**深 圳 大 学 实 验 报 告**

**课程名称：­ 概率论与数理统计**

**实验项目名称： 验证中心极限定理的正确性及其应用**

**学院： 电子与信息工程学院**

**专业： 电子信息工程**

**指导教师： 陈昌盛**

**报告人： 陈应权 学号： 2022280297**

**班级： 文华班**

**实验时间： 2023.11.30-2023.12.19**

**实验报告提交时间： 2023.12.19**

**教务处制**

|  |
| --- |
| Aim of Experiment:  1. Familiar with the central limit theorem.  2. Understand the implementation of the central limit theorem in python.  3. Know how to visualize data in different distributions. |
| Experiment Content:  一. Experiment 1 - Exponentially distributed population  1. We can see that the distribution of our population is far from normal! In the following code, assuming that 𝜃=4, please calculate the mean and the standard deviation of the population.  2. Repeat the above process, but with a much larger sample size (𝑛=500)  3. Calculate the standard deviation of sample means for 500 samples  二. Experiment 2 - Binomially distributed population  1. Follow a similar approach and plot the sampling distribution obtained with a large sample size ( 𝑛=500) for a Binomially distributed variable with parameters 𝑘=30 and 𝑝=0.9. Drawing 50 random samples of size 500 from a Binomial distribution.  2. Compute the mean of sample means. If you calculate correctly, mean of sample means is close to the population mean.  3. compute the standard deviation of sample means. If you calculate correctly, standard deviation of sample means is close to population standard deviation divided by square root of sample size.  三. Experiment 3 - An Application of CLT in Investing/Trading  Visualize both the returns and their distribution according to the diagrams below.  四. Experiment 4  Look at the temperature recording for the city of Detroit,use continuous distributions we have studied to approximate the observed distributions.  In addition, the 7 Day Moving Average Plot, the temparature variation of different cities in a single figure, the visualization of humidity, pressure and temperature of Detroit in a single figure. |
| Experiment Process：  一.  1.   1. mu=theta 2. sd=theta   2.   1. *# Sample size* 2. sample\_size = 500 3. # Creating DataFrame with index x1 to x500 4. index\_values = [f'x{i}' for i in range(1, sample\_size + 1)] 5. df500 = pd.DataFrame(index=index\_values) 6. for i in range(1,51): 7. exponential\_sample = np.random.exponential(theta, sample\_size) 8. col = f'sample {i}' 9. df500[col] = exponential\_sample 10. # Calculating sample means and plotting their distribution 11. df500\_sample\_means = df500.mean(axis=0)  # Calculate means for each column (sample) 12. sns.distplot(df500\_sample\_means)   3.   1. std\_dev\_sample\_means\_500 = df500\_sample\_means.std() 2. print(std\_dev\_sample\_means\_500)   二.  1.   1. k = 30  *# Number of trials* 2. p = 0.9  *# Probability of success* 3. *# Sample size* 4. sample\_size = 500 5. *# Creating DataFrame with index x1 to x500* 6. index\_values = [f'x{i}' for i in range(1, sample\_size + 1)] 7. df\_binomial = pd.DataFrame(index=index\_values) 8. *# Generating 50 samples of binomial distribution with parameters n=k, p=p, and size=sample\_size* 9. for i in range(1,51): 10. binomial\_sample = np.random.binomial(n=k, p=p, size=sample\_size) 11. col = f'sample {i}' 12. df\_binomial[col] = binomial\_sample 14. df\_binomial\_means = df\_binomial.mean(axis=0)  *# Calculate means for each column (sample)* 15. sns.distplot(df\_binomial\_means) *# Plotting the distribution of binomial sample means*   2.   1. *# Calculating the means of each sample in the binomial DataFrame* 2. sample\_means = df\_binomial.mean() 3. *# Calculating the overall mean of all sample means* 4. overall\_mean = sample\_means.mean() 5. *# Printing the overall mean* 6. print(overall\_mean) 7. *# Calculating the expected mean according to the Central Limit Theorem (CLT) for the binomial distribution* 8. clt\_expected\_mean = k \* p 9. *# Printing the CLT expected mean* 10. print(clt\_expected\_mean)   3.   1. *# Calculating the standard deviation of binomial sample means* 2. std\_dev\_binomial\_means = df\_binomial\_means.std() 3. *# Printing the calculated standard deviation* 4. print(std\_dev\_binomial\_means) 5. *# Calculating the expected standard deviation according to the Central Limit Theorem (CLT) for the binomial distribution* 6. clt\_expected\_std = np.sqrt(k \* p \* (1 - p) / sample\_size) 7. *# Printing the CLT expected standard deviation* 8. print(clt\_expected\_std)   三.   1. *# modify this cell* 2. *# Visualizing the daily log returns* 3. sns.set(style="white", palette="muted", color\_codes=True) 4. plt.figure(figsize=(12,5)) 5. plt.subplot(1,2,1) 6. *# Plot a simple histogram with binsize determined automatically.