F1

Tae Hyeon Kwon, undergrad(ITM)

2021-01-25

```
import numpy as np
import pandas as pd


states = np.arange(0,80,10).astype('str')

P_normal = pd.DataFrame(np.matrix([[0,1,0,0,0,0,0,0],
                    [0,0,1,0,0,0,0,0],
                    [0,0,0,1,0,0,0,0],
                    [0,0,0,0,1,0,0,0],
                    [0,0,0,0,0,1,0,0],
                    [0,0,0,0,0,0,1,0],
                    [0,0,0,0,0,0,0,1],
                    [0,0,0,0,0,0,0,1]]), index=states,columns=states)

P_speed=pd.DataFrame(np.matrix([[.1,0,.9,0,0,0,0,0],
                   [.1,0,0,.9,0,0,0,0],
                   [0,.1,0,0,.9,0,0,0],
                   [0,0,.1,0,0,.9,0,0],
                   [0,0,0,.1,0,0,.9,0],
                   [0,0,0,0,.1,0,0,.9],
                   [0,0,0,0,0,.1,0,.9],
                   [0,0,0,0,0,0,0,1]]), index=states, columns=states)

R_s_a=pd.DataFrame(np.matrix([-1,-1,-1,-1,0.0,-1,-1,0,-1.5,-1.5,-1.5,-1.5,-0.5,-1.5,-1.5,0]).reshape(len

print(R_s_a.T)
```

```
##      0   10   20   30   40   50   60   70
## n -1.0 -1.0 -1.0 -1.0  0.0 -1.0 -1.0  0.0
## s -1.5 -1.5 -1.5 -1.5 -0.5 -1.5 -1.5  0.0
```

```
pi_speed=pd.DataFrame(np.c_[np.repeat(0,len(states)),np.repeat(1,len(states))], index=states, columns=[

pi_50=pd.DataFrame(np.c_[np.repeat(0.5,len(states)), np.repeat(0.5,len(states))],index=states, columns=



print(pi_speed.T)
```

```
##    0  10  20  30  40  50  60  70
## n  0    0    0    0    0    0    0    0
## s  1    1    1    1    1    1    1    1
```

```
print(pi_50.T)

#simulator pi_speed
```

```
##       0   10   20   30   40   50   60   70
## n  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5
## s  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5
```

```
pi = pi_speed
np.random.seed(1234)
history = list()
MC_N = 10000
for MC_i in range(MC_N):
    s_now = '0'
    history_i = list(s_now)

    while s_now != '70':
        if np.random.uniform(0, 1) < pi.loc[s_now]['n']:
            a_now = 'n'
            P = P_normal
        else:
            a_now = 's'
            P = P_speed

        r_now = str(R_s_a.loc[s_now][a_now])
        s_next = states[np.argmin(P.loc[s_now].cumsum() < np.random.uniform(0, 1))].item()
        history_i.extend([a_now, r_now, s_next])
        s_now = s_next

    history.append(history_i)

history_speed = history
func = np.vectorize(lambda x: ','.join(x))
print(pd.Series(func(history_speed[:20])))

#simulator pi_50
```

```
## 0              0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70
## 1              0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70
## 2      0,s,-1.5,0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1....
## 3              0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70
## 4              0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70
## 5              0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70
## 6      0,s,-1.5,20,s,-1.5,10,s,-1.5,30,s,-1.5,50,s,-1...
## 7      0,s,-1.5,20,s,-1.5,40,s,-0.5,30,s,-1.5,50,s,-1...
## 8              0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70
## 9              0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70
## 10             0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70
## 11             0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70
## 12             0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70
## 13           0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70
## 14           0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70
## 15    0,s,-1.5,20,s,-1.5,10,s,-1.5,30,s,-1.5,50,s,-1...
## 16           0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70
## 17           0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70
## 18           0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70
## 19           0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70
## dtype: object
##
## C:\Users\user\AppData\Local\R-MINI~1\envs\R-RETI~1\lib\site-packages\numpy\core\_asarray.py:83: Visi
##   return array(a, dtype, copy=False, order=order)
```

```
pi = pi_50
np.random.seed(1234)
history = list()
MC_N = 10000
for MC_i in range(MC_N):
    s_now = '0'
    history_i = list(s_now)

    while s_now != '70':
        if np.random.uniform(0, 1) < pi.loc[s_now]['n']:
            a_now = 'n'
            P = P_normal
        else:
            a_now = 's'
            P = P_speed

        r_now = str(R_s_a.loc[s_now][a_now])
        s_next = states[np.argmin(P.loc[s_now].cumsum() < np.random.uniform(0, 1))].item()
        history_i.extend([a_now, r_now, s_next])
        s_now = s_next

    history.append(history_i)

history_50 = history
func = np.vectorize(lambda x: ','.join(x))
pd.Series(func(history_50[:20]))

#implementation pi_speed (vectorized)
```

