21-344 Course Project: Discrete-Time Markov Chains

Juliette Wong, jnwong
Mathematical Sciences, Carnegie Mellon University, Pittsburgh, USA
May 14th, 2021

Abstract

A first-order Markov chain that is finite state and discrete-time follows the following condition: for the stochastic process $\{X_t : t \in \mathcal{T}\}$, where the time-space $\mathcal{T}$ is discrete and the state space is finite, $P(X_{n+1} = j|X_0 = i_0, X_1 = i_1, \cdots, X_n = i_n) = P(X_{n+1} = j|X_n = i_n)$. We can denote $P_{ij} = P(X_{n+1} = j|X_n = i_n)$ and create a transition matrix P, which has many properties specific to right stochastic matrices. To further explore Markov chains, the paper covers n-step transition probabilities and calculating the limiting distribution. To illustrate this concept further, an example of a calculating the transition matrix for Pittsburgh Weather is shown.

1 Introduction

A market research team is doing a study to determine preference between two toothpaste brands. Of those who bought Brand A in any month, 70% buy that same brand the next month, while 30% switch to Brand B. Of those using Brand B in one month, 80% continue to buy that same brand, while 20% switch to Brand A (Poole, 2015). Converting percentages into probabilities, we can visualize these findings in the figure below:

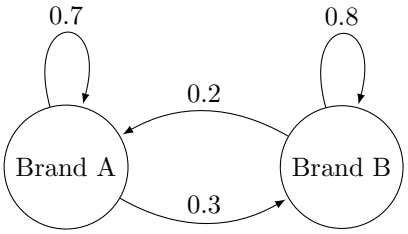


Figure 1: Transition Diagram, Toothpaste Example

The process above is defined as a finite-step Markov Chain. At a given point in time, the process is in a specific state. The state must be within the state space, denoted by S, and the time state is given by $\mathcal{T}$. Each Markov Chain follows the Markov process, which states that the prediction of future state only depends on the current state, and is independent of the states of the past.

While there are many variations of Markov Chains, this paper will cover first-order Markov Chains that are discrete-time and finite-state. By discrete-time, we mean that the time state $\mathcal{T}$ only contains discrete values, such as $\{0, 1, 2, \cdots\}$. By finite-state, we mean that there are only a finite-number of states in the state space S. And finally, by first-order, we mean that for the stochastic process $\{X_t : t \in \mathcal{T}\}$,

$$P(X_{n+1} = j|X_0 = i_0, X_1 = i_1, \cdots, X_n = i_n) = P(X_{n+1} = j|X_n = i_n)$$

The probability of the Markov chain going from one state to the next is called the transition probability. By assumption, the Markov chain is time homogeneous: transition probabilities at time n do not depend on n. We can use the transition probabilties to construct a transition matrix. In the transition matrix P, P_{ij} denotes the probability of the process transitioning from state i to state j, that is, for times n, $n+1$ in the time space, $P_{ij} = P(X_{n+1} = j|X_n = i)$. For our example above, the transition matrix is as follows:

$$P = \begin{array}{c} \\ A \\ B \end{array} \begin{array}{c} \begin{array}{cc} A & B \end{array} \\ \begin{pmatrix} 0.7 & 0.3 \\ 0.2 & 0.8 \end{pmatrix} \end{array}$$

2 Properties of Discrete-Time Markov Chains

There are many properties of the transition matrices for discrete-time Markov chains.

In the transition matrix P, since the probabilities are non-negative and the process must transition to another state (Ross, 2014) (including transitioning back to itself!), we have that for states $i, j \in S$,

$$P_{ij} \geq 0, \sum_{j \in S} P_{ij} = 1$$

.

Thus, a transition matrix can also be called a right stochastic matrix, as the transition matrix is a real square matrix where each entry is non-negative and each row sums to 1. Each row in the transition matrix is also known as a stochastic vector (*Stochastic Matrix*, n.d.).

Note that raising a transition matrix P to a positive integer power k will also yield a right stochastic matrix.

Proof. Let P be a transition matrix. Thus, $P_{ij} \geq 0$, and $\sum_{j \in S} P_{ij} = 1$ for all $i \in S$.

We wish to show that P^2 is a right stochastic matrix. For arbitrary $i, j \in S$,

$$\sum_{j \in S} P_{ij}^2 = \sum_{j \in S} \sum_{k \in S} P_{ik} P_{kj} = \sum_{k \in S} P_{ik} \sum_{j \in S} P_{kj} = 1 \cdot 1 = 1$$

In addition, $P_{ij}^2 \geq 0$ because it is formed by multiplying and adding values that are greater than or equal to 0.

Since i, j were chosen arbitrarily, the above argument can be made for any entry in P^2. Thus, we have shown that P^2 is a right stochastic matrix.

Note that the case of P^k, where $k \in \mathbb{Z}^+$, is proven via induction. □

Equivalent to $P_{ij} \geq 0, \sum_{j \in S} P_{ij} = 1, i = 0, 1, \cdots$, we can write that $P\vec{1} = \vec{1}$, where $\vec{1}$ is an S-dimensional vector containing all 1s. Thus, every Markov matrix has eigenvalue $\lambda = 1$. This is actually the largest eigenvalue in magnitude, so the spectral radius of a transition matrix is 1. (Offner, 2020) Indeed, if λ is an eigenvalue for for an transition matrix P, then $|\lambda| \leq 1$. A proof is shown below:

Proof. We prove by contradiction. Assume $|\lambda| > 1$.

