Analiza Przeżycia

Raport 1

Wiktor Niedźwiedzki (258882) Filip Michewicz (282239)

2 listopada 2025 Anno Domini

Spis treści

Spis wykresów

Spis tabel

1 Lista 1

Lista pierwsza obejmuje analizę rozszerzonego rozkładu Weibulla $\mathcal{EW}(\alpha, \beta, \gamma)$; definicje jego funkcji, generowanie danych, wizualizację oraz porównanie statystyk empirycznych i teoretycznych.

1.1 Zadanie 1

Zadanie polega na deklaracji funkcji:

- gęstości,
- dystrybuanty,
- kwantylową,
- hazardu,

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

1.1.1 Funkcja gęstości

Na wykładzie rozstała podana funkcja gęstości rozkładu $\mathcal{EW}(\alpha, \beta, \gamma)$ wyrażana jest wzorem:

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

Odpowiadający tej funkcji gęstości kod w języku Python ma postać:

```
def EW_density(x, alpha, beta, gamm):
    return(alpha * gamm / beta * (x / beta)**(alpha - 1)*
           (1 - np.exp(-(x / beta)**alpha))**(gamm - 1)*
           np.exp(-(x / beta)**alpha))
```

1.1.2 Dystrybuanta

Na wykładzie została podana dystrybuanta rozkładu $\mathcal{EW}(\alpha, \beta, \gamma)$, która wyrażona jest wzorem:

$$F(t) = \left(1-\exp\left(-\left(\frac{x}{\beta}\right)^{\alpha}\right)\right)^{\gamma}\mathbf{1}_{(0;\infty)}(t)$$

Odpowiadający tej dystrybuancie kod w języku Python przedstawiono poniżej.

```
def EW_distribution(t, alpha, beta, gamm):
    return (1 - np.exp(-(t / beta)**alpha))**gamm
```

1.1.3 Funkcja kwantylowa (dystrybuanta odwrotna)

Funkcja kwantylowa została wyznaczona z dystrybuanty jako funkcja do niej odwrotna. Oznacza to, że dla danej wartości prawdopodobieństwa $p \in (0,1)$ szukamy takiej wartości t, dla której $F(t) = p$.

Punktem wyjścia jest dana dystrybuanta: $F(t) = \left(1 - \exp\left(-\left(\frac{t}{\beta}\right)^{\alpha}\right)\right)^{\gamma}$

Podstawiamy $F(t) = p$ i rozwiązujemy równanie względem t:

$$\begin{aligned}
p &= \left(1 - \exp\left(-\left(\frac{t}{\beta}\right)^{\alpha}\right)\right)^{\gamma} \\
p^{1/\gamma} &= 1 - \exp\left(-\left(\frac{t}{\beta}\right)^{\alpha}\right) \\
\exp\left(-\left(\frac{t}{\beta}\right)^{\alpha}\right) &= 1 - p^{1/\gamma} \\
\left(\frac{t}{\beta}\right)^{\alpha} &= -\ln\left(1 - p^{1/\gamma}\right) \\
t^{\alpha} &= \beta^{\alpha}\left(-\ln\left(1 - p^{1/\gamma}\right)\right) \\
t &= \beta\left[-\ln(1 - p^{1/\gamma})\right]^{1/\alpha} \\
Q(p) &= \beta\left[-\ln(1 - p^{1/\gamma})\right]^{1/\alpha}
\end{aligned}$$

Otrzymana funkcja jest funkcją kwantylową (odwrotnością dystrybuanty) rozkładu $\mathcal{EW}(\alpha, \beta, \gamma)$ i pozwala dla zadanego prawdopodobieństwa p wyznaczyć wartość dystrybuanty t.

Odpowiadający tej funkcji kantylowej kod w języku Python ma postać:

```
def EW_quantile(p, alpha, beta, gamm):
    return beta * (-np.log(1.0 - p**(1.0 / gamm)))**(1.0 / alpha)
```

1.1.4 Funkcja hazardu

Funkcja hazardu rozkładu $\mathcal{EW}(\alpha, \beta, \gamma)$ wyrażona jest wzorem:

$$h(x) = \frac{f(x)}{1-F(x)} = \frac{\alpha\gamma\left(\frac{x}{\beta}\right)^{\alpha-1}\left(1-\exp\left(-\left(\frac{x}{\beta}\right)^{\alpha}\right)\right)^{\gamma-1}\exp\left(-\left(\frac{x}{\beta}\right)^{\alpha}\right)}{\beta\left(1-\left(1-\exp\left(-\left(\frac{x}{\beta}\right)^{\alpha}\right)\right)^{\gamma}\right)}\mathbf{1}_{(0;\infty)}(x)$$

Odpowiadający tej funkcji gęstości kod w języku Python ma postać:

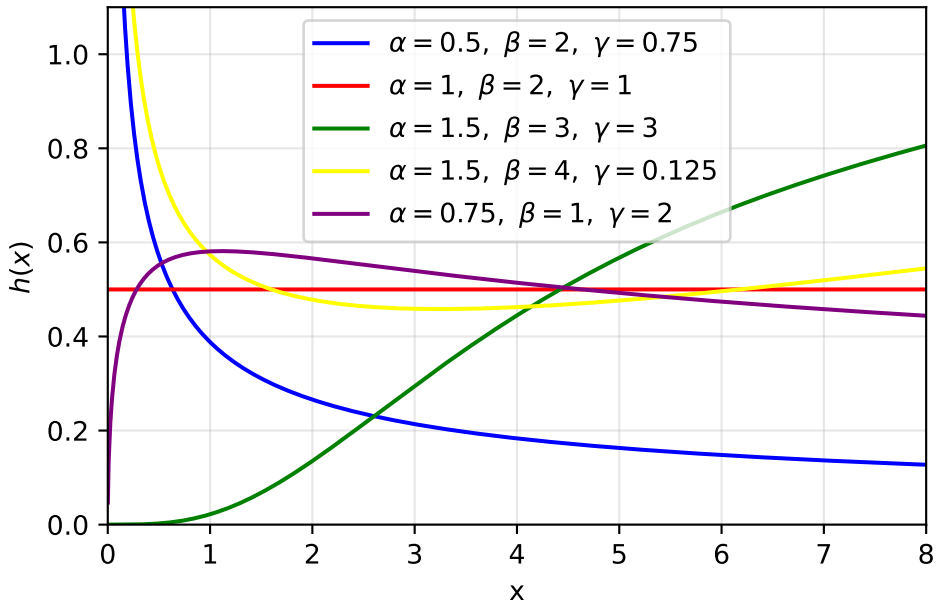
```
def EW_hazard(x, alpha, beta, gamm):
  f = EW_density(x, alpha, beta, gamm)
  F = EW_distribution(x, alpha, beta, gamm)
  if F == 1:
    return np.inf
  return f / (1 - F)
```

1.2 Zadanie 2

W tym zadaniu wygenerowano wykresy przykładowych funkcji hazardu rozkładu $\mathcal{EW}(\alpha, \beta, \gamma)$ dla 5 różnych trójek parametrów.

Użyte parametry są następujące:

- $\alpha = \frac{1}{2}, \beta = 2, \gamma = \frac{3}{4}$,
- $\alpha = 1, \beta = 2, \gamma = 1$,
- $\alpha = \frac{3}{2}, \beta = 3, \gamma = 3$,
- $\alpha = \frac{3}{2}, \beta = 4, \gamma = \frac{1}{8}$,
- $\alpha = \frac{3}{4}, \beta = 1, \gamma = 2$,



Wykres 1: Funkcja hazardu - rozkład rozszerzony Weibulla - przykładowe parametry

Wykres 1. przedstawia przykładowy przebieg funkcji hazardu rozkładu $\mathcal{EW}(\alpha, \beta, \gamma)$ dla pięciu różnych trójek parametrów. Widać, że w zależności od doboru parametrów funkcja hazardu może być malejąca, rosnąca, stała, unimodalna lub przyjmować kształt „wanny".

1.3 Zadanie 3

W tym zadaniu skonstruowano funkcję generującą zmienne losowe z rozkładu $\mathcal{EW}(\alpha, \beta, \gamma)$. W zadaniu 1 została wyznaczona funkcja kwantylowa $F^{-1}(p)$, dlatego korzystając z metody dystrybuanty odwrotnej można wygenerować zmienne losowe w następujący sposób:

1. Generujemy zmienną losową z rozkładu jednostajnego $U_i \sim \mathcal{U}(0, 1)$,
2. Obliczamy $X_i = F^{-1}(U_i)$.

Otrzymana zmienna losowa (X_i) ma rozkład jednoznacznie określony przez dystrybuantę $(F(t))$, czyli w naszym przypadku przez dystrybuantę rozkładu $\mathcal{EW}(\alpha, \beta, \gamma)$.

Poniżej przedstawiono kod w Pythonie realizujący ten algorytm.

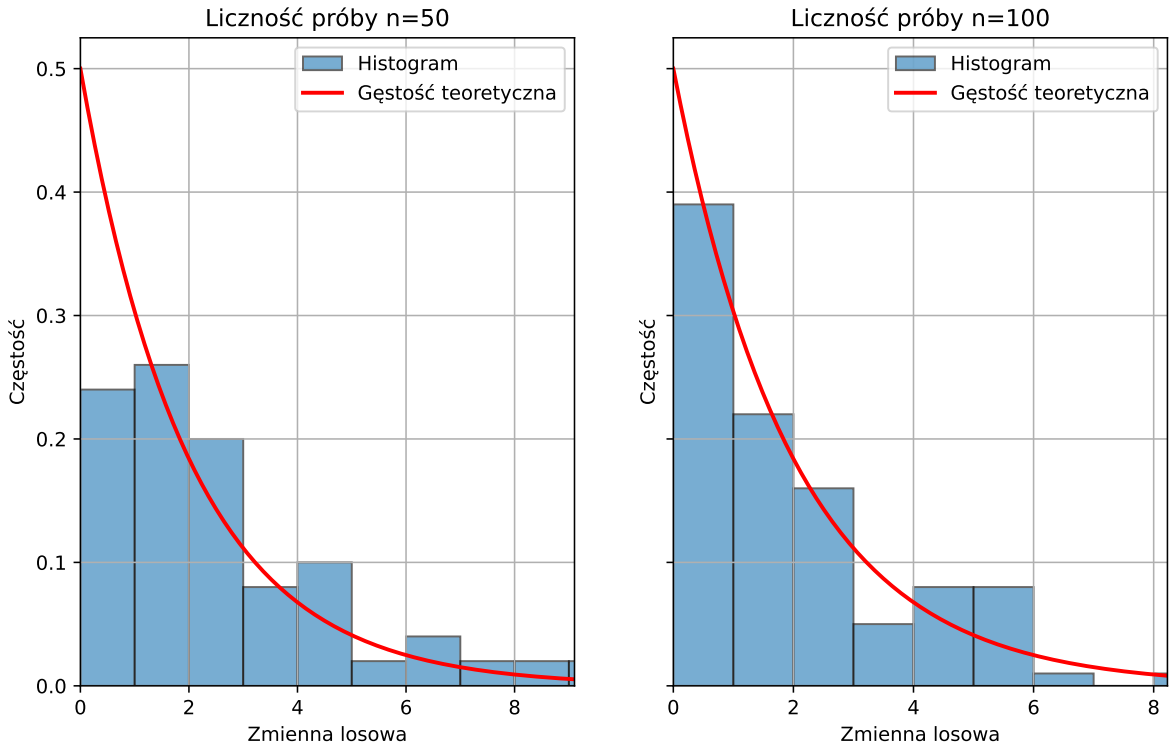
```
def EW_generator(alpha, beta, gamm, size=1):
  u = np.random.rand(size)
  return(EW_quantile(u, alpha, beta, gamm))
```

1.4 Zadanie 4

W tym zadaniu wygenerowano realizację próby z rozkładu $\mathcal{EW}(\alpha, \beta, \gamma)$ o liczności $n = 50$ oraz $n = 100$ dla następujących parametrów:

- $\alpha = 1, \beta = 2, \gamma = 1$,
- $\alpha = \frac{1}{2}, \beta = 1, \gamma = 4$,

Wyniki przedstawiono na poniższych wykresach.

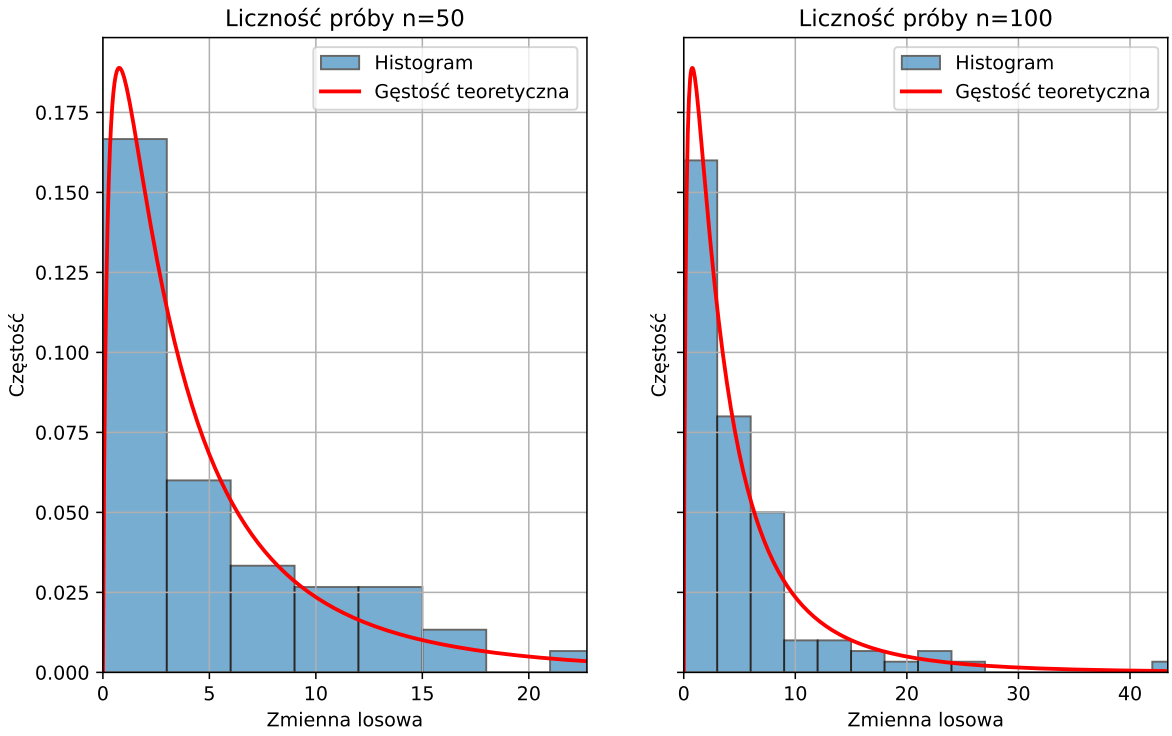


Wykres 2: Histogram i gęstość teoretyczna dla próby z rozkładu $\mathcal{EW}(\alpha = 1, \beta = 2, \gamma = 1)$ - liczność prób n = 50 oraz n = 100

Na Wykresie 2 widać, że histogram próby jest bardzo podobny do funkcji gęstości, jednak pewne różnice wynikają z małej liczności próby $n = 50$ - w tym przypadku Prawo Wielkich Liczb jeszcze nie zapewnia dokładnego przybliżenia rozkładu prawdopodobieństwa.

Dla większej próby $n = 100$ dopasowanie histogramu do gęstości wygląda lepiej, choć nadal widać drobne odchylenia.

