dam-analysis

February 26, 2024

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[51]:
```
# import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from IPython.display import display
import plotly.graph_objects as go
import plotly.express as px
```

[59]:
```
# Load dataset
data = pd.read_csv('/content/drive/MyDrive/Data/mam_saba_naz/sukkur.csv')
data = data.drop(columns=['PEAK DISCHARGE', 'Volume in Cuseecs'])
data = data.rename(columns={"PD in m3/s":"Peak Discharge", "V in m3/s":
 ↪"Volume"})
data.head()
```

[59]:

	YEAR	Duration	Peak Discharge	Volume
0	1977	102	12813.71039	2.849925e+05
1	1978	101	27125.25911	1.014680e+06
2	1979	89	12541.20428	4.299144e+05
3	1980	87	14581.91801	2.773503e+05
4	1981	130	14440.01299	4.328008e+05

1 Exploratory Data Analysis

[ ]:
```
data_description = data[["Volume (m3/s)","Peak Discharge (m3/s)", "Duration"]].
 ↪describe()
print(data_description)
```

```
       Volume (m3/s)  Peak Discharge (m3/s)    Duration
count   3.600000e+01              36.000000   36.000000
mean    3.470409e+05           12972.927726   97.361111
std     2.850717e+05            8313.143418   18.425375
min     1.515986e+04            1755.413860   55.000000
25%     1.345482e+05            5539.826823   86.250000
50%     2.811714e+05           12274.580400   99.000000
75%     4.525969e+05           18296.956437  106.000000
max     1.016445e+06           29888.720140  142.000000
```

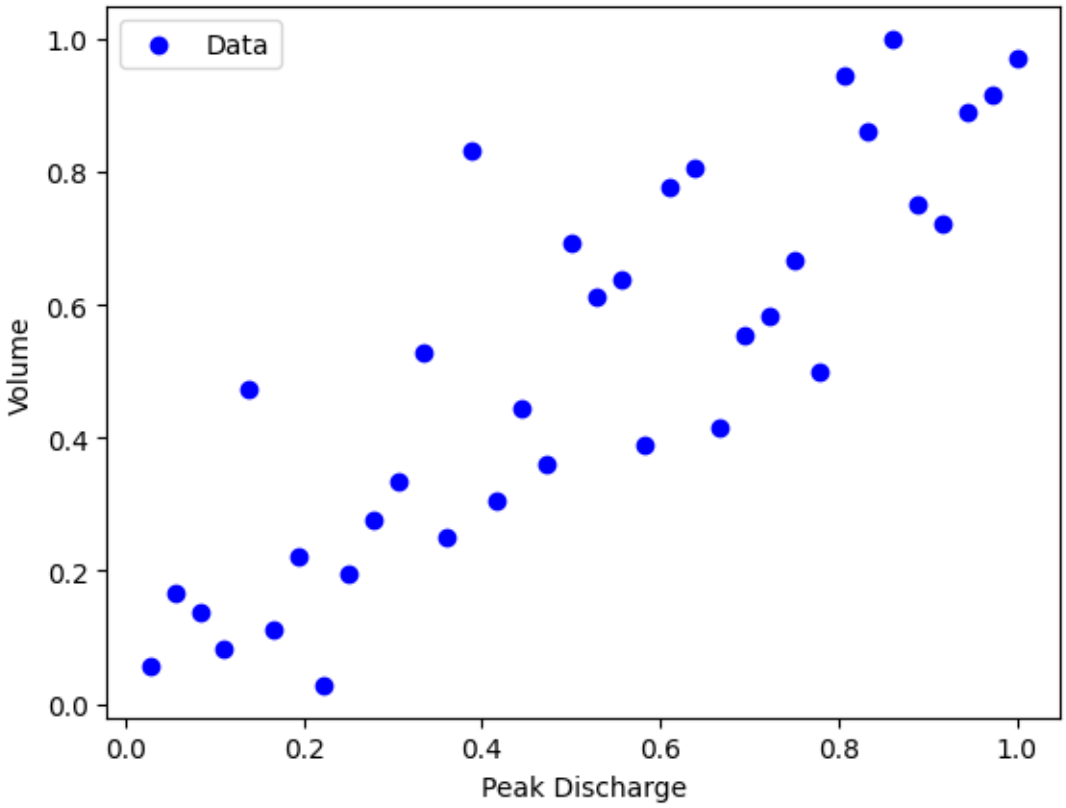
[ ]:
```
print("SKEWNESS")
print(data[["Volume (m3/s)","Peak Discharge (m3/s)"]].skew())
print("****************************************")
print("KURTOSIS")
print(data[["Volume (m3/s)","Peak Discharge (m3/s)"]].kurtosis())
print("****************************************")

print("MEAN")
print(data[["Volume (m3/s)","Peak Discharge (m3/s)"]].mean())
print("****************************************")

print("MEDIAN")
print(data[["Volume (m3/s)","Peak Discharge (m3/s)"]].median())

print("****************************************")
```

```
SKEWNESS
Volume (m3/s)            0.934677
Peak Discharge (m3/s)    0.550155
dtype: float64
****************************************
KURTOSIS
Volume (m3/s)            0.034333
Peak Discharge (m3/s)   -0.678294
dtype: float64
****************************************
MEAN
Volume (m3/s)            347040.887786
Peak Discharge (m3/s)     12972.927726
dtype: float64
****************************************
MEDIAN
Volume (m3/s)            281171.3849
Peak Discharge (m3/s)     12274.5804
dtype: float64
****************************************
```

2 Correlation

[ ]:
```
volume = data['Volume (m3/s)']
duration = data['Peak Discharge (m3/s)']

# Calculate Pearson correlation
pearson_corr = np.corrcoef(volume, duration)[0, 1]

# Calculate Kendall correlation
kendall_corr = volume.corr(duration, method='kendall')

# Calculate Spearman correlation
spearman_corr = volume.corr(duration, method='spearman')

print("Pearson Correlation:", pearson_corr)
print("Kendall Correlation:", kendall_corr)
print("Spearman Correlation:", spearman_corr)

# # Plot the data
# plt.scatter(volume, duration, color='blue', label='Data')
# plt.xlabel('Peak Discharge')
# plt.ylabel('Volume')
# plt.legend()
# plt.show()

# Plot the data after rank transformation (Kendall Plot)
u = volume.rank(pct=True)
v = duration.rank(pct=True)
plt.scatter(u, v, color='blue', label='Data')
plt.xlabel('Peak Discharge')
plt.ylabel('Volume')
plt.legend()
plt.show()
```

