es21btech11025-assign2

January 22, 2024

Ranveer Sahu (ES21BTECH11025) #Assignment - 2

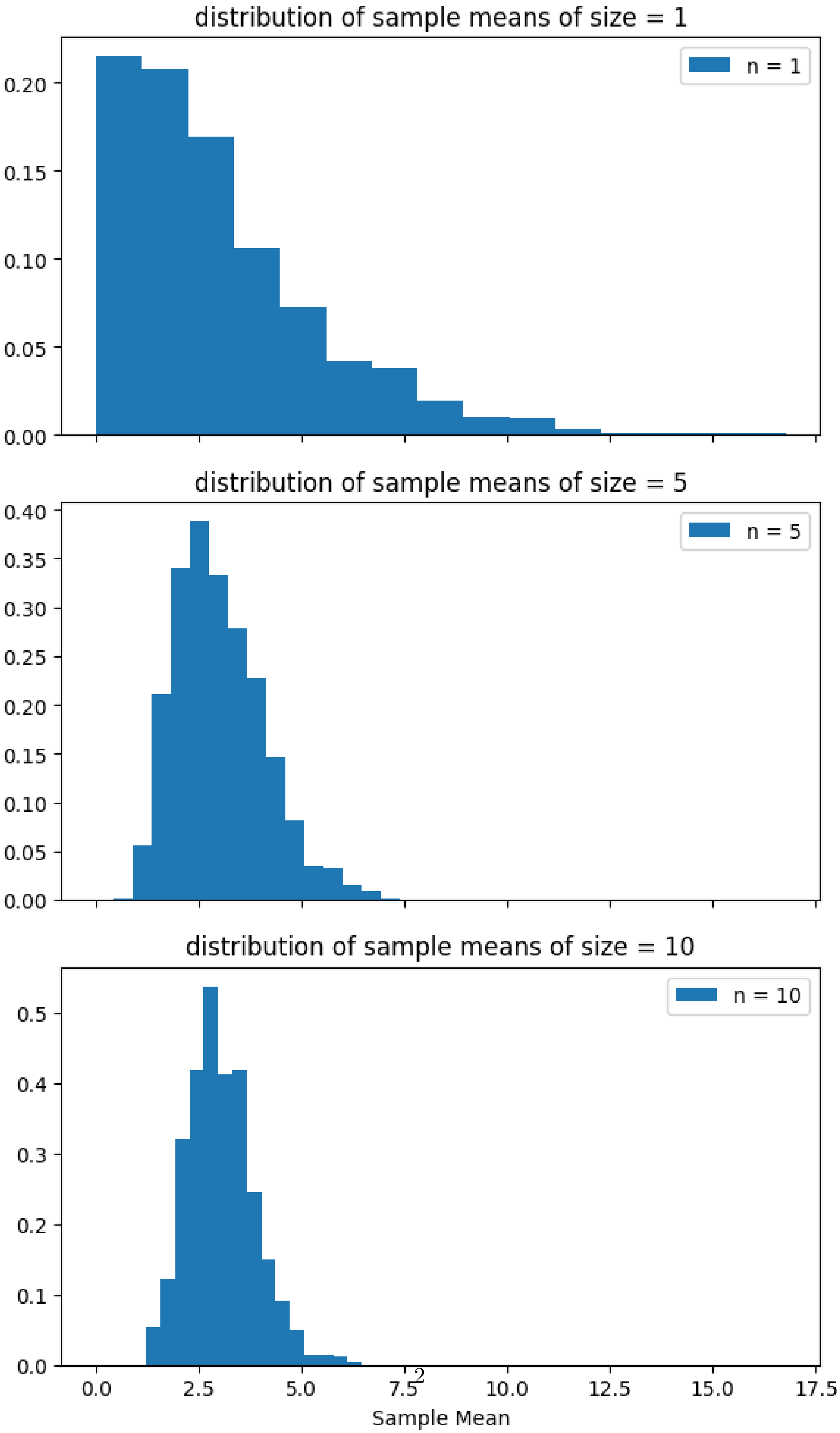
[37]:

```python
import numpy as np
from scipy.stats import pearsonr, t
import matplotlib.pyplot as plt
import pandas as pd


np.random.seed(42)
```

Q1 In the class, we demonstrated the Central Limit Theorem for a sample drawn from a uniform distribution. Reproduce a similar plot for a sample drawn the from chi-square distribution with degrees of freedom equal to 3, for samples drawn once, 5 times, and 10 times. Either plot all of these on one multipanel figure similar to AstroML figure 3.20. (20 points) (Hint: look up numpy.random.chisquare and show the distribution of x from 0 to 10)

