Metody Monte Carlo

Raport nr 2

Ksawey Józefowski 277513

Zadanie 1: Generowanie prób z rozkładów wielowymiarowych (5 pkt)

1. Zaproponuj i zaimplementuj w R algorytm generowania prób z 4-wymiarowego rozkładu normalnego o parametrach
$$\boldsymbol{\mu} = \begin{bmatrix} 1 \\ 3 \\ -1 \\ -3 \end{bmatrix}, \quad \Sigma = \begin{bmatrix} 1 & 2 & 1 & 2 \\ 2 & 8 & 4 & 8 \\ 1 & 4 & 3 & 6 \\ 2 & 8 & 6 & 16 \end{bmatrix}.$$

2. Rozważmy następujący sympleks w $\mathbb{R}^2$:
$$S = \{(x, y) \in [0, \infty)^2 : \ x + y \leq 1\}$$
Zaproponuj i zaimplementuj w R algorytm generowania prób z rozkładu o gęstości
$$f(x, y) = 60xy^2 \mathbf{1}_S(x, y).$$
korzystając z rozkładów warunkowych. Narysuj wykres łączny (jointplot) wygenerowanej próby $((x_j, y_j))_{j=1}^n$, wraz z histogramami rozkładów brzegowych.

3. Zaproponuj i zaimplementuj w R algorytm generowania prób z rozkładu dwuwymiarowego o gęstości
$$f(x, y) = \frac{1}{C}(1 + x^2 + 2y^2)^{-4/3},$$
gdzie $C = \int_{\mathbb{R}^2}(1 + x^2 + 2y^2)^{-4/3}\mathrm{d}x\mathrm{d}y$ korzystając z metody ilorazów jednostajnych.Narysuj wykres łączny (jointplot) wygenerowanej próby $((x_j, y_j))_{j=1}^n$, wraz z histogramami rozkładów brzegowych.

Wskazówka: Należy zastosować ogólniejszą metodę ilorazów jednostajnych dla pewnego $r > 1$ (trzeba dobrać r, tak by funkcje $h_j(x, y)$ były ograniczone)

Rozwiązanie

Każdy wektor losowy $X \sim \mathcal{N}_k(\mu, \Sigma)$ można wygenerować z wykorzystaniem faktu:

$$X = \mu + LZ,$$

gdzie:

$$Z \sim \mathcal{N}_k(0, I_k)$$

L jest dolną macierzą trójkątną uzyskaną z dekompozycji Cholesky'ego

$$\Sigma = LL^T.$$

Kroki algorytmu, który zaimplementujemy:

1. Obliczamy macierz Cholesky'ego
$$L = \text{chol}(\Sigma).$$
2. Generujemy próbkę standardowych zmiennych normalnych
$$Z \sim \mathcal{N}(0, I_4).$$
3. Obliczamy
$$X = \mu + L^T Z$$
w R `chol()` zwraca górną macierz trójkątną, więc używamy transpozycji. Poniżej znajdziemy implementacje naszego algorytmu:

```r
cholesky <- function(n, mu, Sigma) {
  L <- chol(Sigma)
  Z <- matrix(rnorm(n*4), ncol=4)
  X <- Z %*% t(L) + mu

  df <- as.data.frame(X)
  colnames(df) <- paste0("X", 1:4)

  return(df)
}
```

Dla sprawdzenia poprawności w tabeli 1 przedstawiamy różnicę między macierzą Σ, a macierzą kowariancji próbkowych. Różnica między poszczególnymi wartościami macierzy wychodzi nie znacząca i potwierdza, że rozkład został wygenerowany poprawnie.

Table 1: Macierz różnicy macierzy Sigma i macierzy kowariancji próbkowych

	X1	X2	X3	X4
X1	0.022	0.022	0.011	0.003
X2	0.022	-0.088	-0.042	-0.119
X3	0.011	-0.042	-0.010	-0.023
X4	0.003	-0.119	-0.023	0.011

Rozważamy sympleks
$$S = \{(x, y) \in [0, \infty)^2 : x + y \leq 1\}$$
i rozkład o gęstości:

$$f(x, y) = 60xy^2 \mathbf{1}_S(x, y),$$

Rozkład brzegowy X wygląda następująco:

$$f_X(x) = \int_0^{1-x} 60xy^2 \, dy = 60x \int_0^{1-x} y^2 \, dy = 60x \left[\frac{y^3}{3}\right]_0^{1-x} = 20x(1-x)^3$$

dla $0 \leq x \leq 1$.

Obliczamy rozkład warunkowy $Y \mid X$

$$f_{Y|X}(y \mid x) = \frac{f(x, y)}{f_X(x)} = \frac{60xy^2}{20x(1-x)^3} = \frac{3y^2}{(1-x)^3}$$

dla $0 \leq y \leq 1 - x$.

Dystrybuanta dla X:

$$F_X(x) = \int_0^x 20t(1-t)^3\,dt = 10x^2 - 20x^3 + 15x^4 - 4x^5$$

Rozkład warunkowy $Y \mid X$ ma dystrybuantę:

$$F_{Y|X}(y \mid x) = \int_0^y \frac{3t^2}{(1-x)^3}\,dt = \frac{y^3}{(1-x)^3}$$

łatwo znaleźć odwrotna dystrybuantę:

$$F^{-1}_{Y|X}(u \mid x) = (1-x)\,u^{1/3}, \quad u \sim U(0,1)$$

Wtedy możemy skorzystać z algorytmu rozkładów warunkowych, którego implementację przedstawiamy poniżej:

```
F_X <- function(x) {
  10*x^2 - 20*x^3 + 15*x^4 - 4*x^5
}

F_X_inv <- function(u) {
  uniroot(function(x) F_X(x) - u, lower = 0, upper = 1)$root
}

con_method <- function(n) {
  u_X <- runif(n)
  x <- sapply(u_X, F_X_inv)
  u_Y <- runif(n)
  y <- (1 - x) * u_Y^(1/3)

  data.frame(x = x, y = y)
}
```

Na obrazku 1 znajdują się oba rozkłady brzegowe dla X i Y jak i wykres łączny próby. Rozkład brzegowy Y, zachowuje się podobnie do rozkładu normalnego, a rozkłąd X przypomina rozkład Poissona dla parametru około $\lambda = 3$. Wykres łączny wskazuje na silną zależność między zmiennymi.

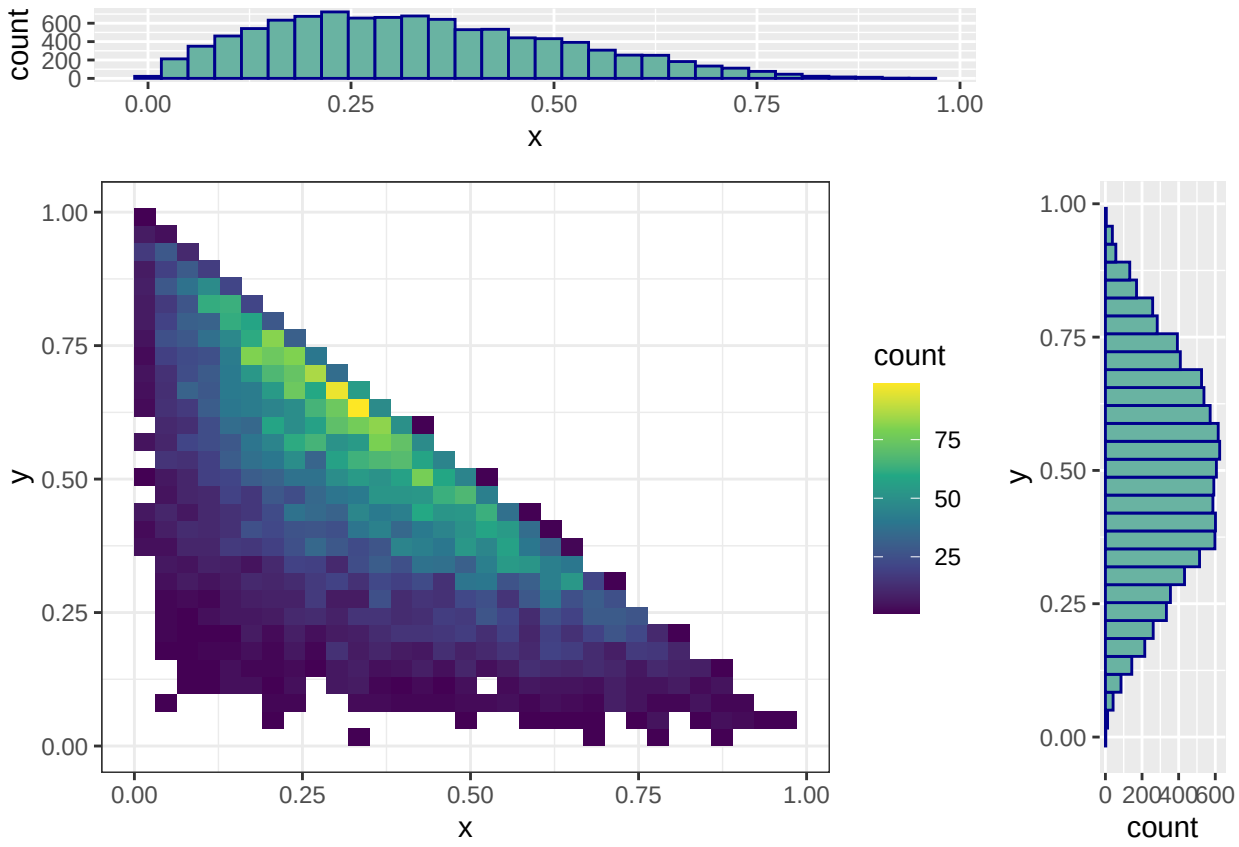


Figure 1: Wykres rozkładów brzegowych na obu osiach i wykres łączny wygenerowanej próby

Definiujemy zbiór

$$\tilde{A}_{f,r} = [0, \sup_{x\in\mathbb{R}^k} f(x)^{1/(1+kr)}] \times \prod_{j=1}^{k} [\inf h_j(x), \sup h_j(x)].$$

Weźmy $r = \frac{3}{3}$, wtedy

$$\sup_{x,y\in\mathbb{R}^2} f(x,y)^{1/(1+2r)} = \sup_{x,y\in\mathbb{R}^2} (1+x^2+2y^2)^{-\frac{1}{3}} = 1,$$

jest to prawdą ponieważ minimum występuje w punkcie $x = 0, y = 0$.

Teraz znajdziemy supremum funkcji h_1 i h_2, które zdefiniowane są w następujący sposób:

$$h_j(x) = x_j\, f(x_1, \ldots, x_k)^{\frac{r}{1+rk}}, \quad j = 1, \ldots, k$$

Dla $h_1(x, y)$ maksimum występuje gdy $y = 0$:

$$\frac{d}{dx}\left[h_1(x,0)\right] = (1+x^2)^{-\frac{3}{2}} > 0, \quad h_1(x,0) = x(1+x^2)^{-\frac{1}{2}} \xrightarrow{x\to\infty} 1$$

więc

$$\sup_{x,y} h_1(x,y) = 1.$$

Dla $h_2(x, y)$ maksimum występuje gdy $x = 0$:

$$\frac{d}{dy}\left[h_2(0,y)\right] = (1+2y^2)^{-\frac{3}{2}} > 0, \quad h_2(0,y) = y(1+2y^2)^{-\frac{1}{3}} \xrightarrow{y\to\infty} \frac{1}{\sqrt{2}}$$

więc

$$\sup_{x,y} h_2(x,y) = \frac{1}{\sqrt{2}}.$$

Ostatecznie zbiór $\tilde{A}_{f,r}$ wygląda następująco:

$$\tilde{A}_{f,r} = [0,1] \times [-1,1] \times \left[-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right].$$

Algorytm generowania próby:

1. Wybieramy $r > 1$ i wyznaczamy kostkę $\tilde{A}_{f,r}$.
2. Generujemy $u, v_1, \ldots, v_k$ z rozkładu jednostajnego na kostce.
3. Sprawdzamy warunek akceptacji:

$$u \leq (f(\frac{v_1}{u^r}, \ldots, \frac{v_k}{u^r}))^{\frac{1}{1+rk}}$$

Jeśli warunek spełniony, akceptujemy próbkę:

$$x_j = \frac{v_j}{u^r}, \quad j = 1, \ldots, k$$

W przeciwnym razie generujemy nowe liczby. Wtedy wektor $x = (x_1, \ldots, x_k)$ ma rozkład o gęstości proporcjonalnej do f.

```
f <- function(x, y) {
  (1 + x^2 + 2*y^2)^(-4/3)
}

uni_method <- function(n, r) {
  X <- matrix(NA, nrow = n, ncol = 2)
  i <- 1
  while (i <= n) {
    u <- runif(1)
    v1 <- runif(1, -1, 1)
    v2 <- runif(1, -1/sqrt(2), 1/sqrt(2))

    x1 <- v1 / (u^r)
    x2 <- v2 / (u^r)

    if(u <= f(x1, x2)^(1/(1+2*r))) {
        X[i, ] <- c(x1, x2)
        i <- i + 1
    }
  }
  df <- as.data.frame(X)
  names(df) <- c("x1","x2")
  return(df)
}
```

Na wykresie 2 obrazujemy wyniki funkcji dla $r = 3/2$ i kostki $[-1,1] \times \left[-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right]$. Wyniki ograniczamy do przedziału $[-15, 15]$ na obu osiach, z racji że badany rozkład ma bardzo ciężkie ogony i algorytm z dużym prawdopodobieństwem losuje *bardzo* odległe punkty od 0.

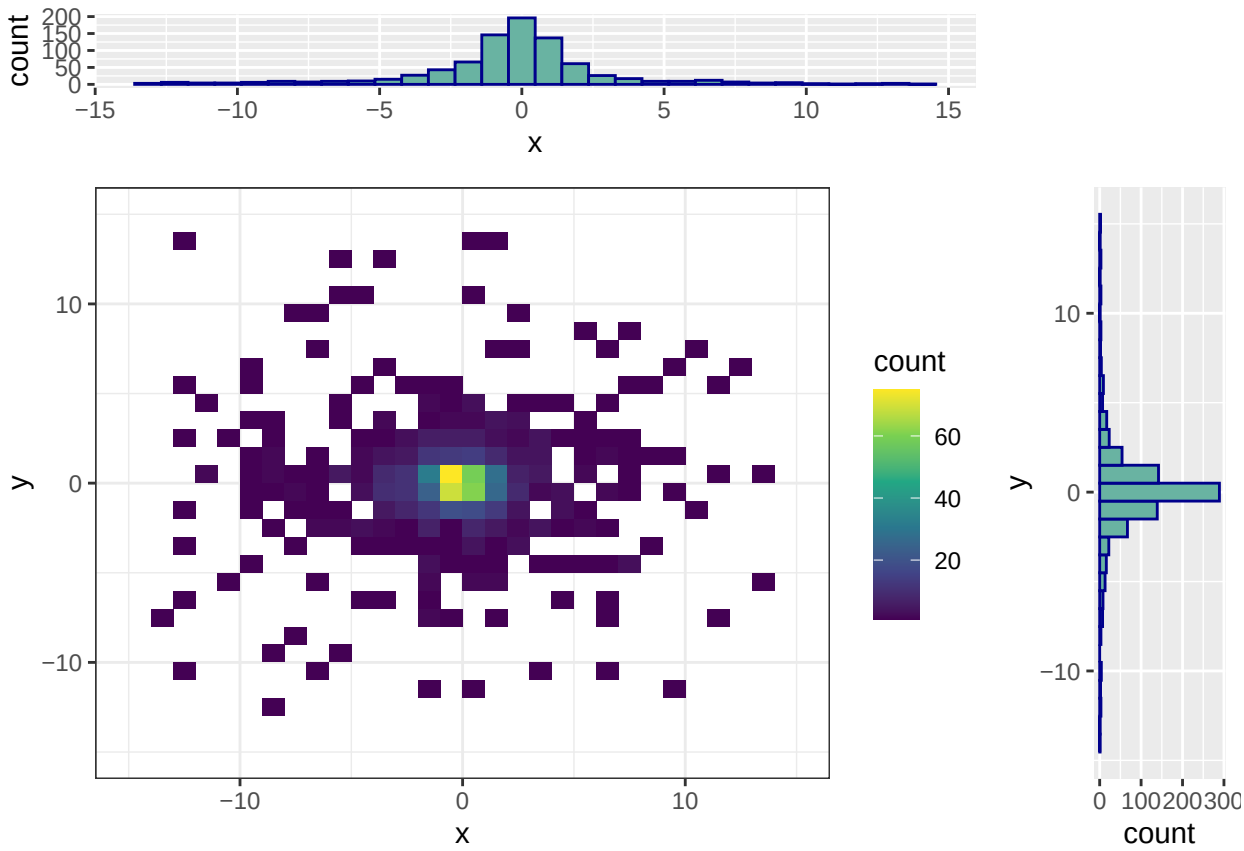


Figure 2: Wykres rozkładów brzegowych na obu osiach i wykres łączny wygenerowanej próby

Zadanie 2: Estymacja całek metodami Monte Carlo (3 pkt)

1. Oblicz w przybliżeniu objętość elipsoidy

$$E = \left\{(x, y, z) \in \mathbb{R}^2 : \frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} \leq 1\right\}$$

   dla wybranych RÓŻNYCH parametrów $a, b, c > 0$.

2. Żuk Mandelbrota jest podzbiorem płaszczyzny zespolonej, do którego należą punkty $p \in \mathbb{C}$, dla których ciąg

$$z_n = \begin{cases} z_0 = 0 \\ z_n = z_{n-1}^2 + p, \ n > 1 \end{cases}$$

   nie zbiega do nieskończoności. Oblicz przybliżoną powierzchnię zbioru Mandelbrota (rozumianego jako podzbiór płaszczyzny $\mathbb{R}^2$).

3. Oblicz przybliżoną wartość całki

$$\int_0^{2\pi} x \sin\left[(1/\cos\left(\ln\left(x+1\right)\right))^2\right] \mathrm{d}x.$$

Dla każdego z zaimplementowanych estymatorów, stwórz wykres zmienności wartości estymatora od liczności próby i zaznacz na nim asymptotyczne przedziały ufności.

Rozwiązanie

W tym zadaniu naszym celem jest przybliżenie objętości elipsoidy E. Algorytm, który zaimplementujemy wygląda następująco:

1. Losujemy n punktów (x_i, y_i, z_i) z prostopadłościanu:

$$x_i \in [-a, a], \quad y_i \in [-b, b], \quad z_i \in [-c, c].$$

2. Sprawdzamy, które punkty należą do elipsoidy:

$$ine_i = \begin{cases} 1 & \text{jeśli } \frac{x_i^2}{a^2} + \frac{y_i^2}{b^2} + \frac{z_i^2}{c^2} \leq 1, \\ 0 & \text{w przeciwnym razie.} \end{cases}$$

3. Estymujemy objętość elipsoidy:

$$\hat{V} = \frac{1}{n} \sum_{i=1}^{n} ine_i \cdot 8abc$$

```
elipse_vol <- function(a, b, c, n) {
  x <- runif(n, -a, a)
  y <- runif(n, -b, b)
  z <- runif(n, -c, c)
  ine <- (x^2 / a^2 + y^2 / b^2 + z^2 / c^2) <= 1
  V_hat <- cumsum(ine) / (1:n) * 8 * a * b * c

  sd <- sqrt(cumsum((ine * 8 * a * b * c - V_hat)^2) / (1:n))

  df <- data.frame(
    n = 1:n,
    est = V_hat,
    x = x,
```

```
    y = y,
    ine = ine,
    low = V_hat - 1.96 * sd / sqrt(1:n),
    up  = V_hat + 1.96 * sd / sqrt(1:n)
  )
}
```

Na rysunku 3 wizualizujemy, któe punkty należą do elipsoidy, a które do niej nie należą. Przyjmujemy $a = 2$, $b = 3$, $c = 4$ Z razji ograniczeń jakie daje knowersja z 3d na wykres 2d niektóre punkty są błędnie pokolorowane, lecz na rysunku możemy zauważyć, że zatacza się ikoniczny jajowaty kształt elipsoidy spłąszczonej do 2 wymiarów.

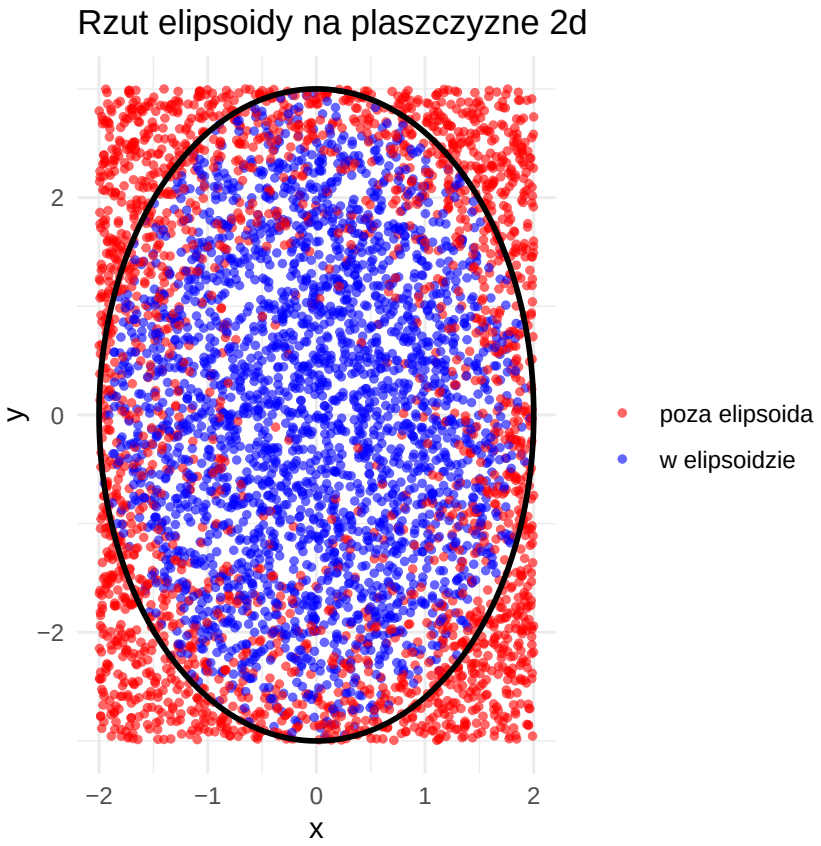


Figure 3: Obrazek przynależności wylosowanych punktów do elipsoidy spłaszczonej do 2d

Na rysunku 4 pokazujemy jak nasz estymator zbiega do prawdziwej objętości rozważanej elipsoidy ($\frac{4}{3}\pi abc$). Estymator wraz ze wzrostem n coraz lepiej radzi sobie z oszacowaniem objętości i od około $n = 3200$ robi to prawie idealnie.

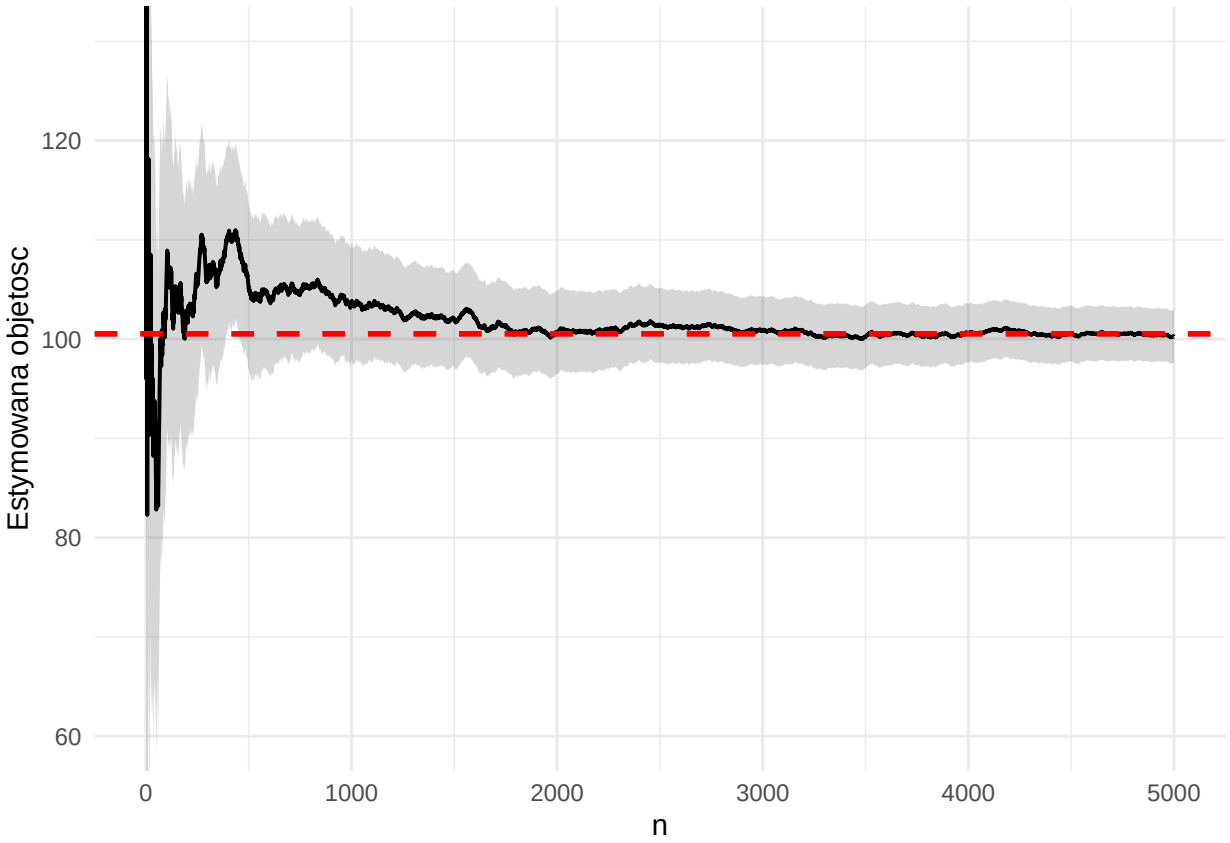


Figure 4: Zbieżność estymatora Monte Carlo do objętości elipsoidy

Do estymacji kolejnego zadania, czyli zbioru Mandelbrota użyjemy:

- Prostokąt ograniczający zbiór:
$$x \in [-2, 1], \quad y \in [-1.5, 1.5]$$
- Całkowita powierzchnia prostokąta:
$$A_{\text{box}} = (x_{\max} - x_{\min}) \cdot (y_{\max} - y_{\min}) = 3 \cdot 3 = 9$$

Algorytm, który zaimplementujemy wygląda nstępująco:

1. Losujemy n punktów (x_i, y_i) jednorodnie z prostokąta ograniczającego:
$$x_i \sim U(x_{\min}, x_{\max}), \quad y_i \sim U(y_{\min}, y_{\max})$$
2. Dla każdego punktu tworzymy liczbę zespoloną $p_i = x_i + iy_i$
3. Iterujemy ciąg z_n do maksymalnej liczby iteracji (w tym przypadku przyjmujemy 1000):
$$z_0 = 0, \quad z_n = z_{n-1}^2 + p_i$$
4. Punkt należy do zbioru Mandelbrota, jeśli $|z_n| < 2$ dla wszystkich n. My zastosujemy przybliżenie i będziemy analizować $n \leq 1000$
5. Przybliżona powierzchnia zbioru Mandelbrota:

$$\hat{P} = (x_{\max} - x_{\min}) \cdot (y_{\max} - y_{\min}) \cdot \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}_{\{|z_i| \leq 2\}}$$

```r
mandelbrot <- function(n) {
  x <- runif(n, -2, 1)
  y <- runif(n, -1.5, 1.5)
  p <- complex(real = x, imaginary = y)

  inm <- logical(n)
  P_hat <- numeric(n)
  for (i in 1:n) {
    z <- 0 + 0i
    for (j in 1:1000) {
      z <- z^2 + p[i]
      if (Mod(z) > 2) {
        inm[i] <- FALSE
        break
      } else {
        inm[i] <- TRUE
      }
    }
    P_hat[i] <- mean(inm[1:i]) * (1 - (-2)) * (1.5 - (-1.5))
  }

  sd <- sqrt(cumsum((inm * 9 - P_hat)^2) / (1:n))
  df <- data.frame(
          n = 1:n,
          x = x,
          y = y,
          inm = inm,
          P_hat = P_hat,
          low = P_hat - 1.96 * sd / sqrt(1:n),
          up = P_hat + 1.96 * sd / sqrt(1:n)
        )
  return(list(df=df, P=P_hat[n]))
}
```

Obrazek 5 pokazuje czy wylosowany punkt należy do zbioru czy nie należy. Czerwone punkty dokładnie kształtują zarys zbioru mandelbrota, co świadczy, że dobrze napisaliśmy funkcję.

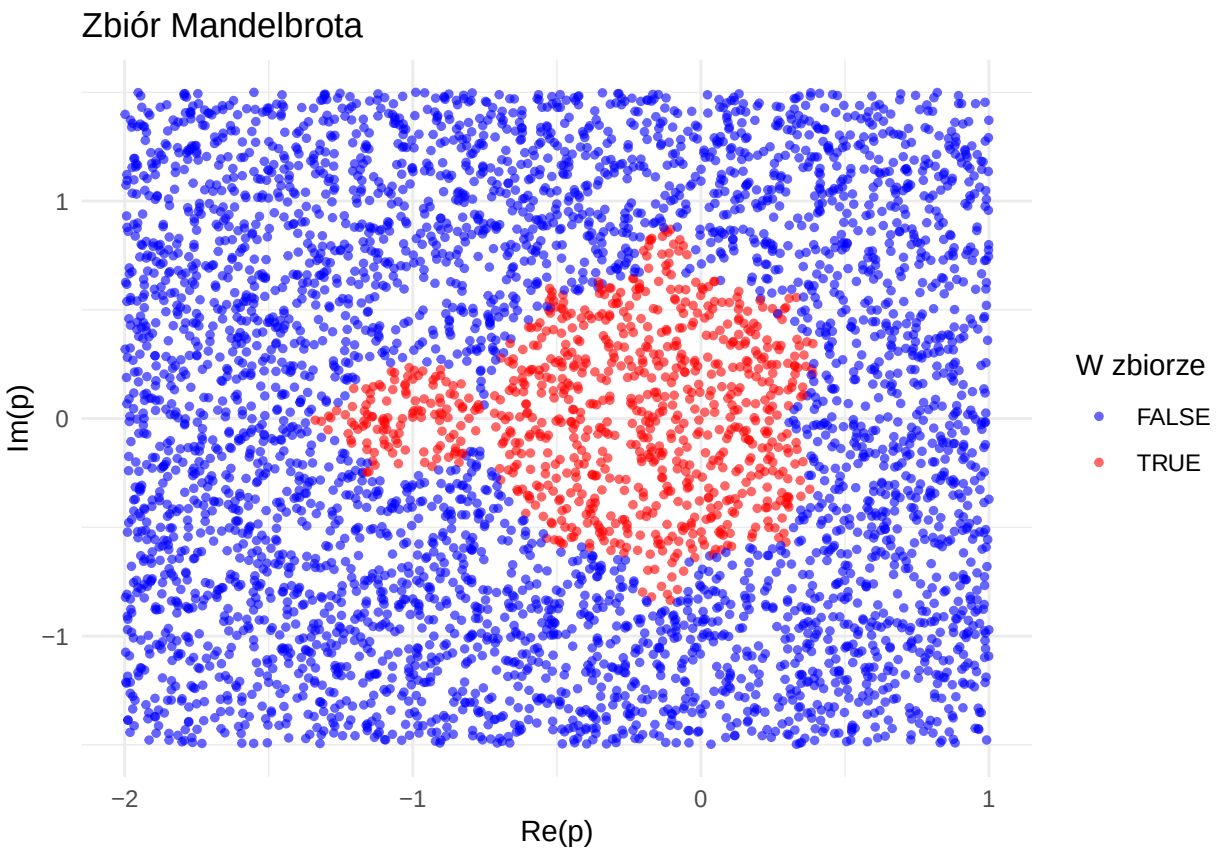


Figure 5: Obrazek przynależności wylosowanych punktów do zbioru Mandelbrota

Obrazek 6 wizualizuje zbieżnośćestymatora do prawdziwej wartości powierzchni zbioru mandelbrota. Estymator od około $n = 4000$ bardzo dobrze szacuje wartość powierzchni.

