations this proves that the time-series methods are helpful in predicting upsets and in others it is not. For example, if we are looking at a seed 8 vs seed 9 matchup, the ~45% accuracy might not be as helpful because 8 vs. 9 is already a very close matchup. Now, if the model predicts an upset for a 6 vs. 11 or 5 vs. 12, then the model's 45% accuracy becomes more useful. Without any prior knowledge it is very difficult to predict an upset of that nature, but if the model says there is a 45% chance that this upset happens then you might become much more inclined to do more research into that matchup.

2023 Results

The method can be applied to the current March Madness tournament. In the first round the ARIMA method predicts three first round upsets, two of which are seed 7 vs. 10 matchups and one which is seed 8 vs. 9. For the second round the ARIMA method predicts six upsets with the largest upset being 10 seed USC beating 2 seed Marquette. The ETS method predicts four upsets in the first round, all of which are either 7 vs. 10 or 8 vs. 9. In the second round, the ETS method predicts six upsets as well, with a high probability prediction on 7 seed Texas A&M beating 2 seed Texas. This year's tournament prediction results by the three methods are shown in Table 7:

2023 Results

	Seed Method	ARIMA Method	ETS Method
Number of Games Correct	34	33	34
Percent of Predicted Upsets Correct	0%	36%	40%
R1 Close Games Correct	12	10	11

Table 7 – 2023 Results

In this year's tournament, the number of games correct were similar among all three methods. The method averages for Seed, ARIMA, ETS are 33.56, 31.78, and 32.44 respectively, therefore all methods predicted more games correctly than expected. The time-series methods performed almost exactly as expected in percentage of upsets correctly predicted. For close game predictions (seven or less difference between seeds in the first round) the time-series methods struggled, while picking the higher seed was a better option. There were some historical upsets this year with 16 seed Fairleigh Dickinson beating 1 seed Purdue, 15 seed Princeton beating 2

seed Arizona, and 13 seed Furman beating 4 seed Virginia. The ETS method gave a 32.1%, 41.5%, and 49.3% chance of these upsets happening, respectively. Princeton would go on to beat 7 seed Missouri, and the ETS model gave Princeton a 46.8% of winning that matchup, not accounting for previous matchups. This shows that Princeton was probably under-seeded (given too high of a seed) and most likely deserved a seed between 12 to 14.

Chapter 7

Conclusions

Limitations

There are many things to consider when carrying out this method of prediction in college basketball. A major limitation to this method was the exclusion of conference games from the time-series model underlying data. As discussed, it is hard to decide how to incorporate conference games into a prediction for the March Madness tournament. Teams should be rewarded for doing well in a tournament format late in the season, but there are a couple things to consider when doing this. All automatic qualifiers will have an artificially high time-series model prediction. Some teams might deserve this prediction, for example teams that win in tough conference tournaments. Others do not, for example teams from low strength one-bid conferences. Teams in those types of conferences will have an artificially high prediction because they must win their conference tournament, therefore their time series is guaranteed to have a series of wins at the end. Also, many at-large bids from tougher conferences that were already guaranteed a March Madness bid before their conference tournament began will most likely accrue a loss in their conference tournament therefore punishing their time-series trend. With no clear way to reward or punish teams for their conference tournament performance, we are limited to only using regular season games.

Another limitation of this statistical method for prediction is the lack of data points we can use in the time-series model. While there are no strict rules for data points needed in an

ARIMA model, it can be mentioned that it can be harder to estimate higher parameter time-series models with small amounts of data. With only about 30 data points for the whole season it can be harder to identify trends in the ARIMA model, so any high parameter model will produce poor estimation errors, limiting us to more basic ARIMA models. Using the ETS method helps with this as well because exponential time smoothing is useful when there is no clear trend in the data, which can be better for small amounts of data.

Access to data used for testing on the model was also a challenge. For this paper, nine years of data are used to test the predictiveness of the model and this is because of lack of access to game data from years prior to 2011. It is unclear how many years of back testing are feasible even if the data was available. Rule changes over the years change the nature of the underlying efficiency metric calculations.

Future Considerations

There are steps that can be taken in the future to make this model more consistent toward the overall prediction of the March Madness tournament. Going forward, the focus would be solely on the ETS method due to the flexibility of the time-series analysis assumptions and the nature of the method is good for the data as the ETS method is heavily weighted to recent periods making it good for short-term forecasting, which is what we want with our data. An improvement that can be made is refining the specifics of the model equation. Reiterating, the current model used is an automatically selected error type, an automatically selected trend type, and an assumption of no seasonal trend. Going forward, fixing the model to one specific type, for

example, a strictly additive model or strictly multiplicative model could be considered. Other smoothing methods can be explored, like Hyndman's method for automatically selecting an ETS model. Many others have time-smoothing methods that usually have some adjustments to a general ETS model. For example, there is the THETA method, created by Assimakopoulos and Nikolopoulos (Assimakopoulos, 2000), which is essentially a single exponential smoothing model with drift.

