Σύντομη εισαγωγή στον τρόπο χρήσης του R

01 Εκκίνηση

Στη γραμμή εντολών γράψτε το γράμμα R (κεφαλαίο) και πατήστε ENTER (στα Windows ή το MacOS κάντε κλικ στο κατάλληλο εικονίδιο του προγράμματος). Πρέπει να δείτε κάτι παρόμοιο με το παρακάτω:

```
R version 2.9.2 (2009-08-24)
Copyright (C) 2009 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Το μήνυμα καλωσορίσματος μας δίνει και χρήσιμες οδηγίες όπως τι να κάνουμε για να πάρουμε βοήθεια ή να φύγουμε από το R. Π.χ. για να φύγουμε μας λέει να γράψουμε q( ). Παρατηρούμε τη χρήση των παρενθέσεων που θυμίζει τη σύνταξη της γλώσσας C. Για βοήθεια γράφουμε help( ) κλπ.

Για να δώσουμε εντολές, τις γράφουμε στην προτροπή (σύμβολο >) και πατάμε ENTER.

Για να φύγουμε από το R γράφουμε q( ). Τότε, το πρόγραμμα θα μας ρωτήσει αν θέλουμε να αποθηκεύσουμε τη δουλειά μας για να συνεχίσουμε από εκεί που σταματήσαμε, την επόμενη φορά.

02 Εκτέλεση πράξεων.

Στη γραμμή εντολών γράψτε τα παρακάτω (το > είναι η προτροπή του R και προφανώς δεν το γράφετε):

```
> 2+2

> 2*4

> 2^3

> sqrt(2)

> 1.414214^2

> exp(1)

> pi

> exp(2)

> exp(1.5)

> sin(pi)
```

Τι αποτελέσματα πήρατε; Τι παρατηρείτε στο τελευταίο αποτέλεσμα;

03 Συναρτήσεις με επιπλέον ορίσματα.
Αλλαγή βάσης λογαρίθμου. Γράψτε:

```
> log(10,10)

> log(10, base=10)
```

Τι αποτέλεσμα πήρατε;

04 Ορισμός μεταβλητών.
Γράψτε τα παρακάτω:

```
> x = 2
> x+3
```

Τι αποτέλεσμα πήρατε και πώς το ερμηνεύετε;

Παρόμοια:

```
> e = exp(1)
> e

> e^2
```

Τι κάναμε εδώ;

Το σύμβολο της ισότητας που χρησιμοποιήσαμε για τον ορισμό μιας μεταβλητής, μπορεί να αντικατασταθεί και από το σύμβολο <- που κάνει το ίδιο (αναθέτει την τιμή του δεξιού στο αριστερό μέλος), αλλά και το σύμβολο -> (αναθέτει την τιμή του αριστερού στο δεξί μέλος).

05 Ανάθεση παραστάσεων σε μεταβλητές.
Συνεχίζοντας, γράψτε τα παρακάτω:

```
> x

> y = x
> x = 2*x+1
> x

> y <- 2*x-1
> y

> (y+1)/2 -> y
> y
```

Τι έγινε σε κάθε βήμα;

Τα ονόματα των μεταβλητών μπορούν να περιέχουν γράμματα, αριθμητικά ψηφία, τελεία και χαρακτήρα υπογράμμισης. Μπορούν να αρχίζουν από γράμμα ή τελεία. Συνηθίζονται, χωρίς να είναι υποχρεωτικό, οι εξής συμβάσεις:

- *n για αριθμό δεδομένων (παρατηρήσεων)*
- *x, y για άνυσμα (πίνακα) δεδομένων (βλ. συνέχεια)*
- *i, j για ακέραιους και αριθμοδείκτες*
- *μεταβλητές που αρχίζουν με τελεία χρησιμοποιούνται στον προγραμματισμό με R*

06 Απόδοση πολλαπλών τιμών. Ανύσματα δεδομένων (data vectors).
Με τη συνάρτηση c( ) μπορούμε να αποδώσουμε πολλές τιμές συγχρόνως, κατασκευάζοντας δομές παρόμοιες με τους πίνακες των άλλων γλωσσών προγραμματισμού.
Γράψτε:

```
> mydata = c(74, 122, 235, 111, 292, 111, 211, 133, 156, 79)
> mydata
```

07 Συνδυασμός δυο ανυσμάτων δεδομένων σε ένα.
Γράψτε:

```
> x = c(74, 122, 235, 111, 292)
> y = c(111, 211, 133, 156, 79)
> c(x,y)

> z = c(x,y)
> z
```

Τα δεδομένα πρέπει να είναι του ίδιου τύπου. Αν δεν είναι θα μετατραπούν σε ένα, συνήθως αλφαριθμητικό, εμποδίζοντας περαιτέρω αριθμητικές πράξεις. Αλφαριθμητικά δεδομένα ορίζονται με τη βοήθεια απλών ή διπλών εισαγωγικών.

08 Εισαγωγή αλφαριθμητικών δεδομένων.
Γράψτε:

```
> simpsons = c("Homer", "Marge", "Bart", "Lisa", "Maggie")
> simpsons
```

09 Απόδοση ονομάτων.
Στα στοιχεία ενός ανύσματος, μπορούμε να δώσουμε ονομασίες με τη συνάρτηση names( ), ώστε όταν εμφανίζονται οι τιμές του, να φαίνονται από πάνω τα ονόματα σαν επικεφαλίδες.
Γράψτε:

```
> names(simpsons) = c("dad", "mom", "son", "daughter 1", "daughter 2")
> names(simpsons)

> simpsons
```

10 Επεξεργασία δεδομένων αποθηκευμένων σε ανύσματα.
Γράψτε:

```
> sum(mydata)

> length(mydata)

> sum(mydata)/length(mydata)

> mean(mydata)

> sort(mydata)

> min(mydata)

> max(mydata)

> range(mydata)

> diff(mydata)

> cumsum(mydata)
```

Τι κάνει κάθε συνάρτηση από τις παραπάνω;
Οι πράξεις και συναρτήσεις στο R είναι "vectorized", δηλαδή επιδρούν σε κάθε στοιχείο του ανύσματος χωριστά.

11 Παράδειγμα vectorized πράξεων.
Γράψτε:

```
> mydata.1=c(89, 254, 306, 292, 274, 233, 294, 204, 204, 90)
> mydata + mydata.1

> mydata - mydata.1

> mydata - mean(mydata)
```

Τι έγινε εδώ;
Στην τελευταία πράξη, ένας αριθμός αφαιρείται από ένα άνυσμα. Αυτό λέγεται "ανακύκλωση δεδομένων", δηλαδή τιμές από το ένα άνυσμα επαναλαμβάνονται ώστε το μήκος του να ταιριάξει με το άλλο.

12 Άλλο παράδειγμα vectorized πράξεων.
Η διασπορά των δεδομένων υπολογίζεται απευθείας με τη συνάρτηση var( ). Εδώ θα υπολογιστεί πρώτα υλοποιώντας το γνωστό ορισμό του μέσου τετραγώνου για να δείξουμε τη χρήση του vectorization.
Γράψτε:

```
> x = c(2, 3, 5, 7, 11)
> sum((x-mean(x))^2)/(length(x)-1)
```

Δοκιμάστε το ίδιο πιο αναλυτικά, ως εξής:

```
> x = c(2, 3, 5, 7, 11)
> xmean = mean(x)
> n=length(x)
> sum((x-xmean)^2)/(n-1)
```

Επαληθεύστε με χρήση της συνάρτησης var( ):

```
> var(x)
```

13 Αναζήτηση βοήθειας.
Αναζητείστε βοήθεια για τη συνάρτηση mean, γράφοντας help("mean") ή ?mean ή ?"mean".

14 Βοήθεια μέσω παραδειγμάτων.
Μπορούμε να δούμε παραδείγματα από τη χρήση των συναρτήσεων με τη βοήθεια της συνάρτησης example( ). Ακόμη, αν γράψουμε help.search("mean") θα βρούμε όλες τις σελίδες της βοήθειας που αναφέρουν κάπου τη συνάρτηση mean, ενώ με apropos("mean") θα δούμε όλα τα ονόματα συναρτήσεων που περιέχουν τη λέξη mean.
Αλλά **προσοχή!** Το παράδειγμα εκτελεί τις πράξεις που περιέχει και αν οι μεταβλητές του έχουν ίδιο όνομα με αυτές που χρησιμοποιείτε, μπορεί να αλλοιώσει τα δεδομένα σας!
Γράψτε:

```
> x = c(2, 3, 5, 7, 11)
> x

> example("mean")
(εδώ θα εμφανιστεί το παράδειγμα)
> x
```

Τι παρατηρείτε ότι συνέβη στο άνυσμα x; Ποιες οι τιμές πριν και μετά την κλήση του παραδείγματος;
Μερικές ευκολίες...

Πάνω βελάκι	*προηγούμενη εντολή (πατώντας πολλές φορές, μας γυρίζει ανάλογα πίσω)*
Κάτω βελάκι μ	*ας γυρίζει μπροστά μέχρι την πιο πρόσφατη εντολή*
Αριστερό/δεξί βελάκι	*πηγαινοφέρνει τον κέρσορα αναλόγως*
HOME ή Ctrl-a	*στην αρχή της τρέχουσας γραμμής*
END ή Ctrl-e	*στο τέλος της τρέχουσας γραμμής*

15 Διόρθωση και προσθήκη δεδομένων.
Για να διορθώσετε τα δεδομένα στο x (ή όποιο άλλο αντικείμενο θέλετε), γράψτε:
> data.entry(x)

Η data.entry χρησιμεύει και στην εισαγωγή δεδομένων: ορίζουμε μια απλή μεταβλητή και με τη data.entry συνεχίζουμε την εισαγωγή δεδομένων.

16 Ορισμός συνεχόμενων δεδομένων.
Με τη διπλή τελεία (:) μπορούμε να δίνουμε συνεχόμενους αριθμούς.
Γράψτε:
```
> 1:10

