Marek Walesiak
Uniwersytet Ekonomiczny we Wrocławiu

LOSOWE GENEROWANIE DANYCH O ZNANEJ STRUKTURZE KLAS W PAKIECIE `clusterSim`

1. Wprowadzenie

Sprawdzenie przydatności wielu metod modelowania i prognozowania zjawisk społeczno-gospodarczych wymaga ich przetestowania na losowo wygenerowanych danych o znanej strukturze klas. W literaturze przedmiotu znane są propozycje generowania danych:

a) o znanej strukturze klas, w których położenie i jednorodność skupień zadaje się za pomocą wektorów wartości przeciętnych (środki ciężkości skupień) i macierzy kowariancji (rozproszenie obiektów) – zob. np. prace [Atlas, Overall 1994; Milligan 1985],

b) zawierających skupienia o zadanym stopniu separowalności – zob. prace [Qiu, Joe 2006; Steinley, Henson 2005].

W artykule zaprezentowana zostanie funkcja `cluster.Gen` pakietu `clusterSim` (zob. [Walesiak, Dudek 2008a; Walesiak, Dudek 2008b]) służąca do losowego generowania zbiorów danych o znanej strukturze klas, pracująca w środowisku **R** (zob. [*R: A Language*... 2008]), która ma następujące zalety:

– pozwala na generowanie danych metrycznych (ilorazowych i przedziałowych), porządkowych oraz symbolicznych przedziałowych dla danej liczby wymiarów (zmiennych) – np. pięciu skupień w przestrzeni trójwymiarowej,

– położenie i jednorodność skupień zadaje się za pomocą wektorów wartości przeciętnych (środki ciężkości skupień) i macierzy kowariancji (rozproszenie obiektów),

– istnieje możliwość wygenerowania klas o różnej gęstości i kształcie,

– istnieje możliwość uwzględnienia zmiennych zakłócających strukturę klas (*noisy variables*) oraz obserwacji odstających (*outliers*).

2. Charakterystyka funkcji `cluster.Gen` pakietu `clusterSim` oraz jej możliwości

Składnię funkcji `cluster.Gen` z pakietu `clusterSim` prezentuje tabela 1.

Tabela 1

Charakterystyka składni funkcji `cluster.Gen` z pakietu `clusterSim`

`cluster.Gen(numObjects=50, means=NULL, cov=NULL, fixedCov=TRUE, model=1, dataType="m", numCategories=NULL, numNoisyVar=0, numOutliers=0, rangeOutliers=c(1,10), inputType="csv2", inputHeader=TRUE, inputRowNames=TRUE, outputCsv="", outputCsv2="", outputColNames=TRUE, outputRowNames=TRUE)`	
`numObjects`	liczba obiektów w każdym skupieniu – dodatnia liczba całkowita lub wektor o tym samym rozmiarze jak *nrow(means)*, np. `numObjects=c(50,20)`
`means`	macierz wartości oczekiwanych (np. `means=matrix(c(0,8,0,8),2,2)`). Jeżeli `means=NULL`, to macierz należy wczytać z pliku *means_<modelNumber>.csv file*
`cov`	macierz kowariancji taka sama dla wszystkich skupień, np. `cov=matrix(c(1,0,0,1),2,2)`. Jeżeli `cov=NULL`, macierz kowariancji należy wczytać z pliku *cov_<modelNumber>.csv file*
`model`	numery modeli: `model=1` – brak struktury klas `model=2` – wartości przeciętne oraz kowariancje odczytane z argumentów `means` i `cov` `model=3,4,...,20` – przykładowe modele z zadanymi wektorami wartości przeciętnych i macierzami kowariancji `model=21,22,...` – jeżeli `fixedCov=TRUE`, wektory wartości przeciętnych means należy wczytać z pliku *means_<modelNumber>.csv*, a macierz kowariancji z pliku *cov_<modelNumber>.csv*. Jeżeli `fixedCov=FALSE`, `means` należy wczytać z pliku *means_<modelNumber>.csv*, a macierze kowariancji dla poszczególnych skupień z plików *cov_<modelNumber>_<clusterNumber>.csv*
`fixedCov`	`fixedCov=TRUE` – macierz kowariancji dla wszystkich skupień jest jednakowa `fixedCov=FALSE` – macierze kowariancji dla skupień są zróżnicowane
`dataType`	`"m"` – dane metryczne (ilorazowe, przedziałowe), `"o"` – dane porządkowe, `"s"` – dane symboliczne przedziałowe
`numCategories`	liczba kategorii (tylko dla danych porządkowych). Dodatnia liczba całkowita lub wektor o rozmiarze: *ncol(means)* plus liczba zmiennych zakłócających
`numNoisyVar`	liczba zmiennych zakłócających strukturę klas (gdy `model=1` `numNoisyVar`, oznacza liczbę zmiennych)
`numOutliers`	liczba obiektów odstających. Dodatnia liczba całkowita oznacza liczbę obiektów odstających, a wartość z przedziału <0, 1> odsetek z całego zbioru obiektów
`rangeOutliers`	rozstęp dla obiektów odstających dla każdego wymiaru (zmiennej) z osobna (domyślnie [1, 10])
...	pozostałe argumenty dotyczą sposobu wczytywania i zapisywania

