Homework 1-2 附錄

November 11, 2021

[1]:
```python
import numpy as np
import pandas as pd
import scipy.stats as ss
from scipy.spatial import distance_matrix
```

1 第三題

[2]:
```python
def Wilcoxon_RankSum(x1, x2, correction=True):
    n1, n2 = len(x1), len(x2)
    N = n1 + n2
    Rank = pd.concat([x1, x2], ignore_index=False).rank()
    if n1 < n2:
        W = Rank.loc[x1.index[0]].sum()
        mu = n1 * (N+1) / 2
    else:
        W = Rank.loc[x2.index[0]].sum()
        mu = n2 * (N+1) / 2

    if correction:
        sigma = np.sqrt(n1*n2 * np.sum(Rank**2) / (N*(N-1)) -
                        n1*n2 * (N+1)**2 / (4*(N-1)))
    else:
        sigma = np.sqrt(n1 * n2 * (N+1) / 12)

    if   W > mu: T = (W - 0.5 - mu) / sigma
    elif W < mu: T = (W + 0.5 - mu) / sigma
    else:        T = 0

    print('Statistic T = {:.3f}'.format(T))
    print('Critical Values: +1.96 / -1.96')
```

1.1 Wilcoxon Rank-sum Test by self-defined function

[3]:
```python
Data = pd.read_csv('JulyDMax.csv', index_col='Date')
DMax1 = pd.Series(Data.loc[:,'1961':'1980'].values.flatten(), index=[1]*620)
DMax2 = pd.Series(Data.loc[:,'1990':'2009'].values.flatten(), index=[2]*620)
Wilcoxon_RankSum(DMax1, DMax2)
```

```
Statistic T = 3.567
Critical Values: +1.96 / -1.96
```

1.2 Wilcoxon Rank-sum Test in SciPy

[4]:
```python
res = ss.ranksums(DMax2, DMax1)
print('Statistic = {:3.3f}'.format(res.statistic))
print('  p-value = {:.5f}'.format(res.pvalue))
```

```
Statistic = 3.566
  p-value = 0.00036
```

1.3 Mann-Whitney U Test in SciPy

[5]:
```
res = ss.mannwhitneyu(DMax2, DMax1)
print('Statistic = {:8.1f}'.format(res.statistic))
print('  p-value = {:.5f}'.format(res.pvalue))
```

```
Statistic = 214683.5
  p-value = 0.00036
```

2 第四題

[6]:
```
chi2 = ss.chi2.ppf(0.95, df=8-3)            # alpha=0.05, df=k(8)-m(2)-1
beta = 1                                    # When beta < 0.05, stop.
n = 2                                       # n = 2,3,4,...
B = []
successive = 0
alpha = np.sqrt(6) / np.pi
scale = alpha * 1                 # Sigma = 1
loc   = 0 - np.euler_gamma * scale #  Mean = 0

while beta >= 0.05 or successive != 3:
    F = ss.uniform.rvs(size=(10000,8*n))
    x = ss.gumbel_r.ppf(F, loc=loc, scale=scale)
    count = 0
    for i in range(10000):
        mean = x[i].mean()
        std  = x[i].std(ddof=1)
        # Equiprobable Intervals
        equiprob = ss.norm.ppf(np.linspace(0, 1, 9), mean, std)
        hist, bin_edges = np.histogram(x[i], bins=equiprob)
        expected = np.ones(8) * n
        statistic, pvalue = ss.chisquare(hist, expected, ddof=2)
        if statistic < chi2: # Do not reject H0. (Type II error)
            count = count + 1
    beta = count / 10000
    B.append(beta)

    if beta < 0.05: successive = successive + 1

    if successive == 3: break
    else: n = n + 1
```

