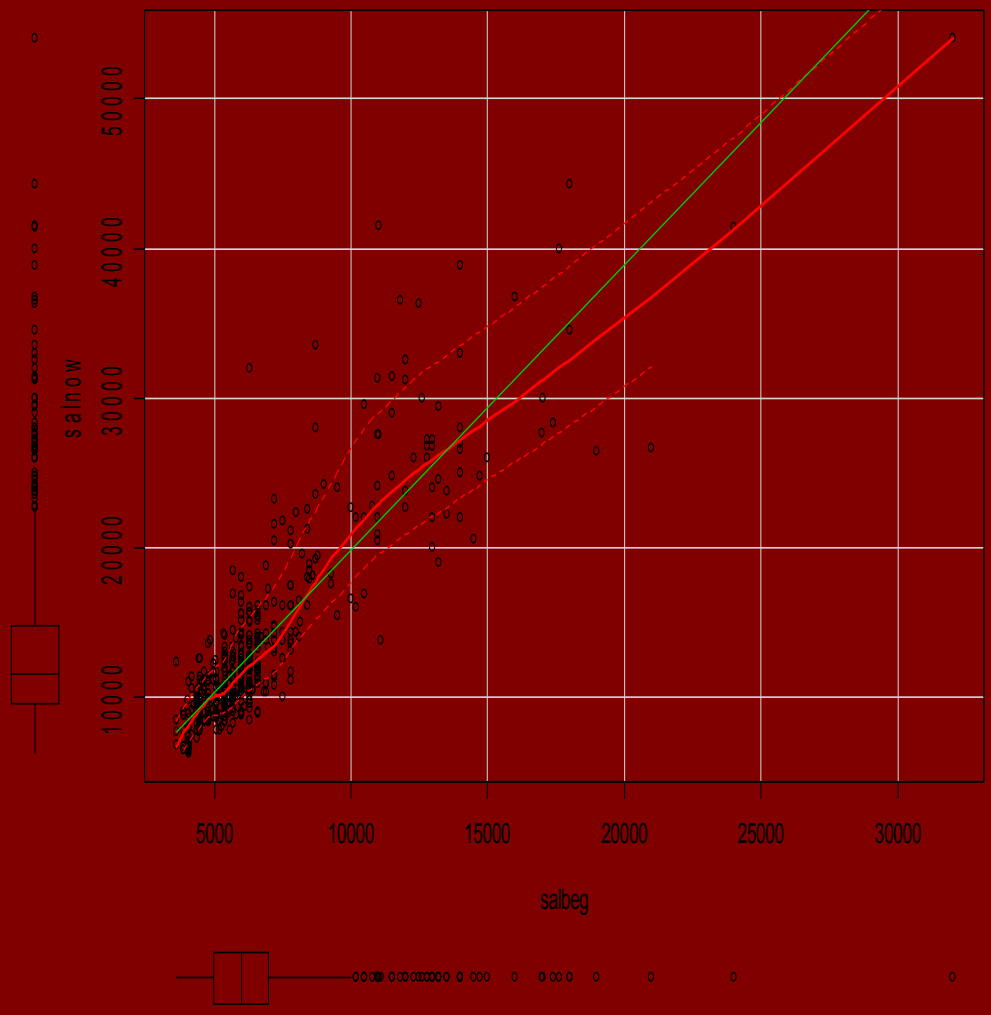
Ελληνικά Ακαδημαϊκά Ηλεκτρονικά Συγγράμματα και Βοηθήματα
www.kallipos.gr
HEALLINK
Σύνδεσμος Ελληνικών Ακαδημαϊκών Βιβλιοθηκών

Εισαγωγή στον Προγραμματισμό και στη Στατιστική Ανάλυση με R



Δημήτρης Καρλής
Ιωάννης Ντζούφρας
Τμήμα Στατιστικής,
Οικονομικό Πανεπιστήμιο Αθηνών

ΙΩΑΝΝΗΣ ΝΤΖΟΥΦΡΑΣ
Καθηγητής Στατιστικής Οικονομικού Πανεπιστημίου Αθηνών

ΔΗΜΗΤΡΗΣ ΚΑΡΛΗΣ
Καθηγητής Στατιστικής Οικονομικού Πανεπιστημίου Αθηνών

Εισαγωγή στον Προγραμματισμό και στη Στατιστική Ανάλυση με R

Εισαγωγή στον Προγραμματισμό και στη Στατιστική Ανάλυση με R

Συγγραφή

Ιωάννης Ντζούφρας

Δημήτρης Καρλής

Κριτικός αναγνώστη

Δημήτρης Φουσκάκης

Συντελεστές έκδοσης

Γλωσσική Επιμέλεια: Κατερίνα Μαντζούνη

Γραφιστική Επιμέλεια: Ιωάννης Ντζούφρας

Τεχνική Επεξεργασία: Ιωάννης Ντζούφρας

ISBN: 978-960-603-449-7

ΣΥΝΔΕΣΜΟΣ ΕΛΛΗΝΙΚΩΝ ΑΚΑΔΗΜΑΪΚΩΝ ΒΙΒΛΙΟΘΗΚΩΝ

Εθνικό Μετσόβιο Πολυτεχνείο

Ηρώων Πολυτεχνείου 9, 15780 Ζωγράφου

www.kallipos.gr

Με αγάπη στη γυναίκα μου Ζωή
και στα παιδιά μου, Κωστή και Ζένια
Δ. Καρλής

Με αγάπη στη γυναίκα μου Ιωάννα
και στις κόρες μου, Νάσια και Μελίνα
Ι. Ντζούφρας

Πρόλογος

Το παρόν διδακτικό σύγγραμμα απευθύνεται σε φοιτητές που σπουδάζουν Στατιστική ή ποσοτικές επιστήμες και επιθυμούν να μάθουν προγραμματισμό, αλλά και βασική ανάλυση δεδομένων με τη χρήση της γλώσσας ανοικτού κώδικα R. Το παρόν εγχειρίδιο επικεντρώνεται περισσότερο στην επεξήγηση βασικών αρχών προγραμματισμού, παρά στην ανάλυση δεδομένων, αν και οι βασικές εντολές ανάλυσης δεδομένων περιγράφονται στο Κεφάλαιο 8. Στο μέλλον οι συγγραφείς σκοπεύουν να συγγράψουν ένα πιο αναλυτικό βιβλίο, το οποίο θα στοχεύει στην ανάλυση δεδομένων με πρακτικά παραδείγματα και περιπτώσεις μελέτης.

Το σύγγραμμα που κρατάτε στα χέρια σας έχει ως σκοπό τόσο την εξοικείωση με την R, όσο και με βασικές ιδέες προγραμματισμού που συναντάει κανείς σε οποιαδήποτε γλώσσα προγραμματισμού ή σε άλλα στατιστικά πακέτα, που επιτρέπουν τη χρήση περιβάλλοντος εντολών. ΄Εχει καταβληθεί μεγάλη προσπάθεια ώστε τα παραδείγματα να είναι σύμφωνα με τις γνώσεις των φοιτητών πρώτου έτους Στατιστικής ή το πολύ δεύτερου έτους σε άλλα συναφή τμήματα. Στο τέλος του συγγράμματος θα βρείτε μια μεγάλη σειρά από άλυτες ασκήσεις διαφορετικού επιπέδου δυσκολίας. Κάποιες από αυτές έχουν δοθεί ως εργασίες ή ως θέματα εξέτασης στο μάθημα που διδάσκουμε στο Τμήμα Στατιστικής του Οικονομικού Πανεπιστημίου Αθηνών με τίτλο «Εισαγωγή στον Προγραμματισμό με R». Θα ήταν επιθυμητό, με το τέλος μελέτης αυτού του βιβλίου ή ενός αντίστοιχου πανεπιστημιακού μαθήματος, ο αναγνώστης ή ο φοιτητής να μπορεί να λύσει τις περισσότερες, αν όχι όλες, από αυτές τις ασκήσεις.

Το σύγγραμμα αυτό έχει βασιστεί σε προηγούμενες εκδόσεις πανεπιστημιακών σημειώσεων του προαναφερόμενου μαθήματος. Στην αρχική έκδοση των σημειώσεων σημαντική συνεισφορά είχε και ο Φώτης Σταυρόπουλος τον οποίο θα θέλαμε να ευχαριστήσουμε θερμά για την πολύτιμη βοήθειά του.

Επίσης, θα θέλαμε να ευχαριστήσουμε θερμά τον Άρη Νικολουλόπουλο για τις διορθώσεις που έκανε κατά τη διάρκεια των φροντιστηριακών ασκήσεων κατά τα ακαδημαϊκά έτη 2004-2007, αλλά και την Κατερίνα Μαντζούνη που βοήθησε σημαντικά στην ενημέρωση και στην επεξεργασία των τελικών κειμένων του βιβλίου που έχετε στα χέρια σας.

