Analiza Przeżycia

Raport 1

Jakub Zdancewicz, Ksawery Józefowski

Wygenerowano 1 października 2025

Wstęp

Sprawozdanie jest podzielone na cztery części.

Część pierwsza dotyczy analizy rozkładu **Exponentiated Weibull**($\mathcal{EW}$) z parametrami α, β oraz γ. Obejmuje implementację podstawowych funkcji rozkładu, wizualizację funkcji hazardu, generowanie danych losowych oraz analizę statystyczną wygenerowanych prób.

Część druga poświęcona jest analizie danych **cenzurowanych** z uogólnionego rozkładu wykładniczego ($\mathcal{GE}(\lambda, \alpha)$). Zadania obejmują generowanie zmiennych losowych cenzurowanych różnych typów, obliczanie rozsądnych statystyk opisowych oraz analizę przykładowych danych eksperymentalnych z obserwacją remisji choroby w dwóch grupach pacjentów.

W części trzeciej wyznaczymy oszacowania największej wiarygodności oraz realizacje przedziałów ufności dla średniego czasu remisji choroby w dwóch grupach pacjentów rozważanych w części drugiej. Następnie powtórzymy obliczenia, zakładając, że obserwacje prowadzono tylko do momentu uzyskania remisji u dziesięciu pacjentów. Na koniec porównamy dwa punktowe estymatory parametru rozkładu wykładniczego na podstawie danych cenzurowanych, przeprowadzając symulację oraz oceniając ich obciążenie i błąd średniokwadratowy na wygenerowanych danych.

Część czwarta sprawozdania poświęcona jest zagadnieniu weryfikacji hipotez dla danych cenzurowanych I-go typu, pochodzących z rozkładu wykładniczego. W szczególności skonstruujemy i zaimplementujemy test ilorazu wiarygodności dla hipotez dwustronnych oraz jednostronnych (lewostronnych i prawostronnych) dotyczących parametru ϑ. Następnie przeprowadzimy symulacje mające na celu ocenę własności tego testu, w szczególności jego rozmiaru oraz mocy dla wybranych alternatyw. Opracowany test zastosujemy następnie do weryfikacji hipotezy o średnim czasie remisji choroby na podstawie danych eksperymentalnych w dwóch rozważanych grupach pacjentów.

Deklaracje funkcji rozkładu $\mathcal{EW}(\alpha, \beta, \gamma)$

Rozkład Exponentiated Weibull ($\mathcal{EW}$) ma trzy parametry $\alpha > 0$, $\beta > 0$ oraz $\gamma > 0$. Definiujemy następujące funkcje:

- **Gęstość prawdopodobieństwa:**

$$f(t; \alpha, \beta, \gamma) = \frac{\alpha\,\gamma}{\beta} \left(\frac{t}{\beta}\right)^{\alpha-1} \left[1 - \exp\left(-\left(\frac{t}{\beta}\right)^{\alpha}\right)\right]^{\gamma-1} \exp\left[-\left(\frac{t}{\beta}\right)^{\alpha}\right] \mathbf{1}_{(0,\infty)}(t),$$

- **Dystrybuanta:**

$$F(t; \alpha, \beta, \gamma) = \left[1 - \exp\left(-\left(\frac{t}{\beta}\right)^{\alpha}\right)\right]^{\gamma}, \; t \geq 0$$

- **Dystrybuanta odwrotna:**

$$Q(p; \alpha, \beta, \gamma) = \beta \left[-\ln\left(1 - p^{\frac{1}{\gamma}}\right)\right]^{\frac{1}{\alpha}}, \; p \in (0, 1)$$

- **Funkcja hazardu:**

$$h(t; \alpha, \beta, \gamma) = \frac{f(t; \alpha, \beta, \gamma)}{1 - F(t; \alpha, \beta, \gamma))}$$

Poniżej podajemy implementacje podanych funkcji w języku Python:

```
# Gęstość prawdopodobieństwa
def f(x, alpha, beta, gamma):
    if x > 0:
        return ((alpha * gamma) / beta) * \
               ((x / beta)**(alpha - 1)) * \
               ((1 - math.exp(-(x / beta)**alpha))**(gamma - 1)) * \
              math.exp(-((x / beta)**alpha))
    return 0

# Dystrybuanta
def F(x, alpha, beta, gamma):
    if x <= 0:
        return 0
    return (1 - math.exp(-(x / beta)**alpha))**gamma

# Dystrybuanta odwrotna
def F_inv(p, alpha, beta, gamma):
    if not 0 < p < 1:a
        raise ValueError("p musi być w (0,1)")
```

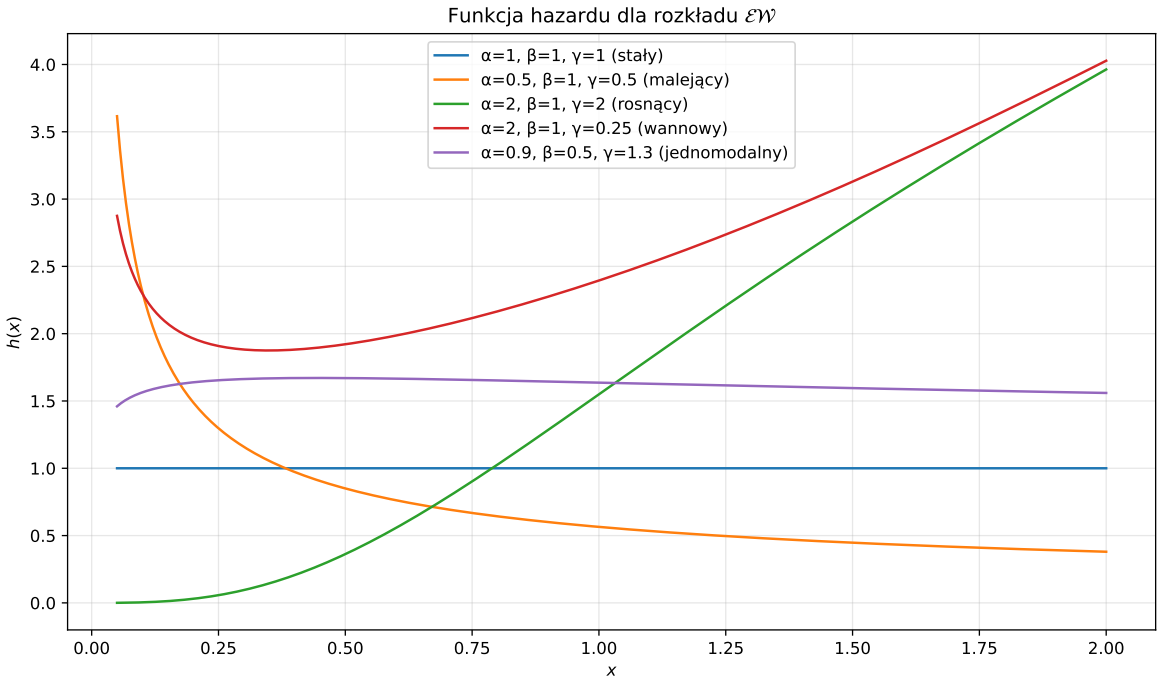
```
    return beta * (-math.log(1 - p**(1/gamma)))**(1/alpha)

# Funkcja hazardu
def h(t, alpha, beta, gamma):
    return f(t, alpha, beta, gamma) / (1 - F(t, alpha, beta, gamma))
```

