Zbiory benchmarkowe do analizy skupień

PDU 2018/19 - praca domowa nr 2

Konrad Komisarczyk

1. Mieszkańcy miast w Polsce

Zbiór reprezentuje uproszczone przybliżone rozmieszczenie ludności miejskiej w Polsce.

Miasta przybliżane są kołami o polu równym obszarowi miasta.

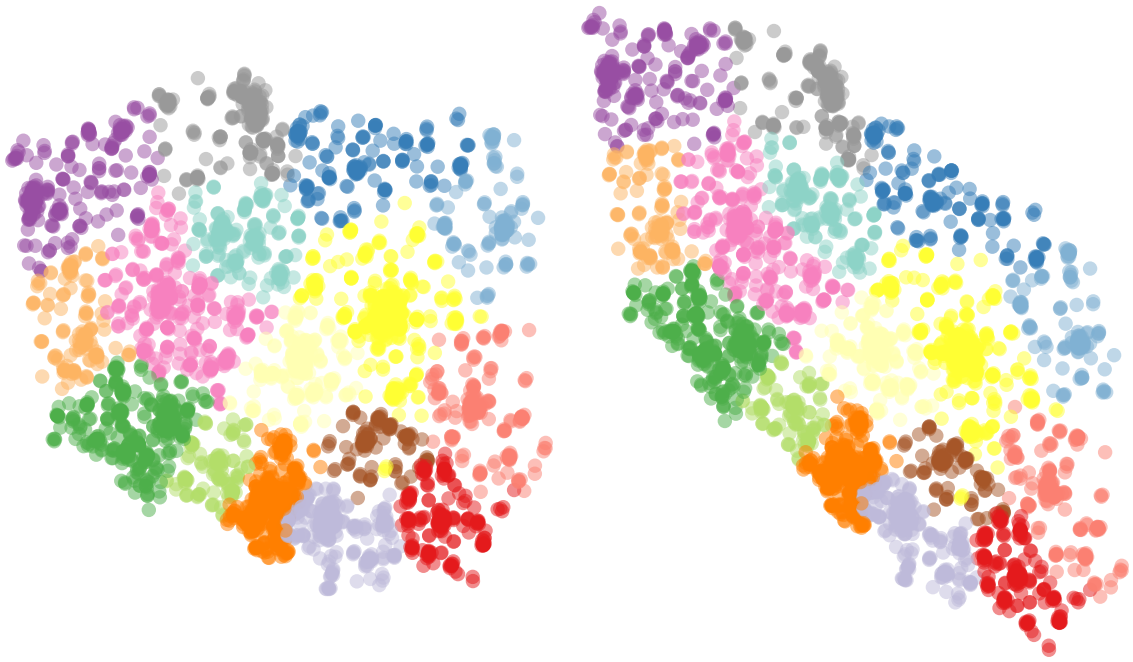
Mieszkańców miasta reprezentują punkty rozmieszczone według rozkładu normalnego o środku w środku koła przybliżającego miasto i odchyleniu standardowym równym promieniowi miasta. Jeden punkt odpowiada 4000 mieszkańcom.

Punkty znajdują się na przybliżonym zrzutowaniu powierzchni ziemi na płaszczyznę euklidesową - 1 stopień szerokości geograficznej odpowiada `110.574 km`, a 1 stopień długości `111.32 * cos(0.01745329 * [szerokość geograficzna w stopniach])`.

Zbiór składa się z 5796 punktów.

Przygotowałem podział zbioru na 3 typy etykiet referencyjnych:

- podział na 16 województw,
- podział na 367 powiatów (nie uwzględniam go w badaniach),
- podział na 940 miast (nie uwzględniam go w badaniach).



Rysunek 1: Ilustracja zbioru z podziałem na województwa. Po lewej przeskalowane tak, żeby wyglądało, jak na mapie, a po prawej współrzędne punktów przechowywane w zbiorze.

Sposób przygotowania zbioru

Dane zebrałem z artykułów na wikipedii

- Listy miast w polsce https://pl.wikipedia.org/wiki/Miasta_w_Polsce
- Tabeli danych statystycznych o miastach w Polsce https://pl.wikipedia.org/wiki/Dane_statystyczne_o_miastach_w_Polsce
- Współrzędnych geograficznych miast z artykułu o każdym z miast

Za pomocą skryptu `cities.py` w języku `python`:

```
from bs4 import BeautifulSoup
import requests

html = requests.get("https://pl.wikipedia.org/wiki/Miasta_w_Polsce").content
soup = BeautifulSoup(html)

coords_file = open('coords', 'w+')

for div in soup.find_all('div'):
    if div.get("style") == '-moz-column-count:3; -webkit-column-count:3; column-count:3;':
        for li in div.find_all('li'):
            for a in li.find_all('a'):
                nazwa = a.get("title")
                link = a.get("href")
                link_html = requests.get("https://pl.wikipedia.org" + link).content
                link_soup = BeautifulSoup(link_html)
                lat = []
                lon = []
                for span in link_soup.find_all('span'):
                    if span.get("class") == ["latitude"]:
                        lat.append(span.get_text())
                    if span.get("class") == ["longitude"]:
                        lon.append(span.get_text())
                print(nazwa, lat[1], lon[1], sep=";", file=coords_file)

link2 = "https://pl.wikipedia.org/wiki/Dane_statystyczne_o_miastach_w_Polsce"
html2 = requests.get(link).content
soup2 = BeautifulSoup(html2)

population_file = open('population', 'w+')

for tr in soup2.find_all('tr'):
    for td in tr.find_all('td'):
        print(td.get_text()[0:-1], file=population_file, end=';')
    print('', file=population_file)
```

i podmieniając ‘,’ na ‘.’ narzędziem `sed` (`cities.sh`):

```
sed -i 's/,/./g' population coords
```

