International Journal of Science and Research (IJSR)
ISSN: 2319-7064
Index Copernicus Value (2016): 79.57 | Impact Factor (2017): 7.296

Markov Chain Model for Time Series and its Application to Forecasting Stock Market Prices

Joel Cheruiyot Chelule[1], Rodgers Otieno[2], Ayubu Anapapa[3]

[1,2]Jomo Kenyatta University of Agriculture and Technology, Department of Statistics and Actuarial Science, Nairobi, Kenya
[3]University of Eldoret, Department of Mathematics and Computer Science, Eldoret, Kenya

Abstract: ***A Markov chain is a discrete-valued Markov process; discrete-valued means that the state space of possible values of the Markov chain is finite or countable. This paper seeks to forecast stock market prices using Markov Chain Model (MCM). A discrete state space is defined for an MCM which is used to calculate fitting probability matrices. Time series data of day closing prices of KenGen Company as listed in the Nairobi Stock Exchange for the period 4th January, 2016 to 31st August 2018, will be used. One of the advantages of this forecasting technique is its flexibility whereby it just requires the ability to calculate the probability at any given point. Numerical analysis is done using R. This forecasting technique is useful, not only to KenGen Company, but also to other companies listed in the NSE, the share brokers as well as the shareholders, and any other individual or company interested in trading in the share market.***

Keywords: Markov Chain Model, Time Series Analysis, Nairobi Stock Market, Trend Prediction.

1. Introduction

There is never ending debate as to whether stock markets are predictable or not. According (Schrimpf, 2010), in the recent years, investors have started to show interest in trading on stock market indices as it provides an opportunity to hedge their market risk, and at the same time it offers a good investment opportunity for speculators and share brokers. This study will provide a good solution to the interested group with major focus on the companies listed in the Nairobi Stock Exchange (NSE).

There has always been a dream of fascination with the attempt to predict the share market prices. This drive has led to several attempts by interested parties to make efforts at predicting these prices, sometimes with little success, sometimes with high success rates and sometimes with no success at all. This study makes a daring attempt to at using Markov Chain Model to make predictions in this market. From the mid-70s, and particularly from 1980, extensive efforts have been made on the predictability of stock prices using new mathematical techniques, long time series and advanced tools such as artificial intelligence and many tests have been performed on stock index and price information so as to show the presence or lack of a certain structure in the information of the stock price and thus violate the assumption of random steps (Moreno & Olmeda, 2007).

According to (Wohar & Rapach, 2006) classical methods such as regression have had relative success in these areas, but the results failed to satisfy the researchers and to predict events that will happen in the future, information obtained from historical events will be relied on. The data is analyzed to obtain a pattern generalizable for the foreseeable future. In most forecasting methods, it is assumed that the relationships between variables will continue in the future and MCM will not be an exception.

Predicting stock market trend has become an important activity with many decisions accepted within the context of randomness. In order to calculate, understand, and predict the effects of randomness, one special type of stochastic processes named Markov chains is examined in this paper. Markov chains are usually used in modeling many practical problems and are useful in studying the evolution of systems where the state of the system cannot be determined with certainty (Anderson, Sweeney, Williams, Camm, & Cochran, 2015). Markov chains are often used for capturing dynamic behaviour with a large stochastic component (Peng, Bao, Zhao, Yi, Xia, & Shen, 2010) and are also effective in modeling time series. If a Markov chain can model the time series accurately, then good predictions and optimal planning in a decision process can be made (Liu, 2010). This study uses discrete time, discrete state first order Markov chains in order to obtain short- term forecasts of the NSE time series.

2. Review of Markov Chain Models

A Markov chain is a discrete-valued Markov process, discrete-valued means that the state spaces of possible values of the Markov chain are finite or countable (Chizhov & Stepchenko , 2016). A Markov chain is a stochastic process that satisfies the Markov property, which means that the past and future are independent when the present is known (Sericola, 2013). Other applicable Markov Chain Models include:

2.1 Markov Chain Monte Carlo (MCMC)

According to (Landauskas & Valakevicius, 2011), Monte carlo process treats time series as a Brownian motion. Thus it satisfies the equation:

$$dS = \mu S dt + \sigma S dz \qquad (2.1)$$

Consider a financial mean with lognormal distributed returns. The random walk of price of such a financial mean is modeled according to this formula (Wilmott, 2007):

$$S(t+\Delta t) = S(t)e^{\left(\delta - \frac{1}{2}\sigma^2\right)\Delta t + \sigma\sqrt{\Delta t}z} \qquad (2.2)$$

Here, random value $Z \sim N(0,1)$ follows standard normal distribution, δ is annual risk free return and σ is annual standard deviation of the logarithm of a stock price.

