# Konstrukcja generatorów liczb losowych i symulacyjna analiza ich własności

Stanisław Olek

---

## Spis treści

# 1 Problem 1 – Generator kongruencyjny

- Zaimplementuj liniowy generator kongruencyjny liczb pseudolosowych $x_1, x_2, \ldots$:

$$(s_0 = 1) \quad s_n = as_{n-1} \bmod m; \quad x_n = s_n/m, \quad n = 1, 2, \ldots, \text{gdzie}$$

  i $m = 37$, $a = 19$,

  ii $m = 2^{31} - 1$, $a = 39373$.

- Sprawdź empirycznie, jaki okres ma pierwszy z tych generatorów.
- Narysuj 50 kolejnych liczb pseudolosowych j.w. jako punkty na odcinku jednostkowym. Narysuj histogram. Skomentuj rysunek.
- Narysuj 50 kolejnych par liczb pseudolosowych j.w. jako punkty na kwadracie jednostkowym. Skomentuj rysunek.

## Rozwiązanie:

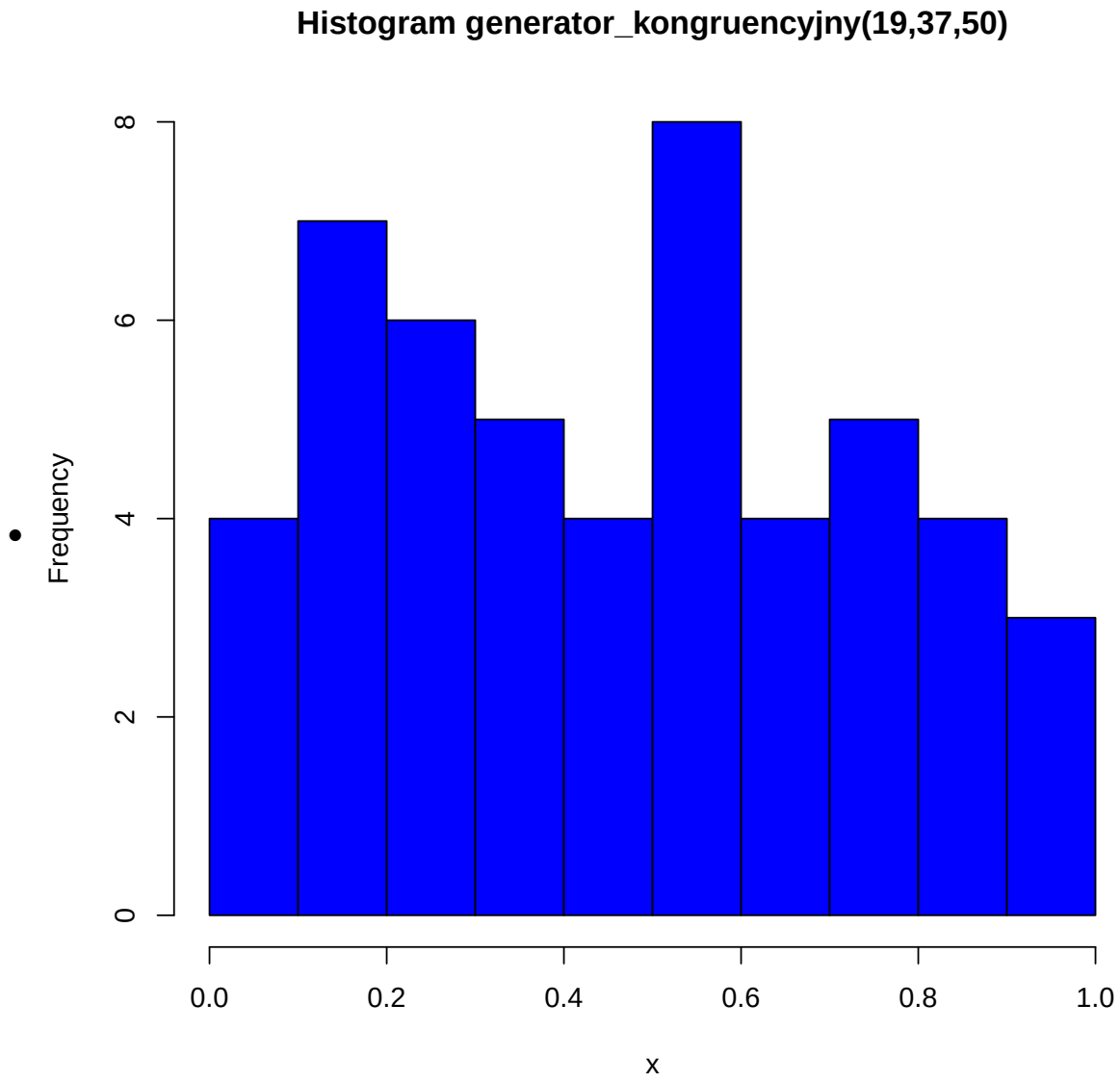
- ```
  generator_kongruencyjny <- function(a, m, n) {
    s_n = c(1)
    for (i in 1:n) {
      k <- s_n[length(s_n)]
      s_n <- c(s_n, (a * k) %% m)
    }
    x_n <- s_n / m
    return(x_n[-1])
  }
  ```

**i** $m = 37, a = 19$

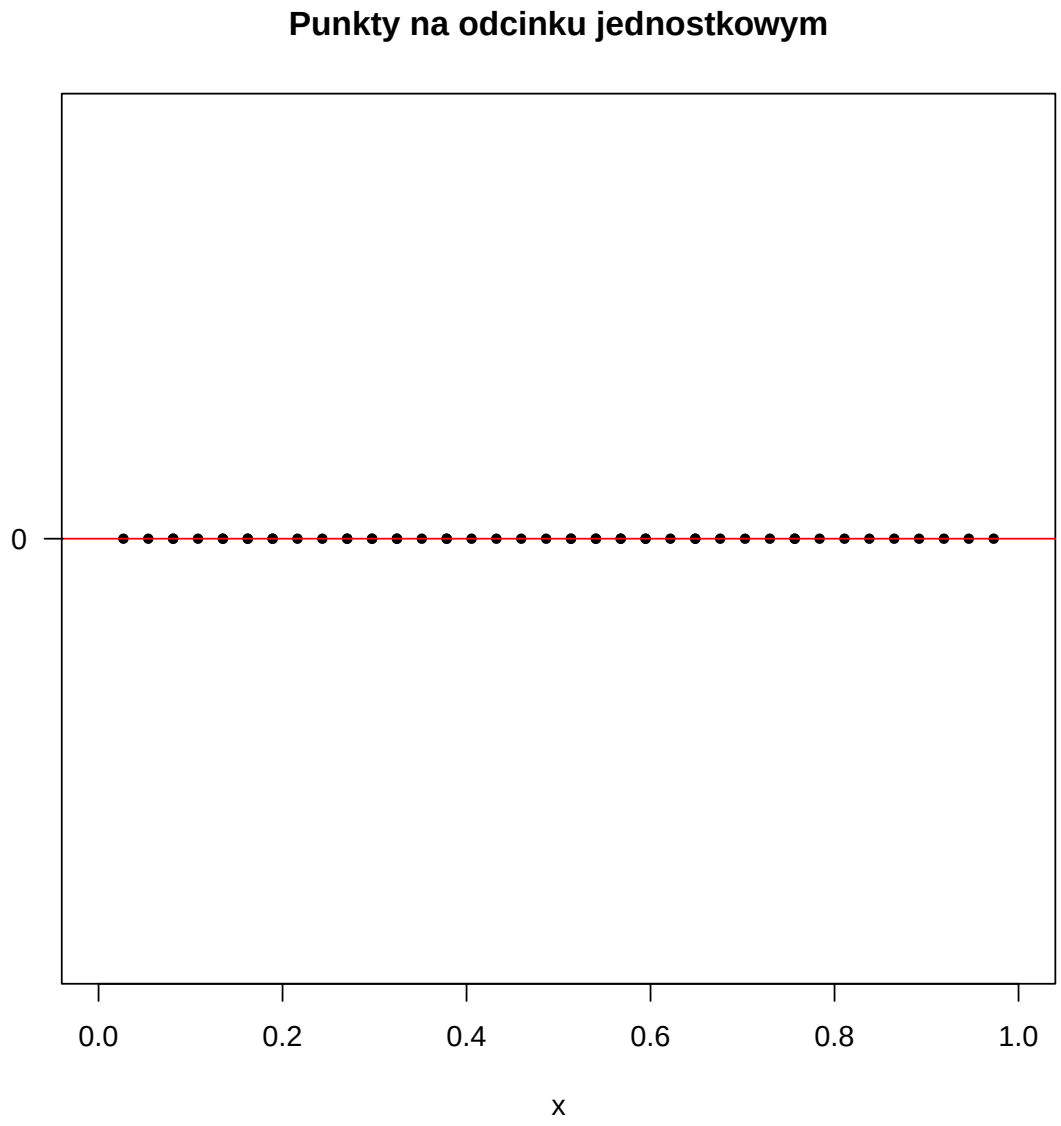
```
##  [1] 0.51351351 0.75675676 0.37837838 0.18918919 0.59459459 0.29729730
##  [7] 0.64864865 0.32432432 0.16216216 0.08108108 0.54054054 0.27027027
## [13] 0.13513514 0.56756757 0.78378378 0.89189189 0.94594595 0.97297297
## [19] 0.48648649 0.24324324 0.62162162 0.81081081 0.40540541 0.70270270
## [25] 0.35135135 0.67567568 0.83783784 0.91891892 0.45945946 0.72972973
## [31] 0.86486486 0.43243243 0.21621622 0.10810811 0.05405405 0.02702703
## [37] 0.51351351 0.75675676 0.37837838 0.18918919 0.59459459 0.29729730
## [43] 0.64864865 0.32432432 0.16216216 0.08108108 0.54054054 0.27027027
## [49] 0.13513514 0.56756757
```

**ii** $m = 2^{31} - 1, a = 39373$

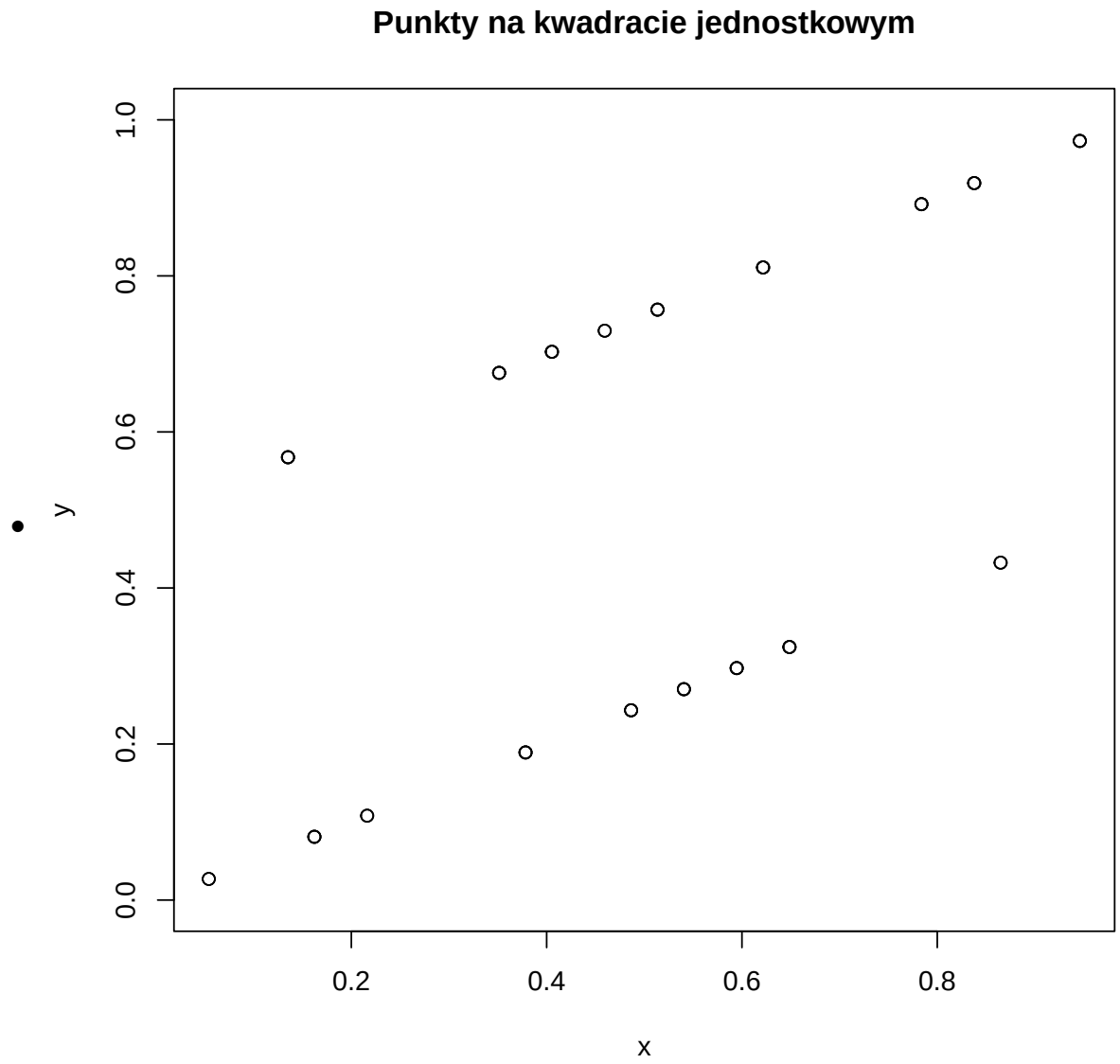
```
##  [1] 1.833448e-05 7.218836e-01 7.212037e-01 9.520167e-01 7.553457e-01
##  [6] 2.248237e-01 9.826929e-01 5.675375e-01 6.550636e-01 8.177157e-01
## [11] 9.210877e-01 9.843399e-01 4.165816e-01 6.655500e-02 4.699210e-01
## [16] 2.001585e-01 8.421137e-01 5.425512e-01 8.698889e-01 1.365497e-01
```
```

```
## [21] 3.718927e-01 5.299994e-01 6.661485e-01 2.644647e-01 7.689364e-01
## [26] 3.318948e-01 6.932724e-01 2.146284e-01 5.649799e-01 9.547553e-01
## [31] 5.812586e-01 8.954412e-01 2.078554e-01 8.923246e-01 4.948566e-01
## [36] 9.879381e-01 8.505122e-02 7.217564e-01 7.158718e-01 1.966913e-02
## [41] 4.328017e-01 7.011114e-01 8.611099e-01 4.784881e-01 5.107689e-01
## [46] 5.052171e-01 9.129803e-01 7.747585e-01 5.649881e-01 2.777356e-01
```

- ```
  okres <- function(a, m) {
    s_n = 1
    wygenerowane = c(s_n)
    okres = 1
    while(TRUE) {
      s_n <- (a * s_n) %% m
      if(s_n %in% wygenerowane) {
        break
      } else {
        wygenerowane <- c(wygenerowane, s_n)
        okres <- okres + 1
      }
    }
    return(okres)
  }
  ```

```
## Okres pierwszego generatora: 36
```
```




Jeśli spojrzymy na histogram to wydaje się, że dane nie są idealnie równomiernie rozłożone, ponieważ liczba wystąpień w poszczególnych przedziałach różni się. Najwięcej obserwacji znajduje się w przedziale od 0.5 do 0.6, gdzie mamy 8 wystąpień. Najmniej obserwacji jest w ostatnim przedziale od 0.9 do 1.0, gdzie mamy tylko 3 wystąpienia. Pozostałe przedziały mają liczbę wystąpień między 4 a 7. Taki rozkład może sugerować, że dane nie są idealnie równomiernie rozłożone, ale nie ma też bardzo dużych odchyleń od równomierności.




Jeśli spojrzymy na rozkład punktów na odcinku jednostkowym, możemy zauważyć, że liczby pseudolosowe wydają się być stosunkowo równomiernie rozłożone bez widocznych wzorców lub skupisk, pomimo że wartości m i a są stosunkowo małe.

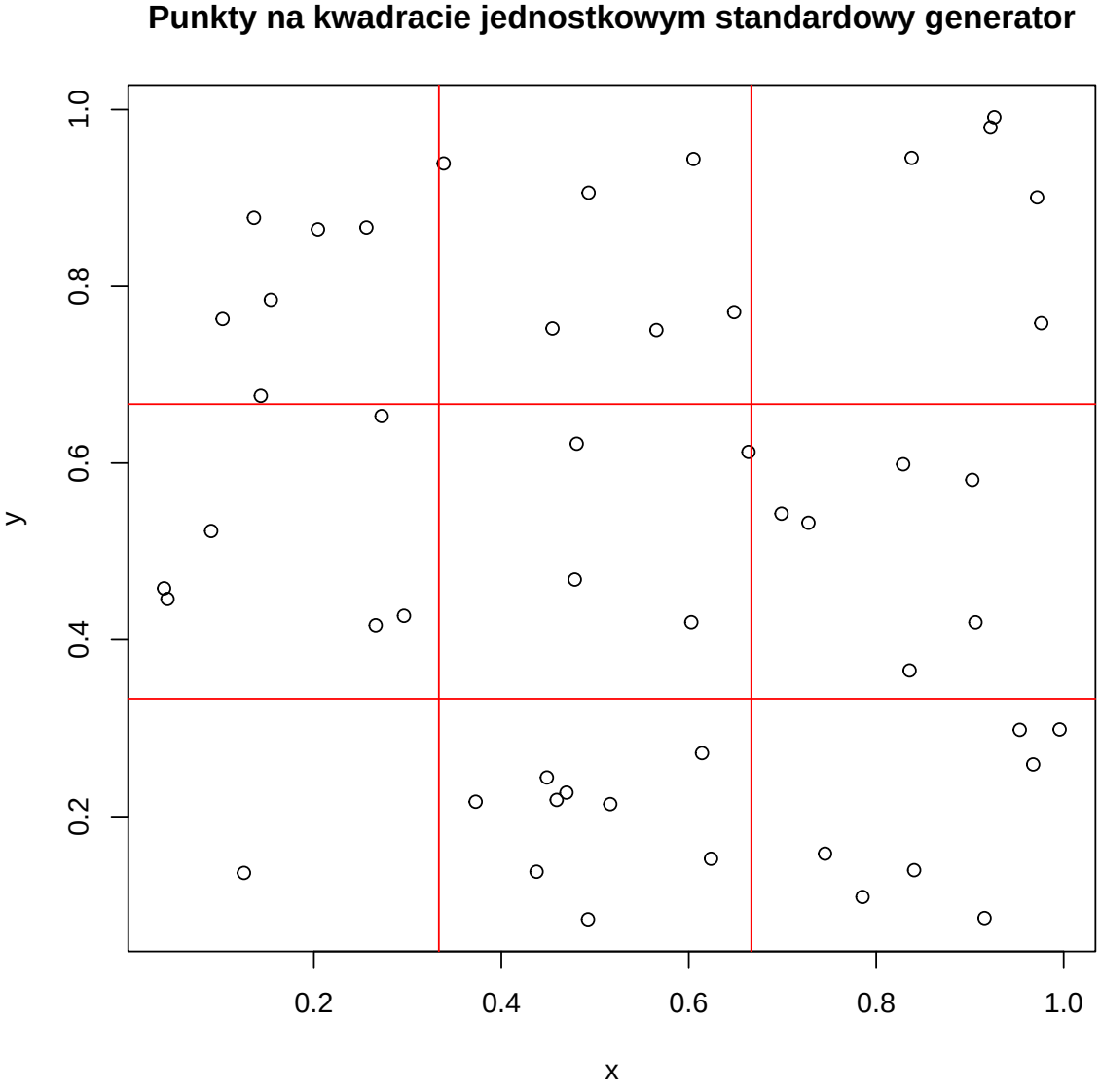



Z uwagi na charakterystykę liniowych generatorów kongruentnych i małą wartość m punkty wykazują wyraźną regularność - układają się w linie. Świadczy to o tym, że liczby nie są równomiernie rozłożona na kwadracie, co jest typową cechą dla generatorów kongruentnych i małych wartości parametrów.

# 2 Problem 2 – Generator standardowy w R

- Narysuj 50 kolejnych par liczb pseudolosowych jako punkty na kwadracie jednostkowym dla standardowego generatora zaimplementowanego w pakiecie R.
- Czy otrzymane pary liczb pseudolosowych mają rozkład jednostajny na kwadracie $[0,1]^2$? Zaproponuj, jak to weryfikować. W jaki sposób ta własność łączy się z własnością niezależności kolejnych liczb?
- Jak inicjować obliczenia, żeby za każdym razem otrzymywać ten sam ciąg 50-ciu par? Czy taka powtarzalność (replikowalność) jest sprzeczna z ideą symulacji pseudolosowych?

## Rozwiązanie:

- 


Tak, jeśli używamy funkcji **runif()**, otrzymujemy liczby pseudolosowe, które mają rozkład jednostajny na przedziale od 0 do 1. Kiedy generujemy pary takich liczb, zakładamy, że każda para jest punktem na kwadracie jednostkowym $[0,1]^2$, gdzie każdy punkt ma taką

samą szansę na pojawienie się w dowolnym miejscu na tym kwadracie. Aby zweryfikować, czy otrzymane pary liczb pseudolosowych mają rozkład jednostajny, można przeprowadzić test statystyczny, na przykład test chi-kwadrat Pearsona. Test ten pozwala sprawdzić, czy obserwowane częstości punktów w różnych obszarach kwadratu jednostkowego są zgodne z oczekiwaniami dla rozkładu jednostajnego.

Oto przykładowy sposób weryfikacji:

1. Podziel kwadrat jednostkowy na mniejsze, równomierne kwadraty (na przykład 3x3, co daje 9 mniejszych kwadratów).
2. Policz, ile punktów wpada do każdego z mniejszych kwadratów.
3. Porównaj obserwowane liczby z oczekiwanymi liczbami dla rozkładu jednostajnego.
4. Użyj testu chi-kwadrat do obliczenia p-wartości i zdecyduj, czy odrzucić hipotezę zerową o rozkładzie jednostajnym.

Własność niezależności kolejnych liczb w generatorze liczb pseudolosowych jest ważna, ponieważ oznacza, że każda liczba jest generowana bez wpływu na poprzednie liczby. Dla rozkładu jednostajnego na kwadracie jednostkowym oznacza to, że położenie jednego punktu nie wpływa na położenie kolejnego. Jeśli liczby nie byłyby niezależne, moglibyśmy zauważyć wzorce lub "skupiska" punktów na kwadracie, co wskazywałoby na problem z generatorem liczb pseudolosowych.

- Aby za każdym razem otrzymywać ten sam ciąg 50-ciu par liczb pseudolosowych, należy zainicjować generator liczb pseudolosowych tym samym "ziarnem", wykorzystując komendę:

```
set.seed(n)
```

  gdzie **n** jest pewną liczbą naturalną.

- Powtarzalność nie jest sprzeczna z ideą symulacji pseudolosowych. Wręcz przeciwnie, jest to jedna z pożądanych cech generatorów liczb pseudolosowych. Generatory liczb pseudolosowych są deterministyczne, co oznacza, że dla tego samego ziarna zawsze wygenerują ten sam ciąg liczb. To pozwala na replikację eksperymentów i testów. Idea symulacji pseudolosowych polega na tym, że choć liczby generowane przez te algorytmy są deterministyczne (i w tym sensie nie są "prawdziwie" losowe), ich właściwości statystyczne sprawiają, że w wielu zastosowaniach są one wystarczająco "dobre" do naśladowania losowości.

# 3 Problem 3 – Symulacja dyskretnych zmiennych losowych

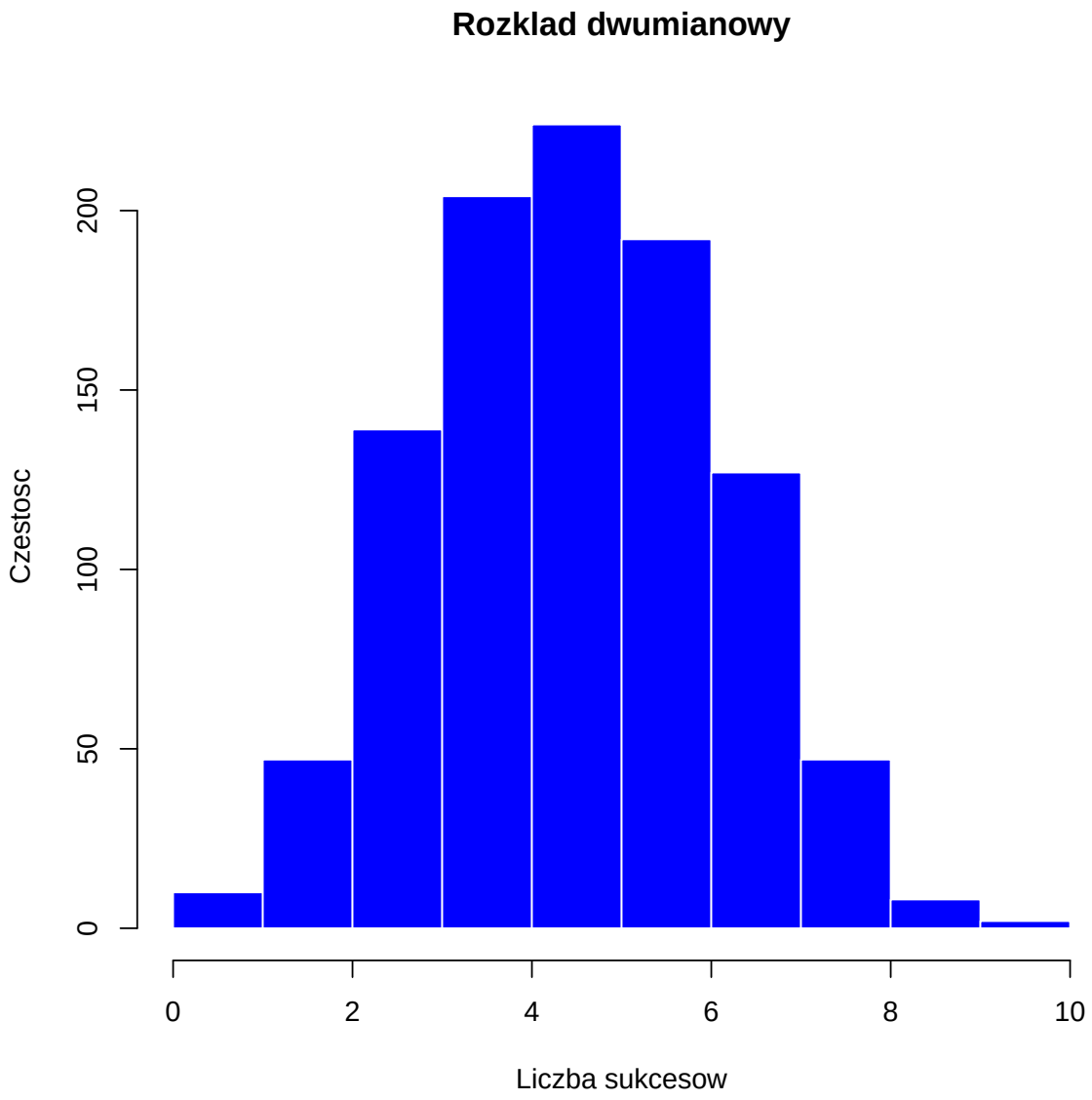
- Napisz generator liczb losowych o rozkładzie dwupunktowym: $\mathbb{P}(X = 1) = p$, $\mathbb{P}(X = 0) = 1 - p$, gdzie $p \in [0, 1]$.
- Napisz generator liczb losowych dla rozkładu dwumianowego. Narysuj histogram dla próby (prostej) z tego rozkładu.
- Napisz generator dla rozkładu Poissona. Wykonaj odręcznie obliczenia przygotowawcze. Działanie generatora sprawdź poprzez analizę histogramów lub w inny sposób.

## Rozwiązanie:

- ```
  gen_0_1 <- function(p) {
    x <- runif(1)
    if(x <= p){
      return(1)
    }
    else{
      return(0)
    }
  }

  generator_dwupunktowy <- function(n,p){
    x <- c(1)
    for (i in 1:n)
      x[i] <- gen_0_1(p)
    return(x)
  }
  ```

- ```
  generator_dwumianowy <- function(n, m, p) {
    x <- c(1)
    for (i in 1:n) {
      u = runif(m);
      b = (u < p);
      x[i] = sum(b)
    }
    return(x)
  }
  ```

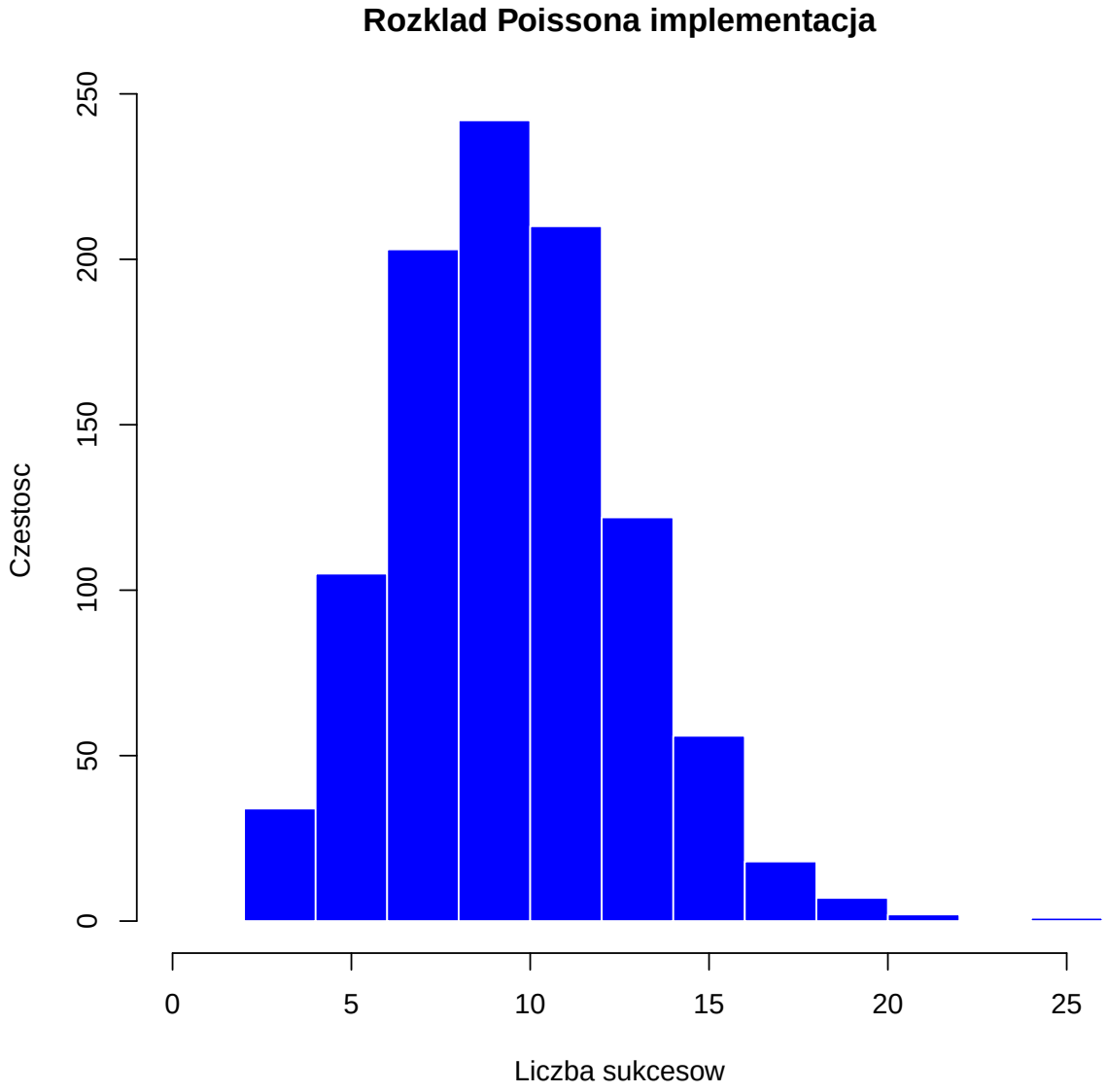
Generator rozkładu dwumianowego dla parametrów: $n = 100, m = 10, p = \frac{1}{2}$

