# 9.1

import numpy as np

import pandas as pd

# 数据

data = pd.DataFrame({

'A': [42, 13, 7],

'B': [62, 28, 18],

'C': [184, 81, 54],

'D': [207, 113, 92]

}, index=['1', '2', '3'])

# 计算行和列的边际总和

row\_sums = data.sum(axis=1)

col\_sums = data.sum(axis=0)

total\_sum = data.values.sum()

# 计算相对频率矩阵

P = data / total\_sum

# 计算行和列的相对频率

row\_marginals = row\_sums / total\_sum

col\_marginals = col\_sums / total\_sum

# 计算期望值矩阵

expected = np.outer(row\_marginals, col\_marginals)

# 计算标准化残差（中心化后的矩阵）

standardized\_residuals = (P - expected) / np.sqrt(expected)

# 奇异值分解 (SVD)

U, singular\_values, Vt = np.linalg.svd(standardized\_residuals)

# 计算主惯量

principal\_inertia = singular\_values\*\*2

# 计算贡献率和累计贡献率

total\_inertia = principal\_inertia.sum()

contribution\_rate = principal\_inertia / total\_inertia

cumulative\_contribution\_rate = np.cumsum(contribution\_rate)

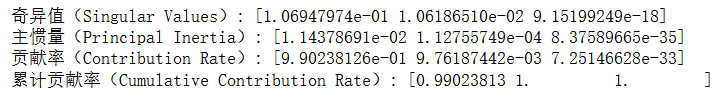
# 输出结果

print("奇异值（Singular Values）:", singular\_values)

print("主惯量（Principal Inertia）:", principal\_inertia)

print("贡献率（Contribution Rate）:", contribution\_rate)

print("累计贡献率（Cumulative Contribution Rate）:", cumulative\_contribution\_rate)



# 9.2

import numpy as np

import pandas as pd

# 考古场所数据

data = pd.DataFrame({

'A': [30, 53, 73, 20, 46, 45, 16],

'B': [10, 4, 1, 6, 36, 6, 28],

'C': [10, 16, 41, 1, 37, 59, 169],

'D': [39, 2, 1, 4, 13, 10, 5]

}, index=['P0', 'P1', 'P2', 'P3', 'P4', 'P5', 'P6'])

# 计算行和列的边际总和

row\_sums = data.sum(axis=1)

col\_sums = data.sum(axis=0)

total\_sum = data.values.sum()

# 计算相对频率矩阵

P = data / total\_sum

# 计算行和列的相对频率

row\_marginals = row\_sums / total\_sum

col\_marginals = col\_sums / total\_sum

# 计算期望值矩阵

expected = np.outer(row\_marginals, col\_marginals)

# 计算标准化残差（中心化后的矩阵）

standardized\_residuals = (P - expected) / np.sqrt(expected)

# 奇异值分解 (SVD)

U, singular\_values, Vt = np.linalg.svd(standardized\_residuals)

# 计算主惯量

principal\_inertia = singular\_values\*\*2

# 计算贡献率和累计贡献率

total\_inertia = principal\_inertia.sum()

contribution\_rate = principal\_inertia / total\_inertia

cumulative\_contribution\_rate = np.cumsum(contribution\_rate)

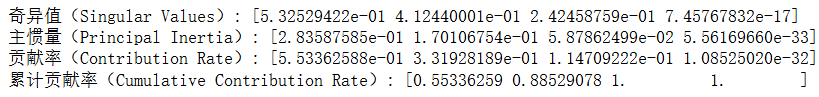
# 输出结果

print("奇异值（Singular Values）:", singular\_values)

print("主惯量（Principal Inertia）:", principal\_inertia)

print("贡献率（Contribution Rate）:", contribution\_rate)

print("累计贡献率（Cumulative Contribution Rate）:", cumulative\_contribution\_rate)



# 9.3

import numpy as np

import pandas as pd

# 学科数据

data = pd.DataFrame({

'1973': [4489, 4101, 3354, 2444, 3338, 1222],

'1974': [4303, 3800, 3286, 2587, 3144, 1196],

'1975': [4402, 3749, 3344, 2749, 2959, 1149],

'1976': [4350, 3572, 3278, 2878, 2791, 1003],

'1977': [4266, 3410, 3137, 2960, 2641, 959],

'1978': [4361, 3234, 3008, 3049, 2432, 959]

}, index=['L', 'P', 'S', 'B', 'E', 'M'])

# 计算行和列的边际总和

row\_sums = data.sum(axis=1)

col\_sums = data.sum(axis=0)

total\_sum = data.values.sum()

# 计算相对频率矩阵

P = data / total\_sum

# 计算行和列的相对频率

row\_marginals = row\_sums / total\_sum

col\_marginals = col\_sums / total\_sum

# 计算期望值矩阵

expected = np.outer(row\_marginals, col\_marginals)

# 计算标准化残差（中心化后的矩阵）

standardized\_residuals = (P - expected) / np.sqrt(expected)

# 奇异值分解 (SVD)

U, singular\_values, Vt = np.linalg.svd(standardized\_residuals)

# 计算主惯量

principal\_inertia = singular\_values\*\*2

# 计算贡献率和累计贡献率

total\_inertia = principal\_inertia.sum()

contribution\_rate = principal\_inertia / total\_inertia

cumulative\_contribution\_rate = np.cumsum(contribution\_rate)

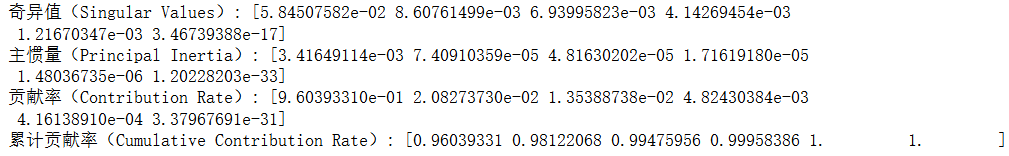
# 输出结果

print("奇异值（Singular Values）:", singular\_values)

print("主惯量（Principal Inertia）:", principal\_inertia)

print("贡献率（Contribution Rate）:", contribution\_rate)

print("累计贡献率（Cumulative Contribution Rate）:", cumulative\_contribution\_rate)



# 10.2

import numpy as np

from scipy.linalg import sqrtm

from scipy.linalg import eigh

# 假设你提供的是多个子矩阵的相关矩阵，手动创建这几个相关矩阵

matrix1 = np.array([[1, 0.6328], [0.6328, 1]])

matrix2 = np.array([[0.2412, 0.0586], [-0.0553, 0.0655]])

matrix3 = np.array([[0.2412, -0.0553], [0.0586, 0.0655]])

matrix4 = np.array([[1, 0.4248], [0.4248, 1]])

# 组合为相关矩阵形式

corr\_matrix = np.block([[matrix1, matrix2], [matrix3, matrix4]])

# 计算协方差矩阵

cov\_XX = corr\_matrix[:2, :2]

cov\_YY = corr\_matrix[2:, 2:]

cov\_XY = corr\_matrix[:2, 2:]

cov\_YX = corr\_matrix[2:, :2]

# 典型相关分析过程

eigvals\_X, eigvecs\_X = eigh(cov\_XX)

eigvals\_Y, eigvecs\_Y = eigh(cov\_YY)

# 标准化协方差矩阵

cov\_XX\_inv\_sqrt = sqrtm(np.linalg.inv(cov\_XX))

cov\_YY\_inv\_sqrt = sqrtm(np.linalg.inv(cov\_YY))

# 计算典型相关矩阵

A = cov\_XX\_inv\_sqrt @ cov\_XY @ cov\_YY\_inv\_sqrt

U, s, Vt = np.linalg.svd(A)

# 输出典型相关系数

print("典型相关系数：", s)



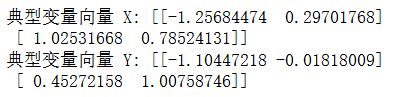
# 计算典型变量向量

canonical\_variate\_X = cov\_XX\_inv\_sqrt @ U

canonical\_variate\_Y = cov\_YY\_inv\_sqrt @ Vt.T

print("典型变量向量 X:", canonical\_variate\_X)

print("典型变量向量 Y:", canonical\_variate\_Y)



