# Κεφάλαιο 5

# Απλός Προγραμματισμός στην R

Η έννοια του προγραμματισμού στην R βασίζεται στη δημιουργία καινούργιων *συναρτήσεων* οι οποίες θα χρησιμοποιηθούν για περαιτέρω ανάπτυξη της γλώσσας. Το κύριο δομικό υλικό είναι οι υπάρχουσες συναρτήσεις (functions) της R, μερικές από τις οποίες ήδη έχουμε εξετάσει σε προηγούμενα κεφάλαια.

## 5.1 Λογικοί Τελεστές και Τελεστές Σύγκρισης

Οι κύριοι λογικοί τελεστές και τελεστές σύγκρισης αναφέρονται στον πίνακα που ακολουθεί.

Οι τελεστές `&` και `|` αξιολογούν τις ανάλογες εκφράσεις στοιχείο με στοιχείο και επιστρέφουν ένα διάνυσμα με τις λογικές τιμές `TRUE` και `FALSE`.

```
> x <- seq(-1,1,length=12)
> x
 [1] -1.00000000 -0.81818182 -0.63636364 -0.45454545 -0.27272727 -0.09090909
 [7]  0.09090909  0.27272727  0.45454545  0.63636364  0.81818182  1.00000000
> x < 0 | x > 0.8
 [1] TRUE TRUE TRUE TRUE TRUE TRUE FALSE FALSE FALSE FALSE TRUE TRUE
> x < 0 & x > 0.8
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

| Τελεστής | Ερμηνεία |
|---|---|
| == | ίσο με |
| > | μεγαλύτερο από |
| != | άνισο από |
| < | μικρότερο από |
| >= | μεγαλύτερο ή ίσο από |
| <= | μικρότερο ή ίσο από |
| & | και |
| && | *και* ελέγχου |
| \| | ή |
| \|\| | *ή* ελέγχου |
| ! | όχι |

Πίνακας 5.1: Λογικοί τελεστές και τελεστές σύγκρισης.

Οι τελεστές ελέγχου χρησιμοποιούνται για να κατασκευάζονται υποθετικές προτάσεις.

## 5.2 Χρησιμοποιώντας Υποσύνολα των Δεδομένων

Αρκετές φορές υπάρχει η ανάγκη να γίνουν διάφοροι υπολογισμοί χρησιμοποιώντας ένα συγκεκριμένο κομμάτι των δεδομένων. Η μέθοδος αυτή ονομάζεται *υποστίξη*. Η R έχει πολύ καλές και εύκολες δυνατότητες στο να πετυχαίνει την υποστίξη. Στο επόμενο παράδειγμα αυτή εφαρμόζεται αρχικά σε διανύσματα.

```
> x
 [1] -1.00000000 -0.81818182 -0.63636364 -0.45454545 -0.27272727 -0.09090909
 [7]  0.09090909  0.27272727  0.45454545  0.63636364  0.81818182  1.00000000
> x[3]    # extract  the third element
[1] -0.6363636
> x[c(1,2,5)]   # extract the first, second and fifth elements.
[1] -1.0000000 -0.8181818 -0.2727273
> x[-(3:10)]    # extract all the elements except those in positions 3 to 10.
[1] -1.0000000 -0.8181818  0.8181818  1.0000000
> x[ x > 0]     # extract the elements that satisfy the condition.
 [1] 0.09090909 0.27272727 0.45454545 0.63636364 0.81818182 1.00000000
> x[ x > 0 & x < 0.5]
```

```
[1] 0.09090909 0.27272727 0.45454545
```

Δηλαδή είναι εφικτό να πάρουμε υπόσυνολο δεδομένων είτε βάση της θέσης των στοιχείων του είτε βάση μιας συνθήκης. Η υπόστιξη μπορεί να γενικευθεί και στην περίπτωση των πινάκων.