A Markov Monte Carlo is then introduced at this stage where we suppose it is needed to generate $\chi_i \sim \pi(x)$. When $\chi_i \sim \pi(x)$ is difficult to sample, then the MCMC sampling technique could be used. This is also the main purpose for using an MCMC. The main idea here is to construct a Markov Chain $\{X_i\}_{i=0}^{\infty}$, such that:

$$\lim_{i\to\infty} P(X_i = x) = \pi(x) \quad (2.3)$$

A Markov Chain is predefined by an initial state $P(X_0 = x_0) = g(x_0)$ and the transition kernel $P(y/x) = P(X_{i+1} = y / X_i = x)$. The stationary distribution $\pi(x) = \lim_{i\to\infty} f(x_i)$ is unique if the chain is ergodic. Then:

$$\pi(y) = \sum_{x\in\Omega} \pi(x) P(y/x), \forall y \in \Omega \quad (2.4)$$

2.2 Hidden Markov Chain

According to (Stephen & Lihn, 2017) Hidden Markov Model (HMM) is the homogeneous first-order HMM, where the state at time t is only dependent on the states of previous time t − 1. The mixing distribution is univariate. The parameter space of the HMM is comprised of $\pi = \{\theta, \Gamma, \delta\}$, where θ is a matrix containing parameters of the mixing distributions, Γ is the transition probability matrix, and δ is the initial state probability vector. In the case of a stationary solution, δ is also the stationary state distribution vector.

The notations are defined as following:

- The latent states are indexed by an integer, i = 1, 2, . . ., m, where m is number of states.
- The time series is the log-returns of a financial instrument, indexed by an integer, t = 1, 2, . . ., T. The time period can be daily, weekly, monthly, etc.
- The observation at time t in the time series is the log-return between time t − 1 and t, which is denoted as x_t. And x_t is unbounded. The vector of all observations can be written as $x^{(T)}$.
- The latent state at time t is denoted as C_t, which can have an integer value from i = 1, 2, . . ., m.
- The transition probability matrix is $\{\gamma_{ij}\}$, or simply Γ in matrix form. γ_{ij} is the probability to transition from state i to state j. It is independent of t.
- The initial state probability vector is $\{\delta_i\}$, or simply δ in vector form. δ also represents the stationary state distribution in which δ = δ. Notice that δ is also defined as α_0.
- The mixing probability of observations is represented as $P_i(x)$, which is the probability density function (PDF) for x when it is in state i. Or simply P (x) in matrix form, where $P_{ii}(x) = P_i(x)$. That is, P (x) is a diagonal matrix. It is independent of t.

2.3 The First-Order Markov Chain Model

We consider modeling a time series x_t by a first-order Markov chains having k states E= {1,2……k}. A first-order discrete-time Markov chain having k states satisfies the following relationship:

$P(x_{t+1} = i_{t+1} \mid x_0 = i_0, x_1 = i_1, \ldots, x_t = i_t) = P(x_{t+1} = i_{t+1} \mid x_t = i_t)$, (2.5)

Where x_t is the state of a time series at time t and $i_j \in E$. The conditional probabilities

$$P(x_{t+1} = i_{t+1} \mid x_t = i_t) \quad (2.6)$$

are called the one-step transition probabilities of the Markov chain. These probabilities can be written as $p_{ij} = P(x_{t+1} = i \mid x_t = j)$ for i and j in E.

The matrix $P = (p_{ij})_{k\times k}$ is called the one-step transition probability matrix. We note that the elements of the matrix P satisfy the following two properties:

$$0 \le p_{ij} \le 1, \ \forall i, j \in E \quad (2.7)$$

$$\sum_{i=1}^{k} p_{ij} = 1, \ \forall j \in E$$

A first order Markov Chain,

$$x_{t+1} = Px_t \quad (2.8)$$

is then constructed for the observed time series.

A transition matrix P has an eigenvalue equal to one and all the eigenvalues of P must have modulus less than or equal to one (Horn & Johnson, 1985).

It can be shown that there exists a positive vector $x = [x_1, x_2, \ldots, a_m]^T$ such that

$$P\,x = x \quad (2.9)$$

if P is irreducible. The vector x in normalized form is called the stationary probability vector of P. Moreover, x_i is the stationary probability that the system is in state i (Ching, Ng & Fung, 2008).

