MARKOV CHAIN MODELING OF SUNSPOT NUMBERS

DATA MODELING PROJECT FOR MATH 7241

Tharini Padmagirisan

NUID: 001586312

Northeastern University

December 2021

Contents

1 Introduction

A Markov chain is a stochastic process where the distribution of a future state of a random sequence depends only on the distribution of the current state and not the past states.

$$P(X_{n+1} = j|X_0 = i_0, ..., X_n = i_n) = P(X_{n+1} = j|X_n = i_n) \quad (1)$$

for all n $\geq$ 0 and all states j,i_0, . . . ,$i_n \in \Omega$.

The objective of this project is to analyze a time-series and find out if the Markov chain method produces a good model for the time series. I will be analyzing the daily number of sunspots for this project.

2 Data Description

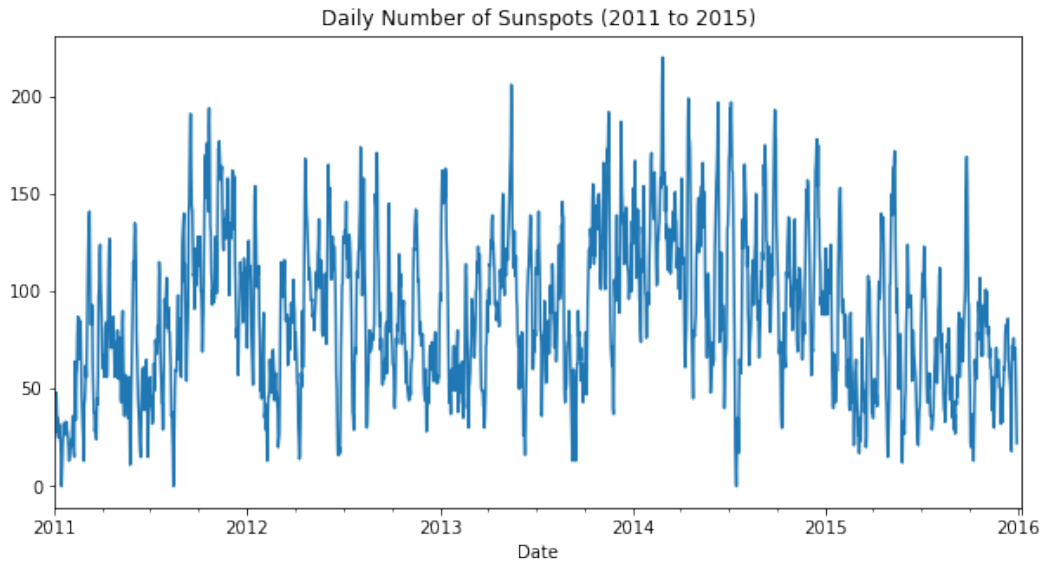
Sunspots are one of the most observed astronomical phenomena. They are dark areas observed on the sun's photosphere. They appear randomly as they are produced as a result of intense magnetic influx. The magnetic pressure increases in these areas while the atmospheric pressure in the surrounding area decreases. The strong magnetic field prevents flow of hot gas from the Sun's interior to the surface lowering the temperature on these areas making them cooler and darker compared to their surrounding area.

I have downloaded the daily sunspot numbers from the repository of WDC-SILSO, Royal Observatory of Belgium, Brussels. Daily sunspot numbers are available from 1818 before which the daily observations were sparse and only yearly and monthly numbers are available. I will be analyzing the sunspot numbers observed in the years 2011 to 2015, as this period falls in the middle of a solar cycle that started in 2008 thus excluding periods of extremities.