Αθήνα, Δεκέμβριος 2015

Δημήτρης Καρλής - Ιωάννης Ντζούφρας

ΠΕΡΙΕΧΟΜΕΝΑ

Πίνακας συντομεύσεων-ακρωνύμια

AIC Akaike Information Criterion

ASCII American Standard Code for Information Interchange

BIC Bayesian Information Criterion

Cos Cosine – Συνημίτονο

Exp Exponent – Δηλώνει τη μαθηματική συνάρτηση e^x, όπου e είναι η σταθερά του Euler και βάση του φυσικού λογάριθμου

FDA Food and Drug Administration

Getwd Get Working Direcory

HSD Honest Significant Different

IC Information Criterion

INF Infinite

Max Maximum

Min Minimum

NA Not Available

NaN Not a Number

OLS Ordinary Least Squares

SAS Statistical Analysis System

Sqrt Square root – Δηλώνει τη τετραγωνική ρίζα ενός αριθμού x

ΕΜΠ Εκτιμήτρια Μεγίστης Πιθανοφάνειας

ΕΣΚ Εμπειρική Συνάρτηση Κατανομής

ΚΟΘ Κεντρικό Οριακό Θεώρημα

ΜΚΔ Μέγιστος Κοινός Διαιρέτης

Ευρετήριο αντιστοίχισης ελληνόγλωσσων και ξενόγλωσσων επιστημονικών όρων

Α

Αγνοούμενες Τιμές – Missing Values
Άθροισμα Τετραγώνων Καταλοίπων – Residual Sum of Squares
Ακέραιος – Integer
Ακολουθία – Sequence
Ακραίες Τιμές – Outliers
Ανακύκλωση – Recycling
Ανάλυση Διακύμανσης κατά έναν Παράγοντα – One-way Analysis of Variance Estimator
Αναμενόμενες Τιμές – Expected Values
Ανθεκτικός – Robust
Αντιγραφή και Επικόλληση – Copy-Paste
Αντικατάσταση – Replacement
Αντικείμενο Δεδομένων – Data Object
Αποθήκευση – Save
Αριθμητικά Δεδομένα – Numeric
Αριθμητικού Τύπου Δεδομένα (στην R) – Numeric Mode
Αρχείο Ιστορικού – History
Αρχή της Απλότητας – Parsimony Principle

Β

Βάρη – Weights
Βασικές Εκφράσεις Σύγκρισης – Basic Regular Expressions
Βιβλιοθήκη (της R) – Library
Βοήθεια – Help

Γ

Γραμμή – Row
Γωνιακοί Περιορισμοί – Corner Constraints

Δ

Δεδομένα Εκπαίδευσης – Training Dataset
Δεδομένα Ελέγχου – Test dataset
Δεδομένα Λογικής – Logical Data

Δεδομένα Χαρακτήρων - Character Data
Δείκτες Επιρροής ή Μόχλευσης - Leverages
Διάγραμμα Διασποράς - Scatter Plot
Διάγραμμα Κλίμακας-Θέσης – Scale Location Plot
Διάγραμμα Πλαισίου-Απολήξεων ή Θηκόγραμμα – Boxplot
Διάγραμμα Χρονικής Σειράς – Time Sequence Plot
Διαδρομή – Path
Διάμεσος – Median
Διάνυσμα – Vector
Διατάξιμοι Παράγοντες – Ordinal Factors
Διατάξιμος – Ordinal
Διδιάστατος Πίνακας – Matrix
Διόρθωση Συνέχειας – Continuity Correction
Διπλής Ακρίβειας – Double (Precision)
Διπλής Εισόδου – Two-way

Ε

Εκτεταμένες Εκφράσεις Σύγκρισης – Extended Regular Expression Matching
Εκτίμηση – Estimate
Εκτύπωση – Print
Ελάχιστο – Minimum
΄Ελεγχος Υπόθεσης – Hypothesis Test
Επίπεδο – Level
Επικρατούσα Τιμή (στη στατιστική) – Mode
Ευαισθησία στη Χρήση Κεφαλαίων και Μικρών Γραμμάτων – Case Sensitive
Εύρος – Range

Θ

Θηκόγραμμα ή Διάγραμμα Πλαισίου-Απολήξεων – Boxplot

Ι

Ισοπαλίες – Ties
Ιστόγραμμα – Histogram

Κ

Καθολικές Μεταβλητές – Global Variables
Καλή Προσαρμογή – Goodness of Fit
Κανονική Κατανομή – Normal Distribution
Κατάλογος – Directory
Κατάλογος Εργασίας – Working Directory
Κατάλοιπα – Residuals
Κατηγορικά Διανύσματα – Factors
Κλάση ενός Αντικειμένου (στην R) – Class
Κλιμακωτή Διαδικασία – Stepwise Procedure
Κοινός Εκτιμητής Διακύμανσης – Pooled Variance Estimator

Λ

Λάθη (προγραμματιστικά) – Bugs
Λίστα – List
Λογικά δεδομένα – Logical Data
Λογικές τιμές – Logical Values

M

Μεγαλύτερο – Bigger
Μέγιστο – Maximum
Μέθοδος Ελαχίστων Τετραγώνων – Ordinary Least Square Method
Μέση Τιμή – Mean
Μέσος Όρος – Average
Μεταβλητή Απόκρισης – Response Variable
Μη Διαθέσιμες Τιμές – Not Available
Μηδενικό – Null
Μήκος – Length
Μιγαδικοί Αριθμοί – Complex
Μικρότερο – Smaller
Μοτίβο – Pattern

N

Νομική Ευθύνη – Liability

O

Ομάδες – Groups
Ονομαστικός – Nomimal
Ονόματα – Names
Οπισθοδρομική Απαλοιφή – Backward Elimination

Π

Πακέτα – Packages
Παράγοντες – Factors
Παραγοντικό – Factorial
Παράλληλη Επεξεργασία των Δεδομένων – Parallel Computing
Παρέκταση – Extrapolation
Περικομμένος Μέσος – Trimmed Mean
Πιθανότητες ανά Γραμμή – Row Probability
Πίνακας – Matrix
Πίνακας Συνάφειας – Contigency Table
Πλαίσιο Δεδομένων – Data Frame
Πολυδιάστατος Πίνακας – Array
Ποινικοποιημένη Πιθανοφάνεια – Penalized Likelihood
Ποσοστό Λανθασμένων Απορρίψεων – False Discovery Rate
Πραγματικός Αριθμός – Real
Πρόγραμμα ή σειρά εντολών – Script
Προγραμματιστικά Λάθη – Bugs
Προειδοποίηση – Warning
Προκαθορισμένος – Default
Προσαρμοσμένες Τιμές – Fitted Values
Πρόσθια Διαδικασία – Forward Procedure
Πρώτος – First

Σ

Σταθερά – Intercept
Στήλη – Column
Στοιχείο με Στοιχείο – Elementwise
Συνάρτηση – Function
Συνδυάζω – Combine
Συντάκτης Δεδομένων – Data Editor
Συντελεστής Προσδιορισμού – Coefficient of Determination

Τ

Τάξη – Rank
Τοπικές Μεταβλητές – Local Variables
Τυπικό Σφάλμα – Standard Error
Τυποποιημένα κατάλοιπα κατά Student – (Externally) Studentized Residuals
Τυποποιημένο κατά Student Εύρος – Studentized Range
Τύπος ενός Αντικειμένου (στην R) – Mode
Τυχαίος – Random

Υ

Υπερχείλιση – Overflow
Υπολογιστική Στατιστική – Computational Statistics
Υποχείλιση – Underflow

Φ

Φάκελος – Folder
Φανταστικός – Imaginary
Φύλλο Εργασίας – Worksheet

Χ

Χαρακτήρας – Character
Χώρος Εργασίας – Workspace

Ψ

Ψευδομεταβλητές – Dummies

ΚΕΦΑΛΑΙΟ 1

Εισαγωγή

1.1 Εισαγωγή

Η στατιστική επιστήμη έχει αναπτυχθεί ραγδαία τα τελευταία χρόνια, κυρίως λόγω της ευρείας διάδοσης των υπολογιστών και της διαθεσιμότητάς τους για περίπλοκους υπολογισμούς που, πριν από τη χρήση των υπολογιστών, φάνταζαν σχεδόν αδύνατοι. Ακόμα και ο υπολογισμός μιας μέσης τιμής χωρίς τη χρήση υπολογιστή μοιάζει μια βαρετή και κουραστική ρουτίνα.

Το πρώτο μισό του $20^{ου}$ αιώνα, η στατιστική άνθισε κυρίως μέσα από μια σειρά θεωρητικών αποτελεσμάτων, τα οποία όμως βρήκαν πολύ λίγες εφαρμογές, αφού πολλά από αυτά, για να εφαρμοσθούν στην πράξη, απαιτούσαν εξαιρετικά χρονοβόρες και επίπονες πράξεις. Ακόμα και η αντιστροφή ενός πίνακα μέτριων διαστάσεων ήταν μια επίπονη διαδικασία, που απαιτούσε αρκετή οξυδέρκεια και εξειδίκευση.

Δεν είναι υπερβολικό να πούμε πως η στατιστική και η ανάπτυξη των υπολογιστών πήγαν χέρι-χέρι. Μερικές σημαντικές ανακαλύψεις και κάποιοι από τους πρώτους υπολογιστές αναπτύχθηκαν, για να βοηθήσουν και να επιλύσουν στατιστικής φύσης προβλήματα, όπως για παράδειγμα να πινακοποιηθούν και να αναλυθούν τα αποτελέσματα της απογραφής.

Ιστορικά οι πρώτοι υπολογιστές ήταν φτιαγμένοι για να εκτελούν ένα ή λίγα πράγματα με σκοπό να επιλύουν πολύ συγκεκριμένα προβλήματα. Στη συνέχεια, αναπτύχθηκαν οι γλώσσες προγραμματισμού, οι οποίες επέτρεπαν στον χρήστη να γράφει τα δικά του προγράμματα και, επομένως να χρησιμοποιεί τον υπολογιστή, για να λύσει πολλά διαφορετικά προβλήματα, πάντα βέβαια μέσα στην εμβέλεια του υπολογιστή. Για παράδειγμα, οι πρώτοι υπολογιστές έλυναν αποκλειστικά αριθμητικά προβλήματα και δεν μπορούσαν να χειριστούν χαρακτήρες (γράμματα).

