Applied Probability I Group Project Report

Part 1

To calculate the proportion of time during which the temperature was above 0 using a Monte Carlo Simulation, we used a stochastic process. The number of steps for the simulation (or the number of discrete time intervals) was calculated using the formula below:

$$steps = 1/\Delta t$$

Through this process, for various step sizes, we generated a standard random variable, Z, using numpy. Through the formula below, we incremented the temperature for every step.

$$T(t) = T(t-1) + \sigma\sqrt{\Delta}Z\ .$$

For each step, if the temperature was larger than the current max temperature, this temperature became the new max temperature. At the end of the simulation, the final max temp is the true max temperature overall.

Additionally, if the temperature was above zero, the above zero counter was incremented by 1, and the probability of the temperature being above zero (Probability, P) was calculated using the formula below:

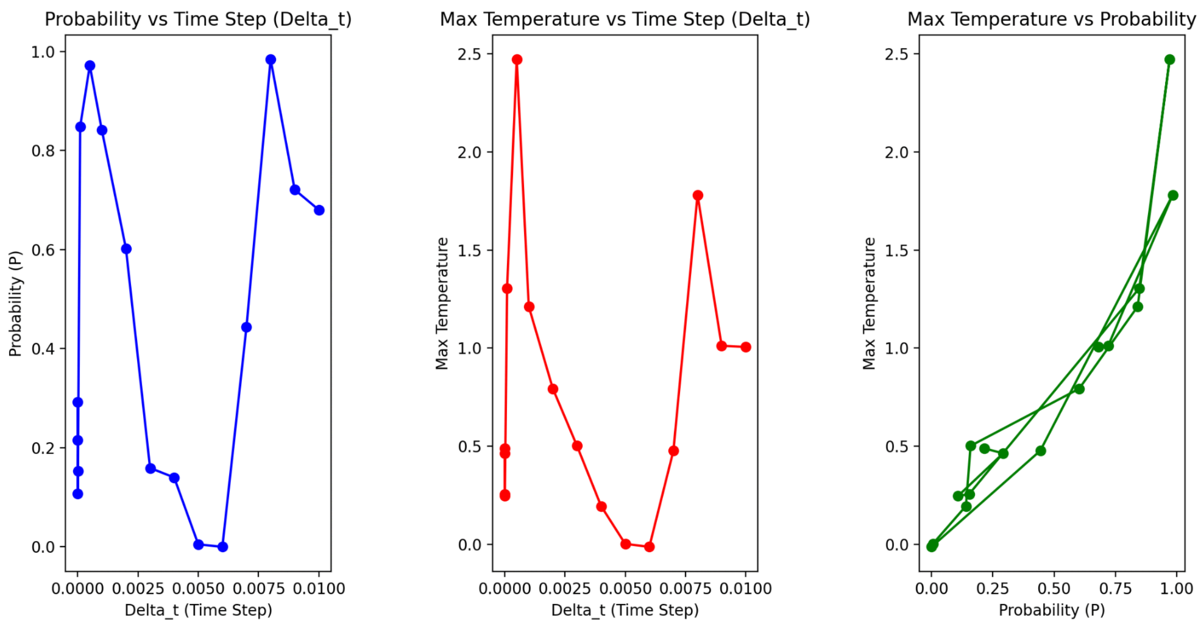
$$Probability\ of\ temperature\ above\ zero\ (P) = steps\ with\ T(t) > 0\ /\ number\ of\ steps$$

This process was repeated for multiple trials to obtain an estimate of the proportion of time the temperature was above zero.

For a trial run of the process as a part of the distribution, please see the results below (additional trials were excluded for brevity).

Note that for this trial run, the max temperature and probability are strongly correlated, with a value of 0.9237 for the correlation coefficient, *R*. This suggests that higher maximum temperatures are associated with a greater proportion of time the temperature remains above zero.