[67]:

```python
degrees = 3
nums = [1, 5, 10]
fig, axs = plt.subplots(len(nums), 1, figsize=(6, 10), sharex=True)


for i, n in enumerate(nums):
    smpl = np.random.chisquare(degrees, size=(n, 1000))
    smpl_mns = np.mean(smpl, axis=0)

    axs[i].hist(smpl_mns, bins=15, density=True, alpha=1, label=f'n = {n}')
    axs[i].set_title(f'distribution of sample means of size = {n}')
    axs[i].legend()


plt.xlabel('Sample Mean')
plt.tight_layout()
plt.show()
```



distribution of sample means of size = 1

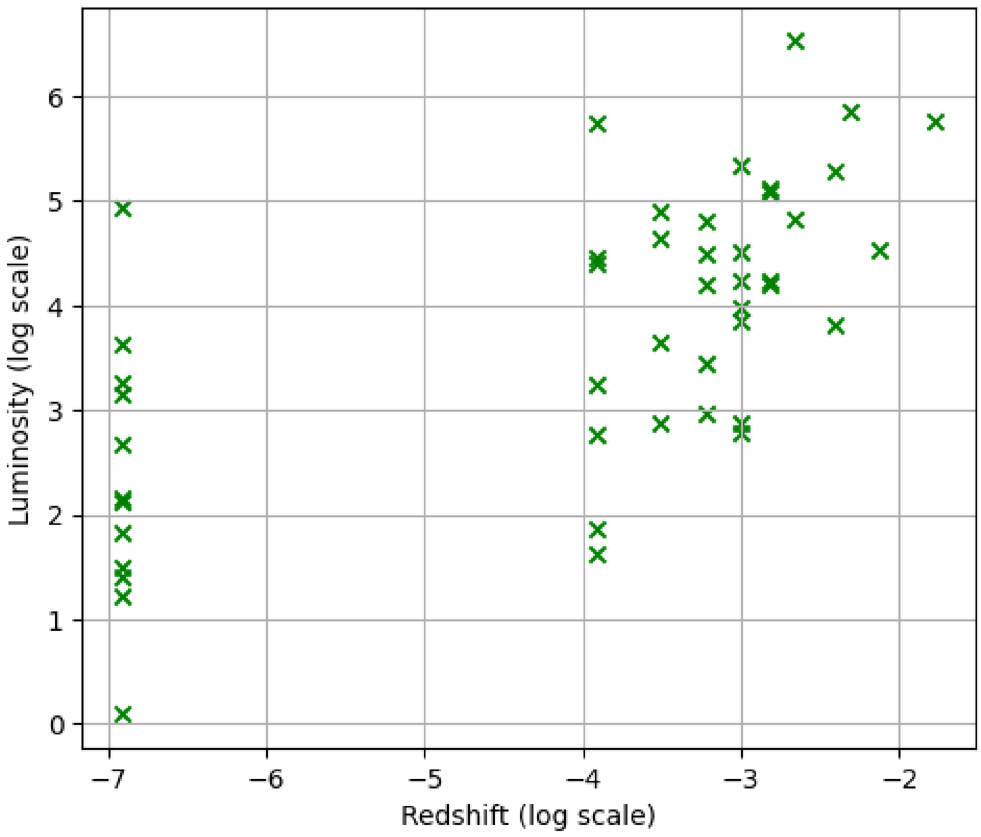
distribution of sample means of size = 5

distribution of sample means of size = 10

Q3 The luminosity and redshift of galaxy clusters from XMM-BCS survey (details available at arXiv:1512.01244) can be downloaded http://www.iith.ac.in/~shantanud/test.dat. Plot the luminosity as a function of redshift on a log-log scale. By eye, do you think the datasets are correlated? Calculate the Spearman, Pearson and Kendall-tau correlation coefficients and the p-value for the null hypothesis.

[52]:

```python
from scipy.stats import spearmanr, pearsonr, kendalltau

file_path = 'q2_data.txt'
df = pd.read_csv(file_path, sep=' ')
plt.figure(figsize=(6, 5))
plt.scatter(np.log(df['L2']), np.log(df['#L1']), marker='x', color='g')
plt.xlabel('Redshift (log scale)')
plt.ylabel('Luminosity (log scale)')
plt.grid(True)
plt.show()


sp_corr, sp_pvalue = spearmanr(df['L2'], df['#L1'])
ps_corr, ps_pvalue = pearsonr(df['L2'], df['#L1'])
kdl_corr, kdl_pvalue = kendalltau(df['L2'], df['#L1'])


print(f"Spearman correlation coeff: {sp_corr}")
print(f"p_value for Spearman correlation: {sp_pvalue}")
print(f"Pearson correlation coeff: {ps_corr}")
print(f"p_value for Pearson correlation: {ps_pvalue}")
print(f"Kendall-tau correlation coeff: {kdl_corr}")
print(f"p_value for Kendall-tau correlation: {kdl_pvalue}")
```