Δεν πρέπει ποτέ να ξεχνάμε πως η βασική αρχή των υπολογιστών βασίζεται στην αντίληψη δύο καταστάσεων 0 και 1, δηλαδή περνάει ηλεκτρικό ρεύμα ή όχι. Αυτό σημαίνει πως ο προγραμματισμός αφορούσε στο παρελθόν μια σειρά από 0 και 1 εντολές και ότι ο κατάλληλος συνδυασμός τους, έδινε στον υπολογιστή τις εντολές για το τι θα κάνει. Προφανώς κάτι τέτοιο αφενός μοιάζει ιδιαίτερα πολύπλοκο και αφετέρου περιοριστικό. Στη συνέχεια, οι εντολές σε δυαδικό κώδικα

μεταφράστηκαν σε εντολές, απλές στην αρχή, όμως κατανοητές σε φυσική γλώσσα, Στην ουσία αυτές οι εντολές μεταφράζονταν στις αντίστοιχες δυαδικές. Είναι σαφές ότι οι εντολές σε φυσική γλώσσα είναι πιο κατανοητές και, συνεπώς, τα προγράμματα πιο εύκολα να γραφτούν και να ελεγχθούν. Σ' αυτήν τη βάση σιγά σιγά αναπτύχθηκαν γλώσσες προγραμματισμού στηριζόμενες πολλές φορές σε συγκεκριμένες ανάγκες. Είναι σημαντικό να θυμάται κανείς πως υπάρχουν ποικίλες γλώσσες προγραμματισμού και ότι κάποιες είναι πιο κατάλληλες για συγκεκριμένης φύσης προβλήματα.

Η ανάπτυξη γλωσσών προγραμματισμού οδήγησε σταδιακά στην ανάπτυξη εργαλείων για τη συγγραφή πιο σύνθετων προγραμμάτων και αυτή με τη σειρά της επέφερε την ανάπτυξη εξειδικευμένων προγραμμάτων που ειδικεύονταν σε συγκεκριμένα προβλήματα. Μ' αυτήν τη λογική και δεδομένης της στενής σχέσης της στατιστικής με τους υπολογιστές εμφανίστηκαν τα πρώτα στατιστικά πακέτα. Θα πρέπει να αναφερθεί πως στη δεκαετία του 1940 και λόγω των πολεμικών αναγκών ήδη είχαν αναπτυχθεί προγράμματα στατιστικής προσομοίωσης.

Τα πρώτα στατιστικά πακέτα αφορούσαν συγκεκριμένες στατιστικές τεχνικές, π.χ. πινακοποιήσεις, ελέγχους υποθέσεων κλπ. Ήταν γραμμένα σε μια γλώσσα προγραμματισμού και προσέφεραν κάποιες δυνατότητες συγγραφής και προγραμματισμού, κυρίως γιατί απευθύνονταν σε εξειδικευμένο προσωπικό. Στη συνέχεια, τα στατιστικά πακέτα έγιναν ολοένα πιο φιλικά προς τον απλό χρήστη χωρίς ιδιαίτερες γνώσεις και μετά την ευρεία είσοδο των παραθυρικών εφαρμογών οι προγραμματιστικές τους ικανότητες σχεδόν παραμελήθηκαν. Σήμερα, πολλά από τα στατιστικά πακέτα προσφέρουν τη δυνατότητα προγραμματισμού με τη χρήση ομάδων εντολών. Κάτι τέτοιο είναι πολύ χρήσιμο στις πραγματικές εφαρμογές, αφού για εκπαιδευτικούς σκοπούς, καθώς μπορεί ο στατιστικός να αναπτύξει και να προσαρμόσει τεχνικές στα δικά του μέτρα και να έχει τον πλήρη έλεγχο στο χειρισμό των δεδομένων.

Η R είναι ένα πακέτο που προσφέρει σημαντικές δυνατότητες προγραμματισμού και συγγραφής προγραμμάτων για την επίλυση πολύπλοκων στατιστικών προβλημάτων. Έχει το τεράστιο πλεονέκτημα να ενσωματώνει μια σειρά από τεχνικές, που είναι ιδιαίτερα σύγχρονες και, επομένως, προσφέρει τα εργαλεία για στατιστικές αναλύσεις με τη χρήση των πιο σύγχρονων μεθόδων.

Στις σημειώσεις αυτές θα ασχοληθούμε με τον προγραμματισμό στην R. Θα προσπαθήσουμε να δώσουμε μερικές συμβουλές - προτροπές σε θέματα προγραμματισμού που ισχύουν πολύ γενικότερα και όχι μόνο για την R.

1.2 Σύντομο ιστορικό της R

Πρόγονος της γλώσσας R υπήρξε η γλώσσα προγραμματισμού S, η οποία αναπτύχθηκε από τον John Chambers στα εργαστήρια της Bell Labs. Στο τέλος της δεκαετίας του 1980 και αρχές του 1990,η S αποτελούσε μια γλώσσα υψηλού επιπέδου που προσέφερε εντυπωσιακές δυνατότητες ανάλυσης δεδομένων και γραφικών για τα χρόνια εκείνα. Αν και στην πραγματικότητα η R βασίστηκε στην S, παρόλα αυτά έχουν σημαντικές διαφορές, κυρίως σε σχέση με τον τρόπο που διαχειρίζονται τη διαθέσιμη μνήμη του υπολογιστή.

Η R αναπτύχθηκε από τους Ross Ihaka και Robert Gentleman στο Πανεπιστήμιο του Auckland

στη Νέα Ζηλανδία στις αρχές της δεκαετίας του 1990 (Ihaka & Gentlman, 1996). Για την ακρίβεια, ο δεύτερος ήταν επισκέπτης στο Auckland με εκπαιδευτική άδεια. Βασίστηκαν στη γλώσσα S και το όνομα που έδωσαν στην καινούρια γλώσσα ήταν αφενός ένα λογοπαίγνιο με τη γλώσσα S και αφετέρου τα αρχικά των ονομάτων τους[1]. Αργότερα, η γλώσσα S εξελίχτηκε στο εμπορικό πακέτο R.

Η πρώτη έκδοση της R εμφανίστηκε το 1994 και η φιλοδοξία ήταν πολύ απλή: να χρησιμοποιηθεί για ένα εισαγωγικό μάθημα στατιστικής. Οι δημιουργοί της, πεπεισμένοι για την περιορισμένη εμπορική δυνατότητα του νέου πακέτου, αποφάσισαν να το διαθέσουν δωρεάν σε όποιον ενδιαφερόταν. Έφτιαξαν μάλιστα και μια λίστα με χρήστες για να ανταλλάσσουν απόψεις και προγράμματα, αλλά και να διορθώνουν προγραμματιστικά λάθη (bugs). Πολύ σύντομα, κατάλαβαν ότι είχαν ξεπεράσει αυτό που πίστευαν. Κάθε μέρα λάμβαναν πολλά μηνύματα και ήταν πια ξεκάθαρο ότι αφενός οι χρήστες ήταν πολύ περισσότεροι και αφετέρου ότι η ανάπτυξη της γλώσσας R είχε φτάσει σε ένα κρίσιμο σημείο. Μάλιστα η λίστα με τα emails έσπασε στα δύο, στη λίστα για βοήθεια και στη λίστα για περαιτέρω ανάπτυξη. Αυτό συνέβη γιατί πολλοί χρήστες έστελναν τους δικούς τους κώδικες ώστε να ενσωματωθούν. Το 1996 δημιουργήθηκε στη Βιέννη ένα αποθετήριο προγραμμάτων της R από τους χρήστες.

Ουσιαστικά είχε αρχίσει η νεα εποχή. Οι δυο βασικοί προγραμματιστές άφησαν πια ελεύθερο το πεδίο σε όποιον άλλο χρήστη ήθελε να δημιουργεί τις δικές του ρουτίνες σε R. Το Φεβρουάριο του 2000 θεωρήθηκε ότι είχε ωριμάσει η πρόοδος και ότι υπήρχε μια αρκετά σταθερή έκδοση της R. Έτσι, η πρώτη έκδοση 1.0 ήταν διαθέσιμη σε όποιον ήθελε να την κατεβάσει. Τώρα πια η ομάδα που ασχολιόταν με την ανάπτυξη της R αποτελούνταν από 20 προγραμματιστές που δούλευαν για την R, ενώ συγχρόνως, πολλοί ακαδημαϊκοί ερευνητές από την περιοχή της Υπολογιστικής στατιστικής (computational statistics) συνεισφέρουν σε αυτή.

Στα χρόνια που θα ακολουθήσουν, η ανάπτυξη θα είναι ακόμα πιο ραγδαία. Η R θα αποτελέσει πολύτιμο εργαλείο διδασκαλίας σε όλες τις άκρες του κόσμου, με αποτέλεσμα οι χρήστες να αυξάνονται. Συγχρόνως, η δυνατότητα που παρέχει είναι η ενσωμάτωση καινούριων βιβλιοθηκών με εξαιρετική ευκολία, αλλά κυρίως, η δυνατότητα των χρηστών να φτιάχνουν και να αποθέτουν βιβλιοθήκες, έχει οδηγήσει σε μια τεράστια ανάπτυξη.

Όλα αυτά τα χρόνια, η R παραμένει γλώσσα ελευθέρου κώδικα και είναι δωρεάν. Αυτό δημιουργεί κάποια προβλήματα και περιορισμούς στη χρήση της για εμπορικούς σκοπούς (π.χ. από εταιρείες), καθώς η νομική ευθύνη (liability) δεν είναι ξεκάθαρο σε ποιον ανήκει. Παρόλα αυτά, τα τελευταία χρόνια υπάρχουν εταιρείες που προσφέρουν πιστοποίηση και άρα αναλαμβάνουν αυτές τις ευθύνες, οδηγώντας σε ολοένα μεγαλύτερη χρήση. Επίσης, μεγάλοι οργανισμοί (π.χ. Food and Drug Administration, FDA, στις ΗΠΑ) αποδέχονται την R σαν εργαλείο για τη στατιστική ανάλυση χωρίς περιορισμούς.

