Assignment 2

Vorobiov Andrii

Imports

In [ ]:
```
import pandas as pd
import numpy as np
from numpy import float128
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
from scipy.special import beta
import math
import pprint
import statsmodels.stats.proportion as prop
from scipy.optimize import minimize, minimize_scalar
from tqdm import tqdm
```

Problem 4

Task 1

Part a

In [ ]:
```
df = pd.read_csv('EUR_UAH.csv')
df['Rate'] = df['Rate'].astype(float)

logn_rates = np.log(df['Rate'] / df['Rate'].shift(1))
logn_rates.dropna(inplace=True)

standartized_data = (df['Rate'] - np.mean(df['Rate'])) / np.std(df['Rate'])

# Perform Kolmogorov-Smirnov test
ks_statistic, p_value = stats.kstest(standartized_data, stats.norm.cdf)
print('For just standartized data:')
print('KS-statistic: ', ks_statistic, '\nP-value: ', p_value)

standartized_data = (logn_rates - np.mean(logn_rates)) / np.std(logn_rates)
ks_statistic, p_value = stats.kstest(standartized_data, stats.norm.cdf)
print('\nFor standartized ln data:')
print('KS-statistic: ', ks_statistic, '\nP-value: ', p_value)
```

```
For just standartized data:
KS-statistic:  0.06460412704778368
P-value:  0.0901592607568884

For standartized ln data:
KS-statistic:  0.17669387039169138
P-value:  1.9391752330629502e-10
```

H_0 hypothesis: F = N

H_1 hypothesis: F ≠ N

Due to the results, we have p-value of standartized returns = 0.09, which means we can't reject the null hypothesis if alpha < 0.09

Part b

In [ ]:
```
def L(params: tuple):
    mu, sigma, _df = params
    _df = math.ceil(_df * 1000)
    sum = 0
    for r in df['Rate']:
        _std = (r - mu) / sigma
        upper = float128(1 + (1/_df) * (_std ** 2))** (-(_df+1)/2)
        lower = beta(_df/2, .5) * math.sqrt(_df) * sigma

        log_likelihood = np.log(upper / lower)
        sum += log_likelihood
    return -sum # Because we need to maximize results, but we use minimize funct

init_params = (0.0001, 0.0001, 1)

result = minimize(L, init_params, bounds=((1e-5, None), (1e-5, None), ( .001, 1)
pprint.pprint({"mu": result.x[0], "sigma": result.x[1], "df": result.x[2] * 1000
```

```
{'mu': 37.90455642777226, 'sigma': 1.8437947710047553, 'df': 1000.0}
/tmp/ipykernel_9540/1138455107.py:10: RuntimeWarning: divide by zero encountered
in log
  log_likelihood = np.log(upper / lower)
/root/anaconda3/envs/py10/lib/python3.10/site-packages/scipy/optimize/_numdiff.p
y:576: RuntimeWarning: invalid value encountered in subtract
  df = fun(x) - f0
```

We can see that df is moving to the maximum bound number, that, in addition to previous task, gives as a conclusion that returns in this case follow Normal Distribution with mean ~ 37.9 and deviation ~ 1.8

Task 2

Part a

H_0 (null hypothesis): The expected age of a CEO is >= 60 years.
H_1 (alternative hypothesis): The expected age of a CEO is < 60 years.

In [ ]:
```
df = pd.read_excel("ceo.xlsx", header=0, )

ALPHA = 0.05
EXPECTED_VALUE = 60

def hr():
    for _ in range(50):
        print('-', end='')
    print()

def get_t_statistic(sample, expected_value):
    sample_mean = sample.mean()
    sample_std = sample.std(ddof=1)
    sample_size = len(sample)
```

```
    t_statistic = (sample_mean - expected_value) / (sample_std / np.sqrt(sample_
    return t_statistic


t_statistic = get_t_statistic(df['age'], EXPECTED_VALUE)

_, p_value = stats.ttest_1samp(df['age'], EXPECTED_VALUE, alternative='less')

df_degrees = len(df['age']) - 1

rejection_area = stats.t.ppf(ALPHA, df_degrees)

print(f"T-statistic: {t_statistic:.3f}")
print(f"Rejection area: {rejection_area:.3f}")
print(f"P-value: {p_value}")

# Decision based on the p-value and alpha level
hr()
if p_value < ALPHA:
    print("Reject the null hypothesis: There is evidence to suggest that the exp
else:
    print("Fail to reject the null hypothesis: There is not enough evidence to s
```

```
T-statistic: -10.965
Rejection area: -1.648
P-value: 3.103801632436528e-25
--------------------------------------------------
Reject the null hypothesis: There is evidence to suggest that the expected age of
a CEO is less than 60.
```