3 第五題

3.1 第 A 小題

[7]:
```
mu = np.ones(5) * 30
sigma = 3
# Coordinates of 5 stations (O-D-A-B-C)
coord = np.array([[0,0], [-20,-60], [20,10], [25,50], [-80,30]])
# Correlation function
corr = lambda d: np.exp(-d/30)
# Distance Matrix of 5 stations (O-D-A-B-C)
dm = distance_matrix(coord, coord)
# Covariance Matrix
cov = corr(dm) * sigma**2
```

$$X = \begin{bmatrix} O \\ D \end{bmatrix} \qquad Y = \begin{bmatrix} A \\ B \\ C \end{bmatrix} \qquad \Sigma = \begin{bmatrix} \Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_{YY} \end{bmatrix} = \begin{bmatrix} 9.0 & 1.093 & 4.271 & 1.396 & 0.522 \\ 1.093 & 9.0 & 0.613 & 0.171 & 0.245 \\ 4.271 & 0.613 & 9.0 & 2.348 & 0.301 \\ 1.396 & 0.171 & 2.348 & 9.0 & 0.255 \\ 0.522 & 0.245 & 0.301 & 0.255 & 9.0 \end{bmatrix}$$

[8]:

```
obs_OD = np.array([33.3, 29.7])
# Mean of Station A, B & C given obs at O & D
mu_ABC = mu[2:] + cov[2:,:2]@np.linalg.inv(cov[:2,:2])@(obs_OD-mu[:2])
# Covariance of Station A, B & C given obs at O & D
cov_ABC = cov[2:,2:] - cov[2:,:2]@np.linalg.inv(cov[:2,:2])@cov[:2,2:]
corr_ABC = np.eye(3)
for i in range(3):
    for j in range(3):
        if i != j:
            corr_ABC[i,j] = cov_ABC[i,j] / np.sqrt(cov_ABC[i,i]*cov_ABC[j,j])
```

$$\rho_Y = \begin{bmatrix} 1.0 & 0.261 & 0.033 \\ 0.261 & 1.0 & 0.028 \\ 0.033 & 0.028 & 1.0 \end{bmatrix}$$

$$m_{Y|X} = \begin{bmatrix} 31.56 \\ 30.51 \\ 30.18 \end{bmatrix} \qquad \Sigma_{Y|X} = \begin{bmatrix} 6.972 & 1.685 & 0.051 \\ 1.685 & 8.783 & 0.174 \\ 0.051 & 0.174 & 8.966 \end{bmatrix} \qquad \rho_{Y|X} = \begin{bmatrix} 1.0 & 0.215 & 0.006 \\ 0.215 & 1.0 & 0.02 \\ 0.006 & 0.02 & 1.0 \end{bmatrix}$$

3.2 第 B 小題

[9]:

```
# Coordinates of 5 stations (A-B-C-D-O)
coord = np.array([[20,10], [25,50], [-80,30], [-20,-60], [0,0]])
# Distance Matrix of 5 stations (A-B-C-D-O)
dm = distance_matrix(coord, coord)
# Covariance Matrix
cov = corr(dm) * sigma**2
```

$$X = \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} \qquad Y = [O] \qquad \Sigma = \begin{bmatrix} \Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_{YY} \end{bmatrix} = \begin{bmatrix} 9.0 & 2.348 & 0.301 & 0.613 & 4.271 \\ 2.348 & 9.0 & 0.255 & 0.171 & 1.396 \\ 0.301 & 0.255 & 9.0 & 0.245 & 0.522 \\ 0.613 & 0.171 & 0.245 & 9.0 & 1.093 \\ 4.271 & 1.396 & 0.522 & 1.093 & 9.0 \end{bmatrix}$$

$$\hat{T_O} - \mu = \Sigma_i w_i (T_i - \mu)$$

$$m_{Y|X} - m_Y = \Sigma_{YX} \Sigma_{XX}^{-1} (x - m_X)$$

$$\Rightarrow w_i = \Sigma_{YX} \Sigma_{XX}^{-1}$$

[10]:

```
weight = cov[4,:4]@np.linalg.inv(cov[:4,:4])
```

$$w_i = \begin{bmatrix} 0.4587 \\ 0.0327 \\ 0.0393 \\ 0.0886 \end{bmatrix}$$