> rev(1:10)

> 10:1
```

Τι πήρατε;

17 Ορισμός δεδομένων με μορφή αριθμητικής προόδου.
Γράψτε:
```
> a = 1; h = 4;   n = 5   (το ; επιτρέπει να γράφουμε πολλές εντολές στην ίδια σειρά)
> a + h*(0:n)

> a + h*(0:(n-1))

> a + h*(0:n-1)

> b  = a + h*(0:n)
> b

> b  = c(a + h*(0:n))
> b
```

Τι αποτέλεσμα πήρατε; Πώς το ερμηνεύετε;
Το παραπάνω παράδειγμα δείχνει ότι χρειάζεται προσοχή στη χρήση παρενθέσεων για να μην πάρουμε λάθος αποτελέσματα. Όπως και με άλλες γλώσσες προγραμματισμού, είναι καλό να χρησιμοποιούμε "καταχρηστικά" τις παρενθέσεις για να είμαστε βέβαιοι για τη σειρά των πράξεων.

18 Επαναλαμβανόμενοι αριθμοί.
Γράψτε:
```
> rep(1, 10)

> rep(1:3, 3)
```

Συνδυάζοντας τα παραπάνω, μπορούμε να χειριστούμε πολύπλοκες ομάδες δεδομένων.

19 Προσπέλαση δεδομένων με δείκτες.
Μπορούμε να προσπελάσουμε μια μεμονωμένη τιμή από ένα άνυσμα δεδομένων χρησιμοποιώντας κατάλληλο αριθμοδείκτη. Ο συμβολισμός είναι ίδιος με τη γλώσσα C, αλλά η μέτρηση αρχίζει από το 1 και όχι από το 0: x[1], x[2], ...
Γράψτε:
```
> mydata

> n=length(mydata)
> n

> mydata[1]
```

```
> mydata[n]

> mydata[length(mydata)]
```

20 Προσπέλαση τομέων και υποσυνόλων δεδομένων.
Γράψτε:
```
> mydata[1:4]

> mydata[c(1,3,6)]
```

21 Αρνητικοί δείκτες για να αποκλείσουμε κάποιες τιμές.
Χρησιμοποιούμε αρνητικούς δείκτες, αν η απόλυτη τιμή τους είναι από 1 μέχρι το μήκος της μεταβλητής, για να εμφανίσουμε όλα τα άλλα στοιχεία εκτός από αυτά στα οποία αναφερόμαστε με την απόλυτη τιμή των δεικτών
Γράψτε:

```
> mydata[-1]

> mydata[-(1:4)]

> mydata[-(c(1,3,6))]
```

22 Προσπέλαση δεδομένων μέσω των ονομάτων τους.
Γράψτε:

```
> x=1:3
> names(x)=c("one", "two", "three")
> x["two"]
```

23 Απόδοση τιμών σε συγκεκριμένα στοιχεία με τη βοήθεια δεικτών.
Γράψτε:

```
> x[1] = 3
> x

> x[4]=4
> x

> x[6]=5
> x

> x[5:6]=5
> x

> x[-2] = 0
> x
```

Τι παίρνετε σε κάθε βήμα; Τι παρατηρείτε σχετικά με το μέγεθος του συνόλου δεδομένων που παριστάνει το x. Τι μπορεί να σημαίνει το NA;

24 Επέκταση ενός συνόλου δεδομένων, με απόδοση τιμών σε ένα κατάλληλο άνυσμα δεικτών.
Γράψτε:
```
> x = 1:4
> x

> x[5:8] = 5
> x
```

```
> y = c(3, 5, 8)
> y

> x[9:11]  = y
> x
```

25 Λογικές τιμές και vectorized χρήση τους.
Γράψτε:
```
> mydata

> mydata > 100

> mydata < 100
```

Τι παίρνετε σε κάθε περίπτωση;

26 Εύρεση δεδομένων που ικανοποιούν συγκεκριμένη συνθήκη.
Γράψτε:
```
> mydata < 100

> mydata[mydata < 100]

> which(mydata < 100)

> mydata[which(mydata < 100)]