Wykresy funkcji hazardu dla różnych parametrów

Tworzymy wykresy funkcji hazardu 1 dla zestawów parametrów α, β, γ przedstawionych w tabeli 1

Tabela 1

α	β	γ	Typ wykresu
1	1	1	stały
0.5	1	0.5	malejący
2	1	2	rosnący
2	1	0.25	wannowy
0.9	0.5	1.3	jednomodalny



Rysunek 1: Wykresy funkcji hazardu dla różnych parametrów rozkładu $\mathcal{EW}$

Generowanie zmiennych z rozkładu $\mathcal{EW}$

Do wygenerowania próbek z rozkładu ($\mathcal{EW}$) stosujemy metodę odwrotnej dystrybuanty. Niech $U \sim \mathcal{U}(0,1)$, wówczas zmienna losowa

$$X = F^{-1}(U; \alpha, \beta, \gamma)$$

gdzie F jest dystrybuantą rozkładu $\mathcal{EW}$ ma rozkład $\mathcal{EW}(\alpha, \beta, \gamma)$.

Poniżej przedstawiamy implementację metody odwrotnej dystrybuanty:

```
# Generowanie zmiennych losowych EW
def generate_EW(n, alpha, beta, gamma):
    u = np.random.uniform(0, 1, n)
    F_inv_vec = np.vectorize(F_inv)
    return F_inv_vec(u, alpha, beta, gamma)
```

Histogramy i gęstości dla wybranych parametrów

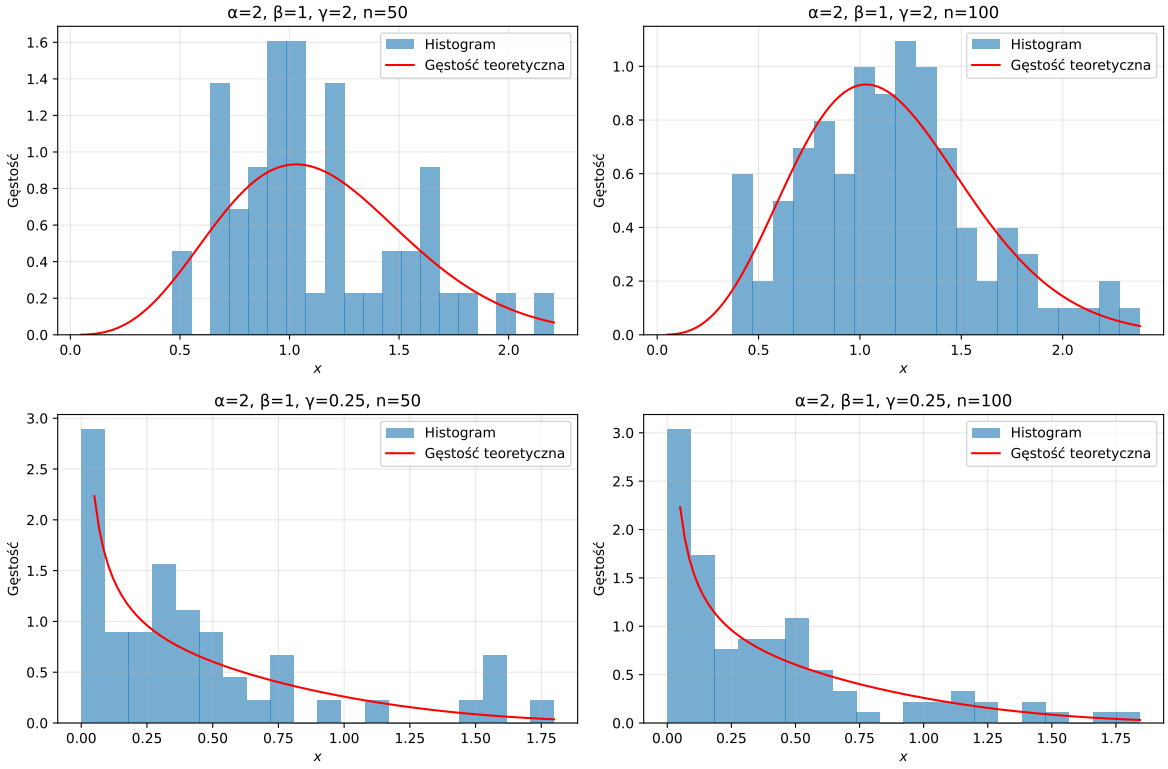
Generujemy dane z rozkładu $\mathcal{EW}$ dla następujących zestawów parametrów:

$$(\alpha, \beta, \gamma) \in \{(2, 1, 2), (2, 1, 0.25)\},$$

oraz dla dwóch rozmiarów prób: $(n = 50)$ i $(n = 100)$.

Dla każdej kombinacji tworzymy histogramy z nałożoną **gęstością teoretyczną**, aby porównać kształt histogramów z kształtem wykresu rozkładu teoretycznego (2).

Można zauważyć, że kształty histogramów są zbliżone do wykresów gęstości teoretycznych, co jest szczególnie widoczne przy większej liczbie obserwacji.



Rysunek 2: Histogramy i gęstości teoretyczne dla rozkładu $\mathcal{EW}$

Statystyki opisowe wygenerowanych próbek

Dla wygenerowanych próbek obliczamy podstawowe statystyki opisowe: średnią, medianę, odchylenie standardowe, pierwszy kwartyl (Q1), trzeci kwartyl (Q3), rozstęp między kwartylowy, minimum oraz maksimum.

Dodatkowo podajemy wartości teoretyczne mediany oraz kwartylów Q1 i Q3, obliczone za pomocą funkcji kwantylowej $F^{-1}(p; \alpha, \beta, \gamma)$, aby umożliwić porównanie wyników empirycznych z wartościami teoretycznymi.

