Ανάλυση Δεδομένων με χρήση του Στατιστικού Πακέτου R

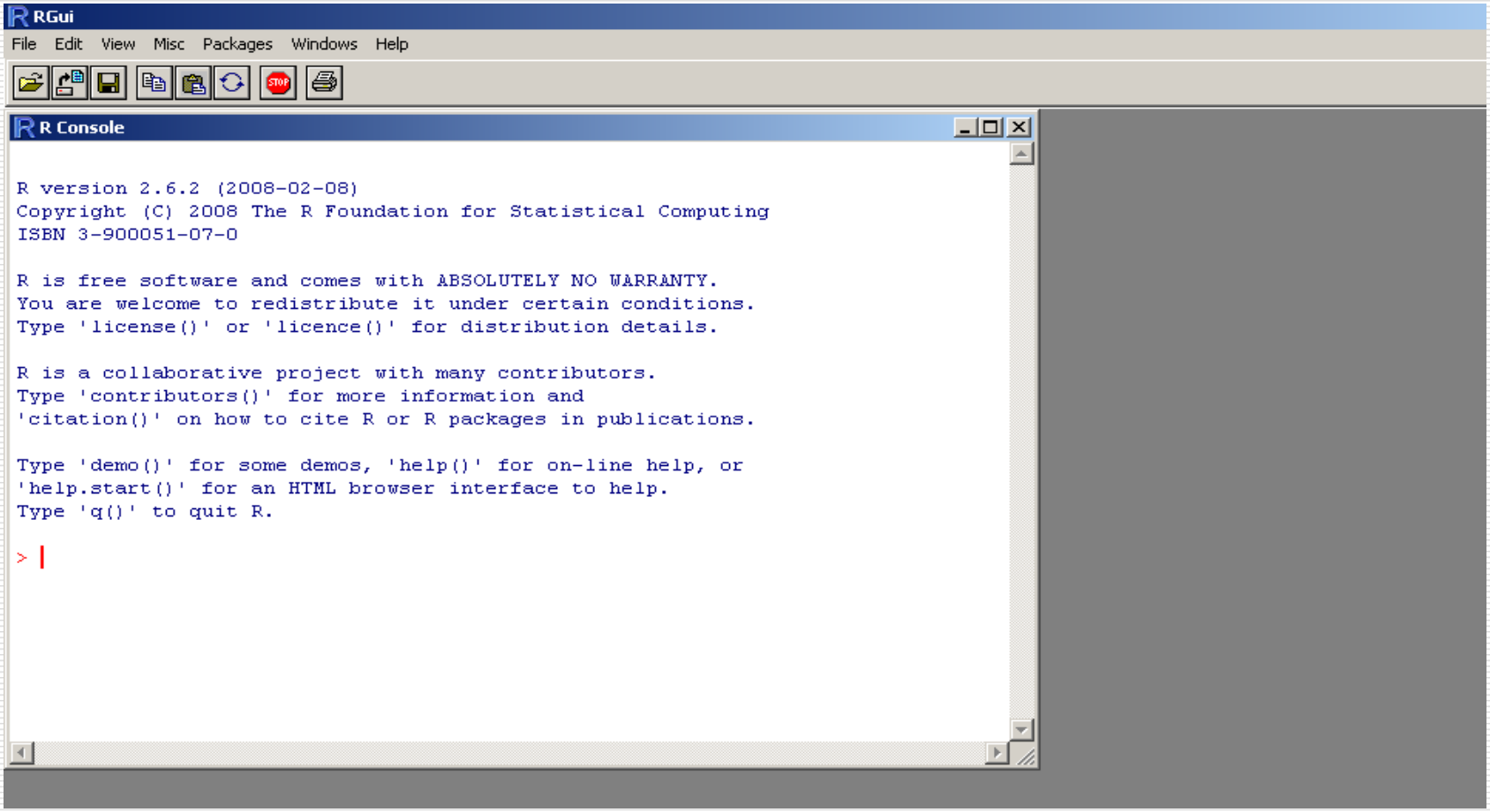
Δημήτρης Φουσκάκης,
Καθηγητής,
Τομέας Μαθηματικών,
Σχολή Εφαρμοσμένων Μαθηματικών και Φυσικών Επιστημών,
Εθνικό Μετσόβιο Πολυτεχνείο.

Περιεχόμενα

Τι είναι η γλώσσα R

- Η R είναι μια γλώσσα προγραμματισμού που χρησιμεύει κυρίως για ανάλυση δεδομένων και εφαρμογή διαφόρων "κλασικών" και "σύγχρονων" στατιστικών τεχνικών.
- Μπορεί να αποκτηθεί δωρεάν από την ιστοσελίδα http://www.r-projects.org ή από ένα από τα πολλά πρότυπα (mirrors) του CRAN (Comprehensive R Archive) http://cran.r-project.org το οποίο είναι ένα δίκτυο διανομής της R σε πολλά μέρη του κόσμου μέσω διαδικτύου. Υποστηρίζει πολλές πλατφόρμες και λειτουργικά όπως Linux, Mac OS, Windows και Playstation 3!
- Μπορεί να χρησιμοποιηθεί είτε με κατευθείαν εντολές που υπάρχουν είτε με προγράμματα που ο χρήστης μπορεί να προγραμματίσει για επίλυση πιο πολύπλοκων στατιστικών προβλημάτων. Επίσης ο χρήστης μπορεί να χρησιμοποιήσει και έτοιμα προγράμματα τα οποία είναι ενσωματωμένα μέσα σε πακέτα τα οποία διατίθενται πάλι ελεύθερα. Οι ποικιλία τέτοιων προγραμμάτων είναι τεράστια.
- Στις συγκεκριμένες σημειώσεις χρησιμοποιούμε την έκδοση 2.6.2.

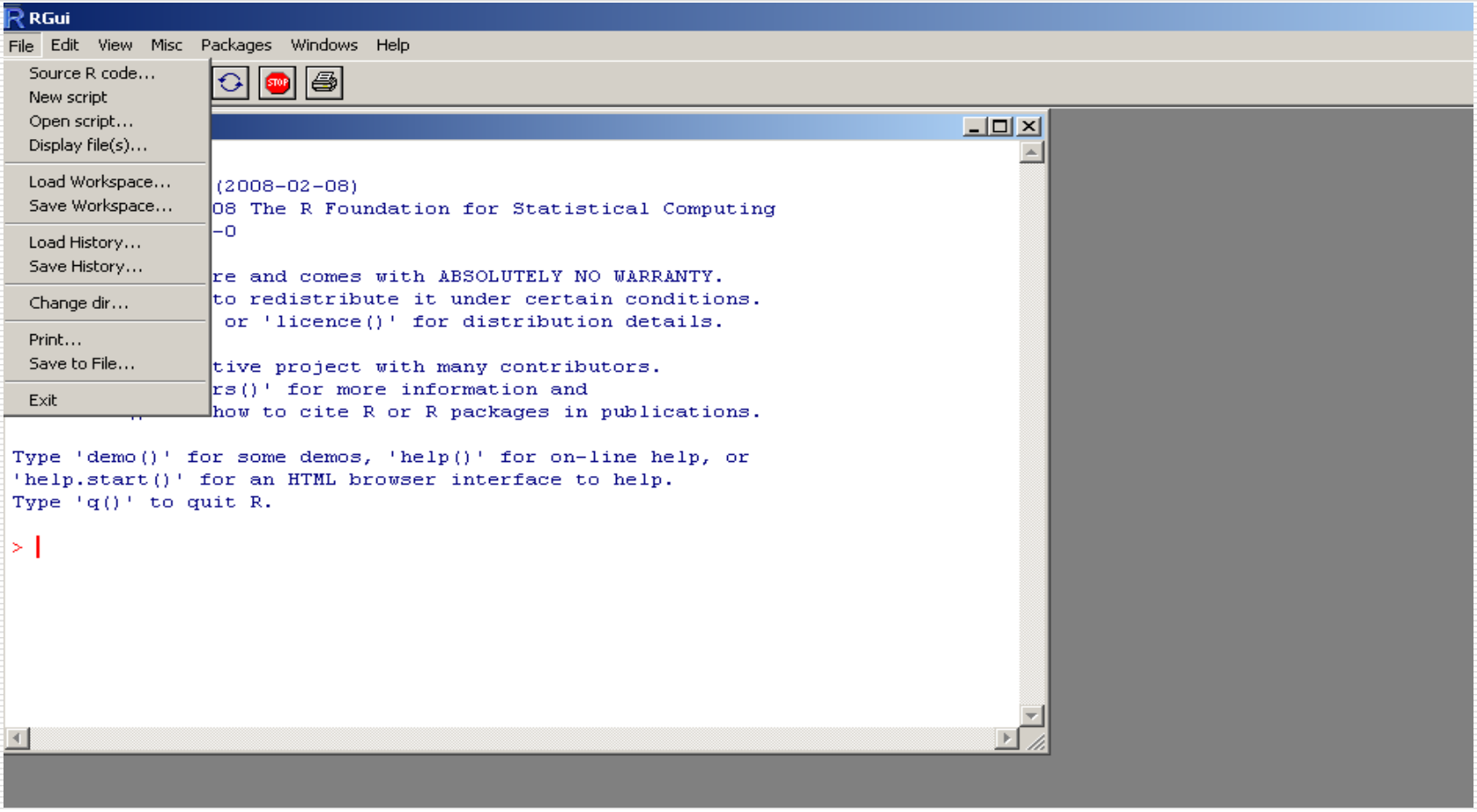
Το περιβάλλον της R



Το περιβάλλον της R

- Η εκκίνηση του προγράμματος γίνεται με διπλό κλικ στο εικονίδιο της R.
- Τότε εμφανίζεται η βασική οθόνη του προγράμματος (σαν αυτή της προηγουμένης διαφάνειας) στην οποία υπάρχει το παράθυρο εντολών (**R-console**). Ο κέρσορας βρίσκεται μετά το σύμβολο “>” και το πρόγραμμα περιμένει τις εντολές σας.
- Για να τερματίσετε το πρόγραμμα είτε πληκτρολογήστε q(), είτε κλείστε την οθόνη του προγράμματος (όχι του παραθύρου εντολών) πάνω δεξιά, είτε από το μενού **file** επιλέξτε το **exit**. Σε κάθε περίπτωση θα ερωτηθείτε αν θέλετε να αποθηκεύσετε ότι έχετε μέχρι τώρα δημιουργήσει (αντικείμενα, συναρτήσεις, κλπ).