Let $\vec{x} = (x_1, x_2, \cdots x_n)^T$ be a unit eigenvalue for P, with eigenvalue λ. By definition of a unit vector, $\sum_{i=1}^{n} x_i^2 = 1$, so $|x_i| \leq 1$.

Since P^k is also Markov, we have that each entry in P^k is between 0 and 1.

Thus, $\left|(P^k \vec{x})_i\right| = P_{i*}^k \cdot \vec{x} \leq n$, since this is a dot product of all numbers that are at most 1.

But we know that $P^k \vec{x} = \lambda^k \vec{k}$. (If λ is an eigenvalue of P, then λ^k is an eigenvalue of M^k.)

Since we assumed $|\lambda| > 1$, then $\lambda^k \vec{x}$ has entries that are greater than n for k sufficiently large.

This contradicts our earlier statement, so $|\lambda| \leq 1$. □

3 Application of Discrete-Time Markov Chains

Discrete-Time Markov Chains serve as a way to represent many phenomena. They are used in a variety of fields, including, but not limited to, Natural Language Processing, population dynamics, information theory, finance, economics, physics, and chemistry (*Stochastic Matrix*, n.d.). Google's Page Rank algorithm, as well as Reddit's Subreddit Simulator, are based on Markov chain processes as well. (Jaiswal, 2019)

An example of how one can apply discrete-time Markov Chains is through weather. Suppose that tomorrow's forecast (rainy, cloudy, etc.) tomorrow depends on on the current weather conditions, and not the past weather conditions. To illustrate this idea better, I have gathered data on Pittsburgh Weather from May 2019-May 2021. Part of the data is shown below:

Date	Weather	High	Low	Precipitation (in)
2019-05-10	Cloudy	58	40	0.0
2019-05-11	Partly Cloudy	55	35	0.0
2019-05-12	Mostly Sunny	59	35	0.0
2019-05-13	Mostly Sunny	63	42	0.0
2019-05-14	Mostly Cloudy	64	43	0.0

Figure 2: Weather Data Table, First 6 Days

This data was obtained via Weather Underground, a commercial weather service under The Weather Company (IBM). In total, 731 days of weather was used, ranging from May 10th, 2019 to May 9th, 2021. While the dates themselves aren't too significant, 2 years of data was recorded to improve the accuracy of the Markov chain. In addition, whole years were used to account for seasonality in the data. (Interestingly enough, May 10th, 2019 was the day of my last final during my sophomore year, and May 9th, 2021 lies on the weekend before my last finals week as an undergraduate student.) Due to bugs on the website, there were 39 days with missing data. I recovered this data by using a second source, the Weather Channel (which is also now by IBM).

In the data, there were 8 states: Mostly Sunny (), Partly Cloudy (), Mostly Cloudy (), Cloudy (), Foggy (), Scattered Showers (), Thunderstorm (), and Snow (). We can use these as our state space, and each day between May 10th, 2019 and May 9th, 2021, to estimate the transition probabilities.

To find the transition matrix for the first-order Markov chain, one must first ensure that the first-order condition holds. To show that the first-order condition holds, one can calculate estimates of the transition matrix by looking at the chain of states and using the approximation: $\hat{P}_{ij} = \frac{\text{number of "ij" that occurs}}{\text{number of "i"s occurs}}$. Asusming the first-order model is correct, the expected counts in the second-order chain are $n_{ij} \cdot \hat{P}_{ij}$, where n_{ij} is the sum of the "ij"th row in the actual second-order chain. (Jin, 2021)

One can compare the expected counts in the second-order chain with the actual counts in the second-order chain with a normal probability plot. To do so, for each observed count O_i and corresponding expected count E_i in our second-order matrix of coutns, we normalize the comparison between the two values using $\frac{O_i - E_i}{\sqrt{O_i}}$. (Jin, 2021) Then, we plot those values in the normal probability plot, which plots the observed (normalized) values against theoretical quantiles. The closer the points in the normal probability plot are to the line y=x, the better the approximation. While the normal probability plot, shown below, is not perfect, it is a much better approximation than the independence model (see code).

