XV Konferencja PLOUG
Kościelisko
Październik 2009

Wykorzystanie projektu R w zadaniach eksploracji danych

Artur Gramacki, Jarosław Gramacki
Uniwersytet Zielonogórski

a.gramacki@iie.uz.zgora.pl, j.gramacki@iie.uz.zgora.pl

Abstrakt. Projekt R jest obecnie jednym z najdynamiczniej rozwijających się pakietów statystycznych udostępnianych na zasadach licencji GNU GPL. Prekompilowane wersje R dostępne są na platformach Windows, Macintosh i różnych wersjach Linuksa. Może on śmiało konkurować z komercyjnymi pakietami statystycznymi, takimi np. jak Statistica, SPSS, Matlab (Statistical Toolbox) czy też SAS. Niewątpliwą jego przewagą nad pakietami komercyjnymi (oprócz oczywiście bezpłatności i dostępności kodów źródłowych) jest to, że bardzo szybko reaguje on na wszelkie nowości pojawiające się w dziedzinie statystyki (i nie tylko). W wielu kręgach naukowych stał się on niepisanym standardem. Wiele zadań powszechnie kojarzonych z eksploracją danych wywodzi się z szeroko rozumianej statystyki, stąd nie dziwi wykorzystanie R w tym właśnie obszarze. W artykule pokazano jak użyć R w wybranych zadaniach eksploracji danych. Krótko omówiono również sposoby jego skomunikowania z relacyjnymi bazami danych oraz innymi zewnętrznymi repozytoriami danych.

1. Wprowadzenie

Eksploracja danych (ang. data mining, DM) to obecnie bardzo dynamicznie rozwijająca się dziedzina wiedzy. Powszechny dostęp do wydajnych i względnie tanich systemów baz danych powoduje, iż ilość przechowywanych danych jest ogromna i ciągle rośnie. Sztuką staje się obecnie nie sam fakt efektywnego i bezpiecznego przechowywania danych, ale ich łatwe analizowanie i wyciąganie sensownych wniosków. Przy ogromnych ilościach danych nie jest to zadanie trywialne. Lista dostępnego oprogramowania dedykowanego tym zagadnieniom jest bardzo długa [DM-Soft]. Nie wymieniamy w tym miejscu z nazwy żadnego produktu, aby nie faworyzować jednych i jednocześnie nie pomijać innych. Generalnie oprogramowanie DM można podzielić na komercyjne oraz niekomercyjne (to drugie zwykle udostępniane jest na licencji GNU GPL, co z grubsza oznacza, że jest ono całkowicie bezpłatne do wszelkich zastosowań, w tym też komercyjnych oraz udostępniany jest kompletny kod źródłowy).

W artykule skupiamy się na projekcie R[1] [RPro] i możliwościach jego użycia w obszarze DM. R jest generalnie uznawany za oprogramowanie statystyczne i może śmiało konkurować z komercyjnymi pakietami, takimi np. jak Statistica, SPSS, Matlab (Statistical Toolbox), SAS często nawet je przewyższając funkcjonalnością. Powstało i nadal powstaje bardzo dużo oprogramowania w R do zastosowań w wielu różnych dziedzinach nauki, nierzadko bardzo odległych od statystyki [RArch]. Oprogramowanie to jest intensywnie rozwijane przez wielką rzeszę użytkowników[2] z całego świata, głównie jeżeli chodzi o tworzenie specjalistycznych pakietów.

Eksploracja danych jest tradycyjnie kojarzona ze statystyką, gdyż z niej się wywodzi, stąd nie dziwi wykorzystanie R w tym właśnie obszarze. W żadnym razie nie jesteśmy w stanie pokazać całego spektrum zagadnień związanych z R i jego wykorzystaniem w dziedzinie eksploracji danych i dlatego ograniczymy się do zademonstrowania kilku prostych przykładów.

2. Krótko na temat projektu R

Pod hasłem Projekt R (ang. R-project) [RPro] kryją się 2 elementy: specyfikacja języka programowania oraz środowisko uruchomieniowe. Projekt ten powstał na bazie komercyjnego środowiska i języka programowania S (aktualna wersja nosi nazwę S-PLUS) [SPLUS] i mówiąc najogólniej jest jego darmową wersją udostępnianą na licencji GNU GPL. Język R jest więc rodzajem dialektu języka S. Istnieją pewne drobne różnice między obu językami, ale są one minimalne i praktycznie programy napisane w jednym języku mogą być prawie bez żadnych zmian uruchamiane w drugim i odwrotnie. Wersje binarne R istnieją dla systemów Windows, UNIX/LINUX oraz Macintosh (S tylko dla tych dwóch pierwszych).

Język R jest językiem skryptowym interpretowanym a nie kompilowanym. Składnia i filozofia języka jest stosunkowo prosta i łatwa do opanowania. Poszczególne komendy wprowadza się z linii poleceń w instalowanej domyślnie prostej konsoli tekstowej[3]. Można też ciąg komend zapisać

[1] Nie mylić z systemem bazodanowym o nazwie R opracowanym przez IBM w latach 70-tych, który jako pierwszy posiadał zaimplementowaną obsługę języka SQL! System ten nie jest już od dawna rozwijany.

[2] Celowo używamy tu słowa użytkownik a nie np. programista, gdyż tworzenie funkcjonalnych programów (pakietów) w R nie jest zadaniem trudnym i nie jest zarezerwowane tylko dla doświadczonych programistów.

[3] Istnieje kilka graficznych edytorów, które wspierają środowisko R i dobrze się z nim integrują. Godnym polecenia jest Tinn-R. Ma on takie przydatne cech jak np. podświetlanie składni, integrację z przestrzenią roboczą R, wysyłanie zawartości edytowanego skryptu lub jego części do R. Osoby znające środowisko Eclipse oraz edytor Emacs być może będą zainteresowane wtyczkami StatET oraz ESS. Podkreślmy jednak, że do prostych, a nawet średnio zaawansowanych zastosowań, w zupełności powinien wystarczyć Tinn-R.

w postaci skryptu i uruchamiać go w dowolnym momencie. Skrypty są niezależne od systemu operacyjnego. Wadą języka R jest głównie długi czas wykonywania w porównaniu do analogicznych programów napisanych w językach kompilowanych. Jest to jednak cecha nie tylko języka R, ale generalnie wszystkich języków interpretowanych. Wraz ze wzrostem mocy obliczeniowej komputerów ta negatywna cecha zaczyna jednak tracić na znaczeniu.

O sile R świadczy to, że istnieje ogromna liczba (1936; stan na dzień 26.08.2009) gotowych do pobrania pakietów [RArch] realizujących różne wyspecjalizowane zadania, nie tylko ze statystyki, ale też z innych dziedzin. Lista pakietów jest na bieżąco uaktualniana. Doskonale rozwiązano kwestię instalowania pakietów. Są one umieszczone na wielu serwerach lustrzanych (ang. mirror) rozsianych po całym świecie. Po wybraniu interesującego nas pakietu następuje automatyczna jego instalacja w środowisku R wraz ze wszystkimi wymaganymi pakietami zależnymi (jeżeli takie są). Każdy może stworzyć swój własny pakiet i jeżeli przejdzie on pozytywnie weryfikację formalną przez zespół nadzorujący projekt R (fundacja *The R Foundation for Statistical Computing*), może zostać umieszczony na serwerze. Należy jednak zaznaczyć, że stworzenie w pełni poprawnego (od strony technicznej; merytoryczna zawartość pakietu to zupełnie inna sprawa) pakietu jest już zadaniem znacznie trudniejszym niż pisanie skryptów [RExt] i wymagana jest szczegółowa wiedza na temat języka R. Bardzo rozbudowane są możliwości tworzenia wysokiej jakości wykresów. W R można korzystać z bibliotek tworzonych w innych językach (np. C, C++) oraz z poziomu tych języków wywoływać funkcje stworzone w R. Nie ma też większych problemów z pobieraniem danych z baz danych (za pośrednictwem interfejsu ODBC oraz DBI). R ma bardzo dobrą i wyczerpującą dokumentację. Na stronie internetowej projektu można znaleźć sporo dokumentów, głównie w postaci plików PDF. Istnieje też bardzo bogata literatura na ten temat, na przykład [Cham00, VeRi02], również w języku polskim [Bie08, BieDM, WaGa09, KomR].

3. Eksploracja danych

3.1. Uwagi wstępne

Cytując za [Wazniak] eksploracja danych to „proces automatycznego odkrywania nietrywialnych, dotychczas nieznanych, potencjalnie użytecznych reguł, zależności, wzorców, schematów, podobieństw lub trendów w dużych repozytoriach danych". Nie będziemy zajmować się w tym miejscu omawianiem eksploracji danych jako takiej. Zainteresowanych odsyłamy do bardzo bogatej literatury przedmiotu. Pozycje [HaKa00, HMS05] można uznać za klasykę gatunku. Natomiast pozycje [Lar05, Lar06] są bardzo przystępnym wprowadzeniem w tematykę DM.

Korzenie eksploracji danych niewątpliwie wywodzą się ze statystyki [KoMi04, KoCw05], czyli nauki badającej prawidłowości występujące w zjawiskach o charakterze masowym (np. badanie preferencji wyborczych wszystkich dorosłych Polaków, przewidywanie ryzyka kredytowego osób z wyższym wykształcenie technicznym itd.). Na podstawie odpowiednio dobranej próby losowej wyciąga się różne wnioski (gdyż nie ma np. technicznie możliwości przeprowadzenia ankiety wśród wszystkich dorosłych Polaków). Wyniki uzyskane na podstawie próby losowej są następnie uogólniane na całą populację. Dochodzimy więc tutaj do podstawowego działu statystyki a mianowicie *wnioskowania statystycznego*. W ramach wnioskowania statystycznego staramy się stworzyć możliwie dokładny *model statystyczny* badanego zjawiska na podstawie posiadanej wspomnianej już próby losowej. Dokonujemy więc *estymacji parametrów* tego modelu. Mając model statystyczny można stawiać różne hipotezy i je weryfikować, czyli innymi słowy stosując odpowiednie testy statystyczne dokonujemy *weryfikacji hipotez statystycznych* (np. stawiamy hipotezę, że ludzie młodzi rzadziej głosują na partie konserwatywne). Przedstawiona powyżej w ogromnym skrócie idea tzw. klasycznego wnioskowania statystycznego zakłada, że wiemy mniej więcej czego szukamy w danych. Stawiając jakąś hipotezę opieramy ją na pewnej „wiedzy eksperckiej".

Eksploracja danych idzie niejako dalej niż klasyczne wnioskowanie statystyczne. Tu nie mamy zwykle możliwości sformułowania hipotezy badawczej, choćby ze względu na wielkość i złożoność posiadanych danych. Klasycznym przykładem jest tutaj zadanie grupowania, gdzie zwykle nie jesteśmy w stanie a priori sformułować hipotezy o ilości grup. Możemy tylko domyślać się, że dane da się jakoś sensownie pogrupować. Mając duże ilości danych bardzo trudno jest formułować wstępne oczekiwania o relacjach zachodzących między nimi. Trzeba je więc stopniowo i coraz dokładniej *zgłębiać*. Często w takich sytuacjach używa się terminu *eksploracyjna analiza danych* (ang. Exploratory Data Analysis, EDA). Do osiągnięcia zamierzonego celu (poznania natury danych i rządzących nimi prawidłowości) stosuje się różne techniki, modele i metody. Te wszystkie elementy określa się wspólnym mianem eksploracji danych. Oczywiście przedstawione wyżej klasyczne wnioskowanie statystyczne można również uznać za jedno z narzędzi eksploracyjnych.

3.2. Dane i metody w eksploracji danych

Różni autorzy zwykle różnie klasyfikują podstawowe metody i techniki związane z eksploracją danych. Wydaje się jednak, że najczęściej spotykany jest następujący podział:

- wstępna obróbka „surowych" danych,
- redukcja wymiarowości,
- budowanie modeli do celów szacowania (estymacji) i przewidywania (predykcji),
- klasyfikacja,
- grupowanie,
- odkrywanie reguł asocjacyjnych,
- odkrywanie wzorców sekwencji.

Jeżeli chodzi o rodzaj danych poddawanych eksploracji zwykle dzielimy je na:

- dane jakościowe (nominalne oraz porządkowe),
- dane ilościowe,
- dane tekstowe,
- mulitimedia,
- dane pochodzące z sieci WWW,
- dane o charakterze przestrzennym.

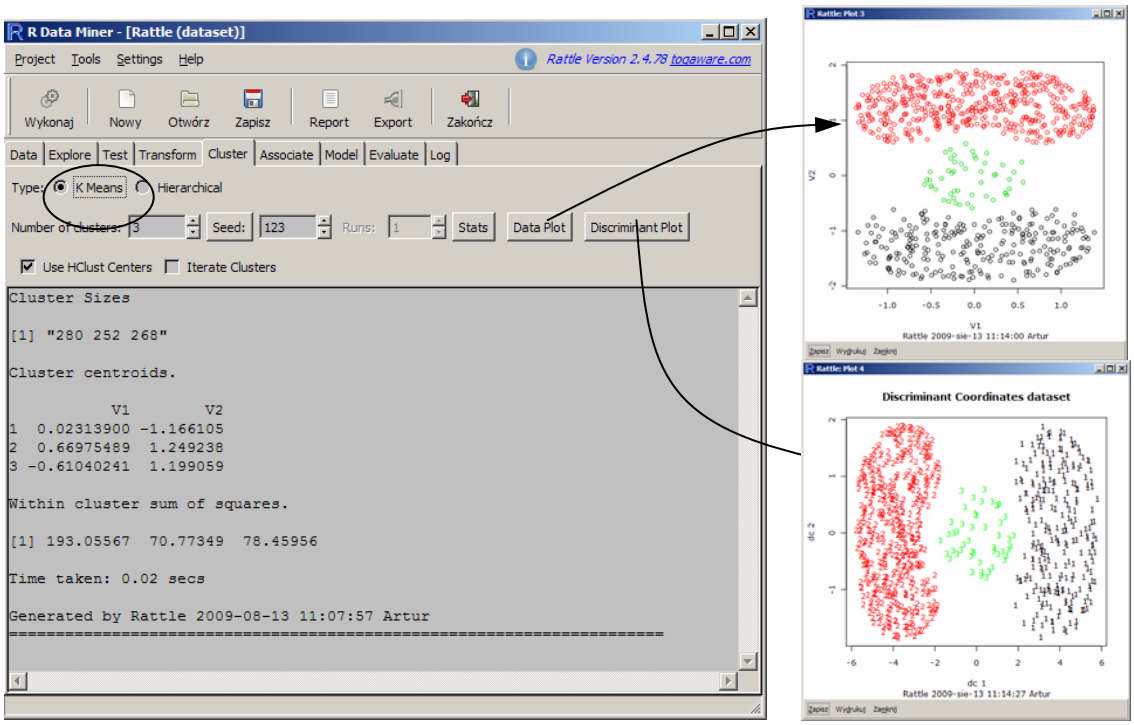
I wreszcie podział ze względu na charakter danych to:

- dane niemające porządku czasowego,
- dane o charakterze szeregów czasowych (ang. time series).

3.3. Pakiet `rattle`

W dalszej części artykułu wszystkie przykłady będą miały postać kodów zapisanych w języku skryptowym R i wykonywanych wprost z poziomu konsoli tekstowej. Mimo to wspominamy w tym miejscu o pewnej wygodnej w użyciu nakładce graficznej dedykowanej zagadnieniom eksploracji danych. Chodzi tutaj o pakiet `rattle` (ang. the **R** **A**nalytical **T**ool **T**o **L**earn **E**asily), który zbiera w jednym miejscu większość najważniejszych funkcji z obszaru eksploracji danych. W czasie instalowania pakietu nie są automatycznie instalowane wszystkie inne wymagane pakiety. W przypadku braku jakiegoś pakietu, w czasie pierwszego użycia funkcji, która go używa, zostaniemy poproszeni o jego zainstalowanie (wydając polecenie `install.packages „nazwa-`

`pakietu")`). W początkowym okresie używania pakietu może to być dość frustrujące, gdyż co chwila system prosi o doinstalowanie kolejnego brakującego pakietu. Zrzut ekranu pakietu `rattle` pokazano na rysunku 1. Na rysunku pokazano wykresy otrzymane w tym pakiecie dla tych samych danych, które są używane w podrozdziale 4.1.



Rys. 1. Pakiet rattle

4. Wybrane metody eksploracji danych w projekcie R

Poniżej zamieszczono przykłady użycia R w najczęściej wykonywanych zadaniach eksploracyjnych. Przykłady są bardzo proste i mają jedynie cel edukacyjny. Użyto tylko najprostszych metod, bez wgłębiania się w ich szczegóły merytoryczne. Jedynie zasygnalizowano fakt obsługiwania przez R bardzo wielu innych metod, czy też modyfikacji metod klasycznych. W szczególności nie są omawiane żadne metody redukcji wymiarowości, które są bardzo ważnym elementem DM. Zagadnienia te są tematem drugiego artykułu autorów, również prezentowanym na niniejszej konferencji [Gra09].

4.1. Analiza skupień (grupowanie)

Analiza skupień, często zwana również grupowaniem[4], (ang. clustering) to forma automatycznego łączenia (grupowania) obiektów, które są do siebie według określonych kryteriów podobne. Grupowanie różni się tym od klasyfikacji (patrz następny podrozdział), że nie ma z góry podanej zmiennej celu. O metodzie tej mówimy, że jest ona nienadzorowana. Algorytm grupowania stara się dokonać takiego podziału danych, aby elementy znajdujące się w jednej grupie były możliwie jak najbardziej do siebie podobne, a jednocześnie elementy należące do różnych grup możliwie jak najbardziej się od siebie różniły. Innymi słowy współczynnik zmienności pomiędzy grupami (ang. between cluster variation, BCV) powinien być duży, a współczynnik zmienności

[4] W języku polskim przyjęło się również używać zwrotu klastrowanie. Nie brzmi on jednak zbyt dobrze i w artykule konsekwentnie posługujemy się zwrotem grupowanie.

wewnątrz grupy (ang. within cluster variation, WCV) powinien być mały. Czyli stosunek BCV/WCV powinien w miarę działania algorytmu grupowania powiększać się[5].

Mnogość istniejących algorytmów jest taka, że rzadko kiedy można mówić o najlepszym rozwiązaniu. Poszczególne algorytmy, stosując różne kryteria tworzenia grup, często generują różniące się od siebie wyniki (chyba, że mamy do czynienia z danymi, gdzie podział na grupy jest bardzo oczywisty i jednoznaczny). Poszczególne algorytmy grupowania można podzielić na:

- hierarchiczne,
  - aglomeracyjne (ang. agglomerative; inna nazwa to skupiające lub łączące),
  - rozdzielające (ang. divisive),
- niehierarchiczne (optymalizacyjno-iteracyjne),
- inne, np. [HaKo00]:
  - metody oparte o analizę gęstości (ang. density-based methods),
  - metody oparte o strukturę gridową (ang. grid-based methods),
  - metody oparte o konstrukcję modelu (ang. model-based methods),
  - metody dedykowane dla danych o bardzo dużym wymiarze (ang. clustering high-dimensional data),
  - metody dla danych kategorycznych,
  - metody częściowo sterowane przez użytkownika (ang. constraint-based methods).

Wiele znanych algorytmów grupowania zostało już zaimplementowana w projekcie R. Dwa najpopularniejsze pakiety R, w których zaimplementowano różne algorytmy grupowania to `cluster` oraz `stats`. Jako pierwszy przykład rozważmy chyba najprostszą metodę grupowania o nazwie k-średnich. Użyjemy funkcji `kmeans(stats)`[6]. Użyjemy również pakietu `mlbench` do wygenerowania sztucznych danych. Ograniczymy się na razie tylko do danych 2-wymiarowych, które łatwo jest zwizualizować. Oczywiście wszystkie omawiane metody grupowania mogą operować na danych wielowymiarowych. Poniżej pokazano kody w języku R[7], które posłużyły do wykonania przykładowego grupowania.

```
# Załadowanie biblioteki do środowiska R.
library(mlbench)
# Zwróćmy uwagę, że utworzone zostaną grupy z nierówną ilością punktów.
dane <- mlbench.cassini(700, relsize = c(10,20,2))
plot(dane$x, pch = as.integer(dane$classes) - 1)
klaster <- kmeans(dane$x, centers = 3)
# Wyświetlenie środków wyznaczonych grup.
points(klaster$centers, cex = 2, pch = 19)
# Pokazanie przypisania poszczególnych punktów do grup.
plot(dane$x, pch = as.integer(klaster$cluster) + 5)
```

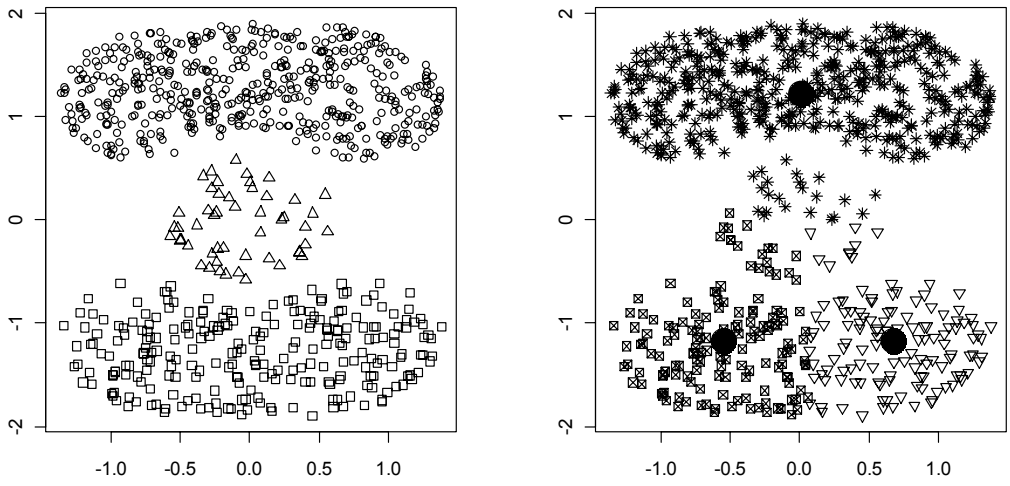
Wyniki grupowania pokazano na rysunku 2. Zauważmy również ciekawą rzecz. Wygenerowaliśmy sztuczne dane w taki sposób, że w jednej z grup jest zdecydowanie mniej danych niż

[5] Często stosowane algorytmy nie gwarantują znalezienia minimum globalnego, zatrzymując się na którymś minimum lokalnym. Tak np. dzieje się w algorytmie k-średnich.

[6] Podając nazwy funkcji w R dobrym zwyczajem jest w nawiasie podawać również nazwę pakietu, gdzie dana funkcja się znajduje. Warto tak robić, gdyż unikamy przez to niejednoznaczności (funkcje o tych samych nazwach mogą być w różnych pakietach). Ponadto w stronie internetowej projektu R nie można wyszukiwać funkcji po ich nazwach.

[7] Czcionką pogrubioną podajemy polecenia wpisywane przez użytkownika w konsoli R. Czcionką pochyłą wpisujemy komentarze a czcionką zwykłą wyniki otrzymywane na konsoli.

w dwóch pozostałych. Ujawniła się tutaj pewna wada algorytmu k-średnich polegająca na tym, że ignorowane są małe (a jednak oczywiste) grupy danych. Na rysunku 2b widać, że punkty z tej małej grupy zostały „zawładnięte" przez inne grupy. Gdyby wykonać podobny eksperyment ale z parametrem np. `relsize=c(10,20,20))` w funkcji `mlbench.cassini()`, wynik grupowania będzie zgodny z oczekiwaniami. W zasadzie identyczny wynik uzyskamy, stosując metodę PAM (ang. partitioning around medoids), która jest zmodyfikowaną wersją algorytmu k-średnich. Modyfikacja polega na tym, że środki grup mogą być brane tylko ze zbioru danych, a nie przyjmowane dowolnie, jak to się dzieje w metodzie k-średnich. Dzięki temu metoda ta staje się bardziej odporna na różnego rodzaju problemy (ang. robust). Akurat dla naszego zbioru danych nie zauważono polepszenia wyników. Metoda PAM jest zrealizowana w R w funkcji `pam(cluster)`. Inne metody grupowania z tej rodziny to np. `clara(cluster)` oraz `fanny(cluster)`. Druga z nich realizuje grupowanie danych z wykorzystaniem logiki rozmytej.



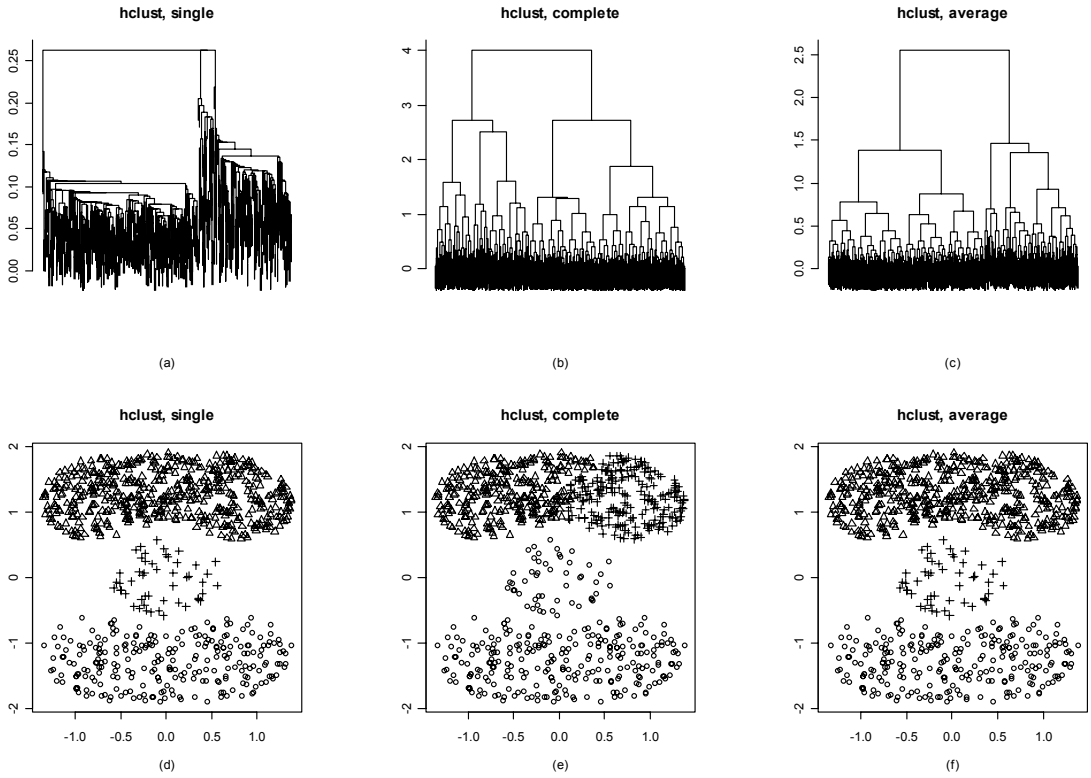
Rys. 2. Wyniki grupowania przykładowego zbioru danych 2D z użyciem funkcji `kmeans(stats)`, (a) – zbiór przykładowych danych, (b) – dokładny wynik grupowania wraz z zaznaczonymi środkami grup

Kolejny przykład dotyczy grupowania hierarchicznego. Użyjemy funkcji `hclust(stats)`. Inne funkcje z tej rodziny to np. `agnes(cluster)`, `diana(cluster)`, `mona(cluster)`. Cechą wspólną metod hierarchicznych jest to, że nie zakłada się wstępnie ilu grup będziemy poszukiwać. Budowane jest natomiast drzewo (dendrogram) i następnie wykonywana jest tzw. procedura przycinania drzewa (ang. prunning) na pewnej wysokości celem przypisania poszczególnych obserwacji do poszczególnych grup. Należy zwrócić tutaj uwagę na to, że uzyskiwane wyniki bardzo różnią się od siebie w zależności od metody służącej do określania odległości między grupami (parametr *method* w funkcji `hclust(stats)`) oraz metody służącej do wyliczania odległości między poszczególnymi obserwacjami (parametr *method* w funkcji `dist(stats)`). Ilość różnych możliwych do otrzymania wariantów rozwiązań jest więc bardzo duża.

Wyniki grupowania pokazano na rysunku 3. Widać, że jedynie metoda *complete* określania odległości między grupami źle pogrupowała dane. Zwykle nie należy się więc sugerować podobieństwem dendrogramów – zwłaszcza, gdy danych jest dużo i dendrogram jest nieczytelny w dolnej części.

```
# Inne metody określania odległości między grupami w funkcji hclust() to:
# ward", "complete", "average", "mcquitty", "median" or "centroid".
# Inne metody określania odległości między obserwacjami w funkcji dist()
to:
# "maximum", "manhattan", "canberra", "binary" or "minkowski".
klaster2 <- hclust(dist(dane$x, method = "euclidean"), method = "single")
# Wyświetlenie dendrogramu.
```

```
plot(klaster2,  main = "hclust, single", labels = FALSE, sub = "", xlab = "")
# Obcięcie drzewa, chcemy otrzymać 3 grupy.
kl2 <- cutree(klaster2, k = 3)
# Pokazanie przypisania poszczególnych punktów do grup.
plot(dane$x, pch = as.integer(kl2) + 5)
```



Rys. 3. Wyniki grupowania hierarchicznego z użyciem funkcji `hclust(stats)`, (a)-(c) – dendrogramy dla metod single, complete oraz avarage, (d)-(f) – odpowiadające dendrogramom wyniki grupowania

Po wykonaniu grupowania kolejnym krokiem jest wyznaczenie jakiejś miary dokładności grupowania (aby np. porównać wyniki otrzymywane różnymi metodami). Tematu tego nie będziemy w tym miejscu rozwijać, odsyłając czytelnika do literatury [HBV01]. Można użyć np. miary Silhouette (`silhouette(stats)`) lub użyć jakiejś innej metody walidacji grup. Wiele takich metod jest zaimplementowanych w pakiecie `clv`. Tam też można np. wyliczyć wspomniane już wyżej współczynniki BCV oraz WCV (funkcje `wcls.matrix(clv)` oraz `bcls.matrix(clv)`).

4.2. Analiza dyskryminacyjna (klasyfikacja)

Analiza dyskryminacyjna, często zwana również klasyfikacją, to metoda na przypisywanie nowych obiektów do znanych, wcześniej określonych klas. Klasyfikacja różni się tym od grupowania, że istnieje pewna zdefiniowana zmienna celu. O metodzie tej mówimy więc, że jest ona nadzorowana. W metodzie tej zawsze mamy do wykonania 3 podstawowe kroki. W pierwszym, posługując się *zbiorem uczącym*, gdzie mamy podane zarówno zmienne opisujące oraz zmienną celu, staramy się nauczyć nasz klasyfikator danych. W drugim kroku używany stworzony klasyfikator do zaklasyfikowania zbioru *danych testowych* (który często jest fragmentem zbioru uczącego) do określonej grupy. W danych testowych zmienna celu jest obecna, ale klasyfikator nie ma do niej dostępu. Następnie sprawdzana jest efektywność klasyfikacji poprzez porównanie z właściwymi wartościami zmiennej celu. Gdy uzyskiwane wyniki nie są satysfakcjonujące, należy próbować poprawić klasyfikator. W ostatnim trzecim kroku klasyfikator stosuje się do zbioru nowych danych (innych niż zbiór uczący) celem wykonania właściwej klasyfikacji, zwanej teraz *predykcją*.

Istnieje sporo różnych algorytmów klasyfikacji. Poszczególne algorytmy klayfikacji można podzielić na:

- klasyfikacja za pomocą indukcji drzew decyzyjnych (ang. decison tree induction),
- klasyfikacja Bayesowska (ang. Bayesian classification),
- klasyfikacja metodą k najbliższych sąsiadów (ang. k-nearest neighbour). Metoda ta należy do grupy metod zwanych uczeniem leniwym (ang. instancje-based learning),
- klasyfikacja za pomocą reguł decyzyjnych (ang. rule-based classification),
- liniowa i kwadratowa klasyfikacja dyskryminacyjna (ang. discrimination classification),
- inne z wykorzystaniem:
  - sieci neuronowych (ang. neural networks),
  - algorytmów genetycznych (ang. genetic algorithms),
  - zbiorów rozmytych (ang. fuzzy sets),
  - zbiorów przybliżonych (ang. rough sets),
  - wektorów wspierających (ang. support vector machines, SVM).

Wiele znanych algorytmów klasyfikacyjnych zostało już zaimplementowana w R. Cztery bardzo popularne pakiety R, w których zaimplementowano różne algorytmy grupowania to `tree`, `rpart`, `party` oraz `class`. Już jednak tylko pobieżny ogląd listy dostępnych pakietów w R [RArch] pozwala stwierdzić, że zadanie klasyfikacji jest tam bardzo silnie reprezentowane i można znaleźć jeszcze wiele innych pakietów dotyczących klasyfikacji. Szczegółów nie będziemy w tym miejscu omawiać.

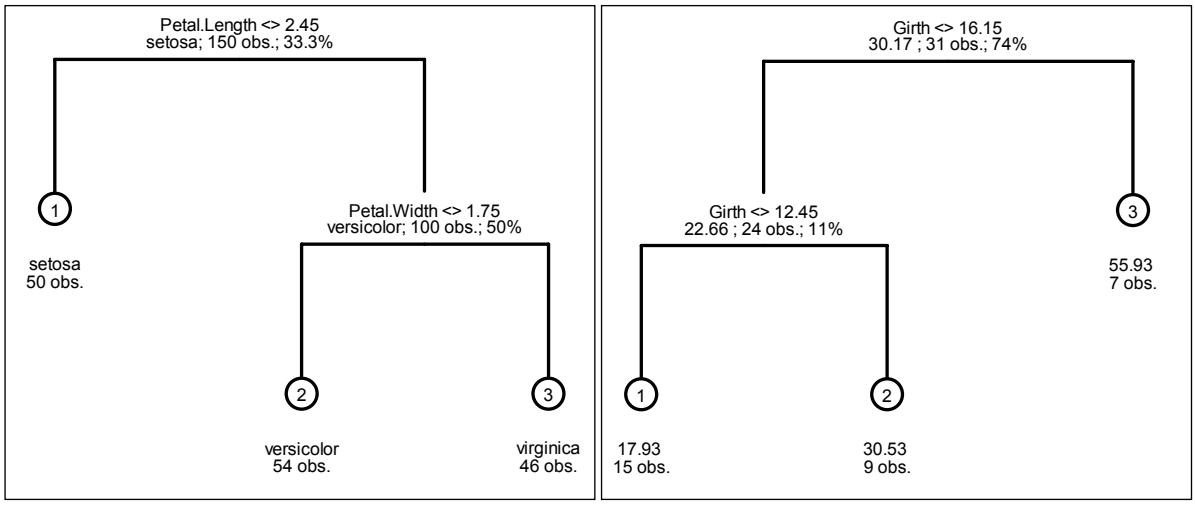
Zagadnienie klasyfikacji jest dużo bardziej złożone niż omówione w poprzednim rozdziale grupowanie. Hasłowo wspomnijmy tylko takie problemy jak: określanie dokładności klasyfikatora, klasyfikacja danych z atrybutami ciągłymi, problem nadmiernego rozgałęziania się drzew (ang. overfitting), przycinanie drzew oraz wiele innych.

Poniżej pokazano użycie R do wykonania klasyfikacji metodą indukcji drzew decyzyjnych. Pozostałe metody, z braku miejsca, nie będą omawiane. Wykorzystamy tylko jedną z najpopularniejszych metod zwaną w skrócie CART (ang. clasiffication and regression trees). Jak wynika z nazwy mamy do czynienia z dwoma rodzajami drzew: klasyfikacyjnymi oraz regresyjnymi. O drzewach klasyfikacyjnych mówimy wówczas, gdy zmienna celu jest jakościowa (nominalna lub porządkowa), natomiast o drzewach regresyjnych, gdy zmienna celu jest ilościowa (a przynajmniej przedziałowa). Nie będziemy w tym miejscu omawiać szczegółów algorytmu CART, odsyłając zainteresowane osoby do bardzo bogatej literatury, np. [HaKa00, HMS05, Lar05].

Poniżej pokazano kody w języku R, które posłużyły do zbudowania dwóch drzew: regresyjnego oraz klasyfikacyjnego. Posłużono się prostymi i bardzo popularnymi zbiorami danych *iris* oraz *trees* [MLRep]. W zbiorze *iris* podane są długości i szerokości działek kielicha i płatków trzech odmian irysa (Setosa, Versicol i Virginic). Celem analizy jest znalezienie sposobu przypisania kwiatu do jednej z trzech odmian, na podstawie zmierzonych czterech, różnych jego rozmiarów. Zbiór *trees* zawiera dane o obwodzie, wysokości i objętości 31 ściętych drzew wiśni. Celem analizy jest znalezienie sposobu przypisania objętości na podstawie obwodu i wysokości drzewa. Oba zbiory danych dostępne są w pakiecie `datasets`. Na rysunku 4 pokazano wynikowe drzewa.

```
library(maptree) # jest tam funkcja draw.tree
data(iris)
# Funkcja rpart ma bardzo dużo parametrów pozwalających dokładnie
```

```
# sterować przebiegiem tworzenia drzewa. W przykładzie nie użyto jawnie
żadnych
# parametrów, stąd przyjmują one wartości domyślne – patrz dokumentacja.
# Drzewo klasyfikacyjne, parametr method = "class".
iris.rp <- rpart(Species ~ Sepal.Length + Sepal.Width + Petal.Length +
                 Petal.Width, data = iris, method = "class")
# Wyświetlenie drzewa.
plot(iris.rp)
text(iris.rp, use.n = TRUE, all = TRUE, cex = 0.5, font = 2)
# Wynik do pliku PosctScript.
post(iris.rp, file = "iris.rp.ps", title = "Drzewo klasyfikacyjne, zbiór
iris")

data(trees)
# Drzewo regresyjne, parametr method = "anova".
trees.rp <- rpart(Volume ~ Girth + Height, data = trees, method = "anova")
 # draw.tree daje nieco ładniejsze rysunki niż funkcje plot() + text()
draw.tree(iris.rp)
post(trees.rp, file = "trees.rp.ps", title = "Drzewo regresyjne, zbiór trees")
```



Rys. 4. Wyniki klasyfikacji przykładowych zbiorów danych, (a) – zbiór danych *iris*, (b) – zbiór danych *trees*

Powyższy przykład jest bardzo prosty i operuje na niewielkich zbiorach danych. Kolejny przykład oparty jest o dużo liczniejszy i bardziej skomplikowany zbiór danych *adult* [MLRep]. W R można go znaleźć np. w pakiecie `arules`. Zbiór ten zawiera 48842 rekordów z danymi spisu ludności w USA. Zmienną celu jest dochód (zmienna *income*; >50000$ oraz <=50000$), liczba zmiennych wejściowych wynosi 14. Jesteśmy zainteresowani klasyfikacją, czy dochód osoby jest w pierwszej, czy też drugiej grupie.