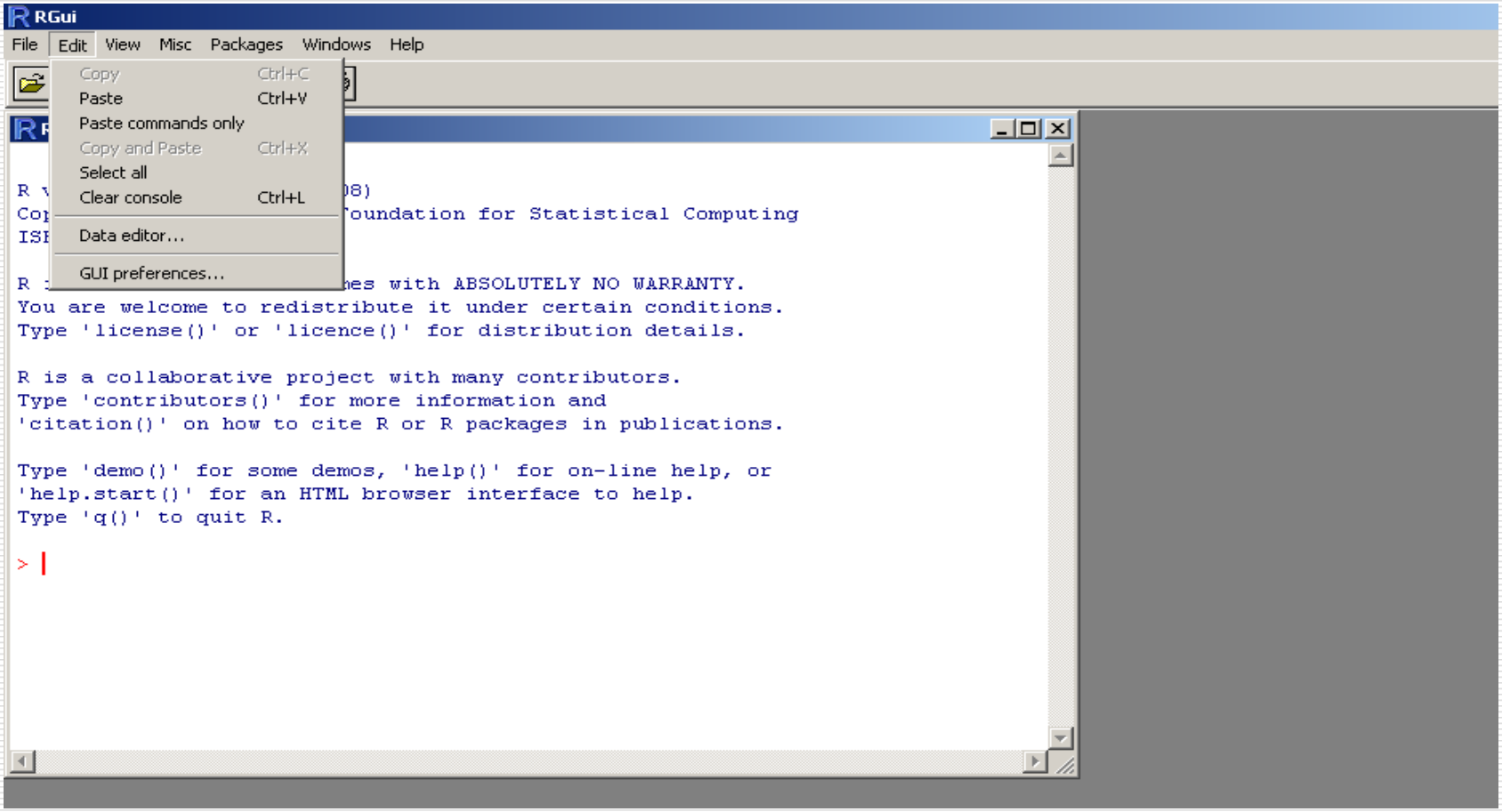
Το μενού File



Το μενού File

- Με την επιλογή αυτή μπορούμε:
  - Να εισάγουμε κώδικα και εντολές από προηγούμενες εφαρμογές μας με το **source R code**.
  - Να ανοίξουμε έναν νέο συντάκτη (**new script**) στον οποίο να τυπώσουμε τις εντολές που θέλουμε να εκτελέσουμε. Μαυρίζουμε με το ποντίκι τις εντολές που θέλουμε να εκτελέσουμε και με δεξί κλικ πάνω στον συντάκτη διαλέγουμε την επιλογή **run line or selection**.
  - Να ανοίξουμε έναν παλιό συντάκτη (**open script**) από τον οποίο θέλουμε να εκτελέσουμε κάποιες εντολές. Οι εκτέλεση γίνεται όπως και πριν.
  - Να δούμε τα διαθέσιμα R αρχεία του φακέλου που είμαστε (**display files**).
  - Να εισάγουμε ή να αποθηκεύσουμε επιφάνειες εργασίας (**workspace**) με αντικείμενα και συναρτήσεις που έχουν δημιουργηθεί (**load/save workspace**).
  - Να εισάγουμε ή να αποθηκεύσουμε εντολές που ήδη έχουμε χρησιμοποιήσει (**load/save history**).
  - Να αλλάξουμε τον φάκελο εργασίας μας (**change dir**).
  - Να εκτυπώσουμε (**print**), να αποθηκεύσουμε την δουλειά μας σε ένα αρχείο κειμένου (**save to file**) και να τερματίσουμε το πρόγραμμα (**exit**).

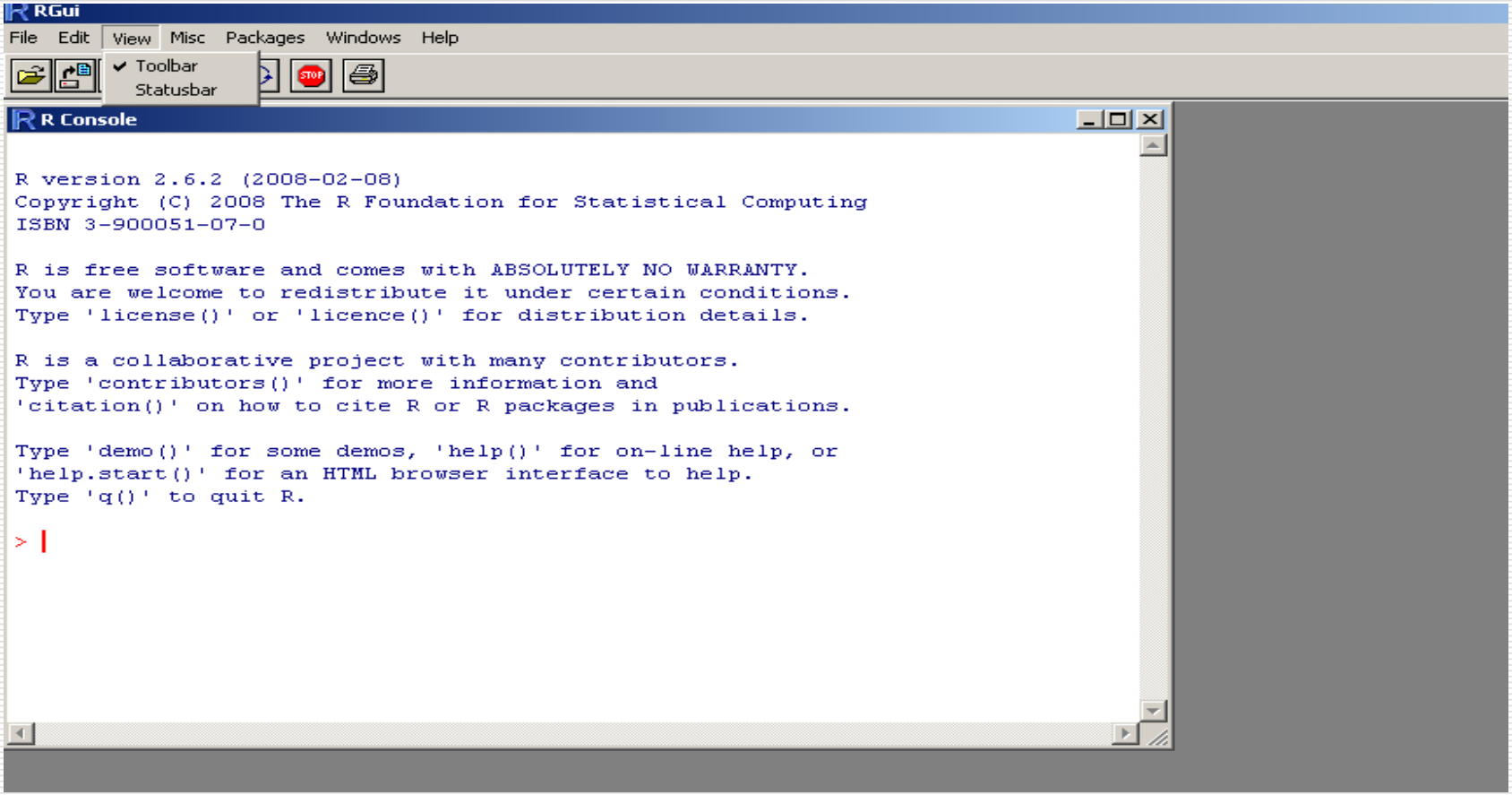
Το μενού Edit



Το μενού Edit

- Εδώ έχουμε τις γνωστές δυνατότητες αντιγραφής (**copy**) και επικόλλησης (**paste**), επιλογής όλων όσων έχουμε πληκτρολογήσει (**select all**), καθαρισμού του παραθύρου εντολών (**clear console**). Επίσης μπορούμε να ανοίξουμε τον συντάκτη δεδομένων (**data editor**) για κάποιο σετ δεδομένων που είναι υπό μορφή πλαισίου δεδομένων – data frame (περισσότερα για τα πλαίσια δεδομένων αργότερα) – και να επεξεργαστούμε αυτά τα δεδομένα. Τέλος μπορούμε να αλλάξουμε το τρόπο εμφάνισης του περιβάλλοντος εργασίας μας (**GUI preferences**).

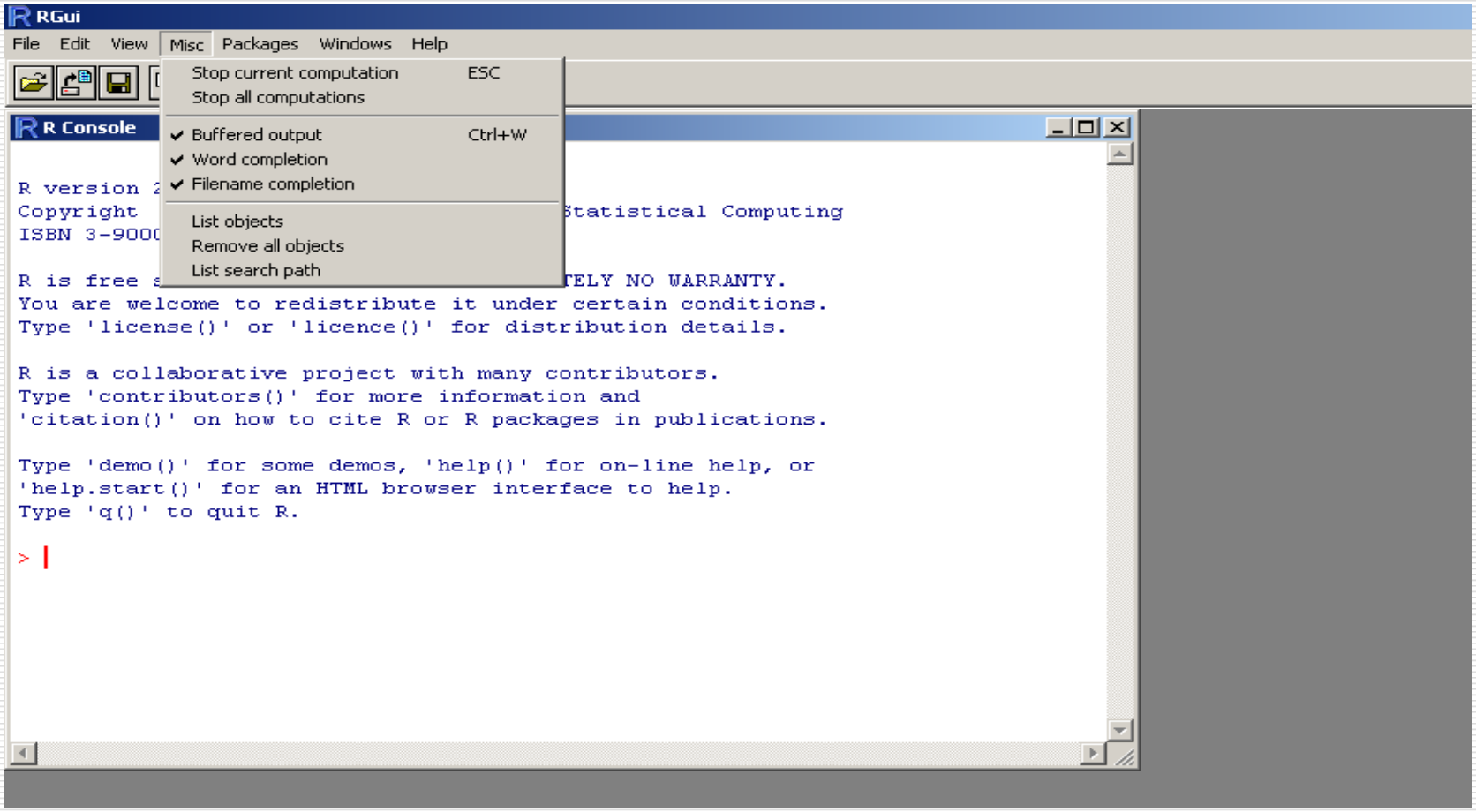
Το μενού View



Το μενού View

- Μπορείτε αν θέλετε να διαγράψετε το toolbar (με όλα τα μενού) από το περιβάλλον εργασίας, όπως επίσης και το statusbar (η μπάρα στο κάτω μέρος με πληροφορίες για την έκδοση του προγράμματος που τρέχετε).

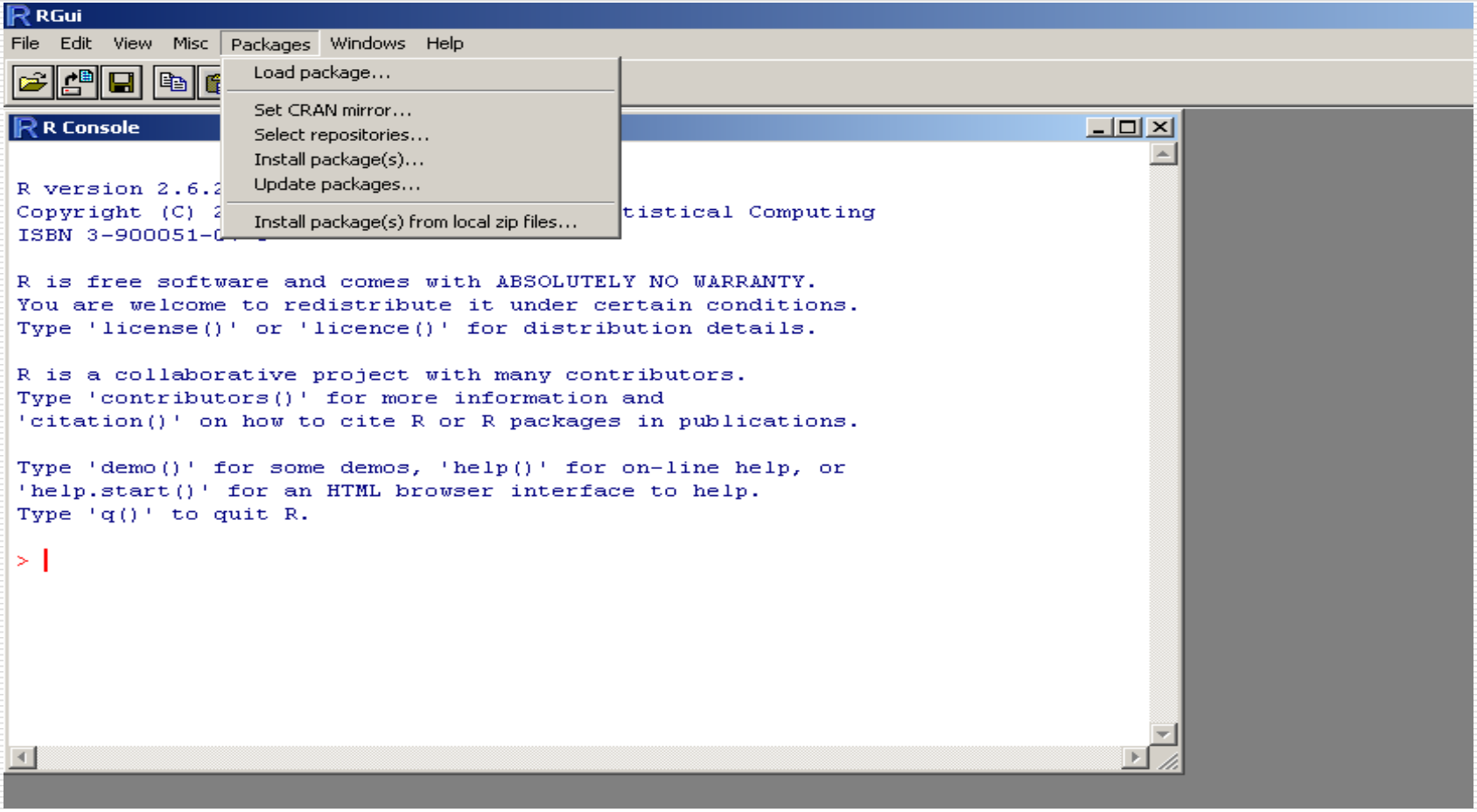
Το μενού Misc



Το μενού Misc

- Από το μενού **Misc** ο χρήστης μπορεί να σταματήσει το τρέχον ή όλα τα προγράμματα που εκτελούνται (**stop current/all computations**), να σταματήσει την εκτύπωση αποτελεσμάτων στην οθόνη (**buffered output**), να δει όλα τα αντικείμενα που έχει δημιουργήσει έως τώρα (**list objects**) - ισοδύναμα μπορεί να χρησιμοποιήσει την εντολή **ls()** ή **objects()**, να διαγράψει όσα αντικείμενα έχει δημιουργήσει έως τώρα (**remove all objects**) - ισοδύναμα μπορεί να χρησιμοποιήσει την εντολή **rm(list=ls(all=TRUE))** και τέλος να δει ποιες βιβλιοθήκες (**libraries**) και πλαίσια δεδομένων (**data frames**) επισυνάπτονται στο τρέχον περιβάλλον εργασίας του.

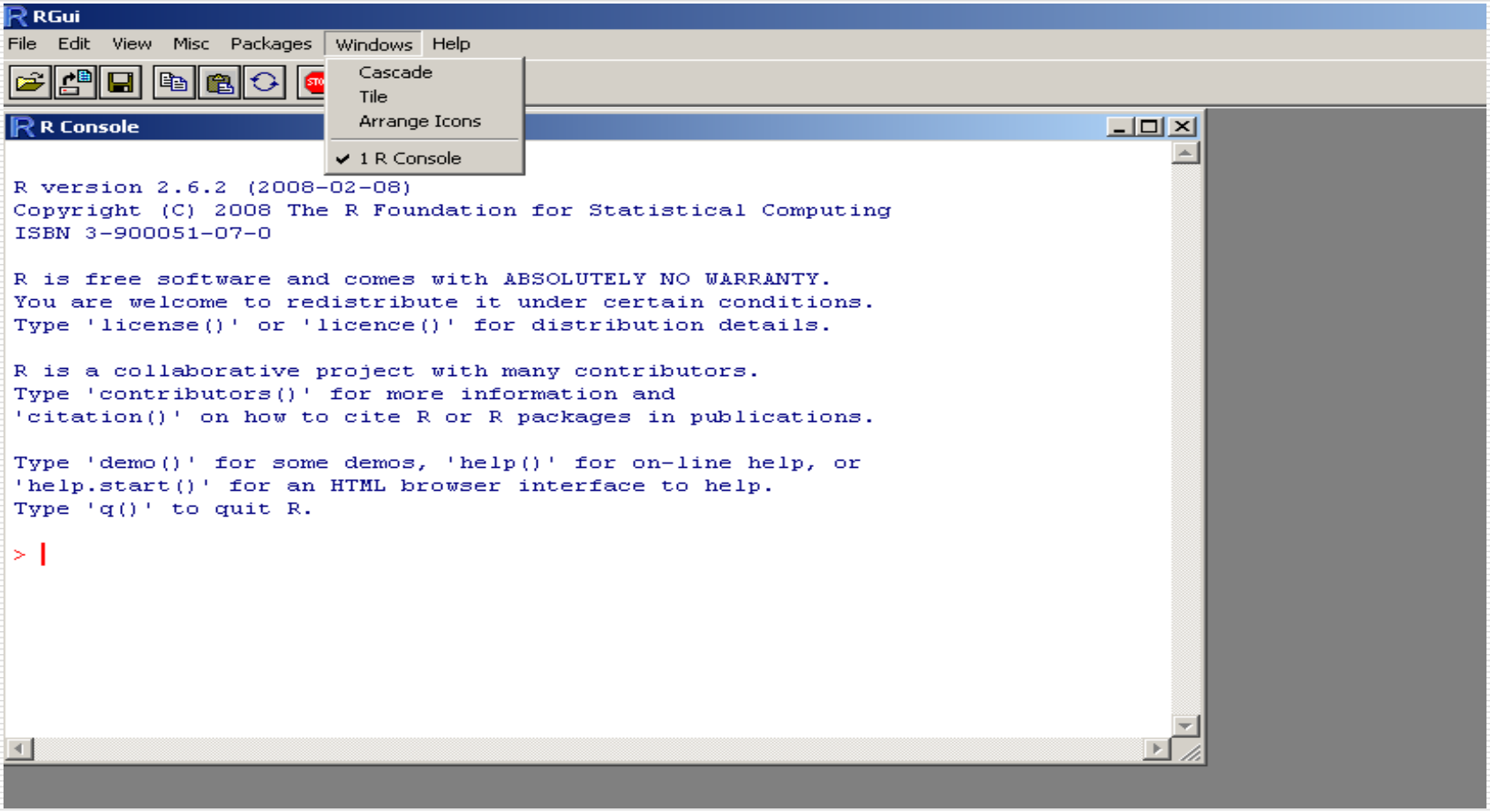
Το μενού Packages



Το μενού Packages

- Από το μενού **Packages** ο χρήστης μπορεί να φορτώσει βιβλιοθήκες που ήδη έχει κατεβάσει (**load package**), να κατεβάσει βιβλιοθήκες από διάφορα πρότυπα του CRAN (**install package(s))** ή από συμπιεσμένα αρχεία του σκληρού του δίσκου (**install package(s) from local zip files**), να ενημερώσει τις βιβλιοθήκες προσθέτοντας νέες (**update packages**), να διαλέξει από ποιο μέρος του κόσμου θα κατεβάσει μέσω του **CRAN** τις βιβλιοθήκες (**set CRAN mirror**) ή να διαλέξει από ποιον διανομέα (πέραν του **CRAN**) θέλει να κατεβάσει τις βιβλιοθήκες (**set repositories**).

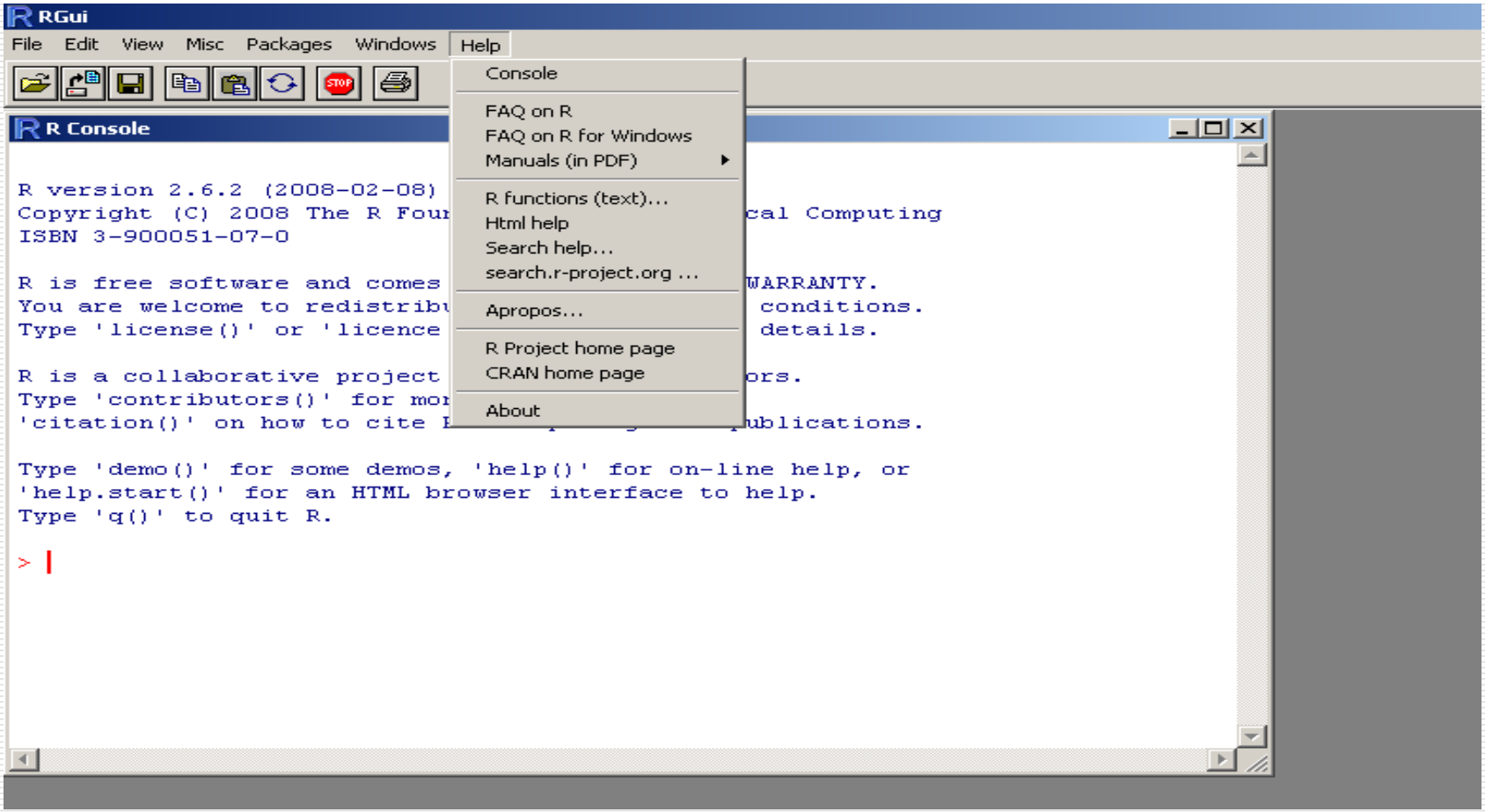
Το μενού Windows



Το μενού Windows

- Με το μενού **Windows** ο χρήστης μπορεί να μετακινηθεί μεταξύ των ανοιχτών παραθύρων και να τα διατάξει με τον τρόπο που επιθυμεί.

Το μενού Help



Το μενού Help

- Με το μενού **Help** δίνεται στον χρήστη ένα εγχειρίδιο για όλες τις εντολές και ιδιότητες του πακέτου. Πιο συγκεκριμένα:
  - **Console**: Πληροφορίες για το πως ο χρήστης μπορεί να χειριστεί την βασική οθόνη του προγράμματος.
  - **FAQ on R** και **FAQ on R for Windows**: Απαντήσεις σε συνήθεις ερωτήσεις για την R και για την R για *windows*.
  - **Manuals (in pdf)**: Βασικό εγχειρίδιο της R σε μορφή pdf.
  - **R functions (text)**: Πληροφορίες για τις εντολές της R που είναι ήδη φορτωμένες (από το βασικό πακέτο ή από τις ήδη φορτωμένες βιβλιοθήκες).