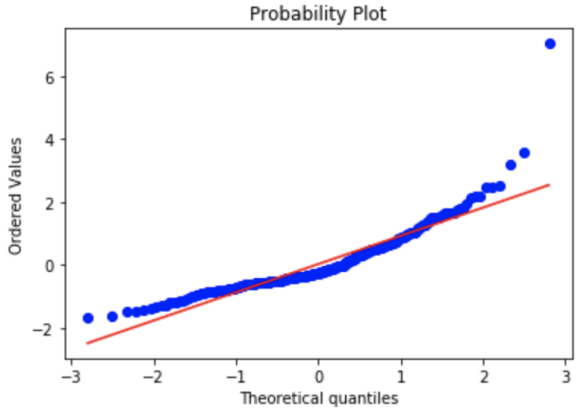


Figure 3: Normal Probability Plot to Ensure First-Order Condition

Our estimate of the transition matrix is shown on the next page (rounded to the nearest 4 decimal places)

$$P = \begin{pmatrix} 0.3204 & 0.1553 & 0.1748 & 0.3301 & 0 & 0 & 0.0097 & 0.0097 \\ 0.1831 & 0.1408 & 0.2113 & 0.3521 & 0.0141 & 0.0986 & 0 & 0 \\ 0.1513 & 0.1513 & 0.2773 & 0.3277 & 0 & 0.0504 & 0.0252 & 0.0168 \\ 0.0714 & 0.0652 & 0.1273 & 0.5373 & 0.0155 & 0.1056 & 0 & 0.0776 \\ 0.25 & 0 & 0.25 & 0.375 & 0.125 & 0 & 0 & 0 \\ 0.1481 & 0.0741 & 0.0926 & 0.4444 & 0.0185 & 0.1481 & 0 & 0.0741 \\ 0.5 & 0 & 0.25 & 0.25 & 0 & 0 & 0 & 0 \\ 0.0816 & 0.0408 & 0.0816 & 0.449 & 0 & 0 & 0 & 0.3469 \end{pmatrix}$$

This transition matrix could be used to answer many questions. As an example, imagine today that it is cloudy (as it is often in Pittsburgh, which we shall show in the next section). The probability that it will be cloudy again tomorrow is 0.5373, while the probability that it will be mostly sunny tomorrow is 0.0714. Ignoring weather forecasts, there will be a 10.56% chance (0.1056 in decimal terms) that there will be scattered showers the next day, given that today is cloudy, so one should bring an umbrella just in case.

4 N-Step Transition Probabilities

Next, we define the n-step transition probabilities P_{ij}^n to be the probability that a process in state i will be at state j in n additional transtions. Thus, $P_{ij}^n = P(X_{n+k} = j | X_k = i)$ for some $k, n+k \in \mathcal{T}$ and $i, j \in S$.

To compute the n-step transition probabilities, we one can use the Chapman-Kolmogorov Equations. The derivation of the Chapman-Kolmogorov equations is shown below:

$$\begin{aligned} P_{ij}^{n+m} &= P(X_{n+m} = j | X_0 = i) \\ &= \sum_{k \in S} P(X_{n+m} = j | X_n = k, X_0 = i) \\ &= \sum_{k \in S} P(X_{n+m} = j | X_n = k, X_0 = i) P(X_n = k | X_0 = i) \\ &= \sum_{k \in S} P(X_{n+m} = j | X_n = k) P(X_n = k | X_0 = i) \\ &= \sum_{k \in S} P_{kj}^m P_{ik}^n \end{aligned}$$

Let $P^{(n)}$ denote the matrix of n-step transition probabilities P_{ij}^n. Then we have that

$$P^{(m+n)} = P^{(m)} \cdot P^{(n)} = P^m \cdot P^n$$

We can use this to show that $P^{(2)} = P^{(2-1+1)} = P^{(1)} \cdot P^{(1)} = P \cdot P = P^2$. Using induction, we can further show that $P^{(n)} = P^{(n-1+1)} = P^{(n-1)} \cdot P^{(1)} = P^n$. Thus, the n-step transition matrix can be obtained by multiplying P by itself n times (Ross, 2014).

Now, we can apply this to our weather transition matrix. Suppose it is mostly sunny on Monday, and we wanted to know the probability there will be snow on Wednesday (2 days from Monday). Using P^2, we find that the probability is 0.0350 (rounded to the nearest 4 decimal places).

$$P^2 = \begin{pmatrix} 0.1868 & 0.12 & 0.1825 & 0.4019 & 0.0073 & 0.059 & 0.0075 & 0.035 \\ 0.1597 & 0.1105 & 0.1778 & 0.4176 & 0.011 & 0.0763 & 0.0071 & 0.04 \\ 0.163 & 0.1125 & 0.1894 & 0.4064 & 0.0082 & 0.071 & 0.0085 & 0.0411 \\ 0.1183 & 0.0856 & 0.15 & 0.4645 & 0.0132 & 0.0852 & 0.0039 & 0.0793 \\ 0.1759 & 0.1011 & 0.192 & 0.4128 & 0.0214 & 0.0522 & 0.0087 & 0.0357 \\ 0.1394 & 0.0904 & 0.1482 & 0.4502 & 0.013 & 0.0808 & 0.0038 & 0.0742 \\ 0.2159 & 0.1318 & 0.1885 & 0.3813 & 0.0039 & 0.039 & 0.0112 & 0.0285 \\ 0.1064 & 0.0742 & 0.131 & 0.4651 & 0.0075 & 0.0555 & 0.0029 & 0.1574 \end{pmatrix}$$

Similarly, suppose it has scattered showers on Saturday, so you had to cancel your plans and reschedule them to the next week. To find the probability that next Saturday you would not have to cancel your plans yet again, you can use P^7, shown below (rounded to the nearest 4 decimal places). Starting from the current state of scattered showers, take the sum of all of the probabilities in where there is no precipitation: $0.1416+0.0975+0.1632+0.4392+0.0109 = 0.8524$ (Alternatively, you could do $1-0.0754-0.0055-0.0667$ to reach the same result.)

$$P^7 = \begin{pmatrix} 0.1415 & 0.0974 & 0.1632 & 0.4393 & 0.011 & 0.0754 & 0.0055 & 0.0667 \\ 0.1414 & 0.0974 & 0.1631 & 0.4394 & 0.011 & 0.0754 & 0.0055 & 0.0668 \\ 0.1414 & 0.0974 & 0.1631 & 0.4394 & 0.011 & 0.0754 & 0.0055 & 0.0668 \\ 0.1411 & 0.0972 & 0.1629 & 0.4397 & 0.011 & 0.0754 & 0.0055 & 0.0673 \\ 0.1415 & 0.0974 & 0.1632 & 0.4393 & 0.011 & 0.0754 & 0.0055 & 0.0667 \\ 0.1412 & 0.0972 & 0.1629 & 0.4396 & 0.011 & 0.0754 & 0.0055 & 0.0672 \\ 0.1416 & 0.0975 & 0.1632 & 0.4392 & 0.0109 & 0.0754 & 0.0055 & .0667 \\ 0.1408 & 0.097 & 0.1627 & 0.4399 & 0.011 & 0.0754 & 0.0055 & 0.0677 \end{pmatrix}$$

5 Limiting Distributions

After seeing what the n-step transition probabilities, perhaps we are interested in knowing the long-term behavior of Markov Chains. This section will discuss limiting distributions (also known as stationary or steady-state distributions), which shows the long-run proportion of time that the Markov Chain spends in each state.

We denote $\pi^{(n)} = \begin{pmatrix} P(X_n = 0) & P(X_n = 1) & \cdots & P(X_n = s) \end{pmatrix}$, that is, $\pi^{(n)}$ is the vector showing the probability the Markov Chain is at each state for a time step n. We wish to study this distribution as $n \to \infty$.

The probability distribution $\pi = \begin{pmatrix} \pi_0 & \pi_1 & \cdots & \pi_s \end{pmatrix}$ is called the limiting distribution of a Markov Chain X_n if

$$\pi_j = \lim_{n\to\infty} P(X_n = j | X_0 = i)$$

for all $i, j \in S$. By the above definition, the limiting distribution does not depend on the initial state, so we can further write $\pi_j = \lim_{n\to\infty} P(X_n = j)$ for all $j \in S$. (Pishro-Nik, 2014)

To find the limiting distribution, we know that $\pi = \lim_{n\to\infty} \pi^{(n)} = \lim_{n\to\infty} [\pi^{(0)} P^n]$. Similarly, we can write:

$$
\begin{aligned}
\pi &= \lim_{n\to\infty} \pi^{(n+1)} \\
&= \lim_{n\to\infty} [\pi^{(0)} P^{n+1}] \\
&= \lim_{n\to\infty} [\pi^{(0)} P^{n} \cdot P] \\
&= \lim_{n\to\infty} [\pi^{(0)} P^{n}] P \\
&= \pi P
\end{aligned}
$$

Intuitively, suppose that X_n has distribution π. Then, πP is the distribution of X_{n+1}. If $\pi = \pi P$, then X_n and X_{n+1} have the same distribution, meaning that the chain converges to a limiting distribution (Pishro-Nik, 2014).

Thus, we can use the system of equations formed by $\pi = \pi P$ to solve for the limiting distribution π. We can use this to find the limiting distribution of the transition matrix for the toothpaste example in our introduction:

$$
P = \begin{array}{c} \\ A \\ B \end{array} \begin{array}{c} \begin{array}{cc} A & B \end{array} \\ \begin{pmatrix} 0.7 & 0.3 \\ 0.2 & 0.8 \end{pmatrix} \end{array}
$$

Using $\pi = \pi P$, we have the following:

$$
\begin{pmatrix} \pi_0 & \pi_1 \end{pmatrix} = \begin{pmatrix} \pi_0 & \pi_1 \end{pmatrix} \begin{bmatrix} 0.7 & 0.3 \\ 0.2 & 0.8 \end{bmatrix}
$$

Thus, we have the system of equations:

$$
\begin{cases} \pi_0 = 0.7\pi_0 + 0.2\pi_1 \\ \pi_1 = 0.3\pi_0 + 0.8\pi_1 \end{cases}
$$

Solving the system of equations, one can find the limiting distribution is $\pi = \begin{pmatrix} \frac{2}{5} & \frac{3}{5} \end{pmatrix}$. Thus, the long-run proportion of individuals who use Brand A and Brand B are $\frac{2}{5}$ and $\frac{3}{5}$, respectively.

