Northeastern University

Sakshi Suman

Data Modeling Project on Markov Chains

MATH 7241 - Probability I

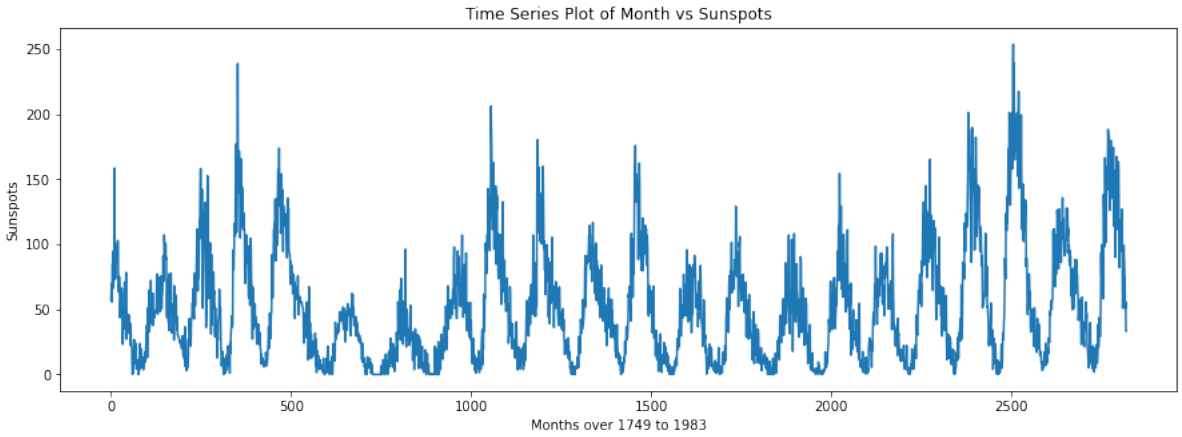
Supervisor: Prof. Christopher King

1 Description of Data

Sunspots are temporary phenomena on the Sun's photosphere that appear as spots darker than the surrounding areas. They are regions of reduced surface temperature caused by concentrations of magnetic field flux that inhibit convection. Sunspots usually appear in pairs of opposite magnetic polarity. Their number varies according to the approximately 11-year solar cycle. I downloaded the Monthly Sunspots data from Kaggle.This dataset describes a monthly count of the number of observed sunspots for 230 years (1749-1983). The units are a count and there are 2,820 observations. The source of the dataset is credited to Andrews & Herzberg (1985).

2 Cleaning of Data

This is a univariate time series data. My data is just a combination of months and sunspots from the year 1749 to the year 1983. The time series plot of months vs sunspots is given below:



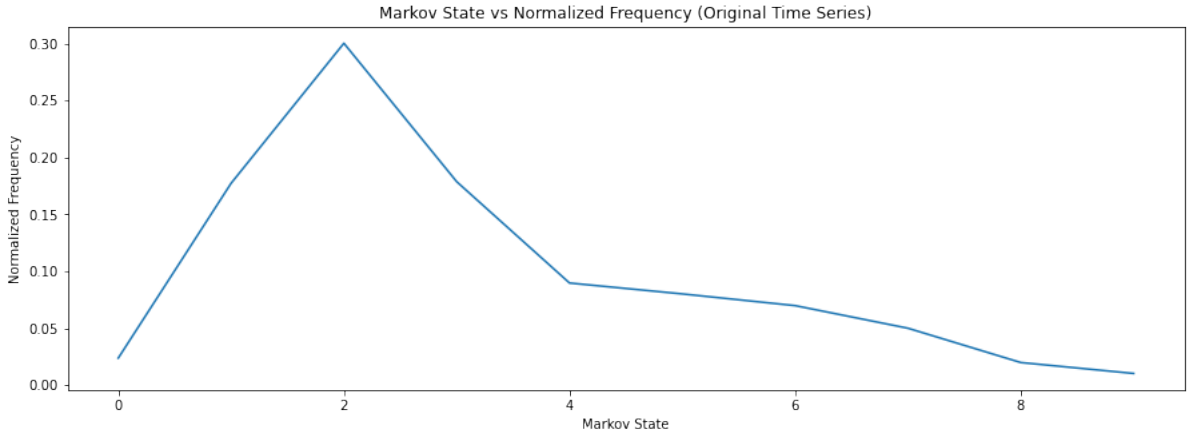
3 Data Analysis & Modeling

From the above graph, I observed that there are several outliers. For example, During 500th and 2500th month, the sunspot is very high. I first tried diving the range of average runs into equal intervals and found that my modelling is behaving poorly for all the states. I realized that is not a fair assumption to divide the intervals of the range of average runs into equal lengths. Hence, upon having a close look at the data I found that diving the values such that each state follows the following distribution is a fair assumption.

```
quantiles = [0.000001, 0.2, 0.5, 0.68, 0.77, 0.85, 0.92, 0.97, 0.99, 1.0]
```