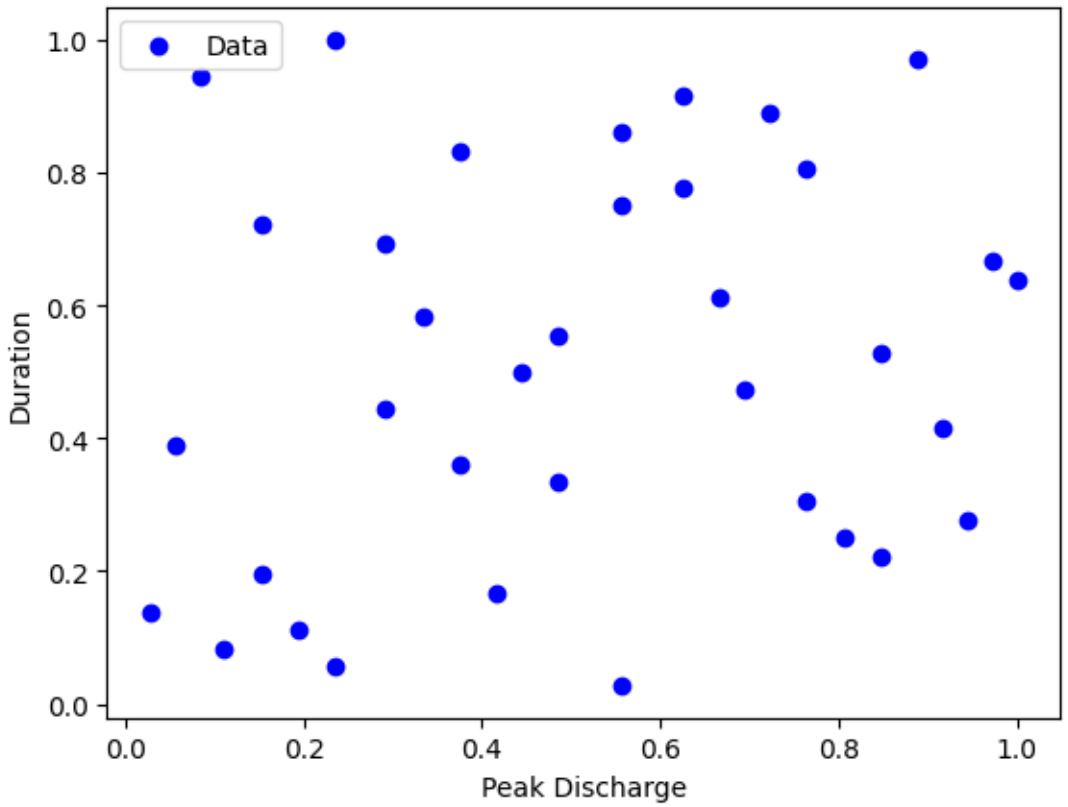
```
Pearson Correlation: 0.848810575771854
Kendall Correlation: 0.6793650793650794
Spearman Correlation: 0.8532818532818532
```



[ ]:
```
volume = data['Duration']
duration = data['Peak Discharge (m3/s)']

# Calculate Pearson correlation
pearson_corr = np.corrcoef(volume, duration)[0, 1]

# Calculate Kendall correlation
kendall_corr = volume.corr(duration, method='kendall')

# Calculate Spearman correlation
spearman_corr = volume.corr(duration, method='spearman')

print("Pearson Correlation:", pearson_corr)
print("Kendall Correlation:", kendall_corr)
print("Spearman Correlation:", spearman_corr)

# Plot the data
# plt.scatter(volume, duration, color='blue', label='Data')
# plt.xlabel('Peak Discharge')
# plt.ylabel('Volume')
# plt.legend()
```

```
# plt.show()

u = volume.rank(pct=True)
v = duration.rank(pct=True)
plt.scatter(u, v, color='blue', label='Data')
plt.xlabel('Peak Discharge')
plt.ylabel('Duration')
plt.legend()
plt.show()
```

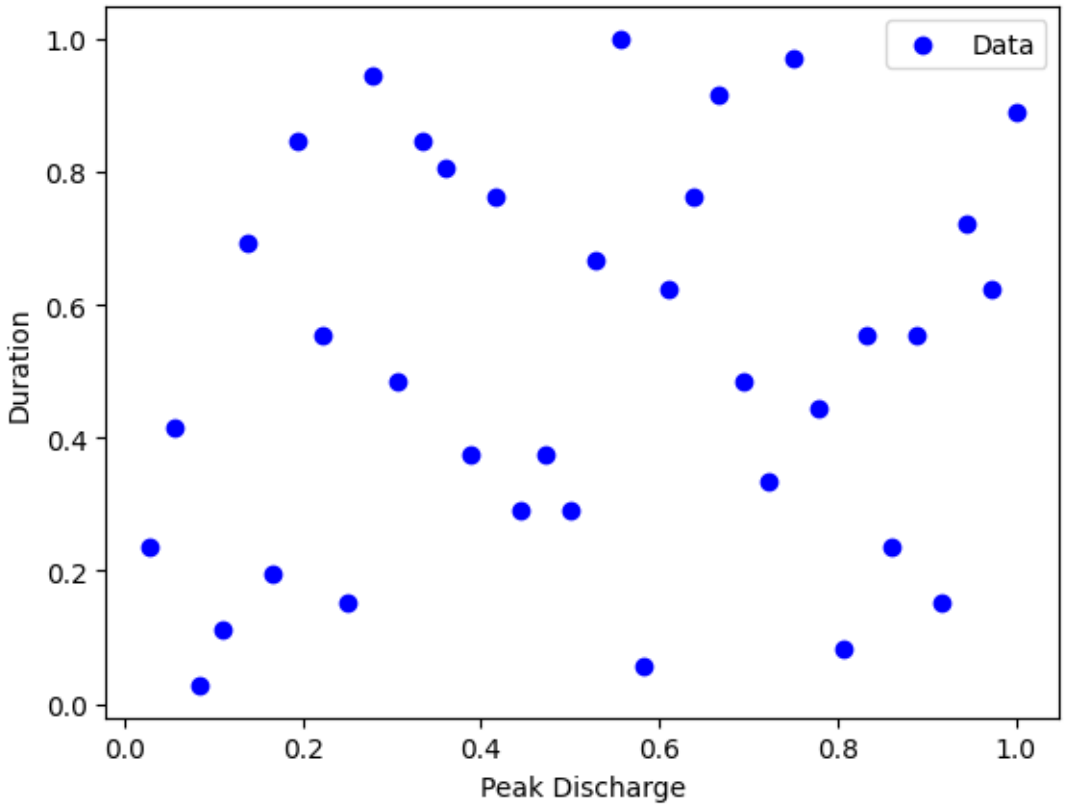
```
Pearson Correlation: 0.09534496810400707
Kendall Correlation: 0.11369536218425486
Spearman Correlation: 0.18147867345941177
```



[ ]:
```
volume = data['Volume (m3/s)']
duration = data['Duration']

# Calculate Pearson correlation
pearson_corr = np.corrcoef(volume, duration)[0, 1]

# Calculate Kendall correlation
```

```
kendall_corr = volume.corr(duration, method='kendall')

# Calculate Spearman correlation
spearman_corr = volume.corr(duration, method='spearman')

print("Pearson Correlation:", pearson_corr)
print("Kendall Correlation:", kendall_corr)
print("Spearman Correlation:", spearman_corr)

# # Plot the data
# plt.scatter(volume, duration, color='blue', label='Data')
# plt.xlabel('Peak Discharge')
# plt.ylabel('Volume')
# plt.legend()
# plt.show()

# Plot the data after rank transformation (Kendall Plot)
u = volume.rank(pct=True) # Percentile Rank = (Total number of values - 1) /␣
 ↪(total number of values - number of tied values) × 100
v = duration.rank(pct=True)
plt.scatter(u, v, color='blue', label='Data')
plt.xlabel('Peak Discharge')
plt.ylabel('Duration')
plt.legend()
plt.show()
```

```
Pearson Correlation: 0.1080893406176211
Kendall Correlation: 0.11049267592554345
Spearman Correlation: 0.17143230260786166
```