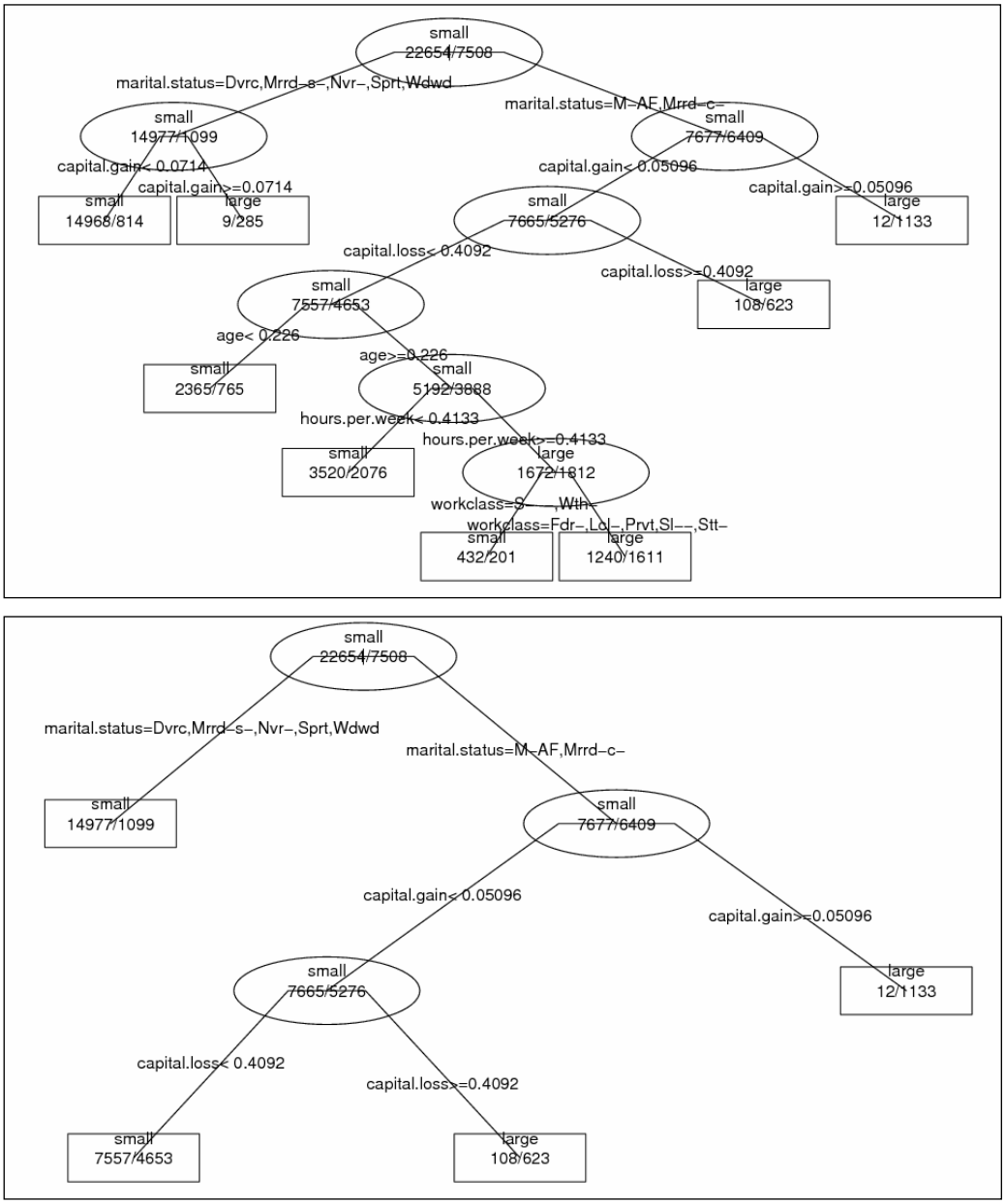
Zbiór danych zawiera zarówno zmienne ilościowe, jak i jakościowe. W wielu miejscach brakuje danych. Gdzie oraz ile danych brakuje można odczytać poleceniem `summary(AdultUCI)`. Do zbudowania drzewa posłużymy się następującymi zmiennymi [Lar05]: *age* (wiek), *education-num* (liczba lat poświęconych nauce), *capital-gain* (przyrost kapitału), *houses-per-week* (liczba godzin pracy w tygodniu), *race* (rasa), *sex* (płeć), *workclass* (rodzaj zatrudnienia), *marital-status* (stan cywilny). Dodatkowo zmienne ilościowe zostaną znormalizowane od przedziału *0-1* za pomocą funkcji `rescaler(reshape)`. Rekordy, gdzie brakuje danych zostaną usunięte za pomocą funkcji `na.omit(stats)`.

```
library(arules)
library(rpart)
```

```
library(reshape) # Jest tam funkcja rescaler.
data(AdultUCI)   # Ładujemy zbiór adult. Jest on w pakiecie arules.
# Podsumowanie danych.
summary(AdultUCI)

# Podsumownie zmiennej celu income.
# Ilość rekordów bez podanej wartości zmiennej celu: 16281.
# Możemy potraktować je jako dane testowe i usunąć na etapie budowania drzewa.
# Usuniemy też inne rekordy, gdzie brakuje danych.
summary(AdultUCI$income)

# Usuwamy rekordy z brakującymi danymi.
# Otrzymujemy ostatecznie 30162 rekorów, gdzie nie brakuje żadnych danych.
adult.omit <- na.omit(AdultUCI)
# Wyświetlamy rozmiar nowej ramki danych.
dim(adult.omit)
# Normalizacja zmiennych ilościowych do przedziału 0-1.
adult.01 <- rescaler(adult.omit, type = "range")
# Funkcja rpart(rpart) ma pewien błąd. Gdy w nazwach zmiennych występują
# znaki minus (-) to są one traktowane jak operatory odejmowania.
# Zmieniamy więc te znaki na kropki.
colnames(adult.01) <-
c("age", "workclass", "fnlwgt", "education", "education.num",
"marital.status",
  "occupation", "relationship", "race", "sex", "capital.gain", "capital.loss",
  "hours.per.week", "native.country", "income")
# Wyświetlamy nazwy zmiennych po zmianie.
dimnames(adult.01)

# Tworzenie drzewa.
adult.rp = rpart(income ~ age + capital.gain + capital.loss +
                 hours.per.week + race + sex + workclass + marital.status,
                 data = adult.01, method = "class")
# Wyświetlanie drzewa oraz utworzenie pliku PostScript.
plot(adult.rp)
text(adult.rp, use.n = T, all = T, cex = 0.8, digits = 6)
post(adult.rp, file = "a1.ps", use.n = TRUE, digits = 6, title = "")
# Struktura drzewa w postaci tekstowej.
print(adult.rp, digits = 2)
# Przycinanie drzewa.
adult.prune <- prune(adult.rp, cp = 0.05) # Parametr cp z adult.rp$cptable.
plot(adult.prune)
text(adult.prune, use.n = FALSE, all = FALSE, cex = 0.8, font = 2)
post(adult.prune, file = "a2.ps", use.n = TRUE, digits = 6, title = "")
print(adult.prune, digits = 2)
```

Na rysunku 5 pokazano oba drzewa w postaci graficznej. Oba rysunki powstały w wyniku działania funkcji `post(rpart)`. Dalej pokazano też 2 drzewa wynikowe w postaci tekstowej. Pierwsze przed obcięciem, oraz drugie po wykonaniu operacji przycinania. Oba wydruki powstały w wyniku użycia funkcji `print(rpart)`[8]. Są one w pełni zgodne z rysunkiem 5.

[8] W języku, R funkcje można przeciążać. Oznacza to, że funkcja o tej samej nazwie wykonuje za każdym razem nieco inne działanie, w zależności od rodzaju przekazanego jako parametr wejściowy argumentu. Tu mamy przykład funkcji `print`, która jeżeli jako parametr wejściowy otrzyma obiekt zwracany przez funkcję, `rpart` drukuje drzewo klasyfikacyjne.



