Generatory liczb pseudolosowych

1 Wstęp

Projekt ten ma na celu sprawdzenie jakości generatorów liczb pseudolosowych za pomocą testów statystycznych. Wiadomym jest, że nie da się skonstruować prawidziwie losowego generatora używając algorytmów komputerowych, ponieważ znając algorytm, w teorii powinniśmy być w stanie go odwrócić. Jednakże wraz z rozwojem technologii komputerowej możliwe są coraz bardziej złożone obliczenia, które pozawalają na zblIżenie się do losowości. W tej pracy zostaną przebadane najprostsze jak i bardziej złożone generatory, które są używane współcześnie, aby przyjrzeć się rozwojowi generatorów pseudolosowych.

2 Generatory

Używane w tej pracy generatory będą produkować 3 typy wyników: liczby całkowite, liczby z przedziału $(0, 1)$ oraz ciągi bitów. Są one jednak uniwersalne, ponieważ jesteśmy w stanie przejść do każdego rodzaju. Każda liczba może zostać przedstawiona jako ciąg bitów, a ciąg bitów może zostać przedstawiony jako liczba w systemie dziesiętnym. Przemnożenie (i użycie funkcji podłogi) oraz podzielenie przez stałą da odpowiednio liczbę całkowitą i liczbę zmiennoprzecinkową.

2.1 LCG(M, a, c)

Linear congruential generator zmienia stan zgodnie z rekurencją:

$$x_n = (ax_{n-1} + c) \mod M$$

Gdzie stanem początkowym (ziarnem) jest x_0. W pracy zostanie przetestowany LCG$(13, 1, 5)$ z ziarnem 1 oraz LCG$(2^{10}, 3, 7)$ z ziarnem 0. W przypadku LCG należy dobrze wybrać parametry, ponieważ dla niektórych kombinacji generator może "wpaść" na taką liczbę, która zatrzyma dalsze generowanie , np. $(6 * 613 + 7) \mod 2^{10} == 613$.

2.2 GLCG$(M, \{a_i\}_{i=1}^{k})$

GLCG jest uogólniem LCG i jego stan zmienia się zgodnie z rekurencją:

$$x_n = (a_1 x_{n-1} + \cdots + a_k x_{n-k}) \mod M$$

Gdzie stanem początkowym (ziarnem) jest $x_0, x_1, \ldots, x_{k-1}$. W pracy zostanie przetestowany GLCG$(2^{10}, \{3, 7, 68\})$ z ziarnem $1, 2, 3$.

2.3 Excel (LCG)

Generator liczb losowych z excela jest modyfikacją LCG, który zwraca liczby z przedziału $(0, 1)$, jego stan zmienia się zgodnie z rekurencją:

$$u_n = (0.9821u_{n-1} + 0.211327) \mod 1$$

Gdzie jego stanem początkowym (ziarnem) jest u_0. W pracy zostanie przetestowany z ziarnem 0 oraz 1812433253.

2.4 Niewymierne stałe

Rozwinięcie części dziesiętnej w systemie dwójkowym takich znanych stałych jak $\pi, e, \sqrt{2}$ może posłużyć jako generator losowych bitów, ze względu na niewymierność tych stałych, ktora powoduje brak okresowości.

2.5 RC4(32)

RC4 jest szyfrem strumieniowym, który można zmodyfikować, aby posłużył jako generator. Polega na zmienianiu permutacji S (stanu wewnętrznego) używając klucza K od długości L oraz zwracaniu elementów tej permutacji.

while $r \in \mathcal{N}$ **do**
  $i := i + 1$
  $j := j + S[i] + K[i \mod L]$
  swap$(S[i], S[j])$
  $Y_r \leftarrow S[S[i] + S[j]]$
end while

W tej pracy użyty generator RC4 zostanie użyty na dwa sposoby:

a) Używając jednego klucza $K = (0, 1, \ldots, 31)$,

b) Używając wielu kluczy $K = 0, 1, 2, \ldots$.

2.6 PCG64

Permuted Congruential Generator jest nowoczesnym generatorem używanym jako domyślny w bibliotece *NumPy* w języku programownia *Python*. Jest ulepszeniem LCG ze znacznie lepszymi statystycznymi własnościami i wysoką wydajnością. W tej pracy zostanie zainicjalizowany ziarnem 0.

3 Testy statystyczne

Testy statystyczne posłużą do sprawdzenia hipotezy o zgodności rozkładu liczb wygenerowanych z rozkładem jednostajnym. Problem statystyczny może zostać zformułowany w postaci hipotez:

$$H_0 : F_X = F_U, \quad H_1 : F_X \neq F_U.$$

Gdzie F_X to dystrybuanta zmiennej wygenerowanej przez generator oraz F_U to dystrybuanta zmiennej losowej o rozkładzie jednostajnym, na odpowiednim odcinku.

Wykonując test otrzymujemy statystykę testową, za pomocą której możemy otrzymać p-wartość, która posłuży do wnioskowania w problemie statystycznym. Przyjmiemy, że generator nie losuje liczb z rozkładu jednostajnego, gdy p-wartość będzie mniejsza od poziomu istotności $\alpha = 0.05$. Będzie to oznaczało, że generator nie jest losowy, a zatem jest "złym" generatorem. Dla prostoty przyjmiemy również, że lepszy generator, będzie miał większą p-wartość.

3.1 Test χ^2

Niech n oznacza liczbę obserwacji, k liczbę katerogrii, Y_s oznacza liczbę obserwacji kategorii s oraz p_s będzie prawdopodobieństwem pojawienia się w kategorii s. Dla dużych n (liczby obserwacji) oczekujemy $Y_s = np_s$. Przy takim założeniu statystyka

$$\chi^2 = \sum_{i=1}^{k} \frac{(Y_i - np_i)^2}{np_i}$$

ma rozkład χ^2_{k-1} (przy założeniu H_0) za pomocą czego możemy otrzymać p-wartość.