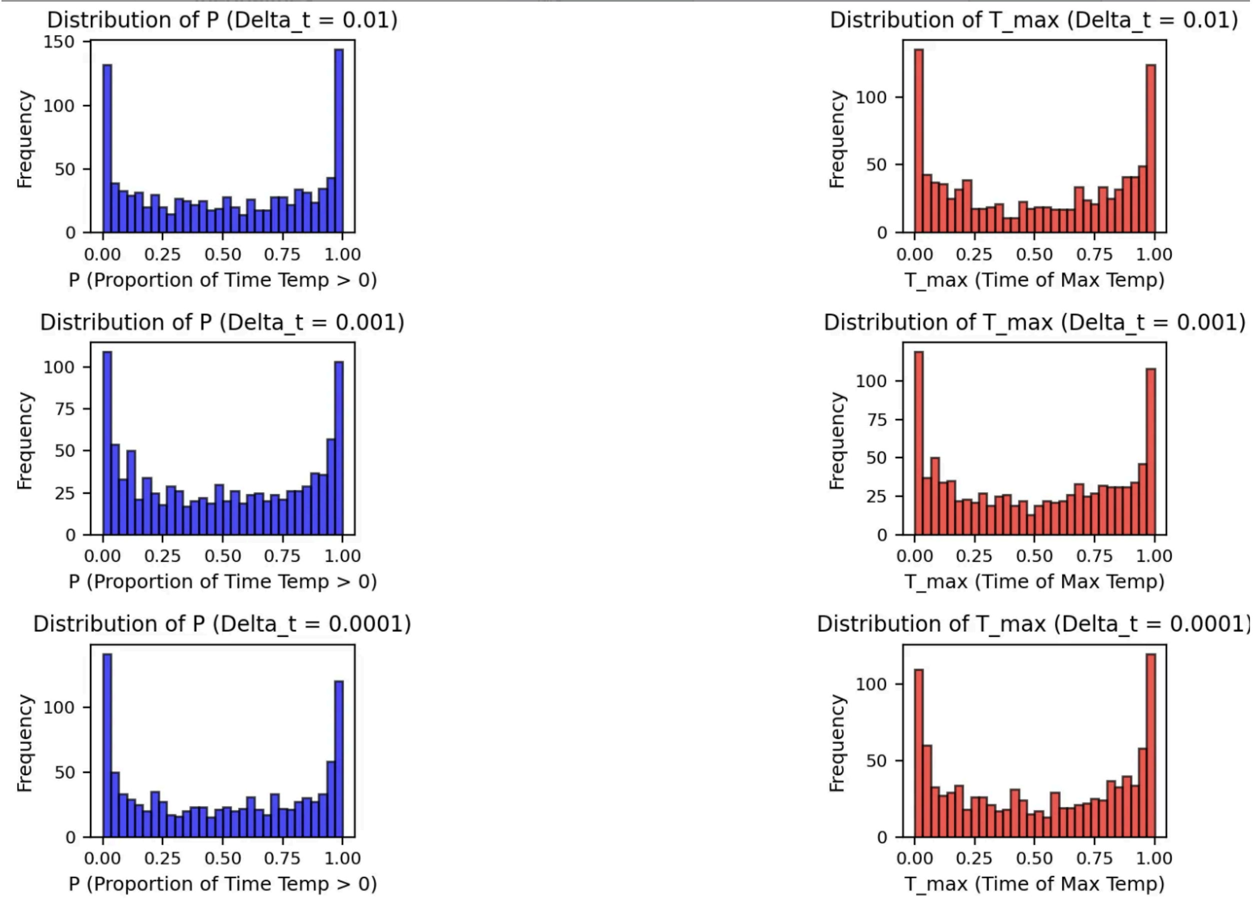
For the larger value of Δt, the results began to converge more towards a higher correlation coefficient *R.* This is because there is more sample point to the temperature evolution, making the simulation more precise, and making the irregularities be evened out. For a smaller value of Δt, where $\lim_{\Delta t \to -\infty}$, we could get a correlation coefficient near R=1, however this would be very computationally expensive as the more precise the simulation is, the more computing power it would require.