Rys. 5. Drzewo CART dla zbioru danych *adult*, (a) – drzewo przed przycięciem, (b) – drzewo po przycięciu

```
  1) root 30162 7500 small (0.751 0.249)
    2) marital.status=Divorced,Married-spouse-absent,Never-married,Separated,Widowed
       16076 1100 small (0.932 0.068)
      4) capital.gain< 0.071 15782  810 small (0.948 0.052) *
      5) capital.gain>=0.071 294    9 large (0.031 0.969) *
    3) marital.status=Married-AF-spouse,Married-civ-spouse 14086 6400 small (0.545
0.455)
      6) capital.gain< 0.051 12941 5300 small (0.592 0.408)
       12) capital.loss< 0.41 12210 4700 small (0.619 0.381)
         24) age< 0.23 3130  760 small (0.756 0.244) *
         25) age>=0.23 9080 3900 small (0.572 0.428)
           50) hours.per.week< 0.41 5596 2100 small (0.629 0.371) *
           51) hours.per.week>=0.41 3484 1700 large (0.480 0.520)
```

```
            102) workclass=Self-emp-not-inc,Without-pay 633  200 small (0.682 0.318) *
            103) workclass=Federal-gov,Local-gov,Private,Self-emp-inc,State-gov 2851
1200
                 large (0.435 0.565) *
       13) capital.loss>=0.41 731  110 large (0.148 0.852) *
      7) capital.gain>=0.051 1145   12 large (0.010 0.990) *
```

```
1) root 30162 7500 small (0.751 0.249)
   2) marital.status=Divorced,Married-spouse-absent,Never-married,Separated,Widowed
      16076 1100 small (0.932 0.068) *
   3) marital.status=Married-AF-spouse,Married-civ-spouse 14086 6400 small (0.545
0.455)
     6) capital.gain< 0.051 12941 5300 small (0.592 0.408)
      12) capital.loss< 0.41 12210 4700 small (0.619 0.381) *
      13) capital.loss>=0.41 731  110 large (0.148 0.852) *
     7) capital.gain>=0.051 1145   12 large (0.010 0.990) *
```

Należy na koniec tego podrozdziału wspomnieć, że jak na razie nie zostały zaimplementowane w R dwa bardzo popularnych algorytmów tworzenia drzew decyzyjnych, a mianowicie C4.5 oraz C5.0. Obejściem tego problemu jest użycie pakietu `RWeka`, który jest interfejsem do innego bardzo popularnego programu do eksploracji danych o nazwie Weka [Weka]. Program ten ma zimplementowany m.in. algorytm C4.5.

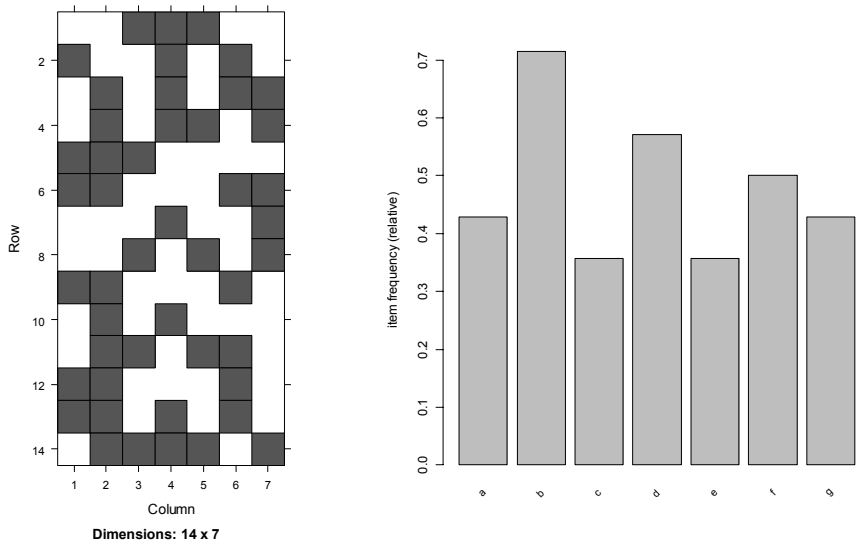
4.3. Odkrywanie reguł asocjacyjnych i zbiorów częstych

Odkrywanie reguł asocjacyjnych oraz zbiorów częstych [HaKa00, Lar05] to popularna i dobrze opracowana metodologia odkrywania interesujących relacji pomiędzy danymi występującymi w dużych zbiorach danych. W R istnieje pakiet `arules`, który zwiera podstawowe narzędzia do manipulowania danymi wejściowymi oraz do wyznaczania zbiorów częstych oraz reguł asocjacyjnych. W pakiecie utworzono interfejsy dostępowe do zaimplementowanych w języku C dwóch popularnych algorytmów: *APriori* oraz *Eclat* [Bor].

Jako przykład użycia pakietu `arules` rozważmy bardzo prosty przykładowy zbiór transakcji stworzony na bazie listy (lista to jeden z typów zmiennych w języku R).

```
a_list <- list(
      c("c", "d", "e"),
      c("a", "d", "f"),
      c("b", "d", "f", "g"),
      c("b", "d", "e", "g"),
      c("a", "b", "c"),
      c("a", "b", "f", "g"),
      c("d", "g"),
      c("c", "e", "g"),
      c("a", "b", "f"),
      c("b", "d"),
      c("b", "c", "e", "f"),
      c("a", "b", "f"),
      c("a", "b", "d", "f"),
      c("b", "c", "d", "e", "g")
      )
# Tworzymy nazwy transakcji.
names(a_list) <- paste("Transaction",c(1:14), sep = "")
# Przypisujemy obiekt to klasy 'transaksja' (w rozumieniu pakietu arules).
trans <- as(a_list, "transactions")
```

```
# Tworzymy wizualną "mapę" transakcji (patrz rysunek 5a).
image(trans)
# Rysujemy wykres częstotliwościowy (patrz rysunke 5b).
itemFrequencyPlot(trans[,itemFrequency(trans) > 0.1], cex.names = 0.8)
```



Rys. 5. (a) – wizualizacja przykładowej transakcji, (b) – wykres częstotliwościowy

Po utworzeniu obiektu typu transakcja możemy wyznaczyć reguły asocjacyjne i je wyświetlić. W przykładzie poniżej wyznaczono reguły dla minimalnego wsparcia (ang. support) 0,2 oraz minimalnego współczynnika ufności (ang. confidence) 0,8.

```
rules <- apriori(trans, parameter = list(supp = 0.2, conf = 0.8))
inspect(rules)
  lhs    rhs   support confidence     lift
1 {c} => {e} 0.2857143  0.8000000 2.240000
2 {e} => {c} 0.2857143  0.8000000 2.240000
3 {a} => {f} 0.3571429  0.8333333 1.666667
4 {a} => {b} 0.3571429  0.8333333 1.166667
5 {f} => {b} 0.4285714  0.8571429 1.200000
6 {a,
   f} => {b} 0.2857143  0.8000000 1.120000
7 {a,
   b} => {f} 0.2857143  0.8000000 1.600000
```

Zbiory częste wyznaczamy za pomocą funkcji `eclat()` jak poniżej.

```
itemsets <- eclat(trans, parameter = list(supp = 0.4, maxlen = 3))
inspect(itemsets)
  items   support
1 {b,
   f}   0.4285714
2 {b}   0.7142857
3 {d}   0.5714286
4 {f}   0.5000000
5 {g}   0.4285714
6 {a}   0.4285714
```

4.4. Eksploracja tekstu

Eksploracja tekstu to bardzo ważny obszar badawczy w eksploracji danych. Dzieje się tak dlatego, że dokumenty tekstowe (albo takie, które da się w miarę łatwo do takiej postaci przekonwertować) są ciągle najpopularniejszą formą gromadzenia i wymiany danych. Do dokumentów tekstowych z powodzeniem stosuje się takie techniki eksploracji jak omówione już powyżej klasyfikacja oraz grupowanie. Bardzo ważne inne zadania eksploracji tekstu to np. wyszukiwanie dokumentów podobnych (np. do celów grupowania lub wykrywania potencjalnych plagiatów), tworzenie rankingów, analiza zależności (indeks cytowań), ekstrakcja cech (np. tworzenie automatycznych podsumowań, rozpoznawanie języka dokumentów, rozpoznawanie tematyki dokumentów) i inne. W odniesieni do dokumentów tekstowych używa się często pojęcia *wyszukiwanie informacji* (ang. Information Retrieval; IR). W R wiele z omawianych powyżej zadań eksploracji tekstu zrealizowane jest w pakiecie `tm`.

Podobnie jak przy omawianiu innych technik eksploracji danych, tak i tym razem jedynie zasygnalizujemy temat, podając kilka przykładów działania pakietu `tm`. Pakiet `tm` pracuje na dokumentach tekstowych. Podstawową strukturą jest tzw. korpus (ang. corpus), który jest obiektem reprezentującym pewną kolekcję dokumentów tekstowych wraz z zapisanymi ew. dodatkowymi informacjami zwanymi metadanymi. Aby utworzyć nowy korpus musimy wyspecyfikować rodzaj źródła z danymi tekstowymi. Aby zorientować się, jakie źródła danych są obsługiwane, należy wykonać funkcję `getSources()`.

```
getSources()
"DataframeSource" "DirSource"       "GmaneSource"     "ReutersSource"
"URISource"       "VectorSource"
```

Przykładowo źródło o nazwie `DirSource` służy do obsługi dokumentów tekstowych zlokalizowanych w katalogach a `DataframeSource` obsługuje źródła zlokalizowane w ramkach danych. Można również wyspecyfikować pewne dodatkowe parametry, jak np. język dokumentów. Poniżej pokazano jak można utworzyć przykładowe korpusy. Przykłady oparte są na opracowaniu [Fei09], którego autorem jest twórca pakietu `tm`.

