Laboratorium 3

Wstęp do Analizy Danych | Politechnika Krakowska

Jakub Kapała

Numer albumu: 151885

Data: 05.04.2025

Zadanie 1 - Plamy słoneczne

Treść

1.1 Ze strony Royal Observatory of Belgium pobrać dane dotyczące liczby plam słonecznych. Użyć polecenia `read.csv`, aby wczytać dane do R. Link do danych: https://www.sidc.be/SILSO/datafiles Bezpośredni link do pliku .csv z danymi: http://www.sidc.be/SILSO/INFO/sndtotcsv.php

1.2 Użyć funkcji `colnames`, aby nadać pobranemu data frame wygodne nazwy kolumn – sprawdzić w tym celu na podanej stronie internetowej znaczenie liczb w poszczególnych kolumnach. Jaka była największa zarejestrowana kiedykolwiek liczba plam na Słońcu? Kiedy miało to miejsce?

1.3 Przedstawić na wykresie liczbę plam słonecznych w funkcji czasu w latach 1990 - 2023.

1.4 Powyższy wykres jest bardzo nieciągły – liczba plam na Słońcu zmienia się chaotycznie w krótkim czasie. Znaleźć i przedstawić na wykresie zależność czasową liczby plam na Słońcu *uśrednioną w skali roku*, tzn. średnią arytmetyczną z dni od 182 dni przed do 182 dni po danym dniu.

1.5 Przedstawić na wykresie liczbę plam słonecznych od stycznia do maja 2023.

1.6 Na tak krótkim przedziale czasu chaotyczne zmiany dominują nad długofalowym trendem i można oczekiwać, że rozkład liczby plam słonecznych jest przynajmniej w pewnym przybliżeniu normalny. Przedstawić histogram liczby plam od stycznia do maja 2023, znaleźć średnią liczbę plam i odchylenie standardowe oraz zbadać normalność rozkładu na wykresie kwantyl-kwantyl (polecenie `qqnorm`).

Rozwiązanie 1.1. Wczytanie danych:

```
# Pobranie danych z podanego linku
url <- "http://www.sidc.be/SILSO/INFO/sndtotcsv.php"

# Wczytanie danych do R
sunspots_data <- read.csv(url, header = FALSE, sep = ";")

# Wyswietlenie pierwszych kilku wierszy danych
head(sunspots_data)
```

```
##     V1 V2 V3       V4 V5 V6 V7 V8
## 1 1818  1  1 1818.001 -1 -1  0  1
## 2 1818  1  2 1818.004 -1 -1  0  1
## 3 1818  1  3 1818.007 -1 -1  0  1
## 4 1818  1  4 1818.010 -1 -1  0  1
## 5 1818  1  5 1818.012 -1 -1  0  1
## 6 1818  1  6 1818.015 -1 -1  0  1
```

1.2.: Nadanie nazwy kolumnom, analiza największej liczby plam słonecznych.

Nadanie nazw kolumnom:

```
colnames(sunspots_data) <- c(
  "Year",
  "Month",
  "Day",
  "Date_In_Fraction_Of_Year",
  "Daily_Total_Sunspot_Number",
  "Daily_Standard_Deviation",
  "Daily_Observations",
  "Definitive_Provisional_Indicator"
)
head(sunspots_data)
```

```
##   Year Month Day Date_In_Fraction_Of_Year Daily_Total_Sunspot_Number
## 1 1818     1   1                 1818.001                         -1
## 2 1818     1   2                 1818.004                         -1
## 3 1818     1   3                 1818.007                         -1
## 4 1818     1   4                 1818.010                         -1
## 5 1818     1   5                 1818.012                         -1
## 6 1818     1   6                 1818.015                         -1
##   Daily_Standard_Deviation Daily_Observations Definitive_Provisional_Indicator
## 1                       -1                  0                                1
## 2                       -1                  0                                1
## 3                       -1                  0                                1
## 4                       -1                  0                                1
## 5                       -1                  0                                1
## 6                       -1                  0                                1
```

Znalezienie największej liczby plam na słońcu:

```
max_sunspots <- max(sunspots_data$Daily_Total_Sunspot_Number, na.rm = TRUE)
max_sunspots
```

```
## [1] 528
```

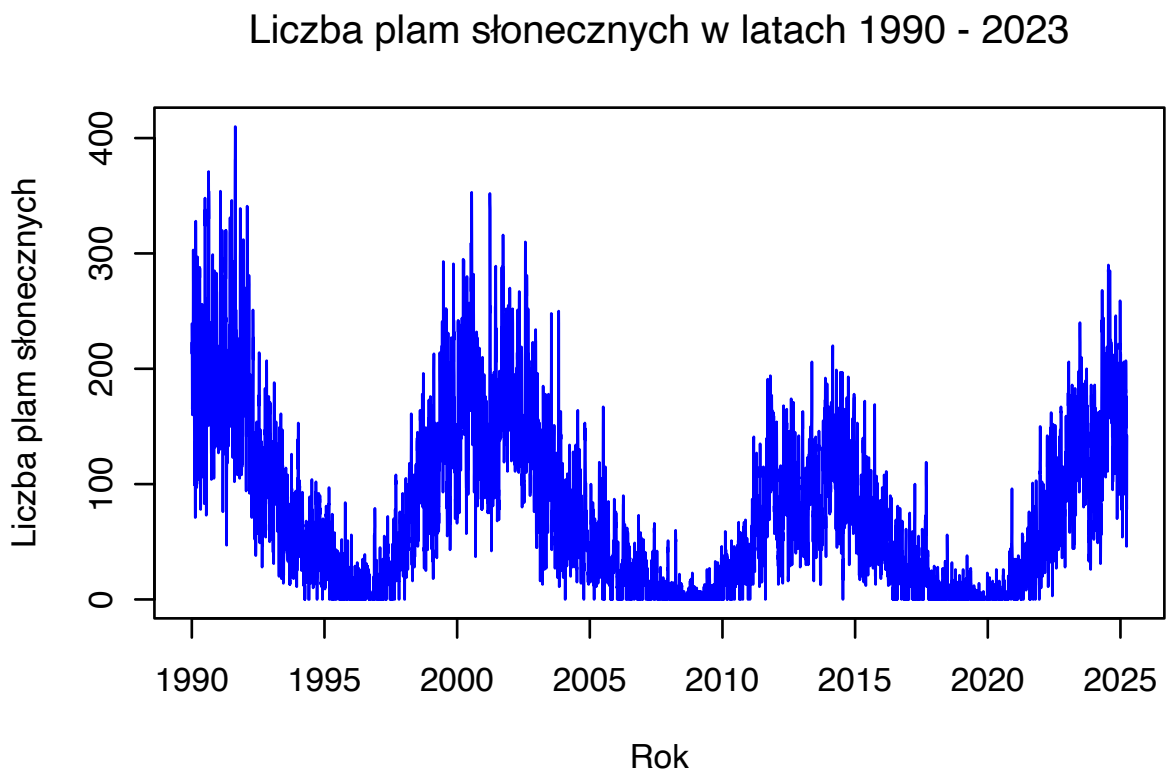
Znalezienie daty kiedy miało to miejsce:

```
max_sunspots_row <- sunspots_data[
  sunspots_data$Daily_Total_Sunspot_Number == max_sunspots,
]
max_sunspots_date <- paste(
  max_sunspots_row$Day,
  max_sunspots_row$Month,
  max_sunspots_row$Year,
  sep = "."
)
max_sunspots_date
```

```
## [1] "26.8.1870"
```

1.3. Wyświetlenie wykresu liczby plam słonecznych w funkcji czasu w latach 1990 – 2023:

```
plot(
  sunspots_data$Date_In_Fraction_Of_Year[sunspots_data$Year >= 1990],
  sunspots_data$Daily_Total_Sunspot_Number[sunspots_data$Year >= 1990],
  type = "l",
  xlab = "Rok",
  ylab = "Liczba plam słonecznych",
  main = "Liczba plam słonecznych w latach 1990 - 2023",
  col = "blue",
)
```



1.4. Uśrednienie liczby plam słonecznych w skali roku

Deklaracja funkcji do obliczania średniej kroczącej:

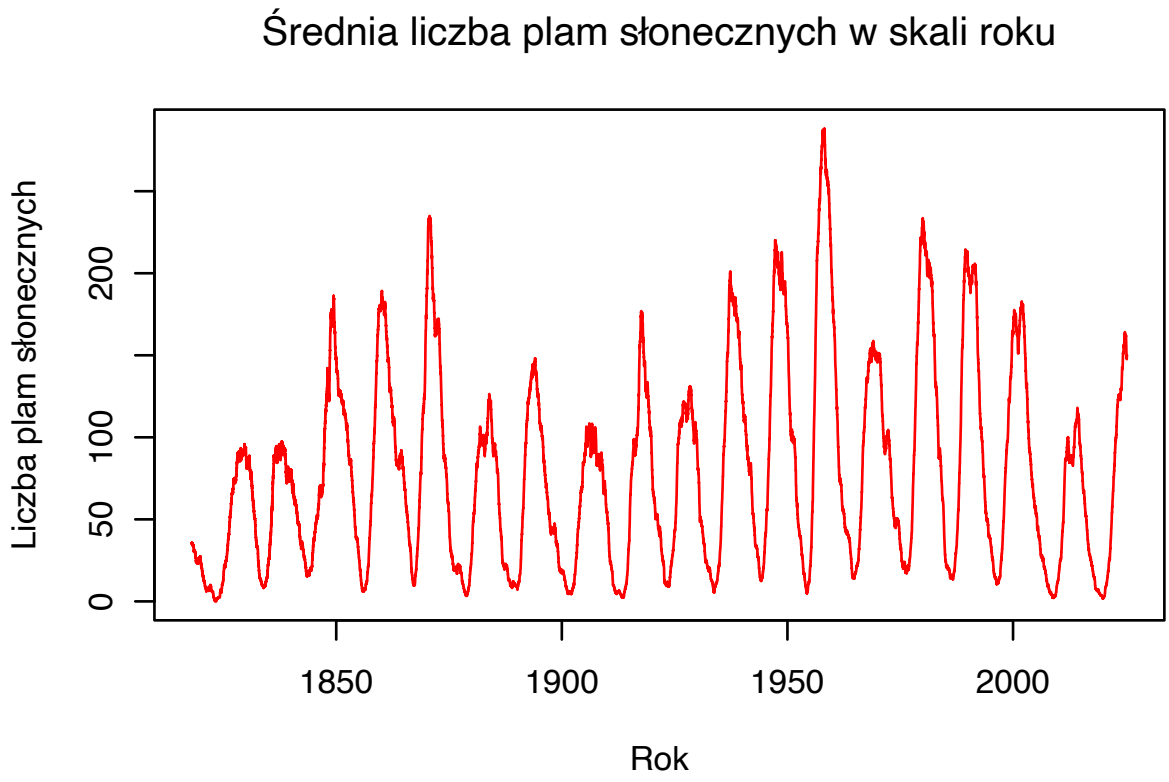
```
moving_average <- function(x, n) {
  result <- rep(NA, length(x))
  for (i in seq_along(x)) {
    start <- max(1, i - floor(n / 2))
    end <- min(length(x), i + floor(n / 2))
    result[i] <- mean(x[start:end], na.rm = TRUE)
  }
  return(result)
}
```

Obliczenie średniej kroczącej dla liczby plam słonecznych:

```
sunspots_data$Yearly_Avg_Sunspots <- moving_average(
  sunspots_data$Daily_Total_Sunspot_Number,
  n = 365
)
```

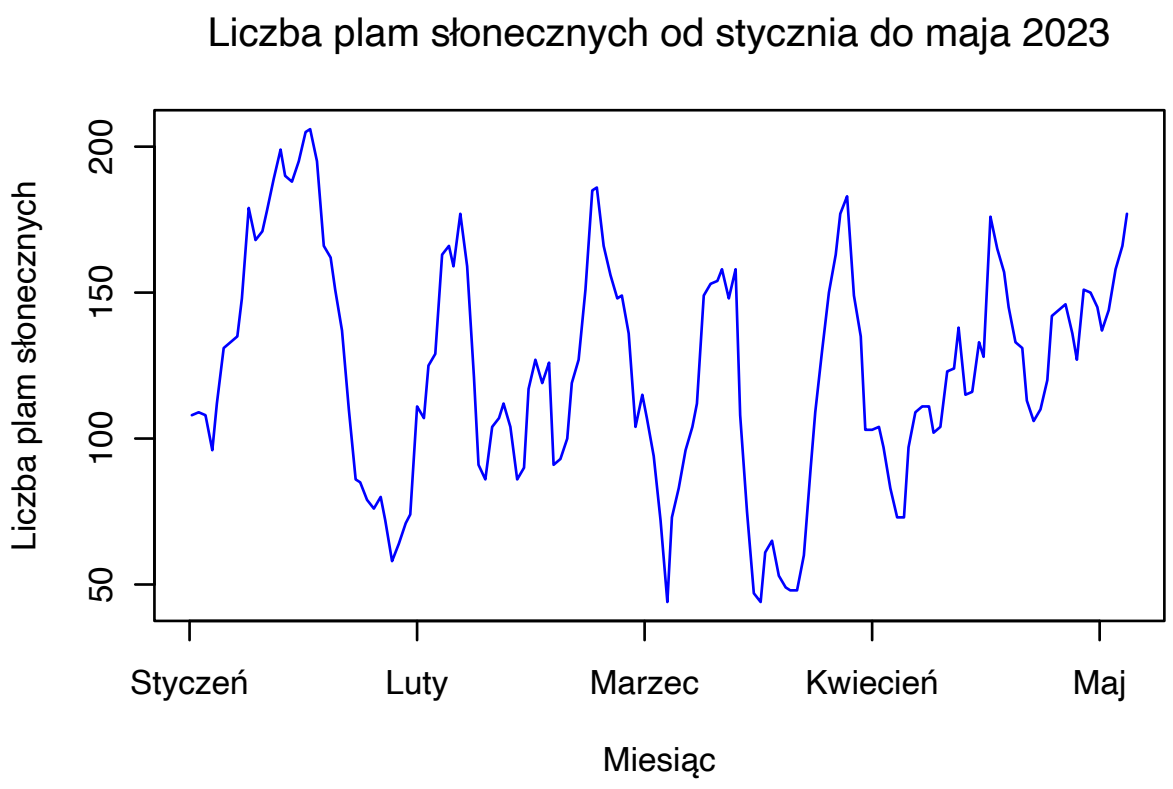
Wyświetlenie wykresu:

```
# Wykres liczby plam słonecznych uśrednionej w skali roku
plot(
  sunspots_data$Date_In_Fraction_Of_Year,
  sunspots_data$Yearly_Avg_Sunspots,
  type = "l",
  xlab = "Rok",
  ylab = "Liczba plam słonecznych",
  main = "Średnia liczba plam słonecznych w skali roku",
  col = "red"
)
```



1.5. Wykres liczby plam słonecznych od stycznia do maja 2023:

```
plot(
  sunspots_data$Date_In_Fraction_Of_Year[
    sunspots_data$Year == 2023 &
    sunspots_data$Month <= 5
  ],
  sunspots_data$Daily_Total_Sunspot_Number[
    sunspots_data$Year == 2023 &
    sunspots_data$Month <= 5
  ],
  type = "l",
  xlab = "Miesiąc",
  ylab = "Liczba plam słonecznych",
  main = "Liczba plam słonecznych od stycznia do maja 2023",
  col = "blue",
  xaxt = "n"
)
axis(
  side = 1,
  at = seq(2023.0, 2023.4, by = 0.1),
  labels = c("Styczeń", "Luty", "Marzec", "Kwiecień", "Maj")
)
```



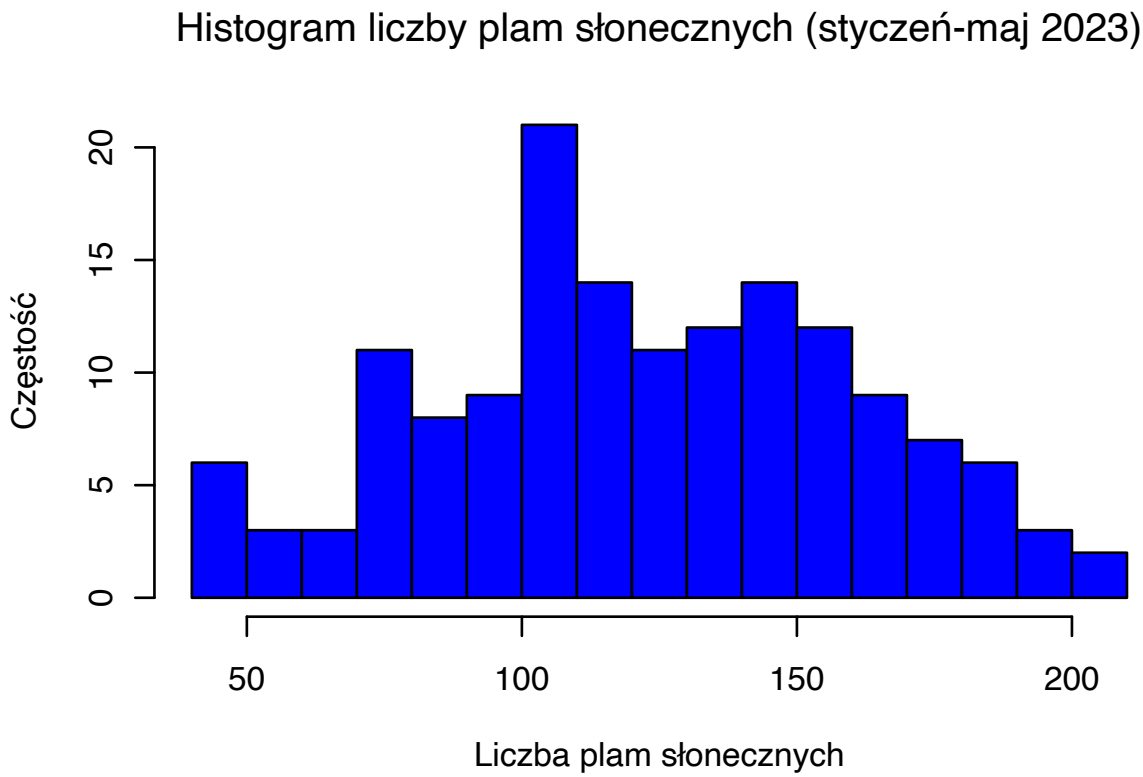
1.6. Analiza danych z wykresu z punktu 1.5.

Pobieram podzbiór danych zawierający liczbę plam słonecznych w styczniu-maju 2023:

```
subset <- sunspots_data[
  sunspots_data$Year == 2023 &
  sunspots_data$Month <= 5,
  "Daily_Total_Sunspot_Number"
]
```

Wyświetlenie histogramu:

```
hist(
  subset,
  breaks = 20,
  main = "Histogram liczby plam słonecznych (styczeń-maj 2023)",
  xlab = "Liczba plam słonecznych",
  ylab = "Częstość",
  col = "blue",
)
```



Obliczenie średniej i odchylenia standardowego:

```
mean_sunspots <- mean(subset, na.rm = TRUE)
mean_sunspots
```

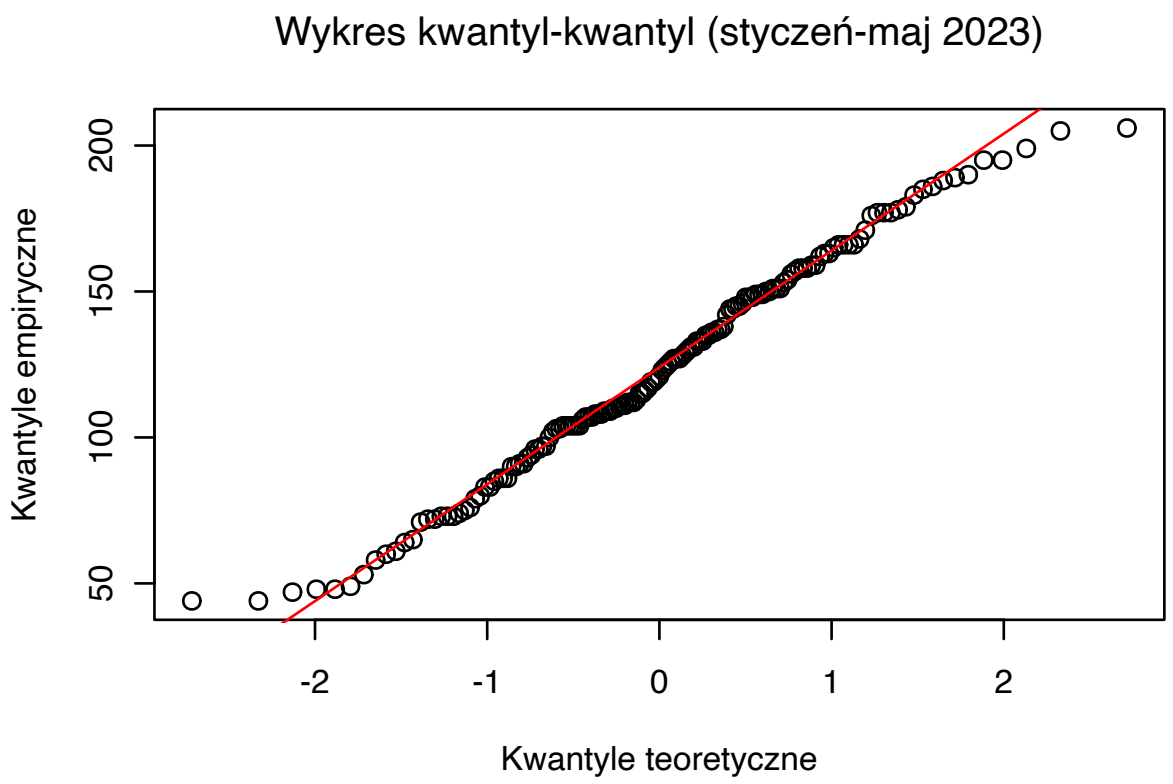
```
## [1] 123.1854
```

```
sd_sunspots <- sd(subset, na.rm = TRUE)
sd_sunspots
```

```
## [1] 38.55548
```

Wyświetlenie wykresu kwantyl-kwantyl:

```
qqnorm(subset, main = "Wykres kwantyl-kwantyl (styczeń-maj 2023)",
  xlab = "Kwantyle teoretyczne", ylab = "Kwantyle empiryczne")
qqline(subset, col = "red")
```



Zadanie 2 - Przedziały ufności i metoda bootstrap

Treść

2.1 Z biblioteki MASS otworzyć zestaw danych cabbages. Ograniczyć data frame do danych odnoszących się wyłącznie do kapusty gatunku "c39" zasadzonej w dniu "d16". Przedstawić histogram mas okazów kapusty.

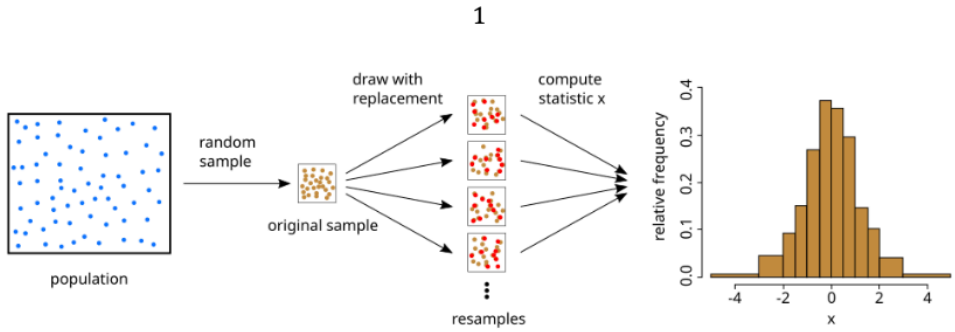


Figure 1: Źródło: https://en.wikipedia.org/wiki/Bootstrapping (statistics), autor: MM-Stat, Biggerj1.

2.2 Znaleźć średnią, odchylenie standardowe i błąd standardowy średniej dla kapusty "c39" zasadzonej dnia "d16". Określić przedziały ufności 90% i 95% (użyć rozkładu t-Studenta, ponieważ liczba próbek < 30, polecenie `qt()`). Porównać wynik obliczeń z wartością zwracaną przez funkcję `t.test`. Przedstawić masy kapust na wykresie kwantyl-kwantyl.

2.3 Niewielka ilość danych, w oparciu o które znajdujemy średnią, jak i wątpliwa normalność rozkładu skłania do znalezienia dokładniejszej metody. Użyć metody bootstrap, aby wyznaczyć przedziały ufności 90% i 95%.

2.3.1. Z naszej próbki kapust o rozmiarze 10 wylosować nową próbkę, ze zwracaniem, również o rozmiarze 10 (funkcja `sample`).

2.3.2. Znaleźć średnią dla tej próbki i zapisać ją.

2.3.3. Powtórzyć kroki 1-2 tysiąc razy.

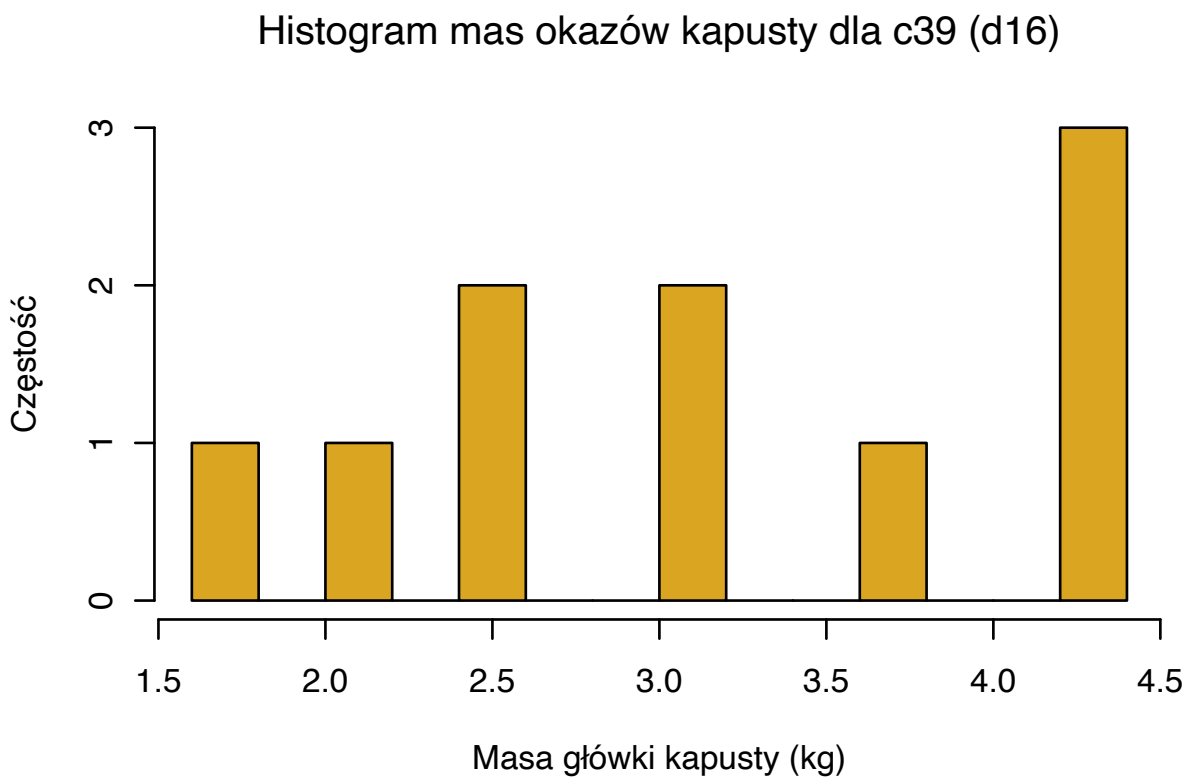
2.3.4. Dla przedziału ufności 90% będziemy używać percentyla 5% zestawu średnich znalezionych w pkcie 2. jako dolną granicę przedziału ufności oraz percentyla 95% jako górną granicę. Określić przedziały ufności 90% i 95%.