3 Distribution Fitting

[44]:
```
!pip install fitter
```

```
Collecting fitter
  Downloading fitter-1.7.0-py3-none-any.whl (26 kB)
Requirement already satisfied: click<9.0.0,>=8.1.6 in
/usr/local/lib/python3.10/dist-packages (from fitter) (8.1.7)
Requirement already satisfied: joblib<2.0.0,>=1.3.1 in
/usr/local/lib/python3.10/dist-packages (from fitter) (1.3.2)
Collecting loguru<0.8.0,>=0.7.2 (from fitter)
  Downloading loguru-0.7.2-py3-none-any.whl (62 kB)
                           62.5/62.5 kB
2.9 MB/s eta 0:00:00
Collecting matplotlib<4.0.0,>=3.7.2 (from fitter)
  Downloading
matplotlib-3.8.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(11.6 MB)
                           11.6/11.6 MB
19.8 MB/s eta 0:00:00
```

```
Requirement already satisfied: numpy<2.0.0,>=1.20.0 in
/usr/local/lib/python3.10/dist-packages (from fitter) (1.25.2)
Requirement already satisfied: pandas<3.0.0,>=0.23.4 in
/usr/local/lib/python3.10/dist-packages (from fitter) (1.5.3)
Collecting rich-click<2.0.0,>=1.7.2 (from fitter)
  Downloading rich_click-1.7.3-py3-none-any.whl (32 kB)
Requirement already satisfied: scipy<2.0.0,>=0.18.0 in
/usr/local/lib/python3.10/dist-packages (from fitter) (1.11.4)
Requirement already satisfied: tqdm<5.0.0,>=4.65.1 in
/usr/local/lib/python3.10/dist-packages (from fitter) (4.66.2)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib<4.0.0,>=3.7.2->fitter)
(1.2.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-
packages (from matplotlib<4.0.0,>=3.7.2->fitter) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib<4.0.0,>=3.7.2->fitter)
(4.49.0)
Requirement already satisfied: kiwisolver>=1.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib<4.0.0,>=3.7.2->fitter)
(1.4.5)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib<4.0.0,>=3.7.2->fitter)
(23.2)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.10/dist-
packages (from matplotlib<4.0.0,>=3.7.2->fitter) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib<4.0.0,>=3.7.2->fitter)
(3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/dist-packages (from matplotlib<4.0.0,>=3.7.2->fitter)
(2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas<3.0.0,>=0.23.4->fitter) (2023.4)
Requirement already satisfied: rich>=10.7.0 in /usr/local/lib/python3.10/dist-
packages (from rich-click<2.0.0,>=1.7.2->fitter) (13.7.0)
Requirement already satisfied: typing-extensions in
/usr/local/lib/python3.10/dist-packages (from rich-click<2.0.0,>=1.7.2->fitter)
(4.9.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.7->matplotlib<4.0.0,>=3.7.2->fitter) (1.16.0)
Requirement already satisfied: markdown-it-py>=2.2.0 in
/usr/local/lib/python3.10/dist-packages (from rich>=10.7.0->rich-
click<2.0.0,>=1.7.2->fitter) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
/usr/local/lib/python3.10/dist-packages (from rich>=10.7.0->rich-
click<2.0.0,>=1.7.2->fitter) (2.16.1)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-
```

```
packages (from markdown-it-py>=2.2.0->rich>=10.7.0->rich-
click<2.0.0,>=1.7.2->fitter) (0.1.2)
Installing collected packages: loguru, matplotlib, rich-click, fitter
  Attempting uninstall: matplotlib
    Found existing installation: matplotlib 3.7.1
    Uninstalling matplotlib-3.7.1:
      Successfully uninstalled matplotlib-3.7.1
Successfully installed fitter-1.7.0 loguru-0.7.2 matplotlib-3.8.3 rich-
click-1.7.3
```

[45]:
```
from fitter import Fitter, get_common_distributions, get_distributions
from scipy.stats import *
```

[46]:
```
dist_list = ['alpha', 'anglit', 'arcsine', 'argus', 'beta', 'betaprime',␣
 ↪'bradford', 'burr', 'burr12',
             'cauchy', 'chi', 'chi2', 'cosine', 'crystalball', 'dgamma',␣
 ↪'dweibull', 'erlang', 'expon',
             'exponnorm', 'exponpow', 'exponweib', 'f', 'fatiguelife', 'fisk',␣
 ↪'foldcauchy', 'foldnorm',
             'frechet_l', 'frechet_r', 'gamma', 'gausshyper', 'genexpon',␣
 ↪'genextreme', 'gengamma',
             'genhalflogistic', 'geninvgauss', 'genlogistic', 'gennorm',␣
 ↪'genpareto', 'gilbrat', 'gompertz',
             'gumbel_l', 'gumbel_r', 'halfcauchy', 'halfgennorm',␣
 ↪'halflogistic', 'halfnorm', 'hypsecant',
             'invgamma', 'invgauss', 'invweibull', 'johnsonsb', 'johnsonsu',␣
 ↪'kappa3', 'kappa4', 'ksone',
             'kstwo', 'kstwobign', 'laplace', 'levy', 'levy_l', 'levy_stable',␣
 ↪'loggamma', 'logistic',
             'loglaplace', 'lognorm', 'loguniform', 'lomax', 'maxwell',␣
 ↪'mielke', 'moyal', 'nakagami',
             'ncf', 'nct', 'ncx2', 'norm', 'norminvgauss', 'pareto',␣
 ↪'pearson3', 'powerlaw', 'powerlognorm',
             'powernorm', 'rayleigh', 'rdist', 'recipinvgauss', 'reciprocal',␣
 ↪'rice', 'semicircular',
             'skewnorm', 't', 'triang', 'truncexpon', 'truncnorm',␣
 ↪'tukeylambda', 'uniform', 'vonmises',
             'vonmises_line', 'wald', 'weibull_max', 'weibull_min',␣
 ↪'wrapcauchy']
```