* 7. *# Please add the labels of the x, y axes and titles in the figure.* 8. *# Tips: sns.lineplot(daily\_data.index,daily\_data['daily\_return'], color="r")* 9. *#* 10. *# YOUR CODE HERE* 11. *#* 12. *# Plotting a line plot of daily log returns* 13. sns.lineplot(x=daily\_data.index, y=daily\_data['daily\_return'], color="r") 14. plt.title('Daily Log Returns')  *# Title of the plot* 15. plt.xlabel('Date')  *# X-axis label* 16. plt.ylabel('Log Return')  *# Y-axis label* 17. *# Creating the second subplot (currently missing subplot creation)* 18. plt.subplot(1, 2, 2)  *# This line sets the position for the second subplot in a 1x2 grid* 19. *# Plot a simple histogram with binsize determined automatically.* 20. *# Please add the labels of the x, y axes and titles in the figure.* 21. *# Tips: sns.distplot(daily\_data['daily\_return'], kde=False, color="r")* 22. *#* 23. *# YOUR CODE HERE* 24. *#* 25. *# Plotting a histogram of daily log returns without kernel density estimate (kde=False)* 26. sns.histplot(daily\_data['daily\_return'], kde=False, color="r") 27. *# Setting title and labels for the histogram* 28. plt.title('Histogram of Daily Log Returns')  *# Title of the histogram* 29. plt.xlabel('Log Return')  *# X-axis label* 30. plt.ylabel('Frequency')  *# Y-axis label* 31. *# To improve layout* 32. plt.tight\_layout() 33. *# Show the plots* 34. plt.show() 35. *# Remember to use the command to visualize.* 36. *# You can use "plt.tight\_layout()" to make the layout more beautiful.*   四.  1.   1. import pandas as pd 2. import numpy as np 3. import matplotlib.pyplot as plt 4. from scipy.stats import norm, expon, gamma, lognorm, beta, weibull\_min 5. *# Load the hourly weather dataset for Detroit from a CSV file* 6. weather\_data = pd.read\_csv('temperature.csv') 7. *# Extract temperature data (assuming it's in the 'temperature' column)* 8. temperature\_data = weather\_data['Detroit'].dropna() 9. *# Plotting histogram with 100 bins* 10. plt.hist(temperature\_data, bins=100, density=True, alpha=0.6, color='g', edgecolor='black') 11. *# Fit different continuous distributions* 12. fitting\_distributions = [norm, expon, gamma, lognorm, beta, weibull\_min] 13. for distribution in fitting\_distributions: 14. *# Estimate distribution parameters* 15. params = distribution.fit(temperature\_data) 17. *# Generate x values for the PDF plot* 18. x = np.linspace(temperature\_data.min(), temperature\_data.max(), 1000) 20. *# Plot the PDF of the distribution* 21. pdf = distribution.pdf(x, \*params) 22. plt.plot(x, pdf, label=distribution.name) 23. *# Add labels and title* 24. plt.xlabel('Temperature') 25. plt.ylabel('Probability Density') 26. plt.title('Histogram and Fitted Distributions of Temperature in Detroit') 27. plt.legend() 28. *# Show the plot* 29. plt.show()   2.   1. import pandas as pd 2. import matplotlib.pyplot as plt 3. *# Load your dataset (replace 'your\_dataset.csv' with the actual file name or data source)* 4. df = pd.read\_csv('Detroit.csv') 5. *# Assuming your dataset has columns like 'datetime', 'humidity', 'temperature', and 'pressure'* 6. *# Convert 'datetime' to datetime format* 7. df['datetime'] = pd.to\_datetime(df['datetime']) 8. *# Create a figure with a single subplot* 9. fig, ax = plt.subplots(figsize=(12, 8)) 10. *# Plot humidity in blue* 11. ax.plot(df['datetime'], df['humidity'], color='blue', label='Humidity', alpha=0.7) 12. *# Plot temperature in red* 13. ax.plot(df['datetime'], df['temperature'], color='red', label='Temperature', alpha=0.7) 14. *# Create a second y-axis for pressure on the right* 15. ax2 = ax.twinx() 16. ax2.plot(df['datetime'], df['pressure'], color='green', label='Pressure', alpha=0.7) 17. *# Customize axes labels and legends* 18. ax.set\_xlabel('Date') 19. ax.set\_ylabel('Humidity (%) and Temperature (°C)', color='black') 20. ax2.set\_ylabel('Pressure (Pa)', color='black') 21. *# Combine legends from both y-axes* 22. lines, labels = ax.get\_legend\_handles\_labels() 23. lines2, labels2 = ax2.get\_legend\_handles\_labels() 24. ax2.legend(lines + lines2, labels + labels2, loc='upper left') 25. *# Customize the overall layout* 26. plt.title('Humidity, Temperature, and Pressure in Detroit') 27. plt.show()   3.   