Markov Chain based Models for Time-Series based Prediction

Experiment Findings · September 2021

DOI: 10.13140/RG.2.2.30607.30881

1 author:

Pranshu Tiwari
IBM
5 PUBLICATIONS **0** CITATIONS

SEE PROFILE

Markov Chain based Models for Time-Series based Prediction

Pranshu Tiwari

Abstract

The paper uses Markov Chain method to predict appliance data at consumer level. Traditionally Markov chain has been used in Stock Market Forecasting and man-power planning [1] The paper initially briefs the mathematical methodology and then details how the model performs as compared to original data. Steady state Transition Matrix was derived and then was used to create a simulated model. Autocorrelation between different states was compared between simulated model and original discrete data. Two step Markov Chain was also developed to predict jump in 2 units of time. We also compute Chi-square analysis and Auto-correlation models to compare how the markov model performs as compared to original discrete Data. Steady State Probability vector has been derived and it was found that there is 0.89 probability that individual will be in very low energy state .On the other hand there is a probability of 0.02 that consumer would be in Very High Energy Consumption state. This shows consumers are energy conscious and would prefer to consume lower energy.

1. Introduction

There are many prediction models that are used to predict and forecast complex engineering problems including ARIMA (Auto Regression and Moving Average), Auto-Correlation, Auto Regression and Recurrent Neural Networks. This paper presents how we can use Markov Chain based model for prediction of energy based on time series data. Markov Chain based models have been extensively used in complex engineering problems including Queueing Problems. Towards this, the paper presents how we can use Markov Chain to predict states of energy consumption. Energy states with high consumption can be predicted which can be used to alert the customer and balance the grid demand -supply accordingly.

The Markov chain is a special case of the stochastic process [1]. Markov is a stochastic process with the Markov property that was named after Andrei Andreevich Markov, a Russian mathematician [2]. Recently, the methods have been used to estimate the matrix of transitive from the observing states of the system [2]. It is a random process where all information about the future is contained in the present state

Definition 1 ([23]). The sequence {Xt , t ≥ 0} is said to be a Markov chain if for all state values $i_0, i_1, i_2 \ldots, i_t \in$ I, then P X_{t+1}= j | X = $i_0, i_1, i_2 \ldots$, where $i_0, i_1, i_2 \ldots$ are the states in the state space i. This type of probability is called Markov chain probability. This indicates that regardless of its history prior to time n, the probability that it will make a transition to another state j depends only on last previous state i_t.Here it should be noted that whether the particle was in that state for only a short period or a long period of time does not matter. For discrete-time Markov chain is a Markov process where the state space is finite.

In addition, the main components in developing the Markov chain model are state transition matrix and probability; both of which will summarize all the essential parameters of dynamic change[3]. We represent the time-series energy-data as Discrete Finite Markov Chain model. We can check if the chain in irreducible and then find the transition probability and stationary distribution. Stationary distribution refers to long run probability of being in state j at any time n is equal to Π_j. That is if

P(X{$x_0 = \bar{J}\} = \Pi_J$. We then simulate the data using markov Chain model and then calculate the auto correlation coefficient to evaluate the efficacy of simulated model.

2. Data Set

We take the energy appliance data set from Kaggle. The data set is at 10 min for about 4.5 months. The house temperature and humidity conditions were monitored with a ZigBee wireless sensor network. Each wireless node transmitted the temperature and humidity conditions around 3.3 min. Then, the wireless data was averaged for 10 minutes periods. The original data set is multi-variate dataset has taken 27 features/measurement vectors. In this case we just want to predict the appliance energy given the past energy usage using Markov Chain. Hence for ease we take appliance energy data as the only feature for prediction