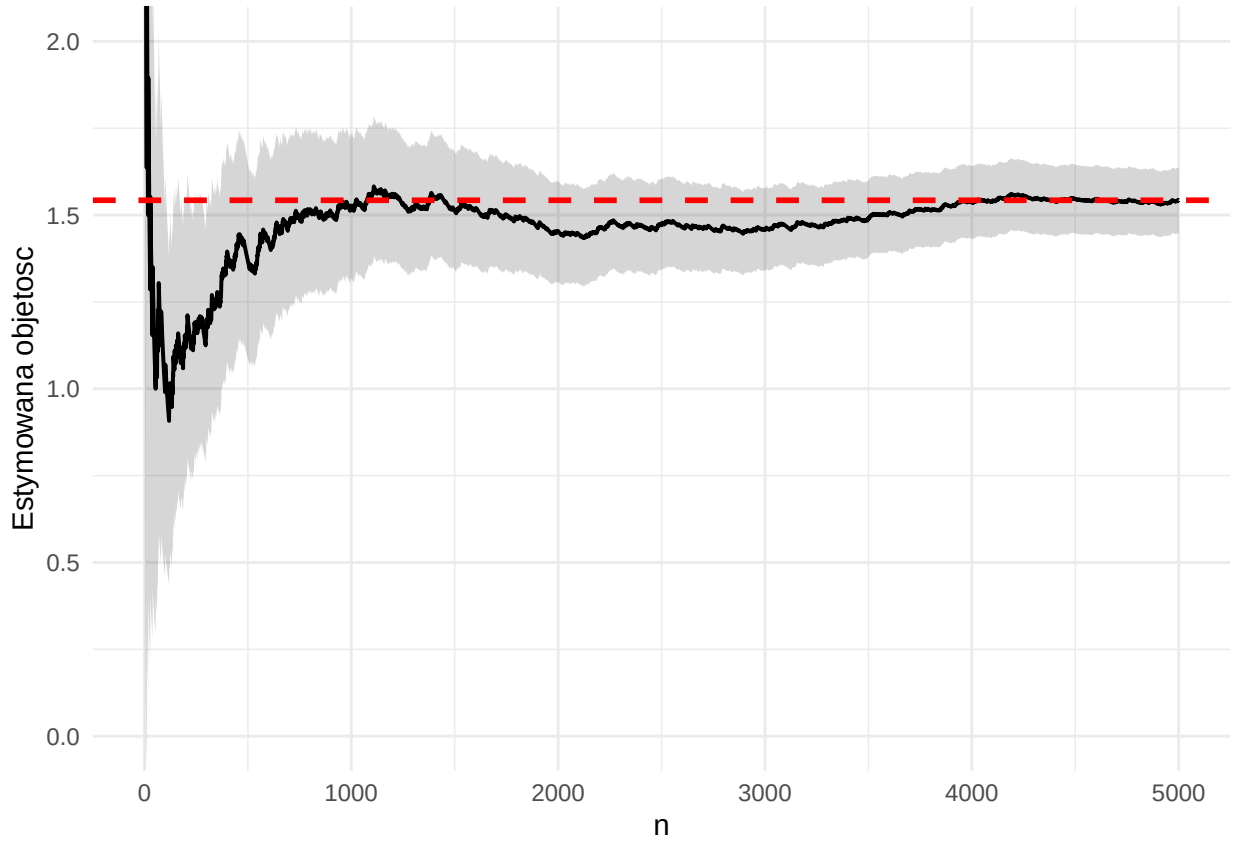


Figure 6: Zbieżność estymatora Monte Carlo do powierzchni zbioru Mandelbrota

W tej części będziemy estymować całkę

$$I = \int_0^{2\pi} x \sin\left[\left(\frac{1}{\cos(\ln(x+1))}\right)^2\right] dx$$

przy użyciu klasycznego estymatora Monte Carlo z różnymi gęstościami próbkowania. Algorytmm, który zaimplementujemy będzie wyglądać następująco: 1. Wybieramy gęstość próbkowania $g(x)$:
Jednostajna:

$$g_1(x) = \frac{1}{2\pi}, \quad x \in [0, 2\pi]$$

Druga gęstość wyrażona wzorem:

$$g_2(x) = \frac{x}{2\pi^2}, \quad x \in [0, 2\pi]$$

2. Generujemy próbę $X_1, \ldots, X_n \sim g(x)$.
3. Obliczamy estymator Monte Carlo:

$$\hat{I}_n = \frac{1}{n}\sum_{j=1}^{n} h(X_j), \quad h(x) = \frac{f(x)}{g(x)}, \quad f(x) = x \sin\left[\left(\frac{1}{\cos(\ln(x+1))}\right)^2\right]$$

4. Wyznaczamy asymptotyczny przedział ufności dla estymatora:

$$\hat{I}_n \pm 1.96\frac{\sigma_n}{\sqrt{n}}, \quad \sigma_n^2 = \frac{1}{n-1}\sum_{j=1}^{n}(h(X_j) - \hat{I}_n)^2$$

```r
mc_integral <- function(n, Gestosc = 1) {
  if (Gestosc == 1) {
    x <- runif(n, 0, 2*pi)
    h <- 2*pi * x * sin((1 / cos(log(x + 1)))^2)
    label <- "Jednostajna"
  } else if (Gestosc == 2) {
    x <- 2*pi * sqrt(runif(n))
    h <- 2*pi^2 * sin((1 / cos(log(x + 1)))^2)
    label <- "Druga"
  }

  I_hat <- cumsum(h) / (1:n)
  sd <- sapply(1:n, function(k) sd(h[1:k]))

  df <- data.frame(
    n = 1:n,
    est = I_hat,
    low = I_hat - 1.96 * sd / sqrt(1:n),
    up  = I_hat + 1.96 * sd / sqrt(1:n),
    Gestosc = label
  )
}
```

Na obrazku 7 widać jak wartości estymatorów zbiegają do wartości całki, jak i również ich przedziały ufności. Estymator oparty na "drugiej" gęstości zbiega lepiej niż ten oparty na jednostajnej, lecz nadal nie idealnie.

