

---

## UMB W04: podstawy opisu statystycznego danych

---

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
from scipy.stats import norm, multivariate_normal

import umb_tools as umb
```

```
# konfiguracja
```

```
plt.rcParams["figure.figsize"] = [5, 4]
```

# 1. Jednowymiarowe rozkłady prawdopodobieństwa (na przykładzie rozkładu normalnego)

## Właściwości teoretycznego rozkładu prawdopodobieństwa

```
# parametry rozkładu teoretycznego
```

```
mi = 4.0  
sigma = 1.0
```

```
# funkcja gęstości prawdopodobieństwa (PDF) i dystrybuanta (CDF)
```

```
x = np.linspace(0.5, 7.5, 100)
```

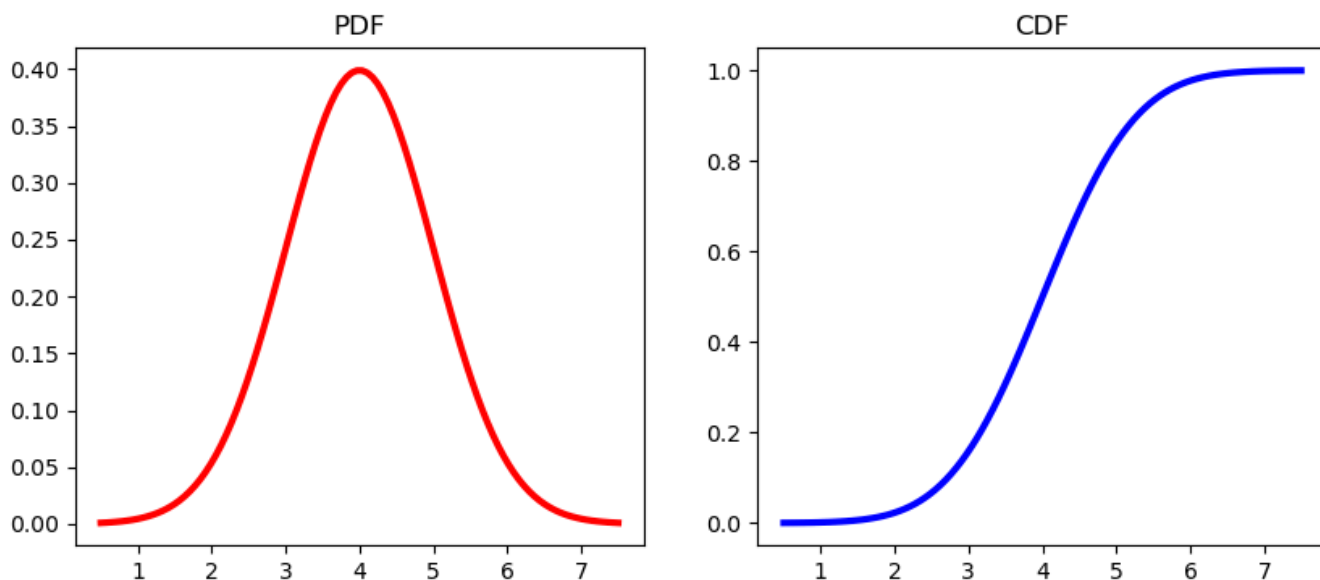
```
pdf = norm.pdf(x, mi, sigma)  
cdf = norm.cdf(x, mi, sigma)
```

```
fig, ax = plt.subplots(1, 2, figsize=(10, 4))
```

```
ax[0].plot(x, pdf, "r-", linewidth=3)  
ax[0].set_title("PDF")
```

```
ax[1].plot(x, cdf, "b-", linewidth=3)  
ax[1].set_title("CDF")
```

```
plt.show()
```



```
# kwantyle rozkładu teoretycznego
```

```
q01 = norm.ppf(0.01, mi, sigma)  
q99 = norm.ppf(0.99, mi, sigma)
```

```
print(f"\n1st percentile = {q01} \n99th percentile = {q99}")
```

```
1st percentile = 1.6736521259591592
99th percentile = 6.326347874040841
```

```
# podstawowe momenty rozkładu teoretycznego
```

```
m, v, s, k = norm.stats(mi, sigma, moments='mvsk')
```

```
print(f"\nmean = {m} \nvariance = {v} \nskewness = {s} \nkurtosis= {k}")
```

```
mean = 4.0
variance = 1.0
skewness = 0.0
kurtosis= 0.0
```

---

## Generowanie wartości zgodnych z teoretycznym rozkładem prawdopodobieństwa