Another consideration going forward that could help the predictiveness of the model is finding some way to give more weight to the probability of success for better seeds. As we can see in Figure 2, and as mentioned in the analysis of close matchups, this model underestimates the probability of success of better seeds, particularly seeds 1 through 4. If the probability of success curve for the time-series model were to be pulled closer to the historical probabilities of success among these top seeds, the predictive power of these models would be greater.

There is also the case of the underlying efficiency calculations. There is a limitation to the game efficiency calculation used, due to the availability of data at the exact time before a tournament, as explained in Chapter 2. For a future model, an exploration of using different types of efficiency calculations to see if other efficiencies are more suited for a time-series model could be done. Also, a lot of rating systems account for large margin of victory mismatches and discount their rating accordingly. This could play a significant part in the trend detection for the time-series model and the efficiencies used in this paper did not have a margin of victory adjustment. Of course, this would come with the complication of collecting the data like it was a problem for this paper.

Conclusions and Recommendations

As a reiteration of a commonly made point, the statistical methods used in this paper are for predictive purposes and it is not created in a typical way by creating a model that is trained on some data and tested on others. Each team's game performance is its training data. The nature of using a flexible time-series model is important because we know that a given team's performance never acts the same compared to another team.

Using the predictions from the time-series model and the probabilistic model, we found that both time-series methods compare, on average, to a baseline method where you always pick the favorite seed. Between the two time-series methods themselves, the ETS method seems to perform slightly better, especially when it comes to variance in the results across the years. Specifically, the ETS has a 0.667 game advantage when it comes to predicting the first 48 games, and a 0.365 reduction in standard deviation. When recommending how this model would be used, it is important to look at the concept of this method. As stated in the introduction, the purpose of this model is to quantify the question, “What teams are hot coming into the tournament?”. When you are predicting the first two rounds you might not want to follow the model if it suggests that a 14, 15, or 16 seed will win because we know historically this doesn’t happen too much at all. Where this model’s prediction will become helpful will be in the games that are hard to decide on, like the 4 vs. 13, 5 vs. 12, and the other middle-seed matchups. It is found that the model is on average 45% correct when it predicts an upset, so if a person was trying to pick their upsets, the model would give a good insight into the matchups that could end in an upset. Again, the accuracy of picks is on average comparable with a baseline picking seed method, but when used correctly, this model can give good insights into which teams might have

improved over the year versus teams that have stayed stagnant or have been regressing, and this is something other methods usually do not model.

