Modeling under uncertainty

Queenie Liu
ql299

Q1 (a) $H \cap P^c \cap S^c$

(b) $P \cap H \cap S^c$

(c) $H \cup P \cup S$

(d) $(H \cap P) \cup (H \cap S) \cup (P \cap S)$

(e) $S \cap H \cap P$

(f) $S^c \cap H^c \cap P^c$

(g) $(H^c \cap P^c) \cup (H^c \cap S^c) \cup (P^c \cap S^c)$

(h) $(H \cap P \cap S)^c$

Q2 R = red

G = green

B = Blue

Sample space:

(RR, RG, RB, BB, BG, BR, GB, GG, GR)

$P(\text{each point in the sample space}) = \frac{1}{3} \times \frac{1}{3} = \frac{1}{9}$

Q4 (a) $P(\text{red}) = \frac{1}{2} \times \frac{1}{2} + \frac{1}{2} \times \frac{2}{3} = \frac{7}{12}$

(b) $P(A \mid \text{white}) = \frac{P(\text{white} \mid A)\, P(A)}{P(\text{white})}$

$= \frac{\frac{1}{2} \times \frac{1}{2}}{\frac{5}{12}} = \frac{3}{5}$

Q3

October 20, 2019

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

(a) Write a recursion of C(N, m)

[2]:
```
def count_ways (S, m, n):
    if n == 0:
        return 1
    if n < 0:
        return 0
    if m <= 0 and n >= 1:
        return 0
    return count_ways (S, m - 1, n) + count_ways(S, m, n-S[m-1])
```

(b) With S1 = 1, S2 = 5, S3 = 10 and S4 = 25, how many ways are there to change N = 213 cents?

[6]:
```
print (count_ways ([ 1, 5, 10, 25 ], 4, 213 ))
S1234 = count_ways ([ 1, 5, 10, 25 ], 4, 213 )
```

```
1670
```

(c) What fraction of these uses only 1, 5 and 10 cents denominations?

[7]:
```
print (count_ways([1, 5, 10], 3, 213))
S123 = count_ways([ 1, 5, 10 ], 3, 213)
```

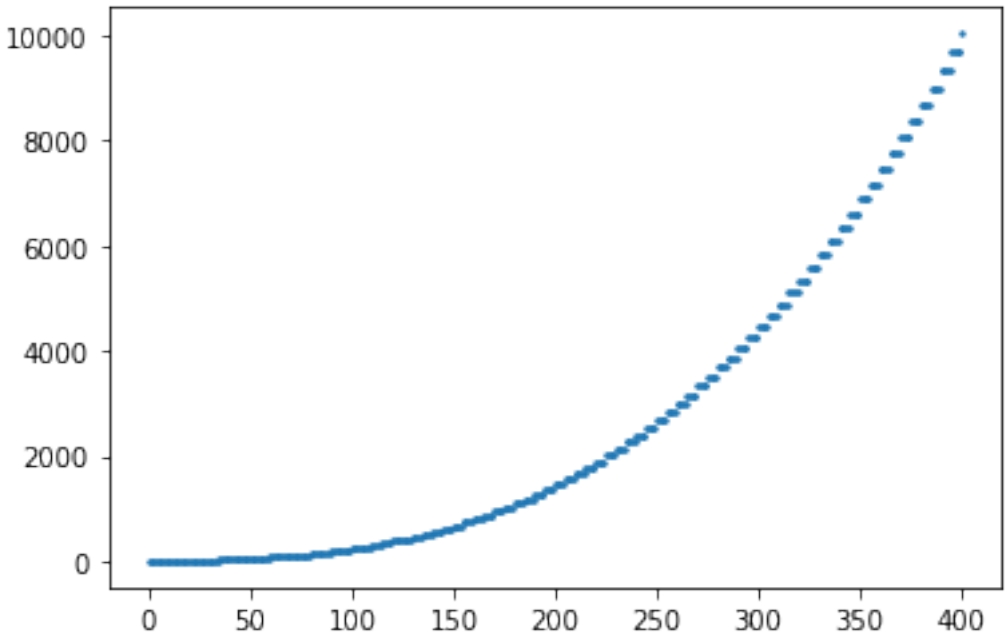
```
484
```

[8]:
```
print (S123/S1234)
```

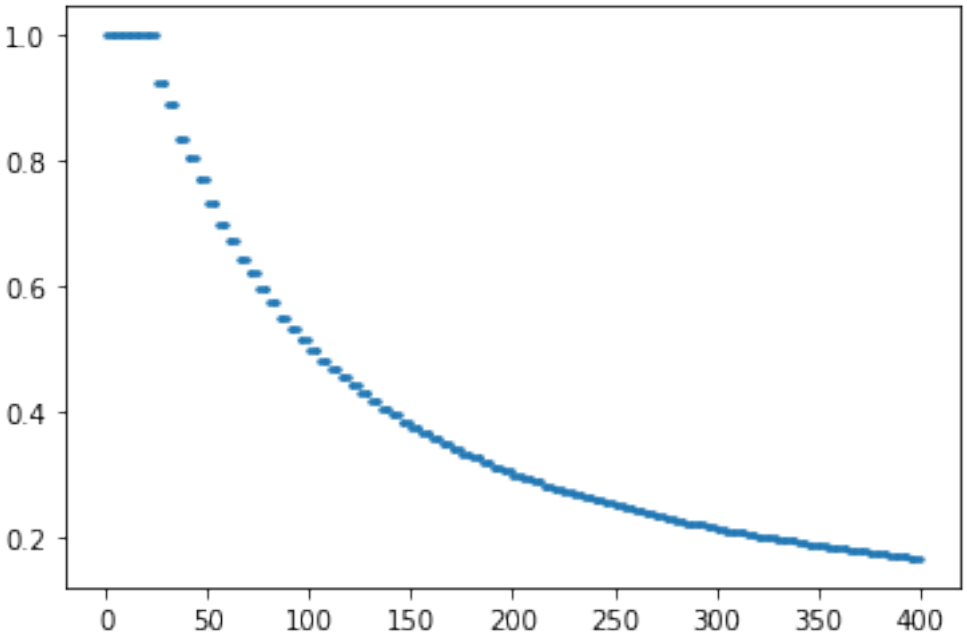
```
0.2898203592814371
```

(d) Repeat the same for multiple values of N and plot the results corresponding to the previous two questions (i.e., number of ways to change N using S1 = 1, S2 = 5, S3 = 10, S4 = 25, and fraction of the total using only 1-, 5- and 10-cent coins) as a function of N ranging from 0 to 400.

[17]:
```
four_count = []
for n in np.arange (0, 401):
    four_count.append(count_ways([1, 5, 10 , 25], 4, n))

plt.scatter(np.arange (0, 401), ways, s = 2)
plt.show()
```



[18]:
```
three_count = []
for n in np.arange(0, 401):
    three_count.append(count_ways([1,5,10], 3, n))

fractions = [x/y for x , y in zip (three_count, four_count)]

plt.scatter(np.arange(0, 401), fractions, s = 2)
plt.show()
```


1.0
0.8
0.6
0.4
0.2
0
50
100
150
200
250
300
350
400

[ ]: