Matrike

A. Blejec

7. maj 2013

Podatki

Podaki so o študentih 3. letnika biologije v letu 2012/13 so v datoteki lfn in na http://bit.ly/16oBVpR

```
fpath <- file.path("../data", lfn)
data <- read.table(fpath, header = TRUE, sep = "\t")
names(data)
```

```
[1] "starost" "mesec"   "spol"    "masa"    "visina"  "
[7] "cevelj"  "lasje"   "oci"     "mati"    "oce"     "
```

Popravljanje podatkov

Odstranimo ta starga

```
data <- data[data$starost < 30, ]
```

Podatke o mesecu 0 spremenimo v `NA`

```
data[data$mesec == 0, "mesec"] <- NA
table(data$mesec)
```

```
1  2  3  4  5  6  7  8  9 10 11
1  3  2  3  4  3  7  5  2  5  6
```

Sprememba podatkov o velikosti majice

```
table(data$majica)

 L  M  S XL XS
 4 19 16  1  2

data$majica[data$majica == "XL"] <- "L"
data$majica[data$majica == "XS"] <- "S"
data$majica <- ordered(data$majica, levels = c("S",
+     "M", "L"))
str(data$majica)

 Ord.factor w/ 3 levels "S"<"M"<"L": 1 1 1 2 3 1 1 2 2 3

table(data$majica)

 S  M  L
18 19  5
```

Vektorji

Vektor pripravimo kot n-terico s pomočjo funkcije c:

```
x <- c(1, 2, -1)
y <- c(2, 1, 6)
x
```

```
[1]  1  2 -1
```

```
y
```

```
[1] 2 1 6
```

Vektor ima dolžino:

```
length(x)
```

```
[1] 3
```

Vektorska aritmetika

Z vektorji lahko računamo po komponentah:

```
 2 * x
[1]  2  4 -2
 x + y
[1] 3 3 5
 x - y
[1] -1  1 -7
 x * y
[1]  2  2 -6
 x/y
[1]  0.5000000  2.0000000 -0.1666667
 x^y
[1] 1 2 1
```

Za ročni vnos in urejanje matrik lahko uporabimo urejevalec tabel `edit`.

```
 if (interactive()) {
+     X0 <- make.matrix(n = 5, m = 2)
+     X0
+ }
```

Množenje %*% in transpozicija t()

```
(B <- cbind(x, y))
```

```
     x y
[1,]  1 2
[2,]  2 1
[3,] -1 6
```

```
(A <- matrix(c(1, 2, -1, 0), 2, 2))
```

```
     [,1] [,2]
[1,]    1   -1
[2,]    2    0
```

```
(C <- (B %*% A))
```

```
     [,1] [,2]
[1,]    5   -1
[2,]    4   -2
[3,]   11    1
```

```
t(C)
```

```
     [,1] [,2] [,3]
[1,]    5    4   11
[2,]   -1   -2    1
```

```
X0 <- cbind(x, y)
X0
```

```
     x y
[1,]  1 2
[2,]  2 1
[3,] -1 6
```

```
X <- X0
dim(X)
```

```
[1] 3 2
```

```
n <- dim(X)[1]
dimnames(X)
```

```
[[1]]
NULL

[[2]]
[1] "x" "y"
```

Povprečja

```
M <- apply(X0, 2, mean)
M
```

```
        x         y
0.6666667 3.0000000
```

matrika povprečij

```
t(t(rep(1, n))) %*% t(M)
```

```
                x y
[1,] 0.6666667 3
[2,] 0.6666667 3
[3,] 0.6666667 3
```

vsredinjena matrika podatkov

```
X <- scale(X, scale = F)
X
```

```
                x  y
[1,]  0.3333333 -1
[2,]  1.3333333 -2
[3,] -1.6666667  3
attr(,"scaled:center")
        x         y
0.6666667 3.0000000
```

vektor povprečij izvlečemo s funkcijo attr

```
attr(X, "scaled:center")
```

```
        x         y
0.6666667 3.0000000
```

Matrika SSP

```
C <- t(X) %*% X
C
```

```
           x  y
x  4.666667 -8
y -8.000000 14
```

Kovariančna matrika

```
S <- C/(n - 1)
S
```

```
           x  y
x  2.333333 -4
y -4.000000  7
```

```
cov(X)
```

```
           x  y
x  2.333333 -4
y -4.000000  7
```

```
diag(S)
```

```
       x        y
2.333333 7.000000
```

Matrika varianc

```
diag(diag(S))
```

```
         [,1] [,2]
[1,] 2.333333    0
[2,] 0.000000    7
```

```
SD1 <- sqrt(diag(1/diag(S)))
SD1
```

```
          [,1]      [,2]
[1,] 0.6546537 0.0000000
[2,] 0.0000000 0.3779645
```

Korelacijska matrika

```
R <- SD1 %*% S %*% SD1
R
```

```
             [,1]       [,2]
[1,]  1.0000000 -0.9897433
[2,] -0.9897433  1.0000000
```

Še enkrat

```
SX <- scale(X)
t(SX) %*% SX/(n - 1)
```

```
             x          y
x  1.0000000 -0.9897433
y -0.9897433  1.0000000
```

```
cor(X)
```

```
             x          y
x  1.0000000 -0.9897433
y -0.9897433  1.0000000
```

Centroidi na več načinov

```
(X <- cbind(x, y))
```

```
      x y
[1,]  1 2
[2,]  2 1
[3,] -1 6
```

```
(colMeans(X))
```

```
        x         y
0.6666667 3.0000000
```

```
(attr(scale(X), "scaled:center"))
```

```
        x         y
0.6666667 3.0000000
```

```
(apply(X, 2, mean))
```

```
        x         y
0.6666667 3.0000000
```

Standardni odkloni

Standardne odklone bi lahko izvlekli kot koren diagonal kovariančne matrike, kot atribut `scale` ali pa z `apply`:

```
(sqrt(diag(var(X))))
```

```
       x        y
1.527525 2.645751
```

```
(attr(scale(X), "scaled:scale"))
```

```
       x        y
1.527525 2.645751
```

```
(apply(X, 2, sd))
```

```
       x        y
1.527525 2.645751
```

Funkcije za linearno algebro

Na voljo imamo funkcije, za manipulacijo matrik in pomembne operacije nad matrikami

- `t()` transponiranje matrik
- `eigen()` lastne vrednosti
- `solve()` inverzna matrika
- `det()` determinanta matrike
- sled izračunamo kot vsoto diagonalnih elementov

Lastni vektorji

```
S <- matrix(c(10, 3, 3, 2), 2, 2)
S
```

```
     [,1] [,2]
[1,]   10    3
[2,]    3    2
```

```
eigen(S)
```

```
$values
[1] 11  1

$vectors
           [,1]       [,2]
[1,] -0.9486833  0.3162278
[2,] -0.3162278 -0.9486833
```

```
det(S)
```

```
[1] 11
```

Inverzna matrika

```
S1 <- solve(S)
S %*% S1
```

```
                [,1] [,2]
[1,]  1.000000e+00    0
[2,] -2.220446e-16    1
```

Takole se znebimo zelo majhnih vrednosti v izpisu

```
zapsmall(S %*% S1)
```

```
     [,1] [,2]
[1,]    1    0
[2,]    0    1
```

Sled bi lahko izračunali kot

```
sum(diag(S))
```

```
[1] 12
```

- Iz podatkov `data` izberite nekaj (3 ali 4) številske spremenljivke in formirajte matriko `X`
- Narišite pare razsevnih diagramov - funkcija `pairs()`
- Poiščite povprečne vrednosti spremenljivk
- Izračunajte kovariančno in korelacijsko matriko

Podatki

```
 X <- data[, c("masa", "visina", "mesec", "roke",
+     "cevelj")]
 head(X)
```

```
  masa visina mesec roke cevelj
2   60    173     1  176     43
3   55    178     7  178     39
4   70    167     8  165     39
5   65    171     4  168     40
6   88    171     3  173     41
7   52    162     7  164     39
```

```
 tail(X)
```

```
   masa visina mesec roke cevelj
38   73    173    10  180     42
39   58    170    10  171     48
40   56    158     7  156     37
41   55    157     4   NA     37
42   50    160    10  160     37
43   73    181    NA  187     43
```

```
cor(X, use = "complete.obs")
```

```
             masa    visina      mesec      roke     ceve
masa    1.00000000 0.6786316 0.03182096 0.6657479 0.54749
visina 0.67863160 1.0000000 0.17764477 0.9393966 0.69042
mesec   0.03182096 0.1776448 1.00000000 0.1609260 0.10377
roke    0.66574793 0.9393966 0.16092600 1.0000000 0.72258
cevelj 0.54749044 0.6904270 0.10377266 0.7225814 1.00000
```