```
# pseudolosowa próba z rozkładu normalnego
```

```
np.random.seed(123)
```

```
sample = norm.rvs(mi, sigma, size=1000)
```

---

## Właściwości empirycznego rozkładu prawdopodobieństwa z próby

```
# podstawowe momenty rozkładu z próby
```

```
d = stats.describe(sample)
```

```
print(f"\nmean = {d.mean:.4f} \nvariance = {d.variance:.4f} \nskewness = {d.skewness:.4f} \nku
```

```
mean = 3.9604
variance = 1.0026
skewness = -0.0290
kurtosis= -0.0254
```

```
# funkcja gęstości prawdopodobieństwa estymowana z prób za pomocą histogramu
```

```
h = np.histogram(sample, bins=30)
```

```
hist = stats.rv_histogram(h, density=True)
```

```
# funkcja gęstości prawdopodobieństwa estymowana z próby za pomocą KDE
```

```
kde = stats.gaussian_kde(sample)
```

```

x = np.linspace(0.5, 7.5, 100)

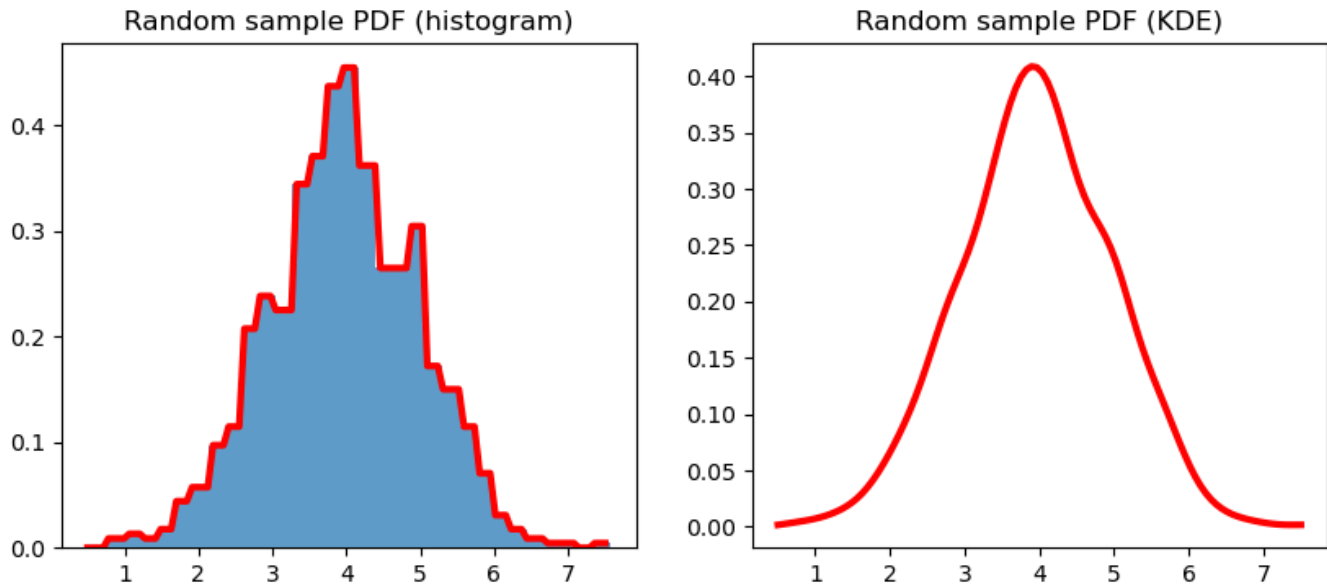
fig, ax = plt.subplots(1, 2, figsize=(10, 4))

ax[0].hist(sample, bins=30, density=True, alpha=0.7)
ax[0].plot(x, hist.pdf(x), "r-", linewidth=3)
ax[0].set_title("Random sample PDF (histogram)")

ax[1].plot(x, kde(x), "r-", linewidth=3)
ax[1].set_title("Random sample PDF (KDE)")

plt.show()

```



```

# kwantyle z próby wykreślone wobec kwantyli rozkładów normalnego i t-Studenta

qs = np.quantile(sample, np.linspace(0.01, 0.99, 99))

qn = norm.ppf(np.linspace(0.01, 0.99, 99), mi, sigma)
qt = stats.t.ppf(np.linspace(0.01, 0.99, 99), 2)

