Assignment 10

1) จาก $P(y2, y1, y0|\alpha) = P(y2|y1)P(y1|y0)P(y0|\alpha)$

เนื่องจาก $y2 \sim N(\alpha y1, \sigma 2), y1 \sim N(\alpha y0, \sigma 2), y0 \sim N(0, \lambda)$

จะได้ว่า $P(y2, y1, y0|\alpha) = (\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_2-\alpha y_1)^2}{2\sigma^2}})(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_1-\alpha y_0)^2}{2\sigma^2}})(\frac{1}{\sqrt{2\pi\lambda}} e^{-\frac{{y_0}^2}{2\lambda}})$

ต้องการหา α ที่ทำให้ $P(y2, y1, y0|\alpha)$ มีค่ามากที่สุด

ซึ่งจะได้ว่าต้องการหาค่าสูงสุดของ $\left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_2-\alpha y_1)^2}{2\sigma^2}}\right)\left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_1-\alpha y_0)^2}{2\sigma^2}}\right)\left(\frac{1}{\sqrt{2\pi\lambda}} e^{-\frac{{y_0}^2}{2\lambda}}\right)$

ซึ่งคือต้องหาค่าสูงสุดของ $\ln\left(\left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_2-\alpha y_1)^2}{2\sigma^2}}\right)\left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_1-\alpha y_0)^2}{2\sigma^2}}\right)\left(\frac{1}{\sqrt{2\pi\lambda}} e^{-\frac{{y_0}^2}{2\lambda}}\right)\right)$

$$= \ln\left(\left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_2-\alpha y_1)^2}{2\sigma^2}}\right)\left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_1-\alpha y_0)^2}{2\sigma^2}}\right)\left(\frac{1}{\sqrt{2\pi\lambda}} e^{-\frac{{y_0}^2}{2\lambda}}\right)\right)$$

$$= -2\ln\left(\sqrt{2\pi\sigma^2}\right) - \frac{(y_2-\alpha y_1)^2}{2\sigma^2} - \frac{(y_1-\alpha y_0)^2}{2\sigma^2} - \ln\left(\sqrt{2\pi\lambda}\right) - \frac{{y_0}^2}{2\lambda}$$

ซึ่งคือต้องหาค่าต่ำสุดของ $2\ln\left(\sqrt{2\pi\sigma^2}\right) + \frac{(y_2-\alpha y_1)^2}{2\sigma^2} + \frac{(y_1-\alpha y_0)^2}{2\sigma^2} + \ln\left(\sqrt{2\pi\lambda}\right) + \frac{{y_0}^2}{2\lambda}$

ทำการ $differentiate$ เพื่อหาค่าต่ำสุดของพจน์ดังกล่าว จะได้

$$0 = \frac{\delta}{\delta\alpha}\left(2\ln\left(\sqrt{2\pi\sigma^2}\right) + \frac{(y_2-\alpha y_1)^2}{2\sigma^2} + \frac{(y_1-\alpha y_0)^2}{2\sigma^2} + \ln\left(\sqrt{2\pi\lambda}\right) + \frac{{y_0}^2}{2\lambda}\right)$$

$$0 = 0 + \frac{\delta}{\delta\alpha}\left(\frac{(y_2-\alpha y_1)^2}{2\sigma^2}\right) + \frac{\delta}{\delta\alpha}\left(\frac{(y_1-\alpha y_0)^2}{2\sigma^2}\right) + 0 + 0$$

$$0 = \frac{2(y_2-\alpha y_1)(-y_1)}{2\sigma^2} + \frac{2(y_1-\alpha y_0)(-y_0)}{2\sigma^2}$$

$$0 = -y_2y_1 + \alpha{y_1}^2 - y_1y_0 + \alpha{y_0}^2$$

$\alpha = \frac{y_2y_1+y_1y_0}{{y_1}^2+{y_0}^2}$ จึงจะได้ค่าต่ำสุด

ดังนั้นจึงสรุปได้ว่าเพื่อให้ได้ $P(y2, y1, y0|\alpha)$ ค่ามากที่สุดต้องเลือกให้ $\alpha = \frac{y_2y_1+y_1y_0}{{y_1}^2+{y_0}^2}$ #

4)

4.1) ไม่สามารถสรุปได้ เนื่องจาก 1. เราไม่ทราบ significant level จึงไม่สามารถตอบได้ว่า reject H_0 หรือไม่ 2. ถึงแม้เราทราบว่า reject H_0 หรือไม่แต่การเปรียบเทียบ $x_0(no\ block + no\ new\ channel)$ กับ $x_1(block + new\ channel)$ ก็ไม่สามารถสรุปผลเรื่องการเปิด new channel ได้เพราะมีตัวแปรเรื่องถูก block หรือไม่อยู่

4.2) ยังไม่เพียงพอ เนื่องจาก 1. เราไม่ทราบ significant level จึงไม่สามารถตอบได้ว่า reject H_0 หรือไม่ 2. แต่หากเราทราบว่า reject H_0 หรือไม่ เราจะสามารถเดาการเปรียบเทียบระหว่าง $x_0(no\ block + no\ new\ channel)$ กับ $x_3(no\ block + new\ channel)$ ได้ว่ามีแนวโน้มไปในทางเดียวกันกับการเปรียบเทียบระหว่าง x_2 กับ x_1