Przygotowałem ramki z danymi

- `population` - kolejne kolumny:
  - nazwa miasta
  - nazwa powiatu
  - nazwa województwa
  - powierzchnia (km2)
  - liczba ludności
  - gęstość zaludnienia
- `coords` - kolejne kolumny:
  - nazwa miasta
  - szerokość geograficzna
  - długość geograficzna

Następnie z nich z użyciem skryptu `cities.R` przygotowałem 3 zbiory danych w odpowiednim formacie:

- `by_voivodeship.data` i `by_voivodeship.labels0`
- `by_powiat.data` i `by_powiat.labels0`
- `by_city.data` i `by_city.labels0`

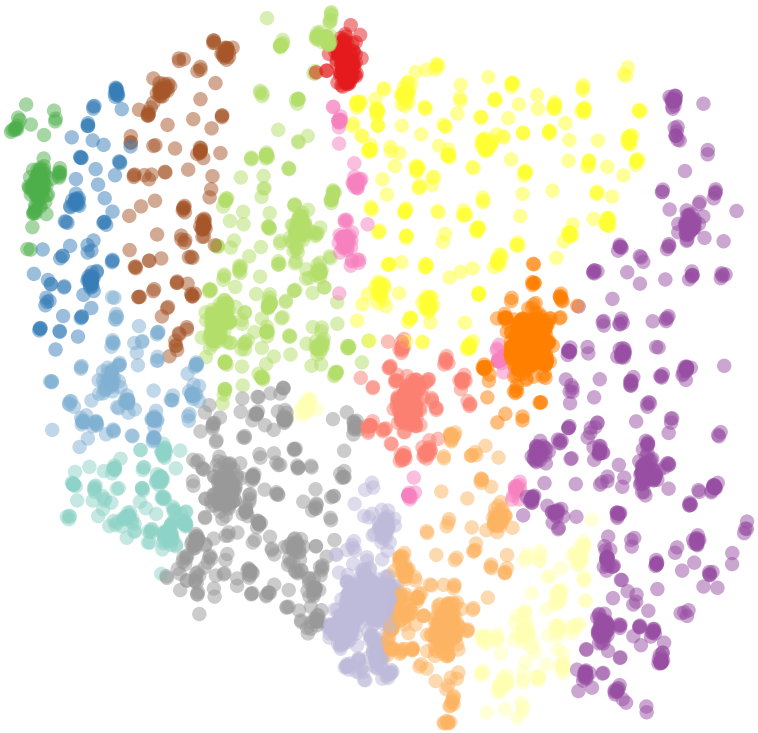
Przygotowałem ramkę `cities2`, z wierszami reprezentującymi punkty w tych zbiorach danych:

```
library(dplyr)
#setwd(file.path("data", "my", "cities"))

# wczytujemy dane zescrapowane pythonem
coords <- read.table("coords", sep = ";", stringsAsFactors = FALSE)
colnames(coords) <- c("C1", "C2", "C3")

population <- read.table("population", sep = ";", stringsAsFactors = FALSE)

# tabele zgadzają się wierszami, złączamy je kolumnami
cities <- cbind(population, coords)


voivodeships <- unique(cities$V3)
powiats <- unique(cities$V2)
n <- nrow(cities)

cities <- cities %>% mutate(
  # skalujemy współrzędne na powierzchnię euklidesową
  y = C3 * 110.574, x = C2 * 111.32 * cos(0.01745329 * C3),
  # obliczamy promień koła przybliżającego miasto
  area = V4, r = sqrt(area / pi),
  # obliczamy liczbę punktów reprezentujących miasto
  popul = V4 * V6, points = round(popul / 4000),
  # kategoryzujemy miasta po województwie, powiecie i mieście
  voiv = match(V3, voivodeships),
  powi = match(V2, powiats),
  city = 1:n)


# tabela zawierająca wiersze odpowiadające naszym punktom - tyle kopii
# wierszy odpowiadających miastu z tabeli cities, ile punktów je reprezentuje
cities2 <- cities %>%
  tidyr::uncount(points)
```

```
# rozrzucamy punkty rozkładami normalnymi względem środków miast, które
# reprezentują i promieni tych miast
n <- nrow(cities2)
t <- runif(n, 0, 2*pi)
a <- rnorm(n, 0, cities2$r)
cities2$x <- sin(t) * a + cities2$x
cities2$y <- cos(t) * a + cities2$y
```