1. import pandas as pd 2. import matplotlib.pyplot as plt 3. *# Load your dataset (replace 'your\_dataset.csv' with the actual file name or data source)* 4. df = pd.read\_csv('Detroit.csv') 5. *# Assuming your dataset has columns like 'datetime', 'temperature'* 6. *# Convert 'datetime' to datetime format* 7. df['datetime'] = pd.to\_datetime(df['datetime']) 8. *# Sort the DataFrame by datetime* 9. df = df.sort\_values(by='datetime') 10. *# Calculate the 7-day moving average for temperature* 11. df['temperature\_7day\_avg'] = df['temperature'].rolling(window=150).mean() 12. *# Create a figure with a single subplot* 13. fig, ax = plt.subplots(figsize=(12, 8)) 14. *# Plot the temperature* 15. ax.plot(df['datetime'], df['temperature'], label='Temperature', alpha=0.7) 16. *# Plot the 7-day moving average* 17. ax.plot(df['datetime'], df['temperature\_7day\_avg'], label='7-day Moving Average', color='red', linewidth=2) 18. *# Customize axes labels and legends* 19. ax.set\_xlabel('Date') 20. ax.set\_ylabel('Temperature (°C)') 21. ax.legend() 22. *# Customize the overall layout* 23. plt.title('Temperature and 7-day Moving Average in Detroit') 24. plt.show()   4.   1. import pandas as pd 2. import matplotlib.pyplot as plt 3. *# Load your dataset (replace 'your\_dataset.csv' with the actual file name or data source)* 4. df = pd.read\_csv('temperature.csv') 5. *# Assuming your dataset has columns like 'datetime', 'Vancouver', 'Portland', 'San Francisco'* 6. *# Convert 'datetime' to datetime format* 7. df['datetime'] = pd.to\_datetime(df['datetime']) 8. *# Create a figure with a single subplot* 9. fig, ax = plt.subplots(figsize=(12, 8)) 10. *# Plot temperature for each city* 11. ax.plot(df['datetime'], df['Vancouver'], label='Vancouver', alpha=0.7) 12. ax.plot(df['datetime'], df['Portland'], label='Portland', alpha=0.7) 13. ax.plot(df['datetime'], df['San Francisco'], label='San Francisco', alpha=0.7) 14. *# Customize axes labels and legend* 15. ax.set\_xlabel('Date') 16. ax.set\_ylabel('Temperature (°C)') 17. ax.legend() 18. *# Customize the overall layout* 19. plt.title('Temperature Variation in Different Cities') 20. plt.show() |
| Data Logging and Processing:  **一. Experiment 1 - Exponentially distributed population**  1. calculate the mean and the standard deviation of the population.      2. Repeat the above process, but with a much larger sample size (𝑛=500)      3. Calculate the standard deviation of sample means for 500 samples    **二. Experiment 2 - Binomially distributed population**  1. Drawing 50 random samples of size 500 from a Binomial distribution.    2. Compute the mean of sample means.    3. compute the standard deviation of sample means.    **三. Experiment 3 - An Application of CLT in Investing/Trading**  Visualize both the returns and their distribution according to the diagrams below.     四. More visualization exercises 1. Use the [hourly weather dataset](https://www.kaggle.com/selfishgene/historical-hourly-weather-data/data) . We will use continuous distributions we have studied to approximate the observed distributions.    2. Temperature and 7-day Moving Average in Vancouver    3. Temperature Variation in Different Cities    4. Humidity, Pressure, and Temperature in Detroit    Experimental Results and Analysis:  1. Through this experiment, I deepened my understanding of the central limit theorem.  2. Each sample experiment has a relative error, which is relative to the model in which the event occurred. Through the central limit theorem, we can simplify many models into normal distribution models, which is convenient for us to observe and draw conclusions. For example, the common binomial distribution, exponential distribution and so on can use the central limit theorem, but the approximation of the central limit theorem is error, when the number of experimental samples is enough, the error will be smaller.  3. In addition, this experiment deepened my ability to process data. I was able to use python to process many data, such as the law of weather humidity, temperature, air pressure and other changes with the date, which could be simulated and visualized by using the central limit theorem. |
| 指导教师批阅意见：  成绩评定：  指导教师签字：  年 月 日 |
| 备注： |

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后10日内。