2.4 The Higher-Order Markov Chain Model

Higher-order (n^{th}-order) Markov chain models have been proposed by Raftery (1985) and Ching et al. (2004) for modeling categorical data sequences.

Ching, Ng & Fung (2008) have suggested a series of modeling methods based on the Markov chain (including the higher-order Markov chain model). We note that a time series $\{x_t\}$ of k states can be represented by a series of vectors (probability distribution)

$\{x_0, x_1, x_2, \ldots\}$

called the canonical form representation. If the system is in state $j \in E$ at time (t + i) then the state probability distribution vector is given by

$$x_{t+i} = \{0, \ldots, 0, \underbrace{1}_{j^{th}\text{ entry}}, 0, \ldots, 0\}^T$$

In addition, we assume that the state probability distribution at time $t = m + 1$ depends on the state probability distribution of the sequence at times $t = m, m-1, \ldots, m-n+1$.

The model is given as follows:

$$x_{m+1} = \sum_{i=1}^{n} \lambda_i P_i x_{n-i+1} \quad i = m-1, m, \ldots \quad (2.10)$$

With initials $x_0, x_1, \ldots, x_{n-1}$. Here the weights λi are non-negative real numbers such that

$$\sum_{i=1}^{n} \lambda_i = 1 \quad (2.11)$$

Here x_m is the state probability distribution at time m, P_i is the i-step transition matrix and λ_i are the non-negative weights. The total number of parameters is of $O(nk^2)$ (Ching, Ng & Fung, 2008).

We estimate the transition probability matrix P_i by the following method. Given the data series, we count the transition frequency from the states in the sequence at time $t = m - i + 1$ to the states in the j^{th} sequence at time $t = m + 1$ for $1 \le i \le n$. Hence one can construct the transition frequency matrix for the data sequences. After making normalization, the estimates of the transition probability matrices $\hat{P}_i$ can also be obtained.

$$Qx \equiv x, \text{ where } Q = \sum_{i=1}^{n} \lambda_i P_i \quad (2.12)$$

It would be expected that $\hat{Q}\hat{x} \approx \hat{x}$

From (iv) one possible way to estimate the parameters λ_i is given as follows. The following minimization problem will be solved:

$$\min_{\lambda} \|\hat{Q}\hat{x} - \hat{x}\|$$

$$s.t \begin{cases} \sum_{i=1}^{n} \lambda_i = 1 \\ \lambda_i \ge 0, i = 1, \ldots, n \end{cases} \quad (2.13)$$

Here $\|\cdot\|$ is certain vector norm. If $\|\cdot\|$ is chosen to be the $\|\cdot\|_\infty$ norm, then the above optimization problem becomes

$$\min_{\lambda} \max_{i} \left| \left\{ \sum_{i=1}^{n} \lambda_i P x - x \right\}_i \right| \quad (2.14)$$

$$s.t \begin{cases} \sum_{i=1}^{n} \lambda_i = 1 \\ \lambda_i \ge 0, i = 1, \ldots, n \end{cases} \quad (2.15)$$

where $[\cdot]_i$ denote the i^{th} entry of the vector. Problem 3.6 can be formulated as s linear programming problems as follows:

$$\min_{\lambda} \varepsilon$$

$$s.t. \begin{cases} -M \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \cdot \\ \cdot \\ \cdot \\ \lambda_n \end{pmatrix} - \begin{pmatrix} \varepsilon \\ \varepsilon \\ \cdot \\ \cdot \\ \cdot \\ \varepsilon \end{pmatrix} \le - \begin{pmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{pmatrix} \\ M \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \cdot \\ \cdot \\ \cdot \\ \lambda_n \end{pmatrix} - \begin{pmatrix} \varepsilon \\ \varepsilon \\ \cdot \\ \cdot \\ \cdot \\ \varepsilon \end{pmatrix} \le \begin{pmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{pmatrix} \\ \sum_{i=1}^{n} \lambda_i = 1 \\ \lambda_i \ge 0, i = 1, \ldots, n \end{cases} \quad (2.16)$$

Where $M = [\hat{P}_1\hat{x} \;\; \hat{P}_2\hat{x} \ldots \hat{P}_n\hat{x}]$.