Appendix A
R Code

```
library(dplyr)
library(tidyverse)
library(data.table)
library(forecast)
library(kableExtra)
library(reshape2)
library(ggplot2)
library(scales)

rm(list=ls())

#This file can be ran all the way through, it will produce many data sets,
#the compare_results one gives overall games predicted for each method in each round

#This will run and put together the advanced data file
# #Pull in Advanced Data
# year = 2011
# gamedataadvanced <- data.frame()
#
# while (year <= 2022) {
#   if (year == 2020) {
#     year = year+1
#   } else {
#     str1 = as.character(year)
#     str2 = "_super_sked.csv"
#     dataread <- paste(str1,str2, sep = "")
#     data <- read_csv(dataread,col_names = FALSE)
#     gamedataadvanced <- rbind(gamedataadvanced, data)
#     year = year+1
#   }
# }
#
# names(gamedataadvanced) <- c('muid', 'date', 'conmatch', 'matchup', 'prediction', 'ttq', 'conf', 'venue',
# 'team1', 't1oe', 't1de', 't1py', 't1wp', 't1propt', 'team2', 't2oe', 't2de', 't2py', 't2wp',
# 't2propt', 'tpro', 't1qual', 't2qual', 'gp', 'result', 'tempo', 'possessions', 't1pts',
# 't2pts', 'winner', 'loser', 't1adjt', 't2adjt', 't1adjo', 't1adjd', 't2adjo', 't2adjd',
# 'gamevalue', 'mismatch', 'blowout', 't1elite', 't2elite', 'ord_date', 't1ppp', 't2ppp', 'gameppp',
# 't1rk', 't2rk', 't1gs', 't2gs', 'gamestats', 'overtimes', 't1fun', 't2fun', 'results')
#
# #Write for Match
# write.csv(gamedataadvanced, "gamedataadvanced.csv", row.names=FALSE)
##############################################################################################
###################
#I made some manual changes in excel to the advanced data file now I bring it back here for the rest of
cleaning
```

```
gamedataadvanced <- read_csv("gamedataadvanced.csv")

#REF TABLES
team_ref <- read_csv("MTeams.csv")
city_ref <- read_csv("Cities.csv")
conf_ref <- read_csv("Conferences.csv")
region_ref <- read_csv("MSeasons.csv")
seed_ref <- read_csv("MNCAATourneySeeds.csv")
adv_id <- read_csv("AdvTeams.csv",col_names = FALSE)

#Game Data
city_results <- read_csv("MGameCities.csv")
tourney_box_results <- read_csv("MNCAATourneyDetailedResults.csv")
regular_box_results <- read_csv("MRegularSeasonDetailedResults.csv")
prob <- read_csv("markov_opp.csv")

adv_id <- rename(adv_id, winner = X1)
gamedataadvanced <- merge(gamedataadvanced, adv_id, by="winner")
gamedataadvanced <- rename(gamedataadvanced, WinId = X2)
adv_id <- rename(adv_id, loser = winner)
gamedataadvanced <- merge(gamedataadvanced, adv_id, by="loser")
gamedataadvanced <- rename(gamedataadvanced, LoseId = X2)

tourney_box_results$Type <- "Tourney"
regular_box_results$Type <- "Regular"

gamedataregular <- bind_rows(regular_box_results,tourney_box_results)
gamedataadvanced$join1 <-
paste(gamedataadvanced$date,gamedataadvanced$WinId,gamedataadvanced$LoseId)
gamedataregular$join1 <-
paste(gamedataregular$date,gamedataregular$WTeamID,gamedataregular$LTeamID)

full <- merge(gamedataadvanced, gamedataregular, by='join1', all.x=TRUE)

tidy <- full %>%
select(winner,loser,date.x,conmatch,prediction,team1,t1adjo,t1adjd,team2,t2adjo,t2adjd,Season,conf,Type
,WTeamSeed,LTeamSeed,City)
tidy$date.x <- as.POSIXct(tidy$date.x, format="%m/%d/%Y")
#Creating seeding example
i=2011
ts <- {}
ts2 <- {}
ets <- {}
ets2 <- {}
seed <- {}
seed2 <- {}
yearw <- {}
while (i <= 2019) {
```

```
