Ajay Kc
013213328
Project 6 Report

1. A three-state Markov Chain

Introduction

The experiment revolves around a three-state Markov chain. A state transition matrix and the initial probability distribution is given. The experiment aims to compare the experimental probabilities with the probabilities obtained through the state transition matrix on three states: R,N, and S.

Methodology

Firstly, a random number is generated to estimate which state is picked first based on their initial probability. Then, another random number is generated to estimate which state does it transition to based on the transition probability. The process is repeated 15 times and different state is recorded. The entire experiment is repeated 10,000 times and the probability of getting different state on various step is calculated. Out of 10,000 instances, two different simulations are plotted. Secondly, the experimental probabilities of different state is also plotted. Finally, the transition matrix is used to calculate the probability and is plotted as well.

Source code

```
"""
Ajay Kc
013213328
EE381
Project 6 Part 1
The experiment deals with three state Markov Chain where
experimental probability and theoretical probability is
plotted.
"""
import numpy as np
import matplotlib.pyplot as plt

step = 15
number = 10000
transition_matrix =
[[1/3,1/3,1/3],[1/2,0,1/2],[1/4,1/4,1/2]]
initial_probability = [1/4,1/2,1/4]

X = np.zeros((step+1,number), dtype = int)
S = np.zeros((step+1), dtype=int)
experimental_probability = np.zeros((step+1,3))
theoretical_probability = np.zeros((step+1,3))
theoretical_probability[0,:]=initial_probability
```

```
def simulate_markov_chain():

    probability = np.random.rand()
    state = 0
    if(probability<=initial_probability[0]):
        state = 0
    elif((probability
>initial_probability[0])and(probability<=initial_probabilit
y[0]+initial_probability[1])):
        state = 1
    else:
        state = 2

    S[0]=state

    for s in range(step):
        state=S[s]
        transition_probability=np.random.rand()
        if(state == 0):

if(transition_probability<=transition_matrix[0][0]):
                state = 0

elif((transition_probability>transition_matrix[0][0])and(tr
ansition_probability<=transition_matrix[0][0]+transition_ma
trix[0][1])):
                state = 1
            else:
                state = 2
        elif(state == 1):
            if (transition_probability <=
transition_matrix[1][0]):
                state = 0
            else:
                state = 2
        else:
            if (transition_probability <=
transition_matrix[2][0]):
                state = 0
            elif ((transition_probability >
transition_matrix[2][0]) and (transition_probability <=
transition_matrix[2][1])):
                state = 1
            else:
                state = 2
        S[s+1]=state
```

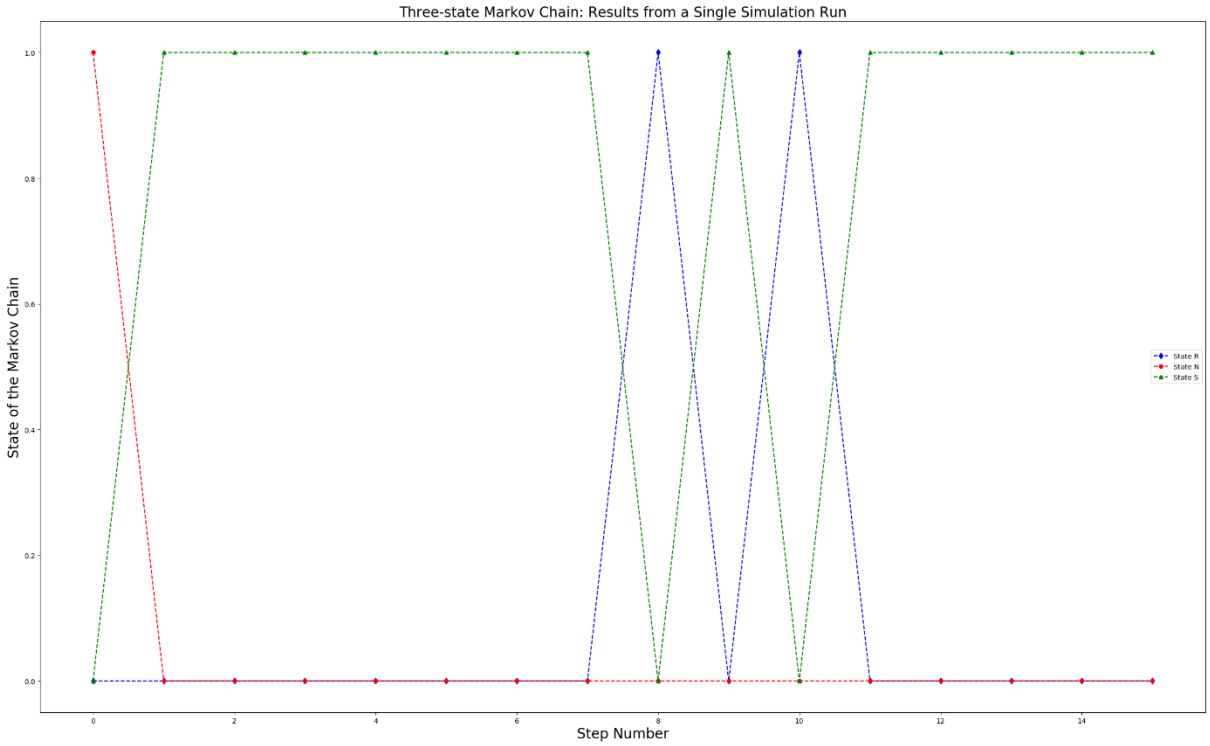
```
    return S

def calculate_experimental_probabilities():
    for i in range(step+1):

experimental_probability[i][0]=X[i,:].tolist().count(0)/num
ber
        experimental_probability[i][1] = X[i,
:].tolist().count(1)/number
        experimental_probability[i][2] = X[i,
:].tolist().count(2)/number

def calculate_theoretical_probabilities():
    for i in range(step):

theoretical_probability[i+1,:]=np.matmul(theoretical_probab
ility[i,:],transition_matrix)

def
plotGraph(state_one,state_two,state_three,xlabel,ylabel,tit
le):

    plt.xlabel(xlabel, fontsize=20)
    plt.ylabel(ylabel, fontsize=20)
    plt.title(title, fontsize=20)

    plt.figure(1)

plt.plot(range(0,step+1),state_one,color='b',marker='d',lin
estyle='--', label='State R')
    plt.plot(range(0, step + 1), state_two, color='r',
marker='H', linestyle='--', label='State N')
    plt.plot(range(0, step + 1), state_three, color='g',
marker='^', linestyle='--', label='State S')
    plt.legend(loc='best')

    plt.show()

for n in range(number):
    X[:,n]=simulate_markov_chain()

simulation_one = X[:,0]
simulation_two = X[:,1]

r_simulation_one=np.zeros(step+1,dtype=int)
n_simulation_one=np.zeros(step+1,dtype=int)
```

