«ustawienia$_{g}lobalne, echo = FALSE >>= library(knitr)library(xtable)pakietdotworzeniatabelwfor$
$opts_{c}hunk$set(fig.align='center', fig.pos='H') @

Metody Monte Carlo

Raport 1

Autor: Anna Bonikowska

17.03.2024

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

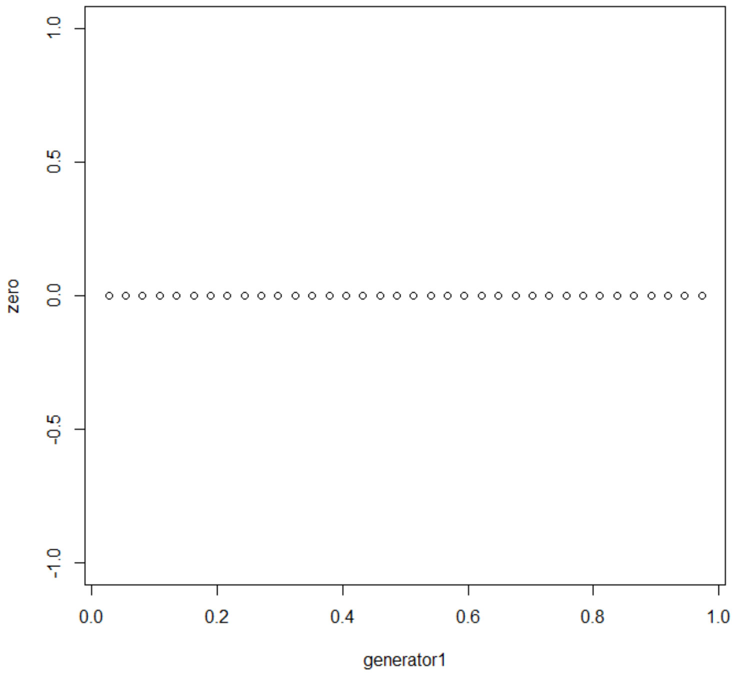
  ii $m = 2^{31} - 1$, $a = 39373$.

- 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.

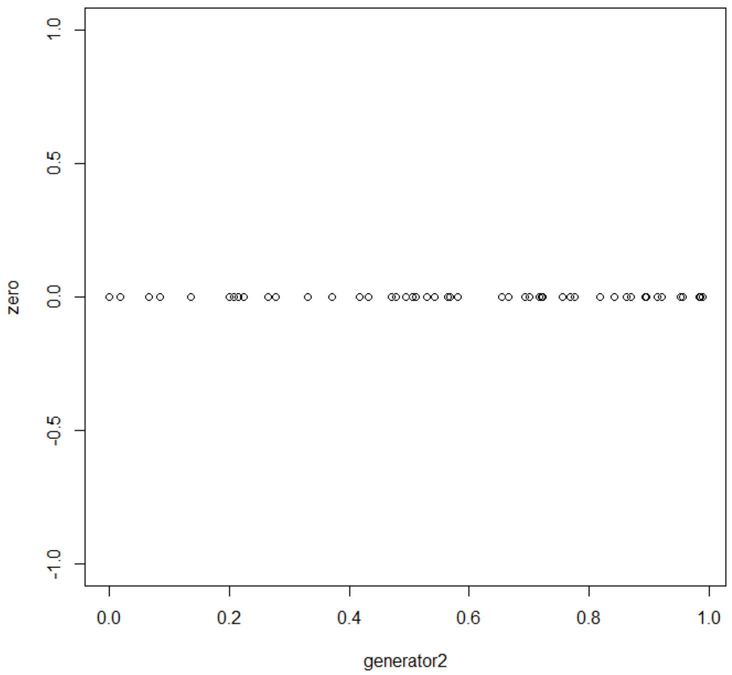
Rozwiązanie:

-
```
#ogólna funkcja generatorów liniowych
f <- function(a,m,n){
  s<- c(1)

  for( i in 1:n){

    k<-s[length(s)]
    s<-c(s,(a*k)%%m)
  }

  xn<-s/m

  return(xn[])

}

#zaimplementowane pozczególne generatory
generator1<-f(19,37,50)
generator2<-f(39373,2**31 -1,50)
```