```
>A <- cbind(c(1,2,-1), c(12,15,18), c(-1,-4,-9))
> A
     [,1] [,2] [,3]
[1,]    1   12   -1
[2,]    2   15   -4
[3,]   -1   18   -9
> A[1,1]  #extracts the (1,1) element
[1] 1
> A[1,3]  # extracts the (1,3) element
[1] -1
> A[1:2,3]  #extracts the elements (1,3), (2,3)
[1] -1 -4
> A[1:2,2:3]   #extracts a two by two matrix
     [,1] [,2]
[1,]   12   -1
[2,]   15   -4
> A[,2:3]      # omission of a dimension gives the corresponding columns
     [,1] [,2]
[1,]   12   -1
[2,]   15   -4
[3,]   18   -9
> A[-1,2:3]   # use of negative indices
     [,1] [,2]
[1,]   15   -4
[2,]   18   -9
```

Γενικεύεται επίσης και στα αντικείμενα λίστας,

```
> mylist <- list(x,A)
> mylist
[[1]]:
 [1] -1.00000000 -0.81818182 -0.63636364 -0.45454545 -0.27272727 -0.09090909
```

```
 [7]  0.09090909  0.27272727  0.45454545  0.63636364  0.81818182  1.00000000
[[2]]:
     [,1] [,2] [,3]
[1,]    1   12   -1
[2,]    2   15   -4
[3,]   -1   18   -9
> mylist[[1]]
 [1] -1.00000000 -0.81818182 -0.63636364 -0.45454545 -0.27272727 -0.09090909
 [7]  0.09090909  0.27272727  0.45454545  0.63636364  0.81818182  1.00000000
> mylist[[2]]
     [,1] [,2] [,3]
[1,]    1   12   -1
[2,]    2   15   -4
[3,]   -1   18   -9
```

καθώς και σε πλαίσια δεδομένων με τη χρήση των συμβόλων `[[]]` και `$`, αντίστιχα.

```
> library(MASS)
> is.data.frame(survey)
[1] TRUE
> names(survey)
 [1] "Sex"    "Wr.Hnd" "NW.Hnd" "W.Hnd"  "Fold"   "Pulse"  "Clap"   "Exer"
 [9] "Smoke"  "Height" "M.I"    "Age"
> survey$Age[1:100]
  [1] 18.250 17.583 16.917 20.333 23.667 21.000 18.833 35.833 19.000 22.333
 [11] 28.500 18.250 18.750 17.500 17.167 17.167 19.333 18.333 19.750 17.917
 [21] 17.917 18.167 17.833 18.250 19.167 17.583 17.500 18.083 21.917 19.250
 [31] 41.583 17.500 39.750 17.167 17.750 18.000 19.000 17.917 35.500 19.917
 [41] 17.500 17.083 28.583 17.500 17.417 18.500 18.917 19.417 18.417 30.750
 [51] 18.500 17.500 18.333 17.417 20.000 18.333 17.167 17.417 17.667 18.417
 [61] 20.333 17.333 17.500 19.833 18.583 18.000 30.667 16.917 19.917 18.333
 [71] 17.583 17.833 17.667 17.417 17.750 20.667 23.583 17.167 17.083 18.750
 [81] 16.750 20.167 17.667 17.167 17.167 17.250 18.000 18.750 21.583 17.583
 [91] 19.667 18.000 19.667 17.083 22.833 17.083 19.417 23.250 18.083 19.083
```

## 5.3 Κατασκευή Συναρτήσεων

Για να γίνει κατανοητή η έννοια της κατασκευής νέων συναρτήσεων στην R, θα εξεταστεί το ακόλουθο παράδειγμα το οποίο δίνει σαν αποτέλεσμα την τυπική απόκλιση ενός διανύσματος $x$:

```
>standard.deviation <- function(x)
{
sqrt(var(x))
}
> x <- rnorm(100, mean=0, sd=2) #100 observations from normal
                                #with mean 0 and variance 4
> var(x)
[1] 3.879332
> standard.deviation(x)
[1] 1.969602
```

Συνεπώς, για να υπολογιστεί η τυπική απόκλιση (`standard.deviation`) αξίζει να σημειωθεί ότι χρησιμοποιήθηκαν δύο από τις προυπάρχουσες συναρτήσεις, η `sqrt()` και η `var()`. Αυτή είναι η θεμελιώδης ιδέα όταν υπάρχει η ανάγκη ορισμού νέας συνάρτησης στην R. Φυσικά υπάρχει συγκεκριμένη συνάρτηση στην R για υπολογισμό της τυπικής απόκλισης και αυτή είναι η `sd`. Παρατίθενται μερικές βασικές δηλώσεις που είναι χρήσιμες στον ορισμό καινούργιων συναρτήσεων:
Μερικές επιπρόσθετες δηλώσεις είναι οι η `switch()` και η `stop()`. Τα επόμενα παραδείγματα αναλύουν τα πιο πάνω. Το πρώτο παράδειγμα επεξηγεί το πώς χρησιμοποιείται η εντολή `if` για να γεννηθούν δείγματα από διάφορες κατανομές.

```
random.gener <- function(n, distribution, shape)
{
# a function to generate n random numbers
if(distribution=="gamma") rgamma(n, shape) else
if(distribution=="exp")   rexp(n) else
if(distribution=="norm") rnorm(n) else
stop("Invalid Distribution")
}
> random.gener(10, "gamma", 2)
 [1] 0.3461286 2.0791867 3.2288429 4.3973702 1.7676279 2.7317868
 [7] 0.4084932 2.4203665 0.7430161 5.1688287
```

| **Εντολή** | **Ερμηνεία** |
|---|---|
| `if` (A) B | ελέγχει αν ισχύει το A, αν ναι τότε εκτελεί το B |
| `if` (A) B1 `else` B2 | ελέγχει αν ισχύει το A, αν ναι τότε εκτελεί το B1, διαφορετικά εκτελεί το B2 |
| `ifelse`(A,B1,B2) | πιο απλός τρόπος γραφής του προηγούμενου |
| `break` | τερματίζει τον τρέχων βρόγχο |
| `next` | τερματίζει τον τρέχων βρόγχο και αρχίζει την επόμενη επανάληψη |
| `return`(A) | τερματίζει την τρέχων συνάρτηση και επιστρέφει το A |
| `while` (A) B | ελέγχει κατ΄ επανάληψη αν ισχύει το A, αν ναι τότε εκτελεί το B |
| `repeat` A | απλούστερη συνάρτηση για το `while` |
| `for` (`index in` A) B | βρόγχος, αλλά απαιτεί αρκετόν υπολογιστικό χρόνο |

Πίνακας 5.2: Βασικές δηλώσεις.

```
> random.gener(10, "unif", 2)
Error in random.gener(10, "unif", 2): Invalid Distribution
```

Η παραπάνω συνάρτηση δημιουργεί δείγμα μεγέθους $n$ από τις κατανομές Γάμμα, Εκθετική και Τυπική Κανονική. Διαφορετικά, αν του δώσουμε μια οποιαδήποτε άλλη κατανομή θα επιστρέψει ότι έγινε σφάλμα.

Το επόμενο παράδειγμα παρουσιάζει τον τρόπο που μπορούν να χρησιμοποιηθούν οι εντολές `for` και `if` για να βρεθεί το πρόσημο ενός πραγματικού αριθμού.

```
new.sign <-  function(x)
{
   for (i in 1:length(x))
   {
         if(x[i] > 0)
            x[i] <- 1
    else if(x[i] < 0)
            x[i] <- -1
   }
   x
}
> new.sign(-10:5)
 [1] -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0  1  1  1  1  1
```

Ωστόσο, υπάρχει καλύτερος τρόπος για να επιτευχθεί αυτό, αποφεύγοντας τις επαναλήψεις, οι οποίες απαιτούν περισσότερο υπολογιστικό χρόνο. Εδώ, φανερώνεται ακόμη μία φορά η χρησιμότητα της υπόστιξης, η οποία βοηθάει στο να κερδίζεται πολύτιμος υπολογιστικός χρόνος.

```
sgnfunction <- function(x)
{
ifelse(x > 0, 1, ifelse(x<0, -1, 0))
}
> sgnfunction(-10:10)
[1] -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0  1  1  1  1  1  1  1  1  1  1
```

Παράδειγματα συναρτήσεων καθώς και περαιτέρω εφαρμογές τους θα δούμε στα κεφάλαια που ακολουθούν.