[ ]:
```
import scipy.stats as stats
from fitter import Fitter

def plot_and_calculate_stats(data, dist_name, params, param_names):
    x = np.linspace(min(data), max(data), 100)
    dist_obj = getattr(stats, dist_name)
```

```
    pdf_values = dist_obj.pdf(x, *params)

    # Plotting
    plt.hist(data, bins=30, density=True, alpha=0.6, color='g', label='Data␣
 ↪Histogram')
    plt.plot(x, pdf_values, 'r-', lw=2, label=f'Fitted {dist_name}')
    plt.title(f'Fitted {dist_name} Distribution')
    plt.xlabel('Data Values')
    plt.ylabel('Density')
    plt.legend()
    plt.show()
    # Calculating statistics
    mean, var = dist_obj.stats(*params, moments='mv')
    print(f"{dist_name} distribution mean: {mean}, variance: {var}")
    # Printing parameters in a dictionary format
    param_dict = dict(zip(param_names, params))
    print(f"Fitted parameters for {dist_name}: {param_dict}")
    # Generate random values from the fitted distribution (example: 1000 values)
    simulated_values = dist_obj.rvs(*params, size=1000)
    print(f"First 5 simulated values from {dist_name}: {simulated_values[:5]}")

col = 'Peak Discharge'
data = data[col].dropna().values

task = input("Type 'common' to fit common distributions, or 'selected' for␣
 ↪specific distributions: ").lower()

if task == 'common':
    f = Fitter(data, timeout=100)
    f.fit()
    best_dist_name = list(f.get_best(method='sumsquare_error').keys())[0]
    best_dist_params = f.fitted_param[best_dist_name]
    plot_and_calculate_stats(data, best_dist_name, best_dist_params, ['loc',␣
 ↪'scale'])

elif task == 'selected':
    dist_parm_dict = {
    "alpha":            ["a", "loc", "scale"],
    "anglit":           ["loc", "scale"],
    "arcsine":          ["loc", "scale"],
    "argus":            ["chi", "loc", "scale"],
    "beta":             ["a", "b", "loc", "scale"],
    "betaprime":        ["a", "b", "loc", "scale"],
    "bradford":         ["c", "loc", "scale"],
    "burr":             ["c", "d", "loc", "scale"],
    "burr12":           ["c", "d", "loc", "scale"],
    "cauchy":           ["loc", "scale"],
```

```
    "chi":               ["df", "loc", "scale"],
    "chi2":              ["df", "loc", "scale"],
    "cosine":            ["loc", "scale"],
    "crystalball":       ["beta", "m", "loc", "scale"],
    "dgamma":            ["a", "loc", "scale"],
    "dweibull":          ["c", "loc", "scale"],
    "erlang":            ["a", "loc", "scale"],
    "expon":             ["loc", "scale"],
    "exponnorm":         ["K", "loc", "scale"],
    "exponpow":          ["b", "loc", "scale"],
    "exponweib":         ["a", "c", "loc", "scale"],
    "f":                 ["dfn", "dfd", "loc", "scale"],
    "fatiguelife":       ["c", "loc", "scale"],
    "fisk":              ["c", "loc", "scale"],
    "foldcauchy":        ["c", "loc", "scale"],
    "foldnorm":          ["c", "loc", "scale"],
    "frechet_l":         ["c", "loc", "scale"],
    "frechet_r":         ["c", "loc", "scale"],
    "gamma":             ["a", "loc", "scale"],
    "gausshyper":        ["a", "b", "c", "z", "loc", "scale"],
    "genexpon":          ["a", "b", "c", "loc", "scale"],
    "genextreme":        ["c", "loc", "scale"],
    "gengamma":          ["a", "c", "loc", "scale"],
    "genhalflogistic":   ["c", "loc", "scale"],
    "geninvgauss":       ["p", "b", "loc", "scale"],
    "genlogistic":       ["c", "loc", "scale"],
    "gennorm":           ["beta", "loc", "scale"],
    "genpareto":         ["c", "loc", "scale"],
    "gilbrat":           ["loc", "scale"],
    "gompertz":          ["c", "loc", "scale"],
    "gumbel_l":          ["loc", "scale"],
    "gumbel_r":          ["loc", "scale"],
    "halfcauchy":        ["loc", "scale"],
    "halfgennorm":       ["beta", "loc", "scale"],
    "halflogistic":      ["loc", "scale"],
    "halfnorm":          ["loc", "scale"],
    "hypsecant":         ["loc", "scale"],
    "invgamma":          ["a", "loc", "scale"],
    "invgauss":          ["mu", "loc", "scale"],
    "invweibull":        ["c", "loc", "scale"],
    "johnsonsb":         ["a", "b", "loc", "scale"],
    "johnsonsu":         ["a", "b", "loc", "scale"],
    "kappa3":            ["a", "loc", "scale"],
    "kappa4":            ["h", "k", "loc", "scale"],
    "ksone":             ["n", "loc", "scale"],
    "kstwo":             ["n", "loc", "scale"],
    "kstwobign":         ["loc", "scale"],
```

```
    "laplace":          ["loc", "scale"],
    "levy":             ["loc", "scale"],
    "levy_l":           ["loc", "scale"],
    "levy_stable":      ["alpha", "beta", "loc", "scale"],
    "loggamma":         ["c", "loc", "scale"],
    "logistic":         ["loc", "scale"],
    "loglaplace":       ["c", "loc", "scale"],
    "lognorm":          ["s", "loc", "scale"],
    "loguniform":       ["a", "b", "loc", "scale"],
    "lomax":            ["c", "loc", "scale"],
    "maxwell":          ["loc", "scale"],
    "mielke":           ["k", "s", "loc", "scale"],
    "moyal":            ["loc", "scale"],
    "nakagami":         ["nu", "loc", "scale"],
    "ncf":              ["dfn", "dfd", "nc", "loc", "scale"],
    "nct":              ["df", "nc", "loc", "scale"],
    "ncx2":             ["df", "nc", "loc", "scale"],
    "norm":             ["loc", "scale"],
    "norminvgauss":     ["a", "b", "loc", "scale"],
    "pareto":           ["b", "loc", "scale"],
    "pearson3":         ["skew", "loc", "scale"],
    "powerlaw":         ["a", "loc", "scale"],
    "powerlognorm":     ["c", "s", "loc", "scale"],
    "powernorm":        ["c", "loc", "scale"],
    "rayleigh":         ["loc", "scale"],
    "rdist":            ["c", "loc", "scale"],
    "recipinvgauss":    ["mu", "loc", "scale"],
    "reciprocal":       ["a", "b", "loc", "scale"],
    "rice":             ["b", "loc", "scale"],
    "semicircular":     ["loc", "scale"],
    "skewnorm":         ["a", "loc", "scale"],
    "t":                ["df", "loc", "scale"],
    "triang":           ["c", "loc", "scale"],
    "truncexpon":       ["b", "loc", "scale"],
    "truncnorm":        ["a", "b", "loc", "scale"],
    "tukeylambda":      ["lam", "loc", "scale"],
    "uniform":          ["loc", "scale"],
    "vonmises":         ["kappa", "loc", "scale"],
    "vonmises_line":    ["kappa", "loc", "scale"],
    "wald":             ["loc", "scale"],
    "weibull_max":      ["c", "loc", "scale"],
    "weibull_min":      ["c", "loc", "scale"],
    "wrapcauchy":       ["c", "loc", "scale"]
}
    print("Available distributions include:", ', '.join(dist_list))
    selected_dists = input("Enter the distributions to fit, separated by commas␣
 ↪(e.g., norm,expon): ").split(',')
```

```
    for dist_name in selected_dists:
        dist_name = dist_name.strip()
        if dist_name in dist_parm_dict:
            try:
                params = getattr(stats, dist_name).fit(data)
                plot_and_calculate_stats(data, dist_name, params,␣
↪dist_parm_dict[dist_name])
            except Exception as e:
                print(f"Could not fit {dist_name} due to: {e}")
        else:
            print(f"{dist_name} is not listed or does not have predefined␣
↪parameter names.")
```

```
Type 'common' to fit common distributions, or 'selected' for specific
distributions: common
```

```
2024-02-26 21:41:10.855 | WARNING  |
fitter.fitter:_fit_single_distribution:347 -
SKIPPED _fit distribution (taking more than 100 seconds)
2024-02-26 21:41:10.978 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted anglit distribution with error=0.0)
2024-02-26 21:41:11.040 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted arcsine distribution with error=0.0)
2024-02-26 21:41:11.042 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted alpha distribution with error=0.0)
2024-02-26 21:41:11.195 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted argus distribution with error=0.0)
2024-02-26 21:41:11.354 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted beta distribution with error=0.0)
2024-02-26 21:41:11.530 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted bradford distribution with error=0.0)
2024-02-26 21:41:11.532 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted betaprime distribution with error=0.0)
2024-02-26 21:41:11.805 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted burr12 distribution with error=1e-06)
2024-02-26 21:41:11.850 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted cauchy distribution with error=0.0)
2024-02-26 21:41:12.065 | INFO     |
```

```
fitter.fitter:_fit_single_distribution:337 -
Fitted chi distribution with error=1e-06)
2024-02-26 21:41:12.210 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted burr distribution with error=0.0)
2024-02-26 21:41:12.249 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted chi2 distribution with error=0.0)
2024-02-26 21:41:12.264 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted cosine distribution with error=0.0)
2024-02-26 21:41:12.351 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted dgamma distribution with error=0.0)
2024-02-26 21:41:12.431 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted dweibull distribution with error=0.0)
2024-02-26 21:41:12.552 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted erlang distribution with error=0.0)
2024-02-26 21:41:12.571 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted expon distribution with error=0.0)
2024-02-26 21:41:12.662 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted exponnorm distribution with error=0.0)
2024-02-26 21:41:12.789 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted crystalball distribution with error=0.0)
2024-02-26 21:41:13.172 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted exponpow distribution with error=0.0)
2024-02-26 21:41:13.439 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted f distribution with error=0.0)
2024-02-26 21:41:13.497 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted exponweib distribution with error=1e-06)
2024-02-26 21:41:13.607 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted fatiguelife distribution with error=0.0)
2024-02-26 21:41:13.800 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted foldcauchy distribution with error=0.0)
2024-02-26 21:41:13.945 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted fisk distribution with error=0.0)
2024-02-26 21:41:14.000 | INFO     |
```

```
fitter.fitter:_fit_single_distribution:337 -
Fitted foldnorm distribution with error=0.0)
2024-02-26 21:41:14.047 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted gamma distribution with error=0.0)
2024-02-26 21:41:14.522 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted gausshyper distribution with error=0.0)
2024-02-26 21:41:14.822 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted genexpon distribution with error=0.0)
2024-02-26 21:41:15.097 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted genextreme distribution with error=1e-06)
2024-02-26 21:41:15.193 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted gengamma distribution with error=1e-06)
2024-02-26 21:41:15.372 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted genhalflogistic distribution with error=0.0)
/usr/local/lib/python3.10/dist-packages/scipy/stats/_continuous_distns.py:3759:
IntegrationWarning: The algorithm does not converge.  Roundoff error is detected
  in the extrapolation table.  It is assumed that the requested tolerance
  cannot be achieved, and that the returned result (if full_output = 1) is
  the best which can be obtained.
  intgrl = integrate.quad(llc, x0, x1,
/usr/local/lib/python3.10/dist-packages/scipy/stats/_continuous_distns.py:3759:
IntegrationWarning: The integral is probably divergent, or slowly convergent.
  intgrl = integrate.quad(llc, x0, x1,
/usr/local/lib/python3.10/dist-packages/scipy/stats/_continuous_distns.py:3754:
IntegrationWarning: The integral is probably divergent, or slowly convergent.
  intgrl = (integrate.quad(llc, x0, mean,
2024-02-26 21:41:15.806 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted genhyperbolic distribution with error=0.0)
2024-02-26 21:41:15.959 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted geninvgauss distribution with error=0.0)
2024-02-26 21:41:16.000 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted genlogistic distribution with error=0.0)
2024-02-26 21:41:16.254 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted gennorm distribution with error=0.0)
2024-02-26 21:41:16.465 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted genpareto distribution with error=0.0)
2024-02-26 21:41:16.498 | INFO     |
```

```
fitter.fitter:_fit_single_distribution:337 -
Fitted gibrat distribution with error=0.0)
2024-02-26 21:41:16.523 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted gumbel_l distribution with error=0.0)
2024-02-26 21:41:16.538 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted gumbel_r distribution with error=0.0)
2024-02-26 21:41:16.792 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted halfcauchy distribution with error=0.0)
2024-02-26 21:41:16.954 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted gompertz distribution with error=0.0)
2024-02-26 21:41:17.091 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted halflogistic distribution with error=0.0)
2024-02-26 21:41:17.139 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted halfgennorm distribution with error=1e-06)
2024-02-26 21:41:17.206 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted hypsecant distribution with error=0.0)
2024-02-26 21:41:17.247 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted halfnorm distribution with error=0.0)
2024-02-26 21:41:17.462 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted invgauss distribution with error=0.0)
2024-02-26 21:41:17.464 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted invgamma distribution with error=0.0)
2024-02-26 21:41:17.940 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted invweibull distribution with error=0.0)
2024-02-26 21:41:17.973 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted johnsonsb distribution with error=0.0)
2024-02-26 21:41:18.202 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted kappa3 distribution with error=0.0)
2024-02-26 21:41:18.289 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted johnsonsu distribution with error=0.0)
2024-02-26 21:41:18.388 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted ksone distribution with error=0.0)
2024-02-26 21:41:18.420 | WARNING  |
```

```
fitter.fitter:_fit_single_distribution:347 -
SKIPPED kstwo distribution (taking more than 100 seconds)
2024-02-26 21:41:18.923 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted kstwobign distribution with error=0.0)
2024-02-26 21:41:18.940 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted laplace distribution with error=0.0)
2024-02-26 21:41:19.059 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted laplace_asymmetric distribution with error=0.0)
2024-02-26 21:41:19.139 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted levy distribution with error=0.0)
2024-02-26 21:41:19.226 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted levy_l distribution with error=0.0)
2024-02-26 21:41:19.675 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted kappa4 distribution with error=0.0)
2024-02-26 21:41:19.862 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted loggamma distribution with error=0.0)
2024-02-26 21:41:19.877 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted levy_stable distribution with error=0.0)
2024-02-26 21:41:19.882 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted logistic distribution with error=0.0)
2024-02-26 21:41:19.900 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted lognorm distribution with error=0.0)
2024-02-26 21:41:20.014 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted loglaplace distribution with error=0.0)
2024-02-26 21:41:20.073 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted loguniform distribution with error=0.0)
2024-02-26 21:41:20.122 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted maxwell distribution with error=0.0)
2024-02-26 21:41:20.201 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted lomax distribution with error=0.0)
2024-02-26 21:41:20.254 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted moyal distribution with error=0.0)
2024-02-26 21:41:20.415 | INFO     |
```

```
fitter.fitter:_fit_single_distribution:337 -
Fitted mielke distribution with error=0.0)
2024-02-26 21:41:20.432 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted nakagami distribution with error=0.0)
2024-02-26 21:41:20.576 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted ncf distribution with error=0.0)
2024-02-26 21:41:21.329 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted nct distribution with error=0.0)
2024-02-26 21:41:21.377 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted norm distribution with error=0.0)
2024-02-26 21:41:22.056 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted ncx2 distribution with error=1e-06)
2024-02-26 21:41:22.081 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted pareto distribution with error=0.0)
2024-02-26 21:41:22.354 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted norminvgauss distribution with error=0.0)
2024-02-26 21:41:22.375 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted pearson3 distribution with error=0.0)
2024-02-26 21:41:22.379 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted powerlaw distribution with error=0.0)
2024-02-26 21:41:23.190 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted powerlognorm distribution with error=1e-06)
2024-02-26 21:41:23.200 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted powernorm distribution with error=nan)
2024-02-26 21:41:23.203 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted rayleigh distribution with error=0.0)
2024-02-26 21:41:23.485 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted rdist distribution with error=0.0)
2024-02-26 21:41:24.014 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted reciprocal distribution with error=0.0)
2024-02-26 21:41:24.484 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted rel_breitwigner distribution with error=0.0)
2024-02-26 21:41:24.872 | INFO     |
```

```
fitter.fitter:_fit_single_distribution:337 -
Fitted rice distribution with error=0.0)
2024-02-26 21:41:24.881 | WARNING  |
fitter.fitter:_fit_single_distribution:347 -
SKIPPED rv_continuous distribution (taking more than 100 seconds)
2024-02-26 21:41:24.884 | WARNING  |
fitter.fitter:_fit_single_distribution:347 -
SKIPPED rv_histogram distribution (taking more than 100 seconds)
2024-02-26 21:41:25.260 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted semicircular distribution with error=0.0)
2024-02-26 21:41:25.890 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted skewcauchy distribution with error=0.0)
2024-02-26 21:41:27.648 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted skewnorm distribution with error=0.0)
2024-02-26 21:41:30.208 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted recipinvgauss distribution with error=0.0)
2024-02-26 21:41:32.533 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted t distribution with error=0.0)
2024-02-26 21:41:32.661 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted trapezoid distribution with error=0.0)
2024-02-26 21:41:32.816 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted trapz distribution with error=0.0)
2024-02-26 21:41:33.424 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted triang distribution with error=0.0)
2024-02-26 21:41:33.563 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted truncexpon distribution with error=0.0)
2024-02-26 21:41:34.257 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted truncnorm distribution with error=1e-06)
2024-02-26 21:41:34.475 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted truncpareto distribution with error=0.0)
2024-02-26 21:41:34.672 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted truncweibull_min distribution with error=0.0)
2024-02-26 21:41:35.097 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted tukeylambda distribution with error=0.0)
2024-02-26 21:41:35.129 | INFO     |
```

```
fitter.fitter:_fit_single_distribution:337 -
Fitted uniform distribution with error=0.0)
2024-02-26 21:41:35.166 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted vonmises distribution with error=2.575671)
2024-02-26 21:41:35.187 | WARNING  |
fitter.fitter:_fit_single_distribution:347 -
SKIPPED vonmises_fisher distribution (taking more than 100 seconds)
2024-02-26 21:41:38.678 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted vonmises_line distribution with error=0.0)
2024-02-26 21:41:38.816 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted wald distribution with error=0.0)
2024-02-26 21:41:38.999 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted weibull_max distribution with error=1e-06)
2024-02-26 21:41:39.119 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted weibull_min distribution with error=0.0)
2024-02-26 21:41:39.386 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted wrapcauchy distribution with error=0.0)
/usr/local/lib/python3.10/dist-packages/scipy/integrate/_quadpack_py.py:1233:
IntegrationWarning: The maximum number of subdivisions (50) has been achieved.
  If increasing the limit yields no improvement it is advised to analyze
  the integrand in order to determine the difficulties.  If the position of a
  local difficulty can be determined (singularity, discontinuity) one will
  probably gain from splitting up the interval and calling the integrator
  on the subranges.  Perhaps a special-purpose integrator should be used.
  quad_r = quad(f, low, high, args=args, full_output=self.full_output,
/usr/local/lib/python3.10/dist-packages/scipy/integrate/_quadpack_py.py:1233:
IntegrationWarning: The integral is probably divergent, or slowly convergent.
  quad_r = quad(f, low, high, args=args, full_output=self.full_output,
/usr/local/lib/python3.10/dist-packages/scipy/integrate/_quadpack_py.py:1233:
IntegrationWarning: The occurrence of roundoff error is detected, which prevents
  the requested tolerance from being achieved.  The error may be
  underestimated.
  quad_r = quad(f, low, high, args=args, full_output=self.full_output,
```

