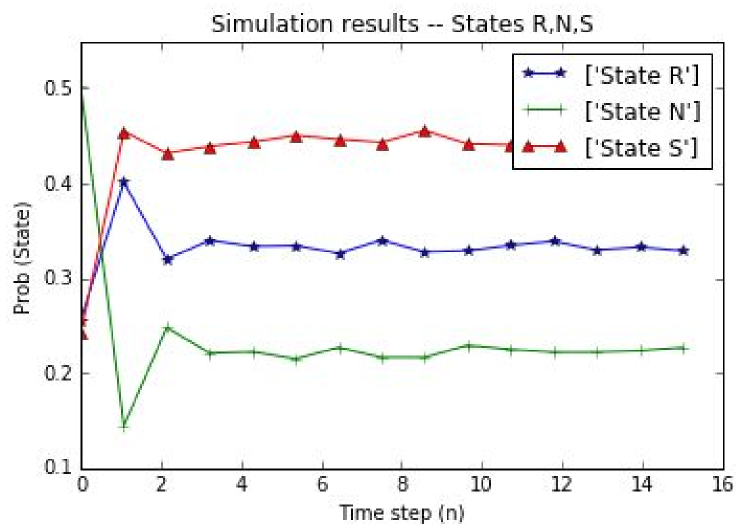
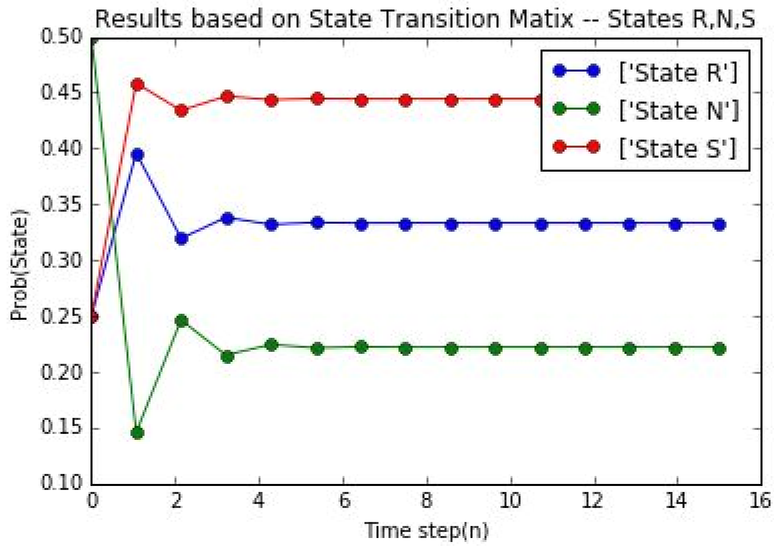
Roberto Sanchez (014587792)
December 07, 2017
EE 381 – Probability and Statistics with Applications to Computing
Lab 06 - Markov Chains

1) A three-state Markov Chain
   a) **INTRODUCTION:**
      i) The section of the project requires us to run the matrix P to get a meaningful statistical data on running the experiment 10,000 times.
   b) **METHODOLOGY:**
      i) We generate values based on what values for probability we have along with its distribution. Once we get that we start to plot the 3 lines on the chart for each state like R, N, S.
   c) **RESULTS AND CONCLUSIONS:**

```
In [1]: runfile('/home/beryl/.spyder2-py3/temp.py', wdir='/home/beryl/.spyder2-
```



      i)




```
In [2]:
```

ii)

2) The Google PageRank Algorithm

a) INTRODUCTION:

i) This is a simulation on how early search engines work back in the late 1990’s. This experiment will use markov chains to find the page ranking based on page interactions.

b) METHODOLOGY:

i) We start with having a matrix of values stored in P and V. The program at that point begins to calculate the markov chain in the matrix to generate values for each page. This page rank determines which pages are more interacted by users.

c) RESULTS AND CONCLUSIONS:

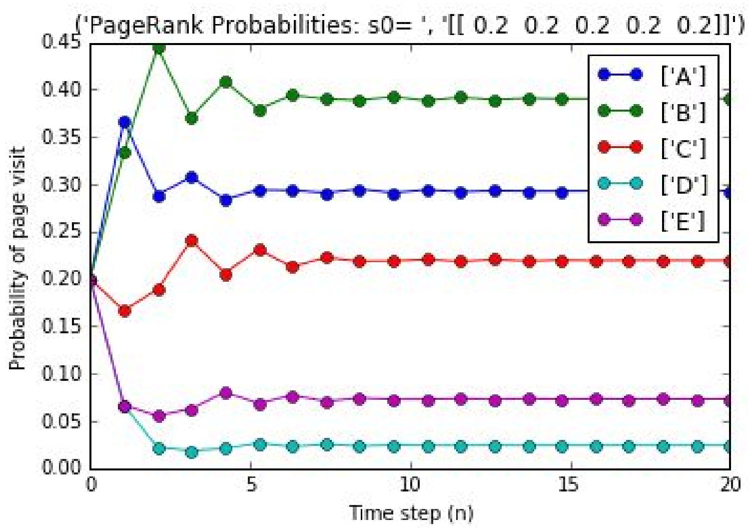
i) The following pages are visible from Rank 1 - 5

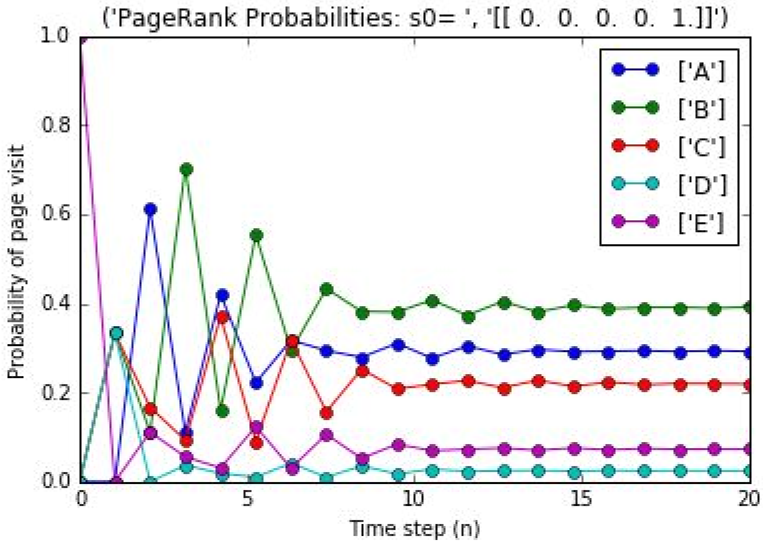
	Intial Probabilty vector V1	
Rank	Page	Probability
1	B	0.4
2	A	0.3
3	C	0.25
4	E	0.7
5	D	0.02

ii) The following pages are visible from Rank 1 - 5

	Initial probabilty vector v2	
Rank	Page	Probability
1	B	0.4
2	A	0.3
3	C	0.2
4	E	0.1
5	D	0.0

```
In [15]: runfile('/home/beryl/.spyder2-py3/temp.py', wdir='/home/beryl/.spyder2-py3')
```




('PageRank Probabilities: s0= ', '[[0. 0. 0. 0. 1.]]')
Probability of page visit
Time step (n)
['A']
['B']
['C']
['D']
['E']
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20

Appendix

1. Markov Chain

```
#-----------------------------------------------
#   Name: Roberto Sanchez
#   Date: December 07, 2017
#   Assignement 6 - Markov Chains
#-----------Importing packages----------------
import numpy as np
import matplotlib.pyplot as plt
import random
# MarkovChain Definition
def MarkovChain(N,n):
    # Delarations
    S = [0 for i in range(0,n)]
    X = np.chararray((n,N))
    X[:] = 0
    M = np.zeros((n,3))
    prob = {'p11' : 1/3, 'p12' : 1/3, 'p13' : 1/3, 'p21' : 1/2, 'p23' :
1/2, 'p22' : 0, 'p31' : 1/4, 'p32' : 1/4, 'p33' : 1/2}
    dist = {'d01' : 1/4, 'd03' : 1/4, 'd02' : 1/2}
    #-------------------------------------------------
    # Calculating Values for S
    for j in range(0,N):
        r0 = np.random.random();
        if r0 <= dist['d01']:
            s0 = 'R'
        elif r0>dist['d01'] and r0<=dist['d01']+dist['d02']:
            s0 = 'N'
        elif r0>dist['d01']+dist['d02']:
            s0 = 'S'
        S[0] = s0
        for k in range(0,n-1):
            s = S[k]
            r = np.random.random();
            if s == 'R':
                if r<=prob['p11']:
                    S[k+1]='R'
                elif r>prob['p11'] and r<=prob['p11']+prob['p12']:
                    S[k+1]='N'
                elif r>prob['p11']+prob['p12']:
                    S[k+1]='S'
            elif s == 'N':
```

```
                if r<=prob['p21']:
                    S[k+1]='R'
                elif r>prob['p21'] and r<=prob['p21']+prob['p22']:
                    S[k+1]='N'
                elif r>prob['p21']+prob['p22']:
                    S[k+1]='S'
            elif s == 'S':
                if r<=prob['p31']:
                    S[k+1]='R'
                elif r>=prob['p31'] and r<=prob['p31']+prob['p32']:
                    S[k+1]='N'
                elif r>prob['p31']+prob['p32']:
                    S[k+1]='S'
        X[:,j]=S
    # Getting M values
    for j in range(0,n):
        ma=0
        mb=0
        mc=0
        x=X[j,:]
        for i in range(N):
            if str(x[i],'utf-8') == 'R':
                ma += 1
        for p in range(N):
            if str(x[p],'utf-8') == 'N':
                mb += 1
        for q in range(N):
            if str(x[q],'utf-8') == 'S':
                mc += 1
        M[j,:] = [ma/N,mb/N,mc/N]

#
---------------------------------Plot------------------------------------
    plt.figure(1)
    nv = np.linspace(0,n,num=15)
    plt.plot(nv,M[:,0],color = 'blue',marker='*',markersize=6)
    plt.plot(nv,M[:,1],color = 'green',marker='+',markersize=6)
    plt.plot(nv,M[:,2],color = 'red',marker='^',markersize=6)
    plt.title('Simulation results -- States R,N,S')
    plt.xlabel('Time step (n)')
    plt.ylabel('Prob (State)')
    plt.legend((['State R'],['State N'],['State S']))
```

```
    plt.show()
    P = 
np.matrix([[prob['p11'],prob['p12'],prob['p13']],[prob['p21'],prob['p22'],p
rob['p23']],[prob['p31'],prob['p32'],prob['p33']]])
    y0 = np.matrix([1/4,1/2,1/4])
    Y = np.zeros((n,3))
    Y[0,:] = y0
    for k in range(0,n-1):
        Y[k+1,:] = np.matmul(Y[k,:],P)
    # Figure 2 Chart
    plt.figure(2)
    plt.plot(nv,Y[:,0],color = 'blue',marker='o',markersize=6)
    plt.plot(nv,Y[:,1],color = 'green',marker='o',markersize=6)
    plt.plot(nv,Y[:,2],color = 'red',marker='o',markersize=6)
    plt.title('Results based on State Transition Matrix -- States R,N,S')
    plt.xlabel('Time step(n)')
    plt.ylabel('Prob(State)')
    plt.legend((['State R'],['State N'],['State S']))
    plt.show()
MarkovChain(10000,15)
```

2. Google Page Rank

```
#----------------------------------------------------------
# Name: Roberto Sanchez
# Date: December 07, 2017
# Assignment 6 - Markov Chains
#----------------------------------------------------------
def PageRank(N,n):
    # Creating character array
    X = np.chararray((n,N))
    X[:] = 0
    P = 
np.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]])
    V = np.matrix([[1/5,1/5,1/5,1/5,1/5],[0,0,0,0,1]])
    # Calculating
    for i in range(0,2):
        s0 = V[i,:]
        Y = np.zeros((n,5))
        Y[0,:] = s0;
```

```
        for j in range(0,n-1):
            Y[j+1,:] = np.matmul(Y[j,:],P)
        nv = np.linspace(0,n,num=20)
        plt.figure()
        plt.plot(nv,Y,marker='o',markersize=6)
        plt.title(('PageRank Probabilities: s0= ',np.str(s0)))
        plt.xlabel('Time step (n)')
        plt.ylabel('Probability of page visit')
        plt.legend((['A'],['B'],['C'],['D'],['E']))
        plt.show
PageRank(10000,20)
```