Źródło: opracowanie własne na podstawie dokumentacji pakietu `clusterSim`.

Zdecydowana większość funkcji programu **R** zwraca złożone obiekty lub listy zawierające wiele informacji wygenerowanych przez wykonywany algorytm. Funkcja `cluster.Gen` zwraca następujące informacje:

`clusters` – numer skupienia dla każdego obiektu. Dla modelu pierwszego (`model=1`), w którym nie ma struktury klas funkcja `clusters` zwraca numery obiektów,

`data` – wygenerowane dane: dla danych metrycznych i porządkowych – macierz (w wierszach obiekty, a w kolumnach zmienne); dla danych symbolicznych przedziałowych trójwymiarowa struktura: pierwszy wymiar oznacza numer obiektu, drugi wymiar – numer zmiennej, a trzeci – dolny i górny kraniec przedziału.

Dane metryczne (`dataType="m"`) generowane są z wielowymiarowego rozkładu normalnego, w którym położenie i jednorodność skupień zadaje się za pomocą wektorów wartości przeciętnych (środki ciężkości skupień) i macierzy kowariancji (rozproszenie obiektów) – zob. [Grabiński, Wydymus, Zeliaś 1989, s. 141–146]. Tylko dla modelu 1, w którym nie ma w zbiorze danych struktury klas, obserwacje generowane są z rozkładu jednostajnego dla jednostkowej hiperkostki o liczbie wymiarów (zmiennych) podanych w `numNoisyVar`.

Funkcja `cluster.Gen` zawiera 14 wbudowanych przykładowych modeli (oznaczonych w pakiecie numerami 3–16), z zadanymi wektorami wartości przeciętnych i macierzami kowariancji, różniących się:

– liczbą zmiennych (wymiarów) i liczbą skupień (*known number of true clusters and true dimensions*),

– gęstością skupień (*cluster density*), tj. liczebnością obiektów w klasach,

– kształtem skupień (*shape of clusters*). I tak modele 3–5 zawierają skupienia wydłużone, modele 6 i 7 – skupienia wydłużone i słabo separowalne, modele 8–11 skupienia o kształcie normalnym. Modele od 13 do 16 zawierają zróżnicowane macierze kowariancji dla poszczególnych skupień, co oznacza różne kształty dla poszczególnych skupień. Model 12 jest nietypowy, z jego wykorzystaniem generuje się bowiem dane zawierające cztery klasy dla jednej zmiennej.

Konstruując zaprezentowane modele, wzorowano się na licznych opracowaniach (zob. np. modele służące do testowania struktury klas zawarte w pracach: [Dudoit, Fridlyand 2002; Soffritti 2003; Tibshirani, Walther, Hastie 2001; Tibshirani, Walther 2005]). Należy podkreślić, że w pakiecie `clusterSim` można wprowadzać własne modele (zob. modele z tabeli 1 oznaczone numerami 21, 22, ...).

Generowanie obserwacji porządkowych (`dataType="o"`) przebiega w sposób następujący. Wygenerowane obserwacje dla modeli mają charakter ciągły (dane metryczne). W celu otrzymania danych porządkowych należy przeprowadzić dla każdej zmiennej proces dyskretyzacji. Liczba kategorii (k_j) zmiennej porządkowej X_j określa szerokość przedziału klasowego $\left\lceil \max_i\{x_{ij}\} - \min_i\{x_{ij}\} \right\rceil / k_j$. Niezależnie dla każdej zmiennej kolejne przedziały klasowe otrzymują kategorie 1, ..., k_j

i aktualna wartość zmiennej x_{ij} jest zastępowana przez te kategorie. Dla poszczególnych zmiennych liczba kategorii może być inna (np. $k_1 = 7$, $k_2 = 4$, $k_3 = 5$), zatem składnia argumentu jest następująca: `numCategories=c(7,4,5)`. W przypadku wprowadzenia np. dwóch zmiennych zakłócających (`numNoisyVar=2`) dodatkowo dla tych zmiennych w składni argumentu numCategories należy podać liczby kategorii (np. `numCategories=c(7,4,5,6,4)`).