4.3) lead to และเราควรทำ hypothesis ที่ $H_0: x_0 \geq x_3$, $H_a: x_0 < x_3$ เพื่อนำมาประกอบกับข้อ 4.2 ซึ่งผลลัพธ์ที่ได้ควรเป็นไปในทางเดียวกันเพื่อสรุปว่าการ create new channel นั้นสำคัญสำหรับ hamtaro หรือไม่

5)

5.1) H_0: the die is fair (โอกาสออกของทุกหน้าคือ 1/6)

H_a: the die is unfair (โอกาสออกของหน้าที่ผู้เล่นเลือกน้อยกว่า 1/6)

5.2) ควรเป็น one-sided เนื่องจาก H_a: the die is unfair (โอกาสออกของหน้าที่ผู้เล่นเลือกน้อยกว่า 1/6) นั่นคือเราสนใจแค่โอกาสที่เล่นเลือกน้อยกว่า 1/6 ไม่ได้สนใจโอกาสที่เล่นเลือกมากกว่า 1/6 ถ้าหากทำการเลือก two-sided ทำให้ rejection region รวมถึงทางฝั่ง มากกว่า 1/6 ด้วย ซึ่งผิดกับสิ่งที่เราต้องการจะ reject นอกจากนี้การเพิ่ม rejection region ที่เราไม่ต้องการจะทำให้ type I error เพิ่มมากขึ้นและทำให้สรุปผลผิดได้

5.3) ทำการสุ่มการเล่น 30 ครั้ง และหาว่าสำหรับ การออกหน้าที่เลือก 3 ครั้งจาก 30 ครั้งอยู่ในช่วง rejection region หรือไม่ โดยทำการคำนวณตาม code ด้านล่าง

```
# TODO#5.3
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import binom

def sample_binomial(sample_size=1000000, n=100, p=0.3):
  return binom.rvs(n = n, p = p, size = sample_size)

sample_size = 1000000
n = 30
p = 1/6
s = sample_binomial(sample_size, n, p)

sig = 0.1
s.sort()
print(np.where(s == 3)[0][0]/sample_size)
```

```
0.102903
```

พบว่าไม่อยู่ใน rejection region เนื่องจากโจทย์บอกว่า significant level = 10% = 0.1

จึงทำการสรุปได้ว่า ผู้เล่นจะยังไม่สามารถ reject H_0 ได้

5.4) ทำการสุ่มการเล่น 200 ครั้งและหาตำแหน่งของค่าที่มากที่สุดที่ไม่เกินตำแหน่งของ significant level โดยทำการคำนวณตาม code ด้านล่าง

```
# TODO#5.4
sample_size = 1000000
n = 200
p = 1/6
s = sample_binomial(sample_size, n, p)

sig = 0.1
sig_position = int(sig*sample_size)
s.sort()
print(s[sig_position]-1)
```

```
26
```

ดังนั้น rejection region คือ การที่หน้าที่เลือกไม่ออกตั้งแต่ 0 – 26 ครั้ง

5.5) ทำแบบเดียวกับข้อ 5.4 แต่เปลี่ยนจากการใช้ binomial distribution เป็น normal distribution เนื่องจากมีการใช้จำนวนตัวอย่างที่มากทำให้ binomial distribution ที่ได้มีรูปร่างใกล้เคียงกับ normal distribution และทำให้ผลลัพธ์ของ rejection region ได้ผลเหมือนกัน ซึ่งคำนวณตาม code ด้านล่าง

```
# TODO#5.5
from scipy.stats import norm

def sample_normal(sample_size=1000000, mu=0, std=1):
  return norm.rvs(loc = mu, scale = std, size = sample_size)

sample_size = 1000000
n = 200
p = 1/6
s = sample_normal(sample_size, n*p, math.sqrt(n*p*(1-p)))

sig = 0.1
sig_position = int(sig*sample_size)
s.sort()
print(round(s[sig_position]-1))
```

```
26
```

ดังนั้น rejection region คือ การที่หน้าที่เลือกไม่ออกตั้งแต่ 0 – 26 ครั้ง

5.6) จาก significant level = 0.01 ทำการหา rejection region ออกมาและทำการปรับ p (ความน่าจะเป็นที่หน้าลูกเต๋าที่ผู้เล่นเลือกจะออก) ของ alternative จนกระทั่งค่า p สอดคล้องกับ power level = 0.05 ซึ่งสามารถคำนวนได้ตาม code ด้านล่าง

```
# TODO#5.6
def cal_cur_power(pa):
  current_s = sample_binomial(sample_size, n, pa)
  current_s.sort()
  accmu_prop = 0
  for c in current_s:
    if c <= reject_region:
      accmu_prop += 1
  return accmu_prop/sample_size

sample_size = 1000000
n = 200
p = 1/6
s = sample_binomial(sample_size, n, p)
```

```
sig = 0.01
sig_position = int(sig*sample_size)
s.sort()
reject_region = s[sig_position]-1

power = 0.05
pa = 1/6
current_power = 0
while current_power < power:
  current_power = cal_cur_power(pa)
  pa -= 0.0001

print(round(pa, 3))
```

```
0.148
```