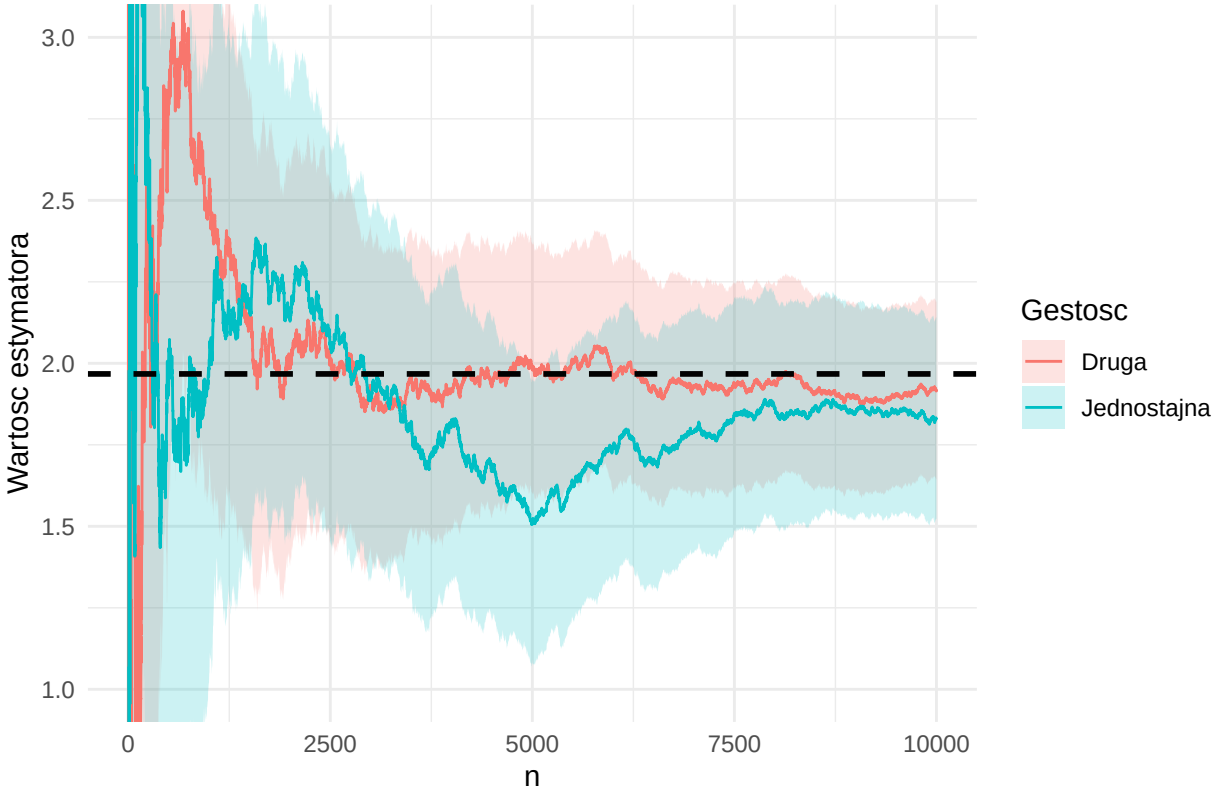


Figure 7: Wykres zbieżności estymatora, dla różnych gęstości z przedziałami ufności

Zadanie 3: Obciążenie i metoda delta (2 pkt)

Załóżmy, że mamy wygenerowaną próbę prostą $(x_j)_{j=1}^n$ ze zmiennej X o rozkładzie normalnym $\mathcal{N}(0, \sigma^2)$, gdzie $\sigma^2 > 0$ jest nieznane. Chcemy estymować wartości funkcji tworzącej momenty (czyli transformaty Laplace'a)

$$M_X(t) = \mathbb{E}e^{tX}.$$

Rozważmy dwie metody:

1. Estymacja $\widehat{M_X}(t) = \mathbb{E}e^{tX}$ metodą Monte-Carlo na podstawie próby $(x_j)_{j=1}^n$.
2. Estymacja wariancji σ^2 za pomocą wariancji próbkowej σ_n^2 z $(x_j)_{j=1}^n$ i następnie obliczenie estymatora $\widehat{M_X}(t) = e^{\frac{t^2}{2}\sigma_n^2}$.

Oceń jakość obu metod estymacji, wyznaczając asymptotyczne przedziały ufności obu estymatorów dla rosnącej liczności próby n z rozkładu $\mathcal{N}(0, \sigma^2)$ i wybranego $\sigma^2 \neq 1$.

Rozwiązanie

Mamy próbę prostą:

$$X_1, \ldots, X_n \sim N(0, \sigma^2), \quad \sigma^2 > 0, \quad \sigma^2 \neq 1$$

Funkcja tworząca momenty dla pojedynczej obserwacji wygląda następująco:

$$M_X(t) = E\left[e^{tX}\right] = \exp\left(\frac{t^2\sigma^2}{2}\right)$$

Ustalamy $\sigma^2 = 2, \quad t = 0.7$. Wtedy:

$$M_X(0.7) = \exp\left(\frac{0.7^2 \cdot 2}{2}\right) = \exp(0.49) \approx 1.632$$

Metoda delta służy do estymacji wariancji funkcji $f(\theta)$ nieznanego parametru θ na podstawie estymatora $\hat{\theta}$. Jeżeli $\hat{\theta}$ jest asymptotycznie normalny:

$$\sqrt{n}(\hat{\theta} - \theta) \xrightarrow{d} \mathcal{N}(0, \sigma^2)$$

oraz f jest różniczkowalna w otoczeniu θ, to estymator $f(\hat{\theta})$ jest również asymptotycznie normalny:

$$\sqrt{n}\big(f(\hat{\theta}) - f(\theta)\big) \xrightarrow{d} \mathcal{N}\big(0, \sigma^2[f'(\theta)]^2\big)$$

W implementacji zastosujemy wersję metody delta opartą na podziale próby na k rozłącznych bloków o równej liczności. Dla każdego bloku wyznaczany jest estymator drugiego momentu

$$\hat{\sigma}_j^2 = \frac{1}{l}\sum_{i \in B_j} X_i^2$$

a następnie wartość funkcji tworzącej momenty

$$M_j(t) = \exp\left(\frac{t^2\hat{\sigma}_j^2}{2}\right)$$

Estymator końcowy ma postać średniej z wartość funkcji tworzącej momenty dla każdego bloku:

$$\hat{M}_k(t) = \frac{1}{k}\sum_{j=1}^{k} M_j(t)$$

Wariancje estymatora szacujemy na podstawie wariancji między blokami, a przedział ufności konstruujemy z wykorzystaniem kwantyla z rozkładu t-Studenta:

$$\hat{M}_k(t) \pm t_{1-\alpha/2,\,k-1}\sqrt{\frac{\widehat{\mathrm{Var}}(M_j(t))}{k}}.$$

Poniżej przedstawiamy implementację metody Delta (przyjmiemy $k = 10$ bloków) oraz metody Monte-Carlo dla porównania.

```
MC_Mx <- function(x, t, alpha) {
  n <- length(x)

  Y <- exp(t * x)
  M_hat <- mean(Y)
  se  <- sqrt(var(Y) / n)

  z <- qnorm(1 - alpha / 2)
  CI <- M_hat + c(-1, 1) * z * se

  df <- data.frame(
    estimator = M_hat,
    CI = CI,
    se = se
  )
}

Delta_method <- function(x, t, k = 10, alpha = 0.05) {
  n <- length(x)
  l <- floor(n / k)

  x <- x[1:(k * l)]
  blocks <- split(x, rep(1:k, each = l))

  theta_blocks <- sapply(blocks, function(b) {
    s2_hat <- mean(b^2)
    exp(0.5 * t^2 * s2_hat)
  })

  CI <- mean(theta_blocks) + c(-1, 1) * qt(1 - alpha / 2, df = k - 1) * sqrt(var(theta_blocks) / k)

  df <- data_frame(
    estimator = mean(theta_blocks),
    CI = CI
  )
}
```

Table 2: Porównanie estymatorów Monte-Carlo i metody delta dla $M_X(t)$

n	MC est.	MC CI low	MC CI high	Delta est.	Delta CI low	Delta CI high
50	1.296900	0.826635	1.767165	1.644951	1.212816	2.077085
100	1.548053	1.301413	1.794693	1.640687	1.193374	2.088000
500	1.811109	1.535669	2.086550	1.740031	1.570472	1.909591
1000	1.628728	1.493172	1.764284	1.659278	1.568150	1.750405
5000	1.637104	1.575268	1.698940	1.635235	1.598967	1.671503
10000	1.652381	1.608841	1.695922	1.634749	1.616951	1.652547
50000	1.630796	1.612418	1.649173	1.628580	1.621119	1.636041

Prawdziwa wartość funkcji tworzącej momenty wynosi:

$$M_X(t) = 1.632316.$$

Dla rosnącej liczności próby n, z tabeli 2 odczytujemy, że obie metody są asymptotycznie poprawne. Metoda Delta wraz ze wzrostem liczności próby wyraźnie stabilizuje się w pobliżu wartości prawdziwej. Długość asymptotycznych przedziałów ufności systematycznie maleje. Dla dużych prób przedziały są wąskie i dobrze wyznaczają wartość prawdziwą. Estymator oparty na metodzie Monte-Carlo również zbiega do wartości prawdziwej, jednak dla małych prób widoczna jest większa zmienność. Wraz ze wzrostem n estymator metody Delta bardzo szybko się stabilizuje, a przedziały ufności są wyraźnie węższe niż w metodzie MC, co wskazuje na większą efektywność asymptotyczną.

