Spring 2023

An Exploratory Study on Methods for Interpolating and Extrapolating Baseball Win-Loss Percentage

Giselle Palacios

An Exploratory Study on Methods for Interpolating and Extrapolating Baseball Win-Loss Percentage

By

Giselle Palacios
B.S., Saint Xavier University, 2021

THESIS

Submitted in partial fulfillment of the requirements

For the Degree of Master of Science,
With a Major in Mathematics

Governors State University
University Park, IL 60484

2023

Acknowledgements

I would like to express my deepest gratitude to my committee chairperson, Dr. Tweddle, for his time and guidance throughout this semester. I genuinely enjoyed taking his Mathematical Modeling course during my first semester; it was this course, along with my interest in baseball, that inspired the writing of this paper. I wish to extend my sincere thanks to my committee members, Dr. Li and Dr. Morlet, for their encouraging words and valuable feedback. Without my committee's assistance and expertise, this thesis would have never been accomplished.

I am also extremely grateful to my family and friends for their love and support, and for helping me stay motivated throughout this process.

Table of Contents

List of Tables

List of Figures

Abstract

The 2020 Major League Baseball season was shortened by a few months due to travel restrictions implemented in response to Coronavirus pandemic. When baseball began in mid-July, the schedule had been changed so that teams were mostly playing against division opponents. This left some fans wondering how different the 2020 season would have looked if the original schedule of 162 games were played. A website called Strat-O-Matic used the Monte Carlo method to simulate a full 2020 season. This paper proposes and explores other possible models for predicting the outcome of the season. The first model we present is a Fourier series model that fits the teams' win-loss percentage (W-L%). We hypothesize that a team's W-L% can be modeled by a sum of cosine and sine curves as every team has winning seasons and losing seasons. The second model we suggest is a cubic spline model, which connects two adjacent W-L% data points with a cubic function. The last model we examine is a seasonal Auto-Regressive Integrated Moving Average (SARIMA) model that was generated following the Box-Jenkins method. With W-L% data from 1998 to 2021, we produce Fourier series, cubic spline, and SARIMA models for each team with the help of Python. The Fourier series and cubic spline model were employed to predict the 2020 W-L%. We analyzed the goodness of fit of the models by computing the sum of the absolute residuals, the sum of the absolute residuals squared, and the maximum absolute residual. Furthermore, we compared these two models to Strat-O-Matic's model. The purpose of the SARIMA model was to forecast the 2022 monthly W-L%. We evaluated this model by calculating the mean absolute error, root mean square error, and mean absolute percentage error. We concluded that using a Fourier series and cubic spline model to predict a team's 2020 W-L% and using a SARIMA model to forecast 2022 monthly W-L% is appropriate and satisfactory.

Keywords: Fourier series, cubic splines, seasonal ARIMA, baseball, win-loss percentage

1 Introduction

In March 2020, Major League Baseball (MLB) was in the midst of Spring Training when the Coronavirus pandemic put a stop to everything. Baseball did not resume until July 1, 2020, with Opening Night being July 23rd and the regular season ending on September 27th. Because the season was shortened by a few months, the teams only played 60 games instead of the usual 162 games, and to limit travel, they faced their division opponents for a majority of the games. Did these changes affect the outcome of the 2020 season? Were 60 games enough for a team to shine or decline? There have been times when a team had a slow start to the season, but then they later competed in playoffs because they had months to improve. The opposite can also happen where a team has a hot start to the season, but then eventually they fall off. Thus, one might believe that the outcome of the 2020 season would have been different if they had played the originally scheduled 162 games. There are models that can predict what the outcome could have been if the teams played a full season in 2020.

A popular modeling technique in the sports industry is the Monte Carlo simulation, a probabilistic model that uses random numbers. Monte Carlo simulation takes into account players' statistics, along with randomness, to estimate the number of runs scored by each team in a game; therefore, it has the ability to predict which team will win the game. Initially a board game that was introduced in 1961, Strat-O-Matic has grown to be a well-known sports website that used this approach to simulate the games that were originally supposed to take place in 2020 and created the postseason bracket based on the simulation results. Although Monte Carlo simulation seems to be a solid method for predicting the 2020 baseball season, this paper will investigate other possible models that are simple in that they only consider the teams' win-loss percentage (W-L%) over time. The win-loss percentage is calculated as

$$\frac{Wins}{Wins + Losses}$$

Throughout the years, a team's W-L% fluctuates as they have outstanding seasons and dismal seasons. It appears that this can be mathematically described by cosine and sine waves; therefore, we propose a Fourier series model as our first model. If we fit a curve composed of cosine and sine functions to yearly W-L% data, then we believe that we can approximate the 2020 W-L% for each team. The second model we suggest is cubic spline interpolation. We believe that by connecting data points with a cubic function, we can obtain an estimate of what the 2020 W-L% would have been. In the first portion of this thesis, we will develop a Fourier series model and cubic spline model for each baseball team in Python, and using data from 1998 to 2021, we will interpolate their 2020 W-L%. We will then determine which teams would have advanced to the postseason based on each model, and compare our results to those of Strat-O-Matic.

Now, the next question a devoted baseball fan or sports bettor would ask is, "How can you predict future W-L%?" The Fourier series model and cubic spline model cannot be used for extrapolation; thus, we propose a third model: seasonal Auto-Regressive Integrated Moving Average, or SARIMA. ARIMA is a common type of time series model that forecasts events based on historical data. It has been employed to forecast stock prices, crime, and most recently, Covid-19 cases. From our research, ARIMA is not used often in sports analytics, perhaps because more advanced baseball projection systems, such as PECOTA or ZiPS, are available and favored. We believe that with monthly W-L% data, a SARIMA model has the potential to make accurate forecasts. In the second part of this thesis, we will generate the best SARIMA model for each team with the assistance of Python, forecast the 2022 season rather than 2020, and deduce if it is an appropriate model for extrapolating W-L%.

2 Models for Interpolating

2.1 Data Collection

For the Fourier series model and the cubic spline model, we collected yearly W-L% for each MLB team from the Baseball Reference website. We started in the year 1998 because that was the first year that all 30 of the current baseball teams existed. We ended at the year 2021 and excluded 2020 as that is the year we wish to interpolate.

2.2 The Fourier Series Model

When data points have an oscillating pattern, such as in Figure 1, we can approximate a continuous curve using Fourier series. For a function with period T, a continuous Fourier series can be expressed as an infinite sum of sine and cosine functions given by

$$f(t) = a_0 + \sum_{k=1}^{\infty} a_k \cos(k\omega t) + b_k \sin(k\omega t)$$

The Fourier coefficients, a_0, a_k, and b_k, can be computed as

$$a_0 = \frac{1}{T}\int_0^T f(t)dt$$

Therefore, a_0 can be interpreted as the average function value between the period interval $[0, T]$ (Kadry, 2014).

$$a_k = \frac{2}{T}\int_0^T f(t)\cos(k\omega t)dt$$

$$b_k = \frac{2}{T}\int_0^T f(t)\sin(k\omega t)dt$$

Lastly, ω is calculated as

$$\omega = \frac{2\pi}{T}$$

where, in our case, $T = 2021 - 1998 = 23$ because we collected data from 1998 to 2021. Hence, for all teams, $\omega = 0.273$.

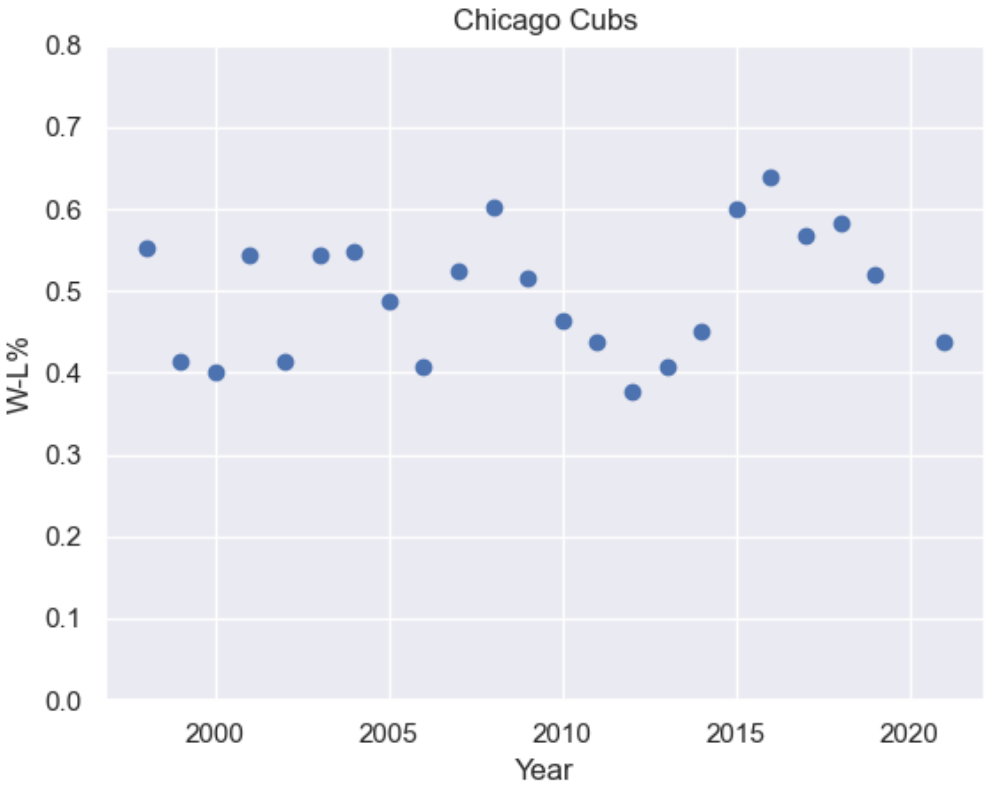


Figure 1: Scatterplot of the Chicago Cubs W-L%.

