C1_황재훈

Jaehun Hwang

1/4/2021

P25.Simulating stochastic paths

```
import numpy as np

def soda_simul(this_state):
    u = np.random.rand()
    if (this_state == "c"):
        if(u<=0.7):
            next_state = "c"
        else:
            next_state = "p"
    else:
        if(u<=0.5):
            next_state = "c"
        else:
            next_state = "p"
    return next_state


for i in range(1,6):
    path = "c"
    for n in range(1,10):
        this_state = path[-1] # read lastest state
        next_state = soda_simul(this_state) # determine next state
        path = path + next_state
    print(i, path)
```

```
## 1 cpcppcppcc
## 2 ccccccppcp
## 3 ccccccccccc
## 4 cccpcccccc
## 5 cppccccpcc
```

P25.Simulating stochastic paths (cont.)

```
import numpy as np

def soda_simul(this_state):
    u = np.random.rand()
    if (this_state == "c"):
        if(u<=0.7):
            next_state = "c"
        else:
            next_state = "p"
    else:
```

```
        if(u<=0.5):
            next_state = "c"
        else:
            next_state = "p"
    return next_state

def cost_eval(path):
    cost_one_path = path.count("c") * 1.5 + path.count("p") * 1
    return cost_one_path


MC_N = 10000
spending_records = [0] * MC_N

for i in range(MC_N):
    path = "c"
    for t in range(1,10):
        this_state = path[-1] # read lastest state
        next_state = soda_simul(this_state) # determine next state
        path = path + next_state
    spending_records[i] = cost_eval(path)

print(np.mean(spending_records))
```

```
## 13.36325
```