Statystyki teoretyczne

Tabela 2 przedstawia wartości teoretyczne mediany oraz kwartylów Q1 i Q3 dla dwóch rozmiarów prób i wybranych zestawów parametrów:

Tabela 2: Wartości teoretyczne mediany i kwartylów Q1 oraz Q3 dla wygenerowanych próbek.

α	β	γ	$Mediana_{teor.}$	$Q1_{teor.}$	$Q3_{teor.}$
2	1	2	1.10813	0.832555	1.41778
2	1	0.25	0.254044	0.0625612	0.616759

Statystyki empiryczne

Tabela 3 przedstawia wartości empiryczne mediany oraz kwartylów Q1 i Q3 obliczone na podstawie wygenerowanych próbek:

Tabela 3: Wartości empiryczne mediany i kwartylów Q1 oraz Q3 dla wygenerowanych próbek.

α	β	γ	n	$Mediana_{emp.}$	$Q1_{emp.}$	$Q3_{emp.}$
2	1	2	50	1.01077	0.831474	1.32946
2	1	2	100	1.12231	0.853751	1.39262
2	1	0.25	50	0.314101	0.083986	0.54074
2	1	0.25	100	0.270129	0.0815788	0.540099

Można zauważyć, że wartości statystyk empirycznych są zbliżone do wartości statystyk teoretycznych, zwłaszcza w przypadku większych wartości n, co jest zgodne z intuicją.

Dodatkowe statystyki opisowe

Tabela 4 przedstawia dodatkowe statystyki opisowe: średnią, odchylenie standardowe, minimum, maksimum oraz rozstęp między kwartylowy dla tych samych próbek:

Tabela 4: Dodatkowe statystyki opisowe dla wygenerowanych próbek.

α	β	γ	n	Średnia	Odch. std.	Min.	Maks.	Rozstęp
2	1	2	50	1.11529	0.390867	0.464871	2.20705	1.74218
2	1	2	100	1.15452	0.43663	0.371057	2.38142	2.01036
2	1	0.25	50	0.446857	0.470365	0.000371554	1.79839	1.79801
2	1	0.25	100	0.397005	0.425948	5.34687e-05	1.84392	1.84387

Widzimy, że w obu przypadkach wartości średnie są wyższe od wartości mediany, co może wskazywać na prawostronną asymetrię rozkładów.

Generowanie zmiennych cenzurowanych z uogólnionego rozkładu wykładniczego ($\mathcal{GE}$)

Generowanie próbek losowych z rozkładu ($\mathcal{GE}$)

W pierwszym kroku zdefiniujmy funkcję odwrotnej dystrybuanty dla rozkładu $\mathcal{GE}(\lambda, \alpha)$. Umożliwi nam ona generowanie próbek z tego rozkładu metodą odwrotnej dystrybuanty.

$$F^{-1}(u; \lambda, \alpha) = -\frac{1}{\lambda} \ln\left(1 - u^{\frac{1}{\alpha}}\right), \quad u \in (0, 1).$$

Poniżej przedstawiamy implementację metody odwrotnej dystrybuanty dla tego rozkładu:

```
def GE(n, L, alpha):
    # generujemy liczby z U(0,1)
    u = np.random.uniform(0, 1, n)
    # funkcja odwrotnej dystrybuanty
    return -1/L * np.log(1 - u**(1/alpha))
```

Cenzurowanie danych z rozkładu $\mathcal{GE}$

W kolejnych krokach definiujemy trzy algorytmy do generowania danych cenzurowanych z rozkładu $\mathcal{GE}(\lambda, \alpha)$, odpowiadające różnym przypadkom prawostronnego cenzurowania:

1. **Typ I**: Ten typ cenzurowania charakteryzuje się ustalonym czasem trwania eksperymentu t_0.

Algorytm:

- Generujemy próbę $X \sim \mathcal{GE}(\lambda, \alpha)$.
- Tworzymy wektor δ, w którym `True` oznacza obserwacje niecenzurowane ($X_s \leq t_0$), a `False` obserwacje cenzurowane ($X_s > t_0$).
- Dane obserwowane, dla których $X_s > t_0$, przycinamy do wartości t_0.

```
def type_I(t0, L, alpha, n):
    X_nc = GE(n, L, alpha)
    delta = (X_nc <= t0).astype(bool)
    X_c = np.minimum(X_nc, t0)
    return pd.DataFrame({"Data": X_c, "Delta": delta})
```

2. **Typ II**: Ten typ cenzurowania charakteryzuje się ustaloną liczbą zaobserwowanych zdarzeń m.

Algorytm:

- Sortujemy wygenerowane próbki $X \sim \mathcal{GE}(\lambda, \alpha)$ w kolejności rosnącej.
- Pierwsze m najmniejszych wartości traktujemy jako dane niecenzurowane (`delta=True`).
- Pozostałe obserwacje przycinamy do wartości m-tej próbki (`delta=False`).

```
def type_II(m, L, alpha, n):
    X_nc = np.sort(GE(n, L, alpha))
    t_m = X_nc[m-1]
    delta = np.zeros(n, dtype=bool)
    delta[:m] = True
    X_c = np.minimum(X_nc, t_m)
    return pd.DataFrame({"Data": X_c, "Delta": delta})
```

3. **Typ Losowy**: Ten typ cenzurowania charakteryzuje się tym, że czas cenzurowania jest generowany losowo i niezależnie dla każdej obserwacji.

Algorytm:

- Generujemy próbkę $X \sim \mathcal{GE}(\lambda, \alpha)$.
- Dla każdej obserwacji generujemy niezależny czas cenzurowania $C_i \sim \text{Exp}(\eta)$.
- Obserwowane dane przycinamy do wartości cenzury (jeśli są większe) C_i : $X_i^{\text{obs}} = \min(X_i, C_i)$.
- Tworzymy wektor δ, w którym `True` oznacza obserwacje niecenzurowane ($X_i \leq C_i$), a `False` obserwacje cenzurowane ($X_i > C_i$).

```
def type_random(eta, L, alpha, n):
    C_i = GE(n, eta, 1)
    X_nc = GE(n, L, alpha)
    X_c = np.minimum(X_nc, C_i)
    delta = X_c != C_i
    return pd.DataFrame({"Data": X_c, "Delta": delta})
```

Generowanie i analiza danych cenzurowanych

Korzystając z wcześniej zdefiniowanych funkcji, generujemy po jednym zbiorze danych cenzurowanych dla każdego z trzech typów cenzurowania.

