MSE

Mentoring Excellence

FC Barcelona La Liga Performance: A Markov Model

Ayushi Gupta, Krantik Das

September 2023

0.1 Problem Statement

The data of the match result of a particular football club (Barcelona FC), for the season 2020-2021 of La Liga has been collected. Barcelona played a total of 76 matches in the entire season. The win loss and draw has been annotated as W L D. We find it trivial to assume that the result of the ith match only depends on the result of the $(i-1)$th match. To avoid biasness the result of the past 3 matches have been clubbed together, e.g, if they have a result sequence like Win, Loss, Loss, we have annotated it as WLL. Every possible such combination has been taken. Then, it has been attempted to predict the probabilities of transition from one particular state space to another in the sense that if Barcelona FC has won the past 2 matches and lost the third one (WWD), what is the probability that it will win the fourth match i.e. transition to state space WDW. In order to model it as a markov chain, we have made certain assumptions. Firstly, the match result in the past three matches gives sufficient information to predict the match result of the next match. Secondly, all other variables, both quantifiable and non quantifiable, such as psychological factors, weather conditions, injuries to the players etc. are kept as constant. Thus, according to our assumptions, we are of the opinion that the current problem statement can be modeled as a Markov Chain.

0.2 Data

The table below shows the first ten data points. Here, the dates have been indexed as time spaces with integer values. Each match result has been assigned a match point. Win, lose and draw are assigned values 1,-1, and 0 respectively. Total score at each time period is the cumulative sum of the match points till that time period. State is the main variable of concern which annotates the match result of the past three matches for each time period. Thus considering every possible state space, there can exist a total of 27 state spaces. For 76 matches, because the first two time periods will not have a state space, a total of 74 data points remain for state spaces. Specific to our data, only 22 state spaces were observed for Barcelona in the 2020-21 season and the same 22 states have been considered.

0.3 Methodology

0.3.1 State Spaces

According to the match result of the team, the new variable state was created as shown in the before mentioned table. Every ith data point of this new variable codes the information of the match results of the ith, $(i-1)$th, and $(i-2)$nd match. In total, we have 74 such data points because of the presence of 76 matches played by the team. There can be three values W, L, D that each match result can have, and as a result, there are 27 unique combinations for the new variable.

As per the data of Barcelona, all the values of the new variable were found out. There were 22 observed unique values which have been taken as the state spaces. While coding in R, the sequence of the new variable has been stored as a vector called `match_results`. All the unique values of this vector have been taken as the state spaces and coded as a new vector named `state_spaces`.

```r

# Create a sequence of states (replace with your own data)
# Define the possible states
match_results <- c(
  "WDD", "DDD", "DDW", "DWD", "WDW", "DWD", "WDD", "DDW",
      "DWW", "WWW",
  "WWD", "WDW", "DWD", "WDW", "DWW", "WWD", "WDD", "DDW",
      "DWD", "WDL",
  "DLL", "LLL", "LLW", "LWW", "WWW", "WWW", "WWW", "WWL",
      "WLW", "LWW",
  "WWW", "WWL", "WLW", "LWD", "WDW", "DWD", "WDD", "DDD",
      "DDL", "DLW",
  "LWW", "WWW", "WWD", "WDD", "DDD", "DDD", "DDW", "DWL",
      "WLW", "LWW",
  "WWW", "WWW", "WWL", "WLD", "LDL", "DLW", "LWL", "WLW",
      "LWD", "WDD",
  "DDW", "DWD", "WDD", "DDW", "DWW", "WWW", "WWW", "WWW",
      "WWW", "WWW",
  "WWD", "WDW", "DWL", "WLW"
)

# Define the possible state spaces
state_spaces <- unique(match_results)
```

Listing 1: State Space Sequence

Date	TimeState	Match Result	Match Point	Total Score	STATE	
27-09-2020	1	W	1	1		
04-10-2020	2	D	0	1		
24-10-2020	3	D	0	1	WDD	
07-11-2020	4	D	0	1	DDD	
29-11-2020	5	W	1	2	DDW	
13-12-2020	6	D	0	2	DWD	
16-12-2020	7	W	1	3	WDW	
19-12-2020	8	D	0	3	DWD	
29-12-2020	9	D	0	3	WDD	
31-01-2021	10	W	1	4	DDW	
13-02-2021	11	W	1	5	DWW	

