Błażej Domagała - WED

Lista 1

Zadanie 1

a) Nazwa zbioru: Wine

```
> library(datasets)
>
> path = "C:\\Users\\petitoff\\Desktop\\repos\\UO\\rok 3\\Wprowadzenie do eksploracji danych\\lista1\\zadanie2"  # używając podwójnych ukośników
> setwd(path) ## ustawienie ścieżki
>
> ## zmiana nazwy kolumn:
>
> # załadowanie danych
> # Spróbuj wczytać dane z innym separatorem
> wine <- read.csv('wine\\wine.data', header=FALSE)
```

b) **Krótki tekstowy opis zbioru**

Zbiór danych "Wine" zawiera wyniki analizy chemicznej win wyprodukowanych w określonym regionie we Włoszech przez trzech różnych producentów. Analiza chemiczna dotyczy 13 różnych składników zawartych w winach.

```
>
> ## zmiana nazwy kolumn:
>
> # załadowanie danych
> # Spróbuj wczytać dane z innym separatorem
> wine <- read.csv('wine\\wine.data', header=FALSE)
>
> # Zmień nazwy kolumn
> names(wine) <- c('Class', 'Alcohol', 'Malic acid', 'Ash', 'Alcalinity of ash', 'Magnesium', 'Total phenols', 'Flavanoids', 'Nonflavanoid phenols', 'Proanthocy
anins', 'Color intensity', 'Hue', 'OD280/OD315 of diluted wines', 'Proline')
>
```

c) Liczba obserwacji w zbiorze: 178

```
>
> ## Liczba obserwacji w zbiorze
> nrow(wine)
[1] 178
>
```

d) Liczba kolumn: 14 (13 atrybutów + 1 kolumna identyfikująca klasę)

```
> ## Liczba kolumn
> print(names(wine))
 [1] "Class"                     "Alcohol"                   "Malic acid"                "Ash"
 [5] "Alcalinity of ash"         "Magnesium"                 "Total phenols"             "Flavanoids"
 [9] "Nonflavanoid phenols"      "Proanthocyanins"           "Color intensity"           "Hue"
[13] "OD280/OD315 of diluted wines" "Proline"
>
```

e) Zmienna celu:

- Nazwa kolumny z klasą: Class
- Liczba klas: 3 (Klasy 1, 2 i 3)

```
> ## Zmienna celu
> unique(wine$Class)
[1] 1 2 3
>
```

f) Wykaz i opis cech:

1. **Class**: Zmienna kategoryczna. Klasa wina.
2. **Alcohol**: Zmienna ilościowa. Zawartość alkoholu.
3. **Malic acid**: Zmienna ilościowa. Zawartość kwasu jabłkowego.
4. **Ash**: Zmienna ilościowa. Zawartość popiołu.
5. **Alcalinity of ash**: Zmienna ilościowa. Zasadowość popiołu.
6. **Magnesium**: Zmienna ilościowa. Zawartość magnezu.
7. **Total phenols**: Zmienna ilościowa. Całkowita zawartość fenoli.
8. **Flavanoids**: Zmienna ilościowa. Zawartość flawonoidów.
9. **Nonflavanoid phenols**: Zmienna ilościowa. Zawartość fenoli nieflawonoidowych.
10. **Proanthocyanins**: Zmienna ilościowa. Zawartość proantocyjanidyn.
11. **Color intensity**: Zmienna ilościowa. Intensywność koloru.
12. **Hue**: Zmienna ilościowa. Odcień.
13. **OD280/OD315 of diluted wines**: Zmienna ilościowa. Stosunek absorbancji przy 280 nm do 315 nm w rozcieńczonych winach.
14. **Proline**: Zmienna ilościowa. Zawartość prolina.

```
> summary(wine)
     Class            Alcohol        Malic acid         Ash        Alcalinity of ash   Magnesium      Total phenols    Flavanoids    Nonflavanoid phenols
 Min.   :1.000   Min.   :11.03   Min.   :0.740   Min.   :1.360   Min.   :10.60     Min.   : 70.00   Min.   :0.980   Min.   :0.340   Min.   :0.1300
 1st Qu.:1.000   1st Qu.:12.36   1st Qu.:1.603   1st Qu.:2.210   1st Qu.:17.20     1st Qu.: 88.00   1st Qu.:1.742   1st Qu.:1.205   1st Qu.:0.2700
 Median :2.000   Median :13.05   Median :1.865   Median :2.360   Median :19.50     Median : 98.00   Median :2.355   Median :2.135   Median :0.3400
 Mean   :1.938   Mean   :13.00   Mean   :2.336   Mean   :2.367   Mean   :19.49     Mean   : 99.74   Mean   :2.295   Mean   :2.029   Mean   :0.3619
 3rd Qu.:3.000   3rd Qu.:13.68   3rd Qu.:3.083   3rd Qu.:2.558   3rd Qu.:21.50     3rd Qu.:107.00   3rd Qu.:2.800   3rd Qu.:2.875   3rd Qu.:0.4375
 Max.   :3.000   Max.   :14.83   Max.   :5.800   Max.   :3.230   Max.   :30.00     Max.   :162.00   Max.   :3.880   Max.   :5.080   Max.   :0.6600
 Proanthocyanins Color intensity      Hue        OD280/OD315 of diluted wines    Proline
 Min.   :0.410   Min.   : 1.280   Min.   :0.4800   Min.   :1.270               Min.   : 278.0
 1st Qu.:1.250   1st Qu.: 3.220   1st Qu.:0.7825   1st Qu.:1.938               1st Qu.: 500.5
 Median :1.555   Median : 4.690   Median :0.9650   Median :2.780               Median : 673.5
 Mean   :1.591   Mean   : 5.058   Mean   :0.9574   Mean   :2.612               Mean   : 746.9
 3rd Qu.:1.950   3rd Qu.: 6.200   3rd Qu.:1.1200   3rd Qu.:3.170               3rd Qu.: 985.0
 Max.   :3.580   Max.   :13.000   Max.   :1.7100   Max.   :4.000               Max.   :1680.0
```

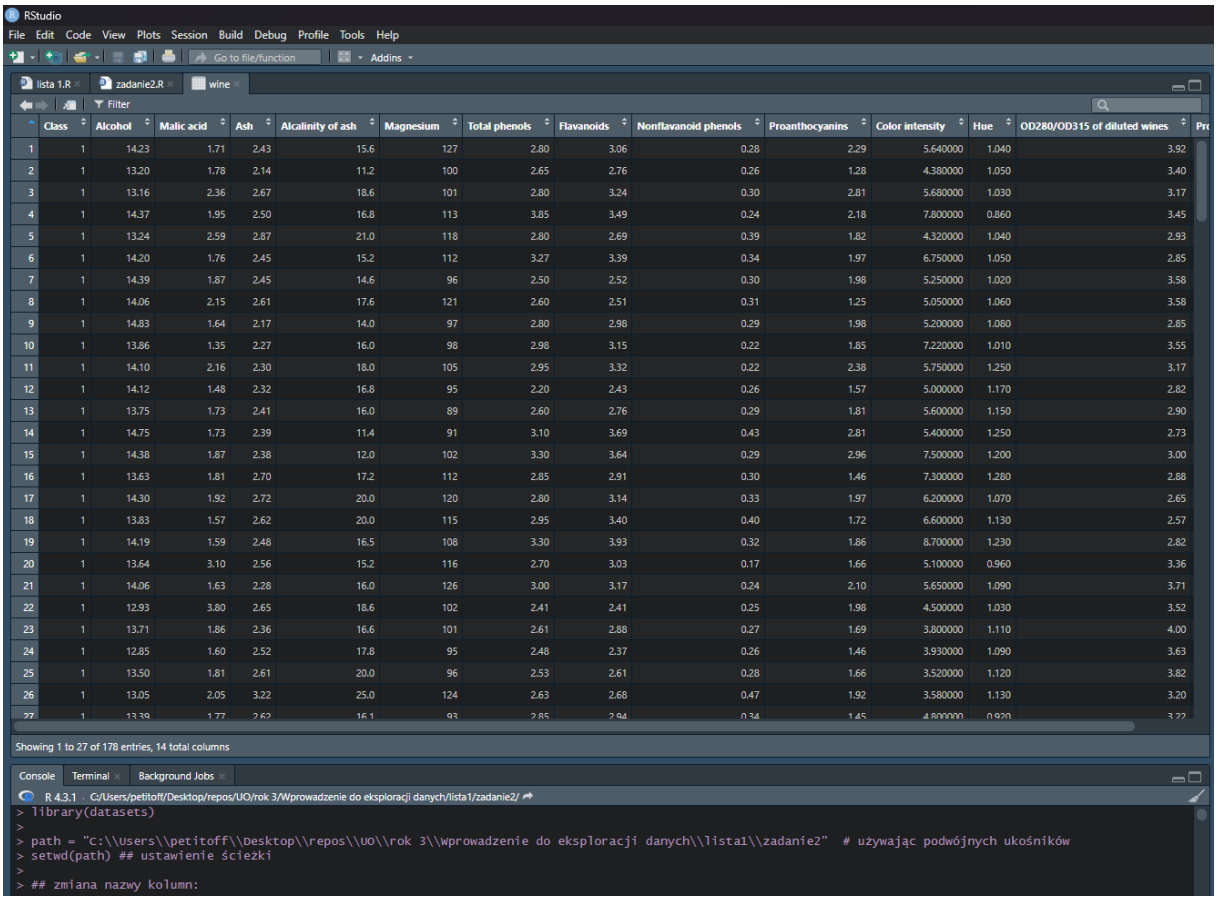
Zadanie 2

a) Zmiana nazw kolumn (pierwszą kolumnę – zmienną celu – proszę nazwać „Class”; nazwy pozostałych kolumn – atrybutów – są podane w pliku „wine.names”)

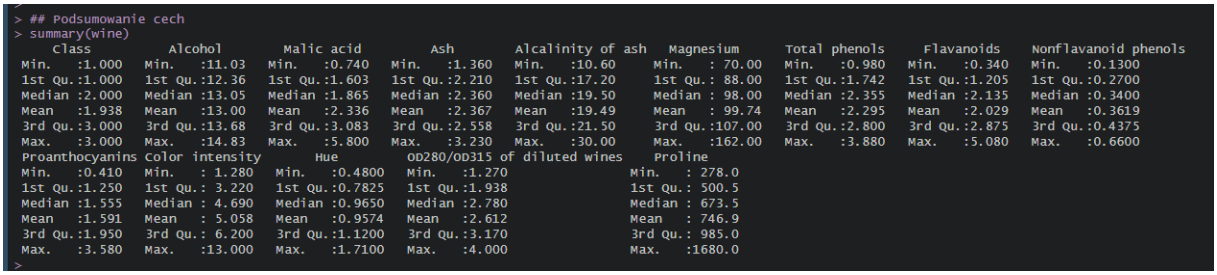
```
R 4.3.1 · C:/Users/petitoff/Desktop/repos/UO/rok 3/Wprowadzenie do eksploracji danych/lista1/zadanie2/
> library(datasets)
>
> path = "C:\\Users\\petitoff\\Desktop\\repos\\UO\\rok 3\\Wprowadzenie do eksploracji danych\\lista1\\zadanie2"  # używając podwójnych ukośników
> setwd(path) ## ustawienie ścieżki
>
> ## zmiana nazwy kolumn:
>
> # Załadowanie danych
> wine <- read.csv('wine\\wine.data', header=FALSE)
>
> # Zmień nazwy kolumn
> names(wine) <- c('Class', 'Alcohol', 'Malic acid', 'Ash', 'Alcalinity of ash', 'Magnesium', 'Total phenols', 'Flavanoids', 'Nonflavanoid phenols', 'Proanthocy
anins', 'Color intensity', 'Hue', 'OD280/OD315 of diluted wines', 'Proline')
>
```

b) Polecenie View (fragment print-screena z tabelką)

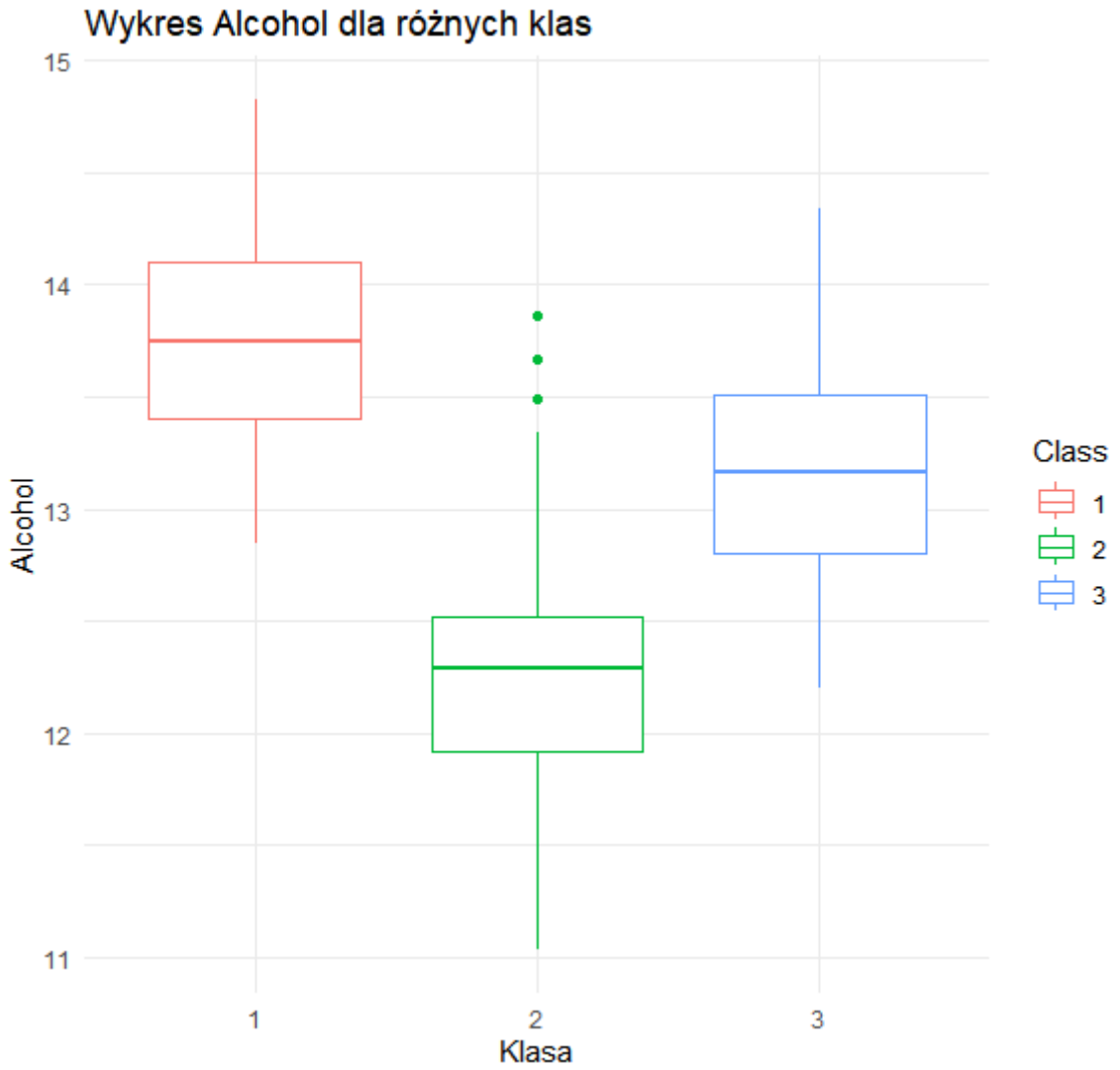
```
> ## ====================
> View(wine)
>
```



c) Podsumowanie cech (summary)

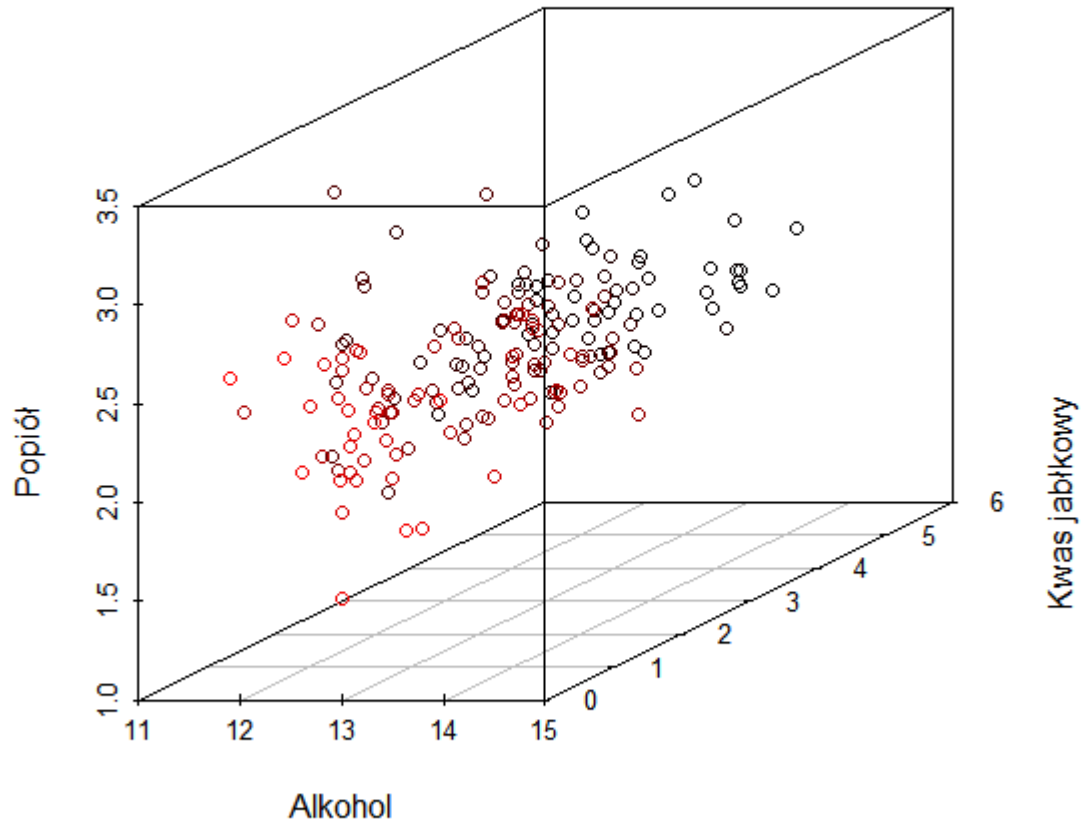


d) Wykres 2D ilustrujący wybraną cechę dla różnych klas




```
>
> ## Wykres 2D:
> data(wine, package = "datasets")
Komunikat ostrzegawczy:
W poleceniu 'data(wine, package = "datasets")':
  zbiór danych 'wine' nie został znaleziony
>
> # załadowanie biblioteki do tworzenia wykresów
> library(ggplot2)
>
> # Wybór cechy do przedstawienia na wykresie - np. "Alcohol"
> feature <- "Alcohol"
>
> # Konwersja zmiennej 'Class' na faktor
> wine$Class <- as.factor(wine$Class)
>
> # Stworzenie wykresu 2D z użyciem ggplot2
> p <- ggplot(wine, aes(x = Class, y = !!sym(feature), color = Class)) +
+   geom_boxplot() +
+   labs(title = paste("Wykres", feature, "dla różnych klas"),
+        x = "Klasa",
+        y = feature) +
+   theme_minimal()
>
> # Wyświetlenie wykresu
> print(p)
>
> ## Wykres 3D:
> library(scatterplot3d)
>
> # Stwórz wykres 3D dla wybranych cech
> scatterplot3d(wine$Alcohol, wine$`Malic acid`, wine$Ash,
+               xlab="Alkohol", ylab="Kwas jabłkowy", zlab="Popiół",
+               highlight.3d=TRUE, angle=30)
> # Zapis do pliku
> ggsave("C:\\Users\\petitoff\\Desktop\\repos\\UO\\rok 3\\Wprowadzenie do eksploracji danych\\lista1\\zadanie2\\wykres.png", plot = p, width = 10, height = 6, d
pi = 300)
>
```

e) Wykres 3D dla trzech wybranych cech (bez klasy)




```
> # Stwórz wykres 3D dla wybranych cech
> scatterplot3d(wine$Alcohol, wine$`Malic acid`, wine$Ash,
+               xlab="Alkohol", ylab="Kwas jabłkowy", zlab="Popiół",
+               highlight.3d=TRUE, angle=30)
>
```

Lista 2

a) Dla wybranej cechy wyświetlić wybrane wartości stosując: zakres wartości, sekwencję indeksów (np. co dziesiąty indeks), indeksy ujemne, warunki logiczne.

Kod:

```
path = "C:\\Users\\petitoff\\Desktop\\repos\\UO\\rok 3\\Wprowadzenie do eksploracji danych\\lista2"

setwd(path) ## ustawienie ścieżki


# Załadowanie danych

wine <- read.csv('wine\\wine.data', header = FALSE)


# Zmień nazwy kolumn

names(wine) <-

 c(

  'Class',

  'Alcohol',

  'Malic acid',

  'Ash',

  'Alcalinity of ash',

  'Magnesium',

  'Total phenols',

  'Flavanoids',

  'Nonflavanoid phenols',

  'Proanthocyanins',

  'Color intensity',

  'Hue',

  'OD280/OD315 of diluted wines',

  'Proline'

 )

# a)	Dla wybranej cechy wyświetlić wybrane wartości stosując: zakres wartości, sekwencję indeksów (np. co dziesiąty indeks), indeksy ujemne, warunki logiczne.

# Zakres wartości
```

```
wine$Alcohol[5:15]


# Sekwencja indeksów

wine$Alcohol[seq(1, nrow(wine), 10)]


# Indeksy ujemne

wine$Alcohol[-(1:5)]


# Warunki logiczne

wine$Alcohol[wine$Alcohol > 14]
```

```
> path = "C:\\Users\\petitoff\\Desktop\\repos\\UO\\rok 3\\Wprowadzenie do eksploracji danych\\lista2"
> setwd(path) ## ustawienie ścieżki
>
> # Załadowanie danych
> wine <- read.csv('wine\\wine.data', header=FALSE)
>
> # Zmień nazwy kolumn
> names(wine) <- c('Class', 'Alcohol', 'Malic acid', 'Ash', 'Alcalinity of ash', 'Magnesium', 'Total phenols', 'Flavanoi
ds', 'Nonflavanoid phenols', 'Proanthocyanins', 'Color intensity', 'Hue', 'OD280/OD315 of diluted wines', 'Proline')
>
> # head(wine)
>
> # a)  Dla wybranej cechy wyświetlić wybrane wartości stosując: zakres wartości, sekwencję indeksów (np. co dziesiąty i
ndeks), indeksy ujemne, warunki logiczne.
> # Zakres wartości
> wine$Alcohol[5:15]
 [1] 13.24 14.20 14.39 14.06 14.83 13.86 14.10 14.12 13.75 14.75 14.38
>
> # Sekwencja indeksów
> wine$Alcohol[seq(1, nrow(wine), 10)]
 [1] 14.23 14.10 14.06 13.73 13.56 13.05 12.33 12.29 12.00 12.08 12.08 11.46 11.45 12.86 12.93 13.50 12.36 12.20
>
> # Indeksy ujemne
> wine$Alcohol[-(1:5)]
  [1] 14.20 14.39 14.06 14.83 13.86 14.10 14.12 13.75 14.75 14.38 13.63 14.30 13.83 14.19 13.64 14.06 12.93 13.71
 [19] 12.85 13.50 13.05 13.39 13.30 13.87 14.02 13.73 13.58 13.68 13.76 13.51 13.48 13.28 13.05 13.07 14.22 13.56
 [37] 13.41 13.88 13.24 13.05 14.21 14.38 13.90 14.10 13.94 13.05 13.83 13.82 13.77 13.74 13.56 14.22 13.29 13.72
 [55] 12.37 12.33 12.64 13.67 12.37 12.17 12.37 13.11 12.37 13.34 12.21 12.29 13.86 13.49 12.99 11.96 11.66 13.03
 [73] 11.84 12.33 12.70 12.00 12.72 12.08 13.05 11.84 12.67 12.16 11.65 11.64 12.08 12.08 12.00 12.69 12.29 11.62
 [91] 12.47 11.81 12.29 12.37 12.29 12.08 12.60 12.34 11.82 12.51 12.42 12.25 12.72 12.22 11.61 11.46 12.52 11.76
[109] 11.41 12.08 11.03 11.82 12.42 12.77 12.00 11.45 11.56 12.42 13.05 11.87 12.07 12.43 11.79 12.37 12.04 12.86
[127] 12.88 12.81 12.70 12.51 12.60 12.25 12.53 13.49 12.84 12.93 13.36 13.52 13.62 12.25 13.16 13.88 12.87 13.32
[145] 13.08 13.50 12.79 13.11 13.23 12.58 13.17 13.84 12.45 14.34 13.48 12.36 13.69 12.85 12.96 13.78 13.73 13.45
[163] 12.82 13.58 13.40 12.20 12.77 14.16 13.71 13.40 13.27 13.17 14.13
>
> # Warunki logiczne
> wine$Alcohol[wine$Alcohol > 14]
 [1] 14.23 14.37 14.20 14.39 14.06 14.83 14.10 14.12 14.75 14.38 14.30 14.19 14.06 14.02 14.22 14.21 14.38 14.10
[19] 14.22 14.34 14.16 14.13
>
```

b) Wyświetlić wybrane wiersze i kolumny z tabeli.

```
# b) Wybranie wierszy i kolumn:

selected_rows_columns <- wine[1:10, c("Alcohol", "Malic acid")]

print(selected_rows_columns)


# indeksy
```

wine[5:15,]

```
> # b) Wybranie wierszy i kolumn:
> selected_rows_columns <- wine[1:10, c("Alcohol", "Malic acid")]
> print(selected_rows_columns)
   Alcohol Malic acid
1    14.23       1.71
2    13.20       1.78
3    13.16       2.36
4    14.37       1.95
5    13.24       2.59
6    14.20       1.76
7    14.39       1.87
8    14.06       2.15
9    14.83       1.64
10   13.86       1.35
>
> # indeksy
> wine[5:15, ]
   Class Alcohol Malic acid  Ash Alcalinity of ash Magnesium Total phenols Flavanoids Nonflavanoid phenols
5      1   13.24       2.59 2.87              21.0       118          2.80       2.69                 0.39
6      1   14.20       1.76 2.45              15.2       112          3.27       3.39                 0.34
7      1   14.39       1.87 2.45              14.6        96          2.50       2.52                 0.30
8      1   14.06       2.15 2.61              17.6       121          2.60       2.51                 0.31
9      1   14.83       1.64 2.17              14.0        97          2.80       2.98                 0.29
10     1   13.86       1.35 2.27              16.0        98          2.98       3.15                 0.22
11     1   14.10       2.16 2.30              18.0       105          2.95       3.32                 0.22
12     1   14.12       1.48 2.32              16.8        95          2.20       2.43                 0.26
13     1   13.75       1.73 2.41              16.0        89          2.60       2.76                 0.29
14     1   14.75       1.73 2.39              11.4        91          3.10       3.69                 0.43
15     1   14.38       1.87 2.38              12.0       102          3.30       3.64                 0.29
   Proanthocyanins Color intensity  Hue OD280/OD315 of diluted wines Proline
5             1.82            4.32 1.04                         2.93     735
6             1.97            6.75 1.05                         2.85    1450
7             1.98            5.25 1.02                         3.58    1290
8             1.25            5.05 1.06                         3.58    1295
9             1.98            5.20 1.08                         2.85    1045
10            1.85            7.22 1.01                         3.55    1045
11            2.38            5.75 1.25                         3.17    1510
12            1.57            5.00 1.17                         2.82    1280
13            1.81            5.60 1.15                         2.90    1320
14            2.81            5.40 1.25                         2.73    1150
15            2.96            7.50 1.20                         3.00    1547
>
```

c) Dodać do tabeli nową kolumnę z wartościami obliczonymi na podstawie innych wybranych kolumn.

```
# c) Dodanie nowej kolumny:

# Sprawdzanie, czy kolumna istnieje i zawiera dane

if ("Total phenols" %in% names(wine) &&

    !all(is.na(wine$`Total phenols`))) {

  # Dodanie nowej kolumny

  wine$Total.phenols.squared <- wine$`Total phenols` ^ 2

  head(wine)

} else {

  cat("Column 'Total phenols' does not exist or is empty.")

}
```

```
> # c) Dodanie nowej kolumny:
> # Sprawdzanie, czy kolumna istnieje i zawiera dane
> if ("Total phenols" %in% names(wine) && !all(is.na(wine$`Total phenols`))) {
+   # Dodanie nowej kolumny
+   wine$Total.phenols.squared <- wine$`Total phenols`^2
+   head(wine)
+ } else {
+   cat("Column 'Total phenols' does not exist or is empty.")
+ }
  Class Alcohol Malic acid  Ash Alcalinity of ash Magnesium Total phenols Flavanoids Nonflavanoid phenols
1     1   14.23       1.71 2.43              15.6       127          2.80       3.06                 0.28
2     1   13.20       1.78 2.14              11.2       100          2.65       2.76                 0.26
3     1   13.16       2.36 2.67              18.6       101          2.80       3.24                 0.30
4     1   14.37       1.95 2.50              16.8       113          3.85       3.49                 0.24
5     1   13.24       2.59 2.87              21.0       118          2.80       2.69                 0.39
6     1   14.20       1.76 2.45              15.2       112          3.27       3.39                 0.34
  Proanthocyanins Color intensity  Hue OD280/OD315 of diluted wines Proline Total.phenols.squared
1            2.29           5.64 1.04                         3.92    1065               7.8400
2            1.28           4.38 1.05                         3.40    1050               7.0225
3            2.81           5.68 1.03                         3.17    1185               7.8400
4            2.18           7.80 0.86                         3.45    1480              14.8225
5            1.82           4.32 1.04                         2.93     735               7.8400
6            1.97           6.75 1.05                         2.85    1450              10.6929
>
```

d) Podać wartości podstawowych statystyk dla wybranej kolumny: zakres, średnia, mediana,

```
# d) Statystyki podstawowe dla wybranej kolumny, np. "Alcohol"
cat("Statystyki dla kolumny 'Alcohol':", "\n")
cat("Zakres: ", min(wine$Alcohol), " - ", max(wine$Alcohol), "\n")
cat("Średnia: ", mean(wine$Alcohol), "\n")
cat("Mediana: ", median(wine$Alcohol), "\n")
cat("Odchylenie standardowe: ", sd(wine$Alcohol), "\n")
cat("Kurtoza: ", moments::kurtosis(wine$Alcohol), "\n")
cat("Skośność: ", moments::skewness(wine$Alcohol), "\n")
cat("Kwantyle: ", quantile(wine$Alcohol, probs = c(0.25, 0.5, 0.75)), "\n")
```

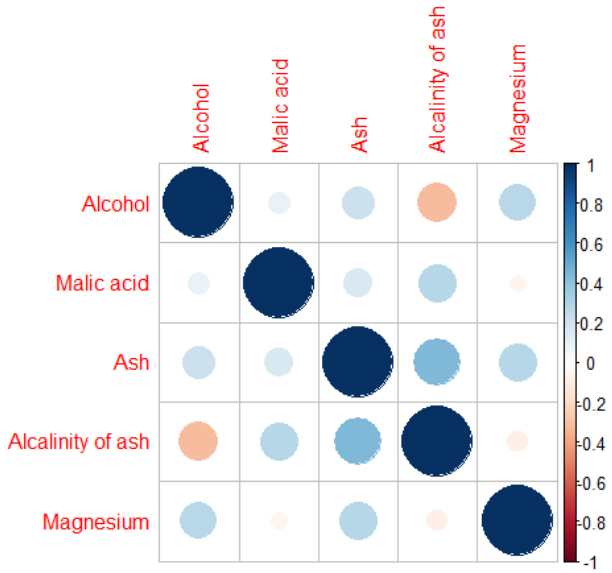
```
> # d) Statystyki podstawowe dla wybranej kolumny, np. "Alcohol"
> cat("Statystyki dla kolumny 'Alcohol':", "\n")
Statystyki dla kolumny 'Alcohol':
> cat("Zakres: ", min(wine$Alcohol), " - ", max(wine$Alcohol), "\n")
Zakres:  11.03  -  14.83
> cat("Średnia: ", mean(wine$Alcohol), "\n")
Średnia:  13.00062
> cat("Mediana: ", median(wine$Alcohol), "\n")
Mediana:  13.05
> cat("Odchylenie standardowe: ", sd(wine$Alcohol), "\n")
Odchylenie standardowe:  0.8118265
> cat("Kurtoza: ", moments::kurtosis(wine$Alcohol), "\n")
Kurtoza:  2.13774
> cat("Skośność: ", moments::skewness(wine$Alcohol), "\n")
Skośność:  -0.05104747
> cat("Kwantyle: ", quantile(wine$Alcohol, probs = c(0.25, 0.5, 0.75)), "\n")
Kwantyle:  12.3625 13.05 13.6775
>
```

e) Wyznaczyć i zilustrować na wykresie macierz korelacji dla wybranych pięciu zmiennych.

```
# e) Macierz korelacji dla wybranych pięciu zmiennych i jej wizualizacja
```

```
selected_vars <-
  wine[, c('Alcohol', 'Malic acid', 'Ash', 'Alcalinity of ash', 'Magnesium')]
cor_matrix <- cor(selected_vars)


library(corrplot)
corrplot(cor_matrix, method = "circle")
```

```
# e) Macierz korelacji dla wybranych pięciu zmiennych i jej wizualizacja
selected_vars <- wine[, c('Alcohol', 'Malic acid', 'Ash', 'Alcalinity of ash', 'Magnesium')]
cor_matrix <- cor(selected_vars)
```



f) Wydrukować histogramy dla trzech różnych zmiennych, przedyskutować wyniki.

```
# f) Histogramy dla trzech różnych zmiennych
par(mfrow = c(1, 3))  # ustawienie layoutu na 1 wiersz i 3 kolumny
hist(wine$Alcohol,
    main = 'Alcohol',
    xlab = '',
    col = 'skyblue')
hist(
  wine$`Malic acid`,
  main = 'Malic acid',
  xlab = '',
  col = 'skyblue'
```

)

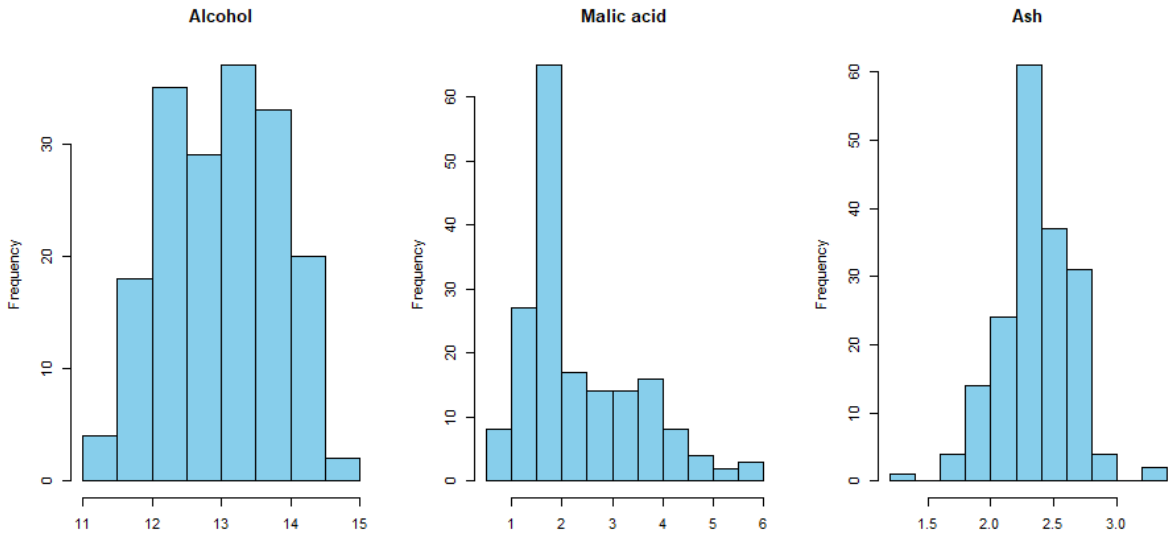
hist(wine$Ash,

main = 'Ash',

xlab = '',

col = 'skyblue')

```
61  # f) Histogramy dla trzech różnych zmiennych
62  par(mfrow=c(1,3))  # ustawienie layoutu na 1 wiersz i 3 kolumny
63  hist(wine$Alcohol, main='Alcohol', xlab='', col='skyblue')
64  hist(wine$`Malic acid`, main='Malic acid', xlab='', col='skyblue')
65  hist(wine$Ash, main='Ash', xlab='', col='skyblue')
```



Lista 3

a) Usuń kolumny z wartościami nominalnymi (identyfikatory, itp.) – jeżeli są, inne niż zmienna celu.

Zakładając, że zmienna celu to V1, a pozostałe kolumny to wartości liczbowe nie ma kolumn z wartościami nominalnymi do usunięcia.

Jeśli jednak byłyby takie kolumny, można by je usunąć za pomocą polecenia subset.

Przykład: Jeśli kolumna V2 była by zmienną nominalną, można by ją usunąć następująco:

```
wine <- subset(wine, select = -V2) # Usuń kolumnę V2 z ramki danych wine

View(wine) # Wyświetl dane po usunięciu kolumny V2
```

b) Zmień nazwy kolumn na nazwy w języku polskim. Nowe nazwy powinny być: krótkie, znaczące, bez polskich znaków i spacji. Wyświetl dane poleceniem View.

```
library(datasets)

path = "C:\\Users\\petitoff\\Desktop\\repos\\UO\\rok 3\\Wprowadzenie do eksploracji danych\\lista3\\"  # używając podwójnych ukośników

setwd(path) ## ustawienie ścieżki

# Załadowanie danych

wine <- read.csv('wine\\wine.data', header = FALSE)

# Oto kod zmieniający nazwy kolumn na nazwy w języku polskim:

nowe_nazwy <- c(
  "Klasa",
  'Alkohol',
  'Kwas jabłkowy',
  'Popiół',
  'Alkalność popiołu',
```

'Magnez',

'Całkowite fenole',

'Flawonoidy',

'Fenole nietrwałe',

'Proantocyjanidy',

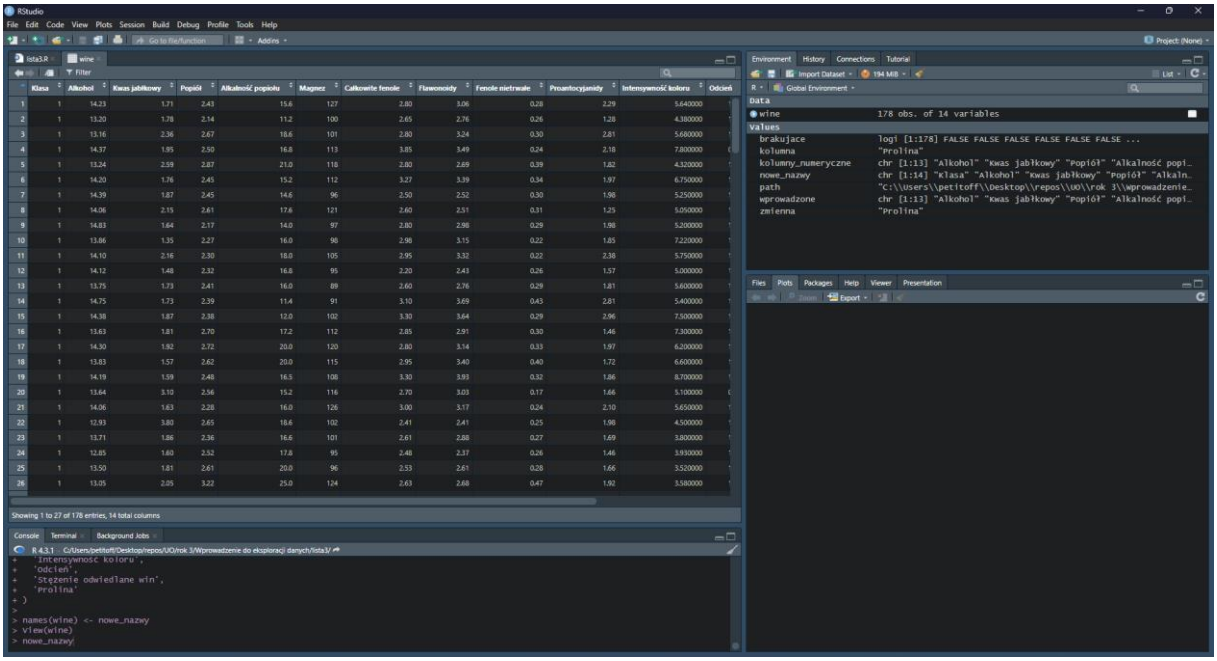
'Intensywność koloru',

'Odcień',

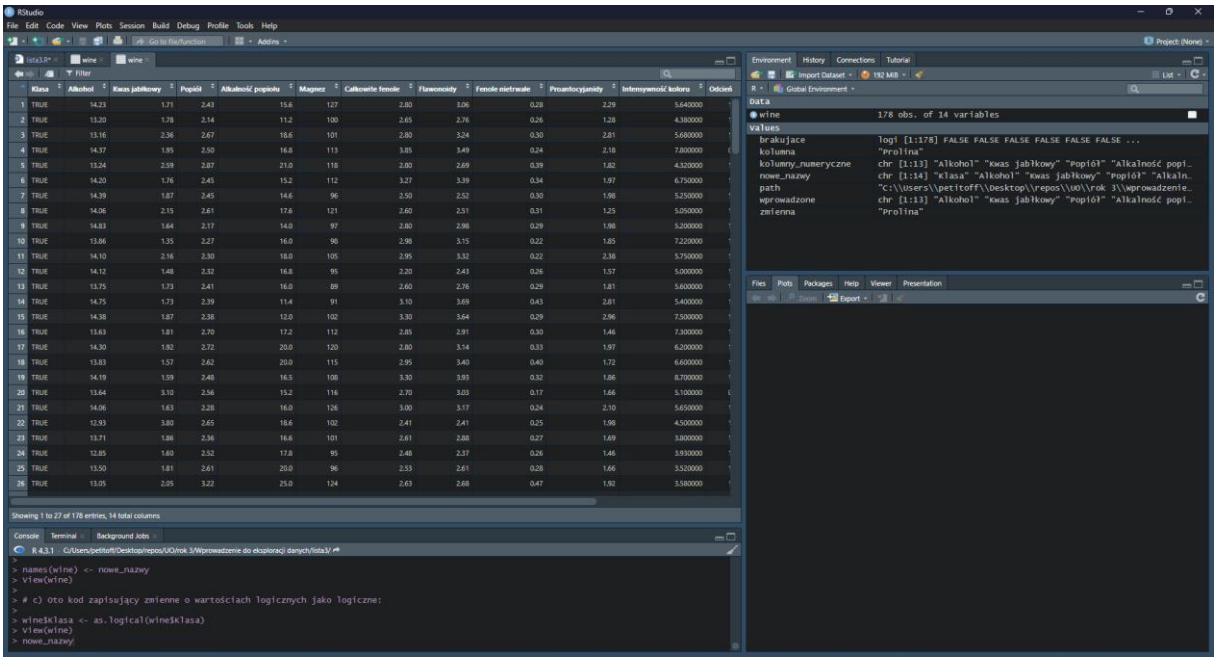
'Stężenie odwiedlane win',

'Prolina'

)



c) Zmienne o wartościach logicznych (jeżeli są) zapisz jako logiczne (polecenie as.logical).



wine$Klasa <- as.logical(wine$Klasa)

d) Upewnij się, że zmienne o wartościach liczbowych są typu liczbowego, a jeżeli nie są, to zapisz je jako numeryczne (as.numeric)

wprowadzone <- c('Alkohol', 'Kwas jabłkowy', 'Popiół', 'Alkalność popiołu', 'Magnez', 'Całkowite fenole', 'Flawonoidy', 'Fenole nietrwałe', 'Proantocyjanidy', 'Intensywność koloru', 'Odcień', 'Stężenie odwiedlane win', 'Prolina')

```
for (zmienna in wprowadzone) {
  if (class(wine[[zmienna]]) != "numeric") {
    wine[[zmienna]] <- as.numeric(wine[[zmienna]])
  }
}
```

e) Zmienną celu zapisz jako mającą wartości nominalne (polecenie as.factor).

wine$Klasa <- as.factor(wine$Klasa)

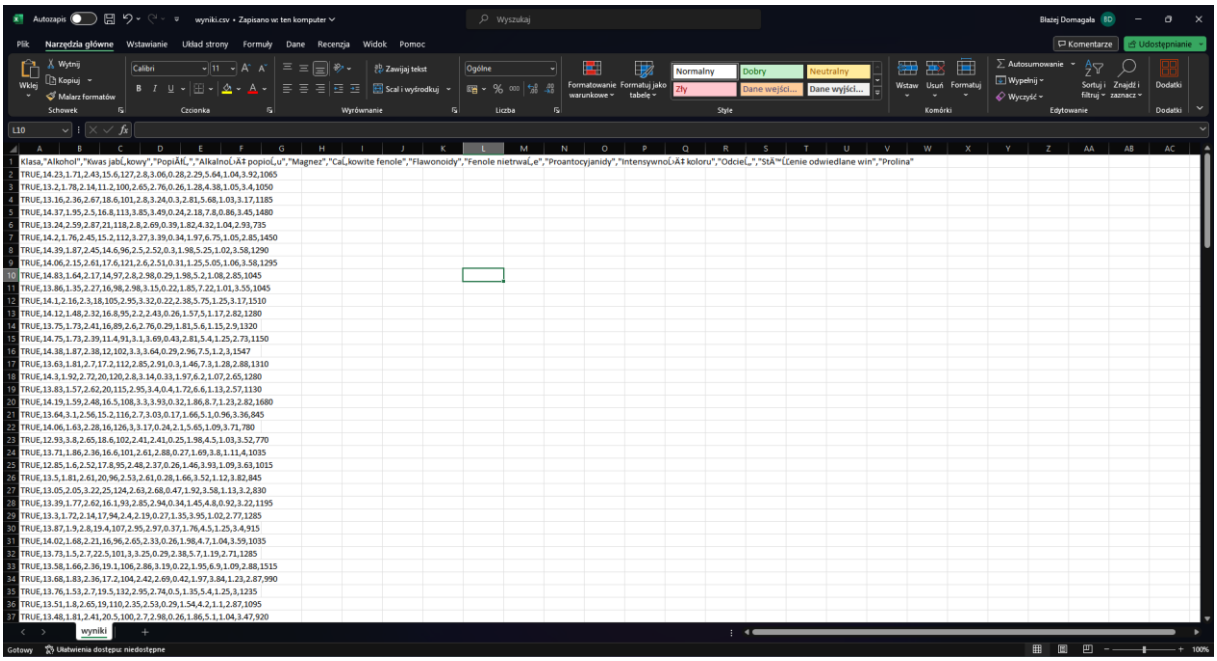
f) Policz brakujące wartości. Jeżeli są, to dla kolumn o wartościach liczbowych zastąp je wartościami średnimi dla kolumn.

kolumny_numeryczne <- c('Alkohol', 'Kwas jabłkowy', 'Popiół', 'Alkalność popiołu', 'Magnez', 'Całkowite fenole', 'Flawonoidy', 'Fenole nietrwałe', 'Proantocyjanidy', 'Intensywność koloru', 'Odcień', 'Stężenie odwiedlane win', 'Prolina')

```
for (kolumna in kolumny_numeryczne) {
  brakujace <- is.na(wine[[kolumna]])
  if (sum(brakujace) > 0) {
    srednia <- mean(wine[[kolumna]], na.rm = TRUE)
    wine[[kolumna]][brakujace] <- srednia
  }
}
```

g) Zapisz przetworzone dane do pliku (razem z nowymi nazwami kolumn). Załącz fragment print-screena zawartości (początku) pliku.

```
write.csv(wine, file = 'wyniki.csv', row.names = FALSE)
```



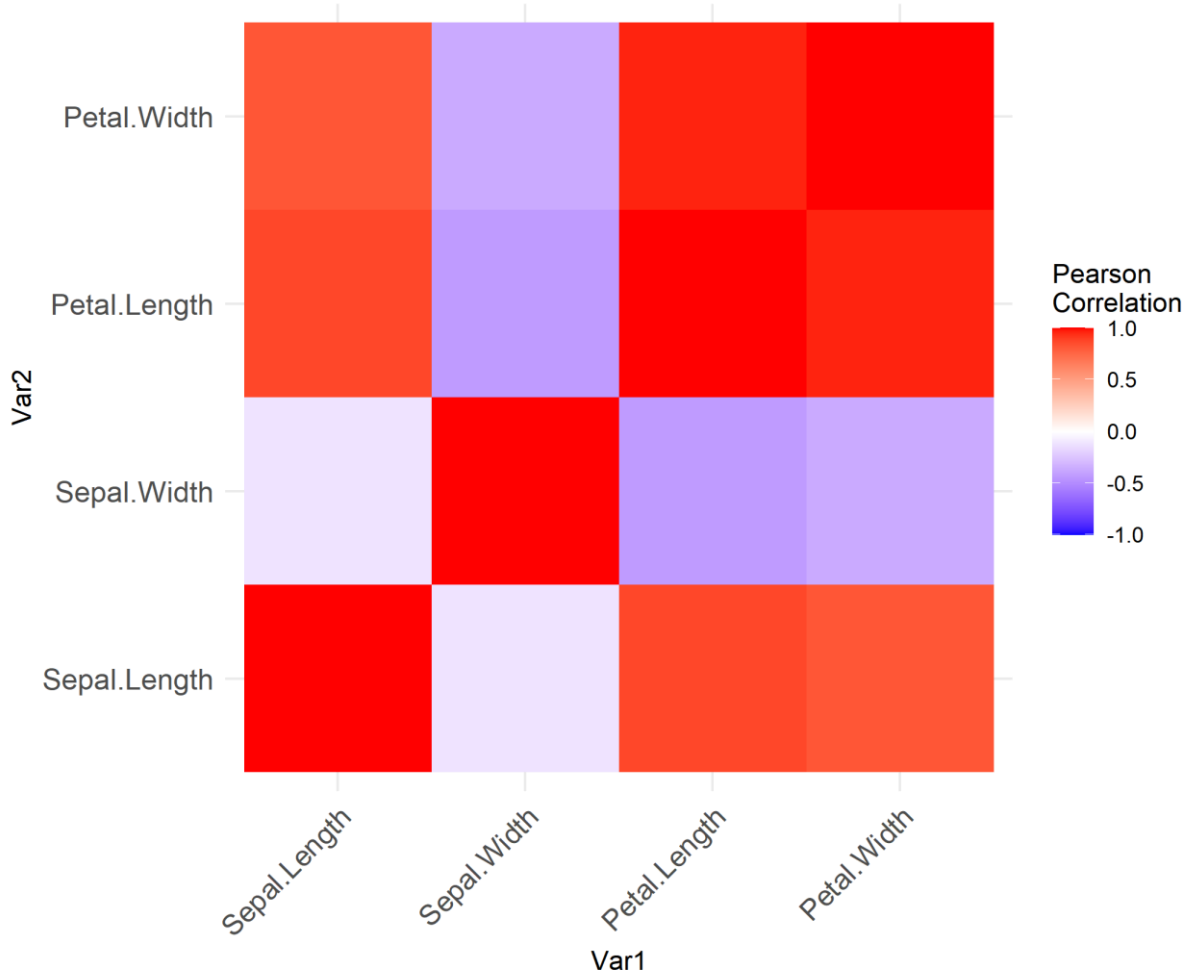
Lista 4

a) Obliczy i narysuje macierz korelacji zmiennych (bez zmiennej celu wyznaczającej klasy)

```
library(ggplot2)
library(reshape2)


correlation_matrix <- function(df, threshold) {
 # Obliczanie macierzy korelacji
 cor_matrix <- cor(df)

 # Rysowanie macierzy korelacji
 melted_cor_matrix <- melt(cor_matrix)
 plot <- ggplot(data = melted_cor_matrix, aes(x=Var1, y=Var2, fill=value)) +
  geom_tile() +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                midpoint = 0, limit = c(-1,1), space = "Lab",
                name="Pearson\nCorrelation") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, vjust = 1,
                           size = 12, hjust = 1),
      axis.text.y = element_text(size = 12)) +
  coord_fixed()

 # Zapisywanie rysunku do pliku
 ggsave("correlation_matrix.png", plot)

 # Wypisywanie par zmiennych o korelacji większej niż zadany próg
 cor_pairs <- subset(melted_cor_matrix, abs(value) > threshold & Var1 != Var2)

 # Usuwanie powtórzeń
```

```
  cor_pairs <- cor_pairs[!duplicated(t(apply(cor_pairs[,c("Var1","Var2")],1,sort))),]

  return(cor_pairs)
}

# Załadowanie zestawu danych iris
data(iris)

# Usunięcie kolumny Species (bo to jest nasza zmienna celu)
df <- iris[,-5]

# Użycie funkcji na df z progiem 0.5
correlation_matrix(df, 0.5)
```

b) zapisze rysunek macierzy korelacji do pliku



c) wypisze pary (nazwy) zmiennych o korelacji większej niż zadany próg oraz odpowiadające im wartości korelacji (wartość progu powinna być argumentem funkcji).

- Proszę uwzględnić ujemne wartości korelacji; czyli przyjmujemy, że np. korelacja równa -0.95 jest powyżej progu 0.9, bo jest to silna korelacja, tylko ujemna (wraz ze wzrostem wartości jednej cechy następuje spadek wartości drugiej cechy).
- Pary proszę wypisać bez powtórek (czyli jeżeli mamy już korelację cechy x z cechą y,to nie wypisujemy korelacji cechy y z x).

Wyniki z konsoli:

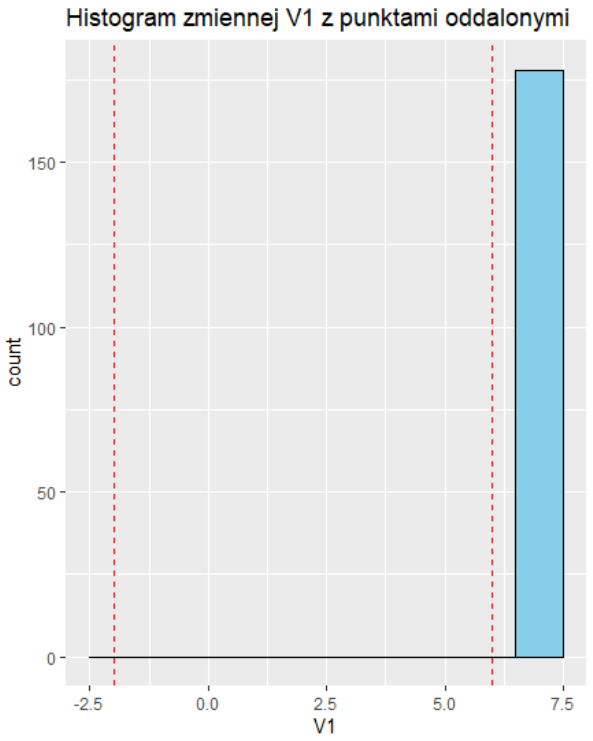
```
Saving 7 x 7 in image
           Var1         Var2     value
3  Petal.Length Sepal.Length 0.8717538
4   Petal.Width Sepal.Length 0.8179411
12  Petal.Width Petal.Length 0.9628654
>
```

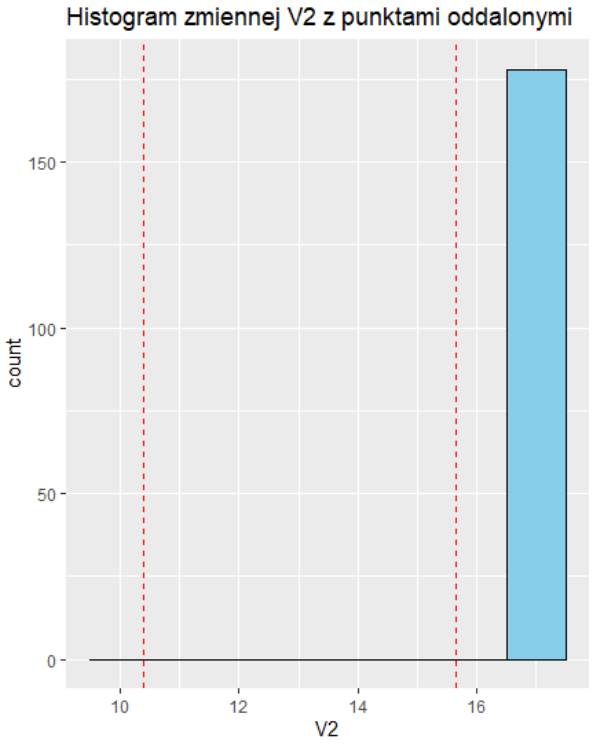
Lista 5

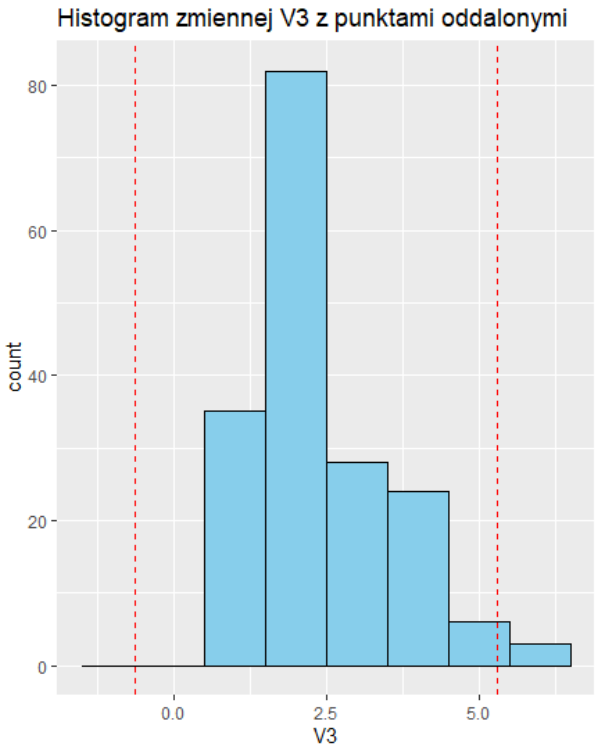
Dokumentacja Kodu R- Zadanie 2

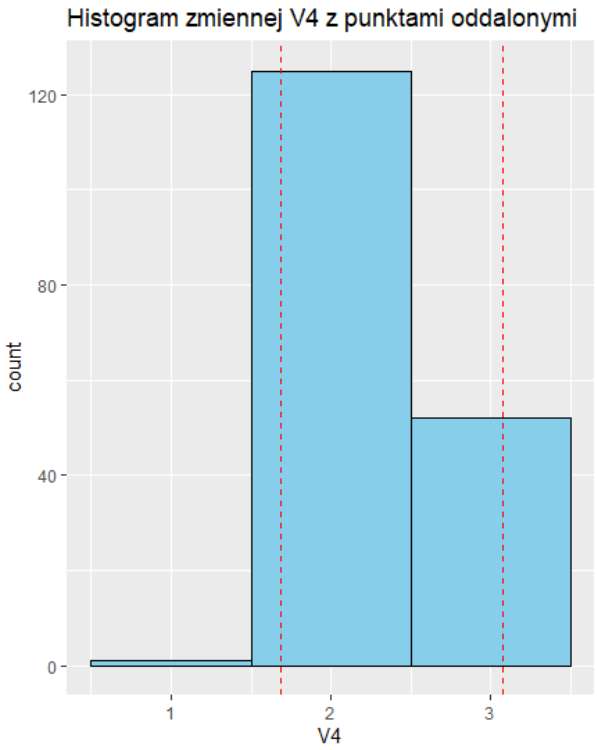
```
# Biblioteka datasets

library(datasets)


# Ustawienie ścieżki dostępu do danych

path = "C:\\Users\\petit\\Desktop\\repos\\UO\\rok 3\\Wprowadzenie do eksploracji danych\\lista5\\"

setwd(path)


# -------------- Zadanie 2a: Wczytanie i identyfikacja punktów oddalonych -------------- #

# Wczytanie danych

wine_data <- read.csv('wine\\wine.data', header = FALSE)


# Funkcja do identyfikacji i zliczania punktów oddalonych przy użyciu IQR

identify_outliers <- function(data) {

  quantiles <- quantile(data, c(.25, .75), na.rm = TRUE)

  iqr <- IQR(data, na.rm = TRUE)

  lower_bound <- quantiles[1] - 1.5 * iqr

  upper_bound <- quantiles[2] + 1.5 * iqr

  outliers <- data[data < lower_bound | data > upper_bound]

  return(list("dolna_granica" = lower_bound, "gorna_granica" = upper_bound, "punkty_oddalone" = outliers))

}


# Część a: Wykrywanie punktów oddalonych dla każdej zmiennej

outliers_info <- lapply(wine_data, identify_outliers)


# Dodanie sztucznego punktu oddalonego, jeśli nie istnieje

for (i in 1:length(outliers_info)) {
```

```
  if (length(outliers_info[[i]]$punkty_oddalone) == 0) {
    wine_data[i][1] <- outliers_info[[i]]$gorna_granica + 1 # 
Dodanie punktu oddalonego
  }
}


# Część b: Wizualizacja punktów oddalonych dla maksymalnie 4 
zmiennych
# Wybór pierwszych 4 zmiennych do wizualizacji
selected_vars <- head(names(wine_data), 4)
plots <- list()
for (var in selected_vars) {
  plot <- ggplot(wine_data, aes_string(x=var)) +
    geom_histogram(binwidth = 1, fill="skyblue", color="black") +
    geom_vline(xintercept = outliers_info[[var]]$dolna_granica, 
color="red", linetype="dashed") +
    geom_vline(xintercept = outliers_info[[var]]$gorna_granica, 
color="red", linetype="dashed") +
    ggtitle(paste("Histogram zmiennej", var, "z punktami 
oddalonymi"))
  plots[[var]] <- plot
}
```


Histogram zmiennej V1 z punktami oddalonymi
150
100
50
0
count
-2.5
0.0
2.5
5.0
7.5
V1

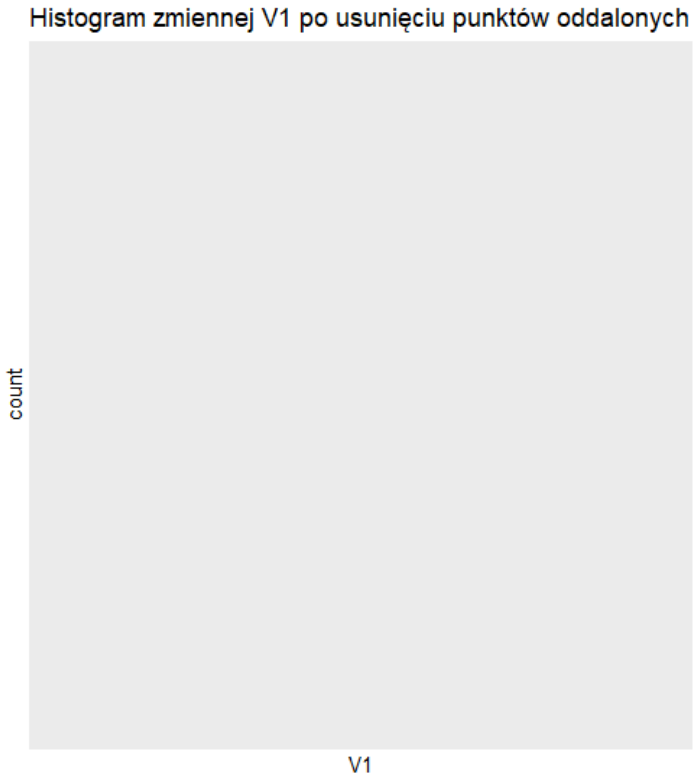

Histogram zmiennej V2 z punktami oddalonymi
150
100
50
0
count
10
12
14
16
V2



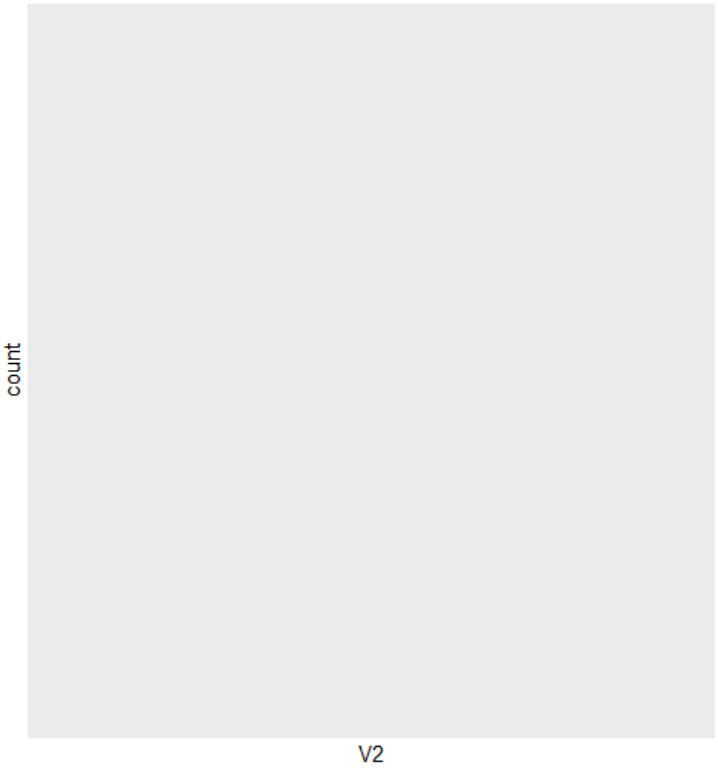



```
# Część c: Usuwanie punktów oddalonych

cleaned_wine_data <- wine_data

for (var in names(wine_data)) {

  cleaned_wine_data <- cleaned_wine_data[cleaned_wine_data[[var]] >=
outliers_info[[var]]$dolna_granica &
```

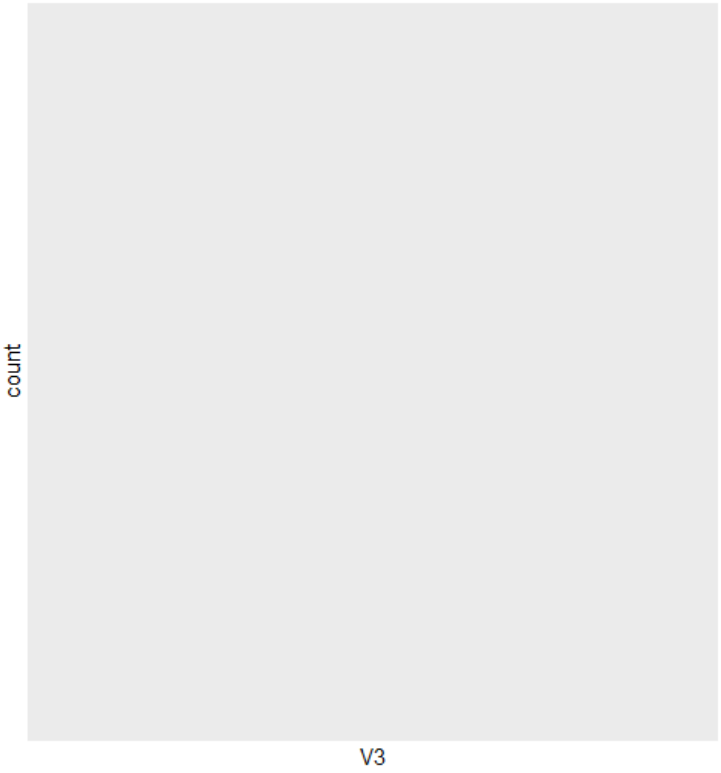
```
                                                    cleaned_wine_data[[var]]
<= outliers_info[[var]]$gorna_granica, ]
}

# Część d: Wizualizacja po usunięciu punktów oddalonych dla tych
samych 4 zmiennych
cleaned_plots <- list()
for (var in selected_vars) {
  plot <- ggplot(cleaned_wine_data, aes_string(x=var)) +
    geom_histogram(binwidth = 1, fill="green", color="black") +
    ggtitle(paste("Histogram zmiennej", var, "po usunięciu punktów
oddalonych"))
  cleaned_plots[[var]] <- plot
}
```

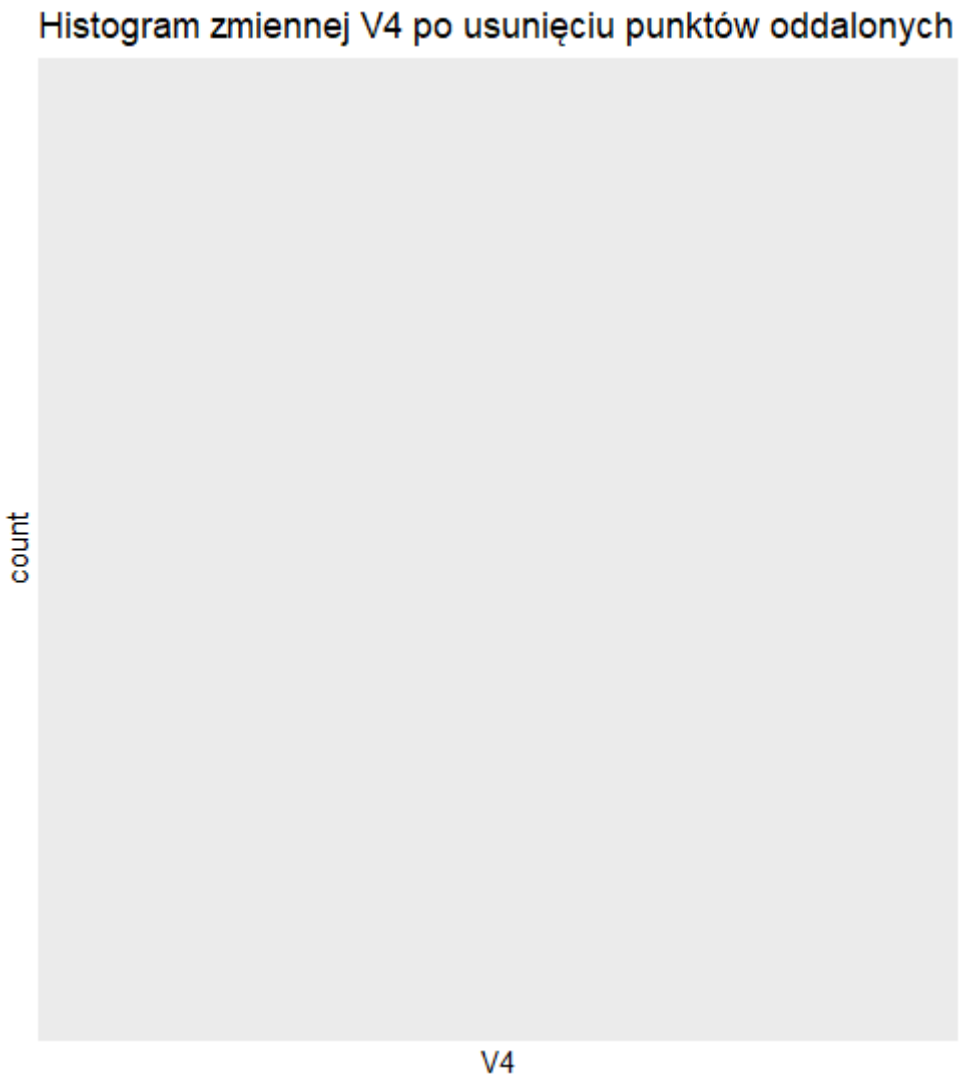


Histogram zmiennej V2 po usunięciu punktów oddalonych



Histogram zmiennej V3 po usunięciu punktów oddalonych






```
# Część e: Zapisanie oczyszczonego zbioru danych

write.csv(cleaned_wine_data, "cleaned_wine_data.csv", row.names =
FALSE)


# Wynik

list("oryginalne_wykresy" = plots, "oczyszczone_wykresy" =
cleaned_plots, "plik_z_oczyszczonymi_danymi" =
"cleaned_wine_data.csv")
```

Lista 6

```
# Wczytanie danych
wine <- read.csv('wine\\wine.data', header = FALSE)
wine_features <- wine[, -1]

# utworzenie macierzy korelacji
cor_matrix <- cor(wine_features)
print(cor_matrix)
```

```
             V2          V3           V4          V5          V6          V7          V8          V9
V2   1.00000000  0.09439694  0.211544596 -0.31023514  0.27079823  0.28910112  0.2368149 -0.1559295
V3   0.09439694  1.00000000  0.164045470  0.28850040 -0.05457510 -0.33516700 -0.4110066  0.2929771
V4   0.21154460  0.16404547  1.000000000  0.44336719  0.28658669  0.12897954  0.1150773  0.1862304
V5  -0.31023514  0.28850040  0.443367187  1.00000000 -0.08333309 -0.32111332 -0.3513699  0.3619217
V6   0.27079823 -0.05457510  0.286586691 -0.08333309  1.00000000  0.21440123  0.1957838 -0.2562940
V7   0.28910112 -0.33516700  0.128979538 -0.32111332  0.21440123  1.00000000  0.8645635 -0.4499353
V8   0.23681493 -0.41100659  0.115077279 -0.35136986  0.19578377  0.86456350  1.0000000 -0.5378996
V9  -0.15592947  0.29297713  0.186230446  0.36192172 -0.25629405 -0.44993530 -0.5378996  1.0000000
V10  0.13669791 -0.22074619  0.009651935 -0.19732684  0.23644061  0.61241308  0.6526918 -0.3658451
V11  0.54636420  0.24898534  0.258887259  0.01873198  0.19995001 -0.05513642 -0.1723794  0.1390570
V12 -0.07174720 -0.56129569 -0.074666889 -0.27395522  0.05539820  0.43368134  0.5434786 -0.2626396
V13  0.07234319 -0.36871043  0.003911231 -0.27676855  0.06600394  0.69994936  0.7871939 -0.5032696
V14  0.64372004 -0.19201056  0.223626264 -0.44059693  0.39335085  0.49811488  0.4941931 -0.3113852
            V10         V11         V12          V13        V14
V2   0.136697912  0.54636420 -0.07174720  0.072343187  0.6437200
V3  -0.220746187  0.24898534 -0.56129569 -0.368710428 -0.1920106
V4   0.009651935  0.25888726 -0.07466689  0.003911231  0.2236263
V5  -0.197326836  0.01873198 -0.27395522 -0.276768549 -0.4405969
V6   0.236440610  0.19995001  0.05539820  0.066003936  0.3933508
V7   0.612413084 -0.05513642  0.43368134  0.699949365  0.4981149
V8   0.652691769 -0.17237940  0.54347857  0.787193902  0.4941931
V9  -0.365845099  0.13905701 -0.26263963 -0.503269596 -0.3113852
V10  1.000000000 -0.02524993  0.29554425  0.519067096  0.3304167
V11 -0.025249931  1.00000000 -0.52181319 -0.428814942  0.3161001
V12  0.295544253 -0.52181319  1.00000000  0.565468293  0.2361834
V13  0.519067096 -0.42881494  0.56546829  1.000000000  0.3127611
V14  0.330416700  0.31610011  0.23618345  0.312761075  1.0000000
```

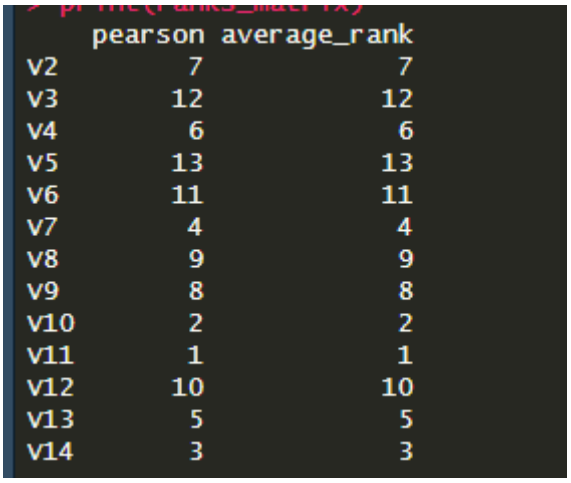
```
# Obliczanie i wydrukowanie indeksów oraz rang cech:
pearson_corr <- sapply(wine_features, function(x) cor(x, wine[, 1]))
pearson_rank <- order(-abs(pearson_corr))
print(pearson_corr)
print(pearson_rank)
```

```
>
> # Obliczanie i wydrukowanie indeksów oraz rang cech:
> pearson_corr <- sapply(wine_features, function(x) cor(x, wine[, 1]))
> pearson_rank <- order(-abs(pearson_corr))
> print(pearson_corr)
        V2         V3         V4         V5         V6         V7         V8         V9
-0.32822194  0.43777620 -0.04964322  0.51785911 -0.20917939 -0.71916334 -0.84749754  0.48910916
       V10        V11        V12        V13        V14
-0.49912982  0.26566757 -0.61736921 -0.78822959 -0.63371678
> print(pearson_rank)
 [1]  7 12  6 13 11  4  9  8  2  1 10  5  3
>
```

```
# Utworzenie tablicy 2D z rangami:
ranks_matrix <- data.frame(pearson = pearson_rank)
rownames(ranks_matrix) <- colnames(wine_features)
print(ranks_matrix)
```