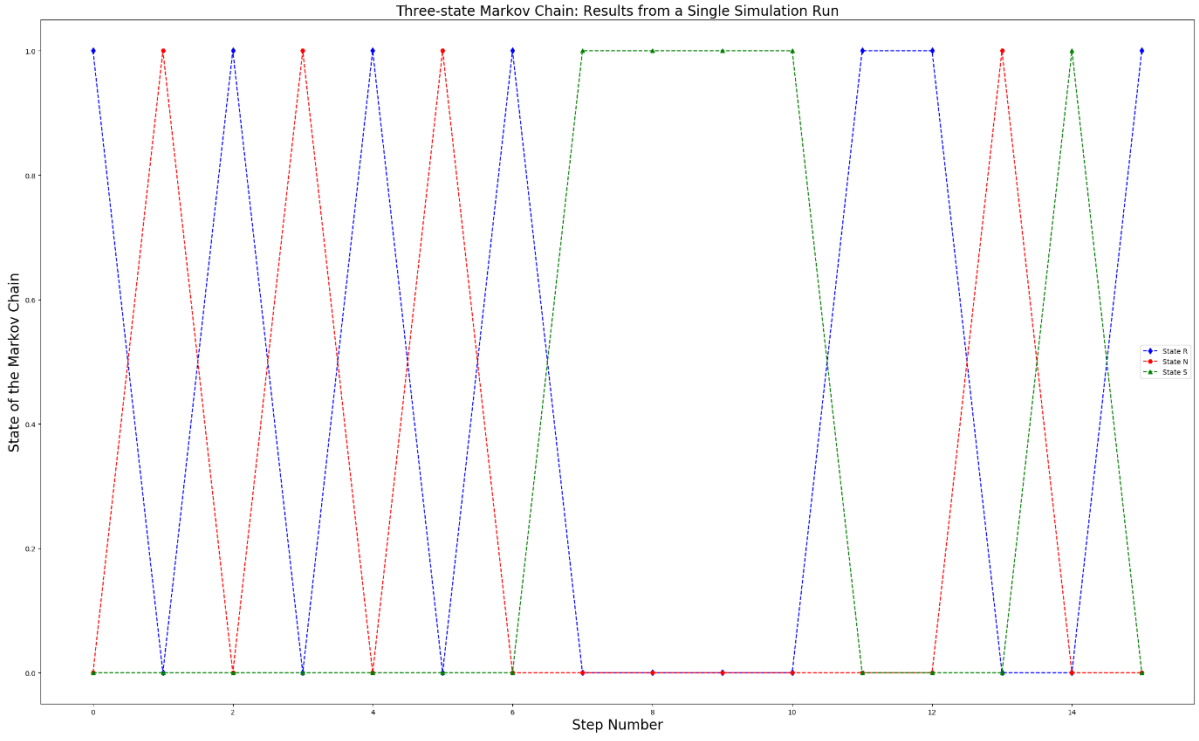
```
s_simulation_one=np.zeros(step+1,dtype=int)
r_simulation_two=np.zeros(step+1,dtype=int)
n_simulation_two=np.zeros(step+1,dtype=int)
s_simulation_two=np.zeros(step+1,dtype=int)

for i in range(step+1):
    if(simulation_one[i]==0):
        r_simulation_one[i]=1
    elif(simulation_one[i]==1):
        n_simulation_one[i]=1
    else:
        s_simulation_one[i]=1

    if (simulation_two[i] == 0):
        r_simulation_two[i] = 1
    elif (simulation_two[i] == 1):
        n_simulation_two[i] = 1
    else:
        s_simulation_two[i] = 1

calculate_experimental_probabilities()
calculate_theoretical_probabilities()

plotGraph(r_simulation_one,n_simulation_one,s_simulation_on
e,'Step Number','State of the Markov Chain',
          'Three-state Markov Chain: Results from a Single
Simulation Run')
plotGraph(r_simulation_two,n_simulation_two,s_simulation_tw
o,'Step Number','State of the Markov Chain',
          'Three-state Markov Chain: Results from a Single
Simulation Run')
plotGraph(experimental_probability[:,0],experimental_probab
ility[:,1],experimental_probability[:,2],'Step Number',
          'Probability','Three-state Markov Chain: Results
from N=10000 Single Simulation Runs')
plotGraph(theoretical_probability[:,0],theoretical_probabil
ity[:,1],theoretical_probability[:,2],'Step Number',
          'Probability','Three-state Markov Chain:
Calculated Using the State Transition Matrix')
```