Το μενού Help

- **Html help**: Διαδικτυακός χώρος με πληροφορίες για την R.
- **Search help**: Αρχεία που σχετίζονται άμεσα ή έμμεσα με την λέξη που αναζητείται από όλες τις διαθέσιμες βιβλιοθήκες.
- **Search.r-project.org**: Σύνδεσμοι στο διαδίκτυο που σχετίζονται άμεσα ή έμμεσα με την λέξη που αναζητείται.
- **Apropos**: Εντολές που είναι ήδη φορτωμένες και σχετίζονται άμεσα ή έμμεσα με την λέξη που αναζητείται.
- **R project home page**: Μεταφορά στην ιστοσελίδα της R.
- **CRAN home page**: Μεταφορά στην ιστοσελίδα της CRAN.
- **About**: Πληροφορία για την έκδοση και τα δικαιώματα του πακέτου.

Αριθμητικοί Τελεστές της R

Σύμβολο	**Πράξη**
+	Πρόσθεση
-	Αφαίρεση
*	Πολλαπλασιασμός
/	Διαίρεση
^	Ύψωση σε δύναμη
% / %	Πηλίκο Ακέραιας Διαίρεσης
% %	Υπόλοιπο Διαίρεσης

Αριθμητικοί Τελεστές της R

```
> 3+3  # this is my first command
[1] 6
> 9-2
[1] 7
> 17/3
[1] 5.666667
> 4*2
[1] 8
> 2^3
[1] 8
> 17%/%3
[1] 5
> 17%%3
[1] 2
> 7/0
[1] Inf
```

- Οτιδήποτε ακολουθεί μετά τον χαρακτήρα # αγνοείται.
- Το Inf δηλώνει άπειρο και μπορεί να χρησιμοποιηθεί, π.χ.

```
> exp(-Inf)
[1] 0
```

$\lim_{x \to \infty} e^{-x}$

- Το NaN δηλώνει ότι η πράξη δεν μπορεί να γίνει, π.χ.

```
> log(-2)
[1] NaN
 Warning message:
 In log(-2) : NaNs  produced
```

Τελεστές Εκχώρησης και Σύγκρισης της R

Τελεστής	Ερμηνεία
<-	Εκχωρεί το αποτέλεσμα στο αριστερό μέλος της σχέσης
->	Εκχωρεί το αποτέλεσμα στο δεξί μέλος της σχέσης
>	Μεγαλύτερο από
<	Μικρότερο από
>=	Μεγαλύτερο ή ίσο από
<=	Μικρότερο ή ίσο από
==	Ίσο με
!=	Όχι ίσο με

Τελεστές Εκχώρησης και Σύγκρισης της R

```
> x<-56
> x
[1] 56

> 5*2->y
> y
[1] 10

> x>y
[1] TRUE

> y<4
[1] FALSE
```

```
> x>=56
[1] TRUE

> y<=9
[1] FALSE

> x==56
[1] TRUE

> y!=1
[1] TRUE
```

Περισσότερα για τους Τελεστές Εκχώρησης

- Η R είναι ευαίσθητη στα κεφαλαία γράμματα (case sensitive), δηλαδή το x και το X είναι διαφορετικά αντικείμενα.
- Στις τελευταίες εκδόσεις της R οι τελεστές "=" και "<-" είναι ισοδύναμοι.
- Στην εντολή εκχώρησης το αποτέλεσμα δεν εμφανίζεται στην οθόνη αλλά καταχωρείται στο αντικείμενο. Για να δούμε την τιμή του αντικειμένου πληκτρολογούμε το όνομά του. Για να δούμε όλα τα αντικείμενα που βρίσκονται στον χώρο εργασίας της R πληκτρολογούμε ls().
- Τα αντικείμενα μπορεί να είναι επίσης διανύσματα, πίνακες, πλαίσια δεδομένων ή λίστες (περισσότερα για τις δομές δεδομένων της R αργότερα).
- Ένα αντικείμενο μπορεί να εμφανιστεί και στα δύο μέρη ενός τελεστή εκχώρησης, αρκεί πριν να το έχουμε ορίσει. Π.χ.

```
> x<-5
> x<-x+3
> x
[1] 8
```

Περισσότερα για τους Τελεστές Σύγκρισης

- Σε περίπτωση που το όνομα του αντικειμένου στο οποίο καταχωρούμε μια τιμή υπάρχει, η τιμή του αντικειμένου θα αντικατασταθεί με την καινούργια.
- Οι τελεστές σύγκρισης ελέγχουν αν ισχύει μια σχέση (π.χ. >) και επιστρέφουν την τιμή TRUE αν ισχύει, ή FALSE αν δεν ισχύει. Αν θέλουμε να συνδυάσουμε 2 ή περισσότερες συγκρίσεις χρησιμοποιούμε το σύμβολο & (και) για να ισχύουν όλες ή το σύμβολο | (ή) για να ισχύει τουλάχιστον μία. Π.χ.

```
> (5>3) & (8>10)
[1] FALSE
> (5>3) | (8>10)
[1] TRUE
```

- Οι τελεστές σύγκρισης χρησιμοποιούνται επίσης όπως θα δούμε και στις βασικές δομημένες εντολές (*βρόχοι*), π.χ if, for, κλπ.
- Ο τελεστής ! δηλώνει το αντίθετο της έκφρασης που ακολουθεί. Π.χ.

```
> !(5>3)
[1] FALSE
```

Βασικές Αριθμητικές Συναρτήσεις της R

Συνάρτηση	Πράξη	Συνάρτηση	Πράξη
sqrt()	τετραγωνική ρίζα	asin()	τόξο ημιτόνου
abs()	απόλυτη τιμή	atan()	τόξο εφαπτομένης
log()	φυσικός λογάριθμος	gamma()	συνάρτηση Γάμμα
log2()	λογάριθμος με βάση 2	lgamma()	φυσικός λογάριθμος της συνάρτησης Γάμμα
log10()	λογάριθμος με βάση 10	beta()	συνάρτηση Βήτα
exp()	εκθετική συνάρτηση	floor()	προηγούμενος ακέραιος
cos()	συνημίτονο	ceiling()	επόμενος ακέραιος
sin()	ημίτονο	factorial()	παραγοντικό
tan()	εφαπτομένη	choose()	συνδυασμοί
acos()	τόξο συνημιτόνου	lchoose()	φυσικός λογάριθμος συνδυασμών

Βασικές Αριθμητικές Συναρτήσεις της R

```
> sqrt(16)
[1] 4
> abs(-2)
[1] 2
> log(10)
[1] 2.302585
> log2(10)
[1] 3.321928
> log10(10)
[1] 1
> exp(3)
[1] 20.08554
> cos(pi)
[1] -1
> sin(2*pi)
[1] -2.449213e-16
> tan(pi/2)
[1] 1.633178e+16
```

```
> sin(pi/2)
[1] 1
> tan(0)
[1] 0
> acos(0.2)
[1] 1.369438
> atan(2)
[1] 1.107149
> asin(0)
[1] 0
> gamma(2)
[1] 1
> beta(2,3)
[1] 0.08333333
```

```
> lgamma(4)
[1] 1.791759
> floor(4.9)
[1] 4
> ceiling(4.1)
[1] 5
> factorial(5)
[1] 120
> choose(5,2)
[1] 10
> lchoose(5,2)
[1] 2.302585
```

Βασικές Αριθμητικές Συναρτήσεις της R

- Το pi στην R δηλώνει τον αριθμό π=3.14…
- Στην R ο αριθμός $2.688117e+43$ είναι ο 2.688117×10^{43} .
- Η συνάρτηση log(x, base=b) υπολογίζει τον λογάριθμο με βάση τον αριθμό b.
- Η συνάρτηση round() παίρνει ως ορίσματα τον αριθμό που θέλουμε να στρογγυλοποιήσουμε και το πλήθος των δεκαδικών ψηφίων, που θέλουμε να εμφανίσουμε (digits).

```
> round(3.14,1)
[1] 3.1
> round(3.16,1)
[1] 3.2
```

Βασικές Αριθμητικές Συναρτήσεις της R

- Η συνάρτηση choose() (ή ισοδύναμα lchoose() για τον νεπέριο λογάριθμο) δέχεται ως ορίσματα έναν πραγματικό αριθμό α και έναν φυσικό αριθμό k και επιστρέφει τον **γενικευμένο διωνυμικό συντελεστή**

$$\binom{a}{k} = \frac{a(a-1)\cdots(a-k+1)}{k(k-1)\cdots 1}$$

- Στην περίπτωση όπου α και k είναι φυσικοί αριθμοί με k μικρότερο ή ίσο του α, η παραπάνω συνάρτηση μας επιστρέφει τον συνηθισμένο **διωνυμικό συντελεστή**.

$$\binom{a}{k} = \frac{a!}{k!(a-k)!}$$

Βασικές Αριθμητικές Συναρτήσεις της R

- Το Inf (άπειρο) μπορεί να χρησιμοποιηθεί στις αριθμητικές συναρτήσεις της R όπως είπαμε και πριν

```
> exp(-Inf)
[1] 0
```

Επίσης μπορεί να δηλώσει το αποτέλεσμα μίας πράξης

```
> 9/0
[1] Inf
```

- Το NaN (*Not A Number*) λαμβάνεται σε περιπτώσεις απροσδιοριστίας.

```
> log(-2)
[1] NaN
Warning message:
In log(-2) : NaNs produced
```

Βασικές Αριθμητικές Συναρτήσεις της R

- Στην R υπάρχουν και τα σύμβολα NA (Not Available) και NULL, που χρησιμοποιούνται για να αναπαραστήσουν ελλιπείς τιμές ή απροσδιόριστες τιμές. Το σύμβολο NULL αναπαριστά το κενό αντικείμενο και συνήθως επιστρέφεται από συναρτήσεις των οποίων οι τιμές δεν υπάρχουν.

```
> x<-1
> names(x)
NULL
```

- Με την εντολή is.null ελέγχουμε αν πράγματι ένα αντικείμενο είναι κενό

```
> x<-1
> names(x)
NULL
> x<-1
> is.null(x)
[1] FALSE
> is.null(names(x))
[1] TRUE
```

Γενικές Συναρτήσεις στην R

- **builtins()**. Η συνάρτηση επιστρέφει μία λίστα με όλες τις ενσωματωμένες συναρτήσεις (built-in functions) της R.
- **cat(x) ή print(x)**. Η συνάρτηση εκτυπώνει στην οθόνη την τιμή του ορίσματος x.
- **ls()**. Η συνάρτηση επιστρέφει μία λίστα με τα αντικείμενα της τρέχουσας επιφάνειας εργασίας.
- **rm(x)**. Η συνάρτηση διαγράφει το αντικείμενο x από την τρέχουσα επιφάνεια εργασίας. Με την εντολή **rm(list=ls())** ή **rm(list=ls(all=TRUE))** μπορείτε να διαγράψετε όλα τα αντικείμενα που έχετε δημιουργήσει.

Γενικές Συναρτήσεις στην R

- **date() ή Sys.time().** Η συνάρτηση επιστρέφει την τρέχουσα ημέρα και ώρα του συστήματος.
- **Sys.Date().** Η συνάρτηση επιστρέφει την τρέχουσα ημέρα του συστήματος.
- **system.time(A).** Η συνάρτηση επιστρέφει χρόνους εκτέλεσης των εντολών Α.
- **getwd().** Η συνάρτηση επιστρέφει τον φάκελο εργασίας.
- **setwd().** Η συνάρτηση αλλάζει τον φάκελο εργασίας (το όρισμά της δηλώνει την επιθυμητή διαδρομή).
- **list.files().** Η συνάρτηση επιστρέφει μία λίστα με όλα τα αρχεία του φακέλου εργασίας.

Βασικοί Τύποι Αντικειμένων

- Κάθε αντικείμενο στην R μπορεί να είναι:
  - Πραγματικός Αριθμός

    ```
    > x<- 3
    ```
  - Μιγαδικός Αριθμός

    ```
    > x<-complex(real=4, imaginary=3)
    > x
    [1] 4 +3i
    ```
  - Δεδομένο Λογικής

    ```
    > x <-3
    > y <- x > 4
    > y
    [1] FALSE
    ```
  - Δεδομένα Χαρακτήρων

    ```
    > x <- 'DIMITRIS'
    > x
    [1] "DIMITRIS"
    ```

Με την εντολή **class()** μπορούμε να δούμε ποιος είναι ο τύπος ενός αντικειμένου.

```
> x<-"STATISTICS"
> class(x)
[1] "character"
```

Βασικές Δομές Αντικειμένων

- Οι κύριες δομές των αντικειμένων στην R είναι:
  - Διανύσματα (*vectors*).
  - Δισδιάστατοι Πίνακες (*matrices*).
  - Πολυδιάστατοι Πίνακες (*arrays*).
  - Πλαίσια Δεδομένων (*data frames*).
  - Λίστες (*lists*).
- Στις 3 πρώτες δομές τα αντικείμενα πρέπει να είναι του ίδιου τύπου.
- Οι λίστες μπορούν να περιέχουν ως στοιχεία διαφορετικές δομές αντικειμένων.
- Η εντολή **class()** επιστρέφει για πίνακες, πλαίσια δεδομένων και λίστες τη δομή του αντικειμένου, ενώ για διανύσματα τον τύπου του διανύσματος.

Διανύσματα

- Τα διανύσματα στην R είναι σύνολα που περιέχουν αντικείμενα του ίδιου τύπου. Ανάλογα με το είδος των αντικειμένων έχουμε
  - Αριθμητικά Διανύσματα.
  - Διανύσματα Χαρακτήρων.
  - Λογικά Διανύσματα.
  - Διανύσματα Κατηγοριών.

Αριθμητικά Διανύσματα

- Ο πιο εύκολος τρόπος δημιουργίας ενός αριθμητικού διανύσματος είναι μέσω της εντολής **c()**. Τα στοιχεία μέσα στην εντολή διαχωρίζονται με κόμμα. Π.χ.

```
> x<-c(1,2,3,4,5)
> x
[1] 1 2 3 4 5
```

- Επίσης η εντολή **c()** μπορεί να λάβει ως όρισμα ένα ήδη ορισμένο διάνυσμα. Π.χ.

```
> x<-c(1,2,3,4,5)
> y<-c(6,7)
> z<-c(10,x,y)
> z
[1] 10  1  2  3  4  5  6  7
```

Αριθμητικά Διανύσματα

- Με την εντολή **numeric()** (ή ισοδύναμα με την εντολή **double()**) δημιουργούμε ένα διάνυσμα μηδενικών τιμών. Ως όρισμα η εν λόγω συνάρτηση δέχεται το μήκος του διανύσματος που θέλουμε να δημιουργηθεί

  ```
  > numeric(3)
  [1] 0 0 0
  ```

- Η εντολή **is.numeric()** ελέγχει αν ένα διάνυσμα είναι αριθμητικό

  ```
  > x<-c(1,2,3,4,5)
  > is.numeric(x)
  [1] TRUE
  ```

Αριθμητικά Διανύσματα

- Χρήσιμες συναρτήσεις για αριθμητικά διανύσματα

  - **Μήκος Διανύσματος**

    ```
    > x<-c(1,2,3,4,5)
    > length(x)
    [1] 5
    ```

  - **Ελάχιστη/Μέγιστη τιμή**

    ```
    > min(x)
    [1] 1
    > max(x)
    [1] 5
    ```

  - **Άθροισμα/Γινόμενο τιμών**

    ```
    > sum(x)
    [1] 15
    > prod(x)
    [1] 120
    ```

Αριθμητικά Διανύσματα

- **Αντιστροφή της Σειράς των Τιμών ενός Διανύσματος**

  > x<-c(3,5,2,1,6)

  > rev(x)

  [1] 6 1 2 5 3

- **Πρόσημο των Τιμών ενός Διανύσματος**

  > x<-c(3,-5,2,1,-6)

  > sign(x)

  [1] 1 -1 1 1 -1

- **Ταξινόμηση τιμών Διανύσματος κατά Αύξουσα/Φθίνουσα Τάξη Μεγέθους**

  > x<-c(3,5,2,1,6)

  > sort(x)

  [1] 1 2 3 5 6

  > sort(x, decreasing=T)

  [1] 6 5 3 2 1

Αριθμητικά Διανύσματα

- **Θέση των ταξινομημένων κατά Αύξουσα Τάξη Μεγέθους Τιμών**

```
> x<-c(3,5,2,1,6)
> order(x)
[1] 4 3 1 2 5
```

Δηλαδή η μικρότερη παρατήρηση βρίσκεται στην 4 θέση, η αμέσως μεγαλύτερη στην 3, κ.λ.π. Με τις εντολή which.min(x) και which.max(x) λαμβάνουμε τις θέσεις της μικρότερης και της μεγαλύτερης παρατήρησης.

- **Σειρά κατάταξης τιμών**

```
> x<-c(3,5,2,1,6)
> rank(x)
[1] 3 4 2 1 5
```

Δηλαδή η πρώτη τιμή του x (το 3) βρίσκεται στην $3^{η}$ θέση στο ταξινομημένο κατά αύξουσα τάξη μεγέθους διάνυσμα, η δεύτερη τιμή (το 5) στην $4^{η}$ θέση στο ταξινομημένο κατά αύξουσα τάξη μεγέθους διάνυσμα, κ.λ.π.

Αριθμητικά Διανύσματα

- **Σειρά κατάταξης τιμών σε περιπτώσεις ισοπαλιών (ties)**
  - ***Average*: Σε περιπτώσεις ισοπαλιών η τελική σειρά κατάταξης προκύπτει από το μέσο όρο των σειρών κατάταξης των παρατηρήσεων με τις ίδιες τιμές, π.χ.**

    ```
    > x2<-c(3,5,2,1,6,3)
    > rank(x2, ties.method="average")
    [1] 3.5 5.0 2.0 1.0 6.0 3.5
    ```
  - ***First*: Η πρώτη παρατήρηση σε σειρά εμφάνισης παίρνει την χαμηλότερη σειρά κατάταξης, π.χ.**

    ```
    > rank(x2, ties.method="first")
    [1] 3 5 2 1 6 4
    ```
  - ***Random*: Τυχαία προκύπτει η σειρών κατάταξης των παρατηρήσεων με τις ίδιες τιμές, π.χ.**

    ```
    > rank(x2, ties.method="random")
    [1] 4 5 2 1 6 3
    > rank(x2, ties.method="random")
    [1] 3 5 2 1 6 4
    ```
  - ***Min/Max*: Δίνεται η μικρότερη/μεγαλύτερη σειρά κατάταξης στις παρατηρήσεις με τις ίδιες τιμές, π.χ.**

    ```
    > rank(x2, ties.method="min")
    [1] 3 5 2 1 6 3
    > rank(x2, ties.method="max")
    [1] 4 5 2 1 6 4
    ```

Αριθμητικά Διανύσματα

- **Ορισμός ονομάτων στις τιμές του αριθμητικού διανύσματος.**

```
> weight<-c(70, 57, 68, 82)
> names(weight)
NULL
> names(weight)<-c("Mary", "Kelly", "Elena", "George")
> names(weight)
[1] "Mary"   "Kelly"  "Elena"  "George"
> weight
Mary  Kelly  Elena George
  70     57     68     82
```

  - Εναλλακτικά μπορούμε να ορίσουμε το διάνυσμα με τα ονόματα εξαρχής:

```
> weight<-c(Mary=70, Kelly=57, Elena=68, George=82)
> weight
Mary  Kelly  Elena George
  70     57     68     82
```

Αριθμητικά Διανύσματα

- **Ελλιπείς τιμές**. Συμβολίζονται στην R με το σύμβολο NA.

```
> x3<-c(1,2,3,NA,9)
> x3
[1]  1  2  3 NA  9
```

  - Με την εντολή **is.na()** ελέγχουμε ποια τιμή είναι ελλιπής.

```
> is.na(x3)
[1] FALSE FALSE FALSE  TRUE FALSE
```

Αριθμητικά Διανύσματα

- **Δημιουργία Αριθμητικών Ακολουθιών**
  - Με την εντολή α:β δημιουργούμε ακολουθίες τιμών από το α στο β με βήμα την μονάδα. Π.χ.

```
> x<-1:10
> x
[1]  1  2  3  4  5  6  7  8  9 10
> x<--4:10
> x
[1] -4 -3 -2 -1  0  1  2  3  4  5  6  7  8  9 10
> x<-6:1
> x
[1] 6 5 4 3 2 1
> x<-3.3:10.3
> x
[1]  3.3  4.3  5.3  6.3  7.3  8.3  9.3 10.3
```

  - Αν η διαφορά των α-β δεν είναι ακέραιος αριθμός, τότε η R δημιουργεί πάλι ακολουθία με βήμα την μονάδα, ξεκινώντας από το α και σταματώντας πριν το β. Π.χ.

```
> x<-3.3:6.9
> x
[1] 3.3 4.3 5.3 6.3
```

Αριθμητικά Διανύσματα

- Για βήμα διαφορετικό της μονάδας μπορεί να χρησιμοποιηθεί η συνάρτηση **seq()**. Ως παραμέτρους η εν λόγω συνάρτηση παίρνει τον πρώτο όρο (**from**) και τελευταίο όρο (**to**) της ακολουθίας, το βήμα της ακολουθίας (**by**), το μήκος της ακολουθίας (**length**) ή το όνομα ενός άλλου διανύσματος (**along**) έτσι ώστε η ακολουθία να έχει ίδιο μήκος με αυτό το διάνυσμα. Η εν λόγω συνάρτηση χρειάζεται 3 από τις παραπάνω αυτές παραμέτρους, ενώ αν δοθούν μόνο 2 η R θεωρεί την παράμετρο by = 1.

```
> seq(from=1,to=9, by=2)
[1] 1 3 5 7 9
> seq(from=1,to=9, length=3)
[1] 1 5 9
> seq(to=9, length=3)
[1] 7 8 9
> seq(from=1,by=2,length=10)
[1]  1  3  5  7  9 11 13 15 17 19
> y<-1:10
> seq(from=1,by=2,along=y)
[1]  1  3  5  7  9 11 13 15 17 19
```

- Αν το πηλίκο της διαφοράς του τελευταίου από τον πρώτο όρο προς το βήμα δεν είναι ακέραιος αριθμός η R θα σταματήσει πριν τον τελευταίο όρο.

```
> seq(from=1,to=10,by=2)
[1] 1 3 5 7 9
```

Αριθμητικά Διανύσματα

- **Επαναλήψεις τιμών ή διανυσμάτων**. Με την εντολή **rep()** μπορούμε να επαναλάβουμε μια τιμή ή ένα διάνυσμα όσες φορές θέλουμε. Ως παραμέτρους δέχεται πρώτα την τιμή ή το διάνυσμα που θέλουμε να επαναλάβουμε και εν συνεχεία τον αριθμό επαναλήψεων της τιμής ή του διανύσματος (**times**) ή τον αριθμό επαναλήψεων κάθε στοιχείου του διανύσματος (**each**).

```
> rep(2,5)
[1] 2 2 2 2 2
> x<-c(1,2,3)
> rep(x,5)
[1] 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3
> rep(x, each=5)
[1] 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3
```

Αριθμητικά Διανύσματα

- **Πράξεις Διανυσμάτων**. Μπορούμε να κάνουμε πράξεις μεταξύ διανυσμάτων (προσοχή να είναι ίδιας διάστασης), μεταξύ αριθμών και διανυσμάτων, όπως και να εφαρμόσουμε αριθμητικές συναρτήσεις σε διανύσματα. Π.χ.

```
> x<-c(1,2,3)
> x*3
[1] 3 6 9
> x^2
[1] 1 4 9
> y<-c(4,5,6)
> y/x
[1] 4.0 2.5 2.0
```

Αριθμητικά Διανύσματα

- Μπορούμε εύκολα να επιλέξουμε **συγκεκριμένα στοιχεία** ενός διανύσματος. Αν π.χ. το διάνυσμα είναι το x και θέλουμε το πρώτο στοιχείο του, τότε το καλούμε με x[1].

```
> x<-seq(from=1,to=9,by=2)
> x
[1] 1 3 5 7 9
> x[2]
[1] 3
> x[2:4]
[1] 3 5 7
> x[c(1,3)]
[1] 1 5
> x[-c(1,3)]
[1] 3 7 9
```

Αριθμητικά Διανύσματα

- Αν οι τιμές του διανύσματος έχουν ονόματα μπορούμε ισοδύναμα να τα χρησιμοποιήσουμε, π.χ.

```
> weight
Mary  Kelly  Elena George
 70     57     68     82
> weight[1]
Mary
70
> weight['Mary']
Mary
 70
```

Αριθμητικά Διανύσματα

- Οι επόμενες εντολές είναι χρήσιμες στις συγκρίσεις διανυσμάτων. Μπορούν με την ίδια σύνταξη να χρησιμοποιηθούν σε οποιαδήποτε διανύσματα (όχι μόνο αριθμητικά).
  - which(): Εντοπίζει τη σειρά των στοιχείων με συγκεκριμένα χαρακτηριστικά.

```
> x<-c(5,1,2,6,7,4)
> which(x==max(x)) # same as which.max(x)
[1] 5
> which(x>3)
[1] 1 4 5 6
```

  - match(): Εντοπίζει κοινά στοιχεία μεταξύ διανυσμάτων.

```
> x<-c(5,1,2,6,7,4)
> y<-c(6,9,0,1,3)
> match(x,y)
[1] NA  4 NA  1 NA NA
> match(y,x)
[1]  4 NA NA  2 NA
```

Το 2ο στοιχείο του x υπάρχει στην 4 θέση στο y
Το 4ο στοιχείο του x υπάρχει στην 1 θέση στο y

Το 1ο στοιχείο του y υπάρχει στην 4 θέση στο x
Το 4ο στοιχείο του y υπάρχει στην 2 θέση στο x

Αριθμητικά Διανύσματα

- Αντίστοιχη είναι και η εντολή **%in%** μόνο που το διάνυσμα που επιστρέφεται είναι λογικό, π.χ.
  - ```
    > x<-c(5,1,2,6,7,4)
    > y<-c(6,9,0,1,3)
    > x%in%y
    [1] FALSE  TRUE FALSE  TRUE FALSE FALSE
    > y%in%x
    [1]  TRUE FALSE FALSE  TRUE FALSE
    ```

  Η εντολή **%in%** είναι ισοδύναμη με την εντολή **is.element(x,y).**
- **intersect()** και **union()**. Τομή και ένωση αντίστοιχα δύο διανυσμάτων. Π.χ.
  - ```
    > x<-c(5,1,2,6,7,4)
    > y<-c(6,9,0,1,3)
     > intersect(x,y)
    [1] 1 6
    > union(x,y)
    [1] 5 1 2 6 7 4 9 0 3
    ```

Αριθμητικά Διανύσματα

- **setdiff().** Κρατάει τα στοιχεία του πρώτου διανύσματος που δεν ανήκουν στο δεύτερο διάνυσμα. Π.χ.

```
> x<-c(5,1,2,6,7,4)
> y<-c(6,9,0,1,3)
> setdiff(x,y)
[1] 5 2 7 4
> setdiff(y,x)
[1] 9 0 3
```

- **unique().** Η εντολή διαγράφει τα κοινά στοιχεία του δοθέντος διανύσματος. Π.χ.

```
> x<-c(5,1,2,6,7,4,5,5,6)
> unique(x)
[1] 5 1 2 6 7 4
```

- **duplicated().** Η εντολή διαγράφει τα κοινά στοιχεία του δοθέντος διανύσματος. Ωστόσο, επιστρέφει ένα λογικό διάνυσμα. Π.χ.

```
> duplicated(x)
[1] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE
```

Διανύσματα Χαρακτήρων

- Τα διανύσματα χαρακτήρων μπορούν να δημιουργηθούν και πάλι με την εντολή **c()**.
  - > x<-c('Dimitris', 'Giorgos')
    > x
    [1] "Dimitris" "Giorgos"
- **Υπάρχει μεγάλη ποικιλία συναρτήσεων για διανύσματα χαρακτήρων.** Μερικές από αυτές είναι οι ακόλουθες:
  - **character(length)**
    > character(length=2)
    [1] "" ""
  - **as.character()**
    > x<-1:10
    > as.character(x)
     [1] "1" "2" "3" "4" "5" "6" "7" "8" "9" "10"

Διανύσματα Χαρακτήρων

- **is.character()**

```
> y<-as.character(x)
 [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"
     "8"  "9"  "10"
> is.character(y)
[1] TRUE
> x<-c("-0.1","2.7","B")
> x
[1] "-0.1" "2.7"  "B"
> is.character(x)
[1] TRUE
> x<-as.numeric(x)
Warning message:
NAs introduced by coercion
> x
[1] -0.1  2.7   NA
> is.character(x)
[1] FALSE
```

- **noquote()**

```
> y<-as.character(x)
 [1] "1"  "2"  "3"  "4"  "5"
     "6"  "7"  "8"  "9"  "10"
> noquote(y)
[1] 1  2  3  4  5  6  7  8  9
    10
```

- **nchar()**

```
> y<-as.character(x)
 [1] "1"  "2"  "3"  "4"  "5"
     "6"  "7"  "8"  "9"  "10"
> nchar(y)
 [1] 1 1 1 1 1 1 1 1 1 2
```

Διανύσματα Χαρακτήρων

- **paste()**

```
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> paste(x)
 [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"
     "9"  "10"
> paste(`Mathematical',`Statistics')
[1] "Mathematical Statistics"
> paste(`3',`5',`8', sep="+")
[1] "3+5+8"
> paste(paste(3,5, sep=` + '), 8, sep=` =
    ')
[1] "3 + 5 = 8"
> paste(`Chapter',2, sep=" ")
[1] "Chapter 2"
> paste("Today is", date())
[1] "Today is Mon Jun 03 20:42:41 2013"
> a<-c(`Kwstas', `Maria')
> b<-c(`Papadopoulos', `Kyriakou')
> paste(a,b)
[1] "Kwstas Papadopoulos" "Maria Kyriakou"
> paste(b,a, sep=`, ')
[1] "Papadopoulos, Kwstas" "Kyriakou, Maria"
> paste("Chapter", 1:2, sep=" ")
[1] "Chapter 1" "Chapter 2"
> a<-c(`Kwstas', `Maria')
> b<-c(`Papadopoulos', `Kyriakou', `Anagnostou')
> paste(a,b)
[1] "Kwstas Papadopoulos" "Maria Kyriakou"
    "Kwstas Anagnostou"
> a<-c(`Kwstas', `Maria')
> paste(a, collapse=",")
[1] "Kwstas,Maria"
> paste(1:10, collapse=`+')
[1] "1+2+3+4+5+6+7+8+9+10"
> b<-c(`Papadopoulos', `Kyriakou', `Anagnostou')
> paste(a, b, collapse=", ")
[1] "Kwstas Papadopoulos, Maria Kyriakou, Kwstas
Anagnostou"
```

Διανύσματα Χαρακτήρων

- **strsplit()**

```
> x<-c("Statistics",
      "Mathematics")
> strsplit(x,split="a")
[[1]]
[1] "St"      "tistics"
[[2]]
[1] "M"    "them" "tics"
> strsplit(x, split='')
[[1]]
 [1] "S" "t" "a" "t" "i" "s" "t" "i" "c"
      "s"
[[2]]
 [1] "M" "a" "t" "h" "e" "m" "a" "t"
      "i" "c" "s"
> strsplit(x, split="th")
[[1]]
[1] "Statistics"
[[2]]
[1] "Ma"      "ematics"
```

- **substr()**

```
> substr("abcdef",2,4)
[1] "bcd"
> x<-c("Statistics", "Mathematics")
> substr(x,2,4)
[1] "tat" "ath"
```

Διανύσματα Χαρακτήρων

- **grep()**

```
> countries<-c("Greece", "United States",
"United Kingdom", "Italy",
"France", "United Arab Emirates")
> grep("United", countries)
[1] 2 3 6
> grep("United", countries, value=TRUE)
[1] "United States"        "United Kingdom"
"United Arab Emirates"
> data[grep("United", data$country), ]
                country           gdp  income continent
23  United Arab Emirates  21000   24200      AS
42        United Kingdom    28300   29400      EU
82         United States      35200   31200      NA
```

- **toupper () & tolower()**

```
> x
[1] "Statistics"  "Mathematics"
> tolower(x)
[1] "statistics"  "mathematics"
> toupper(x)
[1] "STATISTICS"  "MATHEMATICS"
```

Διανύσματα Χαρακτήρων

- **sub() & gsub()**

```
> values<-c("1,700", "2,300")
> as.numeric(values)
[1] NA NA
Warning message:
NAs introduced by coercion
> as.numeric(gsub(",","",values))
[1] 1700 2300
> as.numeric(sub(",","",values))
[1] 1700 2300
> sub(",","",values)
[1] "1700" "2300"
> gsub(",","",values)
[1] "1700" "2300"
> values<-c("1,000,000", "2,000,000")
> sub(",","",values)
[1] "1000,000" "2000,000"
> gsub(",","",values)
[1] "1000000" "2000000"
```

Λογικά Διανύσματα

- **Βασικές συναρτήσεις**

```
> logical(3)   (δημιουργεί διάνυσμα με ψευδείς τιμές)
[1] FALSE FALSE FALSE
> as.logical(c(0:10)) (μετατρέπει αριθμητικά διανύσματα σε λογικά. Η τιμή 0 μετατρέπεται σε False και όλες οι άλλες σε True.)
[1] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
    TRUE  TRUE  TRUE  TRUE  TRUE
```

Διανύσματα Κατηγοριών

- Κατηγορικές Μεταβλητές δίνονται στην R με την βοήθεια της εντολής **factor**.

```
> gender<-c('Male', 'Female', 'Male', 'Male', 'Female')
> gender
[1] "Male"   "Female" "Male"   "Male"   "Female"
> factor(gender)
[1] Male   Female Male   Male   Female
Levels: Female Male
> levels(factor(gender))
[1] "Female" "Male"
```

- Αν η μεταβλητή είναι διάταξης τότε χρησιμοποιούμε την εντολή **ordered**.

```
> opinion<-c('Low', 'Low', 'High', 'High', 'High', 'Medium')
> ordered(opinion, levels=c('Low', 'Medium', 'High'))
[1] Low    Low    High   High   High   Medium
Levels: Low < Medium < High
```

Δισδιάστατοι Πίνακες

- Ένας **δισδιάστατος πίνακας** (matrix) είναι μια δομή δεδομένων της οποίας τα στοιχεία είναι διατεταγμένα σε γραμμές και στήλες. Για να τους δημιουργήσουμε χρησιμοποιούμε την εντολή **matrix()** με παραμέτρους τα στοιχεία (μπορεί να είναι αριθμοί ή χαρακτήρες ή λογικές τιμές) και τον αριθμό γραμμών (**nrow**) ή στηλών (**ncol**). Επίσης δηλώνουμε αν θέλουμε τα στοιχεία να διαβαστούν κατά στήλη (προκαθορισμένη τιμή) ή κατά γραμμή (**byrow=T**).

Δισδιάστατοι Πίνακες

```
> x<-1:10
> X<-matrix(x, ncol=2)
> X
     [,1] [,2]
[1,]    1    6
[2,]    2    7
[3,]    3    8
[4,]    4    9
[5,]    5   10
> X<-matrix(x, nrow=5)
> X
     [,1] [,2]
[1,]    1    6
[2,]    2    7
[3,]    3    8
[4,]    4    9
[5,]    5   10
> X<-matrix(x, nrow=5, byrow=T)
> X
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
[4,]    7    8
[5,]    9   10
```

Δισδιάστατοι Πίνακες

- Η διάσταση του πίνακα δίνεται με την εντολή **dim()**

```
> dim(X)
[1] 5 2
```

- Μπορούμε εύκολα να δούμε κάποιο ή κάποια στοιχεία ενός πίνακα απλά δίνοντας την θέση του μέσα σε [].

```
> X[3,2]
[1] 6
```

- Επίσης μπορούμε να δούμε μια γραμμή ή μια στήλη του πίνακα, παραλείποντας την διάσταση για την οποία δεν ενδιαφερόμαστε

```
> X[3,]
[1] 5 6
> X[,2]
[1]  2  4  6  8 10
```

Δισδιάστατοι Πίνακες

- Με τις εντολές **rbind** και **cbind** δημιουργούμε πίνακες συνενώνοντας ως στήλες ή ως γραμμές αντίστοιχα διανύσματα.

```
> x1<-1:5
> x2<-6:10
> cbind(x1,x2)
     x1 x2
[1,]  1  6
[2,]  2  7
[3,]  3  8
[4,]  4  9
[5,]  5 10
> rbind(x1,x2)
     [,1] [,2] [,3] [,4] [,5]
x1    1    2    3    4    5
x2    6    7    8    9   10
```

Δισδιάστατοι Πίνακες

- Μπορούμε να δημιουργήσουμε διαγώνιους πίνακες με την εντολή **diag()**.

```
> diag(1:5)
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    0    0    0    0
[2,]    0    2    0    0    0
[3,]    0    0    3    0    0
[4,]    0    0    0    4    0
[5,]    0    0    0    0    5
```

Δισδιάστατοι Πίνακες

- Για την δημιουργία ενός ταυτοτικού πίνακα χρησιμοποιούμε πάλι την εντολή **diag()**

```
> diag(5)
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    0    0    0    0
[2,]    0    1    0    0    0
[3,]    0    0    1    0    0
[4,]    0    0    0    1    0
[5,]    0    0    0    0    1
```

Δισδιάστατοι Πίνακες

- Πράξεις Πινάκων. Μπορούμε να χρησιμοποιήσουμε τους αριθμητικούς τελεστές και τις μαθηματικές συναρτήσεις της R για πράξεις μεταξύ πινάκων ή μεταξύ πινάκων και διανυσμάτων ή μεταξύ πινάκων και αριθμών. Οι μόνοι νέοι τελεστές, ειδικά για πίνακες είναι οι ακόλουθοι:

Δισδιάστατοι Πίνακες

Σύμβολο	**Πράξη**
%*%	Πολλαπλασιασμός Πινάκων
t()	Ανάστροφος Πίνακα
solve()	Αντίστροφος Πίνακα
eigen()	Ιδιοτιμές και Ιδιοδιανύσματα Πίνακα
det()	Ορίζουσα Πίνακα

Δισδιάστατοι Πίνακες

```
> x<-matrix(c(1,2,3,4,5,6), ncol=2)
> x
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
> dim(x)
[1] 3 2
> y<-matrix(c(0,1,1,1), ncol=2)
> y
     [,1] [,2]
[1,]    0    1
[2,]    1    1
```

```
> x%*%y
     [,1] [,2]
[1,]    4    5
[2,]    5    7
[3,]    6    9
> t(x)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
> solve(y)
     [,1] [,2]
[1,]   -1    1
[2,]    1    0
```

Δισδιάστατοι Πίνακες

- > y<-matrix(c(0,1,1,1),ncol=2)
- > eigen(y)

```
$values
[1]  1.618034 -0.618034

$vectors
          [,1]       [,2]
[1,] 0.5257311 -0.8506508
[2,] 0.8506508  0.5257311
```

- > det(y)

```
[1] -1
```

- > prod(eigen(y)$values)

```
[1] -1
```

Δισδιάστατοι Πίνακες

- colSums(), rowSums(), colMeans(), rowMeans() & max.col().

- > x<-
matrix(c(1,2,3,4,5,6),ncol=2)
- > colSums(x)
  [1] 6 15
- > rowSums(x)
  [1] 5 7 9

Για κάθε γραμμή επέστρεψε την θέση (στήλη) του μέγιστου.

- > colMeans(x)
  [1] 2 5
- > rowMeans(x)
  [1] 2.5 3.5 4.5
- > x
     [,1] [,2]
  [1,] 1 4
  [2,] 2 5
  [3,] 3 6
- > max.col(x)
  [1] 2 2 2

Δισδιάστατοι Πίνακες

- Η εντολή apply.

```
> x<-matrix(c(1,2,3,4,5,6), ncol=2)
> x
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
> apply(x,1,sum)
[1] 5 7 9
> apply(x,2,sum)
[1]  6 15
```

Άθροισε τα στοιχεία του πίνακα ως προς όλες τις γραμμές.

Άθροισε τα στοιχεία του πίνακα ως προς όλες τις στήλες.

Πολυδιάστατοι Πίνακες

- Οι **πολυδιάστατοι πίνακες** (arrays) είναι πίνακες με 3 ή περισσότερες διαστάσεις. Δημιουργούνται με την εντολή array() και το μέγεθος κάθε διάστασης δηλώνεται από την παράμετρο dim. Π.χ. αν θέσουμε dim=c(2,3,4), θα έχουμε έναν 3-διάστατο πίνακα με μέγεθος διαστάσεων 2 (nrow), 3 (ncol) και 4.

Πολυδιάστατοι Πίνακες

```
> X<-
    array(c(1:12,36:48
    ),dim=c(2,3,4))
> X
, , 1

     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6

, , 2

     [,1] [,2] [,3]
[1,]    7    9   11
[2,]    8   10   12

, , 3

     [,1] [,2] [,3]
[1,]   36   38   40
[2,]   37   39   41

, , 4

     [,1] [,2] [,3]
[1,]   42   44   46
[2,]   43   45   47
```

Πλαίσια Δεδομένων

- Τα **πλαίσια δεδομένων** (data frames) είναι δισδιάστατοι πίνακες στους οποίους δεν χρειάζεται οι στήλες να είναι όλες του ίδιου τύπου. Συνήθως στα πλαίσια δεδομένων κατοχυρώνουμε τις παρατηρήσεις που έχουμε συλλέξει από ένα δείγμα.
- Ένα πλαίσιο δεδομένων δημιουργείται με την εντολή data.frame().

Πλαίσια Δεδομένων

```
> Gender<-c('Male', 'Male', 'Male', 'Female')
> Gender<-factor(Gender)
> Gender
[1] Male   Male   Male   Female
Levels: Female Male
> Smoking<-c(T, T, F, F)
> Smoking
[1]  TRUE  TRUE FALSE FALSE
> Choresterol<-c(200, 220, 180, 172)
> Choresterol
[1] 200 220 180 172
> sample<-data.frame(Gender, Smoking, Choresterol)
> sample
  Gender Smoking Choresterol
1   Male    TRUE         200
2   Male    TRUE         220
3   Male   FALSE         180
4 Female   FALSE         172
```

Πλαίσια Δεδομένων

- Με την εντολή as.data.frame() μπορείτε να μετατρέψετε έναν δισδιάστατο πίνακα σε πλαίσιο δεδομένων. Με την εντολή **names()** μπορείτε να δώσετε ονόματα στις στήλες (μεταβλητές) του πλαισίου σας. Επίσης με την παράμετρο **row.names()** της εντολής data.frame μπορείτε να δώσετε ονόματα και στις γραμμές (παρατηρήσεις).

Πλαίσια Δεδομένων

```
> x<-matrix(c(1,1,200,1,1,220,
      1,0,180,0,0,172), ncol=3, byrow=T)
> x
     [,1] [,2] [,3]
[1,]    1    1  200
[2,]    1    1  220
[3,]    1    0  180
[4,]    0    0  172
> x<-as.data.frame(x)
> x
  V1 V2  V3
1  1  1 200
2  1  1 220
3  1  0 180
4  0  0 172
> names(x)
[1] "V1" "V2" "V3"
> names(x)<-c('Gender', 'Smoking',
      'Choresterol')
> x
  Gender Smoking Choresterol
1      1       1         200
2      1       1         220
3      1       0         180
4      0       0         172

> x<-data.frame(x, row.names=c('obs1',
      'obs2', 'obs3', 'obs4') )
> x
     Gender Smoking Choresterol
obs1      1       1         200
obs2      1       1         220
obs3      1       0         180
obs4      0       0         172
```

Πλαίσια Δεδομένων

- Ό,τι εντολές χρησιμοποιήσαμε στους πίνακες μπορούμε να χρησιμοποιήσουμε και εδώ.

```
> x
     Gender Smoking Choresterol
obs1      1       1         200
obs2      1       1         220
obs3      1       0         180
obs4      0       0         172
> dim(x)
[1] 4 3
> x[1,]
Gender Smoking Choresterol
obs1      1       1         200
> x[1,2]
[1] 1
> x$Gender
[1] 1 1 1 0
> rbind(1,x)
  Gender Smoking Choresterol
1      1       1         1
obs1   1       1         200
obs2   1       1         220
obs3   1       0         180
obs4   0       0         172
> cbind(1,x)
     1 Gender Smoking Choresterol
obs1 1      1       1         200
obs2 1      1       1         220
obs3 1      1       0         180
obs4 1      0       0         172
```

Πλαίσια Δεδομένων

- **subset()**

```
> sample<-data.frame(Gender,
    Smoking, Choresterol)
> sample
  Gender Smoking Choresterol
1   Male    TRUE         200
2   Male    TRUE         220
3   Male   FALSE         180
4 Female   FALSE         172
> subset(sample, Choresterol >
    mean(Choresterol))
  Gender Smoking Choresterol
1   Male    TRUE         200
2   Male    TRUE         220
> subset(sample,
    Gender=="Female")
  Gender Smoking Choresterol
4 Female   FALSE         172
```

Πλαίσια Δεδομένων

- **transform()**

```
> transform(sample, Choresterol2 =
      Choresterol^2)
```

	Gender	Smoking	Choresterol	Choresterol2
1	Male	TRUE	200	40000
2	Male	TRUE	220	48400
3	Male	FALSE	180	32400
4	Female	FALSE	172	29584

```
> country<-c("GR","GR","IT","UK")
> transform(sample, Country = country)
```

	Gender	Smoking	Choresterol	Country
1	Male	TRUE	200	GR
2	Male	TRUE	220	GR
3	Male	FALSE	180	IT
4	Female	FALSE	172	UK

Πλαίσια Δεδομένων

- **merge()**

```
> sample1
  Gender Smoking Choresterol
1   Male    TRUE         200
2   Male    TRUE         220
3   Male   FALSE         180
4 Female   FALSE         172
> sample2
  Choresterol
1         220
2         199
3         187
```

```
> merge(sample1,sample2)
  Choresterol Gender Smoking
1         220   Male    TRUE
> merge(sample1,sample2,all=TRUE)
  Choresterol Gender Smoking
1         172 Female   FALSE
2         180   Male   FALSE
3         187   <NA>      NA
4         199   <NA>      NA
5         200   Male    TRUE
6         220   Male    TRUE
```

Λίστες

- Οι **λίστες** (lists) είναι διανύσματα των οποίων τα στοιχεία δεν είναι ανάγκη να είναι της ίδιας δομής. Δημιουργούνται με την εντολή list δίνοντας ως παραμέτρους τα αντικείμενα που θέλουμε να τα πλαισιώνουν μαζί με τα ονόματά τους.

Λίστες

```
> Gender<-c('Male', 'Male', 'Male',
      'Female')
> Gender<-factor(Gender)
> Gender
[1] Male   Male   Male   Female
Levels: Female Male
> x<-1:10
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> sample
  Gender Smoking Choresterol
1   Male    TRUE         200
2   Male    TRUE         220
3   Male   FALSE         180
4 Female   FALSE         172
> y<-list(my_sample=sample, x=x,
      the_gender=Gender)
```

```
> y
$my_sample
  Gender Smoking Choresterol
1   Male    TRUE         200
2   Male    TRUE         220
3   Male   FALSE         180
4 Female   FALSE         172

$x
 [1]  1  2  3  4  5  6  7  8  9 10

$the_gender
[1] Male   Male   Male   Female
Levels: Female Male
```

Λίστες

- Με το σύμβολο $ ή την διπλή αγκύλη [[ ]], μπορούμε να δούμε τα επιμέρους στοιχεία μιας λίστας.

```
> y$x
[1]  1  2  3  4  5  6  7  8  9 10
> y[[3]]
[1] Male   Male   Male   Female
Levels: Female Male
> y$x[1:3]
[1] 1 2 3
```

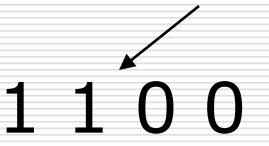
Αποθήκευση Αντικειμένων

- Υπάρχουν αρκετοί τρόποι αποθήκευσης αντικειμένων της R στον σκληρό δίσκο. Αν δώσουμε μόνο το όνομα του αρχείου όπου θα γίνει η αποθήκευση, το εν λόγω αρχείο δημιουργείται στον φάκελο από όπου τρέχουμε την R. Αν θέλουμε η αποθήκευση να γίνει κάπου αλλού τότε πρέπει να δώσουμε την πλήρη διαδρομή. Ένας από τους ευκολότερους τρόπους να αποθηκεύσουμε αντικείμενα είναι με χρήση της εντολής **write()**.

Αποθήκευση Αντικειμένων

- Διανύσματα. Μπορείτε να αποθηκεύσετε διανύσματα (αριθμητικά, χαρακτήρων ή λογικά) με την εντολή **write()**.

```
> Gender<-c("Male", "Male", "Male", "Female")
> write(Gender,file="g.txt", ncol=4)
> x
[1]  1  2  3  4  5  6  7  8  9 10
> write(x,file="x.txt", ncol=length(x))
> Smoking
[1]  TRUE  TRUE FALSE FALSE
> write(Smoking,file="s.txt", ncol=4)
```



Αποθήκευση Αντικειμένων

- Δισδιάστατοι Πίνακες. Με την εντολή **write()** μπορείτε επίσης να αποθηκεύσετε πίνακες 2 διαστάσεων. Εδώ χρειάζεται **προσοχή** διότι η εντολή αναστρέφει τον πίνακα, οπότε ζητούμε αποθήκευση του ανάστροφου.

```
> X
     [,1] [,2]
[1,]    1    7
[2,]    2    8
[3,]    3    9
[4,]    4   10
[5,]    5   11
[6,]    6   12
> write(t(X), "X.txt", ncol=2)
```

Αποθήκευση Αντικειμένων

- Πλαίσια Δεδομένων. Τα πλαίσια δεδομένων τα αποθηκεύουμε με την εντολή **write.table()**.

```
> sample
  Gender Smoking Choresterol
1   Male    TRUE         200
2   Male    TRUE         220
3   Male   FALSE         180
4 Female   FALSE         172
> write.table(sample, file="sample.txt")
```

Ανάκτηση Δεδομένων

- Μπορούμε εύκολα στην R να διαβάσουμε δεδομένα από ένα αρχείο του σκληρού μας δίσκου. Όπως και στην αποθήκευση έτσι και εδώ πρέπει να δώσουμε την πλήρη διαδρομή του αρχείου από όπου θέλουμε να ανακτήσουμε δεδομένα, εκτός και αν το αρχείο βρίσκεται στον φάκελο που δουλεύουμε την R οπότε το όνομά του αρκεί. Για διανύσματα και δισδιάστατους πίνακες χρησιμοποιούμε την εντολή scan().

Ανάκτηση Δεδομένων

```
> x<-scan("x.txt")
Read 10 items
> x
[1]  1  2  3  4  5  6  7  8  9 10
> X<-matrix(scan("XX.txt"), ncol=2, byrow=T)
Read 12 items
> X
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
[4,]    7    8
[5,]    9   10
[6,]   11   12
```

Αρχείο από όπου θα πάρει τις τιμές ο πίνακας

Να διαβάσει τις τιμές ανά γραμμή

στηλών του πίνακα

Ανάκτηση Δεδομένων

- Επίσης με την εντολή scan μπορούμε να εισάγουμε δεδομένα και με το πληκτρολόγιο.

```
> z<-scan()
1: 2
2: 4
3: 6
4: 8
5: 2
6:
Read 5 items
> z
[1] 2 4 6 8 2
```

Ανάκτηση Δεδομένων

- Υπάρχουν μερικές συναρτήσεις για εισαγωγή δεδομένων στην R.
  - **read.csv**, για αρχεία στα οποία τα δεδομένα διαχωρίζονται με κόμμα.
  - **read.delim**, για δεδομένα τα οποία βρίσκονται σε αρχεία διαχωρισμένα με στηλοθέτες.
  - **readLines**, για εισαγωγή γραμμών ενός αρχείου.