3.2 Test Kołogomorowa-Smirnova

Załóżmy, że zmienna X_i, $i = 1, \ldots, n$ ma rozkład ciągły o dystrybuancie $F_X(t)$. Zdefiniujemy dystrybuantę empiryczną jako:

$$\hat{F_n}(t) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}(X_i \leq t)$$

Wtedy statystyka:

$$\hat{D_n} = \sqrt{n} \sup_{x \in \mathcal{R}} |\hat{F_n}(x) - F(x)|$$

dąży do rozkładu Kołogomorowa-Smirnova (przy założeniu H_0), co pozwala na obliczenie p-wartości.

3.3 Test frequency monobit

Załóżmy, że mamy ciąg bitów $(b_i)_{i=1}^n$, przekształcamy ten ciąg do ciągu $(x_i)_{i=1}^n$ o wartościach $-1, 1$. Statystyka:

$$s_n(obs) = \frac{1}{\sqrt{2}} \sum_{i=1}^{n} x_i$$

dąży do rozkładu standardowego normalnego (przy założeniu H_0), co pozwala na obliczenie p-wartości.

3.4 Test frequency block

Celem tego testu jest sprawdzenie proporcji wystąpień 1 w blokach M-bitowych. Załóżmy, że mamy ciąg bitów $(b_i)_{i=1}^n$, $N = \lfloor \frac{n}{M} \rfloor$ jest liczbą bloków (resztkę bitów odrzucamy), π_i jest proporcją 1 w M-bitowych blokach. Statystyka:

$$\chi^2(obs) = 4M \sum_{i=1}^{N} (\pi_i - \frac{1}{2})^2$$

ma rozkład χ^2_N (przy założeniu H_0) za pomocą czego możemy otrzymać p-wartość. Długość bloków M należy tak dobrać aby: $M \geq 20$, $N < 100$, $M > .01n$.

3.5 Second-level testing

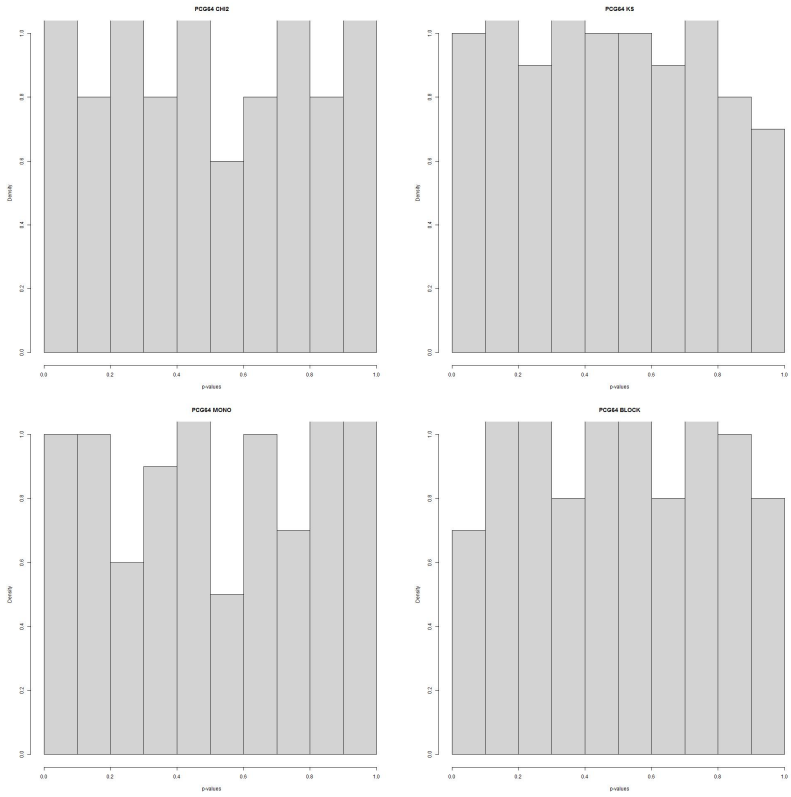
Second-level testing to metoda, która służy do upewnienia wyników. Jak okaże się w sekcji z wynikami (sekcja 4) wykonanie jednego testu jest nie wystarczające do oceny dobroci testu. Z tego powodu, wszystkie testy zostaną powtórzone r razy, co oznacza obliczenie r p-wartości. Z teorii prawdopodobieństwa wynika, że przy hipotezie H_0 p-wartości pochodzą z rozkładu jednostajnego $U(0, 1)$, co również możemy przetestować, za pomocą testu χ^2 (sekcja 3.1).

4 Wyniki

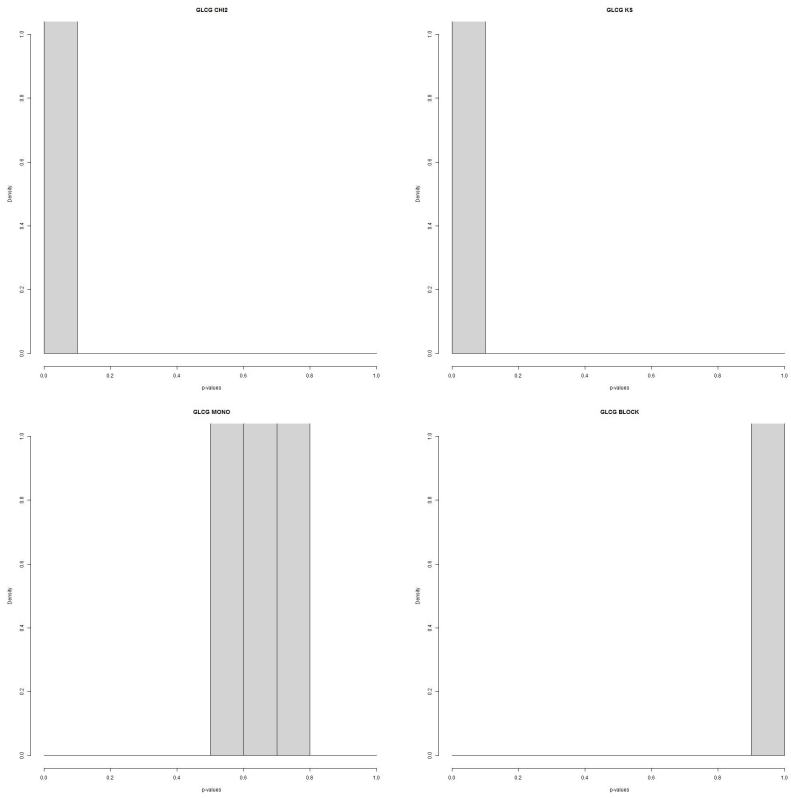
Wszystkie testy zostały wykonane na ciągach o długości $n = 2^{18}$ i powtórzone $r = 100$. P-wartości testów zostały zaprezentowane w tabeli 4.1 oraz wybranych histogramach (wszystkie znajdują się w dodatku 1) 4.1 i 4.2:

	CHI2	KS	MONO	BLOCK
LCG	$1.314539 \times 10^{-185}$	$7.115939 \times 10^{-190}$	$7.115939 \times 10^{-190}$	$7.115939 \times 10^{-190}$
LCG^10	$7.115939 \times 10^{-190}$	$1.314539 \times 10^{-185}$	$1.314539 \times 10^{-185}$	$1.314539 \times 10^{-185}$
GLCG	$7.115939 \times 10^{-190}$	$7.115939 \times 10^{-190}$	1.087128×10^{-91}	$1.314539 \times 10^{-185}$
excel	$7.115939 \times 10^{-190}$	$7.115939 \times 10^{-190}$	$7.115939 \times 10^{-190}$	$9.884149 \times 10^{-102}$
excel18	$7.115939 \times 10^{-190}$	$7.115939 \times 10^{-190}$	$7.115939 \times 10^{-190}$	$7.991558 \times 10^{-100}$
pi	-	-	0.3843359	0.7948
e	-	-	0.961893	0.4372742
sqrt2	-	-	0.7369418	0.3512874
RC4	0.8131215	$1.314539 \times 10^{-185}$	0.9327065	0.3052924
RC4_keys	0.5133364	$1.314539 \times 10^{-185}$	0.03177504	0.7169282
PCG64	0.4938104	0.9532085	0.1736566	0.9434777

Tablica 4.1: Second-level p-wartosci.



Rysunek 4.1: Histogramy p-wartości dla 4 testów generatora PCG64.



Rysunek 4.2: Histogramy p-wartości dla 4 testów generatora GLCG.

5 Wnioski

Z tabeli 4.1 jasno wynika, że proste i podstawowe generatory, oparte na pomyśle liniowej kongruencji okazują się słabymi generatorami. Przyglądając się histogramom ich p-wartości, można zauważyć, że mają pewne punkty, które wielokrotnie się powtarzają.

LCG$(13, 1, 5)$ ma za krótki okres dla ciągów o długości 2^18, co oznacza, że wylosowane liczby zaczynają się powtarzać i idealnie się rozkładać, co daje zawsze takie same statystyki testowe mają dają te same wyniki. Zwiększenie tego okresu w LCG$(2^10, 3, 7)$ daje pewną poprawę, lecz wciąż nie powoduje daje to rozkładu jednostajnego.

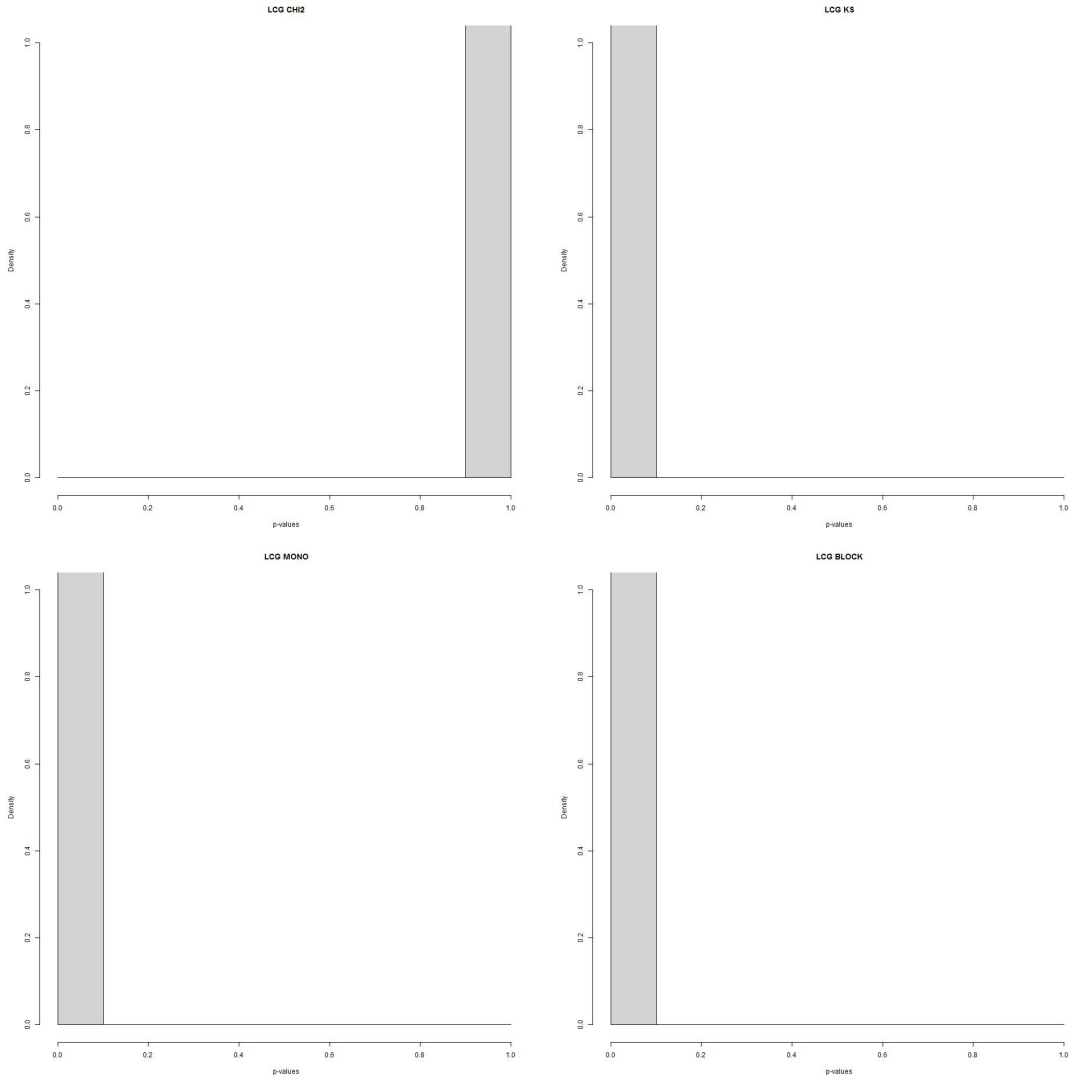
Każda stała okazała się dobrym generatorem, czego można było się spodziewać, ze zwględu na ich nieokresowość. Zależy to jednak od doboru długości ciągów i powtórzeń testów. Ze względu na to, że są to trudne generatory w użytkowaniu (trzeba być w stanie wyprodukować wiele cyfr takiej stałej) jesteśmy ograniczeni do skończonej ilości liczb do wygenerowania – w naszym przypadku około miliona bitów. Być może dla dłuższych ciągów wyniki były by odmienne.

Dobrym generatorem okazuje się RC4, który do niedawna był bardzo używanym generatorem, do czasu odkrycia jego słabości związanej z kluczami, które prezentuje testowanie RC4 z innymi kluczami. Widać tu znaczący spadek p-wartości, w przypadku jednego testu nawet poniżej poziomu istotniości. Z rezultatów testu K-S na tym generatorze, można zauważyć, że test ten nie nadaję się do generatorów liczb całkowitych, z małym zakresem (złamanie założenia o ciągłości).

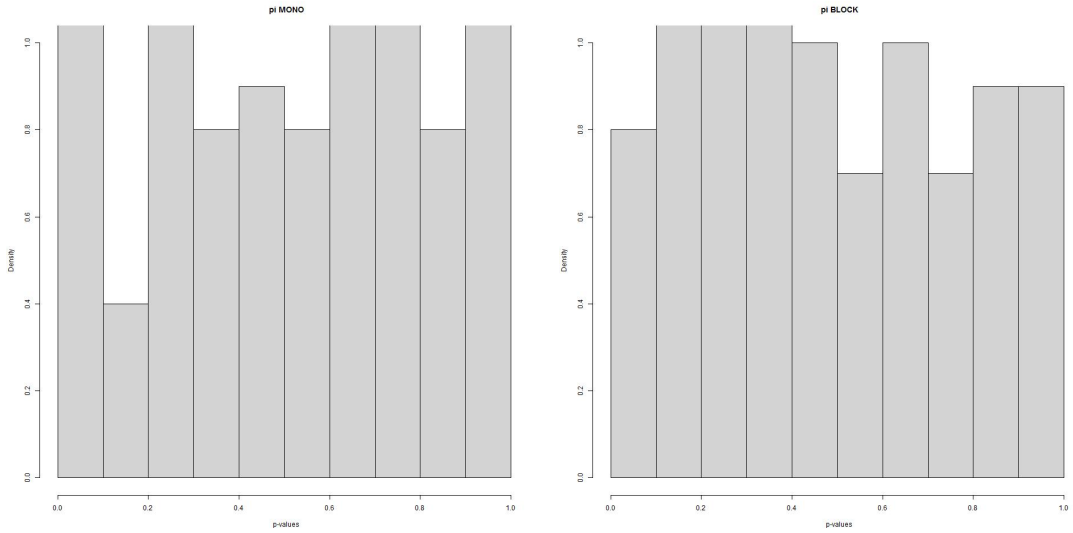
Zdecydowanie najlepszym generatorem jest PCG64, który przechodzi wszystkie testy statystyczne pozytywnie. Takich wyników można by się spodziewać, po współczesnym i wciąż używanym w środowisku matematycznym generatorze.

Appendices

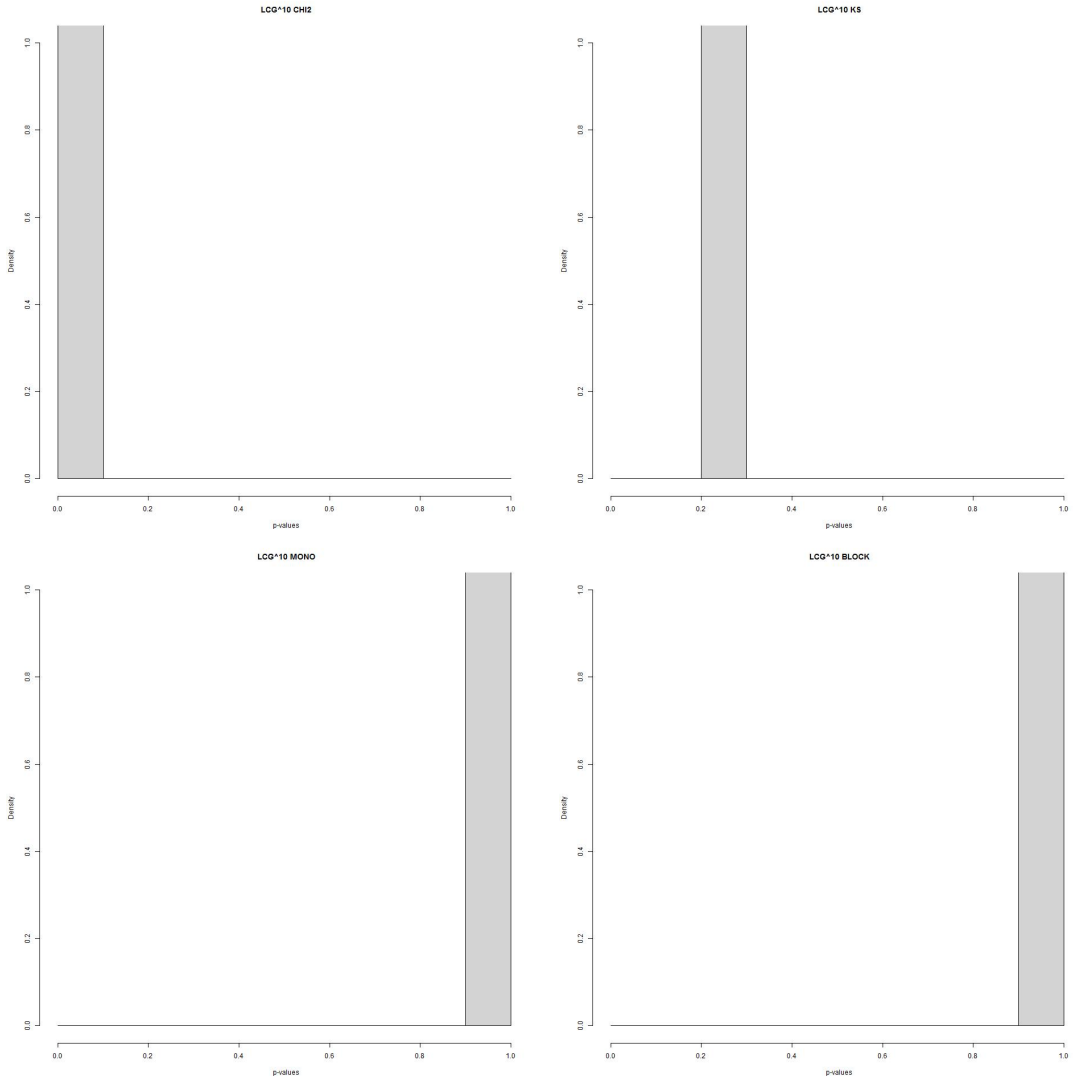
Dodatek 1



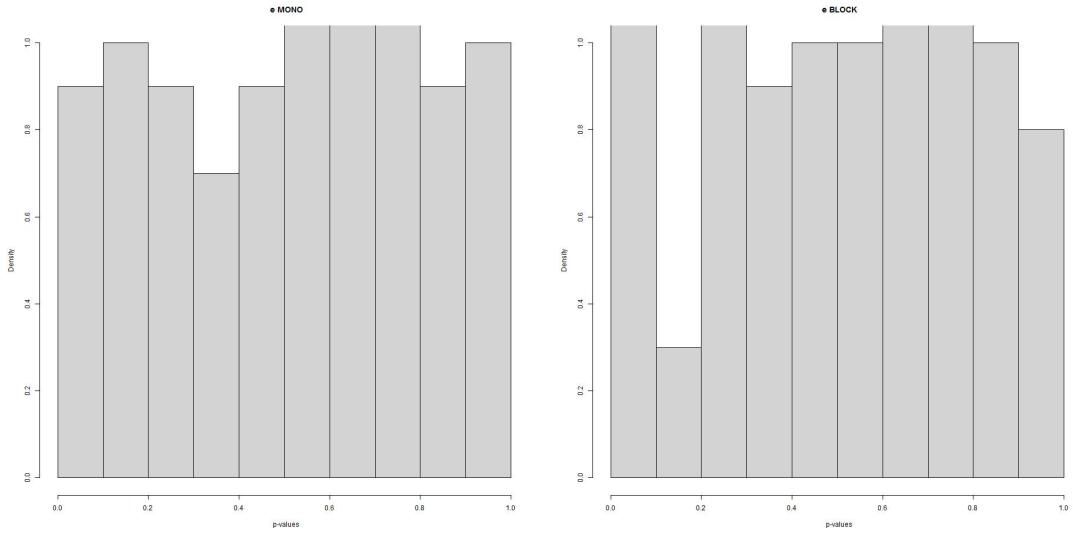
Rysunek 1.1: Histogramy p-wartości dla 4 testów generatora LCG$(13, 1, 5)$.