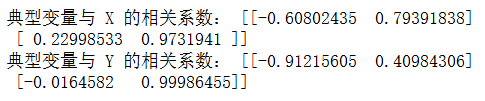
# 计算典型变量和原始变量的相关系数

corr\_X\_U = cov\_XX @ canonical\_variate\_X

corr\_Y\_V = cov\_YY @ canonical\_variate\_Y

print("典型变量与 X 的相关系数：", corr\_X\_U)

print("典型变量与 Y 的相关系数：", corr\_Y\_V)



import numpy as np

from scipy.stats import chi2

# 假设样本数 n

n = 140 # 根据你的数据实际样本量进行替换

# 典型相关系数 s 是前面计算得到的奇异值向量

# 计算 Wilks' Lambda

wilks\_lambda = np.prod(1 - s\*\*2)

# 两个变量组的维度

p = cov\_XX.shape[0]

q = cov\_YY.shape[0]

# 计算卡方统计量

chi\_square\_stat = -(n - 0.5 \* (p + q + 3)) \* np.log(wilks\_lambda)

df = p \* q

# 计算显著性水平（p 值）

p\_value = 1 - chi2.cdf(chi\_square\_stat, df)

print("Wilks' Lambda:", wilks\_lambda)

print("卡方统计量:", chi\_square\_stat)

print("自由度:", df)

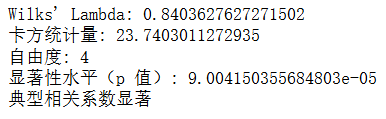
print("显著性水平（p 值）:", p\_value)

if p\_value < 0.05:

print("典型相关系数显著")

else:

print("典型相关系数不显著")



import numpy as np

from scipy.stats import chi2

# 假设样本数 n

n = 140 # 根据你的数据实际样本量进行替换

# 典型相关系数 s 是前面计算得到的奇异值向量

# 计算 Wilks' Lambda

wilks\_lambda = 1 - s[1]\*\*2

# 两个变量组的维度

p = cov\_XX.shape[0]

q = cov\_YY.shape[0]

s\_n2 = 1/s[0]\*\*2

# 计算卡方统计量

chi\_square\_stat = -(n - 1 - 0.5 \* (p + q + 3) + s\_n2) \* np.log(wilks\_lambda)

df = p \* q

# 计算显著性水平（p 值）

p\_value = 1 - chi2.cdf(chi\_square\_stat, df)

print("Wilks' Lambda:", wilks\_lambda)

print("卡方统计量:", chi\_square\_stat)

print("自由度:", df)

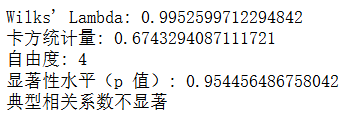
print("显著性水平（p 值）:", p\_value)

if p\_value < 0.05:

print("典型相关系数显著")

else:

print("典型相关系数不显著")



# 10.3

import numpy as np

import pandas as pd

from scipy.linalg import sqrtm, eigh

# 数据

data = {

"长子头长": [191, 195, 181, 183, 176, 208, 189, 197, 188, 192, 179, 183, 174, 190, 188, 163, 195, 186, 181, 175, 192, 174, 176, 197, 190],

"长子头宽": [155, 149, 148, 153, 144, 157, 150, 159, 152, 150, 158, 147, 150, 159, 151, 137, 155, 153, 145, 140, 154, 143, 139, 167, 163],

"次子头长": [179, 201, 185, 188, 171, 192, 190, 189, 197, 187, 186, 174, 185, 195, 187, 161, 183, 173, 182, 165, 185, 178, 176, 200, 187],

"次子头宽": [145, 152, 149, 149, 142, 152, 149, 152, 159, 151, 148, 147, 152, 157, 158, 130, 158, 148, 146, 137, 152, 147, 143, 158, 150]

}

# 转为 DataFrame

df = pd.DataFrame(data)

# 计算协方差矩阵

cov\_matrix = df.cov()

# 提取子矩阵并去掉索引

cov\_xx = cov\_matrix.loc[["长子头长", "长子头宽"], ["长子头长", "长子头宽"]].to\_numpy()

cov\_xy = cov\_matrix.loc[["长子头长", "长子头宽"], ["次子头长", "次子头宽"]].to\_numpy()

cov\_yx = cov\_matrix.loc[["次子头长", "次子头宽"], ["长子头长", "长子头宽"]].to\_numpy()

cov\_yy = cov\_matrix.loc[["次子头长", "次子头宽"], ["次子头长", "次子头宽"]].to\_numpy()

# 输出结果

print("Cov\_XX:\n", cov\_xx)

print("Cov\_XY:\n", cov\_xy)

print("Cov\_YX:\n", cov\_yx)

print("Cov\_YY:\n", cov\_yy)

# 计算相关矩阵

corr\_matrix = df.corr()

# 提取子矩阵

corr\_xx = corr\_matrix.loc[["长子头长", "长子头宽"], ["长子头长", "长子头宽"]].to\_numpy()

corr\_xy = corr\_matrix.loc[["长子头长", "长子头宽"], ["次子头长", "次子头宽"]].to\_numpy()

corr\_yx = corr\_matrix.loc[["次子头长", "次子头宽"], ["长子头长", "长子头宽"]].to\_numpy()

corr\_yy = corr\_matrix.loc[["次子头长", "次子头宽"], ["次子头长", "次子头宽"]].to\_numpy()

# 输出结果

print("Corr\_XX:\n", corr\_xx)

print("Corr\_XY:\n", corr\_xy)

print("Corr\_YX:\n", corr\_yx)

print("Corr\_YY:\n", corr\_yy)

# 计算典型相关

inv\_corr\_xx\_sqrt = np.linalg.inv(sqrtm(corr\_xx))

inv\_corr\_yy\_sqrt = np.linalg.inv(sqrtm(corr\_yy))

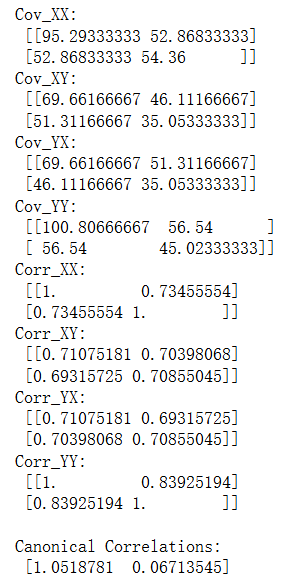
matrix\_to\_decompose = inv\_corr\_xx\_sqrt @ corr\_xy @ inv\_corr\_yy\_sqrt @ corr\_yx

eigvals, eigvecs = eigh(matrix\_to\_decompose)

# 输出典型相关系数

canonical\_correlations = np.sqrt(eigvals[::-1]) # 按降序排列

print("\nCanonical Correlations:\n", canonical\_correlations)



# 计算负的二分之一次方

corr\_xx\_inv\_sqrt = sqrtm(np.linalg.inv(corr\_xx))

corr\_yy\_inv\_sqrt = sqrtm(np.linalg.inv(corr\_yy))

# 计算典型相关矩阵

A = corr\_xx\_inv\_sqrt @ corr\_xy @ corr\_yy\_inv\_sqrt

U, s, Vt = np.linalg.svd(A)

# 输出典型相关系数

print("典型相关系数：", s)



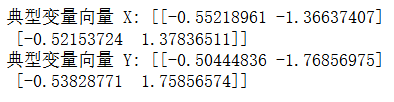
# 计算典型变量向量

canonical\_variate\_X = corr\_xx\_inv\_sqrt @ U

canonical\_variate\_Y = corr\_yy\_inv\_sqrt @ Vt.T

print("典型变量向量 X:", canonical\_variate\_X)

print("典型变量向量 Y:", canonical\_variate\_Y)



# 数据计算完成后，协方差矩阵和典型变量向量已准备好

# Step 1: 计算 D1 和 D2

D1 = np.diag(np.sqrt(np.diag(cov\_xx))) # 长子变量的对角矩阵

D2 = np.diag(np.sqrt(np.diag(cov\_yy))) # 次子变量的对角矩阵

# Step 2: 计算相关矩阵

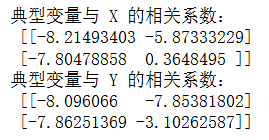
corr\_X\_U = np.linalg.inv(D1) @ (cov\_xx @ canonical\_variate\_X) # 标准化 X 与典型变量的相关矩阵

corr\_Y\_V = np.linalg.inv(D2) @ (cov\_yy @ canonical\_variate\_Y) # 标准化 Y 与典型变量的相关矩阵

# 输出结果

print("典型变量与 X 的相关系数：\n", corr\_X\_U)

print("典型变量与 Y 的相关系数：\n", corr\_Y\_V)



import numpy as np

from scipy.stats import chi2

# 假设样本数 n

n = 25 # 根据你的数据实际样本量进行替换

# 典型相关系数 s 是前面计算得到的奇异值向量

# 计算 Wilks' Lambda

wilks\_lambda = np.prod(1 - s\*\*2)

