第一题

# import necessary libraries

import numpy as np

import matplotlib.pyplot as plt

from scipy.stats import norm, t

# 设置中文字体（可选）

plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签

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

# 创建x轴数值范围（从-4到4，包含1000个点）

x = np.linspace(-4, 4, 1000)

# 计算标准正态分布在每个x点的概率密度值

y = norm.pdf(x, 0, 1)  # 均值为0，标准差为1

# compute t-distr

y\_t1 = t.pdf(x, df = 1)

y\_t3 = t.pdf(x, df = 3)

y\_t30 = t.pdf(x, df = 30)

y\_t100 = t.pdf(x, df = 100)

# 创建图形

plt.figure(figsize=(8, 6))

# plot stan normal distr

plt.plot(x, y, 'b-', linewidth=2, label='标准正态分布')

# plot t-distr

plt.plot(x, y\_t1, 'c--', linewidth=1, label='t分布（df=1）')

plt.plot(x, y\_t3, 'm--', linewidth=1, label='t分布（df=3）')

plt.plot(x, y\_t30, 'g--', linewidth=1, label='t分布（df=30）')

plt.plot(x, y\_t100, 'r--', linewidth=1, label='t分布（df=100）')

# 添加标题和标签

plt.title('标准正态分布与t分布密度曲线对比', fontsize=16)

plt.xlabel('x值', fontsize=12)

plt.ylabel('概率密度', fontsize=12)

# 添加网格

plt.grid(True, linestyle='--', alpha=0.7)

# 添加图例

plt.legend(fontsize=12)

# 设置x轴和y轴范围

plt.xlim(-4, 4)

plt.ylim(0, 0.45)

# 显示图形

plt.tight\_layout()

plt.show()

第二题

import numpy as np

import sys

import io

# 设置标准输出编码为UTF-8

sys.stdout = io.TextIOWrapper(sys.stdout.buffer, encoding='utf-8')

# 输出 m 个样本，sigma2 方差情况下，100 次重复实验结果的方差均值

def simulate\_variance(m, sigma2, R=100, seed=123):

    """模拟计算样本方差"""

    np.random.seed(seed)

    # 循环 100 次实验的方差，存入 s2\_values

    s2\_values = []

    for i in range(R):

        sample = np.random.normal(0, np.sqrt(sigma2), m)

        s2 = np.var(sample, ddof=1)

        s2\_values.append(s2)

    # compute mean of 100 experiments 的方差

    s2\_mean = np.mean(s2\_values)

    return {

        'm': m,

        'sigma2': sigma2,

        'simulated\_mean': s2\_mean

    }

# 计算两组样本方差的加权平均值

def simulate\_weighted\_mean(m, n, sigma2):

    """模拟计算加权均值的方差"""

    var\_x = simulate\_variance(m, sigma2)['simulated\_mean']

    var\_y = simulate\_variance(n, sigma2)['simulated\_mean']

    weighted\_var = ((m-1) \* var\_x + (n-1) \* var\_y) / (m + n - 2)

    return weighted\_var

# 测试多组参数

parameter\_sets = [

    (5, 100, 1), (100, 100, 1), (5, 100, 25), (100, 100, 25)

]

print("不同参数下的样本方差模拟结果：")

print("=" \* 70) # create table

# 存储不同参数输出的数组, 并将计算结果与 sigma2 进行比较（这里采用计算误差的方式）

resultsx = []

resultsy = []

resultsw = []

errorsx = []

errorsy = []

errorsw = []

for n, m, sigma2 in parameter\_sets:

    resultx = simulate\_variance(m, sigma2)

    resulty = simulate\_variance(n, sigma2)

    resultw = simulate\_weighted\_mean(m, n, sigma2)

    errorx = resultx['simulated\_mean'] - sigma2

    errory = resulty['simulated\_mean'] - sigma2

    errorw = resultw - sigma2

    resultsx.append(resultx)

    resultsy.append(resulty)

    resultsw.append(resultw)

    errorsx.append(errorx)

    errorsy.append(errory)

    errorsw.append(errorw)

    print(f"m = {m:3d}, n = {n:3d}, σ² = {sigma2:3d} | "

        f"模拟均值X: {resultx['simulated\_mean']:7.4f} | "

        f"模拟均值Y: {resulty['simulated\_mean']:7.4f} | "

        f"模拟均值S\_w^2: {resultw:7.4f} | "

        f"误差X: {errorx:7.4f} | "

        f"误差Y: {errory:7.4f} | "

        f"误差S\_w^2: {errorw:7.4f}")

print("=" \* 70) # end table

第三题

import numpy as np

import matplotlib.pyplot as plt

from scipy.stats import norm

from scipy.special import gammaln  # 使用伽马函数的对数

import sys

import io

# 设置标准输出编码为UTF-8

sys.stdout = io.TextIOWrapper(sys.stdout.buffer, encoding='utf-8')

# 绘图中文

plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签

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

# 比较不同样本量下，样本均值的密度函数与其渐近分布的密度函数图像

k = 1

true\_mean = 4 \* k

true\_var = 8 \* k  # variance of X\_i

n\_values = [5, 10, 50, 100]

x\_plot = np.linspace(0.01, 8, 1000)

# 使用对数空间计算的样本均值密度函数

def sample\_mean\_pdf(x, n, k):

    degree1 = 2 \* n \* k

    degree2 = 1 / 2

    # 在对数空间计算，避免数值溢出

    log\_term1 = degree1 \* np.log(degree2) - gammaln(degree1)

    log\_term2 = (degree1 - 1) \* np.log(n \* x)

    log\_term3 = -n \* x \* degree2

    log\_result = log\_term1 + log\_term2 + log\_term3 + np.log(n)

    return np.exp(log\_result)

# 样本均值的渐近正态分布密度函数

def asymptotic\_normal\_pdf(x, n, k):

    return norm.pdf(x, 4 \* k, np.sqrt(8 \* k / n))

# 创建图形

fig, axes = plt.subplots(2, 2, figsize=(8, 6))

axes = axes.flatten()

# 为每个n值绘制图像

for i, n in enumerate(n\_values):

    # 计算精确密度

    exact\_density = sample\_mean\_pdf(x\_plot, n, k)

    # 计算渐近正态密度

    asymptotic\_density = asymptotic\_normal\_pdf(x\_plot, n, k)

    # 绘制图像

    ax = axes[i]

    ax.plot(x\_plot, exact\_density, 'b-', linewidth=2, label='精确分布')

    ax.plot(x\_plot, asymptotic\_density, 'r--', linewidth=2, label='渐近正态')

    # 设置标题和标签

    ax.set\_title(f'n = {n}')

    ax.set\_xlabel('x')

    ax.set\_ylabel('密度')

    ax.legend()

    ax.grid(True, alpha=0.3)

plt.tight\_layout()

plt.show()