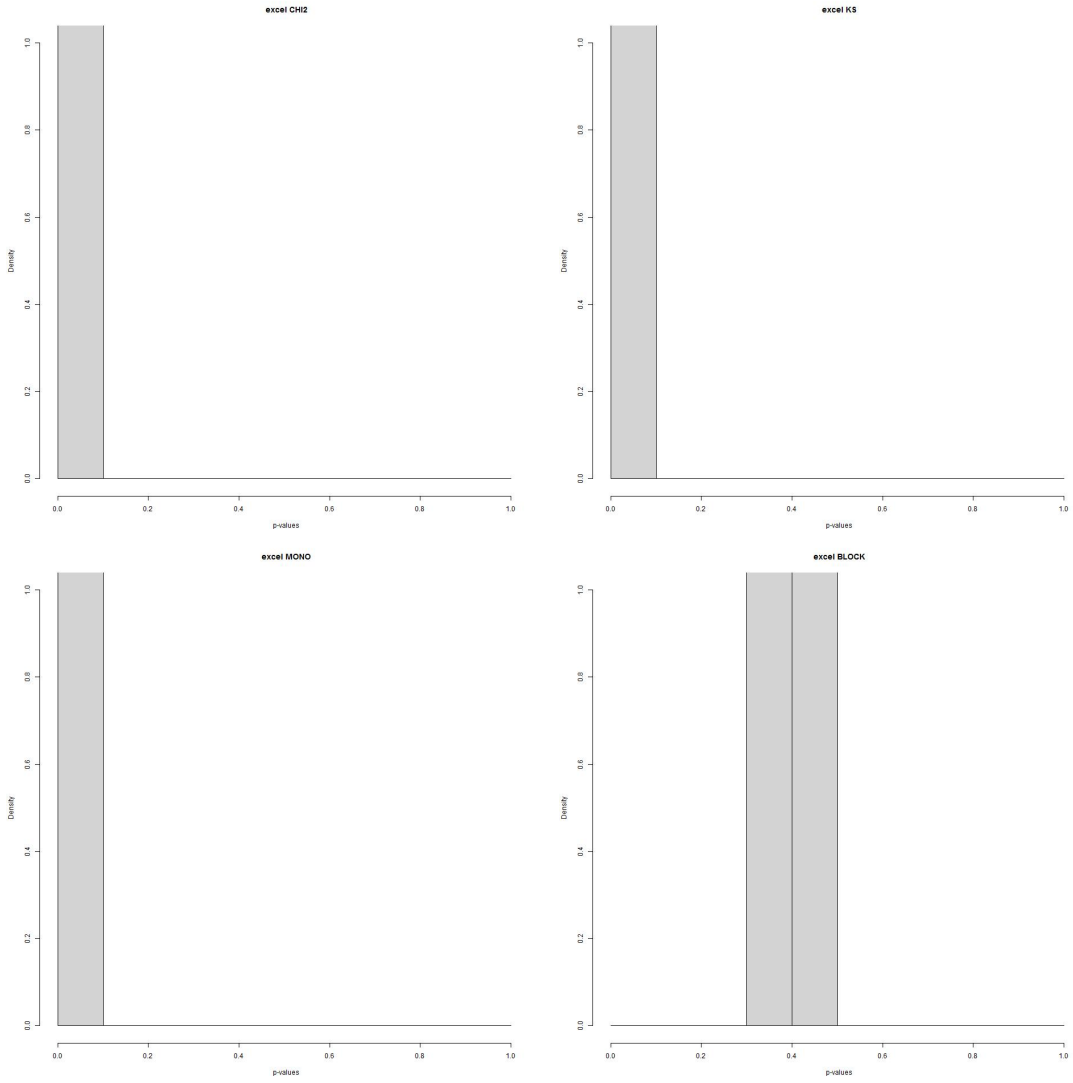
Rysunek 1.2: Histogramy p-wartości dla 4 testów generatora π.



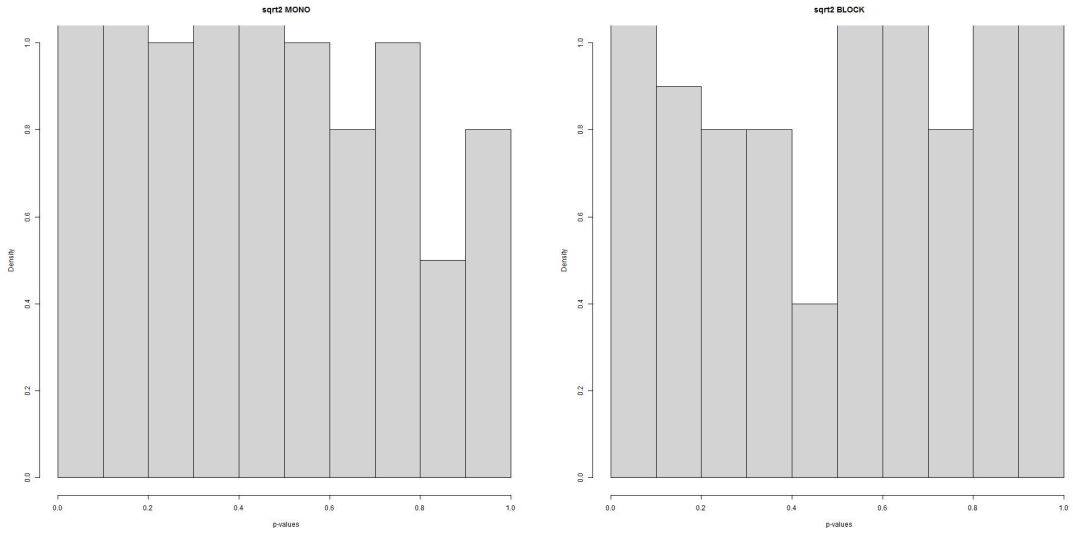
Rysunek 1.3: Histogramy p-wartości dla 4 testów generatora LCG$(2^{10}, 3, 7)$.