# 两个变量组的维度

p = cov\_xx.shape[0]

q = cov\_yy.shape[0]

# 计算卡方统计量

chi\_square\_stat = -(n - 0.5 \* (p + q + 3)) \* np.log(wilks\_lambda)

df = p \* q

# 计算显著性水平（p 值）

p\_value = 1 - chi2.cdf(chi\_square\_stat, df)

print("Wilks' Lambda:", wilks\_lambda)

print("卡方统计量:", chi\_square\_stat)

print("自由度:", df)

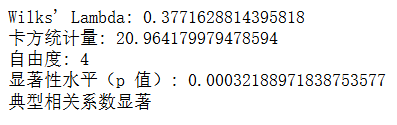
print("显著性水平（p 值）:", p\_value)

if p\_value < 0.05:

print("典型相关系数显著")

else:

print("典型相关系数不显著")



import numpy as np

from scipy.stats import chi2

# 假设样本数 n

n = 25 # 根据你的数据实际样本量进行替换

# 典型相关系数 s 是前面计算得到的奇异值向量

# 计算 Wilks' Lambda

wilks\_lambda = 1 - s[1]\*\*2

# 两个变量组的维度

p = cov\_xx.shape[0]

q = cov\_yy.shape[0]

s\_n2 = 1/s[0]\*\*2

# 计算卡方统计量

chi\_square\_stat = -(n - 1 - 0.5 \* (p + q + 3) + s\_n2) \* np.log(wilks\_lambda)

df = p \* q

# 计算显著性水平（p 值）

p\_value = 1 - chi2.cdf(chi\_square\_stat, df)

print("Wilks' Lambda:", wilks\_lambda)

print("卡方统计量:", chi\_square\_stat)

print("自由度:", df)

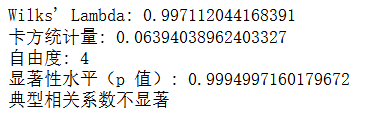
print("显著性水平（p 值）:", p\_value)

if p\_value < 0.05:

print("典型相关系数显著")

else:

print("典型相关系数不显著")



# 10.4

# 定义原始矩阵

data = np.array([

[1, 0.785, 0.81, 0.775, 0.086, 0.144, 0.14, 0.222, 0.101, 0.189, 0.199, 0.239],

[0.785, 1, 0.816, 0.813, 0.2, 0.119, 0.211, 0.301, 0.223, 0.221, 0.274, 0.235],

[0.81, 0.816, 1, 0.845, 0.041, 0.06, 0.126, 0.12, 0.039, 0.108, 0.139, 0.1],

[0.775, 0.813, 0.845, 1, 0.228, 0.122, 0.277, 0.214, 0.201, 0.156, 0.271, 0.171],

[0.086, 0.2, 0.041, 0.228, 1, 0.562, 0.457, 0.579, 0.802, 0.595, 0.512, 0.492],

[0.144, 0.119, 0.06, 0.122, 0.562, 1, 0.36, 0.705, 0.578, 0.796, 0.413, 0.739],

[0.14, 0.211, 0.126, 0.277, 0.457, 0.36, 1, 0.273, 0.606, 0.337, 0.798, 0.24],

[0.222, 0.301, 0.12, 0.214, 0.579, 0.705, 0.273, 1, 0.594, 0.725, 0.364, 0.711],

[0.101, 0.223, 0.039, 0.201, 0.802, 0.578, 0.606, 0.594, 1, 0.605, 0.698, 0.605],

[0.189, 0.221, 0.108, 0.156, 0.595, 0.796, 0.337, 0.725, 0.605, 1, 0.428, 0.697],

[0.199, 0.274, 0.139, 0.271, 0.512, 0.413, 0.798, 0.364, 0.698, 0.428, 1, 0.394],

[0.239, 0.235, 0.1, 0.171, 0.492, 0.739, 0.24, 0.711, 0.605, 0.697, 0.394, 1]

])

# 分割矩阵

# X: 前4行和前4列

# Y: 后8行和后8列

# XY: 前4行和后8列

# YX: 后8行和前4列

R\_XX = data[:4, :4]

R\_YY = data[4:, 4:]

R\_XY = data[:4, 4:]

R\_YX = data[4:, :4]