Mimo to można stwierdzić, że kod poprawnie generuje zmienne losowe z rozkładu $\mathcal{EW}(\alpha, \beta, \gamma)$, co potwierdza wizualne porównanie histogramu z krzywą gęstości.



Wykres 3: Histogram i gęstość teoretyczna dla próby z rozkładu $\mathcal{EW}\left(\alpha = \frac{1}{2}, \beta = 1, \gamma = 4\right)$ - liczność prób n = 50 oraz n = 100

Na Wykresie 3., podobnie jak na Wykresie 2., histogram próby jest bardzo podobny do funkcji gęstości, choć nadal widać pewne odchylenia wynikające z ograniczonej liczności próby $n = 50$ lub $n = 100$.

Tym razem zakres wartości, z którego wylosowały się zmienne losowe, jest większy, jednak liczność próby pozostała bez zmian.

Mimo zmiany parametrów rozkładu, histogram nadal wizualnie przypomina funkcję gęstości, co stanowi dodatkowe potwierdzenie poprawnego działania algorytmu generującego zmienne losowe z rozkładu $\mathcal{EW}(\alpha, \beta, \gamma)$.

1.5 Zadanie 5

W tym zadaniu wyznaczono podstawowe statystyki opisowe (zarówno teoretyczne, jak i empiryczne) dla prób wygenerowanych w poprzednim zadaniu. Dodatkowo obliczono teoretyczne wartości niektórych statystyk opisowych odpowiadających wartościom empirycznym.

Tabela 1: Podstawowe statystyki opisowe prób z rozkładu $\mathcal{EW}(\alpha, \beta, \gamma)$

α	1	1	0.5	0.5
β	2	2	1	1
γ	1	1	4	4
Liczność próby	50	100	50	100
Średnia (emp.)	2.6213	2.089	5.3582	5.135
Mediana (emp.)	2.1432	1.5629	3.065	3.2231
Mediana (teor.)	1.3863	1.3863	3.379	3.379
Odchylenie standardowe	2.1937	1.8643	5.1906	6.2999
Kwartyl dolny (emp.)	1.0604	0.6573	1.3841	1.439
Kwartyl dolny (teor.)	0.5754	0.5754	1.5079	1.5079
Kwartyl górny (emp.)	3.8715	2.9642	8.1259	6.8508
Kwartyl górny (teor.)	2.7726	2.7726	7.1179	7.1179
Rozstęp	9.1031	8.2061	22.5351	43.2056
Rozstęp międzykwartylowy (emp.)	2.8111	2.3069	6.7418	5.4119
Rozstęp międzykwartylowy (teor.)	2.1972	2.1972	5.61	5.61
Minimum	0.0007	0.0233	0.1943	0.18
Maksimum	9.1038	8.2293	22.7295	43.3856

Tabeli 1 widać, że wartości statystyk empirycznych (średnia, mediana, odchylenie standardowe, kwartyle) w przybliżeniu odpowiadają wartościom teoretycznym, choć dla mniejszych prób $n = 50$ różnice są bardziej widoczne.

1.6 Zadanie dodatkowe 1

W tym zadaniu zadeklarowane zostaną funkcje do liczenia wartości oczekiwanej oraz wariancji zmiennej losowej o rozkładzie $\mathcal{EW}(\alpha, \beta, \gamma)$, który jest absolutnie ciągły. Z tego powodu możemy zdefiniować ogólny wzór na n-ty moment oraz wariancję jako:

$$\mathbb{E}[X^n] = \int_{\mathbb{R}} x^n f(x)\, dx,$$
$$Var[X] = \mathbb{E}[X^2] - (\mathbb{E}[X])^2$$

Niestety nie istnieją wzory analityczne na wyliczenie tych całek dla rozszerzonego rozkładu Weibulla, zatem otrzymane wartości będziemy porównywać z wbudowaną funkcją *scipy.integrate.quad*. Dla jak najlepszej dokładności wyników, przedziałem całkowania będzie $[0; Q(0.999)]$, gdzie $Q(\cdot)$ - funkcja kwantylowa. Całkę będziemy liczyć dwoma metodami: parabol oraz Monte Carlo.

1.6.1 Metoda parabol

```
def simpson_rule(a, b, function, n):

    if n%2 != 0: raise Exception("The number of intervals must be even.")

    x_points = np.linspace(a, b, n+1)
    y_points = []
    for i in x_points: y_points.append(function(i))

    h = (b-a)/(n)
    integral = 0
    integral += y_points[0]
    even = False

    for i in range(1, n):
        if even:
            integral += 2*y_points[i]
            even = False
        else:
            integral += 4*y_points[i]
            even = True

    integral += y_points[n]
    integral = h*integral/3

    return(integral)

def expected_value_simpson(alpha, beta, gamma, moment = 1):
    quan = EW_quantile(0.9999, alpha, beta, gamma)
    intervals = floor(quan/0.01)
    if intervals % 2 != 0: intervals += 1
    return(simpson_rule(0.001, quan,
           lambda x: x**moment * EW_density(x, alpha, beta, gamma), intervals))

def variance_simpson(alpha, beta, gamma):
    return(expected_value_simpson(alpha, beta, gamma, 2) -
           expected_value_simpson(alpha, beta, gamma)**2)
```

1.6.2 Metoda Monte Carlo

Kod

1.6.3 Wyniki

Tabelka z wynikami metod parabol i Monte Carlo oraz wyniki ze scipy.integrate.quad, dla pewności czy dobrze liczy można dobrać parametry tak, aby wyszedł rozkład wykładniczy (bądź rozszerzony wykładniczy) i porównać rzeczywiste wyniki (które możemy wyliczyć analitycznie).

Tabela 2: Wartości oczekiwane rozkładu $\mathcal{EW}(\alpha, \beta, \gamma)$ dla wybranych parametrów

Metoda	α=1, β=2, γ=1	α=0.5, β=1, γ=4	α=2, β=1, γ=1.5
Parabol	1.9979576819881102	5.750340743208503	1.0390899761365686
Monte Carlo	-	-	-
SciPy	1.9979576820089229	5.750340736207721	1.039089977151145

Tabela 3: Wariancje rozkładu $\mathcal{EW}(\alpha, \beta, \gamma)$ dla wybranych parametrów

Metoda	α=1, β=2, γ=1	α=0.5, β=1, γ=4	α=2, β=1, γ=1.5
Parabol	3.9660646806966393	53.12562122253677	0.19960274728522465
Monte Carlo	-	-	-
SciPy	3.966064514653979	53.12562130319005	0.19960274521615373

1.7 Zadanie dodatkowe 2

TO DO

2 Lista 2

Lista 2 obejmuje generowanie danych cenzurowanych z rozkładu uogólnionego wykładniczego $\mathcal{GE}(\lambda, \alpha)$ oraz opis statystyk opisowych zarówno dla wygenerowanych danych cenzurowanych, jak i dla podanych danych cenzurowanych.

2.1 Zadanie 1

W tym zadaniu zadeklarujemy funkcje do prawostronnego cenzurowania danych kompletnych: cenzurowanie typu I, typu II oraz losowe. Chociaż zadanie dotyczy głównie danych z rozkładu $\mathcal{GE}(\lambda, \alpha)$, funkcje te można łatwo rozszerzyć na prawostronnie cenzurowane dane z dowolnego rozkładu, podając zamiast parametrów $\mathcal{GE}(\lambda, \alpha)$ dane kompletne. Pozwoli to potencjalnie na analizę i symulację cenzurowanych danych w szerszym kontekście.

W kontekście zadania łatwo widać że rozkład uogólniony wykładniczy $\mathcal{GE}(\lambda, \alpha)$ jest szczególnym przypadkiem rozkładu wykładniczego Weibulla $\mathcal{EW}(\alpha, \beta, \gamma)$ gdzie,

$$\mathcal{GE}(\lambda, \alpha) = \mathcal{EW}(1, \lambda, \alpha)$$

Dlatego też dane z tego rozkładu można generować za pomocą funkcji z Listy 1. Zadania 3.