Because we only have data points instead of a function $f(t)$, it is more difficult to compute the Fourier series as we cannot utilize the equations above. Thus, we decided to use Python to aid with the computational work. Python has a curve-fitting function that allows us to define a function, and with the provided data, it estimates the best values for the Fourier coefficients via nonlinear least squares. We first defined the Fourier function with one or two sine and cosine terms, but it became clear that a Fourier series with several sine and cosine terms will most likely yield a more accurate model. Hence, we decided to define the Fourier function with nine terms for all 30 baseball teams. The following equation illustrates a Fourier series model with nine terms for the Chicago Cubs.

$$f(x) = \frac{0.988}{2} - 0.012\cos(1 * 0.273x) - 0.010\sin(1 * 0.273x) - 0.011\cos(2 * 0.273x)$$
$$+ 0.062\sin(2 * 0.273x) + 0.033\cos(3 * 0.273x) - 0.042\sin(3 * 0.273x)$$
$$+ 0.010\cos(4 * 0.273x) + 0.027\sin(4 * 0.273x) + 0.001\cos(5 * 0.273x)$$
$$+ 0.013\sin(5 * 0.273x) - 0.024\cos(6 * 0.273x) + 0.011\sin(6 * 0.273x)$$
$$- 0.013\cos(7 * 0.273x) - 0.022\sin(7 * 0.273x) - 0.024\cos(8 * 0.273x)$$
$$- 0.014\sin(8 * 0.273x) - 0.011\cos(9 * 0.273x) + 0.022\sin(9 * 0.273x)$$

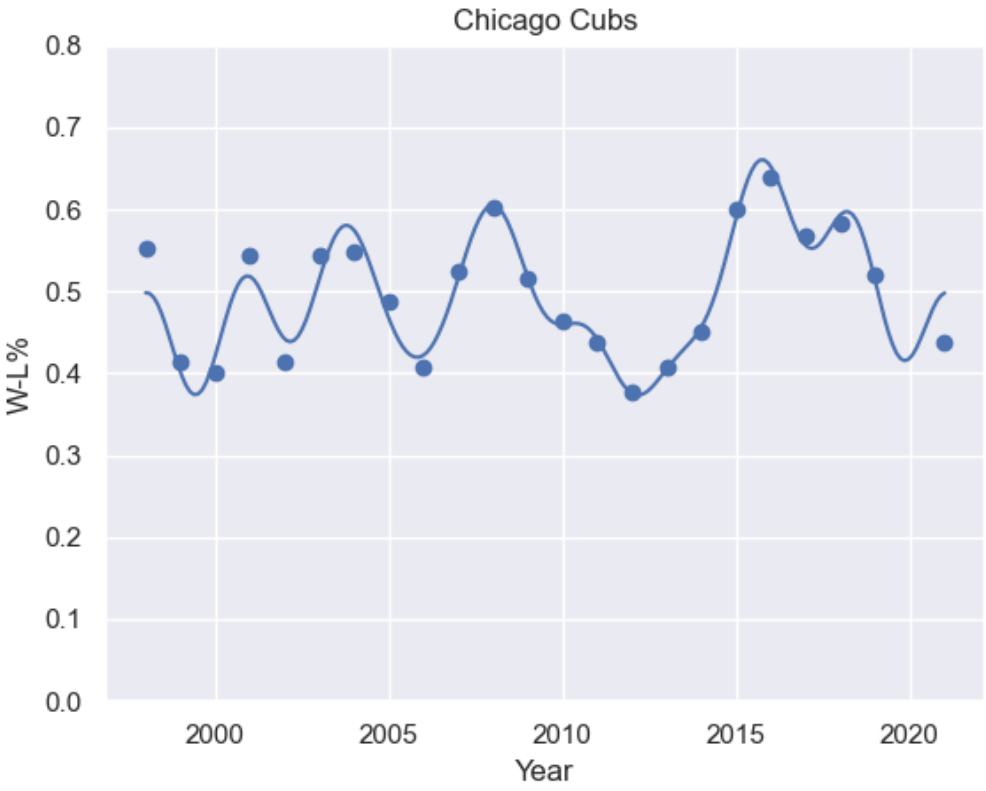


Figure 2: Fourier series model for the Chicago Cubs.

Plugging in 2020 for x, we get that the 2020 W-L% prediction for the Cubs is 0.419. We also calculated the R^2 value to be 0.914. The R^2 value is the square of the correlation between the response values and the predicted response values (Jena et al., 2011). It takes on any value between 0 and 1. When the R^2 value is close to 1, it indicates that a greater proportion

of variance is accounted for by the model (Jena et al., 2011). Therefore, an R^2 value of 0.914 suggests that the fit explains 91.4% of the variance in the data about the average. For the other teams, the R^2 value ranged from 0.733 to 0.969. While the graph of the Fourier series (Figure 2) and the R^2 value for the Cubs seem convincing, we cannot conclude that the Fourier series model is a good fit for the data just yet. We need to calculate other statistical measures to better analyze the goodness of fit, which will be done in section 2.4.

2.3 The Cubic Spline Model

For cubic spline interpolation, the data points are smoothly connected by a set of piecewise cubic functions. More specifically, two adjacent points (x_i, y_i) and (x_{i+1}, y_{i+1}) are joined by a cubic polynomial in the general form of

$$S_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i$$

for $x_i \leq x \leq x_{i+1}$ where $i = 1, ..., n-1$ (Kong et al., 2020). There are four unknown coefficients that need to be determined in the equation above. Thus, for n points, there are $n-1$ cubic functions and a total of $4(n-1)$ unknowns that require $4(n-1)$ independent equations to find all the coefficients (Kong et al., 2020). The required equations come from the following conditions:

1. The cubic functions must pass through the data points. Hence,

$$S_i(x_i) = y_i, \qquad i = 1, .., n-1,$$
$$S_i(x_{i+1}) = y_{i+1}, \qquad i = 1, .., n-1.$$

   This gives $2(n-1)$ equations.

2. The splines need to have continuous first derivatives at the interior points. Thus,

$$S_i'(x_{i+1}) = S_{i+1}'(x_{i+1}), \qquad i = 1, .., n-2.$$

3. The splines also need to have continuous second derivatives at the interior points. Therefore,

$$S_i''(x_{i+1}) = S_{i+1}''(x_{i+1}), \qquad i = 1, .., n-2.$$

   This condition, along with the second condition, gives $2(n-2)$ equations.

4. The last two equations are arbitrary and determine the end behavior of the spline. We chose a natural spline, meaning that the curve is a straight line at the endpoints. The second derivatives are zero at the endpoints, so

$$S_1''(x_1) = 0,$$
$$S_{n-1}''(x_n) = 0.$$

To find the coefficients, a_i, b_i, c_i, and d_i, of each cubic function, we can write the constraints as a system of linear equations with $4(n-1)$ unknowns and utilize linear algebra techniques. For the constraints $S_i(x_i) = y_i$, we have

$$a_1x_1^3 + b_1x_1^2 + c_1x_1 + d_1 = y_1,$$
$$a_2x_2^3 + b_2x_2^2 + c_2x_2 + d_2 = y_2,$$
$$\vdots$$
$$a_{n-1}x_{n-1}^3 + b_{n-1}x_{n-1}^2 + c_{n-1}x_{n-1} + d_{n-1} = y_{n-1}.$$

For $S_i(x_{i+1}) = y_{i+1}$, we get

$$\begin{aligned} a_1x_2^3 + b_1x_2^2 + c_1x_2 + d_1 &= y_2, \\ a_2x_3^3 + b_2x_3^2 + c_2x_3 + d_2 &= y_3, \\ &\vdots \\ a_{n-1}x_n^3 + b_{n-1}x_n^2 + c_{n-1}x_n + d_{n-1} &= y_n. \end{aligned}$$

For $S_i'(x_{i+1}) = S_{i+1}'(x_{i+1})$, we obtain

$$\begin{aligned} 3a_1x_2^2 + 2b_1x_2 + c_1 - 3a_2x_2^2 - 2b_2x_2 - c_2 &= 0, \\ 3a_2x_3^2 + 2b_2x_3 + c_2 - 3a_3x_3^2 - 2b_3x_3 - c_3 &= 0, \\ &\vdots \\ 3a_{n-2}x_{n-1}^2 + 2b_{n-2}x_{n-1} + c_{n-2} - 3a_{n-1}x_{n-1}^2 - 2b_{n-1}x_{n-1} - c_{n-1} &= 0. \end{aligned}$$

For $S_i''(x_{i+1}) = S_{i+1}''(x_{i+1})$, we get

$$\begin{aligned} 6a_1x_2 + 2b_1 - 6a_2x_2 - 2b_2 &= 0, \\ 6a_2x_3 + 2b_2 - 6a_3x_3 - 2b_3 &= 0, \\ &\vdots \\ 6a_{n-2}x_{n-1} + 2b_{n-2} - 6a_{n-1}x_{n-1} - 2b_{n-1} &= 0. \end{aligned}$$

Lastly, for $S_1''(x_1) = 0$ and $S_{n-1}''(x_n) = 0$, we have

$$\begin{aligned} 6a_1x_1 + 2b_1 &= 0, \\ 6a_{n-1}x_n + 2b_{n-1} &= 0. \end{aligned}$$

We can put a_i, b_i, c_i, and d_i in a matrix and apply linear algebra methods, such as Gaussian elimination, to solve it. The matrix will always be square and invertible as long as the x_i values in the dataset are unique (Kong et al., 2020). Rather than writing all the equations by hand, we take advantage of Python's cubic spline function to find the cubic functions and calculate the 2020 W-L% predictions. For example, for the Cubs, Python gives the coefficients for the cubic functions in the form $S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$, so we have

$$S(x) = \begin{cases} 0.0157(x-1998)^3 + 0.0(x-1998)^2 - 0.1537(x-1998) + 0.552 & [1998, 1999] \\ 0.0467(x-1999)^3 + 0.0470(x-1999)^2 - 0.0771(x-1999) + 0.414 & (1999, 2000] \\ \vdots & \\ -0.0046(x-2019)^3 + 0.0274(x-2019)^2 - 0.0771(x-2019) + 0.519 & (2019, 2021] \end{cases}$$

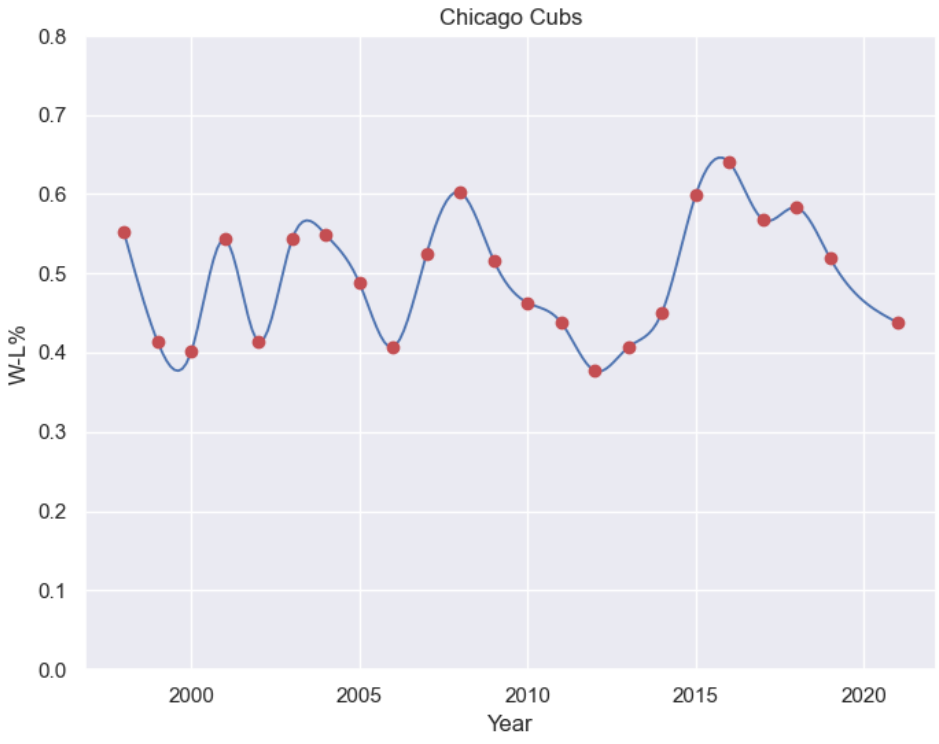


Figure 3: Cubic spline model for the Chicago Cubs.