tourney <- tidy %>%
  group_by(winner,loser,WTeamSeed,LTeamSeed) %>%
  filter(Season == i & conf == 3) %>%
  ungroup()
tourney <- tourney %>%
  arrange(date.x)
tourney <- tourney[5:52,]
tourney$WTeamSeed <- substr(tourney$WTeamSeed, 1, 3)
tourney$LTeamSeed <- substr(tourney$LTeamSeed, 1, 3)
list1 <- tourney$winner
list2 <- tourney$loser
appended_list <- append(list1, list2)
appended_list <- as.data.frame(appended_list)
appended_list <- unique(appended_list)

df1 <- dplyr::select(tourney,winner,WTeamSeed)
df1 <- rename(df1, team = winner,seed = WTeamSeed)
df2 <- select(tourney,loser,LTeamSeed)
df2 <- rename(df2, team = loser,seed= LTeamSeed)
df3 <- bind_rows(df1,df2)

appended_list <- rename(appended_list, team = appended_list)
appended_list <- merge(appended_list, df3, by='team', all.x=TRUE)
tourney_teams <- unique(appended_list)

pred <- {}
pred2 <- {}
pred3 <- {}
for (z in tourney_teams$team) {
  regular1 <- tidy %>%
    group_by(team1,date.x) %>%
    filter(Season == i & (conf == 1 | conf == 0) & team1 == z) %>%
    ungroup() %>%
    select(t1adjo,t1adjd,date.x)

  regular2 <- tidy %>%
    group_by(team2,date.x) %>%
    filter(Season == i & (conf == 1 | conf == 0) & team2 == z) %>%
    ungroup() %>%
    select(t2adjo,t2adjd,date.x)
  regular1 <- rename(regular1, adjo = t1adjo,adjd = t1adjd)
  regular2 <- rename(regular2, adjo = t2adjo,adjd = t2adjd)
  regular3 <- bind_rows(regular1,regular2)
  regular3 <- regular3 %>%
    arrange(date.x)

  regular3$adj <- regular3$adjo - regular3$adjd

  myts <- ts(regular3$adj, frequency=1)
```

```
  MA <- auto.arima(myts,allowdrift = TRUE,allowmean = TRUE, seasonal = FALSE)
  MA2 <- ets(myts,model = "ZZN", allow.multiplicative.trend = TRUE)

  # ts.plot(myts)
  # points(MA$fitted, type = "l", col = 2, lty = 2)
  # points(MA2$fitted, type = "l", col = 2, lty = 2)

  predict_MA <- forecast(MA, h = 1)
  predict_MA2 <- forecast(MA2, h = 1)
  pred <- append(pred,predict_MA$mean[1])
  pred2 <- append(pred2,predict_MA2$mean[1])

}
tourney_teams$pred1 <- parse_number(tourney_teams$seed)
tourney_teams$pred2 <- pred
tourney_teams$pred2 <- tourney_teams$pred2 + (1-min(tourney_teams$pred2))
tourney_teams$pred3 <- pred2
tourney_teams$pred3 <- tourney_teams$pred3 + (1-min(tourney_teams$pred3))

tourney_teams <- tourney_teams %>% distinct()


year <- paste("tourney",as.character(i), sep = "")
assign(year,tourney_teams)

mkv <- merge(prob, tourney_teams %>% select(seed, team, pred1), by.x = "teamseed", by.y = "seed")
mkv <- rename(mkv, prob1 = pred1)
mkv <- merge(mkv, tourney_teams %>% select(seed, pred1), by.x = "opp1", by.y = "seed")
mkv <- rename(mkv, prob2 = pred1)
mkv <- merge(mkv, tourney_teams %>% select(seed, pred1), by.x = "opp2", by.y = "seed")
mkv <- rename(mkv, prob3 = pred1)
mkv <- merge(mkv, tourney_teams %>% select(seed, pred1), by.x = "opp3", by.y = "seed")
mkv <- rename(mkv, prob4 = pred1)

mkv$pr1 <- mkv$prob2 / (mkv$prob1 + mkv$prob2)
mkv$pr2 <- mkv$pr1*(((mkv$prob4/(mkv$prob3 + mkv$prob4))*(mkv$prob3/(mkv$prob1 +
mkv$prob3)))+((mkv$prob3/(mkv$prob3 + mkv$prob4))*(mkv$prob4/(mkv$prob1 + mkv$prob4))))
mkv <- mkv %>%
  select(team,teamseed,pr1,pr2)
year <- paste("seed",as.character(i), sep = "")
assign(year,mkv)

winseed <- merge(prob, mkv %>% select(teamseed, pr1) , by.x = "teamseed", by.y = "teamseed")
winseed <- rename(winseed, t1 = pr1)