> mydata[c(1,10)]
```

Τι πήρατε σε κάθε βήμα; Ποια η χρησιμότητα της συνάρτησης which;

Για να διατυπώνουμε λογικές συνθήκες έχουμε στη διάθεσή μας τα σύμβολα >, >=, <, <=, ==, !=, &, &&, | και ||. Τα & και | είναι vectorized, δηλαδή δίνουν αποτέλεσμα σύγκρισης αντίστοιχων τιμών από δύο μεταβλητές, ενώ τα && και || εκτιμώνται από αριστερά προς τα δεξιά μέχρι να προσδιοριστεί ένα TRUE ή FALSE, δηλαδή επιστρέφουν μόνο μια λογική τιμή.

27 Παράδειγμα χρήσης λογικών τελεστών.
Γράψτε:
```
> x=1:5
> x<5

> x>1

> x>1 & x<5

> x>1 && x<5

> x>1 | x<5

> x>1 || x<5

> x==3

> x!=3

> ! x==3

> x == c(2,4)
```
(συνέχεια στην άλλη σελίδα)

```
> x %in% c(2,4)
```

Για σύγκριση ανύσματος με εύρος τιμών είναι προτιμώτερος ο τελεστής %in%.

<u>Σημαντική σημείωση</u>: για αριθμητικές συγκρίσεις, οι τελεστές ==, != κλπ δεν είναι κατάλληλοι και αντί γι' αυτούς χρησιμοποιούμε τις συναρτήσεις identical και all.equal.
Όταν λείπουν τιμές (NA) από τα δεδομένα μας χρησιμοποιούμε τη συνάρτηση is.na για να τις εντοπίζουμε, π.χ. x[!is.na(x)]

28 Περιβάλλον εργασίας.
Αν έχουμε ορίσει πολλές μεταβλητές, μπορούμε να τις θυμηθούμε με τη συνάρτηση ls( ) ή την objects( )
Γράψτε:
```
> ls()

> objects()
```

Τι μπορεί να σημαίνουν τελικά οι αριθμοί σε αγκύλες [ ] που εμφανίζονται στην αρχή κάθε γραμμής;

Γράψτε:
> browseEnv( )

```
> ls(pattern="data")
```

Τι πήρατε με την τελευταία εντολή;

29 Επέμβαση στο περιβάλλον εργασίας.
Για να καταργήσουμε αντικείμενα που δε μας ενδιαφέρουν πια, χρησιμοποιούμε τη συνάρτηση rm( ) ή remove( ).
Γράψτε:
```
> ls()

> rm(x)
ls()

> rm (xm, xmean, a,b,y,w)
> ls()
```

30) Εξωτερικές πηγές δεδομένων.
Δημιουργείστε στον κατάλογο εργασίας, το αρχείο data1.txt με τα εξής δεδομένα:

```
74 122 235 111 292 111 211 133 156  79
```

Μετά, επιστρέψτε στο R και γράψτε:

```
> data1 = scan(file="data1.txt")
Read 10 items
> data1
```

Δημιουργείστε στον κατάλογο εργασίας, το αρχείο data2.txt με τα εξής δεδομένα:
```
x     y
1     2.05
2     3.95
3     5.89
4     7.99
5    10.01
6    12.11
```

```
7    14.07
8    15.94
9    18.04
10    20.06
```

Μετά, επιστρέψτε στο R και γράψτε:

> read.table("data2.txt",header=TRUE)

```
> data2=read.table("data2.txt",header=TRUE)
> data2
```

Αν το αρχείο είναι τύπου CSV (comma separated values) θα χρησιμοποιήσουμε τη συνάρτηση read.csv( )
Το όνομα του αρχείου μπορούμε να το δώσουμε και διαδραστικά με τη συνάρτηση file.choose( ):
```
> read.table(file=file.choose( ))
```

31 Αποθήκευση εντολών του R για μελλοντική χρήση.
Μπορούμε να συγκεντρώσουμε εντολές του R σε εξωτερικά αρχεία και να τις διαβάσουμε και εκτελέσουμε με τη συνάρτηση source( ).
Γράψτε ένα σε ένα αρχείο με το όνομα commands τα παρακάτω:

```
x = 1:10
y = x^2
z = log(x, 10)
```

Μετά, επιστρέψτε στο R και γράψτε:

```
> source("commands")
> x; y; z
```

Για να εντοπίσουμε ένα αρχείο, μπορούμε να αναφερόμαστε και στη διαδρομή του, με τους κανόνες του λειτουργικού συστήματος. Ο κατάλογος εργασίας βρίσκεται με τη συνάρτηση getwd( ) και επανακαθορίζεται με τη setwd( ).
Αρχεία μπορούν να βρεθούν και να διαβαστούν ακόμη και στο διαδίκτυο αν δοθεί η διεύθυνσή τους.