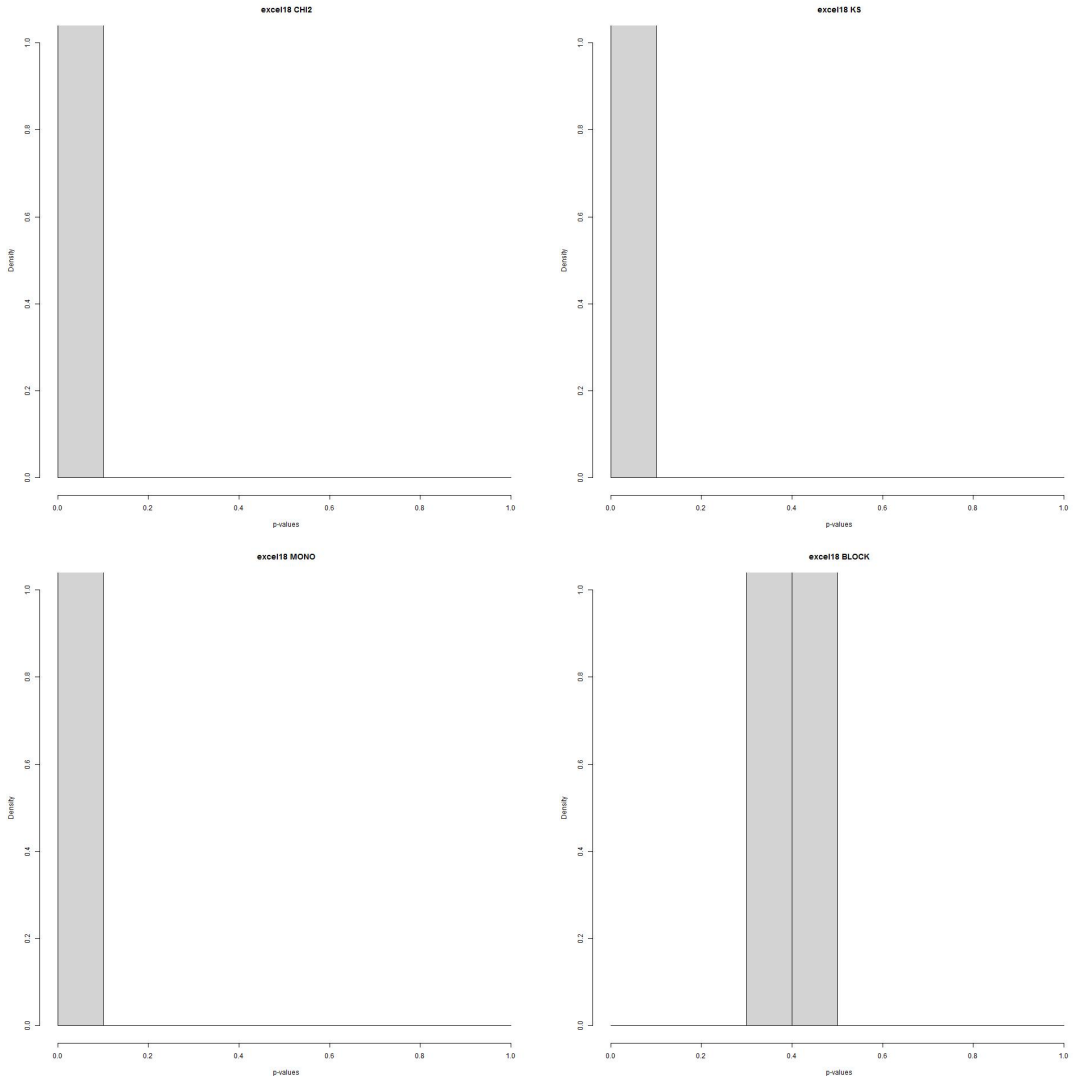
Rysunek 1.4: Histogramy p-wartości dla 4 testów generatora *e*.



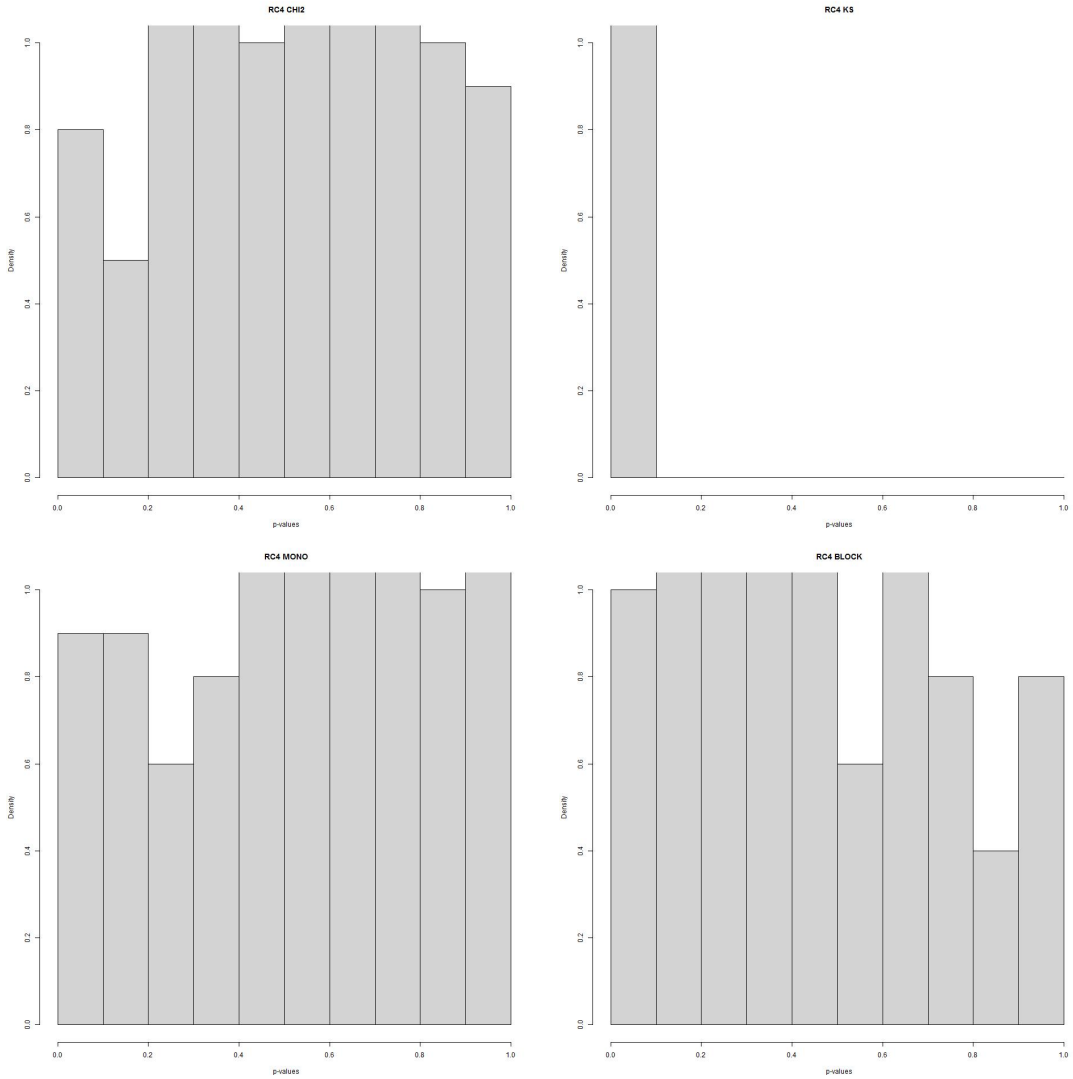
Rysunek 1.5: Histogramy p-wartości dla 4 testów generatora excel.



