Projekt III - Algorytm k-średnich

Patrycja Lewczuk

5 czerwca 2022

Wpierw należy wylosować 5 obserwacji kolejno z grupy kobiet i z grupy mężczyzn. Następnie zaokrągla się dane wartości do liczby całkowitej.

nr.	wzrost	waga
1	65	147
2	71	187
3	70	198
4	67	175
5	69	192
6	66	155
7	65	149
8	59	96
9	65	147
10	67	132

Następnie należy wybrac ilość kklastrów i losowo podzielić obserwacje, żeby wybór grup był niezależny. Do tego został użyty R - do wybrania podziału. Wybrana ilość klastrów to 2. Otrzymano:

nr.	wzrost	waga
3	70	198
6	66	155
9	65	147

nr.	wzrost	waga
1	65	147
2	71	187
4	67	175
5	69	192
7	65	149
8	59	96
10	67	132

Teraz można zastosować **algorytm k-średnich**
Wpierw należy obliczyć centroidy dla poszczególnych klastrów:

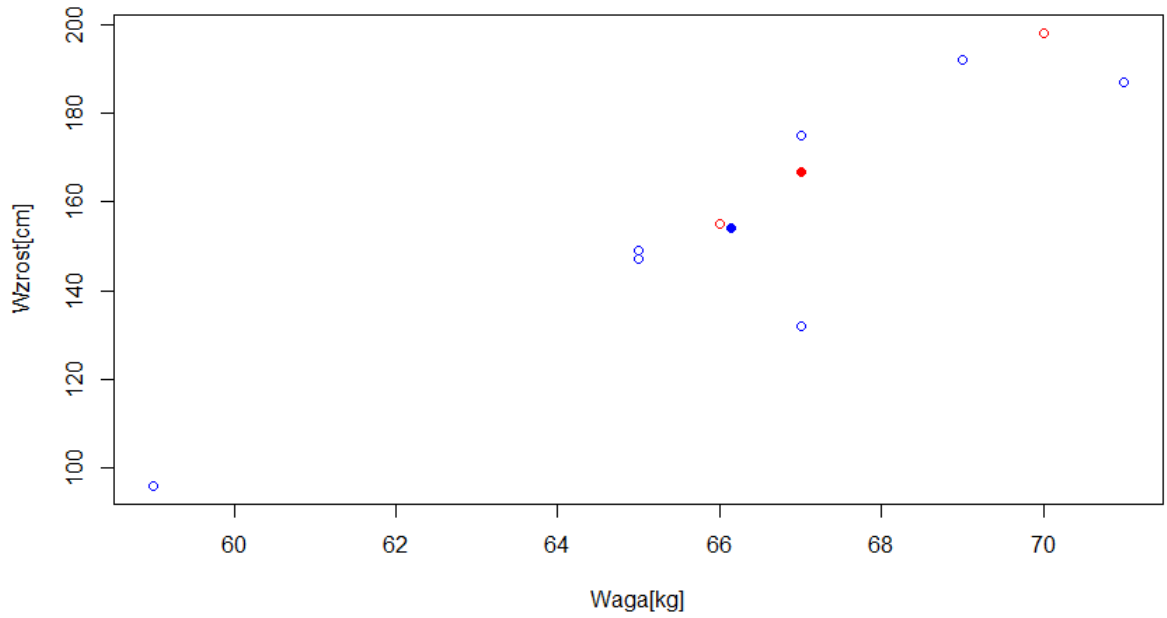
$$x_1 = \frac{70 + 66 + 65}{3} = 67$$

$$y_1 = \frac{198 + 155 + 147}{3} \approx 166.7$$

$$x_2 = \frac{65 + 71 + 67 + 65 + 59 + 67}{7} \approx 66.14$$

$$y_2 = \frac{147 + 187 + 175 + 192 + 149 + 96 + 132}{7} = 154$$

Na wykresie zostały oznaczone klastry wraz z centroidami:



Następnie należy obliczyć odległość pomiędzy każdym z punktów, a środkami obu centroidów. Używamy do tego odległości euklidesowej. Następnie będziemy wybierać dla każdego punktu jego nowy klaster tzn. klaster, którego odległość jego centroida od danego punkty jest jak najmniejsza.
Ręcznie zostało obliczone dla punktu pierwszego, reszta została obliczona przy pomocy programu R. Obliczono odległość od pirwszego punktu z pierwszego klastra $(70, 198)$ dla:

- pierwszego centroida $(67, 166.7)$

  1. $$|x - x_1| = |70 - 67| = 3$$
  2. $$|y - y_1| = |198 - 16.7| = 31.3$$
  3. $$\sqrt{(3)^2 + (31.3)^2} =\approx 31.44$$

