

ΤΑΧΥΡΡΥΘΜΑ ΜΑΘΗΜΑΤΑ
ΚΕΝΤΡΟ ΔΙΔΑΣΚΑΛΙΑΣ ΚΑΙ ΜΑΘΗΣΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ
ΕΙΣΑΓΩΓΗ ΣΤΗ ΓΛΩΣΣΑ
ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ R

Μιχάλης Κολοσιάτης

23 ΦΕΒΡΟΥΑΡΙΟΥ 2019

- ▶ Η R είναι μια **γλώσσα προγραμματισμού**, της οποίας η ανάπτυξη και η χρήση έχει αυξηθεί κατακόρυφα τα τελευταία χρόνια.
- ▶ Είναι ίσως η πιο δημοφιλής γλώσσα προγραμματισμού για μαθηματικούς και στατιστικούς.
- ▶ Ένας κύριος λόγος είναι ότι είναι **open-source**, και άρα **δωρεάν** και ότι ο καθένας μπορεί να δημιουργήσει τα δικά του πακέτα, τα οποία και να μοιραστεί με τους άλλους χρήστες της R.
- ▶ Υπάρχουν χιλιάδες τέτοια πακέτα, τα οποία είναι διαθέσιμα (δωρεάν) στο διαδίκτυο.

ΤΙ ΘΑ ΚΑΛΥΨΟΥΜΕ ΣΤΟ ΣΗΜΕΡΙΝΟ ΜΑΘΗΜΑ

- ▶ Εισαγωγή (εγκατάσταση, παράθυρα εργασίας, βοήθεια, αποθήκευση δουλειάς κτλ.)
- ▶ Χρήση ως (απλή) υπολογιστική.
- ▶ Είδη και δημιουργία αντικειμένων (απλοί αριθμοί, διανύσματα, πίνακες, λίστες κ.α.). Πράξεις με αυτά. Εξαγωγή στοιχείων από αυτά.
- ▶ Πακέτα και βιβλιοθήκες.
- ▶ Εισαγωγή και εξαγωγή δεδομένων και αποτελεσμάτων.
- ▶ Τυχαία δείγματα και απλές στατιστικές πράξεις (περιγραφική στατιστική).
- ▶ Γραφήματα.
- ▶ Συναρτήσεις.
- ▶ Απλός προγραμματισμός. Δημιουργία επαναληπτικών και υπό συνθήκη διαδικασιών.

- ▶ Κατεβάστε το δωρεάν στο <https://cran.r-project.org/>.
- ▶ Άνοιγμα: από το αντίστοιχο παράθυρο στο Desktop ή από το Start Menu (για Windows). Θα ανοίξει ένα παράθυρο που θα ονομάζεται RGUI (R Graphical User Interface). Εκεί θα είναι ανοικτή η επιφάνεια εργασίας (R Console) όπου θα εκτελούνται οι εντολές. Στη συνέχεια θα δούμε πως μπορούμε να ανοίγουμε, να γράφουμε και να εκτελούμε εντολές και σε άλλα παράθυρα μέσα στο πρόγραμμα.
- ▶ Ένα άλλο, ισοδύναμο πρόγραμμα, είναι το R Studio.
- ▶ Κλείσιμο: είτε με το κουμπί κλεισίματος του παραθύρου του προγράμματος, είτε πληκτρολογώντας `q()` στο R Console.
- ▶ Προσοχή! Μην ξεχνάτε να σώζετε ό,τι θα χρειαστείτε πριν βγείτε από το πρόγραμμα (κατ'ακρίβεια, η R μας το υπενθυμίζει αυτό μόλις πάμε να βγούμε)!

- ▶ Στην αρχή: `help.start()`
- ▶ Για συγκεκριμένες εντολές: `help(xx)` ή `help("xx")` ή `?xx`, όπου `xx` είναι το όνομα της εντολής (π.χ. `help("abs")`).
- ▶ Για να ψάξουμε εντολές για ένα θέμα: `help.search("xx")`, όπου `xx` είναι ο όρος που θέλουμε να ψάξουμε (π.χ. `help.search("regression")`).

ΣΩΖΩΝΤΑΣ ΕΝΤΟΛΕΣ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ

- ▶ Μπορούμε να σώσουμε το τι δημιουργήσαμε (συναρτήσεις, αντικείμενα κτλ) σαν .Rdata. Από το παράθυρο, File— >Save Workspace.
- ▶ Μπορούμε να σώσουμε ένα script (ένα αρχείο με εντολές που γράψαμε για ευκολία μας ή για πιο εύκολη αποθήκευση) με File— >Save όταν είμαστε στο παράθυρο του script αυτού.
- ▶ Γραφικές παραστάσεις μπορούν να σωθούν (και μάλιστα σε διαφορετικές μορφές όπως pdf ή ps) με File— >Save όταν είμαστε στο παράθυρο του γραφήματος αυτού.
- ▶ Μπορούμε τέλος να σώσουμε το ιστορικό των εντολών που χρησιμοποιήσαμε με File— >Save History από το κύριο παράθυρο. Μετά, μπορούμε να τα επαναφέρουμε αυτά με load ή open.
- ▶ Επίσης χρήσιμο είναι να κρατάμε σημειώσεις με #. Οτιδήποτε περιλαμβάνεται μετά από αυτό το σύμβολο δεν εκτελείται.
- ▶ **Αν θέλετε, μπορείτε να γράψετε τις εντολές που θα κάνουμε σήμερα είτε σε script και να το σώσετε, είτε να σώσετε το ιστορικό. Πιθανόν να θέλετε να κρατήσετε και τα αντικείμενα που θα δημιουργήσετε.**

- ▶ Στο κυρίως παράθυρο R Console, πάμε File— >New Script. Θα ανοίξει ένα νέο παράθυρο.
- ▶ Εκεί γράφουμε τις εντολές που θέλουμε. ΔΕΝ εκτελούνται, όμως!
- ▶ Για να εκτελεστούν, τα τονίζουμε (highlight) και πατάμε Edit— >Run Line of Selection ή πατάμε Ctrl+R ή τα αντιγράφουμε στο R Console.
- ▶ Αν είμαστε στο script αλλά δεν έχουμε τονίσει κάτι, εκτελείται η εντολή στην οποία είναι ο δείκτης του ποντικιού (και ο δείκτης μετακινείται μια γραμμή πιο κάτω).

Στο κυρίως παράθυρο ή στο script γράφουμε εντολές της μορφής:

- ▶ `1+3, 1-3, 1*3, 1/3, 2^3.`
- ▶ Ο αριθμός $\pi=3.14159\dots$ γράφεται σαν `pi`.
- ▶ Άλλες συναρτήσεις: `sqrt(5), abs(-4), log(5), exp(5), log10(4), sin(-2.3), cos(-0.2), tan(5), factorial(8).`

ΕΙΔΗ ΑΝΤΙΚΕΙΜΕΝΩΝ (objects) ΣΤΗΝ R

Αυτά δηλώνονται (δημιουργούνται) με `<-` ή `=`.

- ▶ Απλοί αριθμοί (scalars), π.χ. `a <- 3` ή `a = 3`.
- ▶ Διανύσματα, π.χ. `a <- c(1,2,3)` ή `a <- rep(2,5)`.
- ▶ Πίνακες, λίστες, παράγοντες, data frames.
- ▶ Συναρτήσεις (functions).

- ▶ Δημιουργία: με `<-` ή `=`, π.χ. `a <- 3` ή `a = 3`.
Προσοχή: με τις πιο πάνω εντολές, η R δεν θα τυπώσει το αποτέλεσμα. Για να δούμε την τιμή του `a`, θα πρέπει να πληκτρολογήσουμε `a` και θα πάρουμε `[1] 3`.
- ▶ Μετά, μπορούμε να χρησιμοποιήσουμε την τιμή αυτή, π.χ. σε συνέχεια του πιο πάνω, `a+3` θα δώσει `[1] 6`.
- ▶ Αν ξαναορίσουμε την τιμή ενός αντικειμένου, η τιμή αυτή αλλάζει, π.χ. `a <- -11` και μετά `a` δίνει `[1] -11`.
- ▶ Βεβαίως, μπορούμε να χρησιμοποιήσουμε μια μεταβλητή για να δημιουργήσουμε μια άλλη, π.χ. `b <- 2 ^ a - 3`.
- ▶ Για να δούμε μια λίστα των αντικειμένων που έχουμε δημιουργήσει (όχι μόνο τα απλά, αλλά όλα): `ls()` ή `objects()`.
- ▶ Διαγραφή: για να διαγράψουμε κάποια αντικείμενα (π.χ. τα `xx,yy`): με `rm(xx,yy)`.

- ▶ Μπορούμε να δημιουργήσουμε αντικείμενα χαρακτήρων (γράμματα, λέξεις κτλ.), και όχι αριθμών. Σε αυτή την περίπτωση, πρέπει να βάλουμε το όνομα της λέξης σε εισαγωγικά, π.χ. `d <- "a"`
- ▶ Αν μετά πληκτρολογήσουμε `d`, θα πάρουμε `[1] "a"`.
- ▶ Προφανώς, δεν μπορούμε να πάρουμε αριθμητικές πράξεις με αυτά, π.χ. `d - 4` θα δώσει μήνυμα σφάλματος.

ΔΗΜΙΟΥΡΓΙΑ ΚΑΙ ΠΡΑΞΕΙΣ ΜΕ ΔΙΑΝΥΣΜΑΤΑ

- ▶ Μπορούμε να κατασκευάσουμε ένα διάνυσμα με την εντολή `c()` και βάζοντας τους αριθμούς που θέλουμε μέσα στην παρένθεση, χωρισμένους με κόμματα. Για παράδειγμα, `c(1,2,4,-4)`.
- ▶ Όλες οι πιο πάνω πράξεις μπορούν να γίνουν και με διανύσματα, όπου η πράξη εκτελείται στο κάθε στοιχείο του διανύσματος. Για παράδειγμα, `c(1,2,4,-4)+5` θα δώσει `[1] 6 7 9 1` και `sqrt(c(1,2,3,4,5))` θα δώσει `[1] 1.000000 1.414214 1.732051 2.000000 2.236068`.
- ▶ Μπορούμε επίσης να πάρουμε πράξεις μεταξύ δύο διανυσμάτων (αρκεί να έχουν το ίδιο μήκος), π.χ. `c(1,2,4)*c(3,3,2)`
`[1] 3 6 8`.
Σημείωση: αν τα διανύσματα δεν έχουν το ίδιο μήκος, πάλι μπορούμε να κάνουμε πράξεις, αλλά δεν είναι ξεκάθαρο (και ίσως αντιδιαισθητικό) το αποτέλεσμα.

- ▶ Επανάληψη αριθμών: `rep(3,5)`.
- ▶ Αριθμητική πρόοδος (απλή): `1:5` ή `seq(1,5,by=1)`.
- ▶ Αριθμητική πρόοδος με καθορισμένη απόσταση:
`seq(1,15,by=2)`.
- ▶ Αριθμητική πρόοδος με καθορισμένο μήκος:
`seq(1,16,length=4)`.
- ▶ Άλλα παραδείγματα:
`1:-5,`
`seq(16,1,by=-2)`.

- ▶ Έχουμε δει εντολές δημιουργίας διανυσμάτων. Χρησιμοποιώντας $<-$ ή $=$, μπορούμε να δημιουργήσουμε αντικείμενα διανυσμάτων. Για παράδειγμα,
`g <- c(1,2,4)`, `h <- 1:5`, `b <- rep(0,6)` κτλ.
Οι εντολές αυτές δεν παρουσιάζουν τα αποτελέσματα. Για να δούμε το `g`, για παράδειγμα, γράφουμε `g`.
- ▶ Όπως με τα απλά αντικείμενα, μπορούμε να ξαναορίσουμε την τιμή κάποιου διανύσματος ή να το διαγράψουμε, με τις ίδιες εντολές.
- ▶ Με `ls()` ή `objects()`, θα δουμε και τα διανύσματα.
- ▶ Μπορούμε να χρησιμοποιήσουμε τα υπάρχοντα διανύσματα για να ορίσουμε καινούρια, π.χ. `h1 <- g-3`.
- ▶ Μπορούμε επίσης να χρησιμοποιήσουμε το `"c"` για να ενώσουμε δύο διανύσματα, π.χ. `a<-c(1,2)`
`b=c(11,12,13,15)`
`c(a,b)` δίνει `[1] 1 2 11 12 13 15`.
- ▶ Μπορούμε να δημιουργήσουμε διανύσματα χαρακτήρων, π.χ.
`g1 =c("a","cc")`.

- ▶ Διάνυσμα και αριθμός: πράξη με το κάθε στοιχείο του διανύσματος με τον αριθμό, π.χ. $g*3$.
 - ▶ Πράξη πάνω σε ένα διάνυσμα: πράξη πάνω στο κάθε στοιχείο του διανύσματος, $\sin(g)$.
 - ▶ Διάνυσμα και διάνυσμα: πράξεις των αντίστοιχων στοιχείων των δύο διανυσμάτων (το 1ο με το 1ο, το 2ο με το 2ο κοκ.), π.χ. $g+h$.
- ΠΡΟΣΟΧΗ: αν τα διανύσματα δεν έχουν το ίδιο μήκος, πάλι μπορούμε να κάνουμε πράξεις, αλλά δεν είναι ξεκάθαρο (και ίσως αντιδιαισθητικό) το αποτέλεσμα.

- ▶ Για το άθροισμα των στοιχείων ενός διανύσματος: με `sum(g)`.
- ▶ Για να δούμε το μήκος ενός διανύσματος: με `length(xx)`.
- ▶ Για να δούμε το k -στό στοιχείο ενός διανύσματος: με `xx[k]`, π.χ. `b[2]`.
- ▶ Πιο γενικά, για να δούμε κάποια στοιχεία ενός διανύσματος: όπως πιο πάνω, αλλά αντί k θα έχουμε ένα διάστημα, π.χ. `b[c(2,4)]` ή `b[2:4]`.
- ▶ Για να μην εμφανιστεί το k -στό στοιχείο ενός διανύσματος: με `xx[-k]`, π.χ. `b[-2]`.
- ▶ Για να διαγράψουμε το k -στό στοιχείο ενός διανύσματος: με `xx<-xx[-k]`, π.χ. `b<-b[-2]`.
- ▶ Για να αλλάξουμε το k -στό στοιχείο ενός διανύσματος: με `xx[k]<- n`, όπου n είναι η νέα τιμή, π.χ. `b[1]<-111`.

ΠΑΡΑΓΟΝΤΕΣ (factors)

- ▶ Τα αντικείμενα αυτά είναι βασικά διανύσματα, στα οποία έχουμε κάποιες συγκεκριμένες τιμές, οι οποίες θεωρούνται ως κατηγορίες.
- ▶ Οι τιμές αυτές μπορεί να είναι είτε αριθμοί είτε κάποιοι χαρακτήρες.
- ▶ Η σχετική εντολή δημιουργίας είναι η `as.factor()`.
- ▶ Για παράδειγμα, το φύλο ή η δοσολογία ενός φαρμάκου. Στην πρώτη περίπτωση, μπορούμε να πάρουμε π.χ. `fact1 <- as.factor(c("M", "M", "F", "F", "F"))` και στη δεύτερη περίπτωση `fact2 <- as.factor(c(1,2,2,2,3,3,3,1))`.

ΠΑΡΑΓΟΝΤΕΣ (factors)

- ▶ Μπορούμε να δούμε τις κατηγορίες ενός παράγοντα με `levels()`.
- ▶ Μπορούμε επίσης να ονομάσουμε διαφορετικά τις κατηγορίες ενός παράγοντα με `levels() <- c("xx1","yy1")`, όπου "xx1", "yy1" είναι οι νέες ονομασίες.
- ▶ Τέλος, με την εντολή `table()` παίρνουμε τον αριθμό παρατηρήσεων σε κάθε κατηγορία, ενώ με την εντολή `prop.table(table())` παίρνουμε το ποσοστό παρατηρήσεων σε κάθε κατηγορία.

ΠΙΝΑΚΕΣ

ΚΑΤΑΣΚΕΥΗ

Υπάρχουν αρκετοί τρόποι να δημιουργήσουμε πίνακα (matrix) στην R:

- ▶ Δημιουργούμε τις στήλες (σαν διανύσματα) και μετά τις ενώνουμε με την εντολή `cbind`.
- ▶ Δημιουργούμε τις γραμμές (σαν διανύσματα) και μετά τις ενώνουμε με την εντολή `rbind`.
- ▶ Με την εντολή `matrix`: `matrix(xx,nrow=m)` ή `matrix(xx,ncol=n)`, όπου `xx` είναι διάνυσμα με τους αριθμούς που θα μπουν στον πίνακα, `nrow` είναι ο αριθμός γραμμών που θέλουμε και `ncol` ο αριθμός στηλών που θέλουμε.
- ▶ Προσοχή: η τελευταία εντολή κατασκευάζει τον πίνακα παίρνοντας τους αριθμούς του διανύσματος και βάζοντάς τους σε στήλες. Αν θέλουμε να μπουν οι αριθμοί αυτοί σε γραμμές, βάζουμε `byrow=T` μέσα στην εντολή του πίνακα.

ΠΙΝΑΚΕΣ ΚΑΤΑΣΚΕΥΗ

Για παράδειγμα:

- ▶ `matrix(1:6,nrow=2)` και `matrix(1:6,nrow=2,byrow=T)` δίνουν διαφορετικά αποτελέσματα (δοκιμάστε τα).
- ▶ Οι πιο πάνω εντολές θέλουν προσοχή! Τί γίνεται αν τα διανύσματα δεν είναι της ίδιας διάστασης στα πρώτα δύο (`cbind`, `rbind`) ή αν ο αριθμός των στοιχείων στο διάνυσμα δεν είναι πολλαπλάσιο του αριθμού γραμμών ή στηλών στο τελευταίο (`matrix`);
- ▶ Εννοείται (και συνιστάται) να σώζουμε πίνακες σαν αντικείμενα, π.χ. `A <- matrix(c(1,13,2,5,3,5),nrow=2,byrow=T)`.

- ▶ Πίνακας με αριθμό: εκτελείται η πράξη σε κάθε στοιχείο του πίνακα ξεχωριστά, π.χ. $A+3$.
- ▶ Απλή εντολή σε πίνακα: εκτελείται η πράξη σε κάθε στοιχείο του πίνακα ξεχωριστά, π.χ. $\log(A)$.
- ▶ Ειδικές εντολές για πίνακες:
 - Ανάστροφος: με $t(A)$.
 - Αντίστροφος: με $\text{solve}(A)$.
 - Ορίζουσα: με $\text{det}(A)$.

Έστω για παράδειγμα δυο πίνακες A, B .

- ▶ Πρόσθεση ή αφαίρεση: για κάθε αντίστοιχία στοιχείων, π.χ. $A+B$. Οι πίνακες πρέπει να είναι της ίδιας διάστασης!
- ▶ Πολλαπλασιασμός: εδώ θέλει προσοχή. Αν γράψουμε $A*B$, τότε οι πράξεις εκτελούνται πάλι στα αντίστοιχα στοιχεία. Οι πίνακες πρέπει να είναι της ίδιας διάστασης!
- ▶ Αν, από την άλλη, γράψουμε $A \% * \% B$, τότε μιλάμε για πολλαπλασιασμό πινάκων (και οι δυο πίνακες πρέπει να έχουν κατάλληλες διαστάσεις, πιο συγκεκριμένα ο αριθμός στηλών του 1ου πρέπει να είναι ίσος με τον αριθμό γραμμών του 2ου πίνακα).

- ▶ Διαστάσεις ενός πίνακα: `nrow(a)`, `ncol(a)`, `dim(a)` (και `length(a)`).
- ▶ Όπως με τα διανύσματα, μπορούμε να πάρουμε ή να αλλάξουμε κάποιο στοιχείο του πίνακα, ή ολόκληρες γραμμές και στήλες. Οι εντολές είναι, για παράδειγμα:
`a[1,3]` (στοιχεία), `a[1,]` (γραμμές), `a[,3]` (στήλες).
- ▶ Μπορούμε να πάρουμε και πιο σύνθετα πράγματα, π.χ.
`a[2,c(1,3)]` (στοιχεία 1 και 3 της 2ης γραμμής) ή `a[1:2,]` (πρώτες 2 γραμμές).
- ▶ Τί γίνεται αν αφαιρέσουμε ένα στοιχείο του πίνακα, π.χ.
`a[-c(1,1)]`;

- ▶ Τα αντικείμενα data frames είναι σαν πίνακες, αλλά χωρίς τον περιορισμό ότι πρέπει να έχουμε μόνο αριθμούς. Κάποιες στήλες μπορεί να είναι, για παράδειγμα, αριθμοί και κάποιες άλλες παράγοντες ή χαρακτήρες.
- ▶ Για παράδειγμα, μπορεί να έχουμε στην 1η στήλη την πόλη, στη 2η τον πληθυσμό και στην 3η την έκταση.
- ▶ Η εντολή κατασκευής είναι η `data.frame()`.
Για παράδειγμα, `a1 <- as.factor(c("N","L","P"))`.
`d1 <-`
`data.frame(city=a1,pop=c(100,200,50),size=c(20,28,33))`.