Table 1: Match Results for Barcelona FC

0.3.2 Transition Probability Matrix

Using the "*match_results*" vector which displays the sequence of the state spaces chronologically, a transition probability matrix was created. The dimension of the matrix has a dimension of 22 rows and 22 columns. Each element of the matrix represents the probability of transitioning from one state space to another. The probability of transition from state A to B is calculated by dividing the frequency of state A going to state B by the total frequency of state A going to all the 22 state spaces.

The same has been coded in R. The transition probability matrix has been calculated and named "transition_matrix_prob". The printed "transition_matrix_prob" is shown as output below. For visual simplicity, only a 4*4 matrix, that is first 16 entries, have been printed.

```r

# Create an empty transition probability matrix
n_states <- length(state_spaces)
transition_matrix <- matrix(0, nrow = n_states, ncol =
    n_states, dimnames = list(state_spaces, state_spaces))

# Function to update the transition matrix
update_transition_matrix <- function(matrix, sequence) {
  from_state <- sequence[1]
  to_state <- sequence[2]
  matrix[from_state, to_state] <- matrix[from_state,
      to_state] + 1
  return(matrix)
}

# Loop through the match results and update the transition
    matrix
for (i in 1:(length(match_results) - 1)) {
  transition_matrix <-
      update_transition_matrix(transition_matrix,
      c(match_results[i], match_results[i + 1]))
}

# Normalize the transition matrix to obtain probabilities
row_sums <- rowSums(transition_matrix)
transition_matrix_prob <- transition_matrix / row_sums

# Print the transition probability matrix
print(transition_matrix_prob)

View(transition_matrix_prob)
```

Listing 2: Transition Probability Matrix

The transition probability matrix is shown below:

	WDD	DDD	DDW	WDW
WDD	0.0	0.4285714	0.5714286	0.0
DDD	0.0	0.2500000	0.5000000	0.0
DDW	0.0	0.0000000	0.0000000	0.5
WDW	0.0	0.0000000	0.0000000	0.6

After calculating the transition probability matrix, it can be clearly observed that the sum of each row of the transition probability matrix is equal to 1. Thus, we can effectively conclude that the model represents a Markov Chain.

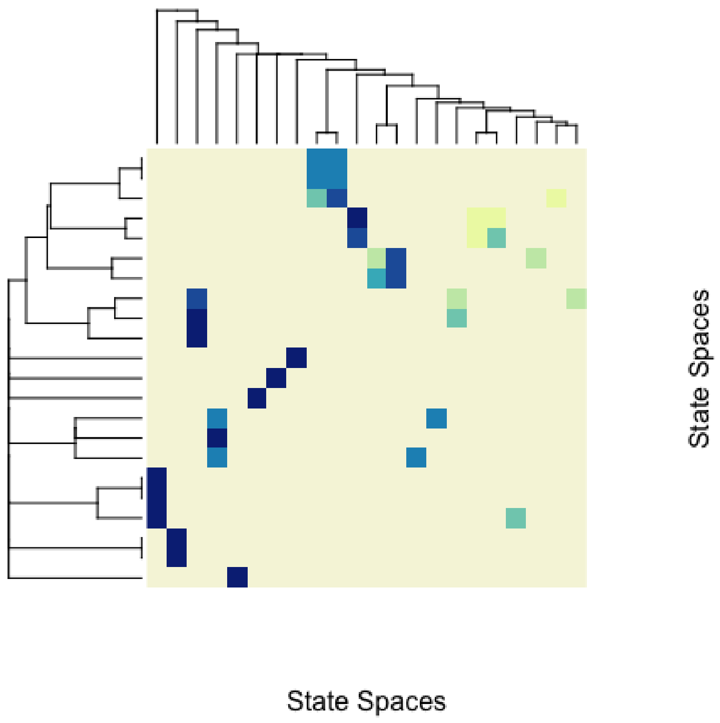
0.3.3 Visualization of Transition Probability Matrix

Since the number of state spaces is huge, it seemed imprudent to construct a transition probability diagram. Instead, the transition probability matrix has been visualized using a heatmap. A heatmap of a transition probability matrix visually represents the probabilities of transitioning from one state to another in a Markov chain or similar stochastic process. Each cell in the heatmap corresponds to the probability of transitioning from the row state to the column state. The color of each cell is typically used to encode the magnitude of the transition probability. The intensity of color in each cell indicates the probability value. Darker colors represent higher probabilities, while lighter colors represent lower probabilities. This allows us to quickly identify which transitions are more likely or less likely.