4 Capola Fitting

[17]:
```
!pip install copulas
```

```
Collecting copulas
  Downloading copulas-0.10.0-py2.py3-none-any.whl (55 kB)
                             55.2/55.2 kB
```

```
1.6 MB/s eta 0:00:00
Requirement already satisfied: plotly<6,>=5.10.0 in
/usr/local/lib/python3.10/dist-packages (from copulas) (5.15.0)
Requirement already satisfied: numpy<2,>=1.23.3 in
/usr/local/lib/python3.10/dist-packages (from copulas) (1.25.2)
Requirement already satisfied: scipy<2,>=1.9.2 in
/usr/local/lib/python3.10/dist-packages (from copulas) (1.11.4)
Requirement already satisfied: pandas>=1.3.4 in /usr/local/lib/python3.10/dist-
packages (from copulas) (1.5.3)
Requirement already satisfied: python-dateutil>=2.8.1 in
/usr/local/lib/python3.10/dist-packages (from pandas>=1.3.4->copulas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas>=1.3.4->copulas) (2023.4)
Requirement already satisfied: tenacity>=6.2.0 in
/usr/local/lib/python3.10/dist-packages (from plotly<6,>=5.10.0->copulas)
(8.2.3)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-
packages (from plotly<6,>=5.10.0->copulas) (23.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.8.1->pandas>=1.3.4->copulas) (1.16.0)
Installing collected packages: copulas
Successfully installed copulas-0.10.0
```

[35]:
```
!pip install scipy
```

```
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages
(1.11.4)
Requirement already satisfied: numpy<1.28.0,>=1.21.6 in
/usr/local/lib/python3.10/dist-packages (from scipy) (1.25.2)
```

[2]:
```
!git clone https://github.com/sdv-dev/Copulas.git
!cd copulas
```

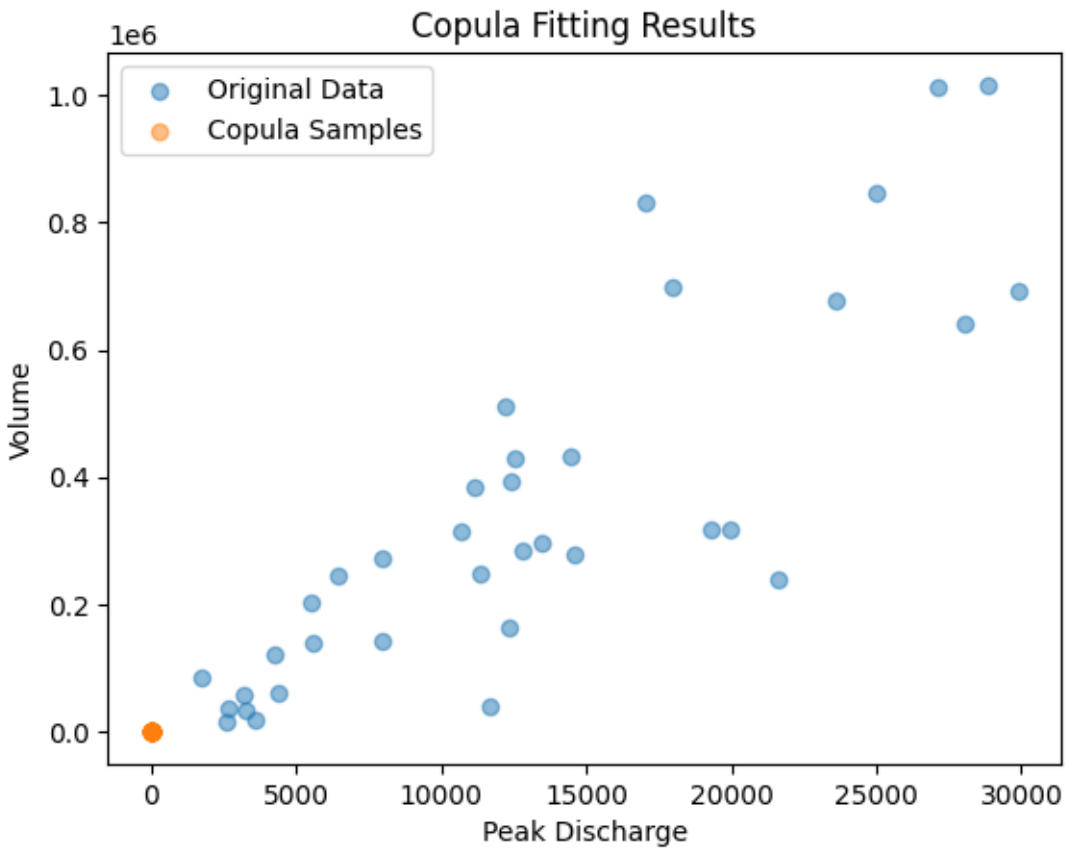
```
Cloning into 'Copulas'…
remote: Enumerating objects: 7120, done.
remote: Counting objects: 100% (1396/1396), done.
remote: Compressing objects: 100% (342/342), done.
remote: Total 7120 (delta 1150), reused 1209 (delta 1049), pack-reused 5724
Receiving objects: 100% (7120/7120), 24.38 MiB | 18.54 MiB/s, done.
Resolving deltas: 100% (4728/4728), done.
/bin/bash: line 1: cd: copulas: No such file or directory
```