Rozwiązanie 2.1. Wczytanie danych:

```
library(MASS)
data(cabbages, package = "MASS")
filtered_cabbages <- cabbages[cabbages$Cult == "c39" & cabbages$Date == "d16", ]
```

Wyświetlenie histogramu mas okazów kapusty:

```
hist(
  filtered_cabbages$HeadWt,
  breaks = 10,
  main = "Histogram mas okazów kapusty dla c39 (d16)",
  xlab = "Masa główki kapusty (kg)",
  ylab = "Częstość",
  col = "goldenrod",
)
```



2.2. Średnia, odchylenie i błąd standardowy, przedziały ufności

Obliczenie średniej, odchylenia standardowego i błędu standardowego:

```
mean_weight <- mean(filtered_cabbages$HeadWt, na.rm = TRUE)
mean_weight
```

```
## [1] 3.18
```

```
sd_weight <- sd(filtered_cabbages$HeadWt, na.rm = TRUE)
sd_weight
```

```
## [1] 0.9566144
```

```
n <- nrow(filtered_cabbages)
se <- sd_weight / sqrt(n)
se
```

```
## [1] 0.302508
```

Wyznaczenie przedziałów ufności 90% i 95%:

```
alpha_90 <- 0.10
alpha_95 <- 0.05
t_90 <- qt(1 - alpha_90 / 2, df = n - 1)
t_95 <- qt(1 - alpha_95 / 2, df = n - 1)
ci_90 <- c(
  mean_weight - t_90 * se,
  mean_weight + t_90 * se
)
ci_90
```

```
## [1] 2.625469 3.734531
```

```
ci_95 <- c(
  mean_weight - t_95 * se,
  mean_weight + t_95 * se
)
ci_95
```

```
## [1] 2.495679 3.864321
```

Porównując z funkcją `t.test()`:

```
t_test_result_90 <- t.test(filtered_cabbages$HeadWt, conf.level = 0.90)$conf.int
t_test_result_90
```

```
## [1] 2.625469 3.734531
## attr(,"conf.level")
## [1] 0.9
```

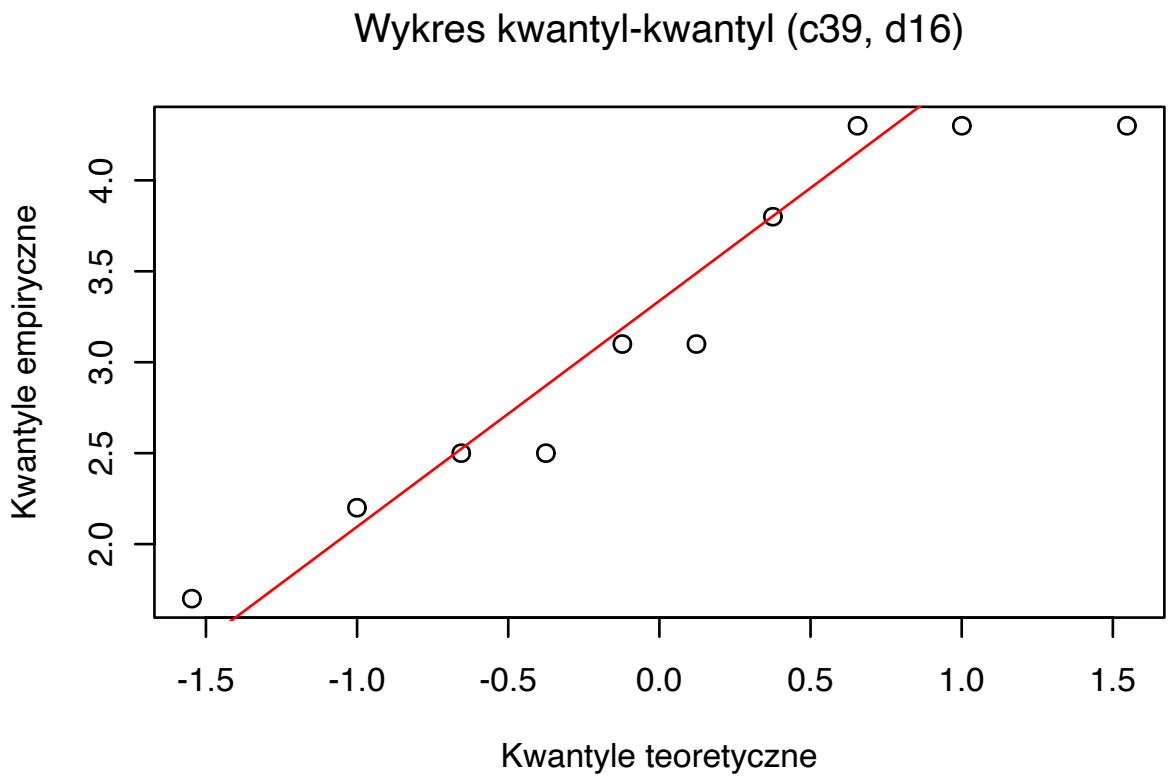
```
t_test_result_95 <- t.test(filtered_cabbages$HeadWt, conf.level = 0.95)$conf.int
t_test_result_95
```

```
## [1] 2.495679 3.864321
## attr(,"conf.level")
## [1] 0.95
```

Jak widać zwrócone wartości są takie same, jak te, które obliczyliśmy za pomocą funkcji `qt()`.

Wyświetlenie wykresu kwantyl-kwantyl:

```
qqnorm(filtered_cabbages$HeadWt, main = "Wykres kwantyl-kwantyl (c39, d16)",
  xlab = "Kwantyle teoretyczne", ylab = "Kwantyle empiryczne")
qqline(filtered_cabbages$HeadWt, col = "red")
```



2.3. Użycie metody bootstrap do wyznaczenia przedziałów ufności 90% i 95%

Ustawiam mój numer albumu jako ziarno dla powtarzalności wyników:

```
set.seed(151885)
```

Deklaracja parametrów:

```
n <- 1000
sample_size <- 10
bootstrap_means <- numeric(n)
```

Zastosowanie metody bootstrap:

```
for (i in 1:n) {
  bootstrap_sample <- sample(filtered_cabbages$HeadWt, size = sample_size, replace = TRUE)
  bootstrap_means[i] <- mean(bootstrap_sample, na.rm = TRUE)
}
```

Wyznaczenie przedziałów ufności 90% i 95%:

```
ci_90 <- quantile(bootstrap_means, probs = c(0.05, 0.95))
ci_90
```

```
##   5%  95%
## 2.72 3.68
```

```
ci_95 <- quantile(bootstrap_means, probs = c(0.025, 0.975))
ci_95
```

```
##  2.5% 97.5%
##  2.64  3.76
```

Zadanie 3 - Przedziały ufności, rozkład t-Studenta

Treść

3.1 Z biblioteki MASS otworzyć zestaw danych cabbages. Ograniczyć data frame do danych odnoszących się wyłącznie do kapusty gatunku "c39" zasadzonej w dniu "d16". Przedstawić histogram mas okazów kapusty.

3.2 Znaleźć średnią, odchylenie standardowe i błąd standardowy średniej ($= \frac{\sigma}{\sqrt{n}}$) dla kapusty "c39" zasadzonej dnia "d16".

3.3 Zakładając, że rozkład mas kapusty w całej populacji jest normalny, a odchylenie standardowe tego rozkładu jest równe błędowi standardowemu znalezionemu w poprzednim zadaniu, znaleźć przedziały ufności 90% i 95% dla średniej masy kapusty. Użyć funkcji `qnorm()`.

3.4 W rzeczywistości nie należy w takim przypadku zakładać, że rozkład jest normalny (co zresztą wyraźnie widać też na histogramie), ponieważ liczba próbek < 30. Użyć rozkładu t-Studenta (funkcja `qt()`). Znaleźć ponownie przedziały ufności 90% i 95]%. Porównać wynik z rezultatem zwracanym przez funkcję `t.test()`.

3.5 Przedstawić masy kapust na wykresie kwantyl-kwantyl.

3.6 Niewielka ilość danych, w oparciu o które znajdujemy średnią, jak i wątpliwa normalność rozkładu skłania do znalezienia dokładniejszej metody. Użyć metody bootstrap, aby wyznaczyć przedziały ufności 90% i 95%.

3.6.1. Z naszej próbki kapust o rozmiarze 10 wylosować nową próbkę, ze zwracaniem, również o rozmiarze 10 (funkcja `sample`).