- ▶ Αυτό θα δώσει

```
city pop size
1  N  100  20
2  L  200  28
3  P   50  33
```

- ▶ Για να δούμε τα ονόματα των στηλών: `names(d1)`.
- ▶ Για να πάρουμε μια συγκεκριμένη στήλη, χρησιμοποιούμε το `$`:
`colu1=d1$city`.
- ▶ Μπορούμε επίσης να πάρουμε συγκεκριμένα στοιχεία από το data frames, π.χ. `d1$city[2]` ή `d1[2,1]` (και τα δύο αυτά θα δώσουν L).

- ▶ Οι λίστες είναι τα πιο γενικά αντικείμενα στην R. Είναι βασικά μια συλλογή απλών αριθμών ή/και διανυσμάτων ή/και πινάκων ή/και παραγόντων ή/και data frames. Επίσης, δεν χρειάζεται να είναι όλα τα αντικείμενα της ίδιας διάστασης.
- ▶ Κατασκευή: με την εντολή `list()`.
Για παράδειγμα,
`a <- 5`
`b <- "hello"`
`c <- 1:4`
`d <- matrix(1:6, ncol=3)`
`list1 <- list(sc=a, ch=b, vec=c, mat=d)`

- ▶ Για να δούμε τα ονόματα των στοιχείων της λίστας: `names(list1)`.
- ▶ Εδώ δεν μπορούμε να χρησιμοποιήσουμε το 2ο τρόπο που είχαμε δει για data frames για να πάρουμε κάποιο στοιχείο. Ο 1ος τρόπος, όμως, δουλεύει: `list1$ch`, `list1$vec[3]`, `list1$mat[1,1]`, `list1$vec[3]`.
Αντί για το `$`, μπορούμε να χρησιμοποιήσουμε `[[]]`.
Για παράδειγμα, `list1[[3]]` αντί `list1$vec`.

- ▶ Όπως έχουμε αναφέρει, υπάρχουν πολλές έτοιμες συναρτήσεις (γραμμένες σε scripts) στην R. Υπάρχουν επίσης πολλές συλλογές δεδομένων. Τέτοιες συλλογές δεδομένων και συναρτήσεων είναι μαζεμένες σε πακέτα (packages) για χρήση.
- ▶ Αυτά μπορεί να προέρχονται από την ομάδα που διαχειρίζεται την R ή από χρήστες, οι οποίοι τις ανεβάζουν για να τις χρησιμοποιήσει όποιος ενδιαφέρεται.
- ▶ Για να χρησιμοποιήσουμε ένα πακέτο (τα περιεχόμενά του, δηλαδή), πρέπει πρώτα να το κατεβάζουμε (load) και να τα εγκαταστήσουμε (install). Το πρώτο χρειάζεται να γίνει μια φορά, ενώ το άλλο κάθε φορά που ανοίγουμε την R (και αν θέλουμε να χρησιμοποιήσουμε το συγκεκριμένο πακέτο, φυσικά!).

ΚΑΤΕΒΑΣΜΑ ΚΑΙ ΕΓΚΑΤΑΣΤΑΣΗ ΠΑΚΕΤΩΝ

- ▶ Μπορούμε να δούμε ποια πακέτα είναι ήδη κατεβασμένα με `library()`.
- ▶ Για να κατεβάσουμε ένα πακέτο, μπορούμε να πάμε είτε μέσω Packages— >Install package(s), είτε πληκτρολογώντας `install.packages()` στο κύριο παράθυρο. Αν γνωρίζουμε το όνομα του πακέτου (έστω `xx`), μπορούμε να χρησιμοποιήσουμε `install.packages("xx")`.
- ▶ Για να εγκαταστήσουμε ένα πακέτο (αφού το έχουμε πρώτα κατεβάσει!): είτε μέσω Packages— >Load package, είτε πληκτρολογώντας `library(xx)` ή `library("xx")` στο κύριο παράθυρο, όπου `xx` είναι το όνομα του πακέτου.
- ▶ Για να δούμε μετά το τι περιέχει το πακέτο: με `??xx` στο κύριο παράθυρο, όπου `xx` είναι το όνομα του πακέτου.

- ▶ Όπως έχουμε αναφέρει, αρκετές συλλογές δεδομένων υπάρχουν σε πακέτα. Άρα, όταν κατεβάσουμε και εγκαταστήσουμε ένα πακέτο, έχουμε πρόσβαση και στα δεδομένα αυτά.
- ▶ Για να δούμε τη λίστα όλων των διαθέσιμων δεδομένων (από τα πακέτα που είναι, δηλαδή, κατεβασμένα και εγκατεστημένα εκείνη τη στιγμή), πληκτρολογούμε `data()` στο R Console.
- ▶ Με την εντολή `data(package = .packages(all.available= TRUE))` βρίσκουμε δεδομένα και από πακέτα που είναι κατεβασμένα αλλά όχι εγκατεστημένα.

- ▶ Για να τρέξουμε κάποια συλλογή δεδομένων: με `data(xx)`, όπου `xx` είναι το όνομα της συλλογής. Το όνομα της συλλογής αυτής θα εμφανίζεται μετά αν πληκτρολογήσουμε `ls()`.
- ▶ Αφού το τρέξουμε, το σύνολο δεδομένων είναι ένα αντικείμενο, όπως τα άλλα, και άρα μπορούμε να το χρησιμοποιήσουμε ανάλογα.
- ▶ Μπορεί να είναι πίνακας, διάνυσμα, λίστα κτλ.

- ▶ Εκτός από δεδομένα που υπάρχουν σε πακέτα, μπορούμε να εισάξουμε δεδομένα και από άλλες πηγές.
- ▶ Η πιο συνηθισμένη μορφή στην οποία αποθηκεύονται, και η οποία είναι πολύ εύκολο να εισαχθεί μετά στην R, είναι σε .txt.
- ▶ Αν τα δεδομένα είναι σε μορφή πίνακα σε ένα .txt αρχείο, μπορούμε να τα εισάξουμε με `read.table`. Για παράδειγμα, `w1 <- read.table(file="BOD-matrix.txt", header=TRUE)`. Το τελευταίο κομμάτι λέει στην R ότι η 1η γραμμή του αρχείου είναι τα ονόματα των στηλών (και όχι μέρος των δεδομένων). ΠΡΟΣΟΧΗ: Βεβαιωθείτε ότι είστε στο σωστό φάκελο (directory) του υπολογιστή (εκεί που είναι αποθηκευμένο το αρχείο, δηλαδή). Μπορούμε να αλλάξουμε το ευρετήριο (directory) που είμαστε με `File -> Change dir.`