Dla zapewnienia powtarzalności wyników ustalamy ziarno generatora liczb losowych `np.random.seed(37)`.

```
np.random.seed(37)
# Generowanie danych cenzurowanych typu I
df_type_I = type_I(t0=1, L=1, alpha=1, n=10000)
# Generowanie danych cenzurowanych typu II
df_type_II = type_II(m=4000, L=1, alpha=1, n=10000)
# Generowanie danych cenzurowanych losowo
df_type_random = type_random(eta=1, L=1, alpha=1, n=10000)
```

Następnie dla wygenerowanych zbiorów danych tworzymy zestawienie sensownych statystyk opisowych w formie tabel 5 oraz 6, które przedstawiają podstawowe miary położenia, rozproszenia oraz liczbę pełnych obserwacji dla każdego typu cenzurowania.

Tabela 5: Liczba danych dla wszystkich typów cenzurowań

	Liczba danych	Liczba pełnych	Liczba cenz.
Cenz. I typu	10000	6334	3666
Cenz. II typu	10000	4000	6000
Cenz. losowe	10000	4997	5003

Tabela 6: Statystyki dla wszystkich typów cenzurowań

	Min	Q1	Mediana	Q3	Max	IQR
Cenz. I typu	0	0.169	0.389	0.642	1	0.473
Cenz. II typu	0	0.103	0.221	0.353	0.498	0.25
Cenz. losowe	0	0.146	0.347	0.706	4.208	0.56

Zrezygnowaliśmy z obliczania średniej oraz odchylenia standardowego, ponieważ w przypadku danych cenzurowanych statystyki te tracą sensowną interpretację i mogą prowadzić do błędnych wniosków.

Analiza czasu do remisji w badaniu klinicznym z cenzurowaniem

Rozważamy dane pochodzące od 40 pacjentów, podzielonych losowo na dwie równoliczne grupy po 20 osób każda. W każdej grupie połowa pacjentów posiada pełne obserwacje czasu do remisji, natomiast pozostała połowa danych jest cenzurowana.

```
# Grupa A
full_A = np.array([0.03345514, 0.08656403, 0.08799947, 0.24385821, 0.27755032,
                   0.40787247, 0.58825664, 0.64125620, 0.90679161, 0.94222208])
censored_A = np.ones(10)  # wartości dla danych cenzurowanych
data_A = np.concatenate([full_A, censored_A])
delta_A = np.array([True]*len(full_A) + [False]*len(censored_A))

# Grupa B
full_B = np.array([0.03788958, 0.12207257, 0.20319983, 0.24474299, 0.30492413,
                   0.34224462, 0.42950144, 0.44484582, 0.63805066, 0.69119721])
censored_B = np.ones(10)  # wartości dla danych cenzurowanych
data_B = np.concatenate([full_B, censored_B])
delta_B = np.array([True]*len(full_B) + [False]*len(censored_B))
```

```
df_A = pd.DataFrame({"Data": data_A, "Delta": delta_A})
df_B = pd.DataFrame({"Data": data_B, "Delta": delta_B})
```

Dla obu grup generujemy zestawienia statystyk opisowych w formie tabel 7 oraz 8, które pozwalają zobrazować podstawowe właściwości rozkładu czasów remisji oraz stopień cenzurowania danych.

Tabela 7: Liczba danych dla obu grup

	Liczba danych	Liczba pełnych	Liczba cenz.
Grupa A	20	10	10
Grupa B	20	10	10

Tabela 8: Statystyki dla grupy A i grupy B

	Min	Q1	Mediana	Q3	Max	IQR
Grupa A	0.033	0.127	0.343	0.628	0.942	0.501
Grupa B	0.038	0.214	0.324	0.441	0.691	0.227

Interpretacja wyników

W obu grupach liczba pełnych obserwacji była taka sama (10 na 20 pacjentów), jednak rozkład czasów remisji różni się między lekami.

Grupa A (lek A) charakteryzuje się:

- większą zmiennością czasów remisji (szerszy rozstęp międzykwartylowy)
- szerszym zakresem obserwowanych wartości (niższe minimum i wyższe maksimum)
- wyższą medianą, co sugeruje dłuższy typowy czas remisji.

Grupa B (lek B) natomiast:

- wykazuje mniejszą zmienność czasów remisji (bardziej skupione obserwacje)
- ma mniejszy rozrzut wartości
- cechuje się niższą medianą, co sugeruje szybszą przeciętną reakcję na leczenie

Lek B wydaje się działać szybciej i w sposób bardziej przewidywalny, prowadząc do remisji w średnio krótszym czasie. Należy jednak pamiętać, że dane są cenzurowane, a wielkość próby niewielka, dlatego nie można wyciągać jednoznacznych wniosków co do skuteczności leków.

Estymacja średniego czasu remisji choroby dla leków A i B

Celem jest oszacowanie średniego czasu do remisji choroby u pacjentów leczonych lekami A oraz B. Ponieważ obserwację zakończono po roku, w tym samym ustalonym momencie $t_0 = 1$, mamy do czynienia z danymi cenzurowanymi prawostronnie I-go typu.

Zakładamy, że czas do remisji X_i ma rozkład wykładniczy o gęstości

$$f(x) = \vartheta e^{-\vartheta x}, \quad x > 0,\ \vartheta > 0$$

Wtedy średni czas do remisji wynosi

$$\mu = \frac{1}{\vartheta}$$

Wyznaczymy punktowe oszacowania największej wiarogodności, oszacowania przy pomocy prostszego estymatora opartego na zmiennej losowej R oznaczającej liczbę obserwacji niecenzurowanych, oraz realizację zbioru ufności opartą na tym uproszczonym estymatorze punktowym.

Estymator największej wiarogodności

Dla danych prawostronnie cenzurowanych I-go typu funkcja wiarogodności jest dana wzorem

$$L(\vartheta; t^*) = \frac{n!}{(n-r)!} \prod_{i=1}^{r} f_\vartheta(x_{(i)}) \left[1 - F_\vartheta(t_0)\right]^{n-r},$$

gdzie $x_{(1)} < x_{(2)} < \cdots < x_{(r)}$, f_ϑ to gęstość, F_ϑ – dystrybuanta rozkładu zmiennych $X_1, \ldots, X_n$, a r jest realizacją zmiennej losowej R.