[15]:
```
!cd Copulas
```

[23]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
```

```python
from copulas.bivariate.frank import Frank

from scipy.stats import rankdata

def transform_to_uniform(column):
    ranks = rankdata(column, method='average')
    uniform = (ranks - 1) / (len(column) - 1)
    uniform = np.clip(uniform, 0.0001, 0.9999)
    return uniform

def fit_frank_copula(df, col1, col2):
    u = transform_to_uniform(df[col1])
    v = transform_to_uniform(df[col2])
    data = np.vstack([u, v]).T
    copula = Frank()
    copula.fit(data)
    copula.theta = copula.compute_theta()
    model_details = {
        'theta': copula.theta,
        'tau': copula.tau,
    }
    return copula, model_details

def plot_copula_results(copula, df, col1, col2):
    samples = copula.sample(len(df))
    plt.scatter(df[col1], df[col2], label='Original Data', alpha=0.5)
    plt.scatter(samples[:, 0], samples[:, 1], label='Copula Samples', alpha=0.5)
    plt.xlabel(col1)
    plt.ylabel(col2)
    plt.legend()
    plt.title('Copula Fitting Results')
    plt.show()

if __name__ == "__main__":
    filename = data
    print("Available columns:", list(data.columns))
    col1 = 'Peak Discharge'
    col2 = 'Volume'
    copula, model_details = fit_frank_copula(data, col1, col2)

    print("Model Details:")
    print(model_details)
    plot_copula_results(copula, data, col1, col2)
```

```
Available columns: ['YEAR', 'Duration', 'Peak Discharge', 'Volume']
Model Details:
{'theta': 10.526079154615662, 'tau': 0.6793650793650794}
```

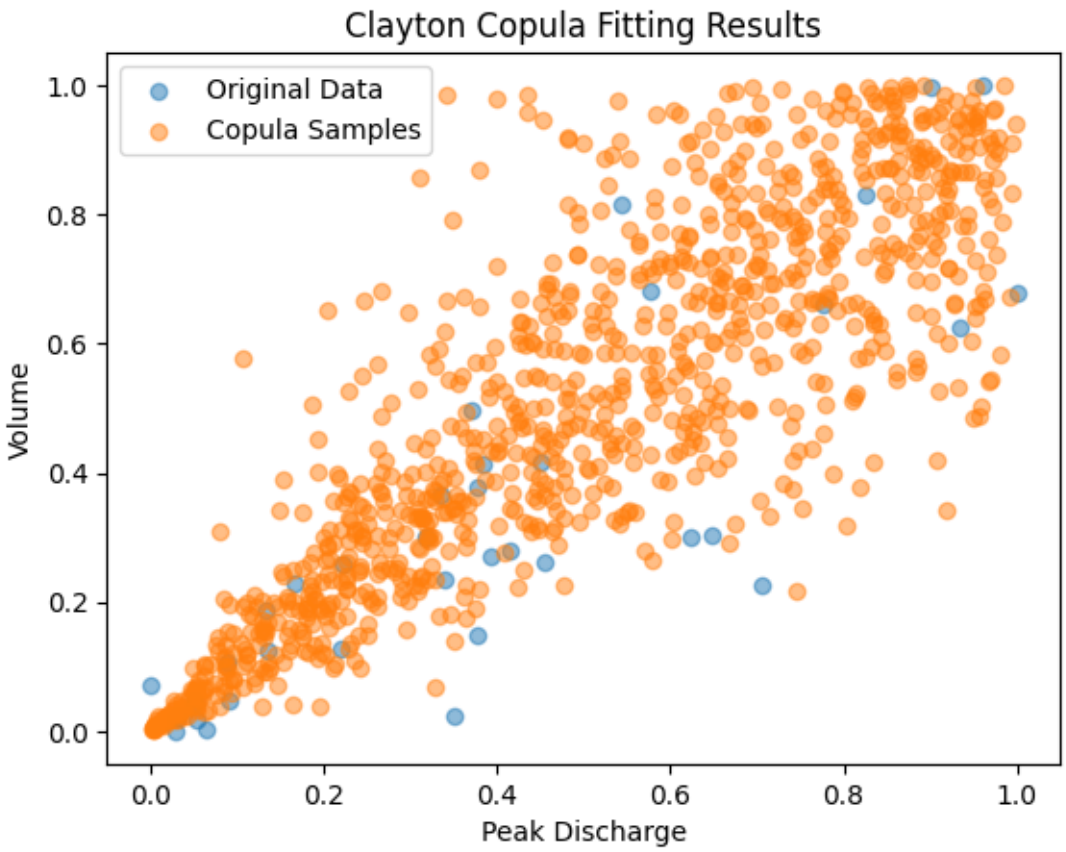


[26]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
from copulas.bivariate.clayton import Clayton
from scipy.stats import rankdata

def preprocess_data(df, col1, col2):
    df[col1] = (df[col1] - df[col1].min()) / (df[col1].max() - df[col1].min())
    df[col2] = (df[col2] - df[col2].min()) / (df[col2].max() - df[col2].min())
    return df[[col1, col2]]

def fit_clayton_copula(df, col1, col2):
    copula = Clayton()
    data = df[[col1, col2]].values
    copula.fit(data)
    theta = copula.compute_theta()
    model_details = {
        'theta': theta,
    }
```

```
    return copula, model_details

def plot_copula_results(copula, df, col1, col2, n_samples=1000):
    samples = copula.sample(n_samples)
    plt.scatter(df[col1], df[col2], label='Original Data', alpha=0.5)
    plt.scatter(samples[:, 0], samples[:, 1], label='Copula Samples', alpha=0.5)
    plt.xlabel(col1)
    plt.ylabel(col2)
    plt.legend()
    plt.title('Clayton Copula Fitting Results')
    plt.show()

# Main execution
if __name__ == "__main__":
    filename = data

    col1 = 'Peak Discharge'
    col2 = 'Volume'
    df_preprocessed = preprocess_data(data, col1, col2)
    copula, model_details = fit_clayton_copula(df_preprocessed, col1, col2)

    print("Model Details:")
    print(model_details)
    plot_copula_results(copula, df_preprocessed, col1, col2)
```

```
/usr/local/lib/python3.10/dist-packages/copulas/bivariate/base.py:162:
RuntimeWarning: Data does not appear to be uniform.
  warnings.warn('Data does not appear to be uniform.', category=RuntimeWarning)
```