Zadanie 4: Zmniejszanie wariancji w estymacji całek metodami Monte-Carlo (4 pkt)

1. Zaproponuj przynajmniej trzy różne estymatory Monte-Carlo estymujące wartość całki

$$I_1 = \int_{\frac{1}{2}}^{1} \frac{e^{1/x} x^{1/3}}{(1+x)^{7/3}} \mathrm{d}x.$$

Skorzystaj z różnych metod zaprezentowanych na wykładzie. Porównaj zaproponowane estymatory, analizując ich wariancje bądź asymptotyczne przedziały ufności. Określ, który z estymatorów jest najlepszy.

2. Zaproponuj estymator Monte-Carlo całki

$$I_2 = \int_{\frac{1}{2}}^{\infty} \frac{e^{1/x} x^{1/3}}{(1+x)^{7/3}} \mathrm{d}x.$$

Uwaga 1: Ogłaszam konkurs na najlepszy estymator całki I_2. Osoba, która zaproponuje najefektywniejszy (o najwęższym asymptotycznym przedziale ufności - czyli jednocześnie najmniejszej wariancji), będzie darzona nieskończoną **estymą** i dodatkowo otrzyma 5 punktów z aktywności (czyli maksymalną możliwą liczbę punktów).

Uwaga 2: Chociaż w funkcji podcałkowej występuje gęstość **odwróconego rozkładu Beta** z parametrami $\alpha = \frac{4}{3}$ i $\beta = 1$, to estymator Monte-Carlo oparty o próbę generowaną z tego rozkładu jest wyjątkowo nieefektywny (można się zastanowić dlaczego). Można jednak poczytać w jaki sposób taki odwrócony rozkład Beta się otrzymuje i w oparciu o tę wiedzę spróbować wymyślić bardziej efektywny estymator.

Rozwiązanie

Rozważamy całkę

$$I = \int_{\frac{1}{2}}^{1} f(x)\,dx, \quad f(x) = \frac{e^{1/x}\, x^{1/3}}{(1+x)^{7/3}}$$

która nie ma prostej postaci analitycznej. Do jej oszacowania wykorzystamy trzy metody Monte Carlo:

1. Zwykła metoda Monte Carlo

$$\hat{I}_{\mathrm{MC}} = \frac{1}{n} \sum_{i=1}^{n} f(X_i), \quad X_i \sim U(0.5, 1)$$

2. Metoda zmiennych antytetycznych

$$\hat{I}_{\mathrm{ant}} = \frac{1}{n} \sum_{i=1}^{n/2} \frac{f(U_i) + f(\frac{3}{2} - U_i)}{2}, \quad U_i \sim U(0.5, 1)$$

3. Metoda Importance Sampling Z funkcją pomocniczą

$$g(x) = \frac{x^{1/3}}{(1+x)^{7/3}}$$

i stałą normalizującą $C_g = \int_{0.5}^{1} g(x)dx$, estymator ma postać

$$\hat{I}_{\mathrm{IS}} = \frac{1}{n} \sum_{i=1}^{n} \frac{f(Y_i)}{\tilde{g}(Y_i)}, \quad Y_i \sim \tilde{g}$$

gdzie $\tilde{g}(x) = g(x)/C_g$ jest gęstością na $[0.5, 1]$.

Poniżej przedstawiamy implementacje estymatorów:

```r
f <- function(x) {
  exp(1/x) * x^(1/3) / (1 + x)^(7/3)
}
```

```r
MC <- function(n) {
  x <- runif(n, 0.5, 1)
  h <- f(x)
  est <- cumsum(h) / (1:n)
  se <- sapply(1:n, function(k) sd(h[1:k]) / sqrt(k))

  data.frame(
    n = 1:n,
    name = "MC",
    est = 0.5 * est,
    low = 0.5 * (est - qnorm(0.975) * se),
    up = 0.5 * (est + qnorm(0.975) * se),
    var = (0.5 * se)^2
  )
}
```

```r
Antytetyczne <- function(n) {
  u <- runif(n)
  x1 <- 0.5 + 0.5 * u
  x2 <- 0.5 + 0.5 * (1 - u)
  h <- (f(x1) + f(x2)) / 2
  est <- cumsum(h) / (1:n)
  se <- sapply(1:n, function(k) sd(h[1:k]) / sqrt(k))

  data.frame(
    n = 1:n,
    name = "Antytetyczne",
    est = 0.5 * est,
    low = 0.5 * (est - qnorm(0.975) * se),
    up = 0.5 * (est + qnorm(0.975) * se),
    var = (0.5 * se)^2
  )
}
```

```r
Imp_samp <- function(n) {
  g <- function(x) x^(1/3) / (1 + x)^(7/3)
  Cg <- integrate(g, 0.5, 1)$value
  x <- numeric(n)
  i <- 1
  while (i <= n) {
    y <- runif(1, 0.5, 1)
    u <- runif(1)
    if (u <= g(y)) {
      x[i] <- y
      i <- i + 1
    }
  }

  w <- f(x) / (g(x) / Cg)
```

```r
  est <- cumsum(w) / (1:n)
  se <- sapply(1:n, function(k) sd(w[1:k]) / sqrt(k))

  data.frame(
    n = 1:n,
    name = "Importance Sampling",
    est = est,
    low = est - qnorm(0.975) * se,
    up = est + qnorm(0.975) * se,
    var = se^2
  )
}
```

Table 3: Porównanie metod Monte Carlo, Antytetycznej i Importance Sampling

	n	name	est	low	up	var
10000	10000	MC	0.538403	0.533870	0.542936	5e-06
20000	10000	Antytetyczne	0.536576	0.535252	0.537900	0e+00
30000	10000	Importance Sampling	0.537059	0.533935	0.540182	3e-06

W tabeli 3 przedstawiamy wyniki estymacji dla $n = 10000$. Wartość referencyjna, obliczona numerycznie za pomocą `integrate` w R, wynosi $I \approx 0.5368601$. Metoda klasyczna daje poprawne oszacowanie, ale ma największą wariancję. Metoda zmiennych antytetycznych estymuje całkę prawie idealnie. Długości jej przedziałów ufności maleją i są ogólnie najmniejsze, a wariancja tej metody jest najmniejsza pośró wszystkich. Metoda Importance Sampling osiąga lepsze wyniki estymacji niż klasyczna metoda, jak i również jej przedziały ufności i wariancja są mniejsze, lecz dalej większe niż Antytetycznych. Najlepszym estymatorem w tym przypadku jest zatem estymator bazujący na metodzie zmiennych antytetycznych.