Podobnie rozkład wykładniczy $\mathcal{E}(\lambda)$ jest szczególnycm przypadkiem uogólnionego rozkładu wykładniczego $\mathcal{GE}(\lambda, \alpha)$, a tym samym wykładniczego rozkładu Weibulla $\mathcal{EW}(\alpha, \beta, \gamma)$ gdzie,

$$\mathcal{E}(\lambda) = \mathcal{GE}(\lambda, 1) = \mathcal{EW}(1, \lambda, 1)$$

Informację na temat danych cenzurowanych będziemy zawierać w tzw. indykatorze cenzurowania:

$$\delta_i = \begin{cases} 1, & \text{jeśli dane są kompletne (niecenzurowane),} \\ 0, & \text{jeśli dane są niekompletne (cenzurowane).} \end{cases}$$

2.1.1 Cenzurowanie I typu

Cenzurowanie typu I polega na niewyznaczaniu wartości zmiennej losowej powyżej pewnego czasu t_0. Oznacza to, że jeśli X_i przekracza t_0, to jest ona zastępowana przez t_0.

Niech X_i oznacza zaobserwowane czasy życia. Wówczas zmienne losowe po cenzurowaniu definiuje się jako:

$$T_i = \begin{cases} X_i, & X_i \leq t_0 \quad (\delta_i = 1), \\ t_0, & X_i > t_0 \quad (\delta_i = 0), \end{cases} \quad \text{czyli} \quad T_i = \min(X_i, t_0).$$

gdzie δ_i jest indykatorem cenzurowania.

Poniżej przedstawiono kod w Pythonie realizujący cenzorowanie I typu dla dowolnych danych:

```
def cenzurowanie_I_typu(t_0, data):
    size = len(data)

    censored_data = []
    deltas = []

    for x in data:
        if x <= t_0:
            censored_data.append(x)
            deltas.append(1)
        else:
            censored_data.append(t_0)
            deltas.append(0)

    return censored_data, deltas
```

Poniżej przedstawiono kod w Pythonie konkretnie do generowania danych cenzurowanych I typu z rozkładu $\mathcal{GE}(\lambda, \alpha)$:

```
def GE_cenzurowanie_I_typu(t0, alpha, lambd, n):
  data = EW_generator(1, lambd, alpha, size=n)
  return cenzurowanie_I_typu(t0, data)
```

2.1.2 Cenzurowanie II typu

Cenzurowanie typu II polega na obserwacji jedynie m najmniejszych wartości zmiennych losowych, a pozostałe wartości są zastępowane przez $X_{(m)}$. Oznacza to, że mierzymy tylko do uzyskania $m \leq n$ danych kompletnych.

Niech $X_{(k)}$ oznacza k-tą statystykę pozycyjną. Wówczas zmienne losowe cenzurowane definiuje się jako:

$$T_i = \begin{cases} X_{(i)}, & i = 1, \dots, m \quad (\delta_i = 1), \\ X_{(m)}, & i = m+1, \dots, n \quad (\delta_i = 0), \end{cases} \qquad \text{czyli} \quad T_i = \min(X_{(i)}, X_{(m)}).$$

gdzie δ_i jest indykatorem cenzurowania.

Poniżej przedstawiono kod w Pythonie realizujący cenzorowanie II typu dla dowolnych danych:

```
def cenzurowanie_II_typu(m, data):
    censored_data = np.sort(data)

    # Wszystkie obserwacje powyżej m-tej zamieniamy na X_(m)
    censored_data[m:] = censored_data[m-1]

    deltas = np.zeros(len(data))
    deltas[:m] = 1

    return censored_data, deltas
```

Poniżej przedstawiono kod w Pythonie konkretnie do generowania danych cenzurowanych I typu z rozkładu $\mathcal{GE}(\lambda, \alpha)$:

```
def GE_cenzurowanie_II_typu(m, alpha, lambd, n):
  data = EW_generator(1, lambd, alpha, size=n)
  return cenzurowanie_I_typu(m, data)
```

2.1.3 Cenzurowanie losowe

Cenzurowanie losowe jest rozszerzeniem cenzurowania typu I, z tym że t_0 nie jest już wartością stałą, lecz zmienną losową C_i, różną dla każdej obserwacji. Zmienna C_i nazywana jest zmienną cenzurującą i ma rozkład g oraz dystrybuantę G (nie musi być taki sam jak dla zmiennej cenzurowanej). Zmienna cenzurująca C_i jest niezależna od X_i i przyjmuje tę samą rolę dla każdej jednostki.

$$T_i = \begin{cases} X_i, & X_i \leq C_i \quad (\delta_i = 1), \\ C_i, & X_i > C_i \quad (\delta_i = 0), \end{cases} \quad \text{czyli} \quad T_i = \min(X_i, C_i).$$

gdzie δ_i jest indykatorem cenzurowania.

Poniżej przedstawiono kod w Pythonie realizujący cenzorowanie losowe dla dowolnych danych:

```
def cenzurowanie_losowe(cen, data):
    deltas = (data <= cen).astype(int)
    censored_data = np.minimum(data, cen)
    return censored_data, deltas
```

Poniżej przedstawiono kod w Pythonie konkretnie do generowania danych cenzurowanych losowo z rozkładu $\mathcal{GE}(\lambda, \alpha)$, cenzurowanych zmienną losową z rozkładu wykładniczego $\mathcal{E}(\lambda)$:

```
def GE_cenzurowanie_losowe(eta, alpha, lambd, n):
  data = EW_generator(1, lambd, alpha, size=n)
  cen = EW_generator(1, eta, 1, size=n)
  return cenzurowanie_losowe(cen, data)
```

2.2 Zadanie 2

W tym zadaniu wygenerowano próbę o rozmiarze $n = 200$ z rozkładu $\mathcal{GE}(1, 1.5)$, a następnie zastosowano cenzurowanie typu I, typu II oraz losowe.

- W cenzurowaniu typu I wybrano parametr $t_0 = 1$.
- W cenzurowaniu typu II wybrano parametr $m = 120$.
- W przypadku cenzurowania losowego wygenerowano próbę zmiennych cenzurujących z rozkładu $\mathcal{E}(1)$.

Poniżej przedstawiono kod w Pythonie generujący wspomniane wyżej dane cenzurowane.

```
lambd = 1.0
alpha = 1.5
size = 200

# Parametry cenzurowania
t0 = 1.0
m = 120
eta = 1.0

data = EW_generator(1, lambd, alpha, size)
cen = EW_generator(1, eta, 1, size)

times_I, deltas_I = cenzurowanie_I_typu(t0, data)
times_II, deltas_II = cenzurowanie_II_typu(m, data)
times_R, deltas_R = cenzurowanie_losowe(cen, data)
```

W następnej kolejności wyliczono podstawowe statystyki opisowe wygenerowanych danych. Jednak w przypadku danych cenzurowanych pominięto średnią i odchylenie standardowe, ponieważ klasyczne wzory na te statystyki zakładają pełne obserwacje i nie uwzględniają faktu, że niektóre wartości są ograniczone przez cenzurę. Obliczenie średniej lub odchylenia standardowego z użyciem klasycznych formuł mogłoby prowadzić do obciążonych i niepoprawnych wyników.

Zamiast tego dodano informację o liczbie danych cenzurowanych, co jest kluczowe w analizie takich danych, ponieważ pozwala ocenić stopień niepełności próby i wpływ cenzury na interpretację statystyk opisowych.