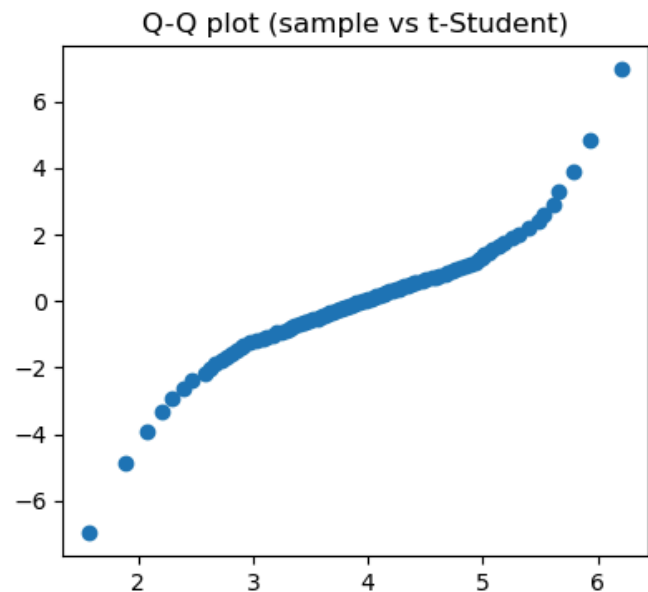
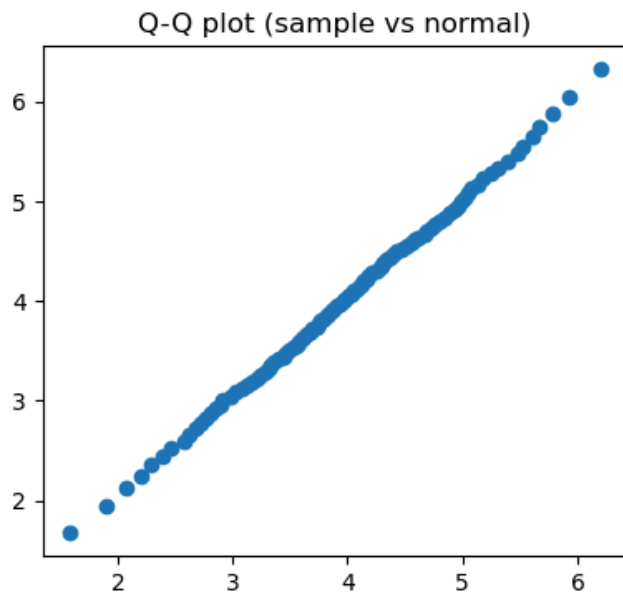
fig, ax = plt.subplots(1, 2, figsize=(10, 4))

ax[0].scatter(qs, qn)
ax[0].set_title("Q-Q plot (sample vs normal)")

ax[1].scatter(qs, qt)
ax[1].set_title("Q-Q plot (sample vs t-Student)")

plt.show()

```



### Wartości odstające (outliers)

```
# próba z rozkładu normalnego  $N(0, 1)$ 
np.random.seed(123)
sample = norm.rvs(0.0, 1.0, size=100)
```

```
# dodanie wartości odstających
sample=np.append(sample, [14, 15])
```

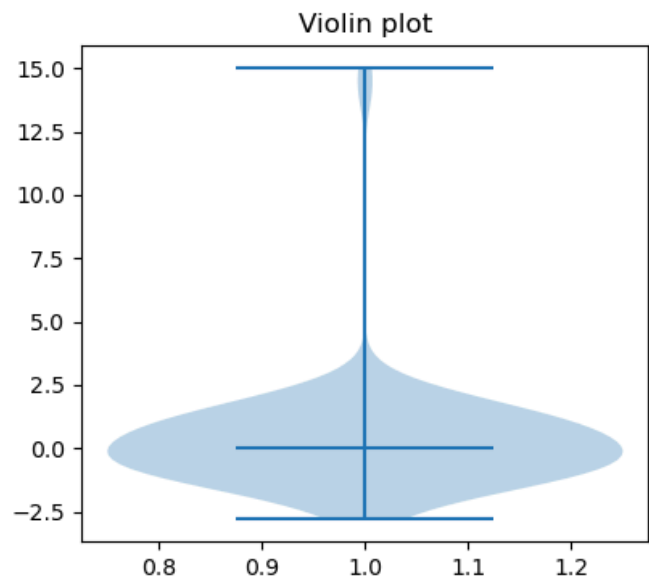
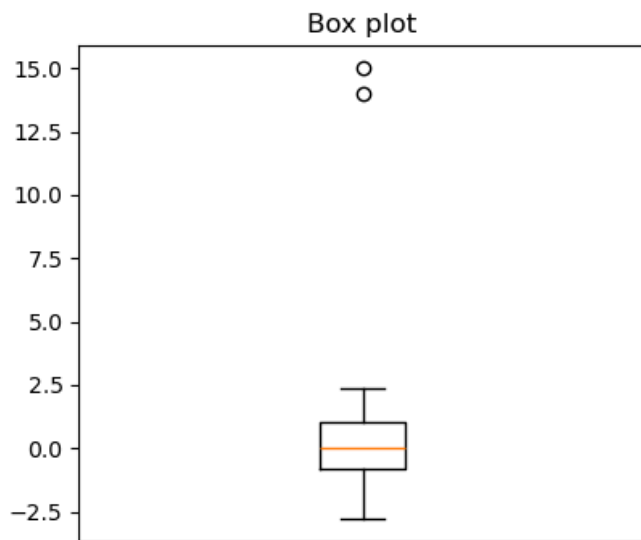
```
# wykresy rozrzutu wartości w próbie
```