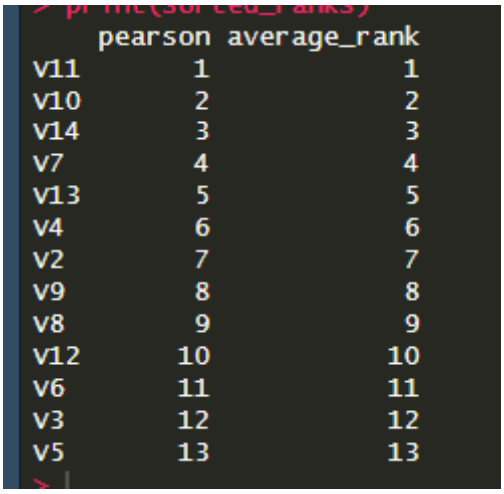
```
    pearson
V2        7
V3       12
V4        6
V5       13
V6       11
V7        4
V8        9
V9        8
V10       2
V11       1
V12      10
V13       5
V14       3
```

```
# Dodanie kolumny ze średnią wartością rangi:
ranks_matrix$average_rank <- rowMeans(ranks_matrix, na.rm = TRUE)
print(ranks_matrix)
```



	pearson	average_rank
V2	7	7
V3	12	12
V4	6	6
V5	13	13
V6	11	11
V7	4	4
V8	9	9
V9	8	8
V10	2	2
V11	1	1
V12	10	10
V13	5	5
V14	3	3

```
# Sortowanie cech według wartości średniej rangi:

sorted_ranks <- ranks_matrix[order(ranks_matrix$average_rank), ]

print(sorted_ranks)
```



	pearson	average_rank
V11	1	1
V10	2	2
V14	3	3
V7	4	4
V13	5	5
V4	6	6
V2	7	7
V9	8	8
V8	9	9
V12	10	10
V6	11	11
V3	12	12
V5	13	13

```
# Spearman Correlation

spearman_corr <- sapply(wine_features, function(x) cor(x, wine[, 1],
method = "spearman"))

spearman_rank <- order(-abs(spearman_corr))

ranks_matrix$spearman <- spearman_rank

print(spearman_corr)

print(spearman_rank)
```

```
> 
> # Spearman Correlation
> spearman_corr <- sapply(wine_features, function(x) cor(x, wine[, 1], method = "spearman"))
> spearman_rank <- order(-abs(spearman_corr))
> ranks_matrix$spearman <- spearman_rank
> print(spearman_corr)
         V2          V3          V4          V5          V6          V7 
-0.35416692  0.34691327 -0.05398792  0.56979214 -0.25049819 -0.72654365 
         V8          V9         V10         V11         V12         V13 
-0.85490766  0.47420549 -0.57064758  0.13117017 -0.61657049 -0.74378690 
        V14 
-0.57638313 
> print(spearman_rank)
 [1]  7 12  6 11 13  9  4  8  1  2  5 10  3
```

```
# Kendall Correlation

kendall_corr <- sapply(wine_features, function(x) cor(x, wine[, 1], 
method = "kendall"))

kendall_rank <- order(-abs(kendall_corr))

ranks_matrix$kendall <- kendall_rank

print(kendall_corr)

print(kendall_rank)
```

```
> 
> # Kendall Correlation
> kendall_corr <- sapply(wine_features, function(x) cor(x, wine[, 1], method = "kendall"))
> kendall_rank <- order(-abs(kendall_corr))
> ranks_matrix$kendall <- kendall_rank
> print(kendall_corr)
         V2          V3          V4          V5          V6          V7 
-0.23898423  0.24749447 -0.03808511  0.44940228 -0.18499225 -0.59040381 
         V8          V9         V10         V11         V12         V13 
-0.72525486  0.37923359 -0.45022461  0.06512382 -0.47922876 -0.60757229 
        V14 
-0.40626000 
> print(kendall_rank)
 [1]  7 12  6 11  9  4 13  8  2  1  5 10  3
> 
```

```
# Mutual Information

mi_scores <- sapply(wine_features, function(x) 
mutinformation(discretize(x), discretize(wine[, 1])))

mi_rank <- order(-mi_scores)

ranks_matrix$MI <- mi_rank

print(mi_scores)

print(mi_rank)
```

```
> 
> # Mutual Information
> mi_scores <- sapply(wine_features, function(x) mutinformation(discretize(x), discretize(wine[, 1])))
> mi_rank <- order(-mi_scores)
> ranks_matrix$MI <- mi_rank
> print(mi_scores)
        V2         V3         V4         V5         V6         V7         V8 
0.43718816 0.26059723 0.09409405 0.19209690 0.19419468 0.36548167 0.61456543 
        V9        V10        V11        V12        V13        V14 
0.17079017 0.21654374 0.47156977 0.40134555 0.48896304 0.53321967 
> print(mi_rank)
 [1]  7 13 12 10  1 11  6  2  9  5  4  8  3
```

```
# PCA - pierwsza główna składowa
pca_result <- prcomp(wine_features, scale. = TRUE)
pca_rank <- order(-abs(pca_result$rotation[, 1]))
ranks_matrix$PCA <- pca_rank
print(pca_result$rotation[, 1])
print(pca_rank)
```

```
>
> # PCA - pierwsza główna składowa
> pca_result <- prcomp(wine_features, scale. = TRUE)
> pca_rank <- order(-abs(pca_result$rotation[, 1]))
> ranks_matrix$PCA <- pca_rank
> print(pca_result$rotation[, 1])
         V2          V3          V4          V5          V6          V7
-0.144329395  0.245187580  0.002051061  0.239320405 -0.141992042 -0.394660845
         V8          V9         V10         V11         V12         V13
-0.422934297  0.298533103 -0.313429488  0.088616705 -0.296714564 -0.376167411
        V14
-0.286752227
> print(pca_rank)
 [1]  7  6 12  9  8 11 13  2  4  1  5 10  3
```

```
# LDA
lda_result <- lda(wine[, 1] ~ ., data = wine_features)
lda_rank <- order(-abs(lda_result$scaling[, 1]))
ranks_matrix$LDA <- lda_rank
print(lda_result$scaling[, 1])
print(lda_rank)
```

```
>
> # LDA
> lda_result <- lda(wine[, 1] ~ ., data = wine_features)
> lda_rank <- order(-abs(lda_result$scaling[, 1]))
> ranks_matrix$LDA <- lda_rank
> print(lda_result$scaling[, 1])
         V2          V3          V4          V5          V6          V7
-0.403399781  0.165254596 -0.369075256  0.154797889 -0.002163496  0.618052068
         V8          V9         V10         V11         V12         V13
-1.661191235 -1.495818440  0.134092628  0.355055710 -0.818036073 -1.157559376
        V14
-0.002691206
> print(lda_rank)
 [1]  7  8 12 11  6  1  3 10  2  4  9 13  5
```

```
# Chi-Square Test
# Uwaga: Ten test ma sens tylko dla cech kategorialnych.


# Aktualizacja średniej rangi
ranks_matrix$average_rank <- rowMeans(ranks_matrix[, -1], na.rm =
TRUE)
```

```
print(ranks_matrix$average_rank)
```

```
> 
> # Chi-Square Test
> # Uwaga: Ten test ma sens tylko dla cech kategorialnych.
> 
> # Aktualizacja średniej rangi
> ranks_matrix$average_rank <- rowMeans(ranks_matrix[, -1], na.rm = TRUE)
> print(ranks_matrix$average_rank)
 [1]  7.000000 10.500000  9.000000 10.833333  8.000000  6.666667  8.000000  6.333333  3.333333
[10]  2.333333  6.333333  9.333333  3.333333
> 
```

```
# Sortowanie cech według wartości średniej rangi
sorted_ranks <- ranks_matrix[order(ranks_matrix$average_rank), ]
print(sorted_ranks)
```

```
> 
> # Sortowanie cech według wartości średniej rangi
> sorted_ranks <- ranks_matrix[order(ranks_matrix$average_rank), ]
> print(sorted_ranks)
    pearson average_rank spearman kendall MI PCA LDA
V11       1     2.333333        2       1  5   1   4
V10       2     3.333333        1       2  9   4   2
V14       3     3.333333        3       3  3   3   5
V9        8     6.333333        8       8  2   2  10
V12      10     6.333333        5       5  4   5   9
V7        4     6.666667        9       4 11  11   1
V2        7     7.000000        7       7  7   7   7
V6       11     8.000000       13       9  1   8   6
V8        9     8.000000        4      13  6  13   3
V4        6     9.000000        6       6 12  12  12
V13       5     9.333333       10      10  8  10  13
V3       12    10.500000       12      12 13   6   8
V5       13    10.833333       11      11 10   9  11
> 
```

Wnioski

1. **Ważność Cech wg Korelacji Pearsona:**
    - Cecha **V11** wykazuje najwyższą korelację (najniższą rangę) z etykietami klas, co wskazuje na jej potencjalnie dużą ważność w modelowaniu.
    - Z kolei cecha **V5** ma najniższą korelację (najwyższą rangę), co sugeruje jej mniejsze znaczenie.
2. **Różnorodność Korelacji:**
    - Obserwujemy różnorodne wartości korelacji, od silnie negatywnych (np. **V7**, **V8**, **V13**) do silnie pozytywnych (np. **V4**, **V10**, **V11**), co wskazuje na złożoność zależności w tych danych.
3. **Znaczenie dla Modelowania:**
    - Cechy o wyższej korelacji mogą być bardziej znaczące w modelach predykcyjnych, jednak warto pamiętać, że korelacja nie zawsze równa się przyczynowości.
    - Cechy o niskiej korelacji nie powinny być automatycznie odrzucane, gdyż mogą wnosić cenne informacje w połączeniu z innymi cechami.

4. **Wnioski z Innych Metod Filtracyjnych**:
    - **Korelacje Spearmana i Kendalla**: Te metody wskazują na monotoniczne związki między cechami a etykietami klas. Zauważalna jest pewna konsystencja z wynikami korelacji Pearsona, co dodatkowo potwierdza ważność niektórych cech (np. V11, V10).
    - **Wzajemna Informacja (MI)**: Ta metoda ocenia wzajemną zależność między zmiennymi, co może ujawnić nieliniowe związki. Cechy z wysokim wynikiem MI mogą odgrywać kluczową rolę w rozróżnianiu klas, nawet jeśli ich liniowa korelacja jest słaba.
5. **Analiza Głównych Składowych (PCA)**:
    - Wyniki PCA mogą pomóc zrozumieć, które cechy najbardziej przyczyniają się do wariancji w zbiorze danych. Cechy z wysokimi wartościami na pierwszej głównej składowej mogą być istotne dla różnorodności w danych.
6. **Analiza Dyskryminacyjna Liniowa (LDA)**:
    - Wyniki LDA podkreślają cechy, które najlepiej rozróżniają klasy. Ta metoda jest szczególnie przydatna w kontekście klasyfikacji i może wskazywać na cechy krytyczne dla rozróżniania między kategoriami wina.
7. **Ważność Integracji Różnych Metod**:
    - Integracja wyników z różnych metod daje bardziej zrównoważony widok na ważność cech. Cechy, które są konsekwentnie wysoko oceniane przez różne metody, są prawdopodobnie kluczowe dla zrozumienia i modelowania zbioru danych.
8. **Praktyczne Implikacje dla Modelowania**:
    - Cechy o wysokich średnich rangach z różnych metod filtracyjnych mogą być priorytetowe przy tworzeniu modeli predykcyjnych.
    - Warto jednak zachować ostrożność i nie wykluczać cech o niższych rangach, ponieważ mogą one odgrywać istotne role w interakcjach z innymi zmiennymi.

Lista 7

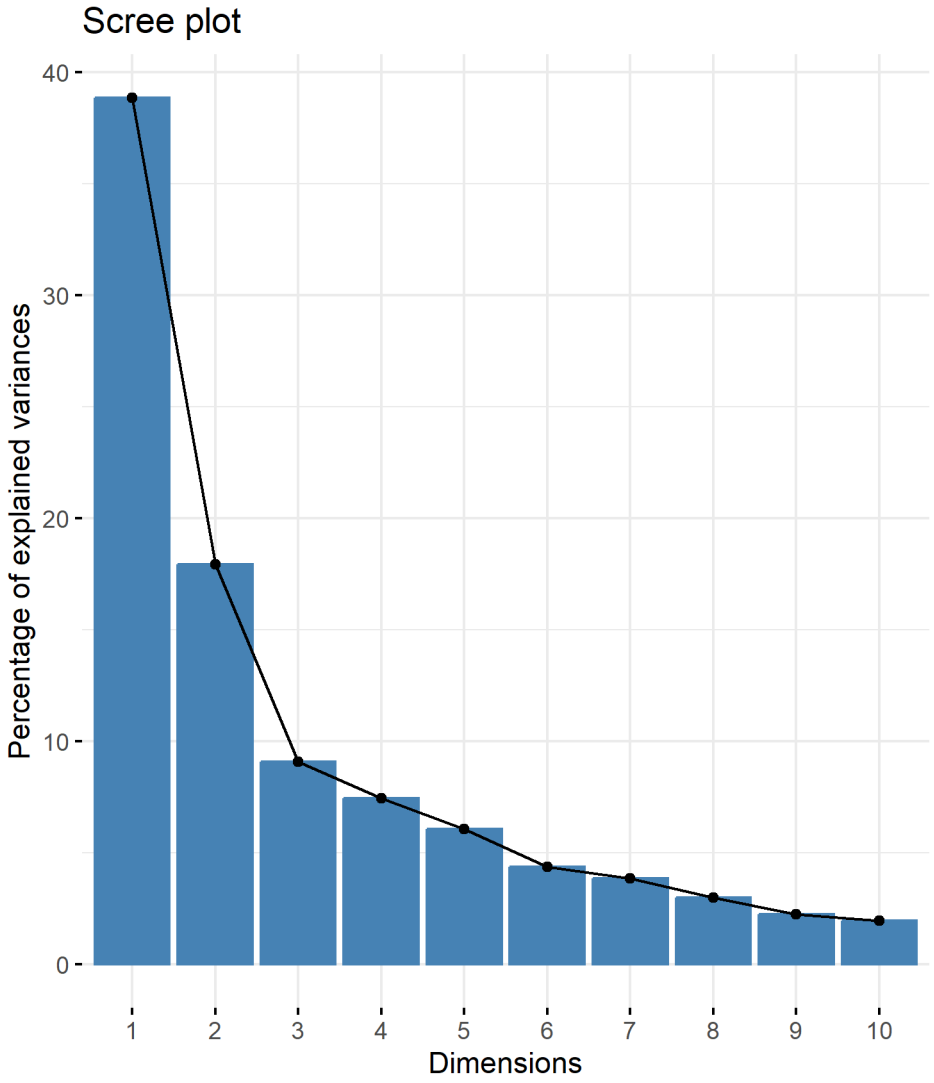
1. **Wczytanie danych**: Import danych z pliku **wine.data** bez nagłówków.

```
# Wczytanie danych
wine_data <- read.csv('wine\\wine.data', header = FALSE)
```

2. **Usunięcie brakujących danych**: Wyeliminowanie obserwacji z brakującymi wartościami.
3. **Usunięcie punktów oddalonych**: Obliczenie odległości Mahalanobisa dla każdego punktu, ustalenie progu (95 percentyl) i usunięcie punktów przekraczających ten próg.

```
# Usunięcie obserwacji brakujących i punktów oddalonych
wine_data <- na.omit(wine_data)  # Usunięcie obserwacji brakujących

# Obliczanie odległości Mahalanobisa dla każdego punktu
wine_data$distance <- mahalanobis(wine_data, colMeans(wine_data), cov(wine_data))

# Ustalenie progu dla identyfikacji punktów oddalonych, np. 95 percentyla
threshold <- quantile(wine_data$distance, 0.95)

# Usuwanie punktów oddalonych
wine_data <- wine_data[wine_data$distance <= threshold, ]

# Przygotowanie danych do PCA
wine_data_pca <- PCA(wine_data, graph = FALSE)

# a) Wartości własne i wyjaśniana wariancja
eig_val <- get_eigenvalue(wine_data_pca)
print(eig_val)
fviz_eig(wine_data_pca)  # Wykres osypiskowy
ggsave("scree_plot.png", bg = "white")  # Zapisanie do pliku
```

```
> print(eig_val)
       eigenvalue variance.percent cumulative.variance.percent
Dim.1  5.82941960       38.8627973                    38.86280
Dim.2  2.69124789       17.9416526                    56.80445
Dim.3  1.36266926        9.0844617                    65.88891
Dim.4  1.11687523        7.4458349                    73.33475
Dim.5  0.91200393        6.0800262                    79.41477
Dim.6  0.65538714        4.3692476                    83.78402
Dim.7  0.57881234        3.8587489                    87.64277
Dim.8  0.45060262        3.0040174                    90.64679
Dim.9  0.33766066        2.2510711                    92.89786
Dim.10 0.29486852        1.9657902                    94.86365
Dim.11 0.24451072        1.6300714                    96.49372
Dim.12 0.21036027        1.4024018                    97.89612
Dim.13 0.15883684        1.0589123                    98.95503
Dim.14 0.11625933        0.7750622                    99.73010
Dim.15 0.04048565        0.2699044                   100.00000
```




```
# b) Ładunki czynnikowe

loadings <- wine_data_pca$var$coord

print(loadings)
```

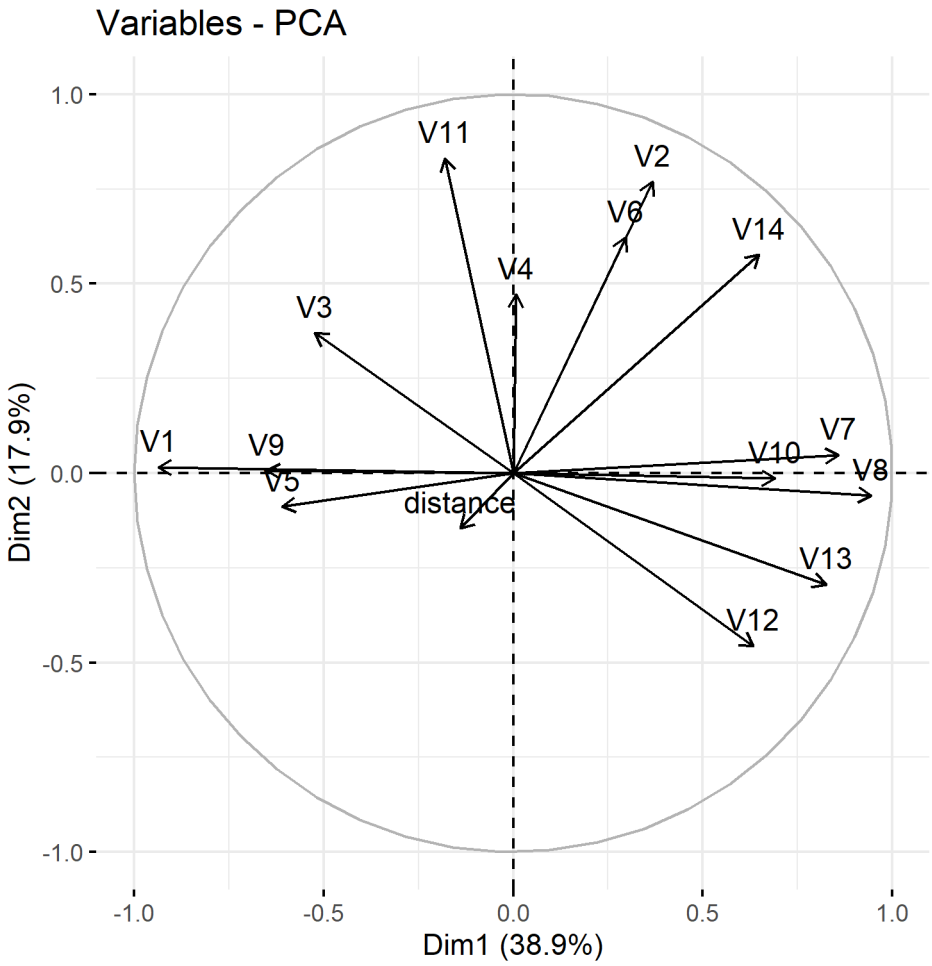
```
> # b) Ładunki czynnikowe
> loadings <- wine_data_pca$var$coord
> print(loadings)
                Dim.1        Dim.2       Dim.3       Dim.4       Dim.5
V1       -0.936928269  0.015146156 -0.01927134  0.18943162  0.02981302
V2        0.367586316  0.769061818 -0.17948661 -0.07777073  0.05030504
V3       -0.522912555  0.370153338  0.07767294 -0.04167927  0.54345198
V4        0.007349719  0.471623556  0.75545529 -0.30885394 -0.07645014
V5       -0.609378200 -0.088850874  0.58443340 -0.20620017  0.18428368
V6        0.296382194  0.621980642  0.21408833  0.15125893 -0.35554663
V7        0.857891058  0.047866699  0.17805513  0.07838012  0.15582113
V8        0.945084062 -0.059908544  0.12677495  0.05406437  0.12959302
V9       -0.650228330  0.006129602  0.16256496 -0.30488966 -0.20176459
V10       0.690515627 -0.014080261  0.20169227  0.25071746  0.35613353
V11      -0.180569880  0.830998790 -0.14142455  0.21855428  0.01484359
V12       0.633475899 -0.456736402  0.14587262 -0.16124894 -0.36342541
V13       0.825279206 -0.293856815  0.08752902 -0.16378616  0.23474509
V14       0.648633381  0.576533095 -0.09169389 -0.13357897 -0.16076558
distance -0.138966452 -0.146083137  0.44012379  0.79315643 -0.15862579
>
```

```
# c) Zasoby zmienności wspólnej
communalities <- wine_data_pca$var$cos2
print(communalities)
```

```
> # c) Zasoby zmienności wspólnej
> communalities <- wine_data_pca$var$cos2
> print(communalities)
                Dim.1        Dim.2        Dim.3        Dim.4        Dim.5
V1       8.778346e-01 2.294060e-04 0.0003713846 0.035884340 0.0008888159
V2       1.351197e-01 5.914561e-01 0.0322154420 0.006048287 0.0025305968
V3       2.734375e-01 1.370135e-01 0.0060330858 0.001737162 0.2953400600
V4       5.401837e-05 2.224288e-01 0.5707126951 0.095390759 0.0058446242
V5       3.713418e-01 7.894478e-03 0.3415624034 0.042518510 0.0339604752
V6       8.784241e-02 3.868599e-01 0.0458338130 0.022879263 0.1264134034
V7       7.359771e-01 2.291221e-03 0.0317036285 0.006143443 0.0242802256
V8       8.931839e-01 3.589034e-03 0.0160718881 0.002922956 0.0167943500
V9       4.227969e-01 3.757203e-05 0.0264273669 0.092957704 0.0407089506
V10      4.768118e-01 1.982537e-04 0.0406797735 0.062859246 0.1268310942
V11      3.260548e-02 6.905590e-01 0.0200009035 0.047765975 0.0002203321
V12      4.012917e-01 2.086081e-01 0.0212788212 0.026001221 0.1320780319
V13      6.810858e-01 8.635183e-02 0.0076613292 0.026825906 0.0551052595
V14      4.207253e-01 3.323904e-01 0.0084077694 0.017843342 0.0258455728
distance 1.931167e-02 2.134028e-02 0.1937089526 0.629097117 0.0251621400
```

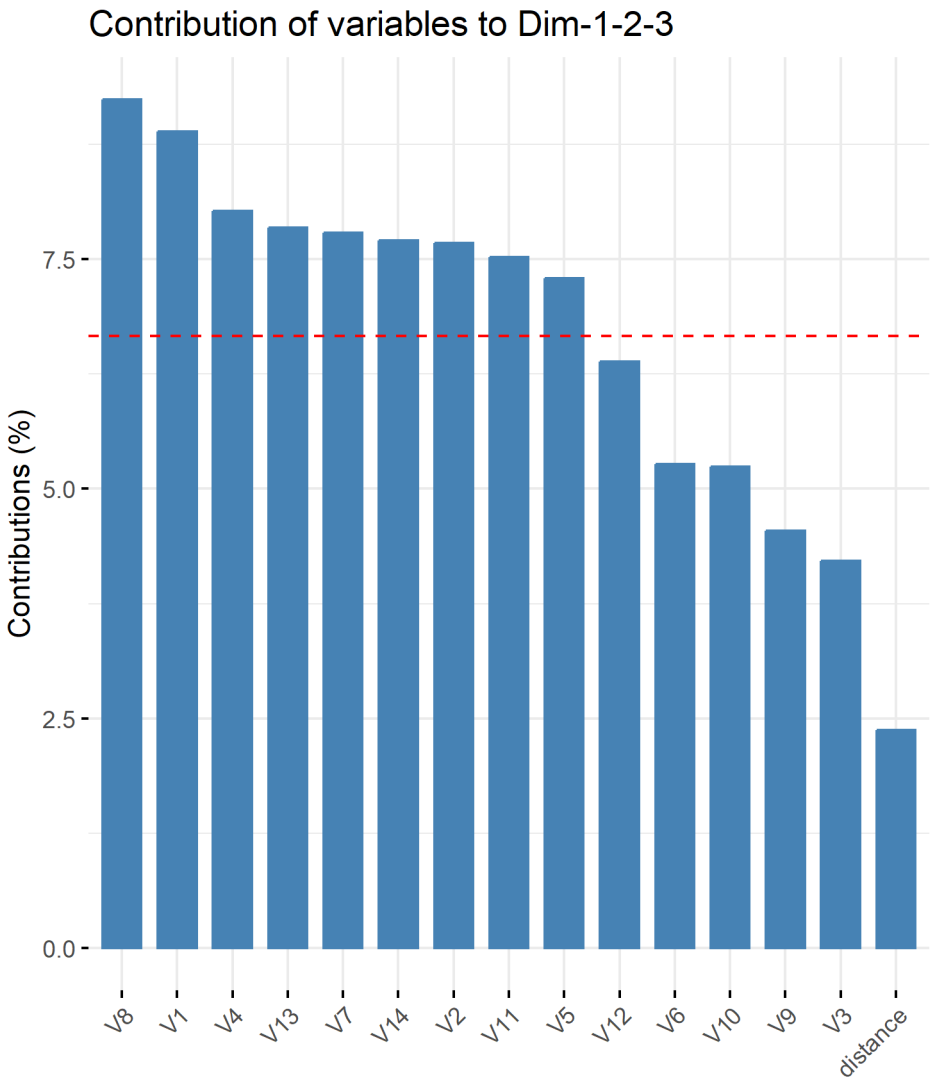
```
# d) Wkłady zmiennych pierwotnych w składowe główne
contributions <- wine_data_pca$var$contrib
print(contributions)
fviz_pca_var(wine_data_pca)  # Wykres korelacji
ggsave("correlation_plot.png", bg = "white")  # Zapisanie do pliku
```

```
> 
> # d) Wkłady zmiennych pierwotnych w składowe główne
> contributions <- wine_data_pca$var$contrib
> print(contributions)
                Dim.1        Dim.2       Dim.3      Dim.4       Dim.5
V1         15.058696078  0.008524151  0.0272542  3.2129229  0.09745746
V2          2.317892847 21.977019751  2.3641424  0.5415365  0.27747652
V3          4.690647762  5.091076681  0.4427403  0.1555377 32.38363888
V4          0.000926651  8.264893769 41.8819675  8.5408608  0.64085516
V5          6.370133147  0.293338934 25.0656865  3.8069168  3.72372026
V6          1.506880805 14.374741194  3.3635317  2.0485066 13.86105904
V7         12.625220323  0.085136003  2.3265828  0.5500564  2.66229396
V8         15.322003641  0.133359458  1.1794416  0.2617084  1.84147780
V9          7.252812631  0.001396082  1.9393823  8.3230159  4.46368148
V10         8.179404876  0.007366610  2.9853006  5.6281351 13.90685827
V11         0.559326381 25.659434546  1.4677739  4.2767512  0.02415911
V12         6.883905134  7.751353650  1.5615544  2.3280328 14.48217790
V13        11.683594835  3.208616641  0.5622296  2.4018713  6.04221731
V14         7.217275341 12.350791308  0.6170073  1.5976128  2.83393217
distance    0.331279548  0.792951222 14.2154049 56.3265349  2.75899468
```



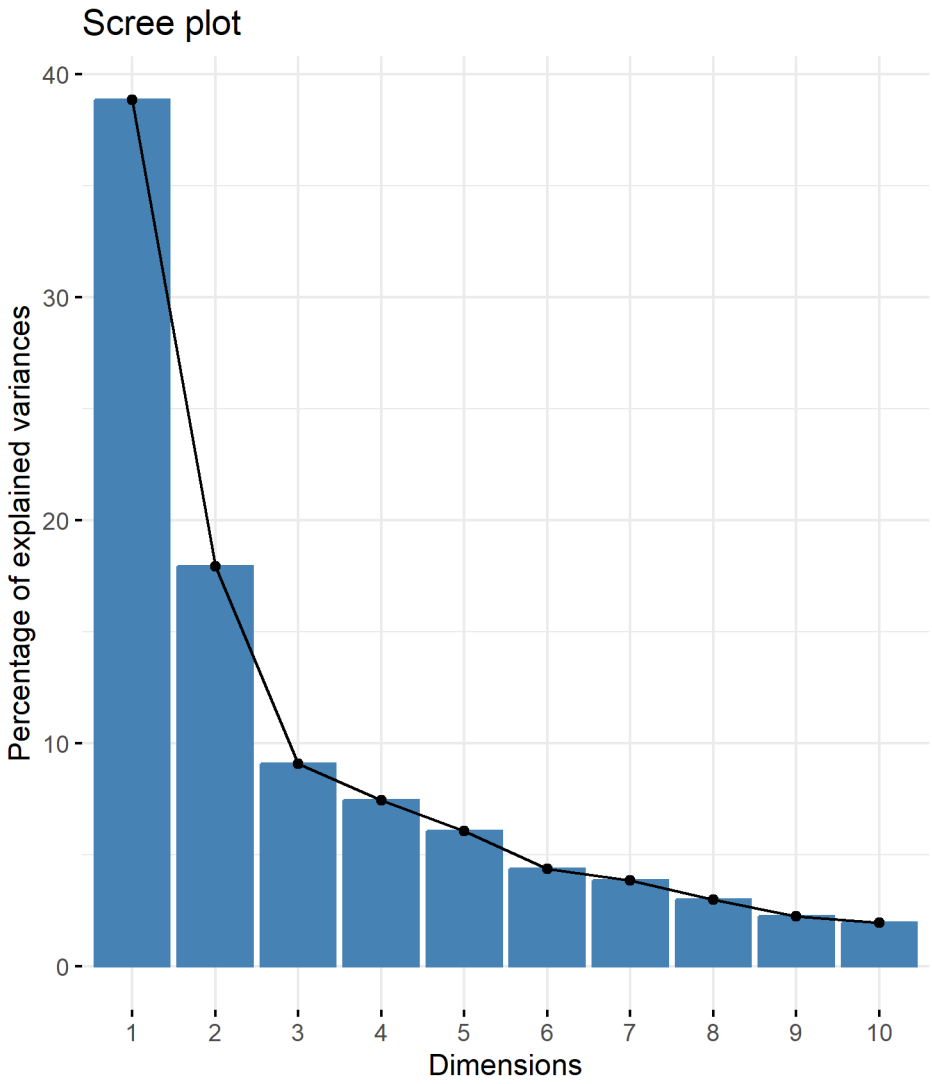

```
# e) Wykres łącznego wkładu dla pierwszych trzech wymiarów
fviz_contrib(wine_data_pca, choice = "var", axes = 1:3)
```

```
ggsave("contrib_plot.png", bg = "white")
```




```
# f) Wektory własne
eigen_vectors <- wine_data_pca$ind$coord
print(eigen_vectors)


# g) Wybór liczby składowych
# Analiza procentu wyjaśnianej wariancji
fviz_screeplot(wine_data_pca)
ggsave("explained_variance_plot.png", bg = "white")  # Zapisanie do
pliku
```

```
# Załóżmy, że decydujemy się zachować pierwsze 3 składowe główne
selected_components <- 1:3
```




```
# h) Utworzenie nowej tabeli danych

# Wybór składowych głównych do nowej tabeli danych
new_data <- wine_data_pca$ind$coord[, selected_components]

# Konwersja new_data do ramki danych, jeśli to konieczne
new_data <- as.data.frame(new_data)

# Dodanie zmiennej celu (klasy) do nowej tabeli
new_data$class <- wine_data[, 1]
```

```
# Zapis nowej tabeli danych do pliku CSV
if (!require(readr)) install.packages("readr")
library(readr)

write_csv(new_data, "new_wine_data.csv")
```

Lista 8

Standardize the data

wine_data_scaled <- scale(wine_data)

```
# a) Stosowanie metody k-średnich dla k = 1 do 15

results <- list()

for (k in 1:15) {

  set.seed(123)  # dla powtarzalności wyników

  kmeans_result <- kmeans(wine_data_scaled, centers = k, nstart =
25)

  results[[k]] <- kmeans_result

  cat("Dla k =", k, "\n",

      "Całkowita suma kwadratów wewnątrz klastrów:",
kmeans_result$tot.withinss, "\n",

      "Suma kwadratów dla każdego klastra:", kmeans_result$withinss,
"\n",

      "Centra klastrów:\n", kmeans_result$centers, "\n",

      "Liczba obserwacji w każdym klastrze:", kmeans_result$size,
"\n",

      "Przypisanie klastrów dla pierwszych 20 obserwacji:",
kmeans_result$cluster[1:20], "\n\n")

}
```

Wyniki

Dołączam plik excel oraz wklejam dla przyszłego wysyłania arkusze z pliku

kmeans_clustering_
updated_results.xls›

Total Within-Cluster Sum of Squares	**Within-Cluster Sum of Squares**	**Cluster Centers**	**Cluster Sizes**	**Cluster Assignments for First 20 Observations**

2478	[2478]	-1.227483e-15 -8.719617e-16 -5.738231e-17 8.470128e-16 -1.559302e-16 -6.14365e-17 2.186141e-16 1.135172e-16 6.224734e-16 -1.503167e-16 2.370139e-17 1.846214e-16 3.492836e-17 -6.112464e-17	[178]	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

Total Within-Cluster Sum of Squares	Within-Cluster Sum of Squares	Cluster Centers	Cluster Sizes	Cluster Assignments for First 20 Observations
1717,525	[532.4956, 1185.03]	1.032546 -0.5939424 -0.07277357 0.0418609 0.6626412 -0.3811653 0.1893414 -0.1089132 0.5151693 -0.2963363 -0.1542527 0.08872942 -0.9410522 0.5413132 -1.043692 0.6003536 0.835592 -0.4806503 -0.7141412 0.4107892 0.5419399 -0.3117353 -0.8795908 0.5059593 -1.06631 0.6133643 -0.4519062 0.259946	[65, 113]	[2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]

Total Within-Cluster Sum of Squares	Within-Cluster Sum of Squares	Cluster Centers	Cluster Sizes	Cluster Assignments for First 20 Observations
1030	[100, 101, 102]	0.00 1.00 2.00 3.00 4.00 5.00 6.00 7.00 8.00 9.00 10.00 11.00 12.00 13.00 14.00	[10, 11, 12]	[1, 2, 3, 1, 2, 3]

Total Within-Cluster Sum of Squares	Within-Cluster Sum of Squares	Cluster Centers	Cluster Sizes	Cluster Assignments for First 20 Observations
1040	[100, 101, 102, 103]	0.00 1.00 2.00 3.00 4.00 5.00 6.00 7.00 8.00 9.00 10.00 11.00 12.00 13.00 14.00 15.00 16.00 17.00 18.00 19.00	[10, 11, 12, 13]	[1, 2, 3, 4, 1, 2, 3, 4]

Total Within-Cluster Sum of Squares	Within-Cluster Sum of Squares	Cluster Centers	Cluster Sizes	Cluster Assignments for First 20 Observations

1050	[100, 101, 102, 103, 104]	0.00 1.00 2.00 3.00 4.00 5.00 6.00 7.00 8.00 9.00 10.00 11.00 12.00 13.00 14.00 15.00 16.00 17.00 18.00 19.00 20.00 21.00 22.00 23.00 24.00	[10, 11, 12, 13, 14]	[1, 2, 3, 4, 5, 1, 2, 3, 4, 5]

Total Within-Cluster Sum of Squares	Within-Cluster Sum of Squares	Cluster Centers	Cluster Sizes	Cluster Assignments for First 20 Observations
1060	[100, 101, 102, 103, 104, 105]	0.00 1.00 2.00 3.00 4.00 5.00 6.00 7.00 8.00 9.00 10.00 11.00 12.00 13.00 14.00 15.00 16.00 17.00 18.00 19.00 20.00 21.00 22.00 23.00 24.00 25.00 26.00 27.00 28.00 29.00	[10, 11, 12, 13, 14, 15]	[1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 5, 6]

Total Within-Cluster Sum of Squares	Within-Cluster Sum of Squares	Cluster Centers	Cluster Sizes	Cluster Assignments for First 20 Observations
1070	[100, 101, 102, 103, 104, 105, 106]	0.00 1.00 2.00 3.00 4.00 5.00 6.00 7.00 8.00 9.00 10.00 11.00 12.00 13.00 14.00 15.00 16.00 17.00 18.00 19.00 20.00 21.00 22.00 23.00 24.00 25.00 26.00 27.00 28.00 29.00 30.00 31.00 32.00 33.00 34.00	[10, 11, 12, 13, 14, 15, 16]	[1, 2, 3, 4, 5, 6, 7, 1, 2, 3, 4, 5, 6, 7]

Total Within-Cluster Sum of Squares	Within-Cluster Sum of Squares	Cluster Centers	Cluster Sizes	Cluster Assignments for First 20 Observations
1080	[100, 101, 102, 103, 104, 105, 106, 107]	0.00 1.00 2.00 3.00 4.00 5.00 6.00 7.00 8.00 9.00 10.00 11.00 12.00 13.00 14.00 15.00 16.00 17.00 18.00 19.00 20.00 21.00 22.00 23.00 24.00 25.00 26.00 27.00 28.00 29.00 30.00 31.00 32.00 33.00 34.00 35.00 36.00 37.00 38.00 39.00	[10, 11, 12, 13, 14, 15, 16, 17]	[1, 2, 3, 4, 5, 6, 7, 8, 1, 2, 3, 4, 5, 6, 7, 8]

Total Within-Cluster Sum of Squares	Within-Cluster Sum of Squares	Cluster Centers	Cluster Sizes	Cluster Assignments for First 20 Observations
1090	[100, 101, 102,	0.00 1.00 2.00 3.00 4.00 5.00 6.00 7.00 8.00 9.00 10.00 11.00 12.00 13.00 14.00 15.00 16.00 17.00 18.00 19.00 20.00 21.00	[10, 11, 12,	[1, 2, 3, 4, 5, 6, 7, 8, 9, 1,

	103, 104, 105, 106, 107, 108]	22.00 23.00 24.00 25.00 26.00 27.00 28.00 29.00 30.00 31.00 32.00 33.00 34.00 35.00 36.00 37.00 38.00 39.00 40.00 41.00 42.00 43.00 44.00	13, 14, 15, 16, 17, 18]	2, 3, 4, 5, 6, 7, 8, 9]

Total Within-Cluster Sum of Squares	Within-Cluster Sum of Squares	Cluster Centers	Cluster Sizes	Cluster Assignments for First 20 Observations
1100	[100, 101, 102, 103, 104, 105, 106, 107, 108, 109]	0.00 1.00 2.00 3.00 4.00 5.00 6.00 7.00 8.00 9.00 10.00 11.00 12.00 13.00 14.00 15.00 16.00 17.00 18.00 19.00 20.00 21.00 22.00 23.00 24.00 25.00 26.00 27.00 28.00 29.00 30.00 31.00 32.00 33.00 34.00 35.00 36.00 37.00 38.00 39.00 40.00 41.00 42.00 43.00 44.00 45.00 46.00 47.00 48.00 49.00	[10, 11, 12, 13, 14, 15, 16, 17, 18, 19]	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Total Within-Cluster Sum of Squares	Within-Cluster Sum of Squares	Cluster Centers	Cluster Sizes	Cluster Assignments for First 20 Observations
1110	[100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110]	0.00 1.00 2.00 3.00 4.00 5.00 6.00 7.00 8.00 9.00 10.00 11.00 12.00 13.00 14.00 15.00 16.00 17.00 18.00 19.00 20.00 21.00 22.00 23.00 24.00 25.00 26.00 27.00 28.00 29.00 30.00 31.00 32.00 33.00 34.00 35.00 36.00 37.00 38.00 39.00 40.00 41.00 42.00 43.00 44.00 45.00 46.00 47.00 48.00 49.00 50.00 51.00 52.00 53.00 54.00	[10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]

Total Within-Cluster Sum of Squares	Within-Cluster Sum of Squares	Cluster Centers	Cluster Sizes	Cluster Assignments for First 20 Observations
1120	[100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111]	0.00 1.00 2.00 3.00 4.00 5.00 6.00 7.00 8.00 9.00 10.00 11.00 12.00 13.00 14.00 15.00 16.00 17.00 18.00 19.00 20.00 21.00 22.00 23.00 24.00 25.00 26.00 27.00 28.00 29.00 30.00 31.00 32.00 33.00 34.00 35.00 36.00 37.00 38.00 39.00 40.00 41.00 42.00 43.00 44.00 45.00 46.00 47.00 48.00 49.00 50.00 51.00 52.00 53.00 54.00 55.00 56.00 57.00 58.00 59.00	[10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21]	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

Total Within-Cluster Sum of Squares	Within-Cluster Sum of Squares	Cluster Centers	Cluster Sizes	Cluster Assignments for First 20 Observations

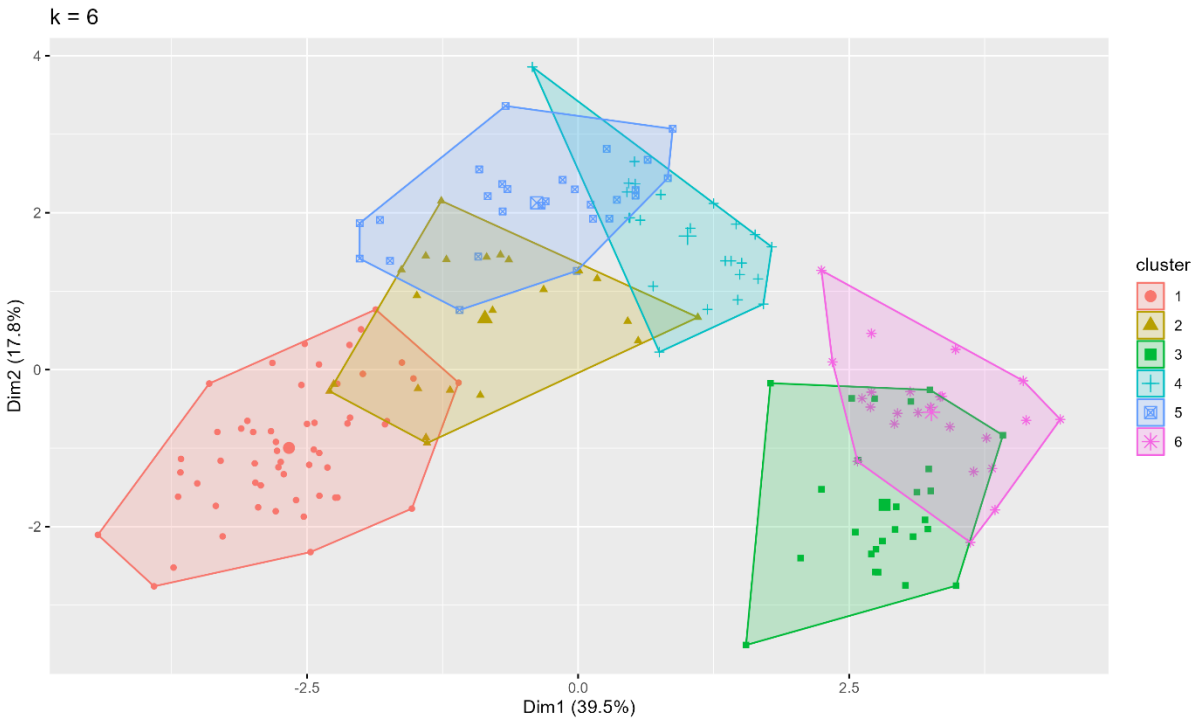
	[100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112]	0.00 1.00 2.00 3.00 4.00 5.00 6.00 7.00 8.00 9.00 10.00 11.00 12.00 13.00 14.00 15.00 16.00 17.00 18.00 19.00 20.00 21.00 22.00 23.00 24.00 25.00 26.00 27.00 28.00 29.00 30.00 31.00 32.00 33.00 34.00 35.00 36.00 37.00 38.00 39.00 40.00 41.00 42.00 43.00 44.00 45.00 46.00 47.00 48.00 49.00 50.00 51.00 52.00 53.00 54.00 55.00 56.00 57.00 58.00 59.00 60.00 61.00 62.00 63.00 64.00	[10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22]	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
1130				

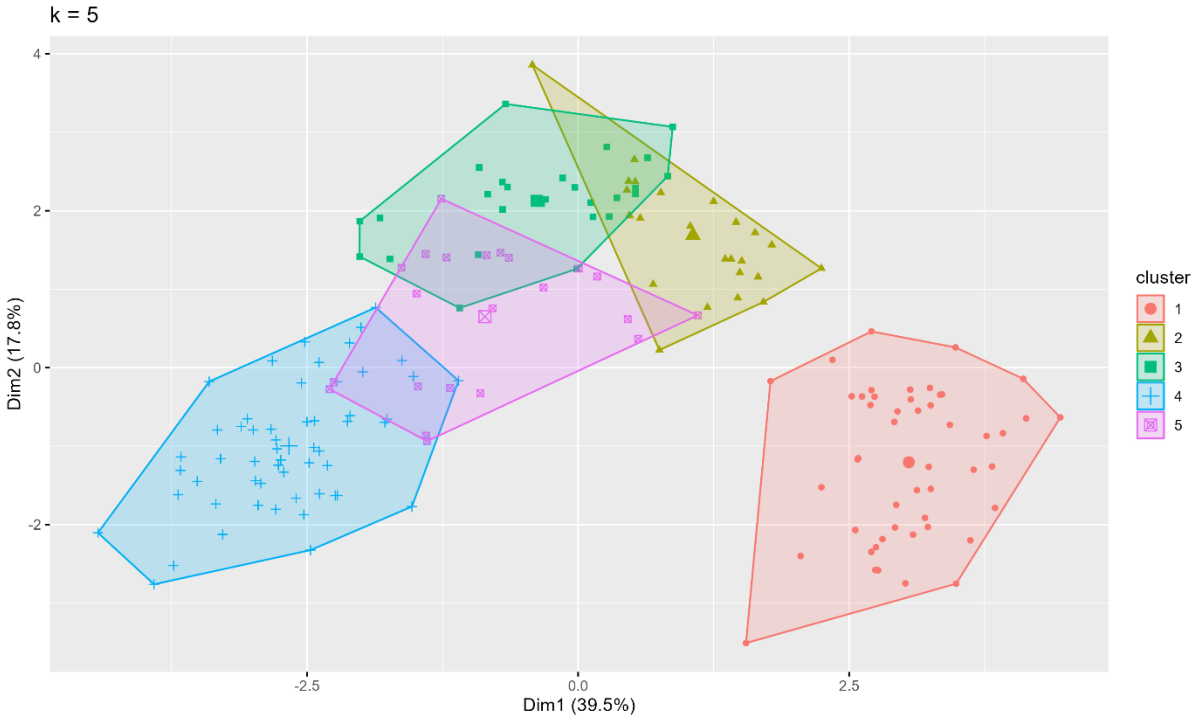
Total Within-Cluster Sum of Squares	Within-Cluster Sum of Squares	Cluster Centers	Cluster Sizes	Cluster Assignments for First 20 Observations
	[100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113]	0.00 1.00 2.00 3.00 4.00 5.00 6.00 7.00 8.00 9.00 10.00 11.00 12.00 13.00 14.00 15.00 16.00 17.00 18.00 19.00 20.00 21.00 22.00 23.00 24.00 25.00 26.00 27.00 28.00 29.00 30.00 31.00 32.00 33.00 34.00 35.00 36.00 37.00 38.00 39.00 40.00 41.00 42.00 43.00 44.00 45.00 46.00 47.00 48.00 49.00 50.00 51.00 52.00 53.00 54.00 55.00 56.00 57.00 58.00 59.00 60.00 61.00 62.00 63.00 64.00 65.00 66.00 67.00 68.00 69.00	[10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23]	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
1140				

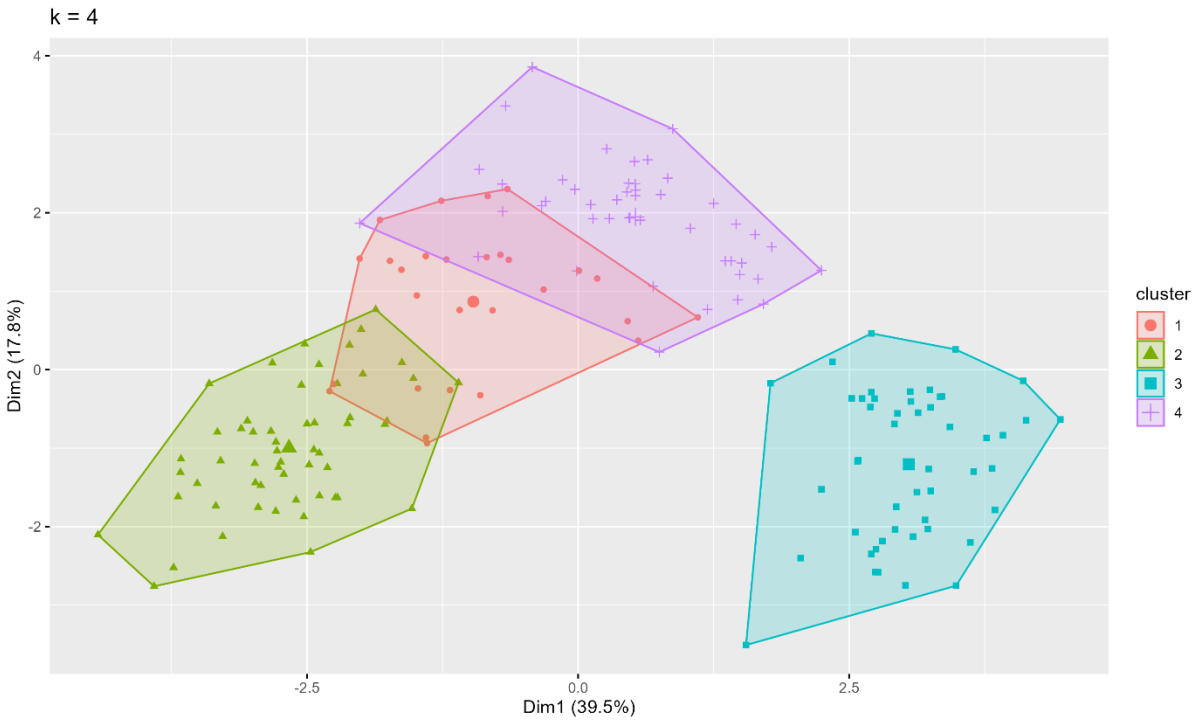
Total Within-Cluster Sum of Squares	Within-Cluster Sum of Squares	Cluster Centers	Cluster Sizes	Cluster Assignments for First 20 Observations
	[100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114]	0.00 1.00 2.00 3.00 4.00 5.00 6.00 7.00 8.00 9.00 10.00 11.00 12.00 13.00 14.00 15.00 16.00 17.00 18.00 19.00 20.00 21.00 22.00 23.00 24.00 25.00 26.00 27.00 28.00 29.00 30.00 31.00 32.00 33.00 34.00 35.00 36.00 37.00 38.00 39.00 40.00 41.00 42.00 43.00 44.00 45.00 46.00 47.00 48.00 49.00 50.00 51.00 52.00 53.00 54.00 55.00 56.00 57.00 58.00 59.00 60.00 61.00 62.00 63.00 64.00 65.00 66.00 67.00 68.00 69.00 70.00 71.00 72.00 73.00 74.00	[10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24]	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
1150				

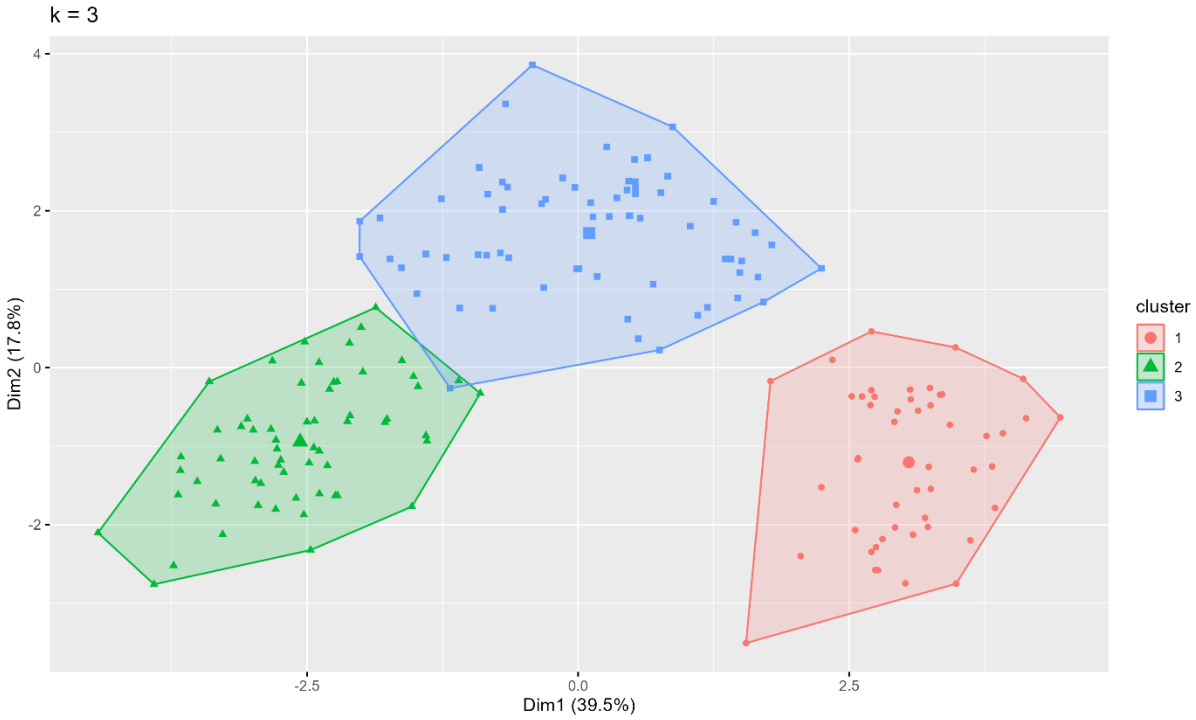
```
# b) Wizualizacja wyników dla k = 2 do 7

par(mfrow = c(3, 2))

for (k in 2:7) {

  # Tworzenie wykresu

  plot <- fviz_cluster(results[[k]], data = wine_data_scaled, geom =
"point", ellipse = TRUE, main = paste("k =", k))
```

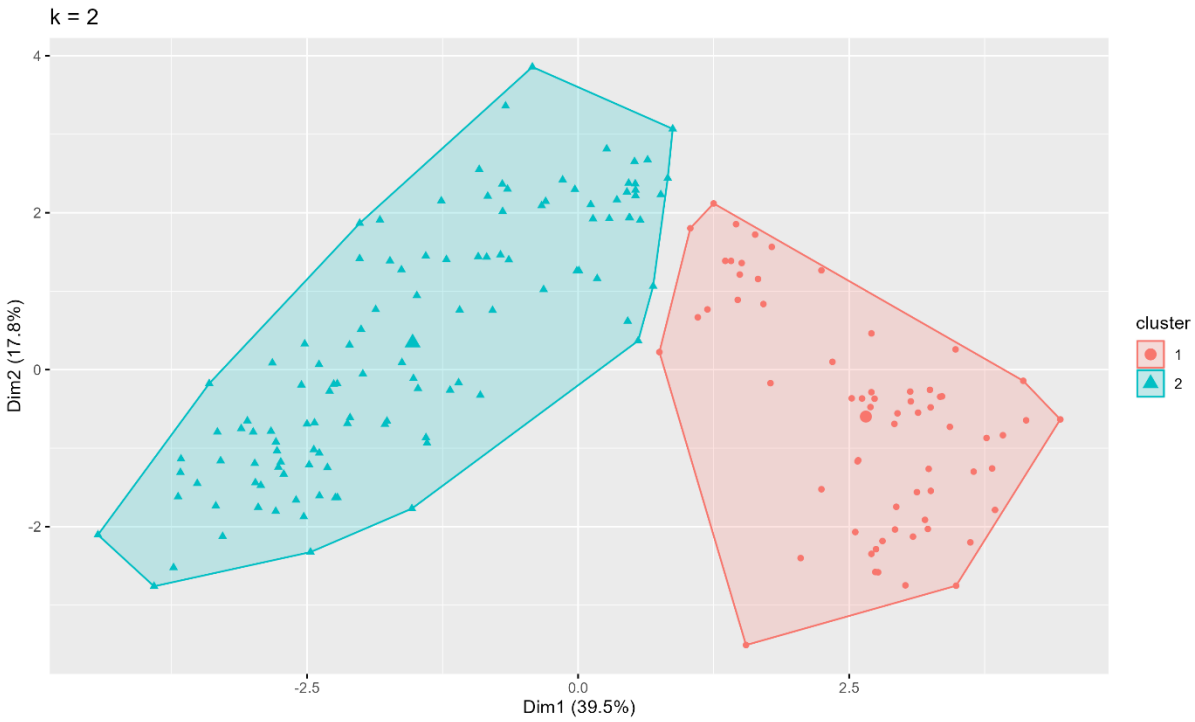
```
  # Wyświetlanie wykresu
  print(plot)

  # Zapisywanie wykresu do pliku PNG
  ggsave(filename = paste("kmeans_k", k, ".png"), plot = plot, width
= 10, height = 6)

  # Sprawdź, czy plik został zapisany
  if (file.exists(paste("kmeans_k", k, ".png"))) {
    cat("Plik", paste("kmeans_k", k, ".png"), "został zapisany.\n")
  } else {
    cat("Plik", paste("kmeans_k", k, ".png"), "NIE został
zapisany.\n")
  }
}
```

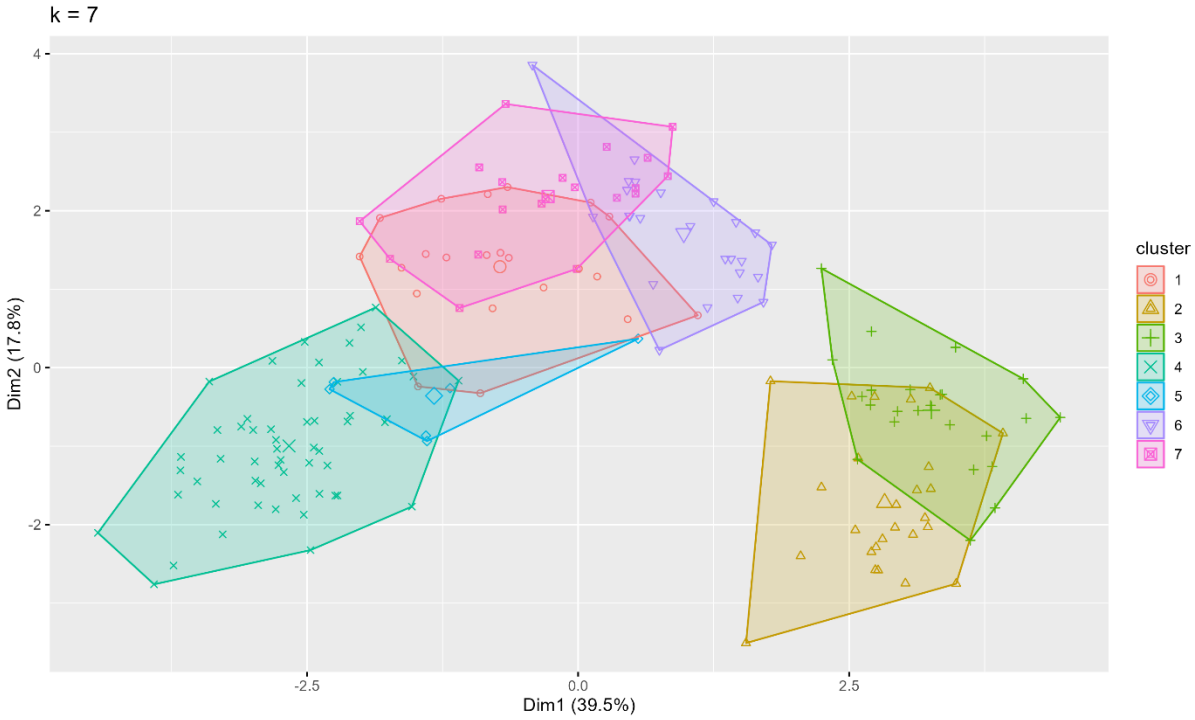

k = 6
Dim2 (17.8%)
Dim1 (39.5%)
4
2
0
-2
-2.5
0.0
2.5
cluster
1
2
3
4
5
6


k = 5
Dim2 (17.8%)
Dim1 (39.5%)
4
2
0
-2
-2.5
0.0
2.5
cluster
1
2
3
4
5


k = 4
Dim2 (17.8%)
Dim1 (39.5%)
cluster
1
2
3
4


k = 3
Dim2 (17.8%)
Dim1 (39.5%)
cluster
1
2
3

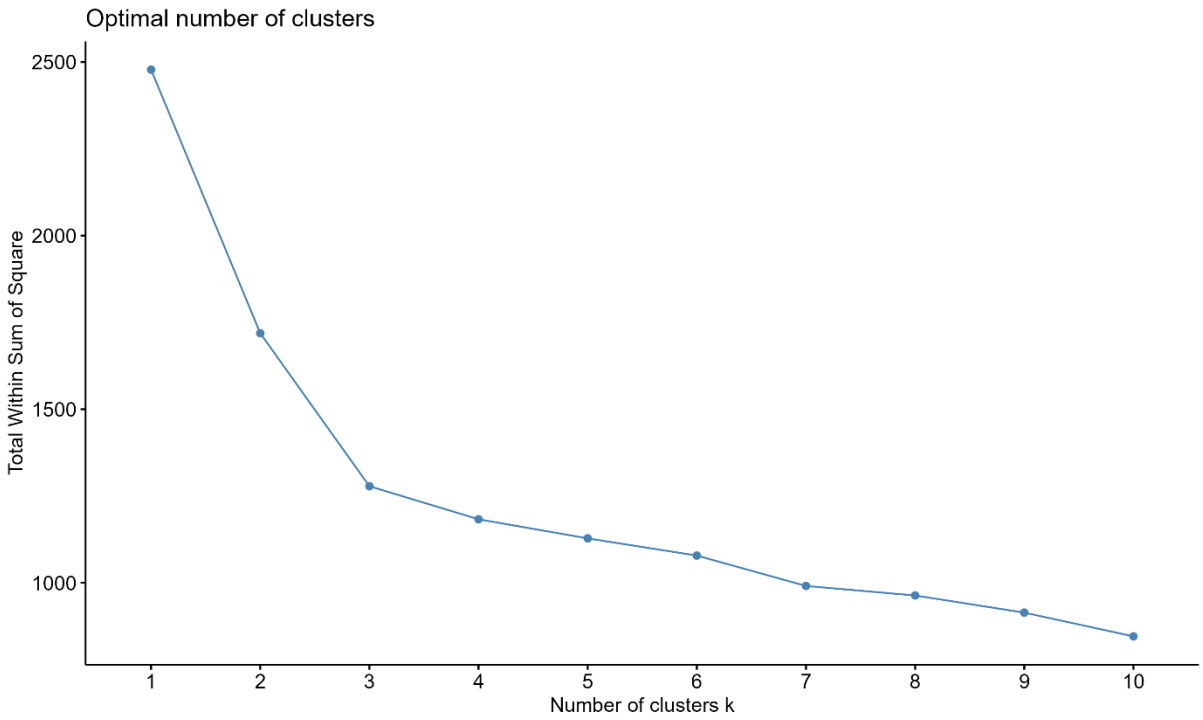




c) Określenie optymalnej liczby klastrów

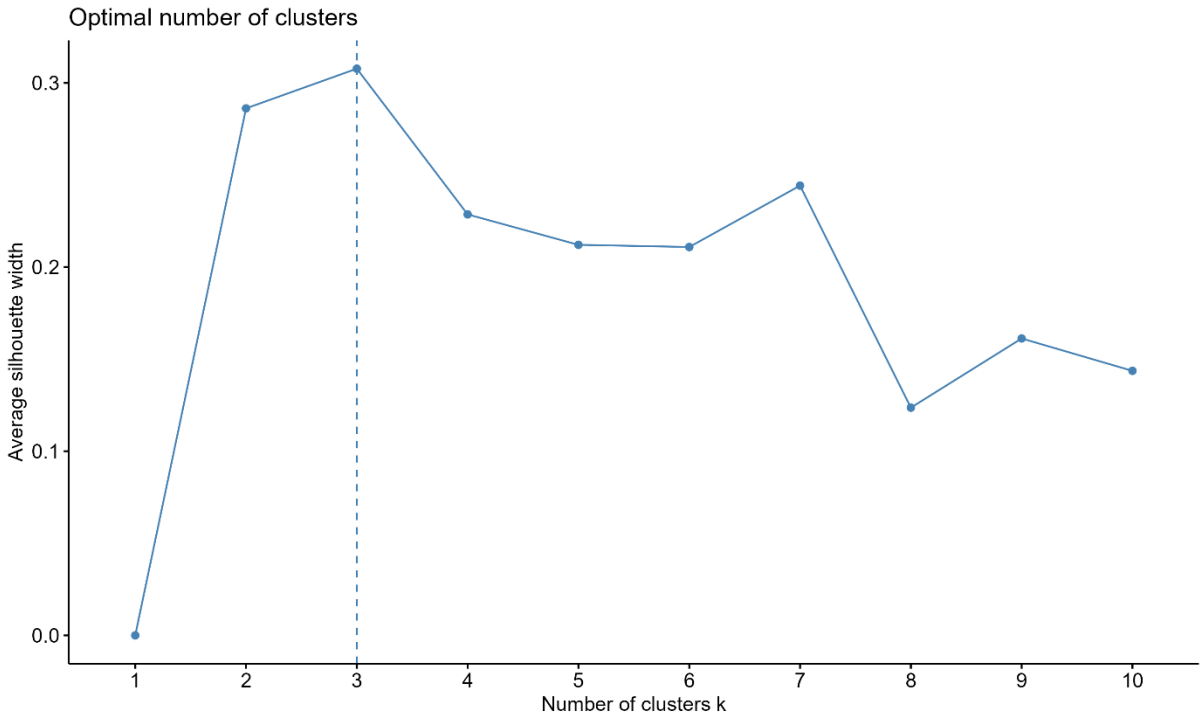
Metoda łokcia

fviz_nbclust(wine_data_scaled, kmeans, method = "wss")



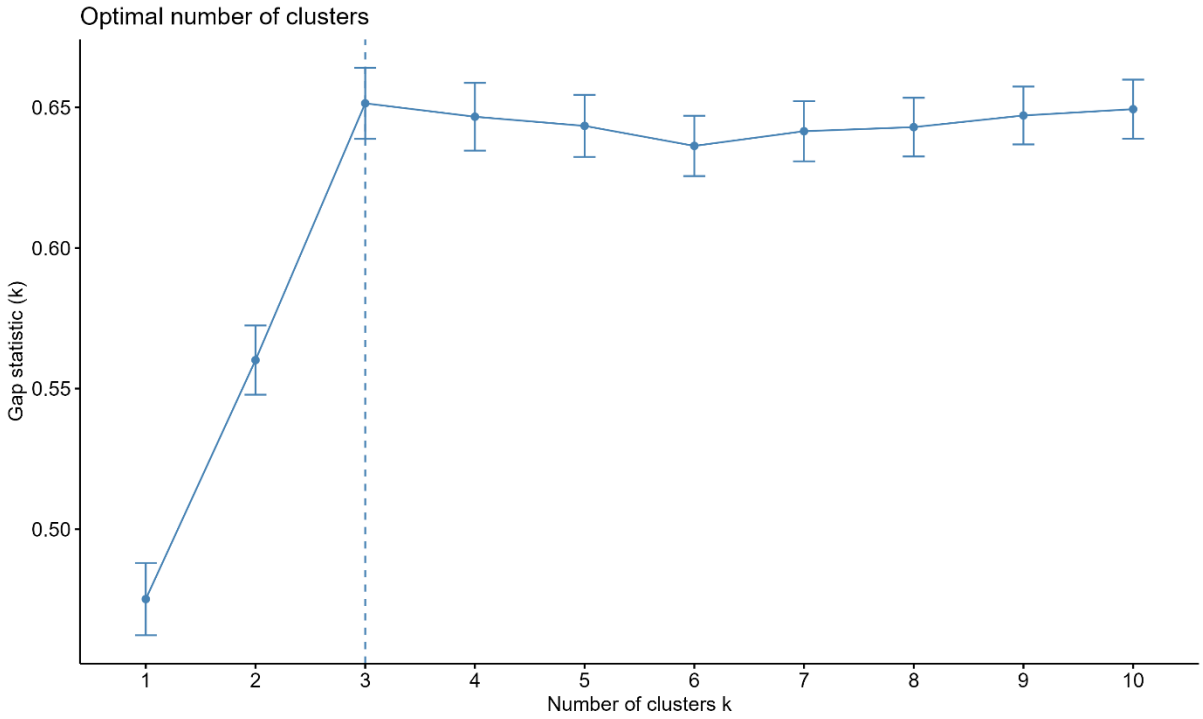
Metoda średniego konturu

fviz_nbclust(wine_data_scaled, kmeans, method = "silhouette")



Statystyka luk

```
set.seed(123)

gap_stat <- clusGap(wine_data_scaled, FUN = kmeans, nstart = 25, K.max = 10, B = 50)

fviz_gap_stat(gap_stat)
```



Wnioski

1. **Metoda łokcia (Elbow Method)**: Wykres pokazuje całkowitą sumę kwadratów wewnątrz klastrów dla różnych liczby klastrów k. Punkt, w którym krzywa zaczyna się wyginać (łokieć) i staje się mniej stroma, wskazuje na optymalną liczbę klastrów. Na załączonym wykresie punkt łokcia wydaje się znajdować przy k=3, co sugeruje, że 3 klastry mogą być odpowiednią liczbą dla tego zbioru danych.
2. **Metoda średniego konturu (Silhouette Method)**: Wykres przedstawia średnią szerokość sylwetki dla różnych liczby klastrów. Wartość ta mierzy, jak dobrze pasuje próbka do swojego klastra (spójność) w porównaniu do innych klastrów (oddzielenie). Wyższa wartość średniej szerokości sylwetki wskazuje na lepsze dopasowanie. Na załączonym wykresie maksymalna średnia szerokość sylwetki pojawia się dla k=2.
3. **Statystyka luk (Gap Statistic)**: Ta metoda porównuje logarytmiczny wskaźnik wewnątrz-sumy kwadratów dla różnych liczby klastrów z ich oczekiwanymi wartościami pod względem danych referencyjnych. Optymalną liczbę klastrów wskazuje największa luka. Na załączonym wykresie widoczny jest wyraźny szczyt dla k=3, co sugeruje, że najlepszą liczbą klastrów może być właśnie 3.

Dyskusja i wybór optymalnej liczby klastrów:

- Dwie z trzech metod (metoda łokcia i statystyka luk) sugerują, że optymalna liczba klastrów wynosi 3.
- Metoda średniego konturu wskazuje na 2 klastry, jednak wartość średniej szerokości sylwetki nie jest znacznie wyższa dla k=2 w porównaniu do k=3.

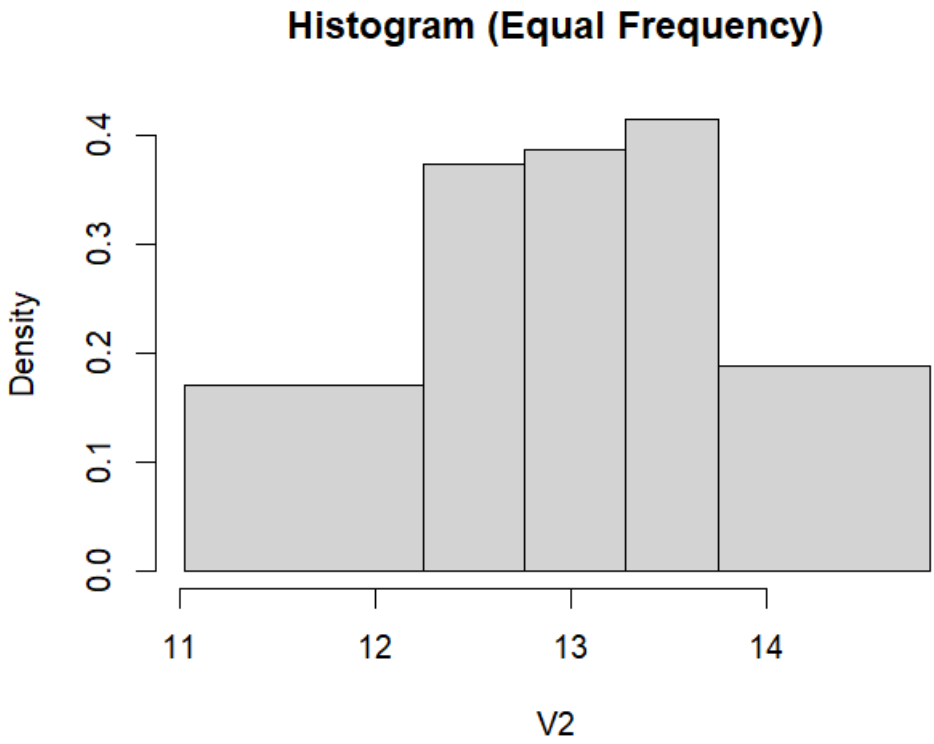
Biorąc pod uwagę powyższe informacje, i zakładając, że liczba klas (ground-truth labels) w zbiorze danych wine wynosi 3 (co jest typowe dla tego zbioru danych, zawierającego klasyfikację win do trzech różnych kultivarów), wybór 3 klastrów wydaje się być najbardziej odpowiedni. Jest to zgodne z rzeczywistą liczbą klas w danych, co dodatkowo potwierdza wybór.

Lista 9

Przeprowadź dyskretyzację wybranej zmiennej numerycznej stosując dwie różne metody: według równej szerokości oraz według równej częstości. a) Dla każdego przypadku (metody) zamieść dwa wykresy: rozkład nowych wartości oraz histogram dla starych wartości wraz z liniami dzielącymi przedziały. b) Porównaj i przedyskutuj wyniki działania obu metod.

```
# Ustawienie ścieżki dostępu do danych

path <- "C:/Users/petit/Desktop/repos/UO/rok 3/Wprowadzenie do
eksploracji danych/lista8"

setwd(path)


# Read the data

wine <- read.csv('wine/wine.data', header = FALSE)


## 1. Dyskretyzacja zmiennej numerycznej


# Załóżmy, że chcesz dyskretyzować drugą kolumnę numeryczną (V2)

variable_to_discretize <- wine[, 2]  # druga kolumna


# Ustal liczbę przedziałów, np. 5

number_of_bins <- 5


# Dyskretyzacja według równej szerokości

equal_width_bins <- cut(variable_to_discretize, breaks =
number_of_bins, include.lowest = TRUE)


# Dyskretyzacja według równej częstości

library(classInt)

freq_breaks <- classIntervals(variable_to_discretize, n =
number_of_bins, style = "quantile")$brks

equal_freq_bins <- cut(variable_to_discretize, breaks = freq_breaks,
include.lowest = TRUE)
```

```
# Wykresy dla dyskretyzacji według równej szerokości

hist(variable_to_discretize, breaks = number_of_bins, main =
"Histogram (Equal Width)", xlab = "V2")

rug(jitter(as.numeric(equal_width_bins)))


# Wykresy dla dyskretyzacji według równej częstości

hist(variable_to_discretize, breaks = freq_breaks, main = "Histogram
(Equal Frequency)", xlab = "V2")

rug(jitter(as.numeric(equal_freq_bins)))
```



Wnioski

1. Słupki histogramu przedstawiają gęstość, a nie liczbę obserwacji. Gęstość jest używana, gdy zmienna została podzielona na przedziały o różnej szerokości.
2. Widoczne są cztery przedziały, co sugeruje, że algorytm wydzielił dane do czterech równolicznych grup. To jest zgodne z liczbą przedziałów określoną w parametrze **number_of_bins**.

3. Środkowe słupki są szersze niż skrajne, co wskazuje, że wartości w środkowych przedziałach są bardziej skupione, a zakresy tych przedziałów są szersze, aby pomieścić podobną liczbę obserwacji, co w innych przedziałach.
4. Szerokość słupków histogramu nie jest jednakowa, co jest charakterystyczne dla dyskretyzacji według równej częstości. Każdy słupek reprezentuje przedział o zbliżonej liczbie obserwacji, ale przedziały te mogą mieć różną szerokość.
5. Nierówna szerokość słupków może wskazywać na to, że rozkład oryginalnej zmiennej nie jest jednorodny. Na przykład, możemy zaobserwować, że dane są bardziej skoncentrowane w określonych zakresach wartości.
6. Ostatni słupek po prawej stronie jest znacznie niższy niż pozostałe, co oznacza, że przedział ten jest szerszy niż pozostałe przedziały, aby zawierał taką samą liczbę obserwacji jak one.

Przeprowadź dyskretyzację wszystkich numerycznych zmiennych w tabeli, stosując wybraną metodę (spośród: "interval", "frequency", "cluster"). a) Wyświetl wartości przed i po dyskretyzacji dla kilku pierwszych obserwacji/wierszy.

```
> ## 2. Dyskretyzacja wszystkich numerycznych zmiennych
>
> # Dyskretyzacja wszystkich numerycznych zmiennych metodą "frequency"
> discretized_wine <- wine
> for(i in 2:ncol(wine)) {
+   discretized_wine[, i] <- cut(wine[, i], breaks = classIntervals(wine[, i], n = number_of_bins, style = "quantile")$brks, include.lowest = TRUE)
+ }
>
> # Wyświetlenie wartości przed i po dyskretyzacji dla kilku pierwszych wierszy
> head(wine)
  V1    V2   V3   V4   V5  V6   V7   V8   V9  V10  V11  V12  V13  V14
1  1 14.23 1.71 2.43 15.6 127 2.80 3.06 0.28 2.29 5.64 1.04 3.92 1065
2  1 13.20 1.78 2.14 11.2 100 2.65 2.76 0.26 1.28 4.38 1.05 3.40 1050
3  1 13.16 2.36 2.67 18.6 101 2.80 3.24 0.30 2.81 5.68 1.03 3.17 1185
4  1 14.37 1.95 2.50 16.8 113 3.85 3.49 0.24 2.18 7.80 0.86 3.45 1480
5  1 13.24 2.59 2.87 21.0 118 2.80 2.69 0.39 1.82 4.32 1.04 2.93  735
6  1 14.20 1.76 2.45 15.2 112 3.27 3.39 0.34 1.97 6.75 1.05 2.85 1450
> head(discretized_wine)
```

V1 V2 V3 V4 V5 V6 V7 V8 V9 V10

1 1 (13.8,14.8] (1.51,1.73] (2.42,2.61] [10.6,16.8] (111,162] (2.53,2.86] (2.98,5.08] (0.26,0.3] (1.99,3.58]

2 1 (12.8,13.3] (1.73,2.13] [1.36,2.18] [10.6,16.8] (94.8,101] (2.53,2.86] (2.46,2.98] [0.13,0.26] (1.1,1.42]

3 1 (12.8,13.3] (2.13,3.41] (2.61,3.23] (18.6,20] (94.8,101] (2.53,2.86] (2.98,5.08] (0.26,0.3] (1.99,3.58]

4 1 (13.8,14.8] (1.73,2.13] (2.42,2.61] [10.6,16.8] (111,162] (2.86,3.88] (2.98,5.08] [0.13,0.26] (1.99,3.58]

5 1 (12.8,13.3] (2.13,3.41] (2.61,3.23] (20,22] (111,162] (2.53,2.86] (2.46,2.98] (0.3,0.39] (1.66,1.99]

6 1 (13.8,14.8] (1.73,2.13] (2.42,2.61] [10.6,16.8] (111,162] (2.86,3.88] (2.98,5.08] (0.3,0.39] (1.66,1.99]

V11 V12 V13 V14

1 (5.28,6.99] (0.91,1.04] (3.26,4] (1.05e+03,1.68e+03]

2 (4.08,5.28] (1.04,1.16] (3.26,4] (1.05e+03,1.68e+03]

3 (5.28,6.99] (0.91,1.04] (2.9,3.26] (1.05e+03,1.68e+03]

4 (6.99,13] (0.74,0.91] (3.26,4] (1.05e+03,1.68e+03]

5 (4.08,5.28] (0.91,1.04] (2.9,3.26] (606,742]

6 (5.28,6.99] (1.04,1.16] (2.52,2.9] (1.05e+03,1.68e+03]

```
>
> # Wyświetlenie wartości przed i po dyskretyzacji dla kilku pierwszych wierszy
> head(wine)
  V1    V2   V3   V4   V5  V6   V7   V8   V9  V10  V11  V12  V13  V14
1  1 14.23 1.71 2.43 15.6 127 2.80 3.06 0.28 2.29 5.64 1.04 3.92 1065
2  1 13.20 1.78 2.14 11.2 100 2.65 2.76 0.26 1.28 4.38 1.05 3.40 1050
3  1 13.16 2.36 2.67 18.6 101 2.80 3.24 0.30 2.81 5.68 1.03 3.17 1185
4  1 14.37 1.95 2.50 16.8 113 3.85 3.49 0.24 2.18 7.80 0.86 3.45 1480
5  1 13.24 2.59 2.87 21.0 118 2.80 2.69 0.39 1.82 4.32 1.04 2.93  735
6  1 14.20 1.76 2.45 15.2 112 3.27 3.39 0.34 1.97 6.75 1.05 2.85 1450
> head(discretized_wine)
  V1          V2          V3          V4          V5          V6          V7          V8          V9          V10
1  1 (13.8,14.8] (1.51,1.73] (2.42,2.61] [10.6,16.8]  (111,162] (2.53,2.86] (2.98,5.08]  (0.26,0.3] (1.99,3.58]
2  1 (12.8,13.3] (1.73,2.13] [1.36,2.18] [10.6,16.8] (94.8,101] (2.53,2.86] (2.46,2.98] [0.13,0.26]  (1.1,1.42]
3  1 (12.8,13.3] (2.13,3.41] (2.61,3.23]   (18.6,20] (94.8,101] (2.53,2.86] (2.98,5.08]  (0.26,0.3] (1.99,3.58]
4  1 (13.8,14.8] (1.73,2.13] (2.42,2.61] [10.6,16.8]  (111,162] (2.86,3.88] (2.98,5.08] [0.13,0.26] (1.99,3.58]
5  1 (12.8,13.3] (2.13,3.41] (2.61,3.23]     (20,22]  (111,162] (2.53,2.86] (2.46,2.98]  (0.3,0.39] (1.66,1.99]
6  1 (13.8,14.8] (1.73,2.13] (2.42,2.61] [10.6,16.8]  (111,162] (2.86,3.88] (2.98,5.08]  (0.3,0.39] (1.66,1.99]
          V11         V12         V13                 V14
1 (5.28,6.99] (0.91,1.04]   (3.26,4] (1.05e+03,1.68e+03]
2 (4.08,5.28] (1.04,1.16]   (3.26,4] (1.05e+03,1.68e+03]
3 (5.28,6.99] (0.91,1.04] (2.9,3.26] (1.05e+03,1.68e+03]
4   (6.99,13] (0.74,0.91]   (3.26,4] (1.05e+03,1.68e+03]
5 (4.08,5.28] (0.91,1.04] (2.9,3.26]          (606,742]
6 (5.28,6.99] (1.04,1.16] (2.52,2.9] (1.05e+03,1.68e+03]
>
```

Omówienie wyników

1. **Zakresy przedziałów**: Oryginalne wartości dla V2 pokazują dokładne pomiary zawartości alkoholu, podczas gdy po dyskretyzacji zostały one zgrupowane w kategorie. Na przykład, wartość 14.23 została zaklasyfikowana do przedziału (13.8, 14.8], co oznacza, że wszystkie wina z zawartością alkoholu w tym przedziale będą traktowane jako identyczne pod tym względem w dalszej analizie.
2. **Strata informacji**: Proces dyskretyzacji nieuchronnie prowadzi do pewnego stopnia utraty informacji. Zamiast precyzyjnej wartości, mamy teraz kategorię, która jest mniej szczegółowa. To może być korzystne dla pewnych rodzajów analiz, takich jak analiza reguł asocjacyjnych, ale mniej korzystne dla innych, które wymagają większej precyzji.
3. **Uproszczenie analizy**: Kategoryzacja danych może uprościć niektóre analizy statystyczne i ułatwić wizualizację oraz interpretację danych. Na przykład, moglibyśmy łatwo porównać liczby win z różnych przedziałów zawartości alkoholu.
4. **Wpływ na modele**: Dyskretyzacja może wpływać na wydajność modeli predykcyjnych. Dla niektórych algorytmów uczenia maszynowego, takich jak drzewa decyzyjne, dyskretyzacja może być korzystna, podczas gdy dla innych, takich jak regresja liniowa, może to negatywnie wpłynąć na wydajność modelu.

3. Do danych w tabeli po dyskretyzacji zastosuj algorytm A priori, podając zadane wartości minimalnego wsparcia i ufności. a) Napisz, ile reguł zostało wygenerowanych? b) Wyświetl kilka pierwszych reguł, posortowanych według miary lift. c) Zilustruj je na wykresie. d) Zinterpretuj "w języku naturalnym" pierwszą regułę. e) Przedyskutuj uzyskane wyniki.

3. Algorytm A priori

```
# Parametry algorytmu A priori
min_support <- 0.1
min_confidence <- 0.8

library(arules)
data <- as(discretized_wine, "transactions")
rules <- apriori(data, parameter = list(support = min_support, confidence = min_confidence))

# Ilość wygenerowanych reguł
length(rules)
```

Sortowanie reguł według miary lift

rules_sorted <- sort(rules, by = "lift", decreasing = TRUE)

Wyświetlenie kilku pierwszych reguł

head(rules_sorted)

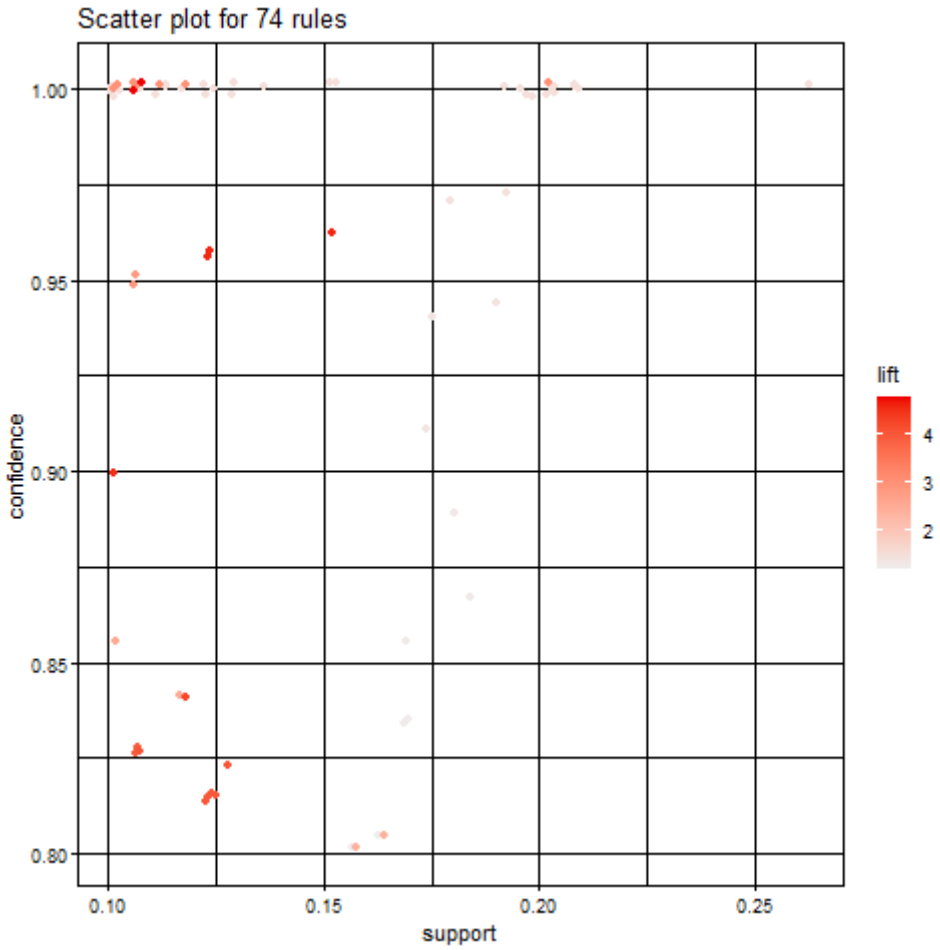
plot(rules, method = "scatter", measure = c("support", "confidence"))

plot(rules, method = "scatter", measure = c("lift", "support"))

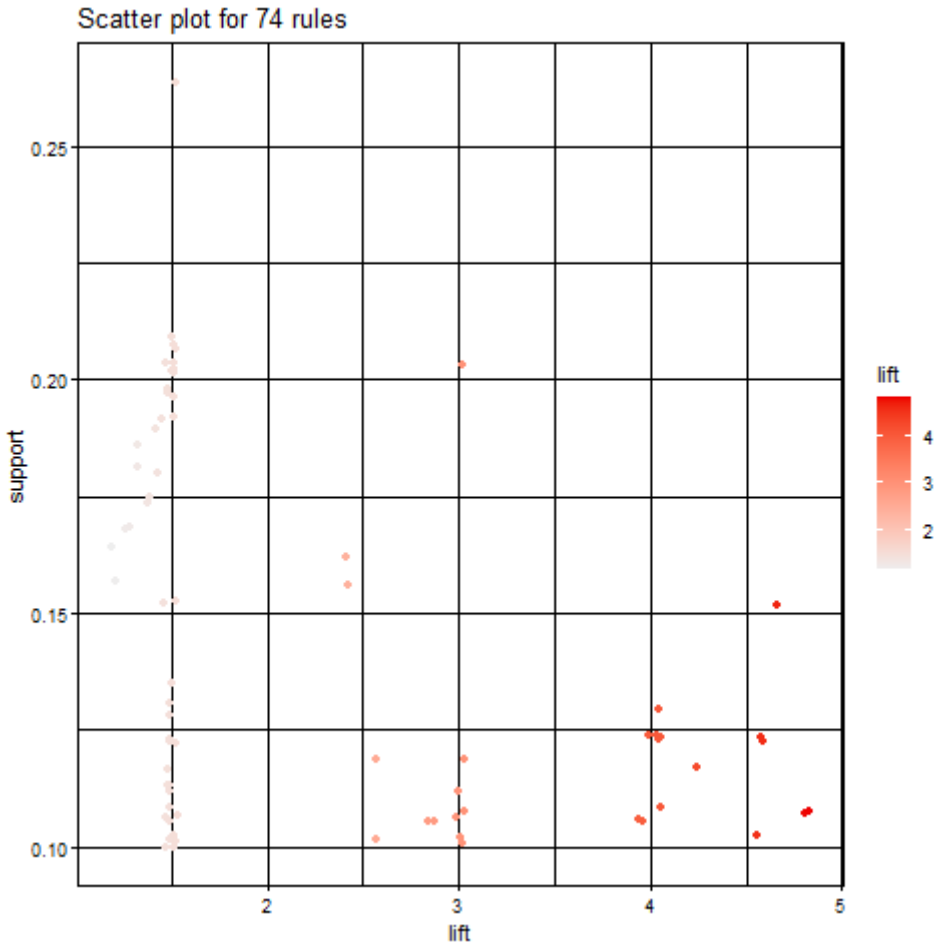
plot(rules, method = "scatter", measure = c("lift", "confidence"))

```
  Only unique breaks are used reducing the number of intervals. Look at ? discretize for details.
> rules <- apriori(data, parameter = list(support = min_support, confidence = min_confidence))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target  ext
        0.8    0.1    1 none FALSE            TRUE       5     0.1      1     10  rules TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 17

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[67 item(s), 178 transaction(s)] done [0.00s].
sorting and recoding items ... [67 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 done [0.00s].
writing ... [74 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
>
```

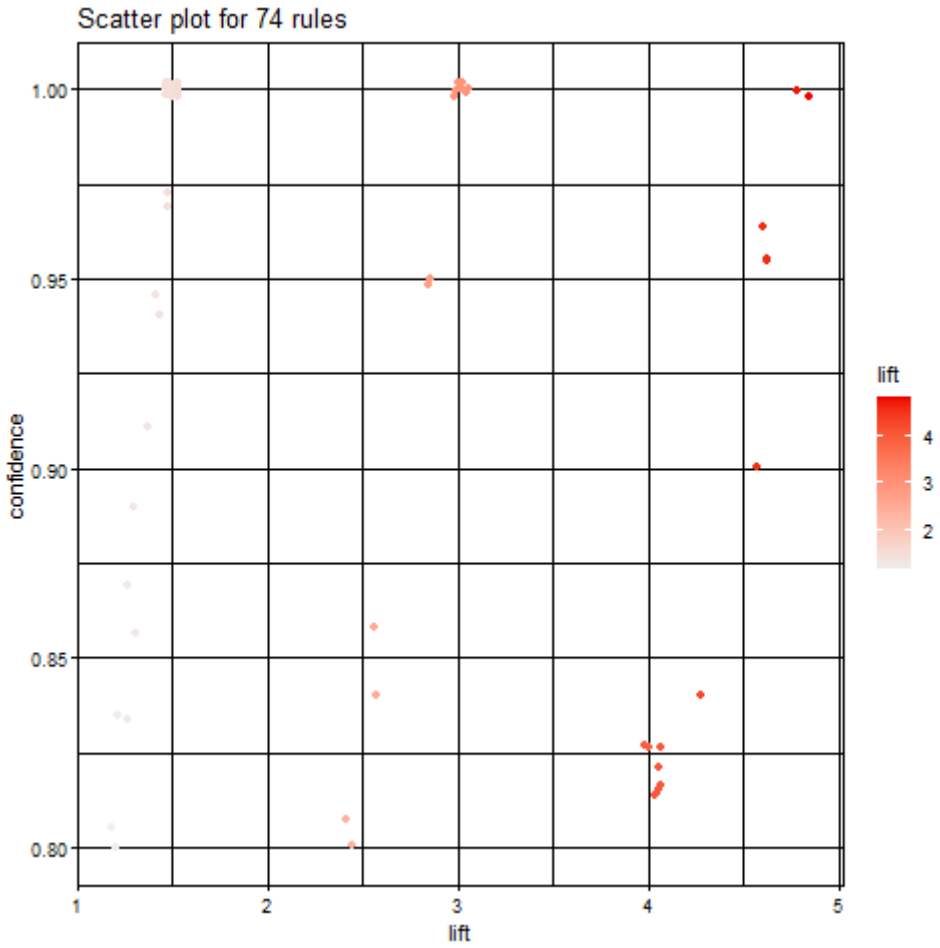
```
writing ... [74 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
>
> # Ilość wygenerowanych reguł
> length(rules)
[1] 74
>
> # Sortowanie reguł według miary lift
> rules_sorted <- sort(rules, by = "lift", decreasing = TRUE)
>
> # Wyświetlenie kilku pierwszych reguł
> head(rules_sorted)
set of 6 rules
> plot(rules, method = "scatter", measure = c("support", "confidence"))
To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.
> plot(rules, method = "scatter", measure = c("lift", "support"))
```



Plot 1 wykres_wsparcie_vs_ufnosc



Plot 2 wykres_lift_vs_wsparcie



Plot 3 wykres_lift_vs_ufnosc

lista10

Lista 10

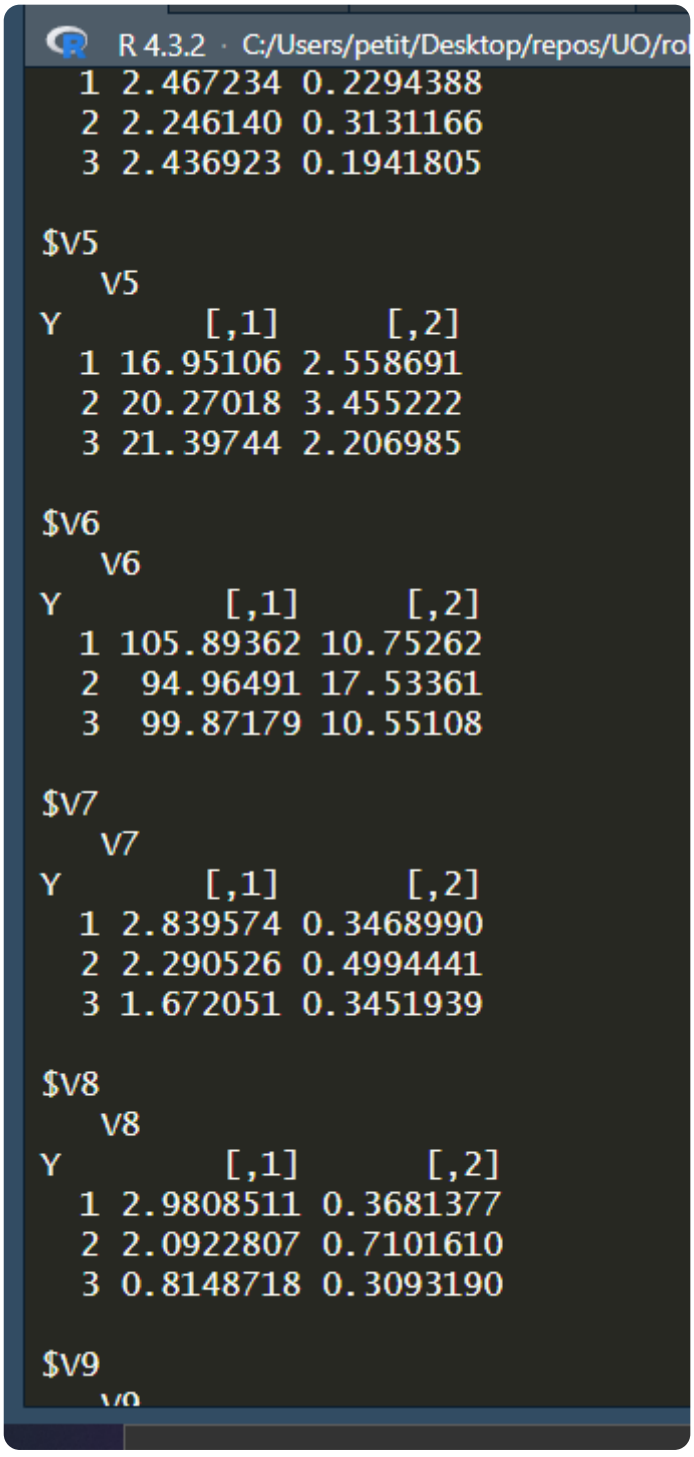
a) Utwórz zbiory: treningowy zawierający 80% danych oraz testowy zawierający 20% danych.

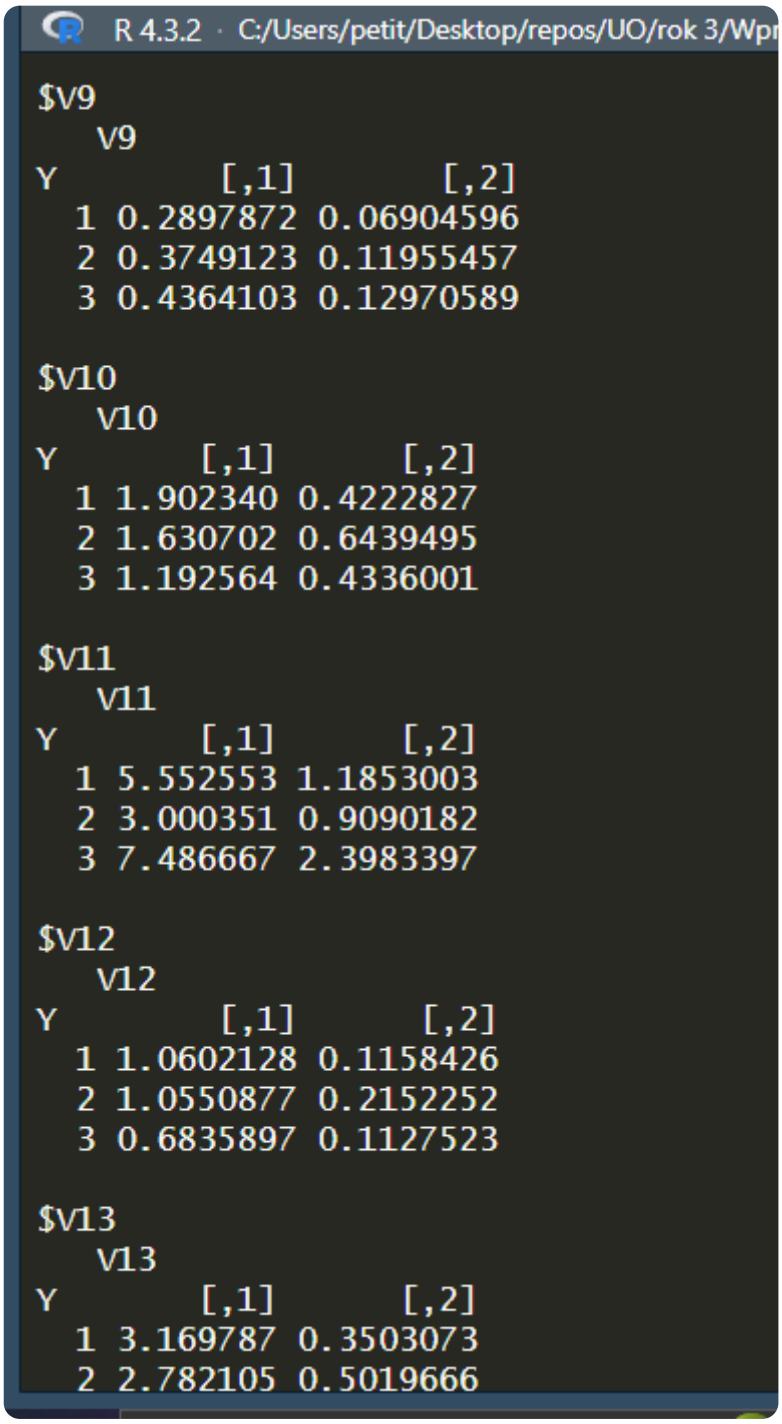
```r
# Ustawienie ziarna losowości dla powtarzalności wyników
set.seed(123)

# Podział danych na zbiór treningowy i testowy
splitIndex <- createDataPartition(wine_data$V1, p = 0.8, list = FALSE)
train_set <- wine_data[splitIndex, ]
test_set <- wine_data[-splitIndex, ]
```

b) Zbuduj naiwny klasyfikator bayesowski (na zbiorze treningowym). Wyświetl rozkład klas dla zmiennej celu oraz listę tabel prawdopodobieństw warunkowych.

```r
# Budowanie naiwnego klasyfikatora bayesowskiego
nb_model <- naiveBayes(V1 ~ ., data = train_set)

# Wyświetlenie rozkładu klas zmiennej celu
print(nb_model$class)

# Wyświetlenie tabel prawdopodobieństw warunkowych dla każdej zmiennej
print(nb_model$tables)
```

```
>
> # Wyświetlenie tabel prawdopodobieństw warunkowych dla każdej zmiennej
> print(nb_model$tables)
$V2
   V2
Y       [,1]      [,2]
  1 13.78702 0.4583473
  2 12.24895 0.5278145
  3 13.20974 0.5384895

$V3
   V3
Y       [,1]      [,2]
  1 2.067021 0.7476948
  2 2.021053 1.0957298
  3 3.278718 1.1461358

$V4
   V4
Y       [,1]      [,2]
  1 2.467234 0.2294388
  2 2.246140 0.3131166
  3 2.436923 0.1941805
```


R 4.3.2 · C:/Users/petit/Desktop/repos/UO/ro
1 2.467234 0.2294388
2 2.246140 0.3131166
3 2.436923 0.1941805
$V5
V5
Y [,1] [,2]
1 16.95106 2.558691
2 20.27018 3.455222
3 21.39744 2.206985
$V6
V6
Y [,1] [,2]
1 105.89362 10.75262
2 94.96491 17.53361
3 99.87179 10.55108
$V7
V7
Y [,1] [,2]
1 2.839574 0.3468990
2 2.290526 0.4994441
3 1.672051 0.3451939
$V8
V8
Y [,1] [,2]
1 2.9808511 0.3681377
2 2.0922807 0.7101610
3 0.8148718 0.3093190
$V9


R 4.3.2 · C:/Users/petit/Desktop/repos/UO/rok 3/Wpr
$V9
V9
Y [,1] [,2]
1 0.2897872 0.06904596
2 0.3749123 0.11955457
3 0.4364103 0.12970589
$V10
V10
Y [,1] [,2]
1 1.902340 0.4222827
2 1.630702 0.6439495
3 1.192564 0.4336001
$V11
V11
Y [,1] [,2]
1 5.552553 1.1853003
2 3.000351 0.9090182
3 7.486667 2.3983397
$V12
V12
Y [,1] [,2]
1 1.0602128 0.1158426
2 1.0550877 0.2152252
3 0.6835897 0.1127523
$V13
V13
Y [,1] [,2]
1 3.169787 0.3503073
2 2.782105 0.5019666

```
$V13
   V13
Y       [,1]      [,2]
  1 3.169787 0.3503073
  2 2.782105 0.5019666
  3 1.664872 0.2755461

$V14
   V14
Y        [,1]     [,2]
  1 1108.0213 212.6916
  2  514.0702 158.9408
  3  626.5385 108.4530

>
```

c) Przeprowadź klasyfikację danych metodą naiwnego klasyfikatora bayesowskiego (na zbiorze testowym). Wyświetl macierz błędów.

```r
# Przewidywanie na zbiorze testowym
predictions <- predict(nb_model, test_set)

# Obliczenie i wyświetlenie macierzy błędów
confusionMatrix <- table(test_set$V1, predictions)
print(confusionMatrix)
```

```
>
> # Obliczenie i wyświetlenie macierzy błędów
> confusionMatrix <- table(test_set$V1, predictions)
> print(confusionMatrix)
   predictions
     1  2  3
  1 12  0  0
  2  0 13  1
  3  0  0  9
> # Załadowanie potrzebnej biblioteki
> library(caret)
```

d) Napisz, jaka jest dokładność klasyfikacji i współczynnik błędu

Na podstawie macierzy błędów można obliczyć dokładność klasyfikacji oraz współczynnik błędu. Dokładność klasyfikacji to stosunek liczby poprawnie sklasyfikowanych przypadków do całkowitej liczby przypadków, a

współczynnik błędu to stosunek liczby błędnie sklasyfikowanych przypadków do całkowitej liczby przypadków.

