Διανύσματα, Πίνακες, και Γραμμικά Συστήματα στην R

Ορισμός Πίνακα

```
> A<-matrix(c(1,2,3,4,5,6),nrow=2,ncol=3)
> A
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> B<-matrix(c(1,2,3,4,5,6),nrow=2,ncol=3,byrow=TRUE)
> B
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
```

Ο ανάστροφος ενός πίνακα προκύπτει από την εντολή `t(A)`.

```
> t(A)
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
> t(B)
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
```

Το γινόμενο δύο πινάκων, A, C, των οποίων οι διαστάσεις είναι συμβατές προκύπτει ως `A %*% C`.

```
> C<-matrix(c(1,0,0,-1,-1,1), nrow=3, ncol=2)
> C
     [,1] [,2]
[1,]    1   -1
[2,]    0   -1
```

```
[3,]    0    1
> A%*%C
     [,1] [,2]
[1,]    1    1
[2,]    2    0
```

Το γινόμενο πίνακα και διανύσματος.

```
> b<-c(1,2,-3)
> b
[1]  1  2 -3
> A%*%b
     [,1]
[1,]   -8
[2,]   -8
```

Αν ο A είναι τετραγωνικός πίνακας $n \times n$ και b διάνυσμα με n συνιστώσες τότε `solve(A)` δίνει τον αντίστροφο A^{-1} (υπό την προϋπόθεση ότι υπάρχει) ενώ η `solve(A, b)` δίνει την (μοναδική) λύση του γραμμικού συστήματος $Ax = b$ δηλ το $x = A^{-1}b$. Η ορίζουσα του A δίδεται από την εντολή `det(A)`. Επίσης η εντολή `eigen(A)` δίνει τόσο τις ιδιοτιμές όσο και τα ιδιοδιανύσματα του πίνακα.

```
> C<-matrix(0,4,4)
> C
     [,1] [,2] [,3] [,4]
[1,]    0    0    0    0
[2,]    0    0    0    0
[3,]    0    0    0    0
[4,]    0    0    0    0
> a1<-c(1,2,3,4)
> for (i in 1:4) C[i,]<-a1^(i-1)
> C
     [,1] [,2] [,3] [,4]
[1,]    1    1    1    1
[2,]    1    2    3    4
[3,]    1    4    9   16
[4,]    1    8   27   64
> solve(C)
     [,1]      [,2] [,3]       [,4]
[1,]    4 -4.333333  1.5 -0.1666667
[2,]   -6  9.500000 -4.0  0.5000000
[3,]    4 -7.000000  3.5 -0.5000000
[4,]   -1  1.833333 -1.0  0.1666667
> det(C)
[1] 12
> eigen(C)
```

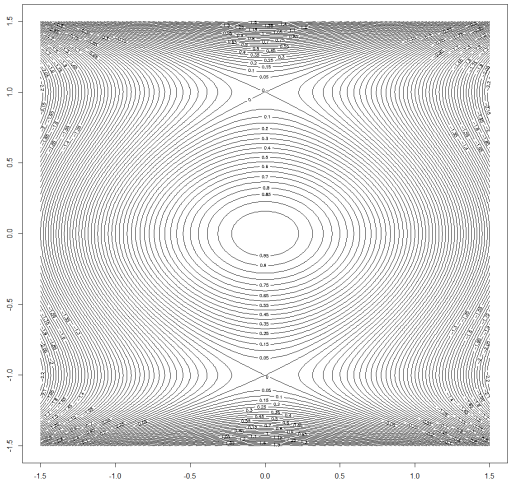
```
eigen() decomposition
$values
[1] 71.59873219  3.61988880  0.71678507  0.06459393

$vectors
            [,1]       [,2]       [,3]       [,4]
[1,] -0.01817783  0.3177346 -0.7101170  0.3533086
[2,] -0.06657990  0.5263751 -0.1985185 -0.7381734
[3,] -0.25130845  0.6892824  0.6336779  0.5553891
[4,] -0.96544329 -0.3832282 -0.2340437 -0.1477026
```

Ισοσταθμικές Καμπύλες

Ισοσταθμικές Καμπύλες της συνάρτησης $f(x,y) = -(x - y^2 + 1)(x + y^2 - 1)$.

```
f4<-function(x,y) {-(x-y^2+1)*(x+y^2-1)};
x <- seq(-1.5, 1.5, len=200)
y <- seq(-1.5, 1.5, len=200)
z <- outer(x, y, f4)
contour(x, y, z, nlevels=100)
```



Προβλήματα Γραμμικού Προγραμματισμού

Για την επίλυση προβλημάτων Γραμμικού Προγραμματισμού καθώς και προβλημάτων Μεταφοράς και Αντιστοίχισης πρέπει να εγκαταστήσουμε τα πακέτα `lpSolve, lpSolveAPI` (με την εντολή `install.packages(c("lpSolve", "lpSolveAPI"))` ή μέσω του μενού της R.
Στη συνέχεια πρέπει να εκτελέσουμε την εντολή `library(lpSolve); library(lPSolveAPI)`.

Για παράδειγμα, ας εξετάσουμε το πρόβλημα γραμμικού προγραμματισμού

$\max 2x_1 + 3x_2$

υπό τους περιορισμούς:

$x_1 + 2x_2 \leq 5$

$2x_1 + x_2 \leq 6$

$x_1 \geq 0, \quad x_2 \geq 0.$

```
f.obj <- c(2, 3)
f.con <- matrix (c(1, 2, 2, 1, 1, 0, 0, 1), nrow=4, byrow=TRUE)
f.dir <- c("<=", "<=", ">=", ">=")
f.rhs <- c(5, 6, 0, 0)

result<-lp("max", f.obj, f.con, f.dir, f.rhs)
result$solution
result$objval
#Output
> result$solution
[1] 2.333333 1.333333
> result$objval
[1] 8.666667


######################
# Cafeteria Problem  #
######################

objective.in <- c(100,80,75,85,90,95)
const.mat <- matrix(c(1,0,0,0,0,1,1,1,0,0,0,0,
                      0,1,1,0,0,0,0,0,1,1,0,0,
                      0,0,0,1,1,0,0,0,0,0,1,1),
                    nrow=6,byrow=TRUE)
const.rhs <- c(4,8,10,7,12,4)
const.dir <- c(">=",">=",">=",">=",">=",">=")
optimum <- lp(direction="min", objective.in, const.mat,
              const.dir, const.rhs)
optimum$solution
optimum$objval
#Output
> optimum$solution
[1]  0  8  2 12  0  4
> optimum$objval
[1] 2190

######################
# fitting y=axˆ2+bx+c #
######################

f.obj <- c(1, 0, 0, 0)
f.con <- matrix (c(-1, 1, 1, 1, -1, -1, -1, -1,
                   -1, 4, 2, 1, -1, -4, -2, -1,
                   -1, 9, 3, 1, -1, -9, -3, -1,
                   -1,16, 4, 1, -1,-16, -4, -1,
                   -1,25, 5, 1, -1,-25, -5, -1), nrow=10, byrow=TRUE)
f.dir <- c("<=", "<=", "<=", "<=","<=", "<=","<=", "<=","<=", "<=")
f.rhs <- c(9.94, -9.94, 18.55, -18.55, 33.02, -33.02, 48.19, -48.19, 71.54, -71.54)
```

```
optimum<- lp ("min", f.obj, f.con, f.dir, f.rhs)
optimum$solution
optimum$objval
#Output
> optimum$solution
[1] 0.96 2.17 1.90 6.83
> optimum$objval
[1] 0.96
```

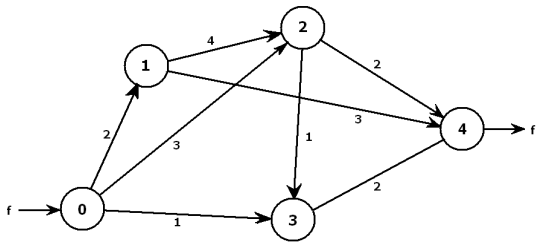
Παράδειγμα προβλήματος αντιστοίχησης.

Κολυμβητής	Ελεύθερο	Πρόσθιο	Πεταλούδα	Ύπτιο
Α	54	54	51	53
Β	51	57	52	52
Σ	50	53	54	56
Δ	56	54	55	53

```
# Assignment Problem 4x4 matrix
assign.costs <- matrix (c(54, 51, 50, 56, 54, 57, 53, 54, 51, 52, 54, 55, 53, 52, 56, 53),
                        4, 4)
optimum<-lp.assign (assign.costs)
optimum$solution
optimum$objval
#Output
> optimum$solution
     [,1] [,2] [,3] [,4]
[1,]    0    0    1    0
[2,]    0    0    0    1
[3,]    1    0    0    0
[4,]    0    1    0    0
> optimum$objval
[1] 207
```

Ένα πρόβλημα μέγιστης ροής δικτύου

```
f.obj <- c(1, 0, 0, 0, 0, 0, 0, 0, 0)
f.con <- matrix
    (c(1, -1, -1, -1,  0,  0,  0,  0,  0,
       0,  1,  0,  0, -1, -1,  0,  0,  0,
       0,  0,  1,  0,  1,  0, -1, -1,  0,
       0,  0,  0,  1,  0,  0,  1,  0, -1,
      -1,  0,  0,  0,  0,  1,  0,  1,  1,
       0,  1,  0,  0,  0,  0,  0,  0,  0,
       0,  0,  1,  0,  0,  0,  0,  0,  0,
       0,  0,  0,  1,  0,  0,  0,  0,  0,
       0,  0,  0,  0,  1,  0,  0,  0,  0,
       0,  0,  0,  0,  0,  1,  0,  0,  0,
       0,  0,  0,  0,  0,  0,  1,  0,  0,
       0,  0,  0,  0,  0,  0,  0,  1,  0,
       0,  0,  0,  0,  0,  0,  0,  0,  1),
                  nrow=13, byrow=TRUE)
f.dir <- c("=", "=", "=", "=", "=", "<=","<=", "<=",
           "<=", "<=","<=", "<=", "<=")
f.rhs <- c(0, 0, 0, 0, 0, 2, 3, 1, 4, 3, 1, 2, 2)
optimum<- lp ("max", f.obj, f.con, f.dir, f.rhs)
optimum$solution
optimum$objval
#Output
> optimum$solution
[1] 6 2 3 1 0 2 1 2 2
> optimum$objval
[1] 6
```




```
costs <- matrix(c(3,5,7,6,2,8,3,5,3,4,8,7), nrow=3, byrow=TRUE)
row.signs <- rep ("=", 3)
row.rhs <- c(200, 300, 350)
col.signs <- rep ("=", 4)
col.rhs <- c(175, 275, 260, 140)
#
# Run
#
trsp<-lp.transport(costs, "min", row.signs, row.rhs, col.signs, col.rhs)
trsp$solution
trsp$objval
#Output
> trsp$solution
     [,1] [,2] [,3] [,4]
[1,]   60    0    0  140
[2,]   40    0  260    0
[3,]   75  275    0    0
> trsp$objval
[1] 3205
```