Summary statistics of daily sunspot numbers (2011-2015),

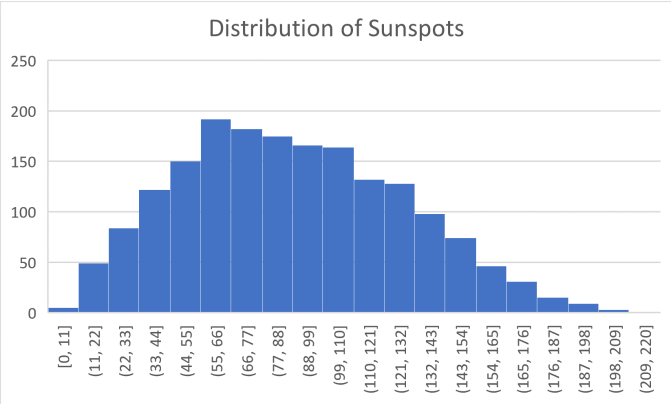
Statistic	Value
Count	1826
Mean	88.5
Median	85
Minimum	0
Maximum	220

Here's a plot of the timeseries,



3 Analysis and Modeling

There are 186 distinct count of sunspots in the data. Here is a distribution of the counts,

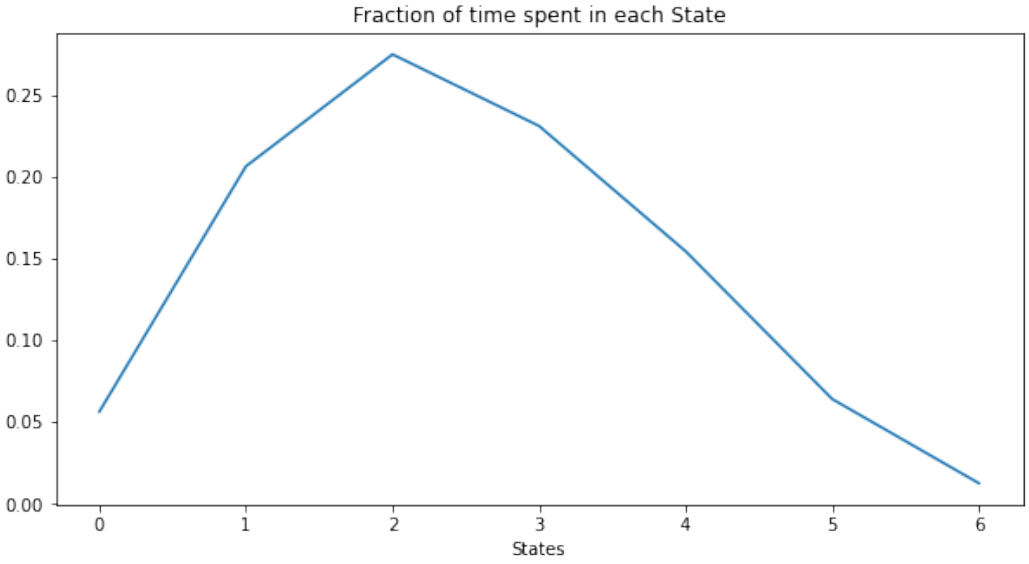


Based on the above distribution, I have divided the data into groups of 30 sunspot counts until 180 and all the counts ≥ 180 are in a single group. This gives **7 Markov states**.

Number of Sunspots	Markov State
0 - 29	0
30 - 59	1
60 - 89	2
90 - 119	3
120 - 149	4
150 - 179	5
≥ 180	6

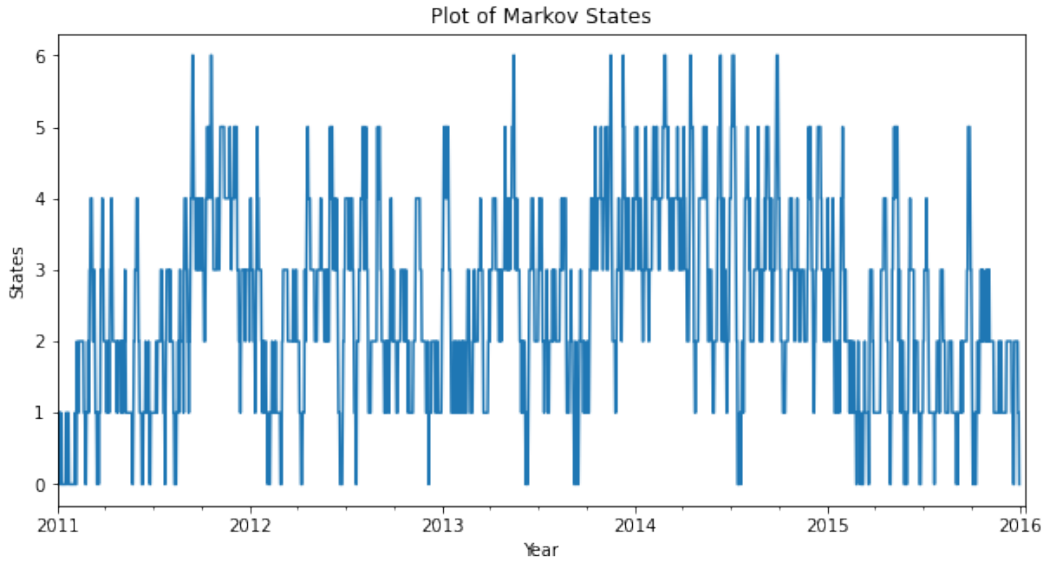
Note: I have used Excel and Python for my analysis.

3.1 Empirical Distribution



The empirical distribution of the states is right-skewed. The third state is the median. Based on the data, we can see that on most days the sunspots range between 55 and 90. Higher number of sunspots (> 100) occur at a lesser frequency compared to the lower number groups.

We could also visualize the Markov states as a time series,



3.2 Transition Matrix

The transition probabilities for the different states are as follows,

0.598	0.343	0.059	0.000	0.000	0.000	0.000	0
0.106	0.658	0.225	0.011	0.000	0.000	0.000	1
0.004	0.183	0.641	0.167	0.004	0.000	0.000	2
0.000	0.002	0.199	0.602	0.194	0.002	0.000	3
0.000	0.000	0.018	0.280	0.521	0.170	0.011	4
0.000	0.000	0.000	0.009	0.410	0.521	0.060	5
0.000	0.000	0.000	0.000	0.130	0.304	0.565	6
0	1	2	3	4	5	6	

Probabilities along the diagonal are higher than the others. Sunspots change gradually as a solar cycle progresses. As I have grouped 30 counts in a state, the probabilities for jumping to the same state is higher than others.

To compute the Transition matrix, I wrote a function in Python available in the Appendix of the report.

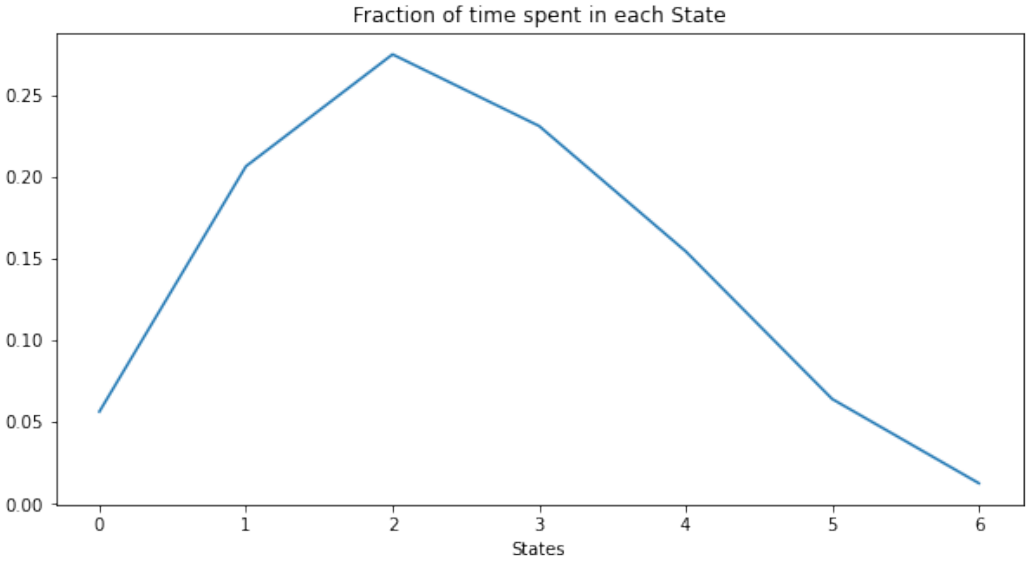
3.3 Stationary Distribution

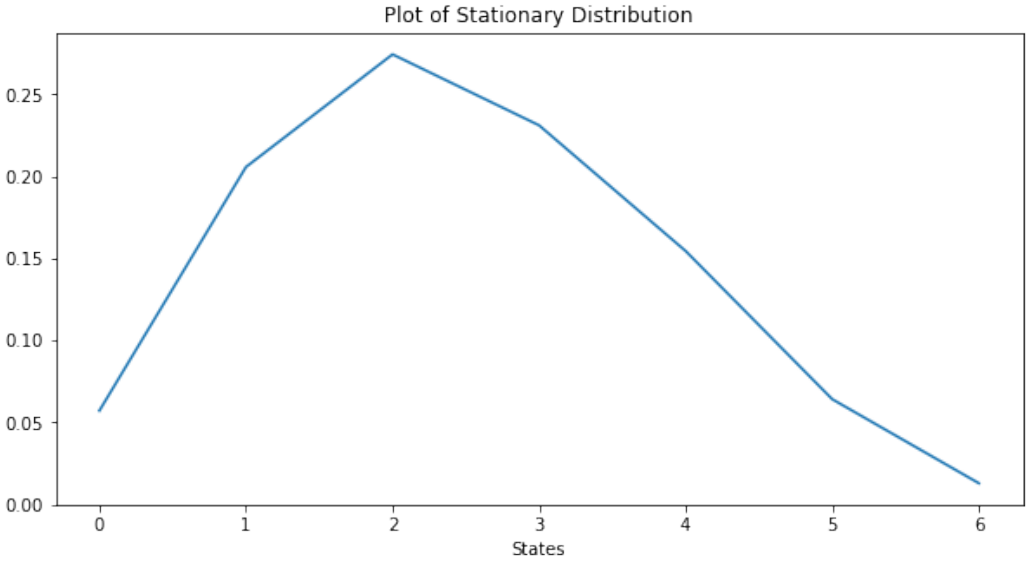
This is the stationary distribution of the chain,

W = 0.057 0.206 0.274 0.231 0.155 0.064 0.013

From the stationary distribution, we can see that the long run probabilities are higher for states 2 and 3 are the highest. For the period I have selected for analysis, which is after a solar minimum we can expect most of the days to have a sunspot count in the range that corresponds to the states 2 3 which is slowly moving towards a solar maximum.
Function to compute the stationary distribution is included in the Appendix of the report.

Comparison of empirical distribution from original distribution to the stationary distribution of the states,





From the above plots we can see that stationary distribution and the distribution of states from the time series are similar.

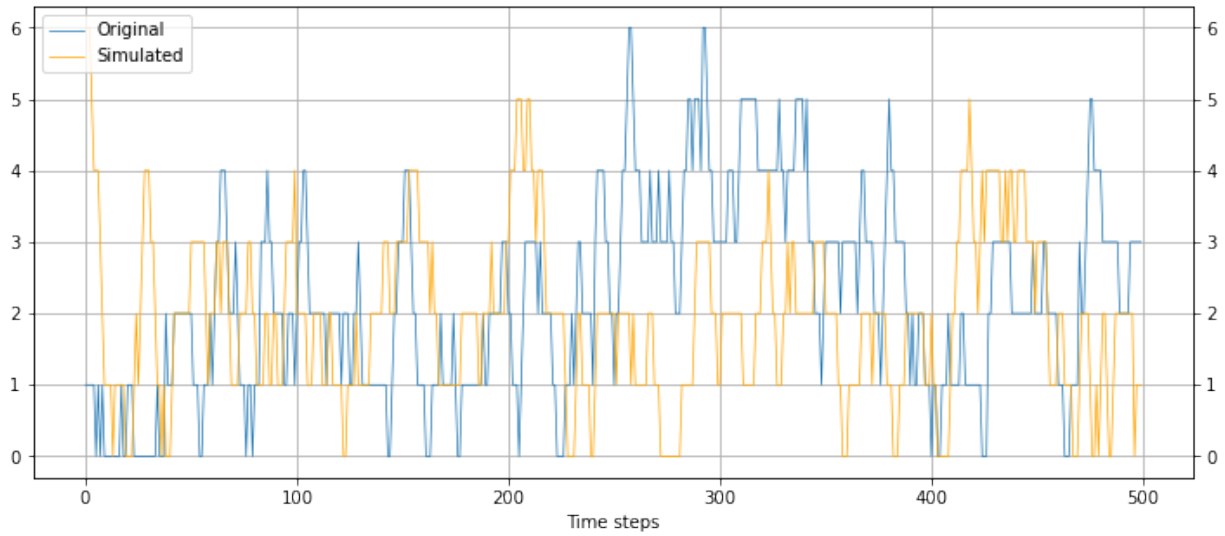
3.4 Simulation of Time Series

I wrote a function to generate a time series based on the transition matrix of the original time series.
First state is generated randomly. The next state is selected based on the transition probabilities of the first

state. I have converted the transition probabilities of each state into a cumulative distribution. Then I select the next state based on a random number generated between 0 and 1.

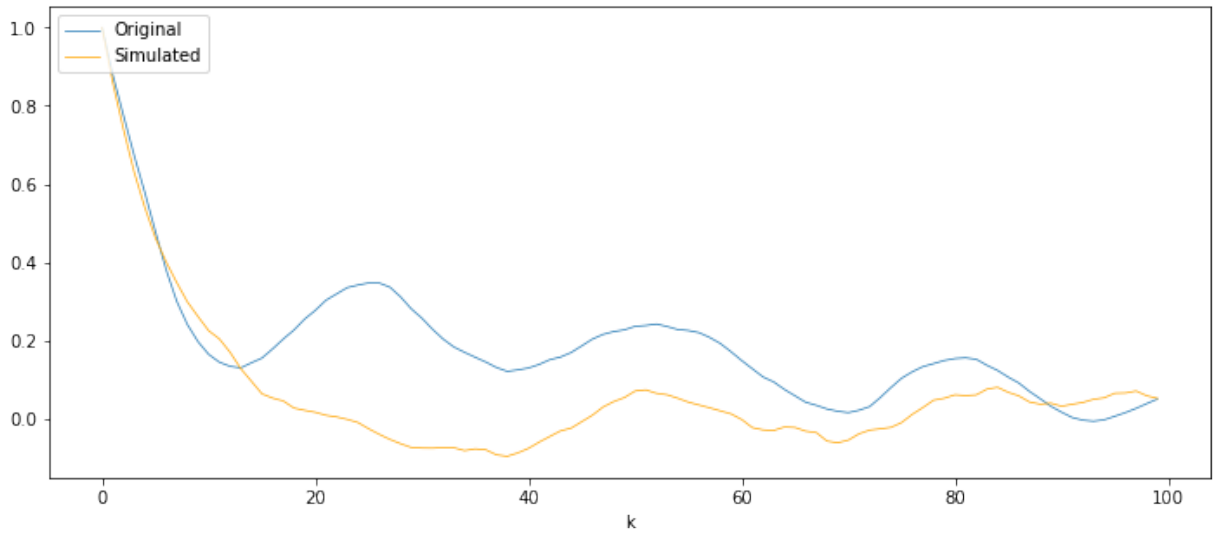
For example, if the first state is 5, the transition probabilities for state 5 are [0 0 0 0.009 0.410 0.521 0.060]. The cumulative distribution is [0 0 0 0.009 0.419 0.94 1]. Now a random number between 0 and 1 is generated, say 0.34. Now, the next state is chosen as 4 based on above distribution. This is iterated and a new series is generated with 1000 values.

Here's a comparison of transition of states of the original series and the simulated series for first 500 time steps,



3.5 Comparison of Auto-Correlation Factors

I have computed Auto-correlation factors for k = 0 to 100 for the original and simulated time series. Here's a comparison of the auto-correlation factors,



The factors are close to each other in the beginning. The difference starts increasing after k = 15 and starts decreasing again shortly thereafter.

As a further analysis, I did a quick correlation check between the two series, which gave a Pearson correlation value of 0.78 which indicates a high correlation between the series. This indicates that the series simulated from the transition probabilities of the original series follows a similar transition pattern to the original.

Additionally we will also be subjecting the original time series to a Goodness of Fit Test. The details are in the following section.