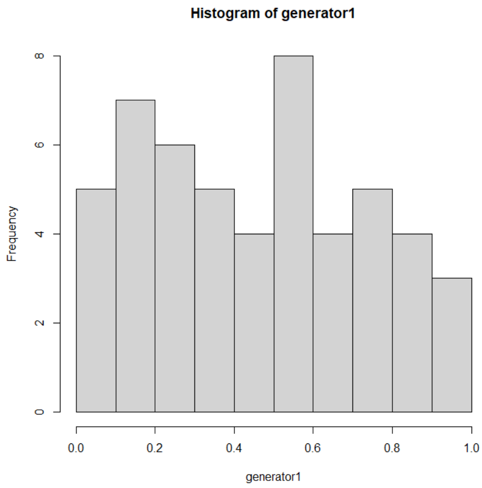
- pierwszy generator ma okres 37
-
```
#generowanie punktów na odcinku
zero <- rep(0,51)

plot(generator1,zero)

plot(generator2,zero)
```



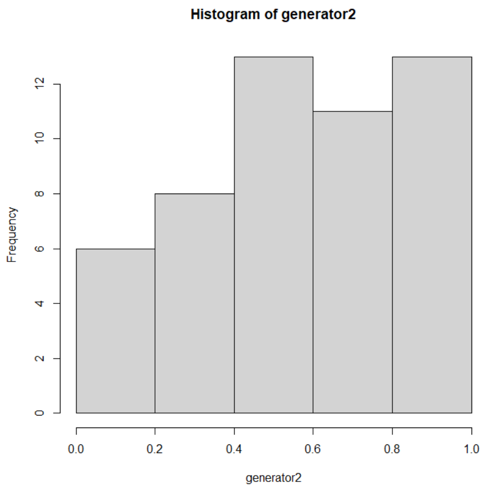
Rysunek 1: Generator 1 punkty na odcinku. Punkty są rozłożone w sposób regularny, co może oznaczać że nie jest to dobry generator liczb losowych



Rysunek 2: Generator 1 punkty na odcinku. Punkty są rozłożone w sposób nieregularny, co może oznaczać że jest to dobry generator liczb losowych

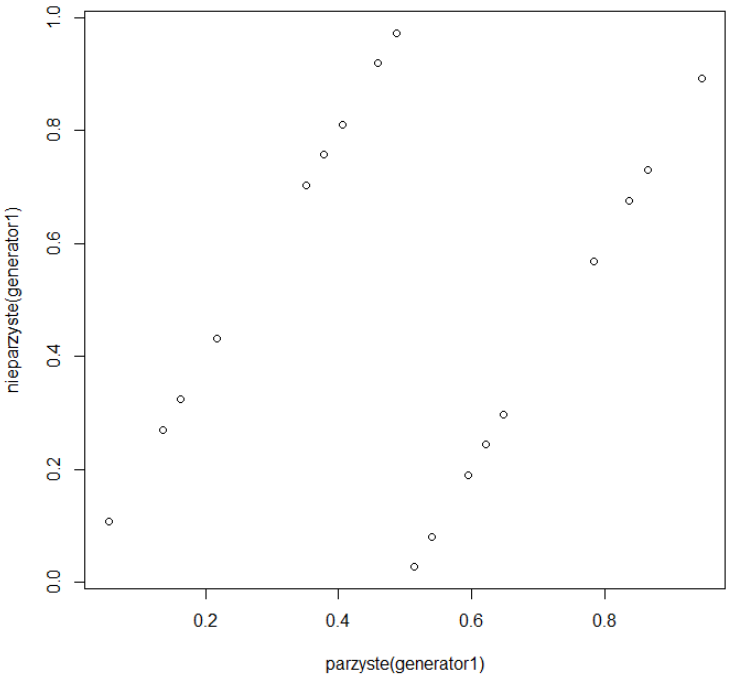


Rysunek 3: Histogram generatora 1. Wyglada dobrze.



