專題

1. The ship Titanic sank in 1912 with the loss of most of its passengers. Details can be obtained on 1309 passengers and crew onboard the ship Titanic. Dataset, description

A. Among these passengers, are there age differences between the survived (survived=1) and the dead (survived=0) groups? Assume the population age is normally distributed. 請問存活與死亡的乘客的年紀是否有差？

使用 one_way ANOVA Test: 因為探討的因素為比較不同存活和死亡乘客的平均年紀關係。one-way ANOVA為單尾檢定

H0: 存活和死亡乘客的平均年紀相等

Ha: 存活和死亡乘客的平均年紀不相等

In [194…

```
import pandas as pd
import numpy as np

# load data file(原始資料)
origin_df = pd.read_csv("D:\統計學\Final_Project\Titanic_R.csv")
# df 為操作之資料
df = origin_df.copy()
df
```

Out[194…

	pclass	survived	Residence	name	age	sibsp	parch	ticket	fare	cabin	embarl
0	3	0	0	Abbing, Mr. Anthony	42	0	0	C.A. 5547	7.55		
1	3	0	0	Abbott, Master. Eugene Joseph	13	0	2	C.A. 2673	20.25		
2	3	0	0	Abbott, Mr. Rossmore Edward	16	1	1	C.A. 2673	20.25		
3	3	1	0	Abbott, Mrs. Stanton (Rosa Hunt)	35	1	1	C.A. 2673	20.25		
4	3	1	2	Abelseth, Miss. Karen Marie	16	0	0	348125	7.65		
…	…	…	…	…	…	…	…	…	…	…	…
1304	3	0	2	Zabour, Miss. Hileni	14.5	1	0	2665	14.4542		

	pclass	survived	Residence	name	age	sibsp	parch	ticket	fare	cabin	embarl
1305	3	0	2	Zabour, Miss. Thamine		1	0	2665	14.4542		
1306	3	0	2	Zakarian, Mr. Mapriededer	26.5	0	0	2656	7.225		
1307	3	0	2	Zakarian, Mr. Ortin	27	0	0	2670	7.225		
1308	3	0	2	Zimmerman, Mr. Leo	29	0	0	315082	7.875		

1309 rows × 15 columns

In [195…

```
# 這邊是利用正則表達式，將pandas中的空白，替換成NaN
df = df.replace(r'\s+', np.nan, regex = True)
df
```

Out[195…

	pclass	survived	Residence	name	age	sibsp	parch	ticket	fare	cabin	embarked	b
0	3	0	0	NaN	42	0	0	NaN	7.55	NaN	S	N
1	3	0	0	NaN	13	0	2	NaN	20.25	NaN	S	N
2	3	0	0	NaN	16	1	1	NaN	20.25	NaN	S	N
3	3	1	0	NaN	35	1	1	NaN	20.25	NaN	S	
4	3	1	2	NaN	16	0	0	348125	7.65	NaN	S	
…	…	…	…	…	…	…	…	…	…	…	…	
1304	3	0	2	NaN	14.5	1	0	2665	14.4542	NaN	C	N
1305	3	0	2	NaN	NaN	1	0	2665	14.4542	NaN	C	N
1306	3	0	2	NaN	26.5	0	0	2656	7.225	NaN	C	N
1307	3	0	2	NaN	27	0	0	2670	7.225	NaN	C	N
1308	3	0	2	NaN	29	0	0	315082	7.875	NaN	S	N

1309 rows × 15 columns

In [196…

```
# 確認隨機一組資料是否有空值
df.iloc[1305]
```

Out[196…

```
pclass            3
survived          0
Residence         2
name            NaN
age             NaN
sibsp             1
parch             0
ticket         2665
fare        14.4542
cabin           NaN
embarked          C
```

```
boat             NaN
body             NaN
home.dest        NaN
Gender             1
Name: 1305, dtype: object
```

In [197…

```
# 檢查資料/目標columns是否有空值，確保資料的完整性
# 並發現，資料內部很多columns都有空值
df.isnull().any()
```

Out[197…

```
pclass       False
survived     False
Residence    False
name          True
age           True
sibsp        False
parch        False
ticket        True
fare          True
cabin         True
embarked      True
boat          True
body          True
home.dest     True
Gender       False
dtype: bool
```

In [198…

```
# 可以看出資料內沒有空值
# 資料內的空值利用當欄位的平均數值相加
for col in df.columns:
    if df[col].any() == np.nan:
        print(col)
#     df[col] = df[col].apply(lambda x: x ==  if x == ' ' else x)
```

In [199…

```
# 將目標 age 欄位的空值處理，並全部利用 age mean值取代空值
without_NaN_in_age = df.dropna(axis = 0, subset = ['age'])
```

In [200…

```
# 將 age 列，除了空值以外的其他值平均
mean_age = without_NaN_in_age['age'].astype(float).mean()

# 並將age的所有數值平均值取代 age內部的空值
df['age'] = df['age'].fillna(value = mean_age)
df['age']
```

Out[200…

```
0             42
1             13
2             16
3             35
4             16
          ...
1304         14.5
1305    29.881135
1306         26.5
1307           27
1308           29
Name: age, Length: 1309, dtype: object
```

第二部分:

將survived = 0/1分開，拆成2部分並做one-way ANOVA檢驗

In [202…

```
## not_alive_list 代表沒有倖存的乘客
not_alive_list = df[df['survived'] == 0]['age'].astype(float).tolist()

# alive_list 代表倖存的乘客
alive_list = df[df['survived'] == 1]['age'].astype(float).tolist()

print(not_alive_list, end = '\n\n')
print(alive_list, end = '\n\n')
```

```
[42.0, 13.0, 16.0, 30.0, 30.0, 26.0, 40.0, 30.0, 26.0, 20.0, 24.0, 25.0, 35.0, 2.0,
30.0, 25.0, 18.0, 32.0, 4.0, 6.0, 2.0, 38.0, 9.0, 11.0, 39.0, 26.0, 39.0, 20.0, 18.
0, 25.0, 39.0, 26.0, 34.0, 25.0, 18.0, 24.0, 71.0, 57.0, 35.0, 5.0, 9.0, 13.0, 40.0,
21.0, 23.0, 47.0, 17.0, 30.0, 23.0, 20.0, 32.0, 40.0, 18.0, 23.0, 26.0, 28.0, 18.0,
45.0, 27.0, 22.0, 51.0, 29.8811345124283, 24.0, 36.0, 28.0, 19.0, 26.0, 22.0, 23.0,
29.8811345124283, 20.0, 21.0, 25.0, 18.0, 45.0, 42.0, 26.0, 26.0, 6.0, 9.0, 29.88113
45124283, 29.8811345124283, 29.8811345124283, 40.0, 32.0, 21.0, 42.0, 27.0, 41.0, 2
0.0, 48.0, 29.0, 22.0, 29.8811345124283, 22.0, 35.0, 60.0, 25.0, 18.5, 19.0, 18.0, 2
5.0, 45.0, 42.0, 21.0, 30.0, 18.0, 38.0, 29.8811345124283, 17.0, 17.0, 29.8811345124
283, 21.0, 21.0, 21.0, 29.8811345124283, 29.8811345124283, 19.0, 28.0, 24.0, 33.0, 3
7.0, 28.0, 17.0, 44.0, 54.0, 28.0, 49.0, 36.0, 24.0, 46.0, 52.0, 37.0, 29.0, 21.0, 2
9.8811345124283, 29.0, 26.0, 18.0, 27.0, 29.0, 29.8811345124283, 20.0, 24.0, 36.0, 2
4.0, 29.0, 28.0, 47.0, 31.0, 37.0, 31.0, 31.0, 30.0, 70.5, 43.0, 35.0, 27.0, 19.0, 3
0.0, 29.8811345124283, 30.0, 21.0, 59.0, 29.8811345124283, 19.0, 44.0, 70.0, 17.0, 3
9.0, 29.8811345124283, 22.5, 22.0, 19.0, 0.3333, 34.0, 28.0, 27.0, 25.0, 31.0, 24.0,
18.0, 22.0, 21.0, 17.0, 29.8811345124283, 32.0, 16.0, 17.0, 26.0, 29.0, 25.0, 29.881
1345124283, 25.0, 29.8811345124283, 22.0, 36.0, 18.0, 17.0, 42.0, 43.0, 29.881134512
4283, 32.0, 50.0, 54.0, 24.0, 33.0, 42.0, 65.0, 39.0, 23.0, 18.0, 23.0, 16.0, 45.0,
29.8811345124283, 39.0, 17.0, 15.0, 47.0, 29.8811345124283, 21.0, 36.0, 40.5, 18.0,
40.5, 29.8811345124283, 40.0, 18.0, 18.0, 29.8811345124283, 29.8811345124283, 29.881
1345124283, 26.0, 29.8811345124283, 21.0, 9.0, 29.8811345124283, 18.0, 16.0, 48.0, 3
0.0, 19.0, 64.0, 29.8811345124283, 36.0, 29.8811345124283, 29.8811345124283, 29.8811
345124283, 29.8811345124283, 38.0, 37.0, 35.0, 38.0, 34.0, 25.0, 29.8811345124283, 1
6.0, 26.0, 47.0, 29.8811345124283, 24.0, 47.0, 21.0, 21.0, 24.0, 22.0, 24.0, 34.0, 3
0.0, 71.0, 33.0, 41.0, 38.0, 9.0, 1.0, 11.0, 10.0, 16.0, 14.0, 40.0, 43.0, 38.0, 51.
0, 52.0, 32.0, 29.8811345124283, 46.0, 20.0, 37.0, 28.0, 19.0, 24.0, 17.0, 29.881134
5124283, 29.8811345124283, 28.0, 30.0, 20.0, 23.5, 41.0, 26.0, 21.0, 44.0, 29.881134
5124283, 25.0, 28.0, 29.8811345124283, 45.0, 30.0, 40.0, 43.0, 29.8811345124283, 11.
0, 55.0, 42.0, 18.0, 23.0, 28.0, 28.0, 29.8811345124283, 49.0, 24.0, 32.0, 21.0, 29.
8811345124283, 18.0, 55.0, 23.0, 36.0, 50.0, 44.0, 43.0, 28.0, 42.0, 21.0, 29.881134
5124283, 63.0, 60.0, 29.8811345124283, 42.0, 33.0, 30.0, 29.8811345124283, 27.0, 25.
0, 50.0, 29.8811345124283, 42.0, 29.8811345124283, 47.0, 24.0, 22.0, 32.0, 20.0, 48.
0, 17.0, 34.0, 22.0, 33.0, 31.0, 29.0, 49.0, 33.0, 19.0, 29.8811345124283, 29.881134
5124283, 29.8811345124283, 29.8811345124283, 46.0, 23.0, 27.0, 50.0, 20.0, 21.0, 17.
0, 21.0, 34.0, 30.0, 33.0, 22.0, 22.0, 29.8811345124283, 18.5, 29.8811345124283, 35.
0, 29.8811345124283, 32.5, 44.0, 34.5, 58.0, 41.0, 29.8811345124283, 29.881134512428
3, 29.8811345124283, 29.8811345124283, 29.8811345124283, 22.0, 26.0, 57.0, 29.881134
5124283, 1.0, 18.0, 36.0, 29.8811345124283, 29.8811345124283, 31.0, 29.881134512428
3, 26.0, 30.0, 37.0, 29.8811345124283, 29.8811345124283, 29.8811345124283, 29.881134
5124283, 25.0, 22.0, 29.0, 29.0, 29.8811345124283, 29.8811345124283, 29.881134512428
3, 29.8811345124283, 29.8811345124283, 32.0, 34.5, 29.8811345124283, 29.881134512428
3, 36.0, 39.0, 36.0, 29.8811345124283, 24.0, 24.0, 25.0, 45.0, 42.0, 36.0, 30.0, 29.
8811345124283, 28.0, 61.0, 29.8811345124283, 30.0, 26.0, 29.8811345124283, 29.0, 30.
0, 50.0, 20.5, 51.0, 29.8811345124283, 57.0, 29.8811345124283, 22.0, 30.0, 29.881134
5124283, 29.8811345124283, 29.8811345124283, 29.0, 29.8811345124283, 31.0, 30.5, 29.
8811345124283, 29.8811345124283, 35.0, 33.0, 19.0, 29.8811345124283, 30.0, 40.0, 46.
0, 54.0, 32.0, 30.0, 29.8811345124283, 35.0, 46.0, 29.8811345124283, 24.0, 19.0, 29.
8811345124283, 29.8811345124283, 29.8811345124283, 55.5, 29.8811345124283, 39.0, 28.
0, 29.8811345124283, 65.0, 48.0, 44.0, 24.0, 21.0, 28.0, 70.0, 29.8811345124283, 25.
0, 55.0, 27.0, 47.0, 29.8811345124283, 29.8811345124283, 29.8811345124283, 54.0, 39.
0, 34.0, 29.8811345124283, 29.8811345124283, 16.0, 29.8811345124283, 18.0, 62.0, 22.
0, 33.0, 29.8811345124283, 29.8811345124283, 32.5, 37.0, 29.8811345124283, 36.5, 29.
8811345124283, 26.0, 58.0, 19.0, 64.0, 29.0, 28.0, 21.0, 41.0, 28.0, 21.0, 28.5, 61.
0, 29.8811345124283, 29.8811345124283, 29.8811345124283, 29.8811345124283, 29.881134
5124283, 29.8811345124283, 29.8811345124283, 23.0, 28.0, 42.0, 29.8811345124283, 31.
0, 28.0, 20.0, 23.0, 20.0, 20.0, 16.0, 65.0, 39.0, 28.5, 23.0, 2.0, 6.0, 3.0, 8.0, 2
9.0, 1.0, 7.0, 2.0, 16.0, 14.0, 41.0, 28.0, 29.8811345124283, 29.8811345124283, 45.
5, 21.0, 19.0, 29.8811345124283, 32.0, 23.0, 0.75, 3.0, 26.0, 29.8811345124283, 29.
0, 29.8811345124283, 29.8811345124283, 21.0, 25.0, 18.0, 19.0, 22.0, 29.881134512428
```

```
3, 41.0, 29.8811345124283, 24.0, 28.0, 19.0, 29.8811345124283, 25.0, 18.0, 43.0, 29.
8811345124283, 17.0, 24.0, 34.0, 47.0, 27.0, 29.8811345124283, 29.8811345124283, 29.
8811345124283, 29.8811345124283, 36.0, 38.0, 34.0, 38.0, 21.0, 10.0, 4.0, 7.0, 2.0,
8.0, 39.0, 23.0, 22.0, 22.0, 35.0, 29.8811345124283, 29.8811345124283, 29.8811345124
283, 50.0, 47.0, 31.0, 19.0, 29.8811345124283, 29.8811345124283, 29.8811345124283,
2.0, 18.0, 41.0, 46.0, 36.0, 55.0, 50.0, 33.0, 16.0, 29.8811345124283, 61.0, 29.8811
345124283, 25.0, 29.8811345124283, 29.8811345124283, 29.8811345124283, 38.5, 29.8811
345124283, 14.5, 29.8811345124283, 29.8811345124283, 29.8811345124283, 29.8811345124
283, 29.8811345124283, 29.8811345124283, 29.8811345124283, 29.8811345124283, 29.8811
345124283, 24.0, 39.0, 29.8811345124283, 29.8811345124283, 29.8811345124283, 20.0, 2
4.5, 29.8811345124283, 26.0, 29.8811345124283, 25.0, 27.0, 29.8811345124283, 29.8811
345124283, 29.8811345124283, 50.0, 29.8811345124283, 22.0, 29.8811345124283, 40.0, 2
1.0, 59.0, 4.0, 10.0, 9.0, 2.0, 40.0, 45.0, 29.8811345124283, 35.0, 29.881134512428
3, 56.0, 29.8811345124283, 56.0, 24.0, 29.8811345124283, 29.8811345124283, 25.0, 19.
0, 30.0, 29.8811345124283, 57.0, 32.0, 29.8811345124283, 33.0, 21.0, 41.0, 62.0, 29.
8811345124283, 25.0, 60.5, 19.0, 22.0, 67.0, 63.0, 27.0, 2.0, 29.0, 25.0, 61.0, 74.
0, 24.0, 18.5, 14.0, 52.0, 49.0, 34.0, 29.8811345124283, 29.8811345124283, 29.881134
5124283, 29.8811345124283, 29.8811345124283, 32.0, 29.8811345124283, 29.881134512428
3, 30.5, 44.0, 29.8811345124283, 29.8811345124283, 23.0, 36.0, 29.0, 27.0, 40.0, 29.
8811345124283, 11.5, 40.5, 61.0, 10.0, 36.0, 30.0, 29.8811345124283, 33.0, 28.0, 28.
0, 47.0, 18.0, 31.0, 16.0, 31.0, 40.0, 20.0, 14.0, 22.0, 22.0, 47.0, 29.881134512428
3, 30.0, 23.0, 29.8811345124283, 64.0, 29.8811345124283, 29.8811345124283, 60.0, 27.
0, 32.5, 36.0, 66.0, 29.8811345124283, 54.0, 21.0, 57.0, 51.0, 50.0, 27.0, 18.0, 21.
0, 29.8811345124283, 29.8811345124283, 29.8811345124283, 51.0, 29.8811345124283, 28.
5, 21.0, 27.0, 29.8811345124283, 36.0, 62.0, 27.0, 45.5, 29.8811345124283, 29.881134
5124283, 24.0, 14.5, 29.8811345124283, 26.5, 27.0, 29.0]

[35.0, 16.0, 25.0, 28.0, 20.0, 18.0, 0.8333, 18.0, 26.0, 29.0, 0.9167, 19.0, 48.0, 1
7.0, 27.0, 63.0, 36.0, 53.0, 3.0, 5.0, 23.0, 38.0, 45.0, 18.0, 24.0, 13.0, 33.0, 0.7
5, 0.75, 5.0, 24.0, 18.0, 36.0, 20.0, 26.0, 80.0, 50.0, 32.0, 32.0, 19.0, 1.0, 4.0,
12.0, 36.0, 37.0, 47.0, 34.0, 26.0, 19.0, 42.0, 32.0, 29.0, 25.0, 19.0, 35.0, 28.0,
40.0, 30.0, 58.0, 45.0, 22.0, 22.0, 29.8811345124283, 24.0, 15.0, 44.0, 59.0, 40.0,
20.0, 21.0, 60.0, 41.0, 36.0, 42.0, 42.0, 0.8333, 26.0, 22.0, 35.0, 53.0, 36.0, 58.
0, 16.0, 11.0, 14.0, 36.0, 36.0, 29.8811345124283, 76.0, 47.0, 27.0, 33.0, 36.0, 30.
0, 45.0, 29.8811345124283, 32.0, 25.0, 45.0, 26.0, 28.0, 22.0, 18.0, 24.0, 8.0, 31.
0, 39.0, 64.0, 22.0, 22.0, 55.0, 9.0, 3.0, 36.0, 17.0, 36.0, 64.0, 38.0, 45.0, 30.0,
29.0, 51.0, 27.0, 33.0, 27.0, 8.0, 48.0, 28.0, 29.8811345124283, 36.5, 36.0, 30.0,
1.0, 0.1667, 33.0, 24.0, 19.0, 31.0, 17.0, 53.0, 4.0, 54.0, 18.0, 34.0, 19.0, 27.0,
48.0, 30.0, 23.0, 8.0, 34.0, 48.0, 49.0, 24.0, 27.0, 30.0, 22.0, 23.0, 5.0, 38.0, 5
4.0, 29.0, 29.8811345124283, 29.8811345124283, 29.8811345124283, 36.0, 29.8811345124
283, 24.0, 28.0, 23.0, 60.0, 30.0, 50.0, 43.0, 29.8811345124283, 60.0, 48.0, 22.0, 3
5.0, 34.0, 35.0, 22.0, 45.0, 16.0, 29.8811345124283, 49.0, 29.8811345124283, 9.0, 3
1.0, 53.0, 19.0, 58.0, 23.0, 45.0, 24.0, 0.6667, 24.0, 45.0, 25.0, 25.0, 6.0, 48.0,
49.0, 62.0, 35.0, 7.0, 45.0, 27.0, 29.8811345124283, 24.0, 52.0, 29.8811345124283, 2
7.0, 29.8811345124283, 26.0, 22.0, 24.0, 24.0, 48.0, 55.0, 16.0, 44.0, 2.0, 22.0, 2
0.0, 54.0, 51.0, 29.0, 35.0, 35.0, 27.0, 42.0, 29.8811345124283, 38.0, 35.0, 29.8811
345124283, 38.0, 17.0, 49.0, 24.0, 29.0, 21.0, 29.8811345124283, 23.0, 29.8811345124
283, 26.0, 4.0, 1.0, 27.0, 32.0, 32.0, 24.0, 21.0, 4.0, 39.0, 29.8811345124283, 29.8
811345124283, 29.8811345124283, 45.0, 29.8811345124283, 29.8811345124283, 42.0, 45.
0, 4.0, 29.0, 26.0, 25.0, 39.0, 29.8811345124283, 22.0, 26.0, 1.0, 3.0, 22.0, 49.0,
22.0, 17.0, 29.8811345124283, 34.0, 30.0, 35.0, 20.0, 55.0, 16.0, 51.0, 21.0, 42.0,
27.0, 23.0, 32.0, 58.0, 29.8811345124283, 15.0, 24.0, 16.0, 1.0, 24.0, 29.8811345124
283, 29.8811345124283, 29.8811345124283, 18.0, 29.8811345124283, 24.0, 29.8811345124
283, 29.8811345124283, 29.8811345124283, 29.8811345124283, 29.8811345124283, 29.8811
345124283, 36.0, 29.8811345124283, 15.0, 13.0, 41.0, 19.0, 29.8811345124283, 21.0, 3
3.0, 37.0, 30.0, 29.8811345124283, 6.0, 27.0, 29.8811345124283, 29.8811345124283, 2
9.8811345124283, 29.8811345124283, 29.8811345124283, 29.8811345124283, 29.8811345124
283, 24.0, 29.8811345124283, 29.8811345124283, 29.8811345124283, 15.0, 1.0, 20.0, 1
9.0, 14.0, 2.0, 3.0, 31.0, 23.0, 19.0, 12.0, 14.0, 18.0, 26.0, 39.0, 20.0, 29.0, 22.
0, 29.8811345124283, 29.8811345124283, 29.8811345124283, 29.8811345124283, 29.881134
5124283, 22.0, 39.0, 9.0, 32.0, 29.8811345124283, 31.0, 22.0, 22.0, 29.881134512428
3, 29.0, 50.0, 22.0, 17.0, 30.0, 25.0, 29.8811345124283, 29.8811345124283, 29.881134
5124283, 52.0, 21.0, 19.0, 32.0, 32.0, 30.0, 56.0, 2.0, 8.0, 33.0, 30.0, 28.0, 29.88
11345124283, 0.8333, 3.0, 24.0, 50.0, 29.8811345124283, 43.0, 45.0, 33.0, 29.8811345
124283, 33.0, 54.0, 21.0, 29.8811345124283, 13.0, 18.0, 21.0, 48.0, 29.881134512428
3, 24.0, 21.0, 29.8811345124283, 1.0, 4.0, 24.0, 25.0, 35.0, 30.0, 34.0, 29.0, 25.0,
29.8811345124283, 40.0, 18.0, 35.0, 39.0, 56.0, 20.0, 30.0, 18.0, 30.0, 28.0, 40.0,
18.0, 29.8811345124283, 24.0, 23.0, 6.0, 45.0, 40.0, 29.8811345124283, 32.0, 23.0, 5
4.0, 43.0, 52.0, 62.0, 31.0, 16.0, 44.0, 14.0, 48.0, 18.0, 39.0, 48.0, 29.8811345124
```

```
283, 25.0, 17.0, 39.0, 0.4167, 16.0, 29.8811345124283, 29.8811345124283, 50.0, 25.0,
7.0, 9.0, 29.0, 28.0, 27.0, 31.0, 18.0, 63.0, 22.0, 31.0, 35.0, 31.0, 60.0, 12.0, 4
0.0, 32.5, 29.0, 2.0, 4.0, 29.0, 0.9167, 5.0, 33.0, 38.0, 55.0, 31.0, 45.0, 50.0, 3
1.0, 47.0, 21.0, 29.8811345124283, 21.0, 31.0, 29.8811345124283, 26.0, 15.0, 36.0]
```

事前檢驗

In [203…

```
## 常態檢定: 確認母體為常態分配，才可以代表母體作分析
## 利用shapiro模型中進行計算，以利確認資料是否呈現常態分佈
## 計算出來的 p-value < 常設標準的 0.05，可以當作常態分布來看
import scipy.stats as st
print(st.shapiro(not_alive_list))
print(st.shapiro(alive_list))
```

```
ShapiroResult(statistic=0.9392056465148926, pvalue=1.1460844640092702e-17)
ShapiroResult(statistic=0.9725306630134583, pvalue=4.516658691500197e-08)
```