We remark that other norms such as $\|\cdot\|_2$ and $\|\cdot\|_1$ can also be considered. The former will result in a quadratic programming problem while the latter will still result in a linear programming problem. Then we use the higher-order Markov model to predict the next state of the sequence $\hat{x}_t$ at time t which can be taken as the state with the maximum probability i.e

$$\hat{x}_t = j, \text{ if } [\hat{x}_t]_j \le [\hat{x}_t]_j \quad \forall 1 \le i \le j \quad (2.17)$$

To evaluate the performance and effectiveness of the higher-order Markov chain model, a prediction result is measured by the prediction accuracy s defined as

$$s = \frac{1}{N-n} \sum_{i=n+1}^{N} a_i \times 100\%, \quad (2.18)$$

Where N is the length of the data sequence and

$$a_i = \begin{cases} 1, & \text{if } \hat{x}_t = x_t \\ 0, & \text{Otherwise} \end{cases} \quad (2.19)$$

3. Application to Analysis and Share Price Prediction

This study simulates Markov Chain under specified conditions likely to be encountered in real life situation. First order Markov Chain was proposed for this study because of its simplicity and directness. The data is then used to illustrate estimation of the future share prices using the model.

We study share market prices for KenGen company where the response variable is the share price. Share price is a function of itself as the main variable. This study considers one-time transition probability from one state to the next with eight major state spaces taken into account. Further, an instrument, p, is generated for a matrix of transition probabilities, which form a major component of the first order Markov Chain Model. As discussed earlier, the Panel Data Regression Model is;

$$x = Px$$

To generate a set of discrete values, data will be put in eight categories and assigned numerical values. The time series of the share price in the NSE is given in the appendix. The

market share price for KenGen were classified into eight possible states (1, 2, 3, 4, 5, 6, 7, 8), see the table 1 below.

Table 1: Share Price Divisions

1 = (not greater than sh.5.606)
2 = (not greater than sh.6.262)
3 = (not greater than sh.6.918)
4 = (not greater than sh.7.574)
5 = (not greater than sh.8.230)
6 = (not greater than 8.886)
7 = (not greater than 9.542)
8 = (not greater than 10.198)

On the one hand the company sales demand for share trade in order to minimize its inventory build-up, while on the other hand the shareholders would like to predict the share price in order to decide the trade strategy. With the share price series, today's price mostly dependents on yesterday's price. We choose the first-order Markov chain model. We first estimate the one-step transition probability matrix P by using the method said above.

The resulting transition matrix is:

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]
[1,]	0.8696	0.1304	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
[2,]	0.0333	0.8556	0.1111	0.0000	0.0000	0.0000	0.0000	0.0000
[3,]	0.0000	0.0481	0.9183	0.0337	0.0000	0.0000	0.0000	0.0000
[4,]	0.0000	0.0000	0.1569	0.7647	0.0784	0.0000	0.0000	0.0000
[5,]	0.0000	0.0000	0.0000	0.0400	0.8600	0.1000	0.0000	0.0000
[6,]	0.0000	0.0000	0.0000	0.0000	0.0769	0.8769	0.0462	0.0000
[7,]	0.0000	0.0000	0.0000	0.0000	0.0000	0.1111	0.8519	0.0370
[8,]	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.2857	0.7142

Having constructed transition probability matrix of our model, vector P(n) is developed from the coded share prices discussed above. Then the R system is implemented, to produce numerical results. The R command is also executed for the stationary distribution x. It has been shown that when n is large, a stationary distribution is reached, where all rows are equal. In other words, regardless of the initial state, the probability of ending up with a certain state is the same. Once such convergence is reached, any row of this matrix is the stationary distribution. For example, you can extract the first row:
a<-mpow(t,750) [1,]
The resulting stationary distribution is:

(0.0371 0.1451 0.3354 0.0719 0.1411 0.1834 0.0762 0.0099)

Having successfully established the transition matrix and the stationary distribution, we now construct the first order Markov Chain transition probability, which is:

	[,1]
[1,]	0.05117340
[2,]	0.16264589
[3,]	0.31734687
[4,]	0.11868801
[5,]	0.14253472
[6,]	0.17518241
[7,]	0.08563184
[8,]	0.02881755

With this milestone, the developed Markov Chain Model is used to predict the next share price, given that the last 8 share prices were (3,3,3,3,3,3,3,3),
We use the model generated to produce the next share price

	[,1]
[1,]	3.246062

The resulting share price, which should have been the price on 3rd September, 2018 is 3.246062

4. Conclusion and Recommendations

We have reviewed Markov Chain models with transition probability matrices, and applied it to forecasting trends of NSE using KenGen share prices. All results show that the effectiveness of the Markov model to forecast the Stock Market Share Prices is very good.