The same has been done in R with the following code. Below is the heatmap obtained:

```

# Load the required package for color palettes
library(RColorBrewer)


# Define the custom color palette
custom_palette <- c("beige", brewer.pal(8, "YlGnBu"))

# Create a heatmap with the custom color palette
heatmap(transition_matrix_prob,
        col = custom_palette,
        xlab = "State Spaces",
        ylab = "State Spaces",
        labRow = NA,  # Remove row labels
        labCol = NA)  # Remove column labels
```

Listing 3: Heatmap



0.3.4 MCMC Simulation using Gibbs Sampling

The goal of the Markov Chain Monte Carlo simulation is to sample prior and posterior distributions for our Markov chain. While there are many methods in MCMC, here the Gibbs sampling method has been used. The following steps are involved in the process of Gibbs sampling to get the prior and posterior distribution:

1. In order to initialize our variables, empty lists named "prior_samples" and "posterior_samples" have been created to store samples from the prior and posterior distributions.

2. The Gibbs sampling loop involves the generation of samples from the prior and posterior distributions. Gibbs sampling is a technique for generating samples from a complicated distribution by iteratively sampling from simpler conditional distributions. In each iteration of the loop, an initial "current_state" has been selected from the "match_results".

3. In the Gibbs sampling process, in order to get the prior distribution, a new state is chosen based on the transition probabilities from the current state to all state spaces. This new state is then included in the "prior_samples" list.

4. For the posterior distribution, a similar process is followed but based on the transition probabilities from the new state that was added in the "prior_samples" list.

5. We repeat this process for a specified number of iterations.

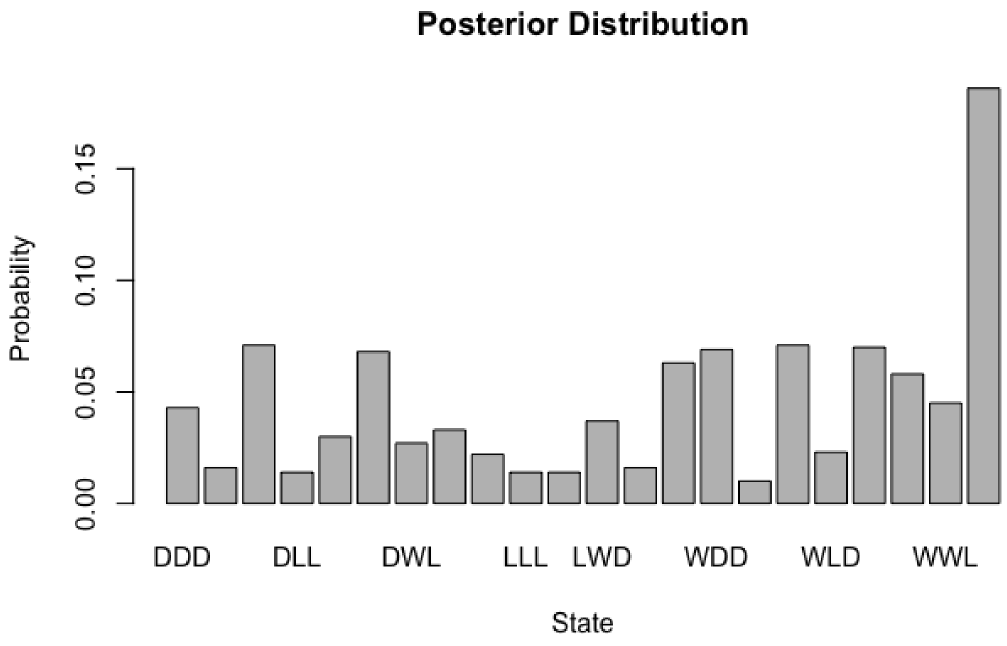
6. Then, the indices are converted back into the state spaces for easier interpretation.