Another way one can find the limiting distribution is by looking at the transition matrix raised to a very high power, such as 1000. This serves as an approximation to P^{∞}. If the matrix P^{1000} has identical rows, then the values in that row is the limiting distribution. Essentially, regardless of the initial state, these will be the long-run proportions that the matrix will be in the given state. If we go back to P^{7} for our weather matrix and compare it to both P and P^{2}, we can see that the values in the rows are starting to "converge", such that the rows begin to look identical. Indeed, we can calculate P^{1000} as an approximation for P^{∞} for the weather transition matrix, shown below (rounded to the nearest 5 decimal places):

$$
P^{1000} = \begin{pmatrix}
0.14124 & 0.09728 & 0.16297 & 0.43957 & 0.01096 & 0.07539 & 0.00548 & 0.0671 \\
0.14124 & 0.09728 & 0.16297 & 0.43957 & 0.01096 & 0.07539 & 0.00548 & 0.0671 \\
0.14124 & 0.09728 & 0.16297 & 0.43957 & 0.01096 & 0.07539 & 0.00548 & 0.0671 \\
0.14124 & 0.09728 & 0.16297 & 0.43957 & 0.01096 & 0.07539 & 0.00548 & 0.0671 \\
0.14124 & 0.09728 & 0.16297 & 0.43957 & 0.01096 & 0.07539 & 0.00548 & 0.0671 \\
0.14124 & 0.09728 & 0.16297 & 0.43957 & 0.01096 & 0.07539 & 0.00548 & 0.0671 \\
0.14124 & 0.09728 & 0.16297 & 0.43957 & 0.01096 & 0.07539 & 0.00548 & 0.0671 \\
0.14124 & 0.09728 & 0.16297 & 0.43957 & 0.01096 & 0.07539 & 0.00548 & 0.0671
\end{pmatrix}
$$

We can see that P^{1000} has identical rows! Thus, the limiting distribution for this problem is:

$$
\pi = \begin{pmatrix} 0.14124 & 0.09728 & 0.16297 & 0.43957 & 0.01096 & 0.07539 & 0.00548 & 0.0671 \end{pmatrix}
$$

Interpreting this with the problem, we can see that the long-run proportion of time that Pittsburgh is mostly sunny is about 14.124%. Of all of the different weather states, Pittsburgh spends the most time being cloudy, as the long-run proportion of time that Pittsburgh is cloudy is approximately 43.957%.

Note that not all transition matrices will have a limiting distribution. While this is beyond the scope of the paper, transition matrices will have a limiting distribution if it is irreducible (only has one recurrent class) and aperiodic (all states have a period of 1). For further reading on this material, refer to Chapter 11.2.4 of Introduction to Probability, Statistics, and Random Processes (Pishro-Nik, 2014).

6 Summary

A first-order Markov chain that is finite state and discrete-time follows the following condition: for the stochastic process $\{X_t : t \in \mathcal{T}\}$, where the time-space $\mathcal{T}$ is discrete and the state space is finite, $P(X_{n+1} = j|X_0 = i_0, X_1 = i_1, \cdots, X_n = i_n) = P(X_{n+1} = j|X_n = i_n)$. We can denote $P_{ij} = P(X_{n+1} = j|X_n = i_n)$. Since by assumption the Markov chain is time homogenous, we can denote P_{ij} to be the transition probability going from state i to state j (regardless of what n is). One can create a transition matrix P by finding the transition probabilities for all states in the state space.

In the transition matrix P, since the probabilities are non-negative and the process must transition to another state (Ross, 2014) (including transitioning back to itself!), we have that for states $i, j \in S$, $P_{ij} \geq 0, \sum_{j \in S} P_{ij} = 1$. In addition, every transition matrix has the eigenvalue $\lambda = 1$. 1 is actually the largest eigenvalue in magnitude for the transition matrix: if λ is an eigenvalue for a transition matrix P, then $|\lambda| \leq 1$.

N-step transition probabilities are defined as $P_{ij}^n = P(X_{n+k} = j|X_k = i)$. These can be calculated via the Chapman-Kolmogorov equations, which in short says that to find the n-step transition matrix, one can just multiply the transition matrix by itself n times.

Finally, the probability distribution $\pi = \begin{pmatrix} \pi_0 & \pi_1 & \cdots & \pi_s \end{pmatrix}$ is called the limiting distribution of a Markov Chain X_n if

$$\pi_j = \lim_{n \to \infty} P(X_n = j|X_0 = i)$$

for all $i, j \in S$. By the above definition, the limiting distribution does not depend on the initial state, so we can rewrite $\pi_j = \lim_{n \to \infty} P(X_n = j)$ for all $j \in S$. If π exists, there are multiple ways to calculate it. One way is to look at an approximation of P^∞ to see if the rows are identical: if they are, then that row is the limiting distribution. Another way is to solve the equation $\pi P = \pi$.

References

Jaiswal, S. (2019). *Markov chains in python: Beginner tutorial.* Retrieved from https://www.datacamp.com/community/tutorials/markov-chains-python-tutorial

Jin, J. (2021). *36-410 lecture notes.* Carnegie Mellon University.

Offner, D. (2020, July). *21-241 lecture 14 notes: Markov matrices.* Carnegie Mellon University.

Pishro-Nik, H. (2014). *Introduction to probability, statistics, and random processes.* Kappa Research LLC. Retrieved from https://www.probabilitycourse.com/chapter11/11_2_6_stationary_and_limiting_distributions.php

Poole, D. (2015). *Linear algebra: A modern introduction, fourth edition.* 200 First Stamford Place, 4th Floor Stamford, CT 06902, USA: Cengage Learning.