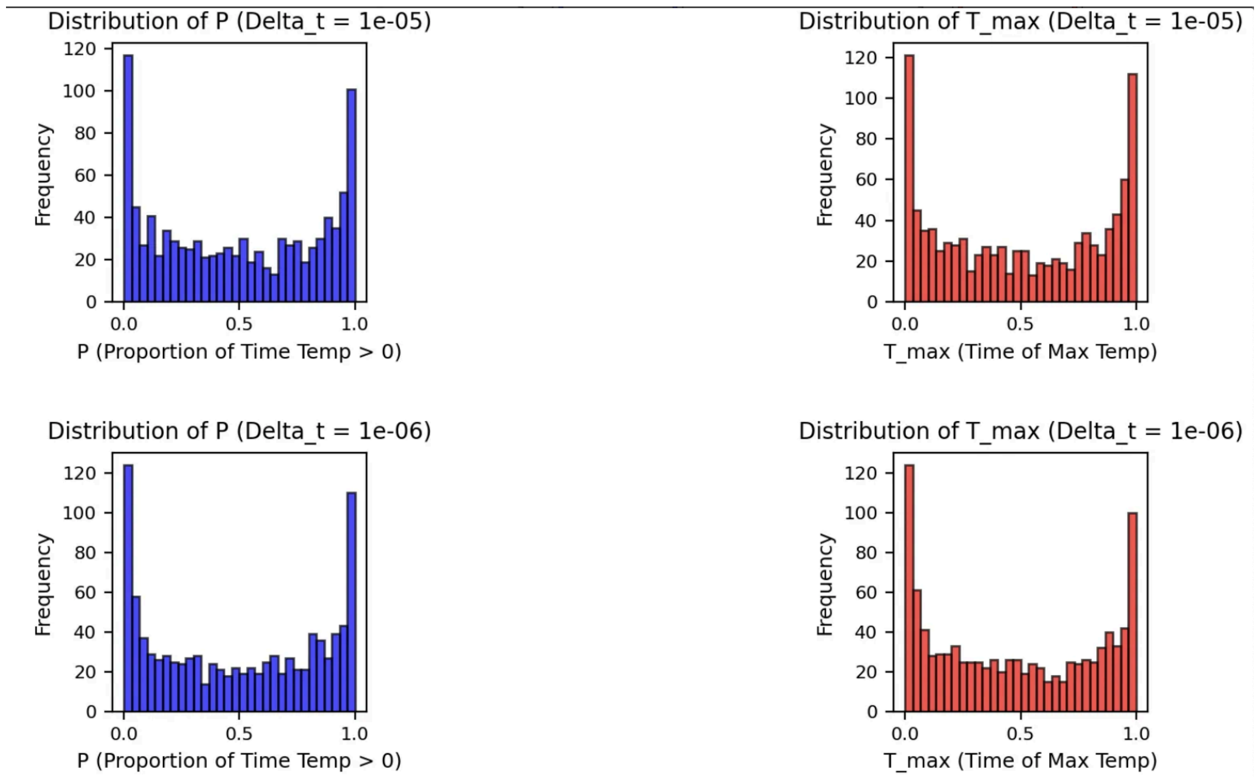


Delta_t (Δt)	Probability (P)	Max Temperature (Tmax)
0.0100	0.6800	1.0063
0.0090	0.7207	1.0126
0.0080	0.9840	1.7816
0.0070	0.4437	0.4777
0.0060	0.0000	-0.0123
0.0050	0.0050	0.0028
0.0040	0.1400	0.1955
0.0030	0.1592	0.5025
0.0020	0.6020	0.7934
0.0010	0.8410	1.2124
0.0005	0.9715	2.4726
0.0001	0.8477	1.3056

0.00001	0.1534	0.2568
0.000001	0.1074	0.2466
0.0000001	0.2919	0.4642
0.00000001	0.2152	0.4895

To estimate the distribution of P and Tmax, the simulation was run 1000 times for each value of Δt, and the distribution of P and Tmax was recorded, as below:





The resulting distributions of both P and Tmax appear to follow a U-shaped Beta distribution, with parameters α=0.5 and β=0.5.

The U-shaped Beta distribution suggests that extreme values of P (either close to 0 or close to 1) are more likely than values in the middle, implying that in some simulations, the temperature remains mostly negative or mostly positive over the time interval.

Part 2

Constructing the model

The model used to predict the league was designed based on the example given in the project brief: the number of shots on target made by each team in a match was taken to be a Poisson-distributed random variable, independent of the team being played against. The average rate at which a shot on target resulted in a goal being scored was computed using the average proportion of goals conceded by the defending team in their first 11 matches of this league. This expected rate of concessions was taken as a constant parameter that did not vary depending on the team being played against. This method was used to run Monte Carlo simulations of each match in the league that had yet to be played at 12/11/2024, the time at which data was gathered for this project.

To justify the choice of a Poisson distribution for shots on target, data was gathered from the Premier League website on each of the 11 matches played by each team in the league at the time of recording. These numbers of shots on target made by each team in these matches were compared to a Poisson distribution.

Team	Arsenal	Aston Villa	Bournemouth	Brentford	Brighton
Arsenal		4	1		7
Aston Villa	3		8		
Bournemouth	4	3		3	
Brentford			6		
Brighton	4				

Table 2.1 Sample of table showing the number of shots on target made by each team in the leftmost column in their matches against each team on the top row. Empty cells indicate matches that had not yet taken place on 12/11/2024.

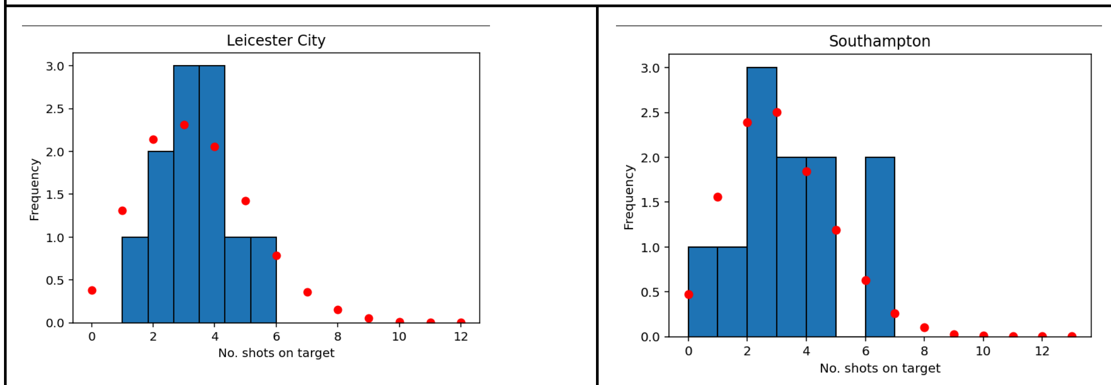


Fig 2.1 Frequency distribution of shots on target (blue) compared to Poisson distribution with rate parameter = average shots on target (red) for Leicester City and Southampton.

There were outliers in these distributions - for example, the frequency of 6 shots on target for Southampton in the above figure is well above what the corresponding Poisson distribution would predict - however, considering the very small sample size of 11, the fit is close enough to suit the purposes of this model. Furthermore, the choice of a Poisson distribution to model the number of shots on target per match makes sense intuitively, as we are counting the number of independent events to take place in a fixed time interval. We therefore proceeded with our model under the assumption of a Poisson distribution for the number of shots on target.

<u>Simulating matches</u>

Matches in the league were simulated using the Monte Carlo method. For each team playing, their average number of shots on target per match and average rate of saves were taken from the dataset compiled from the Premier League website as at 12/11/2024. The score for each team was calculated as follows:

$$Score_A = s.o.t._A * (1 - save_B)$$

Being a Monte Carlo simulation, this was repeated some arbitrarily large number N times, and the average result taken as the expected outcome of the match.