Dane symboliczne przedziałowe otrzymuje się w wyniku dwukrotnego generowania obserwacji dla danego modelu. Otrzymuje się dwa zbiory obserwacji A i B, dla których wartość minimalna (maksymalna) z wartości $\left\{x_{ij}^A, x_{ij}^B\right\}$ jest traktowana jako początek (koniec) przedziału klasowego.

Obserwacje na zmiennych zakłócających generowane są niezależnie z rozkładu jednostajnego. Przedział zmienności zmiennych zakłócających jest podobny do zmiennych wyznaczających strukturę klas (por. [Milligan 1985; Qiu, Joe 2006, s. 322]).

Obiekty odosobnione (*outliers*) generowane są tylko dla danych metrycznych oraz symbolicznych przedziałowych, niezależnie dla każdej zmiennej i całego zbioru obserwacji z rozkładu jednostajnego. Następnie wygenerowane wartości są losowo dodawane do wartości maksymalnej j-tej zmiennej lub odejmowane od wartości minimalnej j-tej zmiennej.

3. Przykłady z wykorzystaniem funkcji `cluster.Gen` pakietu `clusterSim`

Przedstawione zostaną przykłady generowania danych metrycznych, porządkowych oraz symbolicznych przedziałowych prezentujące klasy o różnej gęstości i kształcie z uwzględnieniem zmiennych zakłócających strukturę klas oraz obserwacji odstających.

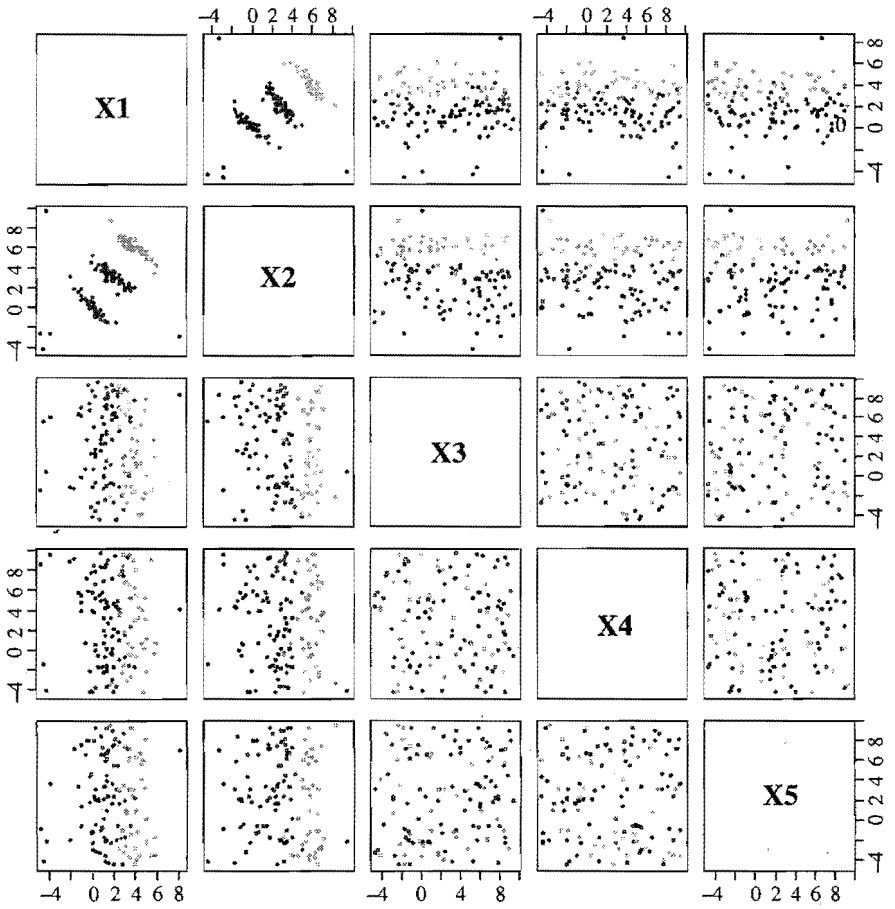
Dane metryczne. Za pomocą dwuwymiarowej zmiennej losowej o rozkładzie normalnym wygenerowano odpowiednio 30, 50 i 40 obserwacji dla trzech skupień o wydłużonym kształcie. Przyjęto następujące wektory wartości oczekiwanych dla skupień (0, 0), (2, 3), (4, 6) oraz identyczne macierze kowariancji $\Sigma(\sigma_{jj} = 1, \sigma_{jl} = -0{,}9)$. Do analizy wprowadzono dodatkowe trzy zmienne zakłócające istniejącą w układzie dwuwymiarowym strukturę klas (tzw. *noisy variables*). Po 120 obserwacji na tych zmiennych wygenerowano niezależnie z rozkładu jednostajnego. Ponadto wprowadzono pięć obiektów odstających.

Do wygenerowania danych wykorzystano funkcję `cluster.Gen` pakietu `clusterSim` z następującą składnią poleceń (zob. przykład 1 oraz rys. 1).

Przykład 1

```
> library(clusterSim)
> means <- matrix(c(0,2,4,0,3,6), 3, 2)
```

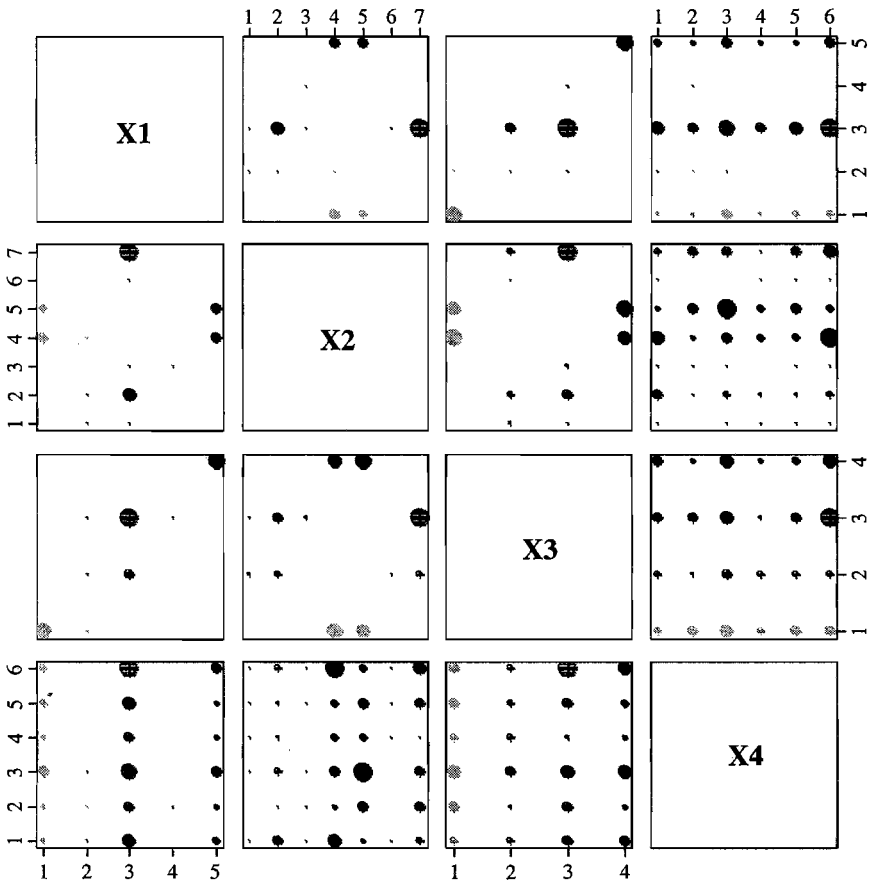
```
> cov <- matrix(c(1,-0.9,-0.9,1), 2, 2)
> grnd1 <- cluster.Gen(numObjects=c(30,50,40), means=means,
                       cov=cov, model=2, numNoisyVar=3,
                       numOutliers=5, rangeOutliers=c(1,3))
> colornames <- c("red","blue","green","brown"))
> grnd1$clusters[grnd1$clusters==0] <- length(colornames)
> plot(grnd1$data, col=colornames[grnd1$clusters])
> write.table(grnd1$data, file="dane_1.csv", sep=";", dec=",")
```



Rys. 1. 120 obserwacji pięciu zmiennych w układach dwuwymiarowych
Źródło: opracowanie własne.