In [204…

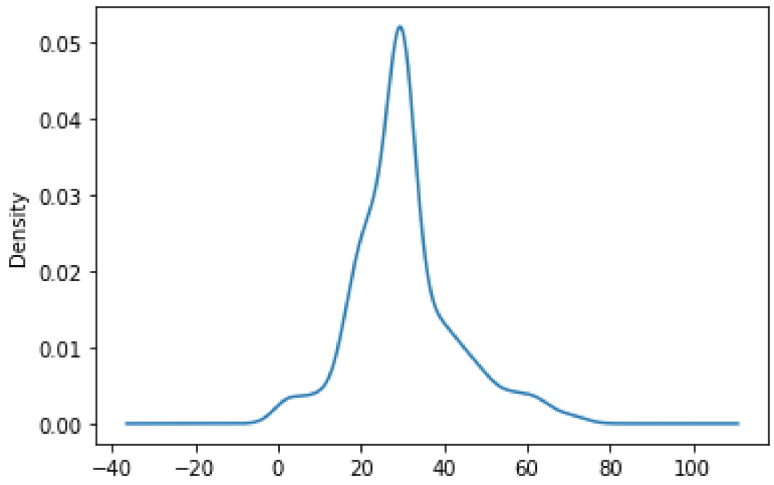
```
## 同質性檢定: 用來確保資料取自變異數相等的母體
## p-value < 0.05，因此有顯著的證據證明在兩者數據之間的變異數是不相等的
st.levene(not_alive_list, alive_list, center='mean')
```

Out[204… `LeveneResult(statistic=12.115059343225045, pvalue=0.0005166346986644329)`

生存/喪生的人分布圖表的呈現

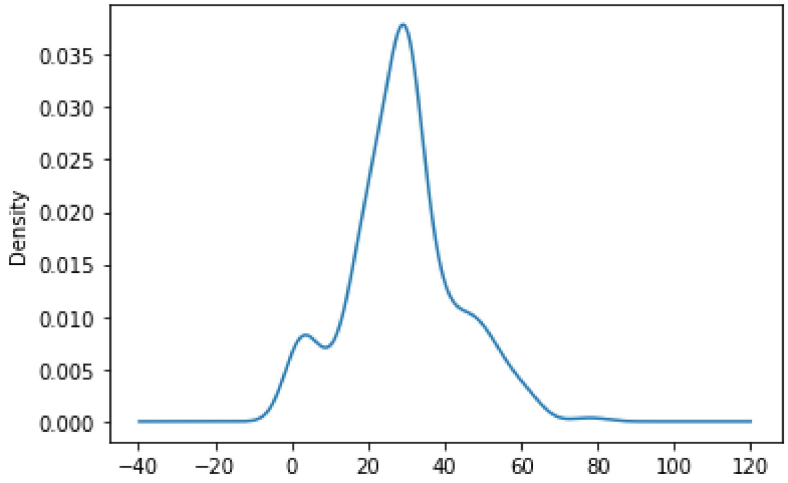
In [17]:

```
not_alive = not_alive_list.plot.kde()
```



In [18]:

```
alive = alive_list.plot.kde()
```



In [205…

```
## 變異數分析
## p-value > 0.05，因此有顯著的證據證明在活下去和沒有活下去的平均年紀是不相等的
## 結論: 活下去和沒有活下去的平均年紀是不相等的
f_value, p_value = st.f_oneway(not_alive_list, alive_list)
p_value
```

Out[205… 0.06942958640263792

三、事後檢定

B. Among these passengers, are there age differences in the first-class cabin (pclass=1), second-class cabin (pclass=2), and third-class cabin (pclass=3) group? Assume the population variances of age in each level of class are the same. 請問三種艙等的乘客年紀是否有差？

使用 one_way ANOVA Test: 因為探討的因素為比較不同艙等之間平均年紀的關係。one-way ANOVA為單尾檢定

H0: 不同艙等的乘客的平均年紀相等

Ha: 至少有一艙等的乘客的平均年紀和其他不同

In [30]:

```
## 第一步: 確認 pclass的欄位沒有空值
## 然後將剛才的 age列，依照 pclass分類

## pclass_1 代表第一階級的乘客
pclass_1 = df[df['pclass'] == 1]['age'].astype(float)

## pclass_2 代表第二階級的乘客
pclass_2 = df[df['pclass'] == 2]['age'].astype(float)

## pclass_3 代表第二階級的乘客
pclass_3 = df[df['pclass'] == 3]['age'].astype(float)

print(pclass_1, end = '\n\n')
print(pclass_2, end = '\n\n')
print(pclass_3, end = '\n\n')
```

```
21      29.000000
23       0.916700
24       2.000000
```

```
25      30.000000
26      25.000000
          ...
1288    21.000000
1289    31.000000
1294    29.881135
1296    62.000000
1299    36.000000
Name: age, Length: 323, dtype: float64

6       30.000000
7       28.000000
16      30.000000
43      18.000000
44      25.000000
          ...
1265    29.881135
1278    31.000000
1285    29.881135
1295    26.000000
1303    24.000000
Name: age, Length: 277, dtype: float64

0       42.000000
1       13.000000
2       16.000000
3       35.000000
4       16.000000
          ...
1304    14.500000
1305    29.881135
1306    26.500000
1307    27.000000
1308    29.000000
Name: age, Length: 709, dtype: float64
```

In [31]:

```
## 常態檢定: 確認母體為常態分配，才可以代表母體作分析
## 利用shapiro模型中進行計算，以利確認資料是否呈現常態分佈
## 計算出來的 p-value < 普遍定義顯著水準的0.05，這些資料的分布可以當作常態分布來看
import scipy.stats as st
print(st.shapiro(pclass_1))
print(st.shapiro(pclass_2))
print(st.shapiro(pclass_3))
```

```
ShapiroResult(statistic=0.9825005531311035, pvalue=0.0005682504270225763)
ShapiroResult(statistic=0.9704985618591309, pvalue=1.7886193745653145e-05)
ShapiroResult(statistic=0.9314501285552979, pvalue=1.7119331611758686e-17)
```

In [52]:

```
## 同質性檢定: 用來確保資料取自變異數相等的母體
## p-value < 0.05，因此有顯著的證據證明不同船艙的人平均年紀是不相等的
## 不同艙等的平均年紀是不相等的
st.levene(pclass_1, pclass_2, pclass_3, center='mean')
```

Out[52]:

```
LeveneResult(statistic=32.16761492976818, pvalue=2.30663588649374e-14)
```

In [33]:

```
## 這邊計算出的p_value也是非常小，代表三者的 mean是顯著不盡相同的
## 所以不同船艙的人平均年紀是不相同的
f_value, p_value = st.f_oneway(pclass_1, pclass_2, pclass_3)
p_value
```

Out[33]:

```
7.790969240746352e-44
```

事後檢定

In [64]:
```
## 刪除不必要的欄位
df_2 = df.copy()
df_2 = df.drop(columns = ['survived', 'Residence', 'name', 'sibsp', 'parch', 'ticket
df_2
```

Out[64]:

	pclass	age
0	3	42
1	3	13
2	3	16
3	3	35
4	3	16
...	...	...
1304	3	14.5
1305	3	29.881135
1306	3	26.5
1307	3	27
1308	3	29

1309 rows × 2 columns

In [68]:
```
## 算出 one-way ANOVA Table 當作參考
from pingouin import pairwise_tukey
m_comp = pairwise_tukey(data = df_2.astype(float), dv = 'age', between = 'pclass')

## 從表中可以看到一些適合作為後續分析的欄位：
## 1. Diff：不同艙等之間的年紀平均差異
## 2. Se：標準誤，可用來協助估算平均值的範圍區間
table = m_comp.drop(columns = ['mean(A)', 'mean(B)', 'T', 'p-tukey', 'hedges'])
table
```

Out[68]:

	A	B	diff	se
0	1.0	2.0	8.511237	0.978549
1	1.0	3.0	11.737347	0.802165
2	2.0	3.0	3.226110	0.846688

C. Among these passengers, are the survival status (survived) different among passengers in different levels of cabins (pclass)? 乘客是否存活和乘坐的艙等是否有關？

使用 Chi-Square Test: 因為探討的因素皆為 Qualitative Data, 比較之間的關係。而所有的卡方檢定均為單尾檢定

H0: 乘坐的艙位等級和平均存活的機率相等

Ha: 至少有一艙位的平均生存機率和其他不同

In [39]:
```
## 第一步: 確認 pclass& survived的欄位沒有空值
## 刪除不必要的欄位
df_3 = df.copy()
df_3 = df.drop(columns = ['Residence', 'name', 'age', 'sibsp', 'parch', 'ticket', 'f
df_3
```

Out[39]:

	pclass	survived
0	3	0
1	3	0
2	3	0
3	3	1
4	3	1
...	...	...
1304	3	0
1305	3	0
1306	3	0
1307	3	0
1308	3	0

1309 rows × 2 columns

In [41]:
```
## 因為資料2者皆為 Qulitative Data，故使用卡方檢定來判斷
## 而且資料數量皆有大於5
## 將資料轉換成卡方檢定可以分析的「頻率」
from collections import Counter
frequency_count = Counter(df_3['pclass'])
frequency_count
```

Out[41]:
```
Counter({3: 709, 2: 277, 1: 323})
```

In [45]:
```
# 將清單中的 pclass取出，稱為清單 f1, 將頻率取出，稱為清單 f2
## 並將這些製作成卡方檢定需要的頻率清單
f1 = list(frequency_count.keys())
f2 = list(frequency_count.values())
frequency_table = pd.DataFrame(zip(f1,f2),
                               columns=['pclass','freq'])
chi_table = frequency_table
chi_table
```

Out[45]:

	pclass	freq
0	3	709
1	2	277
2	1	323

In [47]:
```
## 將頻率表單轉換成卡方檢定形式的資料格式 - array
import numpy as np
obs = np.array([chi_table.iloc[0,:].tolist(),
```

```
                chi_table.iloc[1,:].tolist(),
                chi_table.iloc[2,:].tolist(),])
obs
```

Out[47]:
```
array([[  3, 709],
       [  2, 277],
       [  1, 323]])
```

In [48]:
```
## 帶入卡方檢定
import scipy.stats
## 得到的資料依序為: 檢定值，p-value, 自由度, 檢測內容
scipy.stats.chi2_contingency(obs, correction = False)
```

Out[48]:
```
(0.5916796090791416,
 0.7439065889339367,
 2,
 array([[  3.2486692, 708.7513308],
        [  1.2730038, 277.7269962],
        [  1.478327 , 322.521673 ]]))
```

In [50]:
```
## 重點提出 p-value來檢驗
## 這裡可以看出 p-value > 常規預設的 0.05
## 沒有足夠多的證據證明 "H0會被拒絕"
## 所以「艙位的等級」和「能不能生還」沒有太大的關係
p_value = scipy.stats.chi2_contingency(obs, correction = False)[1]
p_value
```

Out[50]:
```
0.7439065889339367
```

---

2. The Crime Rate data set gives a variety of variables by US state at two time points 10 years apart. Dataset, description

a. Is the Crime Rate (CrimeRate) in the southern states (Southern=1) higher than the other states (Southern=0)? 在南方的州犯罪率(CrimeRate) 是否比其他州高？

使用 z Test: 比較十年前和十年後平均犯罪率是否有差異，單尾檢定

H0: 南方的犯罪率比其他州低

Ha: 南方的犯罪率比其他州高

In [2]:
```
import pandas as pd
import numpy as np

# load data file(原始資料)
origin_df = pd.read_csv("D:\統計學\Final_Project\Crime_R.csv")
# df 為操作之資料
df = origin_df.copy()
df
```

Out[2]:

	CrimeRate	Youth	Southern	Education	ExpenditureYear0	LabourForce	Males	MoreMales	Sta

	CrimeRate	Youth	Southern	Education	ExpenditureYear0	LabourForce	Males	MoreMales	Sta
0	45.5	135	0	12.4	69	540	965	0	
1	52.3	140	0	10.9	55	535	1045	1	
2	56.6	157	1	11.2	47	512	962	0	
3	60.3	139	1	11.9	46	480	968	0	
4	64.2	126	0	12.2	106	599	989	0	
5	67.6	128	0	13.5	67	624	972	0	
6	70.5	130	0	14.1	63	641	984	0	
7	73.2	143	0	12.9	66	537	977	0	
8	75.0	141	0	12.9	56	523	968	0	
9	78.1	133	0	11.4	51	599	1024	1	
10	79.8	142	1	12.9	45	533	969	0	
11	82.3	123	0	12.5	97	526	948	0	
12	83.1	135	0	13.6	62	595	986	0	
13	84.9	121	0	13.2	118	547	964	0	
14	85.6	166	1	11.4	58	521	973	0	
15	88.0	140	0	12.9	71	632	1029	1	
16	92.3	126	0	12.7	74	602	984	0	
17	94.3	130	0	13.3	128	536	934	0	
18	95.3	125	0	12.0	90	586	964	0	
19	96.8	151	1	10.0	58	510	950	0	
20	97.4	152	1	10.8	57	530	986	0	
21	98.7	162	1	12.1	75	522	996	0	
22	99.9	149	1	10.7	61	515	953	0	
23	103.0	177	1	11.0	58	638	974	0	
24	104.3	134	0	12.5	75	595	972	0	
25	105.9	130	0	13.4	90	623	1049	1	
26	106.6	157	1	11.1	65	553	955	0	
27	107.2	148	0	13.7	72	601	998	0	
28	108.3	126	0	13.8	97	542	990	0	
29	109.4	135	1	11.4	123	537	978	0	
30	112.1	142	1	10.9	81	497	956	0	
31	114.3	127	1	12.8	82	519	982	0	
32	115.1	131	0	13.7	78	574	1038	1	
33	117.2	136	0	12.9	95	574	1012	1	
34	119.7	119	0	11.9	166	521	938	0	
35	121.6	147	1	13.9	63	560	972	0	