3. Approach

The following Diagram illustrates the method as below

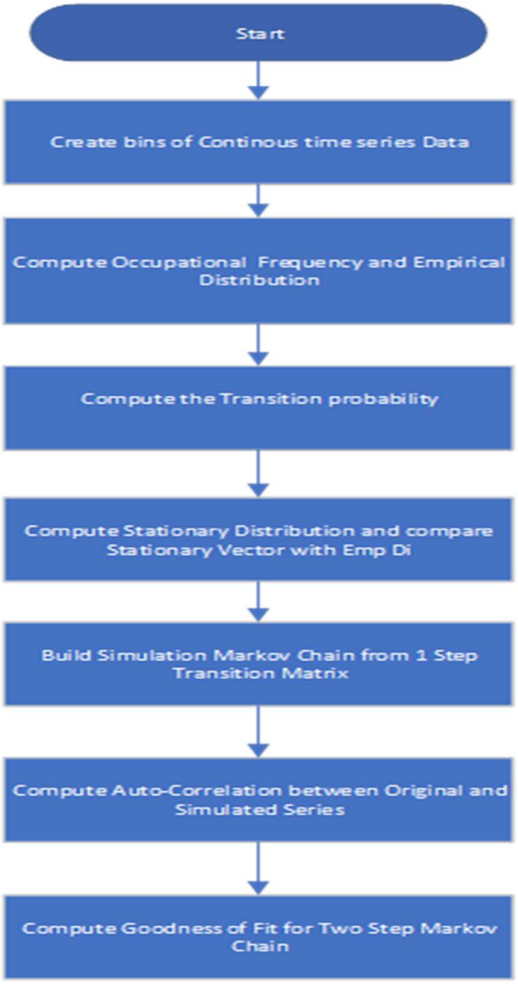

Figure 1: Overall Approach for Markov Chain based Prediction Model

4. Detailed Mathematical Construct

Step 1 Create bins of Continuous time Series Data.

We used bins of difference of 200 Units and created 6 bins from energy consumption as per table below. These states have been derived by creating bins of 200 units each

Energy Consumption	State
Less than 200 Units	1
200-400 Units	2
400 Units -600 Units	3
600 Units-800 Units	4
800 Units-1000 Units	5
1000-1200 Units	6

Table 1: Energy States

Step 2 & Step 3: Empirical Distribution and Creating a Transition Matrix

Empirical Distribution represents the count of occurrences of each state. N represents the state jumps as defined by the Markov chain, indicates the observed frequency of transition or jump from one state to another state. These are given by N which represents actual number of occurrences of state $[i, j]$

$$N = \begin{bmatrix} N_{11} & \cdots & N_{1J} \\ \vdots & \ddots & \vdots \\ N_{i1} & \cdots & N_{iJ} \end{bmatrix} \ldots\ldots eq(1)$$

Let P be a transition matrix or stochastic matrix that describes all the transition probabilities for each state of the Markov chain model. Hence, P is denoted as below.

$$Transition\ Matrix = P = \begin{bmatrix} P_{11} & \cdots & P_{1J} \\ \vdots & \ddots & \vdots \\ P_{i1} & \cdots & P_{iJ} \end{bmatrix} \ldots eq\ (2)$$

A state i is said to be recurrent if and only if there is a probability of 1 that a process, starting in state i will at some point return to state i. And finally, in a related definition, a state i is said to be transient if the probability that the process, starting in state i, will at some point return to state i is less than 1. In the terms that we have already defined, for a state i to be transient there exists another state that is accessible from $state\ j$, but the two states do not communicate and therefore are not in the same class.

Step 4 Compute Stationary Distribution and Compare it with Empirical Distribution.

Steady State Probability -For this step, stationary probability distribution and mean return time can be obtained for Markov process probability values. Stationary probability distribution will describe

the behaviour of energy in long term forecasting where the chain is sufficient for a long period of time with steady-state probabilities that are independent from initial conditions.

Given we have taken a finite State chain, we find each state is communicating to almost every class at some state and chain happens to be irreducible markov chain. Also, chain is persistent or recurrent if $f(ii) = 1$

$$f(ii) = \sum_{0}^{N} f_{ii}(n) = 1 \ldots \text{eq}(3) \text{ where f[ii] is the probability of 1st return to state i if it begins at i}$$

Also, since chain re-starts again infinitely often. We then want to compute the expected number of times it returns to original state. We can
$E(I_n = 1)$ $when\ state\ i\ returns\ back\ to\ where\ it$ started. We can also compute if chain is recurrent if we know P_{ii}-Probability of transition from state i to i

$$\sum_{i=1}^{\alpha} P_{ii}[N] = Infinity..eq(4)$$

Considering chain is persistent, we find a probability vector which when passed through n state transition matrix will result in same probability Vector. This means

$$W * P[N] = W..Eq(5)$$

$where\ N \epsilon R$,P[N] is N step Probability Matrix which is row stochastic.