7. To get the prior and posterior distributions, we normalize the prior and posterior samples by dividing by the total number of samples to get probabilities.

We have run these codes in R and have gotten the prior and posterior distributions. They are then printed, and for better visualization of the posterior distribution, a bar plot has been constructed for the same.

```r
# Define the number of iterations for Gibbs sampling
num_iterations <- 1000


# Number of burn-in iterations
burn_in_iterations <- 100

# Ensure state_sequence contains unique state sequences
match_results <- unique(match_results)

# Create an index map to match state sequences to matrix
    indices
state_indices <- match(match_results,
    rownames(transition_matrix_prob))

# Normalize the transition probability matrix
transition_matrix_prob_normalized <- transition_matrix_prob
    / rowSums(transition_matrix_prob)

# Initialize an initial state
current_state_index <- sample(1:length(match_results), 1)

# Initialize empty vectors to store prior and posterior
    samples
prior_samples <- character(0)
posterior_samples <- character(0)

# Gibbs sampling loop
for (iteration in 1:num_iterations) {
  # Sample the prior distribution
  prior_sample_index <- sample(1:length(match_results), 1,
      prob =
      transition_matrix_prob_normalized[current_state_index,
      ])
  prior_samples <- c(prior_samples,
      match_results[prior_sample_index])

  # Update the current state
  current_state_index <- prior_sample_index

  # Sample the posterior distribution
  posterior_sample_index <- sample(1:length(match_results),
      1, prob =
      transition_matrix_prob_normalized[current_state_index,
      ])
  posterior_samples <- c(posterior_samples,
      match_results[posterior_sample_index])
}


# Optionally, you can convert state indices back to state
    sequences for interpretation
posterior_samples_sequences <-
    match_results[match(posterior_samples, match_results)]

# Calculate the prior and posterior distributions
prior_distribution <- table(prior_samples) /
    length(prior_samples)
posterior_distribution <-
    table(posterior_samples_sequences) /
    length(posterior_samples_sequences)

# Print or visualize the prior and posterior distributions
print("Prior␣Distribution:")
## [1] "Prior Distribution:"
print(prior_distribution)
## prior_samples
##   DDD   DDL   DDW   DLL   DLW   DWD   DWL   DWW   LDL
    LLL   LLW   LWD   LWL
## 0.044 0.013 0.067 0.016 0.032 0.082 0.020 0.035 0.019
    0.016 0.016 0.030 0.016
##   LWW   WDD   WDL   WDW   WLD   WLW   WWD   WWL   WWW
## 0.062 0.081 0.016 0.069 0.019 0.060 0.054 0.043 0.190

print(posterior_distribution)
## posterior_samples_sequences
##   DDD   DDL   DDW   DLL   DLW   DWD   DWL   DWW   LDL
    LLL   LLW   LWD   LWL
## 0.050 0.010 0.065 0.016 0.032 0.077 0.027 0.032 0.019
    0.016 0.016 0.030 0.020
##   LWW   WDD   WDL   WDW   WLD   WLW   WWD   WWL   WWW
## 0.058 0.092 0.012 0.062 0.014 0.065 0.057 0.045 0.185
# Create a bar plot of the posterior distribution
barplot(posterior_distribution, names.arg =
    names(posterior_distribution),
         xlab = "State", ylab = "Probability", main =
            "Posterior␣Distribution")
```