	CrimeRate	Youth	Southern	Education	ExpenditureYear0	LabourForce	Males	MoreMales	Sta
36	123.4	145	1	11.7	82	560	981	0	
37	127.2	132	0	10.4	87	564	953	0	
38	132.4	152	0	12.0	82	571	1018	1	
39	135.5	125	0	12.5	113	567	985	0	
40	137.8	141	0	14.2	109	591	985	0	
41	140.8	150	0	12.0	109	531	964	0	
42	145.4	131	1	12.2	115	542	969	0	
43	149.3	143	0	12.3	103	583	1012	1	
44	154.3	124	0	12.3	121	580	966	0	
45	157.7	136	0	15.1	149	577	994	0	
46	161.8	131	0	13.2	160	631	1071	1	

47 rows × 27 columns

In [3]:
```
# 這邊是利用正則表達式，將pandas中的空白，替換成NaN
df = df.replace(r'\s+', np.nan, regex = True)
df
```

Out[3]:

	CrimeRate	Youth	Southern	Education	ExpenditureYear0	LabourForce	Males	MoreMales	Sta
0	45.5	135	0	12.4	69	540	965	0	
1	52.3	140	0	10.9	55	535	1045	1	
2	56.6	157	1	11.2	47	512	962	0	
3	60.3	139	1	11.9	46	480	968	0	
4	64.2	126	0	12.2	106	599	989	0	
5	67.6	128	0	13.5	67	624	972	0	
6	70.5	130	0	14.1	63	641	984	0	
7	73.2	143	0	12.9	66	537	977	0	
8	75.0	141	0	12.9	56	523	968	0	
9	78.1	133	0	11.4	51	599	1024	1	
10	79.8	142	1	12.9	45	533	969	0	
11	82.3	123	0	12.5	97	526	948	0	
12	83.1	135	0	13.6	62	595	986	0	
13	84.9	121	0	13.2	118	547	964	0	
14	85.6	166	1	11.4	58	521	973	0	
15	88.0	140	0	12.9	71	632	1029	1	
16	92.3	126	0	12.7	74	602	984	0	
17	94.3	130	0	13.3	128	536	934	0	

	CrimeRate	Youth	Southern	Education	ExpenditureYear0	LabourForce	Males	MoreMales	Sta
18	95.3	125	0	12.0	90	586	964	0	
19	96.8	151	1	10.0	58	510	950	0	
20	97.4	152	1	10.8	57	530	986	0	
21	98.7	162	1	12.1	75	522	996	0	
22	99.9	149	1	10.7	61	515	953	0	
23	103.0	177	1	11.0	58	638	974	0	
24	104.3	134	0	12.5	75	595	972	0	
25	105.9	130	0	13.4	90	623	1049	1	
26	106.6	157	1	11.1	65	553	955	0	
27	107.2	148	0	13.7	72	601	998	0	
28	108.3	126	0	13.8	97	542	990	0	
29	109.4	135	1	11.4	123	537	978	0	
30	112.1	142	1	10.9	81	497	956	0	
31	114.3	127	1	12.8	82	519	982	0	
32	115.1	131	0	13.7	78	574	1038	1	
33	117.2	136	0	12.9	95	574	1012	1	
34	119.7	119	0	11.9	166	521	938	0	
35	121.6	147	1	13.9	63	560	972	0	
36	123.4	145	1	11.7	82	560	981	0	
37	127.2	132	0	10.4	87	564	953	0	
38	132.4	152	0	12.0	82	571	1018	1	
39	135.5	125	0	12.5	113	567	985	0	
40	137.8	141	0	14.2	109	591	985	0	
41	140.8	150	0	12.0	109	531	964	0	
42	145.4	131	1	12.2	115	542	969	0	
43	149.3	143	0	12.3	103	583	1012	1	
44	154.3	124	0	12.3	121	580	966	0	
45	157.7	136	0	15.1	149	577	994	0	
46	161.8	131	0	13.2	160	631	1071	1	

47 rows × 27 columns

In [4]:
```
# 檢查資料/目標columns是否有空值，確保資料的完整性
# 並發現，資料內部columns都沒有空值
df.isnull().any()
```

Out[4]:
```
CrimeRate              False
Youth                  False
```

```
Southern               False
Education              False
ExpenditureYear0       False
LabourForce            False
Males                  False
MoreMales              False
StateSize              False
YouthUnemployment      False
MatureUnemployment     False
HighYouthUnemploy      False
Wage                   False
BelowWage              False
CrimeRate10            False
Youth10                False
Education10            False
ExpenditureYear10      False
LabourForce10          False
Males10                False
MoreMales10            False
StateSize10            False
YouthUnemploy10        False
MatureUnemploy10       False
HighYouthUnemploy10    False
Wage10                 False
BelowWage10            False
dtype: bool
```

In [6]:

```
## 將 CrimeRate和 Southern兩種欄位挑出來看
df_1 = df[['CrimeRate', 'Southern']]
df_1
```

Out[6]:

	CrimeRate	Southern
0	45.5	0
1	52.3	0
2	56.6	1
3	60.3	1
4	64.2	0
5	67.6	0
6	70.5	0
7	73.2	0
8	75.0	0
9	78.1	0
10	79.8	1
11	82.3	0
12	83.1	0
13	84.9	0
14	85.6	1
15	88.0	0
16	92.3	0
17	94.3	0
18	95.3	0

	CrimeRate	Southern
19	96.8	1
20	97.4	1
21	98.7	1
22	99.9	1
23	103.0	1
24	104.3	0
25	105.9	0
26	106.6	1
27	107.2	0
28	108.3	0
29	109.4	1
30	112.1	1
31	114.3	1
32	115.1	0
33	117.2	0
34	119.7	0
35	121.6	1
36	123.4	1
37	127.2	0
38	132.4	0
39	135.5	0
40	137.8	0
41	140.8	0
42	145.4	1
43	149.3	0
44	154.3	0
45	157.7	0
46	161.8	0

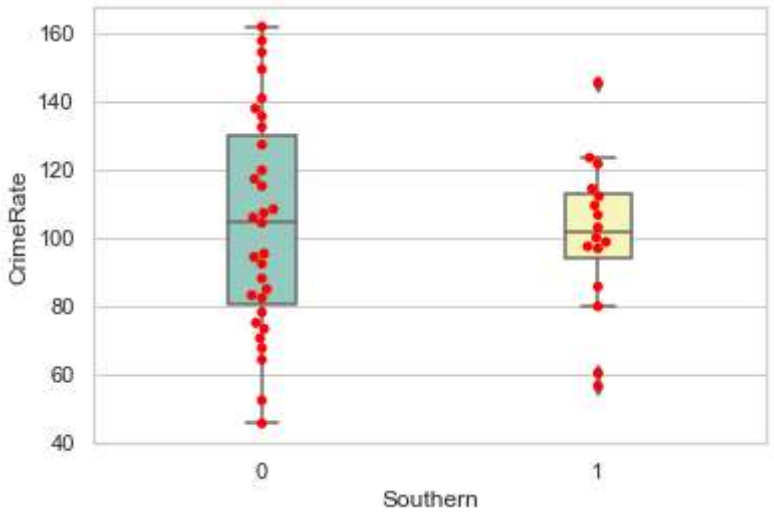
In [7]:
```
is_southern = df_1[df_1['Southern'] == 1]['CrimeRate'].astype(float)
not_southern = df_1[df_1['Southern'] == 0]['CrimeRate'].astype(float)

print(is_southern)
print(not_southern)
```

```
2      56.6
3      60.3
10     79.8
14     85.6
19     96.8
20     97.4
```

```
21     98.7
22     99.9
23    103.0
26    106.6
29    109.4
30    112.1
31    114.3
35    121.6
36    123.4
42    145.4
Name: CrimeRate, dtype: float64
0      45.5
1      52.3
4      64.2
5      67.6
6      70.5
7      73.2
8      75.0
9      78.1
11     82.3
12     83.1
13     84.9
15     88.0
16     92.3
17     94.3
18     95.3
24    104.3
25    105.9
27    107.2
28    108.3
32    115.1
33    117.2
34    119.7
37    127.2
38    132.4
39    135.5
40    137.8
41    140.8
43    149.3
44    154.3
45    157.7
46    161.8
Name: CrimeRate, dtype: float64
```

In [8]:

```
## 畫盒狀圖看不同位置的州數據的分布
import seaborn as sns
sns.set(style="whitegrid")
ax = sns.boxplot(x = "Southern", y = "CrimeRate", data = df_1, width=0.2, palette="S
ax = sns.swarmplot(x = "Southern", y = "CrimeRate", data = df_1, color = "red")
```



In [9]:
```
##  檢查資料是否為常態分布
## 這裡的 p-value > 常設的標準 0.05
## 代表在南方州或者是其他州犯罪率的分布皆為常態分布
import scipy.stats
print(scipy.stats.shapiro(is_southern))
print(scipy.stats.shapiro(not_southern))
```

```
ShapiroResult(statistic=0.9599415063858032, pvalue=0.6607295870780945)
ShapiroResult(statistic=0.9717987179756165, pvalue=0.5698212385177612)
```

In [82]:
```
## 變異數檢測
## 這裡的 p-value < 常設的顯著標準 0.05，但是可以近似，表示兩者的標準差相等
## 代表在南方州或者是其他州犯罪率的標準差相等
import scipy.stats
scipy.stats.levene(is_southern, not_southern, center = 'mean')
```

Out[82]:
```
LeveneResult(statistic=4.138603328571712, pvalue=0.0478345650089015)
```

In [10]:
```
## 現在做 z-Test 檢測，檢測南方的犯罪率是否有比其他州來的高(Ha: 南方犯罪率大於其他州)

import statsmodels.stats.weightstats
statsmodels.stats.weightstats.ztest(is_southern, not_southern, alternative='larger')

## p_value > 顯著標準 0.05，代表沒有足夠多的證據表示「南方的犯罪率比其他州高」
## H0沒有被拒絕，南方的犯罪率比其他州低
p_value = statsmodels.stats.weightstats.ztest(is_southern, not_southern, alternative
p_value
```

Out[10]:
```
0.6402663660609731
```

2. Have crime rates increased in 10 years (CrimeRate vs. CrimeRate10)? 比較各州資料，十年後的犯罪率是否比十年前高？

使用 paired t Test: 比較兩平均犯罪率是否有顯著差異，單尾檢定

H0: 十年後平均犯罪率比十年前低

Ha: 十年後平均犯罪率比十年前高

In [43]:
```
## 將 CrimeRate和 Southern兩種欄位挑出來看
df_2 = df[['CrimeRate', 'CrimeRate10']]
df_2
```

Out[43]:

	CrimeRate	CrimeRate10
0	45.5	26.5
1	52.3	35.9
2	56.6	37.1
3	60.3	42.7
4	64.2	46.7
5	67.6	47.9
6	70.5	50.6
7	73.2	55.9

	CrimeRate	CrimeRate10
8	75.0	61.8
9	78.1	65.4
10	79.8	71.4
11	82.3	75.4
12	83.1	77.3
13	84.9	78.6
14	85.6	80.6
15	88.0	82.2
16	92.3	87.5
17	94.3	92.9
18	95.3	94.1
19	96.8	96.2
20	97.4	97.8
21	98.7	99.9
22	99.9	101.4
23	103.0	103.5
24	104.3	104.5
25	105.9	106.4
26	106.6	107.8
27	107.2	110.1
28	108.3	110.5
29	109.4	113.5
30	112.1	116.3
31	114.3	119.7
32	115.1	124.5
33	117.2	127.8
34	119.7	129.8
35	121.6	130.7
36	123.4	132.5
37	127.2	134.6
38	132.4	137.5
39	135.5	140.5
40	137.8	145.7
41	140.8	150.6
42	145.4	157.3
43	149.3	162.7