Αποθήκευση και Ανάκτηση δεδομένων

- Υπάρχουν ανάλογες συναρτήσεις για αποθήκευση δεδομένων στην R.
  - **write.table**
  - **writeLines**

Ανάκτηση Δεδομένων

- Για πλαίσια δεδομένων χρησιμοποιούμε την εντολή **read.table()**.
  - file: το όνομα του αρχείου, ή η πλήρης διαδρομή αυτού.
  - header: δηλώνει αν στην πρώτη γραμμή του αρχείου δίνονται τα ονόματα ή όχι.
  - sep: δηλώνει πως γίνεται ο διαχωρισμός των δεδομένων.
  - colClasses: ένα διάνυσμα χαρακτήρων, το οποίο δηλώνει τον τύπο των δεδομένων για κάθε στήλη.
  - nrows: ο αριθμός των γραμμών του πλαισίου δεδομένων.
  - comment.char: ένα διάνυσμα χαρακτήρων, το οποίο δηλώνει τον χαρακτήρα, που αποτελεί σχόλιο.
  - skip: ο αριθμός των γραμμών, που θέλουμε να εξαιρεθούν από την αρχή του πλαισίου δεδομένων.
  - stringsAsFactors: θα έπρεπε τα διανύσματα χαρακτήρων να μετατραπούν σε κατηγορικά διανύσματα?

Ανάκτηση Δεδομένων

```
> zz<-read.table("sample.txt", header=T)
> zz
     Gender Smoking Choresterol
1    Male    TRUE         200
2    Male    TRUE         220
3    Male   FALSE         180
4 Female   FALSE          172
```

Ανάκτηση Δεδομένων από Excel

- Μετατροπή του data.xlsx σε data.csv
  - Menu File, Save As… και διαλέξτε .csv as type.
- Τρέξτε την R από τον ίδιο φάκελο που βρίσκετε το αρχείο data.csv.
- Πληκτρολογήστε στην R

```
> data<-read.table(file="data.csv",header=TRUE,sep=";",
    na.strings=c(":"))
```

- Εναλλακτικά:

```
> install.packages("openxlsx")
> library(openxlsx)
> data <- read.xlsx("data.xlsx", sheet = 1, na.strings=":")
```

Header=TRUE είναι η προκαθορισμένη τιμή

Εισαγωγή Δεδομένων από άλλα Στατιστικά Πακέτα

- Με την βοήθεια του πακέτου **foreign** μπορούμε να διαβάσουμε δεδομένα από άλλα στατιστικά πακέτα.

```
> install.packages("foreign")
> library("foreign")
> library(help=foreign)
```

Εισαγωγή Δεδομένων από άλλα Στατιστικά Πακέτα

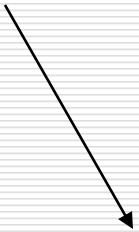
Πακέτο	**Εντολή**
SPSS	read.spss()
S	data.restore() ή read.S()
STATA	read.dta()
SAS	read.xport()
Epi Info	read.epiinfo()
Minitab	read.mtb()
Octave	read.octave()

Συναρτήσεις στην R

```
> x
 [1]  46 104  94 114  35  70 120  29  19 135
   200 222  89 100  55 214  15  81 118 193
> range<-function(x){
   y<-max(x)-min(x)
   return(y)
   }
> range(x)
[1] 207
```

Συναρτήσεις στην R

```
> calc<-function(a,b=2){
   y<-a^b
   return(y)
  }
> calc(4)
[1] 16
> calc(4,3)
[1] 64
```



b=2 προκαθορισμένη τιμή, εκτός αν δοθεί αλλιώς.

a=4, b=3

Συναρτήσεις στην R

Εντολή	Ερμηνεία
if (A) B	Ελέγχει αν ισχύει το A. Αν ναι εκτελεί το B
if (A) B1 else B2	Ελέγχει αν ισχύει το A. Αν ναι εκτελεί το B1 αλλιώς το B2
ifelse(A, B1, B2)	Ίδιο με πριν
break	Τερματίζει τρέχοντα βρόχο
next	Τερματίζει τρέχων βρόχο και αρχίσει επόμενη επανάληψη
return(A)	Τερματίζει τρέχουσα συνάρτηση και επιστρέφει A
while(A) B	Ελέγχει κατά επανάληψη αν ισχύει το A. Αν ναι επιστρέφει B
repeat A	Όπως το while
for(index in A) B	Βρόχος. Εκτελεί το B όσο το index ανήκει στο A

Η εντολή if else

```
if(A)
 {
  A1
 } else
 {
  A2
 }
```

$$h(x) = \begin{cases} x^2 & , x \le 0.05 \\ 0.25 & , x > 0.05 \end{cases}$$

```
> x<-0.10
> if(x<=0.05)
   {
    h<-x^2
   } else
   {
    h<-0.25
   }
```

Η εντολή if else

```
if(A)
 {
  B1
 } else if(C)
 {
  B2
 } else
 {
  B3
 }
```

$$h(x) = \begin{cases} x^2, & x \le 0.05 \\ 0.25, & 0.05 < x \le 1 \\ 1, & x > 1 \end{cases}$$

```
> x<-0.10
> if(x<=0.05)
   {
    h<-x^2
   } else if(x>0.05 & x<=1)
   {
    h<-0.25
   } else
   {
    h<-1
   }
```

Ο Βρόχος for

for(index in A) B

```
> x<-c(3,6,2,7)
> n<-length(x)
> proda<-1
> summ<-0
> for(i in 1:n)
  {
   summ<-summ+x[i]
   proda<-proda*x[i]
  }
```

Ο Βρόχος for

```
> A<-
    matrix(1:1000^2,ncol=1000,
    nrow=1000)
> summ<-0
> for(i in 1:1000)
 {
  for(j in 1:1000)
   {
    summ<-summ+A[i,j]
   }
 }
```

```
> system.time({summ<-0; for(i in
    1:1000){for(j in 1:1000){summ<-
    summ+A[i,j]}}})
   user  system elapsed
   1.94    0.00    1.96
>system.time({sum(as.numeric(apply(A,
    1,sum)))})
   user  system elapsed
   0.05    0.00    0.36
```

Χρόνος, που δαπανάται από τη CPU

Χρόνος, που το σύστημα ξόδεψε για άλλες εργασίες

Χρόνος εκτέλεσης του προγράμματος

Οι Βρόχοι while & repeat

while(A) B | **repeat(B; if(A) break)**

Ας υποθέσουμε ότι θέλουμε να εφαρμόσουμε την επαναληπτική μέθοδο Newton-Raphson για την εύρεση ρίζας της εξίσωσης

$$f(x) = x^3 + 2x^2 - 7 = 0$$

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

$$f'(x) = 3x^2 + 4x$$

Οι Βρόχοι while & repeat

```
> x<-1
> tolerance<-0.000001
> f<-x^3+2*x^2-7
> f.prime<-3*x^2+4*x
> while(abs(f)>tolerance)
 {
  x<-x-f/f.prime
  f<-x^3+2*x^2-7
  f.prime<-3*x^2+4*x
 }
```

```
> x<-1
> tolerance<-0.000001
> f<-x^3+2*x^2-7
> f.prime<-3*x^2+4*x
> repeat
   {
    x<-x-f/f.prime
    f<-x^3+2*x^2-7
    f.prime<-3*x^2+4*x
    if(abs(f) <= tolerance) break
   }
```

Συναρτήσεις στην R

- Παράδειγμα 1: Έστω ότι θέλουμε να κατασκευάσουμε μια συνάρτηση που να υπολογίζει το x! (x παραγοντικό, όπου x φυσικός αριθμός)

```
fact1<-function(x){
y<-floor(x)
if (y!=x | x<0)
print("Your number is not natural")
else
{
f<-1
if (x<2) return(f)
for (i in 2:x) {
f<-f*i
}
return(f)
}
}
```

```
> fact1(3)
[1] 6
> fact1(1)
[1] 1
> fact1(0)
[1] 1
> fact1(4)
[1] 24
> fact1(2.3)
[1] "Your number is not natural"
```

Συναρτήσεις στην R

```
fact2<-function(x){
    y<-floor(x)
    if (y!=x | x<0)
    print("Your number is not natural")
    else
    {
    f<-1
    t<-x
    while(t>1){
    f<-f*t
    t<-t-1
    }
    return(f)
    }
    }
```

```
> fact2(3)
[1] 6
> fact2(1)
[1] 1
> fact2(0)
[1] 1
> fact2(4)
[1] 24
> fact2(2.3)
[1] "Your number is not natural"
```

Συναρτήσεις στην R

```
fact3<-function(x){
      y<-floor(x)
      if (y!=x | x<0)
      print("Your number is not natural")
     else
      {
     f<-1
     t<-x
     repeat{
     if (t<2) break
     f<-f*t
     t<-t-1
      }
     return(f)
     }
     }
```

```
> fact3(3)
[1] 6
> fact3(1)
[1] 1
> fact3(0)
[1] 1
> fact3(4)
[1] 24
> fact3(2.3)
[1] "Your number is not natural"
```

Συναρτήσεις στην R

- Οι **βρόχοι** στην R μπορεί να καθυστερήσουν αρκετά μια συνάρτηση και για αυτό τον λόγο καλό είναι να αποφεύγονται αν είναι δυνατόν. Οι βρόχοι μπορούν να αποφευχθούν κάποιες φορές με χρήση λογικών συναρτήσεων για διανύσματα. Π.χ. ο βρόχος

```
for(i in 1:length(y)) {if(y[i]<0} y[i]<-0}
```

μπορεί να αντικατασταθεί με την πολύ πιο γρήγορη εντολή

```
y[y<0]<-0
```

Συναρτήσεις στην R

- Καλό είναι επίσης να χρησιμοποιούμε τις έτοιμες συναρτήσεις της R όπου είναι εφικτό. Η συνάρτηση π.χ. **cumprod(x)** παίρνει ως όρισμα ένα αριθμητικό διάνυσμα και επιστρέφει το αθροιστικό γινόμενο. Π.χ.

```
> cumprod(c(1,2,4))
[1] 1 2 8
```

- Μπορούμε λοιπόν να χρησιμοποιήσουμε την συγκεκριμένη συνάρτηση για τον υπολογισμό του παραγοντικού.

```
fact4<-function(x){
    y<-floor(x)
    if (y!=x|x<0)
    print("Your number is not natural")
   else
    {
    return(max(cumprod(1:x)))
    }
    }
```

```
> fact4(3)
[1] 6
> fact4(1)
[1] 1
> fact4(0)
[1] 1
> fact4(4)
[1] 24
> fact4(2.3)
[1] "Your number is not natural"
```

Συναρτήσεις στην R

- Τέλος θα μπορούσαμε να είχαμε χρησιμοποιήσει την συνάρτηση Γάμμα, μιας και για φυσικούς αριθμούς x!=Γ(x+1) ή την έτοιμη συνάρτηση factorial.

```
> gamma(4)
[1] 6
> gamma(2)
[1] 1
> gamma(1)
[1] 1
> factorial(3)
[1] 6
> factorial(1)
[1] 1
> factorial(0)
[1] 1
```

- Οι συναρτήσεις gamma και factorial επιστρέφουν τιμές και για μη φυσικούς αριθμούς, χωρίς να ερμηνεύονται τότε ως παραγοντικά.

Συναρτήσεις στην R

- Προβλήματα υπερχείλισης: Η R εκτελεί τις πράξεις με την σειρά που αυτές ορίζονται χωρίς να προβαίνει σε απλοποιήσεις. Έτσι υπάρχει περίπτωση να αντιμετωπίσουμε προβλήματα **υπερχείλισης** (overflow). Π.χ.

```
> factorial(200)/(factorial(100)*factorial(100))
[1] NaN
Warning message:
In factorial(200) : value out of range in 'gammafn'
```

$\binom{200}{100}$

- Για να υπολογίσουμε την άνω ποσότητα λοιπόν είναι καλό να κάνουμε εμείς την απλοποίηση και να ζητήσουμε στην R να κάνει τις πράξεις εν συνεχεία.

Συναρτήσεις στην R

Προφανώς

$$\binom{200}{100} = \frac{101 \cdot \ldots \cdot 200}{1 \cdot \ldots \cdot 100}$$

Άρα στην R γράφουμε

```
> prod(101:200)/prod(1:100)
[1] 9.054851e+58
```

$\longrightarrow \quad 9.05 \cdot 10^{58}$

ή ακόμα καλύτερα

```
> x<-1:100
> y<-101:200
> z<-y/x
> prod(z)
[1] 9.054851e+58
```

Συναρτήσεις στην R

- Ένας ακόμα τρόπος να αποφύγουμε προβλήματα υπερχείλισης είναι με χρήση του λογαρίθμου (φυσικού) log.

$$\binom{200}{100} = \exp\left[\log\left\{\binom{200}{100}\right\}\right] = \exp[\log(200!) - 2\log(100!)]$$

Αλλά

$$\log(n!) = \sum_{i=1}^{n} \log(i)$$

```
> exp(sum(log(1:200))-2*sum(log(1:100)))
[1] 9.054851e+58
```