Ross, S. M. (2014). *Introduction to probability models, eleventh edition.* 225 Wyman Street, Waltham, MA 02451, USA: Academic Press.

Stochastic matrix. (n.d.). Wikipedia. Retrieved from https://en.wikipedia.org/wiki/Stochastic_matrix

Supplementary Files

I have created a Github repository for this project (Access here). The Github repository contains an Excel file, titled "Weather_PGH.xlsx," that contains the dataset I used for my project. The Jupyer Notebook, "21-344_Final_Project_Code.ipynb" is where I did my calculations. It is also shown below in PDF form.

21-344 Final Project Code

May 14, 2021

[1]:
```
import numpy as np
import pandas as pd
import math
import matplotlib.pyplot as plt
import scipy.stats
```

1 Importing data

[2]:
```
df = pd.read_excel("Weather_PGH.xlsx")
df.head() # to see what the dataset looks like
```

[2]:

	Date	Weather	High	Low	Precipitation (in)	Wunderground?
0	2019-05-10	Cloudy	58	40	0.0	No
1	2019-05-11	Partly Cloudy	55	35	0.0	No
2	2019-05-12	Mostly Sunny	59	35	0.0	No
3	2019-05-13	Mostly Sunny	63	42	0.0	No
4	2019-05-14	Mostly Cloudy	64	43	0.0	No

[3]:
```
df.shape # 731 days of data were used
```

[3]: (731, 6)

[4]:
```
# Not all the data could be obtained from Wunderground due to a bug in their␣
↪system
# The ones that are labeled as "No" were obtained from the Weather Channel
df.groupby("Wunderground?").count()
```

[4]:

Wunderground?	Date	Weather	High	Low	Precipitation (in)
No	39	39	39	39	39
Yes	692	692	692	692	692

2 Converting Weather to an Integer Value

To index into the matrix, I converted the weather states for each day into a non-negative integer value

[5]:
```
def weather_number(weather_name):
    num = -1
    if weather_name == "Mostly Sunny":
        num = 0
    elif weather_name == "Partly Cloudy":
        num = 1
    elif weather_name == "Mostly Cloudy":
        num = 2
    elif weather_name == "Cloudy":
        num = 3
    elif weather_name == "Foggy":
        num = 4
    elif weather_name == "Scattered Showers":
        num = 5
    elif weather_name == "Thunderstorm":
        num = 6
    elif weather_name == "Snow":
        num = 7
    return num
```

[6]:
```
df["weather_num"] = df["Weather"].apply(weather_number)
df.head()
```

[6]:

	Date	Weather	High	Low	Precipitation (in)	Wunderground?	\
0	2019-05-10	Cloudy	58	40	0.0	No	
1	2019-05-11	Partly Cloudy	55	35	0.0	No	
2	2019-05-12	Mostly Sunny	59	35	0.0	No	
3	2019-05-13	Mostly Sunny	63	42	0.0	No	
4	2019-05-14	Mostly Cloudy	64	43	0.0	No	

	weather_num
0	3
1	1
2	0
3	0
4	2

3 Creating 1-Step Transition Matrix P

[7]:
```
# General function to create a transition matrix P, assuming first-order␣
 ↪condition is met
# Input: a series, also known as a stochastic process {X(t)}
# Outputs: count matrix (used to prove first-order condition is met),␣
 ↪transition_matrix (P)
def create_transition_matrix(series):
```

```
    # Initialize to make
    unique_vals = len(np.unique(series))
    counts = np.zeros((unique_vals, unique_vals))
    n = len(series)

    # increment count when transition occures
    for i in range(n-1):
        current = series[i]
        future = series[i+1]
        counts[current][future] += 1

    # Convert counts to probabilities
    transition_matrix = counts/counts.sum(axis=1)[:,None]
    return counts, transition_matrix
```

[8]:
```
count_mat, mat = create_transition_matrix(df["weather_num"])
np.around(mat, decimals=4)
# For the final presentation and report, I rounded to the nearest 4␣
 ↪decimal-places for formatting purposes.
```

[8]:
```
array([[0.3204, 0.1553, 0.1748, 0.3301, 0.    , 0.    , 0.0097, 0.0097],
       [0.1831, 0.1408, 0.2113, 0.3521, 0.0141, 0.0986, 0.    , 0.    ],
       [0.1513, 0.1513, 0.2773, 0.3277, 0.    , 0.0504, 0.0252, 0.0168],
       [0.0714, 0.0652, 0.1273, 0.5373, 0.0155, 0.1056, 0.    , 0.0776],
       [0.25  , 0.    , 0.25  , 0.375 , 0.125 , 0.    , 0.    , 0.    ],
       [0.1481, 0.0741, 0.0926, 0.4444, 0.0185, 0.1481, 0.    , 0.0741],
       [0.5   , 0.    , 0.25  , 0.25  , 0.    , 0.    , 0.    , 0.    ],
       [0.0816, 0.0408, 0.0816, 0.449 , 0.    , 0.    , 0.    , 0.3469]])
```

