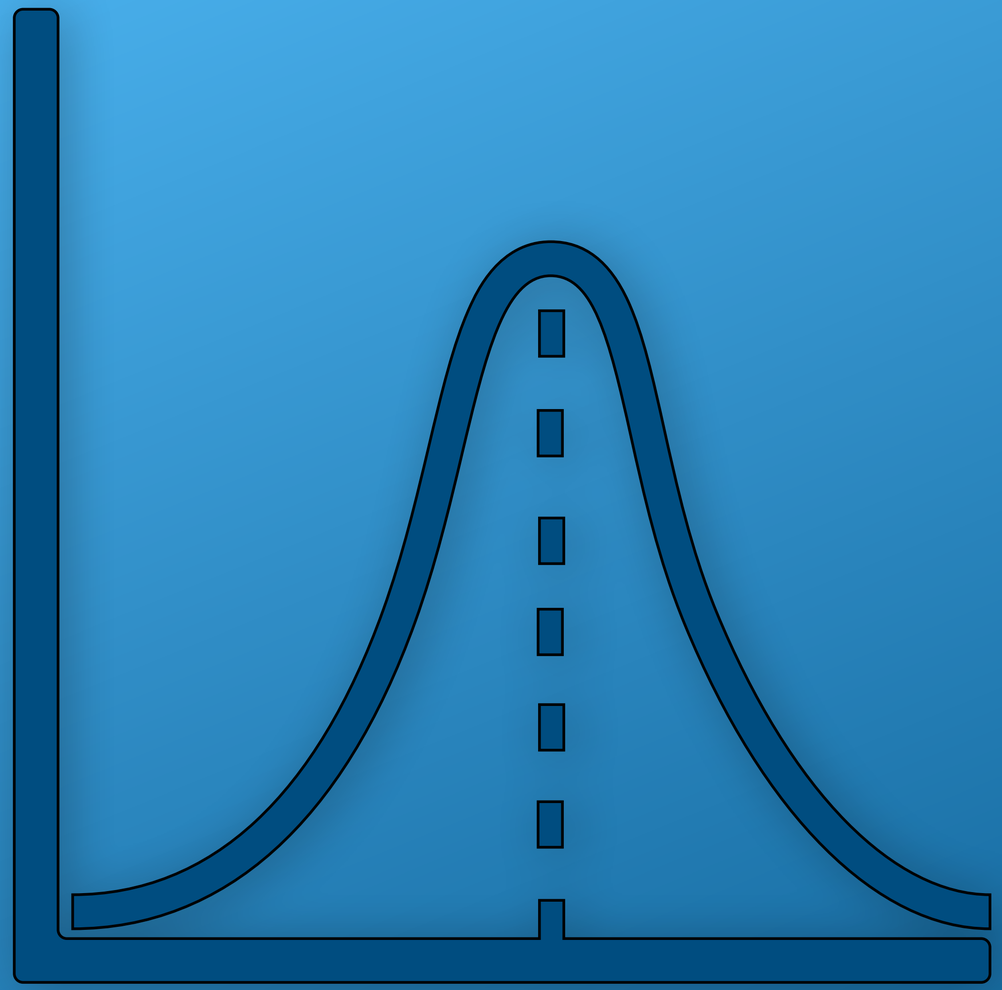
Projekt zaliczeniowy

Statystyczne Modelowanie Danych Biologicznych

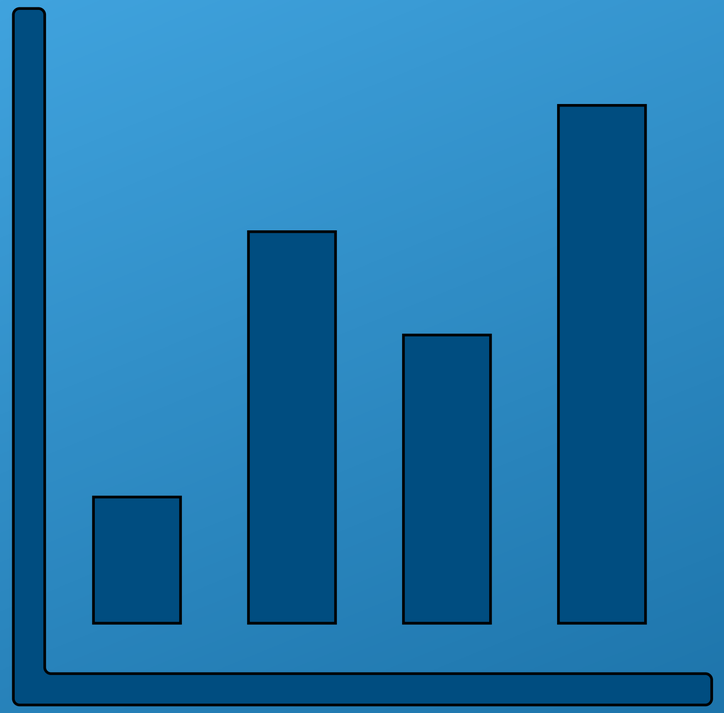
Kacper Kaszuba
Daria Plewa
Karol Rorat
Bioinformatyka
Studia magisterskie rok 1 2022/2023

Plan Prezentacji

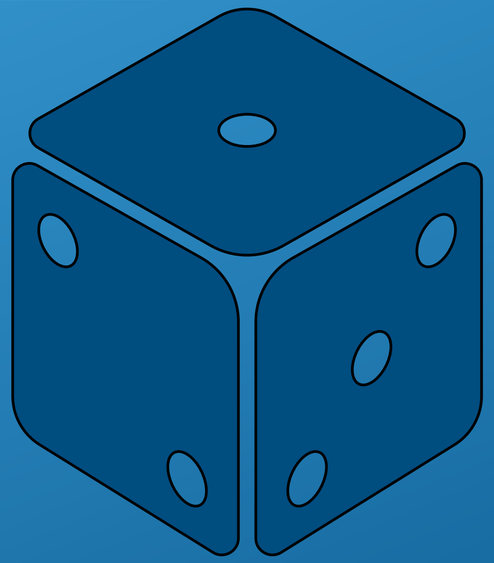


- Prezentacja problemu
- Rozwiązanie problemu
- Przedstawienie wyników
- Kod
- Podsumowanie i porównanie wyników