winseed <- merge(winseed, mkv %>% select(teamseed, pr1) , by.x = "opp1", by.y = "teamseed")
winseed <- rename(winseed, t2 = pr1)
winseed$winr1 <- ifelse((winseed$t1 > winseed$t2),winseed$teamseed,ifelse((winseed$t2 >
winseed$t1),winseed$opp1,0))
winseed <- select(winseed, teamseed, opp1, opp2, opp3, winr1)
```

```
  winseed <- merge(winseed, mkv %>% select(teamseed, pr2) , by.x = "teamseed", by.y = "teamseed")
  winseed <- rename(winseed, t1 = pr2)
  winseed <- merge(winseed, mkv %>% select(teamseed, pr2) , by.x = "opp1", by.y = "teamseed")
  winseed <- rename(winseed, t2 = pr2)
  winseed <- merge(winseed, mkv %>% select(teamseed, pr2) , by.x = "opp2", by.y = "teamseed")
  winseed <- rename(winseed, t3 = pr2)
  winseed <- merge(winseed, mkv %>% select(teamseed, pr2) , by.x = "opp3", by.y = "teamseed")
  winseed <- rename(winseed, t4 = pr2)
  winseed$winr2 <- ifelse((winseed$t1 > winseed$t2)&(winseed$t1 > winseed$t3)&(winseed$t1 >
winseed$t4),winseed$teamseed,ifelse((winseed$t2 > winseed$t1)&(winseed$t2 >
winseed$t3)&(winseed$t2 > winseed$t4),winseed$opp1,ifelse((winseed$t3 > winseed$t1)&(winseed$t3
> winseed$t2)&(winseed$t3 > winseed$t4),winseed$opp2,ifelse((winseed$t4 >
winseed$t1)&(winseed$t4 > winseed$t2)&(winseed$t4 > winseed$t3),winseed$opp3,0))))
  winseed <- select(winseed, winr1, winr2)
  uniqueseedr1 <- as.data.frame(unique(winseed$winr1))
  uniqueseedr1 <- rename(uniqueseedr1, winr1 = "unique(winseed$winr1)")
  uniqueseedr1$test <- 1
  uniqueseedr2 <- as.data.frame(unique(winseed$winr2))
  uniqueseedr2 <- rename(uniqueseedr2, winr2 = "unique(winseed$winr2)")
  uniqueseedr2$test <- 1


  test <- tourney %>%
    arrange(date.x)
  test <- test[1:32,]
  test <- merge(test, uniqueseedr1, by.x = "WTeamSeed", by.y = "winr1", all.x = T)
  test$test <- ifelse(is.na(test$test),0,test$test)
  testseed <- sum(test$test)/32

  year <- paste("seed",as.character(i),"r1", sep = "")
  assign(year,testseed)
  year <- paste("predictseedr1",as.character(i), sep = "")
  assign(year,test)


  test <- tourney %>%
    arrange(date.x)
  test <- test[33:48,]
  test <- merge(test, uniqueseedr2, by.x = "WTeamSeed", by.y = "winr2", all.x = T)

  test$test <- ifelse(is.na(test$test),0,test$test)
  testseed2 <- sum(test$test)/16

  year <- paste("seed",as.character(i),"r2", sep = "")
  assign(year,testseed2)
  year <- paste("predictseedr2",as.character(i), sep = "")
  assign(year,test)
```

```
  mkv2 <- merge(prob, tourney_teams %>% select(seed, team, pred2), by.x = "teamseed", by.y = "seed")
  mkv2 <- rename(mkv2, prob1 = pred2)
  mkv2 <- merge(mkv2, tourney_teams %>% select(seed, pred2), by.x = "opp1", by.y = "seed")
  mkv2 <- rename(mkv2, prob2 = pred2)
  mkv2 <- merge(mkv2, tourney_teams %>% select(seed, pred2), by.x = "opp2", by.y = "seed")
  mkv2 <- rename(mkv2, prob3 = pred2)
  mkv2 <- merge(mkv2, tourney_teams %>% select(seed, pred2), by.x = "opp3", by.y = "seed")
  mkv2 <- rename(mkv2, prob4 = pred2)

  mkv2$pr1 <- mkv2$prob1 / (mkv2$prob1 + mkv2$prob2)
  mkv2$pr2 <- mkv2$pr1*(((mkv2$prob3/(mkv2$prob3 + mkv2$prob4))*(mkv2$prob1/(mkv2$prob1 +
mkv2$prob3)))+((mkv2$prob4/(mkv2$prob3 + mkv2$prob4))*(mkv2$prob1/(mkv2$prob1 +
mkv2$prob4))))
  mkv2 <- mkv2 %>%
    select(team,teamseed,pr1,pr2)
  year <- paste("ts",as.character(i), sep = "")
  assign(year,mkv2)


  wints <- merge(prob, mkv2 %>% select(teamseed, pr1) , by.x = "teamseed", by.y = "teamseed")