We can derive individual probability for each final state alternatively using below equation

$$v_{j=} \sum_{i} P_{ij}[n] * v_I \ldots eq(6)$$

We compute steady state probability vector by iterating over different steps until there is no change is Probability Vector. Hence at steady state W[n+1]=W[N]

Step 5 Creating a new Model from Markov Simulation and Comparison with original Model

Since we have a transition matrix T with $P_{ij}[n]$=x where 0 < x<1, we can simulate random number and if random number falls in the range of probability it can create a target state based on random number generation. This will help us evaluate other possibilities of different states during different point of time.

We then simulate a new model and compare the two models

Step 6 Auto-Correlation Comparison between original model and simulated model

We then compute the autocorrelation model between the simulated model and the original Markov model and compute the autocorrelation for each state. Auto-correlation for each state for a given model is computed as below. Here $\bar{x}$ represents the average value of state in the discrete time series markov state model.

$$r[k] = \frac{\sum_1^{n-k}(x_i - \bar{x})(x_{i+k} - \bar{x})}{\sum_1^N (x_i - \bar{x})^2} \ldots eq(7)$$

We then compare the autocorrelation of each state for the original discrete markov model as compared to original model.

Step 7 Chi Square Analysis of Model

The chi square analysis is used to evaluate how two step model $P_{IJ}[2]$ compares with actual count of 2 step jumps in the Actual Markov Chain. This is denoted by N[i,j][2].We compute this by counting the Observed Counts from jumps from different states in two steps. We then compare this with two state probability matrix denoted as P[2]. P[2] can be derived using the below equation

$$P[2] = P^2 \ \ eq(8)$$

$P[2]$ $r$epresent the two step probability matrix.

H_o: Null Hypothesis – The expected Model probabilities $p_1p_2p_3p_4p_5p_6$ is a good model for N[I,J][2]

H_1: Alternate Hypothesis – At-least one state probability is different

We then compute the observed and expected frequency gap between the two to check if the-two step transition model represent the model. In case the value is less than 0.05 we reject the null hypothesis that Observed model is good model to the expected model.

Statistically p-value is the probability of obtaining a result as extreme as, or more extreme than, the result actually obtained when the null hypothesis is true. We can also say p value is the probability value that we reject the null hypothesis given the null is true. This happens when p value is less than 0.05.

5. Case Study

We now show case our experimental findings from the UCI Data Set on Energy Appliance Data. We have used R studio to build Markov Models

Step 1: Count of Discrete State Value

We generate different States from time-series data. The count of different States are as below. We generate the below by counting the discrete counts of time-series data

State	1	2	3	4	5	6
Count	17761	1330	138	12	2	492

Table 2- Frequency Count-Occurrence of States

Step 2 and Step 3: Stochastic Matrix and Empirical Distribution

Stochastic Transition Matrix

The following is the transition Matrix that has been arrived for appliance energy. We have named energy state 6 as “Very-Very High Energy State” and State 1 as Very Low Energy state.

Energy State
A 6 - dimensional discrete Markov Chain defined by the following states:
VL, L, M, H, VH, VVH
The transition matrix (by rows) is defined as follows:

	VL	L	M	H	VH	VVH
VL	0.96869546	0.02083216	0.001463882	5.630314e-05	0.00000000	0.008952199
L	0.31729323	0.54736842	0.025563910	7.518797e-04	0.00000000	0.109022556
M	0.05797101	0.26086957	0.246376812	3.623188e-02	0.00000000	0.398550725
H	0.00000000	0.08333333	0.333333333	0.000000e+00	0.00000000	0.583333333
VH	0.00000000	0.00000000	0.500000000	5.000000e-01	0.00000000	0.000000000
VVH	0.25458248	0.39714868	0.079429735	8.146640e-03	0.00407332	0.256619145

Table 3: 6 State Transition Matrix for Energy Consumption: VL denotes 1 and VVH denotes 6

The markov State Model as derived after creating a transition Matrix for original time series discrete finite state model.

