Project 2 - “Conditional Probabilities”

CECS 381 - Sec 06

Dustin Martin - 015180085

Problem 1: Probability of erroneous transmission

- **Introduction**
  In the given setup for bit transmission, accuracy is not a guarantee. When a bit is transmitted, there is a chance that it is flipped from 1 to 0 (or vice versa). The probability that a 1 will be flipped to a 0 is given by ε_1 = 0.07, and the probability that a 0 will be flipped to a 1 is ε_0 = 0.04. Bits are generated with an unfair probability, with P(0) = 0.35 (denoted by p_0).
  This experiment aims to calculate the probability of error when sending a single-bit transmission by repeatedly generating bits and transmitting them before calculated a failure rate.

- **Methodology**
  A 1 or 0 is generated by sending a probability to the nSidedDie() function, which will roll an unfair die with the probabilities given. In this test. 0s are generated slightly less at a probability of 0.35.
  If the generated bit is 1, another die is rolled to simulate whether or not the bit is flipped in transmission (using p0.07). The chance a 0 is flipped is slightly lower at 0.04**.**
  The resulting transmitted bit is then compared with the orignal bit. If they match, it is a success. If they dont match, it is a failure.
  The test is repeated 100k times and each failure is counted to calculate the probability of erroneous transmissions.

- **Results**

Probability of transmission error	
Ans.	**p=** ~0.06031

After running the test 100k times, the probability that a bit will be flipped in transmission is around 0.06031.

- ## Appendix/Code

```
import numpy as np

p0=0.35
e0=0.04
e1=0.07

N = 100000 #number of times to repeat the experiment

def nSidedDie(p): #flips the unfair die a single time and returns the result
    n = len(p)
    cs = np.cumsum(p)
    cp = np.append(0,cs)
    r = np.random.rand()
    for k in range(0,n):
        if r>cp[k] and r<=cp[k+1]:
            d=k+1
    return d

failures = 0
for num in range(N):
    S = nSidedDie(np.array ([p0,1-p0])) - 1 #this function is designed for die, so
subtracting 1 will give 0 or 1.
    if S == 1:
        R = nSidedDie(np.array ([e1,1-e1])) - 1 #probability that R=1 given S=1
    elif S == 0:
        R = nSidedDie(np.array ([1-e0, e0])) - 1#probability that R=0 given S=0
    if S!=R:
        failures+=1 #if S does not equal R, count as a failure

pote = failures/N #"probabilities of transmission error
print("1. probabilities of transmission error: ",pote)
```

Problem 2: Conditional probability: P(R=1| S=1)

- **Introduction**

  This experiment aims to conclude the probability that the transmitted message comes through as 1, given that 1 was the originally-generated message.

  The experiment assumes 1 was generated, then transmits a 1-bit repeatedly, counting the number of times a 1 comes out the other side. This is done 100k times to ensure accurate probabilities.

- **Methodology**

  It is assumed a 1 is the originally generated message.

  1 is put in the transmission simulation and if it was transmitted without flipping, it was deemed a success. Otherwise, it is a failure.

  The test is repeated 100k times and each failure is counted to calculate the probability of erroneous transmissions.

- **Results**

  As expected, the result is

Conditional probability P(R=1|S=1)	
Ans.	**p=** ~0.9303

We knew the answer would be more or less 0.93 as the probability that a 1 is sent incorrectly is 0.07, and 1-0.07=0.93.

- **Appendix/Code**

```
import numpy as np

p0=0.35
e0=0.04
e1=0.07

N = 100000 #number of times to repeat the experiment

def nSidedDie(p): #flips the unfair die a single time and returns the result
    n = len(p)
    cs = np.cumsum(p)
    cp = np.append(0,cs)
    r = np.random.rand()
    for k in range(0,n):
        if r>cp[k] and r<=cp[k+1]:
            d=k+1
    return d


successes = 0
for num in range(N):
    R = nSidedDie(np.array ([e1,1-e1])) - 1 #runs the probability that R=1 if S=1
    if R == 1:
        successes+=1 #if R=1, count as success

pote = successes/N
print("2. conditional probability P(R=1|S=1). p=",pote)
```

Problem 3: Conditional probability: P(S=1| R=1)

- **Introduction**
  This experiment is a little more involved. It aims to conclude the probability that the originally generated message is a 1, given that a 1 was received on the other side. We do this by sending bits as normal, but for every 1 we receive, we go back and check to see if 1 was the originally generated message. If so, it is deemed a success. Otherwise it is a failure.