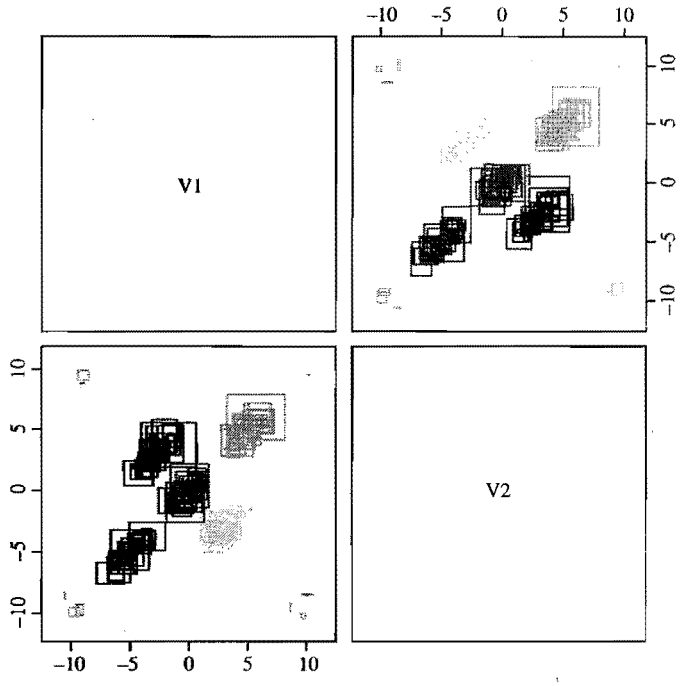
Dane porządkowe. Składnię poleceń pozwalającą na wygenerowanie danych porządkowych zawiera przykład 2. Do wygenerowania danych wykorzystano

```
> grnd3$clusters[grnd3$clusters==0] <- length(colornames)
> plotInterval(grnd3$data, cl=grnd3$clusters,
              clColors=colornames)
> write.table(grnd3$data, file="dane_3.csv", sep=";", dec=",")
```



Rys. 2. Graficzna prezentacja wygenerowanych danych porządkowych dla modelu 14
Źródło: opracowanie własne.

Do stworzenia wykresu rozrzutu dla danych symbolicznych przedziałowych przeznaczona jest funkcja `plotInterval` pakietu `clusterSim`, która przedstawia pojedynczy obiekt nie jako punkt, ale jako prostokąt wyznaczony przez zmienne symboliczne przedziałowe. Efekt działania funkcji przedstawia rys. 3.



Rys. 3. Wykres rozrzutu dla danych symbolicznych przedziałowych

Źródło: opracowanie własne.

Literatura

Atlas R.S., Overall J.E. [1994], *Comparative Evaluation of Two Superior Stopping Rules for Hierarchical Cluster Analysis*, „Psychometrika", nr 59.

Dudoit S., Fridlyand J. [2002], *A Prediction-based Resampling Method for Estimating the Number of Clusters in a Dataset*, „Genome Biology", nr 3(7).

Grabiński T., Wydymus S., Zeliaś A. [1989], *Metody taksonomii numerycznej w modelowaniu zjawisk społeczno-gospodarczych*, PWN, Warszawa.

Milligan G.W. [1985], *An Algorithm for Generating Artificial Test Clusters*, „Psychometrika", nr 1.

Qiu W., Joe H. [2006], *Generation of Random Clusters with Specified Degree of Separation*, „Journal of Classification", vol. 23.

R: A Language and Environment for Statistical Computing [2008], R Development Core Team, R Foundation for Statistical Computing, Vienna, URL, http://www.R-project.org.

Soffritti G. [2003], *Identifying Multiple Cluster Structures in a Data Matrix*, „Communications in Statistics. Simulation and Computation", vol. 32, nr 4.

Steinley D., Henson R. [2005], *OCLUS: An Analytic Method for Generating Clusters with Known Overlap*, „Journal of Classification", vol. 22.

Tibshirani R., Walther G., Hastie T. [2001], *Estimating the Number of Clusters in a Data Set Via the Gap Statistic*, „Journal of the Royal Statistical Society", ser. B, vol. 63, part 2.

Tibshirani R., Walther G. [2005], *Cluster Validation by Predicting Strength*, „Journal of Computational and Graphical Statistics", vol. 14, nr 3.

Walesiak M., Dudek A. [2008a], *clusterSim package*, URL http://www.R-project.org.

Walesiak M., Dudek A., *Identification of Noisy Variables for Nonmetric and Symbolic Data in Cluster Analysis* [w:] *Data Analysis, Machine Learning and Applications* [2008b], red. C. Preisach, H. Burkhardt, L. Schmidt-Thieme, R. Decker, Springer-Verlag, Berlin, Heidelberg.