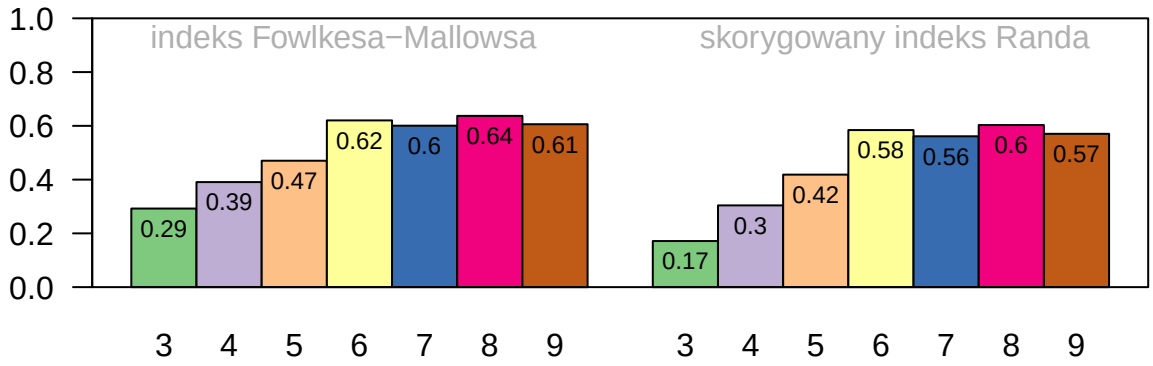
i zapisałem dane do plików w odpowiednich formatach:

```
cities_coords <- t(cbind(cities2$x, cities2$y))

## podział na grupy po województwie
write(cities_coords, "by_voivodeship.data", ncolumns = 2)
write(cities2$voiv, "by_voivodeship.labels0", ncolumns = 1)

## podział na grupy po powiecie
write(cities_coords, "by_powiat.data", ncolumns = 2)
write(cities2$powi, "by_powiat.labels0", ncolumns = 1)

## podział na grupy po mieście
write(cities_coords, "by_city.data", ncolumns = 2)
write(cities2$city, "by_city.labels0", ncolumns = 1)
```



Rysunek 2: Podział algorytmem spektralnym zbioru na 16 części dla parametru M=8

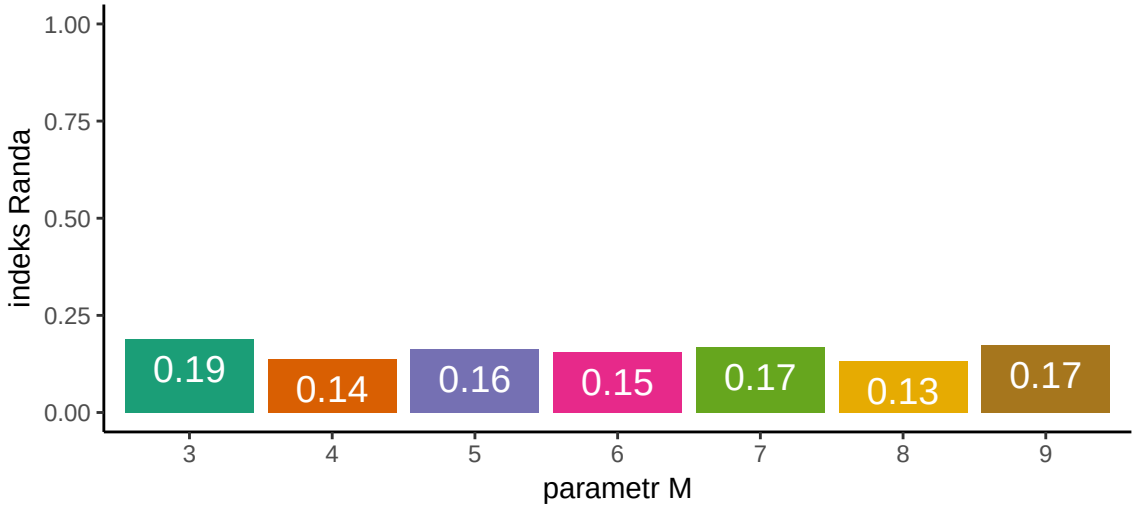
1.1. Podział na województwa (`by_voivodeship`)



Rysunek 3: Wartości indeksów Fowlkesa-Mallowsa i Randa dla algorytmu spektralnego na zbiorze dzielonym na województwa w zależności od parametru M.

Algorytm zwraca dość dobre wyniki. Są nieco niższe niż średnie dla zbiorów benchmarkowych, ale lepsze niż najgorsze z pozostałych zbiorów benchmarkowych. Nie oczekiwałem od tego zbioru bardzo dobrej jakości wyników, oczekiwałem właśnie gdzieś mniej więcej takiej. Województwa w Polsce często mają w centrum duże miasto wojewódzkie i często miasta skupiają się wokół nich, czasami tworząc nawet duże aglomeracje miejskie, jak aglomeracja Warszawy, czy Trójmiasto.

1.2. Podział na powiaty (`by_powiat`)



Rysunek 4: Wartości indeksu Randa dla algorytmu spektralnego na zbiorze dzielonym na powiaty w zależności od parametru M.

Algorytm zwraca bardzo słabe wyniki, czego oczekiwałem po tym podziale. W związku z tym, że nie jest on dobrym referencyjnym podziałem na etykiety, nie będę dołączał go do zbiorów benchmarkowych.

2. Proste aglomeracje

Zbiory zawierają kilka skupisk (aglomeracjii) punktów rozkładających się rozkładem normalnym względem pewnych środków.

Zbiory generuję za pomocą funkcji `simple_agglomerations` przyjmującej jako parametry: * liczbę punktów należących do aglomeracji * promienie aglomeracji * kwadrat wewnątrz którego będziemy losowali punkty zbioru

Etykiety referencyjne zawierają dla każdego punktu informację o tym, do której aglomeracji należy.

Funkcja losuje najpierw środki aglomeracji, a następnie losuje należące do nich punkty według rozkładu normalnego o odchyleniu równym promieniowi aglomeracji.

```
## n_points - wektor z liczbą punktów w kolejnych aglomeracjach
## radius - promienie kolejnych aglomeracji
## lim - bok kwadratu, na którym umieszczone są środki aglomeracji
simple_agglomerations <- function(n_points, radius, lim) {
  n_aggls <- length(n_points)

  agglomeration <- function(center, n, radius) {
    rnorm(n, mean = center, sd = radius)
  }

  centers_x <- runif(n_aggls, min = 0, max = lim)
  centers_y <- runif(n_aggls, min = 0, max = lim)

  X <- matrix(numeric(0), ncol = 2)
  labels <- numeric(0)
  for (i in 1:n_aggls) {
    X <- rbind(X, (cbind(agglomeration(centers_x[i], n_points[i],
                                       radius[i]),
                         agglomeration(centers_y[i], n_points[i],
                                       radius[i]))))
    labels <- c(labels, rep(i, n_points[i]))
  }

  return(list(coords = X, labels = labels))
}
```

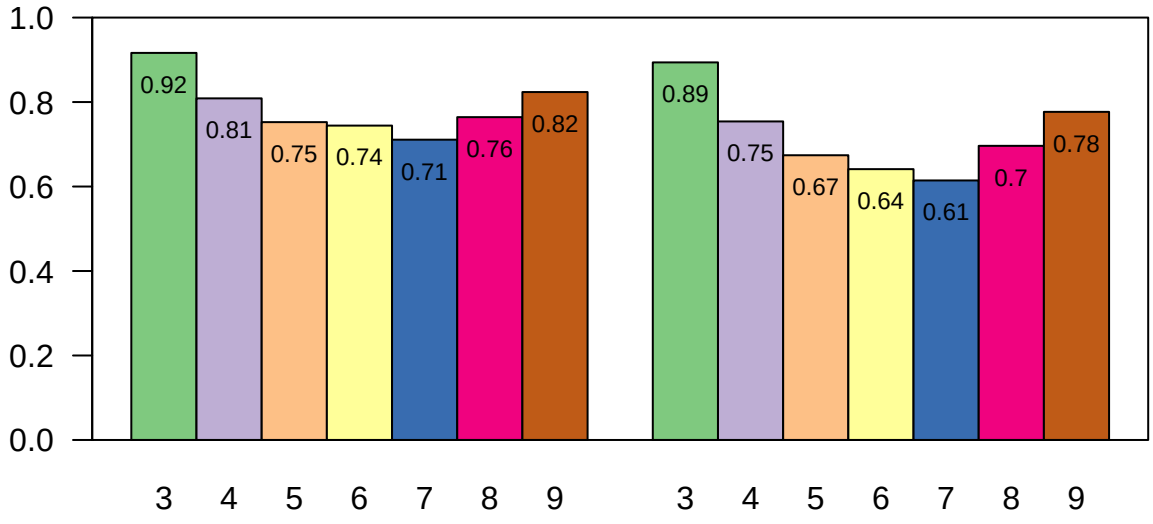
Przygotowałem trzy warianty zbioru:

2.1. simple_agglomerations1

Zbiór składa się z 1347 punktów rozmieszczonych w 5 aglomeracjach.

```
set.seed(1234)
n <- 5
set1 <- simple_agglomerations(n_points = runif(n, 100, 400),
                              lim = 400, radius = runif(n, 20, 36))
```

Rysunek 5: Referencyjny podział zbioru i podział zbioru algorytmem spektralnym dla parametru M=3

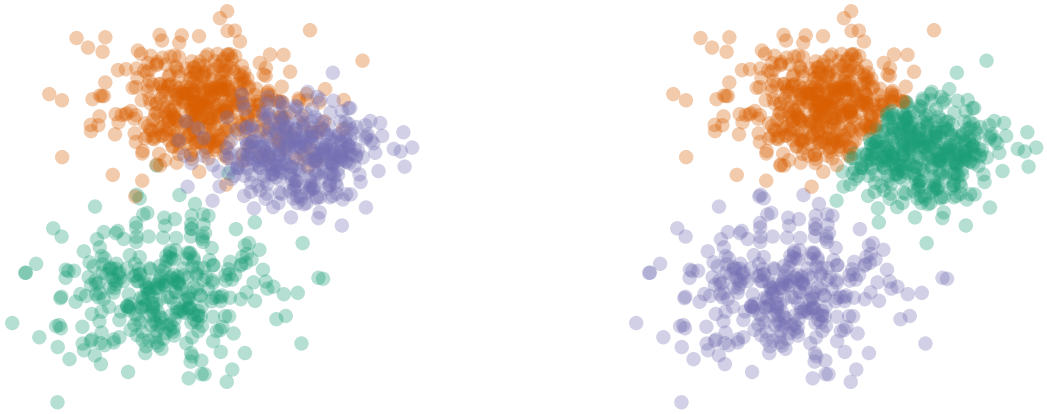


Rysunek 6: Wartości indeksów Fowlkesa-Mallowsa i Randa dla algorytmu spektralnego na zbiorze simple-agglomerations1 w zależności od parametru M.