Part b

H_0 (null hypothesis): $\mu_1 = \mu_2$

H_1 (alternative hypothesis): $\mu_1 \neq \mu_2$

In [ ]:

```
younger = df[df['age'] < EXPECTED_VALUE]['salary']
older = df[df['age'] >= EXPECTED_VALUE]['salary']

n1 = len(younger)
n2 = len(older)
s1 = younger.std(ddof=1)
s2 = older.std(ddof=1)
x1 = younger.mean()
x2 = older.mean()

_, p_value = stats.ttest_ind(younger, older, equal_var=False)

v = (x1 - x2) / (math.sqrt((s1**2 / n1) + (s2**2 / n2)))
rejection_area = (stats.t.ppf(ALPHA/2, n1-1), stats.t.ppf(1-ALPHA/2, n1-1))
print(f"T-statistic: {v:.3f}")
print(f"Rejection area: ({rejection_area[0]:.3f}; {rejection_area[1]:.3f})")
print(f"P-value: {p_value:.3f}")
hr()

if p_value < ALPHA:
    print("Reject the null hypothesis: \n\tThere is enough evidence to suggest t
else:
    print("Fail to reject the null hypothesis: \n\tThere is enough evidence to s
```

```
T-statistic: -1.619
Rejection area: (-1.968; 1.968)
P-value: 0.107
--------------------------------------------------
Fail to reject the null hypothesis:
        There is enough evidence to suggest that the expected salary of younger C
EOs isn't significantly different or equal to/than the expected salary of older C
EOs.
```

Part c

H_0 (null hypothesis): $\rho = 0$

H_1 (alternative hypothesis): $\rho \neq 0$

In [ ]:
```
sales = df['sales']
profits = df['profits']

_, p_value = stats.pearsonr(sales, profits)
r = sum((sales - sales.mean()) * (profits - profits.mean())) / (math.sqrt(sum((s
v = math.sqrt(len(sales) - 2) * (r/math.sqrt(1 - r**2))

rejection_area = (stats.norm.ppf(ALPHA/2), stats.norm.ppf(1-ALPHA/2)) # using nc

print(f"Pearson correlation coefficient: {r:.3f}\np-value: {p_value}")
print(f"Statistic: {v:.3f}")
print(f"Rejection area: ({rejection_area[0]:.2f}; {rejection_area[1]:.2f})")

hr()

if p_value < ALPHA:
    print("Reject the null hypothesis: \n  There is enough evidence to suggest t
else:
    print("Fail to reject the null hypothesis: \n  There is enough evidence to s
```

```
Pearson correlation coefficient: 0.726
p-value: 2.3209381272863423e-74
Statistic: 22.268
Rejection area: (-1.96; 1.96)
--------------------------------------------------
Reject the null hypothesis:
  There is enough evidence to suggest that the correlation between sales and prof
its is significant.
```

Part d

H_0 (null hypothesis): p >= 50% (fraction of CEOs older than 60 higher than 50%)

H_1 (alternative hypothesis): p < 50% (fraction of CEOs older than 60 lower than 50%)

In [ ]:
```
EXPECTED_P = .5

proportion = sum(df['age'] > 60) / len(df)
print(f"Proportion of younger: {proportion}")

_, p_value = prop.proportions_ztest(sum(df['age'] > 60), len(df), EXPECTED_P, al
z = (proportion - EXPECTED_P) / math.sqrt((proportion * (1 - proportion)) / len(
rejection_area =  stats.norm.ppf(ALPHA)

print(f"p-value: {p_value}")
```

```
print(f"Statistic: {z:.3f}")
print(f"Rejection area: {rejection_area:.2f}")

hr()

if p_value < ALPHA:
    print("Reject the null hypothesis: \n  There is enough evidence to suggest t
else:
    print("Fail to reject the null hypothesis: \n  There is enough evidence to s
```

```
Proportion of younger: 0.2684563758389262
p-value: 1.1380583261240903e-28
Statistic: -11.047
Rejection area: -1.64
--------------------------------------------------
Reject the null hypothesis:
  There is enough evidence to suggest that the proportion of older CEOs is signif
icantly less than 50%.
```