In further research we would like to concentrate ourselves to constructing different state spaces S within the same MCM, the results obtained will be compared for different state spaces and made a decision on the effect of the number of state spaces on the prediction accuracy. Finally, another opening of interest that could be delved into is making a continuous versus discrete Markov processes for a set of market share indices. This would involve superimposing a mechanism that dictates when the process changes.

References

[1] Anderson, D. R., Sweeney, D. J., Williams, T. A., Camm, J. D., & Cochran, J. J. (2015). Quantitative Methods for Business. *13th ed.South-Western College Pub*, 771.
[2] Chizhov, A., & Stepchenko , J. (2016). Applying Markov Chains for NDVI Time Series Forecasting of Latvian Regions. *Information Technology and Management Science*, 57–61.
[3] Hjalmarsson, E. (2010). Predicting global stock returns. *Journal of Financial and Quantitative Analysis 45(1)*, 49-80.
[4] Liu, T. (2010). Application of Markov Chains to Analyze and Predict the Time Series. *Modern Applied Science, vol. 4, no. 5*, 162-166.
[5] Moreno, I., & Olmeda, D. (2007). Is the predictability of emerging and developed stock markets really exploitable? *European Journal of Operations Research 182(1)*, 436-454.

[6] Peng, Z., Bao, C., Zhao, Y., Yi, H., Xia, L., & Shen, H. (2010). Weighted Markov chains for forecasting and analysis in Incidence of infectious diseases in jiangsu Province, China. *Journal of Biomedical Research, vol. 24, no. 3*, 207-214.

[7] Schrimpf, A. (2010). International stock returns predictability under model uncertainty. *Journal of International Money and Finance 29(7)*, 1256-1282.

[8] Wohar, & Rapach, D. E. (2006). In-sample vs. out-of-sample tests of stock return predictability in the context of data mining. *Journal of Empirical Finance 13(2)*, 231-247.

Appendix

#Coded share prices:

(4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 2, 2, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 3, 3, 3, 3, 3, 4, 3, 4, 4, 4 , 4, 4, 5, 5, 5, 5, 5, 5,
5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 5, 5, 5, 4, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 2, 2, 2, 2, 3, 2, 2, 2, 2, 2, 2, 2, 2, 3, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 2, 3, 3, 2, 2, 2, 2, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4,
4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 6, 5, 5, 5, 6, 6,
6, 7, 7, 8, 8, 7, 7, 7, 7, 7, 7, 7, 8, 8, 8, 8, 8, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 6, 6, 6, 6, 6, 6, 6, 6, 6, 7, 7, 7, 6, 6, 6, 6,
6, 5, 6, 6, 6, 6, 6, 7, 7, 7, 7, 7, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 7, 7, 7, 7, 7, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 5, 5,
5, 6, 6, 6, 6, 6, 6, 6, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 6, 6, 5, 6, 6, 6, 6, 6, 6, 5, 5, 6, 6, 6, 6, 7, 7, 7, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 7, 7,
7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 5, 6, 6, 6, 6, 6, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 4,
4, 3, 3, 4, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 3, 4, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 2, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3).

