Northeastern University

Rakesh Kumar Rathod

Data Modeling Project - Markov Chain

MATH 7241 - Probability I

Supervisor: Prof. Christopher King

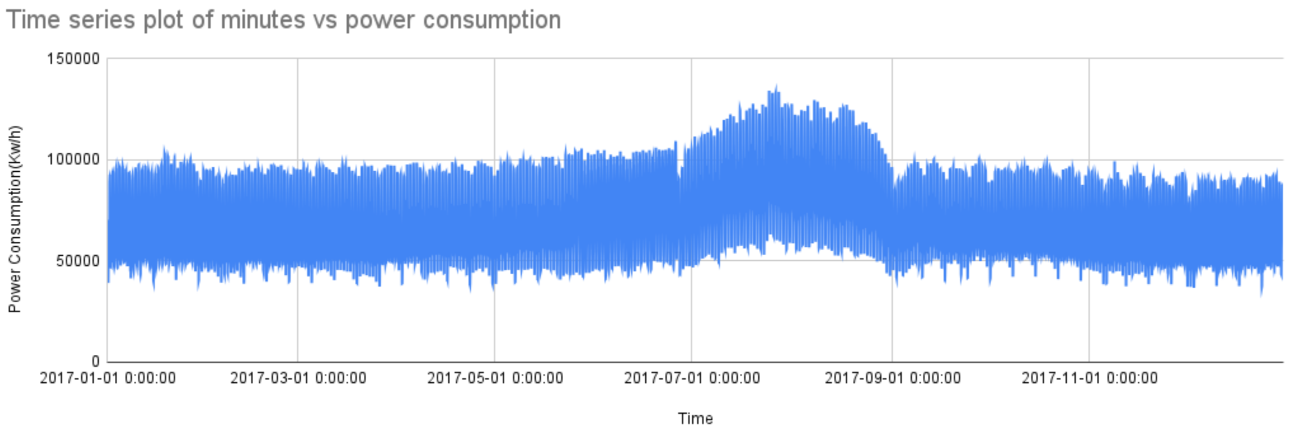
APPLIED MATHEMATICS, MS **2022**

1 Description of Data

I downloaded a compilation of power consumption of Tetouan city [UCI]. It contains the data of power consumption of three different distribution networks of Tetouan city which is located in North Morocco. The data contains 9 attributes such as date time, temperature, humidity, power consumption of zone 1, zone 2 and zone 3, diffuse flows etc. The dataset contains 52417 instances.

2 Cleaning of Data

I picked the aggregate power consumption column which is the sum of power consumptions in zone 1, zone 2 and zone 3 for the modeling. Hence my new data will contain two columns, date-time and total power consumption of the city where date-time is ranging from 1/1/2017 00:00 to 12/30/2017 23:59 in the intervals of 10 mins and power consumption ranges from 36785 to 134208 with a mean of 71222 and standard deviation 17143. The goal is to compare the total power consumption across the year and if it is dependent on the previous timestamp. The total entries are 52415. So, the time series plot of minutes vs total power consumption would look like the below graph:

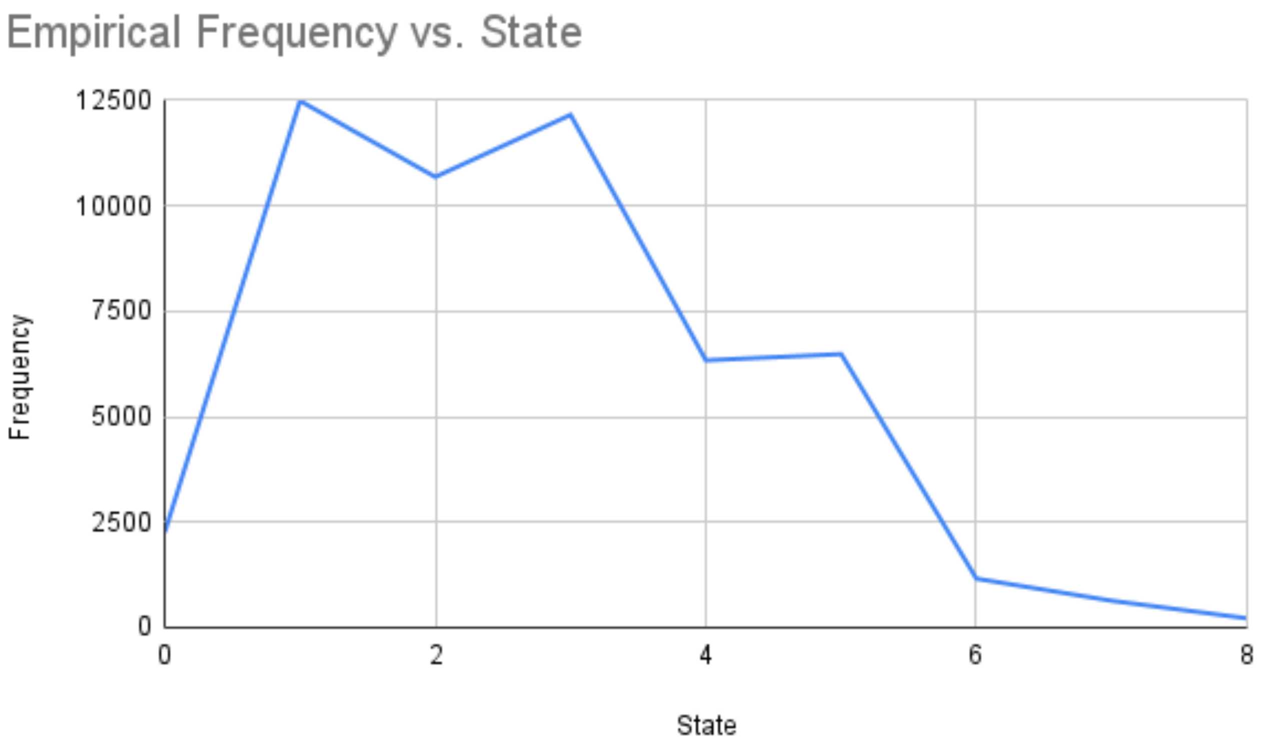


3 Data Analysis & Modeling

Based on the above graph, the spread of the most of data is confined to an upper and lower limit and have less outliers. Hence I divided the range of data into 9 equal intervals representing data in each of 9 states.

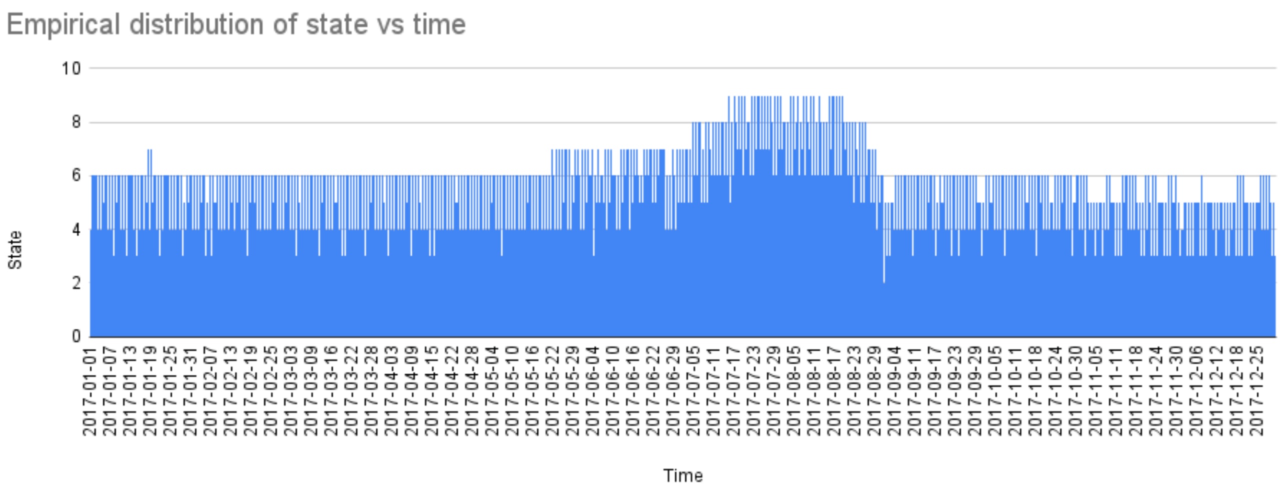
```
States = [0,  1,  2,  3,  4,  5,  6,  7,  8]
```

The total number of states in the Markov Chain are 9. This produces a left skewed curve with median lying at the third state. This looks like a fair assumption because if we closely observe the average run values it is less in the recent days. So the time series frequency plot of Markov State vs the 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 Frequency Plot would look like the below graph:



3.1 Empirical Distribution

The empirical distribution of time series is a plot of time vs Markov State, which would look like the below graph:



3.2 Transition Matrix

I wrote a function in order to compute the transition matrix for the time series which follows the above distribution. The transition matrix looks like following:

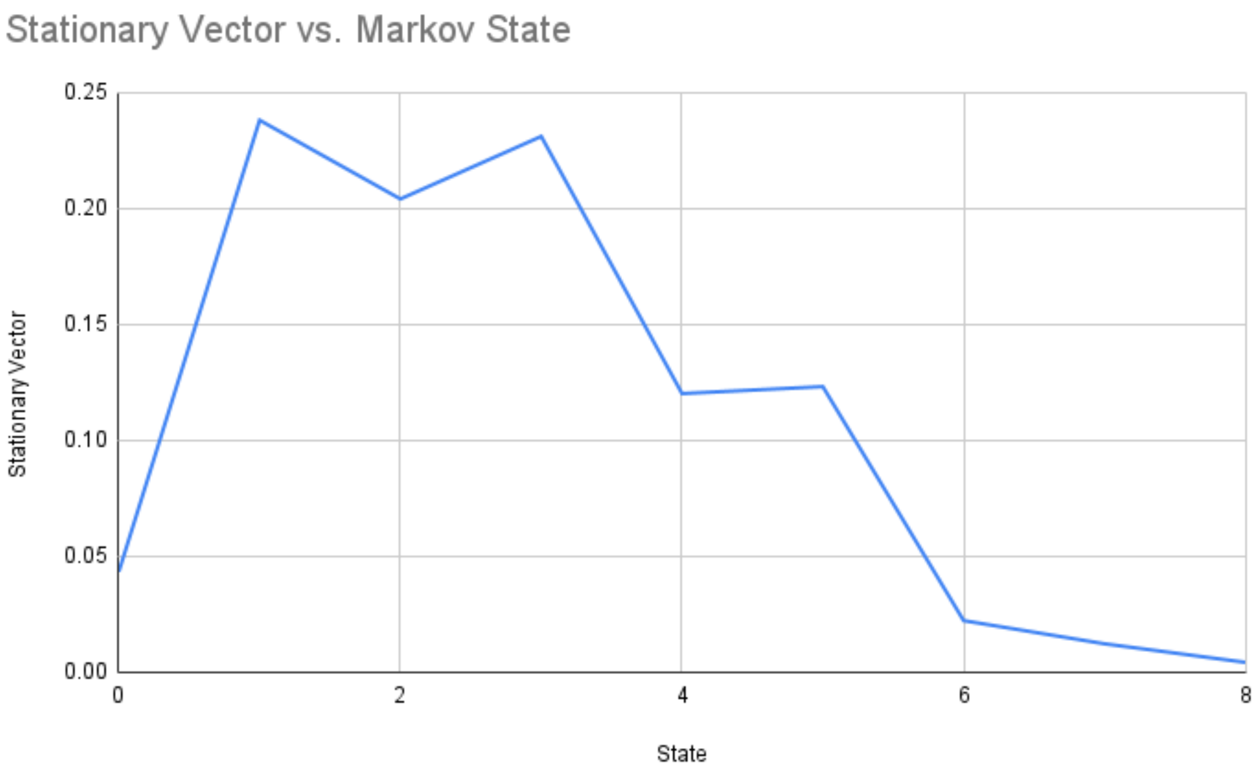
0.88	0.12	0.250	0.000	0.000	0.000	0.000	0.000	0.000443
0.021	0.949	0.03	0.033	0.000	0.000	0.000	0.000	0.000
0.000	0.035	0.913	0.052	0.000	0.000	0.000	0.000	0.000
0.000	0.00008	0.046	0.92	0.032	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.614	0.88	0.059	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.058	0.923	0.019	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.107	0.847	0.046	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.083	0.87	0.046
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.132	0.863