Θα πρέπει επίσης να τονιστεί ότι, αν και η όλη ανάπτυξη της R έχει σημειωθεί σαν μια γλώσσα εντολών, δηλαδή ο χρήστης δεν χρησιμοποιεί παράθυρα αλλά γράφει τον δικό του κώδικα,

[1]Πηγή: https://cran.r-project.org/doc/FAQ/R-FAQ.html\#Why-is-R-named-R_003f

τα τελευταία χρόνια υπάρχουν γραφικά περιβάλλοντα της R που υποστηρίζουν τη γλώσσα και προσφέρουν επιλογές μέσω παραθύρων για μεγαλύτερη ευκολία στο χρήστη.

Το Μάρτιο του 2015 υπήρχαν πάνω από 6200 πακέτα διαθέσιμα στην R, τα οποία οι χρήστες έχουν γράψει και ανεβάσει. Πλέον θεωρείται εξαιρετικά συνηθισμένο ένας ερευνητής να προτείνει μια καινούρια μεθοδολογία, για την οποία δημιουργεί και ανεβάζει το πακέτο που την υποστηρίζει στην R. Έτσι καινούριες μεθοδολογίες είναι διαθέσιμες στον τελικό χρήστη σχεδόν ταυτόχρονα με τη δημοσίευση τους κάτι που στο παρελθόν ήταν αδύνατο.

Τελειώνοντας, σε αυτή τη σύντομη περιγραφή της ιστορίας της R, είναι σημαντικό να αναφέρουμε ότι η R αποτελεί σημαντική επιλογή για τη μετάβαση σε αυτό που είναι πια γνωστό ως η «*Πρόκληση των Μεγάλων Δεδομένων*» (Big Data Challenge). Στις μέρες που ζούμε ο όγκος των διαθεσίμων δεδομένων έχει αυξηθεί με εντυπωσιακό ρυθμό, ενώ καινούριες δομές δεδομένων είναι διαθέσιμες για στατιστική επεξεργασία και ανάλυση σε τεράστιους όγκους (π.χ. κείμενα, φανταστείτε όλα τα twitts από το twitter). Η R αποτελεί σημαντική επιλογή των χρηστών προς αυτή την κατεύθυνση, καθώς υπάρχουν ήδη βιβλιοθήκες που επιτρέπουν παράλληλη επεξεργασία των δεδομένων (parallel computing), ενώ το γεγονός ότι καθένας μπορεί να πάρει έτοιμο κώδικα και να τον τροποποιήσει, καθιστά εξαιρετικά εύκολο το να δουλέψει με τεράστιους όγκους δεδομένων.

1.3 Γρήγορη εισαγωγή στην R

Το στατιστικό πακέτο R είναι ένα πλήρες πακέτο για στατιστική ανάλυση που προσφέρει μια σειρά από πλεονεκτήματα.

- Ποικιλία σύγχρονων στατιστικών μεθόδων.
- Τις πιο σύγχρονες τεχνικές σχετικά με θέματα στατιστικής μοντελοποίησης.
- Μεγάλη ποικιλία και εξαιρετική ποιότητα γραφικών.
- Εύκολο προγραμματισμό νέων συναρτήσεων και μεθοδολογιών.

Στο ανά χείρας σύγγραμμα θα ασχοληθούμε εν συντομία με την R από την άποψη του προγραμματισμού και της χρήσης των εντολών και καθόλου με την χρήση του πακέτου μέσα από τις παραθυρικές εφαρμογές. Επίσης θα παρουσιάσουμε λίγα από τα πολλά χρήσιμα χαρακτηριστικά της, αναφέροντας τα βασικά μόνο στοιχεία που χρειάζεται κανείς για να δουλέψει στην R.

Πριν ξεκινήσουμε θα πρέπει να αναφέρουμε μερικά γενικά χαρακτηριστικά.

- Η R διατίθεται δωρεάν στο διαδίκτυο από την ιστοσελίδα `http://www.r-project.org/`. Οποιοσδήποτε μπορεί να κατεβάσει και να χρησιμοποιήσει δωρεάν την R. Τα πλεονεκτήματα της R είναι πως είναι απλή, χρειάζεται λιγότερο χώρο στο σκληρό δίσκο, εκτελείται γρήγορα (ειδικά σε ότι έχει να κάνει με επαναλήψεις).

- Ουσιαστικά, η R είναι ένα πακέτο γραμμένο με τη χρήση κάποιας άλλης γλώσσας προγραμματισμού και μάλιστα κανείς μπορεί να ενσωματώσει κώδικα από άλλες γλώσσες προγραμματισμού (π.χ. Fortran, C++). Το πακέτο περιέχει μια σειρά από βιβλιοθήκες/πακέτα που εκτελούν αυτόματα κάποιες μεθόδους και που δεν χρειάζεται ο χρήστης να προγραμματίσει. Από την άλλη, μπορεί εύκολα κανείς να εισάγει δικές του ρουτίνες ή και να τροποποιήσει τις ρουτίνες που ήδη υπάρχουν. Συνεπώς, με αυτόν τον τρόπο η R συνέχεια αναπτύσσεται και εξαπλώνεται. Στην ιστοσελίδα της R υπάρχουν χιλιάδες διαθέσιμα πακέτα για ελεύθερη χρήση. Επομένως, πριν γράψετε κάποια πολύπλοκη ρουτίνα, σας συμβουλεύουμε να ψάξετε τα διαθέσιμα πακέτα και είναι πολύ πιθανό να βρείτε έτοιμο αυτό που χρειάζεστε.

  Ένα πρόβλημα που μπορεί να προκύψει με τη χρήση των πακέτων είναι ότι μπορεί να αντιμετωπίσετε προβλήματα κατανόησης των συναρτήσεων που διαθέτουν. Ο λόγος μπορεί να είναι τα ελλειπή αρχεία βοήθειας, μη ύπαρξη παραδειγμάτων ή η μη χρήση μιας τυποποιημένης γραφής των συναρτήσεων. Μια καλή πηγή υλικού που εξηγεί και αξιολογεί πακέτα της R, είναι το επιστημονικό περιοδικό *Journal of Statistical Software*, στο οποίο μπορείτε να βρείτε επιπλέον χρήσιμο υλικό για αρκετά δημοφιλή πακέτα. Η δημοσίευση ενός άρθρου και του αντίστοιχου πακέτου στο συγκεκριμένο περιοδικό εξασφαλίζει ότι έχει γίνει κριτική ανάγνωση και χρήση του πακέτου, με αποτέλεσμα να υπάρχουν κάποιες ελάχιστες προδιαγραφές (π.χ. αξιοπρεπή βοήθεια και σύνταξη παρόμοια με άλλες συναφείς συναρτήσεις).

1.4 Βασικά χαρακτηριστικά της R

Μερικά χρήσιμα και βασικά χαρακτηριστικά της R είναι τα ακόλουθα.

- Όταν δουλεύουμε με τη γραμμή εντολών της R, δηλαδή το παράθυρο με τον τίτλο «R Console», στην αρχή μπορούμε να δούμε ένα από τα ακόλουθα σύμβολα:

  > το οποίο είναι το σύμβολο αναμονής και υπονοεί πως το πρόγραμμα περιμένει να γράψει ο χρήστης την εντολή του την εντολή μας,

  \+ το οποίο είναι το σύμβολο που υποδηλώνει ότι η εντολή που εισάγαμε είναι ατελής και αναμένει να συνεχιστεί σε αυτή τη γραμμή.

  Επιπλέον, τα αποτελέσματα ξεκινούν πάντα με έναν αριθμό μέσα σε αγκύλες , π.χ. `[1]` για να μπορεί κανείς εύκολα να βρει το αποτέλεσμα που χρειάζεται. Όταν τα αποτελέσματα δίνονται σε απλή μορφή (δηλ. σε μορφή διανύσματος), τότε το νούμερο μέσα στις αγκύλες υποδηλώνει τον αύξοντα αριθμό του αντικειμένου που τυπώνει. Σε πιο σύνθετα αντικείμενα, η μορφή αλλάζει ανάλογα με την διάταξη του αντικειμένου.

- Ένα σημαντικό σύμβολο σε όλες τις γλώσσες προγραμματισμού είναι το σύμβολο της εκχώρησης ή ανάθεσης. Στην R, οι εκχωρήσεις ή αναθέσεις σε ένα αντικείμενο γίνονται με το σύμβολο <− του βέλους, σχηματισμένου από 2 χαρακτήρες («< » και «− »). Με

την εκχώρηση η τιμή του δεξιού μέλους μιας παράστασης εκχωρείται στο αριστερό μέλος. Για παράδειγμα η εντολή `x<-5` σημαίνει πως εκχωρούμε στο αντικείμενο `x` την τιμή `5`. Στις τελευταίες εκδόσεις της R προστέθηκε και η χρήση του χαρακτήρα της ισότητας `=` που συνήθως χρησιμοποιείται σε πολλές γλώσσες προγραμματισμού, ως σύμβολο εκχώρησης. Σε ολόκληρο το κείμενο που ακολουθεί θα χρησιμοποιήσουμε τον πρώτο τρόπο για να εκχωρούμε τιμές σε μεταβλητές καθώς είναι ο πιο σωστός τρόπος συμβολισμού της εκχώρησης και παραδοσιακά πιο διαχρονικός στην R.