3.6.2. Znaleźć średnią dla tej próbki i zapisać ją.

3.6.3. Powtórzyć kroki 1-2 tysiąc razy.

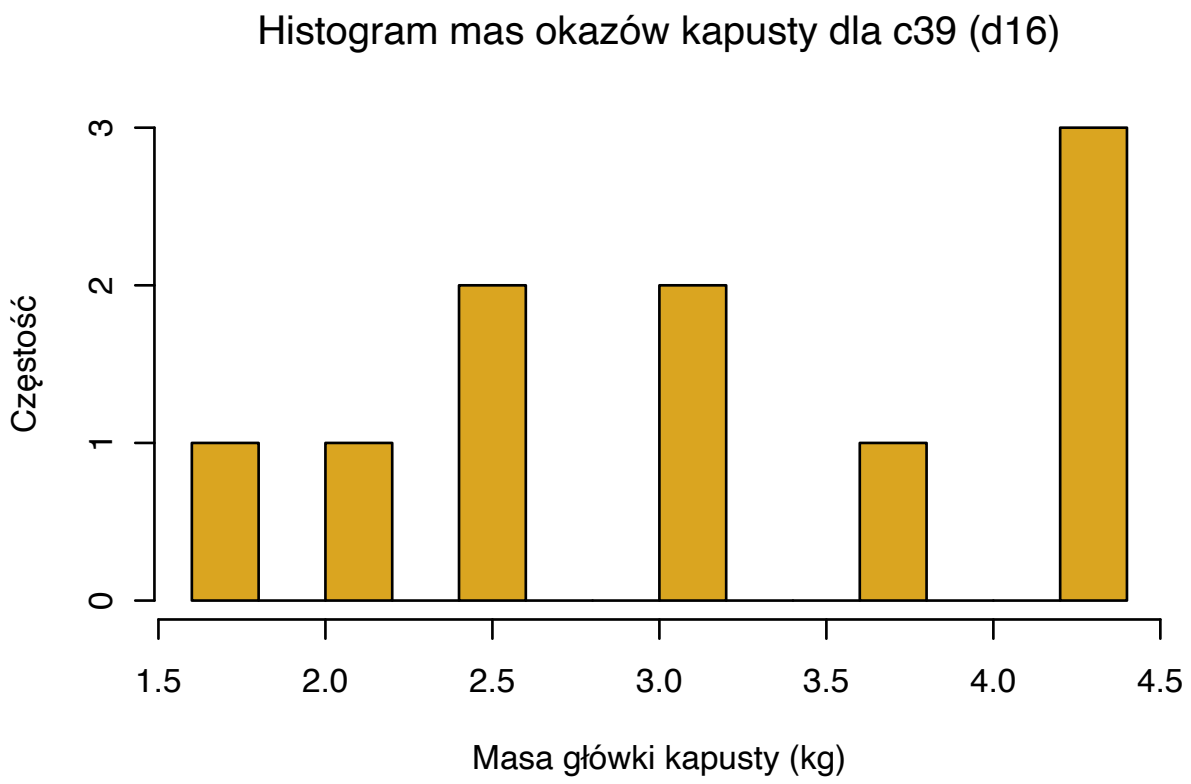
3.6.4. Dla przedziału ufności 90% będziemy używać percentyla 5% zestawu średnich znalezionych w pkcie 2. jako dolną granicę przedziału ufności oraz percentyla 95% jako górną granicę. Określić przedziały ufności 90% i 95%.

Rozwiązanie 3.1. Wczytanie danych:

```
library(MASS)
data(cabbages, package = "MASS")
filtered_cabbages <- cabbages[cabbages$Cult == "c39" & cabbages$Date == "d16", ]
```

Wyświetlenie histogramu mas okazów kapusty:

```
hist(
  filtered_cabbages$HeadWt,
  breaks = 10,
  main = "Histogram mas okazów kapusty dla c39 (d16)",
  xlab = "Masa główki kapusty (kg)",
  ylab = "Częstość",
  col = "goldenrod",
)
```



3.2. Średnia, odchylenie i błąd standardowy średniej.

Obliczenie średniej, odchylenia standardowego i błędu standardowego:

```
mean_weight <- mean(filtered_cabbages$HeadWt, na.rm = TRUE)
mean_weight
```

```
## [1] 3.18
```

```
sd_weight <- sd(filtered_cabbages$HeadWt, na.rm = TRUE)
sd_weight
```

```
## [1] 0.9566144
```

```
n <- nrow(filtered_cabbages)
se <- sd_weight / sqrt(n)
se
```

```
## [1] 0.302508
```

3.3. Przedziały ufności dla rozkładu normalnego.

Obliczenie przedziały ufności, przy założeniu, ze rozkład mas kapusty w całej populacji jest normalny, a odchylenie standardowe tego rozkładu jest równe błędowi standardowemu z poprzedniego zadania:

```
alpha_90 <- 0.10
alpha_95 <- 0.05

z_90 <- qnorm(1 - alpha_90 / 2)
z_95 <- qnorm(1 - alpha_95 / 2)

ci_90 <- c(mean_weight - z_90 * se, mean_weight + z_90 * se)
ci_90
```

```
## [1] 2.682419 3.677581
```

```
ci_95 <- c(mean_weight - z_95 * se, mean_weight + z_95 * se)
ci_95
```

```
## [1] 2.587095 3.772905
```

3.4. Przedziały ufności przy pomocy rozkładu t-Studenta.

Wyznacznie przedziałów ufności:

```
alpha_90 <- 0.10
alpha_95 <- 0.05
t_90 <- qt(1 - alpha_90 / 2, df = n - 1)
t_95 <- qt(1 - alpha_95 / 2, df = n - 1)
ci_90 <- c(
  mean_weight - t_90 * se,
  mean_weight + t_90 * se
)
ci_90
```

```
## [1] 2.625469 3.734531
```

```
ci_95 <- c(
  mean_weight - t_95 * se,
  mean_weight + t_95 * se
)
ci_95
```

```
## [1] 2.495679 3.864321
```

Porównując z funkcją `t.test()`:

```
t_test_result_90 <- t.test(filtered_cabbages$HeadWt, conf.level = 0.90)$conf.int
t_test_result_90
```

```
## [1] 2.625469 3.734531
## attr(,"conf.level")
## [1] 0.9
```

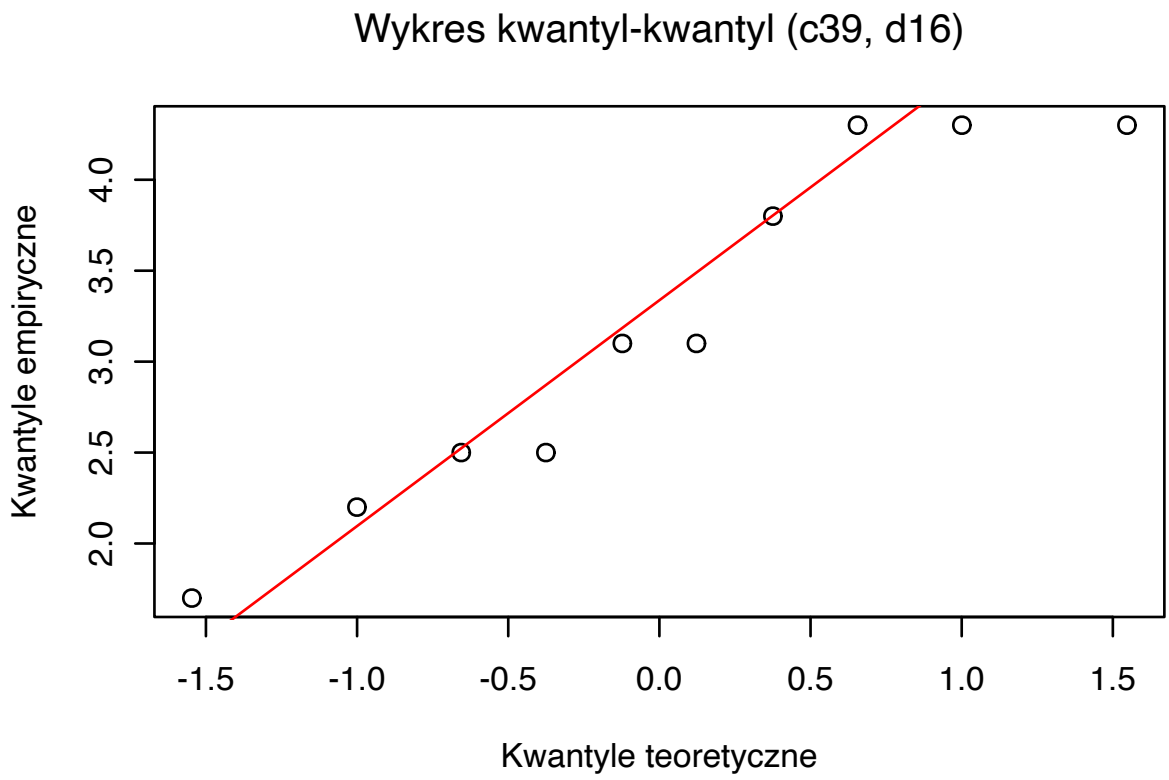
```
t_test_result_95 <- t.test(filtered_cabbages$HeadWt, conf.level = 0.95)$conf.int
t_test_result_95
```

```
## [1] 2.495679 3.864321
## attr(,"conf.level")
## [1] 0.95
```

Jak widać zwrócone wartości są takie same, jak te, które obliczyliśmy za pomocą funkcji `qt()`.