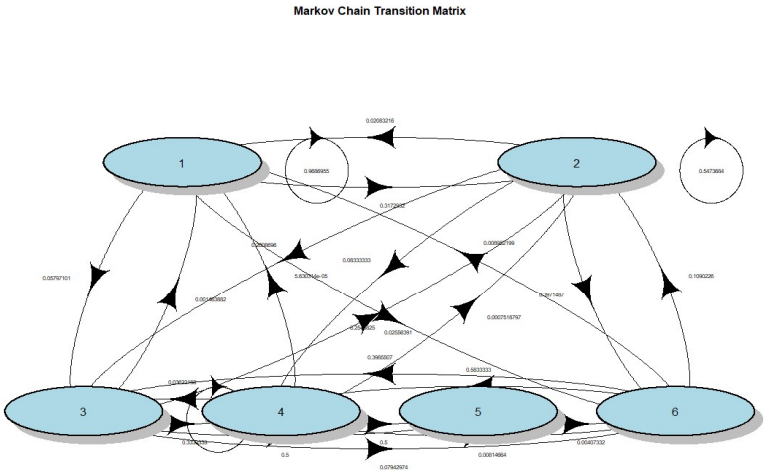

Figure 2- Original Markov Transition Model

Step 4

Comparison of Frequency of Occurrence of States in Markov State Model as compared to Simulated Model. We see a similarity in occurrence of states in the original Empirical model and simulated model

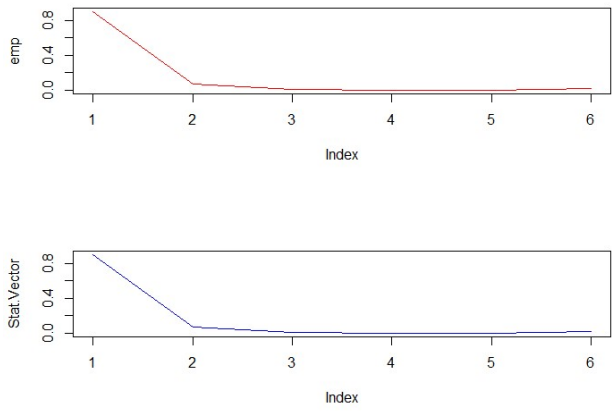

Figure 3 Comparison of Steady State Values of Different States with Empirical Distribution [Top figure shows -empirical distribution and bottom shows steady state probability]

Step 5 Simulated Time Series Vs Original Series

We can find some differences between original and simulated time series model as explained in Step 4 section of the paper. The simulated time series have been derived by random generator and if the random generator has a probability less than p it chooses the original path and if it is more than that it chooses the alternative path.

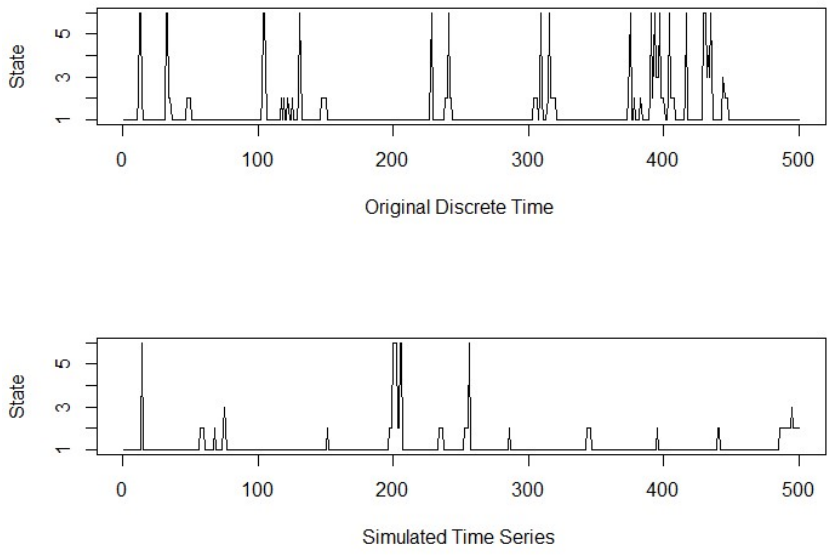

Figure 4-Original and Simulated Series

Step 6 Comparison of Auto Correlation Model

We compute Auto-Correlation Coefficients for Different States of Original and Simulated Time-series data. This has been shown in table 4 using equation 7

Series\,State	1	2	3	4	5	6
Original Time Series	1.0	0.4629	0.2748	0.1956	0.1628	0.150
Simulated Series	1.00	0.44664711	0.28257130	0.19759903	0.133	.08780

Table 4- Auto-Correlation Comparison between Simulated model and Original Model

Step 7 – Chi-Square Analysis -p values for each state & Two Step Expected Probabilities

We compute two step Transition Matrix P[2] as below and get value as below

```
#Two Step Transition Matrix for Original Time Series
p22=tranistionmatrix%^%2
p22
```

```
##            [,1]       [,2]        [,3]         [,4]         [,5]       [,6]
## [1,] 0.94734473 0.03552481 0.003041112 0.0001961734 3.646517e-05 0.01385671
## [2,] 0.51027401 0.35625176 0.029665971 0.0022438159 4.440837e-04 0.10112036
## [3,] 0.25467515 0.36957485 0.111189315 0.0123729516 1.623425e-03 0.25056430
## [4,] 0.19427122 0.36424062 0.130589942 0.0168921577 2.376103e-03 0.29162996
## [5,] 0.02898551 0.17210145 0.289855072 0.0181159420 0.000000e+00 0.49094203
## [6,] 0.44256085 0.34600579 0.055230394 0.0073180743 1.045292e-03 0.14783961
```

Table 5: Two Step Transitional Probabilities

The value below in the table shows if the there is statistically significant difference between observed frequency and 2 state transition probabilistic model shown above. The observed 2 state jumps are computed by counting number of jumps from state i, j in two steps where $i, j \epsilon \{1 \cdot \ldots 6\}$

Series\State	1	2	3	4	5	6
p-value	0.2243	0.2243	0.2424	0.2851	0.3062	0.2243
Chi-Square	30	30	24	12	6	30

Table 6- Chi-square Test Value to compare observed 2 state jumps to Probabilistic Model

6. Conclusion

In this paper, a markov-chain based prediction model is developed to predict the state of energy power consumption. This model can be used in energy forecasting. However, unlike other models, this model would predict state and not the absolute values of energy consumption. We have focussed on single step probabilistic model as well as two step transition model. This is particularly useful when energy and utility companies levy tariff from customers with different consumption level.

Based on the over all results-it can be said that if customer has low or very low energy consumption, would continue to have low /very low energy consumption. As a future study, this model needs to be compared with different seasons to create a bi-variate Finite Markov Chain and see any correlations between them.

References

[1] Winston, W.L.; Goldberg, J.B. Operations Research: Applications and Algorithms; Thomson Brooks/Cole: Belmont, CA, USA, 2004.

[2]Chatfield, C. Statistical Inference Regarding Markov Chain Models. J. R. Stat. Society. Ser. C (Appl. Stat.) 1973, 22, 7–20.

[3] Markov Chain Model Development for Forecasting Air Pollution Index of Miri, Sarawak

[4] https://archive.ics.uci.edu/ml/datasets/Appliances+energy+prediction

Appendix

```
data=read.csv('C:\\Project\\NEU\\Probability\\Markov Chain\\energydata_com
plete (1).csv')
x=data$Appliances
summary(x)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10.00   50.00   60.00   97.69  100.00 1080.00
```

```
for(i in 1:length(x)){
  if (x[i]<200){
    data$state[i]=1
  }
  else if ((x[i]<400) & (x[i]>200)){
      data$state[i]=2
    }
  else if ((x[i]<800) & (x[i]>600)){
      data$state[i]=3
    }
  else if((x[i]<1000) &(x[i]>800)){
      data$state[i]=4
  }
  else if ((x[i]<1200) & (x[i]>1000)){
      data$state[i]=5
    }
  else {
      data$state[i]=6
    }
}

dfnew<-data.frame(data$state,data$Appliances)
library(markovchain)
```

```
## Warning: package 'markovchain' was built under R version 3.6.3
```

```
## Package:  markovchain
## Version:  0.8.5-2
## Date:     2020-09-07
## BugReport: https://github.com/spedygiorgio/markovchain/issues
```

```
library(diagram)
```

```
## Warning: package 'diagram' was built under R version 3.6.3
```

```
mcFit <- markovchainFit(data=data$state)
Frequency=table(dfnew$data.state)
Frequency
```

```
##
##     1     2     3     4     5     6
## 17761  1330   138    12     2   492
```

```r
emp=rep(0,6)
for(i in (1:6)){
  emp[i]=Frequency[i]/sum(Frequency)
}
emp
```

```
## [1] 0.8999746643 0.0673929567 0.0069926526 0.0006080568 0.0001013428
## [6] 0.0249303268
```

```r
mcFit <- markovchainFit(data=data$state)
mcFit
```

```
## $estimate
## MLE Fit
##  A  6 - dimensional discrete Markov Chain defined by the following stat
es:
##  1, 2, 3, 4, 5, 6
##  The transition matrix  (by rows)  is defined as follows:
##            1          2           3            4          5           6
## 1 0.96869546 0.02083216 0.001463882 5.630314e-05 0.00000000 0.008952199
## 2 0.31729323 0.54736842 0.025563910 7.518797e-04 0.00000000 0.109022556
## 3 0.05797101 0.26086957 0.246376812 3.623188e-02 0.00000000 0.398550725
## 4 0.00000000 0.08333333 0.333333333 0.000000e+00 0.00000000 0.583333333
## 5 0.00000000 0.00000000 0.500000000 5.000000e-01 0.00000000 0.000000000
## 6 0.25458248 0.39714868 0.079429735 8.146640e-03 0.00407332 0.256619145
```

```r
library(msm)
```

```
## Warning: package 'msm' was built under R version 3.6.3
```

```r
TableJumpt=statetable.msm(dfnew$data.state,dfnew)
```

```
## Warning in prevsubj != subject: longer object length is not a multiple
of
## shorter object length
```

```r
TableJumpt
```

```
##     to
## from     1     2     3     4     5     6
##    1 17205   370    26     1     0   159
##    2   422   728    34     1     0   145
##    3     8    36    34     5     0    55
##    4     0     1     4     0     0     7
##    5     0     0     1     1     0     0
##    6   125   195    39     4     2   126
```

```r
p=matrix(, nrow = 6, ncol = 6)
for (j in (1:6)){
  x[j]=sum(TableJumpt[j,])
  for (i in(1:6)){
     p[j,i]=TableJumpt[j,i]/x[j]
  }
}
```

```r
par(mfrow=c(2,1))
state=matrix(c(1.0, 0.0, 0.0,0,0,0),nrow=1,ncol=6)
state
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    0    0    0    0    0
```

```r
p
```

```
##            [,1]       [,2]        [,3]         [,4]       [,5]        [
,6]
## [1,] 0.96869546 0.02083216 0.001463882 5.630314e-05 0.00000000 0.008952
199