Estymatorem największej wiarogodności (NW) parametru ϑ jest wtedy statystyka postaci

$$\hat{\vartheta} = \frac{R}{T_1},$$

gdzie

$$T_1 = \sum_{i=1}^{R} X_{(i)} + t_0(n - R)$$

Poniżej przedstawiamy kod oraz estymację parametru $\mu = \frac{1}{\vartheta}$ na podstawie posiadanych danych.

```
r = 10 # liczba remisji
n = 20 # liczba pacjentów
t0 = 1 # czas cenzurowania

# Suma czasów remisji dla leków A i B
sum_A = full_A.sum()
sum_B = full_B.sum()

# Estymacja NW
T_A = sum_A + t0 * (n - r)
theta_A = r / T_A
mu_A = 1 / theta_A

T_B = sum_B + t0 * (n - r)
theta_B = r / T_B
mu_B = 1 / theta_B

print(f"Lek A: Średnia = {mu_A}")
print(f"Lek B: Średnia = {mu_B}")
```

```
Lek A: Średnia = 1.4215826169999999
Lek B: Średnia = 1.3458668850000002
```

Alternatywny estymator punktowy

Prostszy estymator parametru ϑ można uzyskać, korzystając jedynie ze zmiennej losowej R. Ponieważ $R \sim \mathcal{B}(n, F_\vartheta(t_0))$, prawdopodobieństwo sukcesu wynosi

$$p = 1 - \exp(-\vartheta t_0).$$

Przyjmując za estymator p statystykę

$$\tilde{p} = \frac{R}{n},$$

otrzymujemy alternatywny estymator parametru ϑ w postaci

$$\tilde{\vartheta} = -\frac{\log\left(1 - \frac{R}{n}\right)}{t_0}$$

Poniżej przedstawiamy kod oraz estymację parametru $\mu = \frac{1}{\vartheta}$ na podstawie posiadanych danych.

```
r = 10 # liczba remisji
n = 20 # liczba pacjentów
t0 = 1 # czas cenzurowania

theta_A_exp = -np.log(1 - r / n) / t0
mu_A_exp = 1 / theta_A_exp

theta_B_exp = -np.log(1 - r / n) / t0
mu_B_exp = 1 / theta_B_exp

print(f"Lek A: Średnia = {mu_A_exp}")
print(f"Lek B: Średnia = {mu_B_exp}")
```

```
Lek A: Średnia = 1.4426950408889634
Lek B: Średnia = 1.4426950408889634
```

Zauważmy, że w tym przypadku estymacje zależą wyłącznie od wartości r, n oraz t_0, w związku z czym średnia w naszym przykładzie będzie taka sama dla obu leków, wynosząc w przybliżeniu ≈ 1.44.
Dlatego zastosowanie tego estymatora w tym kontekście nie ma sensu.

Estymacja przedziałowa

Zbiór ufności dla parametru ϑ można skonstruować w oparciu o estymator

$$\tilde{\vartheta} = -\frac{\log(1 - \frac{R}{n})}{t_0},$$

który zależy wyłącznie od zmiennej losowej R.

Ponieważ $R \sim \mathcal{B}(n, F_\vartheta(t_0))$, mając przedział ufności $[T_L, T_U]$ dla prawdopodobieństwa sukcesu

$$p = F_\vartheta(t_0) = 1 - \exp(-\vartheta t_0),$$

na poziomie ufności $1 - \alpha$, można go przekształcić w przedział ufności dla parametru ϑ:

$$[\tilde{T}_L, \tilde{T}_U] = \left[-\frac{\ln(1 - T_L)}{t_0}, -\frac{\ln(1 - T_U)}{t_0}\right],$$

również na poziomie ufności $1 - \alpha$.

Poniżej przedstawiamy realizację tego przedziału ufności dla parametru $\mu = \frac{1}{\vartheta}$ na podstawie posiadanych danych.
Do obliczeń wykorzystamy przedział ufności dla paramtru p w rozkładzie Bernoulliego, dostępny w pakiecie *scipy*.

```
r = 10  # liczba sukcesów
n = 20  # liczba prób
alphas = [0.05, 0.01] # poziomy istotności

result = binomtest(r, n)

for alpha in alphas:
    # Przedział ufności dla p
    confidence_level = 1 - alpha
    ci = result.proportion_ci(confidence_level=confidence_level)
    # Transformacja do przedziału
    T_L = -np.log(1 - ci.low)
    T_R = -np.log(1 - ci.high)

    print(f"Poziom istotności {1-alpha}: [{1/T_R}, {1/T_L}]")
```

```
Poziom istotności 0.95: [0.7679853394899884, 3.1506350087666912]
Poziom istotności 0.99: [0.6559875341960035, 4.072031273887426]
```

Zauważmy, że przedziały ufności są identyczne dla obu leków.

Oszacowanie parametrów przy cenzurowaniu II-go typu

Załóżmy teraz, że obserwacje czasu do remisji choroby były prowadzone do momentu, w którym remisja zostanie zaobserwowana u dziesięciu pacjentów.
W takim przypadku mamy do czynienia z danymi cenzurowanymi prawostronnie II-go typu.

Ponownie wyznaczymy punktowe oszacowania największej wiarogodności oraz realizację zbioru ufności, tym razem opartą na estymatorze NW.

Estymator największej wiarogodności

Dla danych prawostronnie cenzurowanych II-go typu funkcja wiarogodności przyjmuje postać

$$L(\vartheta; t) = \frac{n!}{(n-m)!} \prod_{i=1}^{m} f_\vartheta(x_{(i)}) \left[1 - F_\vartheta(x_{(m)})\right]^{n-m},$$

gdzie $0 < x_{(1)} < x_{(2)} < \cdots < x_{(m)} < \infty$, f_ϑ oznacza gęstość, F_ϑ - dystrybuantę rozkładu zmiennych $X_1, \ldots, X_n$, a m jest ustalone.

Wówczas estymatorem największej wiarogodności (NW) parametru ϑ jest statystyka

$$\hat{\vartheta} = \frac{m}{T_2},$$

gdzie

$$T_2 = \sum_{i=1}^{m} X_{(i)} + (n-m)X_{(m)}.$$

Poniżej przedstawiamy kod oraz estymację parametru $\mu = \frac{1}{\vartheta}$ na podstawie posiadanych danych.

```
# Liczba obserwacji niecenzurowanych
m = 10

# Estymacja dla leku A
T_A = full_A.sum() + m * full_A[-1]
theta_A = m / T_A
print("Średni czas do remisji (A):", 1 / theta_A)

# Estymacja dla leku B
T_B = full_B.sum() + m * full_B[-1]
theta_B = m / T_B
print("Średni czas do remisji (B):", 1 / theta_B)
```

```
Średni czas do remisji (A): 1.363804697
Średni czas do remisji (B): 1.037064095
```

Estymacja przedziałowa

Do konstrukcji przedziału ufności dla parametru ϑ wykorzystujemy fakt, że zmienna losowa

$$\frac{\vartheta \sum_{i=1}^{m} D_i}{m} = \frac{\sum_{i=1}^{m} D_i}{m\mu},$$

gdzie

$$D_i = (n-i+1)(X_{(i)} - X_{(i-1)}), \quad i = 1, \ldots, m,$$

ma rozkład gamma $\mathcal{G}(m, 1/m)$ i może służyć jako funkcja centralna w konstrukcji przedziałów ufności.