	CrimeRate	CrimeRate10
44	154.3	169.6
45	157.7	177.2
46	161.8	178.2

In [44]:
```
##  檢查資料是否為常態分布
## 這裡的 p-value > 常設的標準 0.05
## 代表十年後和十年前的平均犯罪率皆為常態分布
import scipy.stats
print(scipy.stats.shapiro(df_2['CrimeRate'].tolist()))
print(scipy.stats.shapiro(df_2['CrimeRate10'].tolist()))
```

```
ShapiroResult(statistic=0.9863165020942688, pvalue=0.8507899045944214)
ShapiroResult(statistic=0.9819384217262268, pvalue=0.6736380457878113)
```

In [46]:
```
## 變異數檢測
## 這裡的 p-value < 常設的顯著標準 0.05，但是可以近似，表示兩者的標準差相等
## 代表十年後和十年前的平均犯罪率的母體標準差相等
import scipy.stats
scipy.stats.levene(df_2['CrimeRate'].tolist(), df_2['CrimeRate10'].tolist(), center
```

Out[46]:
```
LeveneResult(statistic=4.136167346634216, pvalue=0.04485619711837613)
```

In [48]:
```
## 使用 t-test做檢定，來確定十年前後十年後的犯罪率差別
import scipy.stats

## p_value > 標準信心水準 0.05，代表沒有足夠證據證明「十年後平均犯罪率比十年前高」
## 所以H0被接受，十年後平均犯罪率比十年前高
t_score, p_value = scipy.stats.ttest_ind(df_2['CrimeRate'].tolist(), df_2['CrimeRate
p_value
```

Out[48]:
```
0.9175511889487775
```

3. Divide the education time (Education) into high education time (>13), median education time (>11 and <=13), and low education time (<=11). Are the Crime Rate (CrimeRate) different among these education groups? Assume the population variances of Crime Rate are the same.將接受教育的時間長度分組，請問各組的犯罪率是否有差異呢？

使用 one_way ANOVA Test: 因為探討的因素為比較不同受教時間的關係。one-way ANOVA為單尾檢定

H0: 犯罪率在各個受教時間皆相等

Ha: 至少有一犯罪率在各個受教時間不同

In [11]:
```
## 將 CrimeRate和 Southern兩種欄位挑出來看
df_3 = df[['CrimeRate', 'Education']]
df_3
```

Out[11]:

	CrimeRate	Education

	CrimeRate	Education
0	45.5	12.4
1	52.3	10.9
2	56.6	11.2
3	60.3	11.9
4	64.2	12.2
5	67.6	13.5
6	70.5	14.1
7	73.2	12.9
8	75.0	12.9
9	78.1	11.4
10	79.8	12.9
11	82.3	12.5
12	83.1	13.6
13	84.9	13.2
14	85.6	11.4
15	88.0	12.9
16	92.3	12.7
17	94.3	13.3
18	95.3	12.0
19	96.8	10.0
20	97.4	10.8
21	98.7	12.1
22	99.9	10.7
23	103.0	11.0
24	104.3	12.5
25	105.9	13.4
26	106.6	11.1
27	107.2	13.7
28	108.3	13.8
29	109.4	11.4
30	112.1	10.9
31	114.3	12.8
32	115.1	13.7
33	117.2	12.9
34	119.7	11.9
35	121.6	13.9

	CrimeRate	Education
36	123.4	11.7
37	127.2	10.4
38	132.4	12.0
39	135.5	12.5
40	137.8	14.2
41	140.8	12.0
42	145.4	12.2
43	149.3	12.3
44	154.3	12.3
45	157.7	15.1
46	161.8	13.2

In [24]:

```
## 將不同受教時間分組，分成3組時間區段
Edu_over_13 = df_3[df_3['Education'].astype(float) > 13]['CrimeRate']
Edu_betw_11_13= df_3[df_3['Education'].astype(float).between(11, 13)]['CrimeRate']
Edu_under_11= df_3[(df_3['Education'].astype(float) <= 11)]['CrimeRate']

print(Edu_over_13, end = '\n\n')
print(Edu_betw_11_13,  end = '\n\n')
print(Edu_under_11,  end = '\n\n')
```

```
5      67.6
6      70.5
12     83.1
13     84.9
17     94.3
25    105.9
27    107.2
28    108.3
32    115.1
35    121.6
40    137.8
45    157.7
46    161.8
Name: CrimeRate, dtype: float64

0      45.5
2      56.6
3      60.3
4      64.2
7      73.2
8      75.0
9      78.1
10     79.8
11     82.3
14     85.6
15     88.0
16     92.3
18     95.3
21     98.7
23    103.0
24    104.3
26    106.6
29    109.4
31    114.3
33    117.2
```

```
34    119.7
36    123.4
38    132.4
39    135.5
41    140.8
42    145.4
43    149.3
44    154.3
Name: CrimeRate, dtype: float64

1      52.3
19     96.8
20     97.4
22     99.9
23    103.0
30    112.1
37    127.2
Name: CrimeRate, dtype: float64
```

In [26]:

```
## 常態檢定: 確認資料呈現常態分配，才可以代表母體
import scipy.stats as st
## 3個 p_value > 常設標準 0.05，因此沒有顯著的證據證明三者不符合常態分布
print(st.shapiro(Edu_over_13))
print(st.shapiro(Edu_betw_11_13))
print(st.shapiro(Edu_under_11))
```

```
ShapiroResult(statistic=0.9455904960632324, pvalue=0.5332165956497192)
ShapiroResult(statistic=0.979121744632721, pvalue=0.8288686275482178)
ShapiroResult(statistic=0.8563746213912964, pvalue=0.1404404491186142)
```

In [27]:

```
## 同質性檢定: 確保資料取自變異數相等的母體
## p_value > 常設標準 0.05，因此沒有顯著的證據證明三個群體的變異數相同
## 3個受教時間的分群具有相同的變異數
st.levene(Edu_over_13, Edu_betw_11_13, Edu_under_11, center='mean')
```

Out[27]:

```
LeveneResult(statistic=1.0659152225890873, pvalue=0.3529471344114184)
```

In [28]:

```
## 變異數分析
## p_value > 常設標準 0.05，因此沒有顯著的證據證明三個群體的犯罪率會因受教時間不同而不同
f_value, p_value = st.f_oneway(Edu_over_13, Edu_betw_11_13, Edu_under_11)
p_value
```

Out[28]:

```
0.6597571348902196
```

4. Is there a relationship between high youth unemployment (HighYouthUnemploy) and southern states (Southern)? 請問是否有高比例的青年失業和是不是南方的州有無關聯？

使用 Chi-Square Test: 因為探討的因素皆為 Qualitative Data, 且為探討兩種因素之間的關係。而所有的卡方檢定均為單尾檢定

H0: 高比例青年失業和南方的州無關

Ha: 高比例青年失業和南方的州有關

In [29]:

```
## 將 CrimeRate和 Southern兩種欄位挑出來看
df_4 = df[['HighYouthUnemploy', 'Southern']]
df_4
```

Out[29]:

	HighYouthUnemploy	Southern
0	1	0
1	1	0
2	0	1
3	0	1
4	1	0
5	1	0
6	1	0
7	1	0
8	0	0
9	1	0
10	0	1
11	0	0
12	0	0
13	0	0
14	0	1
15	1	0
16	1	0
17	0	0
18	0	0
19	0	1
20	0	1
21	0	1
22	0	1
23	0	1
24	0	0
25	0	0
26	0	1
27	1	0
28	0	0
29	0	1
30	0	1
31	0	1
32	1	0
33	1	0
34	0	0
35	1	1

	HighYouthUnemploy	Southern
36	0	1
37	0	0
38	1	0
39	0	0
40	1	0
41	0	0
42	0	1
43	0	0
44	0	0
45	0	0
46	0	0

In [35]:
```
## 因為資料2者皆為 Qulitative Data，故使用卡方檢定來判斷
## 而且資料數量皆有大於5
## 將資料轉換成卡方檢定可以分析的「頻率」
from collections import Counter
frequency_count = Counter(df_4['HighYouthUnemploy'])
print("'HighYouthUnemploy = 1' means the unemployeed person is from southern, and vi
frequency_count
```

```
'HighYouthUnemploy = 1' means the unemployeed person is from southern, and vice vers
a.
```

Out[35]:
```
Counter({1: 15, 0: 32})
```

In [36]:
```
## 將清單中的 HighYouthUnemploy，稱為清單 f1, 將頻率取出，稱為清單 f2
## 並將這些製作成卡方檢定需要的頻率清單
f1 = list(frequency_count.keys())
f2 = list(frequency_count.values())
frequency_table = pd.DataFrame(zip(f1,f2),
                               columns=['HighYouthUnemploy','freq'])
chi_table = frequency_table
chi_table
```

Out[36]:

	HighYouthUnemploy	freq
0	1	15
1	0	32

In [39]:
```
## 將頻率表單轉換成卡方檢定形式的資料格式 - array
## 並將這個 array定義成為 obs(觀察值)，並將之帶入卡方檢定中，取得 p_value
import numpy as np
obs = np.array([chi_table.iloc[0,:].tolist(),
                chi_table.iloc[1,:].tolist(),])
obs
```

Out[39]:
```
array([[ 1, 15],
       [ 0, 32]])
```

In [42]:
```
## 將 obs(觀察值)帶入卡方檢定中，取得 p_value
```

```
import scipy.stats
scipy.stats.chi2_contingency(obs, correction = False)
## p_value > 標準水準 0.05，因此可以了解沒有足夠的證據證明「高比例青年失業和南方的州有關」
## H0被接受，高比例青年失業和南方的州無關
p_value = scipy.stats.chi2_contingency(obs, correction = False)[1]
p_value
```

Out[42]: 0.1529527338441347

---

4. In the JHU CSSE COVID-19 Dataset

In the global dataset, is the Fatality Ratio (Case_Fatality_Ratio) on 2021/06/07 (file name 06-06-2021) lower than the Fatality Ratio on 2021/05/07? (file name 05-06-2021) (description)以全球的資料來看，2021/06/07的疾病死亡比例是否有比2021/05/07的死亡比例低呢？ 資料備用連結：0607、0507

使用 z Test: 比較2021/06/07的疾病死亡比例是否有比2021/05/07的死亡比例是否有大小差異，單尾檢定
H0: 2021/06/07的疾病死亡比例比2021/05/07的高
Ha: 2021/06/07的疾病死亡比例比2021/05/07的低

In [97]:

```
import pandas as pd
import numpy as np

# load data file(原始資料)
month_ahead_df = pd.read_csv(r"D:\統計學\Final_Project\05-06-2021.csv")
month_later_df = pd.read_csv(r"D:\統計學\Final_Project\06-06-2021.csv")

# df 為操作之資料
df_head = month_ahead_df.copy()
df_later = month_later_df.copy()
df_head
```

Out[97]:

	FIPS	Admin2	Province_State	Country_Region	Last_Update	Lat	Long_	Confirn
0	NaN	NaN	NaN	Afghanistan	2021-05-07 04:20:40	33.939110	67.709953	61
1	NaN	NaN	NaN	Albania	2021-05-07 04:20:40	41.153300	20.168300	131
2	NaN	NaN	NaN	Algeria	2021-05-07 04:20:40	28.033900	1.659600	123
3	NaN	NaN	NaN	Andorra	2021-05-07 04:20:40	42.506300	1.521800	13
4	NaN	NaN	NaN	Angola	2021-05-07 04:20:40	-11.202700	17.873900	27
...	...	...	...	...	...	...	...	
3979	NaN	NaN	NaN	West Bank and Gaza	2021-05-07 04:20:40	31.952200	35.233200	300
3980	NaN	NaN	NaN	Yemen	2021-05-07 04:20:40	15.552727	48.516388	6

	FIPS	Admin2	Province_State	Country_Region	Last_Update	Lat	Long_	Confirn
3981	NaN	NaN	NaN	Zambia	2021-05-07 04:20:40	-13.133897	27.849332	91
3982	NaN	NaN	NaN	Zimbabwe	2021-05-07 04:20:40	-19.015438	29.154857	38
3983	NaN	NaN	NaN	Kiribati	2021-05-07 04:20:40	-3.370400	-168.734000	

3984 rows × 14 columns

In [90]:
```
# 這邊是利用正則表達式，將pandas中的空白，替換成NaN
df_head = df_head.replace(r'\s+', np.nan, regex = True)
df_later = df_later.replace(r'\s+', np.nan, regex = True)
df_head
```

Out[90]:

	FIPS	Admin2	Province_State	Country_Region	Last_Update	Lat	Long_	Confirn
0	NaN	NaN	NaN	Afghanistan	NaN	33.939110	67.709953	61
1	NaN	NaN	NaN	Albania	NaN	41.153300	20.168300	131
2	NaN	NaN	NaN	Algeria	NaN	28.033900	1.659600	123
3	NaN	NaN	NaN	Andorra	NaN	42.506300	1.521800	13
4	NaN	NaN	NaN	Angola	NaN	-11.202700	17.873900	27
...	...	...	...	...	...	...	...	
3979	NaN	NaN	NaN	NaN	NaN	31.952200	35.233200	300
3980	NaN	NaN	NaN	Yemen	NaN	15.552727	48.516388	6
3981	NaN	NaN	NaN	Zambia	NaN	-13.133897	27.849332	91
3982	NaN	NaN	NaN	Zimbabwe	NaN	-19.015438	29.154857	38
3983	NaN	NaN	NaN	Kiribati	NaN	-3.370400	-168.734000	

3984 rows × 14 columns

In [91]:
```
# 檢查資料/目標columns是否有空值，確保資料的完整性
# 並發現，資料內部columns大部分都有空值
print(df_head.isnull().any(), end = '\n\n')
print(df_later.isnull().any())
```

```
FIPS                 True
Admin2               True
Province_State       True
Country_Region       True
Last_Update          True
Lat                  True
Long_                True
Confirmed           False
Deaths              False
Recovered            True
Active               True
Combined_Key         True
```

```
Incident_Rate           True
Case_Fatality_Ratio     True
dtype: bool

FIPS                    True
Admin2                  True
Province_State          True
Country_Region          True
Last_Update             True
Lat                     True
Long_                   True
Confirmed              False
Deaths                 False
Recovered               True
Active                  True
Combined_Key            True
Incident_Rate           True
Case_Fatality_Ratio     True
dtype: bool
```

In [118…

```
# 將目標 age 欄位的空值處理，並全部利用 age mean值取代空值
without_NaN_in_age = df_head.dropna(axis = 0, subset = ['Case_Fatality_Ratio'])

# 將 age 列，除了空值以外的其他值平均
mean_CFR = without_NaN_in_age['Case_Fatality_Ratio'].astype(float).mean()

# 並將age的所有數值平均值取代 age內部的空值
df_head['Case_Fatality_Ratio'] = df_head['Case_Fatality_Ratio'].fillna(value = mean_
df_head['Case_Fatality_Ratio']
```

Out[118…

```
0        4.355646
1        1.829519
2        2.678318
3        0.950385
4        2.227714
          ...
3979     1.107238
3980    19.685652
3981     1.366019
3982     4.101776
3983     0.000000
Name: Case_Fatality_Ratio, Length: 3984, dtype: float64
```

In [119…

```
# 將目標 Case_Fatality_Ratio 欄位的空值處理，並全部利用 Case_Fatality_Ratio mean值取代空
without_NaN_in_age = df_later.dropna(axis = 0, subset = ['Case_Fatality_Ratio'])

# 將 age 列，除了空值以外的其他值平均
mean_CFR = without_NaN_in_age['Case_Fatality_Ratio'].astype(float).mean()
# print(mean_CFR)
# 並將age的所有數值平均值取代 age內部的空值
df_later['Case_Fatality_Ratio'] = df_later['Case_Fatality_Ratio'].fillna(value = mea
df_later['Case_Fatality_Ratio']
```

```
2.0840817987791365
```

Out[119…

```
0        3.969757
1        1.851502
2        2.686357
3        0.923099
4        2.228000
          ...
3979     0.602889
3980     1.134421
3981    19.581553
3982     1.304374
3983     4.098089
Name: Case_Fatality_Ratio, Length: 3984, dtype: float64
```

In [84]:
```
df_later['Case_Fatality_Ratio'].isnull().any()
```

Out[84]:
```
False
```

In [103…
```
df_head_Ratio = pd.DataFrame([])
df_later_Ratio = pd.DataFrame([])
df_head_Ratio
```

Out[103…

In [104…
```
## 將資料加上年限
## 創建兩個全新的DataFrame用來包裝年限和CFR
df_head_Ratio['Case_Fatality_Ratio'] = df_head['Case_Fatality_Ratio']
df_head_Ratio['year'] = [20200506]*len(df_head)
df_later_Ratio['Case_Fatality_Ratio'] = df_later['Case_Fatality_Ratio']
df_later_Ratio['year'] = [20200606]*len(df_head)

df_head_Ratio
# print(df_later_Ratio)
```

Out[104…

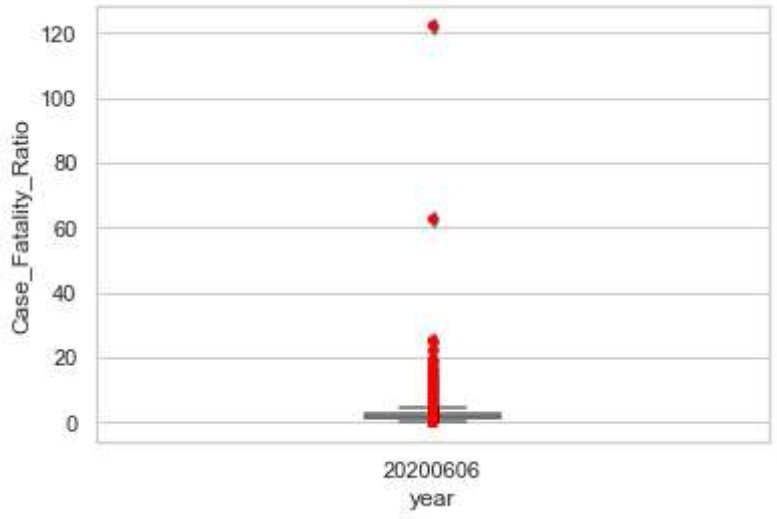
	Case_Fatality_Ratio	year
0	4.355646	20200506
1	1.829519	20200506
2	2.678318	20200506
3	0.950385	20200506
4	2.227714	20200506
…	…	…
3979	1.107238	20200506
3980	19.685652	20200506
3981	1.366019	20200506
3982	4.101776	20200506
3983	0.000000	20200506

3984 rows × 2 columns

In [105…
```
## 畫盒狀圖看不同位置的州數據的分布
import seaborn as sns
sns.set(style="whitegrid")
ax = sns.boxplot(x = 'year', y = "Case_Fatality_Ratio", data = df_later_Ratio, width
ax = sns.swarmplot(x = 'year', y = "Case_Fatality_Ratio", data = df_later_Ratio, col
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-105-06f415a7d9b8> in <module>
      3 sns.set(style="whitegrid")
      4 ax = sns.boxplot(x = 'year', y = "Case_Fatality_Ratio", data = df_later_Rati
o, width=0.2, palette="Set3")
----> 5 ax = sns.swarmplot(x = 'year', y = "Case_Fatality_Ratio", data = df_later_Ra
tio, color = "red")
```

```
c:\users\user1\appdata\local\programs\python\python38\lib\site-packages\seaborn\_dec
orators.py in inner_f(*args, **kwargs)
     44             )
     45         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
---> 46         return f(**kwargs)
     47     return inner_f
     48

c:\users\user1\appdata\local\programs\python\python38\lib\site-packages\seaborn\cate
gorical.py in swarmplot(x, y, hue, data, order, hue_order, dodge, orient, color, pal
ette, size, edgecolor, linewidth, ax, **kwargs)
   3014                         linewidth=linewidth))
   3015
-> 3016     plotter.plot(ax, kwargs)
   3017     return ax
   3018

c:\users\user1\appdata\local\programs\python\python38\lib\site-packages\seaborn\cate
gorical.py in plot(self, ax, kws)
   1418     def plot(self, ax, kws):
   1419         """Make the full plot."""
-> 1420         self.draw_swarmplot(ax, kws)
   1421         self.add_legend_data(ax)
   1422         self.annotate_axes(ax)

c:\users\user1\appdata\local\programs\python\python38\lib\site-packages\seaborn\cate
gorical.py in draw_swarmplot(self, ax, kws)
   1414         for center, swarm in zip(centers, swarms):
   1415             if swarm.get_offsets().size:
-> 1416                 self.swarm_points(ax, swarm, center, width, s, **kws)
   1417
   1418     def plot(self, ax, kws):

c:\users\user1\appdata\local\programs\python\python38\lib\site-packages\seaborn\cate
gorical.py in swarm_points(self, ax, points, center, width, s, **kws)
   1316
   1317         # Do the beeswarm in point coordinates
-> 1318         new_xy = self.beeswarm(orig_xy, d)
   1319
   1320         # Transform the point coordinates back to data coordinates

c:\users\user1\appdata\local\programs\python\python38\lib\site-packages\seaborn\cate
gorical.py in beeswarm(self, orig_xy, d)
   1268
   1269             # Find the first candidate that does not overlap any neighbours
-> 1270             new_xy_i = self.first_non_overlapping_candidate(candidates,
   1271                                                             neighbors, d)
   1272

c:\users\user1\appdata\local\programs\python\python38\lib\site-packages\seaborn\cate
gorical.py in first_non_overlapping_candidate(self, candidates, neighbors, d)
   1227             dy = neighbors_y - y_i
   1228
-> 1229             sq_distances = np.power(dx, 2.0) + np.power(dy, 2.0)
   1230
   1231             # good candidate does not overlap any of neighbors

KeyboardInterrupt:
```

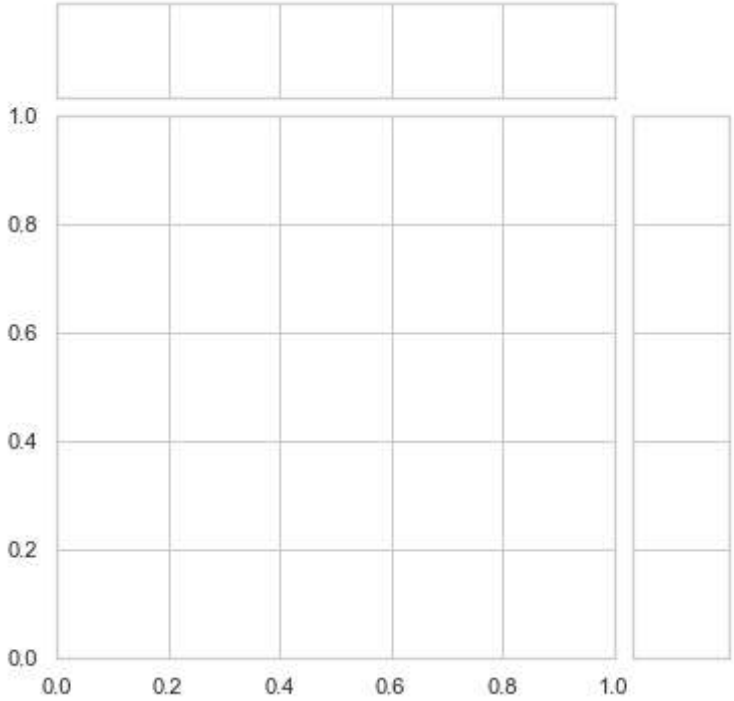


In [124…

```
import seaborn as sns
sns.scatterplot(data=df_head, x="total_bill", y="tip")
```