```
fig, ax = plt.subplots(1, 2, figsize=(10, 4))
```

```
ax[0].boxplot(sample)
ax[0].set_xticks([])
ax[0].set_title("Box plot")
```

```
ax[1].violinplot(sample, showmedians=True)
ax[1].set_title("Violin plot")
```

```
plt.show()
```



```
# wykrywanie odstających wartości
```

```
Q1 = np.quantile(sample, 0.25)
```

```
Q3 = np.quantile(sample, 0.75)
```

```
IQR = Q3 - Q1
```

```
print(f"\nOutliers: \n {sample[sample > (Q3 + 1.5*IQR)]} \n {sample[sample < (Q1 - 1.5*IQR)]}")
```

Outliers:

```
[14. 15.]
```

```
[]
```

```
# wartość oczekiwana i odchylenie standardowe
```

```
m = np.mean(sample)
```

```
sd = np.std(sample)
```

```
print(f"\nM = {m:.4f} \nSD = {sd:.4f}")
```

M = 0.3109

SD = 2.2977

```
# odporne estymatory wartości oczekiwanej (median i średni obcięta)
```

```
m_median = np.median(sample)
```

```
m_tmean = stats.trim_mean(sample, 0.1)
```

```
print(f"\nM (median) = {m_median:.4f} \nM (trimmed mean) = {m_tmean:.4f}")
```

M (median) = -0.0045

M (trimmed mean) = 0.0551

```
# odporne estymatory odchylenia standardowego
```

```
sd_iqr = stats.iqr(sample)/1.349
```

```
sd_mad = stats.median_abs_deviation(sample)/0.6745

print(f"\nSD (IQR) = {sd_iqr:.4f} \nSD (MAD) = {sd_mad:.4f}")
```

SD (IQR) = 1.3544  
SD (MAD) = 1.3394

---

## 2. Współczynnik korelacji

```
x = np.array([ 0.0, 1.0, 2.0, 3.0, 4.0])
y = np.array([-1.0, 0.2, 0.9, 2.1, 8.9])

# współczynnik korelacji liniowej (Pearsona)
rp = stats.pearsonr(x, y)

# rangowy współczynnik korelacji (Spearmana)
rs = stats.spearmanr(x, y)

print(f"\nRp = {rp.statistic:.4f} \nRs = {rs.statistic:.4f}")
```

Rp = 0.8798  
Rs = 1.0000

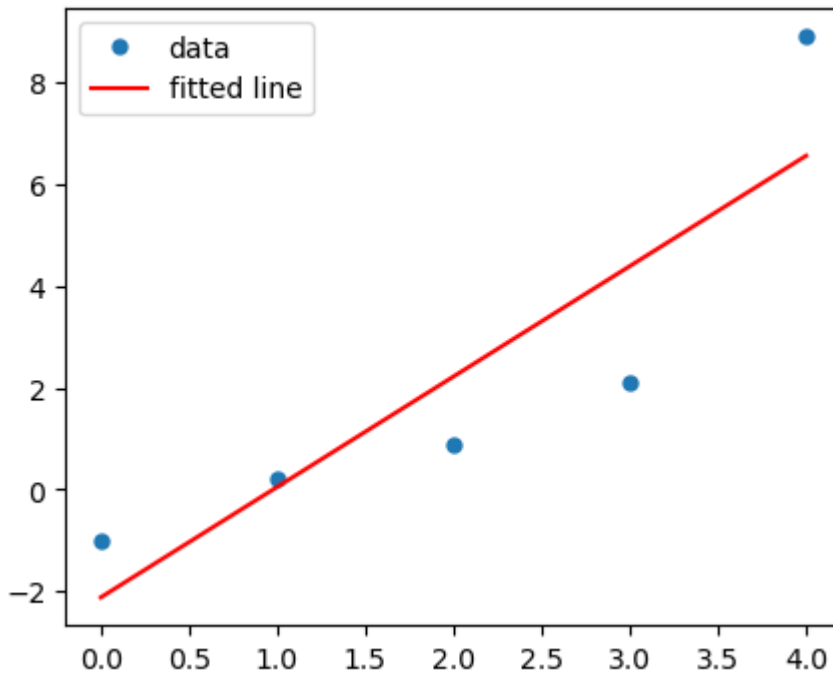
```
# wyznaczanie współczynników regresji liniowej