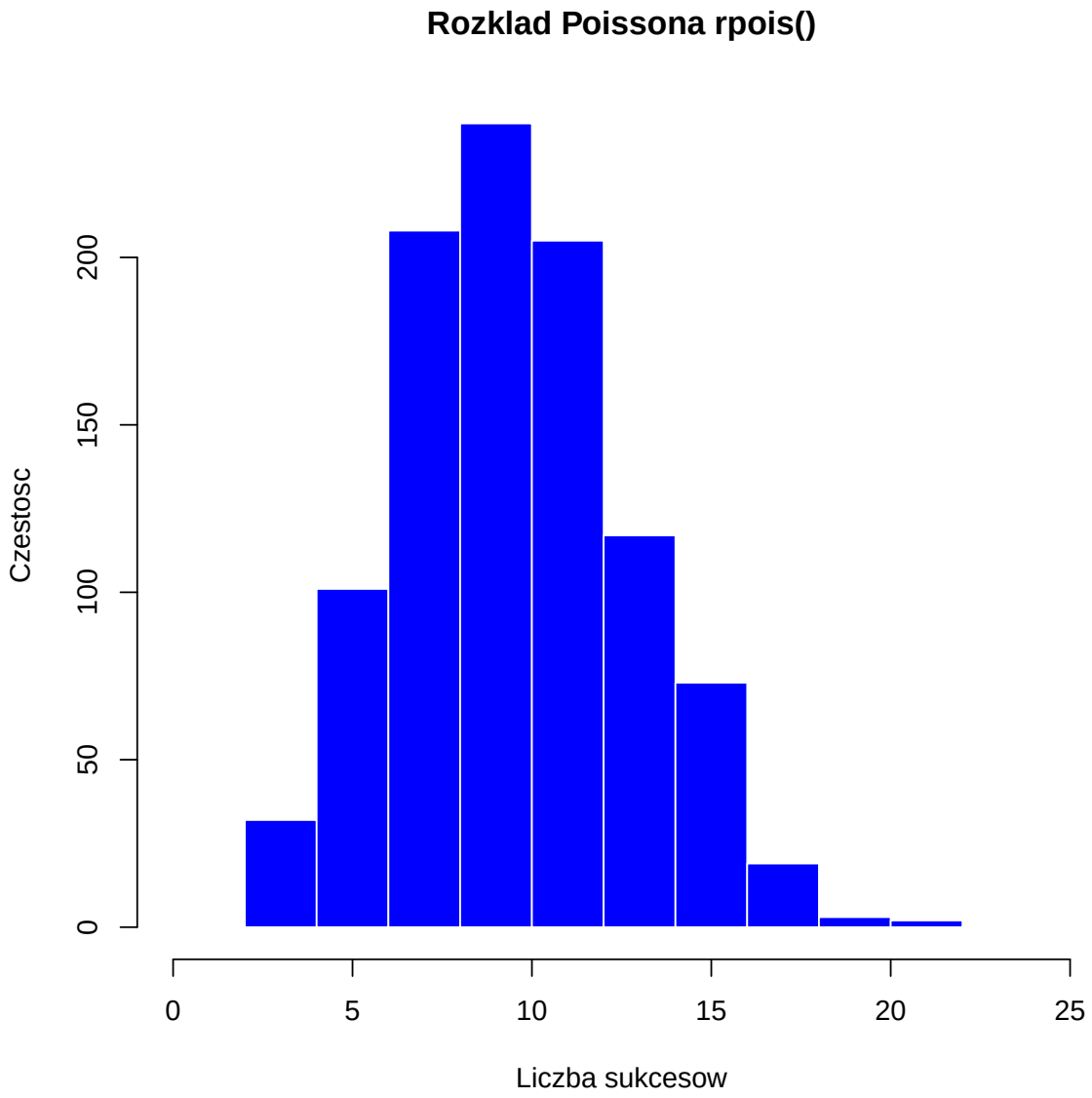



- Generator rozkładu Poissona dla parametrów: $n = 100, \lambda = 5$

```
generator_poissona <- function(n,lambda) {
  tab <- c(0)
  for(i in 1:n){
    L <- exp(-lambda)
    k <- 0
    iloczyn_p <- 1
    while(TRUE) {
      u <- runif(1)
      iloczyn_p <- iloczyn_p * u
      if(iloczyn_p >= L){
        k <- k + 1
      } else{
        tab[i] <- k
```

```
        break
      }
    }
  }
  return(tab)
}
```




Rozklad Poissona rpois()
Czestosc
0
50
100
150
200
0
5
10
15
20
25
Liczba sukcesow