4 Showing why Markov Chains work

4.1 Independence Model does not Hold

For the independence model, one would assume that $\{X_0, X_1, ...\}$ are all independent. The best estimates for the probability that the chain is in state J is the total number of occurrences where the matrix is in state j divided by the total number of events, or $\hat{P}_i = \frac{\text{number i's}}{731}$

[9]:
```
weather_counts = df.groupby("weather_num").count()["Date"]
weather_prop = weather_counts / 731 # calculates approximation of P_i
```

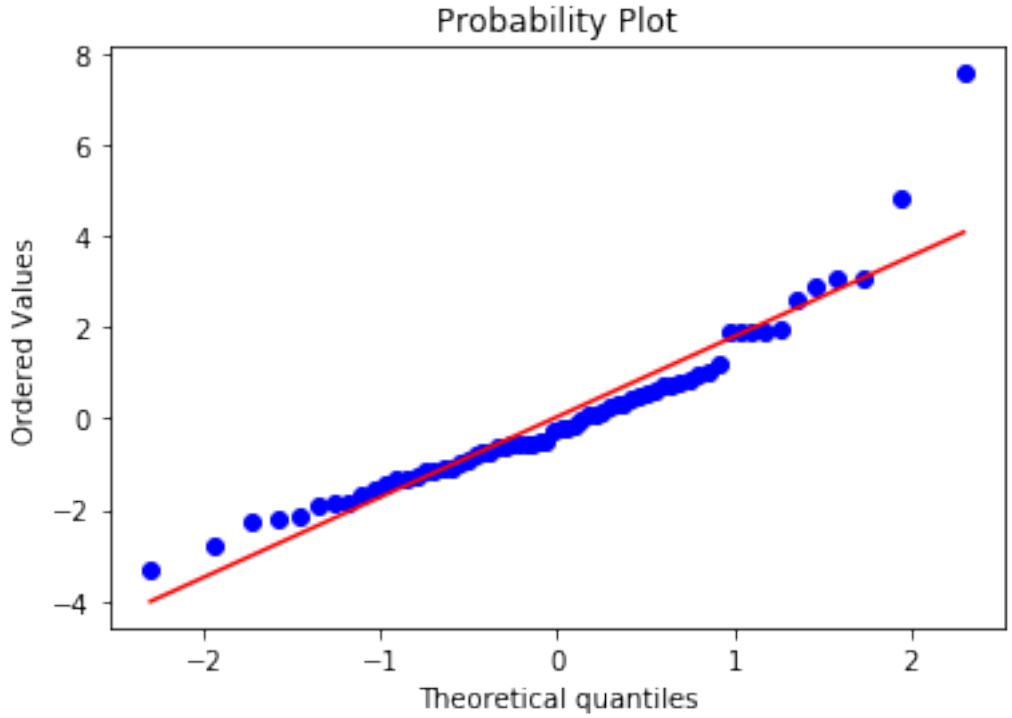
One would use the formula $N \times \hat{P}_i \times \hat{P}_j$ to calculate the expected number of occurences there will be "ij" in the chain, assuming the independence model works.

[10]:
```
expected = np.zeros((8, 8))
for i in range(8):
    for j in range(8):
```

```
        expected[i][j] = weather_prop[i]*weather_prop[j]*731 # N = 731 in our␣
↪example
```

Use a probability plot to assess the fit of the indepdence model. We see that the ordered values don't fit the model too well, with some bigger deviation in the tails of the normal probability plot

[11]:
```
normalized = []
for i in range(8):
    for j in range(8):
        if expected[i][j] == 0: continue # skip, since we don't want 0 in the␣
↪denominator
        normalized.append((count_mat[i][j] - expected[i][j]) / math.
↪sqrt(expected[i][j])) # normalize
scipy.stats.probplot(normalized, plot = plt) # normal probability plot
plt.show()
```



4.2 1st-Order Markov Chain

[12]:
```
def create_second_order(series):
    counts = np.zeros((64, 8))
    n = len(series)
    for i in range(n-2):
        current = series[i]
```