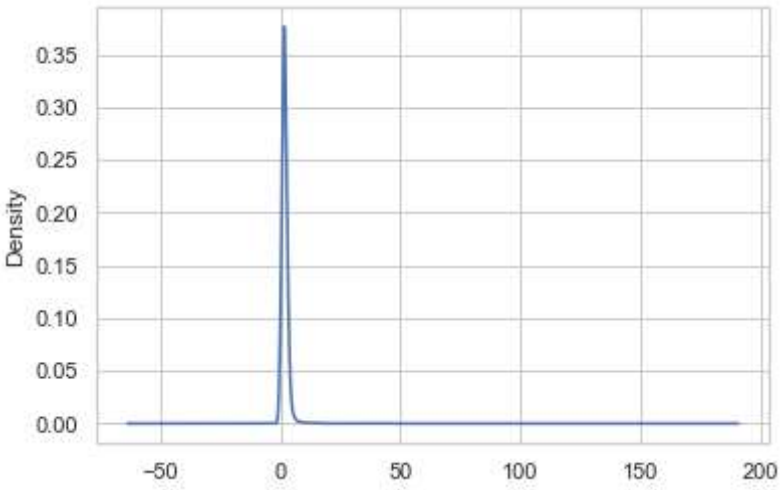
```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-124-61d4f7183874> in <module>
      3 # Case_Fatality_Ratio = sns.load_dataset("Case_Fatality_Ratio")
      4
----> 5 sns.jointplot(x= '2020', y='Case_Fatality_Ratio',data=df_head['Case_Fatality
_Ratio'])

c:\users\user1\appdata\local\programs\python\python38\lib\site-packages\seaborn\_dec
orators.py in inner_f(*args, **kwargs)
     44             )
     45         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
---> 46         return f(**kwargs)
     47     return inner_f
     48

c:\users\user1\appdata\local\programs\python\python38\lib\site-packages\seaborn\axis
grid.py in jointplot(x, y, data, kind, color, height, ratio, space, dropna, xlim, yl
im, marginal_ticks, joint_kws, marginal_kws, hue, palette, hue_order, hue_norm, **kw
args)
   2119
   2120     # Initialize the JointGrid object
-> 2121     grid = JointGrid(
   2122         data=data, x=x, y=y, hue=hue,
   2123         palette=palette, hue_order=hue_order, hue_norm=hue_norm,

c:\users\user1\appdata\local\programs\python\python38\lib\site-packages\seaborn\_dec
orators.py in inner_f(*args, **kwargs)
     44             )
     45         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
---> 46         return f(**kwargs)
     47     return inner_f
     48

c:\users\user1\appdata\local\programs\python\python38\lib\site-packages\seaborn\axis
grid.py in __init__(self, x, y, data, height, ratio, space, dropna, xlim, ylim, siz
e, marginal_ticks, hue, palette, hue_order, hue_norm)
   1628
   1629         # Process the input variables
-> 1630         p = VectorPlotter(data=data, variables=dict(x=x, y=y, hue=hue))
   1631         plot_data = p.plot_data.loc[:, p.plot_data.notna().any()]
   1632

c:\users\user1\appdata\local\programs\python\python38\lib\site-packages\seaborn\_cor
e.py in __init__(self, data, variables)
    602     def __init__(self, data=None, variables={}):
```

```
    603
--> 604         self.assign_variables(data, variables)
    605
    606         for var, cls in self._semantic_mappings.items():

c:\users\user1\appdata\local\programs\python\python38\lib\site-packages\seaborn\_cor
e.py in assign_variables(self, data, variables)
    665         else:
    666             self.input_format = "long"
--> 667             plot_data, variables = self._assign_variables_longform(
    668                 data, **variables,
    669             )

c:\users\user1\appdata\local\programs\python\python38\lib\site-packages\seaborn\_cor
e.py in _assign_variables_longform(self, data, **kwargs)
    900
    901                 err = f"Could not interpret value `{val}` for parameter `{ke
y}`"
--> 902                 raise ValueError(err)
    903
    904             else:

ValueError: Could not interpret value `2020` for parameter `x`
```



In [122…

```
df_head_plot = df_head['Case_Fatality_Ratio'].plot.kde()
```



In [127…
```
##  檢查資料是否為常態分布
## 這裡的 p-value < 常設的標準 0.05，其實p-value並不是0，而是非常小的數字
## 代表2021/06/07的疾病死亡比例比2021/05/07死亡比例的分布皆為常態分布
import scipy.stats
print(scipy.stats.shapiro(df_head['Case_Fatality_Ratio']))
print(scipy.stats.shapiro(df_later['Case_Fatality_Ratio']))
```

```
ShapiroResult(statistic=0.27582335472106934, pvalue=0.0)
ShapiroResult(statistic=0.3070448637008667, pvalue=0.0)
```

In [126…
```
## 變異數檢測
## 這裡的 p-value > 常設的顯著標準 0.05，代表兩者的標準差相等
## 代表2021/06/07的疾病死亡比例比2021/05/07死亡比例的標準差相等
import scipy.stats
scipy.stats.levene(df_head['Case_Fatality_Ratio'], df_later['Case_Fatality_Ratio'],
```

Out[126… `LeveneResult(statistic=0.38280072626246275, pvalue=0.5361258243969971)`

In [128…
```
## 現在做 z-Test 檢測2021/06/07的疾病死亡比例比2021/05/07死亡比例(Ha: 2021/06/07的疾病死
import statsmodels.stats.weightstats
statsmodels.stats.weightstats.ztest(df_head['Case_Fatality_Ratio'], df_later['Case_F

## p_value > 顯著標準 0.05，代表沒有足夠多的證據表示「2021/06/07的疾病死亡比例比2021/05/0
## H0沒有被拒絕，2021/06/07的疾病死亡比例比2021/05/07死亡比例高
p_value = statsmodels.stats.weightstats.ztest(df_head['Case_Fatality_Ratio'], df_lat
p_value
```

Out[128… `0.7603329324301022`

5. An international e-commerce company based wants to discover key insights from its customer database. The company sells electronic products. (dataset from Kaggle)

1. Is the customer rating (Customer_rating) associated with if the products reached on time (Reached.on.Time_Y.N) and customer gender (Gender)? That is, please test if the customer rating affected by reached on-time status and customer gender. 顧客評分跟貨品有沒有準時到還有顧客性別是否有關？

使用 Chi-Square Test: 因為探討的因素皆為 Qualitative Data, 比較之間的關係。而所有的卡方檢定均為單尾檢定

H0: 乘坐的艙位等級和平均存活的機率相等

Ha: 至少有一艙位的平均生存機率和其他不同

In [164…
```
import pandas as pd
import numpy as np

# Load data file(原始資料)
df = pd.read_csv(r"D:\統計學\Final_Project\Train.csv")
```

```
# df 為操作之資料
df_1 = df.copy()
df_2 = df.copy()
df_1
```

Out[164...

	ID	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cost_of_t
0	1	D	Flight	4	2	
1	2	F	Flight	4	5	
2	3	A	Flight	2	2	
3	4	B	Flight	3	3	
4	5	C	Flight	2	2	
...	...	...	...	...	...	
10994	10995	A	Ship	4	1	
10995	10996	B	Ship	4	1	
10996	10997	C	Ship	5	4	
10997	10998	F	Ship	5	2	
10998	10999	D	Ship	2	5	

10999 rows × 12 columns

In [165...

```
# 這邊是利用正則表達式，將pandas中的空白，替換成NaN
df_1 = df_1.replace(r'\s+', np.nan, regex = True)
df_2 = df_2.replace(r'\s+', np.nan, regex = True)
```

In [133...

```
# 檢查資料/目標columns是否有空值，確保資料的完整性
# 並發現，資料內部columns全部都沒有空值
print(df_1.isnull().any(), end = '\n\n')
```

```
ID                   False
Warehouse_block      False
Mode_of_Shipment     False
Customer_care_calls  False
Customer_rating      False
Cost_of_the_Product  False
Prior_purchases      False
Product_importance   False
Gender               False
Discount_offered     False
Weight_in_gms        False
Reached.on.Time_Y.N  False
dtype: bool
```

In [140...

```
## 先對 Customer_rating & Reached.on.Time_Y.N 做卡方檢定
df_1 = df_1[['Customer_rating', 'Reached.on.Time_Y.N']]
df_1
```

Out[140...

	Customer_rating	Reached.on.Time_Y.N
0	2	1

	Customer_rating	Reached.on.Time_Y.N
1	5	1
2	2	1
3	3	1
4	2	1
...	...	...
10994	1	1
10995	1	0
10996	4	0
10997	2	0
10998	5	0

10999 rows × 2 columns

In [141...
```
## 因為資料2者皆為 Qulitative Data，故使用卡方檢定來判斷
## 而且資料數量皆有大於5
## 將資料轉換成卡方檢定可以分析的「頻率」
from collections import Counter
frequency_count = Counter(df_1['Customer_rating'])
frequency_count
```

Out[141... `Counter({2: 2165, 5: 2171, 3: 2239, 1: 2235, 4: 2189})`

In [146...
```
# 將清單中的Customer_rating取出，稱為清單 f1, 將頻率取出，稱為清單 f2
## 並將這些製作成卡方檢定需要的頻率清單
f1 = list(frequency_count.keys())
f2 = list(frequency_count.values())
frequency_table = pd.DataFrame(zip(f1,f2),
                               columns=['Customer_rating','freq'])

frequency_table = frequency_table.sort_values(by = 'Customer_rating', ascending = Tr
frequency_table
```

Out[146...

	Customer_rating	freq
3	1	2235
0	2	2165
2	3	2239
4	4	2189
1	5	2171

In [158...
```
## 再將即時到貨放入頻率測試
CR_Index = frequency_table['Customer_rating'].tolist()

## 將有沒有及時到貨分別取出，並組合成 List
## CR_1: 沒有及時到貨 & CR = 1
CR_1 = df_1[df_1['Reached.on.Time_Y.N'] == 0][df_1['Customer_rating'] == 1].shape[0]
## CR_2: 沒有及時到貨 & CR = 2，後面依樣畫葫蘆
CR_2 = df_1[df_1['Reached.on.Time_Y.N'] == 0][df_1['Customer_rating'] == 2].shape[0]
```

```
CR_3 = df_1[df_1['Reached.on.Time_Y.N'] == 0][df_1['Customer_rating'] == 3].shape[0]
CR_4 = df_1[df_1['Reached.on.Time_Y.N'] == 0][df_1['Customer_rating'] == 4].shape[0]
CR_5 = df_1[df_1['Reached.on.Time_Y.N'] == 0][df_1['Customer_rating'] == 5].shape[0]

RT_zero_list = [CR_1, CR_2, CR_3, CR_4, CR_5]

CR_6 = df_1[df_1['Reached.on.Time_Y.N'] == 1][df_1['Customer_rating'] == 1].shape[0]
## CR_2: 沒有及時到貨 & CR = 2，後面依樣畫葫蘆
CR_7 = df_1[df_1['Reached.on.Time_Y.N'] == 1][df_1['Customer_rating'] == 2].shape[0]
CR_8 = df_1[df_1['Reached.on.Time_Y.N'] == 1][df_1['Customer_rating'] == 3].shape[0]
CR_9 = df_1[df_1['Reached.on.Time_Y.N'] == 1][df_1['Customer_rating'] == 4].shape[0]
CR_10 = df_1[df_1['Reached.on.Time_Y.N'] == 1][df_1['Customer_rating'] == 5].shape[0

RT_one_list = [CR_6, CR_7, CR_8, CR_9, CR_10]
```

```
<ipython-input-158-abbded5cc757>:6: UserWarning: Boolean Series key will be reindexe
d to match DataFrame index.
  CR_1 = df_1[df_1['Reached.on.Time_Y.N'] == 0][df_1['Customer_rating'] == 1].shape
[0]
<ipython-input-158-abbded5cc757>:8: UserWarning: Boolean Series key will be reindexe
d to match DataFrame index.
  CR_2 = df_1[df_1['Reached.on.Time_Y.N'] == 0][df_1['Customer_rating'] == 2].shape
[0]
<ipython-input-158-abbded5cc757>:9: UserWarning: Boolean Series key will be reindexe
d to match DataFrame index.
  CR_3 = df_1[df_1['Reached.on.Time_Y.N'] == 0][df_1['Customer_rating'] == 3].shape
[0]
<ipython-input-158-abbded5cc757>:10: UserWarning: Boolean Series key will be reindex
ed to match DataFrame index.
  CR_4 = df_1[df_1['Reached.on.Time_Y.N'] == 0][df_1['Customer_rating'] == 4].shape
[0]
<ipython-input-158-abbded5cc757>:11: UserWarning: Boolean Series key will be reindex
ed to match DataFrame index.
  CR_5 = df_1[df_1['Reached.on.Time_Y.N'] == 0][df_1['Customer_rating'] == 5].shape
[0]
<ipython-input-158-abbded5cc757>:15: UserWarning: Boolean Series key will be reindex
ed to match DataFrame index.
  CR_6 = df_1[df_1['Reached.on.Time_Y.N'] == 1][df_1['Customer_rating'] == 1].shape
[0]
<ipython-input-158-abbded5cc757>:17: UserWarning: Boolean Series key will be reindex
ed to match DataFrame index.
  CR_7 = df_1[df_1['Reached.on.Time_Y.N'] == 1][df_1['Customer_rating'] == 2].shape
[0]
<ipython-input-158-abbded5cc757>:18: UserWarning: Boolean Series key will be reindex
ed to match DataFrame index.
  CR_8 = df_1[df_1['Reached.on.Time_Y.N'] == 1][df_1['Customer_rating'] == 3].shape
[0]
<ipython-input-158-abbded5cc757>:19: UserWarning: Boolean Series key will be reindex
ed to match DataFrame index.
  CR_9 = df_1[df_1['Reached.on.Time_Y.N'] == 1][df_1['Customer_rating'] == 4].shape
[0]
<ipython-input-158-abbded5cc757>:20: UserWarning: Boolean Series key will be reindex
ed to match DataFrame index.
  CR_10 = df_1[df_1['Reached.on.Time_Y.N'] == 1][df_1['Customer_rating'] == 5].shape
[0]
```

In [157…

```
chi_table = pd.DataFrame(zip(RT_zero_list, RT_one_list),
                         columns=["RT = 0", "RT = 1"],
                         index=['CR = 1', 'CR = 2',  'CR = 3', 'CR = 4', 'CR = 5'])
chi_table
```