ซึ่งจะได้ว่าความน่าจะเป็นของการที่หน้าลูกเต๋าที่ผู้เล่นเลือกจะออกที่น้อยที่สุดที่คำนวนได้คือ 0.148

5.7) เนื่องจากค่า significant level เป็นค่าเดียวกับข้อ 5.6 จึงทำแบบเดียวกับข้อ 5.6 เพียงแค่เปลี่ยนค่า power level จาก 0.05 เป็น 0.01 ซึ่งสามารถคำนวนได้ตาม code ด้านล่าง

```
# TODO#5.7
power = 0.01
pa = 1/6
current_power = 0
while current_power < power:
  current_power = cal_cur_power(pa)
  pa -= 0.0001

print(round(pa, 3))
```

```
0.166
```

ซึ่งจะได้ว่าความน่าจะเป็นของการที่หน้าลูกเต๋าที่ผู้เล่นเลือกจะออกที่น้อยที่สุดที่คำนวนได้คือ 0.166

7)

7.1) H_0: เครื่องจักรเก่าดีมากกว่าหรือเท่ากับเครื่องจักรใหม่

H_a: เครื่องจักรใหม่ดีกว่าเครื่องจักรเก่า

7.2)

```
# TODO#7.2
import math, scipy

def z(fac):
  z = (np.mean(fac)-miu)*math.sqrt(n)/std
  p_value = scipy.stats.norm.sf(abs(z))
  if p_value < sig:
    print("Reject H0")
  else:
    print("Not Reject H0")
```

```
miu = 5000
n = 30*4
std = 20
sig = 0.05
all_fac = np.array([fac_0, fac_1, fac_2, fac_3])
print("All factory")
z(all_fac)
```

```
All factory
Reject H0
```

จากการทดสอบโดยใช้ z-test กับทุกโรงพบว่า reject H_0 นั่นคือสำหรับทุกโรงงานโดยรวมการใช้เครื่องจักรใหม่ดีกว่าเครื่องจักรเก่า

7.3)

```
# TODO#7.3
n = 30
print("Factory 0")
z(fac_0)
print("\nFactory 1")
z(fac_1)
print("\nFactory 2")
z(fac_2)
print("\nFactory 3")
z(fac_3)
```

```
Factory 0
Reject H0

Factory 1
Reject H0

Factory 2
Reject H0

Factory 3
Not Reject H0
```

ทำการทดสอบแบบเดียวกับข้อ 7.2 พบว่าทุกโรงงานยกเว้นโรงงาน 3 นั้น reject H_0 นั่นคือสำหรับโรงงาน 0, 1, 2 การใช้เครื่องจักรใหม่ดีกว่าเครื่องจักรเก่า ส่วนสำหรับโรงงาน 3 นั้น not reject H_0

7.4) จากผลลัพธ์ของข้อ 7.1-7.3 เทียบกับ 7.4 จะเห็นว่าได้ผลลัพธ์เหมือนกัน เนื่องจากจำนวนของเครื่องจักรมากกว่า 30 (large number) ทำให้ student's t distribution มีรูปร่างคล้าย normal distribution ซึ่งส่งผลให้ผลลัพธ์เหมือนกัน

7.4.1) H_0: เครื่องจักรเก่าดีมากกว่าหรือเท่ากับเครื่องจักรใหม่

H_a: เครื่องจักรใหม่ดีกว่าเครื่องจักรเก่า

7.4.2)

```
# TODO#7.4.2
def t(fac):
  t = (np.mean(fac)-miu)*math.sqrt(n)/np.std(fac)
  p_value = scipy.stats.t.sf(np.abs(t), n-1)
  if p_value < sig:
    print("Reject H0")
  else:
    print("Not Reject H0")

n = 30*4
all_fac = np.array([fac_0, fac_1, fac_2, fac_3])
print("All factory")
t(all_fac)
```

```
All factory
Reject H0
```

จากการทดสอบโดยใช้ t-test กับทุกโรงพบว่า reject H_0 นั่นคือสำหรับทุกโรงงานโดยรวมการใช้เครื่องจักรใหม่ดีกว่าเครื่องจักรเก่า

7.4.3)

```
# TODO#7.4.3
n = 30
print("Factory 0")
t(fac_0)
print("\nFactory 1")
t(fac_1)
print("\nFactory 2")
t(fac_2)
print("\nFactory 3")
t(fac_3)
```

```
Factory 0
Reject H0

Factory 1
Reject H0

Factory 2
Reject H0

Factory 3
Not Reject H0
```

ทำการทดสอบแบบเดียวกับข้อ 7.4.2 พบว่าทุกโรงงานยกเว้นโรงงาน 3 นั้น reject H_0 นั่นคือสำหรับโรงงาน 0, 1, 2 การใช้เครื่องจักรใหม่ดีกว่าเครื่องจักรเก่า ส่วนสำหรับโรงงาน 3 นั้น not reject H_0