Tabela 4: Podstawowe statystyki opisowe danych cenzurowanych wygenerowanych z rozkładu $\mathcal{GE}(1, 1.5)$

Rodzaj cenzurowania	I typu ($t_0 = 1.0$)	II typu ($m = 120$)	Losowe ($\eta = 1.0$)
Rozmiar próby	200	200	200
Liczba obserwacji kompletnych	102	120	80
Kwantyl dolny (Q1)	0.5158	0.5158	0.2149
Mediana	0.9843	0.9843	0.4430
Kwantyl górny (Q3)	1.0000	1.1835	0.8348
Rozstęp międzykwartylowy (IQR)	0.4842	0.6678	0.6198
Minimum	0.0016	0.0016	0.0013
Maksimum	1.0000	1.1835	2.5113
Rozstęp	0.9984	1.1819	2.5100

Z Tabeli 2. można odczytać m.in., że cenzurowanie wpływa na rozkład danych. Cenzurowanie typu I i typu II są do siebie zbliżone pod względem sposobu ograniczania obserwacji - w cenzurowaniu typu I z góry ustalany jest czas obserwacji t_0, na podstawie którego wyznaczana jest liczba obserwacji kompletnych, natomiast w cenzurowaniu typu II określa się liczbę obserwacji kompletnych m, a odpowiadający jej czas obserwacji X_m wyznaczany jest na podstawie danych. W przypadku cenzurowania losowego sytuacja wygląda inaczej, ponieważ moment cenzurowania nie jest ustalony z góry, lecz stanowi zmienną losową C_i, co wprowadza dodatkowy element losowości i powoduje większe zróżnicowanie wśród obserwacji.

2.3 Zadanie 3

W tym zadaniu wyznaczono statystyki opisowe danych cenzurowanych I-go typu dotyczących leczenia dwoma różnymi lekami A i B. Grupa 40 pacjentów została podzielona losowo na dwie równoliczne podgrupy. Przez jeden rok obserwacji, jednej z nich podawano lek A, drugiej - lek B, i obserwowano czas do remisji choroby.

W grupie otrzymującej lek A uzyskano następujące dane. U dziesięciu pacjentów remisja choroby nastąpiła w chwilach: 0.03345514, 0.08656403, 0.08799947, 0.24385821, 0.27755032, 0.40787247, 0.58825664, 0.64125620, 0.90679161, 0.94222208, natomiast u pozostałych dziesięciu pacjentów w ciągu roku nie zaobserwowano remisji.

W grupie otrzymującej lek B uzyskano następujące dane. U dziesięciu pacjentów remisja choroby nastąpiła w chwilach: 0.03788958, 0.12207257, 0.20319983, 0.24474299, 0.30492413,

0.34224462, 0.42950144, 0.44484582, 0.63805066, 0.69119721, natomiast u pozostałych dziesięciu pacjentów w ciągu roku nie zaobserwowano remisji.

Tabela 5: Podstawowe statystyki opisowe danych czasu remisji choroby po stosowaniu leku A lub leku B - dane cenzurowane I typu

Statystyka Opisowa	Lek A	Lek B
Rozmiar próby	20	20
Liczba obserwacji kompletnych	10	10
Liczba danych cenzurowanych	10	10
Minimum	0.0335	0.0379
Kwartyl dolny (Q1)	0.3753	0.3329
Mediana	0.9711	0.8456
Kwartyl górny (Q3)	1	1
Maksimum	1	1
Rozstęp międzykwartylowy (IQR)	0.6247	0.6671
Rozstęp (max-min)	0.9665	0.9621

W Tabeli 3. widać że grupy stosujące lek A oraz lek B nie różnią się znacząco między sobą.

3 Lista 3

Lista 3 polega na wyznaczaniu estymatorów parametrów dla rozkładów, których obserwacje zostały częściowo ocenzurowane, a także na wyznaczaniu przedziałów ufności dla tych estymatorów. Dodatkowo przeprowadzono porównanie różnych estymatorów na danych wygenerowanych.

UWAGA: Podane poniżej kody generują estymację punktową oraz przedziałową dla rozkładów wykładniczych postaci $f(x) = \theta \exp(-\theta x)\, \mathbb{1}_{(0;\infty)}(x)$, dla którego wartość oczekiwana wynosi $\mathbb{E}[X] = \frac{1}{\theta}$. W zadaniach 1. oraz 2. chcemy otrzymać średni czas do remisji choroby, czyli $\mathbb{E}[X] = \mu$. Z tego wynika, że $\theta = \frac{1}{\mu}$. Z tego powodu wyniki będą podawane w postaci:

- $\widehat{\mu} = \frac{1}{\widehat{\theta}}$, w przypadku estymacji punktowej,
- $\mu \in \left(\frac{1}{T_U}; \frac{1}{T_L}\right)$, gdzie T_L oraz T_U - odpowiednio dolna i górna granica przedziału ufności parametru θ.

3.1 Zadanie 1

Zadanie polega na oszacowaniu średniego czasu do remisji choroby dla pacjentów leczonych lekami A i B na podstawie danych poddanych cenzurowaniu typu I przyjmując, że dane pochodzą z rozkładu wykładniczego $\mathcal{E}(1)$. Obejmuje ono:

(a) wyznaczenie estymatorów największej wiarogodności średniego czasu do remisji dla obu grup,
(b) wyznaczenie przedziałów ufności dla średniego czasu do remisji na poziomach ufności 95% ($\alpha = 0.05$) i 99% ($\alpha = 0.01$) dla obu grup.

3.1.1 Estymacja punktowa

Z wykładu wiadomo że estymatorem największej wiarogodności jest estymator $\widehat{\theta} = \frac{R}{T_1}$, gdzie:

$$R = \sum_{i=1}^{n} \mathbf{1}\left(X_i \leq t_0\right), \quad T_1 = \sum_{i=1}^{R} X_{(i)} + t_0(n - R)$$

Poniżej przedstawiono kod w języku Python służący do wyznaczania estymatora największej wiarogodności dla danych cenzurowanych I typu z rozkładu wykładniczego $\mathcal{E}(\lambda)$. Dodatkowo dokonano estymacji parametrycznej parametru θ dla danych z Listy 2. Zadania 3. za pomocą metody największej wiarogodności.

```python
def MLE_cenzurowanie_I(times, deltas, t0):
    R = np.sum(deltas)
    n = len(times)
    T1 = np.sum(times[deltas==1]) + t0*(n - R)
    theta_hat = R / T1
    return theta_hat

mu_hat_A = 1/MLE_cenzurowanie_I(times_A, deltas_A, t0)
mu_hat_B = 1/MLE_cenzurowanie_I(times_B, deltas_B, t0)
```

Dla danych dotyczących osób przyjmujących lek A otrzymano estymator $\hat{\mu}_A = 1.422$, natomiast dla osób przyjmujących lek B - estymator $\hat{\mu}_B = 1.346$.

3.1.2 Estymacja przedziałowa

Do estymacji przedziałowej nie stosujemy estymatorwa NW $\hat{\theta} = \frac{R}{T_1}$ do bezpośredniego konsturowania przedziałów ufności, ponieważ jest on ilorazem zmiennej dyskretnej R i zmiennej ciągłej T_1. Taka mieszana natura utrudnia uzyskanie analitycznej funkcji centralnej o znanym rozkładzie w próbie skończonej, więc konstrukcja dokładnych przedziałów ufności jest trudna (wymaga przybliżeń asymptotycznych lub metod numerycznych).

Z tego względu używa się alternatywnego estymatora $\tilde{\theta} = -\frac{log(1-\frac{R}{n})}{t_0}$, który zależy wyłącznie od R i korzysta z faktu, że $R \sim \mathcal{B}(n, p)$ z $p = 1 - \exp(-\vartheta t_0)$.