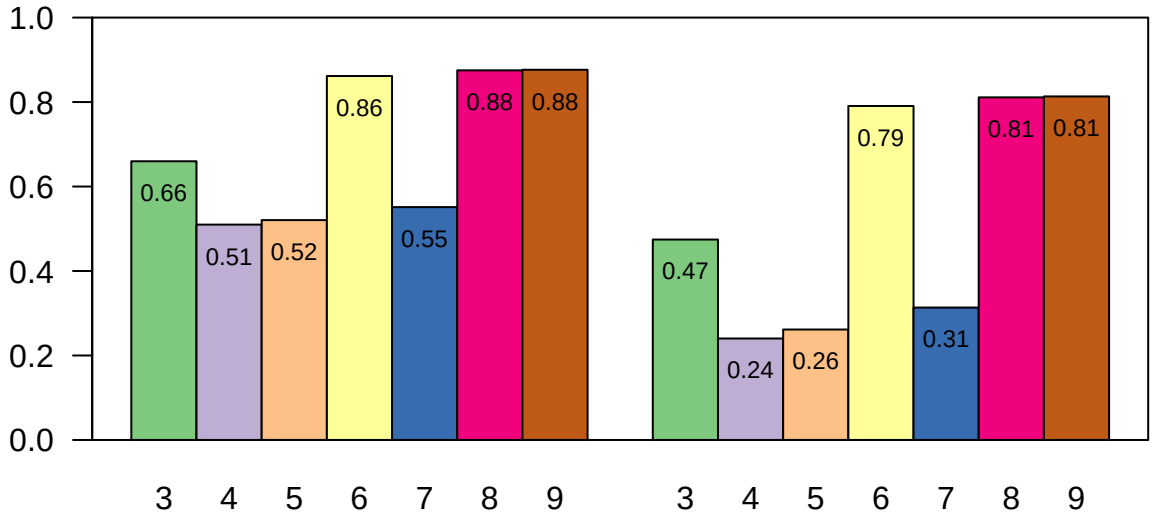
2.2. simple_agglomerations2

Zbiór składa się z 1184 punktów rozmieszczonych w 3 aglomeracjach.

```
set.seed(7312)
n <- 3
set2 <- simple_agglomerations(n_points = runif(n, 300, 600),
                              lim = 300, radius = runif(n, 30, 50))
```



Rysunek 7: Referencyjny podział zbioru i podział zbioru algorytmem spektralnym dla parametru M=9

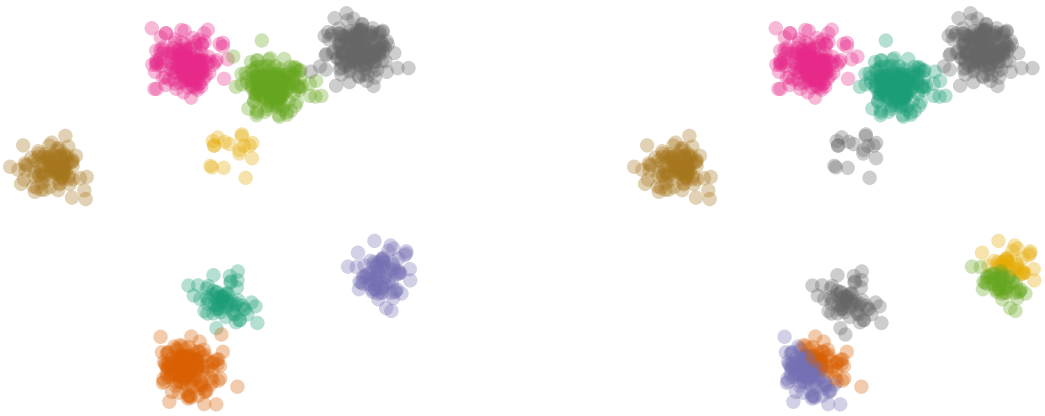


Rysunek 8: Wartości indeksów Fowlkesa-Mallowsa i Randa dla algorytmu spektralnego na zbiorze simple-agglomerations2 w zależności od parametru M.

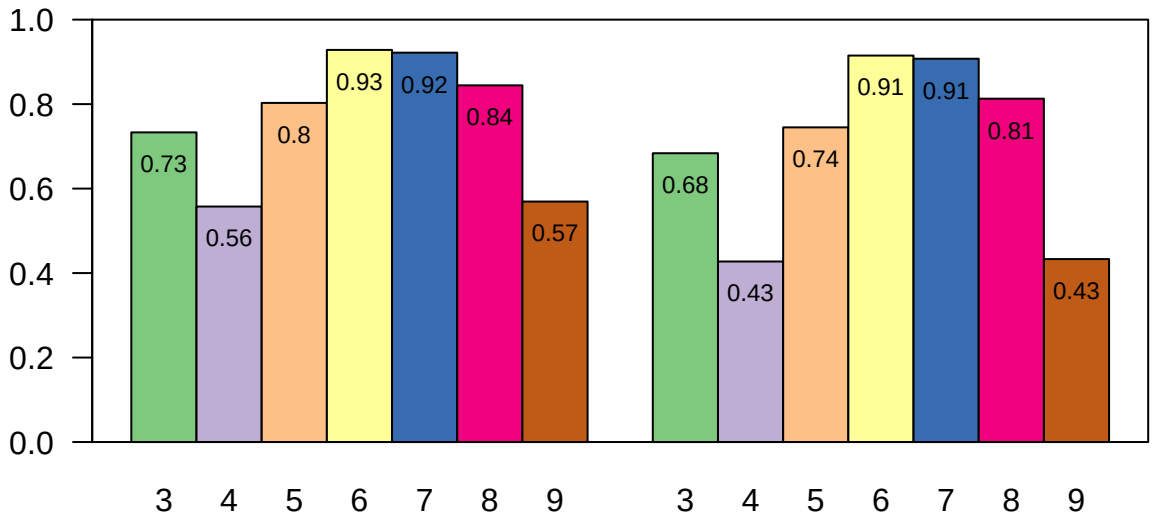
2.3. simple_agglomerations3

Zbiór składa się z 982 punktów rozmieszczonych w 8 aglomeracjach.

```
set.seed(123)
n <- 8
set3 <- simple_agglomerations(n_points = runif(n, 10, 200),
                              lim = 200, radius = rep(n, 13))
```



Rysunek 9: Referencyjny podział zbioru i podział zbioru algorytmem spektralnym dla parametru M=6



Rysunek 10: Wartości indeksów Fowlkesa-Mallowsa i Randa dla algorytmu spektralnego na zbiorze simple-agglomerations3 w zależności od parametru M.