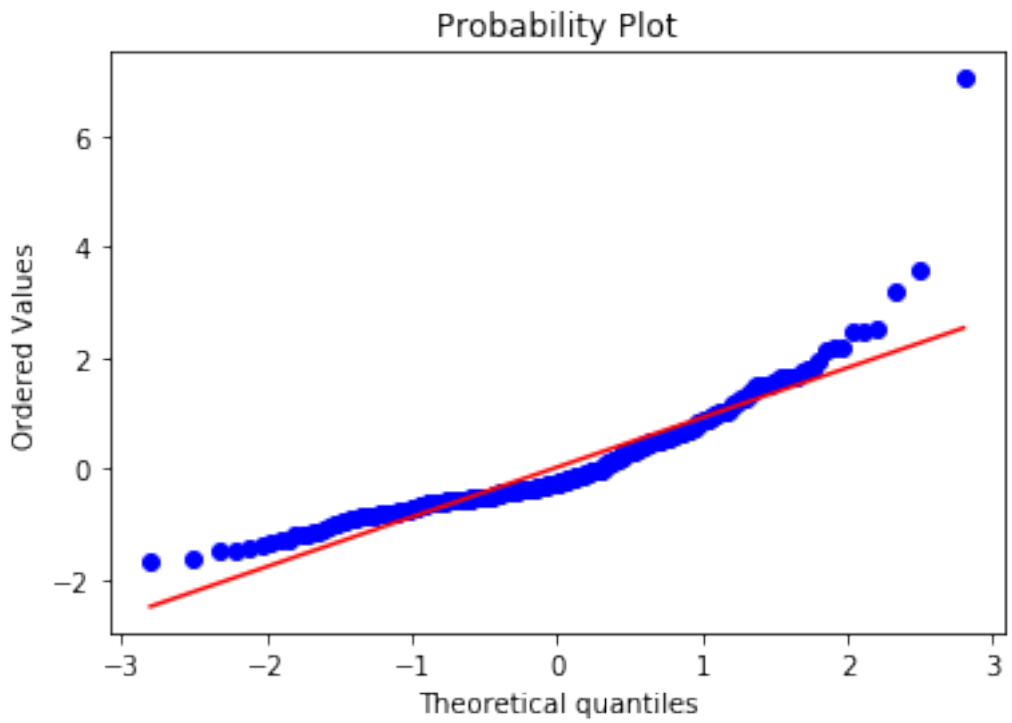
```
        next_val = series[i+1]
        future_val = series[i+2]
        counts[8*current + next_val][future_val] += 1
    return counts
second_order = create_second_order(df["weather_num"])
```

[13]:
```
sec_order_sum = np.sum(second_order, axis = 1)
expected_sec_order = np.zeros((64, 8))

for i in range(64):
    for j in range(8):
        expected_sec_order[i][j] = mat[i % 8][j] * sec_order_sum[i]
```

[14]:
```
# normalize the deviation of the expected from the actual
normalized = []
for i in range(64):
    for j in range(8):
        if expected_sec_order[i][j] == 0: continue  # skip, since we don't want␣
↪0 in the denominator
        normalized.append((second_order[i][j] - expected_sec_order[i][j]) /␣
↪math.sqrt(expected_sec_order[i][j]))
```

[15]:
```
scipy.stats.probplot(normalized, plot = plt)
plt.show()
```



Looking at the normal probability plot, we see that there is an improvement than when an independence model was assumed. We thus choose to use the first-order condition.

5 Creating Multi-Step Transition Matrix

$P^{(n)} = P^n$, by Chapman-Kolmogorov Equations

[16]:
```
np.around(np.linalg.matrix_power(mat, 2), 4)
```

[16]:
```
array([[0.1868, 0.12  , 0.1825, 0.4019, 0.0073, 0.059 , 0.0075, 0.035 ],
       [0.1597, 0.1105, 0.1778, 0.4176, 0.011 , 0.0763, 0.0071, 0.04  ],
       [0.163 , 0.1125, 0.1894, 0.4064, 0.0082, 0.071 , 0.0085, 0.0411],
       [0.1183, 0.0856, 0.15  , 0.4645, 0.0132, 0.0852, 0.0039, 0.0793],
       [0.1759, 0.1011, 0.192 , 0.4128, 0.0214, 0.0522, 0.0087, 0.0357],
       [0.1394, 0.0904, 0.1482, 0.4502, 0.013 , 0.0808, 0.0038, 0.0742],
       [0.2159, 0.1318, 0.1885, 0.3813, 0.0039, 0.039 , 0.0112, 0.0285],
       [0.1064, 0.0742, 0.131 , 0.4651, 0.0075, 0.0555, 0.0029, 0.1574]])
```

[17]:
```
np.around(np.linalg.matrix_power(mat, 7), 4)
```

[17]:
```
array([[0.1415, 0.0974, 0.1632, 0.4393, 0.011 , 0.0754, 0.0055, 0.0667],
       [0.1414, 0.0974, 0.1631, 0.4394, 0.011 , 0.0754, 0.0055, 0.0669],
       [0.1414, 0.0974, 0.1631, 0.4394, 0.011 , 0.0754, 0.0055, 0.0668],
       [0.1411, 0.0972, 0.1629, 0.4397, 0.011 , 0.0754, 0.0055, 0.0673],
       [0.1415, 0.0974, 0.1632, 0.4393, 0.011 , 0.0754, 0.0055, 0.0668],
       [0.1412, 0.0972, 0.1629, 0.4396, 0.011 , 0.0754, 0.0055, 0.0672],
       [0.1416, 0.0975, 0.1632, 0.4392, 0.0109, 0.0754, 0.0055, 0.0666],
       [0.1408, 0.097 , 0.1627, 0.4399, 0.011 , 0.0754, 0.0055, 0.0677]])
```

6 Limiting Distribution

An approximation to P^{∞} is P^{1000}. We can see that there are identical rows, so that is the limiting distribution

[18]:
```
P_thousand = np.linalg.matrix_power(mat, 1000)
```

[19]:
```
pi = P_thousand[0]
np.around(pi, 5)
```

[19]:
```
array([0.14124, 0.09728, 0.16297, 0.43957, 0.01096, 0.07539, 0.00548,
       0.0671 ])
```