3.5. Przedstawienie mas kapust w wykresie kwantyl-kwantyl:

```
qqnorm(filtered_cabbages$HeadWt, main = "Wykres kwantyl-kwantyl (c39, d16)",
  xlab = "Kwantyle teoretyczne", ylab = "Kwantyle empiryczne")
qqline(filtered_cabbages$HeadWt, col = "red")
```



3.6. Użycie metody bootstrap do wyznaczenia przedziałów ufności 90% i 95%

Deklaracja parametrów:

```
n <- 1000
sample_size <- 10
bootstrap_means <- numeric(n)
```

Zastosowanie metody bootstrap:

```
for (i in 1:n) {
  bootstrap_sample <- sample(filtered_cabbages$HeadWt, size = sample_size, replace = TRUE)
  bootstrap_means[i] <- mean(bootstrap_sample, na.rm = TRUE)
}
```

Wyznaczenie przedziałów ufności 90% i 95%:

```
ci_90 <- quantile(bootstrap_means, probs = c(0.05, 0.95))
ci_90
```

```
##   5%  95%
## 2.68 3.62
```

```
ci_95 <- quantile(bootstrap_means, probs = c(0.025, 0.975))
ci_95
```

```
##    2.5%   97.5%
## 2.60975 3.67025
```

Zadanie 4 - Asymptotyka rozkładu t-Studenta

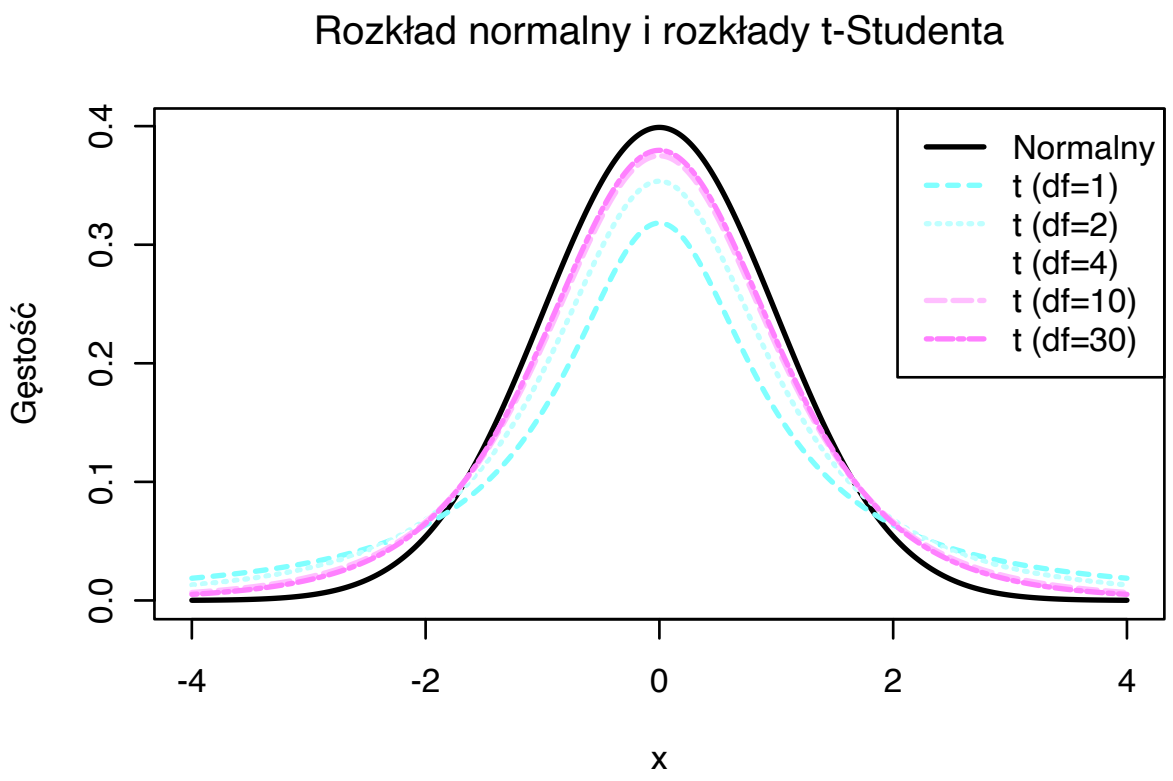
Treść Przedstawić na jednym wykresie krzywą standardowego rozkładu normalnego oraz krzywe rozkładu t-Studenta o $1, 2, 4, 10$ i 30 stopniach swobody.

Rozwiązanie Generowanie wartości dla osi `x`:

```
x <- seq(-4, 4, length.out = 1000)
```

Obliczenie wartości i wyświetlenie wykresu:

```
# Krzywa rozkladu normalnego
normal_density <- dnorm(x)
plot(x, normal_density, type = "l", col = "black", lwd = 2,
     xlab = "x", ylab = "Gęstość", main = "Rozkład normalny i rozkłady t-Studenta")
# Krzywe rozkladu t-Studenta
n <- c(1, 2, 4, 10, 30)
colors <- cm.colors(length(n))
for (i in seq_along(n)){
  lines(x, dt(x, df = i), col = colors[i], lwd = 2, lty = i + 1)
}
# Legenda
legend("topright", legend = c("Normalny", paste0("t (df=", n, ")")),
       col = c("black", colors), lty = c(1, 2:(length(n) + 1)), lwd = 2)
```



Zadanie 5 - Współczynniki Studenta-Fischera

Treść Znaleźć przedziały ufności 95% oraz 68.27% dla rozkładów t-Studenta o $1, 2, 3, ..., 30$ stopniach swobody. Porównać wynik z tablicami współczynników Studenta-Fischera:

https://www.ifiz.umk.pl/panel/wp-content/uploads/wspSF5cyf.pdf

Rozwiązanie Obliczenie przedziałów ufności:

```
df_range <- 1:30
alpha_95 <- 0.05
t_critical_95 <- qt(1 - alpha_95 / 2, df = df_range)
alpha_68 <- 1 - 0.6827
t_critical_68 <- qt(1 - alpha_68 / 2, df = df_range)
```

Wyświetlenie tabeli:

```
results <- data.frame(
  Degrees_of_Freedom = df_range,
  T_Critical_95 = t_critical_95,
  T_Critical_68 = t_critical_68
)

print(results)
```

```
##    Degrees_of_Freedom T_Critical_95 T_Critical_68
## 1                   1     12.706205      1.837409
## 2                   2      4.302653      1.321315
## 3                   3      3.182446      1.196913
## 4                   4      2.776445      1.141655
## 5                   5      2.570582      1.110533
## 6                   6      2.446912      1.090595
## 7                   7      2.364624      1.076739
## 8                   8      2.306004      1.066553
## 9                   9      2.262157      1.058752
## 10                 10      2.228139      1.052586
## 11                 11      2.200985      1.047591
## 12                 12      2.178813      1.043463
## 13                 13      2.160369      1.039993
## 14                 14      2.144787      1.037036
## 15                 15      2.131450      1.034486
## 16                 16      2.119905      1.032265
## 17                 17      2.109816      1.030313
## 18                 18      2.100922      1.028583
## 19                 19      2.093024      1.027040
## 20                 20      2.085963      1.025656
## 21                 21      2.079614      1.024406
## 22                 22      2.073873      1.023272
## 23                 23      2.068658      1.022240
## 24                 24      2.063899      1.021295
## 25                 25      2.059539      1.020427
## 26                 26      2.055529      1.019627
## 27                 27      2.051831      1.018887
## 28                 28      2.048407      1.018201
## 29                 29      2.045230      1.017564
## 30                 30      2.042272      1.016969
```

Tabela współczynników Studenta-Fishera $t_{m,p}$ $m = N - 1$ (N – liczba pomiarów w serii)

m – liczba stopni swobody, p - poziom ufności

m \ p	0,6827	0,90	0,95	0,9545	0,99	0,9973
1	1,8374	6,3138	12,706	13,968	63,657	235,78
2	1,3213	2,9200	4,3027	4,5266	9,9248	19,206
3	1,1969	2,3534	3,1824	3,3068	5,8409	9,2187
4	1,1417	2,1318	2,7764	2,8693	4,6041	6,6201
5	1,1105	2,0150	2,5706	2,6487	4,0321	5,5070
6	1,0906	1,9432	2,4469	2,5165	3,7074	4,9040
7	1,0767	1,8946	2,3646	2,4288	3,4995	4,5299
8	1,0666	1,8595	2,3060	2,3664	3,3554	4,2766
9	1,0588	1,8331	2,2622	2,3198	3,2498	4,0942
10	1,0526	1,8125	2,2281	2,2837	3,1693	3,9569
11	1,0476	1,7959	2,2010	2,2549	3,1058	3,8499
12	1,0435	1,7823	2,1788	2,2314	3,0545	3,7642
13	1,0400	1,7709	2,1604	2,2118	3,0123	3,6941
14	1,0370	1,7613	2,1448	2,1953	2,9768	3,6358
15	1,0345	1,7531	2,1314	2,1812	2,9467	3,5864
16	1,0323	1,7459	2,1199	2,1689	2,9208	3,5441
17	1,0303	1,7396	2,1098	2,1583	2,8982	3,5075
18	1,0286	1,7341	2,1009	2,1489	2,8784	3,4754
19	1,0270	1,7291	2,0930	2,1405	2,8609	3,4472
20	1,0257	1,7247	2,0860	2,1330	2,8453	3,4221
25	1,0204	1,7081	2,0595	2,1051	2,7874	3,3296
30	1,0170	1,6973	2,0423	2,0868	2,7500	3,2703

Figure 2: Fragment tabeli współczynników Studenta-Fischera. Źródło: https://www.ifiz.umk.pl/panel/wp-content/uploads/wspSF5cyf.pdf

Porównując tabelę z poprzedniej strony z tabelą współczynników Studenta-Fischera mozemy zobaczyć, ze wyliczone wartości są zbiezne z tabelą.