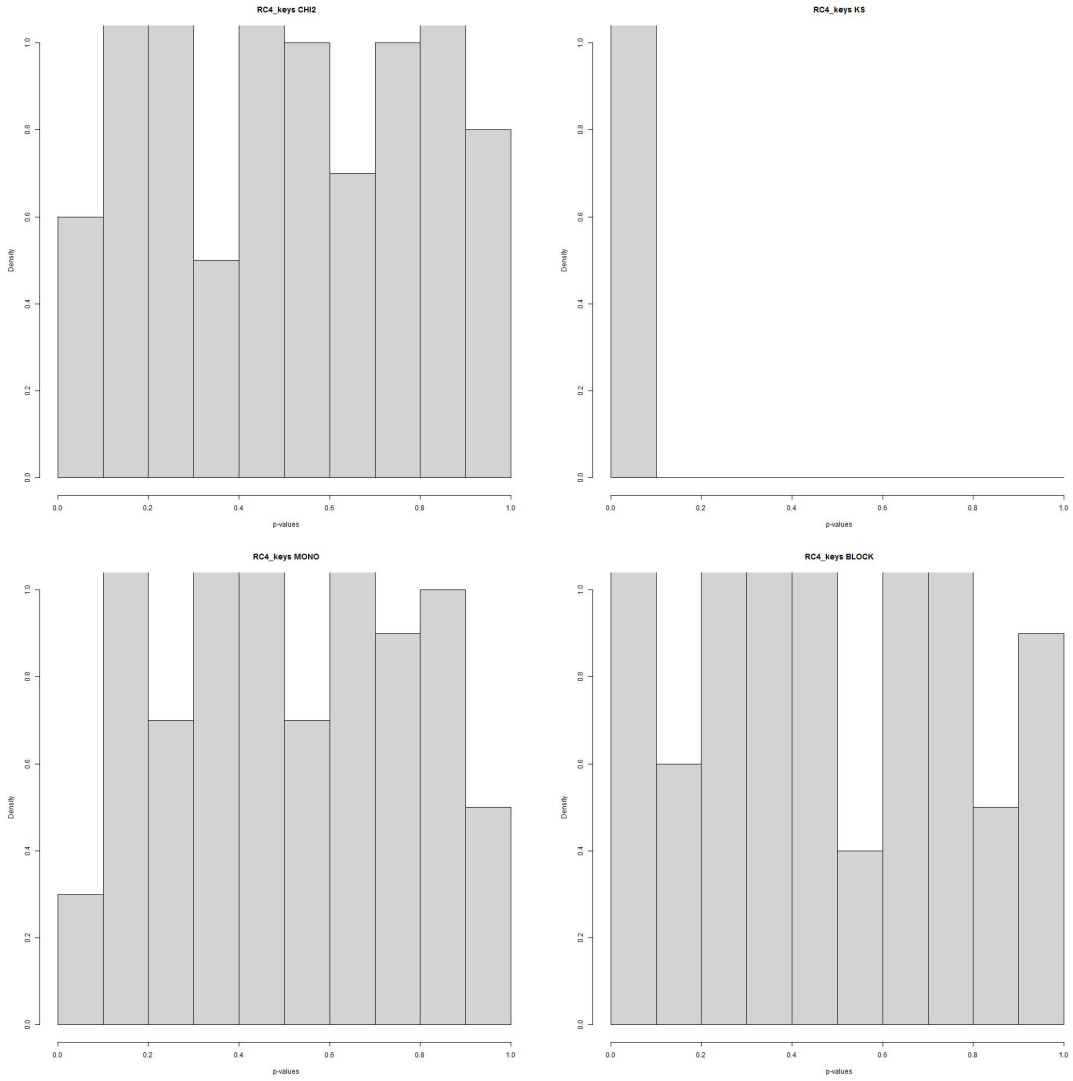
Rysunek 1.6: Histogramy p-wartości dla 4 testów generatora $\sqrt{2}$.



Rysunek 1.7: Histogramy p-wartości dla 4 testów generatora excel(18 . . .).



Rysunek 1.8: Histogramy p-wartości dla 4 testów generatora RC4 z użyciem jednego klucza.



Rysunek 1.9: Histogramy p-wartości dla 4 testów generatora RC4 z użyciem wielu kluczy.