## [2,] 0.31729323 0.54736842 0.025563910 7.518797e-04 0.00000000 0.109022
556
## [3,] 0.05797101 0.26086957 0.246376812 3.623188e-02 0.00000000 0.398550
725
## [4,] 0.00000000 0.08333333 0.333333333 0.000000e+00 0.00000000 0.583333
333
## [5,] 0.00000000 0.00000000 0.500000000 5.000000e-01 0.00000000 0.000000
000
## [6,] 0.25458248 0.39714868 0.079429735 8.146640e-03 0.00407332 0.256619
145
```

```r
par(mfrow=c(1,1))
plotmat(p,pos = c(2,4),
        lwd = 1, box.lwd = 2,
        cex.txt = 0.5,
        box.size = 0.10,
        box.type = "circle",
        box.prop = 0.3,
        box.col = "light blue",
        arr.length=.5,
        arr.width=.3,
        self.cex = .4,
        self.shifty = -.01,
        self.shiftx = .17,
        main = "Markov Chain Transition Matrix")
```

```r
sum(p[2,])
```

```
## [1] 1
```

```r
stateHist=state
dfStateHist=data.frame(state)
state2=matrix(data=NA,nrow=50,ncol=6)
for (i in (1:50)){
  state2[i,]=state%*%p
    state=state2[i,]
  }
```

```r
#Simulation
```

```
tranistionmatrix=p
byRow=TRUE
EnergyMarkov2 <- new("markovchain", states = c("VL", "L", "M","H","VH","VV
H"),transitionMatrix = tranistionmatrix,name = "Energy State")
EnergyMarkov2

## Energy State
##  A  6 - dimensional discrete Markov Chain defined by the following stat
es:
##  VL, L, M, H, VH, VVH
##  The transition matrix  (by rows)  is defined as follows:
##             VL          L           M            H         VH         V
VH
## VL  0.96869546 0.02083216 0.001463882 5.630314e-05 0.00000000 0.0089521
99
## L   0.31729323 0.54736842 0.025563910 7.518797e-04 0.00000000 0.1090225
56
## M   0.05797101 0.26086957 0.246376812 3.623188e-02 0.00000000 0.3985507
25
## H   0.00000000 0.08333333 0.333333333 0.000000e+00 0.00000000 0.5833333
33
## VH  0.00000000 0.00000000 0.500000000 5.000000e-01 0.00000000 0.0000000
00
## VVH 0.25458248 0.39714868 0.079429735 8.146640e-03 0.00407332 0.2566191
45

#Simulated Markocv Chain
EnergyState <- rmarkovchain(n = length(data$Appliances), object = EnergyMa
rkov2, t0 = "VL")
# conversion to factor
EnergyState=as.factor(as.character(EnergyState))
sim1=EnergyState[1:19200]
sim2=c()
#Converting simulation to 0-4 states with numerical valiues
for(i in (1:length(sim1))){
  if (sim1[i]=='VL'){
    sim2[i]=1
  } else if (sim1[i]=='L'){
    sim2[i]=2
  } else if(sim1[i]=='M'){
    sim2[i]=3
  } else if (sim1[i]=='H'){
    sim2[i]=4
  } else if(sim1[i]=='VH')
    sim2[i]=5
    else {
      sim2[i]=6
    }
}

# Calcualtion of x.bar
frequency.sim=table(sim2)
frequency.sim
```

```
## sim2
##     1     2     3     4     5     6
## 17181  1338   154    10     1   516

xbar1=frequency.sim[1]
xbar2=frequency.sim[2]
xbar3=frequency.sim[3]
xbar4=frequency.sim[4]
xbar5=frequency.sim[5]
xbar6=frequency.sim[6]
frequency.sim[1]

##     1
## 17181

x.bar=(1*xbar1+2*xbar2+3*xbar3+4*xbar4+5*xbar5+6*xbar6)/sum(frequency.sim)
x.bar

##        1
## 1.221875

# Calculation of xbar
sim3=sim2[1:500]
plot(1:length(sim3),sim3,type='l',ylab='State',xlab="Simulated States")
dfnew2.state<-dfnew$data.state[1:500]
plot(1:length(dfnew2.state),dfnew2.state,type='l',xlab="Original Discrete
Time ",ylab="State")
```

```
plot(1:length(sim3),sim3,type='l',xlab="Simulated Time Series ",ylab='Stat
e')
#Correlation Coefficient