Dla $\tilde{\theta}$ przedziały ufności buduje się dwuetapowo: najpierw wyznaczamy przedział $[p_L, p_U]$ dla p na poziomie ufności $1 - \alpha$ (np. metodą Cloppera–Pearsona), a następnie transformujemy jego końce przez odwrotność $p = 1 - \exp(-\vartheta t_0)$, otrzymując

$$\vartheta_L = -\frac{1}{t_0}\ln(1 - p_L), \qquad \vartheta_U = -\frac{1}{t_0}\ln(1 - p_U).$$

W praktyce przedział Cloppera–Pearsona dla p definiuje się jako $S_{\leq} \cap S_{\geq}$ (równoważnie $[\inf S_{\geq}; \sup S_{\leq}]$), gdzie dla $k = R$:

$$S_{\leq} = p : \Pr(\mathrm{Bin}(n, p) \leq k) > \tfrac{\alpha}{2}, \qquad S_{\geq} = p : \Pr(\mathrm{Bin}(n, p) \geq k) > \tfrac{\alpha}{2}.$$

Po uzyskaniu $[p_L, p_U]$ powyższa transformacja daje przedział ufności $[\vartheta_L, \vartheta_U]$ o poziomie ufności $1 - \alpha$.

Należy mieć na uwadzę że wynik będzie jedynie przybliżony - co wynika z błędów numerycznych.

Poniżej przedstawiono kod w Pythonie realizujący generowanie wspomnianego wcześniej przedziału ufności:

```
def binom_cdf(t, n, p):
    s = 0.0
    for i in range(0, t+1):
        s += comb(n, i) * (p**i) * ((1-p)**(n-i))
    return s

def clopper_pearson_binom(R, n, alpha):
    if R == 0:
        p_L = 0
        P_U = 1 - (alpha / 2) **(1 / n)
        return p_L, p_U
    if R == n:
        p_U = 1.0
        p_L = (alpha / 2.0) ** (1.0 / n)
        return p_L, p_U

    target_low = 1 - alpha / 2
    L, H = 0, 1

    while H - L > 1e-12:
        M = (L + H) / 2
        cdf = binom_cdf(R, n, M)
        if cdf > target_low:
            L = M
        else:
            H = M
    p_L = (L + H) / 2

    target_up = alpha / 2
    L, H = 0, 1

    while H - L > 1e-12:
        M = (L + H) / 2
        cdf = binom_cdf(R, n, M)
        if cdf > target_up:
            L = M
        else:
            H = M
    p_U = (L + H) / 2

    return p_L, p_U
```

```python
def p_to_theta(p, t0):
    return -np.log(1 - p) / t0

def Ci_theta_cenzurowanie_I(times, deltas, t0, alpha):
    R = np.sum(deltas)
    n = len(times)

    p_L, p_U = clopper_pearson_binom(R, n, alpha)

    theta_L = p_to_theta(p_L, t0)
    theta_U = p_to_theta(p_U, t0)

    return theta_L, theta_U
```

Poniżej przedstawiono wywołanie kodu dla danych z Listy 2. Zadanie 3. Wyniki przedstawiono w tabeli poniżej.

```python
alphas = [0.05, 0.01]
rows = []

for alpha in alphas:
    # Lek A
    thA_L, thA_U = Ci_theta_cenzurowanie_I(times_A, deltas_A, t0, alpha)
    rows.append(["Lek A", alpha, 1/thA_U, 1/thA_L])

    # Lek B
    thB_L, thB_U = Ci_theta_cenzurowanie_I(times_B, deltas_B, t0, alpha)
    rows.append(["Lek B", alpha, 1/thB_U, 1/thB_L])
```

Tabela 6: Przedziały ufności dla danych cenzurowanych I typu - różne poziomy ufności

Grupa	α	$\hat{\mu}_L$	$\hat{\mu}_U$
Lek A	0.05	0.768	2.64
Lek B	0.05	0.768	2.64
Lek A	0.01	0.656	3.363
Lek B	0.01	0.656	3.363

W Tabeli 4. widać, że otrzymane przedziały ufności są identyczne dla obu leków. Wynika to z faktu, że oba zbiory danych zawierają taką samą liczbę obserwacji kompletnych oraz ocenzurowanych, co przy tym samym czasie obserwacji prowadzi do identycznych wartości estymatora i granic przedziału ufności.

3.2 Zadanie 2

Zadanie polega na oszacowaniu średniego czasu do remisji choroby dla pacjentów leczonych lekami A i B na podstawie danych poddanych cenzurowaniu typu II, przyjmując, że pochodzą one z rozkładu wykładniczego. Obejmuje ono:

(a) wyznaczenie estymatorów największej wiarogodności średniego czasu do remisji dla obu grup,
(b) wyznaczenie przedziałów ufności dla średniego czasu do remisji na poziomach ufności 95% ($\alpha = 0.05$) i 99% ($\alpha = 0.01$) dla obu grup.

3.2.1 Estymacja punktowa

Dla danych cenzurowanych II typu pochodzących z rozkładu wykładniczego $\mathcal{E}(\lambda)$ estymator największej wiarogodności przyjmuje postać:

$$\hat{\theta} = \frac{m}{T_2}, \quad \text{gdzie} \quad T_2 = \sum_{i=1}^{m} X_{(i)} + (n-m)X_{(m)}.$$

Poniżej przedstawiono kod w języku Python dla danych z Listy 2, Zadania 3, które tym razem przyjmujemy jako cenzurowane II typu z parametrem cenzurowania $m = 10$.

```
m = 10

def MLE_cenzurowanie_II(times, deltas, m):
    n = len(times)
    T2 = np.sum(times[deltas==1]) + (n - m)*np.max(times[deltas==1])
    theta_hat = m / T2
    return theta_hat

mu_hat_A = 1/MLE_cenzurowanie_II(times_A, deltas_A, m)
mu_hat_B = 1/MLE_cenzurowanie_II(times_B, deltas_B, m)
```

Dla danych dotyczących osób przyjmujących lek A otrzymano estymator $\hat{\mu}_A = 1.364$, natomiast dla osób przyjmujących lek B - estymator $\hat{\mu}_B = 1.037$.

3.2.2 Estymacja przedziałowa

Do wyznaczenia przedziałów ufności dla danych cenzurowanych II typu pochodzących z rozkładu wykładniczego $\mathcal{E}(\theta)$ potrzebna będzie funkcja kwantylowa rozkładu gamma $\Gamma(k, \theta)$. Wynika

to stąd, że dla statystyki

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

zachodzi zależność

$$\frac{T_2}{m\theta} \sim \text{Gamma}(m, \frac{1}{m})$$

Dystrybuanta rozkładu gamma wyraża się przez niepełną funkcję gamma:

$$F(x; k, \sigma) = \frac{\gamma(k, x/\sigma)}{\Gamma(k)},$$

gdzie

$$\gamma(z, x) = \int_0^x t^{z-1} e^{-t}\, dt \qquad \text{(niepełna funkcja gamma)},$$
$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t}\, dt \qquad \text{(funkcja gamma)}.$$

Ponieważ dystrybuanta gamma nie ma prostego wzoru elementarnego, korzystamy z jej funkcji kwantylowej (numerycznie dostępnej w pakietach statystycznych).

Niech $q_{\alpha/2}$ i $q_{1-\alpha/2}$ będą kwantylami rozkładu $\mathcal{G}amma(m, 1/m)$. Z faktu, że

$$\Pr\left(q_{\alpha/2} \leq \frac{T_2}{\theta} \leq q_{1-\alpha/2}\right) = 1 - \alpha$$

otrzymujemy przedział ufności dla θ:

$$[\theta_L, \theta_U] = \left[\frac{m \cdot T_2}{q_{1-\alpha/2}},\ \frac{m \cdot T_2}{q_{\alpha/2}}\right].$$

W praktyce kwantyle $q_{\alpha/2}$ i $q_{1-\alpha/2}$ oblicza się numerycznie, np. funkcją `scipy.stats.gamma.ppf` w Pythonie z parametrami `a=m` i `scale=1/m`.