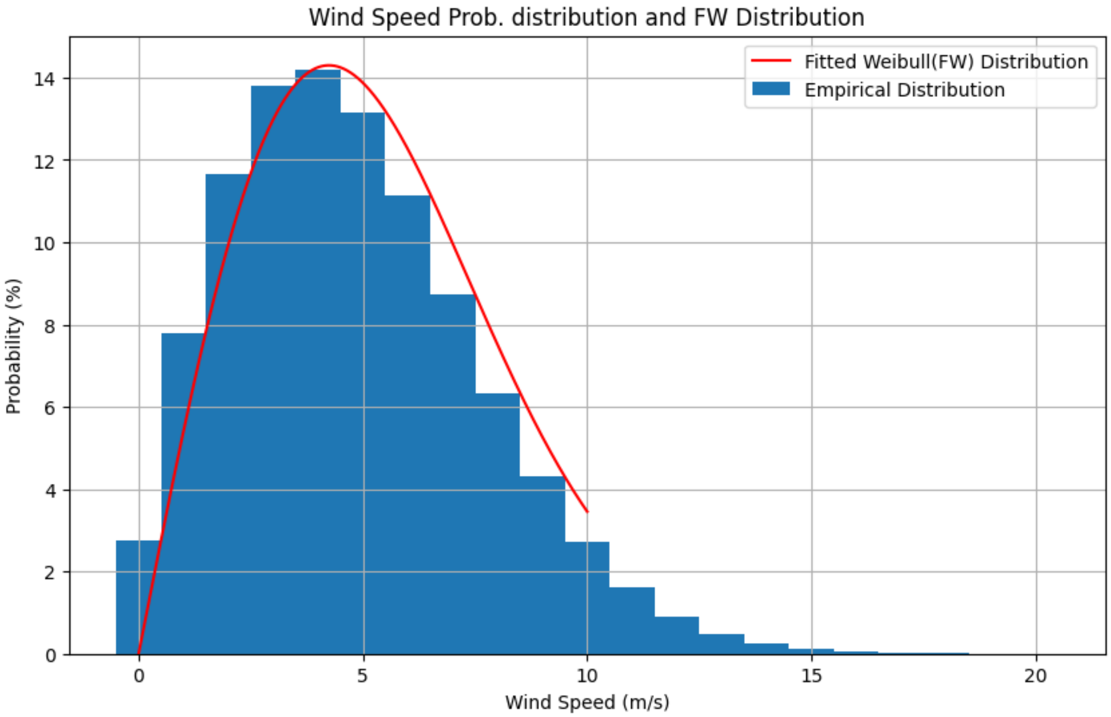


```
Spearman correlation coeff: 0.6596325957535454
p_value for Spearman correlation: 6.166489759081011e-07
Pearson correlation coeff: 0.5144497852670242
p_value for Pearson correlation: 0.0002546471657612427
Kendall-tau correlation coeff: 0.5029584682704178
p_value for Kendall-tau correlation: 2.9696862274734036e-06
```

Q3 Wind speed data from the Swiss Wind Power data website can be found at http://wind-data.ch/tools/ weibull.php. Using the data provided on the website, plot the probability distribution and overlay the best-fit Weibull distribution (with the parameters shown on the website). (20 points) (Hint : A on the website is same as , which was used in class to parameterize the Weibull distribution.)

[66]:
```
from scipy.stats import weibull_min

l_param = 6
k_param = 2

A = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,␣
  ↪19, 20])
```