  wints <- rename(wints, t1 = pr1)
  wints <- merge(wints, mkv2 %>% select(teamseed, pr1) , by.x = "opp1", by.y = "teamseed")
  wints <- rename(wints, t2 = pr1)
  wints$winr1 <- ifelse((wints$t1 > wints$t2),wints$teamseed,ifelse((wints$t2 > wints$t1),wints$opp1,0))
  wints <- select(wints, teamseed, opp1, opp2, opp3, winr1)
  wints <- merge(wints, mkv2 %>% select(teamseed, pr2) , by.x = "teamseed", by.y = "teamseed")
  wints <- rename(wints, t1 = pr2)
  wints <- merge(wints, mkv2 %>% select(teamseed, pr2) , by.x = "opp1", by.y = "teamseed")
  wints <- rename(wints, t2 = pr2)
  wints <- merge(wints, mkv2 %>% select(teamseed, pr2) , by.x = "opp2", by.y = "teamseed")
  wints <- rename(wints, t3 = pr2)
  wints <- merge(wints, mkv2 %>% select(teamseed, pr2) , by.x = "opp3", by.y = "teamseed")
  wints <- rename(wints, t4 = pr2)
  wints$winr2 <- ifelse((wints$t1 > wints$t2)&(wints$t1 > wints$t3)&(wints$t1 >
wints$t4),wints$teamseed,ifelse((wints$t2 > wints$t1)&(wints$t2 > wints$t3)&(wints$t2 >
wints$t4),wints$opp1,ifelse((wints$t3 > wints$t1)&(wints$t3 > wints$t2)&(wints$t3 >
wints$t4),wints$opp2,ifelse((wints$t4 > wints$t1)&(wints$t4 > wints$t2)&(wints$t4 >
wints$t3),wints$opp3,0))))
  wints <- select(wints, winr1, winr2)
  uniquetsr1 <- as.data.frame(unique(wints$winr1))
  uniquetsr1 <- rename(uniquetsr1, winr1 = "unique(wints$winr1)")
  uniquetsr1$test <- 1
  uniquetsr2 <- as.data.frame(unique(wints$winr2))
  uniquetsr2 <- rename(uniquetsr2, winr2 = "unique(wints$winr2)")
  uniquetsr2$ind <- parse_number(uniquetsr2$winr2)
  uniquetsr2$ind <- ifelse(uniquetsr2$ind >= 5 ,1,0)
  upset <- sum(uniquetsr2$ind)
  year <- paste("upsetts",as.character(i), sep = "")
  assign(year,upset)
```

```
uniquetsr2 <- select(uniquetsr2, winr2)
uniquetsr2$test <- 1


test <- tourney %>%
  arrange(date.x)
test <- test[1:32,]
test <- merge(test, uniquetsr1, by.x = "WTeamSeed", by.y = "winr1", all.x = T)
test$test <- ifelse(is.na(test$test),0,test$test)
testts <- sum(test$test)/32
year <- paste("ts",as.character(i),"r1", sep = "")
assign(year,testts)
year <- paste("predicttsr1",as.character(i), sep = "")
assign(year,test)



test <- tourney %>%
  arrange(date.x)
test <- test[33:48,]
test <- merge(test, uniquetsr2, by.x = "WTeamSeed", by.y = "winr2", all.x = T)
test$test <- ifelse(is.na(test$test),0,test$test)

testts2 <- sum(test$test)/16
year <- paste("ts",as.character(i),"r2", sep = "")
assign(year,testts2)
year <- paste("predicttsr2",as.character(i), sep = "")
assign(year,test)


mkv3 <- merge(prob, tourney_teams %>% select(seed, team, pred3), by.x = "teamseed", by.y = "seed")
mkv3 <- rename(mkv3, prob1 = pred3)
mkv3 <- merge(mkv3, tourney_teams %>% select(seed, pred3), by.x = "opp1", by.y = "seed")
mkv3 <- rename(mkv3, prob2 = pred3)
mkv3 <- merge(mkv3, tourney_teams %>% select(seed, pred3), by.x = "opp2", by.y = "seed")
mkv3 <- rename(mkv3, prob3 = pred3)
mkv3 <- merge(mkv3, tourney_teams %>% select(seed, pred3), by.x = "opp3", by.y = "seed")
mkv3 <- rename(mkv3, prob4 = pred3)

mkv3$pr1 <- mkv3$prob1 / (mkv3$prob1 + mkv3$prob2)
mkv3$pr2 <- mkv3$pr1*(((mkv3$prob3/(mkv3$prob3 + mkv3$prob4))*(mkv3$prob1/(mkv3$prob1 +
mkv3$prob3)))+((mkv3$prob4/(mkv3$prob3 + mkv3$prob4))*(mkv3$prob1/(mkv3$prob1 +
mkv3$prob4))))
mkv3 <- mkv3 %>%
  select(team,teamseed,pr1,pr2)