- ▶ Αν τα δεδομένα είναι σε μορφή λίστας σε ένα .txt αρχείο, μπορούμε να τα εισάξουμε με `dget`. Για παράδειγμα, `w1 <- dget(file="BOD-list.txt")`.
ΠΡΟΣΟΧΗ: Βεβαιωθείτε ότι είστε στο σωστό φάκελο (directory) του υπολογιστή (εκεί που είναι αποθηκευμένο το αρχείο, δηλαδή). Μπορούμε να αλλάξουμε το ευρετήριο (directory) που είμαστε με `File -> Change dir`.
- ▶ Τέλος, μπορούμε να εισάξουμε δεδομένα και σε άλλες μορφές. Για παράδειγμα, για δεδομένα σε Excel σωσμένα σε μορφή .csv, μπορούμε να χρησιμοποιήσουμε την εντολή `read.csv()`, για παράδειγμα `read.csv("BOD-excel.csv")`.
Για δεδομένα από την SPSS (σε μορφή .sav), μπορούμε να χρησιμοποιήσουμε την εντολή `read.spss()`, για παράδειγμα `read.spss("BOD-spss.sav")`.
- ▶ Προσοχή! Η πρότελευταία εντολή είναι μέσα στο πακέτο `utils`. Η τελευταία εντολή είναι μέσα στο πακέτο `foreign`. Πρέπει πρώτα να εγκαταστήσουμε και να τρέξουμε το αντίστοιχο πακέτο.

- ▶ Έχουμε δει πως να σώζουμε επιφάνειες εργασίας, scripts κτλ. Έχουμε επίσης δει ότι μπορούμε να σώσουμε δεδομένα της R σε μορφή `.Rdata`.
- ▶ Αν δεν χρειαζόμαστε να σώσουμε τα πάντα που υπάρχουν εκείνη τη στιγμή στην R, μπορούμε να επιλέξουμε τι να σώσουμε με την εντολή `save`, π.χ.
`save(X,y,file="small.matrix.and.vector.rdata")`.
- ▶ Εναλλακτικά, μπορούμε να σώσουμε και κάποια δεδομένα της R σε μορφή `.txt`. Αυτό μπορεί να γίνει με την εντολή `write.table`, π.χ. `write.table(X,file="small.matrix.txt")`.
- ▶ ΠΡΟΣΟΧΗ: Βεβαιωθείτε ότι σώζετε στο φάκελο (directory) του υπολογιστή που θέλετε να σώσετε, για να μην γυρεύετε μετά τα αρχεία αυτά!

ΑΠΛΕΣ ΣΤΑΤΙΣΤΙΚΕΣ ΔΙΑΔΙΚΑΣΙΕΣ (ΠΕΡΙΓΡΑΦΙΚΗ ΣΤΑΤΙΣΤΙΚΗ)

- ▶ Όταν έχουμε να αναλύσουμε κάποια δεδομένα, ένα από τα πρώτα πράγματα που κάνουμε είναι να πάρουμε κάποια αρχικά αποτελέσματα από αυτά, για παράδειγμα τη μέση τιμή, τη διασπορά, το ιστόγραμμα κτλ.
- ▶ Αυτό μπορεί να γίνει με απλές εντολές στην R.
- ▶ Για παράδειγμα, ας πάρουμε

```
w1 <- read.table(file="BOD-matrix.txt", header=TRUE)
```


και ας πάρουμε τη 2η στήλη αυτού (θα πρέπει να είναι προφανές πως να το κάνουμε αυτό πλέον!). Ας την ονομάσουμε `w2`.

ΑΠΛΕΣ ΣΤΑΤΙΣΤΙΚΕΣ ΔΙΑΔΙΚΑΣΙΕΣ ΓΙΑ ΜΙΑ ΜΕΤΑΒΛΗΤΗ

- ▶ Μέσος: `mean(w2)`.
- ▶ Διάμεσος: `median(w2)`.
- ▶ Μέγιστη/ελάχιστη τιμή: `max(w2)` / `min(w2)`.
Εναλλακτικά, η εντολή `range(w2)` δίνει και τις δύο αυτές τιμές.
- ▶ Διασπορά/τυπική απόκλιση: `var(w2)` / `sd(w2)`.
- ▶ Μια άλλη χρήσιμη εντολή, η οποία δίνει κάποια από τα πιο πάνω, συν τα τεταρτημόρια, είναι η `summary(w2)`.

- ▶ Ιστόγραμμα: `hist(w2)`.
- ▶ Γράφημα των τιμών με τη σειρά που παρουσιάζονται (trace plot): `plot(w2)`.
- ▶ Θηκόγραμμα: `boxplot(w2)`.
- ▶ Διάγραμμα πίττα (pie chart): `pie(table(w2))`.

ΑΠΛΕΣ ΣΤΑΤΙΣΤΙΚΕΣ ΔΙΑΔΙΚΑΣΙΕΣ ΓΙΑ ΔΥΟ ΜΕΤΑΒΛΗΤΕΣ

- ▶ Οι επιλογές εδώ είναι πολύ περισσότερες. Αρχικά, μπορούμε να πάρουμε τα πιο πάνω για την κάθε μεταβλητή ξεχωριστά.
- ▶ Μπορούμε επίσης να δούμε τη σχέση μεταξύ των δύο (και τα αντίστοιχα γραφήματα).
- ▶ Έστω η 2η μεταβλητή είναι `w3 <- c(1,3,5,7,3,4.2)` (προσέξτε ότι πρέπει να είναι της ίδιας διάστασης με το `w2`).
- ▶ Συνδιακύμανση (συνδιασπορά): `cov(w2,w3)`.
- ▶ Συντελεστής συσχέτισης: `cor(w2,w3)`.

- ▶ Διάγραμμα διασποράς (scatter plot): `plot(w2,w3)`.
- ▶ Θηκόγραμμα και των δύο, δίπλα-δίπλα: `boxplot(w2,w3)`.

- ▶ Η R, σαν μαθηματική/στατιστική γλώσσα που είναι, έχει ενσωματωμένες εντολές για τη δημιουργία τυχαίων αριθμών από όλες τις συνηθισμένες κατανομές.
- ▶ Η μορφή των εντολών αυτών είναι `rxx()`, όπου `xx` είναι το όνομα της κατανομής (όπως συμβολίζεται στην R) και μέσα στην παρένθεση μπαίνει πρώτα ο αριθμός δειγμάτων που θέλουμε, και μετά οι τιμές των παραμέτρων της κατανομής.
- ▶ ΠΡΟΣΟΧΗ: Οι παράμετροι της κατανομής μπορεί να μην είναι όπως τους έχετε υπόψιν σας, καθώς μπορεί να υπάρχουν 2 ή περισσότερες παραμετρικοποιήσεις (για παράδειγμα, η Εκθετική κατανομή). Ελέγχετε πριν χρησιμοποιήσετε μια κατανομή το πως ορίζεται στην R.

ΔΗΜΙΟΥΡΓΙΑ ΤΥΧΑΙΩΝ ΑΡΙΘΜΩΝ ΣΥΝΗΘΙΣΜΕΝΕΣ ΚΑΤΑΝΟΜΕΣ

Συνάρτηση	Κατανομή
<code>runif(n,min,max)</code>	Uniform
<code>rnorm(n,mean,sd)</code>	Normal
<code>rt(n,df)</code>	Student-t
<code>rexp(n,rate)</code>	Exponential
<code>rgamma(n,shape,rate)</code>	Gamma
<code>rpois(n,lambda)</code>	Poisson
<code>rbinom(n,size,prob)</code>	Binomial
<code>rchisq(n,df)</code>	Chi-square
<code>rweibull(n,shape,scale)</code>	Weibull

- ▶ Εκτός από τα πιο πάνω, μπορούμε να πάρουμε και τις τιμές της συνάρτησης (πυκνότητας ή μάζας) πιθανότητας, τις τιμές της (αθροιστικής) συνάρτησης κατανομής και τα ποσοστημόρια για όλες τις συνηθισμένες κατανομές.
- ▶ Συνάρτηση πιθανότητας: με `d` αντί `r` μπροστά.
- ▶ Συνάρτηση κατανομής: με `p` αντί `r` μπροστά.
- ▶ Ποσοστημόρια: με `q` αντί `r` μπροστά.
- ▶ Επίσης, για τα πιο πάνω, σαν πρώτη καταχώρηση μέσα στην παρένθεση, δεν θα έχουμε αριθμό δειγμάτων που θέλουμε, αλλά την τιμή (ή τιμές) στην οποία θέλουμε να εφαρμόσουμε τις πιο πάνω. Για παράδειγμα,
`dnorm(c(1.2,2.7,0.8),1,.5),`
`pt(c(-3.9,0,1.7),5),`
`qpois(0.05,3).`

- ▶ Έχουμε ήδη δει κάποιες εντολές που παράγουν γραφήματα. Εδώ θα τις δούμε λίγο πιο αναλυτικά, καθώς και πως μπορούμε να αλλάξουμε, να προσθέσουμε ή να αφαιρέσουμε στοιχεία στα γραφήματα.
- ▶ Σαν γενική παρατήρηση, μπορούμε να πούμε ότι οι επιλογές που έχουμε σε κάθε συνάρτηση που παράγει γραφήματα στην R είναι πάρα πολλές. Μπορείτε να παίξετε με αυτά και να δείτε ποιες επιλογές παράγουν το καλύτερο γράφημα στην κάθε περίπτωση.
- ▶ Μπορούμε επίσης, αν έχουμε βρει κάποιο συνδυασμό επιλογών που θέλουμε να χρησιμοποιήσουμε σε πολλά από τα γραφήματά μας, να τον σώσουμε.

- ▶ Αυτό μπορεί να γίνει με την εντολή `par`.
- ▶ Αρχικά, για να αποθηκεύσουμε τις υπάρχουσες ρυθμίσεις (σε περίπτωση που χρειαστεί να τις επαναφέρουμε),
`oldpar <- par(no.readonly=TRUE)`.
- ▶ Μετά, για να δούμε τι επιλογές έχουμε, πληκτρολογούμε `?par`.
- ▶ Αν θέλουμε να δούμε την παρούσα ρύθμιση για μια επιλογή, π.χ. για το χρώμα, πληκτρολογούμε `par("col")`.
Θα μας βγάλει `[1] "black"`.
- ▶ Ας πούμε ότι θέλουμε να το αλλάξουμε σε κόκκινο.
Πληκτρολογούμε: `par(col="red")`.
- ▶ Ομοίως, μπορούμε να αλλάξουμε και άλλες ρυθμίσεις. Όταν κάνουμε τις αλλαγές αυτές, τα γραφήματα που θα γίνουν μετά θα έχουν αυτές τις ρυθμίσεις.
- ▶ Αν θέλουμε να επαναφέρουμε τις προηγούμενες ρυθμίσεις:
`par(oldpar)`.

- ▶ Ας δούμε πρώτα την εντολή `plot`. Από τη σελίδα βοήθειας, μπορείτε να δείτε τις επιλογές.
- ▶ Επιλογές περιλαμβάνουν: τίτλο της γραφικής, ονομασίες και τιμές των δύο αξόνων, το είδος της γραμμής, το χρώμα της γραμμής και των αξόνων κ.α.
- ▶ Ένα άλλο σημαντικό/ενδιαφέρον στοιχείο της εντολής αυτής είναι ότι μπορεί να εφαρμοστεί είτε σε ένα διάνυσμα, είτε σε δύο.
- ▶ Έστω π.χ. `a=c(1,3,5,8)` και `b=c(10,12,15,18)`. Στην πρώτη περίπτωση, για `plot(a)`, έχουμε το δείκτη της παρατήρησης (δηλαδή 1,2,3,4) στον άξονα των X (τον οριζόντιο άξονα) και το a στον άξονα των Y (τον κάθετο άξονα). Στη δεύτερη περίπτωση, `plot(a,b)` έχουμε το a στον άξονα των X και το b στον άξονα των Y .

Με επιπλέον εντολές, μπορούμε επίσης να προσθέσουμε γραμμές, τόξα, κείμενο κ.α. στο γράφημα. Μπορούμε επίσης να προσθέσουμε ή να αλλάξουμε κάποια στοιχεία τα οποία θα μπορούσαμε να ορίσουμε μέσα στην εντολή `plot`.

- ▶ Για γραμμές: με την εντολή `lines`, π.χ. `lines(a,b,lty=24,col="blue")`.
- ▶ Για σημεία: με την εντολή `points`, π.χ. `points(a,b^2,pch=24)`.
- ▶ Προσοχή: από την εντολή `plot` πιθανόν να έχουν δημιουργηθεί ήδη τα σημεία ή/και γραμμή, και με τις δύο τελευταίες εντολές, τα καινούρια σημεία και οι γραμμές μπορεί να μην ξεχωρίζουν από αυτά που υπάρχουν ήδη.
- ▶ Μπορούμε επίσης να προσθέσουμε ένα κουτί με το υπόμνημα του κάθε στοιχείου (legend). Αυτό γίνεται με την εντολή `legend`. Π.χ. αν έχουμε κάνει `plot(a,b,pch="+")` και `plot(a,b^2,pch="*")`, μπορούμε να γράψουμε `legend(0.4,3,c("x2", "x"),pch=c("+", "*"))` (οι δύο πρώτοι αριθμοί ορίζουν τις συντεταγμένες που θα τοποθετηθεί το υπόμνημα).

- ▶ Για την ονομασία (και το χρώμα, το πάχος της γραμματοσειράς κτλ) των αξόνων: με `mtext`. Για παράδειγμα, `mtext("some text",3,col="blue")`.
- ▶ Για να προσθέσουμε γραμμή μέσα στο γράφημα: με `abline`. Π.χ. `abline(h=3)` για οριζόντια γραμμή στο $y = 3$ και `abline(v=3)` για κάθετη γραμμή στο $x = 3$. Επίσης, αν θέλουμε να προσθέσουμε μια ευθεία $y = a + bx$, π.χ. $y = 2 - 3x$, `abline(a=2,b=-3)`.
- ▶ Αν θέλουμε να προσθέσουμε την ευθεία γραμμικής παλινδρόμησης: με `abline(lm(a~ b))`.
- ▶ Για να προσθέσουμε κείμενο μέσα στο γράφημα: με `text`. Π.χ. `text(4,11,labels="whatever")`.
- ▶ Για να προσθέσουμε τόξο μέσα στο γράφημα: με `arrows`. Π.χ. `arrows(-0.4,1.1,2,3,code=1)`. Τα πρώτα δύο νούμερα είναι οι συντεταγμένες της μύτης του τόξου, τα επόμενα δύο είναι οι συντεταγμένες της αρχής του τόξου.

- ▶ Τα πιο πάνω (άξονες, τόξα, κείμενο, γραμμές κτλ.) ισχύουν και για τα άλλα γραφήματα, π.χ. για τα ιστογράμματα και τα θηκογράμματα.
- ▶ Οι εντολές για τα γραφήματα αυτά και οι επιλογές που έχουμε για αυτά είναι, φυσικά, διαφορετικές από τις εντολές για το απλό `plot` γράφημα. Μπορούμε, όμως, με τις επιπλέον εντολές που είδαμε μόλις τώρα, να αλλάξουμε και να προσθέσουμε αντικείμενα στα γραφήματά μας.
- ▶ Πειραματιστείτε!

Τα γραφήματα μπορούν να σωθούν σε πολλές διαφορετικές μορφές. Αυτό μπορεί να γίνει με δύο τρόπους:

- ▶ Όταν είμαστε στο παράθυρο του γραφήματος, πάμε File— >Save as
- ▶ Με την εντολή `dev.print`. Για παράδειγμα:
`dev.print(device=pdf, file="myFile.pdf")` (για pdf) ή
`dev.print(file="myFile.ps",horizontal=FALSE)` (για ps).

Προσοχή στο που σώζετε τα αρχεία!

- ▶ Μπορούμε να έχουμε ένα αρχείο με 2 ή περισσότερα γραφήματα, ταξινομημένα σε γραμμές και στήλες. Η εντολή είναι `par(mfrow())`. Για παράδειγμα, `par(mfrow=c(2,3))` θα δημιουργήσει θέσεις για 6 γραφήματα, σε διάταξη 2x3.
- ▶ Τα γραφήματα εισάγονται στη διάταξη αυτή, συμπληρώνοντας πρώτα τις σειρές (δηλαδή, το 1ο γράφημα που θα εκτελέσουμε μετά την εντολή αυτή μπαίνει πάνε αριστερά, το 2ο πάνω στη μέση, το 3ο πάνω δεξιά, το 4ο κάτω αριστερά κ.ο.κ.).
- ▶ Αν θέλουμε να συμπληρώνονται πρώτα οι σειρές: με `par(mfcol=c(2,3))`.

ΣΥΝΑΡΤΗΣΕΙΣ (functions)

- ▶ Οι συναρτήσεις είναι αντικείμενα στην R τα οποία εκτελούν συγκεκριμένες εντολές.
- ▶ Πολλές εντολές που έχουμε δει ήδη είναι συναρτήσεις. Για παράδειγμα, οι εντολές `sum`, `var`, `hist`, `rnorm`.
- ▶ Συναρτήσεις υπάρχουν επίσης και σε πολλά πακέτα.
- ▶ Εκτός από αυτές τις έτοιμες συναρτήσεις, μπορούμε εύκολα να γράψουμε τις δικές μας συναρτήσεις.
- ▶ Αυτά μπορούν να γραφτούν είτε στο R Console είτε σε ένα R script. Προτιμήστε το 2ο, λόγω του ότι όταν γράφουμε κάποια συνάρτηση, και ειδικά για τις κάπως σύνθετες συναρτήσεις, πολύ πιθανόν να κάνουμε κάποια λάθη ή απλά να θέλουμε να δοκιμάσουμε διάφορα πράγματα. Σε αυτές τις περιπτώσεις το script είναι πολύ πιο βολικό.

ΔΟΜΗ ΜΙΑΣ ΣΥΝΑΡΤΗΣΗΣ

Μια συνάρτηση πρέπει να είναι της μορφής

```
xx < -function(arg) {  
  command 1  
  command 2  
  .....  
  last command  
  return(yy)  
}
```

όπου `xx` είναι το όνομα της συνάρτησης, `arg` είναι οι παράμετροι εισαγωγής (`inputs`) που θα δίνουμε για να τρέξει η συνάρτηση και `yy` είναι το αποτέλεσμα που θα μας δώσει η συνάρτηση (`output`). Οι εντολές `command 1` κτλ είναι οι εντολές που εκτελούνται μέσα στη συνάρτηση.

Μπορούμε να έχουμε πολλά `inputs` αλλά μόνο ένα `output`. Άρα, αν θέλουμε να σώσουμε περισσότερο από ένα αντικείμενο, μπορούμε να τα γράψουμε σε μια λίστα και να πάρουμε αυτή σαν το `output`.

Επίσης σημαντική παρατήρηση είναι ότι η εντολή `return` πρέπει να είναι η τελευταία μέσα στην συνάρτηση. Οτιδήποτε είναι μετά από αυτή παραβλέπεται.

- ▶ Όπως είπαμε, καλύτερα να γράφουμε τη συνάρτηση σε ένα script. Είναι επίσης καλή ιδέα να προσθέσουμε σημειώσεις (με το `#`) για να θυμόμαστε τι κάνει η κάθε εντολή.
- ▶ Την τρέχουμε στο R Console και αν δεν υπάρχει κάποιο λάθος, αυτή αποθηκεύεται σαν αντικείμενο.
- ▶ Αν πληκτρολογήσουμε το όνομα της συνάρτησης στο περιβάλλον εργασίας χωρίς inputs, θα μας γράψει όλη τη συνάρτηση.
- ▶ Αν θέλουμε να τρέξουμε την εντολή για κάποιους αριθμούς, πληκτρολογούμε το όνομα της συνάρτησης με τα inputs σε παρένθεση.
Για παράδειγμα, `cubic(4)`. Αυτό θα δώσει
[1] 56.

- ▶ Αφού η συνάρτηση είναι ένα αντικείμενο, μπορούμε να τη διαχειριστούμε σαν τέτοια. Για παράδειγμα, μπορούμε να δημιουργήσουμε ένα νέο αντικείμενο βάσει αυτής, π.χ.
`t <- cubic(4).`
- ▶ Επίσης, σαν αντικείμενο εισόδου μπορούμε να δώσουμε ένα διάνυσμα. Η συνάρτηση τότε θα εκτελεστεί για κάθε στοιχείο του διανύσματος και το αποτέλεσμα θα είναι πάλι ένα διάνυσμα. Για παράδειγμα, `x <- 1:4` και μετά `cubic(x)` θα δώσει `[1] 1 8 27 64.`
- ▶ Άλλο παράδειγμα συνάρτησης είναι μέσα στο αρχείο `sample_function1.r` (και προσέξτε ότι δεν είναι απαραίτητο το όνομα του αρχείου να είναι το ίδιο με το όνομα της συνάρτησης).

ΠΑΡΑΔΕΙΓΜΑ ΜΕ 2 ΠΑΡΑΜΕΤΡΟΥΣ ΕΙΣΟΔΟΥ

Υπάρχει ένα παράδειγμα μέσα στο αρχείο `sample_function2.r`.
Εννοείται ότι η σειρά που γράφουμε τα inputs παίζει ρόλο (δοκιμάστε `sample_function2.r(1,2)` και `sample_function2.r(2,1)`).
Επίσης, και εδώ μπορούμε να χρησιμοποιήσουμε διανύσματα. Θέλει όμως προσοχή: αν τα διανύσματα είναι του ίδιου μήκους ή αν το ένα από τα δύο είναι αριθμός, τότε είναι εντάξει. Για δύο διανύσματα άνισου μήκους, δε είναι ξεκάθαρο τι θα γίνει.

- ▶ Αυτά είναι αντικείμενα τα οποία είναι είτε ορθά (TRUE) είτε λάθος (FALSE).
- ▶ Συμβολίζονται στην R είτε με TRUE / FALSE είτε με τις συντομεύσεις T / F.
- ▶ Συνήθως τα παίρνουμε αυτά από τον έλεγχο μιας σχέσης, η οποία θα είναι είτε ορθή είτε λανθασμένη. Για παράδειγμα, `x < 5` ή `x == 2`. Αν είχαμε θέσει `x <- 3` πιο πριν, η 1η θα δώσει TRUE και η 2η FALSE.
- ▶ Αν δοκιμάσουμε οποιαδήποτε αριθμητική πράξη με λογικά αντικείμενα, η R τα μετατρέπει σε 1 (τα TRUE) και 0 (τα FALSE).
- ▶ Αφού αυτά είναι αντικείμενα, μπορούμε επίσης να τα χρησιμοποιήσουμε σαν τέτοια, π.χ. `y <- x == 2`.

- ▶ Οι λογικοί έλεγχοι είναι της μορφής: $<$, $>$, $<=$, $>=$, $==$, $!=$.
- ▶ Το $!$ μπροστά από ένα έλεγχο δηλώνει την άρνηση του ελέγχου.
- ▶ Μπορούμε επίσης να συνδυάσουμε ελέγχους με
 $\&\&$ (ΚΑΙ),
 $||$ (Ή).

- ▶ Τα είδη ελέγχων και οι συνδυασμοί που αναφέραμε πιο πάνω μπορούν να χρησιμοποιηθούν και για διανύσματα.
- ▶ Μπορούμε να πάρουμε επίσης διανύσματα, όπου το κάθε στοιχείο είναι T ή F.
- ▶ Για παράδειγμα, `x=1:5` και μετά `y <- x<3`.

Μια άλλη χρήσιμη χρήση αυτών είναι σαν δείκτες. Για παράδειγμα,

```
> x <- c(6,5,2,4,4,6,4)
```

```
> y <- c(5,3,4,2,6,5,4)
```

```
> xeq4 <- x == 4
```

#which of x equals 4?

```
> xeq4
```

```
[1] FALSE FALSE FALSE TRUE TRUE FALSE TRUE
```

```
> y[xeq4]
```

#which of y corresponds
to x==4?

```
[1] 2 6 4
```

Άλλο παράδειγμα:

```
> x[x > 4]
```

```
[1] 6 5 6
```

- ▶ Μπορούμε να χρησιμοποιήσουμε αυτά τα λογικά αντικείμενα στη δημιουργία συναρτήσεων, έτσι ώστε να ελέγχουμε πότε κάποιες εντολές θα εκτελούνται.
- ▶ Οι εντολές που χρησιμοποιούνται σε συνδυασμό με τα λογικά αντικείμενα είναι οι `if` και `while`.
- ▶ Μια άλλη εντολή που χρησιμοποιείται, χωρίς όμως τη χρήση λογικών αντικειμένων είναι η `for`.

Η ΕΝΤΟΛΗ for

Η εντολή αυτή μας λέει βασικά πόσες φορές θα εκτελεστούν κάποιες εντολές. Η γενική μορφή είναι:

```
for (i in 1:n) {  
    command 1  
    command 2  
    ....  
last command  
}
```

όπου n είναι ο αριθμός επαναλήψεων, και οι εντολές *command1* κτλ είναι αυτές που θα εκτελεστούν. Στην πράξη, οι εντολές αυτές συνήθως θα αλλάζουν συναρτήσει του i .

Επίσης, η αρίθμηση μπορεί να ξεκινάει και να τελειώνει σε οποιοδήποτε αριθμό. Στην πράξη, όμως, το να αρχίζουμε από το 1 είναι συνήθως το πιο λογικό.

Η εντολή αυτή μας λέει βασικά σε ποιές περιπτώσεις να εκτελεστούν κάποιες εντολές, ανάλογα με το αν ένα λογικό αντικείμενο είναι *TRUE* ή *FALSE*.

Οι εντολές εκτελούνται μόνο στην περίπτωση που έχουμε *TRUE*.

Η γενική μορφή είναι:

```
if (A) {  
    command 1  
    ....  
    last command  
}
```

όπου *A* είναι το λογικό αντικείμενο. Οι εντολές είναι αυτές που θα εκτελεστούν αν και μόνο αν *A = TRUE*.

Η ΕΝΤΟΛΗ `if` ΜΕ ΕΝΑΛΛΑΚΤΙΚΕΣ ΕΝΤΟΛΕΣ

Αν θέλουμε να εκτελούνται κάποιες άλλες εντολές αν έχουμε *FALSE*, τότε:

```
if (A) {  
    command 1  
    ....  
last command  
} else {  
    command 1'  
    ....  
last command'  
}
```

όπου *A* είναι το λογικό αντικείμενο. Οι πρώτες εντολές είναι αυτές που θα εκτελεστούν αν και μόνο αν *A = TRUE*. Αν *A = FALSE*, εκτελούνται οι εντολές μετά το `else`.

Η ΕΝΤΟΛΗ if ΜΕ ΠΟΛΛΕΣ ΕΝΑΛΛΑΚΤΙΚΕΣ ΕΝΤΟΛΕΣ

Αν θέλουμε να έχουμε περισσότερους από 1 έλεγχο, μπορούμε να χρησιμοποιήσουμε (1 ή περισσότερα) else if:

```
if (A) {  
    command 1  
    ....  
    last command  
} else if (B) {  
    command 1'  
    ....  
    last command'  
} else {  
    command 1''  
    ....  
    last command''  
}
```


Η ΕΝΤΟΛΗ while

Η εντολή αυτή μας λέει βασικά πότε εκτελούνται κάποιες εντολές. Οι εντολές εκτελούνται όσο η συνθήκη ικανοποιείται. Όταν πάψει να ικανοποιείται, εξερχόμαστε από το κομμάτι αυτό του κώδικα. Η γενική μορφή είναι:

```
while (A) {  
    command 1  
    ....  
    last command  
}
```

όπου A είναι το λογικό αντικείμενο. Οι εντολές εκτελούνται για όσο η συνθήκη $A = \text{TRUE}$. Αν $A = \text{FALSE}$, βγαίνουμε από το κομμάτι αυτό.