There is a total of 22 equations, but the most important equation is S_{22} as 2020 falls within the interval $(2019, 2021]$. To obtain the predicted 2020 W-L% for the Cubs, we substitute 2020 for x in S_{22} and get 0.465. In the next section, we will compare and contrast the results of the Fourier series model, the cubic spline model, and Strat-O-Matic's Monte Carlo simulation.

2.4 Results on Interpolating the 2020 Season

In Table 1, we show the actual 2020 W-L% for each team and the predicted W-L% from the Strat-O-Matic, Fourier series, and cubic spline model. We start comparing the three models by determining which teams advance to the postseason based on their results. We then compute the residuals, the sum of the absolute residuals, the sum of the absolute residuals squared, and the maximum absolute residual for each of them to determine if one is evidently a better model.

Table 1: W-L% Predictions.

W-L%				
Team	2020 Results	Strat-O-Matic	Fourier series	Cubic Spline
Tampa Bay Rays	0.667	0.630	0.608	0.611
New York Yankees	0.550	0.586	0.610	0.613
Boston Red Sox	0.400	0.519	0.375	0.479
Toronto Blue Jays	0.533	0.432	0.480	0.456
Baltimore Orioles	0.417	0.358	0.421	0.356
Chicago White Sox	0.583	0.457	0.646	0.516
Cleveland Guardians	0.583	0.549	0.508	0.555
Detroit Tigers	0.397	0.414	0.262	0.321
Kansas City Royals	0.433	0.438	0.532	0.413
Minnesota Twins	0.600	0.562	0.685	0.615

Houston Astros	0.483	0.673	0.583	0.642
Seattle Mariners	0.450	0.383	0.374	0.420
Oakland Athletics	0.600	0.556	0.484	0.562
Los Angeles Angels	0.433	0.512	0.575	0.440
Texas Rangers	0.367	0.414	0.333	0.466
Atlanta Braves	0.583	0.549	0.583	0.585
Philadelphia Phillies	0.467	0.556	0.375	0.495
New York Mets	0.433	0.438	0.467	0.523
Miami Marlins	0.517	0.389	0.409	0.369
Washington Nationals	0.433	0.562	0.605	0.541
Milwaukee Brewers	0.483	0.506	0.537	0.546
St. Louis Cardinals	0.517	0.531	0.619	0.564
Cincinnati Reds	0.517	0.500	0.488	0.498
Chicago Cubs	0.567	0.562	0.419	0.465
Pittsburgh Pirates	0.317	0.377	0.307	0.377
San Francisco Giants	0.483	0.383	0.588	0.543
Los Angeles Dodgers	0.717	0.691	0.744	0.693
San Diego Padres	0.617	0.543	0.565	0.464
Colorado Rockies	0.433	0.451	0.448	0.406
Arizona Diamondbacks	0.417	0.481	0.370	0.466

In baseball, 10 teams get to advance to the postseason, but in 2020, the postseason was expanded to consist of 16 teams. Table 2 shows the 10 teams that each model predicted would make the postseason. The teams in italics did not actually qualify for the postseason in 2020. As one can see, eight of the 10 teams that Strat-O-Matic's model and Fourier series model predicted were actually in the 2020 postseason, and all of the teams that the cubic spline model predicted played in the postseason. Perhaps a different team would have been

crowned World Series Champion, but overall, it is likely that the 2020 postseason would not have looked that much different if they had played a full 162 games. Now, we continue to analyze the models in a more mathematical manner.

Table 2: 2020 postseason qualifiers based on models.

Strat-O-Matic	Fourier series	Cubic Spline
Tampa Bay Rays	Tampa Bay Rays	Tampa Bay Rays
New York Yankees	New York Yankees	New York Yankees
Minnesota Twins	Minnesota Twins	Minnesota Twins
Houston Astros	Houstons Astros	Houston Astros
Los Angeles Dodgers	Los Angeles Dodgers	Los Angeles Dodgers
Atlanta Braves	Atlanta Braves	Atlanta Braves
Oakland Athletics	Chicago White Sox	Oakland Athletics
Chicago Cubs	St. Louis Cardinals	St. Louis Cardinals
Washington Nationals	*Washington Nationals*	Cleveland Guardians
Philadelphia Phillies	*San Francisco Giants*	Milwaukee Brewers

When analyzing the goodness of fit of a model, the first important step is to compute the residuals. Given a model $y = f(x)$, the residuals, denoted as r_i, are computed by taking the difference between the observed values, y_i, and the predicted values, $f(x_i)$, where i is the team. For example, using Strat-O-Matic's model, the residual for the Chicago Cubs is as follows

$$\begin{aligned} r_{Cubs} &= y_{Cubs} - f(x_{Cubs}) \\ &= 0.567 - 0.562 \\ &= 0.005 \end{aligned}$$

Using the Fourier series model, we get

$$\begin{aligned} r_{Cubs} &= 0.567 - 0.419 \\ &= 0.148 \end{aligned}$$

Lastly, using the cubic spline model, we have

$$\begin{aligned} r_{Cubs} &= 0.567 - 0.465 \\ &= 0.102 \end{aligned}$$

Next, we plot the residuals of the Strat-O-Matic's model, the Fourier series model, and the cubic spline model to verify that the residuals are randomly scattered. Otherwise, if a pattern is detected in the residual plot, it is an indication that the model is a poor fit for the data as there might be another variable that is not being considered.

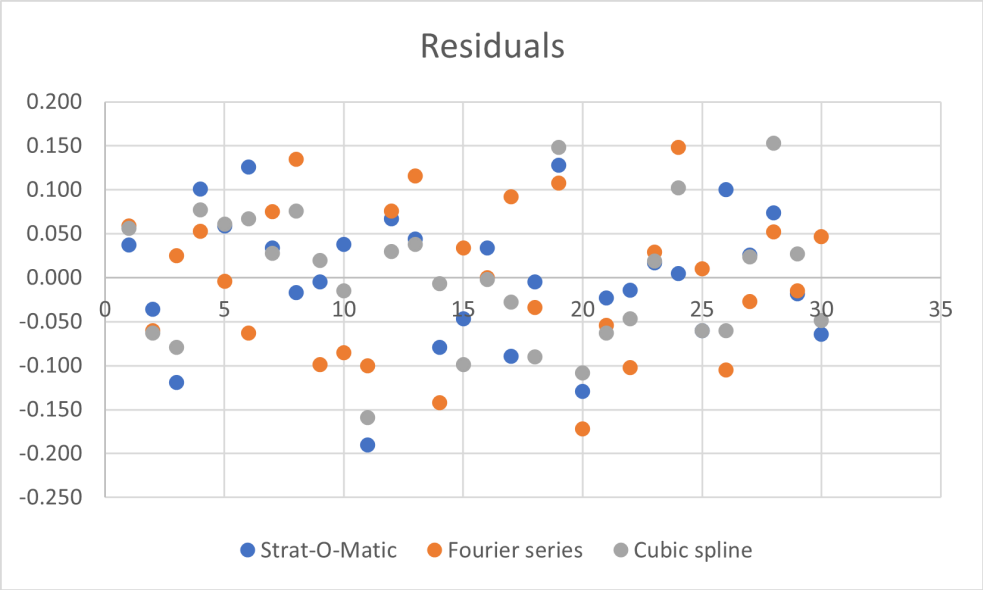


Figure 4: Residual plot.

The residual plot is randomly scattered, so now we move on to calculating the sum of the absolute residuals, the sum of the absolute residuals squared, and the maximum absolute

residual. The equations for these measures are as follows

$$\text{Sum of Absolute Residuals} = \sum_{i=1}^{N} |r_i|$$

$$\text{Sum of Absolute Residuals Squared} = \sum_{i=1}^{N} |r_i|^2$$

$$\text{Maximum Absolute Residual} = \max |r_i|$$

We summarized the values of these measures for Strat-O-Matic's model, the Fourier series model, and the cubic spline model in Table 3. It is important to take into account all three of these measures because looking at them in isolation can be very misleading (Giordano et al., 2013). For example, different models to have the same sum of squared absolute residuals, which would make one believe that the models fit the data about the same. However, the graphs will show a significant variation in each model's ability to capture the trend of the data (Giordano et al., 2013).

Model	$\sum_{i=1}^{N} \lvert r_i \rvert$	$\sum_{i=1}^{N} \lvert r_i \rvert^2$	$\max \lvert r_i \rvert$
Strat-O-Matic	1.785	0.168	0.19
Fourier series	2.121	0.210	0.172
Cubic spline	1.855	0.166	0.159

Table 3: Computed values of the three measures.

Looking at the sum of the absolute residuals, Strat-O-Matic's value of 1.785 is smaller than the Fourier series' value of 2.121 and the cubic spline's value of 1.855. In contrast, the cubic spline model has the smallest value, 0.166, for the sum of the squared absolute residuals compared to Strat-O-Matic's value of 0.168 and the Fourier series' value of 0.210. The same can be said of the maximum absolute residual as the cubic spline's value is 0.159, while 0.19 and 0.172 are the values for Strat-O-Matic's and Fourier series' model respectively. It seems that the cubic spline model is the better model, but notice that the differences between the three measures for the three models are not large. It is intriguing that the cubic spline model, which focuses only on the teams' W-L% data, and the Strat-O-Matic model, a Monte

Carlo simulation that includes players' statistics and randomness, both make good models despite them having their vast differences. The Fourier series model, also a model that only uses W-L% data, is almost just as good of a model. However, its biggest flaw is that it tends to create a large overshoot near the right endpoint, called Gibb's phenomenon. The graph of the Los Angeles Dodgers highlights this issue. This is concerning because 2020 is near the right endpoint, so the prediction might not be as accurate. For instance, the Fourier series model calculated that the 2020 W-L% for the Los Angeles Dodgers would have been 0.744. This is an extremely high W-L%, but it is not impossible. Nonetheless, we conclude that the Fourier series model is acceptable for fitting a team's W-L%, along with the cubic spline model, but when making predictions, they should only be used for interpolating and not extrapolating. For the remainder of the paper, we will discuss a time series model that can possibly be used for extrapolating baseball W-L%.