3. Oliwki na pokrojonej pizzy

Zbiory reprezentują oliwki na pokrojonej pizzy z odsuniętymi od siebie kawałkami.

Punkty są równomiernie rozłożone na każdym kawałku. Kawałki są rozsunięte, czyli są pomiędzy nimi pasy, w których nie ma żadnych punktów.

Zbiór generuje następująca funkcja sparametryzowana szerokością przerwy pomiędzy kawałkami i liczbą punktów w poszczególnych kawałkach:

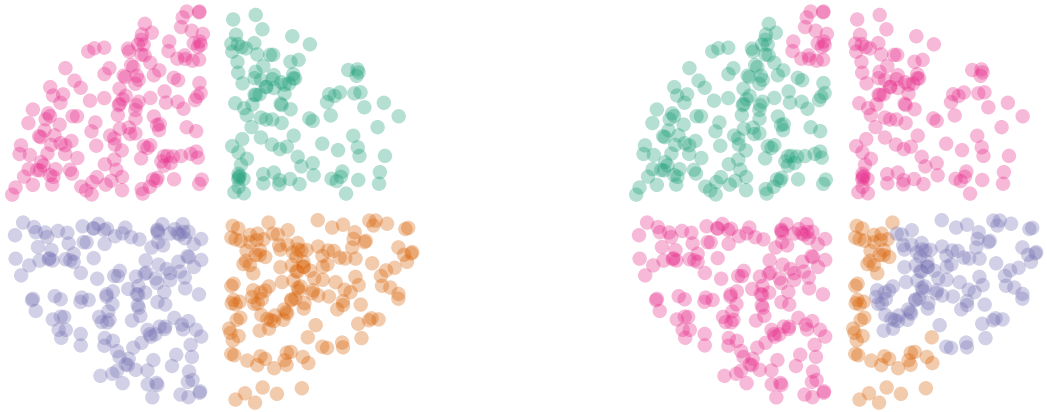
```
## sets - wektor z liczbą punktów należących do kolejnych kawałków pizzy
## space - promień, o jaki odsunięte są kawałki od środka
pizza <- function(sets, space) {
  n <- sum(sets)
  parts <- length(sets)

  # punkty należące do nr-tego kawałka
  points_in_part <- function(n, nr, parts, space) {
    # generujemy n punktów rozłożonych równomiernie na nr-tym kawałku pizzy
    t <- runif(n, (nr-1)*2*pi/parts, nr*2*pi/parts)
    a <- runif(n, 0, 1)
    a <- acos(a)
    x <- sin(t) * a
    y <- cos(t) * a

    # odsuwamy kawałek od środka
    v <- (nr - 0.5)*2*pi/parts
    x <- x + sin(v) * space
    y <- y + cos(v) * space

    return(cbind(x, y))
  }

  X <- matrix(numeric(0), ncol = 2)
  labels <- numeric(0)
  for (i in 1:parts) {
    p <- points_in_part(sets[i], i, parts, space)
    X <- rbind(X, p)
    labels <- c(labels, rep(i, sets[i]))
  }

  return(list(coords = X, labels = labels))
}
```

Przygotowałem dwa warianty zbioru:

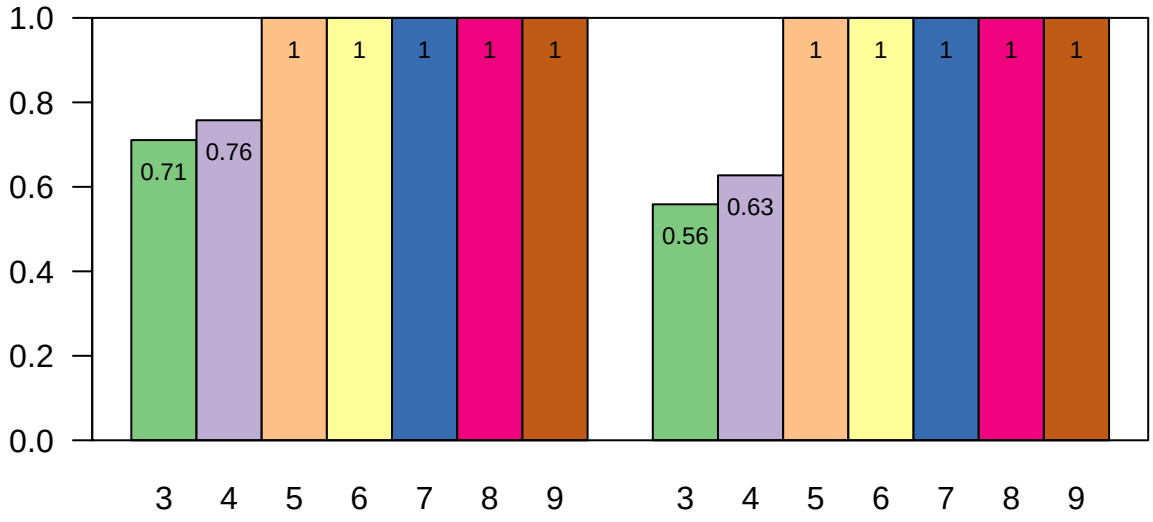
3.1. Mała pizza (4 kawałki)

```
set.seed(1234)
n <- 4
sets <- runif(n, 100, 200)
pitca1 <- pizza(sets, 0.15)
```

Zbiór składa się z 595 punktów.