- Η R είναι ευαίσθητη στη χρήση κεφαλαίων και μικρών γραμμάτων (case sensitive). Αυτό σημαίνει ότι θεωρεί ως διαφορετικούς χαρακτήρες το ίδιο γράμμα ως κεφαλαίο και ως μικρό. Συνεπώς, η ίδια λέξη γραμμένη με μικρά ή κεφαλαία γράμματα θα αντιστοιχεί σε διαφορετικές τιμές ή αντικείμενα στην R.

- Χρησιμοποιώντας τα βελάκια μπορούμε να επανακτήσουμε προηγούμενες εντολές χωρίς να χρειαστεί να τις ξαναπληκτρολογήσουμε. Παρόμοιες δυνατότητες υπάρχουν μέσα από το αρχείο ιστορικού (history), όπου ο χρήστης μπορεί να δει όλες τις εντολές που έχει πληκτρολογήσει μέχρι τη στιγμή της τελευταίας αποθήκευσης του ιστορικού.

- Ίσως η πιο σημαντική εντολή της R είναι η εντολή `help`. Η R διαθέτει λεπτομερέστατη βοήθεια για τη σύνταξη όλων των εντολών αλλά και παραδείγματα χρήσης τους.

- Στην R επιτρέπεται να γράφουμε μια μόνο εντολή σε κάθε γραμμή. Ο μόνος τρόπος για να γράψουμε περισσότερες εντολές σε μια γραμμή είναι να χωρίσουμε τις εντολές με το χαρακτήρα «;».

- Μια άλλη επιλογή της R είναι η χρήση σειράς εντολών, δηλαδή ενός προγράμματος (script), μέσω του παραθύρου R editor. Αυτό μπορεί να γίνει από το μενού File>New script. Σε αυτό το παράθυρο μπορούμε να γράψουμε πολλές εντολές και να τις τρέξουμε όλες μαζί απλά με το να τις μαρκάρουμε και στη συνέχεια πατώντας το πλήκτρο Run line or Selection που εμφανίζεται πάνω από τον `R editor`, όταν είναι ενεργός. Το πλήκτρο αυτό είναι το μεσαίο από τα πέντε που εμφανίζονται και απεικονίζεται με το εικονίδιο  . Στη συνέχεια θα δούμε συνοπτικά αυτό τον τρόπο εκτέλεσης εντολών.

- Οτιδήποτε ακολουθεί μετά το χαρακτήρα `#` η R το αγνοεί, καθώς το καταλαβαίνει σαν σχόλιο και όχι σαν εντολή. Αυτό ισχύει ακόμα και αν σε μια γραμμή πληκτρολογήσουμε μια εντολή και στην ίδια γραμμή προσθέσουμε κάποιο σχόλιο με τη χρήση του χαρακτήρα `#`.

- Ένα βασικό χαρακτηριστικό της R είναι τα αντικείμενα. Ουσιαστικά, όλες οι μεταβλητές αλλά και οι συναρτήσεις (εντολές) που ορίζουμε είναι αντικείμενα. Τα αντικείμενα αυτά αποτελούνται από μια μεγάλη ποικιλία διαφορετικών μεταβλητών ή τύπων δεδομένων. Τα αντικείμενα είναι αυτόνομα και ουσιαστικά είναι δομές που οργανώνουν τα δεδομένα μας.

Αποθηκεύονται στο χώρο εργασίας (workspace) της R και μπορούν να ανακληθούν άμεσα, όποτε χρειαστεί.

΄Υστερα από την παράθεση των προαναφερόμενων βασικών σχολίων, ας ξεκινήσουμε να βλέπουμε μερικά από τα χαρακτηριστικά της γλώσσας. Στο κείμενο που ακολουθεί θα επεκταθούμε ελάχιστα στη στατιστική συμπερασματολογία των αποτελεσμάτων και απλά θα μιλήσουμε για τις εντολές και τον προγραμματισμό με R. Θα πρέπει να τονίσουμε πως αυτό το σύγγραμμα είναι απλά εισαγωγικό, οπότε η στατιστική ανάλυση και μοντελοποίηση (όπου η R είναι πρωτοπόρα) θα μας απασχολήσει μόνο επιδερμικά και περιληπτικά. Είναι, όμως, σημαντικό να γνωρίζει ο αναγνώστης, πως η ποικιλία στατιστικών μοντέλων που προσφέρει η R υπάρχει μόνο σε πολύ λίγα στατιστικά πακέτα (π.χ. το SAS). Συνεπώς, η R είναι πολύτιμο εργαλείο για όποιον ενδιαφέρεται να ασχοληθεί σοβαρά με τη στατιστική. Στη συνέχεια, θα μιλήσουμε για κάποιες βασικές αρχές προγραμματισμού που προφανώς ισχύουν, όχι μόνο για την R, αλλά και γενικότερα.

1.5 Βασικές ιδέες προγραμματισμού

Ας δούμε κάποιες βασικές αρχές προγραμματισμού πριν εξετάσουμε λεπτομερέστατα την R. Ο χώρος δεν αρκεί για μια πλήρη ανάπτυξη των ιδεών, αλλά είναι βασικό να γνωρίζει κανείς κάποιες βασικές τεχνικές πριν ξεκινήσει την προσπάθεια να γράψει ένα πρόγραμμα.

Γενικά, δεν είναι σωστό να προσπαθεί να λύσει κάποιος ένα πρόβλημα γράφοντας κατευθείαν το πρόγραμμα (τις εντολές δηλαδή που χρειάζονται). Αυτό απαιτεί πολύ βαθιά γνώση και πείρα για ρεαλιστικά προβλήματα, τα οποία δεν είναι απλοϊκά και το να προσπαθήσει κανείς να λύσει απευθείας το πρόβλημα γράφοντας εντολές, συνήθως δεν αποδίδει. Μια πολύ βασική ιδέα είναι να γράψει κάποιος το πρόβλημα σε αλγοριθμική μορφή, δηλαδή να γράψει τα βήματα που χρειάζεται για την επίλυση του προβλήματος και μετά να προσπαθεί να επιλύσει τα μικρότερα επιμέρους προβλήματα. Μια τέτοια προσέγγιση σπάει ένα μεγάλο πρόβλημα σε επιμέρους προβλήματα και κάνει πιο εύκολη την επίλυση τους και άρα την επίλυση ενός μεγαλύτερου προβλήματος.

Παράδειγμα 1.1 *΄Εστω πως ενδιαφερόμαστε για την εύρεση της διαμέσου.*

΄Ενας αλγόριθμος που βρίσκει τη διάμεσο είναι ο ακόλουθος.

1. Βάλε τα δεδομένα σε αύξουσα σειρά.
2. Αν το πλήθος των δεδομένων n είναι άρτιος αριθμός, τότε:
   - βρες τις παρατηρήσεις στις θέσεις $n/2$ και $n/2+1$,
   - πάρε τον αριθμητικό μέσο τους ως τη διάμεσο.
3. Αν το πλήθος των δεδομένων n είναι περιττός αριθμός.
   - Η διάμεσος είναι η παρατήρηση στη θέση $(n+1)/2$.

.

Ο παραπάνω αλγόριθμος έχει σπάσει το πρόβλημα της εύρεσης της διαμέσου σε μικρότερα και πιο εύκολα να λυθούν προβλήματα. Από εκεί και πέρα, οι ακριβείς εντολές που θα χρησιμοποιηθούν εξαρτώνται από τη γλώσσα, στην οποία ο αλγόριθμος θα υλοποιηθεί, όμως δουλεύει για οποιαδήποτε γλώσσα. Τέτοιες λύσεις σε φυσική γλώσσα είναι ιδιαίτερα χρήσιμες και πρέπει να χρησιμοποιούνται σε κάθε περίπτωση (εκτός ίσως από την περίπτωση πολύ απλών προβλημάτων, που ουσιαστικά απαιτούν μια μόνο ή πολύ λίγες εντολές).

Τα χαρακτηριστικά ενός καλού αλγορίθμου είναι:

- Η γενικότητά του, δηλαδή πρέπει να είναι έτσι φτιαγμένος έτσι ώστε να μπορεί να αντιμετωπίσει ποικιλία διαφορετικών περιπτώσεων με επιτυχία.

- Η αποτελεσματικότητά του, δηλαδή πρέπει να επιλύει με επιτυχία το πρόβλημα σε λογικά χρονικά πλαίσια και κυρίως με αποτελεσματική χρήση των πηγών (resources) που χρησιμοποιεί.

- Η πειστικότητά του, δηλαδή να είναι εύκολα κατανοητό ότι ο αλγόριθμος λύνει πραγματικά το πρόβλημα για το οποίο φτιάχτηκε.

Τα βήματα ενός αλγορίθμου πρέπει να είναι ξεκάθαρα και αν δεν είναι, ίσως αυτό να σημαίνει πως πρέπει να τα αναλύσουμε ακόμα περισσότερο.