Figure 5: Fourier series model for the Los Angeles Dodgers.

3 Model for Extrapolating

3.1 Data Collection and Preparation

For the SARIMA model, we collected monthly W-L% for each of the 30 baseball teams for the years 1998 to 2022, excluding 2020. We decided on monthly data rather than yearly data because usually more than 50 data points are needed for time series analysis. Moreover, the data was modified so that if a season started in March, those games were incorporated into April's games, and if a season ended in October, those games were added to September's games. We made this adjustment because only a few games were played in March and October, so these months sometimes had extreme W-L%. For instance, if a team played only one game in March, their W-L% is either 0.000 or 1.000 depending if they won or lost. We were concerned that this might negatively affect the accuracy of the SARIMA model. Making this change to the data also made it easier to specify the period when accounting for seasonality in the Python code as each season is now exactly six months long. In addition, the data was split into a training set and a test set. The training set consists of the years 1998 to 2021, and the test set is the year 2022 as this is the year we want to forecast.

3.2 The Seasonal ARIMA Model

Auto-Regressive Integrated Moving Average, or ARIMA for short, is a modeling technique for time series forecasting. It essentially uses a linear combination of past values of the variable and/or past forecast errors to predict future values. To get a better understanding of the model, we will examine the three components of ARIMA separately.

The idea of an autoregressive (AR) model is that the current values of the series can be explained as a function of p past values (Shumway & Stoffer, 2011). An autoregressive model of order p, or $AR(p)$, can be expressed as

$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + ... + \phi_p Y_{t-p} + \epsilon_t,$$

where Y_t is a linear combination of the p most recent past values of itself, $\phi_1, ..., \phi_p$ are coefficients, and ϵ_t is white noise that is independent and identically distributed (Shumway & Stoffer, 2011). A moving average (MA) model uses past forecast errors instead of past observations of the variable and can be written as

$$Y_t = \epsilon_t + \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2} + ... + \theta_q\epsilon_{t-q},$$

where $\theta_1, ..., \theta_q$ are coefficients and ϵ_t is white noise (Shumway & Stoffer, 2011). This is referred to as a moving average model of order q, $MA(q)$. At this point, we can create a mixed Auto-Regressive Moving Average (ARMA) model as long as the time series is stationary, meaning that there is no systematic change in the mean, no systematic change in the variance, and no strictly periodic variations. If the time series is not stationary, then we can compute the differences between consecutive observations, which is called first differencing. Hence, the "integrated" part of ARIMA accounts for differencing and is denoted as

$$\nabla Y_t = Y_t - Y_{t-1}.$$

It might be necessary to difference more than once to obtain a series that is stationary, but it is uncommon to difference more than twice (Cryer & Chan, 2010). In the end, we should have that the dth difference $W_t = \nabla^d Y_t$ is stationary. With W_t denoting a time series that has been differenced d times and is stationary, an $ARIMA(p, d, q)$ model is defined as

$$W_t = \phi_1 W_{t-1} + ... + \phi_p W_{t-p} + \epsilon_t + \theta_1\epsilon_{t-1} + ... + \theta_q\epsilon_{t-q},$$

where p is the order of the autoregressive part, d is the number of times the time series was differenced, and q is the order of the moving average part (Cryer & Chan, 2010). Moreover, with our data, we want to consider seasonality; therefore, we need to add a seasonal component to the ARIMA model. A seasonal $AR(P)_m$ model with seasonal period m is written

similar to an $AR(p)$ model, but with a few minor changes:

$$Y_t = \Phi_1 Y_{t-m} + \Phi_2 Y_{t-2m} + ... + \Phi_P Y_{t-Pm} + \epsilon_t$$

(Cryer & Chan, 2010). Likewise, we can write a seasonal $MA(Q)_m$ model as

$$Y_t = \epsilon_t + \Theta_1 \epsilon_{t-m} + \Theta_2 \epsilon_{t-2m} + ... + \Theta_Q \epsilon_{t-Qm}$$

(Cryer & Chan, 2010). Finally, a seasonal difference is the difference between an observation and the corresponding observation from the previous season and is expressed as

$$\nabla_m Y_t = Y_t - Y_{t-m}.$$

Combining the ARIMA model discussed earlier and these seasonal components, we get an $ARIMA(p,d,q) \times (P,D,Q)_m$ model, sometimes referred to as SARIMA, where p, d, and q are defined as before, P and Q are the seasonal AR and MA terms respectively, D is the count of seasonal differencing, and m is the number of observations per year. In order to write out the general equation, we introduce the backward shift operator B as it helps simplify the notation:

$$BY_t = Y_{t-1}.$$

Essentially, the operator B is a shift of one time unit. Thus, a SARIMA model is given by

$$\Phi_P(B^m)\phi(B)\nabla_m^D \nabla^d Y_t = \Theta_Q(B^m)\theta(B)\epsilon_t,$$

where the operators are defined as

$$\begin{aligned} \text{AR:} \quad \phi(B) &= 1 - \phi_1 B - \phi_2 B^2 - ... - \phi_p B^p \\ \text{MA:} \quad \theta(B) &= 1 + \theta_1 B + \theta_2 B^2 + ... + \theta_q B^q \end{aligned}$$

$$\begin{aligned}
\text{seasonal AR:} \quad & \Phi_P(B^m) = 1 - \Phi_1 B^m - \Phi_2 B^{2m} - ... - \Phi_P B^{Pm} \\
\text{seasonal MA:} \quad & \Theta_Q(B^m) = 1 + \Theta_1 B^m + \Theta_2 B^{2m} + ... + \Theta_Q B^{Qm} \\
\text{first difference:} \quad & \nabla^d \qquad = (1 - B)^d \\
\text{seasonal difference:} \quad & \nabla_m^D \qquad = (1 - B^m)^D
\end{aligned}$$

(Shumway & Stoffer, 2011).

Our ultimate goal is to build a SARIMA model for each of the 30 MLB teams and forecast their 2022 monthly W-L%. Before doing so, we must discuss the steps for generating a SARIMA model. As we build the SARIMA models, we closely follow the Box-Jenkins Method that was introduced by George Box and Gwilym Jenkins in their 1970 seminal work, *Time Series Analysis: Forecasting and Control*. This is an iterative approach that comprises three steps:

1. Model Identification
2. Estimation
3. Model Diagnostics

For model identification, we have to assess whether the time series is stationary, and if it is not, we have to determine the number of times we need to difference. While there are a few ways to determine stationarity, we implement the Augmented Dickey-Fuller (ADF) test in Python. The null hypothesis of the ADF test is that a unit root is present in the time series; therefore, the data series is non-stationary. Thus, the alternative hypothesis is that the time series is stationary. If the p-value is less than 0.05, we will reject the null hypothesis and conclude that the series is stationary. Running the ADF test on the Chicago Cubs and White Sox, we get a p-value of 0.055 for the Cubs and a p-value of 0.001 for the White Sox. This tells us that the Cubs data is non-stationary and the White Sox data is stationary. After differencing the Cubs data once and applying the ADF test again, we obtain a p-value

of 0.000, so the differenced Cubs data is stationary. However, one will notice that none of the SARIMA models for the teams have $d = 1$, even though seven of the teams' data are non-stationary. This is because when seasonal differencing is done first, the resulting series will sometimes be stationary, so there is no need for a first difference (Hyndman & Athanasopoulos, 2018).

Next, we identify the p, q, P, and Q parameters for the SARIMA models. A common method is to use the Auto-Correlation function (ACF) plot to determine the order of a pure MA model and the Partial Auto-Correlation (PACF) plot to find the order of a pure AR model. If p and q are both positive, the plots will not be helpful in finding suitable values for p and q (Hyndman & Athanasopoulos, 2018). Likewise, if both P and Q are positive, the plots will not be informative. Table 4 summarizes how one could determine the parameters using the ACF and PACF plots.

	$AR(p)$	$MA(q)$	$ARMA(p,q)$	$AR(P)$	$MA(Q)$
ACF	tails off	cuts off after lag q	tails off	exponential decay at each m lag	significant at m lag
PACF	cuts off after lag p	tails off	tails off	significant at m lag	exponential decay at each m lag

Table 4: General behavior of ACF and PACF plots.

Figure 6 shows the ACF and PACF plots for the seasonally differenced Chicago White Sox data. It appears that the ACF and PACF plots do not provide information for the non-seasonal parameters. We observe a decaying behavior in the ACF plot and a significant spike in the PACF plot at lag 6, which may be suggestive of a seasonal AR term. To find the best value for all the parameters, we utilized Python's auto_arima, an automatic method that performs a grid search to find the optimal order for a SARIMA model. Python specifically uses the Hyndman-Khandakar algorithm, which involves unit root tests and the minimization of the information criterion, when choosing the best model. A more detailed outline of the Hyndman-Khandakar algorithm is given in Hyndman and Khandakar (2008).

In the code, we specified that the ADF test should be used to determine d, $D = 1$, and $m = 6$. Auto-ARIMA then returns the SARIMA model with the lowest AIC value by default. The AIC value stands for Akaike's Information Criterion and is defined as

$$AIC = -2 \log L_k + 2k,$$

where L_k is the maximized log-likelihood and k is the number of parameters in the model (Shumway & Stoffer, 2011). For the White Sox, auto-ARIMA chose $ARIMA(3, 0, 1) \times (2, 1, 0)_6$ to be the best model with an AIC score of -185.466. This SARIMA model can be written as

$$(1 - \Phi_1 B^6 - \Phi_2 B^{12})(1 - \phi_1 B - \phi_2 B^2 - \phi_3 B^3)\nabla_6^1 Y_t = (1 + \theta_1 B)\epsilon_t. \tag{1}$$

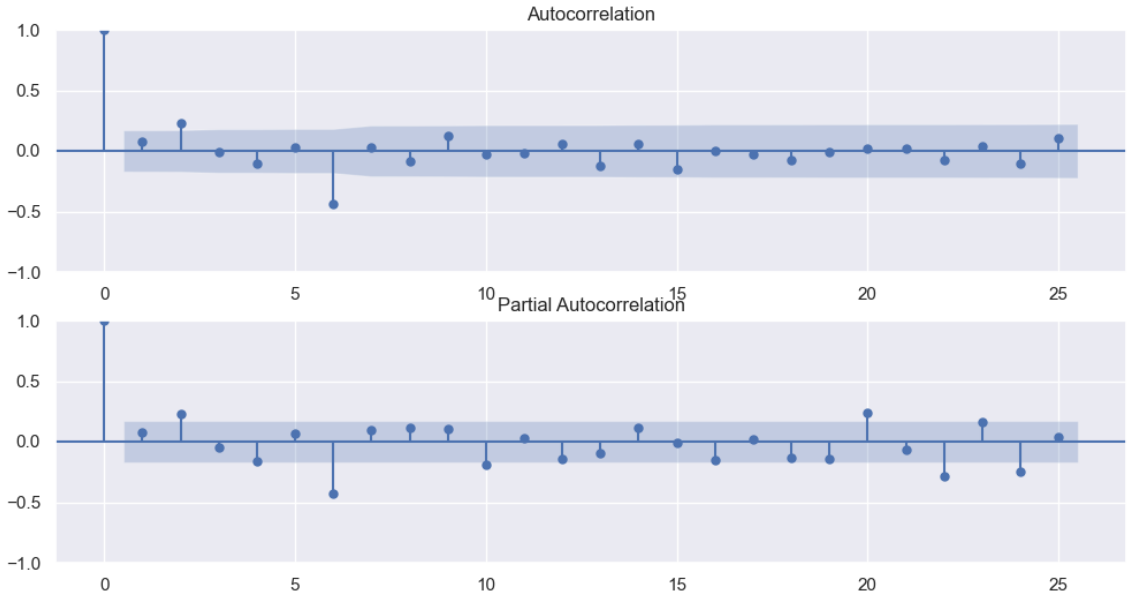


Figure 6: ACF and PACF plots of seasonally differenced White Sox data.

