

# Porządkowanie nieliniowe za pomocą wybranych metod aglomeracyjnych

## Wstęp

Zostanie tutaj zaprezentowane zastosowanie nieliniowego porządkowania danych przy pomocy istniejących funkcji biblioteki cluster, w której to funkcja agnes umożliwiającą uporządkowanie zbioru po wyborze odpowiedniej metody aglomeracyjnej. Mamy tu do wyboru metody: single - metoda najbliższego sąsiedztwa, complete - metoda najdalszego sąsiedztwa, ward - metoda Warda, average - metoda średniej między grupowej. W poniżej zostanie zaprezentowane zastosowanie metod single oraz complete wraz z porównaniem wyników porządkowania.

## Import danych

Na początku należy zaimportować dane, które chcemy poddać porządkowaniu. W tym celu należy zaimportować bibliotekę readxl - gdyż dane pobieramy z excela, a w kolejnym kroku wywołujemy plik, podając naszą ścieżkę pliku z rozszerzeniem.xlsx. My użyjemy tutaj zbioru zawierającego 8 obiektów, będących ofertami sprzedaży aut.

```
library(readxl)
zbior_danych <- read_excel("datasets/zbior_danych.xlsx",
                           sheet = "DANE_INNA_WERSJA")
```

Podgląd danych:

```
head(zbior_danych)

## # A tibble: 6 x 30
##   Nr MARKA  MODEL  WERSJA TYP      WOJEWODZTWO `CENA.NETTO_[pln]`
##   <dbl> <chr>   <chr>   <chr> <chr>   <chr>             <dbl>
## 1  1.00 Hyundai i20     II     kompakt malopolskie      NA
## 2  2.00 Hyundai i20     I      kompakt mazowieckie      NA
## 3  3.00 Subaru  Legacy V      kombi    mazowieckie      NA
## 4  4.00 Ford    Mondeo Mk4     sedan  dolnoslaskie      NA
## 5  5.00 Opel    Astra  G      kompakt slaskie        NA
## 6  6.00 Mazda   Premacy <NA> minivan dolnoslaskie      NA
## # ... with 23 more variables: `CENA.BRUTTO_[pln]` <dbl>, `MOC_[km]` <dbl>,
## # `POJEMNOSC.SKOKOWA_[cm3]` <dbl>, ROK.PRODUKCJI <dbl>, `PRZEBIEG_[km]`
## # <dbl>, KOLOR <chr>, L.DZRZWI <dbl>, RODZAJ.PALIWA <chr>,
## # SKRZYNIA.BIEGOW <chr>, NAPED <chr>, KRAJ.AKTUALNEJ.REJESTRACJI <chr>,
## # KRAJ.POCHODZENIA <chr>, STATUS.POJAZDU.SPROWADZONEGO <chr>,
## # PIERWSZY.WLASCICIEL <dbl>, KTO.SPRZEDAJE <chr>, STAN <chr>,
## # SERWISOWANY <dbl>, ABS <dbl>, KOMPUTER.POKLADOWY <dbl>, ESP <dbl>,
## # KLIMATYZAJCA <dbl>, BEZWYPADKOWY <dbl>, USZKODZONY <dbl>
```

## Podzbiór danych

W kolejnym, kroku po przyjrzeniu się zbiorowi danych, użytkownik musi zdecydować na których danych ilościowych chce pracować - ważna jest znajomość danych. Dodatkowo pierwszą kolumną musi być kolumna zawierająca numery indeksów obiektów, ze względu na to, że w wyniku zastosowania funkcji odpowiedzialnej

za porządkowanie, zostaną zwrócone w kolejności malejącej numery indeksów, mówiące o kolejności uporządkowania. W związku z tym, za pomocą poniższej procedury użytkownik tworzy podzbiór zaimportowanego zbioru, gdzie w miejsce “” wpisuje nazwy kolumn zawierających zmienne ilościowe, wybrane do porządkowania (przyjmijmy założenie, że podzbiór będzie nazywał się dane\_porzadkowanie - będzie to pomocne w dalszej części programu). U mnie wybranymi kolumnami są: cena, moc, pojemność, rok produkcji, przebieg.

```
dane_porzadkowanie<-zbior_danych[c("Nr", "CENA.BRUTTO_[pln]", "MOC_[km]",  
                                "POJEMNOSC.SKOKOWA_[cm3]",  
                                "ROK.PRODUKCJI", "PRZEBIEG_[km]")]
```