The total number of states in the Markov Chain are 10. This produces a right skewed curve. So, the time series frequency plot of Markov State vs the Normalized Frequency should be designed in such a way that median values are closer to least numbered Markov State as it would allow less number of intervals (hence more gap occupied) below the median thus allowing the future

value to choose a Markov State of smaller number with higher probability than a Markov State of larger number. According to the above theory, the Markov State vs Normalized Frequency Plot would look like the below graph:



3.1 Transition Matrix

I wrote a function to calculate the transition matrix from the original time series data by first calculating a count matrix of the transitions from state i to state j in one step. I then normalized each entry in the matrix by diving it by the total transitions from that state to any other state which is nothing but the sum of that particular row. The transition matrix is:

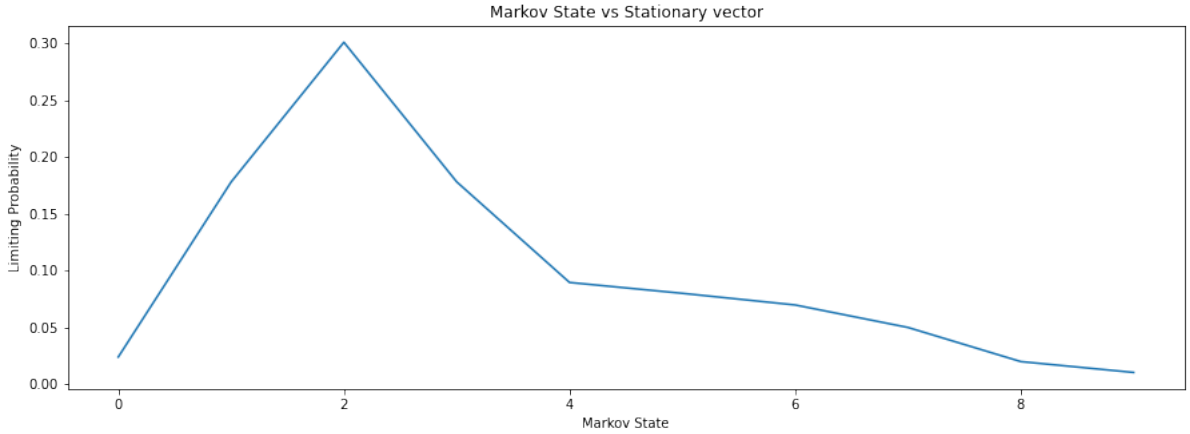
0.537	0.402	0.250	0.059	0.000	0.000	0.000	0.000	0.000	0.000
0.056	0.668	0.274	0.002	0.000	0.000	0.000	0.000	0.000	0.000
0.003	0.163	0.662	0.114	0.021	0.005	0.001	0.000	0.000	0.000
0.000	0.001	0.244	0.488	0.174	0.075	0.011	0.003	0.000	0.000
0.000	0.000	0.059	0.387	0.296	0.173	0.071	0.007	0.003	0.000
0.000	0.000	0.030	0.123	0.216	0.331	0.225	0.061	0.004	0.004
0.000	0.000	0.005	0.040	0.101	0.228	0.375	0.208	0.030	0.010
0.000	0.000	0.000	0.000	0.021	0.134	0.297	0.382	0.113	0.049
0.000	0.000	0.000	0.000	0.000	0.017	0.089	0.446	0.357	0.089
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.103	0.413	0.482

3.2 Stationary Distribution

I wrote a function to compute the limiting probability vector (stationary vector) for the empirical distribution. I used the least squares method to get the approximate values for my stationary vector as the rank(P) for a randomly chosen data is most likely going to be $'n'$ whereas the system is inconsistent as I have total n+1 independent equations including the normalisation condition, that is the sum of $w_i = 1$. I chose to get the least squares approximation for this linear system as my stationary vector. The stationary vector looks like the following:

```
W = [0.023, 0.177, 0.301, 0.178, 0.089, 0.079, 0.069, 0.049, 0.019, 0.010]
```