Once the best SARIMA model is identified, we move on to the second step of the Box-Jenkins approach: estimation. We need to estimate the coefficients, $\phi_1, ..., \phi_p$, $\theta_1, ..., \theta_q$,

$\Phi_1, ..., \Phi_P$, and $\Theta_1, ..., \Theta_Q$, but fortunately, auto-ARIMA does this for us. Below is a table of the coefficients that auto-ARIMA calculated for the White Sox. We interpret $ar.L1$ to be the AR term with lag 1, and similarly, $ar.L2$ and $ar.L3$ are the AR terms with lags 2 and 3 respectively. The coefficient $ma.L1$ refers to the MA term with lag 1. The seasonal AR terms are represented by $ar.S.L6$ and $ar.S.L12$. Lastly, $sigma2$ is the error term. The given p-values tell us whether the coefficient is statistically significant or not. All but one are statistically significant for the White Sox as their p-values are under 0.05; only $ar.S.L12$ is not statistically significant. We can now substitute the values of these coefficients into Equation 1 to obtain

$$(1 + 0.5292B^6 + 0.1861B^{12})(1 + 0.8062B - 0.2983B^2 - 0.2562B^3)\nabla_6^1 Y_t = (1 + 0.9857B)\epsilon_t$$

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.8062	0.098	-8.199	0.000	-0.999	-0.613
ar.L2	0.2983	0.120	2.495	0.013	0.064	0.533
ar.L3	0.2562	0.098	2.620	0.009	0.065	0.448
ma.L1	0.9857	0.044	22.523	0.000	0.900	1.071
ar.S.L6	-0.5292	0.111	-4.757	0.000	-0.747	-0.311
ar.S.L12	-0.1861	0.100	-1.856	0.063	-0.383	0.010
sigma2	0.0126	0.002	6.481	0.000	0.009	0.016

Ljung-Box (L1) (Q):	0.00	Jarque-Bera (JB):	3.06
Prob(Q):	0.98	Prob(JB):	0.22
Heteroskedasticity (H):	1.03	Skew:	-0.26
Prob(H) (two-sided):	0.91	Kurtosis:	2.46

Figure 7: Coefficients for the White Sox SARIMA model computed by auto-ARIMA.

Now that we have identified the parameters and estimated the coefficients for the SARIMA model, we have to check the model diagnostics to ensure that the model is valid. We need to examine the time series plot of the standardized residuals for any pattern and look at the ACF plot of the residuals to verify that there is no autocorrelation in the errors. In addition to the ACF plot, we can use the p-value of the Ljung-Box test that auto-ARIMA provides to determine if the residuals are autocorrelated and violate the assumption that the residuals

are independent. If the residuals are autocorrelated, the p-value will be less than 0.05. We also study the QQ plot and histogram of the residuals to assess normality. We expect the residuals to be normally distributed as ϵ_t is white noise, so we expect the points to lie mostly on the line in the QQ plot and for the histogram to be roughly bell-shaped. Auto-ARIMA also gives us the p-value for the Jarque-Bera test, which tests for the normality of the errors. A p-value greater than 0.05 implies that the residuals are normally distributed. The last test that auto-ARIMA conducts is to check for heteroscedasticity. The residuals should be homoscedastic, meaning that they have a constant variance. If the p-value is greater than 0.05, the residuals are homoscedastic; otherwise, they are heteroscedastic, which is an issue that would have to be addressed. Figure 8 displays the model diagnostics for the White Sox. There is no obvious pattern in the standardized residuals plot. The histogram and QQ plot show that the residuals are normal, and the Jarque-Bera p-value confirms this. The correlogram and the Ljung-Box p-value convey that there is no autocorrelation. Lastly, the heteroscedasticity p-value suggests that the residuals are homoscedastic. Hence, we can continue to forecast with our $ARIMA(3,0,1) \times (2,1,0)_6$ model for the White Sox.

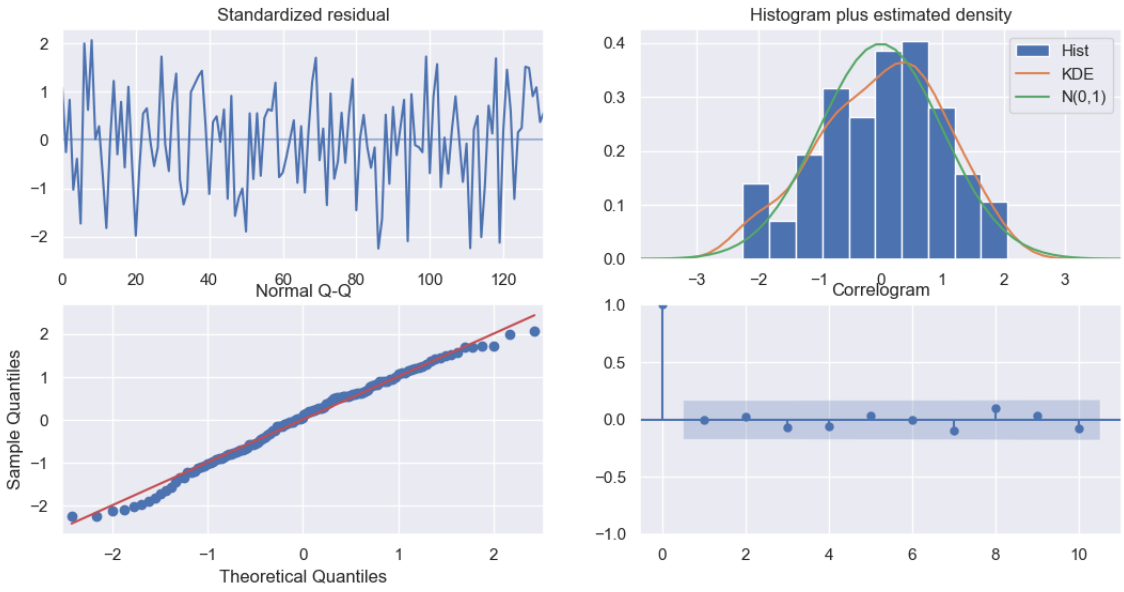


Figure 8: Model diagnostics for the White Sox SARIMA model.

Thus far, we have trained the SARIMA model with W-L% data from 1998 to 2021. We

now apply the model to the test set. SARIMA models do not usually do well forecasting far into the future; therefore, we only forecast the 2022 season, which is the next six months. Figure 9 is the visualization of the $ARIMA(3,0,1) \times (2,1,0)_6$ model and the 2022 forecasts with a 95% confidence interval for the White Sox that was generated by Python's predict function.

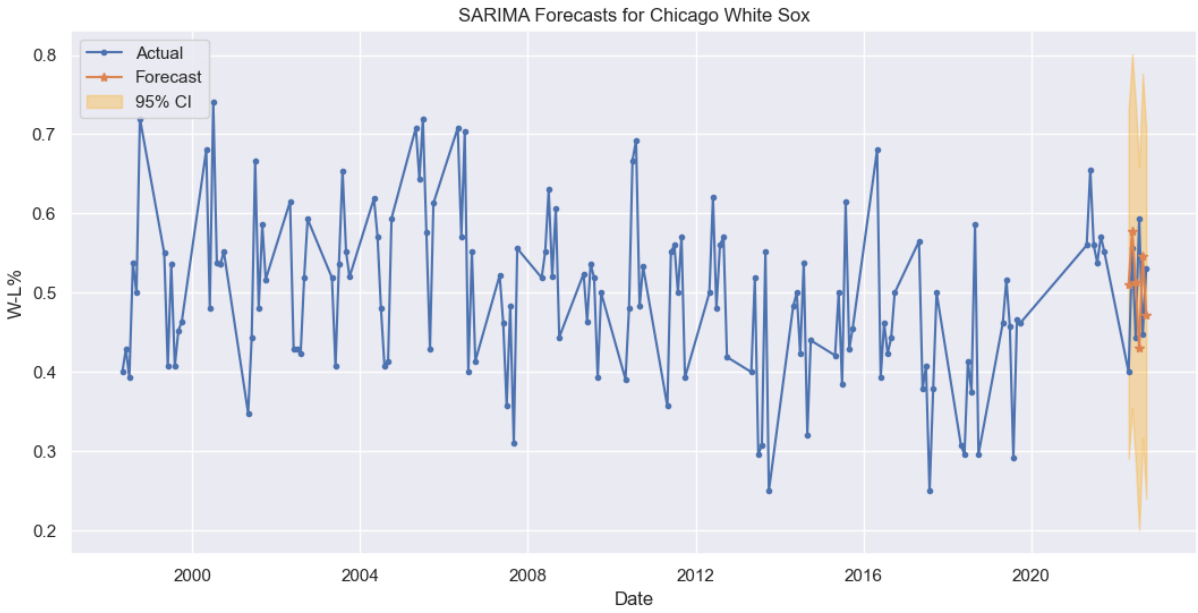


Figure 9: SARIMA model forecasts for 2022 White Sox.

