

Dokumentacja użytkownika

Opis pakietu R (SOaCRaport)

"Seriously, it doesn't have to be about sharing your code (although that is an added benefit!). It is about saving yourself time."

Hilary Parker

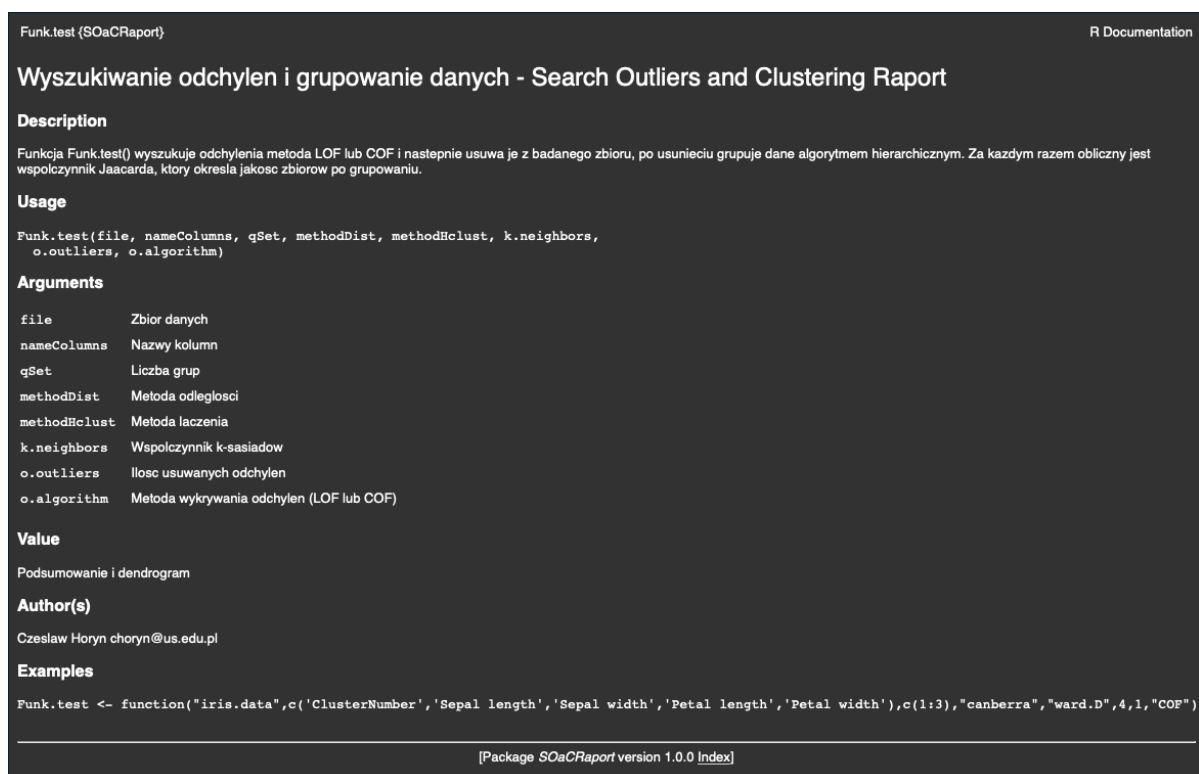
(a quote from R packages by Hadley Wickham)

Pakiety są podstawowymi jednostkami odtwarzalnego kodu w R. Obejmują one funkcje R do wielokrotnego użytku, dokumentację opisującą sposób ich użycia oraz przykładowe dane. Pakiety są przydatne, nawet jeśli nigdy nie udostępniasz swojego kodu, chodzi o oszczędność czasu. Filozofia pakietu opiera się na założeniu, że wszystko co może być zautomatyzowane powinno być zautomatyzowane.

Pakiet SOaCRaport - zasada działania.

Na potrzeby eksperymentów zaimplementowano pakiet SOaCRaport (ang. Search Outliers and Clustering Raport), który zawiera funkcję główną **Funk.test()** oraz funkcje pomocnicze.

Poniżej zademonstrowano wywołanie komendy `help("Funk.test")`.



```
Funk.test (SOaCRaport) R Documentation

Wyszukiwanie odchylen i grupowanie danych - Search Outliers and Clustering Raport

Description
Funkcja Funk.test() wyszukuje odchylenia metoda LOF lub COF i następnie usuwa je z badanego zbioru, po usunieciu grupuje dane algorytmem hierarchicznym. Za kazdym razem obliczony jest wspolczynnik Jaacarda, który okresla jakosc zbiorow po grupowaniu.

Usage
Funk.test(file, nameColumns, qSet, methodDist, methodHclust, k.neighbors,
  o.outliers, o.algorithm)

Arguments
file          Zbior danych
nameColumns   Nazwy kolumn
qSet          Liczba grup
methodDist    Metoda odleglosci
methodHclust  Metoda laczenia
k.neighbors   Wspolczynnik k-sasiadow
o.outliers    Ilosc usuwanych odchylen
o.algorithm   Metoda wykrywania odchylen (LOF lub COF)

Value
Podsumowanie i dendrogram

Author(s)
Czeslaw Horyn choryn@us.edu.pl

Examples
Funk.test <- function("iris.data",c('ClusterNumber','Sepal length','Sepal width','Petal length','Petal width'),c(1:3),"canberra","ward.D",4,1,"COF")

[Package SOaCRaport version 1.0.0 Index]
```

Rysunek 1: Wywołanie komendy `help("Funk.test")`. Źródło: Zrzut ekranu z programu RStudio.

Funkcji `Funk.test()` możemy podawać wybrane miary odległości:

- "euclidean",
- "maximum" - czybyszewa,
- "manhattan",
- "canberra".

i dla każdej z czterech miar sprawdzać siedem wybranych metod łączenia:

- "single",
- "mcquitty",
- "average",
- "centroid",
- "median",
- "complete",
- "ward.D".

Opisane wyżej kombinacje przedstawiono w osobnych tabelach z wynikami dla każdego badanego zbioru, opisanego w punkcie 5.3 Eksperymenty.