Ajay Kc
013213328
Project 6 Report

Plots

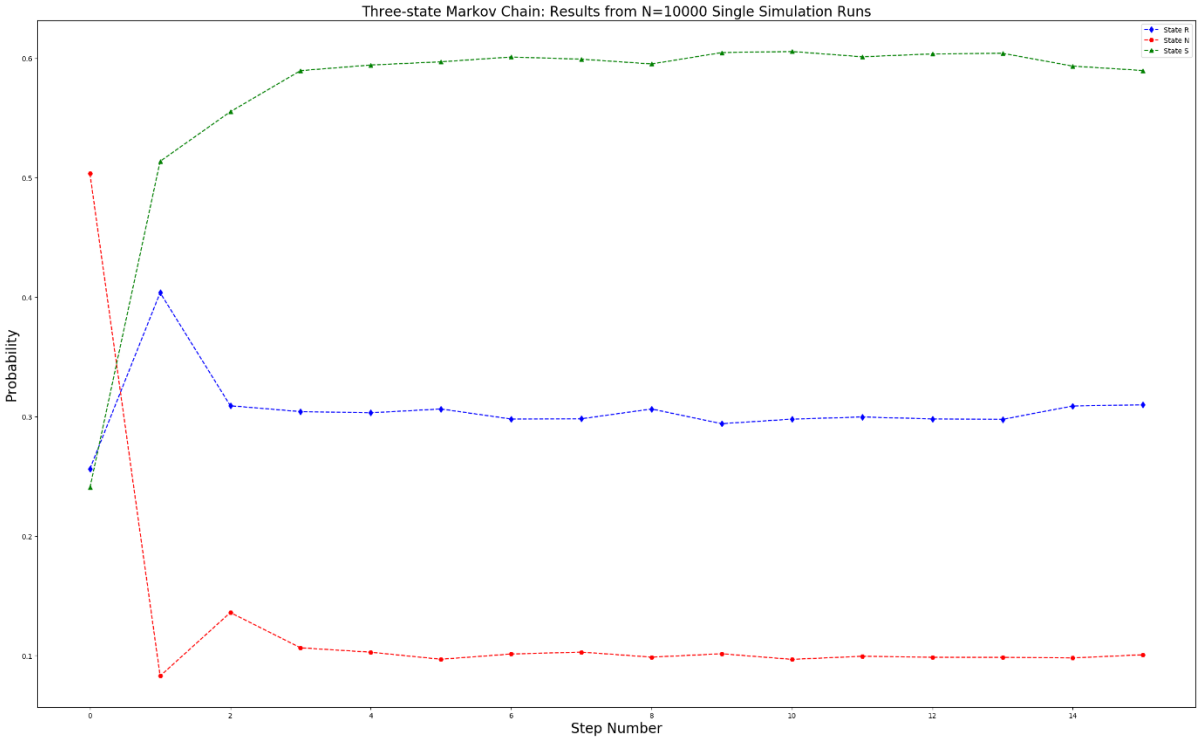
Plots showing two different single simulation runs for step-size = 15