year <- paste("ets",as.character(i), sep = "")
assign(year,mkv3)

winets <- merge(prob, mkv3 %>% select(teamseed, pr1) , by.x = "teamseed", by.y = "teamseed")
```

```
  winets <- rename(winets, t1 = pr1)
  winets <- merge(winets, mkv3 %>% select(teamseed, pr1) , by.x = "opp1", by.y = "teamseed")
  winets <- rename(winets, t2 = pr1)
  winets$winr1 <- ifelse((winets$t1 > winets$t2),winets$teamseed,ifelse((winets$t2 >
winets$t1),winets$opp1,0))
  winets <- select(winets, teamseed, opp1, opp2, opp3, winr1)
  winets <- merge(winets, mkv3 %>% select(teamseed, pr2) , by.x = "teamseed", by.y = "teamseed")
  winets <- rename(winets, t1 = pr2)
  winets <- merge(winets, mkv3 %>% select(teamseed, pr2) , by.x = "opp1", by.y = "teamseed")
  winets <- rename(winets, t2 = pr2)
  winets <- merge(winets, mkv3 %>% select(teamseed, pr2) , by.x = "opp2", by.y = "teamseed")
  winets <- rename(winets, t3 = pr2)
  winets <- merge(winets, mkv3 %>% select(teamseed, pr2) , by.x = "opp3", by.y = "teamseed")
  winets <- rename(winets, t4 = pr2)
  winets$winr2 <- ifelse((winets$t1 > winets$t2)&(winets$t1 > winets$t3)&(winets$t1 >
winets$t4),winets$teamseed,ifelse((winets$t2 > winets$t1)&(winets$t2 > winets$t3)&(winets$t2 >
winets$t4),winets$opp1,ifelse((winets$t3 > winets$t1)&(winets$t3 > winets$t2)&(winets$t3 >
winets$t4),winets$opp2,ifelse((winets$t4 > winets$t1)&(winets$t4 > winets$t2)&(winets$t4 >
winets$t3),winets$opp3,0))))
  winets <- select(winets, winr1, winr2)
  uniqueetsr1 <- as.data.frame(unique(winets$winr1))
  uniqueetsr1 <- rename(uniqueetsr1, winr1 = "unique(winets$winr1)")
  uniqueetsr1$test <- 1
  uniqueetsr2 <- as.data.frame(unique(winets$winr2))
  uniqueetsr2 <- rename(uniqueetsr2, winr2 = "unique(winets$winr2)")
  uniqueetsr2$ind <- parse_number(uniqueetsr2$winr2)
  uniqueetsr2$ind <- ifelse(uniqueetsr2$ind >= 5 ,1,0)
  upset <- sum(uniqueetsr2$ind)
  year <- paste("upsetets",as.character(i), sep = "")
  assign(year,upset)
  uniqueetsr2 <- select(uniqueetsr2, winr2)
  uniqueetsr2$test <- 1


  test <- tourney %>%
    arrange(date.x)
  test <- test[1:32,]
  test <- merge(test, uniqueetsr1, by.x = "WTeamSeed", by.y = "winr1", all.x = T)
  test$test <- ifelse(is.na(test$test),0,test$test)
  testets <- sum(test$test)/32
  year <- paste("ets",as.character(i),"r1", sep = "")
  assign(year,testets)
  year <- paste("predictetsr1",as.character(i), sep = "")
  assign(year,test)

  test <- tourney %>%
    arrange(date.x)
  test <- test[33:48,]
  test <- merge(test, uniqueetsr2, by.x = "WTeamSeed", by.y = "winr2", all.x = T)
```

```
  test$test <- ifelse(is.na(test$test),0,test$test)

  testets2 <- sum(test$test)/16
  year <- paste("ets",as.character(i),"r2", sep = "")
  assign(year,testets2)
  year <- paste("predictetsr2",as.character(i), sep = "")
  assign(year,test)

  ts<-append(ts,testts)
  ts2<-append(ts2,testts2)
  ets<-append(ets,testets)
  ets2<-append(ets2,testets2)
  seed<-append(seed,testseed)
  seed2<-append(seed2,testseed2)
  yearw <- append(yearw,i)


  i = i + 1
}

compare_results <- data.frame(yearw)
compare_results$ts <- ts
compare_results$ts2 <- ts2
compare_results$ets <- ets
compare_results$ets2 <- ets2
compare_results$seed <- seed
compare_results$seed2 <- seed2
compare_results$tsnum <- (ts*32) + (ts2*16)
compare_results$etsnum <- (ets*32) + (ets2*16)
compare_results$seednum <- (seed*32) + (seed2*16)
```