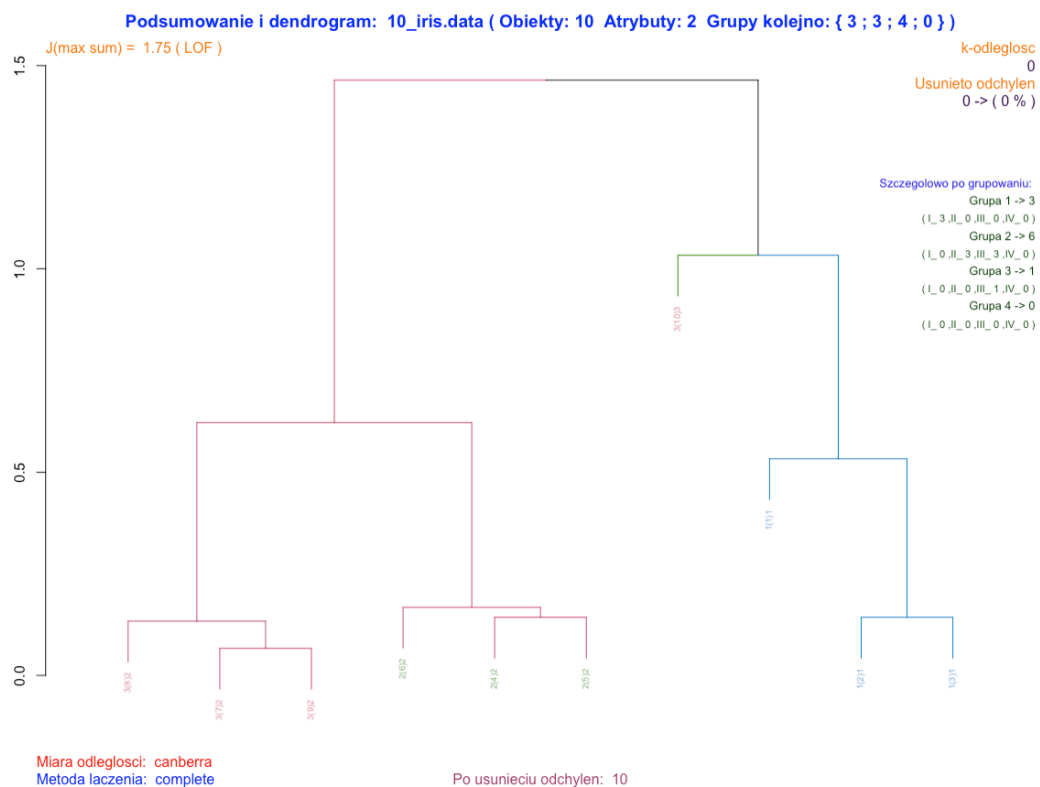
Głównym celem funkcji głównej jest możliwość przeszukania 1600 zadanych przypadków dla każdej omówionej wyżej kombinacji. Przypadków możemy oczywiście zadać więcej (zwiększając parametr ***k*** lub ***o***), każdy z zadanych przypadków to zmieniający się parametr ***k*** (***k***-najbliższych sąsiadów) w przedziale $< 0 ; 40 >$ oraz parametr ***o*** (odchylenia) w przedziale $< 0 ; 40 >$ informujący funkcję, ile obiektów odstających (outliers) należy usunąć ze zbioru przed grupowaniem. Funkcja porządkuje wykryte odchylenia - w zależności od wybranej metody LOF lub COF - sortując je od tych najbardziej odstających do najmniej w danym zbiorze, a następnie usuwa te najbardziej odstające w liczbie zadanej przez parametr ***o***.

Za każdym razem funkcja porównuje wykorzystując do tego współczynnik Jaccarda każdą grupę badanego właśnie zbioru danych do grup jakie powstały po grupowania algorytmem hierarchicznym aglomeracyjnym. Zatem każda grupa, która powstała po podziale (grupowaniu) jest porównywana ze znaną grupą wzorcową sprzed procesu grupowania i obliczany jest współczynnik Jaccarda ***J*** dla takiego porównania. Najlepsze wyniki takich indywidualnych porównań są sumowane i otrzymujemy wartość zbiorczą ***J (sum)*** dla całego badanego właśnie zbioru danych przy określonych w danym przebiegu parametrach ***k*** i ***o***.

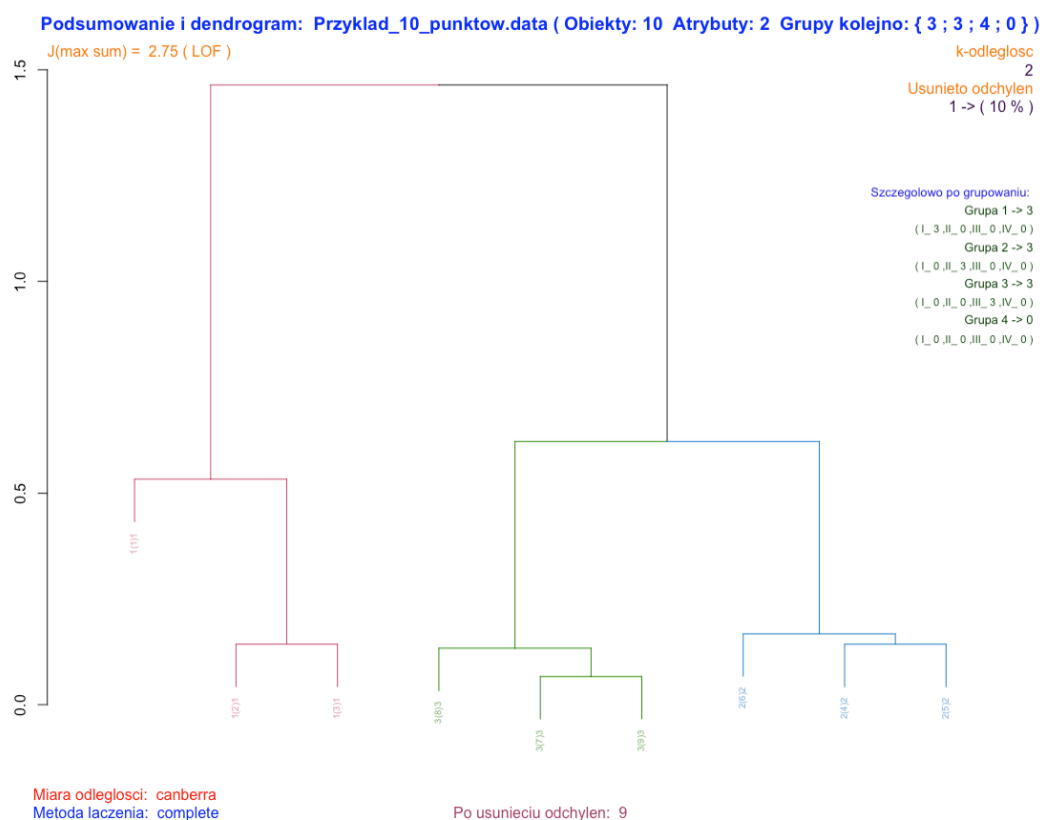
Na koniec analizując wszystkie 1600 wartości ***J (sum)*** funkcja wybiera maksymalną wartość nazwaną: ***J (max sum)***, znając jednocześnie wartość parametrów ***k.max*** i ***o.max***, dla których ta maksymalna wartość wystąpiła. Sposób wyliczania ***J (max sum) = J_XYZ_MAX*** zaprezentowano na przykładzie zbioru IRIS na rysunkach 35 i 36 - przedstawiono tam wizualizacje działania funkcji. Jeżeli ***J (max sum)*** jest największe dla wartości ***o.max = 0*** to oznacza, że usunięcie jakichkolwiek wykrytych odchyleń, przy jakimkolwiek zadanym parametrze ***k*** nie poprawiło jakości grupowania.