3.3 Stationary Distribution

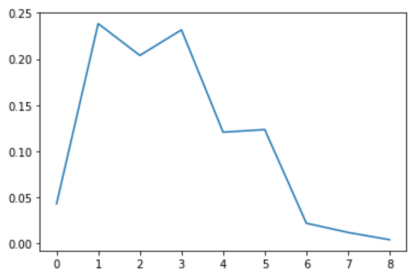
I wrote a function to compute the limiting probability vector (stationary vector) for the empirical distribution. The stationary vector looks like the following:

W = [0.043, 0.238, 0.204, 0.231, 0.120, 0.123, 0.022, 0.012, 0.004]

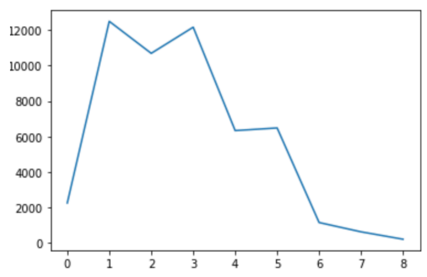
The Markov State vs stationary vector plot would look like the following graph:



This plot is clearly similar to the Markov State vs Frequency for original time series which is an indication that the frequency distribution of the original data was not disturbed.
For comparison, below are both the charts of stationary distribution and empirical distribution.