## Transformacje danych

Przed samym porządkowaniem, wymagane jest aby zmienne miały charakter stymulant oraz by zostały poddane transformacji normalizacyjnej. Aby funkcja dokonująca porządkowania dawała poprawny wynik, użytkownik musi zająć się transformacją przed jej zastosowaniem. Poniżej podałam tego przykład. Dla zmiennych które stymulantami nie są, należy dokonać stymulacji. Wśród moich zmiennych poddanych porządkowaniu, do stymulant nie należy zmienna zmienna: przebieg - jest destymulantą, w związku z tym, została przekształcona na stymulantę, za pomocą przekształcenia ilorazowego.

```
stymulacja_przekształcenie_ilorazowe<-function(x,y){  
  for (i in 1:nrow(x)){  
    x[i,which(colnames(x)==y)]=1/x[i,which(colnames(x)==y)]  
  }  
  return(x)  
}
```

Gdy użytkownik chce skorzystać z tej funkcji, w miejsce x musi wpisać nazwę zbioru, a w miejsce y nazwę kolumny w “”, którą chce poddać stymulacji. UWAGA - kolumny wymagające stymulacji, muszą zostać osobno poddane działaniu poniższej funkcji, dodatkowo po każdym zastosowaniu funkcji, należy nadpisać zbiór by zmiany zostały zapisane.

```
dane_porzadkowanie<-stymulacja_przekształcenie_ilorazowe(dane_porzadkowanie, "PRZEBIEG_[km]")
```

W celu uzyskania porównywalności między zmiennymi, zostały one poddane transformacji normalizacyjnej - unitaryzacja

```
unitaryzacja<-function(x){  
  maksi=0  
  minim=0  
  for (j in 2:ncol(x)){  
    maksi[j]=max(x[j])  
    minim[j]=min(x[j])  
    for (i in 1:nrow(x)){  
      x[i,j]=(x[i,j]-minim[j])/(maksi[j]-minim[j])  
    }  
  }  
  return(x)  
}  
  
dane_porzadkowanie<-unitaryzacja(dane_porzadkowanie)
```

## Nieliniowe porządkowanie przy pomocy metod aglomeracyjnych

Chcąc zastosować funkcję `agnes`, należy w pierwszej kolejności wyznaczyć macierz odległości pomiędzy wszystkimi parami obiektów. Do wyznaczenia odległości zostanie użyta metryka euklidesowa - w tym celu zostanie wykorzystana funkcja `dist(x, method="")` - w miejsce `x` należy wpisać nazwę tabeli zawierających dane do uporządkowania, a w nawiasie `[,]` na miejscu drugiej współrzędnej należy podać wektor kolumn, na podstawie którego wartości zostanie wyznaczona macierz odległości. W miejsce argumentu `method` należy wpisać nazwę metryki na podstawie której zostanie obliczona odległość - w naszym przypadku będzie to `euclidean` - euklidesowa.

```
odleglosci <- dist(dane_porzadkowanie[,c("CENA.BRUTTO_[pln]", "MOC_[km]",
    "POJEMNOSC.SKOKOWA_[cm3]", "ROK.PRODUKCJI", "PRZEBIEG_[km]"]],
```

Następnie należy zaimportować bibliotekę `cluster`, by móc skorzystać z metod aglomeracyjnych. Jak już zostało wspomniane we wstępie, wywołanie metod aglomeracyjnych odbywa się dzięki funkcji `agnes(x, method="")`. W miejsce argumentu `x` - zostanie podana wyznaczona macierz odległości z kolei, w kolejnym argumencie - `method` zostanie podana reguła wyznaczania odległości pomiędzy nową grupą a pozostałymi obiektów. Regułą tą może być metoda najbliższego sąsiedztwa, najdalszego, Warda lub średniej między grupowej. My wykorzystamy metodę najbliższego sąsiedztwa oraz najdalszego.

```
library(cluster)
metoda_najblizszego <- agnes(odleglosci, method = "single")
metoda_najdalszego<- agnes(odleglosci, method = "complete")
```

Ostatni etap to graficzne zaprezentowanie wyniku w postaci dendrogramu. W tym celu należy zaimportować bibliotekę `factoextra`, w której to jest funkcja `fviz_dend(x, main = "")`. W miejsce argumentu `x` należy wpisać nazwę obiektu powstałego przy pomocy funkcji `agnes`, `main` to tytuł wykresu. Dodatkowo na osi pionowej zaprezentowane są odległości między obiektami, z kolei na osi poziomej znajdują się numery indeksów obiektów.

Dendrogram dla metody najbliższego sąsiada:

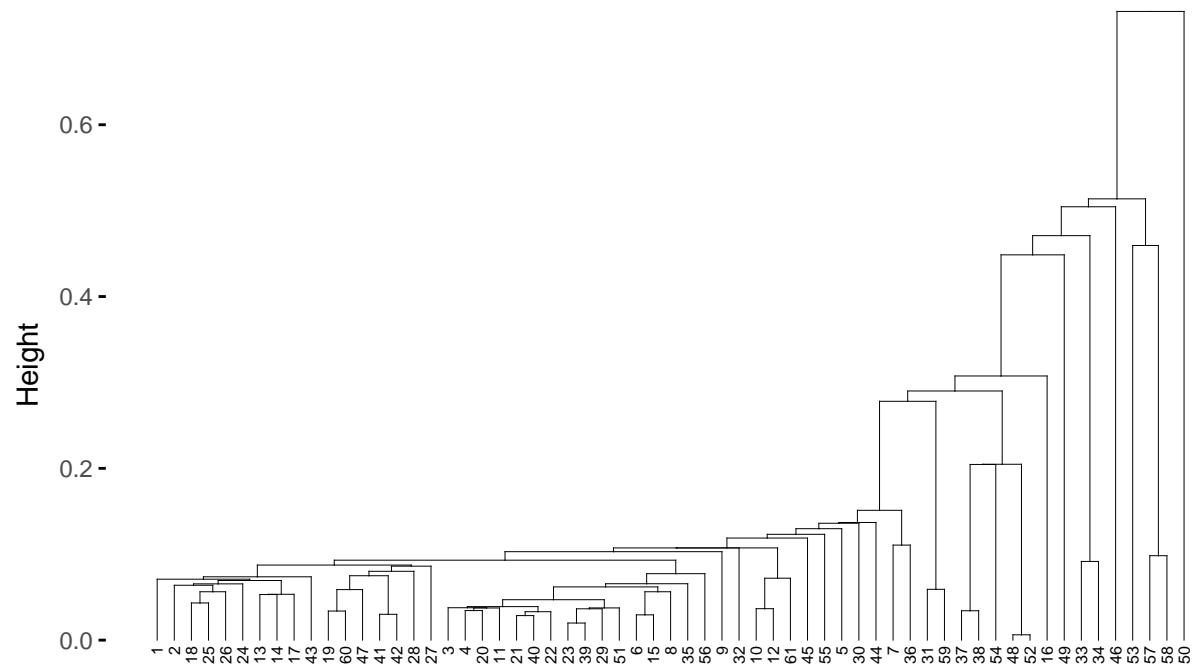
```
library(factoextra)
```

```
## Loading required package: ggplot2

## Welcome! Related Books: `Practical Guide To Cluster Analysis in R` at https://goo.gl/13EFCZ

fviz_dend(metoda_najblizszego, lwd=0.1, cex=0.45, main = "Metoda najbliższego sąsiedztwa")
```