Wynikiem działania funkcji jest raport końcowy składający się z podsumowania i dendrogramu. Przykład raportu (rysunek 2 i 3) zaprezentowano na następnej stronie dla przykładowego zbioru danych z rysunku 11 (w pracy). Schemat przedstawiający działanie kodu źródłowego w pakiecie SOaCRaport zademonstrowano na schemacie, rysunek 4.

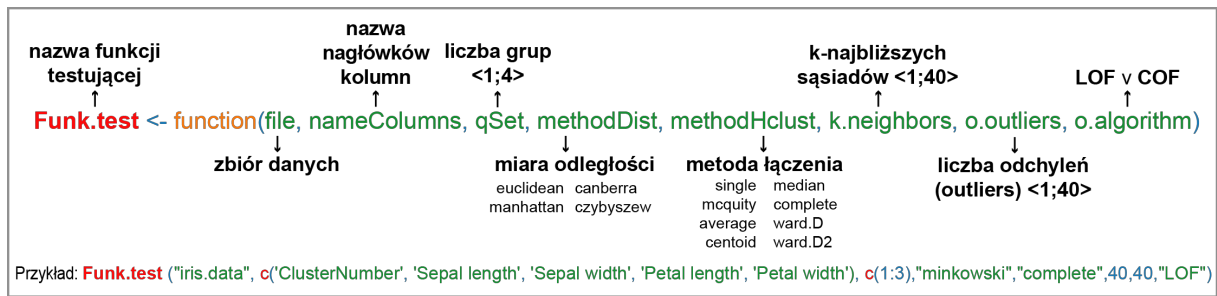
Wyniki eksperymentów (do badań wykorzystano zbiory: Iris, Wine i Breast Tissue z UCI Machine Learning Repository) oraz pozostałe raporty dla badanych zbiorów z zastosowaniem funkcji przedstawiono w punkcie 5.3. Eksperymenty.



Rysunek 2: Podsumowanie i dendrogram: Przykład_10_punktow.data z odchyleniami. Źródło: Obliczenia własne – przykładowy wynik funkcji Funk.test() z pakietu SOaCRaport.



Rysunek 3: Podsumowanie i dendrogram: Przykład_10_punktow.data bez odchyliń. Źródło: Obliczenia własne – przykładowy wynik funkcji Funk.test() z pakietu SOaCRaport.



- Funkcja wczytująca zbior danych**
`read.data <- function(file_data, nameColumns)`

```
data_inf <- read.table(file=file_data, header=FALSE,
  sep=' ', stringsAsFactors=FALSE, col.names=nameColumns)
data_infTrain <- data_inf[, which(names(data_inf) != "ClusterNumber")]
```
- Funkcja pomocnicza zliczająca w pętli liczbę obserwacji w poszczególnych grupach**
`Collection <- function(data_inf2, c)`

```
for (l in qSet) {
  name <- paste('Col', l, sep='')
  totalCollection[[name]] <- Collection(l) + length(Collection(l-1))
  + length(Collection(l-2)) + length(Collection(l-3)) }
```
- Funkcje wyszukujące odchylenia**
`LOF.outliers <- function(data_set, k)` `COF.outliers <- function(data_set, k)`
 ,gdzie k = k-najbliższych sąsiadów

```
if (o.algorithm == "LOF") {
  outlier.scores <- lofactor(data_set, k)
  outliers <- order(outlier.scores, decreasing=T) }

if (o.algorithm == "COF") {
  outlier_score <- COF(data_set, k)
  names(outlier_score) <- 1:nrow(data_set)
  outliers_sort <- sort(outlier_score, decreasing = TRUE)
  outliers <- c(as.numeric(names(outliers_sort))) }
```
- Funkcja usuwająca odchylenia przed grupowaniem**
`delete.outliers2 <- function(data_inf, o)`
 ,gdzie o - liczba usuwanych wartości odstających

```
while(o < o.outliers) { o <- o + 1
  if (o == 0) {
    data_infTrain <- data_inf[, which(names(data_inf) != "ClusterNumber")] }
  if (o > 0) {
    p <- outliers[1: o]
    data_infTrain <- data_inf[-p, which(names(data_inf) != "ClusterNumber")] }
```
- Funkcja obliczająca odległości między obiektami**
`fun.dist <- function(data_infTrain, methodDist)`
 ,gdzie methodDist wybrana miara odległości

```
d_data_infTrain <- dist(data_infTrain, method = methodDist)
```
- Funkcja łącząca dwa najbliższe skupienia i przycinająca drzewo**
`linkage_cutree <- function(d_data_infTrain, methodHclust, qSet)`
 ,gdzie methodHclust wybrana metoda łączenia

```
hc_data_infTrain <- hclust(d_data_infTrain, method = methodHclust)
hc_data_infTrainCutree <- cutree(hc_data_infTrain, k=length(qSet))
```

7

Funkcja zliczająca zbiory po usunięciu odchyleń i po nowym podziale

```
CollectionAfterDivision <- function(hc_data_infTrainCutree,m)
,gdzie m - etykieta danego zbioru
```

```
for (i in 1:length(hc_data_infTrainCutree)) {
  if (hc_data_infTrainCutree[i] == m)
    x[i] <- i }
x[!is.na(x)] }
for (t in qSet) {
  name <- paste("X", t, sep = ")
  totalAfterDivision[[name]] <- CollectionAfterDivision(t) }
```

8

Funkcja obliczająca indeksy Jaccarda (J), określające podobieństwo między zbiorami przed i po podziale (analizująca wszystkie przypadki) i wybierająca maksymalną wartość J(max sum) = J_XYZA_MAX

$$J = \frac{\text{intersect}(\text{Coli}, \text{Xi})}{\text{union}(\text{Coli}, \text{Xi})}$$

J <- function () - funkcja zwraca J_XYZA_MAX = J (max sum)

#Przykład dla jednego zbioru

```
J.A_X1 <- round((length(intersect(totalCollection[["Col1"]],totalAfterDivision[["X1"]])))/
  (length(union(totalCollection[["Col1"]],totalAfterDivision[["X1"]]))),digits = 4)
J.A_Y2 <- round((length(intersect(totalCollection[["Col1"]],totalAfterDivision[["X2"]])))/
  (length(union(totalCollection[["Col1"]],totalAfterDivision[["X2"]]))),digits = 4)
J.A_Z3 <- round((length(intersect(totalCollection[["Col1"]],totalAfterDivision[["X3"]])))/
  (length(union(totalCollection[["Col1"]],totalAfterDivision[["X3"]]))),digits = 4)
#Przykład - rozważamy możliwe przypadki dla trzech zbiorów
J.XYZ_I <- round(sum(c(J.A_X1, J.B_Y2, J.C_Z3), na.rm=TRUE), digits = 4)
J.XZY_II <- round(sum(c(J.A_X1, J.B_Z3, J.C_Y2), na.rm=TRUE), digits = 4)
J.YXZ_III <- round(sum(c(J.A_Y2, J.B_X1, J.C_Z3), na.rm=TRUE), digits = 4)
J.YZX_IV <- round(sum(c(J.A_Y2, J.B_Z3, J.C_X1), na.rm=TRUE), digits = 4)
J.ZXY_V <- round(sum(c(J.A_Z3, J.B_X1, J.C_Y2), na.rm=TRUE), digits = 4)
J.ZYX_VI <- round(sum(c(J.A_Z3, J.B_Y2, J.C_X1), na.rm=TRUE), digits = 4)
#Wybieramy maksymalną sumę
J <- J_XYZ_MAX <- max(J.XYZ_I, J.XZY_II, J.YXZ_III, J.YZX_IV, J.ZXY_V, J.ZYX_VI, na.rm=TRUE)
```

9

Funkcja zapisująca wyniki do pliku Excel

```
writelnExcel.xlsx<- function (df_total_)
```

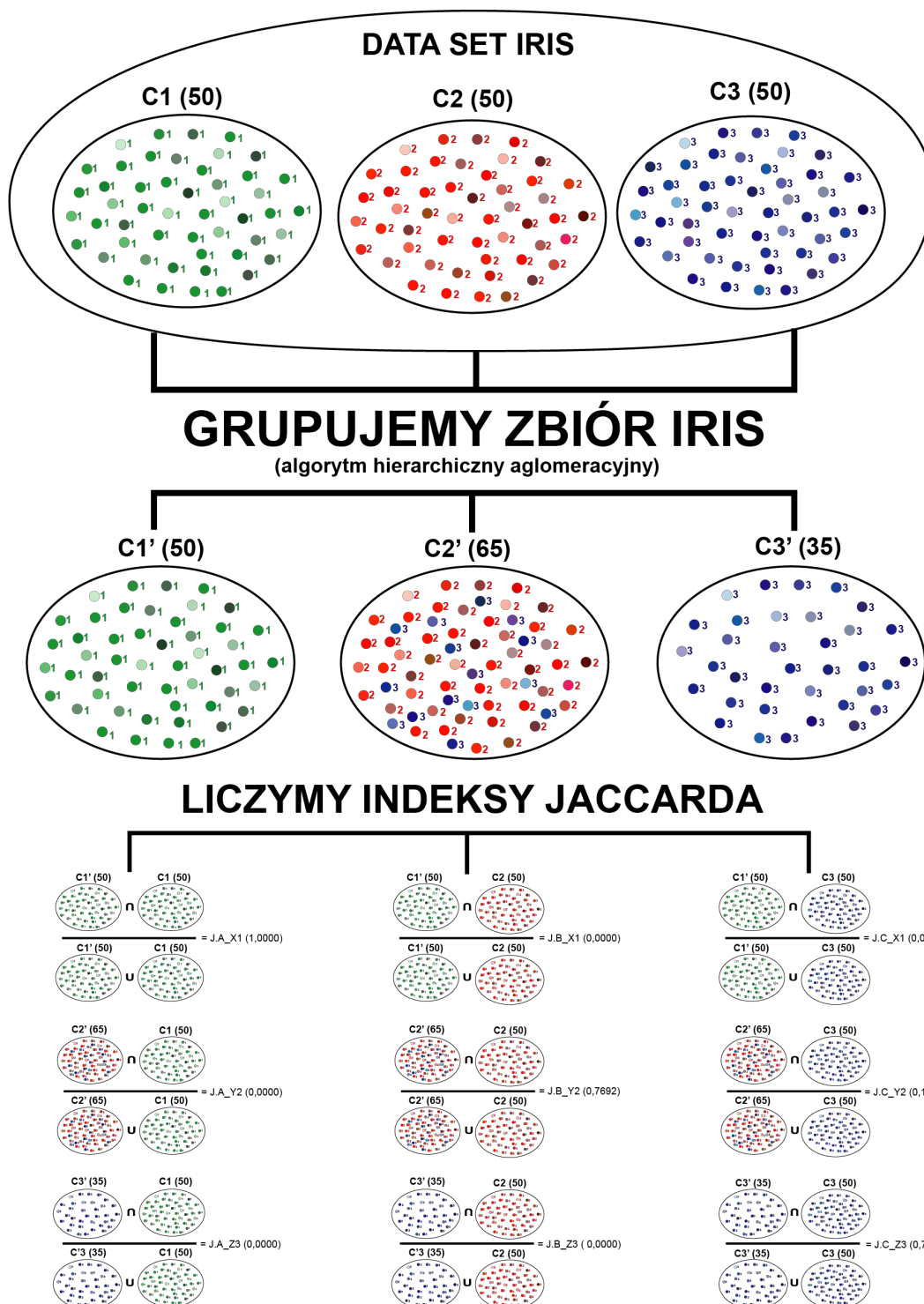
```
write.xlsx(df_total_, file="ZRZUT.xlsx", sheetName = "Arkusz1",
col.names = TRUE, row.names = TRUE, append = FALSE)
```

10

Dendrogram i posumowanie dla danego współczynnika osobliwości (LOF lub COF) oraz najlepszych parametrów k.max i o.max - fragment kodu

```
Train_species <- data_inf[,1]
dend <- as.dendrogram(hc_data_infTrain)
dend %>% head
dend <- color_branches(dend, k=length(qSet))
groupLabels = Train_species
labels_colors(dend) <- rainbow_hcl(length(qSet)) [sort_levels_values(as.numeric(data_inf[,1])
[order.dendrogram(dend)])]
labels(dend) <- paste(as.character(data_inf[,1])
[order.dendrogram(dend)],"(",labels(dend),")", (hc_data_infTrainCutree_num)
[order.dendrogram(dend)], sep = "")
dend <- hang.dendrogram(dend,hang_height=0.1)
dend <- set(dend, "labels_cex", 1.0)
plot(dend,main = "Podsumowanie i dendrogram", horiz = FALSE, nodePar = list(cex = 0.007))
```

Rysunek 4: Schemat przedstawiający działanie kodu źródłowego w pakiecie. Źródło: Opracowanie własne.



ROZWAŻAMY 6 MOŻLIWYCH PRZYPADKÓW

$J.XYZ_I = \text{SUM}(J.A_X1, J.B_Y2, J.C_Z3) (2,4692)$
 $J.XZY_II = \text{SUM}(J.A_X1, J.B_Z3, J.C_Y2) (1,0000)$
 $J.YXZ_III = \text{SUM}(J.A_Y2, J.B_X1, J.C_Z3) (0,7000)$
 $J.YZX_IV = \text{SUM}(J.A_Y2, J.B_Z3, J.C_X1) (0,0000)$
 $J.ZXY_V = \text{SUM}(J.A_Z3, J.B_X1, J.C_Y2) (0,0000)$
 $J.ZYX_VI = \text{SUM}(J.A_Z3, J.B_Y2, J.C_X1) (0,7692)$

Wybieramy

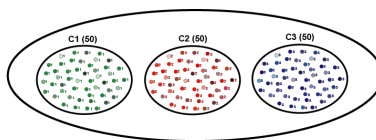
maksymalną sumę

$J_XYZ_MAX = \max(J.XYZ_I, J.XZY_II, J.YXZ_III, J.YZX_IV, J.ZXY_V, J.ZYX_VI)$

(2,4692)

Rysunek 5: Schemat przedstawiający działanie kodu źródłowego w pakiecie. Źródło: Opracowanie własne.

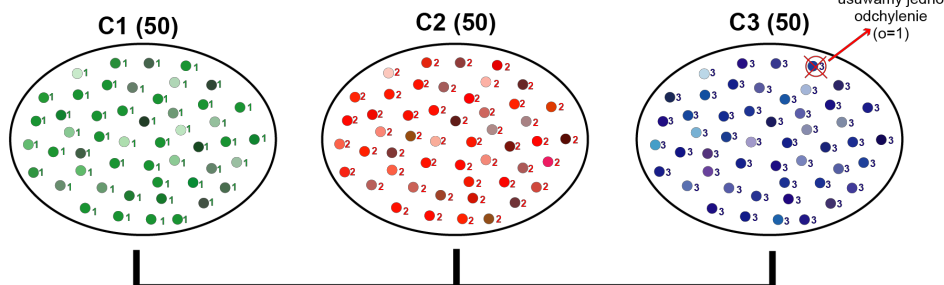
DATA SET IRIS



Wyszukujemy odchylenia (algorytm LOF lub COF) rozważamy
1600 przypadków dla $k < 1; 40$ i $o < 1; 40$

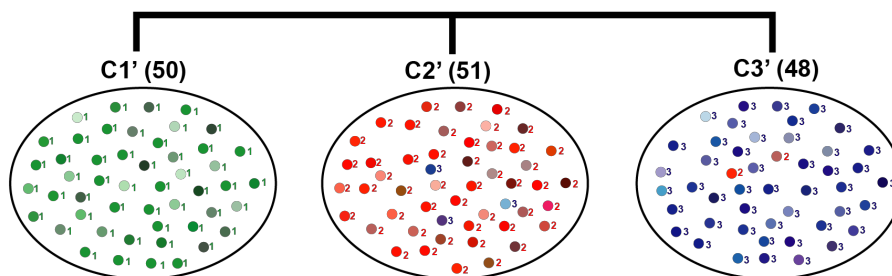
Dla każdego przypadku usuwamy odchylenia, grupujemy i wyliczamy index Jaccarda (J_{XYZ_MAX})

Szukamy największej wartości (J_{XYZ_MAX}), w naszym przypadku największa wartość
(J_{XYZ_MAX}) jest dla $k = 4$, $o = 1$ (COF)