BIBLIOGRAPHY

Assimakopoulos, V., & Nikolopoulos, K. (2000). The Theta model: A decomposition approach to forecasting. International Journal of Forecasting, 16(4), 521–530. https://doi.org/10.1016/s0169-2070(00)00066-2

Boyd, J. (2023, March 13). March madness records by seed: Probability of each making final Four. Retrieved March 15, 2023, from www.boydsbets.com/bracket-tips-by-seed/

History of NCAA Basketball Rule Changes. NCAA basketball rule change history. (n.d.). Retrieved March 15, 2023, from www.orangehoops.org/NCAA/NCAA%20Rule%20Changes.htm

Hyndman R, Athanasopoulos G, Bergmeir C, Caceres G, Chhay L, O'Hara-Wild M, Petropoulos F, Razbash S, Wang E, Yasmeen F (2023)._forecast: Forecasting functions for time series and linear models_. R package version 8.21, <URL: https://pkg.robjhyndman.com/forecast/>.

Hyndman, R.J., & Athanasopoulos, G. (2018) Forecasting: principles and practice, 2nd edition, OTexts: Melbourne, Australia. Retrieved March 15, 2023, from OTexts.com/fpp2

Hyndman, R. J., & Khandakar, Y. (2008). Automatic Time Series Forecasting: The forecast Package for R. Journal of Statistical Software, 27(3). https://doi.org/10.18637/jss.v027.i03

Hyndman, R. J., Koehler, A. B., Snyder, R. D., & Grose, S. (2002). A state space framework for automatic forecasting using exponential smoothing methods. International Journal of Forecasting, 18(3), 439–454. https://doi.org/10.1016/s0169-2070(01)00110-8

Miller, S. J. (2007). A derivation of the Pythagorean won-loss formula in baseball. CHANCE, 20(1), 40–48. https://doi.org/10.1080/09332480.2007.10722831

NCAA.com, D. W. (2023, March 15). What is March madness: The NCAA tournament explained. NCAA.com. Retrieved March 15, 2023, from www.ncaa.com/news/basketball-men/bracketiq/2022-03-14/what-march-madness-ncaa-tournament-explained#:~:text=The%20NCAA%20Division%20I%20men's%20basketball%20tournament%20is%20a%20single,only%20four%20teams%20are%20left

Ord, J. K., Koehler, A. B., & Snyder, R. D. (1997). Estimation and prediction for a class of Dynamic Nonlinear Statistical Models. Journal of the American Statistical Association, 92(440), 1621–1629. https://doi.org/10.1080/01621459.1997.10473684

Pomeroy, K. (2004, April 19). The possession: The Kenpom.com blog. The kenpomcom blog. Retrieved March 16, 2023, from https://kenpom.com/blog/the-possession/

Pomeroy, K. (2006, November 29). Ratings Explanation. Retrieved March 15, 2023, from https://kenpom.com/blog/ratings-explanation/

Pomeroy, K. (2016, October 4). Ratings methodology update. Retrieved March 15, 2023, from https://kenpom.com/blog/ratings-methodology-update/

R Core Team (2021). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Roundtable Technology Solutions. (n.d.). RATINGS METHODOLOGY. College Football & basketball ratings. Retrieved March 15, 2023, from www.rtsratings.com/method.htm

Sagarin, J. (2023, March 15). College basketball team ratings 2022-23 - jeff sagarin ratings. USA Today. Retrieved March 15, 2023, from https://sagarin.usatoday.com/2023-2/college-basketball-team-ratings-2022-23/

Schwertman, N. C., McCready, T. A., & Howard, L. (1991). Probability models for the NCAA Regional Basketball Tournaments. The American Statistician, 45(1), 35. https://doi.org/10.2307/2685236

Torvik, B. (n.d.). Rank FAQ. Retrieved March 16, 2023, from http://adamcwisports.blogspot.com/p/every-possession-counts.html#:~:text=How%20is%20T%2DRank%20calculated,as%20points%20per%20100%20possessions).

ACADEMIC VITA

Christopher Miller

Education

Pennsylvania State University, Schreyer Honors College University Park, PA
B.S. Mathematics, B.S. Applied Statistics, Minor Economics Expected May 2023

Awards

- Schreyer Honors College Scholarship
- Matthew Rosenshine Fund for Excellence in Statistics
- Phi Beta Kappa – National Honors Society
- Rensselaer Medal Award (Math and Science Excellence)
- NYS Scholarship for Academic Excellence
- Dean's List

Relevant Experience

MetLife Virtual - Bridgewater, NJ
Actuarial Intern (Disability Pricing) May 2022-Current

- Assisted with multiple projects that involved analysis with Excel VBA code, Python scripts, SQL, Alteryx, and R
- Improved disability claim predictive model in R by increasing the effectiveness of model code and helping decision making for factor analysis to create a model that improves pricing rates

Ginsberg's Foods Hudson, NY
Data Analyst Intern May 2021-March 2022

- Worked for a mid-size regional food distributor and trained in a data analyst role
- Lead longer-term projects by providing insight to the company such as operations utilization tools and zone pricing analysis and logic
- Investigated KPI vs. Distance Metrics using R Markdown while collaborating with management such as the CEO, COO, and VP of Operations

Extracurriculars

Esports Varsity Team (Division Head) Aug 2019-Current

- Lead a division where I held bi-weekly meetings with members, worked on creating tournament opportunities for members and moderated the community within the division

Actuarial Science Club Aug 2019-Dec 2022

- Former Executive Board member of the club, also a mentor for a student