Μία πολύ σύντομη εισαγωγή στην **R**

Σάμης Τρέβεζας

2016-17

Οι σημειώσεις αυτές αποτελούν μία προσαρμογή των σημειώσεων του Arthur Tenenhaus, INITIATION AU LOGICIEL R

Περιεχόμενα

1 Μία σύντομη παρουσίαση του λογισμικού R **3**
1.1 Χρήσιμες Πληροφορίες 3
1.2 Τα πακέτα 3

2 Οι βασικές έννοιες **5**
2.1 Δημιουργία βασικών αντικειμένων 5
2.2 Διαχείρηση του χώρου μνήμης 6
2.2.1 Πώς βλέπουμε όλα τα αντικείμενα που είναι καταχωρημένα στη μνήμη . 6
2.2.2 Πώς σβήνουμε όλα τα αντικείμενα που είναι καταχωρημένα στη μνήμη . 7
2.3 Οι επιλογές Βοήθειας 7
2.4 Τα δεδομένα στην R 8
2.4.1 Παραγωγή δεδομένων και αποθήκευση σε διαφορετικά αντικείμενα . . . 9
2.4.2 Μετατροπές τύπου στοιχείων 16
2.4.3 Μετατροπές Αντικειμένων 18
2.5 Πρόσβαση στα στοιχεία ενός αντικειμένου 18
2.5.1 Πρόσβαση με δείκτες 18
2.5.2 Πρόσβαση με το όνομα 24
2.5.3 Πρόσβαση με τη βοήθεια παραθύρου 25
2.6 Ανάγνωση δεδομένων απο αρχείο 25
2.7 Αποθήκευση δεδομένων σε αρχείο 26

3 Τα γραφήματα **27**
3.1 Διαχείρηση γραφικών παραθύρων 27
3.2 Διαμέριση ενός γραφήματος 28

Κεφάλαιο 1

Μία σύντομη παρουσίαση του λογισμικού R

1.1 Χρήσιμες Πληροφορίες

Το λογισμικό R είναι ένα λογισμικό που προσφέρεται ιδιαιτέρως για στατιστικές αναλύσεις. Διανέμεται δωρεάν απο το σύνδεσμο του *CRAN* (*Comprehensive R Archive Network*): `http://www.r-project.org`.

Δημιουργήθηκε από τον *Ross Ihaka* και τον *Robert Gentleman* και διανέμεται ελεύθερα για χρήση κάτω απο τους όρους της *GNU General Public Licence*. Η περαιτέρω ανάπτυξη και διανομή του, διασφαλίζονται απο πολλούς στατιστικούς και προγραμματιστές της ομάδας *R Development Core Team*. Σήμερα πλέον προσφέρονται πολυάριθμα στατιστικά πακέτα και συναρτήσεις που είναι χρήσιμα για διάφορα είδη στατιστικών αναλύσεων. Το λογισμικό R προτείνει επίσης ένα πολύ μεγάλο αριθμό γραφικών συναρτήσεων που βοηθάνε στην οπτικοποίηση και την εξερεύνηση των δεδομένων.

Η γλώσσα προγραμματισμού R είναι διερμηνευμένη γλώσσα και όχι μεταγλωττισμένη. Ως αποτέλεσμα, οι εντολές που πληκτρολογούνται εκτελούνται απ΄ ευθείας (σειριακά) χωρίς να είναι απαραίτητο να φτιαχτεί ένα πλήρες πρόγραμμα όπως συμβαίνει στις περισσότερες γλώσσες προγραμματισμού. Επιπλέον, η σύνταξη του κώδικα είναι αρκετά διαισθητική, και αυτό είναι ένα απο τα σημαντικά πλεονεκτήματά της. Οι μεταβλητές, τα δεδομένα, οι συναρτήσεις, τα αποτελέσματα, κτλ., αποθηκεύονται στη μνήμη *RAM* του υπολογιστή, στη μορφή αντικειμένων, που κάθε ένα έχει το δικό του όνομα. Σημειώνουμε ότι το όνομα κάθε αντικειμένου πρέπει να ξεκινάει απο ένα γράμμα (κεφαλαίο ή μικρό) και μπορεί να περιλαμβάνει γράμματα, αριθμούς και τελείες. Στη συνέχεια, ο χρήστης μπορεί να επιδράσει πάνω στα αντικείμενα μέσω κατάλληλων συναρτήσεων και πράξεων.

1.2 Τα πακέτα

Όλες οι συναρτήσεις αποθηκεύονται σε μία βιβλιοθήκη. Αυτή η βιβλιοθήκη περιέχει πακέτα συναρτήσεων. Το πακέτο με το όνομα `"base"` είναι η καρδιά της R και περιέχει τις βασικές συναρτήσεις αυτής της γλώσσας προγραμματισμού για την ανάγνωση και τον χειρισμό των δεδομένων, για τη δημιουργία ορισμένου τύπου γραφικών και για πραγματοποίηση κάποιων

βασικών στατιστικών αναλύσεων. Ένας μεγάλος αριθμός στατιστικών πακέτων περιλαμβάνεται ήδη στη βασική έκδοση της R και είναι διαθέσιμα μετά την εγκατάστασή της. Μπορούμε να έχουμε πρόσβαση σε αυτή τη λίστα των πακέτων με τη βοήθεια της εντολής `installed.packages()`, π.χ., ένα τμήμα της οθόνης μετά την εκτέλεση της εντολής δείχνει:

```
Package                    Libpath                                  Version   Priority
MASS       "MASS"          "C:/Program Files/R/R-3.0.2/library"  "7.2-8"  "recommended"
Base       "base"          "C:/Program Files/R/R-3.0.2/library"  "2.0.0"  "base"
Boot       "boot"          "C:/Program Files/R/R-3.0.2/library"  "1.2-19" "recommended"
translations "translations" "C:/Program Files/R/R-3.0.2/library"  "3.0.2"   NA
```

Τα πακέτα που εγκαθίστανται αυτόματα την πρώτη φορά που γίνεται η εγκατάσταση της R στον υπολογιστή δεν είναι όλα απ΄ ευθείας χρησιμοποιήσιμα. Κάποια απο αυτά είναι υποχρεωτικό να φορτωθούν πρώτα σε ένα προκαταρκτικό βήμα. Για να δεί κανείς για ποιά πακέτα έχει γίνει προεπιλογή φόρτωσης και για ποιά όχι, είναι αρκετό να δεί τη στήλη με το όνομα `Priority` (Προτεραιότητα) του αποτελέσματος της εντολής `installed.packages()`, όπως παραπάνω. Τα πακέτα που αντιστοιχούν σε `priority = «"base"»`, δεν χρειάζονται φόρτωση και μπορούν να χρησιμοποιηθούν απ΄ ευθείας. Αντίθετα, εκείνα που έχουν `priority = «"recommended"»` ή `«NA»` πρέπει να φορτωθούν στη μνήμη διαμέσου της συνάρτησης `library()`. Για παράδειγμα, ας υποθέσουμε ότι κάποιος θέλει να χρησιμοποιήσει τις συναρτήσεις του πακέτου, `MASS`. Εφ΄ όσον η προτεραιότητα είναι `recommended`, το πακέτο πρέπει πρώτα να φορτωθεί μέσω της εντολής `library(MASS)`. Με την εντολή `search()` μπορεί κανείς να δεί τα πακέτα που είναι φορτωμένα στη βιβλιοθήκη.