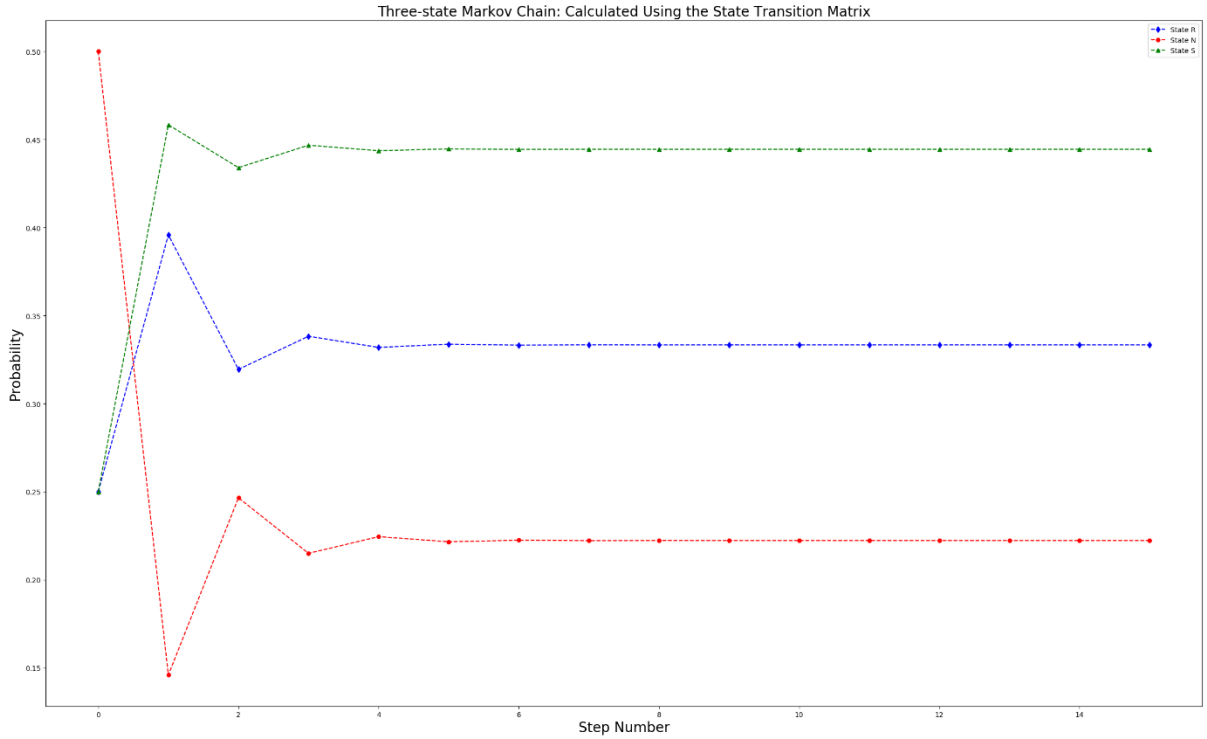


Ajay Kc
013213328
Project 6 Report

Plot with the experimental probabilities for the states R,N,S



Plot with the probabilities obtained through the state transition matrix



Ajay Kc
013213328

2. The Google PageRank Algorithm

Introduction

The experiment revolves around 5-state Markov chain represented in an individual's probability of visiting five different webpages: A, B, C, D, E. A State Transition Matrix(P) has to be calculated based on which webpages can be reached from the current page. After State Transition Matrix is calculated, the probability of visiting each webpage is calculated over the span of 20 steps. The experiment has to instances; first, where each webpage has equal probability of being visited first, and the second one where, E is treated as the home page. The objective of the experiment is to rank the webpages and find the probability of each webpage being visited.

Methodology

First, the State Transition Matrix is calculated and used to find the probability of each page being visited. Firstly, the probability is calculated when each webpage has equal chances of being visited first. Then, the probability is calculated when page 'E' is treated as homepage. For each of the two scenarios, the probability is plotted and then webpages are ranked based on visiting importance.

State Transition Matrix

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \\ 1/3 & 1/3 & 0 & 0 & 1/3 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1/3 & 1/3 & 1/3 & 0 \end{bmatrix}$$