Rysunek 11: Referencyjny podział zbioru i podział zbioru algorytmem spektralnym dla parametru M=4

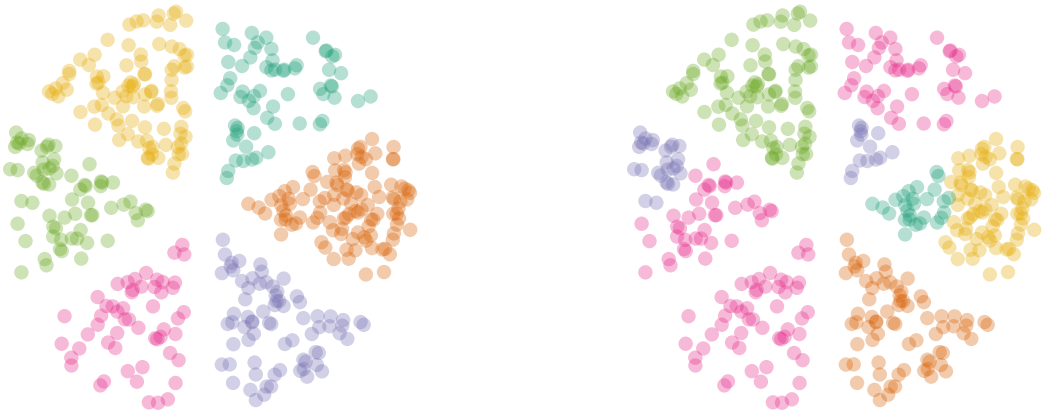


Rysunek 12: Wartości indeksów Fowlkesa-Mallowsa i Randa dla algorytmu spektralnego na zbiorze pizza1 w zależności od parametru M.

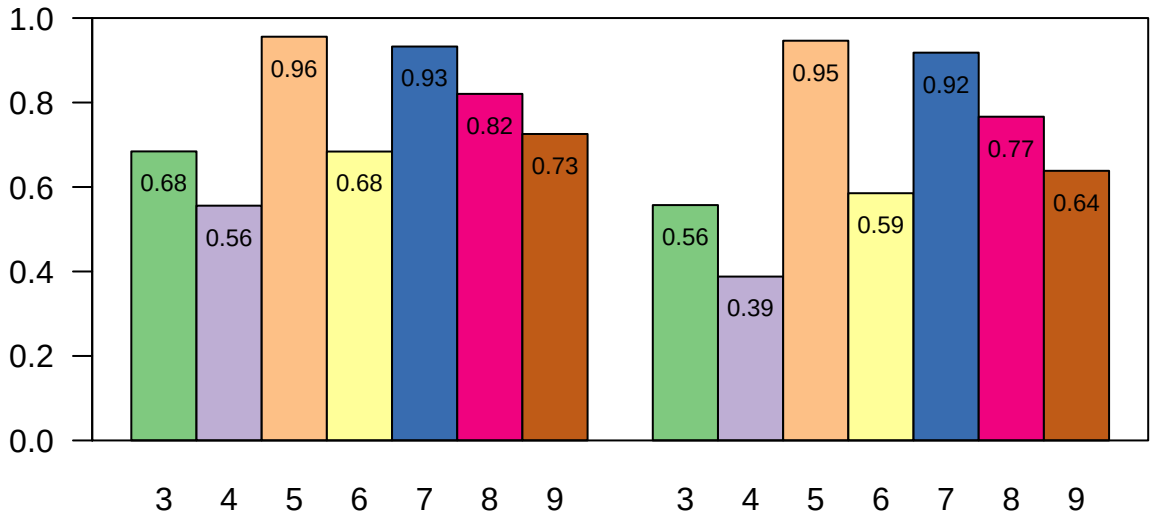
3.2. Średnia pizza (6 kawałków)

```
set.seed(7312)
n <- 6
sets <- runif(n, 50, 150)
pitca2 <- pizza(sets, 0.3)
```

Zbiór zkłada się z 454 punktów.