Οι χρήστες της R έχουν στην διάθεσή τους ένα πολύ μεγάλο αριθμό επιπλέον πακέτων, τα οποία έχουν φτιαχτεί απο άλλους χρήστες, και είναι διαθέσιμα στο δικτυακό τοπο της *CRAN* `http://cran.r-project.org/web/packages/available_packages_by_date.html`, ανάλογα με την ημερομηνία διάθεσης ή ακόμα και αλφαβητικά: `http://cran.r-project.org/web/packages/available_packages_by_name.html`. Για να εγκαταστήσει κάποιος ένα πακέτο με το όνομα `name` απο το δικτυακό τόπο της *CRAN* αρκεί να καλέσει τη συνάρτηση `install.packages("name")` στο παράθυρο εντολών. Σημειώνεται ότι η προτεραιότητα των πακέτων που έχουν εγκατασταθεί απο την *CRAN* είναι `NA`. Άρα για να χρησιμοποιηθούν οι συναρτήσεις των πακέτων αυτών πρέπει πρώτα να φορτωθεί το αντίστοιχο πακέτο.

Κεφάλαιο 2

Οι βασικές έννοιες

2.1 Δημιουργία βασικών αντικειμένων

Μπορούμε να αναθέσουμε απ΄ ευθείας μία τιμή σε ένα αντικείμενο, χωρίς αυτό να έχει δηλωθεί εκ των προτέρων. Για παράδειγμα:

```
> n=8
> n
[1] 8
```

Στην 1η γραμμή γίνεται ανάθεση της τιμής 8, στο αντικείμενο `n` χωρίς αυτό να έχει πρώτα δηλωθεί. Το σύμβολο `>` που εμφανίζεται αυτόματα στην αρχή της γραμμής, ειναι το σύμβολο ετοιμότητας και προηγείται κάθε εντολής που δίνουμε (δεν χρειάζεται να το πληκτρολογήσουμε, εμφανίζεται αυτόματα μετά την εκτέλεση κάθε εντολής με τη χρήση του πλήκτρου `Enter`). Πληκτρολογώντας το όνομα του αντικειμένου στη 2η γραμμή, τότε εμφανίζεται ως αποτέλεσμα η τιμή που αναθέσαμε. Το σύμβολο `[1]` που προηγείται, δείχνει ότι η εμφάνιση του αποτελέσματος αρχίζει απο το πρώτο στοιχείο του αντικειμένου `n`. Θα μπορούσαμε επίσης να είχαμε κάνει την ανάθεση ως εξής:

```
> n<- 8
> n
[1] 8
```

Παρ΄ όλα αυτά, στη συνέχεια θα χρησιμοποιήσουμε το σύμβολο `=`, εφόσον είναι πιο διαισθητικό.

Αν αναθέσουμε μία τιμή σε ένα υπάρχων αντικείμενο, τότε η προηγούμενη χάνεται.

```
> x=5.5
> x
[1] 5.5
> x=2.6
> x
[1] 2.6
```

Η ανάθεση μπορεί να είναι το αποτέλεσμα μιας πράξης.

```
> x=7 + 2
> x
[1] 9
> x=7
> y=2
> x= y+6
> x
[1] 8
```

Για να διακρίνουμε ξεχωριστές εντολές στην ίδια γραμμή, βάζουμε ;

```
> w = 8; name = "Giorgos"; frasi = "Kalimera";
> w; name; frasi
[1] 8
[1] "Giorgos"
[1] "Kalimera"
```

2.2 Διαχείριση του χώρου μνήμης

2.2.1 Πώς βλέπουμε όλα τα αντικείμενα που είναι καταχωρημένα στη μνήμη

Εντολή: ls()

```
> x = 5; X= 6; w=10 + .2; name = "Giorgos"; frasi = "Kalimera";
> x; X; w; name; frasi
[1] 5
[1] 6
[1] 10.2
[1] "Giorgos"
[1] "Kalimera"
```

Η συνάρτηση `ls()` μας επιτρέπει να εμφανίσουμε στην οθόνη **τη λίστα όλων των αντικειμένων που υπάρχουν στη μνήμη** (μόνο τα ονόματα των αντικειμένων εμφανίζονται)

```
> ls()
[1] "frasi" "name"  "w"     "x"     "X"
```

Σημειώνεται ότι η R διαφοροποιεί τα μικρά απο τα κεφαλαία γράμματα.

Αν κάποιος επιθυμεί να δεί τα αντικείμενα που **περιέχουν ένα δεδομένο χαρακτήρα** στο όνομά τους, μπορεί να χρησιμοποιήσει την επιλογή `pat` (συντόμευση του *pattern*)

```
> ls(pat = "m")
[1] "name"
```

Για αυτά που **ξεκινάνε απο ένα δεδομένο χαρακτήρα**:

```
> ls(pat = "^n")
[1] "name"
```

Για μία αναλυτική περιγραφή όλων των αντικειμένων που υπάρχουν στη μνήμη, πληκτρολογούμε: ls.str()

```
>  ls.str()
frasi :  chr "Kalimera"
name :  chr "Giorgos"
w :  num 10.2
x :  num 5
X :  num 6
```

2.2.2 Πώς σβήνουμε όλα τα αντικείμενα που είναι καταχωρημένα στη μνήμη

Εντολή: rm()
Η συνάρτηση `rm()` μας επιτρέπει να **σβήσουμε όλα τα αντικείμενα που υπάρχουν στη μνήμη**.

```
> ls()
[1] "frasi" "name"  "w"     "x"     "X"
```

Για να σβήσουμε το αντικείμενο με το όνομα x, πληκτρολογούμε:

```
> rm(x);ls()
[1] "frasi" "name"  "w"     "X"
```

Για να σβήσουμε όλα τα αντικείμενα απο τη μνήμη, πληκτρολογούμε:

```
> rm(list = ls()); ls()
character(0)
```

2.3 Οι επιλογές Βοήθειας

Εντολή: ? ή help
Η βοήθεια στο *Internet* είναι πολύ χρήσιμη όταν γράφουμε προγράμματα στην R. Για παράδειγμα, ας υποθέσουμε ότι θέλουμε να κάνουμε μία γραμμική παλινδρόμηση. Δύο περιπτώσεις παρουσιάζονται:

1. Δεν γνωρίζουμε το όνομα της συνάρτησης που πρέπει να καλέσουμε, και επομένως δεν ξέρουμε να την χρησιμοποιούμε.