Part e

H_0 hypothesis: F = N

H_1 hypothesis: F ≠ N

In [ ]:
```
data = df['salary']

mean = data.mean()
std = data.std()

d, p_value = stats.kstest(data, 'norm', args=(mean, std))

print(f"K-S statistic: {d}, p-value: {p_value}")
hr()

if p_value < ALPHA:
    print("Reject the null hypothesis: The data is not normally distributed.")
else:
    print("Fail to reject the null hypothesis: No evidence to suggest the data i
```

```
K-S statistic: 0.18615180994924452, p-value: 4.93926502382922e-14
--------------------------------------------------
Reject the null hypothesis: The data is not normally distributed.
```

Task 3

Part a

H_0 hypothesis: μ = 500

H_1 hypothesis: μ ≠ 500

with α = 0.04

- Also we need to estimate σ^2. But we can calculate sample variance(s^2), which is an unbiased estimator of the σ^2

In [ ]:
```
MEAN = 500
VAR = 50
SIGMA = math.sqrt(VAR)
```

```
N = 100
ALPHA = .04

sample = np.random.normal(MEAN, SIGMA, N)

sample_mean = np.mean(sample)
sample_std = np.std(sample, ddof=1)
sample_var = sample_std**2
sample_size = len(sample)

v = (sample_mean - MEAN) / sample_std * math.sqrt(sample_size)
rejection_areas = (stats.norm.ppf(ALPHA/2), stats.norm.ppf(1-ALPHA/2))

print(f"Sample mean: {sample_mean:.3f}")
print(f"Sample variance: {sample_var:.3f}")
print(f"Statistic: {v:.3f}")
print(f"Rejection areas: (-Inf, {rejection_areas[0]:.3f}) U ({rejection_areas[1]
hr()
if v > rejection_areas[0] and v < rejection_areas[1]:
    print("Fail to reject the null hypothesis: \n\tThe sample mean is not signif
else:
    print("Reject the null hypothesis: \n\tThe sample mean is significantly diff
```

```
Sample mean: 500.151
Sample variance: 53.439
Statistic: 0.207
Rejection areas: (-Inf, -2.054) U (2.054, Inf)
--------------------------------------------------
Fail to reject the null hypothesis:
        The sample mean is not significantly different from the 500
```

Part b

In [ ]:

```
M = 1000
results = []

for i in range(M):
    np.random.seed(i)
    sample = np.random.normal(MEAN, SIGMA, N)

    sample_mean = np.mean(sample)
    sample_std = np.std(sample, ddof=1)
    sample_size = len(sample)

    v = (sample_mean - MEAN) / sample_std * math.sqrt(sample_size)
    rejection_areas = (stats.norm.ppf(ALPHA/2), stats.norm.ppf(1-ALPHA/2))
    p = not (v > rejection_areas[0] and v < rejection_areas[1])

    results.append(p)
np.random.seed(None)
ecl = 1/M * sum(results)
print(f"Empirical confidence level: {ecl:.3f}")
print(f"Confidence level: {ALPHA:.3f}")
```

```
Empirical confidence level: 0.045
Confidence level: 0.040
```

I expected the difference to be large, because of low sample size, which can't fully describe Normal Distribution without enough samples.