The 2022 monthly W-L% forecasts for the White Sox do not appear to be far from the actual monthly W-L%, but we will mathematically analyze the forecasts with three widely used accuracy metrics: mean absolute error (MAE), root mean square error (RMSE), and mean absolute percentage error (MAPE). The formulas given by Python's sklearn.metrics module are as follows

$$MAE = \frac{1}{n}\sum_{i=0}^{n-1}|y_i - \hat{y}_i|$$

$$RMSE = \left(\frac{1}{n}\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2\right)^{\frac{1}{2}}$$

$$MAPE = \frac{1}{n}\sum_{i=0}^{n-1}\frac{|y_i - \hat{y}_i|}{\max(\epsilon, |y_i|)}$$

In these equations, $\hat{y}_i$ is the predicted value of the i-th sample, y_i is the corresponding actual value, n is the sample size, and ϵ is an arbitrary small positive number to avoid undefined results when y is zero ("3.3 Metrics and Scoring," n.d.). MAE and RMSE are scale-dependent measures, meaning that they are expressed on the same scale as the data. MAE measures the average magnitude of the absolute errors between the predicted value and actual value, while RMSE is the average vertical distance between the predicted value and actual value on the fit line (Jierula et al., 2021). RMSE punishes large errors more than MAE. MAPE is a scale-independent measure, so it is unit-free. It measures the size of errors in percentage terms, making it easy to interpret (Jierula et al., 2021). However, using MAPE has a couple of disadvantages. First, the MAPE value can be extreme if the actual value is small or it can be undefined if the actual value is zero. Another flaw of MAPE is that it penalizes positive errors more than negative errors (Jierula et al., 2021). Taking all of this into consideration, we will use MAE and RMSE as primary accuracy measures and MAPE as a supplementary measure. For the White Sox, we have

$$MAE = 0.0870$$

$$RMSE = 0.0977$$

$$MAPE = 0.1795$$

The MAE value implies that the average error between the forecasted W-L% and actual W-L% is 0.0870. The RMSE value is interpreted in a similar manner; the weighted average error between the forecasted and actual W-L% is 0.0977. Finally, the MAPE value tells us that the average difference between the forecasted W-L% and actual W-L% is 17.95%. The closer all three of these values are to zero, the better the accuracy of the model. Therefore, based on these numbers for the White Sox, the $ARIMA(3,0,1) \times (2,1,0)_6$ model is fairly accurate in forecasting the 2022 monthly W-L%. In section 3.3, we will evaluate the overall results of the SARIMA models of all 30 baseball teams.

3.3 Results on Extrapolating the 2022 Season

The specific SARIMA model and the MAE, RMSE, and MAPE values of each baseball team are shown in Table 5. An interesting observation is that the SARIMA models are not vastly different like one might think they would be. Looking at the seasonal component, 22 teams have two AR terms and either zero or one MA term, 7 teams have either zero or one AR term and one MA term, and only one team has one AR term and two MA terms. This tells us that for most of the teams, the model is using the last two corresponding months' W-L% to predict the 2022 month W-L%. To clarify, the model predicts the July 2022 W-L% from July 2019 W-L% and July 2021 W-L%. If a MA term is present, the model is using the corresponding month's error from the previous year to predict the 2022 month W-L%. As for the non-seasonal part of the ARIMA models, there is a little more variation as eight teams have no AR and MA terms and the rest of the teams have no more than three AR or MA terms. Thus, the models are using the values and errors of the last three months, at most, to predict the 2022 monthly W-L%. More specifically, the models are looking no further than the July 2021 W-L%. The similarities in the 30 SARIMA models may be surprising at first, but it is reasonable that using the most recent W-L% results in a better model because one would not expect a drastic change in W-L% from month to month. To get an idea of how the models performed, we now analyze the MAE, RMSE, and MAPE.

Table 5: SARIMA model and accuracy metrics for each team. The teams in italics have an acceptable SARIMA model.

Team	$ARIMA(p,d,q) \times (P,D,Q)_6$	MAE	RMSE	MAPE
Tampa Bay Rays	$(0,0,0) \times (2,1,0)_6$	0.0775	0.1046	0.1589
New York Yankees	$(0,0,0) \times (2,1,1)_6$	0.1574	0.1703	0.2867
Boston Red Sox	$(0,0,0) \times (2,1,0)_6$	0.1508	0.1735	0.3754
Toronto Blue Jays	$(0,0,0) \times (2,1,0)_6$	0.0935	0.1086	0.1604

Baltimore Orioles	$(1,0,0) \times (2,1,0)_6$	0.2139	0.2398	0.4021
Chicago White Sox	$(3,0,1) \times (2,1,0)_6$	0.0870	0.0977	0.1795
Cleveland Guardians	$(1,0,0) \times (2,1,0)_6$	0.0790	0.1023	0.1369
Detroit Tigers	$(2,0,2) \times (1,1,1)_6$	0.1037	0.1117	0.2749
Kansas City Royals	$(0,0,0) \times (2,1,0)_6$	0.0738	0.0804	0.1898
Minnesota Twins	$(0,0,1) \times (2,1,0)_6$	0.0855	0.1150	0.2089
Houston Astros	$(1,0,3) \times (0,1,1)_6$	0.0691	0.0757	0.1049
Seattle Mariners	$(3,0,0) \times (1,1,1)_6$	0.1097	0.1306	0.2074
Oakland Athletics	$(2,0,0) \times (2,1,0)_6$	0.1654	0.2173	0.6334
Los Angeles Angels	$(0,0,0) \times (2,1,0)_6$	0.1230	0.1455	0.3457
Texas Rangers	$(2,0,0) \times (0,1,1)_6$	0.0906	0.1034	0.2334
Atlanta Braves	$(0,0,2) \times (0,1,1)_6$	0.1599	0.1807	0.2417
Philadelphia Phillies	$(1,0,1) \times (1,1,1)_6$	0.1090	0.1374	0.2078
New York Mets	$(2,0,0) \times (2,1,1)_6$	0.1544	0.1604	0.2424
Miami Marlins	$(1,0,1) \times (2,1,0)_6$	0.1023	0.1294	0.2363
Washington Nationals	$(1,0,1) \times (2,1,1)_6$	0.1724	0.1834	0.5558
Milwaukee Brewers	$(2,0,0) \times (1,1,2)_6$	0.0811	0.0888	0.1513
St. Louis Cardinals	$(0,0,0) \times (2,1,0)_6$	0.0611	0.0661	0.1090
Cincinnati Reds	$(0,0,1) \times (2,1,0)_6$	0.1040	0.1350	0.4640
Chicago Cubs	$(1,0,0) \times (2,1,0)_6$	0.0971	0.1079	0.2247
Pittsburgh Pirates	$(1,0,1) \times (2,1,0)_6$	0.0405	0.0499	0.1025
San Francisco Giants	$(1,0,1) \times (2,1,0)_6$	0.1561	0.1875	0.3662
Los Angeles Dodgers	$(1,0,3) \times (1,1,1)_6$	0.0884	0.1119	0.1202
San Diego Padres	$(0,0,1) \times (2,1,0)_6$	0.1045	0.1255	0.1917
Colorado Rockies	$(0,0,0) \times (2,1,0)_6$	0.1099	0.1180	0.2641
Arizona Diamondbacks	$(2,0,0) \times (2,1,0)_6$	0.0762	0.1272	0.1490

At a glance, it is difficult to distinguish which SARIMA models performed well and which did not from the table. Thus, we created three histograms to depict the distribution of the MAE, RMSE, and MAPE. From the histograms, it is evident that most teams have a MAE value below 0.1285, a RMSE value below 0.1479, and a MAPE value below 0.2525. The

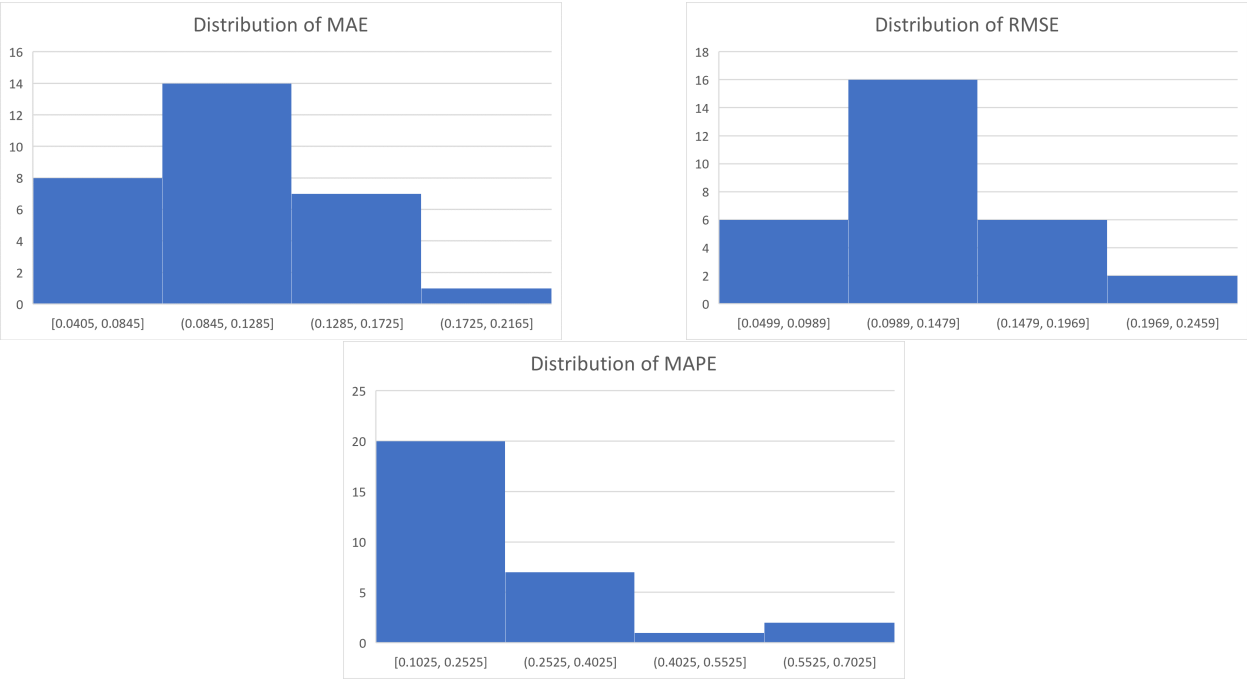
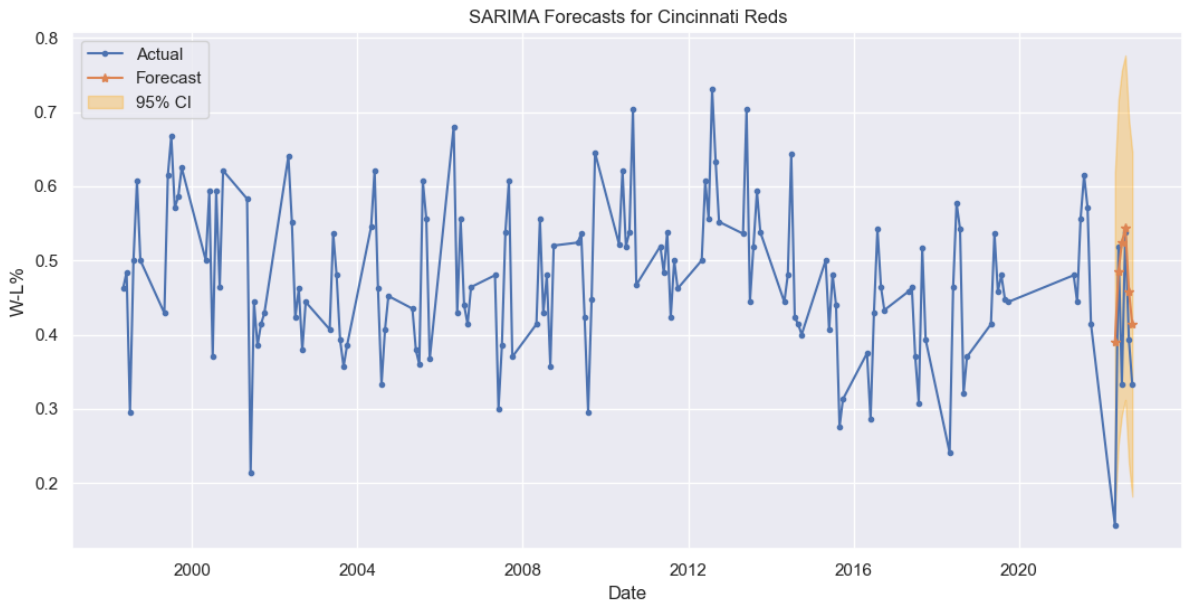