Tworzymy korpus oparty o 6 przykładowych plików pewnej grupy dyskusyjnej (pliki te są włączone do pakietu `tm`).

```
(newsgroup <- system.file("texts", "newsgroup", package = "tm"))
"C:/Programy/R-2.8.1/library/tm/texts/newsgroup"
(ng = Corpus(DirSource(newsgroup), readerControl = list(reader =
readNewsgroup,
            language = "eng")))
A corpus with 6 text documents
```

Tworzymy korpus oparty o wektory tekstowe:

```
docs <- c(
   "Podoba mi się ten głaz. To naprawdę ładny głaz!",
   "I wtedy zapada taka niezręczna cisza...",
   "Ale będzie ubaw! Męskie rozmowy, takie o życiu i o śmierci.")
d <- Corpus(VectorSource(docs))
inspect(d)
A corpus with 3 text documents
[[1]]
[1] Podoba mi się ten głaz. To naprawdę ładny głaz!
[[2]]
[1] I wtedy zapada taka niezręczna cisza...
[[3]]
[1] Ale będzie ubaw! Męskie rozmowy, takie o życiu i o śmierci.
```

Tworzymy korpus oparty o 10 przykładowych plików ze zbioru Reuters-21578. Jest to bardzo popularny zbiór danych testowym [MLRep, Fei08, Fei09]. Mały fragment tego zbioru jest włączony do pakietu `tm`.

```
(reut21578 <- system.file("texts", "reut21578", package = "tm"))
"C:/Programy/R-2.8.1/library/tm/texts/reut21578"
(reuters <- Corpus(DirSource(reut21578),
                   readerControl = list(reader = readReut21578XML)))
A corpus with 10 text documents
```

Bardzo ważnym etapem w eksploracji tekstów jest ich wstępna obróbka i przygotowanie do wykonywania właściwych analiz. Surowe teksty są bowiem prawie zawsze bardzo mocno „zanieczyszczone" różnymi elementami utrudniającymi lub czasem wręcz uniemożliwiającymi ich analizę. Chodzi tutaj np. o takie rzeczy jak istnienie słów mało znaczących (tzw. stoplista-a), istnienie elementów formatujących (np. znaczniki w dokumentach XML-owych), odmiana słów itd. Poniżej pokazano przykłady wykonania różnych transformacji na tekstach bazowych. Listę dostępnych transformacji można uzyskać wykonując funkcję `getTransformations()`.

```
getTransformations()
"asPlain"           "loadDoc"           "removeCitation"
"removeMultipart"   "removeNumbers"     "removePunctuation"
"removeSignature"   "removeWords"       "replacePatterns"
"stemDoc"           "stripWhitespace"   "tmTolower"
```

Poniżej pokazano przykłady wykonania różnych transformacji wraz z wynikami (dla zaoszczędzenia miejsca teksty wynikowe na wydruku skrócono). Wydaje się, że kluczową transformacją jest wydobycie z poszczególnych słów ich rdzenia znaczeniowego (ang. stem, stemming). Nie jest to zadanie łatwe! Dla języka angielskiego istnieje popularny i często stosowany algorytm Portera [Por]. Próbę opracowania odpowiedniego narzędzia dla języka polskiego podjął swego czasu pan Dawid Weiss [Lam] (rozwój projektu został chyba ostatnio przyhamowany).

```
# Konwersja dokumentów do potaci czystego tekstu.
reuters <- tmMap(reuters, asPlain)
inspect(reuters[1])
[1] Showers    continued throughout the      week        in the Bahia cocoa
zone,    alleviating the   drought since   early January and improving
prospects for the coming temporao,although normal humidity levels have not
been restored, Comissaria Smith said in its       weekly review.

# Usunięcie nadmiarowych białych znaków.
reuters <- tmMap(reuters, stripWhitespace)
inspect(reuters[1])
Showers continued throughout the week in the Bahia cocoa zone, alleviating the
drought since early January and improving prospects for the coming temporao,
although normal humidity levels have not been restored, Comissaria Smith said
in its weekly review.

# Zamiana dużych liter na małe.
reuters <- tmMap(reuters, tmTolower)
inspect(reuters[1])
showers continued throughout the week in the bahia cocoa zone, alleviating the
drought since early january and improving prospects for the coming temporao,
although normal humidity levels have not been restored, comissaria smith said
in its weekly review.

# Usunięcie fraz z tzw. stoplisty.
reuters <- tmMap(reuters, removeWords, stopwords("english"))
```

```
inspect(reuters[1])
showers continued throughout  week  bahia cocoa zone, alleviating  drought
january  improving prospects  coming temporao,  normal humidity levels
restored, comissaria smith   weekly review.

# Zamiana słów na formę podstawową (wydobycie rdzenia znaczeniowego).
reuters <- tmMap(reuters, stemDoc)
inspect(reuters[1])
shower continu throughout  week  bahia cocoa zone, allevi  drought  januari
improv prospect  come temporao,  normal humid level   restored, comissaria
smith   week review.
```

Po wstępnym przetworzeniu tekstów zwykle tworzona test tzw. macierz słowo-dokument (ang. term-document matrix, TDM), na której wykonuje się już właściwe analizy. Przykład ciekawej analizy treści książek z serii *Czarnoksiężnik z krainy Oz*, z wykorzystaniem pakietu `tm` zamieszczono w [Fei08]. Wiersze w macierzy TDM reprezentują poszczególne dokumenty a kolumny wszystkie występujące w dokumentach słowa (często używa się też określenia term). Jest to macierz zwykle bardzo rzadka (niemal zawsze nie wszystkie słowa występują we wszystkich dokumentach – zwłaszcza, gdy mamy dużo dokumentów z wielu różnych dziedzin).

Macierz TDM tworzymy na bazie dokumentów przetworzonych wszystkimi powyższymi transformacjami (dokumenty zawierają w sumie 491 różnych termów. Czytelnik może samodzielnie sprawdzić, że po zamianie tylko na postać tekstową poleceniem `tmMap(reuters, asPlain)` i bez wykonywania pozostałych transformacji jest ich 582, czyli prawie o 100 więcej. Przy większej liczbie dokumentów zysk zwykle będzie jeszcze większy).

```
dtm <- DocumentTermMatrix(reuters)
dim(dtm)
10 491

show(dtm)
A document-term matrix (10 documents, 491 terms)

Non-/sparse entries: 605/4305
Sparsity           : 88%
Maximal term length: 14
Weighting          : term frequency (tf)

inspect(dtm[1:5, 150:155])
A document-term matrix (5 documents, 6 terms)

Non-/sparse entries: 4/26
Sparsity           : 87%
Maximal term length: 12
Weighting          : term frequency (tf)

  exclusive exercisable expect expected expects experiencing
1         0           0      0        1       0            1
2         0           0      0        0       0            0
3         0           0      0        0       0            0
4         0           0      1        1       0            0
5         0           0      0        0       0            0
```

Przykładową operacją, którą wykonamy na macierz TDM będzie wyświetlenie tych termów, które pojawiają się w tekstach co najmniej 5 razy.

```
findFreqTerms(dtm, 5)
"bags"        "cocoa"       "comissaria"  "crop"        "dec"
"dlrs"        "july"        "mln"         "sales"       "sept"
"smith"       "times"       "york"        "analysts"    "bankamerica"
"debt"        "stock"       "level"       "price"       "apr"
"feb"         "mar"         "nil"         "prev"        "total"
"computer"    "terminal"
```

5. Import i export danych w R

Podstawowe funkcje do importu oraz eksportu danych to `read.table(utils)` oraz `write.table(utils)` oraz ich warianty (np. `read.csv(utils)`, patrz [RImpExp]). Używając pakietu `XML` możliwa jest praca z niezwykle obecnie popularnym formatem XML-owym. R potrafi również odczytywać dane z plików (też binarnych) wykorzystywanych przez inne popularne pakiety statystyczne, takie jak S-PLUS, Weka, SAS, SPSS, Stata, Minitab, Matlab (Statistical Toolbox), Systat.

Warto wspomnieć, że używając funkcji `read.table` można odczytywać również dane znajdujące się w schowku podręcznym (ang. clipboard) oraz zlokalizowane w Internecie. Wówczas jako nazwę pliku podajemy albo słowo kluczowe *clipboard*, albo też kompletny adres URL do pliku. Czytając dane ze schowka systemowego pamiętajmy, że danych tych „nie widać" i może zdarzyć się, że R źle odczyta dane, gdy nieświadomy tego użytkownik, źle wyspecyfikuje format danych (najczęściej chodzi o specyfikację separatorów kolejnych pól (parametr *sep*) oraz dziesiętnego (parametr *dec*)). Dobrze więc zawczasu upewnić się jaką dokładnie strukturę mają kopiowane do bufora dane (np. z arkusza kalkulacyjnego Excel).

Innym sposobem odczytywania i zapisywania plików jest użycie tzw. połączeń (ang. connections). Jest to niskopoziomowy sposób korzystania z plików podobny do tego, jaki znamy z różnych języków programowania, np. C, Java. Wykonujemy wówczas 3 podstawowe czynności: inicjacja połączenia, wykorzystanie połączenia, zamknięcie połączenia.

Poniżej pokazano kilka wybranych przykładów pracy z plikami. Szczegóły użycia poszczególnych poleceń można znaleźć łatwo w plikach pomocy podręcznej projektu R oraz np. w [Bie08].

```
dane <- read.table("http://adres-serwera/nazwa-pliku", sep = "\t", dec = ".")
dane <- read.table(file = "clipboard")
dane <- read.table(file = "clipboard", sep = ";", header = TRUE)
write.table(dane, "clipboard", col.names = FALSE, row.names = FALSE)
write.csv(dane, file = "plik.csv", dec = ",")

polaczenie <- url("http://adres-serwera/nazwa-pliku.html", open = "r")
dane <- readLines(polaczenie)
close(polaczenie)

# Uwaga na slasze, muszą być w stylu unix-owym.
download.file(("http://adres-serwera/nazwa-pliku", "c:/temp/plik.pdf")
```

Ważnym elementem pracy z danymi jest możliwość korzystania z relacyjnych baz danych. R oferuje dwa mechanizmy dostępowe: za pomocą protokołu ODBC (ang. Open Database Connectivity) oraz DBI (ang. Database Interface) [DeBu00, RImpExp]. Oba protokoły działają bardzo podobnie: definiują i implementują niezależny od konkretnej bazy danych interfejs dostępowy (zbiór funkcji zwany API – Application Programming Interface). Historycznie rzecz biorąc protokół ODBC wywodzi się z systemu Windows, natomiast protokół DBI z systemów UNIX-owych i języka Perl. Obecnie jednak obie rodziny systemów operacyjnych wspierają zarówno ODBC,

jaki i DBI. Najpierw powstała specyfikacja DBI (pierwsza połowa lat 90-tych), wkrótce po niej firma Microsoft opracowała specyfikację ODBC.