Poniżej przedstawiono kod realizujący wyżej wspomniają estymację przedziałową.

```
def Ci_theta_cenzurowanie_II(times, deltas, m, alpha):
    n = len(times)
    T2 = np.sum(times[deltas==1]) + (n - m)*np.max(times[deltas==1])

    q_lower = gamma.ppf(alpha/2, a=m, scale=1/m)
    q_upper = gamma.ppf(1 - alpha/2, a=m, scale=1/m)
```

```
    theta_L = m * q_lower / T2
    theta_U = m * q_upper / T2

    return theta_L, theta_U
```

Poniżej przedstawiono wywołanie kodu dla danych z Listy 2. Zadanie 3. Wyniki przedstawiono w tabeli poniżej.

```
alphas = [0.05, 0.01]
rows = []

for alpha in alphas:
    # Lek A
    thA_L, thA_U = Ci_theta_cenzurowanie_II(times_A, deltas_A, m, alpha)
    rows.append(["Lek A", alpha, 1/thA_U, 1/thA_L])

    # Lek B
    thB_L, thB_U = Ci_theta_cenzurowanie_II(times_B, deltas_B, m, alpha)
    rows.append(["Lek B", alpha, 1/thB_U, 1/thB_L])
```

Tabela 7: Przedziały ufności dla danych cenzurowanych II typu - różne poziomy ufności

Grupa	α	$\hat{\mu}_L$	$\hat{\mu}_U$
Lek A	0.05	0.798	2.844
Lek B	0.05	0.607	2.163
Lek A	0.01	0.682	3.669
Lek B	0.01	0.519	2.79

Z Tabeli 5. można odczytać, że powstałe przedziały ufności są węższe dla danych dotyczących leku A.

3.3 Zadanie 3

Zadanie polega na symulacyjnym porównaniu dokładności dwóch estymatorów punktowych ϑ, zdefiniowanych wzorami:

$$\hat{\vartheta} = \frac{R}{T_1}, \quad \tilde{\vartheta} = -\frac{\log\left(1 - \frac{R}{n}\right)}{t_0}$$

gdzie

$$R = \sum_{i=1}^{n} \mathbf{1}_{\{X_i \leq t_0\}}, \quad T_1 = \sum_{i=1}^{R} X_{(i)} + t_0(n - R).$$

Porównanie estymatorów zostanie dokonane na podstawie:

- obciążenia (bias):

$$\text{Bias}(\hat{\vartheta}, \vartheta) = E_\vartheta(\hat{\vartheta} - \vartheta), \quad \text{Bias}(\tilde{\vartheta}, \vartheta) = E_\vartheta(\tilde{\vartheta} - \vartheta),$$

- średniego błędu kwadratowego (MSE):

$$\text{MSE}(\hat{\vartheta}, \vartheta) = E_\vartheta(\hat{\vartheta} - \vartheta)^2, \quad \text{MSE}(\tilde{\vartheta}, \vartheta) = E_\vartheta(\tilde{\vartheta} - \vartheta)^2,$$

dla wartości $\vartheta = 1$, rozmiarów próby $n = 10, 30$ oraz parametrów cenzurowania $t_0 = 0.5, 1, 2$.

Celem jest ocena, który z estymatorów daje mniejsze obciążenie i mniejszy błąd średniokwadratowy w różnych scenariuszach cenzurowania.

Symulacja będzie polegała na generowaniu prób 100 000 razy. W przypadku, gdy $R = n$ (czyli wszystkie dane są niecenzurowane), estymatory przyjmują wartości skrajne $\hat{\vartheta} = 0$, $\tilde{\vartheta} = -\infty$, co czyni je niepraktycznymi w analizie. W takiej sytuacji iteracja zostanie powtórzona. Wartości $\hat{\vartheta}$ oraz $\tilde{\vartheta}$ zawsze będą wyznaczane na podstawie tych samych prób. Takie podejście ogranicza wpływ sytuacji skrajnych i zapewnia poprawność porównania estymatorów.

Poniżej przedstawiono kod w Pythonie dokonujący wyżej wspomnianej symulacji. Wyniki przedstawiono w tabeli poniżej.

```
theta_true = 1
n_values = [10, 30]
t0_values = [0.5, 1, 2]
n_sim = 100000

rows = []

for n in n_values:
    for t0 in t0_values:
        theta_hat_vals = []
        theta_wave_vals = []
        i = 0
        while i < n_sim:
          # generowanie próbki i cenzurowanie I typu
          sample, deltas = GE_cenzurowanie_I_typu(t0, theta_true, 1, n)
```

```
        R = np.sum(deltas)

        if R == n:
              continue

        T1 = np.sum(sample[deltas==1]) + t0*(n - R)

        theta_hat = R / T1
        theta_wave = -np.log(1 - R/n) / t0

        theta_hat_vals.append(theta_hat)
        theta_wave_vals.append(theta_wave)
        i += 1

    bias_hat = np.mean(theta_hat_vals) - theta_true
    bias_wave = np.mean(theta_wave_vals) - theta_true
    mse_hat = np.mean((np.array(theta_hat_vals) - theta_true)**2)
    mse_wave = np.mean((np.array(theta_wave_vals) - theta_true)**2)

    rows.append({
    "Liczność próby": n,
    "Czas obserwacji": t0,
    r"$\mathrm{Bias}(\hat{\theta})$": bias_hat,
    r"$\mathrm{Bias}(\tilde{\theta})$": bias_wave,
    r"$\mathrm{MSE}(\hat{\theta})$": mse_hat,
    r"$\mathrm{MSE}(\tilde{\theta})$": mse_wave
    })
```

Tabela 8: Porównanie estymatorów $\hat{\vartheta}$ oraz $\tilde{\vartheta}$

Liczność próby	Czas obserwacji	Bias($\hat{\theta}$)	Bias($\tilde{\theta}$)	MSE($\hat{\theta}$)	MSE($\tilde{\theta}$)
10	0.5	0.3786	0.0758	1.2898	0.3349
10	1.0	0.9084	0.0864	2.9909	0.2257
10	2.0	1.3189	-0.0756	3.0273	0.0610
30	0.5	0.3161	0.0228	0.3633	0.0936
30	1.0	0.7916	0.0316	1.2193	0.0665
30	2.0	2.6710	0.0513	12.9035	0.0682

Z Tabeli 6. wynika, że estymator największej wiarogodności $\hat{\vartheta}$ ma większe wartości błędu średniokwadratowego (MSE) oraz większe odchylenie w stosunku do prawdziwej wartości pa-

rametru dla małych prób i krótkich czasów obserwacji w porównaniu do estymatora $\tilde{\vartheta}$, który jest oparty wyłącznie na liczbie obserwacji kompletnych R.

Dla większych prób ($n = 30$) oraz dłuższych czasów obserwacji różnice między estymatorami stają się mniejsze, a oba estymatory wykazują zbliżone wartości MSE i biasu. Warto zauważyć, że $\tilde{\vartheta}$ jest stabilniejszy i mniej wrażliwy na cenzurowanie danych, co czyni go bardziej praktycznym w zastosowaniach z danymi cenzurowanymi I typu.

3.4 Zadanie dodatkowe 1

3.5 Zadanie dodatkowe 2

3.6 Zadanie dodatkowe 3

4 Lista 4

Lista 4 polega na testowaniu hipotez dotyczących średniego czasu życia na podstawie danych cenzurowanych I typu z rozkładu wykładniczego, poprzez wyznaczanie poziomu krytycznego testu ilorazu wiarogodności oraz ocenę jego mocy i rozmiaru.

4.1 Zadanie 1

Zadanie dotyczy deklaracji funkcji do wyznaczania poziomu krytycznego (*eng. p-value*) w teście ilorazu wiarogodności do testowania hipotez prawo-, lewo-i dwustronnych dla danych cenzurowanych I-go typu pochodzących z rozkładu wykładniczego $\mathcal{E}(\theta)$.

Zacznijmy od udowodnienia że funkcja wiarygodności jest funkcją unimodalną (ma jedno maksimum globalne).

