Οδηγός χρήσης της γλώσσας R

ΕΙΣΑΓΩΓΙΚΑ

Ονομασία αντικειμένων στην R

Τα ονόματα που χρησιμοποιούνται για την ονομασία των αντικειμένων στη γλώσσα R, σχηματίζονται με τα κεφαλαία και πεζά λατινικά γράμματα, με τα ψηφία 0-9 σε οποιαδήποτε μη αρχική θέση και την τελεία "**.**", η οποία θεωρείται ως γράμμα εκτός από περιπτώσεις όπως .49, 12.3, όπου θεωρείται ως υποδιαστολή δεκαδικού αριθμού. Ονόματα όπως **prod, Prod** ή **pRod** θεωρούνται διαφορετικά στη γλώσσα S., δεν επιτρέπεται. Ως διαχωριστικό των λέξεων χρησιμοποιείται η τελεία, π.χ. example.regr.01 και η υπογράμμιση "**_**", π.χ. example_regr.01. Ορισμένα ονόματα δεν συνιστάται να χρησιμοποιούνται διότι χρησιμοποιούνται από το πρόγραμμα ή έχουν ειδικές λειτουργίες. Τέτοια είναι τα **c** (παράθεση στοιχείω ή διανυσμάτων για σχηματισμό διανύσματος), **t** (ανάστροφο), **F** (False), **T** (True), **diff** (πρώτες διαφορές), **range, for, function, if, in, next, repeat, return και while**.

Σύνταξη εντολών στην R

Για την εισαγωγή δεδομένων και, γενικότερα, για την απόδοση τιμών σε μεταβλητές χρησιμοποιούμε συνήθως το συνδυασμό "**<-**" (assignment symbol), που "φαίνεται" σαν βέλος από δεξιά προς τα αριστερά. Η λειτουργία του συμβόλου αυτού είναι να υπολογίσει την έκφραση που δίνεται δεξιά του και να την αποδώσει στη μεταβλητή που βρίσκεται αριστερά του, χωρίς να τυπωθεί το αποτέλεσμα. Έτσι η εντολή

```
> x <- 3.1
```

που διαβάζεται "x παίρνει (gets) την τιμή 3.1", αποδίδει για επόμενη χρήση την τιμή 3.1 στη μεταβλητή x. Άλλος τρόπος για την αντιστοίχιση τιμών σε μεταβλητές είναι το ίσον "**=**", ή το αντίστροφο βέλος. Οι παρακάτω εντολές είναι ισοδύναμες με την προηγούμενη:

```
> x = 3.1
> 3.1 -> x
```

Αν το x είχε άλλη τιμή προηγούμενα, αυτή αντικαθίσταται με το 3.1. Δίνοντας στη συνέχεια το όνομα της μεταβλητής ή κάποια επιτρεπτή πράξη με αυτήν παίρνουμε ως εξαγόμενο την αντίστοιχη τιμή. Π.χ.

```
> x
[1] 3.1
> 3 * x
[1] 9.3
> x <- 3 * x
> x
[1] 9.3
```

Το σύμβολο **[1]** στο εξαγόμενο σημαίνει ότι η καταγραφή ξεκινά από το πρώτο στοιχείο του διανύσματος. Σημειώνεται ότι στην S δεν υπάρχουν απλοί αριθμοί, αλλά μόνον διανύσματα, συναρτήσεις λίστες και άλλα αντικείμενα (objects) Οι απλοί αριθμοί θεωρούνται ως διανύσματα διαστάσεως 1. Έτσι το **x=3.1** του παραδείγματος, είναι διάνυσμα με μοναδικό στοιχείο το 3.1. Όμοια, το **3+x** είναι επίσης διάνυσμα διαστάσεως 1 με μοναδικό στοιχείο το 9.3.

Για την καλύτερη αναγνωσιμότητα των εντολών και των εκφράσεων συνιστάται να αφήνονται κενά μεταξύ των συμβόλων, εκτός αν πρόκειται για συνδυασμό συμβόλων, όπως π.χ. το "**<-**". Γενικά το S-Plus αγνοεί τα κενά. Έτσι "**3+x**" ή "**3+ x**", είναι ισοδύναμες εκφράσεις. Δεν επιτρέπονται όμως κενά μεταξύ των ψηφίων ενός αριθμού ή στο όνομα μιας μεταβλητής. Για παράδειγμα είναι λάθος η έκφραση "**12 3+45**".

Μπορούν να δοθούν και περισσότερες από μία εντολές σε μία γραμμή με το διαχωριστικό "**;**". Το σύμβολο "**#**" σημαίνει ότι τα επόμενα μέχρι το τέλος της γραμμής είναι σχόλια. Έτσι αν οι μεταβλητές **a** και **b**, έχουν αρχικές τιμές **a=2.45**, **b=6.315**, μπορεί η εντολή να έχει τη μορφή:

```
> a <- 2.45 ; b <- 6.315       # initial values
```

Όταν ζητείται ο υπολογισμός και η εκτύπωση (στην οθόνη) μιας έκφρασης τότε η τιμή της καταγράφεται στη μνήμη με το όνομα **.Last.value**. Αν επομένως ξεχάσατε να ονομάσετε το αποτέλεσμα μιας έκφρασης, θυμηθείτε ότι αυτό βρίσκεται στην **.Last.value**. Η τιμή αυτή παραμένει μέχρι την επόμενη εκτέλεση εντολής που ζητά εκτύπωση στην οθόνη. Για παράδειγμα έχουμε:

```
> 1-pi+sin(pi/3)
[1] -1.275567
> g <- (1-pi)*cos(pi/3); 2 - .Last.value
[1] 3.070796
> .Last.value
[1] 3.070796
> gamma(1:10)
> factorials <- .Last.value
> facorials
[1]     1     1     2     6    24   120   720  5040 40320 362880
```

όπου η τιμή της **.Last.value** μετά την πρώτη εντολή έγινε -1.275567, μεταβλήθηκε σε 3.070796 παρόλο που το g δεν εμφανίστηκε διότι η τιμή του g ήταν η τελευταία τιμή, ενώ μετά την εντολή υπολογισμού των τιμών Γ(1), Γ(2), …, Γ(10), η τελευταία τιμή είναι το διάνυσμα των 0!, 1!, …,9!.
Ο παρακάτω πίνακας δίνει μερικούς βασικούς τελεστές στη γλώσσα R

Πίνακας τελεστών της R

Περιγραφή	R σύμβολο	Παράδειγμα
Σχόλια	#	# αυτό είναι σχόλιο
Απόδοση τιμής	<-	x<-sin(pi/2)
Τελεστής παράθεσης	c	c(1,3.5,2.8)
Στοιχείο επί στοιχείο πολλαπλασιασμός	*	a*b
Ύψωση σε δύναμη	^	a^b
Υπόλοιπο διαίρεσης ακεραίων x/y	%%	17%%5
Πηλίκο διαίρεσης ακεραίων x/y	%/%	17%/%5
Ακολουθία αριθμών από a έως b ανά h	seq	seq(0,20,2)
Ακολουθία αριθμών ανά 1	:	0:20

Η γλώσσα R χρησιμοποιεί όπως και οι περισσότερες γλώσσες και πακέτα τα γνωστά ονόματα και λειτουργίες των βασικών μαθηματικών συναρτήσεων που για αναφορά δίνονται στον επόμενο πίνακα:

Πίνακας συνήθων συναρτήσεων στην R

Περιγραφή	R σύμβολο	Παράδειγμα
ημίτονο, συνημίτονο, εφαπτομένη	sin,cos,tan	sin(pi/3) (=0.866..)
τόξο ημιτόνου, συνημιτόνου, εφαπτομένης	asin,acos,atan	atan(seq(0,1,.25))/pi
τετραγωνική ρίζα	sqrt	sqrt(x)
ακέραιο μέρος	⌊x⌋, ⌈x⌉	floor(x), ceiling(x)
φυσικός λογάριθμος	log	log(x)

Εκθετική συνάρτηση e^x	e^x	`exp(x)`
παραγοντικό	`n!`	`factorial(n)`
τυχαίοι αριθμοί στο (0,1)	`runif`	`runif(100)`
τυχαίοι κανονικοί αριθμοί	`rnorm`	`u=rnorm(100000,2,4)`
κανονική κατανομή	`pnorm, dnorm`	`pnorm(1,2,4)`
βαθμίδες, διάταξη	`rank, sort`	`z=floor(10*runif(10));z;` `rank(z);sort(z)`
διασπορά, συνδιασπορά	`var, cov`	`var(u), cov(x,y)`
τυπ. απόκλιση, συντ. συσχέτισης	`sd, cor`	`sd(u), cor(x,y)`

ΑΝΤΙΚΕΙΜΕΝΑ ΣΤΗ ΓΛΩΣΣΑ R

Διανύσματα και πίνακες

Τα διανύσματα είναι ο κύριος τρόπος εισόδου και εξόδου σταθερών ή μεταβλητών ποσοτήτων στην R. Τα στοιχεία των διανυσμάτων μπορεί να είναι αριθμητικές ή λογικές τιμές ή και συμβολοσειρές (strings), δεν μπορούν όμως να αναμειγνύονται. Δεν μπορεί δηλαδή το ένα στοιχείο του διανύσματος να είναι αριθμός και το άλλο συμβολοσειρά. Για την περίπτωση των αριθμητικών τιμών δεν ενδιαφέρει αν είναι ακέραιες, πραγματικές ή και μιγαδικές ούτε και αν είναι απλής ή διπλής ακρίβειας.

Ο απλούστερος τρόπος να ορίσουμε ένα διάνυσμα είναι με το να δώσουμε τα στοιχεία του χρησιμοποιώντας τη συνάρτηση **c** (από το "concatenate" που σημαίνει συνδέω κατά σειρά). Για παράδειγμα αν οι μαθητές **Alfred**, **Denis**, **Barbara**, **John**, **Mary** και **Nick** πήραν σε κάποια τεστ κατά μέσον όρο 29.3, 34.5, 22.7, 31.8, 27.3 και 28.5 τότε η επόμενη εντολή:

```
> grades <- c(29.3,34.5,22.7,31.8,27.3,28.5)
```

ορίζει το διάνυσμα **grades** με στοιχεία τους βαθμούς των έξι μαθητών. Η εντολή που αποτελείται από το όνομα του διανύσματος, τυπώνει στην οθόνη το διάνυσμα. Δηλαδή:

```
> grades
[1] 29.3  34.5  22.7  31.8  27.3  28.5
```

Χρησιμοποιώντας αγκύλες παίρνουμε συνιστώσες του διανύσματος. Π.χ.

```
> grades[2]
[1] 34.5
```

Μέσα στις αγκύλες μπορούμε να δώσουμε και σύνθετες εκφράσεις. Π.χ. η εντολή **x[5:20]** δίνει τα στοιχεία του διανύσματος **x** από το 5ο μέχρι και το 20ό. Η εντολή **x[c(2,5,10)]** δίνει το 2ο, 5ο και 10ο στοιχείο του **x**.

Οι λογικές τιμές παριστάνονται με τα κεφαλαία γράμματα T (για το TRUE αληθές) και F (για το FALSE ψευδές). Έτσι η έκφραση **grades > 30** παριστάνει ένα διάνυσμα με στοιχεία T όπου ο βαθμός είναι μεγαλύτερος του 30 και F αλλού. Πράγματι:

```
> grades > 30
[1] F T F T F F
```

Τα στοιχεία ενός διανύσματος μπορούν να ονομαστούν και να αναφέρονται ονομαστικά. Τα ονόματα είναι χαρακτήρες γι' αυτό χρειάζονται εισαγωγικά για τον ορισμό τους. Τα ονόματα των μαθητών που πήραν τους παραπάνω βαθμούς δίνονται ως εξής:

```
> students.names <- c("Alfred", "Denis", "Barbara",
+ "John", "Mary", "Nick")
```

Με την επόμενη εντολή τα ονόματα αυτά αντιστοιχίζονται με τους βαθμούς:

```
> names(grades) <- students.names
```

Τώρα το **grades** έχει αλλάξει. Πράγματι:

```
> grades
 Alfred Denis Barbara  John  Mary  Nick
   29.3  34.5    22.7  31.8  27.3  28.5
```

ενώ μόνο τα ονόματα τα παίρνουμε με την εντολή:

```
> names(grades)
[1] "Alfred" "Denis" "Barbara" "John" "Mary" "Nick"
```

Η ονομαστική αναφορά φαίνεται στις εντολές:

```
> grades["John"]
 John
 31.8
> grades[grades > 30]
 Denis  John
 34.5   31.8
```

Το μήκος του διανύσματος δίνεται από τη συνάρτηση

```
> length(grades)
[1] 6
```

Το είδος του διανύσματος ή άλλου αντικειμένου δίνεται από τη συνάρτηση

```
> class(grades)
[1] "numeric"
```

ενώ

```
> class(names(grades))
[1] "character"
```

και

```
> class(c(29.3, 34.5, 22.7, 31.8, 27.3, 28.5))
[1] "numeric"
> class(grades > 30)
[1] "logical",
```

Η αρίθμηση διανυσμάτων γίνεται καλύτερα με την paste. Π.χ.

```
> let <- letters[1:5]
> names(let) <- paste("letter", 1:5, sep = "") ; let
 letter1 letter2 letter3 letter4 letter5
    "a"     "b"     "c"     "d"     "e"
> num <- 1:5
> names(num) <- paste("number", 1:5, sep = "") ; num
 number1 number2 number3 number4 number5
       1       2       3       4       5
> grades <- as.vector(grades) ;  grades
[1] 29.3 34.5 22.7 31.8 27.3 28.5
#(Η συνάρτηση as.vector(x) απαλλάσσει το x από διάφορα attributes
#και το κάνει απλό διάνυσμα)
> names(grades) <- paste("student", 1:6, sep = ".")
> grades
 student.1 student.2 student.3 student.4 student.5 student.6
      29.3      34.5      22.7      31.8      27.3      28.5
```

Υπάρχουν διάφοροι τρόποι να μετατραπεί ένα διάνυσμα σε πίνακα. Ένας τρόπος είναι με την απόδοση διάστασης στο διάνυσμα. Για να μετατρέψουμε το διάνυσμα **grades** σε πίνακα 2×3, γράφουμε:

```
> dim(grades) <- c(2,3)
```

Τότε το διάνυσμα γίνεται πίνακας. Πράγματι:

```
> grades
      [,1]  [,2]  [,3]
[1,]  29.3  22.7  27.3
[2,]  34.5  31.8  28.5
```

Παρατηρούμε ότι η μετατροπή έγινε κατά στήλες. Δηλαδή τα πρώτα δύο στοιχεία του αρχικού

διανύσματος αποτελούν την πρώτη στήλη του πίνακα, τα επόμενα δύο τη δεύτερη και τα τελευταία δύο την τρίτη.

Αν θέλουμε την i γραμμή του πίνακα M γράφουμε **M[i,]**, για την j στήλη γράφουμε **M[, j]**, ενώ για το (i,j) στοιχείο γράφουμε **M[i, j]**, για παράδειγμα:

```
> grades[2,  ]
[1] 34.5 31.8 28.5
```

μας έδωσε τη δεύτερη γραμμή του πίνακα **grades**

```
> grades[, 3]
[1] 27.3 28.5
```

μας έδωσε τη τρίτη γραμμή του πίνακα **grades**

```
> grades[2, 3]
[1] 28.5
```

μας έδωσε το στοιχείο της δεύτερης γραμμής και τρίτης στήλης του πίνακα **grades**.

Η ακύρωση της προηγούμενης μετατροπής και η επαναφορά του **grades** στην κατάσταση του διανύσματος πετυχαίνεται με την εντολή:

```
> dim(grades) <- NULL
```

Ένας ευκολότερος τρόπος για τη δημιουργία πινάκων είναι με τη βοήθεια της συνάρτησης **matrix**. Το διάνυσμα **grades** μετατρέπεται σε πίνακα ως εξής:

```
> matrix(grades, 2, 3)
     [,1]  [,2]  [,3]
[1,]  29.3  22.7  27.3
[2,]  34.5  31.8  28.5
```

Αν θέλουμε η μετατροπή να γίνει κατά γραμμές, γράφουμε:

```
> matrix(grades, 2, 3, byrow=T)
     [,1]  [,2]  [,3]
[1,]  29.3  34.5  22.7
[2,]  31.8  27.3  28.5
```

Η συνάρτηση **c( )**, συνδέει ομοειδή αντικείμενα, ή τα μετατρέπει, π.χ.

```
> c(T, F, pi, 7)
[1] 1.000000 0.000000 3.141593 7.000000
```

Αν έχουμε ανόμοια αντικείμενα, τότε έχουμε λίστα.

Η λίστα είναι μία συλλογή ανόμοιων πληροφοριών που αφορούν το ίδιο άτομο ή κατάσταση. Π.χ. ένας κατάλογος στο σχολείο περιέχει το ονοματεπώνυμο του μαθητή, τα ονόματα των γονιών του, τη διεύθυνσή του, την ημερομηνία γέννησής του, τους βαθμούς του στα διάφορα μαθήματα και πολλά άλλα στοιχεία. Όμοια ένας κατάλογος εργαζομένων, μία μισθολογική κατάσταση, μία καταγραφή σε ένα ερωτηματολόγιο κλπ. περιέχουν στήλες με αριθμητικές τιμές, με ονόματα (χαρακτήρες) ή με ημερομηνίες κλπ.

Έστω ότι για τους προηγούμενους μαθητές γνωρίζουμε την ηλικία τους, και τους βαθμούς στα επιμέρους τεστ που έλαβαν μέρος. Έστω ακόμη ότι δεν έδωσαν το ίδιο πλήθος τεστ και ότι οι βαθμοί τους δίνονται με τα διανύσματα **sc1** έως **sc6**. Μια λίστα που αφορά τους μαθητές αυτής της τάξης μπορεί να είναι η επόμενη:

```
> sc1 <- c(27,31,30) ; sc2 <- c(28,38,32,40)
> sc3 <- c(28,20,20) ; sc4 <- c(30,35,35,27)
> sc5 <- c(36,30,16) ; sc6 <- c(27,32,25,30)


> Class <- list(students=students.names, age=c(12,13,
```

```
+ 12,12,13,12), tests=c(3, 4, 3, 4, 3, 4),
+ scores=c(sc1, sc2, sc3, sc4, sc5, sc6))
```

Δίνοντας την εντολή **class** παίρνουμε:

```
> Class
$students:
[1] "Alfred" "Denis" "Barbara" "John" "Mary" "Nick"
$age:
[1] 12 13 12 12 13 12
$tests:
[1] 3 4 3 4 3 4
$scores:
[1] 27 31 30 28 38 32 40 ..... 27 32 25 30
```

Παρατηρούμε ότι η λίστα του παραδείγματος έχει τέσσερις κατηγορίες, έχει, όπως θα αναφέρεται, μήκος (length) 4 και κάθε μία απ' αυτές αναφέρεται ως **Class[[1]]**, **Class[[2]]**, **Class[[3]]** και **Class[[4]]**. Επειδή κάθε κατηγορία είναι διάνυσμα το **Class[[4]][4:7]** ορίζει τα στοιχεία του από το 4ο μέχρι το έβδομο που ταυτίζεται με το **sc2**. Μπορούμε όμως να αναφερόμαστε και απ' ευθείας στα ονόματα. Έτσι:

```
> Class$students
[1] "Alfred" "Denis" .....  "Nick"
> Class$age[3]
[1] 12
```

Τα ονόματα των συνιστωσών μπορούν να συμπτυχθούν στο μικρότερο δυνατό πλήθος γραμμάτων που τα επιτρέπει να διακρίνονται από τα υπόλοιπα. Αντί του **Class$students** μπορεί να χρησιμοποιηθεί το **Class$st** για να διακρίνεται από την τέταρτη κατηγορία την **Class$sc**. Όμως αντί του **Class$tests** μπορεί να χρησιμοποιηθεί απλά το **Class$t** αφού δεν υπάρχει άλλη κατηγορία που αρχίζει από **t**.

Η συνάρτηση **c** προσθέτει και άλλες κατηγορίες σε μια λίστα. Π.χ. αν οι μαθητές κατάγονται από Θεσ/νίκη, Καβάλα, Θεσ/νίκη, Θεσ/νίκη, Κατερίνη, Γιαννιτσά, αντίστοιχα, τότε ο τόπος γέννησης προστίθεται στη λίστα ως εξής:

```
> places <- c("Thes","Kav","Thes","Thes","Kat","Gian")
> Class <- c(Class, list(Place.birth=places)).
```

Η συνάρτηση **unlist** μετατρέπει τη λίστα σε διάνυσμα:

```
> unlist(Class)
student1 …  student6 age1 …  age6 tests1 …  test6
"Alfred" …    "Nick" "12" …  "12"    "3" …     "4"
scores1 scores2   scores21 … Place.birth5 Place.birth6
   "27"    "31"       "30" …        "Kat"       "Gian"
```

Αν δεν μας ενδιαφέρουν τα ονόματα των συνιστωσών, τότε:

```
> unlist(Class, use.names=F)
[1] "Alfred" "Denis" "Barbara"  "John"  "Mary"
[6]   "Nick"    "12"      "13"    "12"    "12"
-----------------------------------------------
[37]   "32"     "25"    "30"    "Kat"    "Gian"
```

Τα ονόματα στηλών και γραμμών ενός πίνακα μπορούν να καθοριστούν με μία λίστα, που η πρώτη συνιστώσα της αφορά τις γραμμές και η δεύτερη τις στήλες. Π.χ.

```
> dim(grades) <- c(2,3); grades
> dimnames(grades) <- list(paste("row",letters[1:2]),
     +    paste("col",LETTERS[1:3])) ; grades


      col A col B col C
```

```
row a  29.3  22.7  27.3
row b  34.5  31.8  28.5
```

Προσέξτε τη διαφορά των ενσωματωμένων στο πρόγραμμα διανυσμάτων **letters** και **LETTERS**.

Αν θέλουμε να αριθμήσουμε μόνο τις στήλες, π.χ. Τάξη Α, Τάξη Β, Τάξη Γ, γράφουμε:

```
> dimnames(grades) <- list(NULL, paste("Τάξη",c("Α","Β","Γ")))
> grades
     Τάξη Α Τάξη Β Τάξη Γ
[1,]   29.3   22.7   27.3
[2,]   34.5   31.8   28.5
```

Πλαίσια δεδομένων (Data Frames), Πολλαπλοί πίνακες (arrays)

Ένα πλαίσιο δεδομένων είναι μία λίστα μεταβλητών ίδιου μήκους αλλά ενδεχομένως διαφορετικού τύπου. Ένα παράδειγμα πλαισίου δεδομένων είναι το iris που είναι ενσωματωμένο στην R και έχει 5 μεταβλητές και 150 παρατηρήσεις. Γράφοντας

```
> iris[c(1:3,147:150),  ,  ]
```

Παίρνουμε τις τρεις πρώτες και τις τρεις τελευταίες γραμμές του iris:

```
    Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
1            5.1         3.5          1.4         0.2    setosa
2            4.9         3.0          1.4         0.2    setosa
3            4.7         3.2          1.3         0.2    setosa
147          6.3         2.5          5.0         1.9 virginica
148          6.5         3.0          5.2         2.0 virginica
149          6.2         3.4          5.4         2.3 virginica
150          5.9         3.0          5.1         1.8 virginica
```

Οι τέσσερις πρώτες στήλες είναι 4 μετρήσεις του μήκους και πλάτους των σεπάλων και του μήκους και πλάτους των πετάλων του λουλουδιού iris(αγριόκρινο). Έγιναν από τον R.A.Fisher σε 50 άνθη από κάθε ένα από τρία είδη, τα Setosa, Versicolor, and Virginica. Το είδος δίνεται με αλφαριθμητική μεταβλητή στην πέμπτη στήλη. Η πρώτη σειρά δίνει τα ονόματα των μεταβλητών που μπορούμε να τα έχουμε ως διάνυσμα με την εντολή:

```
> names(iris)
[1] "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width" "Species"
```

Οι αριθμοί πριν κάθε γραμμή δίνουν τον αύξοντα αριθμό της περίπτωσης.

Οι εντολές:

```
> z<-iris$Sepal.Width
> z<-iris[[2]]
```

Είναι ισοδύναμε και μας δίνουν τη δεύτερη μεταβλητή, ενώ η επόμενη μας δίνει τη μέση τιμή και την τυπική απόκλιση της δεύτερης μεταβλητής.

```
> c(mean=mean(z),st_dev=sd(z))
     mean    st_dev
3.0573333 0.4358663
```

Η επόμενη εντολή δίνει ένα πίνακα συχνοτήτων ως προς το είδος των λουλουδιών

```
> table(iris$Species)
    setosa versicolor  virginica
        50         50         50
```

Για να μη γράφουμε το όνομα του πλαισίου δεδομένων κάνουμε attach, π.χ.

```
> attach(iris)
> x1=Sepal.Length[1:50];x2=Sepal.Length[51:100];
```

```
+ x3=Sepal.Length[101:150]
> summary(x1)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  4.300   4.800   5.000   5.006   5.200   5.800
> summary(x2)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  4.900   5.600   5.900   5.936   6.300   7.000
> summary(x3)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  4.900   6.225   6.500   6.588   6.900   7.900
```

Οι τρεις τελευταίες εντολές δίνουν τα περιγραφικά μέτρα θέσεις για τη μεταβλητή Sepal.Length στα τρία διαφορετικά είδη λουλουδιών.

Ένας πίνακας πολλαπλής εισόδου (Array) είναι μία ταξινόμηση p×q×r×.. αριθμών σε p γραμμές, q στήλες, r επίπεδα(ορόφους), …και πετυχαίνεται με αντιστοίχιση πολλαπλών δεικτών. Δηλ. κάθε στοιχείο του Array έχει τρεις δείκτες (i,j,k,…) που κυμαίνονται από 1 έως αντίστοιχα p, q, r, …
Η δημιουργία μιας **array** είναι ανάλογη με αυτήν του πίνακα:

```
> Aarray <- array(c(1:8, 11:18, 111:118),
+ dim = c(2, 4, 3)) ;Aarray
, , 1
     [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8

, , 2
     [,1] [,2] [,3] [,4]
[1,]   11   13   15   17
[2,]   12   14   16   18

, , 3
     [,1] [,2] [,3] [,4]
[1,]  111  113  115  117
[2,]  112  114  116  118
```

Παρατηρούμε ότι για την στάθμη 1 της τρίτης διάστασης (1ος όροφος) τα πρώτα 2×4=8 στοιχεία του διανύσματος σχηματίζουν πίνακα διπλής εισόδου. Η πρώτη στάθμη της δεύτερης διάστασης (1η στήλη) αποτελείται από τα πρώτα 2 στοιχεία, η 2η στήλη από τα άλλα δύο στοιχεία κ.ο.κ. Το ίδιο για το 2ο όροφο με τα επόμενα 8 στοιχεία και τέλος ο 3ος όροφος με τα τελευταία 8 στοιχεία.
Αν θέλουμε το δεύτερο όροφο γράφουμε:

```
> Aarray[,  , 2]
     [,1] [,2] [,3] [,4]
[1,]   11   13   15   17
[2,]   12   14   16   18
```

ενώ με την παρακάτω εντολή, παίρνουμε όλες τις πρώτες γραμμές και στους τρεις ορόφους:

```
> Aarray[1,  ,  ]
     [,1] [,2] [,3]
[1,]    1   11  111
[2,]    3   13  113
[3,]    5   15  115
[4,]    7   17  117
```

Το πλαίσιο δεδομένων iris μπορεί να μετατραπεί σε array με διαστάσεις (50,3,4) με τις επόμενες εντο-

λές:

```
> x<-as.matrix(iris[,1:4])
> y<-array(x,dim=c(50,3,4))
```

Οι δύο πρώτες (από τις 50) γραμμές αυτού του πίνακα τριπλής εισόδου είναι

```
> y[1:2,,]
, , 1
     [,1] [,2] [,3]
[1,]  5.1  7.0  6.3
[2,]  4.9  6.4  5.8

, , 2
     [,1] [,2] [,3]
[1,]  3.5  3.2  3.3
[2,]  3.0  3.2  2.7

, , 3
     [,1] [,2] [,3]
[1,]  1.4  4.7  6.0
[2,]  1.4  4.5  5.1

, , 4
     [,1] [,2] [,3]
[1,]  0.2  1.4  2.5
[2,]  0.2  1.5  1.9
```

Με τις επόμενες εντολές εξάγουμε τα ονόματα των τριών ειδών (στάθμες της δεύτερης διάστασης) και των τεσσάρων μεταβλητών (στάθμες της τρίτης διάστασης) και αυτά τα ονόματα τα δίνουμε στις διαστάσεις του array για να είναι ευανάγνωστο:

```
> ns=names(table(Species))
> nm=names(iris)[-5]
> y<-array(x,dim=c(50,3,4),dimnames=list(NULL,ns,nm))
> y[1:2,,]
, , Sepal.Length
     setosa versicolor virginica
[1,]    5.1        7.0       6.3
[2,]    4.9        6.4       5.8

, , Sepal.Width
     setosa versicolor virginica
[1,]    3.5        3.2       3.3
[2,]    3.0        3.2       2.7

, , Petal.Length
     setosa versicolor virginica
[1,]    1.4        4.7       6.0
[2,]    1.4        4.5       5.1

, , Petal.Width
     setosa versicolor virginica
[1,]    0.2        1.4       2.5
[2,]    0.2        1.5       1.9
```

Με την εντολή

```
> apply(y,2:3,FUN=mean)
           Sepal.Length Sepal.Width Petal.Length Petal.Width
setosa            5.006       3.428        1.462       0.246
versicolor        5.936       2.770        4.260       1.326
virginica         6.588       2.974        5.552       2.026
```

παίρνουμε τη μέση τιμή όλων των 4 μεταβλητών ξεχωριστά για κάθε είδος (αυτό καθορίζεται με το δεύτερο όρισμα της εντολής apply). Αν ως συνάρτηση θέσουμε π.χ. var θα πάρουμε τις διασπορές και μπορούμε να θέσουμε διάφορες συναρτήσεις.

Παράγοντες

Έτσι ονομάζουμε μεταβλητές κατηγοριοποίησης οι οποίες παίρνουν μικρό πλήθος διακεκριμένων τιμών, που αναφέρονται ως στάθμες ή βαθμίδες. Πρόκειται δηλαδή για διανύσματα ειδικού τύπου. Π.χ. αν Sm παριστάνει τον καπνιστή (smoker), No-Sm τον μη-καπνιστή και NA την έλλειψη απάντησης, τότε ο παράγοντας που περιγράφει το κάπνισμα επτά ατόμων μπορεί να είναι:

```
> smok <- factor(c("Sm","No-Sm","No-Sm","NA","Sm","NA","Sm"))
> smok
[1] Sm    No-Sm  No-Sm  NA    Sm    NA    Sm
Levels: NA No-Sm Sm
```

Παρατηρείστε ότι οι στάθμες του παράγοντα αν και συμβολοσειρές τυπώνονται χωρίς τα εισαγωγικά. Αυτό διευκολύνει την καλύτερη παρουσίαση αποτελεσμάτων των στατιστικών αναλύσεων, όπου υπεισέρχονται παράγοντες. Η συνάρτηση **factor** αντιστοιχίζει κωδικούς στα ονόματα των σταθμών του παράγοντα, ενώ τα ίδια τα ονόματα θεωρούνται ως attributes.

```
> print.default(smok)
[1] 3 2 2 1 3 1 3    # οι αριθμοί αντιστοιχούν στα levels δηλ. αλφαβητικά
> attributes(smok)
$levels
[1] "NA"    "No-Sm" "Sm"

$class
[1] "factor"
```

Οι στάθμες καθορίζονται αυτόματα με αλφαβητική κωδικοποίηση των ονομάτων τους. Μπορούμε όμως να τις καθορίσουμε με τον τρόπο που θέλουμε ως εξής:

```
> smoka <- factor(c("Sm", "No-Sm", "No-Sm","NA",
+"Sm","NA","Sm"), levels=c("No-Sm","Mid-Sm","Sm"))
> smoka
[1] Sm    No-Sm No-Sm <NA>  Sm    <NA>  Sm
Levels: No-Sm Mid-Sm Sm
> print.default(smoka)
[1]  3  1  1 NA  3 NA  3
```

Όταν οι στάθμες του παράγοντα είναι διαταγμένες όπως συμβαίνει συχνά στην ανάλυση διασποράς, μπορούμε να χρησιμοποιήσουμε τη συνάρτηση ordered αντί της factor. Π.χ.

```
> income <- ordered(c("Mid","Lo","Lo","Hi","Mid",
+"Hi","Hi","Lo"), levels=c("Lo","Mid","Hi"))
```

Δίνοντας income παίρνουμε εκτός από τους κωδικούς και τη διάταξη:

```
> income
[1] Mid Lo Lo Hi Mid Hi Hi Lo
Lo < Mid < Hi
```

Θα μετατρέψουμε στη συνέχεια τη λίστα που είχαμε δημιουργήσει προηγούμενα την Class σε

πλαίσιο δεδομένων χρησιμοποιώντας τις μεταβλητές που είχαμε καθώς και το βαθμό του κάθε φοιτητή (ως μέσο όρο στρογγυλευμένο σε ένα δεκαδικό ψηφίο) των τεστ που έδωσε και το εάν πέρασε το μάθημα (βαθμός >30) ή όχι. Η τελευταία είναι παράγοντας. Οι μεταβλητές (συνιστώσες της λίστας) που είχαμε είναι:

Κάνουμε πρώτα

```
> attach(Class)
> students
[1] "Alfred"  "Denis"   "Barbara" "John"    "Mary"    "Nick"
> age
[1] 12 13 12 12 13 12
> tests
[1] 3 4 3 4 3 4
> scores
 [1] 27 31 30 28 38 32 40 28 20 20 30 35 35 27 36 30 16 27
[19] 32 25 30
> Place.birth
[1] "Thes" "Kav"  "Thes" "Thes" "Kat"  "Gian"
```

Βρίσκουμε τους βαθμούς των φοιτητών από τα τεστ και τους εισάγουμε σε ένα διάνυσμα grades.

```
> g1=round(mean(scores[1:3]),1);g1
[1] 29.3
> g2=round(mean(scores[4:7]),1);g2
[1] 34.5
> g3=round(mean(scores[8:10]),1);g3
[1] 22.7
> g4=round(mean(scores[11:14]),1);g4
[1] 31.8
> g5=round(mean(scores[15:17]),1);g5
[1] 27.3
> g6=round(mean(scores[18:21]),1);g6
[1] 28.5
> grades=c(g1,g2,g3,g4,g5,g6);grades
[1] 29.3 34.5 22.7 31.8 27.3 28.5
```

Που είναι αυτό που είχαμε ορίσει αρχικά. Ορίζουμε και ένα διάνυσμα (παράγοντα) pass με τιμές FAIL και PASS με τη βοήθεια της **factor** ή της **ordered**:

```
> pass <- factor(grades > 30, labels = c("FAIL","PASS")) ; pass
[1] FAIL PASS FAIL PASS FAIL FAIL
```

Χρησιμοποιώντας τώρα την εντολή data.frame, τα αριθμητικά διανύσματα **tests**, **grades**, **age**, το αλφαριθμητικό **Place.birth** και ο παράγοντας **pass**, ορίζουν ένα πλαίσιο δεδομένων στο S-Plus.

```
> stds <- data.frame(tests, grades, pass, age, Place.birth); stds
  tests grades pass age Place.birth
1     3   29.3 FAIL  12        Thes
2     4   34.5 PASS  13         Kav
3     3   22.7 FAIL  12        Thes
4     4   31.8 PASS  12        Thes
5     3   27.3 FAIL  13         Kat
6     4   28.5 FAIL  12        Gian
```

Αν κάποιες από τις στήλες θέλουμε να έχουν διαφορετικά ονόματα από αυτά των μεταβλητών, το δηλώνουμε ως «νέο όνομα» = «παλιό όνομα», στην εντολή **data.frame**. Αρκεί μόνο το όνομα να είναι επιτρεπτό (χωρίς κενά και ειδικά σύμβολα).

```
> stds <- data.frame(n.of.tests = tests, grades, pass, age,
```

```
+ birth.places = Place.birth) ; stds
  n.of.tests grades pass age birth.places
1          3   29.3 FAIL  12         Thes
2          4   34.5 PASS  13          Kav
3          3   22.7 FAIL  12         Thes
4          4   31.8 PASS  12         Thes
5          3   27.3 FAIL  13          Kat
6          4   28.5 FAIL  12         Gian
```

Αν θέλουμε και τα ονόματα των παιδιών (γραμμών), απλά δίνουμε την εντολή:

```
> row.names(stds) <- students ; stds
        n.of.tests grades pass age birth.places
Alfred           3   29.3 FAIL  12         Thes
Denis            4   34.5 PASS  13          Kav
Barbara          3   22.7 FAIL  12         Thes
John             4   31.8 PASS  12         Thes
Mary             3   27.3 FAIL  13          Kat
Nick             4   28.5 FAIL  12         Gian
```

Αρχεία, Workspace και Working Directory

Όλα τα αντικείμενα που δημιουργούνται κατά την εκτέλεση των εντολών της R σώζονται στο Workspace και μπορούμε να τα δούμε με την εντολή

```
> ls()
```

Μπορούμε να διαγράψουμε όποια από αυτά θέλουμε με τις εντολές **rm(x,y,..)** ή **remove(x,y,…)**. Αν θέλουμε να υπάρχουν διαθέσιμα τα αντικείμενα αυτά την επόμενη φορά που θα ανοίξουμε την R, τότε πριν κλείσουμε το πρόγραμμα επιλέγουμε να σώσουμε το Workspace. Αλλιώς δεν το σώζουμε και την επόμενη φορά εκτελούμε εκείνα που μας χρειάζονται.

Το να γνωρίζουμε ποια αντικείμενα υπάρχουν στο Workspace και κυρίως τι περιεχόμενο έχουν, φαίνεται από το εξής παράδειγμα:

Έστω ότι σε προηγούμενη εκτέλεση είχαμε ορίσει το διάνυσμα x ως x=(1.2,2,5,7,8) και κατόπιν ορίσαμε το data.frame που το ονομάσαμε dat με δύο μεταβλητές x,y, π.χ

```
> x=c(1.2,2,5,7,8)
> dat <- data.frame(x=c(1:10,1:10), y=1:20)
```

Κάνοντας **attach** μπορούμε π.χ. να προσθέσουμε τα x,y πιστεύοντας ότι θα προστεθούν τα διανύσματα του dat. Όμως παίρνουμε ως εκτέλεση:

```
> attach(dat)
The following object(s) are masked _by_ '.GlobalEnv':

    x
The following object(s) are masked from 'dat (position 3)':

    x, y
The following object(s) are masked from 'dat (position 4)':

    x, y
> x+y
 [1]  2.2  4.0  8.0 11.0 13.0  7.2  9.0 13.0 16.0 18.0
[11] 12.2 14.0 18.0 21.0 23.0 17.2 19.0 23.0 26.0 28.0
```

που προφανώς δεν είναι αυτό που περιμέναμε. Βέβαια στην περίπτωση αυτή το πρόγραμμα μας προειδοποιεί ότι υπάρχει και άλλο x εκτός από αυτό που είναι στο dat και αν το προσέξουμε θα διορθώσουμε το λάθος διαγράφοντας το x (αυτό που εξάγεται από το dat δεν διαγράφεται) και εκτελώντας πάλι το

ζητούμενο.

```
> rm(x)
> x+y
 [1]  2  4  6  8 10 12 14 16 18 20 12 14 16 18 20 22 24 26 28 30
```

Βρίσκουμε το σωστό αποτέλεσμα.

Όταν σώζουμε αρχεία ή όταν θέλουμε να διαβάσουμε υπάρχοντα αρχεία το πρόγραμμα τα σώζει ή τα αναζητά στο Working Directory. Για να ξέρουμε ποιο είναι αυτό εκτελούμε την εντολή:

```
> getwd()
[1] "c:/Users/Chronis/Desktop/My R Dir"
```

ενώ μπορούμε εύκολα να το αλλάξουμε με την εντολή

```
> setwd("f:/temp")
> getwd()
[1] "f:/temp"
```

Εναλλακτικά βρισκόμενοι στην R Console από τον κατάλογο FILE του προγράμματος επιλέγουμε Change dir και κατευθύνουμε το πρόγραμμα στον κατάλογο που μας ενδιαφέρει.

Ας υποθέσουμε ότι με έναν οποιονδήποτε editor έχουμε σώσει ένα αρχείο bathmoi.dat που περιέχει σε γραμμές και στήλες τις παρακάτω πληροφορίες:

```
        sex   school  g1      g2      g3      g4      pref
stud001 M     "15"    6.2     12.5    19.6    17.6    3
stud002 M     "12"    12.8    15.0    17.2    19.9    1
stud003 F     "05"    07.2    09.5    11.6    13.1    9
stud004 M     "11"    15.5    14.3    10.9    16.4    2
stud005 F     "11"    19.2    18.6    15.9    18.6    1
stud006 M     "05"    17.6    14      10.5    8.2     6
```

Τοποθετούμε το αρχείο στο Working Directory και εκτελούμε την εντολή

```
> bathmoi<-read.table("bathmoi.dat");bathmoi
        sex school   g1   g2   g3   g4 pref
stud001   M     15  6.2 12.5 19.6 17.6    3
stud002   M     12 12.8 15.0 17.2 19.9    1
stud003   F      5  7.2  9.5 11.6 13.1    9
stud004   M     11 15.5 14.3 10.9 16.4    2
stud005   F     11 19.2 18.6 15.9 18.6    1
stud006   M      5 17.6 14.0 10.5  8.2    6
```

και παίρνουμε το πλαίσιο δεδομένων bathmoi.

Αν έχουμε αρχεία από το SPSS μπορούμε να τα εισάγουμε στην R αφού τρέξουμε το πακέτο foreign

```
> library(foreign)
> ?read.spss
> read.spss(file.sav)
```

Αν έχουμε αρχείο από το Excel το εισάγουμε με το πακέτο gdata

```
> library(gdata)
> ?read.xls
> read.xls(lile.xls
```

Εκφράσεις και τελεστές

Η γλώσσα S είναι εφοδιασμένη με αριθμητικούς και λογικούς τελεστές. Εκτός από τους συνήθεις τελεστές πρόσθεσης, αφαίρεσης πολλαπλασιασμού διαίρεσης, ύψωσης σε δύναμη, έχουμε:

Οι λογικοί τελεστές της R για τις συγκρίσεις μικρότερο, μεγαλύτερο, μικρότερο ή ίσο, μεγαλύτερο ή ίσο, ίσο και άνισο είναι:

```
<    >    <=   >=   ==   !=
```

όπου πρέπει να τονιστεί ότι τα διπλά σύμβολα δεν χωρίζονται με κενά. Υπάρχουν επίσης οι λογικοί τελεστές

```
&    |    !
```

που χρησιμοποιούνται για τις λογικές πράξεις ΚΑΙ, ή, ΟΧΙ.
Οι τελεστές ελέγχου

```
&&   ||
```

χρησιμοποιούνται για υπολογισμό κάτω από προϋποθέσεις. Έτσι αν **a** και **b** είναι δύο εκφράσεις, τότε **a && b** είναι αληθής (TRUE) αν και οι δύο εκφράσεις είναι αληθείς. Αν η **a** είναι ψευδής η **b** δεν υπολογίζεται καθόλου. Παραδείγματος χάριν για τον υπολογισμό των λογαρίθμων του διανύσματος **x** είναι προφανώς απαραίτητο το **x** να είναι αριθμητικό διάνυσμα και στην περίπτωση αυτή να αποτελείται μόνο από θετικά στοιχεία. Έτσι δίνουμε

```
> if(mode(x)!="character" && min(x)>0) log(x)
```

που επιτρέπει τον υπολογισμό των λογαρίθμων μόνο όταν επιτρέπεται. Για παράδειγμα

```
> x <- cos(((1:10) * pi)/3) ; > x
 [1]  0.5 -0.5 -1.0 -0.5  0.5  1.0  0.5 -0.5 -1.0 -0.5
> if(mode(x) != "character" && x > 0) log(x)
 [1] -0.6931     NaN     NaN     NaN -0.6931  0.000
 [7] -0.6931     NaN     NaN     NaN
> if(mode(x) != "character" && min(x) > 0) log(x)
```

όπου στη δεύτερη περίπτωση έπρεπε το ελάχιστο των τιμών του x να είναι θετικό, για να ικανοποιηθεί η απαιτούμενη προϋπόθεση και γι' αυτό δεν δίνει αποτέλεσμα.
Όμοια η έκφραση **a || b** είναι αληθής (TRUE) αν μία από τις δύο εκφράσεις είναι αληθείς. Αν η **a** είναι αληθής η **b** δεν υπολογίζεται καθόλου. Ως παράδειγμα η εκτέλεση του:

```
k=0
for (i in 1:1000) {
x <- round(runif(100),2) ; y <- round(runif(100),2)
sort(x) ; sort(y)
if (any(x == 0.55) || any(y == 0.55)) { ".55 encountered"
k=k+1}
}
k
```

έδωσε 864, δηλαδή από 1000 τυχαίες επιλογές των διανυσμάτων x και y με 100 τυχαίους αριθμούς στο (0,1) στρογγυλευμένους σε δύο δεκαδικά, υπήρξαν 864 περιπτώσεις που είτε το x είτε το y είχε το 0.55.

Ο κανόνας ανακύκλωσης

Αναφέρθηκε ήδη ότι η έκφραση **x + 1.3** προσθέτει το 1.3 σε κάθε στοιχείο του **x**. Αυτό φαίνεται συντακτικά σωστό και εύκολο, μαθηματικά όμως είναι λάθος αφού το 1.3 είναι ένας αριθμός, δηλ. διάνυσμα διάστασης 1 ενώ το **x** μπορεί να έχει οποιαδήποτε διάσταση. Ανάλογο πρόβλημα εμφανίζεται κάθε φορά που συνυπάρχουν στην ίδια έκφραση διανύσματα διαφορετικής διάστασης.

Χρειάζεται λοιπόν να καθοριστεί μία σύμβαση ώστε να είναι δυνατόν το πρόγραμμα να χειρίζεται τέτοιες περιπτώσεις. Τη σύμβαση αυτή εκφράζει ο κανόνας ανακύκλωσης (**recycling rule**). Σύμφωνα με αυτόν:
Το αποτέλεσμα κάθε έκφρασης στη γλώσσα R εκφράζεται με ένα διάνυσμα που έχει διάσταση τη μεγαλύτερη από τις διαστάσεις των διανυσμάτων που συμμετέχουν, αρκεί οι μικρότερες διαστάσεις να διαιρούν ακριβώς τη μεγαλύτερη. Τα διανύσματα με μικρότερη διάσταση συμπληρώνονται μέχρι τη μεγαλύτερη με επανάληψη των στοιχείων τους για όσες φορές απαιτείται. Ιδιαίτερα ένας αριθμός επαναλαμβάνεται τόσες φορές όσες η διάσταση του μεγαλύτερου μήκους διανύσματος.

Έτσι για παράδειγμα αν **x** είναι το διάνυσμα των πρώτων 16 φυσικών αριθμών και **y** το διάνυσμα των πρώτων 10 αριθμών, τότε το **x+y** υπολογίζεται με συμπλήρωση του y με τα πρώτα 6 στοιχεία του ώστε να γίνει διάνυσμα μήκους 16. Το πρόγραμμα όμως τυπώνει ένα μήνυμα (προειδοποίηση) για

να μας ειδοποιήσει ότι υπάρχει πρόβλημα με τις διαστάσεις.

```
> x <- 1:16 ; x
[1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
> y <- 1:10 ; y
[1]  1  2  3  4  5  6  7  8  9 10
> x + y
[1]  2  4  6  8 10 12 14 16 18 20 12 14 16 18 20 22
Warning message:
In x + y : longer object length is not a multiple of
shorter object length
```

Ενώ, αν είχαμε δώσει

```
> y <- 1:8
> x + y
 [1]  2  4  6  8 10 12 14 16 10 12 14 16 18 20 22 24
```

δεν θα τύπωνε καμία προειδοποίηση.

Ακόμη, μπορεί αυτό το διάνυσμα να γραφεί ως πίνακας διάστασης 3×6 και ας μην υπάρχουν αρκετά στοιχεία. Πράγματι:

```
> matrix(x + y, 3, 6)
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    2    8   14   12   18   24
[2,]    4   10   16   14   20    2
[3,]    6   12   10   16   22    4
Warning message:
In matrix(x + y, 3, 6) :
  data length [16] is not a sub-multiple or multiple of the num-
ber of rows [3]
```

Η εκτέλεση, όμως, της εντολής:

```
> 2 ^ y - 1
[1] 1 3 7 15 31 63 127 255 511 1023
```

δεν συνοδεύεται από κανένα μήνυμα. Γενικά, αν η διάσταση του μεγαλύτερου μήκους διανύσματος είναι πολλαπλάσιο των μικρότερων μηκών διανυσμάτων που υπεισέρχονται στην έκφραση, τότε η εκτέλεση δεν συνοδεύεται από κανένα μήνυμα. Αλλιώς το πρόγραμμα προειδοποιεί ότι αναγκάστηκε να κάνει συμπληρώσεις. Άρα, πρέπει να γνωρίζουμε τι θέλουμε να κάνουμε και σε κάθε περίπτωση να ελέγχουμε τα αποτελέσματα.

Οι δικές μας Συναρτήσεις

Η γλώσσα R είναι εφοδιασμένη με ένα μεγάλο πλήθος συναρτήσεων όπως ήδη αναφέρθηκε. Πολλές φορές όμως χρειαζόμαστε να ορίσουμε και δικές μας συναρτήσεις. Ας υποθέσουμε για παράδειγμα ότι ζητούμε τη μέση τιμή των διαφορών $τοξημ\sqrt{x+0.5} - τοξημ\sqrt{x}$ όπου **x** είναι διάνυσμα.

Ορίζουμε τότε μια συνάρτηση έστω την **msq** ως εξής:

```
> msq <- function(x){
+ x0 <- sqrt(x) ; x1 <- sqrt (x+0.5)
+ y <- asin(x1)-asin(x0)
+ mean(y)
+ }
```

Στην πρώτη γραμμή ορίζεται ότι η **msq** είναι συνάρτηση με όρισμα ένα διάνυσμα x. Στις δύο επόμενες ορίζονται οι βοηθητικές ποσότητες $x_0 = \sqrt{x}$ και $x_0 = \sqrt{x+0.5}$, η διαφορά $y = τοξημx_1 - τοξημx_0$ ορίζεται στην τέταρτη, ενώ στην τελευταία υπολογίζεται η ζητούμενη μέση τιμή. Δίνοντας τώρα:

```
> msq(c(.25,.32,.37,.47,.5))
[1] 0.6059519
```

βρίσκουμε τη μέση τιμή για ένα συγκεκριμένο διάνυσμα.

Είναι φανερό ότι η συνάρτηση **msq** ορίζεται μόνο για τιμές **x** με 0≤**x**≤0.5. Οι συναρτήσεις **sqrt** και **asin** έχουν ενσωματώσει ελέγχους που να σταματούν την εκτέλεση αν δεν επιτρέπεται. Μπορούμε να κάνουμε κάτι ανάλογο και εμείς στη συνάρτηση που ορίσαμε πριν, κάνοντας πριν την εκτέλεση τον έλεγχο ότι όλα τα **x** είναι θετικά και μικρότερα του 0.5. Παρατηρείστε τις δύο επιπλέον γραμμές στη συνάρτηση:

```
> msq <- function(x) {
          if(min(x) < 0 || max(x) > 0.5) {
            stop("vector x is outside (0,.5)")
                }
          x0 <- sqrt(x)
          x1 <- sqrt(x + 0.5)
          y <- asin(x1) - asin(x0)
          mean(y)
      }
```

Αν για οποιοδήποτε λόγο τυπώθηκε κάποιο λάθος στη δημιουργία της συνάρτησης, έχουμε τη δυνατότητα να τη διορθώσουμε. Είναι προτιμότερο να εργαζόμαστε σε **script** παράθυρο, για να διορθώνουμε τα σφάλματα και να εκτελούμε ξανά τη συνάρτηση.

Ως δεύτερο παράδειγμα έστω ότι θέλουμε μία συνάρτηση που να υπολογίζει το στατιστικό για τη διαφορά των μέσων τιμών δύο ανεξάρτητων κανονικών με άγνωστες και ίσες διακυμάνσεις. Γνωρίζουμε ότι το στατιστικό που ελέγχει την υπόθεση H0:μ_1-μ_2=δ, δίνεται από τη σχέση:

$$t = \frac{\bar{x}_1 - \bar{x}_2 - \delta}{s\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}},\quad s^2 = \frac{(n_1-1)s_1^{\,2} + (n_2-1)s_2^{\,2}}{n_1 + n_2 - 2}$$

όπου $n_1, n_2, \bar{x}_1, \bar{x}_2, s_1^{\,2}, s_2^{\,2}$ τα μεγέθη, οι μέσες τιμές και οι διασπορές των δύο δειγμάτων και ακολουθεί κατανομή t με $n_1 + n_2 - 2$ βαθμούς ελευθερίας.

```
> TwoIndPop <- function(x1, x2, dif=0) {
+   n1 <- length(x1); n2 <- length(x2)
+   mx1 <- mean(x1); mx2 <- mean(x2)
+   vx1 <- var(x1); vx2 <- var(x2)
+   vpooled <- ((n1-1)*vx1 + (n2-1)*vx2)/(n1+n2-2)
+   tstat <- (mx1 - mx2-dif)/sqrt(vpooled*(1/n1 + 1/n2))
+   tstat
+ }
```

Έστω για παράδειγμα ότι δύο ανεξάρτητα δείγματα 12 και 10 μαθητών αντίστοιχα έδωσαν ύψη:

x_1=(100, 105, 102, 96, 106, 95, 110, 120, 115, 112, 112, 90) και

x_2=(104, 98, 100, 90, 110, 116, 120, 115, 95, 88)

Εκτελούμε τις εντολές

```
> x1=c(100,105, 102,96,106,95,110,120, 115,112,112,90);x1
> x2=c(104,98,100,90,110,116,120,115,95,88);x2> TwoIndPop(x1,x2)
[1] 0.3826262
> qt(.95,length(x1)+length(x2)-2)
[1] 1.724718
```

Η τελευταία που δίνει την κρίσιμη τιμή σε στάθμη σημαντικότητας 0.05, αποδεικνύει ότι το μέσο ύψος των μαθητών στα δύο δείγματα δεν διαφέρει σημαντικά σε α=0.05.

Συστηματικές ακολουθίες αριθμών

Πολύ συχνά χρειάζεται να ορίσουμε μεταβλητές με τιμές ακέραιους (ή και πραγματικούς) αριθμούς, που να συσχετίζονται μεταξύ τους συστηματικά, π.χ. για τη δημιουργία μεταβλητών κατηγοριοποίησης. Είδαμε ήδη τον τελεστή **":"** που δημιουργεί ακολουθίες αριθμών με πρόσθεση (ή αφαίρεση) μιας μονάδας στον προηγούμενο αριθμό. Μάλιστα, ο τελεστής αυτός έχει μεγαλύτερη προτεραιότητα από τα σύμβολα των πράξεων και έτσι προσφέρει μεγαλύτερη ευελιξία. Συγκρίνετε για παράδειγμα τις εντολές **1:k-1** και **1:(k-1)**

```
> k <- 6;     1:k-1;     1:(k-1)
[1] 0 1 2 3 4 5
[1] 1 2 3 4 5
```

Ένας πιο δυναμικός τρόπος δημιουργίας συστηματικών ακολουθιών αριθμών είναι με τις συναρτήσεις **seq** και **rep**. Η σύνταξη της πρώτης είναι:

```
> seq(from = 1, to = 1, by = ((to - from)/(length.out - 1)),
    length.out = NULL, along.with = NULL, ...)
```

και δημιουργεί την αριθμητική ακολουθία με πρώτο όρο αυτόν που δίνεται στο from, τελευταίο αυτόν που δίνεται στο to, διαφορά αυτόν που δίνεται στο by, συνολικό μήκος αυτόν που δίνεται στο length.out, ή τόσο ώστε να συμφωνεί με το μήκος του διανύσματος που δίνεται στο along.with. Είναι φανερό ότι δεν χρειάζεται να συμπληρωθούν όλες οι παράμετροι, ούτε είναι απαραίτητο να γραφούν τα ονόματα των πεδίων. Αρκεί να καθορίζεται με κόμματα η θέση τους. Όλες οι παρακάτω εντολές δίνουν το ίδιο διάνυσμα 5, 7, 9, 11.

```
> seq(5,11,2)
[1]  5  7  9 11
> seq(5,11,,4)
[1]  5  7  9 11
> seq(,11,2,4)
[1]  5  7  9 11
> seq(5,,2,4)
[1]  5  7  9 11
> seq(5,,2,,10:13)
[1]  5  7  9 11
```

Η σύνταξη της εντολής **rep** είναι:

```
> rep(x, times=r ή v, len=n).
```

Αν στην παράμετρο **times** θέσουμε ένα φυσικό αριθμό r, δημιουργείται ένα διάνυσμα με την επανάληψη r φορών του **x=(x1, x2, …,xt)**. Αν στην παράμετρο **times** θέσουμε ένα διάνυσμα ίδιας διάστασης με το **x**, π.χ. **v=(v1, v2, …,vt)** τότε σχηματίζεται διάνυσμα με την επανάληψη του x1 v1 φορές, του x2 v2 φορές, και τέλος του xt vt φορές. Αν δίνεται η παράμετρος length=n συνεχίζουμε την επανάληψη των στοιχείων του **x** ώσπου να συμπληρωθούν n στοιχεία. Τα παρακάτω παραδείγματα διασαφηνίζουν την εντολή

```
x <- c(1,3,4,7)        # τίθεται x=(1, 3, 4, 7)
y <- rep(x,2)          # τίθεται y=(1, 3, 4, 7, 1, 3, 4, 7)
w <- rep(x,,6)         # τίθεται w=(1, 3, 4, 7, 1, 3)
i <- rep(2,4)          # τίθεται i=(2, 2, 2, 2)
z <- rep(x,i)          # τίθεται z=(1, 1, 3, 3, 4, 4, 7, 7)
u <- rep(x,1:4)        # τίθεται y=(1, 3, 3, 4, 4, 4, 7, 7, 7, 7)
```

Μία πολύ χρήσιμη εφαρμογή των εντολών αυτών έχουμε σε δεδομένα όπως αυτά του διπλανού πίνακα, που παριστάνουν τους χρόνους επιβίωσης (σε 10-ωρα διαστήματα) ζώων σε ένα 3×4 παραγοντικό πείραμα και χρησιμοποιήθηκαν από τους Box και Cox (1964). Εδώ υπάρχει μία μεταβλητή, έστω η **y** με "μήκος" 48 ταξινομημένη σε 3 γραμμές (I, II, III) και 4 στήλες (A, B, C, D) με 4 παρατηρήσεις σε κάθε κελί. Η μεταβλητή y περιέχει πρώτα τις 4 παρατηρήσεις της γραμμής I και της στήλης A, μετά τις 4 παρατηρήσεις της γραμμής I και της στήλης B, κ.ο.κ., μετά τις 4 παρατηρήσεις της γραμμής II και της στήλης A, κ.ο.κ. Για να σχηματίσουμε δύο δείκτριες μεταβλητές που να μας δίνουν τη γραμμή (z1) και τη στήλη (z2) αντίστοιχα της παρατήρησης **y[i]**, χρησιμοποιούμε την εντολή **rep** :

```
> y <- c(0.31, 0.45, 0.46,
0.43, 0.82, 1.1, 0.88, 0.72,
0.43, 0.45, 0.63, 0.76, 0.45,
0.71, 0.66, 0.62, 0.36, 0.29,
0.4, 0.23, 0.92, 0.61, 0.49,
1.24, 0.44, 0.35, 0.31, 0.4,
0.56, 1.02, 0.71, 0.38, 0.22,
0.21, 0.18, 0.23, 0.3, 0.37,
0.38, 0.29, 0.23, 0.25, 0.24,
0.22, 0.3, 0.36, 0.31, 0.33)
> z1 <- rep(1:3, each = 16) ;
z1
 [1] 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2
[31] 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3
> z2 <- rep(rep(1:4, 3), each = 4) ; z2
 [1] 1 1 1 1 2 2 2 2 3 3 3 3 4 4 4 4 1 1 1 1 2 2 2 2 3 3 3 3 4 4
[31] 4 4 1 1 1 1 2 2 2 2 3 3 3 3 4 4 4 4
```

	A	B	C	D
Poison I	0.31	0.82	0.43	0.45
	0.45	1.10	0.45	0.71
	0.46	0.88	0.63	0.66
	0.43	0.72	0.76	0.62
Poison II	0.36	0.92	0.44	0.56
	0.29	0.61	0.35	1.02
	0.40	0.49	0.31	0.71
	0.23	1.24	0.40	0.38
Poison III	0.22	0.3	0.23	0.30
	0.21	0.37	0.25	0.36
	0.18	0.38	0.24	0.31
	0.23	0.29	0.22	0.33

Για παράδειγμα οι 4 παρατηρήσεις της 2ης γραμμής και 3ης στήλης είναι:

```
> y[(z1 == 2) & (z2 == 3)]
[1] 0.44 0.35 0.31 0.40
```

Άλλος τρόπος ορισμού αυτών των μεταβλητών είναι με την **ceiling** :

```
x1 <- ceiling(1:48/16)
x2 <- 1+(ceiling(1:48/4)-1) %% 4
```

Γενικότερα η εντολή **1+(ceiling(1:n/r)-1) %% m** δημιουργεί ένα διάνυσμα μήκους n που αποτελείται από τους αριθμούς 1, 2 έως m καθένας από τους οποίους επαναλαμβάνεται r φορές (πρέπει n να είναι πολλαπλάσιο του mr).

Τυχαίοι Αριθμοί

Οι τυχαίοι αριθμοί είναι πολύ χρήσιμοι στη Στατιστική. Χρησιμοποιούνται κυρίως για την επιλογή τυχαίου δείγματος από κάποιον πληθυσμό. Για τη δημιουργία τους χρησιμοποιούνται συναρτήσεις που εφαρμόζονται σε κάποιες αρχικές τιμές. Οι τιμές αυτές αποθηκεύονται σε ένα διάνυσμα με την ονομασία **.Random.seed** και αντικαθίστανται με νέες κάθε φορά που ζητείται ο υπολογισμός κάποιων τυχαίων αριθμών. Η γλώσσα R εκτός από απλούς τυχαίους αριθμούς (δηλ. ομοιόμορφα κατανεμημένους), μπορεί να μας δώσει και τυχαίους αριθμούς που ακολουθούν άλλες κατανομές , όπως κανονική, t, F, βήτα, γάμα κλπ. Οι κατανομές αυτές δίνονται στον πίνακα 2. Για να πάρουμε τυχαίους αριθμούς από μια κατανομή χρησιμοποιούμε τη συνάρτηση που ορίζεται με το γράμμα r ακολουθούμενο από το κωδικό όνομα της κατανομής. Ανάλογα με την περίπτωση η συνάρτηση απαιτεί παραμέτρους. Έτσι για παράδειγμα, για να πάρουμε 100 τυχαίους αριθμούς από την ομοιόμορφη κατανομή στο διάστημα (1,3), εκτελούμε την εντολή

```
> runif(100,1,3)
```

ενώ η εντολή

```
> mean(rchisq(50,12))
```

δίνει τη μέση τιμή 50 τυχαίων τιμών από την κατανομή χ^2 με 12 βαθμούς ελευθερίας. Να σημειωθεί ότι επανειλημμένη εκτέλεση αυτών των εντολών δίνει διαφορετικά αποτελέσματα. Αν για κάποιο λόγο θέλουμε να χρησιμοποιήσουμε κάποια άλλη φορά τις ίδιες τυχαίες τιμές, χρησιμοποιούμε την **set.seed**, όπως στο παράδειγμα

```
> set.seed(1949)
> mean(rchisq(50, 12))
```

```
[1] 13.14979
```

Οι κατανομές του Επόμενου πίνακα χρησιμοποιούνται και για τον υπολογισμό τιμών της αντίστοιχης συνάρτησης πυκνότητας πιθανότητας, της συνάρτησης κατανομής καθώς επίσης και ποσοστιαίων σημείων της κατανομής. Πράγματι αρκεί να θέσουμε μπροστά από τον κωδικό της κατανομής αντί του "r" ένα από τα γράμματα "d", "p" και "q". Για τον τρόπο σύνταξης των εντολών για τον υπολογισμό αυτών των τιμών, μπορούμε να πάρουμε άμεση βοήθεια με την εντολή help (κωδικό κατανομής ή όνομα συνάρτησης) . Π.χ. **`help (beta)`** ή **`help (rbeta)`** ή **`?rbeta`**, κλπ. Παραδείγματα υπολογισμού τέτοιων τιμών δίνονται παρακάτω:

```
> dnorm(1.25,0,2); pnorm(1.25); qt(.85,10)
[1] 0.1640805
[1] 0.8943502
[1] 1.093058
```

Η πρώτη από αυτές δίνει την πιθανότητα P(Y=1.25)=0.1640805, όπου Y έχει την $N(0,2^2)$ κατανομή, η δεύτερη την πιθανότητα P(Z<1.25), όπου η Z έχει την τυπική κανονική κατανομή και η τρίτη το 85ο εκατοστιαίο σημείο της κατανομής t_{10} με 10 βαθμούς ελευθερίας.

	Κατανομή	Παράμετροι	Αρχικές τιμές
beta	Βήτα	shape1, shape2, ncp	-, -,0
binom	Διωνυμική	size, prob	-, -
cauchy	Caushy	location, scale	0, 1
chisq	χ^2	df, ncp	-,0
exp	Εκθετική	rate	1
f	F	df1, df2, ncp	-, -, -
gamma	Γάμα	shape, rate, scale	-, 1, 1
geom	Γεωμετρική	prob	-
hyper	Υπεργεωμετρική	m, n, k	-, -, -
lnorm	Λογαριθμοκανονική	meanlog,sdlog	0, 1
logis	Λογιστική	location, scale	0, 1
nbinom	Αρν. Διωνυμική	size, prob, mu	-, -, -
norm	Κανονική	mean, sd	0, 1
pois	Poison	lambda	-
t	t του Student	df, ncp	-
unif	Ομοιόμορφη	min, max	0, 1

Διάταξη

Η συνάρτηση **`sort()`** διατάσσει τα στοιχεία διανύσματος σε αύξουσα σειρά, χρησιμοποιώντας τη διάταξητων χαρακτήρων ASCII. Δηλαδή προηγούνται οι αριθμοί, ακολουθούν τα κεφαλαία γράμματα και έπονται τα πεζά. Έτσι έχουμε:

```
> x <- c(3,5,9,1,7,4,15,4,3,9,7) ; x
[1]  3  5  9  1  7  4 15  4  3  9  7
> sort(x)
[1]  1  3  3  4  4  5  7  7  9  9 15,
```

ενώ:

```
> sort(names(grades))
[1] "Alfred" "Barbara" "Denis" "John" "Mary" "Nick"
```

Αν έχουμε αντιστοιχίσει ονόματα στα στοιχεία του x (π.χ. xi), τότε η σειρά των ίσων τιμών διατηρείται. Πράγματι:

```
> names(x) <- paste("x",1:length(x),sep="") ; x
 x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11
  3  5  9  1  7  4 15  4  3   9   7,
```

ενώ

```
> sort(x)
 x4 x1 x9 x6 x8 x2 x5 x11 x3 x10 x7
  1  3  3  4  4  5  7   7  9   9 15
```

Σχετική με την **sort** και πολύ χρήσιμη είναι η συνάρτηση **order**, που δίνει τους δείκτες του αρχικού **x** όπως θα είναι στο τελικό.

```
> order(x)
[1]  4  1  9  6  8  2  5 11  3 10  7
```

Παρατηρείστε ότι ισχύει **x[order(x)]=sort(x)**. Πράγματι:

```
> x[order(x)]
 x4 x1 x9 x6 x8 x2 x5 x11 x3 x10 x7
  1  3  3  4  4  5  7   7  9   9 15
```

ή για να το δούμε καλύτερα δίνουμε την εντολή ισότητας (με δύο =), οπότε:

```
> x[order(x)] == sort(x)
 x4 x1 x9 x6 x8 x2 x5 x11 x3 x10 x7
TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

Ένα άλλο διάνυσμα **y** ίδιου μήκους με το **x** διατάσσεται σύμφωνα με τη διάταξη στο **x** με την εντολή **y[order(x)]**. Παραδείγματος χάριν, αν **y** είναι το διάνυσμα μήκους όσο το **x**, που διαμερίζει το [1,10] σε ίσου εύρους διαστήματα, τότε:

```
> y <- seq(1, 10,  , length(x)) ; y
[1]  1.0  1.9  2.8  3.7  4.6  5.5  6.4  7.3  8.2  9.1 10.0
> y[order(x)]
[1]  3.7  1.0  8.2  5.5  7.3  1.9  4.6 10.0  2.8  9.1  6.4
```

Η συνάρτηση **rev()** αντιστρέφει τη σειρά των στοιχείων κάποιου διανύσματος. Έτσι η εντολή **rev(sort(x))** διατάσσει τα στοιχεία του **x** σε φθίνουσα σειρά.

Η εντολή **rank()** δίνει τις θέσεις των αρχικών στοιχείων στο διαταγμένο διάνυσμα, λαμβανομένου υπόψη των ίσων τιμών. Δηλαδή

```
> rank(x)
[1] 2.5 6.0 9.5 1.0 7.5 4.5 11.0 4.5 2.5 9.5 7.5
> sort(rank(x))
[1] 1.0 2.5 2.5 4.5 4.5 6.0 7.5 7.5 9.5 9.5 11.0
```

Η συνάρτηση **diff()** δίνει ένα διάνυσμα, κατά 1 μικρότερο σε μήκος, που δίνει τις διαφορές των διαδοχικών στοιχείων του αρχικού διανύσματος, κάτι πολύ χρήσιμο στις χρονοσειρές.

Ισχύει **diff(x)=x[-1]-x[-length(x)]** . Π.χ.

```
> diff(x)
2  4 -8  6 -3 11 -11 -1   6  -2
```

Σχετική με τη διάταξη είναι επίσης και η συνάρτηση **match()** , που συγκρίνει τιμές μεταξύ δύο διανυσμάτων. Για παράδειγμα:

```
> z <-c("L","M","M","H","M","H","L","H","M")
> match(z,c("L","M","H"))
[1] 1 2 2 3 2 3 1 3 2
```

ΑΝΑΓΝΩΣΗ ΔΕΔΟΜΕΝΩΝ

Αναφέρθηκε ήδη η συνάρτηση read.table που διαβάζει αρχεία δεδομένων που δίνονται σε στήλες

και χωρίζονται είτε με κενά ή κόμματα ή στηλοθέτες (**csv, txt, dat**, κλπ) και τα εισάγει ως πλαίσια δεδομένων. Υπάρχουν δυνατότητες και βιβλιοθήκες που μπορούμε να κατεβάσουμε για εισαγωγή δεδομένων από άλλα προγράμματα, όπως SPSS, STATA, EXCEL, MINITAB κλπ, τα οποία μπορεί κάποιος που ενδιαφέρεται να τα μελετήσει σε κατάλληλα βιβλία οδηγιών. Ένα αρκετά καλό είναι ενσωματωμένο στη βοήθεια της R (Help, Manuals (in pdf), R Data Import/Export) που ανοίγει το αρχείο **r-data.pdf** με χρήσιμες οδηγίες. Ωστόσο υπάρχουν προβλήματα συμβατότητας με υπολογιστές 64 bits που δεν λύνονται εύκολα. Ευκολότερο συνήθως είναι να αντιγράψουμε τα δεδομένα σε έναν απλό editor και να τα εισάγουμε ως csv (comma separated value) αρχείο.

Εκτός από αυτούς τους τρόπους μπορούμε να εισάγουμε δεδομένα από το πληκτρολόγιο με τη συνάρτηση **scan** που χρησιμοποιείται για να εισάγουμε διανύσματα ή λίστες ή και πιο πολύπλοκα σύνολα δεδομένων, που έχουν δημιουργηθεί με άλλους τρόπους. Για να εισάγουμε ένα αριθμητικό διάνυσμα δίνουμε απλά **scan( )**, οπότε το πρόγραμμα απαντά με τον αριθμό 1: που σημαίνει ότι περιμένει τα διαδοχικά στοιχεία του διανύσματος. Δίνουμε τα στοιχεία αυτά με ένα κενό ανάμεσα. Τελειώνουμε με μία κενή γραμμή. Π.χ. για το διάνυσμα x=(2.3, 12, 3.5, 456, 34, 23.1, 2, 12) γράφουμε:

```
> x <- scan()
1: 2.3 12 3.5 456
5: 34 23.1
7: 2 12
9:
> x
[1]   2.3  12.0   3.5 456.0  34.0  23.1   2.0  12.0
```

Για να εισάγουμε διάνυσμα χαρακτήρων το δηλώνουμε με το **what**:

```
x <- scan(what="")
1: f  F mm 3F 4 F1
7:
> x
[1] "f" "F"   "mm"  "3F"  "4"   "F1"
```

ενώ για λίστες χρησιμοποιούμε την παράμετρο **what** για να δηλώσουμε τη μορφή κάθε συνιστώσας της λίστας. Για παράδειγμα έστω ότι θέλουμε να εισάγουμε τα δεδομένα του διπλανού πίνακα, που αποτελούνται από μία αλφαριθμητική μεταβλητή (**name**), έναν παράγοντα (**group**) και δύο αριθμητικές μεταβλητές (x1 και x2). Ένας τρόπος εισαγωγής των δεδομένων αυτών με το όνομα **heart.list**, είναι ο εξής (η παράμετρος **skip**=1 ήταν εδώ απαραίτητη επειδή το αρχείο είχε ονόματα στηλών στην 1η γραμμή):

Name	Group	x1	x2
John	1	46	18.2
Nick	1	49	21.3
Mary	1	32	16.4
Helen	2	58	20.7
Teo	2	25	18.4
Peter	2	38	20.5

```
> heart.list <- scan("c:/…/heart.dat", skip = 1, what =
+ list(name = "", group = 0, x1 = 0, x2 = 0))
> heart.list
$name:
[1] "John"  "Nick"  "Mary"  "Helen" "Teo"   "Peter"
$group:
[1] 1 1 1 2 2 2
$x1:
[1] 46 49 32 58 25 38
$x2:
[1] 18.2 21.3 16.4 20.7 18.4 20.5
```

Αν θέλουμε να δώσουμε τα δεδομένα από το πληκτρολόγιο δίνουμε την εντολή, π.χ.:

```
> heart1.list <- scan( what=list(name="", group=0,x1=0,x2=0))
```

Οπότε το πρόγραμμα περιμένει στη συνέχεια να δώσουμε 4-δες τιμών από τις οποίες η πρώτη αλφα-

ριθμητική και οι επόμενες τρεις αριθμητικές. Τελειώνουμε με κενή γραμμή. Δηλαδή (μετά την scan) δίνουμε:

```
John 1 46 18.2
Nick 1 49 21.3
Mary 1 32 16.4
Helen 2 58 20.7
Teo 2 25 18.4
Peter 2 38 20.5
```

όπου να προσέξουμε ότι δεν χρειάζονται εισαγωγικά για τα αλφαριθμητικά. Η λίστα που προκύπτει είναι ίδια με την προηγούμενη, δηλ. έχει τέσσερις συνιστώσες με ονόματα **name**, **group**, x1 και x2, όπως αυτά καθορίστηκαν στην παράμετρο **what**.

Για να εισάγουμε πίνακα με τη συνάρτηση **scan** χρησιμοποιούμε τη συνάρτηση **matrix** μαζί με την **scan()** . Έτσι η εντολή:

```
> mat <- matrix(scan( ), byrow=T, ncol=2)
```

μας περιμένει να δώσουμε τα στοιχεία του πίνακα κατά σειρές. Δίνοντας διαδοχικά π.χ. 450 54.6 760 73.4 325 50.3 285 58.1 παίρνουμε τον 4×2 πίνακα:

```
> mat
     [,1] [,2]
[1,]  450 54.6
[2,]  760 73.4
[3,]  325 50.3
[4,]  285 58.1
```

Αν θέλουμε να εισάγουμε τα δεδομένα **heart.dat** ως πλαίσιο δεδομένων, δίνουμε τις εντολές:

```
> names<-c("John","Nick","Mary","Helen","Teo", "Peter")
> colns<-c("Group","x1","x2")
> heart.df <- matrix(scan(),ncol=3,dimnames=list(names,colns))
```

και στη συνέχεια δίνουμε διαδοχικά τις στήλες

```
1 1 1 2 2 2
46 49 32 58 25 38
18.2 21.3 16.4 20.7 18.4 20.5
```

και τελειώνουμε με κενή γραμμή. Η εντολή

```
> heart.df
```

δίνει τον πίνακα με ονόματα γραμμών-στηλών (πλαίσιο δεδομένων).

```
       Group x1   x2
John       1 46 18.2
Nick       1 49 21.3
Mary       1 32 16.4
Helen      2 58 20.7
Teo        2 25 18.4
Peter      2 38 20.5
```

Αν τα δεδομένα είναι γραμμένα με τρόπο ώστε να κατέχουν σταθερό πλήθος θέσεων (**fixed width format**), τότε μπορούμε να χρησιμοποιήσουμε την εντολή **read.fwf** , στην οποία η παράμετρος **widths** μας επιτρέπει να διαβάσουμε τα δεδομένα χωρισμένα σύμφωνα με συγκεκριμένο format. Π.χ. αν στα δεδομένα δίπλα χρησιμοποιήσουμε widths=c(1,3,4), τα δεδομένα διαβάζονται 1 \ .05 \ 0.05 \ 2 \ .05 \ 0.10 \ 3 \.....

```
1.050.052.050.10
3.250.25
4.250.3550.50.75
60.50.85
```

Η επόμενη εντολή

```
> fix.d <- as.list(read.fwf("fixwidth.dat",
+ widths=c(1,3,4),col.names=c("a","b","c")))
```

διαβάζει τα δεδομένα και τα μετατρέπει σε λίστα.

```
> fix.d
$a:
[1] "1" "2" "3" "4" "5" "6"
$b:
[1] 0.05 0.05 0.25 0.25 0.50 0.50
$c:
[1] 0.05 0.10 0.25 0.35 0.75 0.85
```

Αν δεν είχαμε χρησιμοποιήσει την **as.list** το αποτέλεσμα θα ήταν πίνακας με στήλες a,b,c.

Έξοδος δεδομένων – Εκτύπωση

Έστω π.χ. ότι θέλουμε τα αποτελέσματα των εντολών να μην τυπώνονται στην οθόνη αλλά να τυπώνεται σε αρχείο που να μπορούμε να το χρησιμοποιήσουμε οπουδήποτε αλλού. Ένας τρόπος είναι με τη χρήση της εντολής **sink**. Έστω ότι εργαζόμαστε με το αρχείο **airquality** που υπάρχει στα **datasets** της R και περιέχει δεδομένα από την ποιότητα του αέρα στη Νέα Υόρκη (New York Air Quality Measurements). Πληροφορίες για το αρχείο παίρνετε με την εντολή:

```
> ?? airquality
```

ενώ για να τα εισάγουμε και να τα έχουμε διαθέσιμα γράφουμε:

```
> data(airquality)
> attach(airqality)
```

Ελέγχουμε τα περιεχόμενα του αρχείου, π.χ. οι τρεις πρώτες γραμμές του είναι:

```
> airquality[1:3,]
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
```

Γράφοντας, τώρα:

```
> sink("airquality.res")
```

δημιουργείται ένα αρχείο το airquality.res στον τρέχοντα κατάλογο (με την εντολή **setwd**(«διαδρομή») αλλάζουμε τον τρέχοντα κατάλογο), όπου θα γράφονται από δώ και μετά τα αποτελέσματα. Για παράδειγμα οι επόμενες εντολές:

```
> cat("Το Αρχείο airquality \n", "από το data sets \n",
   " ---------------------------- ", "\n")
> airquality
> cat("\n \n summary(airquality) \n")
> summary(airquality)
> plot(Temp,Ozone)
```

εμφανίζουν τα περιεχόμενα του αρχείου ($2^{η}$ εντολή), υπολογίζει περιγραφικά μέτρα θέσης για τις μεταβλητές-στήλες του ($4^{η}$ εντολή) και κάνει μια γραφική παράσταση ($5^{η}$ εντολή). Η $2^{η}$ και η $4^{η}$ εντολή τυπώνουν επικεφαλίδες ώστε να μπορούμε αργότερα να ερμηνεύσουμε τα αποτελέσματα. Το στοιχείο \n μέσα στα εισαγωγικά χρησιμοποιείται για αλλαγή γραμμής. Για να κλείσουμε το προηγούμενο αρχείο και να επανέλθουμε στην οθόνη, δίνουμε:

```
> sink()
```

και τώρα το αρχείο **airquality.res** μπορούμε να το ανοίξουμε με ένα οποιοδήποτε επεξεργαστή (π.χ. Notepad) και να δούμε τι γράφτηκε. Θα παρατηρήσουμε ότι η τελευταία εντολή δεν προσθέτει τίποτε στο αρχείο, αλλά ανοίγει ένα παράθυρο γραφικών το οποίο μπορεί να αποθηκευτεί μόνο του.

Αν θέλουμε να προσθέσουμε στο προηγούμενο αρχείο και άλλα αποτελέσματα χωρίς να χάσουμε τα προηγούμενα χρησιμοποιούμε την παράμετρο **append=T**, δηλ.:

```
> sink("airquality.res ",append=T)
> cat("\n \n variance of airquality \n")
```

```
> var(airquality)
> sink()
```

Αφού κλείσει το εξωτερικό αρχείο τα αποτελέσματα τυπώνονται πάλι στην κονσόλα. Αν για κάποιο λόγο δεν τυπώνονται στην οθόνη τρέξτε την αυτόματη εντολή για εκτύπωση στην οθόνη:

```
> sink(file = NULL, append = FALSE,
+ type = c("output", "message"),split=F)
```

Ας υποθέσουμε τώρα ότι ζητούμε να αποθηκεύσουμε σε ένα εξωτερικό αρχείο μια συνάρτηση την **larger**, που υπολογίζει το διάνυσμα $\max_{i=1,\ldots,n}(x_i, y_i)$ για δοσμένα διανύσματα **x** και **y**. (λάβετε υπόψιν ότι αν $x=(x_1,x_2,\ldots,x_n)$ και a ένα διάνυσμα μήκους επίσης n με στοιχεία τις λογικές τιμές T,F, τότε x[a] είναι το διάνυσμα με μόνο εκείνα τα στοιχεία του x που αντιστοιχούν στα T του a).