- drugiego centroida $(66.14, 154)$

  1. $$|x - x_1| = |70 - 66.14| = 3.86$$
  2. $$|y - y_1| = |198 - 154| = 44$$
  3. $$\sqrt{(3.86)^2 + (44)^2} =\approx 44.17$$

Odległość pierwszego punktu od pierwszego centroida jest mniejsza, niż jego odległość od środka drugiego klastra, zatem pierwszy punkt zostaje przydzielony do pierwszego klastra. Pozostałe obliczenia naley przeprowadzić analogicznie. W poniższej tabeli zostały umieszczone wyniki.

nr.	odległość od x_1, y_1	odległość od x_2, y_2
1	19.7681	7.092681
2	20.72304	33.35554
3	31.47662	44.16874
4	8.333333	21.01749
5	25.41216	38.10726
6	11.70945	1.010153
7	17.77951	5.128949
8	71.11806	58.43818
9	19.7681	7.092681
10	34.66667	22.01669

Można teraz przegrupować dane do klastrów ze względu na najmniejszą odległość punktu od centrodiu. Podział wygląda następująco:

nr.	wzrost	waga
2	71	187
3	70	198
4	67	175
5	69	192

nr.	wzrost	waga
1	65	147
6	66	155
7	65	149
8	59	132
9	65	147
10	67	132

Ponownie liczymy centroidy dla aktualnych klastrów.

$$x_1 = \frac{71 + 70 + 67 + 69}{4} = 69.25$$

$$y_1 = \frac{187 + 198 + 175 + 192}{4} = 188$$

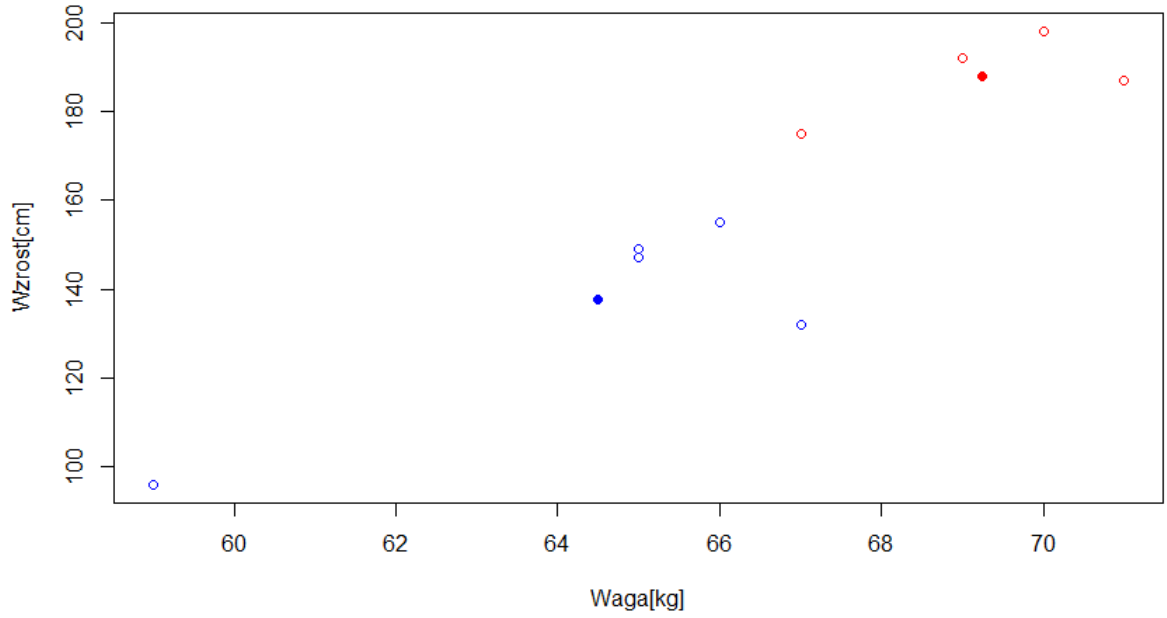
$$x_2 = \frac{65 + 66 + 65 + 59 + 65 + 67}{6} = 64.5$$

$$y_2 = \frac{147 + 155 + 149 + 132 + 147 + 132}{6} \approx 137.67$$

Następuje ponowne powtórzenie algorytmu.
Aby policzyć odległości między punktami i nowymi centroidami korzystano głównie z programu R. W poprzedniej części zostało pokazane, jak oblicza się daną odległość euklidesową. Wyniki są następujące:

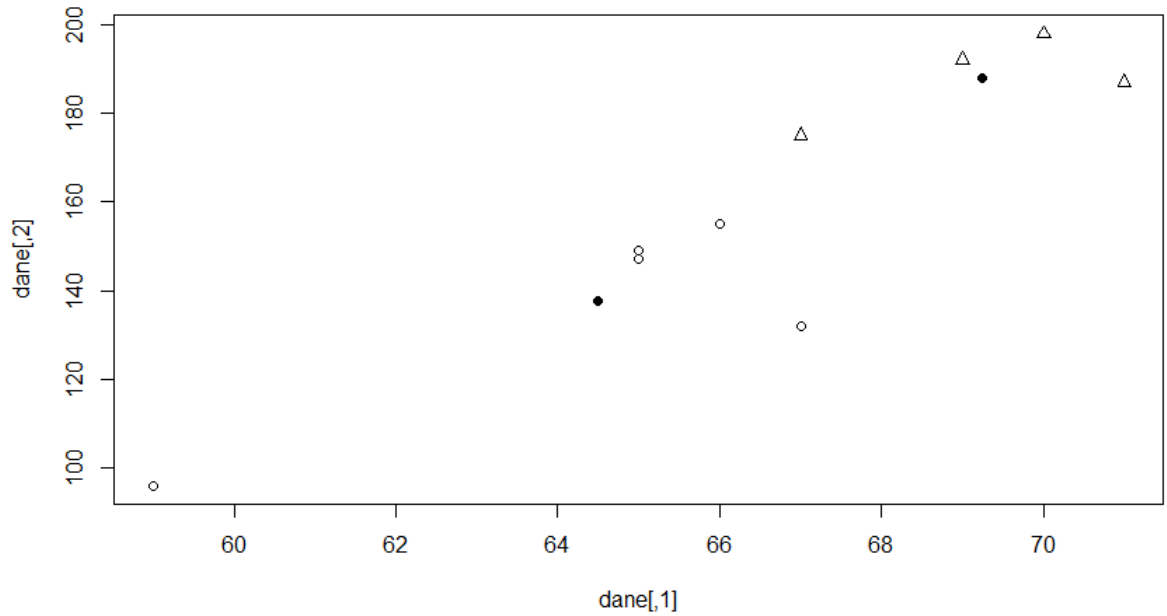
nr.	odległość od x_1, y_1	odległość od x_2, y_2
1	41.21969	9.346717
2	2.015564	49.7597
3	10.02809	60.58351
4	13.19327	37.41695
5	4.007805	54.51936
6	33.15965	17.39812
7	39.23089	11.34436
8	92.56923	42.0281
9	41.21969	9.346717
10	56.04518	6.193635

Podział danych na klastry nie zmieni się, co oznacza, że dane zostały odpowiednio przegrupowane i jest to koniec algorytmu. Zobrazujemy nasze wyznaczone klastry zraz z centroidami na wykresie:



Porównanie wyniku z wbudowanym algorytmem w R, aby sprawdzić poprawność:

```
> #wykorzystanie wbudowany=ego w R algorytmu
> klaster = kmeans(dane,2)
> str(klaster)
List of 9
 $ cluster     : int [1:10] 1 2 2 2 2 1 1 1 1 1
 $ centers     : num [1:2, 1:2] 64.5 69.2 137.7 188
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:2] "1" "2"
  .. ..$ : NULL
 $ totss       : num 8840
 $ withinss    : num [1:2] 2411 295
 $ tot.withinss: num 2706
 $ betweenss   : num 6134
 $ size        : int [1:2] 6 4
 $ iter        : int 1
 $ ifault      : int 0
 - attr(*, "class")= chr "kmeans"
> plot(dane, pch=klaster$cluster)
> points(klaster$centers, pch=19)
>
```



Wnioski:

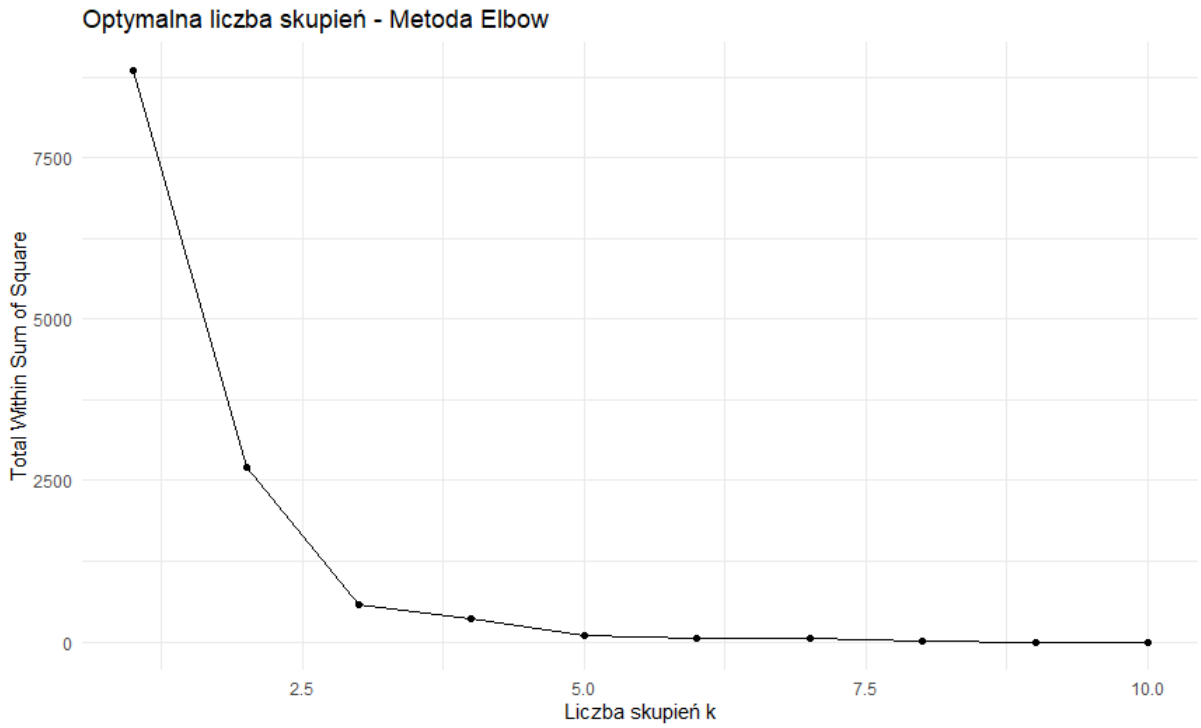
1. Wyliczenie ręczne pokrywa się z wynikami funkcji wbudowanej w R.
2. nasze obserwacje zostały podzielone prawie według płci, wyjątkiem jest pierwsza wylosowana obserwacja, która należy do grupy mężczyzn, ale taka sama wartość znajdowała się wśród kobiet i obie zakwalifikowały się do klastra oznaczonego kolorem niebieskim.

Wybór ilości klastrów Wybór liczby skupień (klastrów) może być dokonany na wiele sposobów, za pomocą różnych metod lub algorytmow. Najłatwiejszym jest narzucenie losowej liczby (musi być większa od 2 i mniejsza bądź równa od ilości danych) i odczytanie z wykresu, czy taka liczba na pewno jest odpowiednia. W zależności od lizcby danych, możemy nie być w stanie znaleźć odpowiedniej liczby skupień.
Wraz z rozwojem technologii oraz samej dziedziny analizy danych i statystyki, zostały stworzone w programach algorytmy, które pozwalają ustalić lub w pewny sposób określić odpowiednią liczbę skupień, np. zawierają implementację v-krotnego sprawdzianu krzyżowego - algorytmu automatycznie wyznaczającego liczbę skupień danych.
Istnieją również statyczyne metody, które mogą ułatwić nam to zadanie, m.in. Elbow, Silhouette lub Gap Statistic. Posłużymy się pierwszą z wymienionych metod, gdyż możemy samodzielnie stworzyć funkcję generującą wykres, na podstawie którego odczytamy najlepszą wartość liczby skupień.
Wraz ze wzrostem liczby skupień suma wariancji będzie malała. Dzieję się tak, ponieważ skupienia są coraz mniejsze. Optymalnej liczby klastrów poszukujemy w miejscu, w którym suma wariancji przestaje gwałtownie maleć, a podział na coraz większą liczbę skupień nie poprawia znacząco wyniku.
Posłużę się 10 obserwacjami, które zostały wykorzystane w danym projekcie.

```r
library(ggplot2)

tot.withinss <- vector("numeric", length = 10)
for (i in 1:9){
  k_d <- kmeans(dane, i)
  tot.withinss[i] <- k_d$tot.withinss
}

ggplot(as.data.frame(tot.withinss), aes(x = seq(1,10), y = tot.withinss)) +
  theme_minimal() +
  geom_point(col = 'black') +
  geom_line(col = 'black') +
```



W naszym przypadku poszukiwaną liczbą klastrów może być liczba 5.