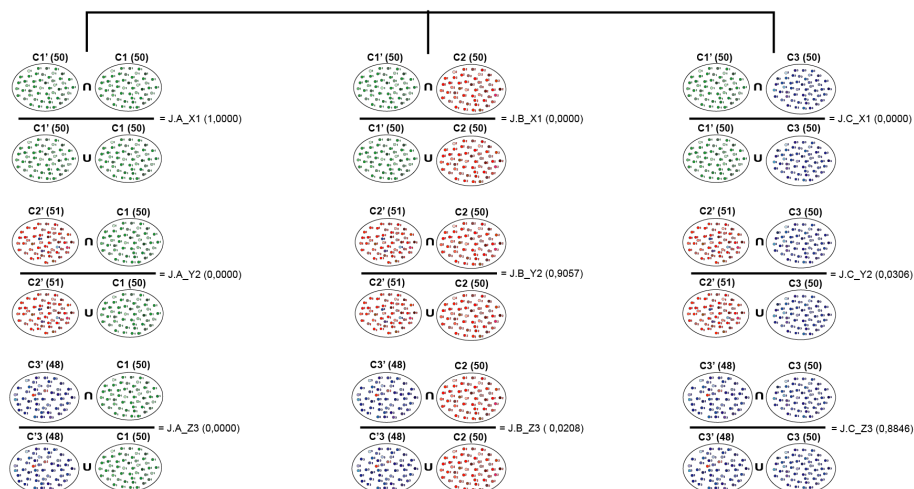


GRUPUJEMY ZBIÓR IRIS

(algorytm hierarchiczny aglomeracyjny)



LICZYMY INDEKSY JACCARDA



ROZWAŻAMY 6 MOŻLIWYCH PRZYPADKÓW

$$J.XYZ_I = \text{SUM}(J.A_X1, J.B_Y2, J.C_Z3) (2,7903)$$

$$J.XZY_II = \text{SUM}(J.A_X1, J.B_Z3, J.C_Y2) (1,0514)$$

$$J.YXZ_III = \text{SUM}(J.A_Y2, J.B_X1, J.C_Z3) (0,8846)$$

$$J.YZX_IV = \text{SUM}(J.A_Y2, J.B_Z3, J.C_X1) (0,0208)$$

$$J.ZXY_V = \text{SUM}(J.A_Z3, J.B_X1, J.C_Y2) (0,0306)$$

$$J.ZYX_VI = \text{SUM}(J.A_Z3, J.B_Y2, J.C_X1) (0,9057)$$

Wybieramy

maksymalną sumę

$$J_{XYZ_MAX} = \max(J.XYZ_I, J.XZY_II, J.YXZ_III, J.YZX_IV, J.ZXY_V, J.ZYX_VI)$$

(2,7903)

Rysunek 6: Schemat przedstawiający działanie kodu źródłowego w pakiecie. Źródło: Opracowanie własne.

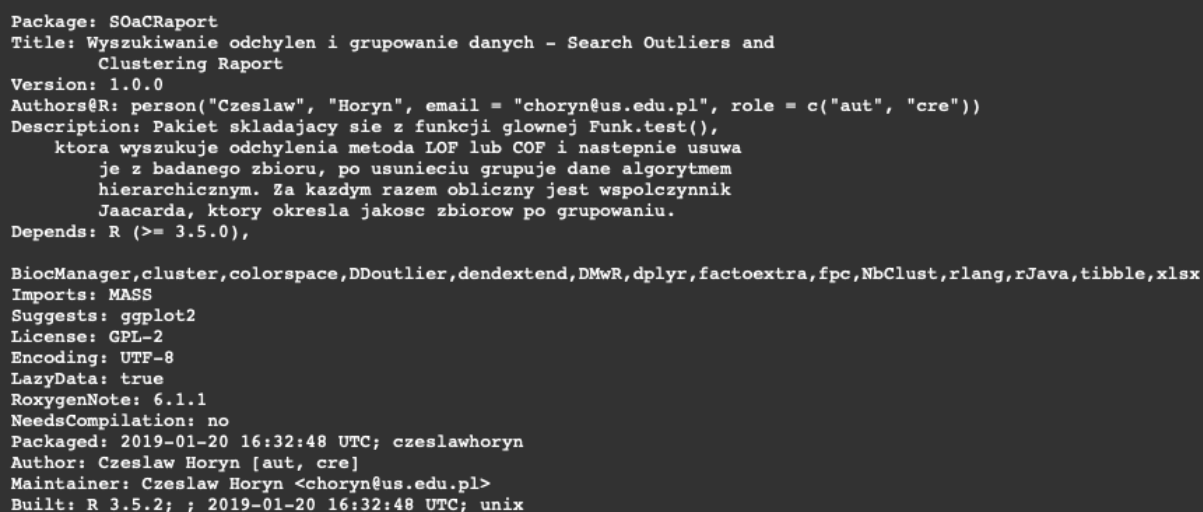
Pakiet SOaCRaport - dokumentacja użytkownika

DESCRIPTION to prosty plik w formacie DCF (format kontrolny Debiana). Plik zawiera informacje o pakiecie, takie jak nazwa, wersja, autor oraz listę innych pakietów wymaganych do działania. Zrzut ekranu pliku DESCRIPTION z pakietu SOaCRaport przedstawia rysunek 7. DESCRIPTION w pakiecie SOaCRaport zawiera zarówno pakiety wymagane wymienione w polach Depends i Imports jak i polecane (choć niewymagane) wypisane w polu Suggests. Aby skorzystać z pakietu SOaCRaport należy zainstalować wszystkie wymagane pakiety. Instalując pakiet SOaCRaport wymagane pakiety powinny zainstalować się automatycznie, w przypadku problemów poniższymi poleceniami instalujemy wszystkie pakiety (z pól Depends, Imports i Suggests):

```
install.packages ( c("cluster", "colorspace", "DDoutlier", "dendextend", "DMwR", "dplyr",  
"factoextra", "fpc", "NbClust", "rJava", "xlsx", "ggplot2", "MASS" ))
```

Aby zainstalować pakiet "BiocManager" R wersja ("3.5") oraz funkcję "genefilter" należy użyć następującego kodu:

```
if (!requireNamespace("BiocManager", quietly = TRUE))  
  install.packages("BiocManager")  
BiocManager::install("genefilter", version = "3.8")1
```



```
Package: SOaCRaport  
Title: Wyszukiwanie odchylen i grupowanie danych - Search Outliers and  
        Clustering Raport  
Version: 1.0.0  
Authors@R: person("Czesław", "Horyn", email = "choryn@us.edu.pl", role = c("aut", "cre"))  
Description: Pakiet składający się z funkcji głównej Funk.test(),  
        która wyszukuje odchylenia metoda IOF lub COF i następnie usuwa  
        je z badanego zbioru, po usunięciu grupuje dane algorytmem  
        hierarchicznym. Za każdym razem obliczony jest współczynnik  
        Jaacarda, który określa jakość zbiorów po grupowaniu.  
Depends: R (>= 3.5.0),  
  
BiocManager, cluster, colorspace, DDoutlier, dendextend, DMwR, dplyr, factoextra, fpc, NbClust, rlang, rJava, tibble, xlsx  
Imports: MASS  
Suggests: ggplot2  
License: GPL-2  
Encoding: UTF-8  
LazyData: true  
RoxygenNote: 6.1.1  
NeedsCompilation: no  
Packaged: 2019-01-20 16:32:48 UTC; czeslawhoryn  
Author: Czesław Horyn [aut, cre]  
Maintainer: Czesław Horyn <choryn@us.edu.pl>  
Built: R 3.5.2; ; 2019-01-20 16:32:48 UTC; unix
```

Rysunek 7: plik DESCRIPTION. Źródło: Zrzut ekranu z RStudio.