Rysunek 4: Histogram generatora 2. Wyglada dobrze.

```
#generowanie histogramów
hist(generator1)
hist(generator2)
```

- ```
  #generowanie liczb pseudolosowych jako punkty na kwadracie

  parzyste <- function(x){

    k<-length(x)
    h<-x[2]
    a<-c(h)

    for(i in 2:(k/2)){
      a<-c(a,x[2*i])
    }
  ```
  ```



Rysunek 5: Generator 1 jako punkty na kwadracie. Punkty tworzą linie. Nie wygląda to dobrze

```
  return(a[])
}

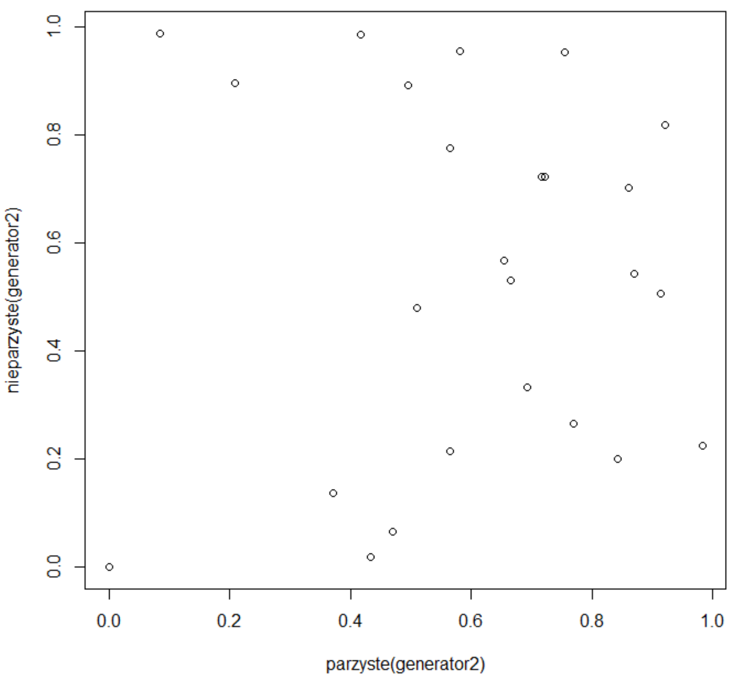

nieparzyste <- function(x){

  k<-length(x)
  h<-x[1]
  a<-c(h)

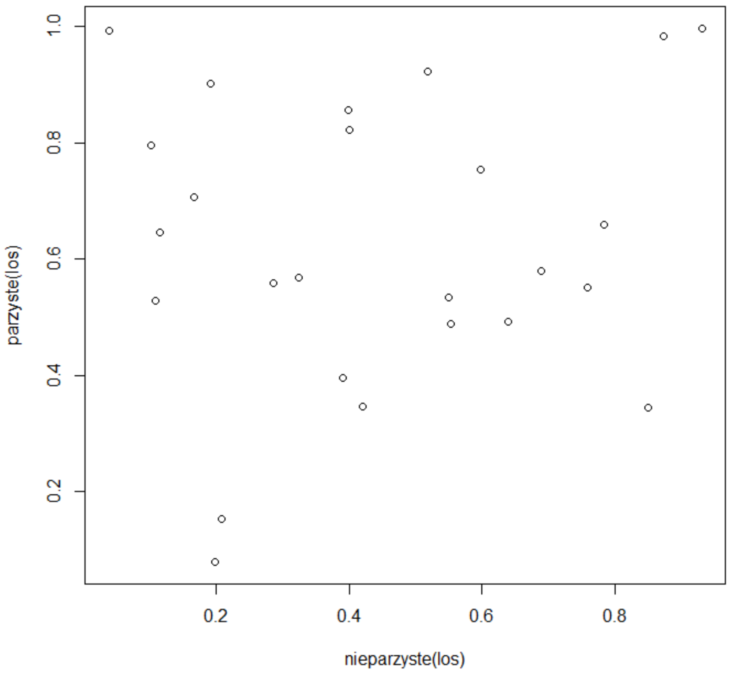
  for(i in 2:(k/2)){
    a<-c(a,x[2*i - 1])
  }

  return(a[])
}

generator1<-f(19,37,50)
generator2<-f(39373,2**31 -1,50)

plot(parzyste(generator1),nieparzyste(generator1))
plot(parzyste(generator2),nieparzyste(generator2))
```



Rysunek 6: Generator 2 jako punkty na kwadracie. Punkty wyglądają losowo czyli może to być dobry generator

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:

- ```
  #generowanie 50 par pseudolosowych liczb za pomocą wbudowanego generatora
  los <- runif(100)

  plot(nieparzyste(los), parzyste(los))
  ```
- Moim pomysłem na sprawdzenie czy jest to rozkład jednostajny na kwadracie jest sprawdzenie czy w każdej cwiartce jest tyle samo punktów, Bo wtedy będzie to oznaczać że ma to podobny rozkład jak jak rozkład jednostajny.

  ```
  #sprawdzanie czy w każdej ćwiartce jest tyle samo punktów
  ```
  ```



Rysunek 7: Wbudowany generator losowy jako punkty losowe na kwadracie

```
x <- runif(1000)



spr_rozk <- function(x,y){

  kx<-length(x)
  ky<-length(y)

  a<-0
  b<-0
  c<-0
  d<-0


  for(i in 1:kx){
    for( j in 1:ky){
      if( x[i]<(1/2) & y[j]<(1/2)){
        a<-a+1
      }

      if( x[i]>(1/2) & y[j]<(1/2)){
        b<-b+1
      }
```

```
      if( x[i]<(1/2) & y[j]>(1/2)){
        c<-c+1
      }

      if( x[i]>(1/2) & y[j]>(1/2)){
        d<-d+1
      }

    }



  }
  return(c(a,b,c,d))
}

spr_rozk(parzyste(x),nieparzyste(x))
```

Po kilku próbach wychodzi że w kazdej ćwiartce jest mniejwięcej tyle samo punktów czyli jest to tozkład jednostajny na kwadracie

Ta własność łączy się z niezależnością kolejnych liczb, bo dzieki niej kolejne liczby nie wpływają na siebie.

- Za pomocą funkcji set.seed(3) za każdym razem możemy otrzymywać te same wyniki.

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.

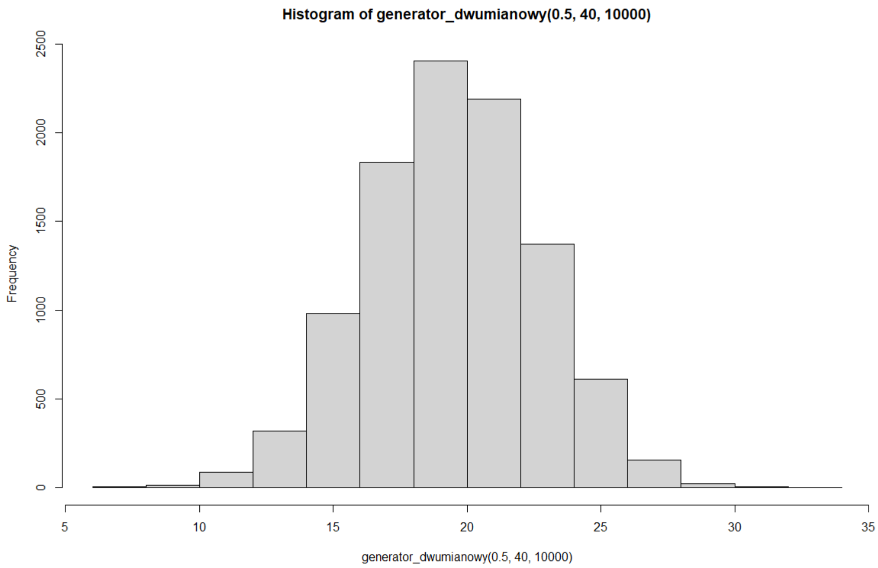
Rozwiązanie:

- Generator rozkładu dwupunktowego

```
# p- prawdobienstwo z przedziału [0,1]
#n ilosc generowanych liczb
generator_dwupunktowy<-function(p,n){

  x<-c()

  y<-runif(n)

  for(i in 1:n){
    if(y[i]<=p){
      x<-c(x,1)
    }else{
      x<-c(x,0)


    }
  }

  return(x[])

}
```

- Generator rozkłady dwumianowego

```
#p pojedyncze prawdobienstwo
#n ilosc sukcesóe
# ilosc  ilosc generowanych liczb
generator_dwumianowy<- function(p,n, ilosc){


  x<-generator_dwupunktowy(p,ilosc)
```

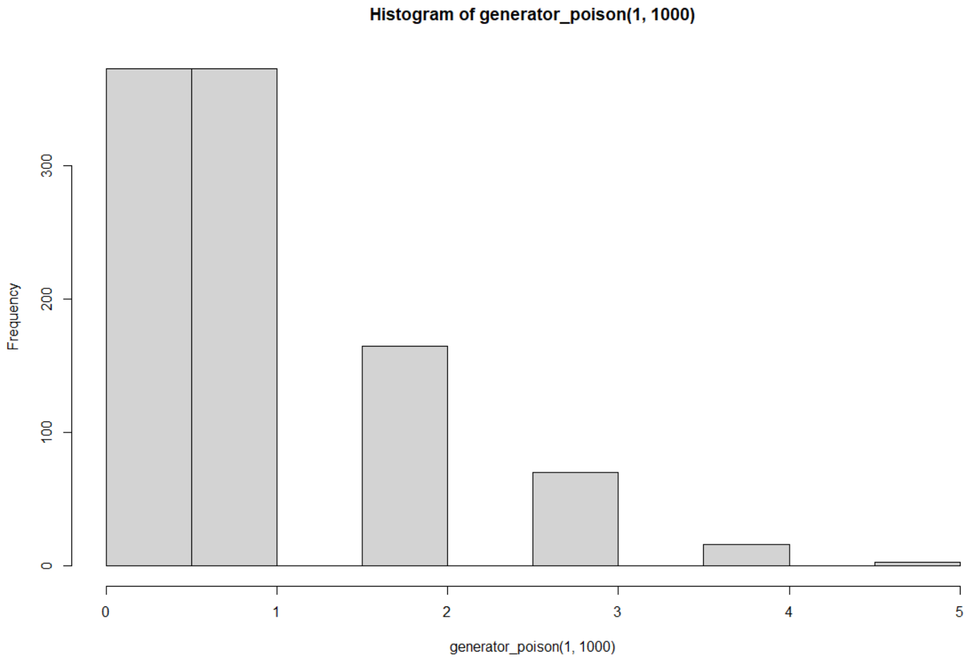


Rysunek 8: Histogram generatora dwumianowego gdy p=0.5 , n=40.

```
  for(i in 2:n){
    x<- x+ generator_dwupunktowy(p,ilosc)
  }


  return(x[])
}
generator_dwumianowy(0.2,9,1000)
```