Listing 4: Gibbs Sampling



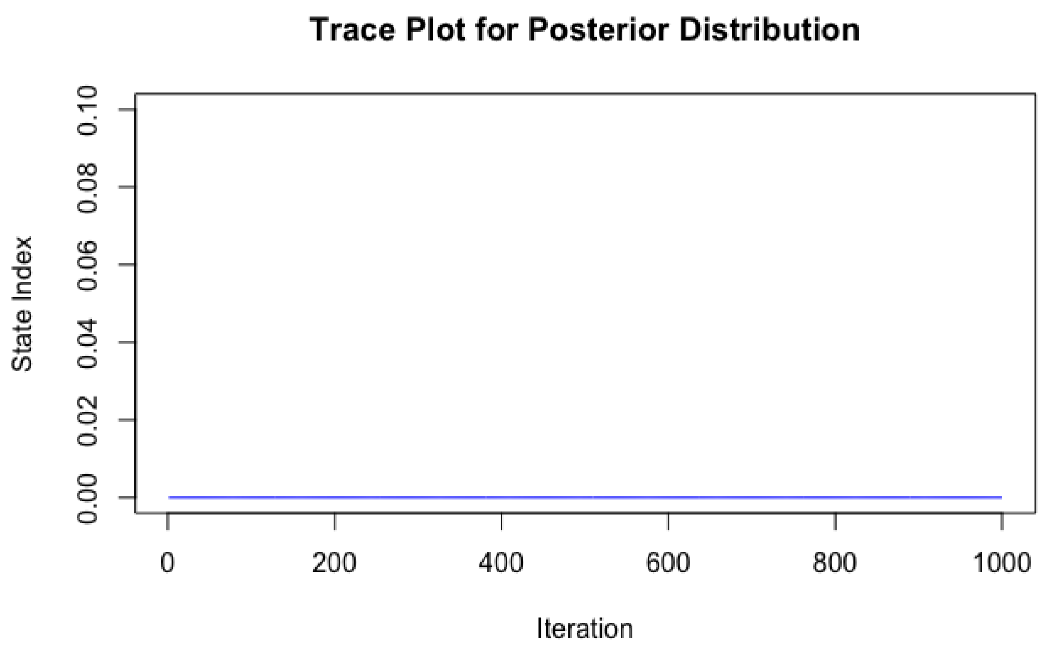
0.3.5 Convergence

There are many techniques to check for convergence of a markov chain. Here, we have used the trace plot to visualize how the sampled states change over iterations. This indicates whether the markov chain has reached a stable distribution or not. The trace plot has been plotted for the posterior distribution using R. The below shown trace plot comes out to be a horizontal line at value 0. This indicates a stable distribution and Markov Chain Monte Carlo (MCMC) algorithm has converged to a state where the sampled values of the parameter are centred around zero.

The same has been done in R with the following code. Below is the traceplot obtained:

```r

#CONVERGENCE
# Initialize the vector for trace plot
trace_plot <- numeric(num_iterations)

y_axis_limits <- c(0.0, 0.1)
# Create a trace plot for the posterior distribution
plot(1:num_iterations, trace_plot, type = "l", col =
    "blue", xlab = "Iteration", ylab = "State␣Index", main =
    "Trace␣Plot␣for␣Posterior␣Distribution",ylim =
    y_axis_limits)
```

Listing 5: Trace Plot



0.3.6 Stationarity

Stationary Distribution computation through Eigen Values

According to our Markov Chain model, we have finite number of state spaces. The markov chain is also irreducible, because there exists some number of matches, that Barcelona FC can play, such that it can it move from any state space of any state space. Thus, our Markov chain must have a unique stationary probability distribution.

In the context of Markov chains, eigenvalues and eigenvectors are used to find the stationary distribution of the chain, which describes its long-term behaviour. So we have calculated the eigen values and eigen vectors of the transpose of the transition probability matrix. We have then stored all the eigen values with values close to one in a vector to get the stationary distribution. This is because stationary distribution corresponds to the eigenvector associated with the eigen value 1.After normalizing the stationary distributions, we have printed the same normalized stationary distribution.

```r

# Compute the eigenvalues and eigenvectors
eigen_result <- eigen(t(transition_matrix_prob))  # Transpose the matrix for column-wise eigenvectors

# Find the index of the eigenvalue closest to 1 (stationary distribution)
index <- which(abs(eigen_result$values - 1) < 1e-8)

# Extract the corresponding eigenvector as the stationary distribution
stationary_distribution <- eigen_result$vectors[, index]

# Normalize the stationary distribution to sum to 1 (if needed)
stationary_distribution <- stationary_distribution / sum(stationary_distribution)

# Print the stationary distribution
print(stationary_distribution)
```