2. Γνωρίζουμε το όνομα της συνάρτησης, όμως δεν ξέρουμε να την χρησιμοποιούμε.

Στην πρώτη περίπτωση μπορούμε να ζητήσουμε βοήθεια μέσω της εντολής `help.search("regression")`. Αυτή η εντολή δίνει μία λίστα συναρτήσεων που σχετίζονται με τη λέξη *regression* . Μέσα απο αυτή τη λίστα μπορούμε να ανακαλύψουμε ποιά συνάρτηση έχει σχεδιαστεί γι΄ αυτό το σκοπό. Έτσι μπορούμε να δούμε ότι η συνάρτηση `lm()` (lm: linear model) πραγματοποεί μία γραμμική παλινδρόμηση.

Στην δεύτερη περίπτωση, εφ΄ οσον γνωρίζουμε το όνομα της συνάρτησης, μπορούμε να δούμε πώς χρησιμοποιείται με τη βοήθεια των ισοδύναμων εντολών `?lm` ή `help(lm)`. Με τις εντολές αυτές παίρνουμε λεπτομερείς πληροφορίες για τη συνάρτηση αυτή:

- Όνομα του πακέτου που προέρχεται η συνάρτηση.
- Description: Σύντομη περιγραφή της συνάρτησης.
- Usage: το όνομα της συνάρτησης με όλα τα ορίσματά της και προεπιλεγμένες τιμές, εφ΄ όσον υπάρχουν.
- Arguments: Αναλυτική περιγραφή των ορισμάτων της συνάρτησης.
- Details: Λεπτομέρειες και πληροφορίες για τη συνάρτηση.
- Value: Ο τύπος του αποτελέσματος που επιστρέφεται απο τη συνάρτηση.
- See also: Επιπλέον σύνδεσμοι για βοήθεια που μπορούμε να έχουμε.
- Examples: Παραδείγματα χρήσης της συνάρτησης.

Κι΄ άλλες παράγραφοι μπορεί να είναι διαθέσιμες, όπως Note, References ή Author(s). Σημειώνεται επίσης ότι μπορούμε να κάνουμε χρήση της βοήθειας σε περιβάλλον **html** με την εντολή **help.start()**.

Σημαντική παρατήρηση: Όταν χρησιμοποιούμε τη συνάρτηση `help.search()`, τότε η έρευνα πραγματοποιείται σε όλα τα πακέτα που ειναι διαθέσιμα στον υπολογιστή. Όμως, όταν χρησιμοποιούμε τις συναρτήσεις `?` ή `help()` τότε η έρευνα περιορίζεται μόνο στα πακέτα που είναι φορτωμένα στη βιβλιοθήκη.

2.4 Τα δεδομένα στην R

Οι μεταβλητές, τα δεδομένα, οι συναρτήσεις και τα αποτελέσματα των αναλύσεων αποθηκεύονται σε αντικείμενα. Σε αυτήν την παράγραφο θα περιγράψουμε τη δομή των δεδομένων. Όλα τα αντικείμενα έχουν δυο χαρακτηριστικές ιδιότητες: τον **τύπο** (**mode**) και το **μήκος** (**length**).

Ο τύπος του αντικειμένου αναφέρεται στον τύπο/είδος των στοιχείων του αντικειμένου. Υπάρχουν 4 βασικοί τύποι: αριθμητικός, χαρακτήρας, λογικός και μιγαδικός (**numeric, character, logical, complex**). Για να δούμε τον τύπο ενός αντικειμένου χρησιμοποιούμε τη συνάρτηση `mode()`.

```
> x=2.3; name= 'test'
> mode(x); mode(name)
[1] "numeric"
[1] "character"
```

Το μήκος του αντικειμένου αναφέρεται στο πλήθος των στοιχείων του αντικειμένου (αν το αντιληφθούμε ως διάνυσμα στοιχείων). Για να δούμε το μήκος ενός αντικειμένου χρησιμοποιούμε τη συνάρτηση `length()`

```
 > length(x); length(name)
[1] 1
[1] 1
```

Οι βασικές δομές δεδομένων που θα αναλυθούν σε αυτήν την πολύ σύντομη εισαγωγή είναι οι εξής: διανύσμα, παράγοντας, πίνακας, πλαίσιο (δεδομένων) και λίστα (**vector, factor, matrix, data.frame, list**).

Σημειώνουμε επίσης ότι **τα χαμένα ή μη διαθέσιμα δεδομένα** αναπαριστώνται στην R με `NA` (*Not Avalaible*).

Table 2.1: Ο παρακάτω πίνακας συνοψίζει τους τύπους δεδομένων που αναφέραμε και τις χαρακτηριστικές ιδιότητες αντικειμένων του κάθε τύπου (num=numerical, char=character, log=logical, comp=complex)

Αντικείμενα	Ορολογία	Τύποι	Στοιχεία με διαφορετικούς τύπους
Διάνυσμα	vector	num, char, log, comp	ΟΧΙ
Παράγοντας	factor	num, char	ΟΧΙ
Πίνακας	matrix	num, char, log, comp	ΟΧΙ
Πλαίσιο	data.frame	num, char, log, comp	ΝΑΙ
Λίστα	list	num, char, log, comp	ΝΑΙ

Στη λίστα υπάρχει η δυνατότητα αποθήκευσης και άλλων τύπων δεδομένων όπως οι συναρτήσεις και οι εκφράσεις (functions and expressions). Αυτά θα εξηγηθούν παρακάτω.

2.4.1 Παραγωγή δεδομένων και αποθήκευση σε διαφορετικά αντικείμενα

Τα διανύσματα: vector()

Η συνάρτηση `vector()` έχει 2 ορίσματα, τον τύπο στοιχείων που συνθέτουν το διάνυσμα και το μήκος του διανύσματος.

```
a = vector("numeric", 5)
b = vector("character", 5)
c = vector("logical", 5)
d = vector("complex", 5)
```

Ισοδυναμες εντολές με τις παραπάνω είναι οι εξής:

```
a = numeric(5)
b = character(5)
c = logical(5)
d = complex(5)
```

Το αποτέλεσμα και στις 2 περιπτώσεις είναι το εξής:

```
a;b;c;d
[1] 0 0 0 0 0
[1] "" "" "" "" ""
[1] FALSE FALSE FALSE FALSE FALSE
[1] 0+0i 0+0i 0+0i 0+0i 0+0i
```

Μπορούμε επίσης να φτιάξουμε άμεσα ένα διάνυσμα με τη συνάρτηση c()

```
> x=c(1.1, 2.2, 3.3)
> x
[1] 1.1 2.2 3.3
```

Οι παράγοντες: factor()

Η συνάρτηση `factor()` φτιάχνει κατηγορικές ονομαστικές μεταβλητές.

```
> factor(1:3)
[1] 1 2 3
Levels: 1 2 3
> factor(1:3, levels=1:5)
[1] 1 2 3
Levels: 1 2 3 4 5
> factor(1:3, levels=1:5, labels = c("A", "B", "C", "D", "E"))
[1] A B C
Levels: A B C D E
```

Η επιλογή `levels` καθορίζει το πλήθος των διαφορετικών επιπέδων για την κατηγορική μεταβλητή (εξ΄ ορισμού το πλήθος των διακεκριμένων τιμών της μεταβλητής).

Η επιλογή `labels` καθορίζει το όνομα των επιπέδων.

Οι πίνακες: matrix()

Ο πίνακας είναι ένα διάνυσμα με ένα επιπλέον όρισμα που καθορίζει τις διαστάσεις του πίνακα.

```
> matrix(0,3,4)
     [,1] [,2] [,3] [,4]
[1,]    0    0    0    0
[2,]    0    0    0    0
[3,]    0    0    0    0
>
```

```
> x=1:12
> mat1=matrix(x,4,3)
> mat1
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12
> mat2 = matrix(x, 4, 3, byrow = TRUE)
> mat2
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
[4,]   10   11   12
```

Οι επιλογές της συνάρτησης `matrix()` είναι χρήσιμες. Για παράδειγμα μπορούμε να δώσουμε ονόματα στις γραμμές ή στις στήλες του πίνακα με τη βοήθεια συναρτήσεων όπως `rownames(), colnames()` ή `dimnames()`.

```
> rownames(mat1)=c('r1','r2','r3','r4')
> colnames(mat1)=c('c1','c2','c3')
> mat1
   c1 c2 c3
r1  1  5  9
r2  2  6 10
r3  3  7 11
r4  4  8 12
```

Το ίδιο αποτέλεσμα θα έχουμε και για τον 2ο πίνακα ως εξής:

```
> dimnames(mat1)
[[1]]
[1] "r1" "r2" "r3" "r4"

[[2]]
[1] "c1" "c2" "c3"

> mat2
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
[4,]   10   11   12
> dimnames(mat2)
NULL
```

```
> dimnames(mat2)=list(rownames(mat1),colnames(mat1))
> dimnames(mat2)
[[1]]
[1] "r1" "r2" "r3" "r4"

[[2]]
[1] "c1" "c2" "c3"
```

ή ακόμα και ως εξής:

```
> dimnames(mat2)=NULL
> dimnames(mat2)
NULL

> dimnames(mat2)=dimnames(mat1)
> dimnames(mat2)
[[1]]
[1] "r1" "r2" "r3" "r4"

[[2]]
[1] "c1" "c2" "c3"
```

Όλα αυτά μπορούν να γίνουν πιο γρήγορα με τη βοήθεια της εντολής `paste()`

```
> nom_ind = paste("I", 1:4, sep = "")
> nom_ind
[1] "I1" "I2" "I3" "I4"
> nom_var = paste("V", 1:3, sep = "")
> nom_var
[1] "V1" "V2" "V3"

> dimnames(mat2) = list(nom_ind, nom_var)
> mat2
   V1 V2 V3
I1  1  2  3
I2  4  5  6
I3  7  8  9
I4 10 11 12
```

Πράξεις με Διανύσματα και Πίνακες

Η R προσφέρει αρκετές δυνατότητες χειρισμού και πράξεων με πίνακες και διανύσματα. Εδώ περιγράφουμε κάποιες περιπτώσεις που εμφανίζονται συχνά.

```
> mat1 = matrix(1 : 4, nrow = 2, ncol = 2)
> mat1
     [,1] [,2]
```

```
[1,]    1    3
[2,]    2    4
> mat2 = matrix(5 : 8, nr = 2, nc = 2)
> mat2
     [,1] [,2]
[1,]    5    7
[2,]    6    8
```

Η συνάρτησηrbind() παραθέτει τον δεύτερο πίνακα κάτω απο τον πρώτο, επομένως διατηρώντας τις γραμμές και των δύο πινάκων.

```
> rbind(mat1, mat2)
     [,1] [,2]
[1,]    1    3
[2,]    2    4
[3,]    5    7
[4,]    6    8
```

Η συνάρτηση cbind() παραθέτει τον δεύτερο πίνακα δεξιά απο τον πρώτο, επομένως διατηρώντας τις στήλες και των δύο πινάκων.

```
> cbind(mat1, mat2)
     [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8
```

Ο πολλαπλασιασμός πινάκων γίνεται μέσω του τελεστή %*% ενώ για τον πολλαπλασιασμό στοιχείο με στοιχείο (γινόμενο Hadamard) χρησιμοποιείται το σύμβολο $*$. Σημειώνεται ότι ο πολλαπλασιασμός ενός πίνακα με ένα βαθμωτό λειτουργεί με τον ίδιο τρόπο όπως το (γινόμενο Hadamard).

```
> diag(2)
     [,1] [,2]
[1,]    1    0
[2,]    0    1
> D=2*diag(2)
> D
     [,1] [,2]
[1,]    2    0
[2,]    0    2
> mat1%*% D
     [,1] [,2]
[1,]    2    6
[2,]    4    8
> mat1* D
     [,1] [,2]
[1,]    2    0
[2,]    0    8
```

Η πρόσθεση και η αφαίρεση πινάκων γίνεται με τα συνήθη σύμβολα + και -.

```
> mat1 + D
     [,1] [,2]
[1,]    3    3
[2,]    2    6
> mat1 - D
     [,1] [,2]
[1,]   -1    3
[2,]    2    2
```

Για να πάρουμε τον ανάστροφο (*transpose*) ενός πίνακα χρησιμοποιούμε τη συνάρτηση `t()`

```
> t(mat1 - D)
     [,1] [,2]
[1,]   -1    2
[2,]    3    2
```

Η συνάρτηση `diag()` είναι αρκετά χρήσιμη. Είδαμε παραπάνω ότι με τη χρήση ενός μόνο ορίσματος (παραμέτρου), μπορούμε να κατασκευάσουμε ταυτοτικούς πίνακες με διάσταση που υποδεικνύεται απο την τιμή της παραμέτρου. Άλλες χρήσεις της συνάρτησης φαίνονται παρακάτω.

```
> A=diag(c(1,2,3,4))
> A
     [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
[2,]    0    2    0    0
[3,]    0    0    3    0
[4,]    0    0    0    4
> diag(A)
[1] 1 2 3 4
```

Η R προτείνει επίσης και αρκετές ειδικές συναρτήσεις για λογισμό πινάκων, όπως η `solve()` και η `eigen()` για την εύρεση του αντιστρόφου ενός πίνακα και των ιδιοτιμών και ιδιοδιανυσμάτων αντίστοιχα.

```
> solve(A)
     [,1] [,2]      [,3] [,4]
[1,]    1  0.0 0.0000000 0.00
[2,]    0  0.5 0.0000000 0.00
[3,]    0  0.0 0.3333333 0.00
[4,]    0  0.0 0.0000000 0.25
> eigen(A)
$values
[1] 4 3 2 1

$vectors
```

```
     [,1] [,2] [,3] [,4]
[1,]    0    0    0    1
[2,]    0    0    1    0
[3,]    0    1    0    0
[4,]    1    0    0    0
```

Τα πλαίσια δεδομένων: data.frame()

Η συνάρτηση `data.frame()` μοιάζει με αυτήν που κατασκευάζει πίνακες, αλλά με τη βασική διαφορά ότι οι στήλες μπορεί να είναι διαφορετικού τύπου (numeric, character, logical, complex). Έτσι μπορούμε να αντιληφθούμε τα πλαίσια δεδομένων ως συλλογές διανυμάτων, ενδεχομένως διαφορετικού τυπου, τοποθετημένα σε στήλες. Αυτή η δομή δεδομένων είναι μία απο τις πιο συχνά χρησιμοποιούμενες στην R.

```
> a = c(1, 2, 3); a; mode(a);
[1] 1 2 3
[1] "numeric"
> b = c("a", "b", "c"); b; mode(b)
[1] "a" "b" "c"
[1] "character"
> df = data.frame(a, b)
> df
  a b
1 1 a
2 2 b
3 3 c
```

Ένας άλλος τρόπος να φτιάξουμε ένα data.frame δίνεται παρακάτω.

```
> df2 = data.frame(a = 1:4, b = letters[1:4])
> df2
  a b
1 1 a
2 2 b
3 3 c
4 4 d
```

Τα ονόματα των γραμμών και των στηλών ειναι παρόμοια με τους πίνακες, αν και στο data.frame υπάρχει προεπιλογή οι γραμμές να παίρνουν το όνομα του αριθμού που αντιστοιχεί στη φυσική τους διάταξη.

```
> dimnames(df2)
[[1]]
[1] "1" "2" "3" "4"

[[2]]
[1] "a" "b"
```

Οι λίστες: list()

Μία λίστα δημιουργείται με τον ίδιο τρόπο όπως και το `data.frame()` και τα στοιχεία που συνθέτουν τη λίστα μπορούν επίσης να είναι διαφορετικού τύπου.Επιπλέον, σε αντίθεση με τα στοιχεία του data.frame τα στοιχεία μιας λίστας μπορούν να έχουν και διαφορετικό μήκος.

```
> a = c(1, 2, 3, 4, 5)
> b = c("a", "b", "c")
> list1 = list(a, b)
> list1
[[1]]
[1] 1 2 3 4 5

[[2]]
[1] "a" "b" "c"
```

Μπορούμε να δώσουμε όνομα στα στοιχεία μιας λίστας όπως φαίνεται παρακάτω.

```
> names(list1) = c("L1", "L2"); list1
$L1
[1] 1 2 3 4 5

$L2
[1] "a" "b" "c"

> list2 = list(L1 = a, L2 = b)
> list2
$L1
[1] 1 2 3 4 5

$L2
[1] "a" "b" "c"
```

2.4.2 Μετατροπές τύπου στοιχείων

Σε πολλές περιπτώσεις είναι χρήσιμο να γίνεται μετατροπή του τύπου των στοιχείων ενός αντικειμένου. Μία τέτοια μετατροπή είναι δυνατή με τη βοήθεια της συνάρτησης `as.mode` (as.numeric, as.logical, as.character,...).

Παρακάτω δινουμε μερικά παραδείγματα:

- Μετατροπή **logical** → **numeric**

```
> logiko = c(FALSE, FALSE, TRUE, TRUE)
> logiko
[1] FALSE FALSE  TRUE  TRUE
```

Table 2.2: Ο παρακάτω πίνακας συνοψίζει τις δυνατές μετατροπές

Μετατροπή σε	Συνάρτηση	Κανόνες
numeric	as.numeric	FALSE→ 0, TRUE → 1, '1' → 1, '2' → 2, 'A' → NA
character	as.character	$1, 2, \ldots \rightarrow$ '1', '2',...,FALSE → 'FALSE', TRUE → 'TRUE'
logical	as.logical	'FALSE' → FALSE, 'TRUE' → TRUE, 0 → FALSE, $x \rightarrow$ TRUE ($x \neq 0$)

```
> as.numeric(logiko)
[1] 0 0 1 1
```

- Μετατροπή **character → numeric**

```
> char = c("1", "2", "3", "A", "/", "T", "%", "-")
> as.numeric(char)
[1]  1  2  3 NA NA NA NA NA
Warning message:
NAs introduced by coercion
```

- Μετατροπή **numeric → logical**

```
> num = 0:5
> as.logical(num)
[1] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
```

- Μετατροπή **character → logical**

```
> char = c("FALSE", "TRUE", "F", "T", "false", "true", "f", "A", "(")
> as.logical(char)
[1] FALSE  TRUE FALSE  TRUE FALSE  TRUE    NA    NA    NA
```

- Μετατροπή **numeric → character**

```
> num = 0:5
> as.character(num)
[1] "0" "1" "2" "3" "4" "5"
```

- Μετατροπή **logical → character**

```
> logiko = c(F, FALSE, T, TRUE)
> as.character(logiko)
[1] "FALSE" "FALSE" "TRUE"  "TRUE"
```

2.4.3 Μετατροπές Αντικειμένων

Σε πολλές περιπτώσεις είναι χρήσιμο να γίνεται μετατροπή του τύπου ενός αντικειμένου. Μία τέτοια μετατροπή είναι δυνατή με τη βοήθεια συναρτήσεων της μορφής (as.matrix, as.data.frame, as.list, as.factor, ...)

Παρακάτω δινουμε μερικά παραδείγματα:

- Μετατροπή **matrix** → **data.frame**

```
> A=matrix(1:12,nr=3); A
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> B=as.data.frame(A); B
  V1 V2 V3 V4
1  1  4  7 10
2  2  5  8 11
3  3  6  9 12
```

- Μετατροπή **factor** → **vector** ή **numeric**

```
> paragontas = factor(c(1, 5, 10));paragontas
[1] 1  5  10
Levels: 1 5 10
> as.vector(paragontas)
[1] "1"  "5"  "10"
> as.numeric(paragontas)
[1] 1 2 3
```

- Μετατροπή **factor** → **numeric** (αλλά διατηρώντας τις τιμές)

```
> as.numeric(as.vector(paragontas))
[1]  1  5 10
```

ή

```
> as.numeric(as.character(paragontas))
[1]  1  5 10
```

2.5 Πρόσβαση στα στοιχεία ενός αντικειμένου

2.5.1 Πρόσβαση με δείκτες

Τα διανύσματα και οι πίνακες

Για να έχουμε πρόσβαση σε ένα στοιχείο ενός διανύσματος σε καθορισμένη θέση:

```
> v=-2:5; v
[1] -2 -1  0  1  2  3  4  5
> v[2]; v[5]
[1] -1
[1] 2
```

Για να έχουμε πρόσβαση σε περισσότερα στοιχεία ενός διανύσματος σε καθορισμένη θέση:

```
> v[1:3]; v[c(2,5,7)]
[1] -2 -1  0
[1] -1  2  4
```

Για να έχουμε πρόσβαση σε εκείνα τα στοιχεία ενός διανύσματος που ικανοποιούν μία συσκεκριμένη συνθήκη:

```
> v>3
[1] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE
> v[v>3]
[1] 4 5
```

Για να δώσουμε μία τιμή στα στοιχεία εκείνα του διανύσματος που ικανοποιούν μία συγκεκριμένη συνθήκη:

```
> v=-5:5
> v[v<=0]=0; v
 [1] 0 0 0 0 0 0 1 2 3 4 5
> v[v==0]=5; v
 [1] 5 5 5 5 5 5 1 2 3 4 5
```

Για να έχουμε πρόσβαση σε ένα στοιχείο ενός πίνακα σε καθορισμένη θέση:

```
> A[2,3]; A[1,4]
[1] 8
[1] 10
```

Ένας πίνακας συμπεριφέρεται και ως διάνυσμα με τα στοιχεία του διατεταγμένα κατα στήλες:

```
> A[1:6]
[1] 1 2 3 4 5 6
```

Μπορούμε να έχουμε πρόσβαση στις γραμμές και τις στήλες ενός πίνακα, ως εξής:

```
> A[1,]
[1]  1  4  7 10
> A[,3]
[1] 7 8 9
```

Μπορούμε να αλλάξουμε τις τιμές μιας γραμμής, ή μιας στήλης ενός πίνακα, ως εξής:

```
> A[1,]=0; A
     [,1] [,2] [,3] [,4]
[1,]    0    0    0    0
[2,]    2    5    8   11
[3,]    3    6    9   12
> A[,4]=c(-1,-2,-3); A
     [,1] [,2] [,3] [,4]
[1,]    0    0    0   -1
[2,]    2    5    8   -2
[3,]    3    6    9   -3
```

Μπορεί να γίνει διαγραφή μιας γραμμής, ή μιας στήλης ενός πίνακα, ως εξής:

```
> A=A[-3,]; A
     [,1] [,2] [,3] [,4]
[1,]    0    0    0   -1
[2,]    2    5    8   -2
> A=A[,-4]; A
     [,1] [,2] [,3]
[1,]    0    0    0
[2,]    2    5    8
```

Όπως και για τα διανύσματα, για να δώσουμε μία τιμή στα στοιχεία εκείνα του πίνακα που ικανοποιούν μία συγκεκριμένη συνθήκη:

```
> A[A==0]=1; A
     [,1] [,2] [,3]
[1,]    1    1    1
[2,]    2    5    8
```

Προσέξτε τα αποτελέσματα που παίρνουμε με την επιλογή στοιχείων του πίνακα μέσω λογικών εκφράσεων:

```
> c(F,T)
[1] FALSE  TRUE
> A
     [,1] [,2] [,3]
[1,]    1    1    1
[2,]    2    5    8
> A[c(F,T,F,T,F,T)]
[1] 2 5 8
> A[c(F,T)]
[1] 2 5 8
> A[c(T,F)]
[1] 1 1 1
> A[c(T,F,T)]
[1] 1 1 5 8
```

```
> A[T]
[1] 1 2 1 5 1 8
> A[F]
numeric(0)
```

Τα πλαίσια δεδομένων

Οι δείκτες χρησιμοποιούνται επίσης και στα πλαίσια δεδομένων, αλλά με την επιπρόσθετη δυσκολία ότι οι στήλες στα πλαίσια δεδομένων μπορεί να είναι διαφορετκού τύπου στοιχείων.

```
> a = c(1, 2, 3)
> b = c("a", "b", "c")
> df = data.frame(a, b); df
  a b
1 1 a
2 2 b
3 3 c
```

Για να έχουμε πρόσβαση σε ένα συγκεκριμένο στοιχείο, πληκτρολογούμε:

```
> df[2,1]; df[3,2]
[1] 2
[1] c
Levels: a b c
```

Παρατηρούμε ότι τα στοιχεία της 2ης στήλης που αρχικά είχαν οριστεί ως χαρακτήρες, στα πλαίσια δεδομένων μετατρέπονται σε παράγοντες. Θα παίρναμε το ίδιο αποτέλεσμα αν πληκτρολογούσαμε το εξής:

```
> df[1][2,1]; df[2][3,1]
[1] 2
[1] c
Levels: a b c
```

Αυτό συμβαίνει διότι με την εντολή `df[i]` παίρνουμε τα στοιχεία της i-στήλης σαν ένα πλαίσιο δεδομένων. Πράγματι,

```
> df[1]; df[2]
  a
1 1
2 2
3 3
  b
1 a
2 b
3 c
```

Για να έχουμε πρόσβαση στα αρχικά αντικείμενα που αντιστοιχούν στις στήλες του πλαισίου δεδομένων, τότε πληκτρολογούμε:

```
> df[,1]; df[,2]
[1] 1 2 3
[1] a b c
Levels: a b c
```

Παρατηρούμε λοιπόν ότι εδώ τα στοιχεία παρουσιάζονται ως ένα αριθμητικό διάνυσμα και ένας παράγοντας αντίστοιχα.

Ας πάρουμε τώρα ένα άλλο παράδειγμα σχηματισμού ενός πλαισίου δεδομένων που αποτελείται απο ένα αριθμητικό διάνυσμα και έναν πίνακα.

```
> df=data.frame(a=20:23,b=matrix(1:16,nr=4)); df
   a b.1 b.2 b.3 b.4
1 20   1   5   9  13
2 21   2   6  10  14
3 22   3   7  11  15
4 23   4   8  12  16
```

Για να έχουμε πρόσβαση στον πίνακα (ως ένα καινούριο πλαίσιο δεδομένων), πληκτρολογούμε:

```
> df[2:5]
  b.1 b.2 b.3 b.4
1   1   5   9  13
2   2   6  10  14
3   3   7  11  15
4   4   8  12  16
```

Αν πληκτρολογήσουμε:

```
> df[,2:5]
  b.1 b.2 b.3 b.4
1   1   5   9  13
2   2   6  10  14
3   3   7  11  15
4   4   8  12  16
```

τότε είναι φανερό ότι το αποτέλεσμα είναι το ίδιο και μπορούμε να επιβεβαιώσουμε ότι δεν αναγνωρίζεται ως πίνακας, αλλά πάλι ως πλαίσιο δεδομένων, με την εντολή:

```
> is.matrix(df[,2:5])
[1] FALSE
> is.data.frame(df[,2:5])
[1] TRUE
```

Ο τρόπος ανάθεσης και αλλαγής τιμών στοιχείων του πλαισίου δεδομένων είναι παρόμοιος με αυτόν στους πίνακες. Για να αλλάξουμε, για παράδειγμα, τις τιμές της 3ης γραμμής του πίνακα, πληκτρολογούμε:

```
> df[3,2:5]=0; df
  a b.1 b.2 b.3 b.4
1 20   1   5   9  13
2 21   2   6  10  14
3 22   0   0   0   0
4 23   4   8  12  16
```

Οι λίστες

Για τις λίστες, αν χρησιμοποιήσουμε απλές αγκύλες

```
> lista=list(a=1:4,b=matrix(1:16,nr=4)); lista
$a
[1] 1 2 3 4

$b
     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    2    6   10   14
[3,]    3    7   11   15
[4,]    4    8   12   16

> lista[1]; lista[2]
$a
[1] 1 2 3 4

$b
     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    2    6   10   14
[3,]    3    7   11   15
[4,]    4    8   12   16
```

τότε παίρνουμε τα επιμέρους αντικείμενα που απαρτίζουν την αρχική λίστα ως ξεχωριστές λίστες. Αν όμως βάλουμε διπλές αγκύλες, τότε έχουμε πρόσβαση στα αντικείμενα της αρχικής λίστας, διατηρώντας την αρχική δομή τους.

```
> lista[[1]]; lista[[2]]
[1] 1 2 3 4
     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    2    6   10   14
[3,]    3    7   11   15
[4,]    4    8   12   16
```

Άρα στην περίπτωση της λίστας μπορούμε να διατηρούμε τον πίνακα ως ξεχωριστό αντικείμενο, σε αντίθεση με τα πλαίσια δεδομένων. Παρ΄ όλα αυτά, για να πάρουμε τις τιμές της 3ης γραμμής του πίνακα, πληκτρολογούμε:

```
>  lista[[2]][3,]
[1]  3  7 11 15
```

ενώ για παράδειγμα οι εντολές:

```
> lista[3,2:5]
Error in lista[3, 2:5] : incorrect number of dimensions
> lista[2][3,]
Error in lista[2][3, ] : incorrect number of dimensions
```

δεν είναι σωστές όταν εφαρμόζονται σε λίστες, σε αντίθεση με τα πλαίσια δεδομένων.

2.5.2 Πρόσβαση με το όνομα

Προσβαση στα στοιχεία ενός αντικειμένου με το όνομα μπορούμε να έχουμε μόνο στην περιπτωση των πλαισίων δεδομένων και των πινάκων Αυτό γινεται εφικτό με τη χρήση του συμβόλου $

```
> lista = list(a = 1:4, b = matrix(1:16, 4)); lista
$a
[1] 1 2 3 4

$b
     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    2    6   10   14
[3,]    3    7   11   15
[4,]    4    8   12   16

> lista$b
     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    2    6   10   14
[3,]    3    7   11   15
[4,]    4    8   12   16
> lista$b[1,]
[1]  1  5  9 13
> lista$a[3]
[1] 3

> df = data.frame(a = 1:6, b = letters[1:3])
> df$a
[1] 1 2 3 4 5 6
> df$b
[1] a b c a b c
Levels: a b c
```

2.5.3 Πρόσβαση με τη βοήθεια παραθύρου

Πρόσβαση στα στοιχεία ενός αντικειμένου μπορούμε να έχουμε και με τη βοήθεια ενός παραθύρου. Αυτό γινεται εφικτό με τη χρήση της συνάρτησης `data.entry()` που ανοίγει ένα παράθυρο για την επεξεργασία των στοιχείων των αντικειμένων. Για παράδειγμα, αν ορίσουμε έναν πίνακα

```
> pinakas = matrix(1:9, 3, 3)
> data.entry(pinakas)
```

και μηδενίσουμε τα στοιχεία της 1ης γραμμής του πίνακα, πληκτρολογώντας 0 στο παράθυρο που ανοίγει, τότε μετά το κλείσιμο του παραθύρου, τα στοιχεία του πίνακα έχουν μεταβληθεί.

```
> pinakas
     var1 var2 var3
[1,]    0    0    0
[2,]    2    5    8
[3,]    3    6    9
```

2.6 Ανάγνωση δεδομένων απο αρχείο

Στην R μπορεί να κάνει κανείς ανάγνωση δεδομένων που είναι αποθηκευμένα σε αρχεία τύπου ASCII με τη βοήθεια εντολών, μεταξύ άλλων, όπως `read.table(), scan()`. Μπορεί επίσης να γίνει ανάγνωση αρχείων τύπου Excel, SAS, SPSS, και άλλων, με τη βοήθεια συναρτήσεων που βρίσκονται σε κατάλληλα πακέτα για αυτό το σκοπό, αλλά αυτό δεν θα μας απασχολήσει σε αυτή τη σύντομη εισαγωγή.

Η συνάρτηση `read.table()` επιστρέφει ένα πλαίσιο δεδομένων, όπου τα ονόματα των μεταβλητών προεπιλέγονται ως: V1, V2,... . Η συνάρτηση αυτή έχει αρκετές επιλογές, και οι προεπιλεγμένες τιμές δίνονται παρακάτω:

```
read.table(file, header = FALSE, sep = "", quote = "\"'", dec = ".",
row.names, col.names, as.is = FALSE, na.strings = "NA",
colClasses = NA, nrows = -1, skip = 0, check.names = TRUE,
fill = !blank.lines.skip, strip.white = FALSE, blank.lines.skip = TRUE,
comment.char = "#")
```

Ο αναγνώστης παραπέμπεται στη βοήθεια που προσφέρει η R για να δεί τις λεπτομέρειες χρησης αυτών των επιλογών, όπως και παραλλαγές της συνάρτησης `read.table()`.

Η συνάρτηση `scan()` είναι αρκετές φορές πιο βολική απο την `read.table()`, διότι μπορούμε να επιλέξουμε τον τύπο του αντικειμένου που επιστρέφεται απο τη συνάρτηση. Η συνάρτηση αυτή έχει επίσης αρκετές επιλογές, και οι προεπιλεγμένες τιμές δίνονται παρακάτω:

```
scan(file = "", what = double(0), nmax = -1, n = -1, sep = "",
quote = if (sep=="\n") "" else "'\"", dec = ".", skip = 0, nlines = 0,
na.strings = "NA", flush = FALSE, fill = FALSE, strip.white = FALSE,
quiet = FALSE, blank.lines.skip = TRUE, multi.line = TRUE, comment.char = "")
```

2.7 Αποθήκευση δεδομένων σε αρχείο

Η συνάρτηση `write.table()` σώζει σε ένα αρχειο τα δεδομένα τύπου διάνυσμα, πίνακα ή πλαισίου δεδομένων. Τα ορίσματα και προεπιλογές αυτής της συνάρτησης είναι:

```
write.table(x, file = "", append = FALSE, quote = TRUE, sep = " ", eol = "\n",
na = "NA", dec = ".", row.names = TRUE, col.names = TRUE,
qmethod = c("escape", "double"))
```

Για τον ίδιο σκοπό χρησιμοποιειται και η συνάρτηση `write.table()` :

```
write(x, file = "data", ncolumns = if(is.character(x)) 1 else 5, append = FALSE)
```

Τέλος, για να αποθηκευτούν δεδομένα οποιουδήποτε τύπου χρησιμοποιειται η συνάρτηση `save()`:

```
save(..., list = character(0), file = stop("'file' must be specified"), ascii =
FALSE)
```

Για να αποθηκευτούν τα αντικειμενα X και Υ στο αρχείο με όνομα `my_data.Rdata` στη διεύθυνση `/Documents/data`, πληκτρολογούμε:

```
save(X, Y, file = "/Documents/data/|my_data.Rdata")
```

Η συνάρτηση `save.image()` είναι μία συντόμευση της συνάρτησης save(list = ls(all = TRUE), file = ".Rdata"). Επιτρέπει την αποθήκευση όλων των δεδομένων στη μνήμη. Έτσι μπορούν αυτά τα δεδομένα να φορτωθούν στη μνήμη με τη βοήθεια της εντολής: `load(".Rdata")`

Παρατήρηση: Αν δεν προσδιορίσουμε το μονοπάτι αποθήκευσης, τότε η R αποθηκεύει τα δεδομένα στη διεύθυνση που καθορίζεται μέσω της συνάρτησης `getwd()`. Μπορούμε να τροποποιήσουμε το μονοπάτι με τη βοήθεια της συνάρτησης `setwd()`, π.χ., `setwd("/Documents/data")`.

Κεφάλαιο 3

Τα γραφήματα

Η R διαθέτει πολυάριθμα γραφικά εργαλεία για την ανάλυση και την αναπαράσταση των δεδομένων. Ένα δείγμα των γραφικών δυνατοτήτων της μπορεί να έχει κανείς μέσω της εντολής `demo(graphics)`. Ο Πίνακας **??** δινει κάποια παραδείγματα γραφικών συναρτήσεων. Στη συνέχεια του Κεφαλαίου θα δωθούν κάποια παραδειγματα χρήσης αυτών των συναρτήσεων.

3.1 Διαχείρηση γραφικών παραθύρων

Όταν εκτελείται μία γραφική συνάρτηση, τότε ανοίγει ένα γραφικό παράθυρο και εμφανίζεται ένα γράφημα. Για να ανοίξουμε ένα καινούριο γραφικό παράθυρο, πληκτρολογούμε `X11()` (X Window System, version 11) ή `windows()`. Για να δούμε τα ορίσματα αυτής της συνάρτησης πληκτρολογούμε:

```
args(windows)

function (width = 7, height = 7, pointsize = 12,
              record = getOption("graphics.record"),
              rescale = c("R", "fit", "fixed"), xpinch, ypinch,
             canvas = "white",  gamma = getOption("gamma"))
NULL
```

Τα δύο πρωτα ορίσματα της συνάρτησης `windows()` ειναι οι διαστάσεις του πλάτους και του ύψους του γραφικού παραθύρου. Για να επαναπροσδιορίσουμε το μέγεθος του παραθύρου, π.χ., σε (5,5) πληκτρολογούμε `windows(5,5)`, όπου μονάδα μέτρησης είναι η ίντσα και η προεπιλεγμένη τιμή της είναι 7 και στις 2 διαστάσεις.

Άλλες συναρτήσεις διαχείρησης είναι για παράδειγμα οι `dev.set()`, `dev.cur()`, `dev.off()`, `graphics.off()`.
Αυτές οι συναρτήσεις είναι ιδιαίτερα χρήσιμες όταν υπάρχουν περισσότερα απο ένα παράθυρα ανοιχτά. Παρόλα αυτά, μόνο ένα μπορεί να είναι ενεργό και μόνο σε αυτό εμφανίζονται τα αποτελέσματα των εντολών που πληκτρολογούμε. Για να επιλέξουμε μέσα απο το σύνολο των ανοιχτών παραθύρων κάποιο συγκεκριμένο παράθυρο, π.χ., το παράθυρο i, πληκτρολογούμε `dev.set(i)`. Για να δούμε το νούμερο του παραθύρου που είναι ενεργό πληκτρολογούμε `dev.cur()`. Η συνάρτηση `dev.off()` χρησιμοποιείται για να κλείσει είτε το ενεργό

παράθυρο, αφήνοντας την παρένθεση κενή, ειτε να κλείσει το παράθυρο που καθορίζεται απο το νούμερο του παραθύρου μέσα στην παρένθεση. Με τη συνάρτηση `graphics.off()` μπορούμε να κλείσουμε όλα τα παράθυρα μαζί.

3.2 Διαμέριση ενός γραφήματος

Υπάρχουν 2 βασικοί τρόποι να διαμερίσουμε την οθόνη του παράθυρου γραφικών. Το κοινό χαρακτηριστικό τους ειναι ότι μας επιτρέπουν να φτιάξουμε περισσότερα απο ένα γραφήματα στο ίδιο παράθυρο γραφικών.

- `split.screen()`: καθορίζει τον τρόπο και τον αριθμό των περιοχών που θα διαμεριστεί το παράθυρο γραφικών. Η συνάρτηση αυτή ορίζεται ως:
  `split.screen(figs, screen, erase = TRUE)`, όπου figs είναι ένα δισδιάστατο διάνυσμα που καθοριζει τον αριθμό των γραμμών και των στηλών που θα διαμεριστει η οθόνη, και screen αντιστοιχεί στον αριθμό της οθόνης που κάνουμε εφαρμογή της συνάρτησης αυτής δύνωντας έτσι τη δυνατότητα να έχουμε διαμέριση μιας οθόνης που ειναι ήδη διαμερισμένη.

- `layout()`: διαμερίζει το ενεργό παράθυρο σε πολλαπλές περιοχές που θα εμφανιστούν διαδοχικά τα γραφήματα. Η συνάρτηση αυτή έχει ως όρισμα έναν πίνακα ακέραιων τιμών που καθορίζει τον αριθμό των υποπαραθύρων. Για να δούμε τη διαμέριση μπορούμε να πληκτρολογήσουμε `layout.show(n)`, όπου το n πρέπει να αντιστοιχει στον αριθμό των υποπαραθύρων. Για παράδειγμα, αν θέλουμε να διαμερίσουμε το αρχικό παράθυρο σε 4 ίσα μέρη, πληκτρολογούμε `layout(matrix(1:4, 2, 2))` και για να δούμε τη διαμέριση, πληκτρολογούμε `layout.show(4)`.