Figure 10: Histograms of accuracy metrics.

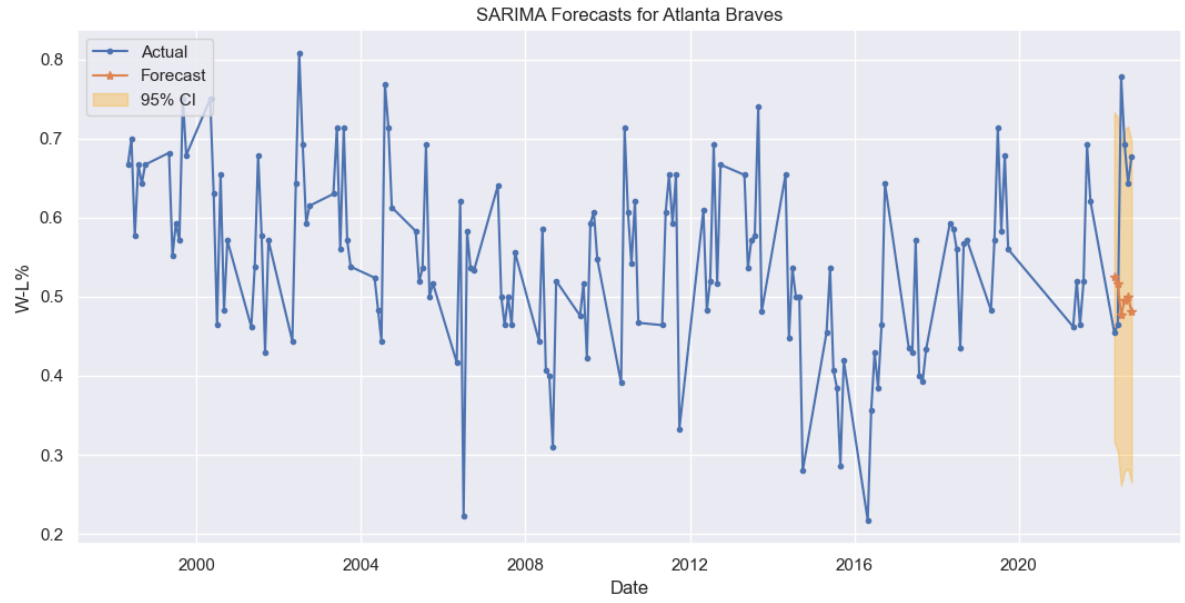
Pirates' SARIMA model has the lowest value for all three accuracy metrics. In contrast, the Orioles' model has the highest MAE and RMSE value, but Athletics' model has the highest MAPE value. We can easily determine that the Pirates' model is acceptable and seemingly accurate, whereas the Athletics' model, which also has a high MAE and RMSE value, is unsatisfactory. Although the Orioles' model has a MAPE value that falls into the second interval $(0.2525, 0.4025]$, their model is still unsatisfactory because of its high MAE and RMSE. Hence, the models that we deemed acceptable are those that have a MAE and RMSE value that fall into the first or second bin of the histograms. The 21 teams that have an acceptable SARIMA model are in italics in Table 5. Of these 21 models, four of them, the Tigers, Angels, Reds, and Rockies, have a MAPE value above 0.2525. We still consider these models to be appropriate not only because of their low MAE and RMSE values but

also because the plots show that the forecasts are reasonable. On the other hand, the Braves' and Mets' MAPE values suggest that their models are acceptable; however, their plots reveal that the forecast errors are indeed large as the MAE and RMSE values indicate. Figure 11 gives the plots of the Reds and Braves to show that even though the Reds have a high MAPE value, the forecasts seem reasonable, whereas it is evident that the forecast errors are large for the Braves despite having a low MAPE value.

There are many possible reasons why the other nine SARIMA models did not perform as well. Referring back to the model diagnostics step, the models for the Padres and Giants both failed the heteroscedasticity test, suggesting that the residuals have a non-constant variance. This heteroscedasticity could lead to forecasting problems. Sometimes taking the logarithm or square root transformation of the data can resolve this issue. If the residuals are still heteroscedastic after the transformation, then we might instead consider an ARCH or GARCH model, which are models specifically for capturing the changing variance of a time series. For now, we will ignore the heteroscedastic residuals as ARCH and GARCH models are beyond the scope of this paper, and we continue with the SARIMA models listed in Table 5 for the Padres and Giants. Even though the Padres' model seemed to have performed well, in the end, we deemed it unreliable because of the heteroscedasticity. As for the other teams, the model diagnostics were good, but it could be that the models severely underestimated or overestimated the teams' W-L%. From the Braves' and Mets' forecast plots, we can see that the model underestimated their W-L%, and this is the case with the Orioles as well. On the contrary, the SARIMA models for the Nationals and Athletics overestimated their W-L%. Lastly, the models were not able to capture the extreme fluctuation in W-L% for the Yankees and Red Sox. The Yankees had a W-L% above 0.649 for the first three months of the 2022 season, and in August, their W-L% was down to 0.357. The Red Sox had a W-L% of 0.769 in June 2022, which dramatically dropped to 0.296 by the end of July. Therefore, it is understandable why the SARIMA models failed to better forecast the Yankees' and Red Sox's 2022 W-L%. Overall, because we determined that 21



(a) Even though the Reds have a high MAPE value, the forecasts seem reasonable.



(b) The Braves' model has a low MAPE value, but the forecast errors appear to be large.

Figure 11: SARIMA model forecasts for 2022 Reds and Braves

of the 30 models are satisfactory, we conclude that a SARIMA model is appropriate for forecasting baseball W-L%, although it may not be as accurate as more advanced models, such as ZiPS or PECOTA.

4 Conclusion and Future Work

The purpose of this thesis is to explore different models, namely Fourier series, cubic spline, and SARIMA, for predicting past and future baseball W-L%. There already exist numerous elaborate models that can make such predictions, but we proposed three unsophisticated models as they only consider W-L% over time rather than a myriad of individual player statistics and randomness. We chose the Fourier series model because we hypothesized that yearly W-L% can be modeled by an infinite sum of cosine and sine functions. The cubic spline model was included in our study because cubic splines are a more traditional approach for interpolation. Both of these models were utilized to interpolate the 2020 W-L% for each team. On the other hand, a type of ARIMA model was chosen for extrapolating the monthly 2022 W-L% for each team because of its fairly accurate forecasting abilities in other areas, such as finance and epidemiology.

Our findings imply that past and future baseball W-L% can in fact be predicted by using only historical W-L% data. Both the Fourier series model and cubic spline model performed well as they gave adequate predictions for 2020 W-L%. Their sum of the absolute residuals, sum of the absolute residuals squared, and maximum absolute residual values are not far from those of Strat-O-Matic's Monte Carlo simulation. To answer the question posed in the introduction, it is likely that the outcome of the 2020 season was not affected much by the abbreviated schedule and lack of travel as all three models depicted a similar picture of the playoffs. Moreover, most of the SARIMA models provided reasonable monthly forecasts for 2022 as they have low MAE and RMSE values. However, the models can still be improved.

As highlighted throughout the paper, further work needs to be done on the Fourier series model and SARIMA model to enhance their accuracy. The flaw in the Fourier series model is that there is an overshoot between the last two data points, which is where 2020 falls. Therefore, in the future, we want to investigate how to remove or avoid Gibb's phenomenon if possible. We might try to reduce the number of terms included in the Fourier series model. In doing so, Gibb's phenomenon is not as apparent as seen in Figure 12. We altered

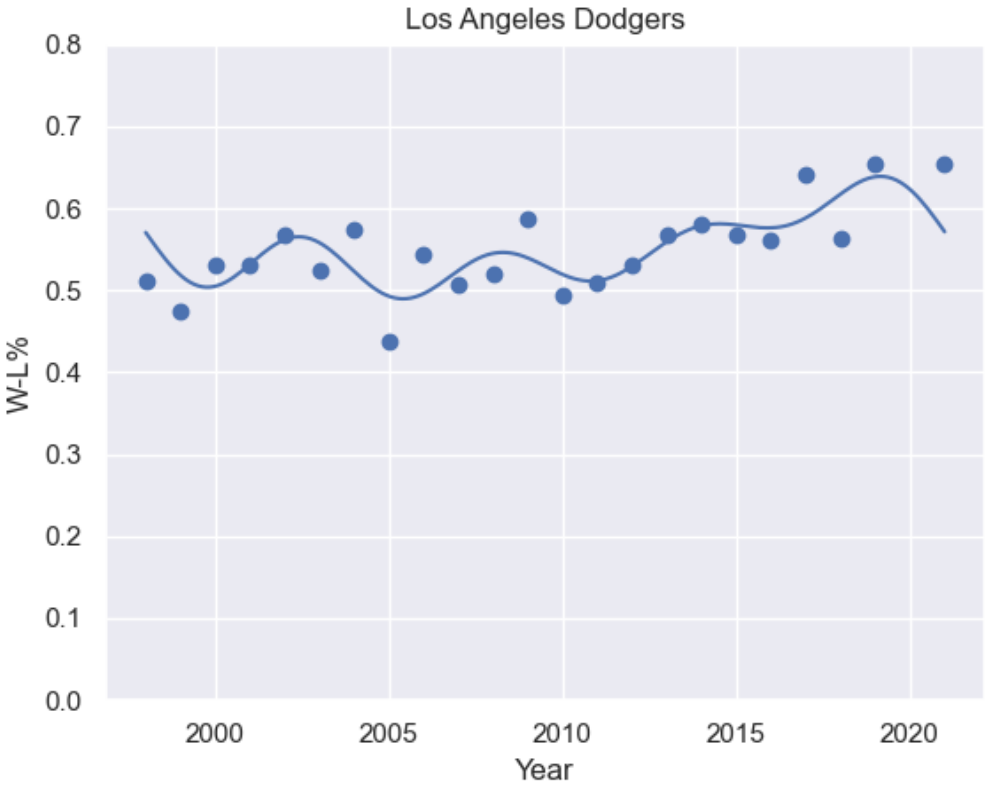


Figure 12: Fourier series model with five terms for the Dodgers.