# 将结果转为DataFrame，方便查看

XX\_df = pd.DataFrame(XX, columns=["x1", "x2", "x3", "x4"], index=["x1", "x2", "x3", "x4"])

YY\_df = pd.DataFrame(YY, columns=["y1", "y2", "y3", "y4", "y5", "y6", "y7", "y8"],

index=["y1", "y2", "y3", "y4", "y5", "y6", "y7", "y8"])

XY\_df = pd.DataFrame(XY, columns=["y1", "y2", "y3", "y4", "y5", "y6", "y7", "y8"],

index=["x1", "x2", "x3", "x4"])

YX\_df = pd.DataFrame(YX, columns=["x1", "x2", "x3", "x4"],

index=["y1", "y2", "y3", "y4", "y5", "y6", "y7", "y8"])

# 输出结果

print("XX:\n", XX\_df)

print("\nYY:\n", YY\_df)

print("\nXY:\n", XY\_df)

print("\nYX:\n", YX\_df)

XX:

x1 x2 x3 x4

x1 1.000 0.785 0.810 0.775

x2 0.785 1.000 0.816 0.813

x3 0.810 0.816 1.000 0.845

x4 0.775 0.813 0.845 1.000

YY:

y1 y2 y3 y4 y5 y6 y7 y8

y1 1.000 0.562 0.457 0.579 0.802 0.595 0.512 0.492

y2 0.562 1.000 0.360 0.705 0.578 0.796 0.413 0.739

y3 0.457 0.360 1.000 0.273 0.606 0.337 0.798 0.240

y4 0.579 0.705 0.273 1.000 0.594 0.725 0.364 0.711

y5 0.802 0.578 0.606 0.594 1.000 0.605 0.698 0.605

y6 0.595 0.796 0.337 0.725 0.605 1.000 0.428 0.697

y7 0.512 0.413 0.798 0.364 0.698 0.428 1.000 0.394

y8 0.492 0.739 0.240 0.711 0.605 0.697 0.394 1.000

XY:

y1 y2 y3 y4 y5 y6 y7 y8

x1 0.086 0.144 0.140 0.222 0.101 0.189 0.199 0.239

x2 0.200 0.119 0.211 0.301 0.223 0.221 0.274 0.235

x3 0.041 0.060 0.126 0.120 0.039 0.108 0.139 0.100

x4 0.228 0.122 0.277 0.214 0.201 0.156 0.271 0.171

YX:

x1 x2 x3 x4

y1 0.086 0.200 0.041 0.228

y2 0.144 0.119 0.060 0.122

y3 0.140 0.211 0.126 0.277

y4 0.222 0.301 0.120 0.214

y5 0.101 0.223 0.039 0.201

y6 0.189 0.221 0.108 0.156

y7 0.199 0.274 0.139 0.271

y8 0.239 0.235 0.100 0.171

# 标准化协方差矩阵

R\_XX\_inv\_sqrt = sqrtm(np.linalg.inv(R\_XX))

R\_YY\_inv\_sqrt = sqrtm(np.linalg.inv(R\_YY))

# 计算典型相关矩阵

A = R\_XX\_inv\_sqrt @ R\_XY @ R\_YY\_inv\_sqrt

U, s, Vt = np.linalg.svd(A)

# 输出典型相关系数

print("典型相关系数：", s)



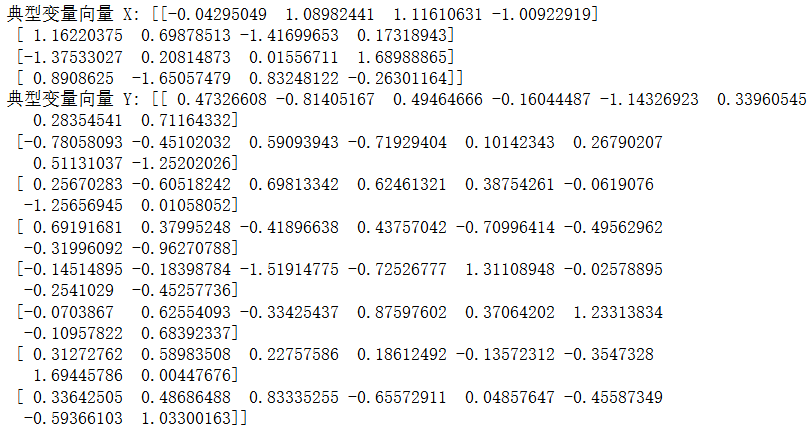
# 计算典型变量向量

canonical\_variate\_X = R\_XX\_inv\_sqrt @ U

canonical\_variate\_Y = R\_YY\_inv\_sqrt @ Vt.T

print("典型变量向量 X:", canonical\_variate\_X)

print("典型变量向量 Y:", canonical\_variate\_Y)