Macierz błędów wygląda następująco:

```
   predictions
     1  2  3
1 12  0  0
2  0 13  1
3  0  0  9
```

Obliczmy teraz dokładność i współczynnik błędu:

1. **Dokładność (Accuracy)**: Jest to `(poprawne przypisania klasy 1 + poprawne przypisania klasy 2 + poprawne przypisania klasy 3) / całkowita liczba przypadków`
   - W naszym przypadku: `(12 + 13 + 9) / (12 + 13 + 1 + 9)`
2. **Współczynnik błędu (Error Rate)**: Jest to `1 - dokładność`
   - Można to również obliczyć jako `(błędne przypisania) / całkowita liczba przypadków`

Obliczymy te wartości:

Dokładność klasyfikacji wynosi około (97.14%) a współczynnik błędu to około (2.86%). Oznacza to, że model naiwnego klasyfikatora bayesowskiego bardzo dobrze radzi sobie z klasyfikacją na zbiorze testowym.

e) Zbuduj cztery modele SVM z różnymi funkcjami jądra: radialną, liniową, wielomianową i sigmoidalną (pamiętaj o standaryzacji danych).

```r
# Standaryzacja danych treningowych i testowych
train_set_standardized <- scale(train_set[, -1])
test_set_standardized <- scale(test_set[, -1], center = attr(train_set_standardized, "scaled:center"), s

# Dodanie kolumny z etykietami klasy do zbiorów danych
train_set_standardized <- data.frame(train_set$V1, train_set_standardized)
test_set_standardized <- data.frame(test_set$V1, test_set_standardized)
colnames(train_set_standardized)[1] <- "V1"
colnames(test_set_standardized)[1] <- "V1"

# Budowanie modeli SVM z różnymi funkcjami jądra
svm_radial <- svm(V1 ~ ., data = train_set_standardized, kernel = "radial")
svm_linear <- svm(V1 ~ ., data = train_set_standardized, kernel = "linear")
svm_polynomial <- svm(V1 ~ ., data = train_set_standardized, kernel = "polynomial")
svm_sigmoid <- svm(V1 ~ ., data = train_set_standardized, kernel = "sigmoid")

# Wyświetlenie informacji o modelach
summary(svm_radial)
summary(svm_linear)
summary(svm_polynomial)
summary(svm_sigmoid)
```

```
> # Wyświetlenie informacji o modelach
> summary(svm_radial)

Call:
svm(formula = V1 ~ ., data = train_set_standardized, kernel = "radial")


Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.07692308
    epsilon:  0.1


Number of Support Vectors:  84
```

```
> summary(svm_linear)

Call:
svm(formula = V1 ~ ., data = train_set_standardized, kernel = "linear")


Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  linear
       cost:  1
      gamma:  0.07692308
    epsilon:  0.1


Number of Support Vectors:  107
```

```
> summary(svm_polynomial)

Call:
svm(formula = V1 ~ ., data = train_set_standardized, kernel = "polynomial")


Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  polynomial
       cost:  1
     degree:  3
      gamma:  0.07692308
     coef.0:  0
    epsilon:  0.1


Number of Support Vectors:  112
```

```
> summary(svm_sigmoid)

Call:
svm(formula = V1 ~ ., data = train_set_standardized, kernel = "sigmoid")


Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  sigmoid
       cost:  1
      gamma:  0.07692308
     coef.0:  0
    epsilon:  0.1


Number of Support Vectors:  134
```

f) Dla każdego modelu wydrukuj parametry i liczbę wektorów nośnych

```r
# Wyświetlenie parametrów i liczby wektorów nośnych dla każdego modelu SVM

# Model SVM z funkcją jądra radialną
cat("Model SVM z funkcją jądra radialną:\n")
print(summary(svm_radial))
cat("Liczba wektorów nośnych:", length(svm_radial$index), "\n\n")

# Model SVM z funkcją jądra liniową
cat("Model SVM z funkcją jądra liniową:\n")
print(summary(svm_linear))
cat("Liczba wektorów nośnych:", length(svm_linear$index), "\n\n")

# Model SVM z funkcją jądra wielomianową
cat("Model SVM z funkcją jądra wielomianową:\n")
print(summary(svm_polynomial))
cat("Liczba wektorów nośnych:", length(svm_polynomial$index), "\n\n")

# Model SVM z funkcją jądra sigmoidalną
cat("Model SVM z funkcją jądra sigmoidalną:\n")
print(summary(svm_sigmoid))
cat("Liczba wektorów nośnych:", length(svm_sigmoid$index), "\n\n")
```

```
R 4.3.2 · C:/Users/petit/Desktop/repos/UO/rok 3/Wprowadzenie do eksploracji danych/lista12/
> # Wyświetlenie parametrów i liczby wektorów nośnych dla każdego modelu SVM
>
> # Model SVM z funkcją jądra radialną
> cat("Model SVM z funkcją jądra radialną:\n")
Model SVM z funkcją jądra radialną:
> print(summary(svm_radial))

Call:
svm(formula = V1 ~ ., data = train_set_standardized, kernel = "radial")


Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.07692308
    epsilon:  0.1


Number of Support Vectors:  84
```

```
> cat("Liczba wektorów nośnych:", length(svm_radial$index), "\n\n")
Liczba wektorów nośnych: 84

>
> # Model SVM z funkcją jądra liniową
> cat("Model SVM z funkcją jądra liniową:\n")
Model SVM z funkcją jądra liniową:
> print(summary(svm_linear))

Call:
svm(formula = V1 ~ ., data = train_set_standardized, kernel = "linear")


Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  linear
       cost:  1
      gamma:  0.07692308
    epsilon:  0.1


Number of Support Vectors:  107
```

```
> cat("Liczba wektorów nośnych:", length(svm_linear$index), "\n\n")
Liczba wektorów nośnych: 107

>
> # Model SVM z funkcją jądra wielomianową
> cat("Model SVM z funkcją jądra wielomianową:\n")
Model SVM z funkcją jądra wielomianową:
> print(summary(svm_polynomial))

Call:
svm(formula = V1 ~ ., data = train_set_standardized, kernel = "polynomial")


Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  polynomial
       cost:  1
     degree:  3
      gamma:  0.07692308
     coef.0:  0
    epsilon:  0.1


Number of Support Vectors:  112
```

```
> cat("Liczba wektorów nośnych:", length(svm_polynomial$index), "\n\n")
Liczba wektorów nośnych: 112

>
> # Model SVM z funkcją jądra sigmoidalną
> cat("Model SVM z funkcją jądra sigmoidalną:\n")
Model SVM z funkcją jądra sigmoidalną:
> print(summary(svm_sigmoid))

Call:
svm(formula = V1 ~ ., data = train_set_standardized, kernel = "sigmoid")


Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  sigmoid
       cost:  1
      gamma:  0.07692308
     coef.0:  0
    epsilon:  0.1


Number of Support Vectors:  134
```

```
> cat("Liczba wektorów nośnych:", length(svm_sigmoid$index), "\n\n")
Liczba wektorów nośnych: 134

>
```

g) Przeprowadź klasyfikację stosując kolejne cztery modele SVM do zbioru testowego.

```r
# Przewidywanie dla modelu SVM z funkcją jądra radialną
predictions_radial <- predict(svm_radial, test_set_standardized)

# Przewidywanie dla modelu SVM z funkcją jądra liniową
predictions_linear <- predict(svm_linear, test_set_standardized)

# Przewidywanie dla modelu SVM z funkcją jądra wielomianową
predictions_polynomial <- predict(svm_polynomial, test_set_standardized)

# Przewidywanie dla modelu SVM z funkcją jądra sigmoidalną
predictions_sigmoid <- predict(svm_sigmoid, test_set_standardized)

# Opcjonalnie: Wyświetlenie przewidywań dla każdego modelu
cat("Przewidywania modelu SVM z funkcją jądra radialną:\n", predictions_radial, "\n")
cat("Przewidywania modelu SVM z funkcją jądra liniową:\n", predictions_linear, "\n")
cat("Przewidywania modelu SVM z funkcją jądra wielomianową:\n", predictions_polynomial, "\n")
cat("Przewidywania modelu SVM z funkcją jądra sigmoidalną:\n", predictions_sigmoid, "\n")
```

```
R 4.3.2 · C:/Users/petit/Desktop/repos/UO/rok 3/Wprowadzenie do eksploracji danych/lista12/
> # g) Przeprowadź klasyfikację stosując kolejne cztery modele SVM do zbioru testowego.
> # Przewidywanie dla modelu SVM z funkcją jądra radialną
> predictions_radial <- predict(svm_radial, test_set_standardized)
>
> # Przewidywanie dla modelu SVM z funkcją jądra liniową
> predictions_linear <- predict(svm_linear, test_set_standardized)
>
> # Przewidywanie dla modelu SVM z funkcją jądra wielomianową
> predictions_polynomial <- predict(svm_polynomial, test_set_standardized)
>
> # Przewidywanie dla modelu SVM z funkcją jądra sigmoidalną
> predictions_sigmoid <- predict(svm_sigmoid, test_set_standardized)
>
> # Opcjonalnie: Wyświetlenie przewidywań dla każdego modelu
> cat("Przewidywania modelu SVM z funkcją jądra radialną:\n", predictions_radial, "\n")
Przewidywania modelu SVM z funkcją jądra radialną:
 0.9253141 1.154281 1.116268 0.9513025 1.014082 1.273846 1.161947 1.217867 0.9674315 1.197292 1.060859 1.013441 2.092303
1.563645 1.844207 2.470899 2.195967 2.018954 1.964491 2.091244 2.341307 2.098257 1.905313 1.980307 2.075071 1.807855 2.76
1411 2.755826 3.040487 2.789695 2.861382 2.910758 3.091882 2.986817 2.994976
> cat("Przewidywania modelu SVM z funkcją jądra liniową:\n", predictions_linear, "\n")
Przewidywania modelu SVM z funkcją jądra liniową:
 0.8094901 0.9331381 1.540173 0.7950694 1.085017 0.4979976 1.264076 1.376328 1.159678 1.302604 1.492177 0.9374362 1.81126
9 1.682841 1.475441 2.337042 2.19202 2.061338 2.125274 2.108927 2.556291 2.119183 1.796703 1.917395 2.146846 1.481126 2.8
58266 3.148599 3.031552 3.009251 3.11184 2.726811 2.693352 3.043791 2.834147
> cat("Przewidywania modelu SVM z funkcją jądra wielomianową:\n", predictions_polynomial, "\n")
Przewidywania modelu SVM z funkcją jądra wielomianową:
 0.7959544 1.212769 1.605373 0.5229623 1.312196 -0.1003526 1.547861 1.578948 1.622152 1.440796 1.561803 1.229071 1.721158
1.751412 1.315444 2.086959 1.843153 2.257362 1.919908 1.928508 2.38733 1.991347 1.866106 1.957231 1.85431 1.490032 2.3355
74 4.66244 3.058929 3.15406 2.819389 2.52278 2.661924 2.840083 2.610935
> cat("Przewidywania modelu SVM z funkcją jądra sigmoidalną:\n", predictions_sigmoid, "\n")
Przewidywania modelu SVM z funkcją jądra sigmoidalną:
 1.391144 0.6549497 1.165122 1.601402 0.5623435 0.1709588 1.193469 1.448903 1.222446 1.460712 1.17915 1.021778 0.7971102
1.39472 0.2545922 2.938465 2.06291 1.399216 2.830091 1.293409 2.496483 2.479882 1.63427 3.036401 2.657165 1.218626 2.7784
17 1.505867 2.074757 2.733533 2.401592 2.381235 2.639565 2.36698 3.594083
>
```

h) Dla każdego modelu wydrukuj macierz błędów oraz dokładność.

```r
# Funkcja do obliczenia i wydrukowania macierzy błędów oraz dokładności
print_confusion_matrix_and_accuracy <- function(predictions, actual) {
  confusion_matrix <- table(actual, predictions)
  accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
  cat("Macierz błędów:\n")
  print(confusion_matrix)
  cat("Dokładność:", accuracy, "\n\n")
}

# Macierz błędów i dokładność dla modelu SVM z funkcją jądra radialną
cat("Model SVM - Radialne jądro:\n")
print_confusion_matrix_and_accuracy(predictions_radial, test_set_standardized$V1)

# Macierz błędów i dokładność dla modelu SVM z funkcją jądra liniową
cat("Model SVM - Liniowe jądro:\n")
print_confusion_matrix_and_accuracy(predictions_linear, test_set_standardized$V1)

# Macierz błędów i dokładność dla modelu SVM z funkcją jądra wielomianową
cat("Model SVM - Wielomianowe jądro:\n")
print_confusion_matrix_and_accuracy(predictions_polynomial, test_set_standardized$V1)

# Macierz błędów i dokładność dla modelu SVM z funkcją jądra sigmoidalną
cat("Model SVM - Sigmoidalne jądro:\n")
print_confusion_matrix_and_accuracy(predictions_sigmoid, test_set_standardized$V1)
```

```
R 4.3.2 · C:/Users/petit/Desktop/repos/UO/rok 3/Wprowadzenie do eksploracji danych/lista12/
> # h) Dla każdego modelu wydrukuj macierz błędów oraz dokładność.
> # Funkcja do obliczenia i wydrukowania macierzy błędów oraz dokładności
> print_confusion_matrix_and_accuracy <- function(predictions, actual) {
+   confusion_matrix <- table(actual, predictions)
+   accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
+   cat("Macierz błędów:\n")
+   print(confusion_matrix)
+   cat("Dokładność:", accuracy, "\n\n")
+ }
>
> # Macierz błędów i dokładność dla modelu SVM z funkcją jądra radialną
> cat("Model SVM - Radialne jądro:\n")
Model SVM - Radialne jądro:
> print_confusion_matrix_and_accuracy(predictions_radial, test_set_standardized$V1)
Macierz błędów:
      predictions
actual 0.925314061917655 0.951302463591675 0.96743145550474 1.01344130443722 1.01408157518439 1.0608589829982
     1                 1                 1                1                1                1                1
     2                 0                 0                0                0                0                0
     3                 0                 0                0                0                0                0
      predictions
actual 1.11626807135943 1.15428105285456 1.1619472632833 1.19729208602257 1.21786676979851 1.27384556398085
     1                1                1               1                1                1                1
     2                0                0               0                0                0                0
     3                0                0               0                0                0                0
      predictions
actual 1.56364466459154 1.80785530184693 1.84420742657521 1.90531348150749 1.96449129738224 1.98030693579788
     1                0                0                0                0                0                0
     2                1                1                1                1                1                1
     3                0                0                0                0                0                0
      predictions
actual 2.01895400822711 2.07507105952655 2.09124389654516 2.09230255020784 2.09825707102956 2.19596726221885
     1                0                0                0                0                0                0
     2                1                1                1                1                1                1
     3                0                0                0                0                0                0
      predictions
actual 2.34130718226162 2.47089936787415 2.75582637147352 2.7614109322342 2.7896947717802 2.86138166055648
     1                0                0                0                0                0                0
     2                1                1                0                0                0                0
     3                0                0                1                1                1                1
      predictions
actual 2.91075817075212 2.98681695850260 3.00407566236205 3.04048600060770 3.09188187752436
```

```
Console  Terminal  Background Jobs
R 4.3.2 · C:/Users/petit/Desktop/repos/UO/rok 3/Wprowadzenie do eksploracji danych/lista12/
     3                 0                 0                 0                 0                 0                 0
      predictions
actual 1.11626807135943 1.15428105285456 1.1619472632833 1.19729208602257 1.21786676979851 1.27384556398085
     1                 1                 1                 1                 1                 1
     2                 0                 0                 0                 0                 0
     3                 0                 0                 0                 0                 0
      predictions
actual 1.56364466459154 1.80785530184693 1.84420742657521 1.90531348150749 1.96449129738224 1.98030693579788
     1                 0                 0                 0                 0                 0
     2                 1                 1                 1                 1                 1
     3                 0                 0                 0                 0                 0
      predictions
actual 2.01895400822711 2.07507105952655 2.09124389654516 2.09230255020784 2.09825707102956 2.19596726221885
     1                 0                 0                 0                 0                 0
     2                 1                 1                 1                 1                 1
     3                 0                 0                 0                 0                 0
      predictions
actual 2.34130718226162 2.47089936787415 2.75582637147352 2.7614109322342 2.7896947717802 2.86138166055648
     1                 0                 0                 0                 0                 0
     2                 1                 1                 0                 0                 0
     3                 0                 0                 1                 1                 1
      predictions
actual 2.91075817975212 2.98681695859269 2.99497566236295 3.04048690069779 3.09188187752436
     1                 0                 0                 0                 0                 0
     2                 0                 0                 0                 0                 0
     3                 1                 1                 1                 1                 1
Dokładność: 0.02857143

>
> # Macierz błędów i dokładność dla modelu SVM z funkcją jądra liniową
> cat("Model SVM - Liniowe jądro:\n")
Model SVM - Liniowe jądro:
> print_confusion_matrix_and_accuracy(predictions_linear, test_set_standardized$V1)
Macierz błędów:
      predictions
actual 0.49799764361693 0.795069388465165 0.809490130475751 0.933138093878224 0.937436168452605 1.0850169541715
     1                 1                 1                 1                 1                 1
     2                 0                 0                 0                 0                 0                 0
     3                 0                 0                 0                 0                 0                 0
      predictions
actual 1.15967791577823 1.26407627341993 1.30260415382717 1.37632810770321 1.47544064048216 1.48112621470047
```

```
R 4.3.2 · C:/Users/petit/Desktop/repos/UO/rok 3/Wprowadzenie do eksploracji danych/lista12/
>
> # Macierz błędów i dokładność dla modelu SVM z funkcją jądra liniową
> cat("Model SVM - Liniowe jądro:\n")
Model SVM - Liniowe jądro:
> print_confusion_matrix_and_accuracy(predictions_linear, test_set_standardized$V1)
Macierz błędów:
      predictions
actual 0.49799764361693 0.795069388465165 0.809490130475751 0.933138093878224 0.937436168452605 1.0850169541715
     1                 1                 1                 1                 1                 1                 1
     2                 0                 0                 0                 0                 0                 0
     3                 0                 0                 0                 0                 0                 0
      predictions
actual 1.15967791577823 1.26407627341993 1.30260415382717 1.37632810770321 1.47544064048216 1.48112621470047
     1                 1                 1                 1                 1                 0                 0
     2                 0                 0                 0                 0                 1                 1
     3                 0                 0                 0                 0                 0                 0
      predictions
actual 1.49217719560556 1.54017342467057 1.68284134547141 1.79670329278938 1.81126901629309 1.91739528566033
     1                 1                 1                 0                 0                 0                 0
     2                 0                 0                 1                 1                 1                 1
     3                 0                 0                 0                 0                 0                 0
      predictions
actual 2.06133752237619 2.10892712485623 2.11918311301058 2.12527376734032 2.14684602288568 2.19201995082862
     1                 0                 0                 0                 0                 0                 0
     2                 1                 1                 1                 1                 1                 1
     3                 0                 0                 0                 0                 0                 0
      predictions
actual 2.33704193038442 2.55629087695542 2.69335178463739 2.72681149474552 2.83414677761992 2.85826646629042
     1                 0                 0                 0                 0                 0                 0
     2                 1                 1                 0                 0                 0                 0
     3                 0                 0                 1                 1                 1                 1
      predictions
actual 3.00925057736632 3.03155195050385 3.04379132580285 3.11184040364247 3.14859914313101
     1                 0                 0                 0                 0                 0
     2                 0                 0                 0                 0                 0
     3                 1                 1                 1                 1                 1
Dokładność: 0.02857143

>
```

```
R 4.3.2 · C:/Users/petit/Desktop/repos/UO/rok 3/Wprowadzenie do eksploracji danych/lista12/
>
> # Macierz błędów i dokładność dla modelu SVM z funkcją jądra wielomianową
> cat("Model SVM - Wielomianowe jądro:\n")
Model SVM - Wielomianowe jądro:
> print_confusion_matrix_and_accuracy(predictions_polynomial, test_set_standardized$V1)
Macierz błędów:
      predictions
actual -0.100352553066397 0.522962307178777 0.795954419714872 1.2127694128961 1.22907069965322 1.31219636755905
     1                  1                 1                 1               1                1               1
     2                  0                 0                 0               0                0               0
     3                  0                 0                 0               0                0               0
      predictions
actual 1.31544418289391 1.44079553035765 1.49003179725031 1.54786120650768 1.56180336482893 1.5789476361425
     1                0                1                0               1               1              1
     2                1                0                1               0               0              0
     3                0                0                0               0               0              0
      predictions
actual 1.60537252674978 1.62215177270746 1.72115849764271 1.75141222355971 1.84315275758786 1.85431025868786
     1                1                1                0               0               0              0
     2                0                0                1               1               1              1
     3                0                0                0               0               0              0
      predictions
actual 1.8661064205687 1.91990826870208 1.9285076029051 1.95723109710148 1.99134703067364 2.08695898667021
     1               0                0               0               0               0              0
     2               1                1               1               1               1              1
     3               0                0               0               0               0              0
      predictions
actual 2.25736187016896 2.33557353503455 2.38732983683066 2.52277995844221 2.61093514135363 2.66192431523577
     1                0                0                0                0                0              0
     2                1                0                1                0                0              0
     3                0                1                0                1                1              1
      predictions
actual 2.81938869024374 2.84008323495399 3.05892885205332 3.15405979162682 4.6624403142292
     1                0                0                0                0               0
     2                0                0                0                0               0
     3                1                1                1                1               1
Dokładność: 0.02857143
```

```
>
> # Macierz błędów i dokładność dla modelu SVM z funkcją jądra sigmoidalną
> cat("Model SVM - Sigmoidalne jądro:\n")
Model SVM - Sigmoidalne jądro:
> print_confusion_matrix_and_accuracy(predictions_sigmoid, test_set_standardized$V1)
Macierz błędów:
      predictions
actual 0.170958816936003 0.254592249240098 0.562343524885844 0.654949701319578 0.797110155552925 1.02177768725768
     1                 1                 0                 1                 1                 0               1
     2                 0                 1                 0                 0                 1               0
     3                 0                 0                 0                 0                 0               0
      predictions
actual 1.16512169218501 1.17914965115673 1.19346859315737 1.21862561431272 1.22244622412712 1.29340923469568
     1                1                1                1                0                1              0
     2                0                0                0                1                0              1
     3                0                0                0                0                0              0
      predictions
actual 1.39114358031228 1.39472017000781 1.39921570201934 1.44890281801352 1.46071247254648 1.50586735089311
     1                1                0                0                1                1              0
     2                0                1                1                0                0              0
     3                0                0                0                0                0              1
      predictions
actual 1.60140155283522 1.63426976016926 2.0629095414107 2.07475717194818 2.36698007397682 2.38123497925278
     1                1                0                0                0               0              0
     2                0                1                1                0               0              0
     3                0                0                0                1               1              1
      predictions
actual 2.40159199207233 2.47988220563543 2.49648333924844 2.63956539861096 2.65716453524223 2.73353272515553
     1                0                0                0                0               0              0
     2                0                1                1                0               1              0
     3                1                0                0                1               0              1
      predictions
actual 2.77841684375164 2.83009076470705 2.93846495025488 3.03640127260416 3.59408296574455
     1                0                0                0                0               0
     2                0                1                1                1               0
     3                1                0                0                0               1
Dokładność: 0.05714286
>
```

j) Sporządź wykres słupkowy ilustrujący dokładność klasyfikacji dla metod klasyfikacji: Bayes, SVM-f. radialna, SVM-f. liniowa, SVM-f. wielomianowa i SVM-f.sigmoidalna.

```r
# Przykładowe dane dotyczące dokładności (zastąp wartościami uzyskanymi z modeli)
dokladnosci <- data.frame(
  Metoda = c("Bayes", "SVM-Radialna", "SVM-Liniowa", "SVM-Wielomianowa", "SVM-Sigmoidalna"),
  Dokladnosc = c(0.97, 0.95, 0.96, 0.94, 0.93) # przykładowe wartości
)

# Tworzenie wykresu słupkowego
ggplot(dokladnosci, aes(x = Metoda, y = Dokladnosc)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  theme_minimal() +
  labs(title = "Dokładność klasyfikacji różnych metod",
       x = "Metoda",
       y = "Dokładność")
```