W polu **Title** zamieszczono krótki opis pakietu, a w polu **Description** bardziej wyczerpujący. Pole **Version** zawiera numer wersji pakietu, **Author** - nazwisko autora, a w polu **License** znajdują się informacje licencyjne.

Zainstalowane pakiety należy załadować poleceniem **library**, przykładowo:

```
library(DMwR); library(rJava); library(xlsx); library(dplyr); library(dendextend);  
library(factoextra); library(SOaCRaport); library(cluster); library(colorspace); library(DDoutlier)
```

Folder z dysku CD o nazwie SOaCRaport należy skopiować do katalogu głównego naszego komputera. Przykładowo w systemie Windows -> Biblioteki\Dokumenty, w przypadku

¹ W przypadku starszych wersji R należy zapoznać się z informacją na stronie:
<http://bioconductor.org/packages/release/bioc/html/genefilter.html>

systemu Mac OS X -> Katalog domowy, następnie najwygodniej zainstalować pakiet pomocniczy "devtools", który posłuży do instalacji pakietu SOaCRaport. Wykonujemy to następującymi poleceniami: `install.packages(c("devtools"))` i wczytujemy bibliotekę poleceniem: `library(devtools)`.

Następnie instalujemy pakiet wykonując następujące kroki (zalecane R w wersji 3.5.2):

1. Zainstaluj pakiet "rJava" i sprawdź czy można załadować `library(rJava)`.
2. Wykonaj polecenie:

```
if (!requireNamespace("BiocManager", quietly = TRUE))
  install.packages("BiocManager")
BiocManager::install("genefilter", version = "3.8")
```
3. Jeśli tego nie zrobiono wcześniej, wykonaj:
`install.packages(c("devtools"))` i wczytaj bibliotekę poleceniem: `library(devtools)`
4. Wykonaj: `install("SOaCRaport")` i wczytaj wydając komendę: `library(SOaCRaport)`

W folderze SOaCRaport na dysku CD znajduje się podkatalog **data**, w podkatalogu tym zamieszczono wykorzystywane w pracy zbiory danych: **iris.data**, **wine.data**, **breast_tissue.data** oraz **Przykład_10_punktow.data**. Zbiory te przed uruchomieniem funkcji głównej należy skopiować do katalogu głównego naszego komputera.

5. Uruchamiamy funkcję główną, przykład:

```
Funk.test("iris.data",c('ClusterNumber','Sepal length','Sepal width','Petal length','Petal width'),
c(1:3),"canberra","ward.D",4,1,"COF")
```

Najczęściej spotykane problemy i zadawane pytania:

W przypadku systemu **MAC OS X** należy zainstalować **XQuartz-2.7.11.dmg** dostępny na stronie: <https://www.xquartz.org/>

Zapis danych do Excela przy pomocy "rJava". Pakiet "rJava" wymaga instalacji najnowszej wersji JAVA².

(a) Instrukcja dla MAC OS X - oto proste kroki:

1. Usuń pakiet "rJava": `remove.packages(rJava)`
2. Zamknij R
3. Zainstaluj najnowszą wersję Java na komputerze Mac
4. Otwórz terminal i wpisz polecenie: `sudo R CMD javareconf`
5. Otwórz R i zainstaluj rJava za pomocą polecenia:
`install.packages("rJava", dependencies=TRUE, type="source")`

(b) Instrukcja dla Windows - oto proste kroki:

1. Zainstaluj najnowszą wersję JAVA - domyślna ścieżka instalacji to: C:\Program Files\Java\jre1.8.0_191\bin
2. Otwórz RStudio³ i wpisz polecenie:

```
Sys.setenv(JAVA_HOME = 'C:\\Program Files\\Java\\jre1.8.0_191') # for 64-bit version
Sys.setenv(JAVA_HOME = 'C:\\Program Files (x86)\\Java\\jre7') # for 32-bit version
```

² <https://www.java.com/en/download/manual.jsp>

³ https://pbiecek.gitbooks.io/przewodnik/content/Programowanie/podstawy/jak_zainstalowac_R.html

W przypadku eksperymentów potwierdzających tezę pracy na innych zbiorach danych istotna jest pierwsza kolumna w której należy przypisać poszczególnym obserwacjom numer klastra do którego przynależą (kolumna ta powinna nazywać się ClusterNumber) w zależności od struktury badanego zbioru danych możemy przypisać od 1 do maksymalnie 4 grup.

Linki do wykorzystywanych w badaniach zbiorów z UCI Machine Learning Databases:

Wine:

file <- 'https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data'

Iris:

file <- 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'

Breast_tissue:

file <- 'http://archive.ics.uci.edu/ml/machine-learning-databases/00192/BreastTissue.xls'

Funkcja wymaga wczytania nazw kolumn dla poszczególnych zbiorów danych. Przykładowe nameColumns - numer grupy jako pierwszy zawsze - 'ClusterNumber':

Wine:

```
nameColumns <- c('ClusterNumber', 'Alcohol', 'Malic.acid', 'Ash', 'Alcalinity.of.ash', 'Magnesium', 'Total.phenol', 'Flavanoids', 'NonFlavanoid.phenols', 'Proanthocyanin', 'Color.intensity', 'Hue', 'OD280.OD315.of.diluted.wines', 'Proline')
```

Iris:

```
nameColumns <- c('ClusterNumber', 'Sepal length', 'Sepal width', 'Petal length', 'Petal width')
```