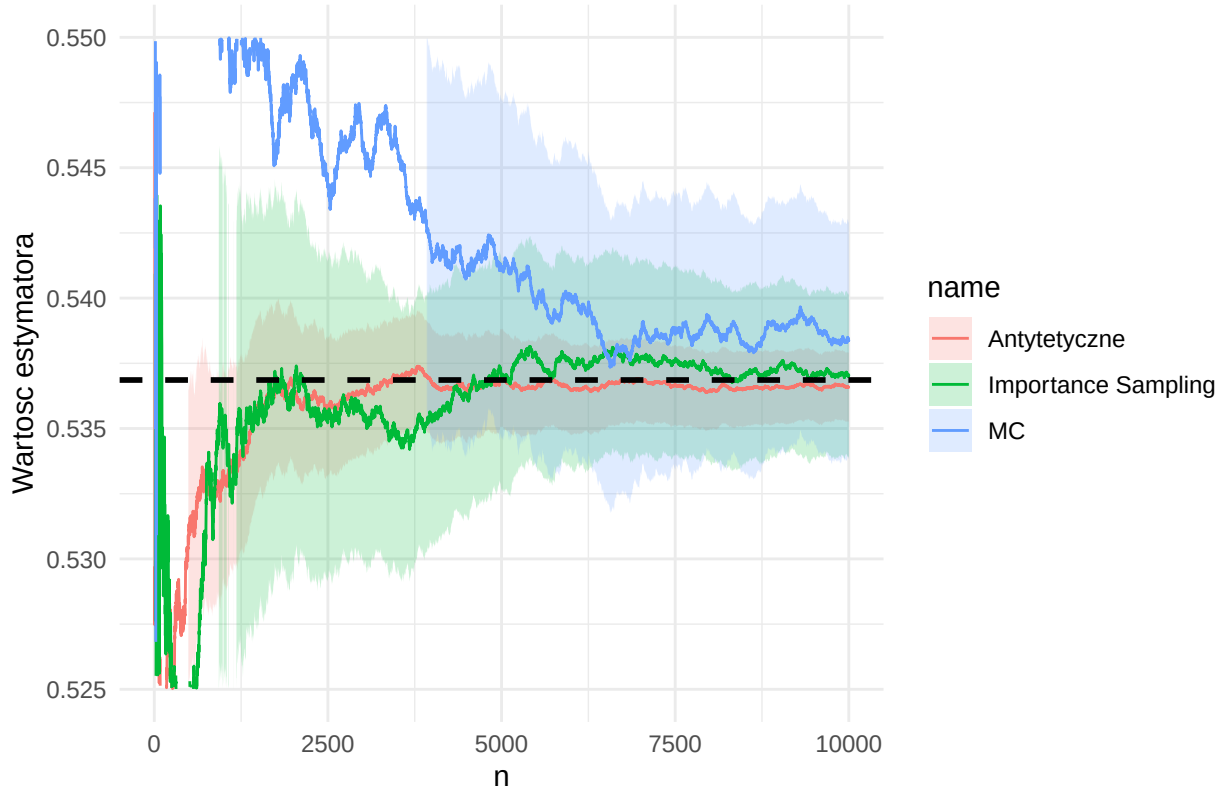


Figure 8: Wykres zbieżności estymatorów do wartości całki

Na wykresie 8 obrazujemy, jak estymatory zbiegają do wartości prawdziwej wraz ze wzrostem liczby próbek n, dla dodatkowej weryfikacji.

Rozważamy całkę

$$I_2 = \int_{1/2}^{\infty} \frac{e^{1/x}\, x^{1/3}}{(1+x)^{7/3}}\, dx.$$

Do stworzenia estymatora tej całki najpierw skonstruujemy gęstość pomocniczą. Zmienną losową

$$U \sim \text{Beta}\left(\frac{4}{3}, 1\right)$$

przekształcamy zgodnie z transformacją

$$X = U^{-\gamma} - \frac{1}{2}, \qquad \gamma > 1.$$

W ten sposób otrzymujemy gęstość pomocniczą o postaci

$$g(x) = \frac{\alpha}{\gamma}\,(x + \tfrac{1}{2})^{-\left(\frac{\alpha}{\gamma}+1\right)}, \qquad x \geq \frac{1}{2},$$

gdzie $\alpha = \frac{4}{3}$.
Estymator całki oparty na importance sampling będzie dany wzorem

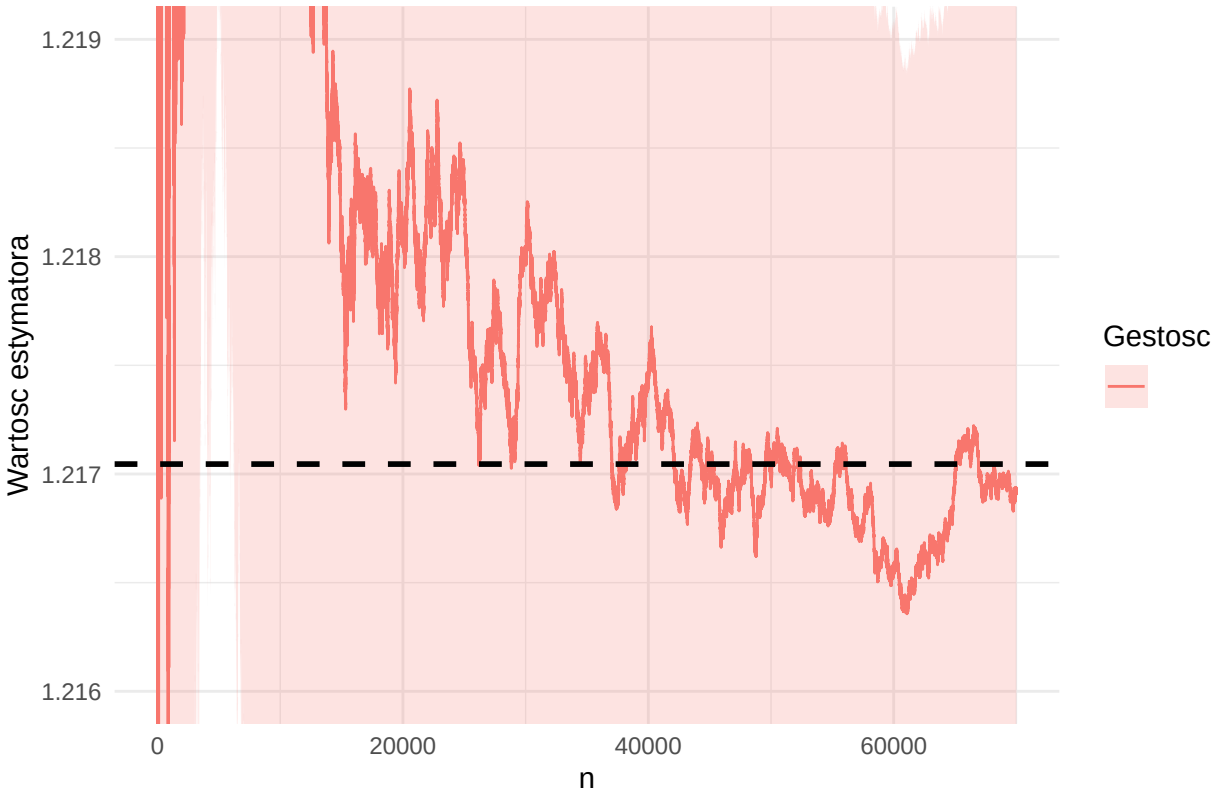
$$\hat{I}_n = \frac{1}{n}\sum_{i=1}^{n} \frac{f(X_i)}{g(X_i)},$$

gdzie

$$f(x) = \frac{e^{1/x}\, x^{1/3}}{(1+x)^{7/3}}, \qquad X_i \sim g.$$

Zastosowana gęstość pomocnicza powinna zachowywać poprawny ogon potęgowy funkcji podcałkowej, dodatkowo zagęszcza próbę w pobliżu $x = 1/2$, gdzie duży wpływ ma $e^{1/x}$. Wykorzystana gęstość nie jest konstruowana bezpośrednio jako odwrócony rozkład Beta, który prowadzi do nieefektywnego estymatora, lecz bazuje na mechanizmie jego powstawania poprzez odpowiednią transformację zmiennej losowej o rozkładzie Beta. Modyfikując tę transformację uzyskuje się lepsze dopasowanie gęstości pomocniczej do struktury funkcji podcałkowej, bez nadmiernego zwiększania wariancji. Estymator oparty na tej gęstości oscyluje blisko wartości rzeczywistej całki $I_2 \approx 1.217$, jednak wymaga stosunkowo dużej liczby prób n, aby osiągnąć wyższą dokładność.

```r
est_I2 <- function(n) {
  alpha <- 4/3
  gamma <- 1.3

  u <- rbeta(n, alpha, 1)
  x <- u^(-gamma) - 1/2

  g <- function(x) {
    alpha / gamma * (x + 1/2)^(-(alpha/gamma + 1))
  }

  w <- f(x) / g(x)

  est <- cumsum(w) / seq_len(n)
  s2 <- cumsum((w - est)^2) / seq_len(n)
  se <- sqrt(s2 / seq_len(n))

  data.frame(
    n = 1:n,
    est = est,
    low = est - 1.96 * se,
    up  = est + 1.96 * se,
    Gestosc = ""
  )
}
```


1.219
1.218
1.217
1.216
Wartosc estymatora
0
20000
40000
60000
n
Gestosc