- Generator rozkładu Poisona

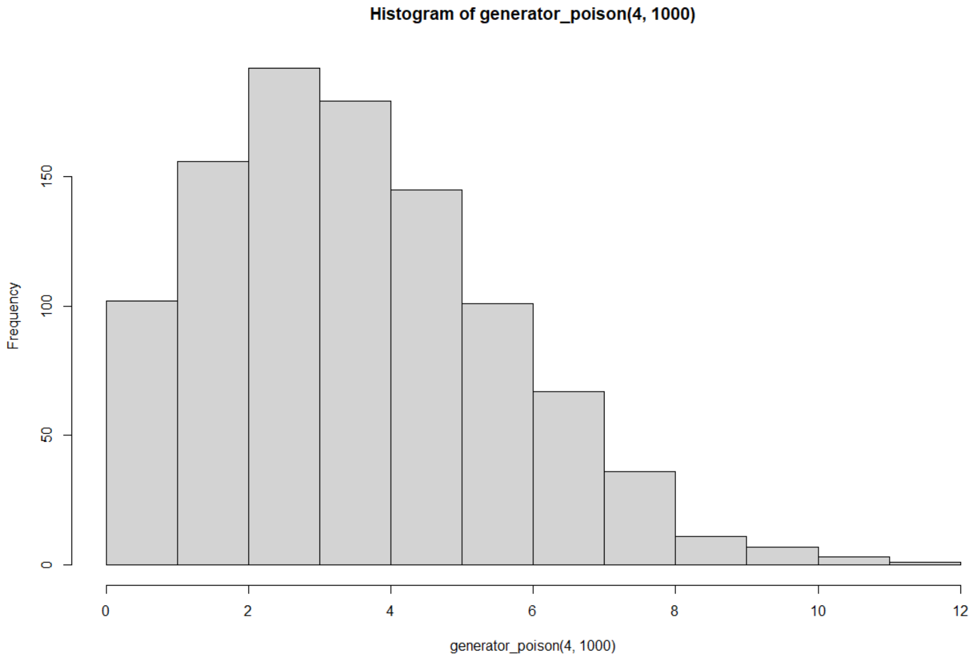
```
#lam - lambda parametr który zmieniamy w tym rozkładzie
# ilość generowanych liczb
generator_poison <- function(lam, n){

  e<- exp(-lam)

  p<-e

  S <- p

  x<-runif(n)

  wynik<-c()

  for(i in 1:n){
```



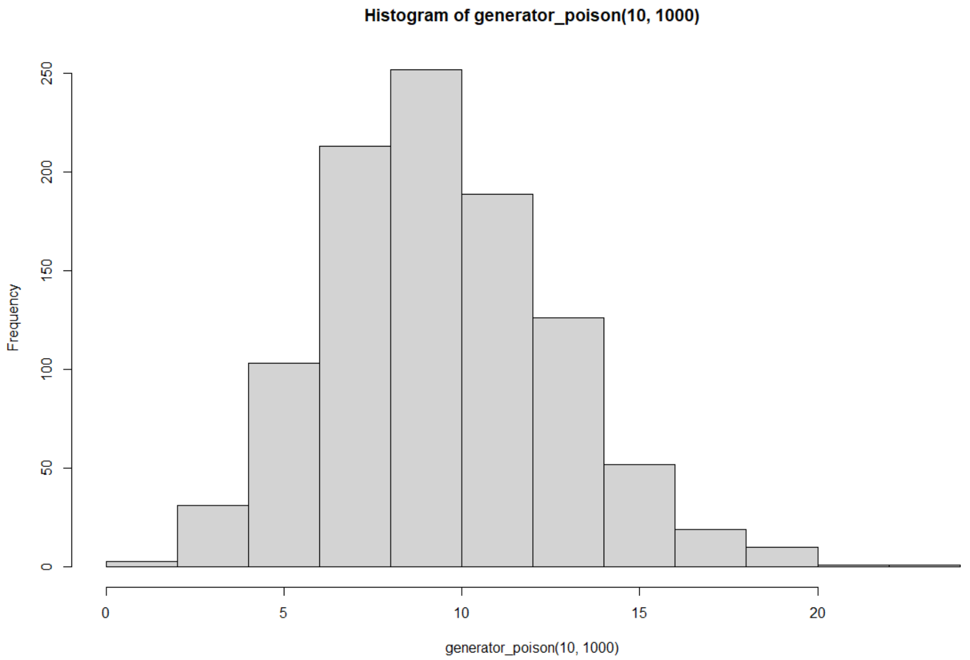
Rysunek 9: Histogram generatora Poisona gdy parametr λ=1.

```
    p<- e
    S <- p
    k<- 0


    while(S<x[i]){
      p<- p*(lam/(k+1))
      S<-S+p
      k<- k+1
    }

    wynik<-c(wynik,k)

  }

  return(wynik[])

}
```

Histogramy odpowiadają rozkładom Poisona więc mój generator jest dobry



Rysunek 10: Histogram generatora Poisona gdy parametr λ=4.



Rysunek 11: Histogram generatora Poisona gdy parametr λ=10.