prob_probs_1

August 19, 2024

[ ]:
```
#2.4
p41 = .7
p42 = .85
p43 = .8

p_all = p41 * p42 * p43
p_none = (1 - p41) * (1 - p42) * (1 - p43)
print(p_all)
print(p_none)
```

```
0.476
0.009000000000000001
```

[ ]:
```
#2.8 (c) under the assumptions laid out in the exercise that we
# pick 1 box followed by the host opening 1 of the other 2 boxes
# and that opened box being empty, this question is really asking
# the probability of box C being the winning box. If we pick box
# A and box C is the winning box, then the host will always open
# box B. Thus the joint probability of the host opening box B and
# box C being the winning box is the probability of box C being
# the winning box = 1/3.
p8Cw = 1/3
print(p8C)
```

```
0.3333333333333333
```

[ ]:
```
#2.8 (d) under the same assumptions described above, if we pick
# box A and box A is the winning box, then # both boxes B and C
# are empty, and the host can open either box. The joint probability
# of box A being the winning box and the host opening box C is the
# probability of box A being the winning box (1/3) multiplied by the
# probability of the host opening box C between boxes B and C (1/2)
# = 1/6.
p8Aw = 1/3
P8_openC_Aw = 1/2
print(p8Aw * P8_openC_Aw)
```

```
0.16666666666666666
```

[ ]:
```
#2.12
import numpy as np
import pandas as pd
from scipy.stats import rv_discrete
x = np.array([9, 10, 11, 12, 13, 14, 15])
p = np.array([.07, .15, .23, .25, .15, .12, .03])
df = pd.DataFrame({
    'return': x,
    'prob': p
})

p_atleast12 = df['prob'][df['return'] >= 12].sum()
print(p_atleast12)

expected_return = np.dot(df['return'], df['prob'])
print(expected_return)

stdi = np.sqrt(np.dot((df['return'] - expected_return)**2, df['prob']))
print(stdi)
```

```
0.55
11.739999999999998
1.514067369703211
```

[ ]:
```
dist = rv_discrete(name="return dist", values=(x, p))
print(dist.mean())
print(dist.pmf(12))
print(dist.sf(11))
print(dist.std())
```

```
11.739999999999998
0.25
0.55
1.5140673697032254
```

[ ]:
```
#footnote: I used both GitHub Copilot and ChatGPT with this quiz.
# GitHub recommended the use of np.dot to calculate the expected
# value from data set, and I then used ChatGPT to explain what that
# function is and how it works. I further used ChatGPT to explore
# some other ways of working with a problem like this, including the
# use of scipy.stats.rv_discrete to create a discrete probability
# distribution object. See link to chatGPT transcript here:

# https://chatgpt.com/share/e/5b26dfc1-e218-4a23-9aca-128ae3fa3a6f
```

[ ]: