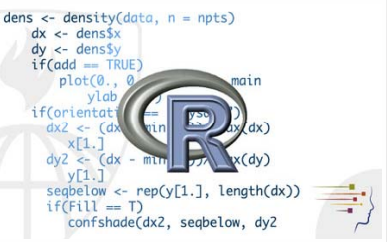
1/11/2022

Αναλυτικές Μέθοδοι στη Γεωπληροφορική

Χειρισμός δεδομένων στο λογισμικό R

Σημερινές ενότητες μαθήματος ...

- Στα ενδότερα της λειτουργίας και της χρήσης του R
  - Σύνταξη εντολών
  - Μεταβλητές και συναρτήσεις
  - Εισαγωγή δεδομένων
  - Αποθήκευση αντικειμένων δεδομένων
  - Χώρος εργασίας στο R
  - Διαχείριση συνεδριών του R
  - ...



Χαρακτηριστικά του R

- **Φορητότητα**
- **Ανεξαρτησία πλατφόρμας**
- **Έγκυρες, συχνές ανανεώσεις**
- **Γραφικά υψηλής αισθητικής**
- **Ευελιξία ευρείας επιλογής πακέτων/ Ενεργή κοινότητα**

Λειτουργικό στυλ του R

- Το R βασίζεται σε μια πολύ ιδιαίτερη γλώσσα προγραμματισμού, την ***r***, μια πλήρη (αν και συγκεκριμένη για τον τομέα της στατιστικής και των γραφικών απεικονίσεων) **λειτουργική\συναρτησιακή γλώσσα** *(functional language)* επηρεασμένη από πλούσια μαθηματική θεωρία.
  - Δίνει έμφαση στην εφαρμογή συναρτήσεων για την *οργάνωση δεδομένων* και την *εκτέλεση κώδικα* για την *ανάλυση τους*

- Ο τρόπος οργάνωσης και εκτέλεσης *κώδικα **r***, επιτρέπει στο R ...
  - Να έχει ορισμένες **ελκυστικές τεχνικές ιδιότητες** (π.χ., αρθρωτός σχεδιασμός, σύνθεση συναρτήσεων σε νέες, κ.λπ.) και,
  - να προσφέρεται για ένα στυλ επίλυσης προβλημάτων με επίκεντρο συναρτήσεις.
    - Αυτές μπορούν εύκολα να εκτελεστούν χρησιμοποιώντας μόνο τοπικές πληροφορίες (μεμονωμένα) ή/και να βελτιστοποιηθούν και να επεκταθούν πολύ πιο εύκολα ή να εκτελεστούν γρήγορα με παράλληλη επεξεργασία και συγχρονισμό

- Για την αποτελεσματική διαχείριση της πολυπλοκότητας για την οργάνωση και εκτέλεση κώδικα και ανάλυσης δεδομένων, το R χρησιμοποιεί έννοιες **αντικειμενοστρεφούς προγραμματισμού** *(object-programming)*
- Στην πραγματικότητα, οτιδήποτε χρησιμοποιείται στο R είναι ένα **αντικείμενο** *(object)*.
  - Αντικείμενα είναι δομές δεδομένων που έχουν **ορισμένα χαρακτηριστικά και ιδιότητες** *(attributes)* και μπορούν να υπόκειται σε **συγκεκριμένες διαδικασίες** *(methods) που δρουν στα χαρακτηριστικά τους*

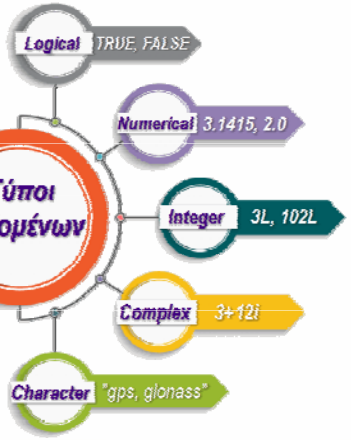
- Χρησιμοποιώντας **'αντικείμενα'**, μπορούμε να κατασκευάσουμε αρθρωτά κομμάτια κώδικα ***r***
- Με τον τρόπο αυτό το R οργανώνει και μοντελοποιεί εφαρμογές του λογισμικού ως **συλλογές αντικειμένων** και **δομές δεδομένων** που επικοινωνούν μεταξύ τους, *παρά συναρτήσεις και λογική που απαιτείται για τον χειρισμό τους*
  - Αυτή η προσέγγιση είναι κατάλληλη για προγράμματα που είναι μεγάλα, περίπλοκα και πρέπει να ενημερώνονται ή να συντηρούνται ενεργά → ευεργετική για τη συνεργατική ανάπτυξη, δυνατότητα επαναχρησιμοποίησης κώδικα, επεκτασιμότητα και αποτελεσματικότητα.

- Η **τάξη** *(class)* ενός αντικειμένου είναι απλά ένα σχέδιο ή ένα σκίτσο του αντικειμένου *(blueprint)* που αντιπροσωπεύει το σύνολο των ιδιοτήτων ή μεθόδων που είναι κοινές για όλα τα **αντικείμενα ίδιου τύπου**.
- Όπως κάθε γλώσσα προγραμματισμού, η ***r*** έχει τους δικούς της ***τύπους δεδομένων*** *(data types)* για την αποθήκευση τιμών ή οποιασδήποτε πληροφορίας
  - Ο χρήστης μπορεί να εκχωρήσει τους διάφορους τύπους δεδομένων σε **μεταβλητές** *(variables)* και να εκτελεί συγκεκριμένες συναρτήσεις/ενέργειες αντίστοιχα → ανάλογα με τους τύπους των δεδομένων.


```
# create a list with required components
student1 <- list(name = "Elena",
  age = 21, GPA = 3.5)

# name the class appropriately
class(student1) <- "Student_Info"

# create and call an object
student1
$name
[1] "Elena"
$age
[1] 21
$GPA
[1] 3.5
attr(,"class")
[1] "student_Info"
```

```
# create a class "Student_Info" with three member variables
setClass("Student_Info", slots=list(name="character",
  age="numeric", GPA="numeric"))

# create an object of class
student1 <- new("Student_Info", name = "Elena",
  age = 21, GPA = 3.5)

# call student1 object
student1

An object of class "Student_Info"
Slot "name":
[1] "Elena"

Slot "age":
[1] 21

Slot "GPA":
[1] 3.5
```

- Τυπικά είναι *οι ταξινομήσεις που δίνουμε σε διαφορετικά είδη πληροφοριών.*
  - **Χαρακτήρες**
  - **Αριθμοί**:
    Δεκαδικοί / Ακέραιοι / Μιγαδικοί
  - **Λογικές εκφράσεις**
  - **Ακατέργαστα δεδομένα**



- Διαφορετικοί τύποι δεδομένων μπορούν να συνδυαστούν για να σχηματίσουν διαφορετικές **δομές δεδομένων** *(data modes ή data structures)* που παρέχουν αποδοτικές μορφές διαχείρισης τους

Βασικές δομές δεδομένων

- **Διάνυσμα** *(Vector)*
  - ***Ατομικό διάνυσμα*** *(Atomic Vector)* περιέχει δεδομένα μόνο ενός τύπου
  - ***Λίστα*** *(List)* - Διάνυσμα που μπορεί να περιέχει πολλούς διαφορετικούς τύπους στοιχείων
- **Πίνακας** ή **μήτρα** *(Matrix)* - δισδιάστατο ορθογώνιο σύνολο δεδομένων (γραμμές & στήλες περιέχουν στοιχεία ίδιου τύπου)
- **Συστοιχία** *(Array)* - πολυδιάστατο σύνολο δεδομένων
- **Παράγοντας** *(Factor)* - διάνυσμα με διακριτές ετικέτες για τις τιμές των στοιχείων του
- **Πλαίσιο δεδομένων** *(Data frame)* – πίνακας όπου κάθε στήλη μπορεί να περιέχει διαφορετικούς τύπους δεδομένων

Μεταβλητές / variables στο R

- Μπορεί να θεωρηθούν ως ένα κοντέινερ/συσκευασία για τα δεδομένα και άλλες πληροφορίες → *Δεσμευμένη θέση μνήμης* για την αποθήκευση δεδομένων
- Με βάση τον τύπο των δεδομένων που εκχωρούνται σε αυτές, η μνήμη που απαιτείται για την αποθήκευση τους κατανέμεται στην εκάστοτε συγκεκριμένη μεταβλητή.
- Σε αντίθεση με άλλες γλώσσες προγραμματισμού (C++, Java, …), στο R οι μεταβλητές ανατίθενται σε αντικείμενα και όχι στους τύπους δεδομένων

- Στο R, όλα τα αντικείμενα / μεταβλητές αντιγράφονται ή/και τροποποιούνται ανάλογα με τις ανάγκες της εκάστοτε επεξεργασίας.
- Για να αποφευχθούν συνθήκες αναποτελεσματικότητας στο χειρισμό τους, το R εφαρμόζει μια προσέγγιση γνωστή ως **σημασιολογική βελτιστοποίηση αντιγραφής σε τροποποίηση** *(copy-on-modify semantics)* που επιτρέπει σε πολλές μεταβλητές να δείχνουν στο ίδιο αντικείμενο / ίδια θέση μνήμης μέχρι να τροποποιηθεί μια από αυτές.


```
COPY - ON - MODIFY
x1 <- c(1, 2, 3)      # create a simple numeric vector x1
x2 <- x1              # assign the value of x1 to another vector x2
#   x1 and x2 have exactly the same value (elements)
x1
[1] 1 2 3
x2
[1] 1 2 3
#   when x1 is changed, x2 will remain unchanged
x1[1] <- 0
x1
[1] 0 2 3
x2
[1] 1 2 3
```

Το R αντιγράφει αντικείμενα μόνο όταν χρειάζεται

– Αυτό εμποδίζει να γίνονται αντίγραφα όταν οι μεταβλητές μεταβιβάζονται σε συναρτήσεις που διαβάζουν αλλά δεν τροποποιούν τα ορίσματά τους

Ονοματολογία μεταβλητών στο R

- Το R δεν έχει κάποια εντολή για τη δήλωση του τύπου μιας μεταβλητής, όπως γίνεται με άλλες γλώσσες προγραμματισμού *(int, null, float, double,…)*
- Το **μοναδικό όνομα** *(identifier)* που δίνεται στη μεταβλητή (ως συνάρτηση ή/και ως αντικείμενο) είναι αναγνωριστικό της μεταβλητής. Μια μεταβλητή μπορεί να έχει ένα σύντομο όνομα (*x, y*, …) ή ένα πιο περιγραφικό όνομα (*gnss, datagroup, volume*, …)
  - Για την επιλογή τους ισχύουν βασικοί κανόνες που πρέπει να τηρούνται κατά τη σύνταξη εντολών στο R (συμπεριλαμβανομένων και των ονομάτων των συναρτήσεων)

- *Βέλτιστες πρακτικές χρήσης των βασικών κανόνων ονοματολογίας των μεταβλητών στο R*
  - *έχουν αλλάξει από τις παλαιότερες προς τις τρέχουσες εκδόσεις του R*
- **Τα αναγνωριστικά μπορούν να είναι ένας** συνδυασμός γραμμάτων, ψηφίων, περιόδου **(.)** και υπογράμμισης **(_)**.
- Ένα αναγνωριστικό πρέπει να ξεκινάει με ένα γράμμα ή μια περίοδο. Αν ξεκινάει με περίοδο, δεν μπορεί να ακολουθήσει ένα ψηφίο.
  - (π.χ., *.2way* δεν είναι επιτρεπτό)

Δεσμευμένες λέξεις ή λέξεις κλειδιά

- Γενικά, στον προγραμματισμό, έχουν μια ειδική σημασία.
- Μια λέξη-κλειδί μπορεί να είναι μια εντολή ή μια παράμετρος.
- Ονομάζονται επίσης ***"δεσμευμένα ονόματα"***.

Δεσμευμένες λέξεις στο R

if	else	repeat	while	function
for	in	next	break	TRUE
FALSE	NULL	Inf	NaN	NA
NA_integer_	NA_real_	NA_complex_	NA_character_	...

Είναι ένα σύνολο λέξεων με ιδιαίτερη σημασία που **δεν μπορούν να χρησιμοποιηθούν ως αναγνωριστικό** (όνομα μεταβλητής, συνάρτησης κ.λπ.).

- Παράδειγμα: *TRUE* είναι μια δεσμευμένη λέξη που δηλώνει μια λογική σταθερά στο R, ενώ το *true (ή True, tRue, …)* μπορεί να χρησιμοποιηθεί ως το όνομα μιας μεταβλητής.

```
> TRUE <- 1
Error in TRUE <- 1 : invalid (do_set) left-hand side
to assignment

> True <- 1

> TRUE
[1] TRUE

> True
[1] 1
```

Ευαισθησία της R σε πεζά

R is a case sensitive language

Οι μεταβλητές TRUE και True δεν είναι ίδιες

Σταθερές στο R

- Οι *σταθερές* είναι οντότητες των οποίων η τιμή δεν μπορεί να μεταβληθεί.
- Βασικοί τύποι σταθερών είναι *αριθμητικές σταθερές* και *σταθερές χαρακτήρων*.
- Υπάρχουν ορισμένες *ενσωματωμένες σταθερές* που ορίζονται στο R μαζί με τις τιμές τους, π.χ.
  - pi (3.141593), LETTERS (A,B,C,), letters(a,b,…)
  - month.abb
    ("Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep" "Oct" "Nov" "Dec")

Δεδομένα αριθμητικού τύπου

```
> typeof(5)
[1] "double"

> typeof(5L)
[1] "integer"

> typeof(5i)
[1] "complex"
```

- Όλοι οι αριθμοί υπάγονται στην κατηγορία αυτή. Μπορούν να είναι ακέραιοι αριθμοί *(integer)*, πραγματικοί αριθμοί διπλής ακρίβειας *(real numbers stored in double precision)* ή μιγαδικοί αριθμοί *(complex numbers)*
- Μπορεί να ελεγχθεί ο τύπος τους με τη συνάρτηση *typeof()*

- Οι αριθμητικές τιμές που ακολουθούνται από τον χαρακτήρα ***L*** ή ***i*** (π.χ., 5L ή 5i) θεωρούνται ως ακέραιοι και μιγαδικοί αριθμοί αντίστοιχα.
- Αριθμητικές σταθερές που ξεκινούν με ***0x*** ή ***0X*** και ακολουθούνται από μια ακολουθία χαρακτήρων από τους *0-9 a-f A-F* αντιμετωπίζονται ως δεκαεξαδικοί αριθμοί *(hexadecimal constants)*
  - π.χ., *0xff* αναπαριστά την τιμή 255
- Οι τιμές χαρακτήρων αντιπροσωπεύονται από απλά (') ή διπλά (") εισαγωγικά *(τους οριοθέτες)*. Αυτός ο τύπος δεδομένων χρησιμοποιείται για την έκφραση συμβολοσειρών, π.χ. "method", "linear", "John", "GNSS", …

- Είναι σύμβολα μέσω των οποίων εντέλλεται ο διερμηνέας της ***R*** να εκτελέσει συγκεκριμένους χειρισμούς (ενέργειες) …
  - Διαφορετικές μαθηματικές και λογικές πράξεις επί των μεταβλητών *(συγκρίσεις, υπολογισμούς κάτω από προϋποθέσεις, υπολογισμούς κατά σειρά ή προτεραιότητα, ελέγχους, …)*

Τελεστές του R

- **Αριθμητικοί τελεστές**
  - **+ , - , * , / , %% , %/%**
- **Σχεσιακοί τελεστές**
  - **< , > , <= , >= , == , !=**
- **Λογικοί τελεστές**
  - **& , | , ! , && , ||**
- **Τελεστές εκχώρησης/ανάθεσης**
  - **<- , <<- , = , -> , ->>**
- **Διάφοροι τελεστές (ειδικού σκοπού)**
  - **: , %in%**

Τελεστής	Περιγραφή
<-, << -, =	Εκχώρηση Αριστερά
->, - >>	Εκχώρηση Δεξιά

```
> x <- 5      > x = 9      > 10 -> x
> x           > x          > x
[1] 5         [1] 9        [1] 10
```

Τελεστές εκχώρησης

- Χρησιμοποιούνται για την εισαγωγή δεδομένων και, γενικότερα, για την απόδοση τιμών σε μεταβλητές
  - χωρίς να τυπωθεί το αποτέλεσμα
- Στους συνδυασμούς συμβόλων εκχώρησης *δεν επιτρέπονται κενά*

Αριθμητικοί τελεστές

- Βοηθούν την εκτέλεση βασικών αριθμητικών πράξεων μεταξύ μεταβλητών.
- Η έξοδος μιας αριθμητικής πράξης είναι πάντα ένα αριθμητικό αποτέλεσμα.

```
x <- 10 ; y <- 6
x
y
x+y     # Πρόσθεση
x-y     # Αφαίρεση
x*y     # Πολλαπλασιασμός
x/y     # Διαίρεση
x^y     # Εκθέτης
x%%y    # Υπόλοιπο από τη διαίρεση
x%/%y   # Ακέραιος από τη διαίρεση
```

```
[1] 10
[1] 6
[1] 16
[1] 4
[1] 60
[1] 1.666667
[1] 1e+06
[1] 4
[1] 1
```

Σχεσιακοί τελεστές

- Βοηθούν την εκτέλεση σχεσιακών πράξεων (συγκρίσεων) μεταξύ μεταβλητών.
- Η έξοδος μιας σχεσιακής πράξης είναι πάντα μια λογική τιμή.

```
x <- 5 ; y <- 16
x
y
x<y     # Μικρότερο από
x>y     # Μεγαλύτερο από
x<=5    # Μικρότερο από ή ίσο με
x>=20   # Μεγαλύτερο από ή ίσο με
x==16   # Ακριβώς ίσο με
x!=5    # Δεν είναι ίσο με
```

```
[1] 5
[1] 16
[1] TRUE
[1] FALSE
[1] TRUE
[1] FALSE
[1] FALSE
[1] FALSE
```


Λογικοί τελεστές

!	Λογικό ΟΧΙ (Logical NOT)	Λογική άρνηση (ΟΧΙ) → Παίρνει κάθε στοιχείο του διανύσματος και δίνει την αντίθετη λογική τιμή
&	Στοιχειώδες λογικό ΚΑΙ (Element-wise logical AND)	Συνδυάζει κάθε στοιχείο του πρώτου διανύσματος με το αντίστοιχο στοιχείο του δεύτερου διανύσματος → Δίνει στην αντίστοιχη έξοδο TRUE εάν και τα δύο στοιχεία είναι TRUE.
&&	Λογικό ΚΑΙ (Logical AND)	Εξετάζει μόνο το πρώτο στοιχείο των δύο διανυσμάτων → Δίνει το TRUE μόνο αν και τα δύο είναι TRUE, δηλαδή (TRUE, TRUE = TRUE), όλοι οι άλλοι συνδυασμοί είναι FALSE, δηλαδή (TRUE, FALSE = FALSE), (FALSE, TRUE = FALSE), (FALSE, FALSE = FALSE)
|	Στοιχειώδες λογικό Ή (Element-wise logical OR)	Συνδυάζει κάθε στοιχείο του πρώτου διανύσματος με το αντίστοιχο στοιχείο του δεύτερου διανύσματος → Δίνει στην αντίστοιχη έξοδο TRUE εάν και τα δύο στοιχεία είναι TRUE.
||	Λογικό Ή (Logical OR)	Εξετάζει μόνο το πρώτο στοιχείο των δύο διανυσμάτων → Δίνει το TRUE εάν ένα από αυτά είναι TRUE

- **Λογικοί τελεστές** : *Όλες οι τιμές μηδέν θεωρείται ότι είναι FALSE και όλες οι μη μηδενικές τιμές ότι είναι TRUE*

!	Λογικό ΟΧΙ (Logical NOT)	v <- c(3,0,TRUE,2+2i) !v [1] FALSE TRUE FALSE FALSE	x <- c(TRUE, FALSE, 0, 6) !x [1] FALSE TRUE TRUE FALSE
&	Στοιχειώδες λογικό ΚΑΙ (Element-wise logical AND)	v <- c(3,1,TRUE,2+3i) t <- c(4,1,FALSE,2+3i) v&t [1] TRUE TRUE FALSE TRUE	x <- c(TRUE, FALSE, 0, 6) y <- c(FALSE, TRUE, FALSE, TRUE) x&y [1] FALSE FALSE FALSE TRUE
&&	Λογικό ΚΑΙ (Logical AND)	v <- c(3,0,TRUE,2+2i) t <- c(1,3,TRUE,2+3i) v&&t [1] TRUE	x <- c(TRUE,FALSE,0,6) y <- c(FALSE,TRUE,FALSE,TRUE) x&&y [1] FALSE
|	Στοιχειώδες λογικό Ή (Element-wise logical OR)	v <- c(3,0,TRUE,2+2i) t <- c(4,0,FALSE,2+3i) v|t [1] TRUE FALSE TRUE TRUE	x <- c(TRUE, FALSE, 0, 6) y <- c(FALSE, TRUE, FALSE, TRUE) x|y [1] TRUE TRUE FALSE TRUE
||	Λογικό Ή (Logical OR)	v <- c(0,0,TRUE,2+2i) t <- c(0,3,TRUE,2+3i) v||t [1] FALSE	x <- c(TRUE, FALSE, 0, 6) y <- c(FALSE, TRUE, FALSE, TRUE) x||y [1] TRUE


```
x=1:6
x
x<6
x>1
x>1 & x<6
x>1 && x<6
x>1 | x<6
x>1 || x<6
x==3
x!=3
! x==3
x == c(2,4)
x %in% c(2,4)
```

```
$Rscript main.r
[1] 1 2 3 4 5 6
[1]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
[1] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
[1] FALSE  TRUE  TRUE  TRUE  TRUE FALSE
[1] FALSE
[1] TRUE TRUE TRUE TRUE TRUE TRUE
[1] TRUE
[1] FALSE FALSE  TRUE FALSE FALSE FALSE
[1]  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
[1]  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
[1] FALSE FALSE FALSE  TRUE FALSE FALSE
[1] FALSE  TRUE FALSE  TRUE FALSE FALSE
```

- Για το εάν οι τιμές ενός διανύσματος εμπεριέχονται σε ένα άλλο διάνυσμα είναι προτιμότερος ο τελεστής *%in%*
- Για αριθμητικές συγκρίσεις, οι τελεστές ==, != κλπ δεν είναι κατάλληλοι και αντί γι' αυτούς χρησιμοποιούμε τις συναρτήσεις *identical* και *all.equal*

Διαφορά μεταξύ των τελεστών '%in%' και '=='

```
# The operator %in% is value matching and "returns a vector of the
# positions of (first) matches of its first argument in its second"
> x <- c(1:2); x
[1] 1 2
> y <- rep(1:2,5) ; y
[1] 1 2 1 2 1 2 1 2 1 2
> x %in% y
[1] TRUE TRUE
> y %in% x
 [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
#  Note this output is longer
# The operator == is a logical operator to compare if two variables are exactly equal
> x == y
 [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
> y == x
 [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```



```
x <- c(2,1,8,3)
y <- c(9,4)
x+y
# Element of y is recycled to 9,4,9
    ,4
x-1
# Scalar 1 is recycled to 1,1,1,1
x+c(1,2,3)
```

```
$Rscript main.r
[1] 11  5 17  7
[1] 1 0 7 2
[1]  3  3 11  4
Warning message:
In x + c(1, 2, 3) :
  longer object length is not a multiple
  of shorter object length
```

- Όταν υπάρχει αναντιστοιχία στο μήκος (αριθμός στοιχείων) των χρησιμοποιούμενων διανυσμάτων τιμών, τα στοιχεία στο διάνυσμα με το μικρότερο μήκος *(λιγότερα στοιχεία)* ***ανακυκλώνονται με κυκλικό τρόπο*** ώστε να ταιριάζουν με τον αριθμό των στοιχείων του διανύσματος με το μεγαλύτερο μήκος *(περισσότερα στοιχεία)*
  - R δίνει μια προειδοποίηση εάν το μήκος του μεγαλύτερου διανύσματος δεν είναι ακέραιο πολλαπλάσιο του μικρότερου διανύσματος

Για όλες σχεδόν τις λειτουργίες του …

- στο R, όλα τα δεδομένα εισόδου και εξόδου (από την ανάλυση μέχρι τα αποτελέσματα των δεδομένων), καθώς και ότι άλλα δεδομένα παράγονται ενδιάμεσα, αποθηκεύονται σε ***αντικείμενα*** (objects) → τις πλέον στοιχειώδεις δομές δεδομένων
  - Κάθε αντικείμενο έχει συγκεκριμένο όνομα και αποτελείται από άλλες οντότητες (***αντικείμενα, συνήθως απλούστερης μορφής***)

- Ένα σενάριο *(script)* ή ακόμα και μια ολόκληρη στατιστική ανάλυση μπορεί να θεωρηθεί ως ένας τρόπος για να δημιουργήσει κάνεις κατάλληλα αντικείμενα ενδιαφέροντος *(π.χ., τα στατιστικά αποτελέσματα και τα γραφικά που χρειάζονται)*
  - Τα αντικείμενα διαφέρουν ως προς τον τύπο τους και τον τρόπο αποθήκευσης, καταχώρησης και διαχείρισης τους

```
c(c(1, 2, 3, 4, 5) * 1, c(1, 2, 3, 4, 5) * 2, c(1, 2, 3, 4, 5) * 3)
[1] 1 2 3 4 5 2 4 6 8 10 3 6 9 12 15
kronecker(c(1, 2, 3, 4, 5), c(1, 2, 3))
[1] 1 2 3 4 5 2 4 6 8 10 3 6 9 12 15
```

Λίγο πιο κομψό: η δράση της **συνάρτησης Kronecker** *σε δύο αντικείμενα !!!*

Η επικοινωνία του χρήστη με το R και οι υπολογισμοί γίνονται …

- Μέσω της διαχείρισης 'αντικειμένων', με βάση συγκεκριμένους κανόνες που αφορούν
  - Τύπο, χαρακτηριστικά γνωρίσματα, και τρόπο δημιουργίας τους *(types, attributes, creation)*
  - Δομή *(data structure)*
  - Ονομασία *(naming convention)*
  - Εκχώρηση *(assignment)*
  - Συναρτήσεις *(functions)*
  - Χώρο εργασίας *(workspace)*
  - Ιστορικό εκτελεσμένων εντολών *(history)*

Βασικός χαρακτηρισμός αντικειμένων

- Ονοματολογία: με πεζά ή/και κεφαλαία λατινικά γράμματα

Κανόνες ονομασίας

- Να αρχίζουν με κάποιο γράμμα (A-Z ή a-z)
  - *Όχι ορισμένα γράμματα μόνα τους ή λέξεις, π.χ. F (=false), T (=true), diff, range, for, repeat, …*
- Να περιέχουν γράμματα, ψηφία (0-9), και/ή "." και/ή "_"
  - Συνιστάται να αποφύγετε τις τελείες (.) μέσα σε ένα όνομα μεταβλητής *(πολλές συναρτήσεις στο R με τελείες στα ονόματά τους)*
- Διάκριση πεζών-κεφαλαίων είναι σημαντική
  - *mydata* διαφορετικό από το *MyData*

Ονοματολογία μεταβλητών στην r

Ισχύοντα και μη έγκυρα ονόματα

- **Ισχύοντα αναγνωριστικά στο R**
  - total, Sum, .fine.with.dot, this_is_acceptable, Number5
- **Μη έγκυρα αναγνωριστικά στο R**
  - tot@l, 5um, _fine, TRUE, .0ne
- Το αναγνωριστικό
  - get.satellite.count *(period.separated συμβολισμός)* συχνά προτιμάται
  - από το get_satellite_count *(snake_case συμβολισμός)* ή εναλλακτικά
  - το getSatelliteCount ή GetSatelliteCount *(lower)camelCase ή (upper)CamelCase συμβολισμός)*

Χαρακτηρισμός αντικειμένων

- **Κλάση** ή **κατηγορία** ή **τάξη** *(class)*: *vector, factor, array, matrix, data.frame, list*
- **Χαρακτηριστικά**
  - *mode*: αριθμός, χαρακτήρας, μιγαδικός αριθμός, λογικές παράμετροι
  - *length*: αριθμός στοιχείων στο αντικείμενο
- **Δημιουργία**
  - Εκχώρηση τιμών
  - Κενά αντικείμενα
- **Εξαγωγή υποσυνόλων από αντικείμενα**
  - Επεξεργασία υποσυνόλων δεδομένων ως ολοκληρωμένες οντότητες

Βασικοί τρόποι χρήσης εντολών στο R

- Ο πρώτος τρόπος χρήσης είναι να εκτελούνται οι εκφράσεις από τη γραμμή εντολών (πολλοί χρήστες πιθανόν να μην απαιτούν κάτι παραπάνω για τις περισσότερες ανάγκες τους)

```
> x<-4
> y<-5
> x+y
[1] 9
> x/y
[1] 0.8
> x%%y
[1] 4
> x%/%y
[1] 0
> x^2+8
[1] 24
```



```
my_var <- 3
my_var
[1] 3

my_var <<- 3
my_var
[1] 3

3 -> my_var
my_var
[1] 3

3 ->> my_var
my_var # print my_var
[1] 3
```

- Η λειτουργία των τελεστών εκχώρησης "<-" *(αντίστοιχα "<<-", "->" , "->>" και "=")* είναι να υπολογίσουν την παρατιθέμενη έκφραση και να την αποδώσουν στη καθορισμένη μεταβλητή, **χωρίς να τυπωθεί το αποτέλεσμα**.

```
x<-c(1,2,3,4,5,6,7)
x
[1] 1 2 3 4 5 6 7
x<-c(1:7)
x
[1] 1 2 3 4 5 6 7
x<-1:4
x
[1] 1 2 3 4
x <- 5.2
x
[1] 5.2
```

Αν ακολουθήσει άλλη εκχώρηση στην ίδια μεταβλητή, η προηγούμενη τιμή αντικαθίσταται


```
# Setup some scalars
x <- 3
y <- 4
z <- sqrt(x*x + y*y)
use.dots.in.names <- z - 5

# Print out z
print(z)
# The default action is print()
z

# So you have a neat scientific calculator
sin(log(2.718281828)*pi)
```

- Ο δεύτερος τρόπος είναι μια μέθοδος κατά την οποία γράφουμε κώδικα για το κάλεσμα συναρτήσεων ή την εκτέλεση ενός συνόλου εντολών του χρήστη → *R scripts*.

- Σε μια ανάλυση δεδομένων στο R λειτουργεί με τη μορφή σειράς υπολογιστικών βημάτων, όπου …
  - σε κάθε βήμα αποθηκεύει τα αποτελέσματα σε αντικείμενα για περαιτέρω ανάλυση
  - Τα αντικείμενα είναι οι δομικές οντότητες που δημιουργεί και χειρίζεται η γλώσσα ***r***
- Συνιστάται να συμβουλευτείτε διάφορα παραδείγματα χρήσιμων εντολών του R
  - https://www.calvin.edu/~scofield/courses/m143/materials/RcmdsFromClass.pdf
  - http://www.personality-project.org/R/r.commands.html

Πρακτική χρήση εντολών προς το R

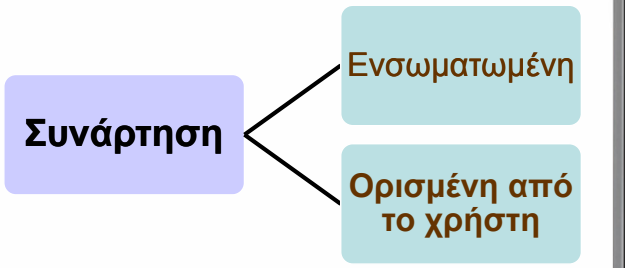
- **Πλεονέκτημα** → σταδιακή ανάπτυξη βημάτων προγραμματισμού, π.χ. …
  - Κάνουν εφικτό να δημιουργηθεί μια συνάρτηση, να εκτελεστεί, να δημιουργηθεί μια άλλη συνάρτηση που να καλεί την πρώτη …
  - Διευκολύνουν με αρκετά απλό τρόπο στη χρήση γραφικών συναρτήσεων → fully programmable graphical capabilities
  - Cut & paste εισαγωγή κώδικα από προηγούμενες εφαρμογές ή παραδείγματα στο Διαδίκτυο

- **Μειονέκτημα** → ανοδική καμπύλη εκμάθησης
  - Απαιτεί αρχικά αρκετή προσπάθεια και ενασχόληση με τη διαλογική λειτουργία του R, για να αποκτηθεί σχετική οικειότητα στη διαμόρφωση και τη χρήση εντολών
  - Ο χρήστης πρέπει να αποφασίσει για τα στάδια μιας ανάλυσης και να τα εκτελέσει βήμα προς βήμα. *Ωστόσο, είναι εύκολο να δημιουργηθούν* ***σενάρια*** *(scripts) με όλα τα επιμέρους βήματα μιας ανάλυσης και να εκτελεστούν δέσμες ενεργειών από τη γραμμή εντολών ή τα μενού του R*

- **Μειονέκτημα** → ***Πλεονεκτήματα***
  - Ο χρήστης του R μπορεί να επωφεληθεί από την ευρεία υποστήριξη από τη σημαντική κοινότητα χρηστών και ερευνητών → **πλεονέκτημα:** η συστηματική ενασχόληση του τον βοηθάει σταδιακά να κατανοήσει το θεωρητικό υπόβαθρο των στατιστικών αναλύσεων
  - Για να χρησιμοποιήσει σωστά το R, ο χρήστης πρέπει να εξοικειωθεί με τον κρίσιμο τρόπο αντίληψης και σύνθεσης εντολών και, να μάθει να χρησιμοποιεί νέους τύπους αντικειμένων και δεδομένων

Χρήση συναρτήσεων στο R

- *Οι συναρτήσεις* είναι ένα θεμελιώδες δομικό στοιχείο του R



- Στο R, υπάρχουν πολλές ενσωματωμένες συναρτήσεις που μπορούν να κληθούν απευθείας χωρίς να τις ορίσουμε.
- Το R επιτρέπει επίσης στους χρήστες να δημιουργήσουν τις δικές τους συναρτήσεις

- *Συναρτήσεις* απαιτούνται για πιο προηγμένες τεχνικές ανάλυσης δεδομένων με το R
- Για την αποτελεσματική χρήση τους απαιτείται μια σταθερή βάση κατανόησης για το *πώς δημιουργούνται, πως χρησιμοποιούνται και πως λειτουργούν*
  - *οι συναρτήσεις είναι αντικείμενα από μόνες τους*
- Σταδιακή εξοικείωση από την υπάρχουσα ή άτυπη γνώση των **ενσωματωμένων** ***(build-in)*** συναρτήσεων του R, μέχρι την αυστηρή κατανόηση εξειδικευμένων ή από τους χρήστες καθορισμένων ***(user-defined)*** συναρτήσεων

Ο ρόλος των συναρτήσεων στο R

- Μια συνάρτηση, ουσιαστικά, είναι ένα αντικείμενο στο οποίο ο διερμηνέας εντολών του R είναι σε θέση να περάσει τον έλεγχο, μαζί με τις παραμέτρους που μπορεί να είναι απαραίτητες για τη συνάρτηση προκειμένου να ολοκληρώσει τις εμπεριεχόμενες εντολές.
  - Στην πραγματικότητα, πολλές από τις συναρτήσεις του R είναι συναρτήσεις μέσα σε συναρτήσεις.
- Η συνάρτηση με τη σειρά της εκτελεί τις εντολές της και επιστρέφει τον έλεγχο στο διερμηνέα,
- καθώς και οποιοδήποτε αποτέλεσμα το οποίο μπορεί να αποθηκεύεται σε άλλα αντικείμενα

Ενδεικτικές συναρτήσεις στο R

	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	$\lfloor x \rfloor$, $\lceil x \rceil$	floor(x), ceiling(x)
φυσικός λογάριθμος	log	log log(x)
Εκθετική συνάρτηση	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)	

Build-in συναρτήσεις		Συναρτήσεις διαχείρισης πινάκων	
abs	acos	cbind	chol
atan	atan2	col	colMeans
beta	ceiling	colnames	colSums
choose	cos	crossprod	det
exp	factorial	diag	dim
floor	gamma	dimnames	matrix
log	log2	ncol, nrow	outer
log10	max	prod	rbind
sqrt	tan	row	rowname
		svd	t

Χρήσιμες build-in συναρτήσεις

options, ifelse, integrate, mtext, cat, pairs, gamma, sign, prod, legend, array, cut, data.restore, outer, dist, rep, if, unlist, abline, arrows, print, deriv, paste, set.seed, ceiling, tabulate, diff, contour, table, quantile, qqplot, boxplot, sample, floor, cor, substring, data.dump, diag, stem, scatter.smooth, text,

Χρήσιμες build-in συναρτήσεις

```
builtins() # List all built-in functions
append()   # Add elements to a vector
cat(x)     # Prints the arguments
cbind()    # Combine vectors by row/column (cf. "paste" in Unix)
julian()     # Return Julian date
length(x)    # Return no. of elements in vector x
ls()         # List objects in current environment
range(x)     # Returns the minimum and maximum of x
rep(1,5)     # Repeat the number 1 five times
rev(x)       # List the elements of "x" in reverse order
seq(1,10,0.4)  # Generate a sequence (1 -> 10, spaced by 0.4)
sequence()     # Create a vector of sequences
sort(x)        # Sort the vector x
order(x)       # list sorted element numbers of x
tolower(),toupper()  # Convert string to lower/upper case letters
unique(x)      # Remove duplicate entries from vector
vector()       # Produces a vector of given length and mode
floor(x), ceiling(x), round(x), signif(x), trunc(x)   # rounding functions
Sys.getenv(x)  # Get the value of the environment variable "x"
Sys.putenv(x)  # Set the value of the environment variable "x"
Sys.time()     # Return system time
Sys.Date()     # Return system date
getwd()        # Return working directory
setwd()        # Set working directory
list.files()   # List files in a give directory
```


Χρήσιμες μαθηματικές συναρτήσεις

```
log(x),logb(),log10(),log2(),exp(),expm1(),log1p(),sqrt()   # Fairly obvious
cos(),sin(),tan(),acos(),asin(),atan(),atan2()       # Usual stuff
cosh(),sinh(),tanh(),acosh(),asinh(),atanh()         # Hyperbolic functions
union(),intersect(),setdiff(),setequal()             # Set operations
+,-,*,/,^,%%,%/%                                     # Arithmetic operators
<,>,<=,>=,==,!=                                      # Comparison operators
eigen()      # Computes eigenvalues and eigenvectors

deriv()      # Symbolic and algorithmic derivatives of simple expressions
integrate()  # Adaptive quadrature over a finite or infinite interval.

sqrt(),sum()
?Control     # Help on control flow statements (e.g. if, for, while)
?Extract     # Help on operators acting to extract or replace subsets of vectors
?Logic       # Help on logical operators
?Mod         # Help on functions which support complex arithmetic in R
?Paren       # Help on parentheses
?regex       # Help on regular expressions used in R
?Syntax      # Help on R syntax and giving the precedence of operators
?Special     # Help on special functions related to beta and gamma functions
```


Συνήθεις συναρτήσεις στατιστικού ενδιαφέροντος

Excel function	R function
NORMSDIST	pnorm(7.2,mean=5,sd=2)
NORMSINV	qnorm(0.9,mean=5,sd=2)
LOGNORMDIST	plnorm(7.2,meanlog=5,sdlog=2)
LOGINV	qlnorm(0.9,meanlog=5,sdlog=2)
GAMMADIST	pgamma(31,shape=3,scale=5)
GAMMAINV	qgamma(0.95, shape=3, scale =5)
GAMMALN	lgamma(4)
WEIBULL	pweibull(6,shape=3,scale =5)
BINOMDIST	pbinom(2,size=20,p=0.3)
POISSON	ppois(2,lambda=3)

Συναρτήσεις Κατανομών Πιθανότητας

Distribution	R name	additional arguments
beta	beta	shape1, shape2, ncp
binomial	binom	size, prob
Cauchy	cauchy	location, scale
chi-squared	chisq	df, ncp
exponential	exp	rate
F	f	df1, df2, ncp
gamma	gamma	shape, scale
geometric	geom	prob
hypergeometric	hyper	m, n, k
log-normal	lnorm	meanlog, sdlog
logistic	logis	location, scale
negative binomial	nbinom	size, prob
normal	norm	mean, sd
Poisson	pois	lambda
Student's t	t	df, ncp
uniform	unif	min, max
Weibull	weibull	shape, scale
Wilcoxon	wilcox	m, n


```
help(package=stats)   # List all stats functions

?Chisquare            # Help on chi-squared distribution functions
?Poisson              # Help on Poisson distribution functions
help(package=survival) # Survival analysis

cor.test()            # Perform correlation test
cumsum(); cumprod(); cummin(); cummax()   # Cumulative functions for vectors
density(x)            # Compute kernel density estimates
ks.test()             # Performs one or two sample Kolmogorov-Smirnov tests
loess(), lowess()     # Scatter plot smoothing
mad()                 # Calculate median absolute deviation
mean(x), weighted.mean(x), median(x), min(x), max(x), quantile(x)
rnorm(), runif()  # Generate random data with Gaussian/uniform distribution
splinefun()           # Perform spline interpolation
smooth.spline()       # Fits a cubic smoothing spline
sd()                  # Calculate standard deviation
summary(x)            # Returns a summary of x: mean, min, max etc.
t.test()              # Student's t-test
var()                 # Calculate variance
sample()              # Random samples & permutations
ecdf()                # Empirical Cumulative Distribution Function
qqplot()              # quantile-quantile plot
```


Στατιστικά μέτρα και μετασχηματισμοί

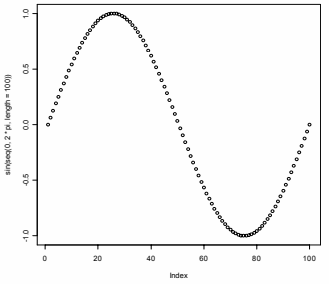
```
max(x, na.rm=TRUE)     #max value in the vector x, no NA values
min(x, na.rm=TRUE)
mean(x, na.rm=TRUE)
median(x, na.rm=TRUE)
sum(x, na.rm=TRUE)
var(x, na.rm=TRUE)     #produces the variance covariance matrix
sd(x, na.rm=TRUE)      #standard deviation
mad(x, na.rm=TRUE)     #(median absolute deviation)
table(x)    #frequency counts of entries
scale(data,scale=FALSE)   #centers around the mean
                          #  but does not scale by the sd)
cumsum(x,na=rm=TRUE)      #cumulative sum, etc.
cumprod(x) ; cummax(x) ; cummin(x)
rev(x)      #reverse the order of values in x
summary(x)       # returns a summary of x: mean, min, max etc.
```


Συναρτήσεις γραφικών

```
help(package=graphics) # List all graphics functions

plot()                 # Generic function for plotting of R objects
par()                  # Set or query graphical parameters
curve(5*x^3.add=T)     # Plot an equation as a curve
points(x,y)            # Add another set of points to an existing graph
arrows()               # Draw arrows [see errorbar script]
abline()               # Adds a straight line to an existing graph
lines()                # Join specified points with line segments
segments()             # Draw line segments between pairs of points
hist(x)                # Plot a histogram of x
pairs()                # Plot matrix of scatter plots
matplot()              # Plot columns of matrices

?device                # Help page on available graphical devices
postscript()           # Plot to postscript file
pdf()                  # Plot to pdf file
png()                  # Plot to PNG file
jpeg()                 # Plot to JPEG file
X11()                  # Plot to X window
persp()                # Draws perspective plot
contour()              # Contour plot
image()                # Plot an image
```


Απλή χρήση ... ως calculator

```
# Natural logs
> log(10)
[1] 2.302585
#  Log using base 10
> log10(10)
[1] 1
# Exponentiation
> exp(2)
[1] 7.389056
# Square Root
> sqrt(4)
[1] 2
# Absolute Value
> abs(-4)
[1] 4
```




```
> plot(sin(seq(0, 2*pi,
length=100)))
```

Βασικές συναρτήσεις διαχείρισης εντολών

```
> source ("script.r")     # Εκτέλεση εντολών από κάποιο αρχείο, π.χ., script.r, στο work directory
#  Για να δούμε τα αντικείμενα (π.χ. μεταβλητές) που έχουν οριστεί στη συγκεκριμένη συνεδρία του R
> objects ()      #  Αντίστοιχα η εντολή rm(.,.,.) διαγράφει αντικείμενα
> ls ()           #  Εντολή ισοδύναμη με την εντολή objects ()
"x" "y"
#  Τα αποτελέσματα των εντολών μιας συνεδρίας με το R αποθηκεύονται σε ένα αρχείο, π.χ., output.txt, στο work directory
> sink ("output.txt")   #  εκτελώντας sink() επαναφέρει την παρουσίαση των αποτελεσμάτων στην κονσόλα
File/save workspace, save history #  Στο τέλος μιας συνεδρίας του R, οι μεταβλητές και οι εντολές που χρησιμοποιήθηκαν αποθηκεύονται
File/load workspace, load history  #  αντίστοιχα ξαναφορτώνονται
```


Μερικοί συμβατικοί συμβολισμοί & κανόνες

- *Το κάλεσμα συναρτήσεων απαιτεί παρενθέσεις ()*, ακόμα και αν δεν απαιτείται να περαστούν παράμετροι στη συνάρτηση:
  - π.χ., *q()* για την έξοδο από το R.
- *Αγκύλες ([ ]) χρησιμοποιούνται για την ανάκληση (subscripting) στοιχείων (π.χ., σε μια λίστα τιμών)*. Οι υποδείκτες μπορεί να είναι θετικοί ή αρνητικοί ακέραιοι, μηδέν, τύπου Boolean (x[x>3]), ή τίποτα.
  - Το σύμβολο [1] στο εξαγόμενο μιας εκχώρησης ή υπολογισμού μιας έκφρασης σημαίνει ότι η καταγραφή ξεκινά από το πρώτο στοιχείο του διανύσματος τιμών
    - αντίστοιχα [k] με κάποιο ακέραιο αριθμό k, σημαίνει ότι η καταγραφή ξεκινά από το k στοιχείο του διανύσματος που εκτυπώνεται


```
> x <- c(3, 1, 4, 1, 5, 9)
> x
[1] 3 1 4 1 5 9
> x[2]
[1] 1
> x[c(2,3)]
[1] 1 4
> x[-2]
[1] 3 4 1 5 9
> x[0]
numeric(0)
> x[x>3]
[1] 4 5 9
> x[x > 3] <- 7
> x
[1] 3 1 7 1 7 7
> x[x > 3] <- c(10, 11, 12)
> x
[1]  3  1 10  1 11 12
```

```
> x <- c(1,2,3)
> names(x) <- c('a','b','c')
> x
a b c
1 2 3
> x <- c(1,2,3)
> x
[1] 1 2 3
> names(x) <- c('a','b','c')
> x
a b c
1 2 3
> x['c']
c
3
```

είδη υποδεικτών στην R

Δημιουργία ακολουθιών δεδομένων

- Πολλές φορές απαιτείται να δημιουργήσουμε ακολουθίες τιμών π.χ., από αριθμητικές επαναλαμβανόμενες ή ισαπέχουσες τιμές σε ένα διάστημα τιμών → χρήσιμες εντολές/συναρτήσεις ***rep*** και ***seq*** . Οι ακολουθίες μπορεί να περιέχουν στοιχεία
  - Σε αύξουσα ή φθίνουσα σειρά
  - Ισαπέχοντα (με βήμα κατά μονάδα ή άλλη τιμή)
  - Μέσα σε επιθυμητό διάστημα τιμών
  - Με θετικούς ή/και αρνητικούς αριθμούς
  - Με ακέραιους ή δεκαδικούς αριθμούς

seq(from= , to= , by= , length= , along=)

from	Δίνει τον πρώτο όρο της ακολουθίας
to	Δίνει τον τελευταίο όρο της ακολουθίας
by	Δίνει το βήμα της ακολουθίας
length	Δίνει το μήκος της ακολουθίας
along	Δίνει το όνομα ενός άλλου διανύσματος. Τότε η R δημιουργεί ακολουθία με μήκος ίδιο με του άλλου διανύσματος.

rep(vector, times= ,each=)

vector	Το διάνυσμα το οποίο θα επαναληφθεί.
times	Αριθμός επαναλήψεων του διανύσματος.
each	Αριθμός επαναλήψεων κάθε στοιχείου του διανύσματος.


```
seq(from = 1, to = 1, by = ((to - from)/(length.out - 1)),
   length.out = NULL, along.with = NULL, ...)
> x <- seq(-6,7)
> x
 [1] -6 -5 -4 -3 -2 -1  0  1  2  3  4  5  6  7
> seq(-2,2,length.out=10)
 [1] -2.0000000 -1.5555556 -1.1111111 -0.6666667
-0.2222222  0.2222222
 [7]  0.6666667  1.1111111  1.5555556  2.0000000

rep(x, ...)
rep.int(x, times)

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


Δημιουργία ακολουθιών δεδομένων

```
> yy <- seq(from=-5.2,to=10.6,by=1.5) ; yy
 [1] -5.2 -3.7 -2.2 -0.7  0.8  2.3  3.8  5.3  6.8  8.3  9.8
>
> zz <- seq(from=-5.2, by=1.5, along=yy) ; zz
[1] -5.2 -3.7 -2.2 -0.7  0.8  2.3  3.8  5.3 6.8 8.3 9.8
>
> ww <- seq(from=-5.2, by=1.5, length=8) ; ww
 [1] -5.2 -3.7 -2.2 -0.7  0.8  2.3  3.8  5.3
>
> vv <- seq(from=-5.2, by=1.5, along=yy) ; vv
 [1] -5.2 -3.7 -2.2 -0.7  0.8  2.3  3.8  5.3  6.8  8.3  9.8
```



```
> rep( c(1,3,5,7), each=3)
 [1] 1 1 1 3 3 3 5 5 5 7 7 7
>
> rep( c(1,3,5,7), c(2,3,3,4))
 [1] 1 1 3 3 3 5 5 5 7 7 7 7
>
> rep( c(1,3,5,7), times=2, each=3)
 [1] 1 1 1 3 3 3 5 5 5 7 7 7 1 1 1 3 3 3 5 5 5 7 7 7
> # first 'each', then 'times' is applied
>
> rep( rep( c(1,3,5,7), each=3), 2)
 [1] 1 1 1 3 3 3 5 5 5 7 7 7 1 1 1 3 3 3 5 5 5 7 7 7
>
> rep ( seq(from=-5.2,to=10.6,by=3.5) , 2)  # seq & rep together
 [1] -5.2 -1.7  1.8  5.3  8.8 -5.2 -1.7  1.8  5.3  8.8
```



```
> a<-LETTERS[1:5] ; a
[1] "A" "B" "C" "D" "E"
> b=seq(1:5) ; b
[1] 1 2 3 4 5
> c=rnorm(5,0,0.36) ; c
[1] -0.8212684 -0.3703826 -0.4873474  0.1154301 -0.1448235
> d <- rep(2.31,5) ; d
[1] 2.31 2.31 2.31 2.31 2.31
> # Which we can combine into a dataframe (more on this later)
> A = data.frame(a,b,c,d) ; A
  a b         c    d
1 A 1 -0.8212684 2.31
2 B 2 -0.3703826 2.31
3 C 3 -0.4873474 2.31
4 D 4  0.1154301 2.31
5 E 5 -0.1448235 2.31
```

```
Sort (or order) a vector into ascending or descending order
sort(x, partial = NULL, na.last = NA, decreasing = FALSE,
        method = c("auto", "shell", "quick", "radix"),
          index.return = FALSE)

rev(x)        # Obtaining vectors sorted into descending order
              # Is more directly achievable by sort(x, decreasing = TRUE)

Rank references the position of the value
in the sorted vector and is in the same order
as the original sequence
rank(x, na.last = TRUE,
     ties.method = c("average", "first", "last",
       "random", "max", "min"))

Order returns the position of the original value
and is in the order of sorted sequence, that
is smallest value to largest value
Order(x)
```

Συναρτήσεις ταξινόμησης και δείκτης δεδομένων

Η συνάρτηση sort() στο R

- Επιστρέφει ταξινομημένα, κατά αύξουσα σειρά από προεπιλογή, τα στοιχεία ενός διανύσματος τιμών που εισάγεται ως είσοδος στη συνάρτηση

```
sort(x,                  # Atomic vector
     decreasing = FALSE, # Whether to sort in increasing or decreasing order
     na.last = TRUE,     # Whether to put NA values at the beginning or at the end
     ...)                # Additional arguments

sort.int(x,              # Atomic vector and factor
         partial = NULL, # Partial sorting indices vector
         decreasing = FALSE, # Same as above
         na.last = TRUE,     # Same as above
         method = c("auto", "shell", "radix"), # Method to be used. Defaults to auto
         index.return = FALSE, # Whether to return the ordering index vector or not
         ...)                  # Additional arguments
```


Η συνάρτηση rank() στο R

- Επιστρέφει την κατάταξη των στοιχείων του ταξινομημένου διανύσματος, με την ίδια σειρά όπως της ακολουθίας του διανύσματος εισόδου
  - Η ελάχιστη τιμή κατατάσσεται στη θέση 1

```
rank(x,                  # Numeric, character or logical vector
     na.last = TRUE,     # Treatment of NAs. How to handle NAs
     ties.method = c("average", "first", "random", "max", "min"))
                         # Treatment of Ties. How to Handle Ties
```


Η συνάρτηση order() στο R

- Επιστρέφει μια μετάθεση *(permutation)* της θέση των στοιχείων του διανύσματος εισόδου, με τη σειρά της ταξινομημένης ακολουθίας τιμών

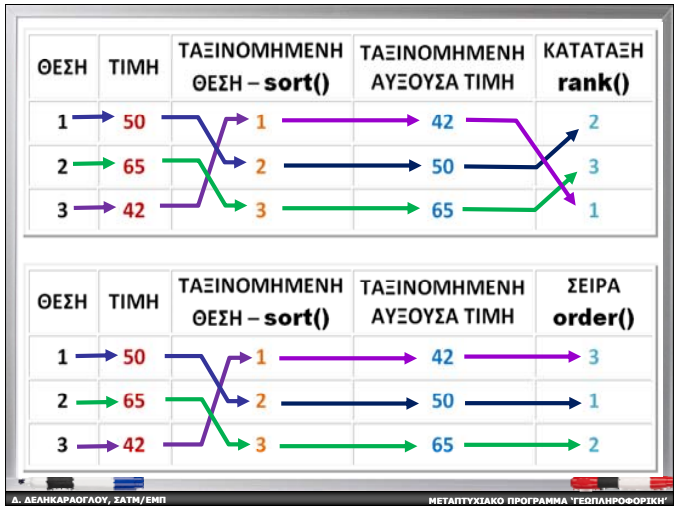
```
order(x,  # Sequence of vectors of the same length
      decreasing = FALSE,  # Whether to sort in increasing or decreasing order
      na.last = TRUE,      # Whether to put NA values at the beginning or at the end
      method = c("auto", "shell", "radix")) # Method to be used. Defaults to auto


sort.list(x, # Atomic vector
          decreasing = FALSE,
          partial = NULL,     # Vector indices for partial sorting
          na.last = TRUE,
          method = c("auto", "shell", "quick", "radix"))
```



```
sort()  -  rank()  -  order()
x <- c(50,65,42)
x
[1] 50, 65, 42
sort(x)
[1] 42, 50, 65
rank(x)
[1] 2, 3, 1
order(x)
[1] 3, 1, 2
```

Rank references the position of the value in the sorted vector and is in the same order as the original sequence

Order returns the position of the original value and is in the order of sorted sequence, that is smallest value to largest value



- *sort()*, *rank()* και *order()* χρησιμοποιούνται όταν απαιτείται να κατατάξουμε ή να αναδιατάξουμε ακολουθίες τιμών σε αύξουσα ή φθίνουσα τάξη μεγέθους

```
> h=rnorm(5,1.70,0.12) ; h
[1] 1.594552 1.814129 1.735637 1.614156 1.538000
>
> height <- round(h, digits=2) ; height
[1] 1.59 1.81 1.74 1.61 1.54
> sort(height)    # ascending order
[1] 1.54 1.59 1.61 1.74 1.81
> rev(height)     # descending order
[1] 1.54 1.61 1.74 1.81 1.59
> rev(sort(height))   # similar to previous
[1] 1.81 1.74 1.61 1.59 1.54
```



```
> x <- c(1:6, 6:4) ; x
[1] 1 2 3 4 5 6 6 5 4
> rev(sort(x))
[1] 6 6 5 5 4 4 3 2 1
> sort(x, partial = c(6,9))   # use indices for partial sorting
[1] 1 2 3 4 4 5 5 6 6
>
> x <- c(1:4, 5:7, 10) ; x
[1]  1  2  3  4  5  6  7 10
> is.unsorted(x)                # FALSE: is sorted
[1] FALSE
> is.unsorted(x, strictly = TRUE) # TRUE : is not (and cannot be)
[1] FALSE
>                               # sorted strictly
```


Παραδείγματα ταξινόμησης

```
> height <- c(1.63, 1.92, 1.84, 1.51, 1.64) ; height
[1] 1.63 1.92 1.84 1.51 1.64
> rank(height)    # ranking of the values
[1] 2 5 4 1 3
>
> height <- c(1.63, 1.92, 1.84, 1.51, 1.64, 1.92) ; height
[1] 1.63 1.92 1.84 1.51 1.64 1.92
> rank(height)    # ranking of the values, when there are ties
[1] 2.0 5.5 4.0 1.0 3.0 5.5
```

```
> height <- c(1.63, 1.92, 1.84, 1.51, 1.64, 1.92, 1.92) ; height
[1] 1.63 1.92 1.84 1.51 1.64 1.92 1.92
> rank(height)  # ranking of the values, when more than 2 ties
[1] 2 6 4 1 3 6 6
> rank(height, ties.method="average") # by averaging the index set of ties
[1] 2 6 4 1 3 6 6
>
> rank(height, ties.method="first")  # increasing by index set of ties
[1] 2 5 4 1 3 6 7
>
> rank(height, ties.method="min")  # replacing by their min index
[1] 2 5 4 1 3 5 5
> rank(height, ties.method="max")  # replacing by their max index
[1] 2 7 4 1 3 7 7
>
> rank(height, ties.method="random")  # puts their indices in random order
[1] 2 7 4 1 3 6 5
```



```
> height <- c(1.63, 1.92, 1.84, 1.51, 1.64, 1.92) ; height
[1] 1.63 1.92 1.84 1.51 1.64 1.92
> zz <- order(height) ; zz    # ordering of the height values
[1] 4 1 5 3 2 6
>
> weight <- c(61, 82, 71, 84, 90, 89) ; weight
[1] 61 82 71 84 90 89
>
> height <- sort(height) # re-arrange heights
> order(height)    # new ordering of the height values
[1] 1 2 3 4 5 6
>
> height
[1] 1.51 1.63 1.64 1.84 1.92 1.92
> weight <- weight[zz] ; weight
[1] 84 61 90 71 82 89
```


Unordered row names

```
set.seed(4)
my_df <- data.frame(x = 1:10, y = 12:21)
rownames(my_df) <- sample(letters, nrow(my_df))
my_df
```

Output

```
   x  y
p  1 12
a  2 13
h  3 14
g  4 15
r  5 16
f  6 17
o  7 18
v  8 19
s  9 20
b 10 21
```

```
my_df[order(rownames(my_df)), ]
```

Output

```
   x  y
a  2 13
b 10 21
f  6 17
g  4 15
h  3 14
o  7 18
p  1 12
r  5 16
s  9 20
v  8 19
```

Ταξινόμηση σειρών αλφαβητικά

Συναρτήσεις ορισμένες από τους χρήστες *(user-defined functions)*

- Αν και η R έχει ένα μεγάλο αριθμό ενσωματωμένων συναρτήσεων *(build-in functions)*, αυτές δεν καλύπτουν πάντα όλες τις παρουσιαζόμενες ανάγκες
  - *Οι χρήστες μπορούν να δημιουργήσουν και τις δικές τους συναρτήσεις*
- Η βασική σύνταξη μιας συνάρτησης είναι:

function.name <- function(*arguments*) {
function body (i.e., purpose of function & computations involving the arguments) }

Συστατικά μιας συνάρτησης

Ονομασία	• Η συνάρτηση αποθηκεύεται ως αντικείμενο με το όνομά της
Ορίσματα	• Είναι προαιρετικά. Μπορούν επίσης να έχουν προεπιλεγμένες τιμές.
Σώμα της συνάρτησης	• Περιέχει ένα σύνολο εντολών που ορίζουν τι κάνει η συνάρτηση
Τιμές εξόδου	• Τα αποτελέσματα που επιστρέφει

Συστατικά στοιχεία των συναρτήσεων

- **Όνομα** / *function name* - είναι το πληροφοριακό όνομα της συνάρτησης, με το οποίο αποθηκεύεται στο περιβάλλον του R ως ένα αντικείμενο με αυτό το όνομα.
  - Είναι προσωπική επιλογή του δημιουργού της συνάρτησης
  - Συνίσταται να αποφεύγονται ονόματα που ήδη χρησιμοποιούνται στη R (π.χ. ονόματα build-in συναρτήσεων, όπως *diff, cumsum, length, factorial, …*)

- **Παράμετροι ή μεταβλητές εισόδου** – Είναι σύμβολα καθορισμού τιμών. Όταν εφαρμόζεται η συνάρτηση, περνάει μια τιμή για κάθε παράμετρο
  - Είναι **τοπικές μεταβλητές / αντικείμενα** (δηλ. αφορούν τη συγκεκριμένη συνάρτηση και δεν έχουν άλλη φυσική αντιστοιχία με άλλες μεταβλητές με το ίδιο όνομα).
    - Το εκάστοτε αντικείμενο που επιστρέφεται μπορεί να είναι οποιοσδήποτε τύπος δεδομένων
  - Είναι **προαιρετικές**, δηλ. μια συνάρτηση μπορεί να περιέχει ή όχι παραμέτρους.
  - Επίσης, **μπορεί να έχουν προκαθορισμένες τιμές.**

- **Το σώμα των εντολών** – Περιέχει μια συλλογή από εντολές που ορίζουν το τι ακριβώς κάνει η συνάρτηση.
  - Μέσα στις εντολές μπορούν να χρησιμοποιηθούν διαφορετικές μεταβλητές (π.χ., για ενδιάμεσους υπολογισμούς)
    - Συνιστάται να δίνονται **αρχικές τιμές** σε αυτές που δεν είναι τοπικές, προκειμένου **να αποφεύγονται 'πλάγια αποτελέσματα'** (δηλ. να αλλάζουν τιμή, αν τις χρησιμοποιούμε και σε άλλες συναρτήσεις, δηλ. είναι καθολικές μεταβλητές που ισχύουν γενικά)
  - Στο R, μπορείτε να δείτε τον κώδικα μιας συνάρτησης πληκτρολογώντας το όνομα της συνάρτησης χωρίς το ()

- **Τιμή επιστροφής** – Αποτελεί την τελευταία έκφραση στο σώμα μιας συνάρτησης που πρέπει να υπολογιστεί

```
test.function <- function(a) {
    for(i in 1:a) {
        b <- i^2
        print(b) } }
```

- Επειδή όλα τα αντικείμενα μέσα στη συνάρτηση είναι τοπικά, δεν θα εμφανίζονται στο χώρο εργασίας. Για να είναι προσιτά εξωτερικά της συνάρτησης, πρέπει να χρησιμοποιηθεί η εντολή *return*
  - π.χ., … *return(list(a,b, absum=sum(a,b)))*

```
# Create a function to print squares of numbers in sequence.
new.function <- function(a) {
   for(i in 1:a) {
      b <- i^2
      print(b)
   }
}
# Call the function new.function supplying 6 as an argument.
new.function(6)
[1] 1
[1] 4
[1] 9
[1] 16
[1] 25
[1] 36
```



```
# Create a function without an argument.
new.function <- function() {
   for(i in 1:5) {
      print(i^2)
   }
}

# Call the function without supplying an argument.
new.function()
[1] 1
[1] 4
[1] 9
[1] 16
[1] 25
```



```
# Create a function with many arguments.
new.function <- function(a,b,c) {
   result <- a * b + c
   print(result)
}

# Call the function by position of arguments.
new.function(5,3,11)
[1] 26
# Call the function by names of the arguments.
new.function(a = 11, b = 5, c = 3)
[1] 58
new.function(a = 11, c = 3, b = 5)
[1] 58
```



```
# Create a function with arguments.
new.function <- function(a = 3, b = 6) {
   result <- a * b
   print(result)
}

# Call the function without giving any argument.
new.function()
[1] 18

# Call the function with giving new values of the argument.
new.function(9,5)
[1] 45
```



```
fahrenheit_to_kelvin <- function(temp_F) {
  temp_K <- ((temp_F - 32) * (5 / 9)) + 273.15
  return(temp_K)
}

fahrenheit_to_kelvin (32)
[1] 273.15


fahrenheit_to_kelvin_to_Celcius <- function(temp_F) {
  temp_K <- ((temp_F - 32) * (5 / 9)) + 273.15
  temp_C <- temp_K - 273.15

result <- list(TK=temp_K,TC=temp_C)
return(result)

}

fahrenheit_to_kelvin_to_Celcius(32)
$TK
[1] 273.15

$TC
[1] 0
```



```
deg2rad <- function(deg) return(deg*pi/180)
rad2deg <- function(rad) return((rad*180)/pi)

# Function to compute the bearing/true course (tc) between two positions
getBearing<-function(lat1, lon1, lat2, lon2) {
lat1<-deg2rad(lat1)       #lat1<-deg2rad(lat1)
lat2<-deg2rad(lat2)       #lat2<-deg2rad(lat2)
lambda<-deg2rad(lon2-lon1)
Y<-sin(lambda)*cos(lat2)
X<-cos(lat1)*sin(lat2)-sin(lat1)*cos(lat2)*cos(lambda)
tc<-atan2(Y,X)
tc<-rad2deg(tc)
if (tc<0.0) {
tc<-360+tc
}
tc
}
getBearing(36.581, 19.099, 38.627, 20.091)
[1] 62.42685
```



```
mytrimmean <- function(x,nstd) {
  averagex=mean(x) ; n=length(x) ; std=sqrt(var(x))
  sum1=0 ; nobs=0
  for (i in 1:n) {
    if(abs((x[i]-averagex)/std) <=nstd) {
    sum1=sum1+x[i] ; nobs=nobs+1
    }
  }
  # the last result to be reported
  sum1/nobs
}

set.seed(1234)
x <- rnorm(5, 0, 1)
x
[1] -1.2070657  0.2774292  1.0844412 -2.3456977  0.4291247
mytrimmean(x,1)
[1] -0.1668373
```

Ο μέσος όρος αποκοπής υπολογίζεται με την απόρριψη συγκεκριμένου ποσοστού από τις χαμηλότερες και τις υψηλότερες τιμές


```
mytrimmean <- function(x,nstd) {
  averagex=mean(x) ; n=length(x) ; std=sqrt(var(x))
  sum1=0 ; nobs=0
  for (i in 1:n) {
    if(abs((x[i]-averagex)/std) <=nstd) {
    sum1=sum1+x[i] ; nobs=nobs+1
    }
  }
  # the last result to be reported
  result = sum1/nobs
list(averagex,n,std,sum1,nobs,result)
}

set.seed(1234)
x <- rnorm(5, 0, 1)
x

mytrimmean(x,1)
```

Αναφορά πολλαπλών στοιχείων

```
[1] -1.2070657  0.2774292
    1.0844412 -2.3456977  0.4291247
[[1]]
[1] -0.3523537

[[2]]
[1] 5

[[3]]
[1] 1.394244

[[4]]
[1] -0.5005118

[[5]]
[1] 3

[[6]]
[1] -0.1668373
```



```
mytrimmean <- function(x,nstd) {
  averagex=mean(x) ; n=length(x) ; std=sqrt(var(x))
  sum1=0 ; nobs=0
  for (i in 1:n) {
    if(abs((x[i]-averagex)/std) <=nstd) {
    sum1=sum1+x[i] ; nobs=nobs+1
    }
  }
  # the last result to be reported
  result=sum1/nobs

  return(c(averagex,n,std,sum1,nobs,result))
}

set.seed(1234)
x <- rnorm(5, 0, 1)
x

mytrimmean(x,1)

[1] -1.2070657  0.2774292  1.0844412 -2.3456977  0.4291247
[1] -0.3523537  5.0000000  1.3942442 -0.5005118  3.0000000 -0.1668373
```

Αναφορά πολλαπλών στοιχείων

Data structures – Δομές δεδομένων

Όλα τα αντικείμενα που χρησιμοποιεί το R έχουν έναν τύπο (type) ή μορφή (mode), καθώς και μια κατηγορία ή κλάση (class)

Type ή mode: πώς τα αντικείμενα αποθηκεύονται στο R

Class: πώς τα αντικείμενα αντιμετωπίζονται από τις συναρτήσεις του R

Data structures – Δομές δεδομένων

- Ο τύπος ή μορφή των αντικειμένων είναι μια ταξινόμηση με βάση την κύρια δομή των στοιχείων τους
- Σύνθετα αντικείμενα μπορούν να περιέχουν στοιχεία του ίδιου ή διαφορετικού τύπου → δηλ. μπορούν να έχουν τύπο ίδιο ή διαφορετικό από τον τύπο των στοιχείων τους
- ***Αντίθετα, η κλάση ή κατηγορία των αντικειμένων είναι μια πιο αναλυτική έννοια*** *που αναφέρεται στο ίδιο το αντικείμενο συνολικά*

Οι βασικές δομές δεδομένων του R μπορούν να οργανωθούν με βάση τη διαστασιολόγησή τους και τον τύπο τους

	Linear (1-D)	Rectangular (2-D)	Multi-dimensional
All Same Type (Homogeneous)	**Atomic VECTORS** are one dimensional, for instance names	**MATRIX** is a collection of data elements arranged in a two-dimensional rectangular layout	**ARRAYS** are multi-dimensional data layouts
Mixed Type (Heterogeneous)	**LIST** is a generic vector containing other objects.	**DATA FRAME** is a list of vectors of equal length, i.e. a data table.	

Data structures – Δομές δεδομένων

Διάσταση διάταξης στοιχείων	Ομοιογενή στοιχεία (ίδιου τύπου)	Ετερογενή στοιχεία (διαφορετικού τύπου)
1d	(Atomic) Vector	List
2d	Matrix	Data frame
nd	Array	

- Τα αντικείμενα μπορούν είναι μεμονωμένες μεταβλητές, διανύσματα αριθμών, μια σειρά χαρακτήρων, συναρτήσεις και γενικότερα σύνθετες δομές που χτίζονται από αυτά τα στοιχεία.

Data structures – Δομές δεδομένων

Διάσταση διάταξης στοιχείων	Ομοιογενή στοιχεία (ίδιου τύπου)	Ετερογενή στοιχεία (διαφορετικού τύπου)
1d	(Atomic) Vector	List
2d	Matrix	Data frame
nd	Array	

- Βαθμωτά μεγέθη *(scalar type)*;
  - Απλοί αριθμοί ή χαρακτήρες, που θα μπορούσαν να εκληφθούν ως βαθμωτά μεγέθη, θεωρούνται ως διανύσματα μιας διάστασης *(single elements)*

Data structures – Δομές δεδομένων

Ο **τύπος** (type) ή **μορφή** (mode) ενός αντικειμένου μπορεί να εξακριβωθεί πληκτρολογώντας ***"typeof"*** ή ***"mode"***

Typical modes are:
- numeric ("integer", "double"),
- complex,
- logical,
- character,
- list,
- raw, expression, name, symbol and function (special, builtin, user-defined)

Data structures – Δομές δεδομένων

Η κατηγορία (class) ενός αντικειμένου μπορεί να εξακριβωθεί πληκτρολογώντας ***"class"***

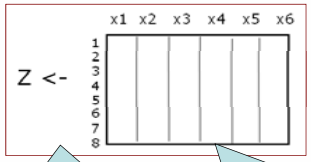
Κατηγορίες:
- Vectors – Διανύσματα είναι ο κύριος τρόπος εισόδου και εξόδου σταθερών ή μεταβλητών ποσοτήτων στο R
- Factors – Παράγοντες είναι μεταβλητές κατηγοριοποίησης που παίρνουν μικρό πλήθος διακεκριμένων τιμών (στάθμες ή βαθμίδες)
- Matrices – Πίνακες (2 διαστάσεων)
- Arrays – Πίνακες (πολλών (>2) διαστάσεων)
- Data frames – Πλαίσια δεδομένων είναι μια συλλογή μεταβλητών ίδιου μήκους αλλά ενδεχομένως διαφορετικού τύπου
- Lists – Λίστες δεδομένων είναι μια συλλογή ανόμοιων αντικειμένων

Data structures – Δομές δεδομένων

Για να διαπιστωθεί αν ένα αντικείμενο ανήκει σε κάποιο τύπο ή μορφή χρησιμοποιείται η εντολή ***is.***⋯ (π.χ. *is.numeric*) και μπορεί να μετατραπεί στο συγκεκριμένο τύπο ή μορφή με την εντολή ***as.***⋯ (π.χ. *as.character*)
- *is.character()*,
- *is.double()*,
- *is.integer()*,
- *is.logical()*, or, more generally,
- *is.atomic()*

```
int_var <- c(1L, 6L, 10L)
typeof(int_var)
    #> [1] "integer"
is.integer(int_var)
    #> [1] TRUE
is.atomic(int_var)
    #> [1] TRUE

dbl_var <- c(1, 2.5, 4.5)
typeof(dbl_var)
    #> [1] "double"
is.double(dbl_var)
    #> [1] TRUE
is.atomic(dbl_var)
    #> [1] TRUE

is.numeric(int_var)
    #> [1] TRUE
is.numeric(dbl_var)
    #> [1] TRUE
```

Παράδειγμα χρήσης ενός **αντικειμένου**



Η ΜΟΡΦΗ του Z καθορίζεται αυτόματα από τα είδη των στοιχείων που αποθηκεύονται σε αυτό το αντικείμενο - αριθμούς, χαρακτήρες, κ.λπ. Εάν είναι ένα μείγμα → mode = λίστα (list)

Η ΚΑΤΗΓΟΡΙΑ του Z μπορεί είτε να προεπιλεγεί, ανάλογα, με το πώς δημιουργήθηκε, ή να καθοριστεί ρητά από το χρήστη. Η κατηγορία των αντικειμένων μπορεί να ελεγχθεί και να αλλαχθεί. Καθορίζει πώς οι διάφορες συναρτήσεις ενεργούν στο Z

Variable Types – Τύποι μεταβλητών

Το R αποθηκεύει και οργανώνει τα δεδομένα ως:

"numeric" ("integer", "double"),
πραγματικοί (ακέραιοι, δεκαδικοί) αριθμοί
"complex", φανταστικοί αριθμοί
"logical", λογικές παράμετροι
"character", αλφαριθμητικοί χαρακτήρες
"list", λίστες
"raw", "expression", "name", "symbol" and
"function" ("special", "builtin", "user-defined")

Ανάλογα με τον τύπο των δεδομένων μιας μεταβλητής, εκχωρείται στο R κατάλληλη μνήμη και αποφασίζεται τι μπορεί να αποθηκευτεί στη δεσμευμένη μνήμη.
Στο R, οι μεταβλητές καθορίζονται με τα αντικείμενα, και ο τύπος ενός αντικειμένου καθορίζει τον τύπο της εκάστοτε μεταβλητής.

Data Types – Numerics / Αριθμητικές μεταβλητές

Το πιο απλό είδος αντικειμένου στο R είναι τα βαθμωτά (scalar) αντικείμενα αριθμητικού τύπου (numeric type ή mode), δηλ. αντικείμενα με μια μόνο αριθμητική τιμή που ωστόσο θεωρούνται διανύσματα με μια συνιστώσα (single elements).

```
> x <- 7   # Δημιουργία του βαθμωτού αντικειμένου x με τιμή 7
> y <- 2   # και του βαθμωτού αντικειμένου y με τιμή 2
#  Τέτοια αντικείμενα επιδέχονται αλγεβρικές πράξεις.
> z <- x+y; z; x-y; x+y+z
[1] 9
[1] 5
[1] 18
> height <- 1.81 ; weight <- 85 ; BMI <- weight / height^2
> BMI
[1] 25.94548
```


Data Types – Numerics / Αριθμητικές μεταβλητές

```
# Ο πιο βασικός τρόπος αποθήκευσης μιας τιμής
# είναι η ανάθεση/εκχώρηση ενός αριθμού σε μια μεταβλητή.
# Όλες οι παρακάτω τρεις εντολές είναι ισοδύναμες
> a <- 3   # Εκχώρηση/αποθήκευση γίνεται ως δεκαδική τιμή
> a=3
> 3 -> a
#  Εκχώρηση μέσω κάποιας επιτρεπτής πράξης με τη μεταβλητή ...
> b <- sqrt(a*a+3)
> b      # Ποια είναι η τιμή της μεταβλητής b; Print the value of b.
[1] 3.464102
# Οι απλοί αριθμοί θεωρούνται ως διανύσματα διάστασης 1.
> x = 10.5
[1] 10.5
> class(x)       # print the class name of x
[1] "numeric"
```


Data Types – Numerics / Αριθμητικές μεταβλητές

```
> k = 1
> k      # Ακόμα και αν έχουμε εκχωρήσει έναν ακέραιο σε μια
[1] 1    # μεταβλητή k, αυτή αποθηκεύεται ως δεκαδικός αριθμός
> class(k)
[1] "numeric"
> is.integer(k)  # is k an integer?
[1] FALSE
> y = as.integer(3)   # Εκχώρηση ακέραιου αριθμού σε μεταβλητή
> y                   # ή αλλιώς, με την ισοδύναμη εντολή ...
> y <- 3L             # χρησιμοποιώντας τον συμβολισμό ' L '
[1] 3
> class(y)
[1] "integer"
> is.integer(y)  # is y an integer?
[1] TRUE
```


Data Types – Numerics / Αριθμητικές μεταβλητές

```
# μπορούμε να εξαναγκάσουμε οποιαδήποτε αριθμητική τιμή,
# σε έναν ακέραιο αριθμό με την ίδια εντολή ' as.integer '
> as.integer(3.14)    # μετατροπή δεκαδικού αριθμού, σε ακέραιο
[1] 3
> as.integer("5.27")  # και αριθμητικοί χαρακτήρες, σε ακέραιο
[1] 5
# Όμως, δεν μπορούμε να εξαναγκάσουμε μεταβλητές χαρακτήρων
> as.integer("temperature")      # coerce an non-decimal string
[1] NA
#  Συχνά, η εντολή as.integer είναι χρήσιμη για την αντιστοίχιση
#     αριθμητικών τιμών σε λογικές τιμές, όπως και στη γλώσσα C.
> as.integer(TRUE)    # TRUE έχει την τιμή 1
[1] 1
> as.integer(FALSE)   # FALSE έχει την τιμή 0
[1] 0
```



```
# Assignment using equal operator.
var.1 = c(0,1,2,3)
# Assignment using leftward operator.
var.2 <- c("Hello","R learner")
# Assignment using rightward operator.
c(TRUE,1) -> var.3
print(var.1)
# The 'cat' function combines multiple items
#      into a continuous print output.
cat ("var.1 is ", var.1 ,"\n")
cat ("var.2 is ", var.2 ,"\n")
cat ("var.3 is ", var.3 ,"\n")
[1] 0 1 2 3
var.1 is 0 1 2 3
var.2 is Hello R learner
var.3 is 1 1
```


Ο τύπος μιας μεταβλητής μπορεί να επανακαθοριστεί δυναμικά

i.e., we can change a variable's data type of the same variable 'var_x' again and again

```
var_x <- "Hello R learner"
cat("The class of var_x is ", class(var_x),"\n")
var_x <- 23.4
cat(" Now the class of var_x is ", class(var_x),"\n")
var_x <- 24L
cat(" Next the class of var_x becomes ", class(var_x),"\n")

The class of var_x is character
Now the class of var_x is numeric
Next the class of var_x becomes integer
```


Data Types – Logical / τιμές λογικών αντικειμένων

Μπορούν να είναι TRUE, FALSE ή και NA (not available), και εκχωρούνται μέσω της σύγκρισης μεταξύ μεταβλητών.

```
> x = 1; y = 2      # εκχωρημένες τιμές στις μεταβλητές x, y
> z = x > y         # Είναι  x > y ;
> z                 # print the logical value
[1] FALSE
> class(z)          # print the 'class' name of z
[1] "logical"
> u = TRUE; v = FALSE
> u & v          # u AND/και v
[1] FALSE
> u | v          # u OR/ή v
[1] TRUE
> !u             # negation of/όχι το u
[1] FALSE
```

Data Types – Characters / Χαρακτήρες

Μια αλληλουχία αλφαριθμητικών χαρακτήρων (character string) ορίζεται χρησιμοποιώντας 'μονά' ή "διπλα" εισαγωγικά

```
> a <- "hello" ; a
[1] "hello"
> b <- c("hello","world") ; b ; b[1]
[1] "hello" "world"
[1] "hello"
> typeof(a)
[1] "character"
> v <- "23.45" ; print(class(v))
[1] "character"
> a = character(20)
> a
[1] "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" ""
```


Data Types – Characters / Χαρακτήρες

Ένα αντικείμενο χαρακτήρων χρησιμοποιείται για να αντιπροσωπεύσει ονομαστικές μεταβλητές. Άλλου τύπου αντικείμενα μετατρέπονται σε μεταβλητές χαρακτήρων με την εντολή as.character ().

```
> x = as.character(3.14) ; x
[1] "3.14"
> class(x)      # print the class name of x
[1] "character"
fname = "Monty"; lname ="Smith"
> paste(fname, lname)  # Concatenate two character values
                       #     with the paste function
[1] "Monty Smith"
> substr("GPS stands for Global Positioning System", start=3,
+ stop=9)
[1] "S stands "      # extract a substring
```


Data Types – Complex / Μηγαδικές μεταβλητές

```
y <- 3 + 5i  ; y
class(y)
[1] "complex"

> sqrt(-1)       # square root of -1
[1] NaN
Warning message:
In sqrt(-1) : NaNs produced

> sqrt(-1+0i)    # square root of -1+0i
[1] 0+1i
> sqrt(as.complex(-1))
[1] 0+1i
```


Special data types – ειδικές τιμές

- *-Inf, Inf :* αντιστοιχούν στις τιμές -∞, +∞
  - Η R τις χρησιμοποιεί για να εκφράσει τα όρια της ακρίβειας υπολογισμών, π.χ.

    > exp(2100) [... $e^{2100} = 2.718^{2100}$?]

    [1] Inf
- *NA* – *Not available*, λογική σταθερά που υποδηλώνει μη διαθέσιμες (δηλ. άγνωστες) τιμές ***(missing values)***
- *NULL* – αντιπροσωπεύει το 'μηδενικό' αντικείμενο στην R
- *NaN* – *Not a Number*, υποδηλώνει τιμές που δεν μπορούν να προσδιοριστούν (π.χ. από μια αριθμητική πράξη με απροσδιόριστο αποτέλεσμα)

Από τα απλούστερα σε πιο σύνθετα αντικείμενα

- Η απλούστερη κατηγορία ή κλάση αντικειμένων είναι τα *ατομικά διανύσματα* (*atomic vectors*), τα οποία απαντώνται σε έξι τύπους: *numeric, logical, integer, complex, character,* και *raw*
- Όλα τα άλλα R αντικείμενα είναι χτισμένα πάνω στα ατομικά διανύσματα, π.χ.,
  - μπορούμε να χρησιμοποιήσουμε πολλά ατομικά διανύσματα και θα δημιουργήσουμε ένα αντικείμενο οποίου η κλάση θα γίνει 'πίνακας (*matrix*)' ή 'πολυδιάστατη διάταξη (*array*)'


```
# Example - Το R παρέχει πολλές συναρτήσεις για
την εξέταση χαρακτηριστικών διανυσμάτων τιμών
και άλλων αντικειμένων

    # class() - τι είδους αντικείμενο είναι
       (υψηλού επιπέδου);
    # typeof() - ποιος είναι ο τύπος δεδομένων
       του αντικειμένου (χαμηλού επιπέδου);
    # length()- ποιό είναι το πλήθος των
       στοιχείων του;
    # attributes() - έχει μεταδεδομένα;

x <- "dataset"
x
typeof(x)          [1] "dataset"
attributes(x)      [1] "character"
class(x)           NULL
                   [1] "character"

y <- 1:10
y                  [1]  1  2  3  4  5  6  7  8  9 10
typeof(y)          [1] "integer"
length(y)          [1] 10
class(y)           [1] "integer"

z <- as.numeric(y)
z                  [1]  1  2  3  4  5  6  7  8  9 10
typeof(z)          [1] "double"
```


Συνοψίζοντας …

- Στην κονσόλα του R, ο χρήστης μπορεί να πληκτρολογήσει διάφορες εκφράσεις.
- Στις περισσότερες γλώσσες προγραμματισμού υπάρχουν μεταβλητές και τύποι μεταβλητών.
  - Η ***r*** "χειρίζεται" τα πάντα ως ***αντικείμενα*** *(objects)*, τα οποία ανήκουν σε μια ***τάξη*** ή ***κλάση*** *(class)*.
    - για την ***r*** τα αντικείμενα είναι οι μεταβλητές, ενώ η κλάση είναι ο τύπος τους.
  - Στην R δεν απαιτείται η ρητή δήλωση της κλάσης στην οποία ανήκουν τα αντικείμενα. Αυτή καθορίζεται αυτόματα από την τιμή που θα ανατεθεί στο αντικείμενο.
- Η ***r*** χρησιμοποιεί, επίσης, βασικές δομές δεδομένων ως κλάσεις αντικειμένων.

Την επόμενη φορά ...

Θα δούμε λεπτομερέστερα τις δομές και χρήσεις μονοδιάστατων και πολυδιάστατων (≥2) αντικειμένων στο R