- **Methodology**
  A 1 or 0 is generated by sending a probability to the nSidedDie() function, which will roll an unfair die with the probabilities given. In this test. 0s are generated slightly less at a probability of 0.35.
  If the generated bit is 1, another die is rolled to simulate whether or not the bit is flipped in transmission (using p0.07). The chance a 0 is flipped is slightly lower at 0.04**.**
  Everytime a 1 is received on the other side, we go back and check to see if the original message is 1 as well. If it is, it is deemed a success, otherwise it is a failure.
  The test is repeated 100k times and each success is counted to calculate the probability.

- **Results**

Conditional probability P(S=1|R=1)	
Ans.	**p=** ~0.977

  - Everytime a 1 is received, there is a 97.7% chance that the original message was 1.

- **Appendix/Code**

```
import numpy as np

p0=0.35
e0=0.04
e1=0.07

N = 100000 #number of times to repeat the experiment

def nSidedDie(p): #flips the unfair die a single time and returns the result
    n = len(p)
    cs = np.cumsum(p)
    cp = np.append(0,cs)
    r = np.random.rand()
    for k in range(0,n):
        if r>cp[k] and r<=cp[k+1]:
            d=k+1
    return d

successes = 0
runs=0
while runs <= N:
    S = nSidedDie(np.array ([p0,1-p0])) - 1 #this function is designed for die, so
subtracting 1 will give 0 or 1.
    if S == 1:
        R = nSidedDie(np.array ([e1,1-e1])) - 1 #probability R=1 given S=1
    elif S == 0:
        R = nSidedDie(np.array ([1-e0, e0])) - 1 #probability R=0 given S=0

    if R==1: #if function calculates the probability that, given R=1, what is the
chance that S=1
        runs+=1
        if S==1:
            successes+=1
pote = successes/N
print('3. conditional probability P(S=1|R=1). p=',pote)
```

Problem 4: Enhanced Transmission Method

- **Introduction**
  This experiment aims to give a way to increase the probability that the correct message is received without actually having to optimize the sending system. To do so, a message is sent three times instead of once, and the 'majority' response from the triply- transmitted message is accepted as the response. For example, 1 is transmitted as (1,1,1). Since there is a probability that a bit flips, the received message might be (1,0,1), but we accept this message as 1 since most of the bits in the array are one. The mirror of this is the same for 0.

- **Methodology**
  A 1 or 0 is generated by sending a probability to the nSidedDie() function, which will roll an unfair die with the probabilities given. In this test. 0s are generated slightly less at a probability of 0.35.
  If the generated bit is 1, another die is rolled to simulate whether or not the bit is flipped in transmission (using p0.07). The chance a 0 is flipped is slightly lower at 0.04**.**
  Everytime a bit is generated, it is duplicated and sent 3 times. Each of the 3 messages are then transmitted. If the majority of the received messages are 1, then the message is received as 1. The opposite is true with 0. If the received message matches the original message, it is deemed a success, otherwise, it is a failure.
  The experiment is run 100k times and the failures are counted to calculate the probability of incorrect transmissions with this updated method.

- **Results**

Probability of error with enhanced transmission	
Ans.	**p=** ~0.0106

The probability of an erroneous message with the new enhanced system is around 0.0106

- **Appendix/Code**

```
import numpy as np
p0=0.35
e0=0.04
e1=0.07
N = 100000 #number of times to repeat the experiment
def nSidedDie(p): #flips the unfair die a single time and returns the result
    n = len(p)
    cs = np.cumsum(p)
    cp = np.append(0,cs)
    r = np.random.rand()
    for k in range(0,n):
        if r>cp[k] and r<=cp[k+1]:
            d=k+1
    return d
successes = 0
for num in range(N):
    S = nSidedDie(np.array ([p0,1-p0])) - 1 #this function is designed for die, so subtracting 1 will give 0 or 1.
    if S == 1:
        R = [nSidedDie(np.array ([e1,1-e1])) - 1, nSidedDie(np.array ([e1,1-e1])) - 1, nSidedDie(np.array ([e1,1-e1])) - 1]
        if sum(R)>=2:
            successes+=1
    elif S == 0:
        R = [nSidedDie(np.array ([1-e0, e0])) - 1, nSidedDie(np.array ([1-e0, e0])) - 1, nSidedDie(np.array ([1-e0, e0])) - 1]
        if sum(R)<2:
            successes+=1
pote = successes/N #calculates successful transmissions

failureP = 1-pote #calculates failure transmissions using q = 1-p

print("4. probability of failure with enhanced transmission:",failureP)
```