Pakiet missMDA

Małgosia Łazęcka

9.05.2018

Wprowadzenie

Otrzymujemy **dane wielowymiarowe** z **brakami danych**.

Znana metoda redukcji wymiaru dla wielowymiarowych zmiennych ciągłych: **PCA**.

Dla zmiennych kategorycznych - MCA (*multiple correspondence analysis*), dla zmiennych mieszanych - FAMD (*factorial analysis for mixed data*), dla zmiennych tworzących grupy (rozszerzenie wszystkich poprzednich) - MFA (*multiple factor analysis*).

Wprowadzenie

Otrzymujemy **dane wielowymiarowe** z **brakami danych**.

Znana metoda redukcji wymiaru dla wielowymiarowych zmiennych ciągłych: **PCA**.

Dla zmiennych kategorycznych - MCA (*multiple correspondence analysis*), dla zmiennych mieszanych - FAMD (*factorial analysis for mixed data*), dla zmiennych tworzących grupy (rozszerzenie wszystkich poprzednich) - MFA (*multiple factor analysis*).

Problem:

Dane wielowymiarowe, występują braki danych, nie chcemy ich imputować przed rozpoczęciem analizy ze względu na to, że wariancja estymowanych współczynników może zostać niedoszacowana - **czy możemy wykorzystać PCA?**

Pakiet `missMDA`

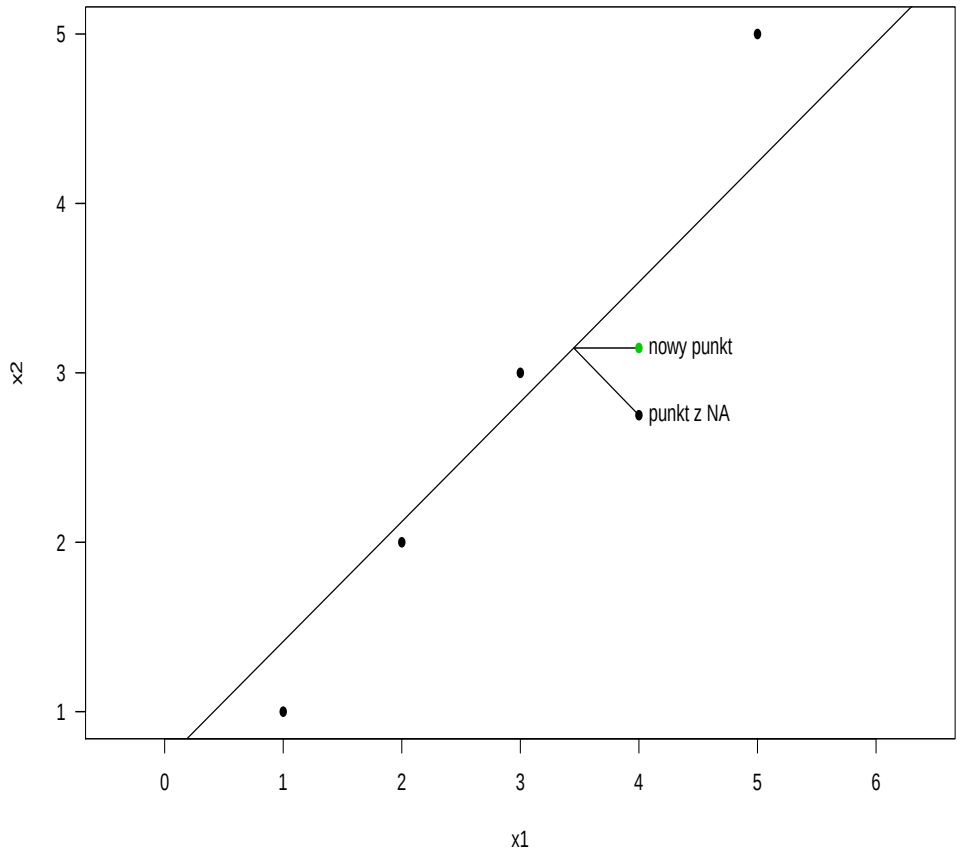
Najważniejsze hasła:

- metody składowych głównych,
- zmienne różnych typów,
- braki danych,
- imputacja braków danych,

Jak to działa?

Iteracyjne PCA

```
DF <- data.frame(x1 = c(1,2,3,4,5),
                 x2 = c(1,2,3,NA,5))
```



Uwagi

Iteracyjne PCA - problem przeuczania. Rozwiązanie: iteracyjne PCA z regularyzacją (`default`).

Liczba składowych głównych S wykorzystywanych do imputacji danych znajdowana jest kroswalidacyjnie na podstawie błędu:

$$\mathrm{MSEP}(S) = \sum_{i=1}^{n}\sum_{j=1}^{p}(x_{i,j} - \overline{x}_{i,j}^{S})^2/np.$$

Przykładowe dane

```
library("missMDA")
data("geno", package = "missMDA")
head(round(geno, 2))
```

```
##      ACOR AORE  ASAL  CALB  CBAD  CCOR  CLER  CSE1  CSE2  CTO1
## C_1 0.64 0.43  0.37    NA -0.64 -0.14    NA    NA  0.16 -0.47
## C_2 2.17 0.86    NA  0.01    NA  0.04 -1.32 -0.56 -0.77 -0.53
## C_3 0.85 0.96 -0.26 -0.13 -0.39  0.27 -0.20 -0.49 -0.33 -0.27
## C_4 1.41 1.50 -0.03 -0.10 -0.32 -0.16 -0.73 -0.54 -1.09  0.07
## C_5 0.87 0.86  0.54 -0.10 -0.03  0.05 -0.88 -0.34 -0.66 -0.32
## C_6 1.71 0.34  0.53  0.17    NA -0.34 -0.83 -0.51 -1.03  0.06
```

16 rows corresponding to genotypes (triticale lines) and 10 columns corresponding to different environments where the genotypes were sown

Znajdowanie optymalnej liczby składowych głównych S

```
ncomp <- estim_ncpPCA(geno)
#                     method.cv = c("gcv","loo","Kfold")
ncomp$ncp
```

```
## [1] 2
```

```
ncomp$criterion
```

```
##         0         1         2         3         4         5
## 0.4059200 0.2069907 0.1521044 0.1999721 0.2308990 0.3711718
```

Imputacja danych

```
res.imp <- imputePCA(geno, ncp = ncomp$ncp)
head(round(res.imp$completeObs, 2))
```

```
##      ACOR AORE  ASAL  CALB  CBAD  CCOR  CLER  CSE1  CSE2  CTO1
## C_1 0.64 0.43  0.37 -0.07 -0.64 -0.14 -0.51 -0.29  0.16 -0.47
## C_2 2.17 0.86  0.50  0.01 -0.56  0.04 -1.32 -0.56 -0.77 -0.53
## C_3 0.85 0.96 -0.26 -0.13 -0.39  0.27 -0.20 -0.49 -0.33 -0.27
## C_4 1.41 1.50 -0.03 -0.10 -0.32 -0.16 -0.73 -0.54 -1.09  0.07
## C_5 0.87 0.86  0.54 -0.10 -0.03  0.05 -0.88 -0.34 -0.66 -0.32
## C_6 1.71 0.34  0.53  0.17 -0.36 -0.34 -0.83 -0.51 -1.03  0.06
```

```
head(round(res.imp$fittedX, 2))
```

```
##      [,1] [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9] [,10]
## [1,] 1.00 0.67  0.16 -0.07 -0.33 -0.07 -0.51 -0.29 -0.48 -0.20
## [2,] 1.86 1.02  0.50 -0.05 -0.56 -0.14 -0.93 -0.60 -0.94 -0.34
## [3,] 0.83 0.81 -0.10 -0.16 -0.31 -0.05 -0.41 -0.19 -0.35 -0.20
## [4,] 1.30 0.89  0.19 -0.10 -0.42 -0.09 -0.65 -0.38 -0.62 -0.26
## [5,] 1.08 0.63  0.27 -0.04 -0.35 -0.08 -0.55 -0.34 -0.54 -0.20
## [6,] 1.29 0.33  0.71  0.13 -0.36 -0.12 -0.69 -0.49 -0.72 -0.18
```

Imputacja danych - argumenty i PCA

```
imputePCA(X, ncp = 2, scale = TRUE, method = c("Regularized","EM"),
          row.w = NULL, coeff.ridge = 1, threshold = 1e-06,
          seed = NULL, nb.init = 1, maxiter = 1000, ...)
```

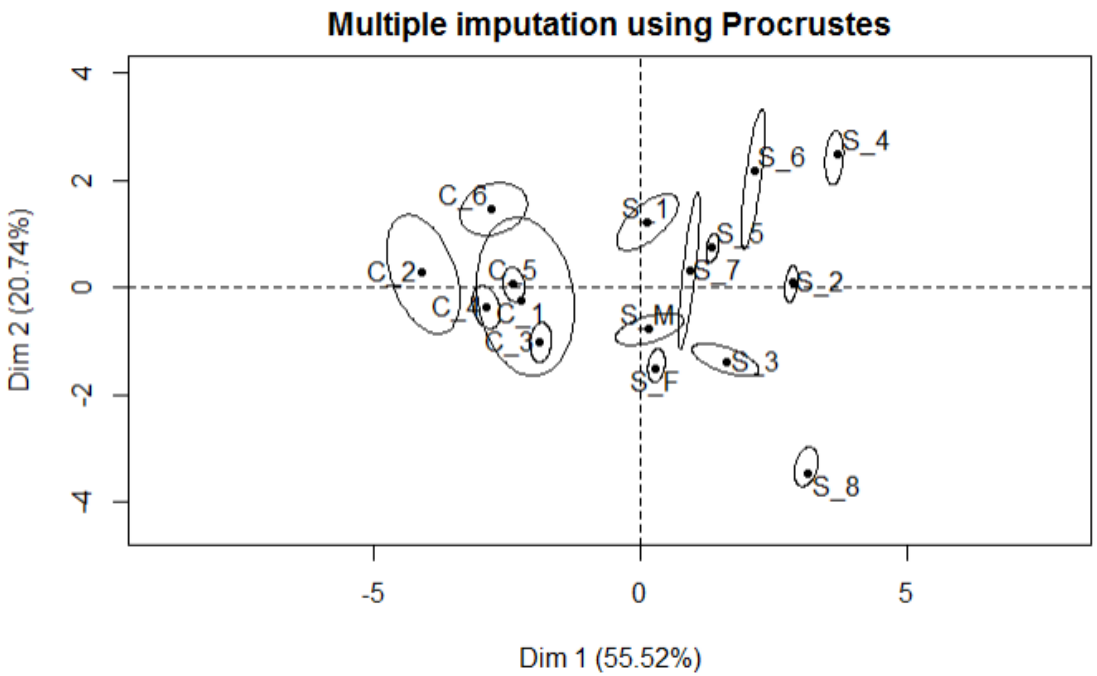
- `ncp` - liczba składowych głównych użytych do analizy
- `method` - `Regularized`: iteracyjne PCA z regularyzacją, `EM`: iteracyjne PCA
- `coeff.ridge` - 1: iteracyjne PCA z regularyzacją, im mniej, tym bliżej `EM`
- `scale`, `row.w` - umożliwiają nadanie wag odpowiednio zmiennym i obserwacjom

PCA

```
library("FactoMineR")
res.pca <- PCA(res.imp$completeObs, graph = FALSE)
```

Wykresy

```
resMIPCA <- MIPCA(geno, ncp = 2, nboot = 200)
plot(resMIPCA, choice= "ind.proc")
```



Dyskusja

Łatwość użycia

Raczej łatwy.

Elastyczność

To zależy.

Inne uwagi

Czas.

Opis.

Brak wyniku PCA.

Kod

Przykład ze slajdów:

```
library("missMDA")
data("geno", package = "missMDA")
ncomp <- estim_ncpPCA(geno)
res.imp <- imputePCA(geno, ncp = ncomp$ncp)
library("FactoMineR")
res.pca <- PCA(res.imp$completeObs, graph = FALSE)
```

Inny przykład:

```
data("vnf", package = "missMDA")
# UWAGA! to chwilę trwa:
ncomp <- estim_ncpMCA(vnf, method.cv = "Kfold")
tab.disj.impute <- imputeMCA(vnf, ncp = 4)$tab.disj
res.mca <- MCA(vnf, tab.disj = tab.disj.impute, graph=FALSE)

# Jeśli nie chcesz czekać, policz z tymi danymi:
set.seed(123)
vnf.male <- vnf[sample(1:nrow(vnf),200),]
```