#Revised R code for simulation of Markov Chain model

For further enquiry contact otienoor@gmail.com

```
> trans.matrix <- function(chain)  {
+   states <- length(unique(chain))
+ p <- matrix (nrow = states, ncol = states, 0) # initialise transition matrix
+   for (t in 1:(length(chain) - 1)) {p[chain[t], chain[t + 1]] <- p[chain[t], chain[t + 1]] + 1}
+   for (i in 1: states) p [i,] <- p [i,] / sum (p [i,]) # make stochastic
+   return (p)
+ }
> chain <- c (4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 2, 2, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 3, 3, 3, 3, 3, 4, 3, 4, 4, 4, 4, 4, 5,
5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 5, 5, 5, 4, 4, 4, 4, 4, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 2, 2, 3, 2, 2, 2, 2, 2, 2, 2, 2, 3, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 2, 3, 3,
2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 6,
5, 5, 5, 6, 6, 6, 7, 7, 8, 8, 7, 7, 7, 7, 7, 7, 7, 8, 8, 8, 8, 8, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 6, 6, 6, 6, 6, 6, 6, 6, 6, 7, 7,
7, 6, 6, 6, 6, 6, 5, 6, 6, 6, 6, 6, 7, 7, 7, 7, 7, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 7, 7, 7, 7, 7, 6,
6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 5, 5, 5, 6, 6, 6, 6, 6, 6, 6, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 6, 6, 5, 6, 6, 6, 6, 6,
6, 5, 5, 6, 6, 6, 6, 7, 7, 7, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 5, 6, 6,
6, 6, 6, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 4, 4, 3, 3, 4, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4,
4, 4, 4, 3, 4, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3)
> trans.matrix(chain)
           [,1]       [,2]      [,3]       [,4]       [,5]      [,6]        [,7]       [,8]
[1,] 0.86956522 0.13043478 0.0000000 0.00000000 0.00000000 0.0000000 0.000000000 0.00000000
[2,] 0.03333333 0.85555556 0.1111111 0.00000000 0.00000000 0.0000000 0.000000000 0.00000000
[3,] 0.00000000 0.04807692 0.9182692 0.03365385 0.00000000 0.0000000 0.000000000 0.00000000
[4,] 0.00000000 0.00000000 0.1568627 0.76470588 0.07843137 0.0000000 0.000000000 0.00000000
[5,] 0.00000000 0.00000000 0.0000000 0.04000000 0.86000000 0.1000000 0.000000000 0.00000000
[6,] 0.00000000 0.00000000 0.0000000 0.00000000 0.07692308 0.8769231 0.04615385  0.00000000
[7,] 0.00000000 0.00000000 0.0000000 0.00000000 0.00000000 0.1111111 0.85185185  0.03703704
[8,] 0.00000000 0.00000000 0.0000000 0.00000000 0.00000000 0.0000000 0.28571429  0.71428571
> t<-trans.matrix(chain)
> mpow <- function(t,n) {
+  if (n == 0) diag(nrow(t))
+  else if (n == 1) t
+  else t %*% mpow(t, n - 1)
```

```
+ }
> mpow(t,0)
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]    1    0    0    0    0    0    0    0
[2,]    0    1    0    0    0    0    0    0
[3,]    0    0    1    0    0    0    0    0
[4,]    0    0    0    1    0    0    0    0
[5,]    0    0    0    0    1    0    0    0
[6,]    0    0    0    0    0    1    0    0
[7,]    0    0    0    0    0    0    1    0
[8,]    0    0    0    0    0    0    0    1

          [,1]       [,2]      [,3]       [,4]       [,5]       [,6]       [,7]       [,8]
[1,] 0.86956522 0.13043478 0.0000000 0.00000000 0.00000000 0.0000000 0.000000000.00000000
[2,] 0.03333333 0.85555556 0.1111111 0.00000000 0.00000000 0.0000000 0.000000000.00000000
[3,] 0.00000000 0.04807692 0.9182692 0.03365385 0.00000000 0.0000000 0.000000000.00000000
[4,] 0.00000000 0.00000000 0.1568627 0.76470588 0.07843137 0.0000000 0.000000000.00000000
[5,] 0.00000000 0.00000000 0.0000000 0.04000000 0.86000000 0.1000000 0.000000000.00000000
[6,] 0.00000000 0.00000000 0.0000000 0.00000000 0.07692308 0.8769231 0.046153850.00000000
[7,] 0.00000000 0.00000000 0.0000000 0.00000000 0.00000000 0.1111111 0.851851850.03703704