Although we have "The rule of thumb", so I believe it should be at least near the correct answer

Task 4

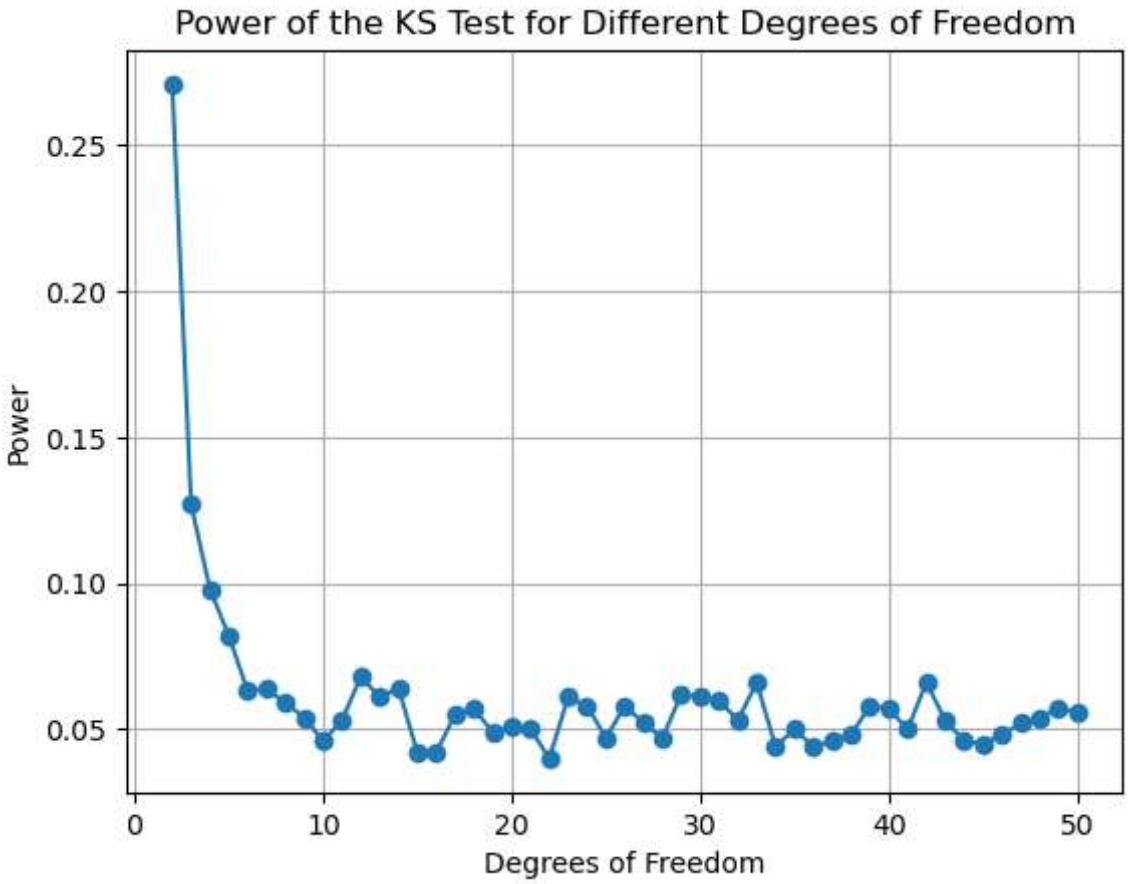
H_0 hypothesis: $t_{2..50} = N$

H_1 hypothesis: $t_{2..50} \neq N$

Though H_0 is declared as normality, we expect it to be rejected, and the H_1 hypothesis to be accepted.

In [ ]:
```
M = 1000
n = 100
dfs = range(2, 51)
alpha = 0.05

rejections = {df: 0 for df in dfs}

for df in tqdm(dfs):
    for i in range(M):
        sample = np.random.standard_t(df, size=n)

        _, p_value = stats.kstest(sample, 'norm')

        if p_value < alpha:
            rejections[df] += 1

power = [rejections[df] / M for df in dfs]

plt.plot(dfs, power, marker='o')
plt.title('Power of the KS Test for Different Degrees of Freedom')
plt.xlabel('Degrees of Freedom')
plt.ylabel('Power')
plt.grid(True)
plt.show()
```

```
100%|██████████| 49/49 [01:21<00:00,  1.66s/it]
```



The provided plot shows the power of the KS test for different degrees of freedom. Initially, the test has high power for low degrees of freedom, indicating good performance in detecting non-normality. As the degrees of freedom increase, the power drops sharply and then levels off, showing that the test is less effective when the t-distribution resembles a normal distribution (roughly at df > 10).
The plot suggests that the KS test may not be the best tool for detecting small deviations from normality in samples that come from distributions similar to the normal distribution. This is relevant in practical applications where the sample distribution may only slightly deviate from normality.