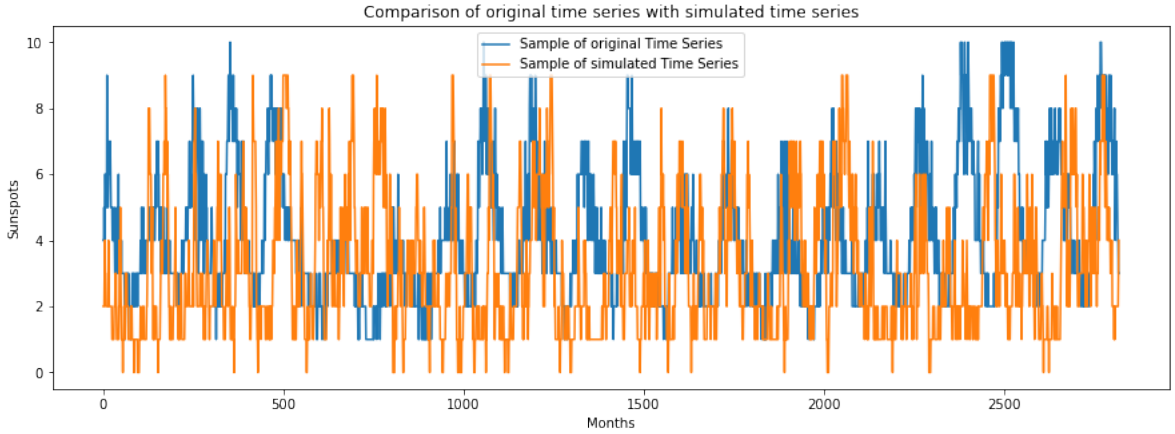
The Markov State vs stationary vector plot would look like the following graph:



This plot is similar to the Markov State vs Normalized Frequency for original time series(since the frequencies for each state is the same).

3.3 Simulated Time Series Generation

I generated a new time series from the stationary vector by first selecting a random seed from state 0 to state 9 (highest Markov state id, because total states = 10 and initial state id = 0). I computed my next state by following the probability distribution that was obtained from the transition matrix computed of the original time series(shown above). For example, if the first random state is 2, then the values in the third row in the transition matrix are picked (the probability vector) and the second state is generated based on this probability vector. If the second state is 4, then the vaalues in the fourth row in transition matrix is picked, the third state is generated based on this probability vector and so on, till I get 2820 observations. This is the new time series. I compared my original time series with the simulated time series by plotting a graph. The graph is given below:



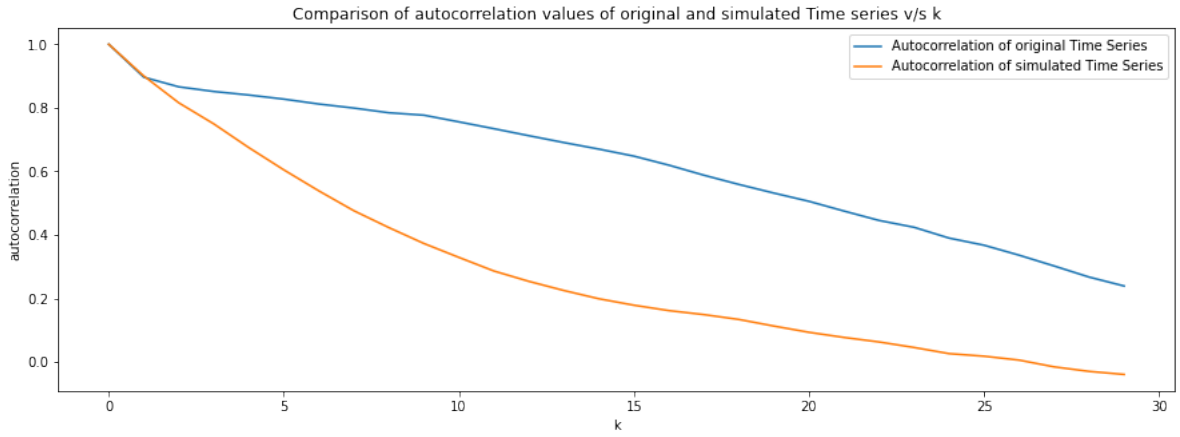
3.4 Auto-Correlation Comparison

Applying the auto correlation formula given in the project descripton, the correlation values for original time series and simulated time series are given in the following table:

Auto correlation values (R)		
k	Original Time Series	Simulated Time Series
0	1.000	1.000
1	0.896	0.905
2	0.866	0.820
3	0.851	0.749
4	0.840	0.678
5	0.827	0.617
6	0.812	0.560
7	0.799	0.503
8	0.784	0.452
9	0.777	0.405
10	0.756	0.360
11	0.734	0.318
12	0.712	0.280
13	0.690	0.250
14	0.670	0.224