[8,] 0.00000000 0.00000000 0.0000000 0.00000000 0.00000000 0.0000000 0.285714290.71428571
> mpow (t,3)

          [,1]          [,2]          [,3]          [,4]          [,5]          [,6]          [,7]          [,8]
[1,] 0.6687974759 0.2924047872 0.0383100001 0.0004877369 0.00000000000.0000000000 0.000000000 0.0000000000
[2,] 0.0747256678 0.6515117089 0.2639769746 0.0094923690 0.00029327970.0000000000 0.000000000 0.0000000000
[3,] 0.0042362019 0.1142208063 0.8024143966 0.0721524165 0.00671222690.0002639517 0.000000000 0.0000000000
[4,] 0.0002513826 0.0191442735 0.3363070618 0.4675985352 0.15671613670.0196206193 0.000361991 0.0000000000
[5,] 0.0000000000 0.0003016591 0.0159559223 0.0799252297 0.66382748710.2278705697 0.011948192 0.0001709402
[6,] 0.0000000000 0.0000000000 0.0004826546 0.0076973199 0.17528505360.7078177200 0.104541080 0.0041761720
[7,] 0.0000000000 0.0000000000 0.0000000000 0.0003418803 0.02212628140.2516729701 0.656968813 0.0688900549
[8,] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.00244200240.0775574807 0.531437567 0.3885629502
> mpow(t,15)

          [,1]          [,2]          [,3]          [,4]       [,5]          [,6]          [,7]          [,8]
[1,] 2.053281e-01 0.3680220327 0.372839008 0.04010839 0.01070757 0.0027459740.0002379845 1.097549e-05
[2,] 9.405008e-02 0.3032244077 0.500952305 0.06723690 0.02495166 0.0085667660.0009633751 5.450979e-05
[3,] 4.122739e-02 0.2167582089 0.569044429 0.09370477 0.05117797 0.0242435330.0035951629 2.485271e-04
[4,] 2.067211e-02 0.1356038322 0.436763989 0.10624315 0.14953955 0.1219182910.0268867522 2.372332e-03
[5,] 2.814561e-03 0.0256645660 0.121657357 0.07626517 0.30515145 0.3547023600.1030701347 1.067440e-02
[6,] 5.552299e-04 0.0067781005 0.044331032 0.04782948 0.27284797 0.4309330640.1750540513 2.167107e-02
[7,] 1.158443e-04 0.0018350001 0.015826326 0.02539304 0.19087062 0.4214264200.2982126563 4.632009e-02
[8,] 4.121409e-05 0.0008009601 0.008439776 0.01728413 0.15249139 0.4024627200.3573264087 6.115340e-02
        > mpow(t,50)

          [,1]       [,2]      [,3]       [,4]       [,5]       [,6]       [,7]       [,8]
[1,] 0.062914347 0.22715784 0.4826085 0.08070534 0.06933843 0.058246260.01712230 0.001906946
[2,] 0.058051449 0.21640047 0.4707670 0.08090221 0.07779425 0.071030810.02246209 0.002591713
[3,] 0.053365366 0.20369727 0.4515446 0.08023296 0.08859286 0.088662800.03028409 0.003620100
[4,] 0.041596031 0.16316411 0.3739710 0.07516389 0.12511649 0.152831730.06047290 0.007683833
[5,] 0.018226101 0.08001694 0.2105979 0.06380941 0.19988737 0.286534810.12455065 0.016376842
[6,] 0.011777267 0.05620020 0.1621263 0.05995707 0.22041140 0.325634790.14470008 0.019192937
[7,] 0.008334665 0.04278493 0.1333141 0.05711330 0.23064935 0.348352050.15826128 0.021190359
[8,] 0.007160778 0.03808232 0.1229356 0.05598221 0.23395489 0.356440260.16346848 0.021975422
   > mpow(t,150)

          [,1]       [,2]      [,3]       [,4]       [,5]       [,6]       [,7]       [,8]
[1,] 0.04272588 0.1648653 0.3737730 0.07460887 0.1234906 0.1518163 0.060922430.007797627
[2,] 0.04213225 0.1627870 0.3697324 0.07432899 0.1253396 0.1551370 0.062526720.008016072
[3,] 0.04133066 0.1599804 0.3642757 0.07395096 0.1278365 0.1596215 0.064693330.008311089
[4,] 0.03845387 0.1499072 0.3446902 0.07259404 0.1367980 0.1757169 0.072469820.009369986
[5,] 0.03246038 0.1289207 0.3038855 0.06976696 0.1554684 0.2092504 0.088671540.011576120
[6,] 0.03069693 0.1227458 0.2918793 0.06893510 0.1609618 0.2191171 0.093438710.012225254
```

```
[7,] 0.02965536 0.1190985 0.2847876 0.06844372 0.1642065 0.2249451 0.096254560.012608684
[8,] 0.02928088 0.1177872 0.2822378 0.06826704 0.1653731 0.2270404 0.097266990.012746545
> mpow(t,300)

        [,1]        [,2]        [,3]        [,4]        [,5]        [,6]        [,7]        [,8]
