Metody Monte Carlo

Raport 3

Autor: Anna Bonikowska

06.05.2024

Spis treści

1 Problem 1 – Generowanie rozkładów warunkowych i wielowymiarowych

- Niech $X \sim Bin(1,p)$ i $Y|X=0 \sim N(0,1)$, $Y|X=1 \sim N(\mu,\sigma)$, gdzie $p \in [0,1]$, $\mu \in \mathbb{R}$, $\sigma > 0$. Zaimplementuj generator zmiennej Y. Uzasadnij, że $Y \sim (1-p)N(0,1)+pN(\mu,\sigma)$. Narysuj histogram i wykresy funkcji gęstości tego rozkładu dla wybranych parametrów.

- Zaproponuj sposób generacji punktów z rozkładu jednostajnego na symplexie $S := \mathbb{R}^2 \cap \{x, y > 0, x+y < 1\}$ oraz z rozkładu o gęstości cxy^2 na S.

- Zaimplementuj generator rozkładu $N(\mu,\Sigma)$, gdzie $\mu = (1,-1,0)$ i $\Sigma = \begin{bmatrix} 2 & 2 & 0 \\ 2 & 5 & -3 \\ 0 & -3 & 9 \end{bmatrix}$.

  Sprawdź, czy Σ jest symetryczna i dodatnio określona. Dla obliczenia $\Sigma^{1/2}$ użyj zaimplementowanej w R metody Choleskiego lub rozkładu spektralnego macierzy Σ. Wygeneruj próbkę z tego rozkładu i zilustruj wszystkie rozkłady 1-wymiarowe, 2-wymiarowe i 3-wymiarowe. Ile ich jest?

Rozwiązanie:

- Generator rozkładu $Y \sim (1-p)N(0,1)+pN(\mu,\sigma)$

```
generator_Y<-function(n, p, mi, gam){
  u <- runif(n )

  x <- c()

  for( i in u){

    if(i<p){
      x<- c(x,rnorm(1, mean=mi,sd=(gam)**(1/2)))
    } else {
      x<- c(x,rnorm(1, mean=0,sd=1))
    }
  }

  return(x[])
}
```

$X \sim Bin(1,p)$

Zdarzenia $X=0$, $X=1$ są rozbiciem przestrzeni Ω

$$\mathbb{P}(Y \in A) = \mathbb{P}(Y \in A \mid X=0)\,\mathbb{P}(X=0) + \mathbb{P}(Y \in A \mid X=1)\,\mathbb{P}(X=1)$$

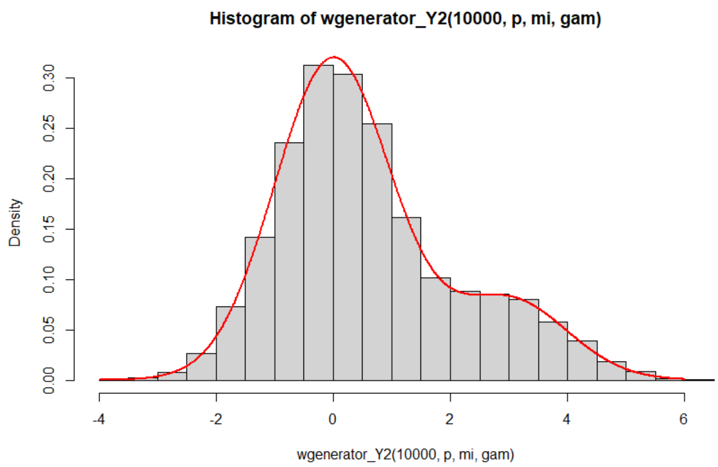
$$= \mathbb{P}(Z \in A)\cdot(1-p) + \mathbb{P}(H \in A)\cdot p$$

$Z \sim N(0,1)$ $\qquad H \sim N(\mu, \sigma)$

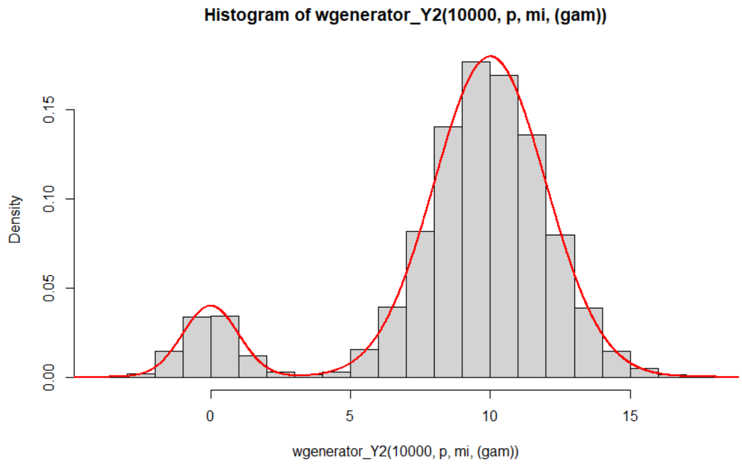
bo $Y|X=0 \sim N(0,1)$ $\qquad Y|X=1 \sim N(\mu, \sigma)$

Więc $Y \sim (1-p)N(0,1) + pN(\mu, \sigma)$

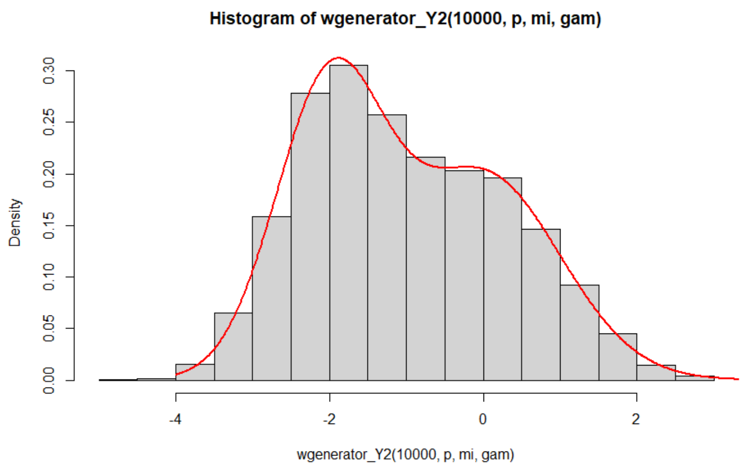
Uzasadnienie, że $Y \sim (1-p)N(0,1) + pN(\mu,\sigma)$



Rysunek 1: Generator rozkładu Y dla parametrów p=0.2 , $\mu = 3, \sigma = 1$



Rysunek 2: Generator rozkładu Y dla parametrów p=0.9 , $\mu = 10, \sigma = 4$



Rysunek 3: Generator rozkładu Y dla parametrów p=0.5 , $\mu = -2, \sigma = 0.5$

- generowanie rozkładu jednostajnego na symplexie:

```
# zwraca macierz gdzie x[i,1] odpowiada współrzednej x
#a x[i,2] odpowiada współrzednej y punktu i
gener1 <- function(n){



  licznik<- 0
  x <- c()
  y <- c()

  while (licznik<n){
```

```
    a <-runif(1)
    b <-runif(1)

    if ((a+b)<1){
      #print(a)
      #print(b)
      #print(a+b)
      #print(3)

      licznik <- licznik + 1

      x <- c(x,a)
      y <- c(y,b)

    }

  }

  u_matrix <- matrix(c(x,y), ncol=2)

  return(u_matrix)
}
}
```

generowanie rozkładu cxy^2 na S:

```
# zwraca macierz gdzie x[i,1] odpowiada współrzednej x
#a x[i,2] odpowiada współrzednej y punktu i
gener2 <- function(n){



  licznik<- 0
  x <- c()
  y <- c()

  while (licznik<n){

    u1 <-runif(1)
    u2 <-runif(1)

    x2 <- u1**(1/2)
    y2 <- u2**(1/3)

    if ((x2+y2)<1){
      #print(a)
      #print(b)
```

```
      #print(a+b)
      #print(3)

      licznik <- licznik + 1

      x <- c(x,x2)
      y <- c(y,y2)

    }

  }


  u_matrix <- matrix(c(x,y), ncol=2)

  return(u_matrix)

}
```

- – Σ jest symetryczna. Sprawdza sie to wprost
  - sprawdzanie czy Σ jest dodatnio okreslona. Jesli Σ ma dodatnie wartosci własne to jest dodatnio okreslona

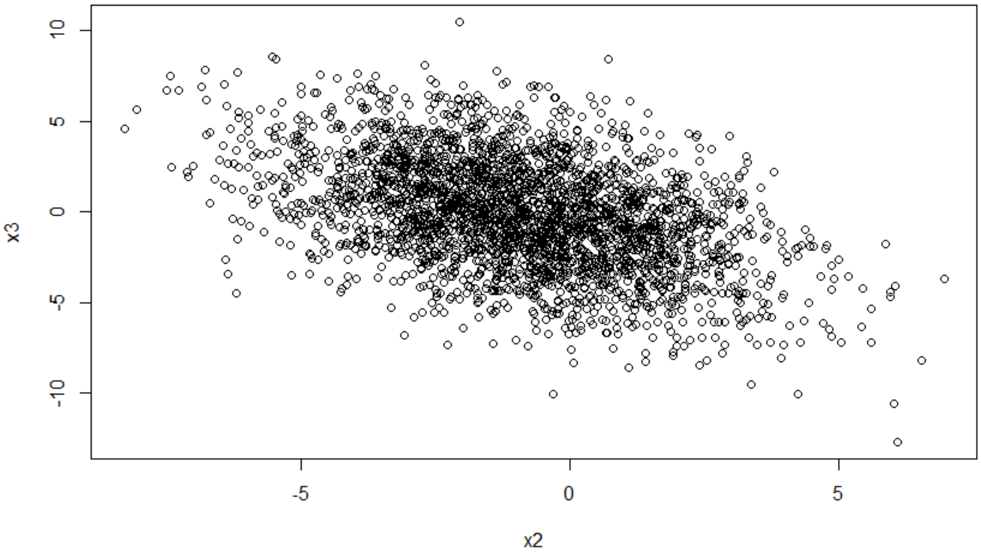
    ```
      A <- matrix(c(2,2,0,2,5,-3,0,-3,9), ncol=3)

    eigen(A)


    eigenvalues <- eigen(A)$values

    # Sprawdzenie czy wszystkie wartości własne są dodatnie
    all_positive_eigenvalues <- all(eigenvalues > 0)
    all_positive_eigenvalues
    ```

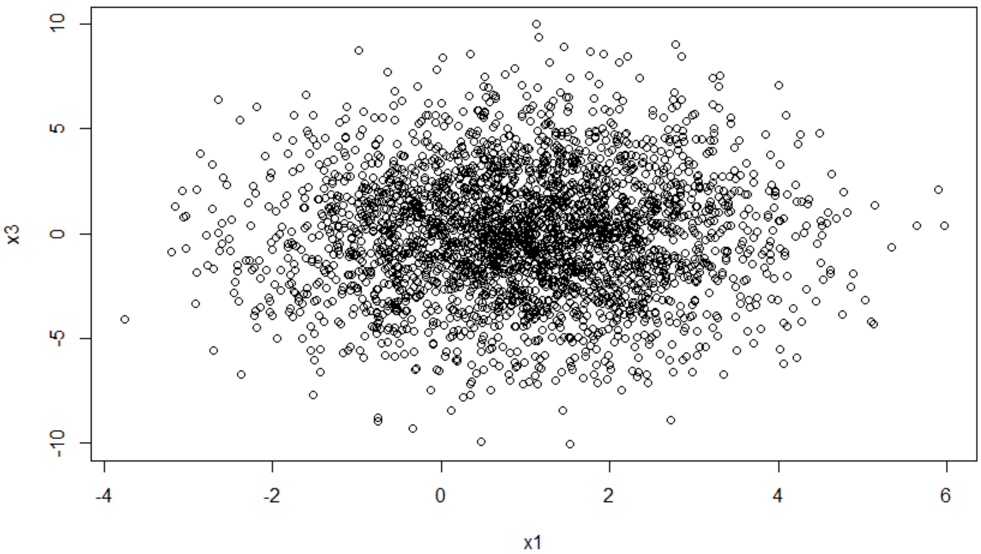
  - generator rozkładu $N(\mu, \Sigma)$

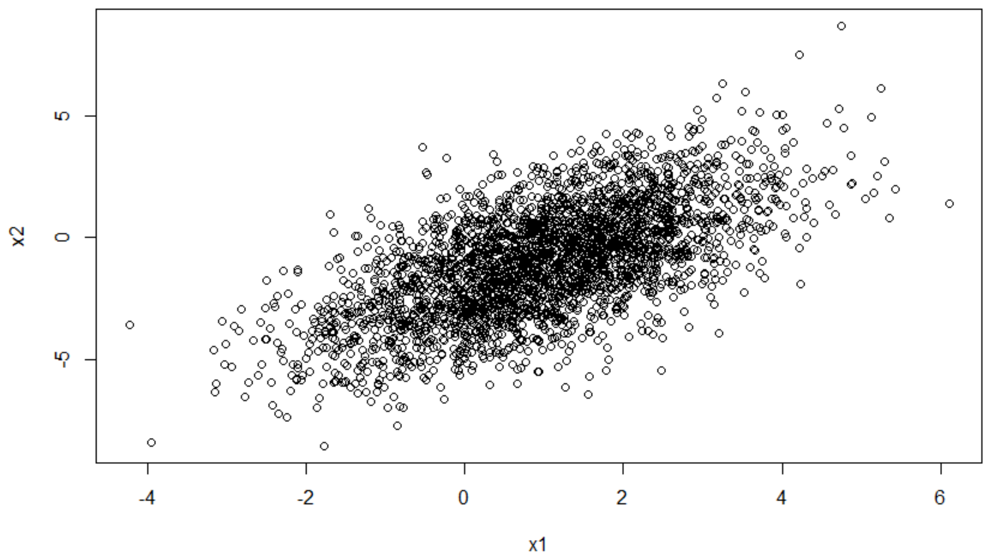
    ```
          gener_wielow_norm <- function(mean, cov_matrix, num_samples) {

      B <- t(chol(cov_matrix))


      z <- matrix(rnorm(length(mean) * num_samples), nrow = length(mean))


      samples <- matrix(mean, ncol=num_samples, nrow=length(mean)) + B %*% z

      return(samples)
    }
    ```

```
#generuje rozkład N(mi, Sigma)
x <- gener_wielow_norm(mean = c(1,-1,0), cov_matrix = matrix(c(2,2,0,2,5,-3,0,-
```

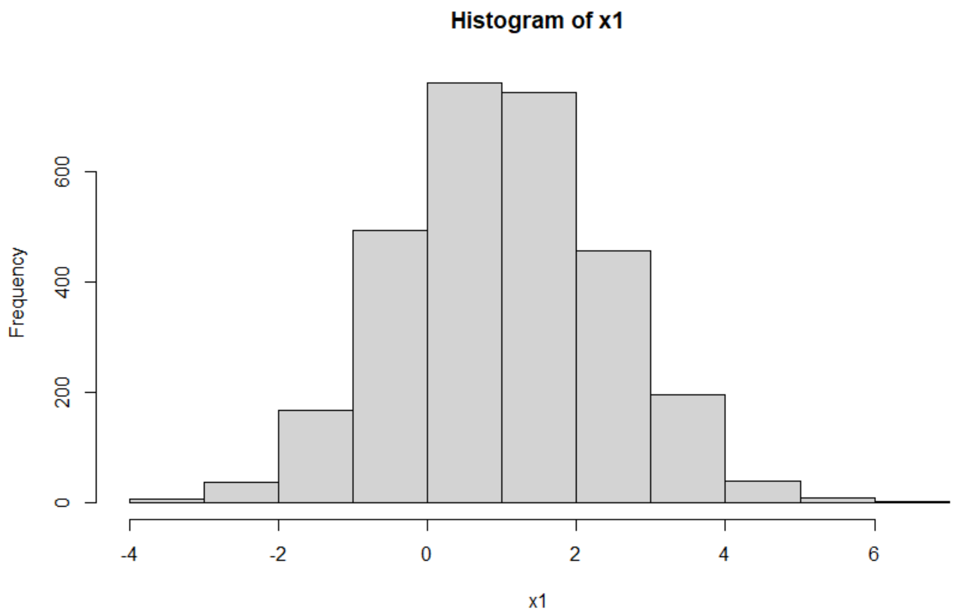
– rozkłady dwuwymiarowe:

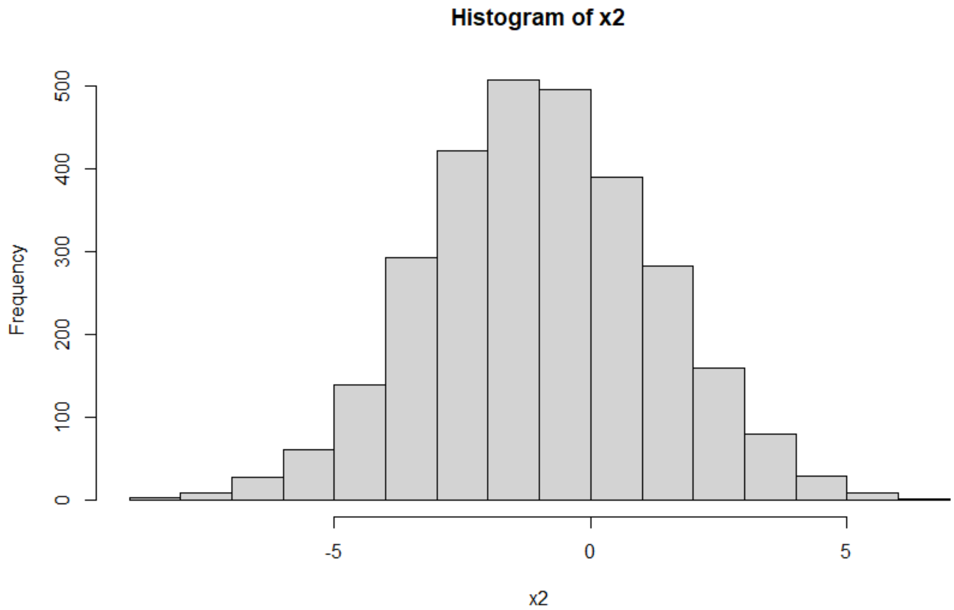


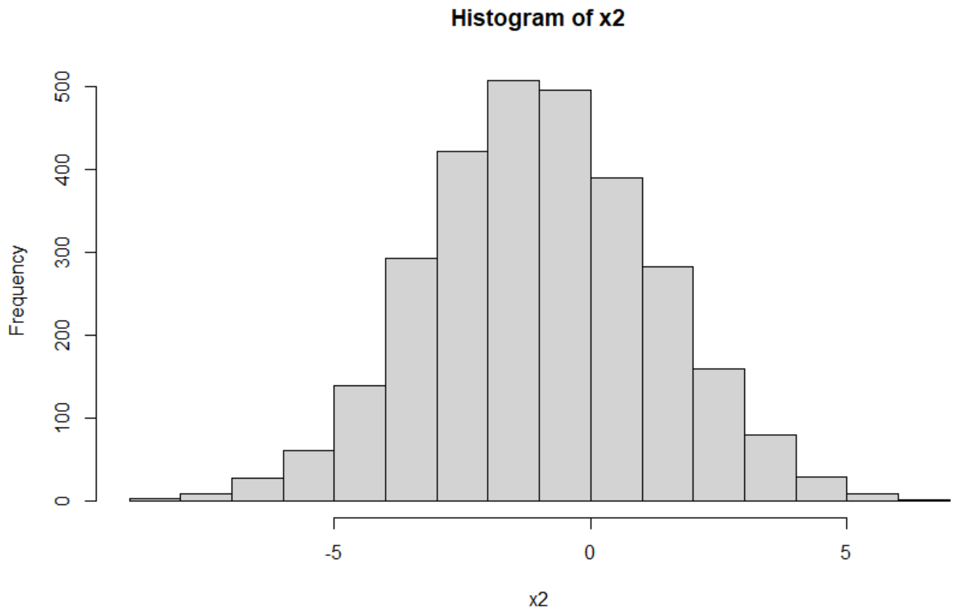




– rozkłady jednowymiarowe




Histogram of x2
Frequency
0
100
200
300
400
500
-5
0
5
x2


Histogram of x2
Frequency
0
100
200
300
400
500
-5
0
5
x2

2 Problem 2 – Przybliżone obliczanie całek metodą Monte Carlo

- Aproksymuj metodą Monte Carlo całkę $\int_0^1 e^{-x^2} dx$. Zbadaj empirycznie szybkość zbieżności i wariancję otrzymanego estymatora w zależności od rozmiaru próby. Znajdź przedział ufności dla tej całki na podstawie $n = 1000$ obserwacji.
- Użyj metody Monte Carlo do aproksymacji całki $\int_0^\infty x^5 e^{-x} \cos x dx$.
- Metodą Monte Carlo oszacuj powierzchnię żuka Mandelbrota. Jak ocenić błąd?

Rozwiązanie:

- aproksymowanie całki $\int_0^1 e^{-x^2} dx$:

```
  fun <- function(n){
I <-0


for (i in 1:n){
  u <- runif(1)

  I <- I + exp(-u**2)


}

I <- I/n

return(I)

}
```

wartość obliczona przez internetowy kalkulator całek:
0.7468241328124270253994674361318530053544996868126063290276544989...
...

dla próby n=10:
mean=0.7470848 var= 0.004037572

dla próby n=100:
mean=0.746775 var=0.0004151597

dla próby n=1000:
mean=0.746892 var=3.999278e-05

dla próby n=10000:
mean=0.7468265 var=4.038532e-06

- aproksymacja całki $\int_0^\infty x^5 e^{-x} \cos x dx$

```
    fun <- function(n){
  I <-0


  for (i in 1:n){
    x <- rexp(1, rate=1)

    I <- I + x**5*cos(x)


  }

  I <- I/n

  return(I)

}
```