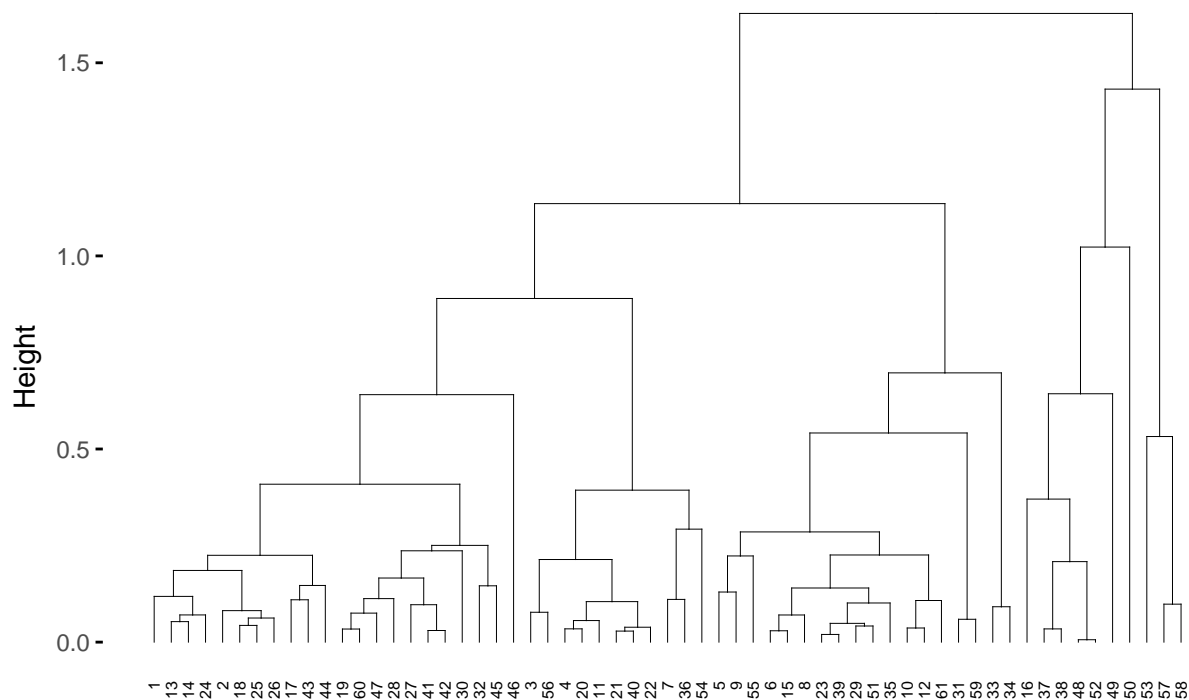
## Metoda najbliższego sąsiedztwa



Dendrogram dla metody najdalszego sąsiada:

```
fviz_dend(metoda_najdalszego, lwd=0.1, cex=0.45, main = "Metoda najdalszego sąsiedztwa")
```

## Metoda najdalszego sąsiedztwa



## Porównanie wyników uporządkowania

W przypadku dendrogramu uzyskanego metodą najdalszego sąsiedztwa można zauważyć większą odległość wiązań niż dla metody najbliższego sąsiedztwa, które to obrazują odległość między grupami. Dodatkowo obiekty charakteryzujące się najbardziej korzystnymi wartościami zmiennych, zostały pogrupowane w jedną grupę (tu mowa o obiektach o numerach indeksów 49, 50, 53, 57, 58), natomiast w przypadku metody najbliższego sąsiedztwa obiekty te zostały rozdzielone w pojedyncze grupy. Można również zauważyć, że w przypadku uporządkowania metodą najdalszego sąsiedztwa obiekty tworzą pewnego rodzaju pogrupy, skupiska, z kolei w przypadku uporządkowania metodą najbliższego sąsiada, wynik porządkowania można porównać do wyglądu lawiny, lub góry, u której podnóża znajdują się obiekty o najniższych wartościach opisywanych przez zmienne, które to łączą się i dochodzą do szczytu, który stanowią obiekty o najwyższych wartościach opisywanych przez zmienne.