the Fourier series model to only have five terms for all 30 teams. For the Dodgers, the model no longer produces such a large overshoot at the right endpoint, and it gives a more accurate prediction for their 2020 W-L%. However, the Fourier series model with five terms overall performs slightly worse than the model with nine terms as it has a larger sum of the absolute residuals, sum of the absolute residuals squared, and maximum absolute residual. This is due to the fact that for some teams, the end behavior flattened significantly. We can also consider other methods for determining the optimal number of terms and for finding the Fourier coefficients. Asmat et al. (2021) identified daily rainfall patterns in Sarawak, Malaysia using Fourier series. In their approach, they obtained the maximum likelihood estimates of the Fourier coefficients and calculated the deviance for each different number of terms that was fitted into the model. They then conducted an analysis of deviance to determine the number of terms that should be fitted; a term was included in the model if it

decreased the deviance significantly. Another method worth researching is the Fast Fourier Transform, which converts the data from the time domain to the frequency domain. Thus, it reveals which frequencies are significant, allowing us to determine the number of terms, and it provides the values of the Fourier coefficients. For the SARIMA models, we have to study how to best handle the heteroscedasticity, whether it be taking the log transformation or adding an ARCH or GARCH model. Despite needing some improvements, we conclude that the Fourier series and cubic splines are suitable models for interpolating baseball W-L%, while SARIMA is an appropriate and satisfactory model for extrapolating W-L%.

5 References

Asmat, A., Syed Wahid, S. N., & Deni, S. (2021). Identifying rainfall patterns using fourier series: A case of daily rainfall data in sarawak, malaysia. *Journal of Physics: Conference Series*, *1988*, 1–8.

Cryer, J. D., & Chan, K.-S. (2010). *Time series analysis: With applications in r* (2nd ed.). Springer.

Giordano, F. R., Fox, W. P., & Horton, S. B. (2013). *A first course in mathematical modeling* (5th ed.). Cengage Learning.

Hyndman, R. J., & Khandakar, Y. (2008). Automatic time series forecasting: The forecast package for r. *Journal of Statistical Software*, *27*(1), 1–22.

Hyndman, R., & Athanasopoulos, G. (2018). *Forecasting: Principles and practice* (2nd ed.). OTexts.

Jena, S. K., Ganesh, A., Balasubramanian, G., & Pradhan, N. (2011). Fourier approach to function approximation. *International Journal of Mathematical Archieve*, *2*, 617–624.

Jierula, A., Wang, S., OH, T.-M., & Wang, P. (2021). Study on accuracy metrics for evaluating the predictions of damage locations in deep piles using artificial neural networks with acoustic emission data. *Applied Sciences*, *11*(5).

Kadry, S. (2014). Calculus. In *Mathematical formulas for industrial and mechanical engineering* (pp. 65–111). Elsevier.

Kong, Q., Siauw, T., & Bayen, A. (2020). *Python programming and numerical methods: A guide for engineers and scientists* (1st ed.). Academic Press.

Shumway, R. H., & Stoffer, D. S. (2011). *Time series analysis and its applications: With r examples* (3rd ed.). Springer.

2020 Season Simulation. (2020). Strat-O-Matic. https://www.strat-o-matic.com/2020-season-simulation/

3.3. Metrics and Scoring: Quantifying the Quality of Predictions. (n.d.). Scikit-Learn Machine Learning in Python. https://scikit-learn.org/stable/modules/model_evaluation.html

Major League Teams and Baseball Encyclopedia. (n.d.). Baseball Reference. https://www.baseball-reference.com/teams/

6 Appendix: Python Code

The following code is specifically for the Chicago Cubs, but the other teams have the same code with different data. The SARIMA code was adapted from a post by David Ordorica on alldatascience.com.

```
from scipy.interpolate import CubicSpline
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
import pmdarima as pm
from sklearn.metrics import mean_absolute_error,
    mean_absolute_percentage_error, mean_squared_error
from statsmodels.tsa.seasonal import seasonal_decompose
from scipy.optimize import curve_fit
from numpy import sin, cos
import seaborn as sns
sns.set()


#Fourier series model
x_data = np.array([1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006,
    2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018,
        2019, 2021])
y_data = np.array([0.552, 0.414, 0.401, 0.543, 0.414, 0.543, 0.549, 0.488,
    0.407, 0.525, 0.602, 0.516, 0.463, 0.438, 0.377, 0.407, 0.451, 0.599,
        0.640, 0.568, 0.583, 0.519, 0.438])
```

```
w = 0.273
def fourier(x, a0, a1, b1, a2, b2, a3, b3, a4, b4, a5, b5, a6, b6,
    a7, b7, a8, b8, a9, b9):
return (a0/2) + a1 * cos ( 1*w*x) + b1 * sin( 1*w*x) +
       a2 * cos ( 2*w*x) +b2 * sin( 2*w*x) +
       a3 * cos ( 3*w*x) +b3 * sin( 3*w*x) +
       a4 * cos ( 4*w*x) +b4 * sin( 4*w*x) +
       a5 * cos ( 5*w*x) +b5 * sin( 5*w*x) +
       a6 * cos ( 6*w*x) +b6 * sin( 6*w*x) +
       a7 * cos ( 7*w*x) +b7 * sin( 7*w*x) +
       a8 * cos ( 8*w*x) +b8 * sin( 8*w*x) +
       a9 * cos ( 9*w*x) +b9 * sin( 9*w*x)

popt, pcov = curve_fit(fourier, x_data, y_data, p0=[0.993, -0.005,
    -0.018, -0.051, 0.033, 0.045, 0.009, -0.025, 0.014, -0.010, 0.001,
        0.006, -0.021, 0.016, 0.006, 0.016, 0.006, 0.008, -0.010])
res = y_data - fourier(x_data, *popt)
ss_res = np.sum(res**2)
ss_tot = np.sum((y_data - np.mean(y_data))**2)
r_squared = 1 - (ss_res/ss_tot)
print("r^2:" , r_squared)
print("2020 prediction: " , fourier(2020, *popt))

#create scatterplot of x vs y
plt.scatter(x_data, y_data)
plt.ylim(0.000,0.800)
plt.title("Chicago Cubs")
```

```
plt.xlabel("Year")
plt.ylabel("W-L\%")
x_axis=np.linspace(min(x_data), max(x_data), num=200)
plt.plot(x_axis, fourier(x_axis, *popt))
plt.show()

#Cubic Spline
f = CubicSpline(x_data, y_data, bc_type='natural')
x_new = np.linspace(1998, 2021, 200)
y_new = f(x_new)
print(f(2020))

plt.figure(figsize = (8,6))
plt.plot(x_new, y_new, 'b', linewidth=1.3)
plt.plot(x_data, y_data, 'ro',markersize=6)
plt.ylim(0.000,0.800)
plt.title('Chicago Cubs')
plt.xlabel('Year')
plt.ylabel('W-L%')
plt.show()

#show values of coefficients
print(f.c)

#Time Series Analysis
data=pd.read_csv(r'C:\ Users\ gisel\ OneDrive\ Desktop\ masters thesis\
    cubsdata.csv', parse_dates=['Date'], index_col='Date')
```

```
plt.plot(data.index, data['W-L%'], marker='.')
plt.xlabel('Date')
plt.ylabel('W-L%')
plt.title('Time Series of Chicago Cubs')
plt.show()

#Train Test Split
test_size=6
train, test = data.iloc[:-test_size], data.iloc[-test_size:]

plt.figure(figsize=(25,6))
plt.xlabel('Date')
plt.ylabel('W-L%')
plt.plot(train, lw=1.5, marker='.',label='Train set')
plt.plot(test, lw=1.5,marker='*', label='Test set')
plt.ylim([0.2, 0.8])
plt.legend()
plt.show()

#ACF and PACF Plots
plot_acf(train['W-L%'])
plot_pacf(train['W-L%'])
plt.show()

#ACF and PACF for seasonally differenced
fig, ax = plt.subplots(2, figsize=(12,6))
```

```
x = (train['W-L%'].dropna() - train['W-L%'].dropna().shift(6)).dropna()
ax[0] = plot_acf(x, ax=ax[0], lags=25)
ax[1] = plot_pacf(x, ax=ax[1], lags=25)
plt.show()

#Seasonal Decompose
d = seasonal\_decompose(data, period=6)
d.plot()
plt.show()

#ADFuller Test
result = adfuller(train['W-L%'].dropna())
print('ADF p-value: ', result[1])

#ARIMA model
model = pm.auto_arima(train['W-L%'], stepwise=True, D=1, m=6)
print(model.summary())

#Check Residuals
model.plot_diagnostics(figsize=(15,10))
plt.show()

#Forecast
prediction, confint = model.predict(n_periods=test_size,
    return_conf_int=True)
prediction.index=test.index
print(prediction)
```

```
cf= pd.DataFrame(confint)
prediction_series = pd.Series(prediction,index=test.index)
fig, ax = plt.subplots(1, 1, figsize=(18, 6))
ax.plot(data['W-L%'],lw=1.5, marker='.', label='Actual')
ax.plot(prediction_series,lw=1.5, marker='*', label='Forecast')
ax.fill_between(prediction_series.index,
        cf[0],cf[1],color='orange',alpha=.3,label='95% CI' )
plt.title('SARIMA Forecasts for Chicago Cubs')
ax.set_xlabel('Date')
ax.set_ylabel('W-L%')
plt.legend(loc='upper left')
plt.show()

#Accuracy Metrics
mae = mean_absolute_error(test['W-L%'], prediction)
print('mae: ', mae)
rmse = np.sqrt(mean_squared_error(test['W-L%'], prediction))
print('rmse: ', rmse)
mape = mean_absolute_percentage_error(test['W-L%'], prediction)
print('mape: ', mape)
```