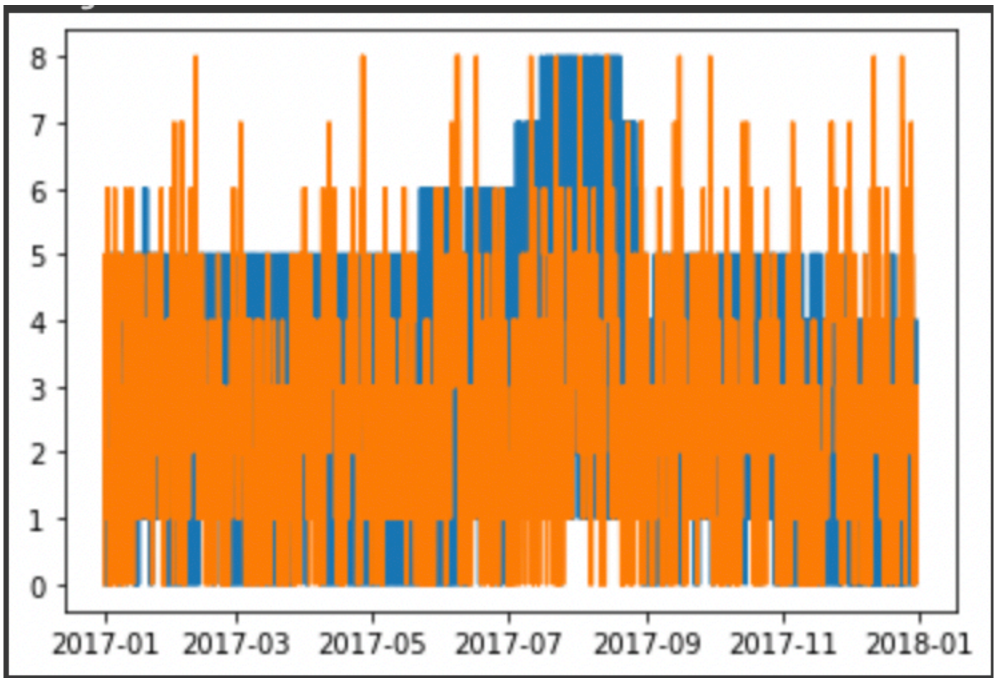
Stationary distribution



Empirical distribution

3.4 Simulated Time Series Generation

I generated a new time series from the stationary vector in the following way. I choose a random seed from state = 0 to state = 8 (my highest Markov state id, because total states = 9 and initial state id = 0). I generate my next state by following a probability distribution as obtained from the transition matrix computed above. For example, if my first random state is, state = 2, then I go to third row in the transition matrix, pick the probability vector (elements in that row) and generate my second state based on this probability vector. If I get my second state as, state = 4, then I go to fourth row in transition matrix, pick the probability vector and generate my third state based on this probability vector and so on ... till I get 52415 observations. This is my new time series. As a comparison, the original time series is compared with this simulated time series as a plot and it looks like the following graph:



3.5 Goodness of Fit Test

A Goodness of Fit Test for two step transition of original time series is conducted in order to see if our original time series obeys Markov Principle (current state is dependent only on previous state and not anything before that). The Pearson Test Statistic values are computed for each state and this value is compared with 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 percent significance level, using the function compute_test_statistic. This comparison yielded following results:

State	0	1	2	3	4	5	6	7	8
Observed Frequency	2259	12487	10682	12155	6335	6483	1159	635	219
Expected Frequency	2566	13745	11393	11874	5871	5380	880	527	178

From the above, I have calculated the test statistic value for each state and the chi2 value for corresponding degrees of freedom as shown in table below

State	Chi square value	Test Statistic Value	Decision
0	5.9914	0.4576	Accepted
1	5.9914	2.3696	Accepted
2	7.8147	8.5861	Rejected
3	9.4877	25.9734	Rejected
4	7.8147	7.8472	Rejected
5	7.8147	29.7419	Rejected
6	7.8147	6.6580	Accepted
7	5.9914	2.2613	Accepted
8	5.9914	1.0049	Accepted

4 Conclusions

This comparison shows that GoF test rejected the states 2, 3, 4, 5 and accepted states 0, 1, 6, 7, 8. This is confirmation that this time series does not obey the Markov Rule (current state is dependent only on previous state and not anything before that). So our initial assumption that the total power consumption is just dependent on the previous amount of power consumption is wrong. It could be true because in practice this is a multidimensional problem which can depend on several features and we tried to simplify the model with an assumption that it follows Markov Rule (current state is dependent only on previous state and not anything before that). In fact, the power consumption depend on many other factors such as season(In winter power consumption might be more than because of electric heaters) and many such factors. 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 Markov Modeling for this problem statement in the future.

5 Appendix

5.1 Function to compute transition matrix

```
def transition_matrix(transitions):
    n = 1+ max(transitions) #number of states

    M = [[0]*n for _ in range(n)]

    for (i,j) in zip(transitions,transitions[1:]):
        M[i][j] += 1

    #now convert to probabilities:
    for row in M:
        s = sum(row)
        if s > 0:
            row[:] = [f/s for f in row]
    return M
```

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])))) # print(A.shape)

    b = np.zeros((P.shape[0] + 1, 1))

    b[-1] = 1

    return np.linalg.lstsq(A, b)[0]
```

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

```
def compute_test_statistic(test_stat_data):
        x=[]
        chi = 0
        for i in range(test_stat_data.shape[0]):
          chi =  (((test_stat_data[i,1]-test_stat_data[i,0])**2)/(test_stat_data[i,0]))
          print(chi)
          x.append(chi)
        print(chi)
        print(chi2.ppf(q=0.95,df=8))
```

References

UCI Repo - https://archive.ics.uci.edu/ml/datasets/Power+consumption+of+Tetouan+city