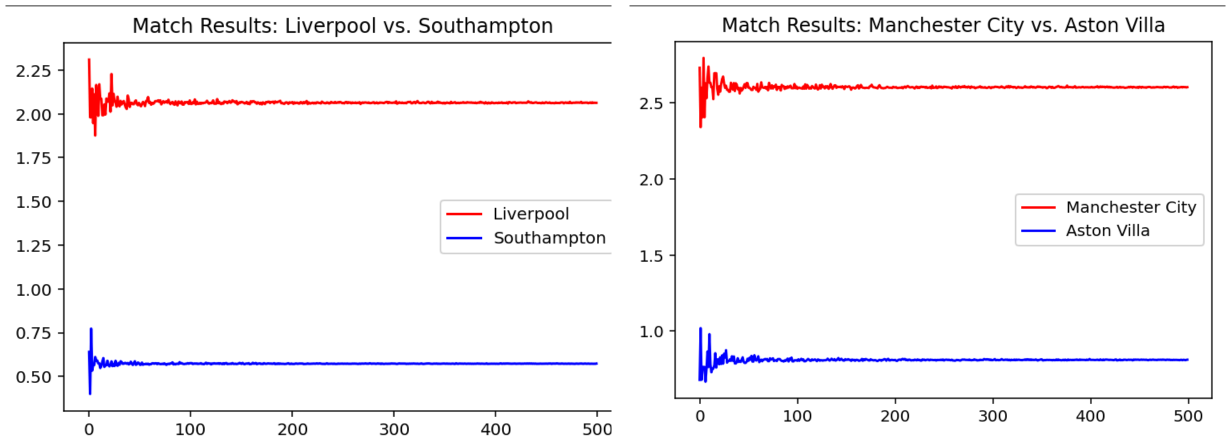


Fig 2.2 Monte Carlo simulation results of matches between Liverpool and Southampton and between Manchester City and Aston Villa

<u>Projecting the league</u>

To create a prediction of the final results of the Premier League, all that remained was to run Monte Carlo simulations of all remaining matches in the league, and amend the existing league table in accordance with the results - 3 league points for a win, 0 for a loss, and 1 for a draw (note that due to the design of the model, a draw is almost impossible to achieve, as the expected results of each match are computed to a large number of decimal places).

This resulted in the final league table as follows:

Team	Points	Team	Points
Liverpool	54	Manchester United	27
Chelsea	48	Crystal Palace	21
Newcastle United	48	Brighton	21
Manchester City	45	West Ham United	15
Fulham	39	Everton	12
Nottingham Forest	39	Aston Villa	6
Bournemouth	33	Ipswich Town	6
Brentford	33	Leicester City	3
Arsenal	30	Southampton	3
Spurs	30	Wolverhampton Wanderers	0

Model reflection

This model for the Premier League was simple and relatively easy to compute. However, it makes a lot of assumptions and omits many potentially important factors. For example, it assumes that the shots on target that a team makes are independent events, and that they do not depend on the team being played against.

One potential improvement to the accuracy of the model could be to consider how a team's number of shots on target varies depending on their opponent's defense. This could be done by changing the random variable of the model to the total number of shots, and computing a "defense quality" stat for each team representing the average proportion of shots taken against them that are on target.

The model used in this simulation ran Monte Carlo simulations on each match, and took the expected value of each team's score in that match to be final. This is sufficient for most matches with a clear outcome. However, where the expected scores of the match would be very similar, this is not necessarily ideal - for example, if the expected score of a match between Liverpool and Arsenal is computed as 2.01-1.99, to award 3 league points to Liverpool and none to Arsenal as this model does is ignoring a large number of cases where Arsenal may in fact win that match.

One way around this limitation might be to run Monte Carlo simulation at the league level - that is, we simulate some large number N instances of the league, where each match is simulated once and a league table generated for each instance using the results of each match. A final league table could then be computed by finding the average league points of each team across every simulated instance of the league.

This method might better account for variation in the outcomes of individual matches. However, as it would produce a large number of different league tables, it would be extremely computationally intensive to run this method with a sufficient number of repetitions to yield accurate results, and it was therefore decided that such an algorithm would be outside the scope of this project.

References

Monte Carlo Simulations for Predicting Stock Prices (Medium article)
Analytics Vidhya, 2020. *Monte Carlo simulations for predicting stock prices using Python*. [online] Available at: https://medium.com/analytics-vidhya/monte-carlo-simulations-for-predicting-stock-prices-python-a64f53585662 [Accessed 30 November 2024].

Matplotlib Documentation
Matplotlib, 2024. *Matplotlib documentation*. [online] Available at: https://matplotlib.org/stable/index.html [Accessed 30 November 2024].

NumPy Documentation
NumPy, 2024. *NumPy documentation*. [online] Available at: https://numpy.org/ [Accessed 30 November 2024].

Beta Distribution (Statistics by Jim)
Jim, S., 2020. *Beta distribution*. [online] Available at: https://statisticsbyjim.com/probability/beta-distribution/ [Accessed 30 November 2024].