[1,] 0.03779278 0.1475922 0.3401889 0.07228215 0.1388574 0.1794160 0.074257100.009613356
[2,] 0.03771801 0.1473304 0.3396799 0.07224688 0.1390903 0.1798343 0.074459200.009640876
[3,] 0.03761705 0.1469769 0.3389925 0.07219926 0.1394049 0.1803992 0.074732130.009678041
[4,] 0.03725466 0.1457080 0.3365254 0.07202832 0.1405337 0.1824267 0.075711730.009811429
[5,] 0.03649967 0.1430644 0.3313853 0.07167220 0.1428856 0.1866509 0.077752640.010089333
[6,] 0.03627752 0.1422865 0.3298729 0.07156742 0.1435776 0.1878938 0.078353150.010171103
[7,] 0.03614631 0.1418270 0.3289795 0.07150552 0.1439864 0.1886280 0.078707850.010219401
[8,] 0.03609913 0.1416619 0.3286584 0.07148327 0.1441333 0.1888919 0.078835380.010236766
> mpow(t,350)

        [,1]        [,2]        [,3]        [,4]        [,5]        [,6]        [,7]        [,8]
[1,] 0.03743822 0.1463507 0.3377750 0.07211490 0.1399619 0.1813998 0.075215550.009743866
[2,] 0.03740074 0.1462195 0.3375199 0.07209722 0.1400787 0.1816095 0.075316860.009757662
[3,] 0.03735012 0.1460423 0.3371753 0.07207335 0.1402364 0.1818927 0.07545368 0.009776292
[4,] 0.03716847 0.1454062 0.3359385 0.07198766 0.1408022 0.1829090 0.075944740.009843158
[5,] 0.03679000 0.1440809 0.3333619 0.07180914 0.1419812 0.1850266 0.076967830.009982468
[6,] 0.03667864 0.1436910 0.3326037 0.07175662 0.1423281 0.1856496 0.077268860.010023458
[7,] 0.03661286 0.1434607 0.3321559 0.07172559 0.1425330 0.1860176 0.077446660.010047669
[8,] 0.03658921 0.1433779 0.3319949 0.07171444 0.1426067 0.1861499 0.077510590.010056374
        > mpow(t,500)

        [,1]        [,2]        [,3]        [,4]        [,5]        [,6]        [,7]        [,8]
[1,] 0.03712672 0.1452600 0.3356543 0.07196797 0.1409323 0.1831426 0.076057600.009858525
[2,] 0.03712200 0.1452434 0.3356222 0.07196575 0.1409470 0.1831690 0.076070360.009860263
[3,] 0.03711562 0.1452211 0.3355787 0.07196274 0.1409669 0.1832047 0.076087600.009862610
[4,] 0.03709274 0.1451410 0.3354230 0.07195194 0.1410381 0.1833327 0.076149460.009871033
[5,] 0.03704506 0.1449741 0.3350984 0.07192946 0.1411867 0.1835995 0.076278330.009888582
[6,] 0.03703103 0.1449249 0.3350029 0.07192284 0.1412304 0.1836780 0.076316250.009893745
[7,] 0.03702275 0.1448959 0.3349465 0.07191893 0.1412562 0.1837243 0.076338650.009896795
[8,] 0.03701977 0.1448855 0.3349262 0.07191753 0.1412655 0.1837410 0.076346700.009897892
        > mpow(t,750)

          [,1]        [,2]        [,3]        [,4]        [,5]        [,6]        [,7]          [,8]
[1,] 0.03708324 0.1451078 0.3353583 0.07194747 0.1410677 0.1833858 0.07617512  0.009874527
[2,] 0.03708309 0.1451072 0.3353573 0.07194740 0.1410682 0.1833867 0.07617552  0.009874582
[3,] 0.03708289 0.1451065 0.3353559 0.07194730 0.1410688 0.1833878 0.07617607  0.009874657
[4,] 0.03708217 0.1451040 0.3353510 0.07194696 0.1410711 0.1833919 0.07617803  0.009874923
[5,] 0.03708066 0.1450987 0.3353407 0.07194625 0.1410758 0.1834003 0.07618211  0.009875479
[6,] 0.03708021 0.1450972 0.3353377 0.07194604 0.1410772 0.1834028 0.07618331  0.009875642
[7,] 0.03707995 0.1450962 0.3353359 0.07194591 0.1410780 0.1834043 0.07618402  0.009875739
[8,] 0.03707986 0.1450959 0.3353353 0.07194587 0.1410783 0.1834048 0.07618427  0.009875773
> mpow(t,750) [1,]
[1] 0.037083243 0.145107757 0.335358325 0.071947466 0.141067721 0.183385842
[7] 0.076175119 0.009874527
> a<-mpow(t,750) [1,]
> y<-(t%*%a)
> y
        [,1]
[1,] 0.05117340
[2,] 0.16264589
[3,] 0.31734687
[4,] 0.11868801
[5,] 0.14253472
[6,] 0.17518241
[7,] 0.08563184
[8,] 0.02881755
> s<-c(3,3,3,3,3,3,3,3)
> q <- matrix (s, nrow = 1, ncol = 8, byrow = TRUE)
```

```
> q
[,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,] 3    3    3    3    3    3    3    3
> r<-(q%*%y)
> r
[,1]
[1,] 3.246062
```