```
> larger <- function(x, y)  {
+ y.is.bigger <- y > x
+ x[y.is.bigger] <- y[y.is.bigger]
+ x
+ }
```

Εκτελώντας τις εντολές:

```
> x<-c(12.3, 32, 15, 21.2, -2.3, 0)
> y<-c(22.5, 25, -24, 2.7, 1, 2.4)
> larger(x,y)
[1] 22.5 32.0 15.0 21.2  1.0  2.4
```

βλέπουμε πως δουλεύει η συνάρτηση **larger**.

Εκτελώντας την εντολή

```
> dump("larger","larger.func")
```

το αρχείο **larger.func** θα περιέχει τον ορισμό της συνάρτησης. Αν **larg.df** είναι ένα πλαίσιο δεδομένων που περιέχει στις δύο πρώτες στήλες τα δοσμένα διανύσματα **x**, **y** και στην τρίτη το διάνυσμα **larger(x,y)**, για να στείλουμε εκτός από τον ορισμό της συνάρτησης και το **larg.df** στο αρχείο **larger.func**, εκτελούμε διαδοχικά:

```
> larg.df <- data.frame(x,y,larger(x,y)); larg.df
> dump(c("larger","larg.df"),"larger.func")
```

Η ίδια συνάρτηση χρησιμοποιείται και για εξαγωγή με μορφή ASCII δεδομένων για ενδεχόμενη χρήση από άλλα προγράμματα. Έτσι, για να εξάγουμε σε μορφή ASCII το πλαίσιο δεδομένων state.x77, που περιέχει στοιχεία για τον πληθυσμό, εισόδημα, εγκληματικότητα κλπ για το έτος 1977 για τις πολιτείες της Αμερικής, γράφουμε την εντολή:

```
> dump("state.x77","state1.dat")
```

Η διαφορά είναι στον τρόπο καταγραφής των στοιχείων στο αρχείο. Ένας άλλος τρόπος είναι με την εντολή:

```
> dput(state.x77,"state2.dat")
```

Παρατηρείστε ότι τώρα το state.x77 δεν είναι σε εισαγωγικά, όπως ήταν στις προηγούμενες εντολές. Με την εντολή:

```
> state <- dget("state2.dat")
```

ξαναφέρνουμε το αρχείο στην R. Μελετώντας τη δομή του αρχείου "state2.dat" έχουμε άλλο ένα τρόπο να διαβάζουμε πλαίσια δεδομένων στην R. Η δομή του αρχείου είναι:

```
> structure(.Data = c(x1,x2,…,xnm), .Dim=c(n,m),
+ .Dimnames=list(c("A1",…,"An"),c("B1",…,"Bm")))
```

όπου n είναι οι γραμμές με ονόματα A1, A2,…,An, m είναι οι στήλες με ονόματα B1, B2,…,Bm. και x1,x2,…,xnm, τα δεδομένα γραμμένα κατά στήλες, δηλαδή τα πρώτα n στοιχεία αφορούν τη στήλη B1, τα επόμενα n αφορούν τη στήλη B2, κλπ.

Ανάλογα με την read.table υπάρχει και η write.table που αποθηκεύει πίνακες ή πλαίσια δεδομένων σε αρχεία csv. Η **write.table** εξάγει σε ASCCII μορφή. Για παράδειγμα η εντολή

```
> write.table(airquality, file="air.dat")
```

εξάγει το ενσωματωμένο αρχείο airquality στο αρχείο που καθορίζεται.

Η **write.table** εξάγει σε ASCCII μορφή και αποτελέσματα σύνθετων εντολών που έχουν τη μορφή πίνακα. Για παράδειγμα η εντολή

```
> write.table(coef(lm(Ozone~Solar.R+Wind+Temp)),
+ file="myresults1.csv")
```

εξάγει σε αρχείο ASCII τους εκτιμώμενους συντελεστές μιας παλινδρόμησης, ενώ η:

```
> read.table("myresults1.csv")
```

επαναφέρει τους συντελεστές στην R. Όμοια, η εντολή:

```
> write.table(cor(na.omit(airquality)[,1:4]),
file="myresults2.csv")
```

εξάγει σε αρχείο ASCII τον πίνακα συσχετίσεων τεσσάρων μεταβλητών, ενώ η:

```
> read.table("myresults2.csv")
```

Εισάγει τον πίνακα στην R.

ΔΙΑΝΥΣΜΑΤΑ ΚΑΙ ΠΙΝΑΚΕΣ

Χρησιμοποιώντας τους δείκτες των πινάκων και τις αριθμητικές πράξεις που περιγράφηκαν στις προηγούμενες παραγράφους, μπορούμε να κάνουμε όλες τις πράξεις των πινάκων. Υπάρχουν όμως πιο ευέλικτες συναρτήσεις που μας εξυπηρετούν όταν έχουμε πράξεις με διανύσματα και πίνακες.

Οι συναρτήσεις crossprod και outer

Η πρώτη υπολογίζει το γινόμενο $X^{\mathrm{T}}Y$, όπου X, Y, διανύσματα ή πίνακες. Η εντολή γράφεται **crossprod(X,Y)** και προϋποθέτει οι πίνακες X, Y έχουν ίδιο πλήθος στηλών. Θα μπορούσαμε να υπολογίσουμε το ίδιο αποτέλεσμα με τον τελεστή %*% πολλαπλασιασμού πινάκων. Οι εντολές **crossprod(X,Y)** και **t(X)%*%Y** έχουν το ίδιο αποτέλεσμα. Για το γινόμενο $X^{\mathrm{T}}X$ γράφουμε απλά **crossprod(X)**.

Για παράδειγμα εισάγουμε το αρχείο salary.df και το σώζουμε ως πλαίσιο δεδομένων salary.df.. Ορίζουμε πίνακα X, που περιέχει τις στήλες 1, exper, educ, manag και διάνυσμα στήλη Y, με τη στήλη salary. Υπολογίζουμε στη συνέχεια τα γινόμενα $X^{\mathrm{T}}Y$, $M = X^{\mathrm{T}}X$. Αυτά γίνονται με τις εντολές:

```
> salary.df=read.table("salary.df")
> attach(salary.df)
> X <- matrix(cbind(1,exper,educ,manag),ncol=4);X
> Y <- matrix(salary)
> B <- crossprod(X,Y);B
          [,1]
[1,]  38082.40
[2,] 303576.72
[3,]  78473.22
[4,]  21577.49
> M <- crossprod(X);M
     [,1] [,2] [,3] [,4]
[1,]   55  399  110   26
[2,]  399 4245  768  182
[3,]  110  768  252   55
[4,]   26  182   55   26
```

Η συνάρτηση **outer** υπολογίζει το γινόμενο $x \cdot y^{\mathrm{T}}$ για δύο οποιαδήποτε διανύσματα x, y. Χρησιμοποιείται όμως και για οποιαδήποτε άλλη διμελή πράξη μεταξύ όλων των στοιχείων των δύο

διανυσμάτων. Αν η πράξη είναι ο πολλαπλασιασμός, τότε δεν χρειάζεται να δηλωθεί. Έτσι η εντολή **g <- outer(a, b, "*")** (και απλούστερα) **g <- outer(a, b),** δίνει τον πίνακα $g = a^{\mathrm{T}} b$. Το ίδιο αποτέλεσμα το παίρνουμε με τις εντολή **matrix (a, length(a)) %*% b** . Για παράδειγμα:

```
> x <- 1:5
> y <- 6:1
> outer(x,y)
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    6    5    4    3    2    1
[2,]   12   10    8    6    4    2
[3,]   18   15   12    9    6    3
[4,]   24   20   16   12    8    4
[5,]   30   25   20   15   10    5
```

Παρατηρείστε ότι **outer(x,y)** είναι εν γένει διάφορο του **outer(x,y)**.

Με τη συνάρτηση **outer** υπολογίζεται εύκολα και ο πίνακας με στοιχεία $f(x_i,y_j)$, όπου f(x,y) συνάρτηση δύο μεταβλητών. Για παράδειγμα, αν είναι $f(x,y) = e^{-(x+y)} \sin(y)$, δίνουμε διαδοχικά τις εντολές:

```
> f <- function (x, y) exp(-x-y)* sin(y)
> z <- outer(x, y, f)
```

Με την εντολή

```
> outer(month.name, 2008:2011, paste)
```

παίρνουμε όλους τους μήνες των ετών 2008 έως 2011.

Αν η k-διάστατη τ.μ. $(X_1, X_2,\ldots,X_k)$ έχει την εμπειρική πολυωνυμική κατανομή με πιθανότητες $p_1, p_2,\ldots,p_k$, $(p_1+ p_2+\ldots+ p_k \le 1)$ και μέγεθος n, τότε είναι γνωστό ότι $E(X_i)=np_i$, $Var(X_i)= np_i\cdot(1-p_i)$ και $Cov(X_i,X_j)=-np_ip_j$, i,j =1,2,..k. Έστω για παράδειγμα k=4, p=(0.30, 0.15, 0.20, 0.25) και n=3. Η συνάρτηση **outer** μαζί με τη συνάρτηση **diag(M)** μας δίνουν τη δυνατότητα να σχηματίσουμε τον πίνακα διασπορών - συνδιασπορών για την κατανομή αυτή. Έχουμε:

```
> p <- c(0.30, 0.15, 0.20, 0.25) ; p
[1] 0.30 0.15 0.20 0.25
> VC <-  -3*outer(p,p) ; VC
       [,1]    [,2]  [,3]    [,4]
[1,] -0.270 -0.1350 -0.18 -0.2250
[2,] -0.135 -0.0675 -0.09 -0.1125
[3,] -0.180 -0.0900 -0.12 -0.1500
[4,] -0.225 -0.1125 -0.15 -0.1875
> diag(VC) <- 3*p*(1-p) ; VC
       [,1]    [,2]  [,3]    [,4]
[1,]  0.630 -0.1350 -0.18 -0.2250
[2,] -0.135  0.3825 -0.09 -0.1125
[3,] -0.180 -0.0900  0.48 -0.1500
[4,] -0.225 -0.1125 -0.15  0.5625
```

Οι συναρτήσεις apply και sweep

Η συνάρτηση **apply** δίνει το αποτέλεσμα της εφαρμογής μιας συνάρτησης, όπως π.χ. της **mean** ή της **var** σε ένα πίνακα αθροίζοντας όλα τα στοιχεία του πίνακα που έχουν δοθείσες διαστάσεις. Για παράδειγμα έστω X ένας p×q×r×s πίνακας αριθμητικών δεδομένων. Με X_{ijkl} συμβολίζουμε το στοιχείο που βρίσκεται στο ijkl κελί του πίνακα. Με $\bar{X}_{i\cdot k\cdot}$ εννοούμε το μέσο όρο όλων των στοιχείων που βρίσκονται στα κελιά με πρώτο δείκτη i και τρίτο δείκτη k. Τέτοιοι μέσοι όροι υπάρχουν για όλους τους συνδυασμούς των δεικτών i, k. Αυτοί οι μέσοι βρίσκονται με την εντολή apply. Αν Aarray είναι ο πίνακας

πολλαπλής εισόδου που ορίστηκε στη σελίδα 8, τότε η εντολή:

```
> meanAar13 <- apply(Aarray, c(1,3),mean); meanAar13
     [,1] [,2] [,3]
[1,]    4   14  114
[2,]    5   15  115
```

δίνει την τιμή $\bar{X}_{i.k} = \frac{1}{4}\begin{pmatrix} \sum_{j=1}^{4} X_{1j1} & \sum_{j=1}^{4} X_{1j2} & \sum_{j=1}^{4} X_{1j3} \\ \sum_{j=1}^{4} X_{2j1} & \sum_{j=1}^{4} X_{2j2} & \sum_{j=1}^{4} X_{2j3} \end{pmatrix} = \begin{pmatrix} 4 & 14 & 114 \\ 5 & 15 & 115 \end{pmatrix}$

Όμοια αν X ο πίνακας που ορίστηκε στη σελίδα 25, η εντολή:

```
> apply(X, 2, mean, trim=.25)
```

δίνει τους αποκομμένους κατά 25% μέσους (**trimmed means**) της 2ης στήλης, δηλ:

```
[1] 1.0000000 6.5172414 2.0000000 0.4482759
```

Όμοια, η εντολή:

```
> apply (X, 2, sort)
```

διατάσσει τα στοιχεία κάθε στήλης του X.

Αν έχουμε βρει τους μέσους ή οποιαδήποτε άλλα στατιστικά του πίνακα **x** με την **apply** μπορούμε να τα αφαιρέσουμε ή να τα προσθέσουμε, διαιρέσουμε κλπ από τα αντίστοιχα στοιχεία του πίνακα με τη συνάρτηση **sweep**.

Για παράδειγμα έστω τα δεδομένα OECDGas που βρίσκονται στο πακέτο AER και αφορούν την κατανάλωση βενζίνης στις χώρες του OECD (Organisation for Economic Co-operation and Development) για τα έτη 1960-1978 (δόθηκαν το 2002 από τον B.H. Baltagi). Για να τα εμφανίσουμε εκτελούμε τις εντολές:

```
> library(AER) # άνοιγμα του πακέτου AER
> data("OECDGas")
> OECDGas        # εμφάνιση των δεδομένων
```

Αν το πακέτου AER δεν υπάρχει στον υπολογιστή σας τρέξτε την επόμενη εντολή

```
> install.packages("AER",  repos = "http://cran.r-project.org",
+        destdir = NULL, dependencies = TRUE)
```

που θα εγκαταστήσει και ότι άλλο απαιτείται.

Μια άλλη εφαρμογή αυτών των συναρτήσεων είναι να σχηματίσουμε τις τυποποιημένες τιμές για ένα πλήθος μεταβλητών (με ίδιο μήκος). Για παράδειγμα χρησιμοποιούμε στα επόμενα τα δεδομένα από το προηγούμενο αρχείο που αφορούν την Ελλάδα. Εκτελούμε διαδοχικά:

```
> grGas <- OECDGas[OECDGas$country=="Greece",]
```

που επιλέγει από το αρχείο μόνο τις εγγραφές που ως country έχουν Greece.

```
> grGasm <- as.matrix(grGas[-c(1:2)]);grGasm
```

που μετατρέπει το προηγούμενο σε πίνακα.

```
> grm <- apply(grGasm,2,mean);grm
       gas     income      price       cars
4.87867884 -6.60637349 -0.03391043 -10.78206599
```

που υπολογίζει τη μέση τιμή για κάθε στήλη του προηγούμενου πίνακα.

```
> grv <- apply(grGasm,2,var);grv
       gas     income      price       cars
0.06485907 0.10933968 0.02729951 0.70368423
```

που υπολογίζει τις διασπορές για κάθε στήλη του προηγούμενου πίνακα. Η επόμενη εντολή «απαλείφει» από κάθε στοιχείο μιας στήλης του πρώτου πίνακα (εδώ του grGasm) τα στοιχεία των αντίστοιχων στοιχείων του τρίτου αντικειμένου (εδώ του διανύσματος grm).

```
> grdevs <- sweep(grGasm, 2, grm);grdevs
```

Η επόμενη εντολή «απαλείφει» από κάθε στοιχείο μιας στήλης του πρώτου πίνακα (εδώ του grdevs) τα στοιχεία των αντίστοιχων στοιχείων του τρίτου αντικειμένου (εδώ του διανύσματος sqrt(grv)). Το «α-

παλείφει» αυτόματα εννοεί αφαίρεση. Εδώ θέλαμε διαίρεση, γι αυτό το δηλώνουμε στην εντολή.

```
> grstnd <- sweep(grdevs, 2, sqrt(grv), "/");grstnd
```

Για να ελέγξουμε ότι αυτά που υπολογίσαμε είναι οι τυπικές τιμές ((x-μ)/σ) υπολογίζουμε τις μέσες τιμές και τις διασπορές του τελευταίου πίνακα (grstnd) και που αναμένεται να είναι 0 και 1, αντίστοιχα για κάθε στήλη.

```
> round(apply(grstnd,2,mean),10)
   gas income  price   cars
     0      0      0      0
> apply(grstnd,2,var)
   gas income  price   cars
     1      1      1      1
```

Συναρτήσεις που εφαρμόζονται σε πίνακες

Η συνάρτηση **solve** έχει διπλή χρήση· αντιστρέφει πίνακες και λύνει γραμμικά συστήματα. Έτσι **solve(A)** είναι ο αντίστροφος του A (όταν υπάρχει) και **solve(A,b)** είναι η λύση του συστήματος $A \cdot x = b$. Π.χ.

```
> A <- matrix(10*round(runif(9),1),ncol=3);A
     [,1] [,2] [,3]
[1,]    1    8    4
[2,]    4    2    3
[3,]    5    4    4
> b <- matrix(10*round(runif(3),1),ncol=1);b
     [,1]
[1,]    9
[2,]    6
[3,]    9
> solve(A)
            [,1]      [,2]      [,3]
[1,] -0.33333333 -1.333333  1.333333
[2,] -0.08333333 -1.333333  1.083333
[3,]  0.50000000  3.000000 -2.500000
> solve(A,b)
     [,1]
[1,]    1
[2,]    1
[3,]    0
```

Αν ο πίνακας R είναι άνω τριγωνικός (τα στοιχεία κάτω από την κύρια διαγώνιο είναι 0) και ενδιαφερόμαστε για τη λύση του συστήματος $\boldsymbol{Rx = b}$, τότε χρησιμοποιείται η εντολή **backsolve(R,b)** , ενώ αν ο πίνακας L είναι κάτω τριγωνικός και ενδιαφερόμαστε για τη λύση του συστήματος $\boldsymbol{Lx = b}$, τότε χρησιμοποιείται η εντολή **forwardsolve(L,b)** . Π.χ.

```
> R=A;L=A
> R[2,1]=0;R[3,1]=0;R[3,2]=0;R
     [,1] [,2] [,3]
[1,]    1    8    4
[2,]    0    2    3
[3,]    0    0    4
> L[1,2]=0;L[1,3]=0;L[2,3]=0;L
     [,1] [,2] [,3]
[1,]    1    0    0
```

```
[2,]    4    2    0
[3,]    5    4    4
> br<-backsolve(R,b) ;br
      [,1]
[1,]  3.000
[2,] -0.375
[3,]  2.250
> R %*% br
     [,1]
[1,]    9
[2,]    6
[3,]    9
> bl<-forwardsolve(L,b);bl
     [,1]
[1,]    9
[2,]  -15
[3,]    6
> L %*% bl
     [,1]
[1,]    9
[2,]    6
[3,]    9
```

Η εντολή **eigen(A)** υπολογίζει τις ιδιοτιμές και τα ιδιοδιανύσματα του πίνακα Α. Το αποτέλεσμα είναι μία λίστα με συνιστώσες values και vectors. Αν θέλουμε μόνο τις ιδιοτιμές δίνουμε την τιμή Τ στην παράμετρο only.values.

```
> eigen(A, only.values=T)$values
[1] 11.455972 -4.206984 -0.248988

> eigen(A)
$values
[1] 11.455972 -4.206984 -0.248988

$vectors
           [,1]       [,2]       [,3]
[1,] -0.6022593 -0.8579110 -0.4282721
[2,] -0.4614360  0.3927720 -0.3497478
[3,] -0.6514296  0.3312383  0.8332223
```

Για τον υπολογισμό της ορίζουσας ενός πίνακα, το πρόγραμμα διαθέτει τις συναρτήσεις **det** και **determinant** μπορεί όμως να χρησιμοποιηθεί και η συνάρτηση **eigen**, διότι όπως γνωρίζουμε το γινόμενο των ιδιοτιμών ισούται με την ορίζουσα. Για το ίχνος του πίνακα Μ, μπορούμε να αθροίσουμε τα διαγώνια στοιχεία ή τις ιδιοτιμές του. Άλλες χρήσιμες εντολές στους πίνακες:

row(M), όπου το (i,j) στοιχείο είναι i για όλα τα j,

col(M), όπου το (i,j) στοιχείο είναι j για όλα τα i,

rbind(a,b), που συνδέει τα διανύσματα ή πίνακες a, b που έχουν ίδιο πλήθος στηλών κατά γραμμές και

cbind(a,b), που συνδέει τα διανύσματα ή πίνακες a, b που έχουν ίδιο πλήθος γραμμών κατά στήλες.

Για παράδειγμα, η εντολή **cbind(1,X)** προσθέτει μία στήλη μονάδες πριν από τις στήλες του Χ, η εντολή **rbind(M,v1,v2)** επαυξάνει τον πίνακα Μ με δύο ακόμη γραμμές (τις v1, v2), η εντολή **X[row(X)>col(X)]<-0** μηδενίζει τα στοιχεία του πίνακα που είναι κάτω από την κύρια διαγώνιο

(ώστε να γίνει κάτω τριγωνικός), η εντολή **X[row(X)-col(X)==1]** δίνει την υποδιαγώνιο του X που είναι παράλληλος προς την κύρια διαγώνιο και κάτω απ' αυτήν (οι δείκτες έχουν διαφορά 1) και η εντολή **X[row(X)==1]<-v1** αντικαθιστά την πρώτη γραμμή του πίνακα X με το διάνυσμα v1.

Η ανάλυση ενός μη-αρνητικά ορισμένου συμμετρικού πίνακα A με τη μορφή $A = U^{\mathrm{T}} \cdot U$, όπου U άνω τριγωνικός (Choleski decomposition) πετυχαίνεται με την εντολή **chol(A)**. Π.χ.

```
> set.seed(200)
> x <- matrix(sample(-3:3, size = 9, replace = T), nrow = 3,
+ ncol = 3) ;  x
     [,1] [,2] [,3]
[1,]    0    1    1
[2,]    1    1   -3
[3,]    1    2    0


> M <- x %*% t(x) ;  M
     [,1] [,2] [,3]
[1,]    2   -2    2
[2,]   -2   11    3
[3,]    2    3    5
```

Ο M που ορίστηκε με τον παραπάνω τρόπο είναι θετικά ορισμένος και συμμετρικός. Έτσι:

```
> U <- chol(M)
> round(U, 3)
      [,1]   [,2]   [,3]
[1,] 1.414 -1.414 1.414
[2,] 0.000  3.000 1.667
[3,] 0.000  0.000 0.471
```

Για έλεγχο εκτελούμε

```
> t(U) %*% U
     [,1] [,2] [,3]
[1,]    2   -2    2
[2,]   -2   11    3
[3,]    2    3    5
```

και βρίσκουμε πάλι τον πίνακα M.

Με την εντολή **svd(X)**, όπου X πίνακας διαστάσεως n×p, υπολογίζονται δύο πίνακες U και V διαστάσεων n×min(n,p) και p×min(n,p) αντίστοιχα, των οποίων οι στήλες είναι ορθοκανονικά διανύσματα και ένας διαγώνιος πίνακας Λ με στοιχεία σε φθίνουσα σειρά, έτσι ώστε $X = U \cdot \Lambda \cdot V^{\mathrm{T}}$. Η ανάλυση αυτή λέγεται singular value decomposition και η ύπαρξή της αποδείχθηκε το 1989 από τους Golub και Van Loan. Το αποτέλεσμα της εντολής είναι λίστα με τρεις συνιστώσες u, v και d. Το d είναι η διαγώνιος του Λ και έχει τόσα μη–μηδενικά στοιχεία όσα η τάξη (rank) του πίνακα X. Αν δεν απαιτείται ο υπολογισμός του πίνακα U ή του V δίνουμε nu=0 ή nv=0.

```
> set.seed(200)
> x <- matrix(sample(-3:6,size = 12,replace = T), nrow = 3) ; x
     [,1] [,2] [,3] [,4]
[1,]    2    3    4   -1
[2,]    2    3   -3    1
[3,]    2    5    2    3
> svdx <- svd(x) ;  svdx
$d:
[1] 8.045104 5.037190 2.214277
```

```
$u
           [,1]       [,2]       [,3]
[1,] 0.5422372 -0.6056377  0.5823931
[2,] 0.3008439  0.7871285  0.5384437
[3,] 0.7845201  0.1167548 -0.6090128

$v
           [,1]       [,2]       [,3]
[1,] 0.4046191  0.1184174  0.4622946
[2,] 0.8019590  0.2239833  0.1433635
[3,] 0.3524451 -0.9033661 -0.2275165
[4,] 0.2625407  0.3460324 -0.8449656
```

Για έλεγχο εκτελούμε το επόμενο γινόμενο

```
> x1 <- svdx$u %*% diag(svdx$d) %*% t(svdx$v)
> round(x1 - x, 3)
     [,1] [,2] [,3] [,4]
[1,]    0    0    0    0
[2,]    0    0    0    0
[3,]    0    0    0    0
```

Για τον έλεγχο της ορθοκανονικότητας των στηλών των u και v, εκτελούμε

```
> round(t(svdx$u) %*% svdx$u, 6)
> round(t(svdx$v) %*% svdx$v, 6)
```

που δίνουν τον μοναδιαίο πίνακα.

Η εντολή **qr(M)** υπολογίζει δύο πίνακες· τον Q με στήλες ορθοκανονικά διανύσματα και τον R έναν άνω τριγωνικό, ώστε να ισχύει $M = QR$ (QR-decomposition). Το αποτέλεσμα είναι λίστα. Αν συμβολίσουμε με L τη λίστα αυτή (**L<-qr(M)**), τότε δίνοντας **qr.Q(L)** παίρνουμε τον πίνακα Q, ενώ δίνοντας **qr.R(L)** παίρνουμε τον πίνακα R. Αρκετές συναρτήσεις εφαρμοζόμενες στη λίστα αυτή δίνουν αποτελέσματα χρήσιμα στην ανάλυση παλινδρόμησης. Για παράδειγμα η **qr.resid(L,y)** δίνει το διάνυσμα υπολοίπων (residuals) μετά την προβολή του διανύσματος y στο χώρο των στηλών του πίνακα M. Άλλες σχετικές συναρτήσεις είναι η **qr.fitted** και η **qr.coeff**.

Για παράδειγμα μπορούμε να εκτελέσουμε τις επόμενες εντολές

```
> L <- qr(x) ; L
> q <- qr.Q(L);q
> r <- qr.R(L);r
> round(q%*%r,6)
> round(t(q)%*%q, 6)
```

όπου x ο πίνακας που ορίστηκε προηγούμενα.

ΓΡΑΦΗΜΑΤΑ

Η γλώσσα R διαθέτει μία μεγάλη ποικιλία τρόπων για τη δημιουργία γραφημάτων οποιουδήποτε τύπου. Η βασική εντολή κατασκευής γραφημάτων είναι η **plot()** η οποία χρησιμοποιώντας διάφορες παραμέτρους που καθορίζονται από τη συνάρτηση **par()**, μπορεί να σχηματίσει μια μεγάλη ποικιλία γραφημάτων. Υπάρχουν όμως και πολλά άλλα πακέτα, όπως το **lattice**, **igraph**, κλπ που βελτιώνουν και επεκτείνουν τις ικανότητες του πακέτου R για δημιουργία γραφικών παραστάσεων, γραφημάτων κλπ

Συσκευές γραφημάτων

Η κατασκευή ενός γραφήματος, η αντιγραφή του σε κάποιο αρχείο ή η εκτύπωσή του σε εκτυ-

πωτή απαιτεί την ύπαρξη κατάλληλου παραθύρου όπου να επιτρέπεται ο σχεδιασμός γραμμών, καμπύλων, συμβόλων κλπ. Με την εκτέλεση κάποιας βασικής εντολής κατασκευής γραφήματος π.χ. **plot(…)**, **barplot(…)**, κλπ ανοίγει στην οθόνη του υπολογιστή μας ένα παράθυρο που φέρει το όνομα **R Graphics: Device 2 (ACTIVE)** και είναι ο χώρος (η συσκευή) όπου έχει σχηματιστεί το γράφημα που ζητήθηκε.

Για παράδειγμα με την εντολή:

```
> plot(rnorm(10))
```

σχηματίζει στην ενεργό συσκευή το γράφημα 10 τυχαίων τιμών της τυπικής κανονικής κατανομής, ενώ η εντολή:

```
> plot(1:10,runif(10))
```

αντικαθιστά στην ενεργό συσκευή το προηγούμενο γράφημα με το γράφημα 10 τυχαίων τιμών της ομοιόμορφης κατανομής.

Μπορούμε να έχουμε έλεγχο στις συσκευές γραφημάτων οι οποίες εκκινούν με τις εντολές **x11()** ή **X11()**, ή **win.graph()**, ενώ κλείνουν με την **dev.off()**. Εκτελέστε το παράδειγμα:

```
> x11()     # ανοίγει μία συσκευή (η 2 που είναι ενεργός)
> plot(1:10) # γράφημα στη συσκευή 2
> x11()     # ανοίγει μία άλλη συσκευή (η 3 που είναι ενεργός, ενώ η 2 ανενεργός))
> plot(rnorm(10)) # γράφημα στη συσκευή 2
> dev.set(dev.prev()) # κάνουμε ενεργό τη συσκευή 2 και ανενεργό την 3
> abline(0,1) # through the 1:10 points # προσθέτουμε την ευθεία y=x
> dev.set(dev.next()) # κάνουμε ενεργό τη συσκευή 3 και ανενεργό την 2
> abline(h=0, col="gray") # προσθέτουμε την ευθεία y=0, με χρώμα γκρι
> dev.set(dev.prev()) # κάνουμε ενεργό τη συσκευή 2 και ανενεργό την 3
> dev.off(); dev.off()# κλείνουμε και τις δύο συσκευές
```

Αν θέλουμε να τυπώσουμε το γράφημα κατευθείαν σε αρχείο, ανοίγουμε μία συσκευή postscript και καθορίζουμε το όνομα του αρχείου, που πρέπει να έχει κατάληξη **ps**. Το αρχείο αυτό μπορούμε στη συνέχεια να το ανοίξουμε με έναν Postscript Viewer (π.χ. GSview) ή να το μετατρέψουμε πρώτα σε **pdf** (π.χ. με το Acrobat Distiller). Εκτελέστε το παράδειγμα:

```
> postscript(file="Plots.ps")  # ανοίγει η συσκευή Postscript που
                               # θα τυπώνει τα επόμενα στο αρχείο Plots.ps
> plot(1:10,runif(10))    # ένα γράφημα
> dev.off()# κλείνουμε τη συσκευή
```

Μία άλλη συσκευή για εκτύπωση σε αρχείο είναι στη βιβλιοθήκη Cairo, όπου μπορούμε να τυπώσουμε σε ps αλλά επίσης και σε pdf, όπως βλέπουμε στο παράδειγμα:

```
> library(Cairo)    # αν δεν υπάρχει τρέξτε την επόμενη εντολή
   #install.packages("Cairo",  repos = "http://cran.r-project.org",
   #destdir = NULL, dependencies = TRUE)
> cairo_ps(file="RPlots.ps")  # ανοίγει η συσκευή Postscript του πακέτου
                          # Cairo που θα τυπώνει τα επόμενα στο αρχείο RPlots.ps
> plot(rnorm(1000),rnorm(1000))   # ένα γράφημα
> dev.off()  # κλείνουμε τη συσκευή
```

Για εκτύπωση σε pdf

```
> cairo_pdf(file="Rplots.pdf")  # ανοίγει η συσκευή pdf του πακέτου
                    # Cairo που θα τυπώνει τα επόμενα στο αρχείο RPlots.pdf
> plot(rnorm(1000),rnorm(1000)) # ένα γράφημα
> dev.off()# κλείνουμε τη συσκευή
null device
          1
```

Σε όλες αυτές τις συσκευές όταν τις κλείνουμε (όλες) πρέπει να γράφει null device.

Οι βασικές εντολές για κατασκευή γραφημάτων

Όνομα	Παράμετροι	Γράφημα
plot	x, y	Σχηματίζει το διάγραμμα διασποράς της y ως προς x.
lines points		Προσθέτει γραμμές στο προηγούμενο γράφημα. Καθορίζει το χαρακτήρα που τυπώνει στα σημεία. Υπάρχουν 18 χαρακτήρες (pch=n, n=1,2,…,18)
hist	x	Σχηματίζει το ιστόγραμμα του x
barplot	height	Σχηματίζει ραβδόγραμμα με ύψη σύμφωνα με το height
tsplot tspoints tslines	x	Σχηματίζει το γράφημα της χρονοσειράς x Καθορίζονται τα σημεία χρονοσειράς Καθορίζεται ο τύπος των γραμμών της χρονοσειράς
matplot	X, Y	Σχηματίζει διαγράμματα διασποράς των στηλών του πίνακα X προς τις αντίστοιχες του Y. Αρχικά διακρίνει τα σημεία με "1" για τις α΄ στήλες, "2" για τις β΄ κλπ.
pairs	X	Σχηματίζει όλα τα ανά ζεύγη στηλών του πίνακα X διαγράμματα διασποράς , για συνολική εποπτεία
abline	a, b reg	Προσθέτει στο προηγούμενο μία γραμμή με εξίσωση y=a+bx της οποίας δίνονται τα a και b. Προσθέτει στο διάγραμμα διασποράς τη γραμμή παλινδρόμησης της y στην x, όπου `reg <- lsfit(x,y)`
box	n	Περικλείει το προηγούμενο γράφημα σε ορθογώνιο με γραμμές τόσο έντονες όσο μεγαλύτερο το n
boxplot stem	x	Σχηματίζεται το θηκόγραμμα του x Σχηματίζεται το φυλλογράφημα του x
pie	x	Σχηματίζει κυκλικό διάγραμμα (πίττα) για το x
qqplot qqnorm	x, y x	Σχηματίζεται κανονικό πιθανοτικό διάγραμμα, δηλαδή διάγραμμα σε normal probability paper.
usa()		Σχηματίζει το χάρτη των ηνωμένων πολιτειών

Δίνουμε παρακάτω ένα παράδειγμα χρήσης αυτών των εντολών στα δεδομένα cats, που περιέχουν ανατομικά δεδομένα (Βάρος καρδιάς Hwt και βάρος σώματος Bwt από 47 θηλικιές και 97 αρσενικιές γάτες) και είναι ενσωματωμένα στη βιβλιοθήκη MASS. Οι εντολές με μια σύντομη επεξήγηση είναι:

```
> library(MASS)  # άνοιγμα βιβλιοθήκης MASS
> attach(cats)   # ενεργοποίηση αρχείου δεδομένων cats (help(cats))
> catsF <- lm(Hwt~Bwt,data=cats, subset=Sex=="F")
                 # γραμμική παλινδρόμηση για τις θηλυκιές γάτες
> catsM <- update(catsF, subset = Sex == "M")
                 # γραμμική παλινδρόμηση για τις αρσενικιές γάτες
> plot(Bwt, Hwt, xlab="Body Weight (kg)",
+ ylab="Heart Weight (gm)",type="n") # άδειο γράφημα με άξονες
> text(Bwt, Hwt, c("+","-")[Sex])
          # πρόσθεση σημείων με "+" οι θηλυκιές γάτες, με "-" οι αρσενικιές
> legend(2.0,20,c("Females"," ","Males"),
+       pch="+ -",bty="n")    # προσθήκη κατάλληλης λεζάντας
> lines(Bwt[Sex == "F"], fitted(catsF), lty=1,col=8,lwd=2)
          # προσθήκη ευθείας παλινδρόμησης για τις θηλυκιές γάτες
> lines(Bwt[Sex == "M"], fitted(catsM), lty=4,col=9,lwd=2)
          # προσθήκη ευθείας παλινδρόμησης για τις αρσενικιές γάτες
```

Η κατασκευή των περισσοτέρων κλασικών γραφημάτων γίνεται με τη συνάρτηση **plot()**, ενώ ο καθορισμός πολλών από τις παραμέτρους που καθορίζουν τους άξονες, τα μεγέθη των γραμματοσειρών που θα χρησιμοποιηθούν, τον τρόπο και τη θέση που θα τοποθετηθούν οι επικεφαλίδες και οι λεζάντες γίνεται με τη συνάρτηση **par()**. Έτσι είναι χρήσιμο να μελετήσει κανείς τη σελίδα βοήθειας για τη συνάρτηση αυτή. Τις πιο συνηθισμένες και απαραίτητες από τις συναρτήσεις και παραμέτρους γραφημάτων, τις έχουμε ήδη χρησιμοποιήσει στα γραφήματα που είδαμε μέχρι τώρα. Μια συνοπτική περιγραφή των παραμέτρων που εξηγούνται στη βοήθεια της **par()**, είναι οι:

- **type=**"p" (points) τυπώνονται σύμβολα στα σημεία, "l" (ελ) (lines) συνδέονται τα σημεία με ευθύγραμμα τμήματα, "b" (both) σημεία και γραμμές όχι ενωμένα, "o" (overstruck) σημεία και γραμμές ενωμένα, "h" (high-density) κατακόρυφες γραμμές μέχρι τον οριζόντιο άξονα, "s" (stairstep) σκαλάκια και "n" (none) δεν τυπώνεται τίποτε. Η έλλειψη της type ισοδυναμεί με type="p".
- **lty=**1 έως 8 για τον καθορισμό του είδους της γραμμής, όπως φαίνεται στον πίνακα Β.14. Η έλλειψη της lty ισοδυναμεί με lty=1.
- **pch=**1 έως 18 για τον καθορισμό του συμβόλου που θα τυπωθεί στο σημείο, όπως φαίνεται στον πίνακα Β.14. Η έλλειψη της pch ισοδυναμεί με pch=1. Μπορούμε επίσης να θέσουμε ως χαρακτήρα οποιοδήποτε γράμμα ή άλλο σύμβολο. Π.χ. μπορούμε να θέσουμε pch="A" (για να τυπωθούν A), pch=="+" (για να τυπωθούν +) ή pch="\"" (για να τυπωθούν ").
- **col=**συνήθως 1 έως 15 για τον καθορισμό των χρωμάτων. Για να δούμε ποια χρώματα αντιστοιχίζει το πρόγραμμα σ' αυτούς τους αριθμούς, εκτελούμε την:
  ```
  > barplot(rep(1,15), col = 1:15)
  ```
- **log=**"x" οπότε ο άξονας x αριθμείται λογαριθμικά (δηλαδή τα διαστήματα 1 έως 10, 10 έως 100, 100 έως 1000 παριστάνονται με ίσο μήκος) (ημι-λογαριθμικό χαρτί), "y" οπότε ο άξονας y αριθμείται λογαριθμικά ή "xy" οπότε και οι δύο άξονες αριθμούνται λογαριθμικά (λογο-λογαριθμικό χαρτί)

Για παράδειγμα πήραμε 1000 τυχαίους κανονικούς αριθμούς και κάναμε τη γραφική τους παράσταση με τρεις διαφορετικούς τρόπους. Στο πρώτο γράφημα οι αριθμοί τυπώνονται σε σχέση με τον αύξονα αριθμό τους. Στο δεύτερο ο οριζόντιος άξονας υποδιαιρέθηκε λογαριθμικά. Αυτό είχε ως συνέπεια να συσσωρευτούν τα σημεία προς τα δεξιά. Στο τρίτο γράφημα χρησιμοποιήσαμε το λογάριθμο του αύξοντα αριθμού. Έτσι προέκυψε το ίδιο γράφημα, με μόνη διαφορά την αρίθμηση. Στο τελευταίο γράφημα προσθέσαμε και διαγραμμίσεις με τη βοήθεια της παραμέτρου **tck=.** Εκτελέστε τις εντολές:

```
> set.seed(1000); ynorm <- rnorm(1000) # 1000 κανονικοί τυχαίοι αριθμοί
> par(mfrow=c(3,1))    # υποδιαίρεση της οθόνης (3 γραμμές, 1 στήλη)
> plot(1:1000, ynorm)                    # το γράφημα
> plot(1:1000, ynorm, log="x")       # το γράφημα με λογαριθμημένο άξονα x
> plot(log10(1:1000), ynorm,tck=1) # το γράφημα με λογαριθμημένες τιμές x
> par(mfrow=c(1,1))    # επαναφορά της οθόνης στην κανονική της μορφή
```

- **xlab=**"ονομασία του άξονα των x"
- **ylab=**"ονομασία του άξονα των y"
- **xlim=**c(x1, x2), όπου x1, x2 τα ακραία σημεία του άξονα των x.
- **ylim=**c(y1, y2), όπου y1, y2 τα ακραία σημεία του άξονα των y.
- **cex=**ο μέγεθος των γραμμάτων και συμβόλων. Αρχική τιμή είναι το 1.
- **crt=**η γωνία τοποθέτησης επικεφαλίδων. Αρχική τιμή είναι το 0.
- **tck=**το μέγεθος των σημαδιών στους άξονες ως ποσοστό του πλάτους του γραφήματος. Αρχική τιμή είναι -0.02, όπου το – σημαίνει προς τα έξω.
- **axes=**T (True) η αρχική τιμή, F (False) ακυρώνει την εκτύπωση των αξόνων.

Ένα ακόμη παράδειγμα με το αρχείο salary.df που ορίστηκε στη σελ. 25. Με τις εντολές

```
> attach(salary.df)
> plot(exper, salary)
```

παίρνουμε ένα διάγραμμα διασποράς με τη μεταβλητή exper στον οριζόντιο άξονα και την salary στον κατακόρυφο άξονα. Τα σύμβολα που χρησιμοποιούνται για τα σημεία, το εάν θα είναι συνδεδεμένα μεταξύ τους με ευθύγραμμα τμήματα, το χρώμα τους, το μέγεθός τους κλπ ακολουθούν κάποιες αρχικές τιμές που εφαρμόζει το πρόγραμμα.

Τροποποιήσαμε το γράφημα με τις επόμενες εντολές.

```
> plot(exper,salary,type="n", xlab="εμπειρία (exper)", ylab=
+  "μισθός (salary)")
> points(exper[educ==1],salary[educ==1],pch=16,col=3)
> points(exper[educ==2],salary[educ==2],pch=17,cex=1.3,col=4)
> points(exper[educ==3],salary[educ==3],pch=18,cex=1.6,col=5)
> title(main="Μισθός ανά εμπειρία (και εκπαίδευση)")
> legend(locator(1), legend=c("educ=1", "educ=2", "educ=3"),
+     pch=16:18, col=c(3,4,5))
```

Η πρώτη από αυτές δημιουργεί το πλαίσιο του γραφήματος μαζί με τις επικεφαλίδες, χωρίς τα σημεία στο εσωτερικό. Οι τρεις επόμενες επιλέγουν εκείνα τα σημεία που αντιστοιχούν ανάλογα σε τιμές της μεταβλητής educ 1, 2 και 3 και τα σημειώνουν με διαφορετικό σύμβολο, διαφορετικό μέγεθος και διαφορετικό χρώμα. Η επόμενη προσθέτει τον κύριο τίτλο, ο οποίος όπως φαίνεται μπορεί να περιέχει και ελληνικούς χαρακτήρες. Τέλος η τελευταία εντολή προσθέτει την λεζάντα. Η πρώτη παράμετρος locator(1) βοηθά στην τοποθέτηση της λεζάντας εκεί που βολεύει. Πράγματι με την εκτέλεση της εντολής αυτής, ο κέρσορας μετασχηματίζεται σε σταυρό όταν τον φέρουμε πάνω από το γράφημα. Με κλικ στη θέση που θέλουμε να είναι η άνω δεξιά γωνία της λεζάντας, πετυχαίνουμε την τοποθέτηση. Αντί της παραμέτρου locator(1) δίνουμε τις συντεταγμένες της άνω δεξιάς γωνίας. Έτσι, στο παράδειγμά μας η εντολή

```
> legend(15,600, legend=c("educ=1", "educ=2", "educ=3"),
+  col=c(3,4,5), pch=16:18)
```

έχει ισοδύναμο αποτέλεσμα με την προηγούμενη.

Με το πρόγραμμα R μπορούμε επίσης να σχεδιάσουμε οποιαδήποτε συνάρτηση καθώς και τις συναρτήσεις πυκνότητας πιθανότητας των γνωστών κατανομών.

Έστω για παράδειγμα η συνάρτηση $f(x)=\begin{cases}\cos(x), & -\pi \le x \le 0\\ e^{-x}, & 0 \le x \le 1\\ \sqrt{x}, & 1 \le x \le 4\end{cases}$.

Η γραφική της παράσταση πετυχαίνεται με τις εντολές:

```
> x1<-seq(-pi,0,length=500)
> x2<-seq(0,1,length=500)
> x3<-seq(1,4,length=500)
> y1<-cos(x1)
> y2<-exp(-x2)
> y3<-sqrt(x3)
> x<-c(x1,x2,x3)
> y<-c(y1,y2,y3)
> plot(x,y,type="n",axes=F,xlab="",ylab="",pch=".")
> points(x1,y1,type="l",col=2,lwd=2)
> points(x2,y2,type="l",col=3,lwd=2)
> points(x3,y3,type="l",col=4,lwd=2)
> axis(1,pos=0, at=c(-pi,-pi/2,0,1:4),
+    labels=c("-pi","-pi/2","0","1","2","3","4"))
> axis(2,pos=0)
```

Οι πρώτες τρεις εντολές ορίζουν ένα πλέγμα γειτονικών ισαπεχουσών τιμών στα διαστήματα που ορίζεται με διαφορετικό τρόπο η συνάρτηση. Οι επόμενες τρεις υπολογίζουν τις αντίστοιχες τιμές της συ-

νάρτησης. Οι άλλες δύο συνενώνουν τις τιμές x και τις τιμές y, ώστε να είναι δυνατό να παρασταθούν γραφικά σε ένα γράφημα. Στην επόμενη, που κάνει το γράφημα, ζητήσαμε να μη τυπωθούν τα σημεία, αλλά ούτε και οι άξονες, για να καθορίσουμε μετά την τοποθέτησή τους. Στις επόμενες τρεις προσθέσαμε στο άδειο γράφημα τα σημεία (x,y) των τριών επιμέρους συναρτήσεων συνενώνοντάς τα με lines (**type="l"**). Στις δύο τελευταίες ζητούμε οι άξονες να περνούν από την αρχή (pos=0), ενώ για τον οριζόντιο άξονα δώσαμε με την παράμετρο "at=" τις θέσεις που θα σημειωθούν με tick στον άξονα και με την labels= τι να γραφεί στα διάφορα σημεία.

Μια εναλλακτική σχεδίαση της γραφικής παράστασης της ίδιας συνάρτησης f(x) γίνεται με τις επόμενες εντολές. Εδώ αφήνουμε το γράφημα και σχηματίζουμε δεύτερο γράφημα πάνω του με την εντολή par(new=T). Σημαντικό στον τρόπο αυτό είναι να ορίζουμε οπωσδήποτε τα xlim, ylim, ώστε να είναι 'ιδιοι οι άξονες στα τρία γραφήματα.

```
> plot(x1, y1, type = "o", axes = F, xlab = "", ylab = "",
+   pch = ".", xlim = c( - pi, 4), ylim = c(-1, 2), col = 2)
> par(new = T)
> plot(x2, y2, type = "o", axes = F, xlab = "", ylab = "",
+   pch = ".", xlim = c( - pi, 4), ylim = c(-1, 2), col = 3)
> par(new = T)
> plot(x3, y3, type = "o", axes = F, xlab = "", ylab = "",
+   pch = ".", xlim = c( - pi, 4), ylim = c(-1, 2), col = 4)
> axis(1,pos=0, at=c(-pi,-pi/2,0,1:4),
+    labels=c("-pi","-pi/2","0","1","2","3","4"))
> axis(2, pos = 0)
```

Μπορούμε επίσης εύκολα να σχηματίσουμε τη γραφική παράσταση της πυκνότητας πιθανότητας οποιασδήποτε κατανομής π.χ. της κατανομής Γάμμα με παράμετρο 1.7. Για το σχηματισμό της χρησιμοποιήσαμε τη συνάρτηση dgamma που δίνει τις πυκνότητες στα ποσοστιαία σημεία που δίνονται από τη συνάρτηση pgamma. Οι εντολές ήταν

```
> drawgamma <- function(shape){
+ x <- qgamma(seq(.001,.999,length=200),shape)
+ plot(x,dgamma(x,shape), type="l",xlab="",ylab="",axes=F)
+ axis(1,pos=0);axis(2,pos=0)
+ }                          # ορισμός σχετικής συνάρτησης
> drawgamma(1.7)             # εφαρμογή
> text(locator(1),"Γάμμα(1.7)")    # λεζάντα
```

Παρατηρούμε ότι οι πρώτες 5 εντολές ορίζουν μια συνάρτηση με όρισμα την παράμετρο shape της κατανομής Γάμμα, έτσι ώστε να μπορεί να χρησιμοποιηθεί και μελλοντικά. Η γραφική παράσταση ζητήσαμε να έχει άξονες που περνούν από την αρχή, όπως συμβαίνει στις παραστάσεις μαθηματικών συναρτήσεων. Η τελευταία εντολή χρησιμοποιεί την locator(1) για την τοποθέτηση κειμένου στο γράφημα με τη βοήθεια του ποντικιού.