第一题

# %% [markdown]

# ## 调用

# %%

import pandas as pd

import matplotlib.pyplot as plt

excel = pd.read\_excel("B题数据.xlsx")

lon\_lat = pd.read\_excel('等高线图.xlsx')

x\_label = excel['断面名称']

y\_labels = {

'高锰酸盐指数(mg/L)': 'purple',

'氨氮(mg/L)': 'red',

'总磷(mg/L)': 'green',

'总氮(mg/L)': 'orange'

}

plt.rcParams['font.sans-serif'] = ['SimHei'] #显示中文

plt.rcParams['axes.unicode\_minus']=False #用来正常显示负号

# %% [markdown]

# ## 第一题

#

# 1、分析北京地区水系中高锰酸盐指数、氨氮、总磷、总氮的总体分布特征。

# \*（分布特征分析）\*

#

# 另外，为了绘图，我把广北滨河路（桥）的（桥）给去掉了，特此说明

# %% [markdown]

# ### 数据分析

# %%

ana = excel[['高锰酸盐指数(mg/L)','氨氮(mg/L)','总磷(mg/L)','总氮(mg/L)']]

print(ana.describe())

# %% [markdown]

# ### 地区分布条形图

# %%

plt.figure(figsize=[27.50, 12.50])

# 遍历y\_labels字典，为每个指标创建子图

for i, (ylabel, color) in enumerate(y\_labels.items(), start=1):

plt.subplot(2, 2, i) # 根据循环的索引选择子图位置

plt.bar(x\_label, excel[ylabel], color=color, label=ylabel)

plt.xticks(rotation=40, fontsize=10)

plt.title(ylabel,fontsize=20)

plt.gca().xaxis.set\_tick\_params(labelrotation=40)

# 显示图例

# plt.legend()

# 显示图形

plt.show()

# %% [markdown]

# ### 等高线图

# %%

point\_label = list(x\_label)

# %%

from scipy.interpolate import griddata

# print(lon\_lat['经度'])

# 定义一维数组

x = lon\_lat['经度']

y = lon\_lat['纬度']

point=np.vstack([x,y]).T

# print(list(point))

print(lon\_lat.columns)

xx,yy=np.meshgrid(np.linspace(x.min(),x.max(),200),np.linspace(y.min(),y.max(),200))

plt.figure(figsize=[27.50, 12.50])

for i, (ylabel, color) in enumerate(y\_labels.items(), start=1):

index = lon\_lat[ylabel]

zz=griddata(points=point, values=index, xi=(xx,yy), method="cubic")

plt.subplot(2, 2, i)

plt.xticks(fontsize=20)

plt.yticks(fontsize=20)

plt.title(ylabel,fontsize=25)

plt.scatter(x, y, c=color, s=30)

plt.contourf(xx, yy, zz, alpha=0.75, cmap=f'{color.title()}s')

for j, name in enumerate(x\_label):

plt.annotate(name, xy = list(point)[j], textcoords="offset points", xytext=(0,10), ha='center', fontsize=10)

plt.tight\_layout()

plt.show()

第二题

# %% [markdown]

# ## 调用

# %%

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# from scipy.stats import spearmanr

import pingouin as pg

from sklearn.linear\_model import LinearRegression

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

plt.rcParams['font.sans-serif'] = ['SimHei'] #显示中文

plt.rcParams['axes.unicode\_minus']=False #用来正常显示负号

plt.rcParams["figure.figsize"] = [27.50, 12.50]

excel = pd.read\_excel("B题数据.xlsx")

data = excel.iloc[0:,2:]

ana\_ff1 = pd.DataFrame(data)

# %% [markdown]

# ## 第二题

#

# 2、分析高锰酸盐指数、氨氮、总磷、总氮的高低，与哪些因素具有相关性，仅限给出的数据项目，并据此评估呈现当前分布特征的原因。

# \*（相关性分析，评估原因）\*

# %% [markdown]

# ### 主成分分析法

# %% [markdown]

# #### 创建类

# %%

import numpy as np

from numpy import linalg

class PCA:

''' dataset 形如array([样本1,样本2,...,样本m]),每个样本是一个n维的ndarray'''

def \_\_init\_\_(self, dataset):