params = stats.linregress(x, y)
a = params.slope
b = params.intercept

fig, ax = plt.subplots()

ax.plot(x, y, 'o', label='data', markersize=5)
ax.plot(x, a*x + b, 'r', label='fitted line')

plt.legend()
plt.show()
```



---

### 3. Wielowymiarowe rozkłady prawdopodobieństwa (na przykładzie rozkładu normalnego)

---

#### Właściwości teoretycznego rozkładu prawdopodobieństwa

```
# parametry rozkładu teoretycznego 2D
```

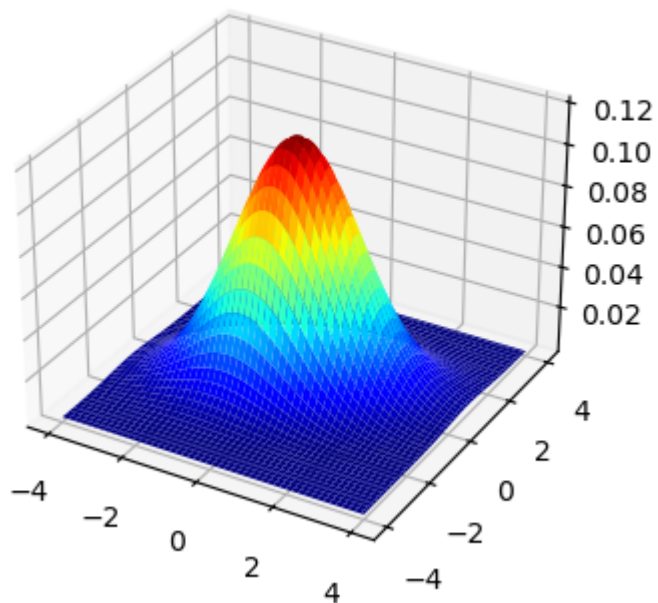
```
mi = [0, 0]  
sigma = [[2.0, 0.5],  
         [0.5, 1.0]]
```

```
# funkcja gęstości prawdopodobieństwa (PDF)
```

```
x, y = np.meshgrid(np.linspace(-4, 4, 100), np.linspace(-4, 4, 100))  
  
pdf = multivariate_normal.pdf(np.dstack((x, y)), mi, sigma)
```

```
fig, ax = plt.subplots(subplot_kw={"projection": "3d"})  
  
ax.plot_surface(x, y, pdf, cmap="jet")  
  
plt.show()
```





---

### Generowanie wartości zgodnych z teoretycznym rozkładem prawdopodobieństwa

```
# pseudolosowa próba z rozkładu normalnego 2D  
  
sample = multivariate_normal.rvs(mi, sigma, size=100)
```

---

### Właściwości empirycznego rozkładu prawdopodobieństwa z próby

```
# podstawowe parametry rozkładu 2D z próby  
  
m = np.mean(sample, axis=0)  
c = np.cov(sample.T)  
  
print(f"\nmean = {m} \n\ncovariance = \n{c}")  
  
mean = [-0.01784551 -0.17107837]  
  
covariance =  
[[1.71200437 0.41235947]  
 [0.41235947 0.99269782]]  
  
fig, ax = plt.subplots()  
  
pdf=multivariate_normal.pdf(np.dstack((x, y)), m, c)  
  
ax.contourf(x, y, pdf, levels=20, cmap="jet")  
ax.scatter(sample[:, 0], sample[:, 1], color='w', s=2)  
  
plt.show()
```