Problem



Estymacja testu Shapiro-Wilka przy pomocy metody Monte Carlo



Wady i Zalety Metody MC

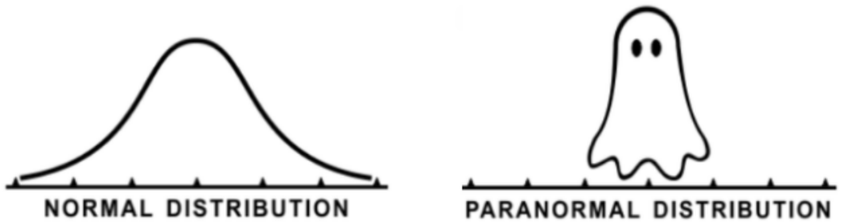
Wady	Zalety
Obecność błędu - wynik jest przybliżeniem	Możliwość rozwiązywania skomplikowanych problemów matematycznych
Wpływ jakości generatora liczb pseudolosowych na jakość przybliżenia (wyniku)	Zastąpienie analitycznego rozumowania na statystyczne podejście do zagadnienia
Często mało koszystne poprawianie dokładności	Przyzwolenie na ignorowanie praw, zasad i wzorów, dzięki zastosowaniu wnioskowania statystycznego
Skończona i określona liczba próby	Malejący czas wymagany na rozwiąza nie problemu (dzięki progresywnej mocy obliczeniowej komputerów)

Test Shapiro-Wilka

Test służący do oceny, czy zebrane przez nas wyniki od badanych osób posiadają rozkład normalny. Hipoteza zerowa dla tego testu zakłada, że nasza próba badawcza pochodzi z populacji o normalnym rozkładzie.

```
import scipy.stats as stats
stats.shapiro(random_data)
```

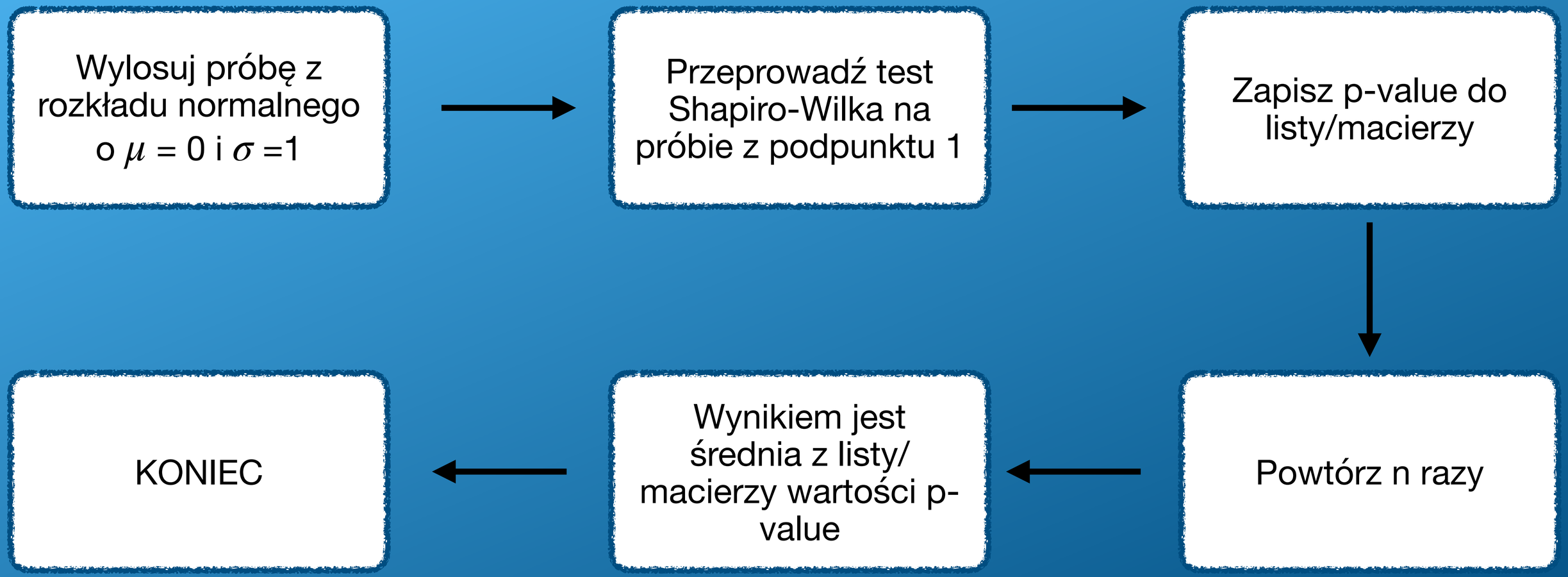
```
shapiro.test(random_data)
```



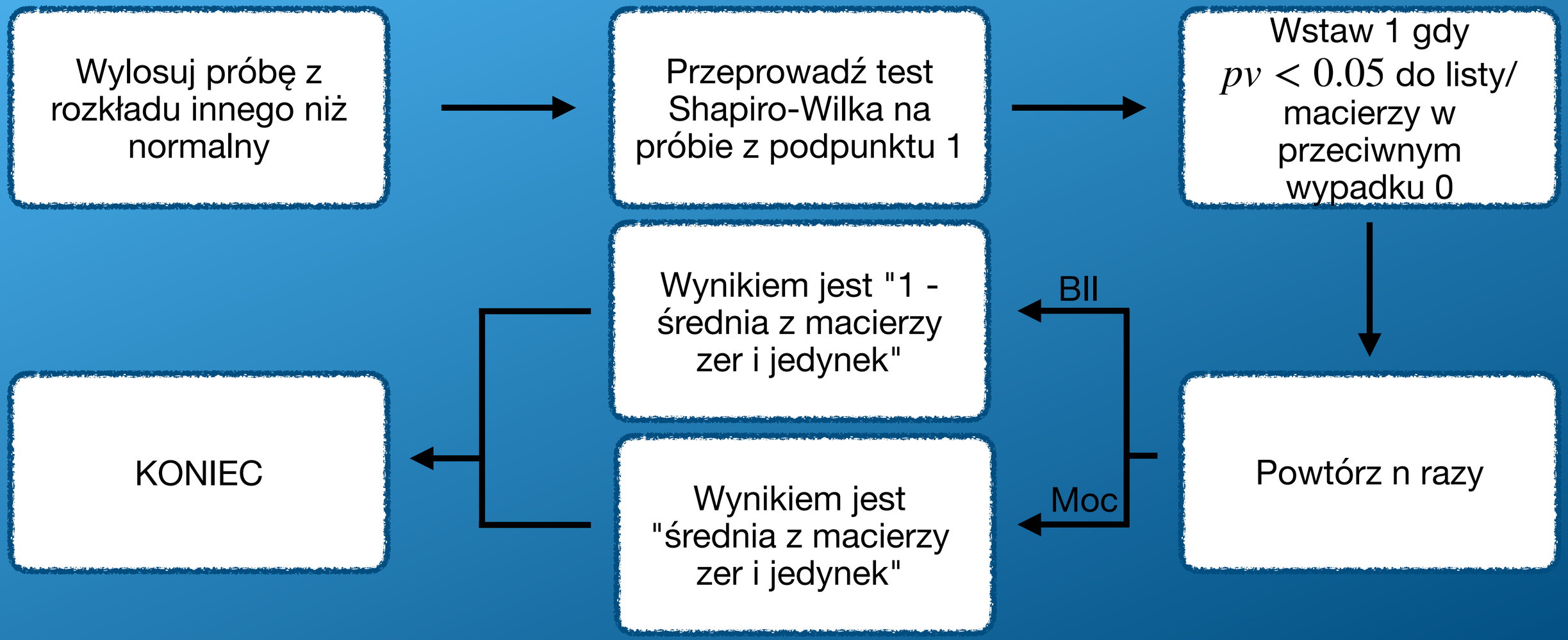
en.wikipedia.org

en.wikipedia.org

Algorytm P-value



Algorytm Błędu II Rodzaju i Mocy



Algorytm Błędu Bezwzględnego

Przed pętlą znajdź p-value dla rozkładu teoretycznego.

Wylosuj próbę z rozkładu normalnego o $\mu = 0$ i $\sigma = 1$.

→

Przeprowadź test Shapiro-Wilka na próbie z podpunktu 1.

→

Zapisz p-value do listy/macierzy

↓

Powtórz n razy

←

Wynikiem jest wartość:
$|pv(teo.) - pv(MC)|$

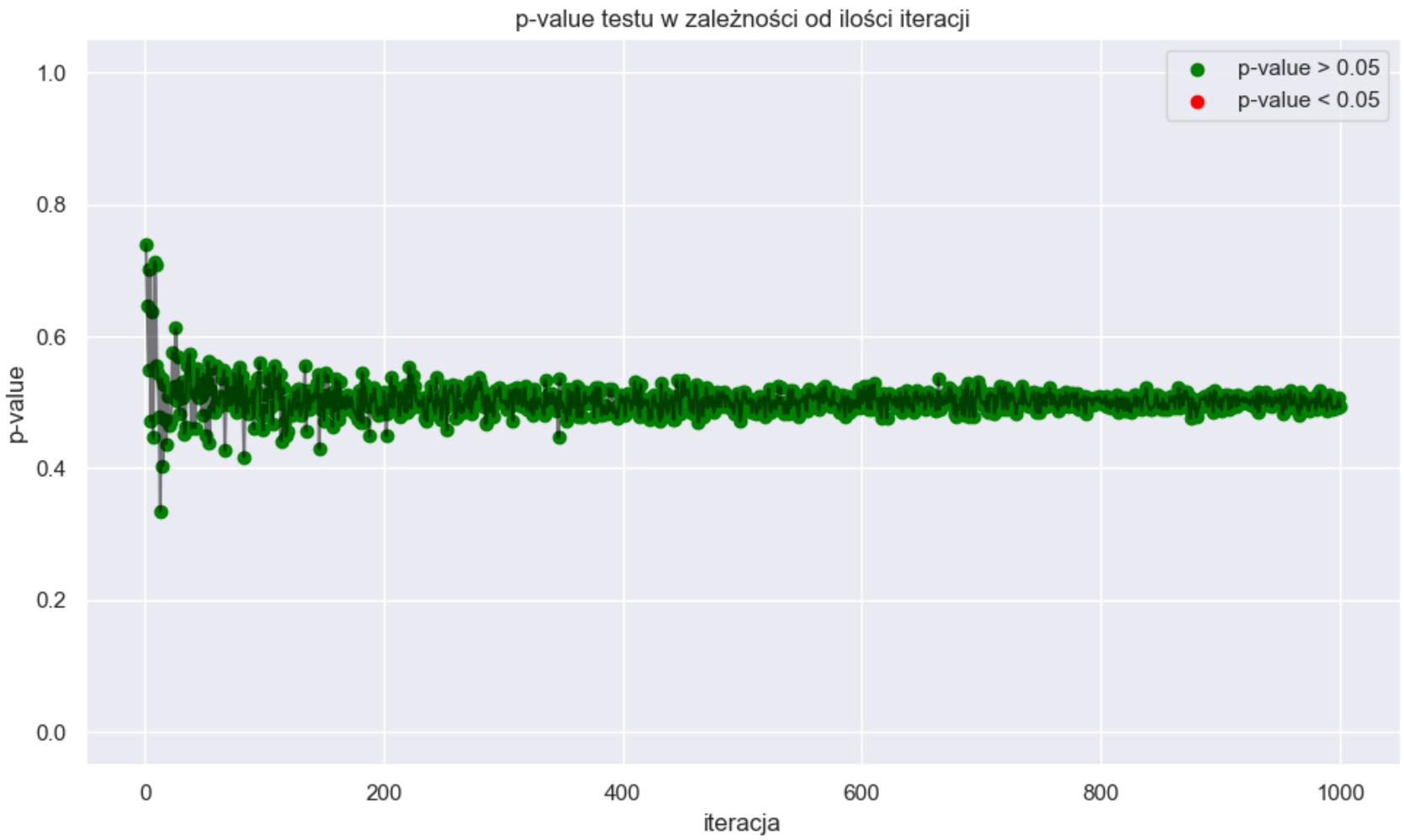
←

KONIEC

Wyniki

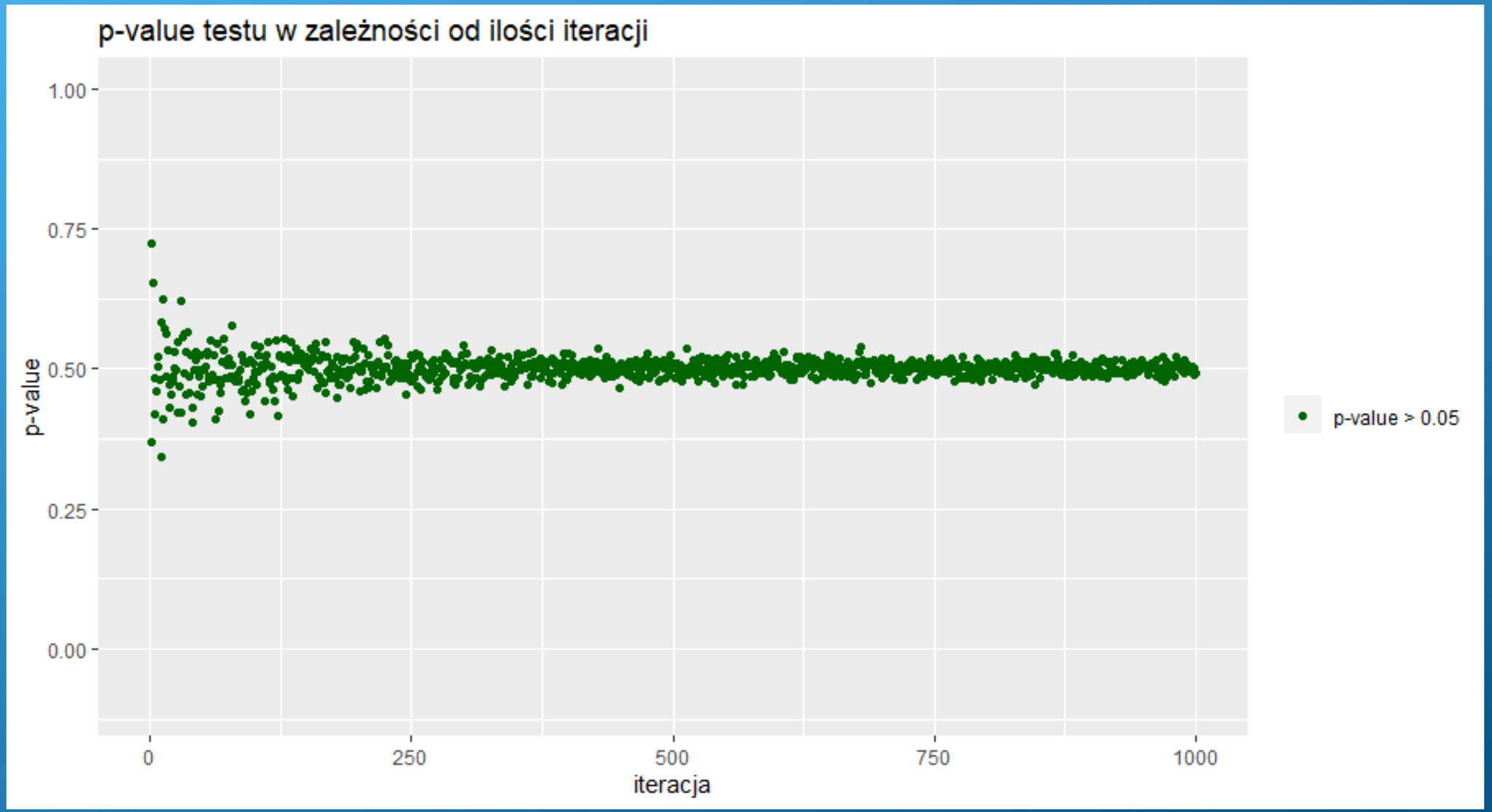

en.wikipedia.org

P-value


p-value testu w zależności od ilości iteracji
p-value > 0.05
p-value < 0.05
1.0
0.8
0.6
0.4
0.2
0.0
p-value
0
200
400
600
800
1000
iteracja

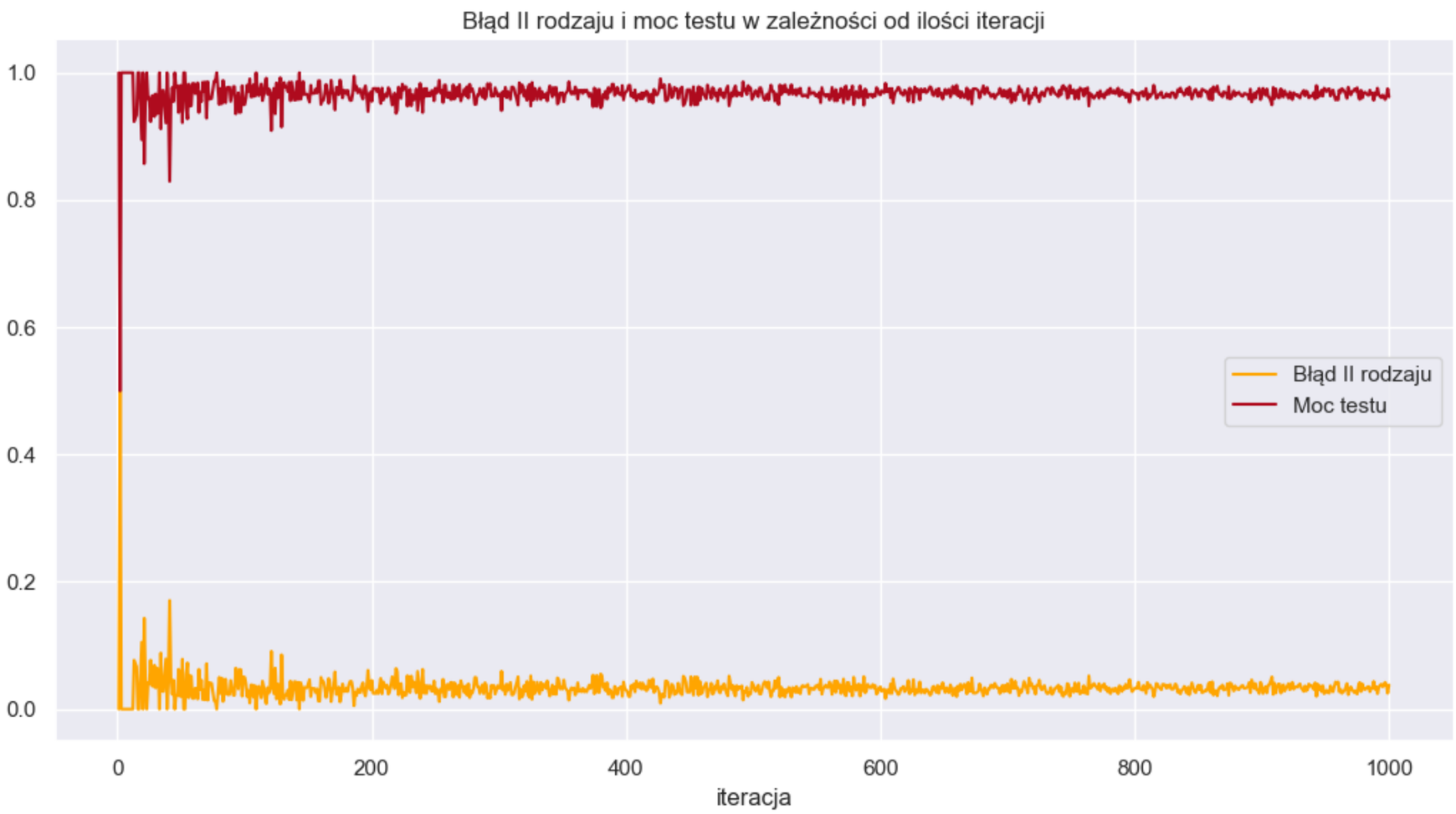

en.wikipedia.org

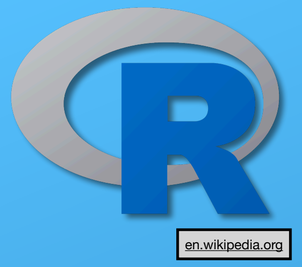
P-value


p-value testu w zależności od ilości iteracji
1.00
0.75
0.50
0.25
0.00
p-value
0
250
500
750
1000
iteracja
p-value > 0.05

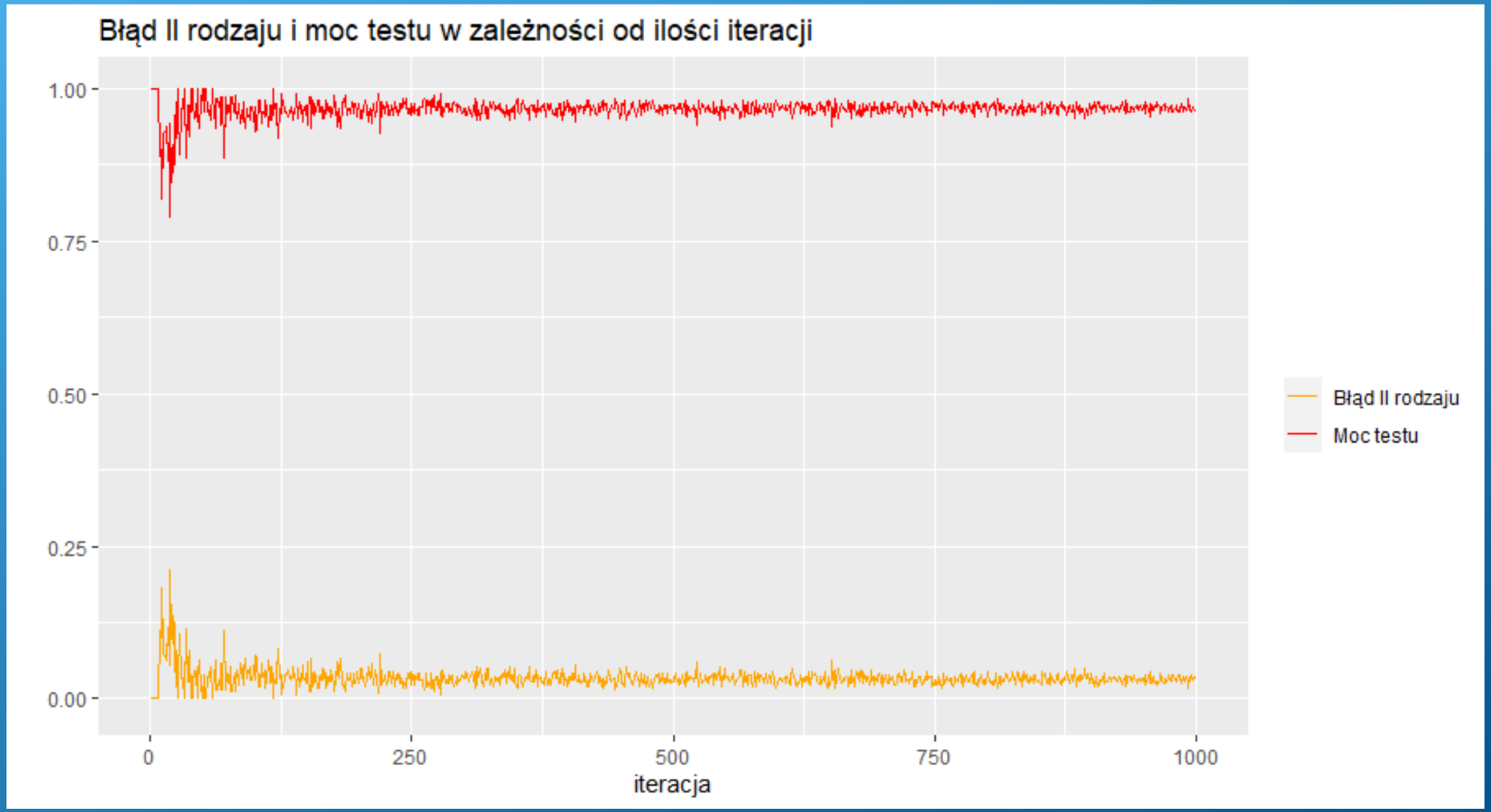
en.wikipedia.org

Błąd II Rodzaju i Moc

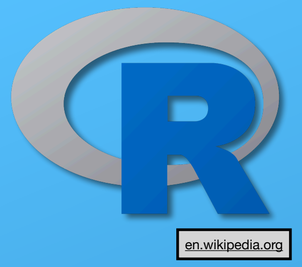




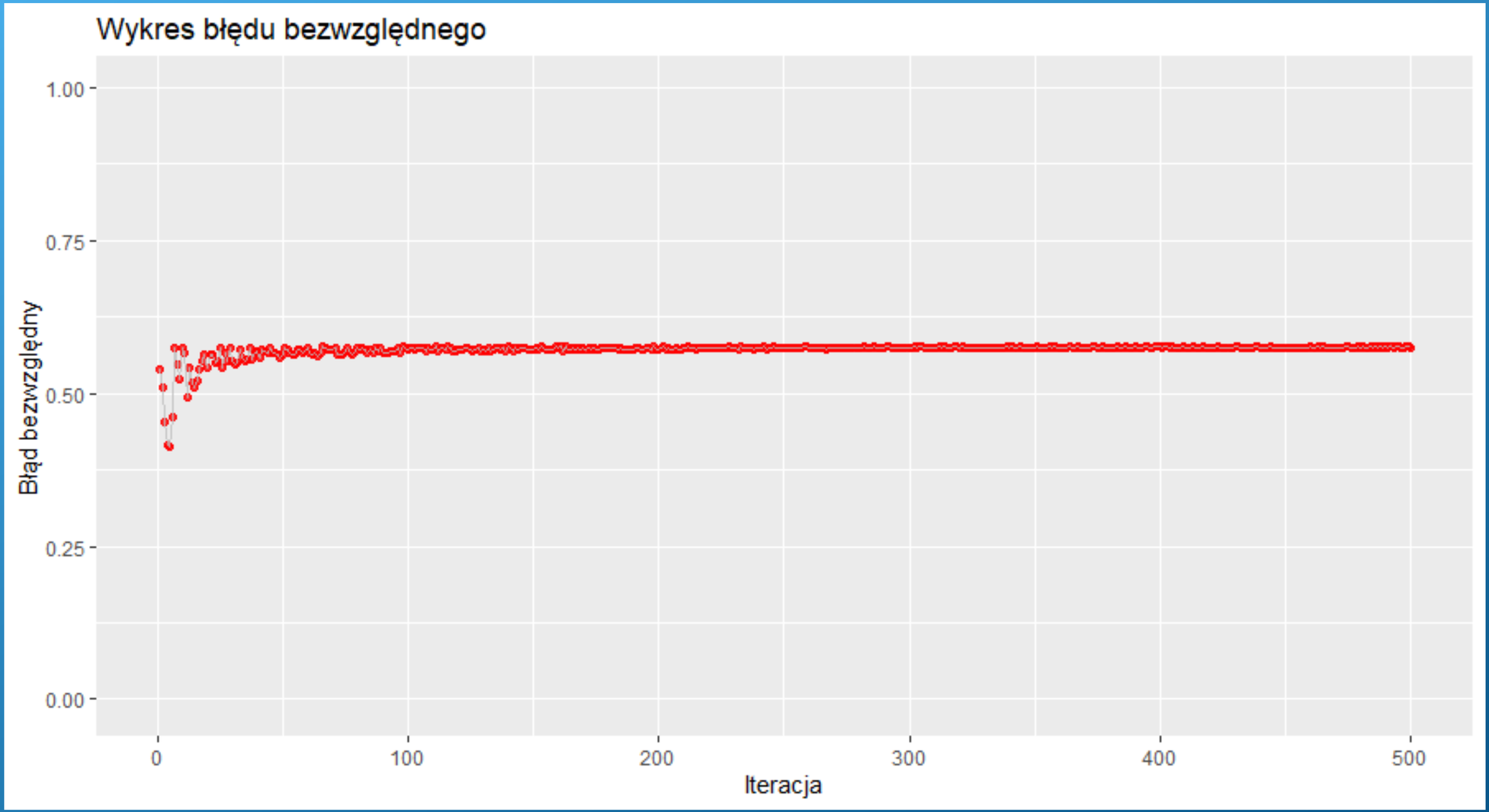
Błąd II Rodzaju i Moc



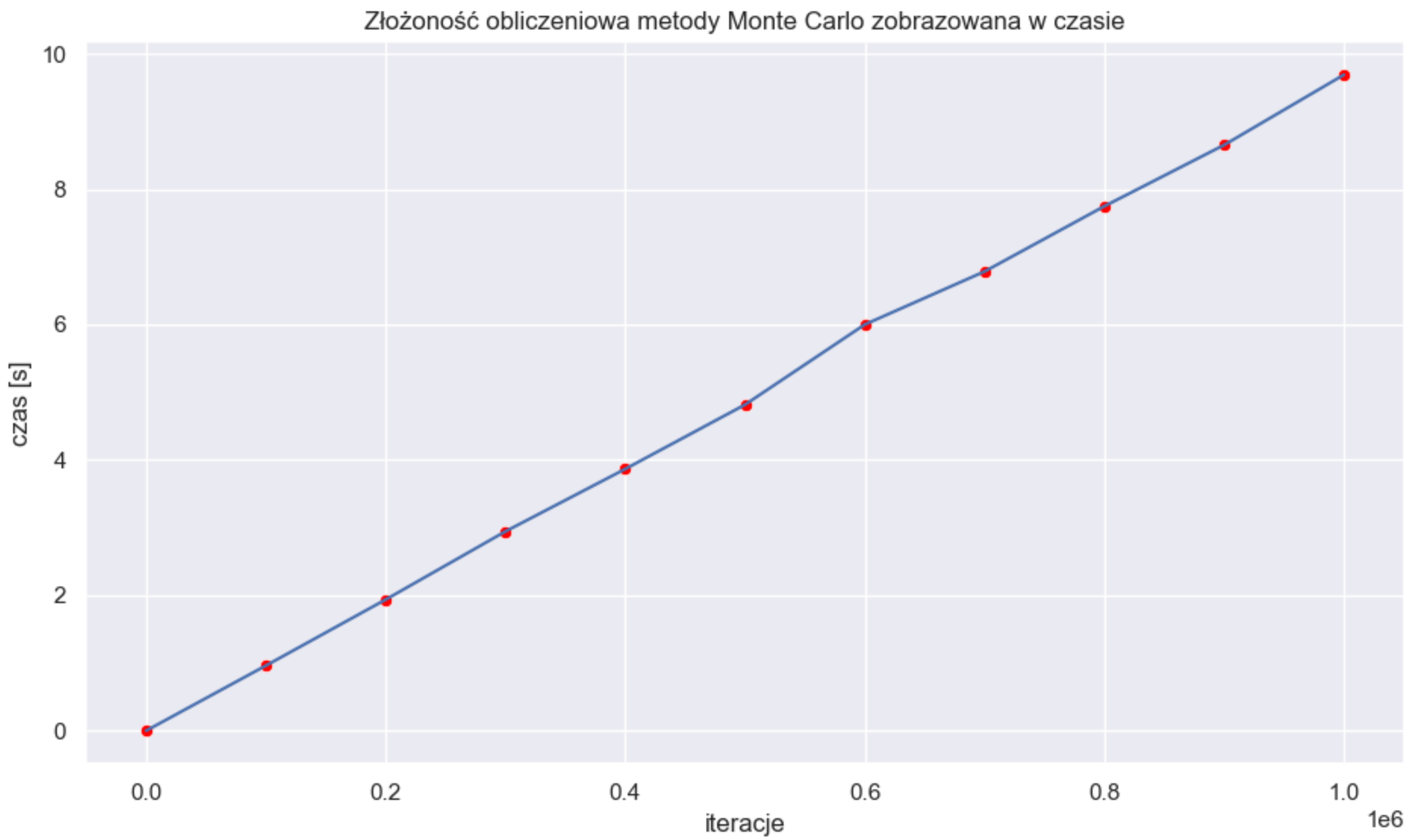
Błąd Bezwzględny

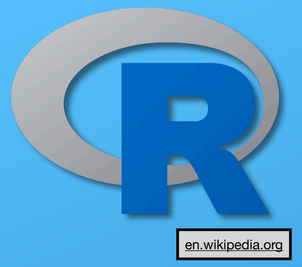


Błąd Bezwzględny

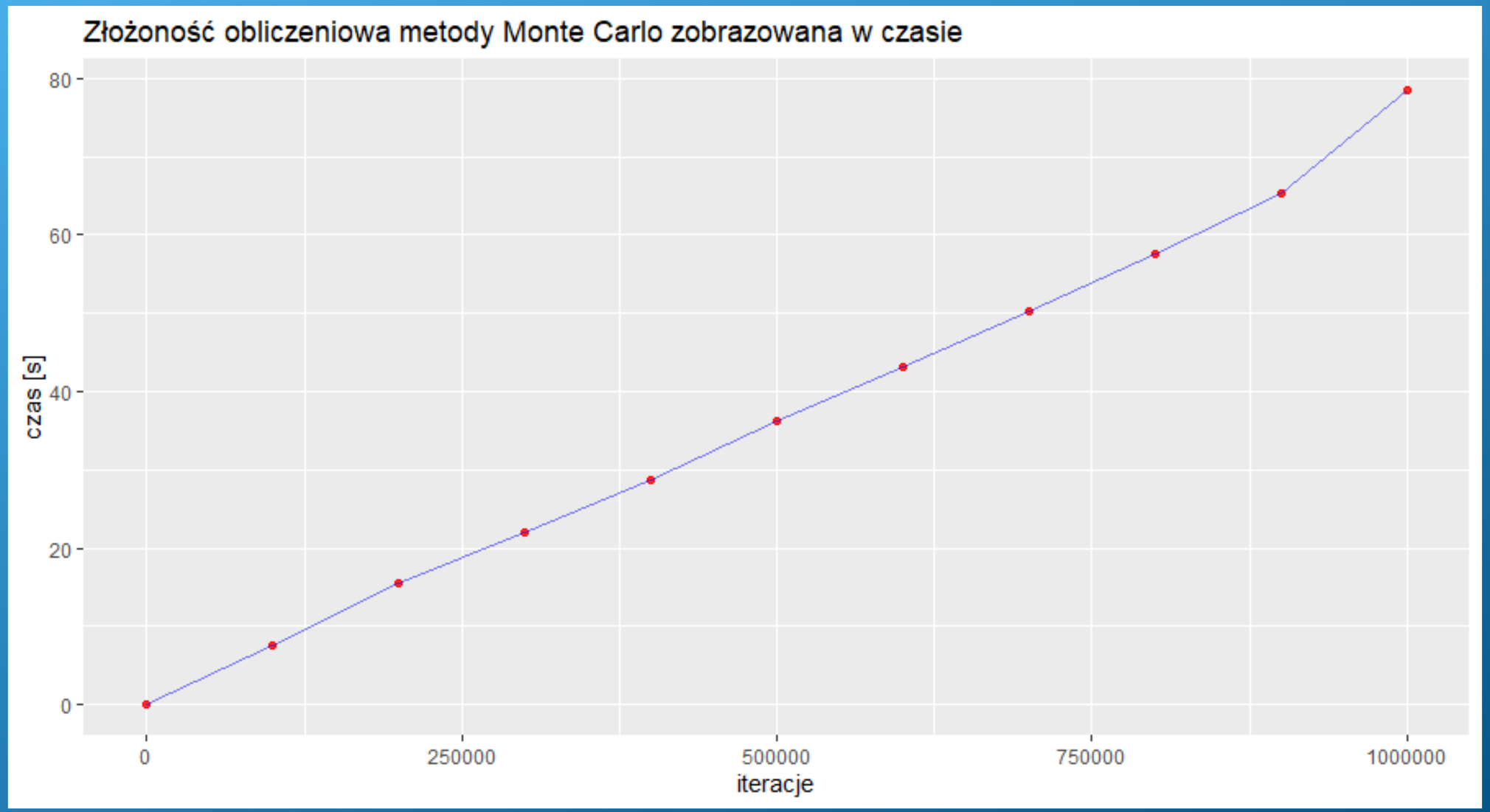


Złożoność Obliczeniowa





Złożoność Obliczeniowa



Kod

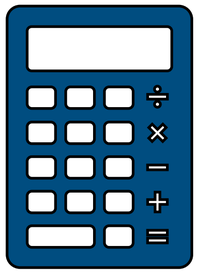
```
for index, n_iter in enumerate(iterations):
    pv_test = np.zeros(n_iter) # Macierz zer na p-value z każdego powtórzenia testu
    po_test = np.zeros(n_iter) # Macierz zer na p-value < alpha z każdego powtórzenia testu
    # Wielokrotne powtarzanie testu shapiro-wilka i zapisywanie p-value do macierzy
    for i in range(0,n_iter):
        random_data = np.random.normal(0, 1, 30)
        other_data = stats.expon.rvs(size = 30)
        p1 = stats.shapiro(random_data).pvalue
        p2 = stats.shapiro(other_data).pvalue
        # Dodawanie p-value do macierzy
        pv_test[i] = p1
        # Wstawianie 1, gdy p-value < alpha
        po_test[i] = int(p2 <= alpha)
    # Przypisywanie danemu indeksowi odpowiedającemu iteracji w liście p_values średnie wszystkich p-value z MC
    p_values[index] = np.mean(pv_test)
    powers[index] = np.mean(po_test)
    blad_II_r[index] = 1 - np.mean(po_test)
```

en.wikipedia.org

Kod

```
for (iter in iterations) {
  test1 = numeric(iter)
  test2 = numeric(iter)
  for (j in 1:iter) {
    x = rnorm(n,0,10)
    y = rexp(n)
    test1[j] = shapiro.test(x)$p.value
    test2[j] = as.integer(shapiro.test(y)$p.value <= alpha)
  }
  p_values = append(p_values, mean(test1))
  powers = append(powers, mean(test2))
  blad_II_r = append(blad_II_r, 1-mean(test2))
}
dane1 = data.frame(cbind(iterations, p_values))
dane1$alpha_status <- ifelse(dane1$p_values > alpha, "p-value > 0.05", "p-value < 0.05")
dane2 = data.frame(cbind(iterations, powers))
dane3 = data.frame(cbind(iterations, blad_II_r))
```

Podsumowanie



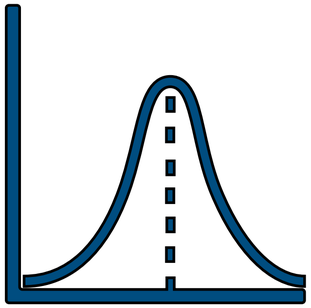
✓ **Python** okazał się rozwiązywać **szybciej** całe zadanie niż **R**.

✓ Wyniki pomiędzy oprogramowaniami **różnią się** przez użycie różnych generatorów liczb.

✓ Wyniki otrzymane przez użycie Pythona generują mniejsze błędy niż wyniki otrzymane z R.

✓ **P-value** w metodzie Monte Carlo **zbiega się do 0.05**.

✓ Wraz ze wzrostem iteracji **moc testu zbliża się do 1** a **błąd II rodzaju do 0**.



Projekt przygotowali

Kacper Kaszuba
Daria Plewa
Karol Rorat
Bioinformatyka
Studia magisterskie rok 1 2022/2023

Statystyczne Modelowanie Danych Biologicznych