Breast_tissue:

```
nameColumns <- c('ClusterNumber', 'I0', 'PA500', 'HFS', 'DA', 'Area', 'A/DA', 'Max', 'IP', 'P')
```

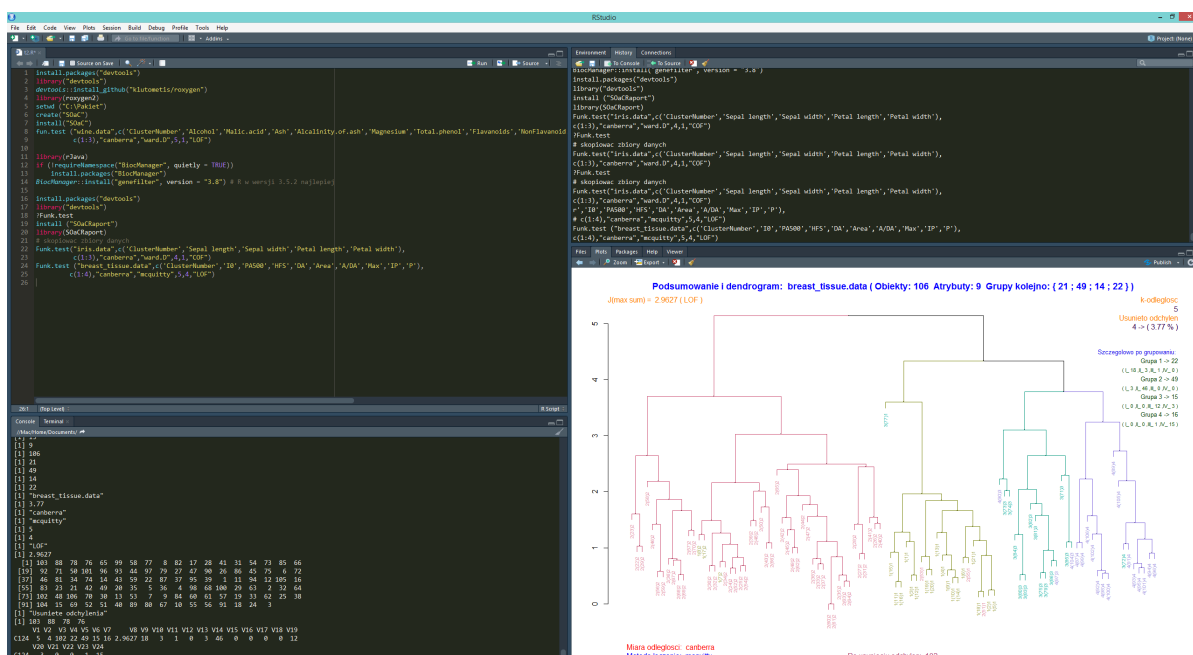
Sposób użycia (ang. Usage):

Funk.test <- function (file, nameColumns, qSet, methodDist, methodHclust, k.neighbors, o.outliers, o.algorithm)

Przykład:

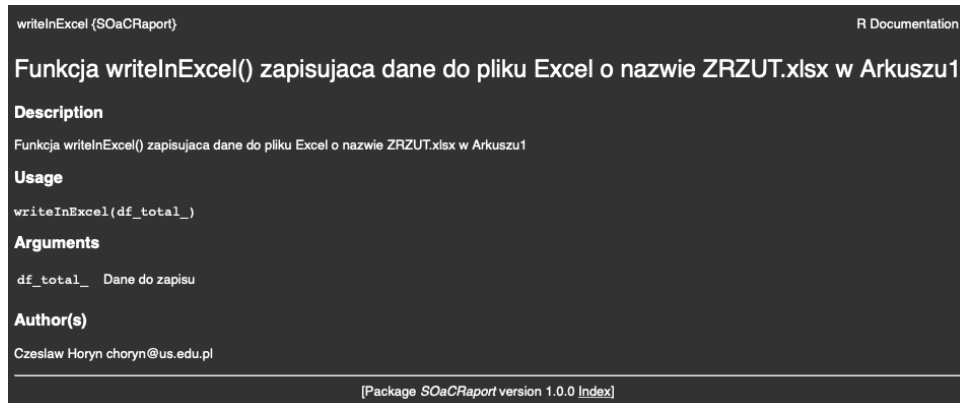
```
Funk.test("iris.data",c('ClusterNumber','Sepal length','Sepal width','Petal length','Petal width'),  
c(1:3),"canberra","ward.D",4,1,"COF")
```

Przykładowy widok po wywołaniu funkcji Funk.test() w programie RStudio pokazano na rysunku 8.



Rysunek 8: Ogólny widok okna w RStudio po wywołaniu funkcji Funk.test(). Źródło: Zrzut z programu RStudio.

Oprócz funkcji głównej w pakiecie znajdują się funkcje pomocnicze współpracujące z funkcją główną, dodatkowe informacje o nich można pozyskać zaglądając do dokumentacji pakietu, rysunek 11. Funkcje te można wykorzystać jako niezależnie działające instrumenty w tym celu w programie R należy wywołać `help('nazwa funkcji')`. Przykładowo wywołanie polecenia `help('writeInExcel')`, pozwoli nam zdobyć informacje, jak wywołać zapisanie danych do pliku Excel, rezultat komendy pokazano na rysunku 9.



Rysunek 9: Wywołanie komendy `help('writeInExcel')`. Źródło: Zrzut ekranu z programu RStudio.

Wynik działania funkcji `writeInExcel()`, rysunek 10:

	A	B	C	D	E	F	G	H	I
1		V1	V2	V3	V4	V5	V6	V7	V8
2	C1	1	0	10	3	6	1	0	1,75
3	C11	1	1	9	3	5	1	0	1,85
4	C12	2	0	10	3	6	1	0	1,75
5	C13	2	1	9	3	3	3	0	2,75

Rysunek 10: Przykładowe dane zapisane w pliku Excel funkcją `writeInExcel`. Źródło: Zrzut ekranu z pliku Excel.

Szczegółowa dokumentacja pakietu 'SOaCRaport' jest dostępna z programu, rysunek 11:



Rysunek 11: Dokumentacja for packages 'SOaCRaport'. Źródło: Zrzut ekranu z programu RStudio.