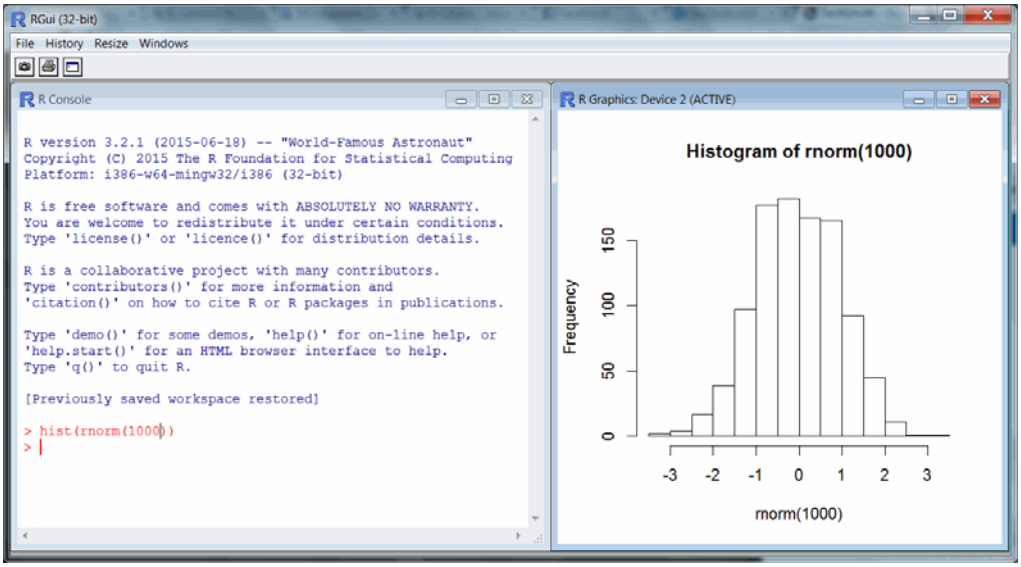
Στη συνέχεια, για λόγους οικονομίας χώρου δεν θα αναφέρουμε συχνά τη λύση των προβλημάτων σε φυσική γλώσσα, δηλαδή την κατασκευή των αλγορίθμων που επιλύουν το πρόβλημα που μας απασχολεί. Παρ'όλα αυτά, συμβουλεύουμε τον αναγνώστη να καταφεύγει πάντα σε μια λύση σε φυσική γλώσσα, πριν προσπαθήσει να γράψει το πρόγραμμα που επιλύει το πρόβλημα με το οποίο ασχολείται.

Ο προγραμματισμός δεν είναι καθόλου εύκολη υπόθεση και για αυτό το λόγο στην αρχή θα συναντήσετε αρκετά προβλήματα, τα οποία με την πάροδο του χρόνου θα βελτιώνονται. Το παρόν σύγγραμα δεν θα βοηθήσει ουσιαστικά τον αναγνώστη του, αν ο ίδιος δεν κάτσει μπροστά στον υπολογιστή και δοκιμάσει τόσο τις εντολές όσο και τον κώδικα που τον συνοδεύει.

1.6 Το περιβάλλον της R

Στο Διάγραμμα 1.1 μπορεί κανείς να δει το βασικό παράθυρο της R, όπου όλοι οι υπολογισμοί γίνονται μέσω εντολών, που πληκτρολογεί κανείς στο παράθυρο `R Console`. Επίσης, μπορείτε να δείτε και το παράθυρο γραφικών, στο οποίο τοποθετούνται τα διαγράμματα, τα οποία κατασκευάζουμε. Γενικά, τα παράθυρα της R είναι απλά και λιτά. Η επιλογή των μενού αφορά κυρίως τον ορισμό παραμέτρων εμφάνισης των παραθύρων, τη δυνατότητα να βρει κανείς και να φορτώσει βιβλιοθήκες (κάτι το οποίο αποτελεί το τεράστιο όπλο της R), όπως επίσης και η βοήθεια που παρέχεται με διάφορους τρόπους.

Μερικές χρήσιμες λειτουργίες στην R είναι οι ακόλουθες:



Εικόνα 1.1: Το βασικό παράθυρο της κονσόλας της R

- Ο χρήστης μπορεί να αποθηκεύσει το χώρο εργασίας (workspace), το οποίο περιλαμβάνει όλα τα δεδομένα μας (με τη μορφή αντικειμένων) και εντολές (με τη μορφή συναρτήσεων), που έχουμε δημιουργήσει στην τελευταία χρήση του προγράμματος. Αυτά, αποθηκεύονται στο αρχείο του χώρου εργασία με την ονομασία .Rdata στο τρέχων φάκελο εργασίας (τον οποίο μπορούμε να δούμε με την εντολή `setwd`). Η αποθήκευση μπορεί να γίνει από το μενού `file` και την επιλογή `Save Workspace` ή με την εντολή `save.image`. Μπορούμε να ανακτήσουμε τα δεδομένα ενός χώρου εργασίας με διπλό πάτημα του αριστερού κουμπιού του ποντικιού στο περιβάλλον των Windows. Εναλλακτικά, μπορούμε να χρησιμοποιήσουμε την εντολή `load` ή την επιλογή `Load Workspace` από το μενού `File` της R. Μπορείτε να βρείτε περισσότερες λεπτομέρειες στην Ενότητα 4.4.1.

- Ο χρήστης μπορεί να αποθηκεύσει ότι εντολές έχει χρησιμοποιήσει μέχρι τώρα με την επιλογή `save history`, η επιλογή αποθηκεύει σε απλό αρχείο χαρακτήρων όλες τις εντολές που έχουν πληκτρολογηθεί μέχρι εκείνη τη στιγμή.

- Ο χρήστης μπορεί να διαβάσει εντολές από αρχείο με την εντολή `source`. Περισσότερες λεπτομέρειες θα βρείτε στην Ενότητα 4.3.3.

- Από το μενού Edit ο χρήστης μπορεί να καθορίσει την εμφάνιση των παραθύρων, αλλά και να κάνει αντιγραφή και επικόλληση εντολών.

- Από το μενού Misc, ο χρήστης μπορεί να δεί όλα τα αντικείμενα που έχει μέχρι τώρα (ισοδύναμα μπορεί να χρησιμοποιήσει την εντολή `ls( )`).

- Από το μενού Packages, μπορεί να συνδεθεί μέσω του διαδικτύου με κατάλληλα sites από τα οποία μπορεί να κατεβάσει και να φορτώσει βιβλιοθήκες, δηλαδή προγράμματα που έχουν γράψει άλλοι χρήστες.

- Τέλος, από το μενού Help, μπορεί να πάρει βοήθεια σε διάφορα formats.

1.7 Χρήσιμες Συμβουλές

Είναι βασικό να κατανοήσει κανείς πως σε οποιαδήποτε γλώσσα προγραμματισμού, ο υπολογιστής εκτελεί τις εντολές, που ο χρήστης του καθορίσει. Συνεπώς, θα πρέπει να περιγράψουμε στον υπολογιστή με κάθε λεπτομέρεια, το πως ακριβώς θα εκτελεστούν αυτές οι εντολές. Για παράδειγμα, αν πρέπει να γράψουμε εντολές για να υπολογιστεί το άθροισμα πολλών αριθμών και δεν είναι διαθέσιμη μια εντολή που να το κάνει, τότε πρέπει να εξηγήσουμε στον υπολογιστή πως βρίσκουμε το άθροισμα πολλών αριθμών. Ουσιαστικά, αυτό μπορούμε να το καταφέρουμε απλά, περιγράφοντας τη διαδικασία που ο άνθρωπος χρησιμοποιεί, αλλά με αρκετή λεπτομέρεια. Έτσι, για να αθροίσουμε 10 αριθμούς, ξεκινάμε με την τιμή του πρώτου και σε αυτή προσθέτουμε το δεύτερο αριθμό και κρατάμε κάπου το άθροισμα τους. Στη συνέχεια, σε αυτό το άθροισμα προσθέτουμε τον τρίτο και ούτω καθ εξής. Προγραμματιστικά, αυτό σημαίνει πως πρέπει να υλοποιήσουμε όλες αυτές τις μικρές και ίσως βαρετές πράξεις με κατάλληλες εντολές.

Ένα άλλο απλοϊκό παράδειγμα, όπου συνήθως η ανθρώπινη αντίληψη μπορεί να μας ξεγελάσει, είναι η περίπτωση ανταλλαγής τιμών ανάμεσα σε δύο μεταβλητές, έστω `x` και `y`. Αν και κάτι τέτοιο φαίνεται εξαιρετικά απλό στον υπολογιστή, θα πρέπει να υλοποιηθεί μέσω μιας τρίτης μεταβλητής, έστω temp ως εξής:

- αποθήκευσε την τιμή του `x` στο `temp`,
- αποθήκευσε την τιμή του `y` στο `x`,
- αποθήκευσε την τιμή του `temp` στο `y`.

Τώρα τα `x` και `y` έχουν ανταλλάξει τιμές. Αν πηγαίναμε και υλοποιούσαμε απλά:

- αποθήκευσε την τιμή του `y` στο `x`,
- αποθήκευσε την τιμή του `x` στο `y`.

Η αρχική τιμή του `x` θα είχε χαθεί μετά το πρωτο βήμα, αφού αποθηκεύσαμε από πάνω της την τιμή του `y`.

Μερικές πολύ χρήσιμες συμβουλές σχετικά με τον προγραμματισμό, που αφορούν και τον προγραμματισμό με R, είναι οι ακόλουθες:

- Προσπαθήστε να χρησιμοποιείται ονόματα για τις μεταβλητές σας που να σας θυμίζουν κάτι. Αν προσπαθήσετε να δουλέψετε με ένα πρόγραμμα μετά από καιρό και έχετε χρησιμοποιήσει απλοϊκά ονόματα θα σας πάρει ώρα να θυμηθείτε και να καταλάβετε τι είχατε κάνει στο παρελθόν.

- Σε συνέχεια του προηγούμενου, προσπαθήστε να βάζετε σχόλια σε διάφορα σημεία του προγράμματος σας, με τη χρήση του συμβόλου #, αυτό θα βοηθάει εσάς αλλά και τους συνεργάτες σας να κατανοούν τι ακριβώς έχετε υλοποιήσει και πώς.

- Αν για κάποιο σκοπό έχετε από κάποια πηγή ένα πρόγραμμα έτοιμο και τσεκαρισμένο, προτιμήστε να το χρησιμοποιήσετε παρά να γράψετε το δικό σας για το οποίο θα πρέπει να αφιερώσετε πολύ χρόνο σε δοκιμές ορθότητας. Ειδικά στην R, όπου οι βιβλιοθήκες είναι διαθέσιμες και αυξάνονται συνεχώς, είναι καλή ιδέα να τις χρησιμοποιείτε αντί να προσπαθείτε να ξαναεφεύρετε τον τροχό.

- Προσπαθήστε να έχετε τις εντολές στοιχισμένες. Αυτό βοηθάει να καταλάβετε πού αρχίζει και πού τελειώνει μια εντολή και συνεπώς, να βρείτε εύκολα και γρήγορα τυχόν λάθη.

- Χρησιμοποιήστε συναρτήσεις για τα ίδια πράγματα που εκτελούνται πολλές φορές στο πρόγραμμά σας. Αυτό αφενός εξυπηρετεί την ταχύτητα εκτέλεσης του προγράμματος και αφετέρου μικραίνει το μέγεθος των εντολών, με συνέπεια να είναι πιο εύκολη η διαχείριση τους. Το ίδιο ισχύει και για ποσότητες που επαναλαμβάνονται συχνά στο πρόγραμμα σας. Για παράδειγμα, αν το μέγεθος του δείγματος το χρησιμοποιείτε συνέχεια, υπολογίστε και αποθηκεύστε το κάπου, ώστε στη συνέχεια να μην χρειάζεται να το υπολογίσετε ξανά και ξανά.

- Για την εύρεση συντακτικών λαθών, μπορείτε να χρησιμοποιήσετε έξυπνους κειμενογράφους, όπως ο «Notepad++», όπου εντοπίζουν λάθη στο κώδικα και κυρίως μπορούν να εντοπίσουν που κλείνουν (ή όχι) παρενθέσεις και αγκύλες.

Ίσως, πολλές από αυτές τις συμβουλές να σας φαίνονται υπερβολικές σε αυτή τη φάση, στην οποία συνήθως τα προγράμματα σας είναι μικρά, αλλά καλό είναι να αποκτήσει κανείς μια σωστή νοοτροπία προγραμματισμού, ώστε αργότερα να αντιμετωπίσει λιγότερα προβλήματα.

1.8 Γιατί κάποιος να προτιμήσει την R

Ένα εύλογο ερώτημα που προκύπτει είναι γιατί κάποιος να μην προτιμήσει ένα εμπορικό πακέτο με παραθυρικό περιβάλλον, αντί της R. Οι κύριοι λόγοι είναι η τεράστια ευελιξία της R και οι άπειρες επιλογές σύγχρονης ανάλυσης που δίνονται από τα πακέτα/βιβλιοθήκες της R. Πιο αναλυτικά τα πλεονεκτήματα της R δίνονται στην ακόλουθη λίστα.

- Η R είναι μια πλήρης και δομημένη γλώσσα προγραμματισμού, που προσφέρει στο χρήστη τεράστιες δυνατότητες επεξεργασίας και επίλυσης ποικίλων προβλημάτων με κάθε μορφή δεδομένων. Επίσης συνεργάζεται με άλλες γλώσσες προγραμματισμού, αφού μπορεί κανείς να καλέσει και να τρέξει μέσα από την R εντολές που έχουν γραφτεί σε C++ ή/και FORTRAN.

- Προσφέρει αποτελεσματικό τρόπο διαχείρισης και αποθήκευσης δεδομένων. Συνεργάζεται με ποικίλες βάσεις δεδομένων και μπορεί κανείς να διαβάσει δεδομένα και να τα αποθηκεύσει σε ένα ευρύ πεδίο εφαρμογών και format.

- Το βασικό πρόγραμμα της R είναι ελαφρύ και μπορεί να τρέξει ακόμα και σε υπολογιστές με περιορισμένες υπολογιστικές ικανότητες. Αυτό σημαίνει ότι δεν υπάρχουν σημαντικές απαιτήσεις σε μνήμη και, άρα μπορεί εύκολα π.χ. να χρησιμοποιηθεί από φοιτητές που διαθέτουν υπολογιστή με μικρή ισχύ.

- Είναι δωρεάν και ανοικτού κώδικα. Ο καθένας μπορεί να την κατεβάσει, να τη χρησιμοποιήσει και να τη αναδιανέμει χωρίς κάποια τιμή. Επίσης, η χρήση αδείας ανοικτού κώδικα σημαίνει ότι ο καθένας μπορεί να συνεισφέρει και να τροποποιήσει το δικό του κώδικα. Με αυτόν τον τρόπο έχει επιτευχθεί μια τεράστια διασπορά των χρηστών. Σκεφτείτε πως αυτή τη στιγμή μπορούν όλοι οι χρήστες να συνεισφέρουν ουσιαστικά στην ανάπτυξη της R.

- Το παραπάνω γεγονός, ότι η R είναι δωρεάν διαθέσιμη στο διαδίκτυο και ανοικτού κώδικα, σε αντίθεση με τα εμπορικά πακέτα, κάποια εκ των οποίων έχουν τέτοιο κόστος που δεν μπορούν να αποκτηθούν από μεμονωμένους χρήστες ή ερευνητές, αλλά μόνο από πανεπιστήμια, ερευνητικά κέντρα ή εταιρείες, έχει οδηγήσει στην ευρεία της χρήση και την περαιτέρω ανάπτυξη της.

- Συνέπεια των προαναφερόμενων είναι ότι η R έχει να παρουσιάσει μια αλματώδη ανάπτυξη τα τελευταία χρόνια με ένα τεράστιο αριθμό πακέτων και βιβλιοθηκών που αναπτύσσονται από τους χρήστες, αλλά είναι διαθέσιμες στον καθένα. Έτσι, για ερευνητικούς σκοπούς αποτελεί πολύτιμο εργαλείο. Αντίθετα, τόσο η ποικιλία των μεθόδων, όσο και ο ρυθμός ενσωμάτωσης τους είναι πολύ μικρός και σε κάποιες περιπτώσεις παίρνει χρόνια να εισαχθεί μια νέα μεθοδολογία σε αυτά. Βέβαια, εδώ χρειάζεται εξαιρετική προσοχή: οι βιβλιοθήκες που αναπτύσσονται στην R δεν σημαίνει ότι είναι εγγυημένα σωστές (ούτε βέλτιστες με βάση την υπολογιστική τους απόδοση), καθώς αναπτύσσονται από ακαδημαϊκούς ερευνητές, των οποίων το κύριο τους αντικείμενο δεν είναι ο προγραμματισμός. Σε άλλες περιπτώσεις είναι δύσχρηστες ή χωρίς ικανοποιητική βοήθεια. Αρκετές από αυτές τις βιβλιοθήκες δεν έχουν την παραμικρή συντήρηση και περαιτέρω ανάπτυξη. Παρ' όλα αυτά, είναι σε πολλές περιπτώσεις σημαντική βοήθεια για τον χρήστη, αφού μπορεί να ξεκινήσει από κάποιο σημείο. Επίσης, συνήθως, λάθη στις βιβλιοθήκες συζητιούνται στα ειδικά forum (που είναι πια αρκετά) και έτσι σύντομα υπάρχει μια εκ των έσω επιβεβαίωση και ανάπτυξη των βιβλιοθηκών αυτών.

- Η R διαθέτει μεγάλη ποικιλία διαγραμμάτων και αρκετές δυνατότητες εξαγωγής τους σε πολλά formats. Αυτή τη στιγμή μπορεί κανείς να βρει εύκολα κώδικα για να φτιάξει εντυπωσιακά, αλλά και πολύπλοκα διαγράμματα για κάθε μορφή δεδομένων. Επίσης, η

ποιότητα τους μπορεί να είναι εξαιρετική, υπάρχουν μάλιστα βιβλιοθήκες μόνο για αυτό (π.χ. `ggplot`) . Το γεγονός ότι ο τελικός χρήστης μπορεί να προσαρμόσει τα διαγράμματα αυτά ανάλογα με τις ανάγκες παρουσίασης που έχει, είναι επίσης σημαντικότατο πλεονέκτημα.

- Την εποχή των big data η R αποτελεί πολύτιμο εργαλείο και επιλογή για την ανάλυση μεγάλων όγκων δεδομένων. Η R επιτρέπει πολύ εύκολα με μικρές αλλαγές στις εντολές της παράλληλους υπολογισμούς (parallel computing), που σημαίνει ότι ο χρόνος υπολογισμών μπορεί να περιοριστεί εντυπωσιακά, το οποίο είναι εξαιρετικά σημαντικό. Επίσης, προσφέρει σημαντικά εργαλεία για την οπτικοποίηση δεδομένων μεγάλου όγκου.

1.9 Κατηγορίες αντικειμένων

Τα αποτελέσματα των μαθηματικών πράξεων παρουσιάζονται στην οθόνη, αλλά δεν αποθηκεύονται στη μνήμη του υπολογιστή, αν δεν καταχωρηθούν σε κάποια μεταβλητή. Η περίπτωση όπου ένας αριθμός καταχωρείται σε μία μεταβλητή είναι η απλούστερη δυνατή. Η R παρέχει μία πληθώρα από **δομές δεδομένων**, που ονομάζονται αντικείμενα δεδομένων (data objects). Κάθε αντικείμενο έχει συγκεκριμένο όνομα και αποτελείται από επιμέρους στοιχεία (στην ουσία είναι άλλα αντικείμενα, συνήθως απλούστερης μορφής). Τα αντικείμενα αυτά διαφέρουν, τόσο ως προς τον τύπο τους, όσο και ως προς τον τρόπο αποθήκευσης και καταχώρησης τους. Έτσι, τα αντικείμενα της R μπορεί να είναι:

- διάνυσμα (vector),
- δισδιάστατος πίνακας (matrix),
- πολυ-διάστατος πίνακας (array),
- πλαίσιο δεδομένων (data.frame),
- λίστα (list).