Out[157…

	RT = 0	RT = 1
CR = 1	922	1313
CR = 2	892	1273

	RT = 0	RT = 1
CR = 3	882	1357
CR = 4	886	1303
CR = 5	854	1317

In [159…
```
## 將城市轉換成卡方檢定的資料刑事
import numpy as np
obs = np.array([chi_table.iloc[0,:].tolist(),
                chi_table.iloc[1,:].tolist(),
                chi_table.iloc[2,:].tolist(),
                chi_table.iloc[3,:].tolist(),
                chi_table.iloc[4,:].tolist()])
obs
```

Out[159…
```
array([[ 922, 1313],
       [ 892, 1273],
       [ 882, 1357],
       [ 886, 1303],
       [ 854, 1317]])
```

In [160…
```
## 將剛剛處理過的資料帶入卡方檢定內
## 得到的資料依序為: 檢定值，p-value, 自由度, 檢測內容
import scipy.stats
scipy.stats.chi2_contingency(obs, correction = False)
```

Out[160…
```
(3.200045474831146,
 0.5249236018493662,
 4,
 array([[ 901.39649059, 1333.60350941],
        [ 873.16483317, 1291.83516683],
        [ 903.00972816, 1335.99027184],
        [ 882.84425857, 1306.15574143],
        [ 875.58468952, 1295.41531048]]))
```

In [161…
```
## 重點提出 p-value來檢驗
## 這裡可以看出 p-value > 常規標準的 0.05
## 沒有足夠多的證據證明 "H0會被拒絕"
## 所以「貨物有沒有及時到達」和「顧客評分」沒有太大的關係
p_value = scipy.stats.chi2_contingency(obs, correction = False)[1]
p_value
```

Out[161…
```
0.5249236018493662
```

在對 Customer_rating & Gender做卡方分配

In [166…
```
## 先對 Customer_rating & Reached.on.Time_Y.N 做卡方檢定
df_2 = df_2[['Customer_rating', 'Gender']]
df_2
```

Out[166…

	Customer_rating	Gender
0	2	F
1	5	M
2	2	M

	Customer_rating	Gender
3	3	M
4	2	F
...	...	...
10994	1	F
10995	1	F
10996	4	F
10997	2	M
10998	5	F

10999 rows × 2 columns

In [167…

```
## 將有沒有及時到貨分別取出，並組合成 List
## CR_1: 沒有及時到貨 & CR = 1
CR_1 = df_2[df_2['Gender'] == 'F'][df_2['Customer_rating'] == 1].shape[0]
## CR_2: 沒有及時到貨 & CR = 2，後面依樣畫葫蘆
CR_2 = df_2[df_2['Gender'] == 'F'][df_2['Customer_rating'] == 2].shape[0]
CR_3 = df_2[df_2['Gender'] == 'F'][df_2['Customer_rating'] == 3].shape[0]
CR_4 = df_2[df_2['Gender'] == 'F'][df_2['Customer_rating'] == 4].shape[0]
CR_5 = df_2[df_2['Gender'] == 'F'][df_2['Customer_rating'] == 5].shape[0]

GenF_list = [CR_1, CR_2, CR_3, CR_4, CR_5]

CR_6 = df_2[df_2['Gender'] == 'M'][df_2['Customer_rating'] == 1].shape[0]
## CR_2: 沒有及時到貨 & CR = 2，後面依樣畫葫蘆
CR_7 = df_2[df_2['Gender'] == 'M'][df_2['Customer_rating'] == 2].shape[0]
CR_8 = df_2[df_2['Gender'] == 'M'][df_2['Customer_rating'] == 3].shape[0]
CR_9 = df_2[df_2['Gender'] == 'M'][df_2['Customer_rating'] == 4].shape[0]
CR_10 = df_2[df_2['Gender'] == 'M'][df_2['Customer_rating'] == 5].shape[0]

GenM_list = [CR_6, CR_7, CR_8, CR_9, CR_10]
```

```
<ipython-input-167-8be6d53dccbc>:3: UserWarning: Boolean Series key will be reindexe
d to match DataFrame index.
  CR_1 = df_2[df_2['Gender'] == 'F'][df_2['Customer_rating'] == 1].shape[0]
<ipython-input-167-8be6d53dccbc>:5: UserWarning: Boolean Series key will be reindexe
d to match DataFrame index.
  CR_2 = df_2[df_2['Gender'] == 'F'][df_2['Customer_rating'] == 2].shape[0]
<ipython-input-167-8be6d53dccbc>:6: UserWarning: Boolean Series key will be reindexe
d to match DataFrame index.
  CR_3 = df_2[df_2['Gender'] == 'F'][df_2['Customer_rating'] == 3].shape[0]
<ipython-input-167-8be6d53dccbc>:7: UserWarning: Boolean Series key will be reindexe
d to match DataFrame index.
  CR_4 = df_2[df_2['Gender'] == 'F'][df_2['Customer_rating'] == 4].shape[0]
<ipython-input-167-8be6d53dccbc>:8: UserWarning: Boolean Series key will be reindexe
d to match DataFrame index.
  CR_5 = df_2[df_2['Gender'] == 'F'][df_2['Customer_rating'] == 5].shape[0]
<ipython-input-167-8be6d53dccbc>:12: UserWarning: Boolean Series key will be reindex
ed to match DataFrame index.
  CR_6 = df_2[df_2['Gender'] == 'M'][df_2['Customer_rating'] == 1].shape[0]
<ipython-input-167-8be6d53dccbc>:14: UserWarning: Boolean Series key will be reindex
ed to match DataFrame index.
  CR_7 = df_2[df_2['Gender'] == 'M'][df_2['Customer_rating'] == 2].shape[0]
<ipython-input-167-8be6d53dccbc>:15: UserWarning: Boolean Series key will be reindex
ed to match DataFrame index.
  CR_8 = df_2[df_2['Gender'] == 'M'][df_2['Customer_rating'] == 3].shape[0]
<ipython-input-167-8be6d53dccbc>:16: UserWarning: Boolean Series key will be reindex
ed to match DataFrame index.
```

```
  CR_9 = df_2[df_2['Gender'] == 'M'][df_2['Customer_rating'] == 4].shape[0]
<ipython-input-167-8be6d53dccbc>:17: UserWarning: Boolean Series key will be reindex
ed to match DataFrame index.
  CR_10 = df_2[df_2['Gender'] == 'M'][df_2['Customer_rating'] == 5].shape[0]
```

In [169…

```
chi_table = pd.DataFrame(zip(GenF_list, GenM_list),
                         columns=["F", "M"],
                         index=['CR = 1', 'CR = 2',  'CR = 3', 'CR = 4', 'CR = 5'])
chi_table
```

Out[169…

	F	M
CR = 1	1149	1086
CR = 2	1062	1103
CR = 3	1143	1096
CR = 4	1096	1093
CR = 5	1095	1076

In [170…

```
## 將城市轉換成卡方檢定的資料刑事
import numpy as np
obs = np.array([chi_table.iloc[0,:].tolist(),
                chi_table.iloc[1,:].tolist(),
                chi_table.iloc[2,:].tolist(),
                chi_table.iloc[3,:].tolist(),
                chi_table.iloc[4,:].tolist()])
obs
```

Out[170…

```
array([[1149, 1086],
       [1062, 1103],
       [1143, 1096],
       [1096, 1093],
       [1095, 1076]])
```

In [171…

```
## 重點提出 p-value來檢驗
## 這裡可以看出 p-value > 常規標準的 0.05
## 沒有足夠多的證據證明 "H0會被拒絕"
## 所以「性別」和「顧客評分」沒有太大的關係
p_value = scipy.stats.chi2_contingency(obs, correction = False)[1]
p_value
```

Out[171…

```
0.565115449371974
```

2. Is the cost of the product (Cost_of_the_Product) different among the mode of shipment (Mode_of_Shipment)? 使用不同運送方式的貨品商品價格是否不同？

使用 one_way ANOVA Test: 因為探討的因素為比較不同運送方式的貨品商品價格關係。one-way ANOVA為單尾檢定

H0: 不同運送方式的平均貨品商品價格相等

Ha: 不同運送方式的平均貨品商品價格不相等

In [172…

```
import pandas as pd
import numpy as np
```

```
# load data file(原始資料)
df = pd.read_csv(r"D:\統計學\Final_Project\Train.csv")
# df 為操作之資料
df_3 = df.copy()
df_3
```

Out[172...

	ID	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cost_of_t
0	1	D	Flight	4	2	
1	2	F	Flight	4	5	
2	3	A	Flight	2	2	
3	4	B	Flight	3	3	
4	5	C	Flight	2	2	
...	...	...	...	...	...	
10994	10995	A	Ship	4	1	
10995	10996	B	Ship	4	1	
10996	10997	C	Ship	5	4	
10997	10998	F	Ship	5	2	
10998	10999	D	Ship	2	5	

10999 rows × 12 columns

In [176...

```
df_3 = df_3[['Cost_of_the_Product', 'Mode_of_Shipment']]
df_3
```

Out[176...

	Cost_of_the_Product	Mode_of_Shipment
0	177	Flight
1	216	Flight
2	183	Flight
3	176	Flight
4	184	Flight
...	...	...
10994	252	Ship
10995	232	Ship
10996	242	Ship
10997	223	Ship
10998	155	Ship

10999 rows × 2 columns

In [190...

```
## 將不同的貨運方式分組，分成2種類型 Flight & Ship
ship_way = df_3[df_3['Mode_of_Shipment'] == 'Ship']['Cost_of_the_Product']
```

```
flight_way = df_3[df_3['Mode_of_Shipment'] == 'Flight']['Cost_of_the_Product']

print(ship_way, end = '\n\n')
print(flight_way,  end = '\n\n')
```

```
17       227
18       239
19       145
20       161
21       232
        ...
10994    252
10995    232
10996    242
10997    223
10998    155
Name: Cost_of_the_Product, Length: 7462, dtype: int64

0        177
1        216
2        183
3        176
4        184
        ...
10972    249
10973    240
10974    219
10975    219
10976    278
Name: Cost_of_the_Product, Length: 1777, dtype: int64
```

In [191…

```
## 常態檢定: 確認母體為常態分配，才可以代表母體作分析
## 利用shapiro模型中進行計算，以利確認資料是否呈現常態分佈
## 計算出來的 p-value < 常設標準的 0.05，因此有顯著的證據可以證明此兩筆資料不符合常態分配
## 不能當作常態分布來看
import scipy.stats as st
print(st.shapiro(ship_way))
print(st.shapiro(flight_way))
```

```
ShapiroResult(statistic=0.9726488590240479, pvalue=1.8421398403631282e-35)
ShapiroResult(statistic=0.9727673530578613, pvalue=7.40923541764978e-18)
c:\users\user1\appdata\local\programs\python\python38\lib\site-packages\scipy\stats
\morestats.py:1760: UserWarning: p-value may not be accurate for N > 5000.
  warnings.warn("p-value may not be accurate for N > 5000.")
```

In [192…

```
## 同質性檢定: 用來確保資料取自變異數相等的母體
## p-value > 0.05，因此有顯著的證據證明在兩者數據之間的變異數是相等的
st.levene(ship_way, flight_way, center='mean')
```

Out[192… `LeveneResult(statistic=0.005246240538302699, pvalue=0.942260534007179)`

In [207…

```
## 變異數分析
## p-value > 0.05，因此有顯著的證據證明在不同運送方式的平均貨品商品價格相等
## 結論: 不同運送方式的平均貨品商品價格相等
f_value, p_value = st.f_oneway(ship_way, flight_way)
p_value
```

Out[207… `0.41489518552508187`

參考資料

期末專題Documentation

第一題:

1. python - 用pandas中的NaN替換空白值（空格）
2. pandas dataframe將任一資料型別轉換資料型別
3. 計算Pandas Datframe行列資料平均
4. Pandas.DataFrame.plot.density 可以用來確認圖形是否常態分布
5. Sharpiro-Wilk Test(確認隨機變數是否為常態分布)
6. 單因子變異數分析 — Python實戰：商務資料結構整理(附Python程式碼)
7. 卡方檢定 — Python實戰：商務資料結構整理(附Python程式碼)
8. pingouin.pairwise_tukey 製作one-way ANOVA table

第二題:

1. 兩組獨立樣本Z檢定 (Two-Sample Z test，parametric)
2. 【python】statsmodels.stats.weightstats.ztest
3. How to select rows in a DataFrame between two values, in Python Pandas?
4. T 檢定 with Python

Jupyter Notebook轉pdf問題:

1. 一分钟搞定jupyter notebook文件转换成pdf（史上最简单的方法）