Listing 6: Stationary Distribution Calculation

Index	Value
1	0.08435003
2	0.04820002
3	0.07230003
4	0.07887276
5	0.07120457
6	0.03834092
7	0.18624882
8	0.05576080
9	0.01314546
10	0.01314546
11	0.01314546
12	0.01314546
13	0.06040038
14	0.04298050
15	0.06813300
16	0.03406650
17	0.01205000
18	0.02637684
19	0.02629092
20	0.01432683
21	0.01432683
22	0.01318842

Table 2: Values as a Table

If a Markov chain reaches stationarity at time period N, the transition probability matrix at time period N will be stagnant for all the transition probability matrices at time periods greater than N, i.e., $N+1, N+2, N+3, \ldots$. Thus, we have performed a simulation of matrix multiplication in R using the transition probability matrix. We have created an R loop where the transition probability matrix is multiplied by itself until we get the "current matrix" = $(\text{transition\_matrix\_prob})^N$.

In order to find out at what time period the transition probability matrix becomes stagnant, we have calculated this N, which comes out to be time period 42. This means that the transition probability matrix for time periods later than 42 should be identical to the current matrix. To verify the same, we have directly calculated the 43rd and 45th power of the transition probability matrix. On comparing them, the three matrices—current matrix and transition probability matrix to the power 43 and 45—come out to be equal.

```

\subsection{Matrix Convergence}

#MATRIX MULTIPLICATION OVER A NUMBER OF TIME PERIODS

```

```
# Define the matrix A
A <- transition_matrix_prob

# Define the tolerance level
tolerance <- 1e-6

# Initialize variables
N <- 1
previous_matrix <- A
current_matrix <- A %*% A  # A^2

# Loop until convergence
while (sum(abs(current_matrix - previous_matrix)) >
    tolerance) {
  N <- N + 1
  previous_matrix <- current_matrix
  current_matrix <- current_matrix %*% A  # Multiply by A
      again
}

# N is the smallest power at which convergence occurs
cat("Convergence_at_power_N=", N, "\n")

# The stationary matrix is in 'current_matrix'
cat("Stationary_Matrix_(A^N):\n")
print(current_matrix)
view(current_matrix)
# Calculate A^43 directly
power_43_direct <- expm::`%^%`(A, 43)
print(power_43_direct)
view(power_43_direct)

# Calculate A^45 directly
power_45_direct <- expm::`%^%`(A, 45)
print(power_45_direct)
view(power_45_direct)
```

Listing 7: Stationarity

0.4 Results

1. After calculating the transition probability matrix, it can be clearly observed that the sum of each row of the transition probability matrix is equal to 1. Thus, we can effectively conclude that the model represents a Markov Chain.

2. The heatmap shows that, there are very few non zero values in the transition probability matrix. This can also be logically verified by taking a

simple instance, that is, if Barcelona won the first three matches (WWW), it cannot go to WLD in the immediate next time period.

3. By performing the Monte Carlo Markov Chain simulation using the Gibbs Sampling algorithm, we were successfully able to arrive at a prior distribution and at a posterior distribution.

4. We have also taken a burn in period to take care of any biasedness that might arrive due to the sampling of the prior distribution.

5. The convergence of the posterior distribution was assessed using the traceplot diagram. The horizontal line at value zero verifies convergence. Additionally, we can conclude that the sampled values of the parameter are centred around zero.

6. The presence of finite state space and irreducibly in our Markov Chain concludes to a unique stationary distribution.

7. The Stationary distribution was calculated by finding out the eigen values of the transposed transition probability metrics. The resultant stationary distribution was summing up to one.

8. The posterior distribution is equal to this stationary distribution with some tolerance indicating a positive sign for convergence.

9. The verification of stationarity in a Markov Chain was performed by matrix multiplication of transition probability matrix for multiple time periods.

10. It was found out that the Markov Chain reaches stationarity at time period 42.

0.5 Inference

It can be observed that Barcelona's past 3 performances are sufficient to predict the result of the upcoming match. The probability transition matrix for Barcelona converges at time period 42, that is, 44th match. After playing 44 matches, the overall performance of the team becomes constant and we can predict the next match outcomes using the same set of probability values.