15	0.648	0.202
16	0.619	0.183
17	0.588	0.162
18	0.559	0.146
19	0.531	0.136
20	0.505	0.119
21	0.475	0.102
22	0.445	0.093
23	0.424	0.084
24	0.390	0.077
25	0.367	0.070
26	0.336	0.062
27	0.302	0.054
28	0.267	0.046
29	0.239	0.044

It is clear that for k=0, the auto-correlation value has to be 1.0. The percentage of difference increases as we go down the table. This is in general not a good sign and can lead to a conclusion that our original time series is not a Markov Chain. The plot for the comparison of auto-correlation values of original and simulated time series for k= 1, 2, 3, 4, ..., 30 is given below.



3.5 Goodness of Fit Test

A Goodness of Fit Test for two step transition using chi square and Pearson's test statistic (where N is the count matrix(calculated while calculating the transition matrix) and Q is P^2) of original time series is computed in order to see if our original time series follows the Markov Principle (the memoryless property). The Pearson Test Statistic values are computed for each state which is compared with the chi-squared probability density function with corresponding degrees of freedom obtained from empirical distribution with two step transition where the entry is greater than zero in theoretical distribution and 5% significance level. This comparison yielded following results: ‘

Chi-square vs Test Statistic for each state			
State	Chi Square Value	Test Statistic	Outcome
0	5.991464547107979	177.4445109841738	Rejected
1	7.814727903251179	1331.340288628769	Rejected
2	12.591587243743977	11537.267950497691	Rejected
3	12.591587243743977	11153.104305794925	Rejected
4	12.591587243743977	2882.144903953539	Rejected
5	14.067140449340169	2277.300951300658	Rejected
6	14.067140449340169	1869.7987804229078	Rejected
7	11.070497693516351	791.6406956552881	Rejected
8	9.487729036781154	808.1391298330243	Rejected
9	5.991464547107979	121.12241604642537	Rejected

4 Conclusions

This comparison shows that the Goodness of Fit test rejected all the states. This is another confirmation apart from the autocorrelation values that the Monthly Sunspots time series does not obey the Markov Rule (Memoryless Property). This is just an analysis for homogenous Markov Chains. We can test it for non-homogenous Markov Chains as well. However, this analysis is still very useful because if it works out well, our model is greatly simplified and if it doesn't work well then we can conclude to not use Homogenous Markov

Chain Modeling for this problem.

5 Appendix

5.1 Function to compute transition matrix

```
def compute_transition_matrix_fast(data, n, step = 1):
    t = np.array(data)
    step = step
    total_inds = t.size - (step + 1) + 1
    t_strided = np.lib.stride_tricks.as_strided(
                                    t,
                                    shape = (total_inds, 2),
                                    strides = (t.strides[0], step * t.strides[0]))

    inds, counts = np.unique(t_strided, axis = 0, return_counts = True)

    P = np.zeros((n, n))
    N = np.zeros((n, n))
    P[inds[:, 0], inds[:, 1]] = counts
    N[inds[:, 0], inds[:, 1]] = counts

    sums = P.sum(axis = 1)

    P[sums != 0] = P[sums != 0] / sums[sums != 0][:, None]

    return P, N
```

5.2 Function to compute Stationary Vector

```
def compute_stationary_distribution(P):
    A = np.vstack((P.T - np.identity(P.shape[0]), np.ones((P.shape[0]))))
    b = np.zeros((P.shape[0] + 1, 1))
    b[-1] = 1
    return np.linalg.lstsq(A, b)[0]
```

5.3 Function to compute Auto Correlation

```
def compute_auto_correlation(x, k):
    x_bar = np.average(x)
    numerator, denominator = 0, 0
    m, M = x.index.min(), x.index.max()

    for i in range(m, M - k):
        numerator += ((x[i] - x_bar) * (x[i + k] - x_bar))

    for i in range(m, M):
        denominator += (x[i] - x_bar)**2
```

```
    return numerator / denominator
```

5.4 Function to compute test statistic and chi square value for each state

```
def compute_test_statistic(N, Q, i):
    # Q is P^2
    n, N2 = N.shape[0], N[i][Q[i] > 0]
    S = N2.sum()
    chi2 = sc.stats.chi2.ppf(q = 0.95, df = len(N2) - 1)
    ts = 0.0
    if S > 0:
        for j in range(n):
            if Q[i][j] > 0:
                observed = N[i][j]
                expected = S * Q[i][j]
                ts += ((observed - expected)**2 / expected)
    return chi2, ts
```