```
## 0     0,n,-1.0,10,n,-1.0,20,s,-1.5,40,n,0.0,50,s,-1....
## 1     0,n,-1.0,10,s,-1.5,30,n,-1.0,40,s,-0.5,30,s,-1...
## 2     0,s,-1.5,20,n,-1.0,30,n,-1.0,40,s,-0.5,60,s,-1...
## 3     0,s,-1.5,20,n,-1.0,30,n,-1.0,40,n,0.0,50,n,-1....
## 4     0,n,-1.0,10,n,-1.0,20,n,-1.0,30,s,-1.5,20,s,-1...
## 5     0,n,-1.0,10,n,-1.0,20,n,-1.0,30,n,-1.0,40,n,0....
## 6     0,n,-1.0,10,n,-1.0,20,n,-1.0,30,n,-1.0,40,n,0....
## 7             0,s,-1.5,20,s,-1.5,40,s,-0.5,60,n,-1.0,70
## 8     0,s,-1.5,20,n,-1.0,30,s,-1.5,50,n,-1.0,60,s,-1...
## 9     0,s,-1.5,20,s,-1.5,40,n,0.0,50,n,-1.0,60,n,-1....
## 10    0,n,-1.0,10,s,-1.5,30,n,-1.0,40,s,-0.5,60,s,-1...
## 11    0,s,-1.5,20,n,-1.0,30,n,-1.0,40,n,0.0,50,n,-1....
## 12    0,n,-1.0,10,s,-1.5,30,n,-1.0,40,n,0.0,50,s,-1....
## 13            0,n,-1.0,10,s,-1.5,30,s,-1.5,50,s,-1.5,70
## 14            0,n,-1.0,10,s,-1.5,30,s,-1.5,50,s,-1.5,70
## 15            0,n,-1.0,10,s,-1.5,30,s,-1.5,50,s,-1.5,70
## 16    0,s,-1.5,20,s,-1.5,40,n,0.0,50,n,-1.0,60,n,-1....
## 17            0,s,-1.5,20,s,-1.5,40,s,-0.5,60,n,-1.0,70
## 18    0,n,-1.0,10,n,-1.0,20,s,-1.5,40,n,0.0,50,n,-1....
## 19            0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70
## dtype: object
##
## C:\Users\user\AppData\Local\R-MINI~1\envs\R-RETI~1\lib\site-packages\numpy\core\_asarray.py:83: Visi
##   return array(a, dtype, copy=False, order=order)
```

```
pol_eval = pd.DataFrame(np.zeros((len(states), 2)), index=states, columns=['count', 'sum'])
print(pol_eval.T)
```

```
##          0   10   20   30   40   50   60   70
## count  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
## sum    0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
```

```
for MC_i in range(len(history_speed)):
    history_i = history_speed[MC_i]

    for j in range(0, len(history_i), 3):
        pol_eval.loc[history_i[j]]['count'] += 1

        if j < len(history_i):
            pol_eval.loc[history_i[j]]['sum'] += pd.Series(history_i)[range(j + 2, len(history_i) - 1, 
                'float').sum()

        else:
            pol_eval.loc[history_i[j]]['sum'] += 0

print(pol_eval.T)
```

```
##              0       10       20      30       40      50       60       70
## count  11225.0  1076.0  10291.0  1887.0   9485.0  2563.0   8563.0  10000.0
## sum   -65136.0 -5619.0 -42703.0 -6539.0 -22275.5 -4472.5 -14355.0      0.0
```

```
print((pol_eval['sum'] / pol_eval['count']))

#implementation pi_speed (running estimate)
```

```
## 0    -5.802762
## 10   -5.222119
## 20   -4.149548
## 30   -3.465289
## 40   -2.348498
## 50   -1.745025
## 60   -1.676398
## 70    0.000000
## dtype: float64
```

```
pol_eval = pd.DataFrame(np.zeros((len(states), 2)), index=states, columns=[’count’, ’est’])
print(pol_eval.T)
```

```
##          0   10   20   30   40   50   60   70
## count  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
## est    0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
```

```
for MC_i in range(len(history_speed)):
    history_i = history_speed[MC_i]

    for j in range(0, len(history_i), 3):
        pol_eval.loc[history_i[j]][’count’] += 1
        current_cnt = pol_eval.loc[history_i[j]][’count’]

        if j < len(history_i):
            new_info = pd.Series(history_i)[range(j + 2, len(history_i) - 1, 3)].astype(’float’).sum()

        else:
            new_info = 0

        alpha = 1 / current_cnt
        pol_eval.loc[history_i[j]][’est’] += alpha * (new_info - pol_eval.loc[history_i[j]][’est’])

print(np.round(pol_eval.T, 2))

#implementation pi_50 (vectorized)
```

```
##              0       10        20       30       40       50       60       70
## count  11225.0  1076.00  10291.00  1887.00  9485.00  2563.00  8563.00  10000.0
## est       -5.8    -5.22     -4.15    -3.47    -2.35    -1.75    -1.68      0.0
```

```
pol_eval = pd.DataFrame(np.zeros((len(states), 2)), index=states, columns=[’count’, ’sum’])
print(pol_eval.T)
```

```
##          0   10   20   30   40   50   60   70
## count  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
## sum    0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
```

```
for MC_i in range(len(history_50)):
    history_i = history_50[MC_i]

    for j in range(0, len(history_i), 3):
        pol_eval.loc[history_i[j]][’count’] += 1

        if j < len(history_i):
            pol_eval.loc[history_i[j]][’sum’] += pd.Series(history_i)[range(j + 2, len(history_i) - 1,
                ’float’).sum()

        else:
            pol_eval.loc[history_i[j]][’sum’] += 0

print(pol_eval.T)
```

```
##                0        10        20        30        40        50       60        70
## count  10863.0   5792.0   8140.0   7121.0   7549.0   7363.0  6991.0  10000.0
## sum   -64904.5 -29662.5 -33549.0 -24133.0 -15410.0 -14874.5 -9436.5      0.0
```

```
print(pol_eval['sum'] / pol_eval['count'])

#implementation pi_speed (running estimate)
```

```
## 0    -5.974823
## 10   -5.121288
## 20   -4.121499
## 30   -3.388990
## 40   -2.041330
## 50   -2.020168
## 60   -1.349807
## 70    0.000000
## dtype: float64
```

```
pol_eval = pd.DataFrame(np.zeros((len(states), 2)), index=states, columns=['count', 'est'])
print(pol_eval.T)
```

```
##          0   10   20   30   40   50   60   70
## count  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
## est    0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
```

```
for MC_i in range(len(history_50)):
    history_i = history_50[MC_i]

    for j in range(0, len(history_i), 3):
        pol_eval.loc[history_i[j]]['count'] += 1
        current_cnt = pol_eval.loc[history_i[j]]['count']

        if j < len(history_i):
            new_info = pd.Series(history_i)[range(j + 2, len(history_i) - 1, 3)].astype('float').sum()

        else:
            new_info = 0

        alpha = 1 / current_cnt
        pol_eval.loc[history_i[j]]['est'] += alpha * (new_info - pol_eval.loc[history_i[j]]['est'])

print(np.round(pol_eval.T, 2))
```

```
##                0        10        20        30        40        50        60        70
## count  10863.00  5792.00  8140.00  7121.00  7549.00  7363.00  6991.00  10000.0
## est       -5.97    -5.12    -4.12    -3.39    -2.04    -2.02    -1.35      0.0
```