Niech $q_m(p)$ oznacza kwantyl rzędu p rozkładu gamma $G(m, 1/m)$. Wówczas przedział ufności dla ϑ na poziomie ufności $1-\alpha$ można zapisać jako

$$[T_L, T_U] = \left[\frac{m q_m(\alpha_1)}{\sum_{i=1}^{m} D_i}, \frac{m q_m(1-\alpha_2)}{\sum_{i=1}^{m} D_i} \right].$$

Poniżej przedstawiamy realizację tego przedziału ufności dla parametru $\mu = \frac{1}{\vartheta}$ na podstawie posiadanych danych.
Do obliczeń wykorzystamy kwantyle rozkładu gamma obliczone przy pomocy pakietu *scipy*.

```
# Liczba obserwacji niecenzurowanych
m = 10

# Poziomy istotności
alphas = [0.05, 0.01]

def ci_gamma(T, m, alpha):
    T_L = m * scipy.stats.gamma.ppf(alpha/2, m, scale=1/m) / T
    T_U = m * scipy.stats.gamma.ppf(1 - alpha/2, m, scale=1/m) / T
    return T_L, T_U

groups = {"A": T_A, "B": T_B}

for group, T in groups.items():
    print(f"Przedziały ufności dla leku {group}:")
    for alpha in alphas:
        T_L, T_U = ci_gamma(T, m, alpha)
        print(f"Poziom istotności {1-alpha}: [{1/T_U}, {1/T_L}]")
    if (group == "A"):
        print("\n")
```

```
Przedziały ufności dla leku A:
Poziom istotności 0.95: [0.7982560062092574, 2.8439919752489153]
Poziom istotności 0.99: [0.6819561154044388, 3.669177477392265]


Przedziały ufności dla leku B:
Poziom istotności 0.95: [0.6070096726303604, 2.162627809162604]
Poziom istotności 0.99: [0.5185729329187228, 2.7901152037066144]
```

Symulacyjne porównanie estymatorów parametru ϑ dla danych prawostronnie cenzurowanych I-go typu

Niech $X_1, \ldots, X_n$ będą niezależnymi zmiennymi losowymi o rozkładzie wykładniczym $\mathcal{E}(\vartheta)$, a t_0 ustalonym czasem obserwacji. Naszym celem jest porównanie estymatorów parametru

ϑ dla danych prawostronnie cenzurowanych I-go typu. Rozważamy estymator największej wiarygodności (NW):

$$\hat{\vartheta} = \frac{R}{T_1},$$

oraz prosty estymator:

$$\tilde{\vartheta} = -\frac{\log(1 - \frac{R}{n})}{t_0}.$$

W celu porównania dokładności obu estymatorów przeprowadziliśmy symulacje dla parametrów:

$$\vartheta = 1, \quad n = 10, 30, \quad t_0 = 0.5, 1, 2,$$

z $k = 10000$ powtórzeniami. Dla każdej kombinacji obliczyliśmy średni bias oraz błąd średniokwadratowy:

```
# Parametry symulacji
theta = 1
n_values = [10, 30]
t0_values = [0.5, 1, 2]
k = 10000  # liczba powtórzeń

results = []

for n in n_values:
    for t0 in t0_values:
        bias_NW = 0
        mse_NW = 0
        bias_simple = 0
        mse_simple = 0

        for _ in range(k):
            while True:
                data = type_I(t0, theta, 1, n)
                X_c = data.Data
                delta = data.Delta
                R = delta.sum()
                if R != n:
                    break

            # Estymator NW
            X_c = sorted(X_c)
            T1 = sum(X_c[:R]) + t0*(n - R)
            theta_NW = R / T1
```

```
            bias_NW += theta_NW - theta
            mse_NW += (theta_NW - theta)**2

            # Prosty estymator
            theta_simple = -np.log(1 - R / n) / t0
            bias_simple += theta_simple - theta
            mse_simple += (theta_simple - theta)**2

        bias_NW /= k
        mse_NW /= k
        bias_simple /= k
        mse_simple /= k

        results.append({
            "n": int(n),
            "t0": t0,
            "Bias_NW": bias_NW,
            "MSE_NW": mse_NW,
            "Bias_simple": bias_simple,
            "MSE_simple": mse_simple
        })

df_results = pd.DataFrame(results)
```

Wyniki symulacji przedstawiamy w tabeli 9

Tabela 9: Wyniki symulacji dla $\vartheta = 1$

n	t_0	bias_NW	MSE_NW	Bias_simple	MSE_simple
10	0.5	0.0739667	0.305753	0.0854036	0.334649
10	1	0.0576342	0.185804	0.0828919	0.22627
10	2	-0.0501649	0.0800284	-0.0763162	0.0612459
30	0.5	0.023567	0.0925241	0.0254824	0.0954239
30	1	0.0219996	0.0572527	0.0314326	0.0670109
30	2	0.0237012	0.0412222	0.0526667	0.0672316

gdzie

$$\text{MSE} = \mathbb{E}(X - \vartheta)^2$$

to błąd średniokwadratowy, natomiast

$$\text{Bias} = \mathbb{E}[X - \theta]$$

oznacza średnie odchylenie estymatora od prawdziwej wartości ϑ. Natomiast _NW oraz _simple odnoszą się odpowiednio do estymatora największej wiarygodności oraz do prostszego estymatora.

Wnioski

Możemy zauważyć, że:

- Estymator największej wiarogodności wykazuje mniejszy bias i niższe MSE w porównaniu do prostszego estymatora.
- Wzrost liczby prób n prowadzi do zmniejszenia biasu i MSE dla obu estymatorów.
- Wzrost czasu obserwacji t_0 powoduje zmniejszenie MSE dla obu estymatorów, natomiast nie ma praktycznie wpływu na bias.
- Prostszy estymator daje bardzo dobre wyniki i, mimo pewnej utraty informacji, nie odbiega znacząco od estymatora NW.

Test ilorazu wiarogodności dla danych cenzurowanych I-go typu

Rozważmy n niezależnych zmiennych losowych $X_1, \dots, X_n$ o rozkładzie wykładniczym z parametrem $\vartheta > 0$.
Gęstość tego rozkładu ma postać:

$$f_\vartheta(x) = \vartheta e^{-\vartheta x} \cdot \mathbf{1}_{(0,\infty)}(x),$$

natomiast dystrybuanta:

$$F_\vartheta(x) = (1 - e^{-\vartheta x}) \cdot \mathbf{1}_{(0,\infty)}(x).$$

Zakładamy, że mamy do czynienia z danymi **cenzurowanymi prawostronnie I-go typu** przy czasie cenzurowania $t_0 > 0$.