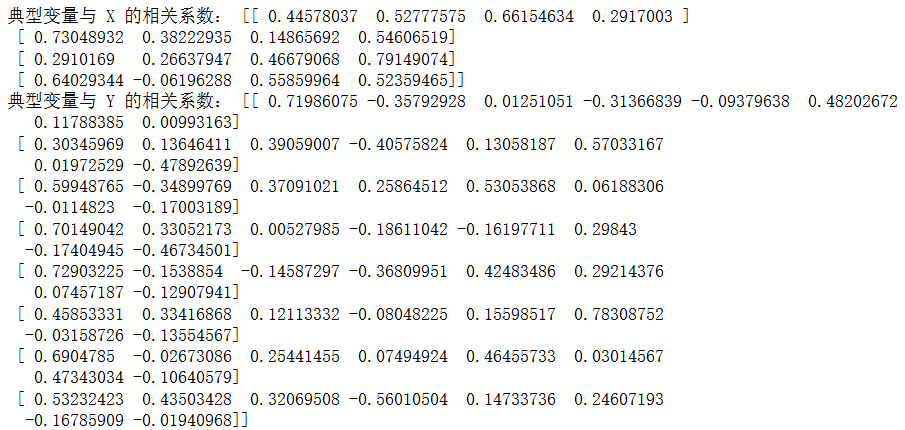
# 计算典型变量和原始变量的相关系数

corr\_X\_U = R\_XX @ canonical\_variate\_X

corr\_Y\_V = R\_YY @ canonical\_variate\_Y

print("典型变量与 X 的相关系数：", corr\_X\_U)

print("典型变量与 Y 的相关系数：", corr\_Y\_V)



import numpy as np

from scipy.stats import chi2

# 假设样本数 n

n = 110 # 根据你的数据实际样本量进行替换

# 典型相关系数 s 是前面计算得到的奇异值向量

# 计算 Wilks' Lambda

wilks\_lambda = np.prod(1 - s\*\*2)

# 两个变量组的维度

p = R\_XX.shape[0]

q = R\_YY.shape[0]

# 计算卡方统计量

chi\_square\_stat = -(n - 0.5 \* (p + q + 3)) \* np.log(wilks\_lambda)

df = p \* q

# 计算显著性水平（p 值）

p\_value = 1 - chi2.cdf(chi\_square\_stat, df)

print("Wilks' Lambda:", wilks\_lambda)

print("卡方统计量:", chi\_square\_stat)

print("自由度:", df)

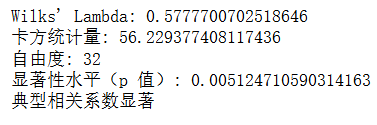
print("显著性水平（p 值）:", p\_value)

if p\_value < 0.05:

print("典型相关系数显著")

else:

print("典型相关系数不显著")



import numpy as np

from scipy.stats import chi2

# 假设样本数 n

n = 110 # 根据你的数据实际样本量进行替换

# 使用从 s[1] 开始的典型相关系数

s\_used = s[1:]

# 典型相关系数 s 是前面计算得到的奇异值向量

# 计算 Wilks' Lambda，从 s[1] 开始

wilks\_lambda = np.prod(1 - s\_used\*\*2)

# 两个变量组的维度

p = R\_XX.shape[0]

q = R\_YY.shape[0]

s\_n2 = 1/s[0]\*\*2

# 计算卡方统计量

chi\_square\_stat = -(n - 1 - 0.5 \* (p + q + 3) + s\_n2) \* np.log(wilks\_lambda)

df = p \* q

# 计算显著性水平（p 值）

p\_value = 1 - chi2.cdf(chi\_square\_stat, df)

print("Wilks' Lambda:", wilks\_lambda)

print("卡方统计量:", chi\_square\_stat)

print("自由度:", df)

print("显著性水平（p 值）:", p\_value)

if p\_value < 0.05:

print("典型相关系数显著")

else:

print("典型相关系数不显著")