W R mamy dostępne pakiety `RODBC`, `DBI` oraz sterowniki DBI dla konkretnych baz danych (np. `RMySQL`, `RPostgreSQL`, `ROracle`[9]). Pakiet DBI pełni rolę tzw. „front-end"-u, natomiast poszczególne sterowniki (np. `RMySQL`) pełnią rolę tzw. „back-end"-u. Poniżej pokazano sposób użycia dwóch wspomnianych wyżej interfejsów.

```
## ----------------------
## Interfejsc DBI
## ----------------------
library(DBI)
library(RMySQL)
# Tworzymy połączenie z bazą MySQL.
polaczenie <- dbConnect(MySQL(), user = "root", password = "mysql",
              dbname = "mysql", host = "localhost")
# Listujemy nazwy tabel.
dbListTables(polaczenie)
# Tworzymy zapytanie do bazy danych.
zapytanie <- dbSendQuery(polaczenie, "SELECT * FROM test")
# Feczujemy dane.
wynik <- fetch(zapytanie, n = -1)
# Zwalnia zasoby.
dbClearResult(zapytanie)
# Zwraca numer i opis ostatniego błędu.
dbGetException(polaczenie)

# Jak wyżej, ale wszystko w jednym poleceniu
# (tworzenie zapytania, feczowanie, wyświetlanie pobranych danych,
# zwalnianie zasobów).
dbGetQuery(polaczenie, "SELECT * FROM test")
# Zwalniamy połączenie.
dbDisconnect(polaczenie)
# Tworzymy ramkę danych.
tekst <- c("a", "b", "c")
liczby <- c(1:3)
ramka.danych = data.frame(tekst, liczby)
# Tworzymy nową tabele i wpisujemy do niej dane z ramki.
dbWriteTable(polaczenie, "test", ramka.danych)
# Tworzymy nową tabelę i wpisujemy do niej dane z pliku.
dbWriteTable(polaczenie, "test2", "dane.csv")
# Odczytujemy dane z tabeli i wpisujemy do nowej ramki danych.
tabela.jako.ramka <- dbReadTable(polaczenie, "test")
# Sprawdzamy, czy tabela istnieje w bazie danych.
dbExistsTable(polaczenie, "test")
# Kasowanie tabeli.
dbRemoveTable(polaczenie, "test")

## ----------------------
## Interfejsc ODBC
## ----------------------
```

[9] Pakiet `ROracle` w obecnej chwili dostępny jest tylko w postaci źródłowej i wymaga samodzielnego skompilowania przez użytkownika. Wymagany jest kompilator Microsoft Visual C/C++ oraz Oracle-owy prekompilator Pro*C/C++.

```
library(RODBC)
# Ustanawiamy nowe połączenie z bazą Oracle.
polaczenie <-odbcConnect("oracle10g", uid = "scott", pwd = "tiger")
# Jak wyżej, ale tryb interaktywny (wyświetla się GUI windowsowe).
polaczenie <- odbcDriverConnect("")
# Listujemy dostępne dla połączenia tabele.
sqlTables(polaczenie)
# Wykonujemy zapytanie i wyświetlenie wyniku.
sqlQuery(polaczenie, "SELECT * FROM emp")
# Pobieramy dane z tabeli i zapisujemy je do ramki danych.
ramka.danych <- sqlFetch(polaczenie, "EMP")
# Kasujemy tabelę.
sqlDrop(polaczenie, "moja_tabela")
# Kasujemy wszystkie rekordy z tabeli.
sqlClear(polaczenie, "moja_tabela")
# Zamykamy połączenie.
odbcClose(polaczenie)
```

6. Podsumowanie

W artykule, w wielkim skrócie, pokazano możliwości użycia R do wykonywania zadań kojarzonych powszechnie z eksploracją danych. Skupiono się tylko na najważniejszych zagadnieniach, pomijając wiele szczegółów, które łatwo jest znaleźć w bardzo bogatej literaturze. Przykładowo nie omówiono wielu innych zaimplementowanych w R metod grupowania i klasyfikacji danych. Wspomniano tylko o takich bardzo ważnych etapach eksploracji danych jak redukcja wymiarowości, czy też wstępne przygotowanie/czyszczenie danych. Pierwsze zagadnienie jest dokładniej omówione w drugim artykule autorów [Gra09], również prezentowanym na niniejszej konferencji. Wydaje się, że R daje na tyle duże wsparcie dla eksploracji danych, że może śmiało konkurować z innymi dedykowanymi dla tych zagadnień programami. Przewagą R nad innymi programami jest jego pełna otwartość kodu źródłowego, bardzo duża popularność w środowiskach naukowych (ale niestety chyba nie w środowiskach przemysłowych) oraz łatwość tworzenia nowych pakietów implementujących nowe algorytmy. Wspomniany tylko hasłowo w artykule pakiet `rattle` stanowi wygodny graficzny interfejs do szybkiego zapoznania się z najważniejszymi możliwościami R w dziedzinie eksploracji danych.

Bibliografia

[Bie08] Biecek P.: Przewodnik po pakiecie R, Oficyna wydawnicza GiS, Wrocław, 2008 (pierwsze 64 strony tej książki znajduje się pod adresem `http://www.biecek.pl/R/R.pdf`)

[BieDM] Biecek P.: Na przełaj przez Data Mining `http://www.biecek.pl/R/naPrzelajPrzezDM.pdf`

[Bor] `http://www.borgelt.net//apriori.html`

[Cham00] Chambers J. M. Programming with Data: A Guide to the S Language, Springer-Verlag, 2000

[DeBu00] Descartes A., Bunce T.: Programming the Perl DBI, O'REILLY, 2000

[DMSoft] `http://www.kdnuggets.com/software/suites.html`

[Gra09] Gramacki J., Gramacki A.: Redukcja wymiarowości oraz wizualizacja danych wielowymiarowych z wykorzystaniem projektu R, XV Konferencja użytkowników i deweloperów ORACLE, 20-23.X.2009 Zakopane-Kościelisko

[ESS] `http://ess.r-project.org/`

[Fei08] Feinerer I.: An introduction to text mining in R, R News, 8(2):19-22, Oct. 2008

http://CRAN.R-project.org/doc/Rnews/

[Fei09] Introduction to the tm Package, Text Mining in R
http://ftp5.gwdg.de/pub/misc/cran/web/packages/tm/vignettes/tm.pdf

[HBV01] Halkidi M., Batistakis Y., Vazirgiannis M.: On Clustering Validation Techniques, Journal of Intelligent Information Systems, 17:2/3, 107–145, 2001

[HaKa00] Han J.., Kamber M.: Data Mining: Concepts and Techniques, Morgan Kaufman, 2000

[HMS05] Hand D. J., Mannila H., Smyth P.: Pricinciples of Data Mining, MIT Press, Cambridge, MA, 2001; tłumaczenie polskie: Eksploracja danych,Wydawnictwa Naukowo-Techniczne, Warszawa, 2005

[KomR] Komsta Ł.: Wprowadzenie do środowiska R,
http://cran.r-project.org/doc/contrib/Komsta-Wprowadzenie.pdf

[KoMi04] Koronacki J., Mielniczuk J.: Statystyka, WTN, 2004

[KoCw05] Koronacki: J., Ćwik J.: Statystyczne systemy uczące się, WTN, 2005

[Lam] http://www.cs.put.poznan.pl/dweiss/xml/projects/lametyzator/index.xml?lang=pl

[Lar05] Larose D. T.: Discovering Knowledge in Data: An Introduction to Data Mining, Wiley, 2005; tłumaczenie polskie: Odkrywanie wiedzy z danych, Wydawnictwo Naukowe PWN, Warszawa, 2006

[Lar06] Larose D. T.: Data Mining Methods and Models, Wiley, 2006; tłumaczenie polskie: Metody i modele eksploracji danych, Wydawnictwo Naukowe PWN, Warszawa, 2008

[MLRep] http://www.ics.uci.edu/~mlearn/MLRepository.html

[Port] http://tartarus.org/~martin/PorterStemmer/

[RArch] http://r.meteo.uni.wroc.pl/

[Rattle] http://www.togaware.com/index.html

[RPro] http://www.r-project.org/

[RExt] R Development Core Team, Writing R Extensions,
http://cran.r-project.org/doc/manuals/R-exts.pdf

[RImpExp] R Development Core Team, R Data Import/Export
http://cran.r-project.org/doc/manuals/R-data.pdf

[SPLUS] http://www.insightful.com/

[STATET] www.walware.de/goto/statet

[TinnR] http://www.sciviews.org/Tinn-R/

[VeRi02] Venables W.N., Ripley B. D.: Modern Applied Statistics with S. Fourth Edition, Springer-Verlag, 2002

[WaGa09] Walesiak M., Gatnar E. (Ed.): Statystyczna analiza danych z wykorzystaniem programu R, Wydawnictwo Naukowe PWN 2009

[Wazniak] http://wazniak.mimuw.edu.pl/index.php?title=Eksploracja_danych

[Weka] http://www.cs.waikato.ac.nz/~ml/weka/