Zadanie 6 – Dystrybuanta empiryczna

Treść

6.1 Z biblioteki `MASS` otworzyć zestaw danych `birthwt` zawierający informacje dotyczące noworodków w Baystate Medical Center w Springfield (1986). Przedstawić histogram mas noworodków (ostatnia kolumna data frame), sprawdzić normalność rozkładu na wykresie kwantyl-kwantyl oraz znaleźć średnią masę i odchylenie standardowe.

6.2 Jaki procent badanych kobiet paliło w ciąży? Przedstawić informację o palących i niepalących na wykresie kołowym wraz z etykietami obu części wykresu (`pie()`, `labels`).

6.3 Przedstawić dystrybuantę empiryczną mas niemowląt na wykresie. Użyć funkcji `ecdf()`.

6.4 Skonstruować własną funkcję obliczającą dystrybuantę empiryczną. Przedstawić dystrybuantę mas niemowląt na wykresie i porównać wynik z poprzednim zadaniem.

6.5 Przedstawić na jednym wykresie dystrybuantę empiryczną z zadania 6.3 oraz dystrybuantę rozkładu normalnego o znalezionych w zadaniu 6.1 średniej i odchyleniu standardowym.

Rozwiązanie

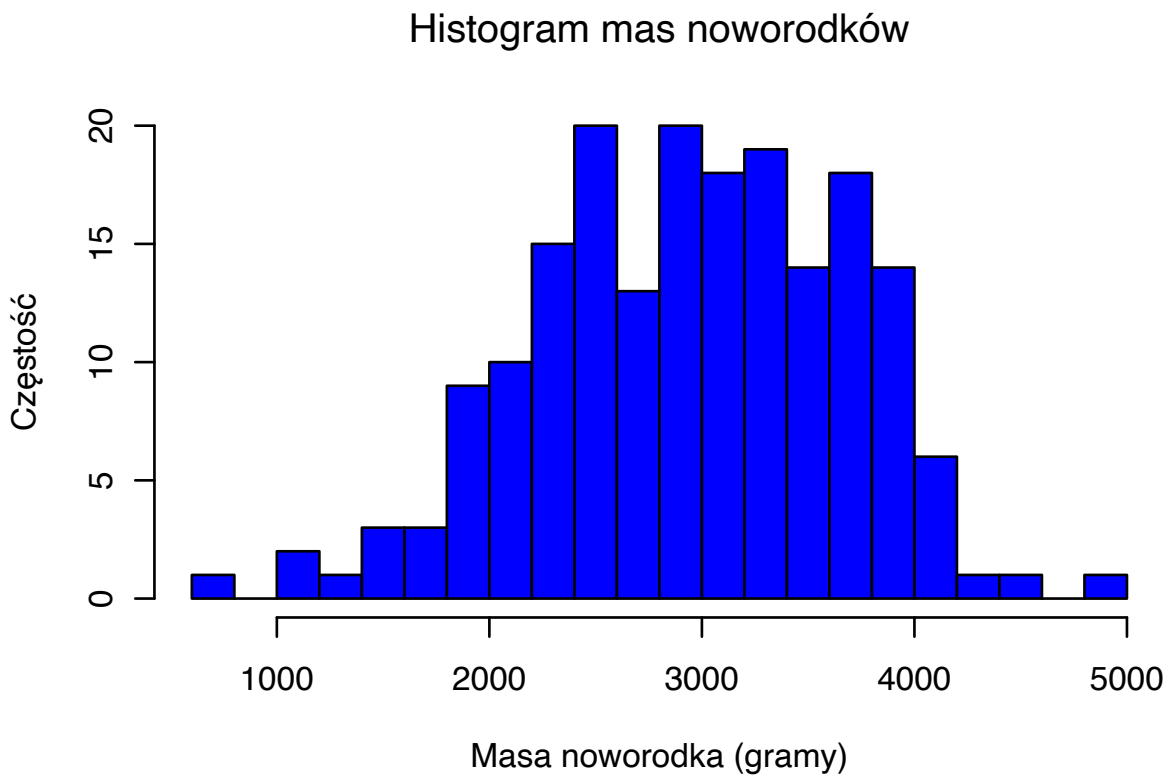
6.1 Wczytanie danych i analiza mas noworodków.

Wczytanie danych:

```
library(MASS)
data(birthwt, package = "MASS")
```

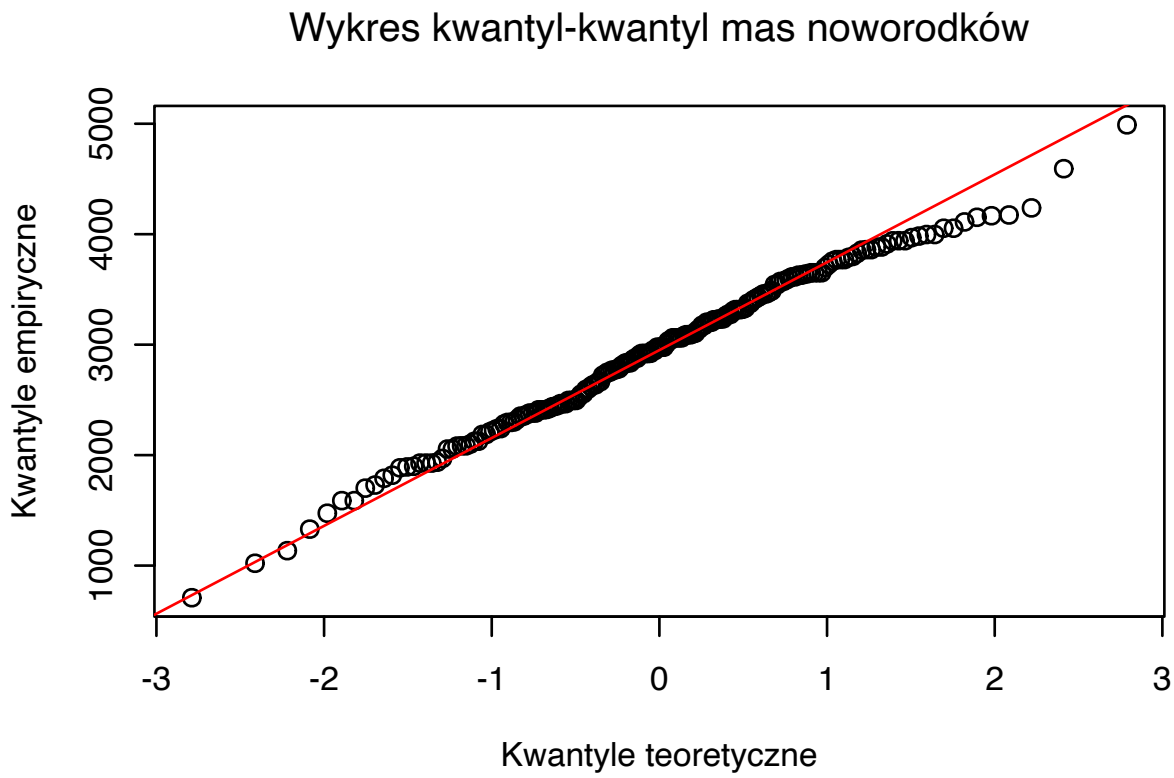
Wyświetlenie histogramu mas noworodków:

```
hist(
  birthwt$bwt,
  breaks = 20,
  main = "Histogram mas noworodków",
  xlab = "Masa noworodka (gramy)",
  ylab = "Częstość",
  col = "blue"
)
```



Sprawdzenie normalności rozkładu - wykres kwantyl-kwantyl:

```
qqnorm(birthwt$bwt, main = "Wykres kwantyl-kwantyl mas noworodków",
  xlab = "Kwantyle teoretyczne", ylab = "Kwantyle empiryczne")
qqline(birthwt$bwt, col = "red")
```



Jak widać z wykresu, rozkład mas noworodków nie jest idealnym rozkładem normalnym, ale przypomina go. Istnieją wartości odstające od prostej, ale w większości przypadków dane są zgodne z rozkładem normalnym.

Obliczenie średniej masy i odchylenia standardowego:

```
mean_bwt <- mean(birthwt$bwt)
mean_bwt
```

```
## [1] 2944.587
```

```
sd_bwt <- sd(birthwt$bwt)
sd_bwt
```

```
## [1] 729.2143
```

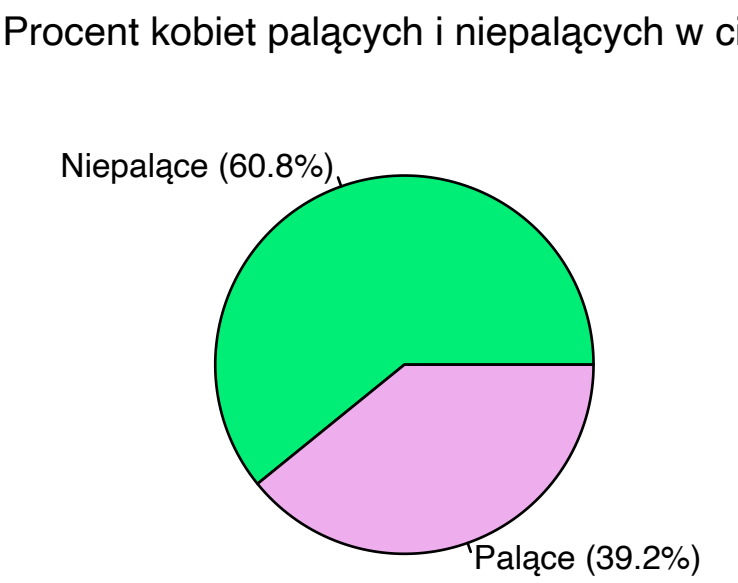
6.2. Analiza palenia w ciąży:

Obliczenie procentu kobiet palących i niepalących:

```
smoking_counts <- table(birthwt$smoke)
smoking_labels <- c("Niepalące", "Palące")
smoking_percent <- round(100 * smoking_counts / sum(smoking_counts), 1)
```

Wyświetlenie wykresu kołowego:

```
pie(
  smoking_counts,
  labels = paste(smoking_labels, " (", smoking_percent, "%)", sep = ""),
  main = "Procent kobiet palących i niepalących w ciąży",
  col = c("springgreen2", "plum2")
)
```



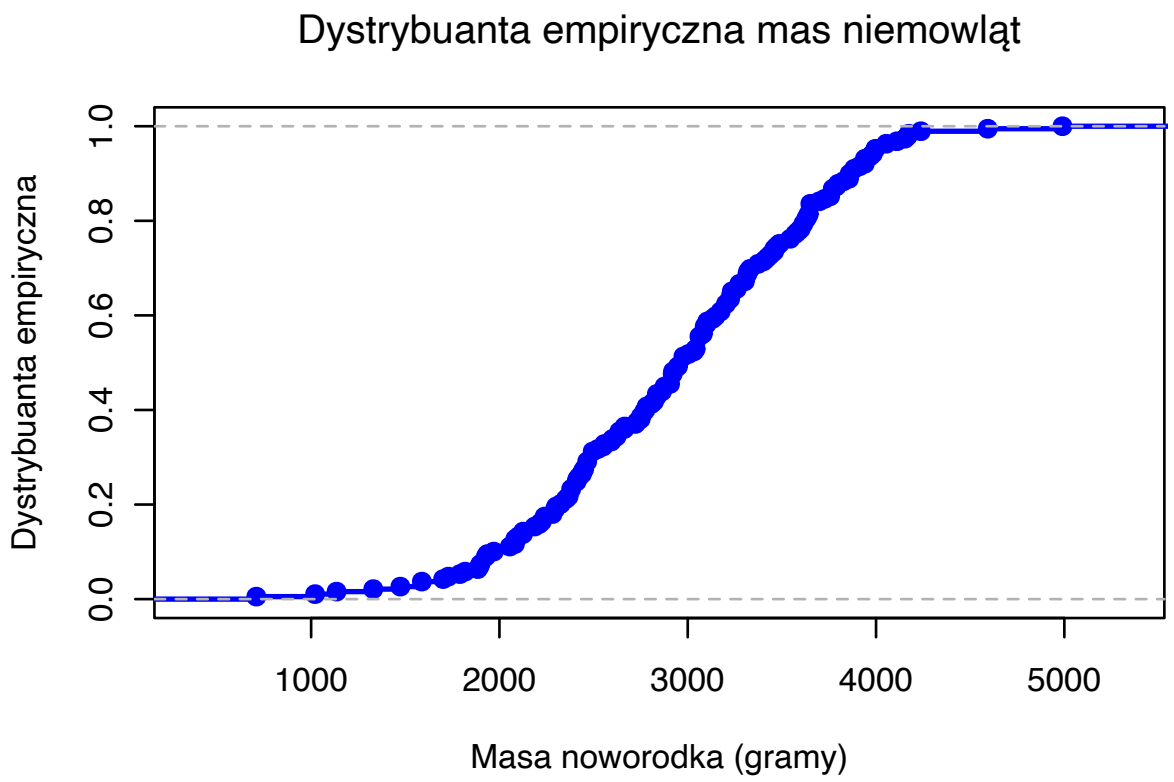
6.3. Dystrybuanta empiryczna mas niemowląt.

Obliczenie dystrybuanty empirycznej:

```
ecdf_weight <- ecdf(birthwt$bwt)
```

Wyświetlenie wykresu dystrybuanty empirycznej:

```
plot(
  ecdf_weight,
  main = "Dystrybuanta empiryczna mas niemowląt",
  xlab = "Masa noworodka (gramy)",
  ylab = "Dystrybuanta empiryczna",
  col = "blue",
  lwd = 2
)
```



6.4. Własna funkcja do obliczania dystrybuanty empirycznej.

Deklaracja funkcji:

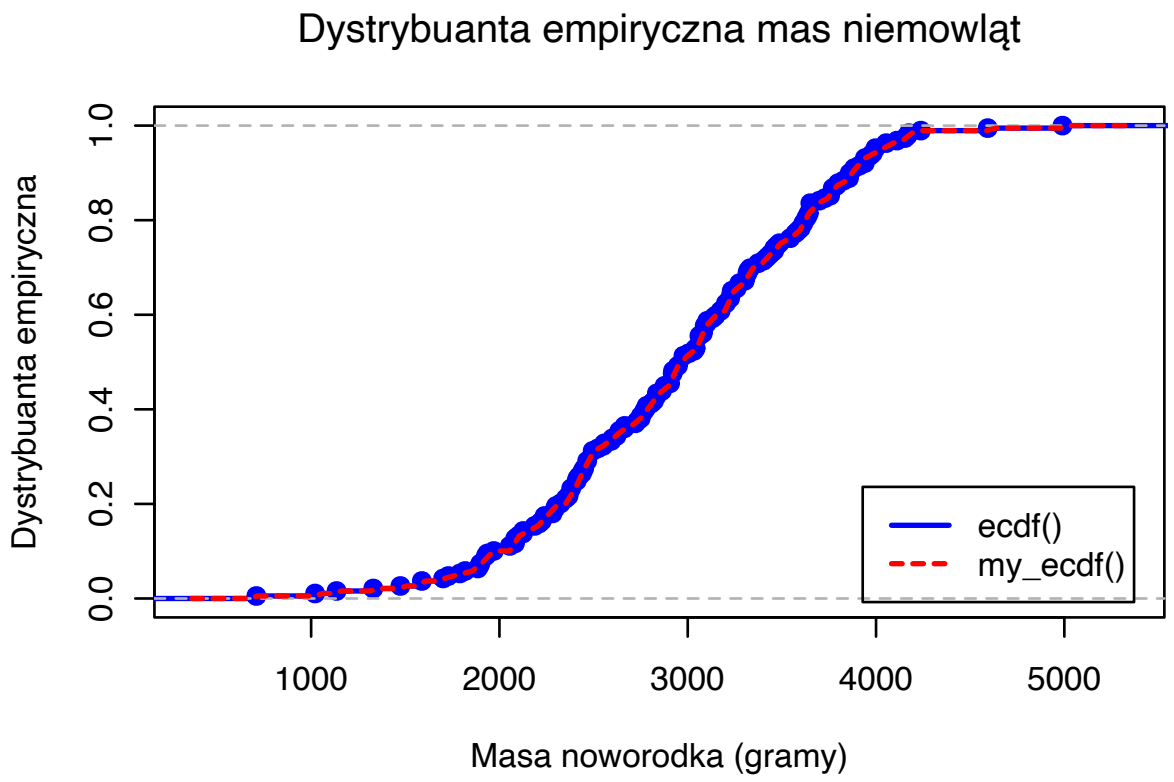
```
my_ecdf <- function(x) {
  x_sorted <- sort(x)
  n <- length(x)
  function(t) sapply(t, function(ti) sum(x_sorted <= ti) / n)
}
```

Obliczanie dystrybuanty empirycznej:

```
custom_ecdf_weight <- my_ecdf(birthwt$bwt)
```

Wyświetlenie wykresu porównawczego dystrybuanty empirycznej:

```
plot(
  ecdf_weight,
  main = "Dystrybuanta empiryczna mas niemowląt",
  xlab = "Masa noworodka (gramy)",
  ylab = "Dystrybuanta empiryczna",
  col = "blue",
  lwd = 2
)
curve(custom_ecdf_weight, add = TRUE, col = "red", lwd = 2, lty = 2)
legend(
  "bottomright",
  legend = c("ecdf()", "my_ecdf()"),
  col = c("blue", "red"),
  lty = c(1, 2),
  lwd = 2,
  inset = 0.03
)
```



Jak widać z wykresu, dystrybuanta empiryczna uzyskana za pomocą funkcji `ecdf()` jest taka sama, jak ta uzyskana za pomocą własnej funkcji `my_ecdf()`.

6.5. Porównanie dystrybuanty empirycznej z zadania 6.3 z dystrybuantą rozkładu normalnego o parametrach z zadania 6.1.

Obliczenie wartości dystrybuanty rozkładu normalnego:

```
x_vals <- seq(min(birthwt$bwt), max(birthwt$bwt), length.out = 1000)
normal_cdf <- pnorm(x_vals, mean = mean_bwt, sd = sd_bwt)
```

Wyświetlenie wykresu porównawczego:

```
plot(
  ecdf_weight,
  main = "Porównanie dystrybuanty empirycznej i dystrybuanty\nrozkładu normalnego",
  xlab = "Masa noworodka (gramy)",
  ylab = "Dystrybuanta",
  col = "blue",
  lwd = 2
)
lines(x_vals, normal_cdf, col = "red", lwd = 2, lty = 2)
legend(
  "bottomright",
  legend = c("Dystrybuanta empiryczna ", "Rozkład normalny"),
  col = c("blue", "red"),
  lty = c(1, 2),
  lwd = 2,
  inset = 0.03,
)
```