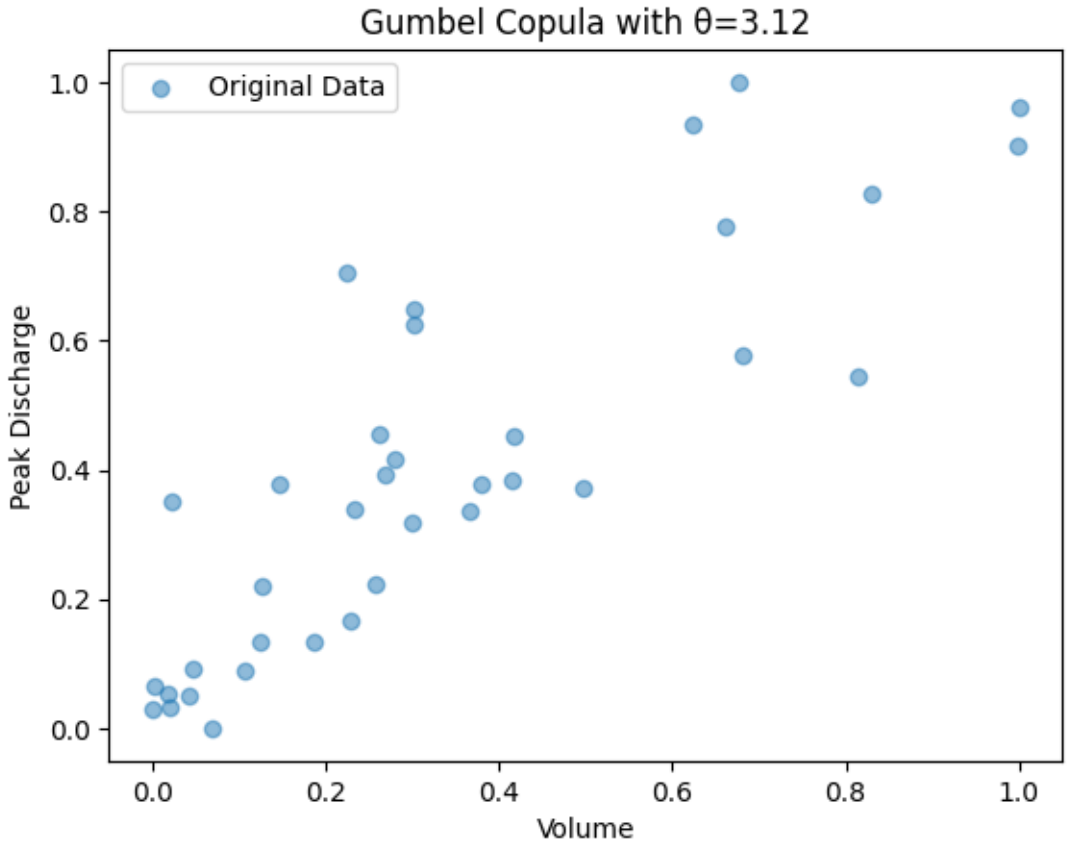
```
Model Details:
{'theta': 4.237623762376238}
```



[37]:
```python
from copulas.bivariate.gumbel import Gumbel
from scipy.stats import kendalltau

def normalize_data(df, col1, col2):
    df[col1] = (df[col1] - df[col1].min()) / (df[col1].max() - df[col1].min())
    df[col2] = (df[col2] - df[col2].min()) / (df[col2].max() - df[col2].min())
    return df[[col1, col2]]

def fit_gumbel_copula(df, col1, col2):
    data = normalize_data(df, col1, col2)
    tau, _ = kendalltau(data[col1], data[col2])
    copula = Gumbel()
    copula.tau = tau
    copula.theta = copula.compute_theta()

    return copula.theta, tau

def plot_data_and_copula(df, col1, col2, theta):
    plt.scatter(df[col1], df[col2], alpha=0.5, label='Original Data')
    plt.xlabel(col1)
```

```
    plt.ylabel(col2)
    plt.legend()
    plt.title(f'Gumbel Copula with  ={theta:.2f}')
    plt.show()

if __name__ == "__main__":
    filename = data
    col1 = 'Volume'
    col2 = 'Peak Discharge'
    theta, tau = fit_gumbel_copula(data, col1, col2)
    print(f"Theta: {theta}, Tau: {tau}")
    plot_data_and_copula(data, col1, col2, theta)
```

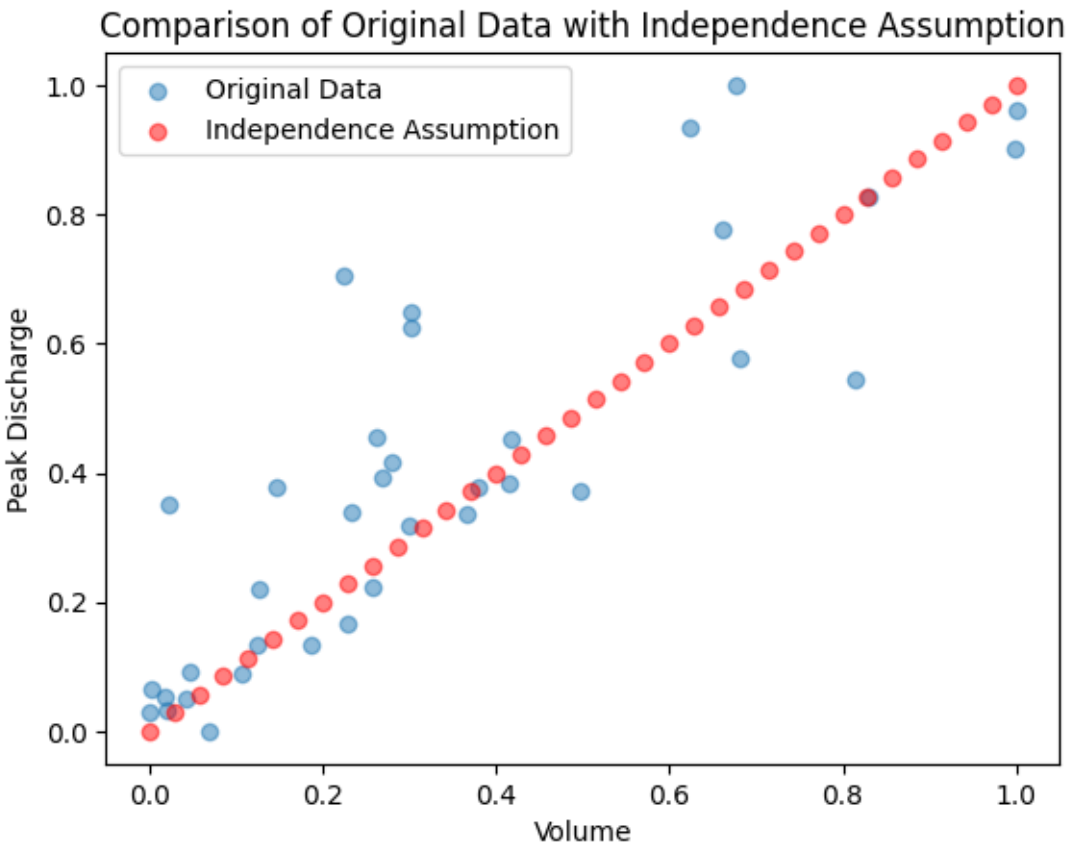
```
Theta: 3.118811881188119, Tau: 0.6793650793650794
```



[41]:
```
from copulas.bivariate.independence import Independence

def normalize_data(df, col1, col2):
    df[col1] = (df[col1] - df[col1].min()) / (df[col1].max() - df[col1].min())
    df[col2] = (df[col2] - df[col2].min()) / (df[col2].max() - df[col2].min())
```

```
    return df[[col1, col2]]

data = normalize_data(data, 'Volume', 'Peak Discharge')
independence_copula = Independence()
plt.scatter(data['Volume'], data['Peak Discharge'], alpha=0.5, label='Original␣
 ↪Data')
u = np.linspace(0, 1, len(data))
v = np.linspace(0, 1, len(data))
plt.scatter(u, v, alpha=0.5, label='Independence Assumption', color='r')
plt.xlabel('Volume')
plt.ylabel('Peak Discharge')
plt.legend()
plt.title('Comparison of Original Data with Independence Assumption')
plt.show()
```

```
<ipython-input-41-def4cc100b71>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col1] = (df[col1] - df[col1].min()) / (df[col1].max() - df[col1].min())
<ipython-input-41-def4cc100b71>:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col2] = (df[col2] - df[col2].min()) / (df[col2].max() - df[col2].min())
```


Comparison of Original Data with Independence Assumption
Original Data
Independence Assumption
Peak Discharge
Volume
0.0
0.2
0.4
0.6
0.8
1.0