x1=Frequency[1]
x2=Frequency[2]
x3=Frequency[3]
x4=Frequency[4]
x5=Frequency[5]
x6=Frequency[6]
xbar=(1*x1+2*x2+3*x3+4*x4+5*x5+6*x6)/(sum(Frequency))
xbar

##        1
## 1.208259

length(dfnew$data.state)

## [1] 19735

k=6
p=matrix(0,nrow = length(dfnew$data.Appliances),ncol = 6)
d=matrix(0,nrow = length(dfnew$data.Appliances),ncol = 6)
r=rep(NA,6)
#Autocorrelation for original timeseries
for (j in (1:k)){
  for (i in (1:(length(data$Appliances)-(j-1)))){
```

```
    p[i,j]=(dfnew$data.state[i]-xbar)*(dfnew$data.state[i+j-1]-x.bar)
    d[i,j]=(dfnew$data.state[i]-xbar)*(dfnew$data.state[i]-x.bar)
  }
  r[j]=(sum(p[,j]))/(sum(d[,j ]))

}
r
```

```
## [1] 1.0000000 0.4629951 0.2748963 0.1956352 0.1689551 0.1503227
```

```
#Autocorrelation for Simulated Series
p1=matrix(0,nrow = length(sim2),ncol = 6)
d1=matrix(0,nrow = length(sim2),ncol = 6)
r1=rep(NA,6)
r1
```

```
## [1] NA NA NA NA NA NA
```

```
#Autocorrelation
for (j in (1:k)){
  for (i in (1:(length(sim2)-(j-1)))){
    p1[i,j]=(sim2[i]-x.bar)*(sim2[i+j-1]-x.bar)
    d1[i,j]=(sim2[i]-x.bar)*(sim2[i]-x.bar)
  }
  r1[j]=(sum(p1[,j]))/(sum(d1[,j ]))

}
r1
```

```
## [1] 1.0000000 0.4467976 0.2930287 0.2056178 0.1543828 0.1189881
```

```
#Two Step Transition Matrix for Original Time Series
p22=tranistionmatrix%^%2
p22
```

```
library(expm)
w2=p22
w2
```

```
##            [,1]       [,2]        [,3]         [,4]         [,5]
[,6]
## [1,] 0.94734473 0.03552481 0.003041112 0.0001961734 3.646517e-05 0.0138
5671
## [2,] 0.51027401 0.35625176 0.029665971 0.0022438159 4.440837e-04 0.1011
2036
## [3,] 0.25467515 0.36957485 0.111189315 0.0123729516 1.623425e-03 0.2505
6430
## [4,] 0.19427122 0.36424062 0.130589942 0.0168921577 2.376103e-03 0.2916
2996
## [5,] 0.02898551 0.17210145 0.289855072 0.0181159420 0.000000e+00 0.4909
4203
## [6,] 0.44256085 0.34600579 0.055230394 0.0073180743 1.045292e-03 0.1478
3961
```

```r
#Let N be the numer of states in the original observation
# Actual Jumps 2 States
alpha=0
count=matrix(data=0,nrow=6,ncol=6)
count
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    0    0    0    0    0    0
## [2,]    0    0    0    0    0    0
## [3,]    0    0    0    0    0    0
## [4,]    0    0    0    0    0    0
## [5,]    0    0    0    0    0    0
## [6,]    0    0    0    0    0    0
```

```r
count[[1,2]]
```

```
## [1] 0
```

```r
r=rep(0,length(dfnew$data.state))
for (l in (1:(length(dfnew$data.state)-2))){
     r=dfnew$data.state[l+1]
     m=dfnew$data.state[l+2]
     p=dfnew$data.state[l]
     if ((p!=r) &(r!=m)){
       count[[p,m]]=count[[p,m]]+1
       count[p,m]=count[[p,m]]
     }
}
count
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]  145   63   45    2    0   89
## [2,]   53   67   13    5    1   36
## [3,]   16   41    4    0    0    3
## [4,]    3    4    0    0    0    3
## [5,]    0    1    0    0    0    1
## [6,]   69   20    6    3    0   32
```

```r
count[4,]
```

```
## [1] 3 4 0 0 0 3
```

```r
w2[4,]
```

```
## [1] 0.194271224 0.364240618 0.130589942 0.016892158 0.002376103 0.29162
9956
```

```r
chisq.test(count[1,],w2[1,])
```

```r
chisq.test(count[2,],w2[2,])
```

```r
chisq.test(count[3,],w2[3,])
```

```r
chisq.test(count[4,],w2[4,])
```

```r
chisq.test(count[5,],w2[5,])
```

```r
chisq.test(count[6,],w2[6,])
```