Source Code

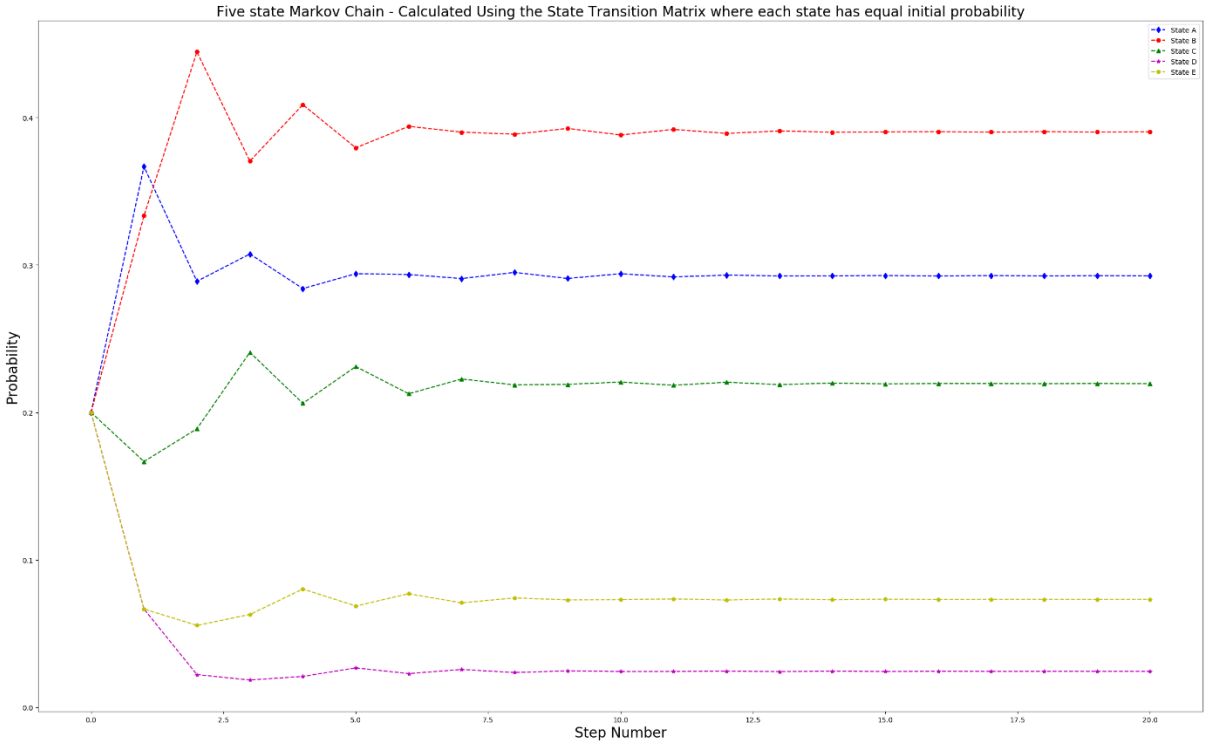
```
"""
Ajay Kc
013213328
EE381
Project 6 Part 2
The experiment deals with five state Markov Chain where
theoretical probability is plotted and the probability of
each
page being visited is calculated.
"""
import numpy as np
import matplotlib.pyplot as plt
```

```
step = 20
transition_matrix = [[0,1,0,0,0],
                     [1/2,0,1/2,0,0],
                     [1/3,1/3,0,0,1/3],
                     [1,0,0,0,0],
                     [0,1/3,1/3,1/3,0]]
v1 = [1/5,1/5,1/5,1/5,1/5]
v2 = [0,0,0,0,1]
theoretical_probability_one = np.zeros((step+1,5))
theoretical_probability_two = np.zeros((step+1,5))
theoretical_probability_one[0,:]=v1
theoretical_probability_two[0,:]=v2

def calculate_theoretical_probabilities():
    for i in range(step):

theoretical_probability_one[i+1,:]=np.matmul(theoretical_pr
obability_one[i,:],transition_matrix)
        theoretical_probability_two[i + 1, :] =
np.matmul(theoretical_probability_two[i, :],
transition_matrix)

def
plotGraph(state_one,state_two,state_three,state_four,state_
five,xlabel,ylabel,title):

    plt.xlabel(xlabel, fontsize=20)
    plt.ylabel(ylabel, fontsize=20)
    plt.title(title, fontsize=20)

    plt.figure(1)

plt.plot(range(0,step+1),state_one,color='b',marker='d',lin
estyle='--', label='State A')
    plt.plot(range(0, step + 1), state_two, color='r',
marker='H', linestyle='--', label='State B')
    plt.plot(range(0, step + 1), state_three, color='g',
marker='^', linestyle='--', label='State C')
    plt.plot(range(0, step + 1), state_four, color='m',
marker='*', linestyle='--', label='State D')
    plt.plot(range(0, step + 1), state_five, color='y',
marker='p', linestyle='--', label='State E')
    plt.legend(loc='best')

    plt.show()
```

```
calculate_theoretical_probabilities()

plotGraph(theoretical_probability_one[:,0],theoretical_prob
ability_one[:,1],theoretical_probability_one[:,2],
          theoretical_probability_one[:,
3],theoretical_probability_one[:,4],"Step
Number","Probability",
          "Five state Markov Chain - Calculated Using the
State Transition Matrix where each state has equal initial
probability")


plotGraph(theoretical_probability_two[:,0],theoretical_prob
ability_two[:,1],theoretical_probability_two[:,2],
          theoretical_probability_two[:,
3],theoretical_probability_two[:,4],"Step
Number","Probability",
          "Five state Markov Chain - Calculated Using the
State Transition Matrix where one site is the home page")
```