# 这里的参数跟上文是反着来的(每行是一个样本)，需要转置一下

self.dataset = np.matrix(dataset, dtype='float64').T

# def standard2(self, dataset):

# scaler = StandardScaler()

# scaled\_data = scaler.fit\_transform(np.asarray(dataset))

# # scaled\_df = pd.DataFrame(scaled\_data)

# # return scaled\_df

def standard(self, dataset):

'''标准化'''

for (index, line) in enumerate(dataset):

dataset[index] -= np.mean(line)

# np.std(line, ddof = 1)即样本标准差(分母为n - 1)

dataset[index] /= np.std(line, ddof = 1)

def get\_eig\_vector(self, data, Cov):

'''求特征值和特征向量'''

eigs, vectors = linalg.eig(Cov)

# print(vectors)

for i in range(len(eigs)):

data.append((eigs[i], vectors[:, i]))

# 按照特征值从大到小排序

data.sort(key = lambda x: x[0], reverse = True)

return eigs

def principal\_comps(self, threshold = 0.85):

'''求主成分。threshold可选参数表示方差累计达到threshold后就不再取后面的特征向量。

返回值是特征值、特征向量、方差贡献率、累计方差贡献率。'''

ret = []

data = []

self.standard(self.dataset)

Cov = np.cov(self.dataset)

eigs = self.get\_eig\_vector(data, Cov)

sum = 0

for comp in data:

sum += comp[0] / np.sum(eigs)

ret.append(

tuple(

map(lambda x: np.round(x, 5)

# 特征值、特征向量、方差贡献率、累计方差贡献率

, (comp[0], comp[1], comp[0] / np.sum(eigs), sum))

)

)

if sum > threshold:

return ret

return ret

# PMx = PCA(anal)

# op = PMx.principal\_comps(threshold=0.85)

# %% [markdown]

# #### 数据准备

# %%

#准备原始数据

anal = excel.drop(['高锰酸盐指数(mg/L)', '氨氮(mg/L)','总磷(mg/L)', '断面名称', '总氮(mg/L)', '水质类别'], axis=1)

index = ['水温(℃)','pH(无量纲)','溶解氧(mg/L)','电导率(μS/cm)','浊度(NTU)']

#输出用

output = [['高锰酸盐', '高锰酸盐指数(mg/L)'], ['氨氮', '氨氮(mg/L)'], ['总磷', '总磷(mg/L)'],['总氮', '总氮(mg/L)']]

# print(excel.columns)

#绘制影响因子图用

x\_label = pd.read\_excel('影响因子图.xlsx', sheet\_name='Sheet5', header=None).squeeze()

y\_labels = {

'高锰酸盐指数(mg/L)': 'purple',

'氨氮(mg/L)': 'red',

'总磷(mg/L)': 'green',

'总氮(mg/L)': 'orange'

}

def standard(data):

'''标准化'''

opt = (data - data.mean()) / data.std()

return opt

# %% [markdown]

# #### 分析

# %%

with pd.ExcelWriter('output.xlsx') as writer:

for ii in output:

PMx = PCA(anal)

op = PMx.principal\_comps(threshold=0.85)

opd = pd.DataFrame(op, columns=['特征值', '特征向量', '方差贡献率', '累计方差贡献率'])

# print(opd)

opd.to\_excel('analyse.xlsx', index=True)

matrix = pd.DataFrame(index=index)

for i in range(len(op)):

matrix[i] = op[i][1]

#result是主成分

result = np.dot(anal, matrix)

# print(result)

model = LinearRegression()

y = np.array(excel[ii[1]])

x = np.array(result)

model.fit(y=standard(y), X=x)

result\_mid = np.dot(matrix, model.coef\_.T)

pd.set\_option('display.float\_format', lambda x: '%.6f' % x)

# print(result2)

matrix['主成分回归系数'] = result\_mid

# print(matrix)

matrix.to\_excel(writer, sheet\_name=ii[0], float\_format='%.6f', index=True)

# print(result)

# 载荷系数=原始变量与主成分之间的相关系数

for i in range(3):

for j in index:

load = pd.DataFrame([result[:, i], anal[j]])