Niech:

- R – liczba pełnych obserwacji,
- $S = \sum_{i=1}^{R} X_{(i)}$ – suma czasów niecenzurowanych,
- $T_1 = S + (n - R)t_0$ – całkowity czas testowania.

Wówczas funkcja wiarogodności dla danych cenzurowanych I-go typu ma postać:

$$L(\vartheta; t^*) = \frac{n!}{(n-r)!}\,\vartheta^r \exp\left(-\vartheta\left[\sum_{i=1}^{r} x_{(i)} + t_0(n-r)\right]\right),$$

gdzie r i s są realizacjami zmiennych losowych R i S.

Estymator największej wiarogodności parametru ϑ ma postać:

$$\widehat{\vartheta} = \frac{R}{T_1}.$$

Zdefiniujmy statystykę:

$$\lambda(R,S) = \frac{\sup\limits_{\vartheta\in\Theta_0} L(\vartheta; R, S)}{\sup\limits_{\vartheta\in\Theta} L(\vartheta; R, S)}.$$

Zauważmy, że jeśli $\widehat{\vartheta} \in \Theta_0$, to $\lambda(R,S) = 1$

Z twierdzenia Wilksa wynika, że przy założeniu prawdziwości hipotezy zerowej H_0 statystyka

$$-2\ln\lambda(R,S)$$

ma asymptotyczny rozkład $\chi^2(1)$.

Zatem wartość p-value możemy obliczyć jako:

$$p\text{-value} = 1 - F_{\chi^2(1)}(-2\ln\lambda(r,s)),$$

gdzie $F_{\chi^2(1)}$ oznacza dystrybuantę rozkładu χ^2 z jednym stopniem swobody.

Hipotezę zerową H_0 odrzucamy, jeśli:

$$p\text{-value} < \alpha,$$

gdzie α to przyjęty poziom istotności.

Poniżej przedstawiamy funkcję w języku Python realizującą test ilorazu wiarogodności dla hipotez:

- $H_0 : \theta = \theta_0$ (dwustronny),
- $H_0 : \theta \leq \theta_0$ (prawostronny),
- $H_0 : \theta \geq \theta_0$ (lewostronny).

```python
def test(r, s, n, t0, theta0, hipoteza):
    T1 = (s + (n - r) * t0)
    theta_hat = r / T1
    L = theta_hat**r * np.exp(-theta_hat*T1)
    L0 = theta0**r * np.exp(-theta0*T1)
    log_lambda = np.log(L0/L)
    if hipoteza == 'dwustronna':
        p_value = 1 - scipy.stats.chi2.cdf(-2 * log_lambda, df=1)

    elif hipoteza == 'prawostronna':
        if theta_hat <= theta0:
            p_value = 1.0
        else:
            p_value = 1 - scipy.stats.chi2.cdf(-2 * log_lambda, df=1)

    elif hipoteza == 'lewostronna':
        if theta_hat >= theta0:
            p_value = 1.0
        else:
            p_value = 1 - scipy.stats.chi2.cdf(-2 * log_lambda, df=1)

    return p_value
```

Symulacyjna estymacja mocy i rozmiaru testu przy wybranych alternatyw

Przeprowadziliśmy symulacje w celu oszacowania rozmiaru testu oraz jego mocy przy wybranych alternatywach dla hipotezy zerowej $H_0 \colon \vartheta = \vartheta_0$.

Przyjeliśmy czas eksperymentu $t_0 = 4$, liczności próby $n \in \{20, 50\}$, wartość parametru $\vartheta_0 = 1/4$ oraz alternatywy

$$\vartheta \in \{0.05,\ 0.1,\ 0.15,\ 0.2,\ 0.25,\ 0.4,\ 0.5,\ 0.7,\ 0.9,\ 1.1\}.$$

Dla każdej konfiguracji parametrów postępowaliśmy według następującego algorytmu:

- Generujemy $M = 10000$ niezależnych prób z rozkładu wykładniczego $\mathrm{Exp}(\vartheta)$, prawostronnie cenzurowanych I-go typu przy czasie eksperymentu t_0.
- Obliczamy wartości p-value z wykorzystaniem wcześniej zdefiniowanego testu.

- Na podstawie uzyskanych wartości p-value podejmujemy decyzję o odrzuceniu lub braku podstaw do odrzucenia hipotezy zerowej na poziomie istotności $\alpha = 0.05$.

Rozmiar testu oszacowaliśmy jako częstość odrzucenia hipotezy zerowej H_0 w symulacjach przeprowadzonych dla $\vartheta = \vartheta_0$.

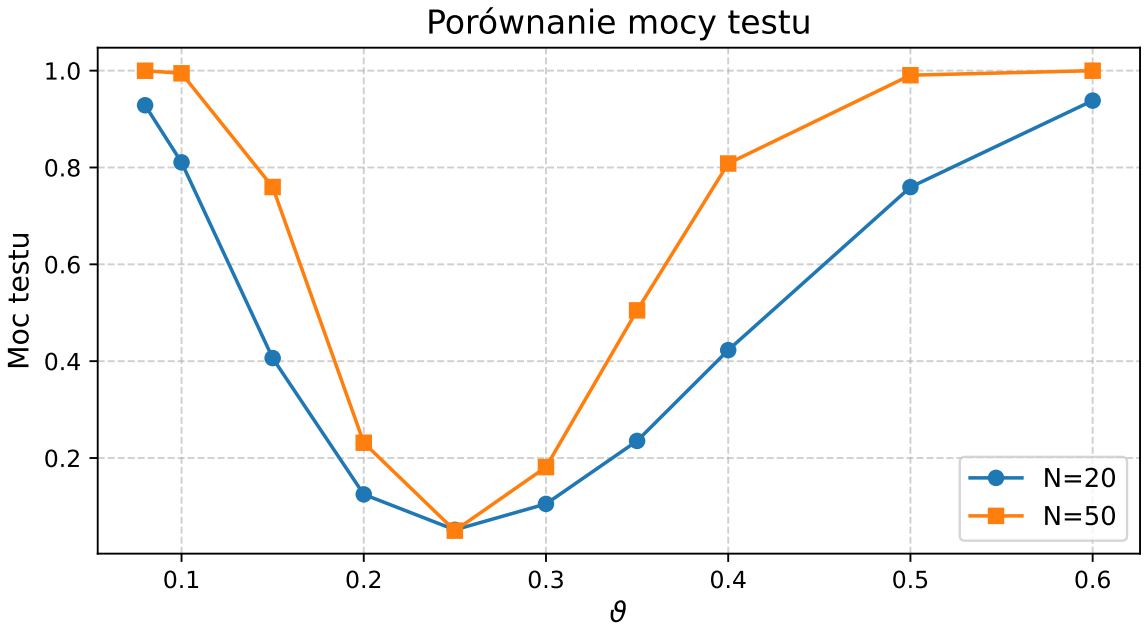
Moc testu przy alternatywach oszacowaliśmy analogicznie dla dziesięciu wybranych wartości parametru $\vartheta \neq \vartheta_0$.