$\int_0^\infty x^5 e^{-x} \cos x dx \approx 0.1629284$
$gdyn = 100000000$

według Wolfram Alpha: $\int_0^\infty x^5 e^{-x} \cos x dx = 0$

- aproksymacja powierzchni rzuka mandelbrota

funkcja sprawdzajaca czy punkt nalezy do zbioru rzuka mandelbrota:

```
mandelbrot <- function(a,b){

  c <-0
  d <- 0
  czy_nalezy <- 0
  i <- 0
  wynik <- 0

  while( i <= 1000 &  wynik < 2  ){

    x <- c^2 - d^2 +a
    y <- 2*d*c + b

    c<-x
    d <-y

    i <- i +1

    wynik <- (c^2 + d^2)^(1/2)
```

```
  }

  wynik <- (c^2 + d^2)^(1/2)

  czy_nalezy <- 0

  if( wynik < 2){
    czy_nalezy <- 1
  }

  return( czy_nalezy)
 }
```

właściwa funkcja licząca pole powierzchni Mandelbrota

```
 fun <- function(n){
  I <-0

  for (i in 1:n){
    x <- runif(1,min=-2, max=1)
    y <- runif(1,min=-1, max=1)

    I <- I + 6*mandelbrot(x,y)

  }

  I <- I/n

  return(I)

}

}
```

Nie da się wprost oszacować błedu bo nie jest znana dokładna wartość powierzchni zbioru Mandelborta, ale jest szacowana na 1.3744 do 1.68288. Wartosci mojej funkcji mieszczą sie w tym przedziale czyli moja funkcja działa z grubsza dobrze.

3 Problem 3 – Twierdzenia graniczne dla ciągów *i.i.d.*

- Udowodnij słabe prawo wielkich liczb, gdy $\mathbb{E}X^2 < \infty$.
- Sformułuj mocne prawo wielkich liczb. Zilustruj je symulacjami, gdy $X \sim Bin(1, p)$.
- Sformułuj Centralne Twierdzenie Graniczne dla średnich i sum *i.i.d.* zmiennych losowych o skończonym drugim momencie. Korzystając z CTG i tablic dystrybuanty rozkładu normalnego oblicz w przybliżeniu $\mathbb{P}(U_1 + \ldots + U_n > 0.5n + 0.1\sqrt{n})$ dla (*i.i.d.*) zmiennych losowych $U_1, \ldots, U_n$ z rozkładu jednostajnego. Zweryfikuj i zilustruj ten wynik przez symulacje.

Rozwiązanie:

- mocne prawo wielkich liczb

  $\mathrm{X}_1, X_2, ..., X_n i.i.d.$

$\mathrm{E}(|\mathrm{X}_1|) < \infty$

$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} X_i = \mathbb{E}(X_1)$ z prawdobieństwem 1

```
    sum <- 0
p<-0.9



n <- 1000000


for (i in 1:n){

  x <- rbinom(1,size=1, prob=p)

  sum <- sum + x

}

sum<- sum/n

sum
```

wartosć oczekiwana to p=0.9 sum= 0.8999617 Czyli jest to zbiezne czyli zachodzi prawo wielkich liczb

- Centralne Twierdzenie Graniczne dla średnich i sum

Jeżeli

$$(X_i)$$

jest ciagiem niezależnych zmiennych losowych o tym samym rozkładzie, $\mathbb{E}(|X_1|) = \mu < \infty$,$Varx_i =^2 <$ ∞ to
$\lim_{n\to\infty} \left(\frac{X_1+X_2+\ldots+X_n-n\mu}{\sigma\sqrt{n}} \leq x \right) = N \sim N(0,1)$
Według moich obliczeń:
$\mathbb{P}(U_1 + \ldots + U_n > 0.5n + 0.1\sqrt{n}) = 0{,}3669$
symulacyjne obliczenie $\mathbb{P}(U_1 + \ldots + U_n > 0.5n + 0.1\sqrt{n})$:

```
sprawdzanie <- function(n){

 h <- 10000


 x <- 0

 for (i in 1:h){

   u <- runif(n)

   if ( sum(u) > 0.5*n +0.1*n**(1/2)){

    x<- x+1
   }

 }

 return(x/h)

}
```

Dla n=10000000 wyszło że ta wartosc wynosi 0.3688. To jest całkiem dobre przyblizenie