3.6 Goodness of Fit Test

We further analyze our original time series using goodness of fit test for the 2-step transition probabilities. We compare the observed 2-step jumps in our original time series with the expected jumps which is obtained by multiplying the 2-step jumps by the 2-step transition probabilities (square of our original single step transition matrix) for each state.

We are testing our hypothesis that the observed frequencies follow the expected frequencies for two-step jumps. We compute the test statistic (Pearson's) and the chi-squared distribution for each state and reject the null hypothesis if TS $>$ chi-squared. We are using 5% significance level.

We get the following results,

State	Test Statistic	Chi-squared Value
0	0.16	7.81
1	2.04	9.48
2	7.74	11.07
3	2.77	12.59
4	1.61	11.07
5	3.82	9.48
6	22.22	9.48

Based on the above results, our null hypothesis can be rejected for state 6 and we don't have enough evidence to reject the null hypothesis for states 0 to 5. So, the Markov chain is a good model for 2-step transitions of the original time series.

4 Conclusion

From the auto-correlation plots, we can see that the factors decrease as lag increases meaning that the correlation is growing less as the gap between the states increases. The randomly simulated time series based on the transition probabilities of the original series also has a similar behaviour. The goodness of fit test also has enough evidence only to reject the hypothesis for one state which is state 6 where the higher counts of sunspots are grouped.

From the above analyses, I am concluding that the Markov chain method produces a good model for the daily number of sunspots observed in the middle of a solar cycle.The number of sunspots over a longer period can be analyzed to understand their pattern over solar cycles.

Appendix

Here are the Python functions I wrote for the project,

Transition Matrix

```
def TransitionMatrix(states):
    pd.value_counts(states)
    y = len(pd.unique(states))
    P = np.zeros([y,y])

    for i in range(y):
        for j in range(y):
            for x in range(len(states)-1):
                if states[x] == i and states[x+1] == j:
                    P[i][j] += 1

    for row in P:
        s = sum(row)
        if s > 0:
            row[:] = [round(f/s,3) for f in row]
    return P
```

Stationary Distribution

```
def StationaryDistribution(P):
    A = P.T-np.identity(P.shape[0])
    A = np.vstack([A,np.ones((P.shape[0]))])
    b = np.zeros((P.shape[0])).T
    b = np.zeros((P.shape[0]+1,1))
    b[-1] = 1
    W = np.linalg.lstsq(A,b,rcond=None)[0]
    return W
```

Note: Of the methods linalg has to solve linear equations, lstsq is a method that does not require the matrices to be square with full rank for solving. We don't have a square matrix after including the normalization condition.

Simulation of Time Series

```
def SimulateSeries(P,k):
    P_dist = [np.cumsum(P[i, :]) for i in range(P.shape[0])]
    A = np.zeros(k, dtype=int)
    A[0] = random.choice(range(P.shape[0]))
    for t in range(k-1):
        x = round(random.uniform(0,1),2)
        arr = P_dist[A[t]]
        A[t+1] = min([i for i, e in enumerate(arr) if e >= x])
    return A
```

Auto-Correlation

```
def Autocorrelation(X,k):
    X_b = np.average(X)
    n, d = 0, 0
    for i in range(0,len(X) - k):
```

```
        n += ((X[i] - X_b)*(X[i+k] - X_b))
    for i in range(0,len(X)):
        d += (X[i] - X_b)**2

    return n/d

ACF1, ACF2 = np.zeros(100), np.zeros(100)
for i in range(len(ACF1)):
    ACF1[i] = Autocorrelation(originalstates,i)
    ACF2[i] = Autocorrelation(simulatedstates,i)
```

Goodness of Fit

```
def GoodnessofFit(N,Q):
    n = 0
    TS = np.zeros((N.shape[0]))
    chi2 = np.zeros(N.shape[0])
    for i in range(N.shape[0]):
        N1 = N[i][Q[i]>0]
        n = N1.sum()
        chi2[i] = stats.chi2.ppf(q = 0.95, df = len(N1) - 1)
        for j in range(N.shape[0]):
            if Q[i][j] > 0:
                O = N[i][j]
                E = n * Q[i][j]
                TS[i] += ((O - E)**2 / E)
    return TS, chi2
```

where N is the matrix with 2-step jump frequencies and Q is the 2-step transition probability matrix.