Ostateczne wyniki symulacji przedstawiliśmy w tabeli 10 oraz na wykresie 3, które ilustrują zależność mocy testu od wartości parametru ϑ.

```
theta0 = 0.25
t0 = 4
n_values = [20, 50]
theta_values = [0.08, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.5, 0.6]
M = 10000
alpha = 0.05
results = {}
size = {}
for n in n_values:
    power = []
    for theta in theta_values:
        rejections = 0
        for _ in range(M):
            X = type_I(t0=t0, L=theta, alpha=1, n=n)
            r = np.sum(X.Delta)
            s = np.sum(X.Data * X.Delta)
            p_val = test(r, s, n, t0, theta0, 'dwustronna')
            if p_val < alpha:
                rejections += 1
        power.append(rejections / M)
        if theta == theta0:
            size[n] = rejections / M
    results[n] = power
```

Tabela 10: Rozmiar testu

n=20	n=50
0.0517	0.05



Rysunek 3: Porównanie mocy testu dla $n = 20$ i $n = 50$

Na wykresie 3 widać wyraźną różnicę w mocy testu dla prób o liczności $n = 20$ i $n = 50$. Test oparty na większej próbie ($n = 50$) osiąga wyższe wartości mocy dla tych samych wartości parametru ϑ, co oznacza, że skuteczniej wykrywa odchylenia od hipotezy zerowej ϑ_0.

Dla obu wielkości próby moc testu jest minimalna w punkcie $\vartheta_0 = 0.25$, co jest zgodne z oczekiwaniami, w tym punkcie hipoteza zerowa jest prawdziwa, a test powinien ją odrzucać z prawdopodobieństwem zbliżonym do poziomu istotności $\alpha = 0.05$. Jest to rozmiar naszego testu, a dokładne wartości dla różnych prób przedstawiamy w tabeli 10. Rzeczywiście, rozmiary testu są bliskie wartości α.

Ponadto wykres wskazuje, że moc testu rośnie wraz z oddaleniem wartości ϑ od ϑ_0, co jest zgodne z intuicją: im większa różnica między rozkładami, tym łatwiej testowi odrzucić hipotezę zerową.

Weryfikacja hipotezy o średnim czasie do remisji

Celem analizy jest weryfikacja hipotezy, że średni czas do remisji choroby w grupie A oraz w grupie B wynosi jeden rok. Dane wykorzystane w badaniu są danymi cenzurowanymi I-go typu. Przyjmujemy $t_0 = 1$, ponieważ obserwacje obejmowały okres jednego roku, oraz $\vartheta_0 = 1$, zgodnie z hipotezą zerową.

W analizie zastosowaliśmy test dwustronny przy poziomie istotności $\alpha = 0.05$.
Poniżej przedstawiamy kod oraz uzyskane wartości p-value testu dla obu grup.

```
r_A = np.sum(delta_A)
s_A = np.sum(data_A * delta_A)
n_A = len(data_A)
t0 = 1
theta0 = 1

p_val_A = test(r_A, s_A, n_A, t0, theta0, hipoteza='dwustronna')

r_B = np.sum(delta_B)
s_B = np.sum(data_B * delta_B)
n_B = len(data_B)

p_val_B = test(r_B, s_B, n_B, t0, theta0, hipoteza='dwustronna')
```

Tabela 11: p-value testów hipotezy $\vartheta = 1$

Grupa	p-value
A	0.237355
B	0.323047

Z tabeli 11 można wywnioskować, że w obu przypadkach brak jest podstaw do odrzucenia hipotezy zerowej, ponieważ wartości p-value są większe od przyjętego poziomu istotności α. Oznacza to, że możemy przyjąć hipotezę, że średni czas do remisji choroby w grupie A oraz w grupie B wynosi jeden rok.

Zadania dodatkowe

1.6

```
def ew_mc(alpha, beta, gamma, n=200000):
    np.random.seed(37)
    u = np.random.rand(n)
    samples = np.array([F_inv(ui, alpha, beta, gamma) for ui in u])
    mean = samples.mean()
    sd = samples.std()
```

```
    return mean, sd

alpha_1, beta_1, gamma_1 = 2, 1, 2

alpha_2, beta_2, gamma_2 = 2, 1, 0.25

mean_1, sd_1 = ew_mc(alpha_1, beta_1, gamma_1)
mean_2, sd_2 = ew_mc(alpha_2, beta_2, gamma_2)

print(f"alfa: {alpha_1}, beta: {beta_1}, gamma: {gamma_1} \
-> E[X] = {mean_1:.4f}, sd = {sd_1:.4f}")
print(f"alfa: {alpha_2}, beta: {beta_2}, gamma: {gamma_2} \
-> E[X] = {mean_2:.4f}, sd = {sd_2:.4f}")
```

```
alfa: 2, beta: 1, gamma: 2 -> E[X] = 1.1443, sd = 0.4326
alfa: 2, beta: 1, gamma: 0.25 -> E[X] = 0.4026, sd = 0.4318
```

3.4

Rozwijając sumę otrzymujemy

$$\sum_{i=1}^{m} D_i = \sum_{i=1}^{m} (n-i+1)X_{(i)} - \sum_{i=1}^{m} (n-i+1)X_{(i-1)}.$$

Zmieniając zmienną sumowania $j = i-1$ oraz korzystając z faktu, że $X_{(0)} = 0$, mamy

$$\sum_{i=1}^{m} D_i = \sum_{i=1}^{m} (n-i+1)X_{(i)} - \sum_{j=1}^{m-1} (n-j)X_{(j)}.$$

Dla $i = 1, \ldots, m-1$ współczynnik przy $X_{(i)}$ wynosi

$$(n-i+1) - (n-i) = 1,$$

natomiast dla $i = m$ jest to $n-m+1$. Zatem

$$\sum_{i=1}^{m} D_i = \sum_{i=1}^{m-1} X_{(i)} + (n-m+1)X_{(m)} = \sum_{i=1}^{m} X_{(i)} + (n-m)X_{(m)}.$$

Otrzymujemy więc

$$T_2 = \sum_{i=1}^{m} X_{(i)} + (n-m)X_{(m)} = \sum_{i=1}^{m} D_i.$$