Funkcja wiarogodności dla rozkładu wykładniczego $\mathcal{E}(\theta)$ ma postać:

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

Najwygodniej maksimum będzie się szukało przez funkcję logarytmiczną (log-wiarogodność), ponieważ logarytm jest funkcją rosnącą i zachowuje maksimum:

$$\ell(\vartheta) = \ln\frac{n!}{(n-r)!} + r\ln\vartheta - \vartheta S, \qquad S = \sum_{i=1}^{r} x_{(i)} + t_0(n-r).$$

Aby znaleźć maksimum, obliczamy pochodną log-wiarogodności po θ.

$$\ell'(\vartheta) = \frac{r}{\vartheta} - S = 0 \quad \Rightarrow \quad \hat{\vartheta} = \frac{r}{S}.$$

Z warunku stacjonanrości wynika że ekstremum znajduje się w $\hat{\vartheta} = \frac{r}{S}$, który jest naszym estymatorem największej wiarogodności.

Sprawdzamy, czy funkcja jest wklęsła (co gwarantuje maksimum):

$$\ell''(\vartheta) = -\frac{r}{\vartheta^2} < 0 \quad \forall\vartheta > 0.$$

Wniosek: ℓ jest ściśle wklęsła, więc L jest unimodalna z maksimum globalnym w $\hat{\vartheta}$.

Funkcja ilorazu wiarogodności ma postać:

$$\lambda(t^*) = \frac{\sup_{\vartheta\in\Theta_0} L(\vartheta; t^*)}{\sup_{\vartheta\in\Theta} L(\vartheta; t^*)}.$$

Przy testowaniu hipotezy dwustronnej zbiór $\Theta_0 = \{\vartheta_0\}$ jest jednoelementowy i właśnie w tym punkcie osiąga swoje supremum.

W przypadku hipotezy prawostronnej, tj. testowania na przedziale $[\vartheta_0, \infty)$, supremum przyjmuje się w punkcie $\hat{\vartheta}$, jeżeli $\hat{\vartheta} \in [\vartheta_0, \infty]$, lub w punkcie ϑ_0 w przeciwnym przypadku.

Analogicznie, dla hipotezy lewostronnej, gdy rozważany przedział to $(-\infty, \vartheta_0]$, supremum osiągane jest w $\hat{\vartheta}$, jeśli $\hat{\vartheta} \in (-\infty, \vartheta_0]$, lub w ϑ_0 w przeciwnym wypadku.

Zgodnie z twierdzeniem Wilksa, wartość krytyczna ma postać:

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

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

Poniżej przedstawiono kod deklarujący funkcję do testowania wyżej wspomianych hipotez danych cenzurowanych z rozkładu wykładniczego $\mathcal{E}(\theta)$.

```
def IW_cenzurowanie_I(r, s, n, t0, theta0, test_type, alpha):
    if r == 0:
        theta_hat = 1e-9
    else:
        S = s + (n - r) * t0 # Całkowity czas testowania
        theta_hat = r / S # Estymator MLM

    Lambda = (theta0 / theta_hat)**r * np.exp(S * (theta_hat - theta0))

    chi2_statistic = -2 * np.log(Lambda)
    lambda_1 = chi2.ppf(1 - alpha, df=1)

    if test_type == "dwustronna":
        # H0: theta = theta0  vs  H1: theta =/= theta0
        p_value = 1 - chi2.cdf(chi2_statistic, df=1)
    elif test_type == "prawostronna":
        # H0: theta <= theta0  vs  H1: theta > theta0
        p_value = ((1 - chi2.cdf(chi2_statistic, df=1))
        if (theta_hat > theta0) else 1)
    else:  # "lewostronna"
        # H0: theta >= theta0  vs  H1: theta < theta0
        p_value = ((1 - chi2.cdf(chi2_statistic, df=1))
        if (theta_hat < theta0) else 1)

    return chi2_statistic, lambda_1, p_value
```

4.2 Zadanie 2

Zadanie polega na przeprowadzeniu symulacji, których celem jest oszacowanie mocy (dla 10 wybranych alternatyw) oraz rozmiaru testu z punktu dwustronnego dla wybranej wartości ϑ_0, t_0 oraz $n \in \{20, 50\}$.

Próba została wylosowana dla $\vartheta \in \{0.1, 0.3, 0.5, 0.7, 0.9, 1, 1.1, 1.3, 1.5, 1.7\}$, ocenzurowana I-typu z parametrem cenzurowania $t_0 = 1$, a następnie badano hipotezę zerową H_0 $\vartheta = \vartheta_0 = 1$ przy hipotezie alternatywnej H_1 $\vartheta \neq \vartheta_0$. Dla każdego przypadku wykonano 10 000 replikacji. Wyniki przedstawiono w tabeli poniżej.

Poniżej znajduje się kod w języku Python dokonujący symulacji.

```
theta0 = 1
t0 = 1.5
n_vals = [20, 50]
alternatives = [0.1, 0.3, 0.5, 0.7, 0.9, 1, 1.1, 1.3, 1.5, 1.7]
alpha = 0.05
N = 10  # liczba replikacji

rows = []
rozmiar = []

for n in n_vals:
    for theta in alternatives:
        results = []

        N_cur = 0
        while N_cur != N:
            sample, deltas = GE_cenzurowanie_I_typu(t0, theta, 1.0, n)

            r = np.sum(deltas)
            s = np.sum(sample[deltas==1])

            _, _, p_value = IW_cenzurowanie_I(r, s, n, t0, theta0,
            "dwustronna", alpha)
            results.append(int(p_value <= alpha))

            if (r != 0):
                N_cur += 1

        moc = np.mean(results)
        if theta == theta0:
```

```
            rozmiar.append(moc)

        rows.append({
            r"\vartheta": theta,
            "Liczność próby": n,
            "Moc testu": moc
        })
```

Tabela 9: Wyniki symulacji mocy testu dwustronnego

ϑ	Liczność próby	Moc testu	Liczność próby	Moc testu
0.1	20	1.0000	50	1.0000
0.3	20	1.0000	50	1.0000
0.5	20	1.0000	50	1.0000
0.7	20	0.9000	50	1.0000
0.9	20	0.7000	50	0.9000
1.0	20	0.5000	50	0.9000
1.1	20	0.7000	50	0.7000
1.3	20	0.4000	50	0.6000
1.5	20	0.1000	50	0.2000
1.7	20	0.3000	50	0.1000

Rozmiar testu wyniósł 0.500 dla liczności próby $n = 20$ oraz 0.900 dla liczności próby $n = 50$.

COŚ NAPISAĆ ŻE CHWALIMY ANTYCHRYSTA

4.3 Zadanie 3

Zadanie polega na weryfikacji hipotezy, że średni czas do remisji choroby w grupie, która brała lek A, oraz w grupie, która brała lek B (na podstawie danych z Listy 2, Zadanie 3), można traktować jako realizacje zmiennych losowych z rozkładu wykładniczego $\mathcal{E}(1)$. Zakładamy również, że dokonano cenzurowania typu I z parametrem cenzurowania $t_0 = 1$. Test przeprowadzimy na poziomie istotności $\alpha = 0.05$.

Poniżej przedstawiono kod dokonujący testowania tej hipotezy statystycznej. Wyniki przedstawiono w tabelce poniżej.

```
theta0 = 1.0
t0 = 1.0
alpha = 0.05
```

```
rows = []

for name, (times, deltas) in datasets.items():
    n = len(times)
    r = int(np.sum(deltas))
    s = float(np.sum(times[deltas==1]))
    chi2_statistic, lambda_1, p_value = IW_cenzurowanie_I(r, s, n, t0, theta0,
    "dwustronna", alpha)
    rows.append({
        "Grupa": name,
        "Liczność próby": n,
        "Liczność danych kompletnych": r,
        "p -wartość": p_value,
    })
```

Tabela 10: Testowanie hipotezy dwustronnej na danych rzeczywistych

Grupa	Liczność próby	Liczność danych kompletnych	p -wartość
Lek A	20	10	0.2374
Lek B	20	10	0.3230

Jak widać, wartości p są większe od przyjętego poziomu istotności, zatem nie ma podstaw do odrzucenia hipotezy zerowej H_0.

4.4 Zadanie dodatkowe