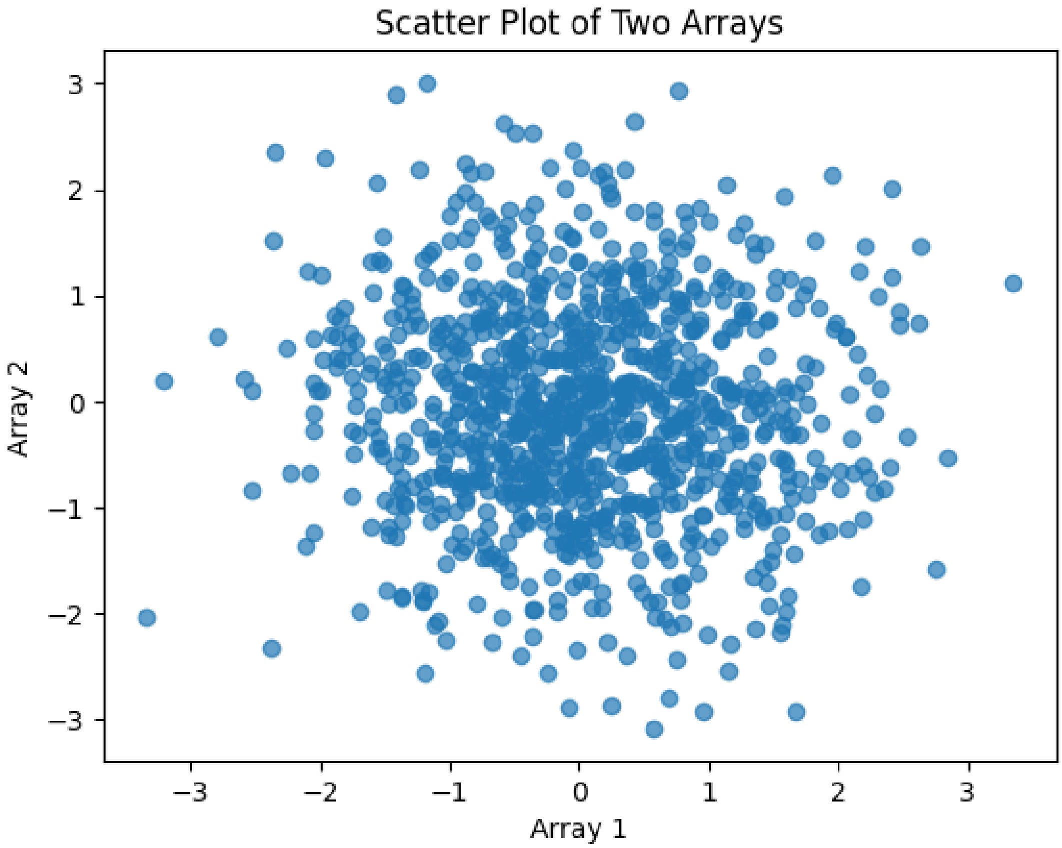
```python
B = np.array([2.75, 7.80, 11.64, 13.79, 14.20, 13.15, 11.14, 8.72, 6.34, 4.30, 2.73, 1.62, 0.91, 0.48, 0.24, 0.11, 0.05, 0.02, 0.01, 0.00, 0.00])
# A = array of speed date and B =  array of probability
cumutative_prob = np.cumsum(B) / 100.0

# Fit a Weibull distribution to the data
def weibull_distribution(x, shape, scale):
    return weibull_min.pdf(x, shape, loc=0, scale=scale)

iguess = [2, 5]
x = np.linspace(0, 10, 1000)
pdf_values = weibull_min.pdf(x, k_param, loc=0, scale=l_param)
plt.figure(figsize=(10, 6))
plt.bar(A, B, width=1, align='center', alpha=0.9, label='Empirical Distribution')
plt.plot(x, pdf_values * 100, 'r-', label='Fitted Weibull(FW) Distribution')
plt.title('Wind Speed Prob. distribution and FW Distribution')
plt.xlabel('Wind Speed (m/s)')
plt.ylabel('Probability (%)')

x = np.linspace(0, 10, 1000)
pdf_values = weibull_min.pdf(x, k_param, loc=0, scale=l_param)
plt.legend()
plt.grid(True)
plt.show()
```



Q4 Generate two arrays of size 1000 drawn from a Gaussian distribution of mean of zero and standard deviation of one. Calculate Pearson correlation coefficient and its p−value using scipy module. Also check if the p− value agrees with that calculated using the Student-t distibution.

[51]:

```
x  = np.random.normal(0, 1, 1000)
y = np.random.normal(0, 1, 1000)
corr_coeff, p_value = pearsonr(x, y)

# Point 1: finding degrees of freedom for Student-t distribution
degree = len(x) - 2

# point 2 : Calculate p-value using Student-t distribution
t_dist = t(degree)
p_value_t_dist = 2 * (1 - t_dist.cdf(np.abs(corr_coeff)))


plt.scatter(x, y, alpha=0.7)
plt.title("Scatter Plot of Two Arrays")
plt.xlabel("Array 1")
plt.ylabel("Array 2")
plt.show()


print("Pearson Correlation Coeff.:", corr_coeff)
print("P_value for Normal distribution :", p_value)
print("P-value for Student-t Distribution:", p_value_t_dist)


if np.isclose(p_value, p_value_t_dist, rtol=1e-10):
    print("agree!")
else:
    print("do not agree.")
```



```
Pearson Correlation Coeff.: -0.04658133076413823
P_value for Normal distribution : 0.14102419157936402
P-value for Student-t Distribution: 0.962856230934058
do not agree.
```