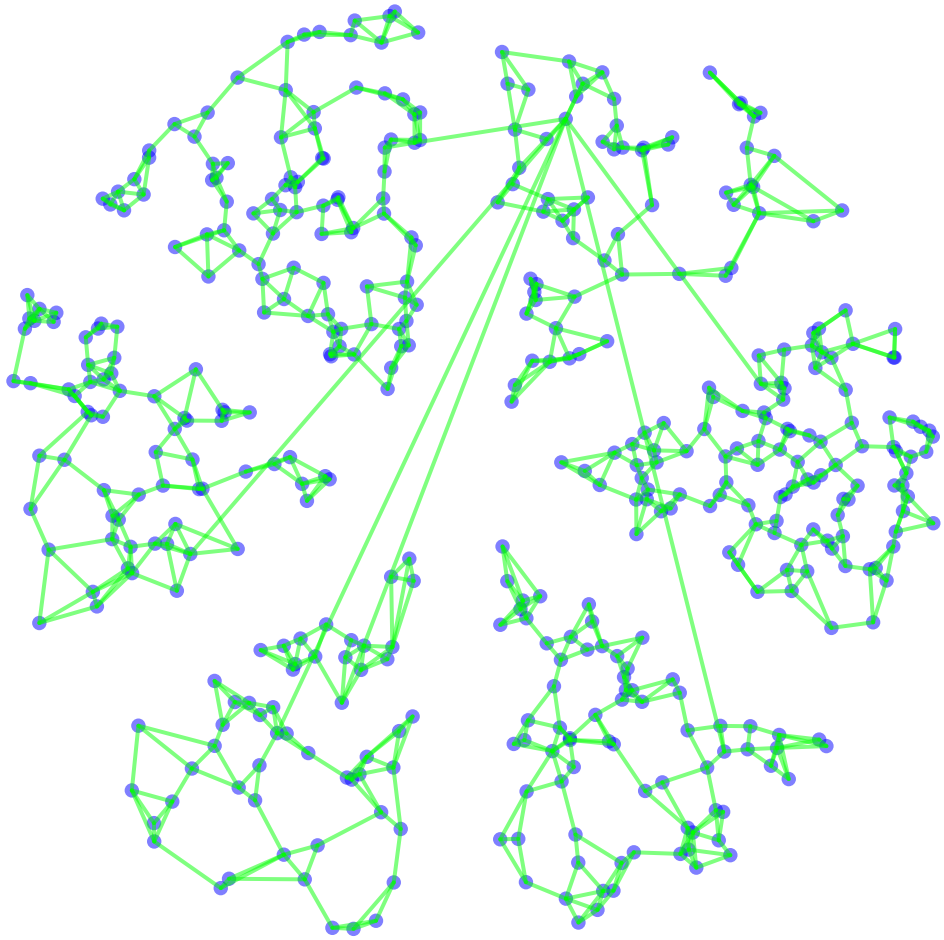


Rysunek 13: Referencyjny podział zbioru i podział zbioru algorytmem spektralnym dla parametru M=6

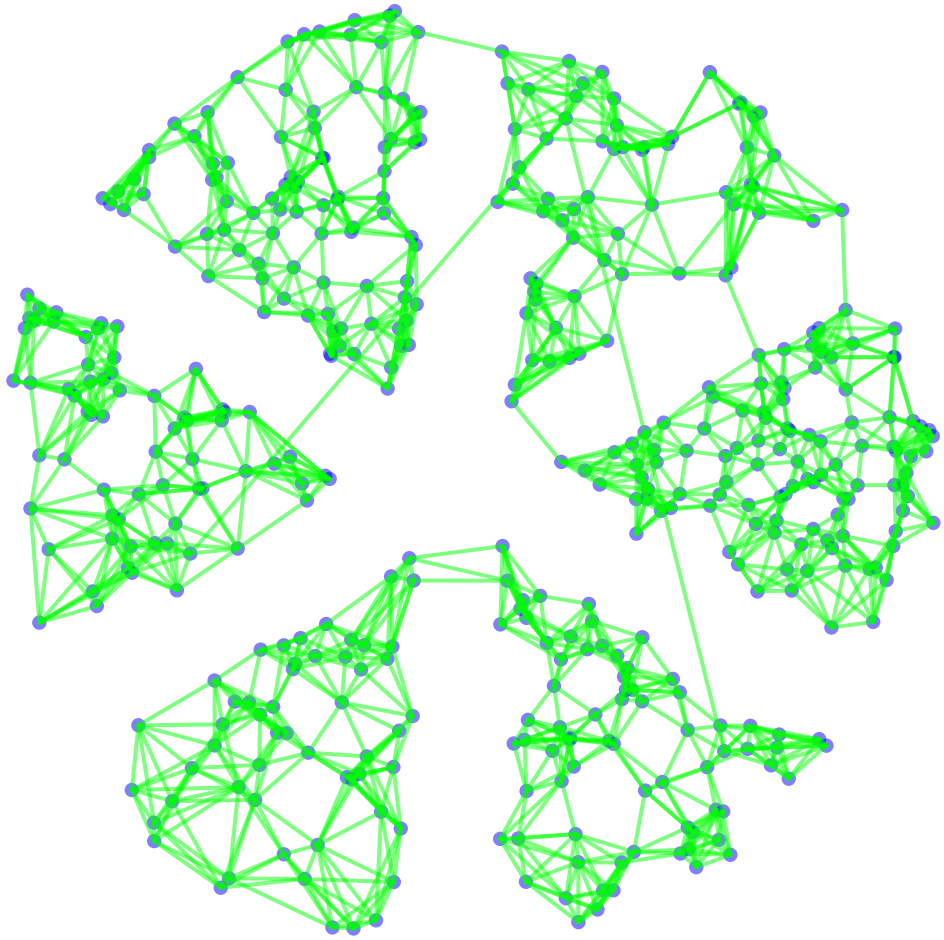


Rysunek 14: Wartości indeksów Fowlkesa-Mallowsa i Randa dla algorytmu spektralnego na zbiorze pizza2 w zależności od parametru M.

4. Dodatek



Rysunek 15: Rysunek przedstawiający graf 3 najbliższych sąsiadów na zbiorze pizza2



Rysunek 16: Rysunek przedstawiający graf 7 najbliższych sąsiadów na zbiorze pizza2