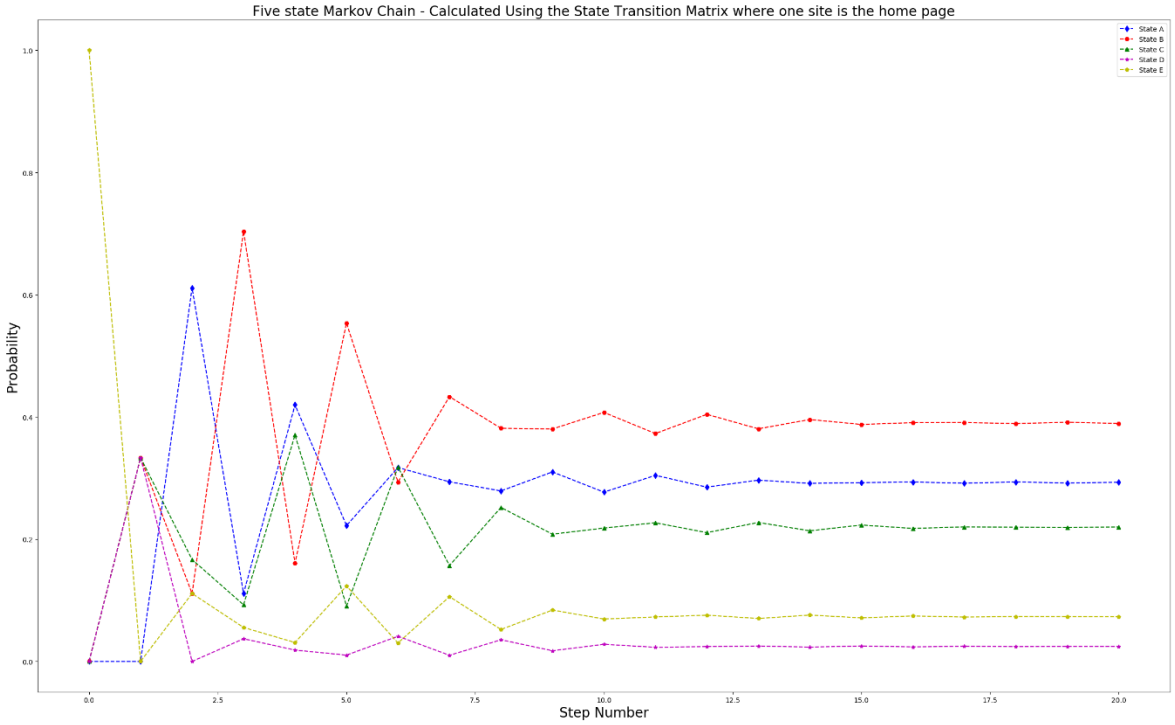
Plot

Plot for five-state Markov chain when each site has equal probability of being visited



Ajay Kc
013213328
Project 6 Report

Plot for five-state Markov chain when site E is the homepage



Results

Initial probability vector: v1		
Rank	Page	Probability
1	B	0.3912290
2	A	0.2933310
3	C	0.2205870
4	E	0.0733394
5	D	0.0244502

Initial probability vector: v1		
Rank	Page	Probability
1	B	0.3922190
2	A	0.2932560
3	C	0.2189320
4	E	0.0753791
5	D	0.0274940