Test hipotez

Ali so fantje večji od deklet

```
 t.test(visina ~ spol, data = data)

Welch Two Sample t-test

data:  visina by spol
t = -6.4643, df = 12.502, p-value = 2.55e-05
alternative hypothesis: true difference in means is not
95 percent confidence interval:
 -17.901862  -8.906219
sample estimates:
mean in group F mean in group M
       166.8182        180.2222
```

Funkcija za Student t-test

Ali so fantje večji od deklet

```
 student <- function(x, y) {
+ }
```

Permutacijski test

$$H_0 : \mu_1 = \mu_2 (= \mu)$$

Če imamo opravka z vzorcema iz iste populacije, potem je lahgko v vsakem vzorcu katerakoli od izmerjenih $n_1 + n_2$ vrednosti. Če imam npr. 10

```
n
[1] 10
sample(10, 4)
[1]  9  8 10  2
```

lahko takole določim, katere vrednosti spadajo v prvi vzorec.

Podatki

Zgenerirajmo podatke iz populacije s povprečjem 10

```
set.seed(555)
n1 <- 5
mu1 <- 10
sd1 <- 1
n2 <- 5
mu2 <- 10
sd2 <- 1
n <- n1 + n2
X1 <- round(rnorm(n1, mu1, sd1), 1)
X2 <- round(rnorm(n2, mu2, sd2), 1)
X1
```

```
[1]  9.7 10.5 10.4 11.9  8.2
```

```
X2
```

```
[1] 10.9  9.8 11.4 10.0 10.6
```

```
X <- c(X1, X2)
ind <- c(rep(1, n1), rep(2, n2))
```

Funkcija za razliko povprečij

```
dx <- function(x, y) mean(y) - mean(x)
dx(X1, X2)
```

```
[1] 0.4
```

Premešajmo podatke

```
set.seed(432)
(smp1 <- sample(n, n1))
```

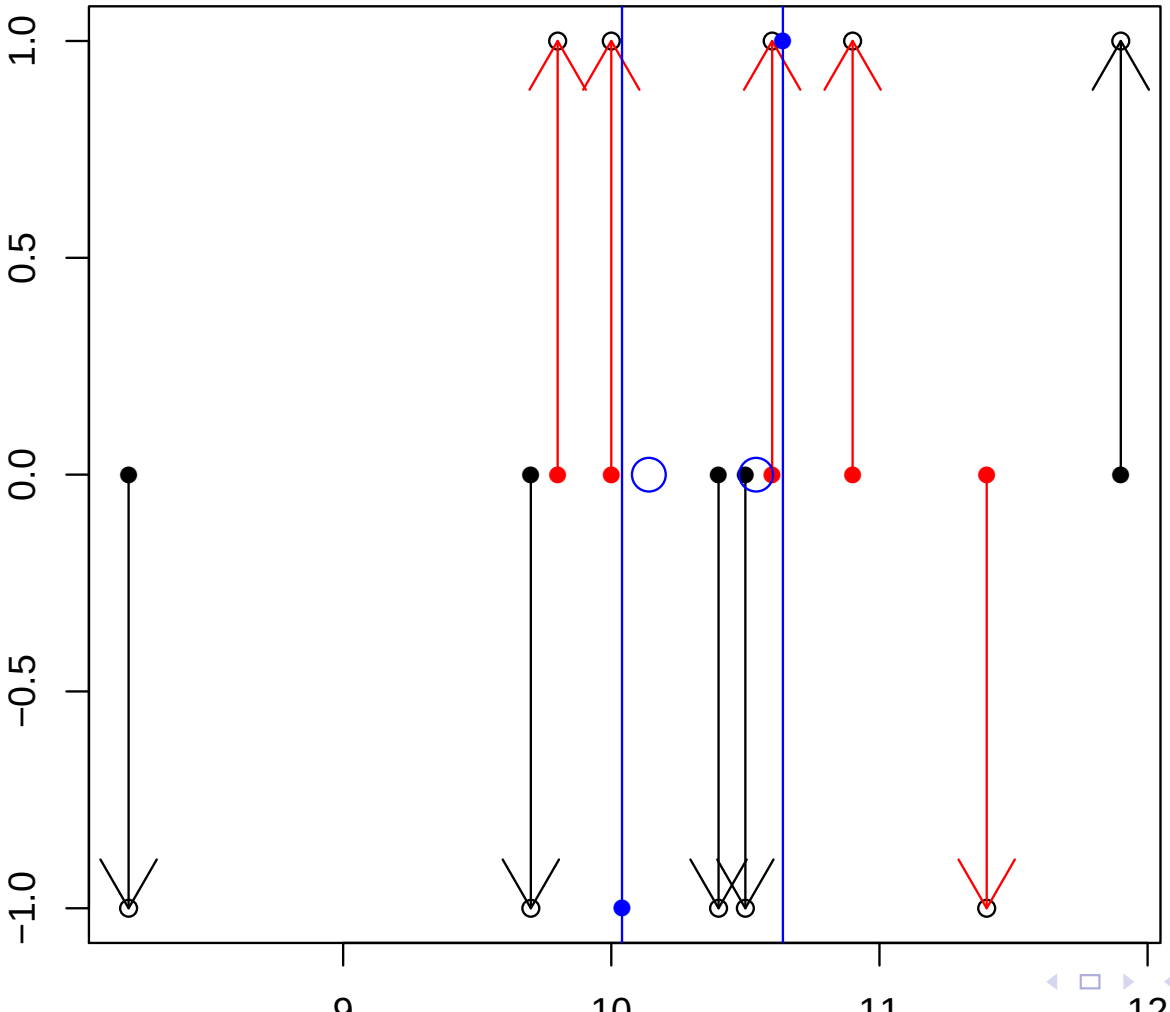
```
[1] 3 1 8 2 5
```

```
(smp2 <- (1:n)[-smp1])
```

```
[1]  4  6  7  9 10
```

Premestitev vrednosti

```
[1] 0.6
```



Kako je narisano

```
 plot(X, rep(0, n), col = ind, pch = 16, ylab = "")
 points(X[smp1], rep(-1, n1))
 points(X[smp2], rep(1, n2))
 arrows(X[smp1], rep(0, n1), X[smp1], rep(-1, n1),
+     col = ind[smp1])
 arrows(X[smp2], rep(0, n2), X[smp2], rep(1, n2),
+     col = ind[smp2])
 abline(v = mean(X[smp1]), col = 4)
 abline(v = mean(X[smp2]), col = 4)
 points(c(mean(X[smp1]), mean(X[smp2])), c(-1, 1),
+     pch = 16, col = 4)
 points(c(mean(X1), mean(X2)), c(0, 0), col = 4, cex = 2
 dx(X[smp1], X[smp2])
```

```
[1] 0.6
```

Funkcija

```
 perm.test <- function(x, X1, X2) {
+     n1 <- length(X1)
+     n2 <- length(X2)
+     ind <- c(rep(1, n1), rep(2, n2))
+     (smp1 <- sample(n, n1))
+     (smp2 <- (1:n)[-smp1])
+     plot(X, rep(0, n), col = ind, pch = 16, ylab = "")
+     points(X[smp1], rep(-1, n1))
+     points(X[smp2], rep(1, n2))
+     arrows(X[smp1], rep(0, n1), X[smp1], rep(-1,
+         n1), col = ind[smp1])
+     arrows(X[smp2], rep(0, n2), X[smp2], rep(1, n2),
+         col = ind[smp2])
+     abline(v = mean(X[smp1]), col = 4)
+     abline(v = mean(X[smp2]), col = 4)
+     points(c(mean(X[smp1]), mean(X[smp2])), c(-1,
+         1), pch = 16, col = 4)
+     points(c(mean(X1), mean(X2)), c(0, 0), col = 4,
+         cex = 2)
```

```
 perm.test(x, X1, X2)
[1] 0.68
 sapply(1:8, FUN = perm.test, X1 = X1, X2 = X2)
[1]  0.88  0.08  0.04  1.24  0.16  0.12  0.16 -0.04
 dx(X1, X2)
[1] 0.4
```