# print(load)

print(load.iloc[0].corr(load.iloc[1]))

print('\n')

# %% [markdown]

# #### 影响因子

# %%

for i, (ylabel, color) in enumerate(y\_labels.items(), start=1):

inf = pd.read\_excel('影响因子图.xlsx', sheet\_name=f'Sheet{i}', header=None).squeeze()

# print(list(inf))

# print(list(x\_label))

plt.subplot(2, 2, i) # 根据循环的索引选择子图位置

plt.bar(list(x\_label[i-1]), list(inf), color=color, label=ylabel)

plt.xticks(rotation=40, fontsize=15)

plt.title(ylabel,fontsize=20)

plt.gca().xaxis.set\_tick\_params(labelrotation=40)

plt.tight\_layout()

plt.show()

# %% [markdown]

# ## 调用

# %%

import numpy as np

import pandas as pd

# plt.rcParams["figure.figsize"] = [27.50, 12.50]

excel = pd.read\_excel("B题数据.xlsx")

ana = pd.DataFrame(excel).iloc[:, 2:]

# print(ana)

第三题

# %% [markdown]

# ## 第三题

#

# 3、国家标准的I-V分类主要考虑水质本身对人和环境的影响，请从居民、游客生活体验的角度出发，对各个监测站的水质重新分类，并给出分类的依据。

# \*（分类，分类依据制定）\*

# %% [markdown]

# ### 熵权法计算原始数据打分到二级标准（实际是一级）的权重

#

# 注：不包括水质类别相对应的数据列。一方面是制定打分标准时忘记了，另一方面是后来反思也认为数据给出的水质类别与其他列是相关的，不必算进新的打分标准中

# %%

stdana = pd.DataFrame()

weight = []

k = 1 / np.log(len(ana.index))

for i, col in enumerate(ana):

stdana[i] = ((ana[col]-min(ana[col]))/max(ana[col]-min(ana[col])))

stdana[i] /= sum(stdana[i])

for ji, j in enumerate(stdana[i]):

if j != 0:

stdana.iloc[ji, i] \*= np.log(j)

else:

stdana.iloc[ji, i] = 0

weight.append(1-(-k\*sum(stdana[i])))

for k in range(len(weight)):

weight[k] /= sum(weight)

weight = {k:v for k,v in zip(ana.columns, weight)}

# print(weight)

# %% [markdown]

# ### 分数计算

# %%

weight1 = [weight['水温(℃)'], weight['浊度(NTU)'], weight['溶解氧(mg/L)']]

weight2 = [weight['高锰酸盐指数(mg/L)'], weight['电导率(μS/cm)'], weight['pH(无量纲)'], weight['氨氮(mg/L)']]

weight3 = [weight['氨氮(mg/L)'], weight['溶解氧(mg/L)'], weight['总磷(mg/L)'], weight['总氮(mg/L)']]

weight1 /= sum(weight1)

weight2 /= sum(weight2)

weight3 /= sum(weight3)

weightn = [weight1, weight2, weight3]

s = pd.DataFrame(columns=['舒适度', '安全性', '清洁度'])

for i in range(3):

score = pd.read\_excel('分.xlsx', sheet\_name=i)

s.iloc[:, i] = np.dot(weightn[i], score.iloc[0:, 1:])

weightupper = [0.2, 0.4 ,0.4]

s\_last = pd.concat([excel['断面名称'], pd.Series(np.dot(s, weightupper))], axis=1)

s\_last.columns = ['断面名称', '总分']

s\_last\_sorted = sorted(s\_last.values, key=lambda x: x[1], reverse=True)

# print(excel['断面名称'])

# print(s\_last\_sorted)

total\_scores = np.array([item[1] for item in s\_last\_sorted])

standardized\_scores = (total\_scores - min(total\_scores)) / (max(total\_scores) - min(total\_scores))

levels = 6 - np.digitize(standardized\_scores, np.linspace(-0.00001, 1.00001, 6), right=True)

s\_last\_sorted\_with\_scores = [np.append(item, [score, level]) for item, score, level in zip(s\_last\_sorted, standardized\_scores, levels)]

for item in s\_last\_sorted\_with\_scores:

print(item)

print(s\_last\_sorted)