Στην ενότητα αυτή θα περιγράψουμε και θα παρουσιάσουμε τις προαναφερόμενες δομές δεδομένων, καθώς επίσης τον τρόπο διαχείρισής τους.

Όπως θα δούμε, οι τρεις πρώτες δομές επιβάλλουν όλα τα δεδομένα που καταχωρούνται σε αυτές να είναι του ίδιου τύπου (π.χ. όλα αριθμοί ή όλα strings), ενώ οι δύο τελευταίες επιτρέπουν την ταυτόχρονη καταχώρηση δεδομένων διαφορετικού τύπου. Οι λίστες μάλιστα μπορεί να περιέχουν ως στοιχεία άλλες δομές δεδομένων.

Η πιο απλή μορφή δεδομένων είναι τα διανύσματα, τα οποία θα περιγράψουμε πρώτα. Τα διανύσματα αυτά, δεν είναι τίποτα άλλο, παρά μια σειρά δεδομένων ιδίου τύπου που είναι τοποθετημένα σειριακά σε ένα αντικείμενο με μία συγκεκριμένη ονομασία. Κάθε μεμονωμένο στοιχείο (ή και αντικείμενο) μπορεί να ανήκει σε έναν από τους ακόλουθους τύπους δεδομένων:

- αριθμητικά δεδομένα (numeric),

- δεδομένα λογικής (logical),
- δεδομένα χαρακτήρων (character),
- μιγαδικοί αριθμοί (complex).

1.9.1 Ορισμός αντικειμένων

Ο ορισμός αντικειμένων γίνεται με εντολές ανάλογες της κλάσης ή τάξης του αντικειμένου που θέλουμε να ορίσουμε. Έτσι, για να ορίσουμε διανύσματα μπορούμε χρησιμοποιούμε τις εντολές `numeric`, `character`, `logical` και `complex` που θα περιέχουν στοιχεία αντίστοιχου τύπου. Επίσης, υπάρχουν και οι γενικότερες εντολές `vector`, `list` και `data.frame`. Όλες αυτές οι εντολές δημιουργούν διανύσματα με κάποιες αρχικές τιμές. Για παράδειγμα, η εντολή

```
> numeric(5)
[1] 0 0 0 0 0
```

δημιουργεί ένα αριθμητικό διάνυσμα μήκους πέντε στοιχείων, τα οποία είναι όλα μηδέν. Αντίστοιχα, για τους υπόλοιπους τύπους δεδομένων αν γράψουμε:

```
> logical(7)
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
> character(7)
[1] "" "" "" "" "" "" ""
> complex(3)
[1] 0+0i 0+0i 0+0i
```

■,

θα πάρουμε ένα λογικό διάνυσμα μήκους επτά με όλα τα στοιχεία να έχουν την τιμή `FALSE`, ένα διάνυσμα χαρακτήρων με επτά κενά στοιχεία, και ένα διάνυσμα με τρεις μιγαδικούς αριθμούς, οι οποίοι είναι ίσοι με μηδέν.

Στις παραπάνω εντολές, τα αντικείμενα που δημιουργήσαμε δεν αποθηκεύτηκαν πουθενά. Η «αποθήκευση» στο χώρο εργασίας μας γίνεται με το σύμβολο της εκχώρησης (`<-`) που αναφέραμε για πρώτη φορά στην Ενότητα 1.4. Για παράδειγμα, η σύνταξη

```
x <-  numeric(100)
```

θα δημιουργήσει ένα αριθμητικό διάνυσμα `x` 100 μηδενικών τιμών, ενώ η σύνταξη

```
x <- c( 1.3, 2.1, 10.5)
```

θα ορίσει ένα αριθμητικό διάνυσμα `x` τριών στοιχείων με τιμές 1.3, 2.1 και 10.5. Στις παραπάνω εντολές, το αποτέλεσμα δεν θα εμφανιστεί στην οθόνη αφού απλά καταχωρείται στο διάνυσμα `x`. Για να δούμε το αποτέλεσμα που έχει αποθηκευτεί στο αντικείμενο, απλά θα πρέπει να καλέσουμε το όνομά του π.χ.

```
> x
[1]  1.3  2.1 10.5
```

■.

Εντολές με το πρόθεμα `is.` ελέγχουν αν το αντικείμενο είναι ενός συγκεκριμένου τύπου. Για παράδειγμα, η εντολή `is.numeric(x)` θα ελέγξει αν το αντικείμενο είναι αριθμητικό διάνυσμα και θα μας επιστρέψει την τιμή `TRUE` αν όντως είναι ή την τιμή `FALSE` διαφορετικά.

Γενικά, είναι εξαιρετικά χρήσιμο να δίνουμε αρχικές τιμές στις μεταβλητές ή αντικείμενα που ορίζουμε. Η R δεν προαπαιτεί να ορίζουμε τα αντικείμενα που χρησιμοποιούμε, ούτε δίνει αυτόματα αρχικές τιμές, όπως άλλες γλώσσες προγραμματισμού. Επομένως, αν μια μεταβλητή βρεθεί στο δεξί μέρος της εντολής εκχώρησης, χωρίς να έχει ήδη κάποια τιμή, τότε θα πάρετε μήνυμα λάθους όπως φαίνεται παρακάτω:

```
> w
Error: object 'w' not found
> x<-w
Error: object 'w' not found                                    ■.
```

Προσοχή, στην περίπτωση που το όνομα που δίνετε το έχετε ξαναχρησιμοποιήσει, τότε το προϋπάρχον αντικείμενο θα αντικατασταθεί με το καινούριο, με συνέπεια τα παλιά περιεχόμενα να χαθούν (χωρίς κάποιο προειδοποιητικό μήνυμα). Αυτό είναι σημαντικό στην R γιατί κρατάει στην μνήμη της όλα τα αντικείμενα που έχουν οριστεί κατά καιρούς (εκτός αν τα διαγράψετε με συγκεκριμένες εντολές ή αλλάξετε χώρο εργασίας) και συνεπώς μπορεί ύστερα από πολύ καιρό να ορίσετε ξανά μια μεταβλητή με το ίδιο όνομα, ξεχνώντας πως έχετε ήδη μια μεταβλητή με το ίδιο όνομα.

1.9.2 Μετατροπές αντικειμένων

Γενικά, στις εντολές ορισμού που αναφέραμε παραπάνω (`numeric`, `character`, `logical`, `complex`, `vector`, `list`, `data.frame`) μπορούμε να προσθέσουμε το πρόθεμα `as.` ή το πρόθεμα `.is`. Οι εντολές με το πρόθεμα `as.` χρησιμοποιούνται για την μετατροπή ενός τύπου αντικειμένου σε άλλο τύπο. Μπορούν όμως να χρησιμοποιηθούν και για τον ορισμό ενός αντικειμένου με συγκεκριμένες τιμές στοιχείων. Για παράδειγμα οι εντολές

```
> as.numeric(1:10)
 [1]  1  2  3  4  5  6  7  8  9 10
> as.character(1:10)
 [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10"
```

ορίζουν ένα διάνυσμα δέκα στοιχείων με τιμές από το ένα εως το δέκα. Στην πρώτη περίπτωση τα στοιχεία λαμβάνονται ως αριθμητικά ενώ στη δεύτερη περίπτωση έχουν μετατραπεί σε χαρακτήρες. Οι μετατροπές από ένα τύπο σε άλλο γίνονται με εσωτερικούς κανόνες της R. Συνήθως, τα αριθμητικά στοιχεία μπορούν να μετατραπούν σε χαρακτήρες και σε λογικά στοιχεία χωρίς προβλήματα. Η μετατροπή σε λογικά στοιχεία γίνεται με τον απλό κανόνα ότι το μηδέν παίρνει την τιμή `FALSE` και όλοι οι υπόλοιποι αριθμοί την τιμή `TRUE`.

```
> as.logical(-1:1)
[1]  TRUE FALSE  TRUE                                          ■.
```

Οι λογικές τιμές μετατρέπονται, τόσο σε διανύσματα χαρακτήρων, όσο και σε αριθμητικά (βλ. παράδειγμα που ακολουθεί).

```
> as.character(c(TRUE,FALSE))
[1] "TRUE"  "FALSE"                                                    ■.
```

Όσον αφορά την μετατροπή των λογικών τιμών σε αριθμητικές, τη τιμή `FALSE` μετατρέπεται σε μηδέν και η τιμή `TRUE` σε ένα:

```
> as.numeric(c(TRUE,FALSE))
[1] 1 0                                                                 ■.
```

Η μετατροπή από χαρακτήρες σε λογικές ή αριθμητικές τιμές είναι πιο δύσκολη (βλ. μερικά παραδείγματα που ακολουθούν).

```
 > as.numeric( c(1:4,"TRUE", 'yiorgos' ))
[1]  1  2  3  4 NA NA
Warning message:
NAs introduced by coercion
> as.logical( c(1:4,"TRUE", 'yiorgos' ))
[1]   NA   NA   NA   NA TRUE   NA
> as.character( c(1:4,"TRUE", 'yiorgos' ))
[1] "1"       "2"       "3"       "4"       "TRUE"    "yiorgos"
> as.logical( c(1:4,"TRUE" ))
[1]   NA   NA   NA   NA TRUE
> as.logical( c(1:4,TRUE ))
[1] TRUE TRUE TRUE TRUE TRUE                                            ■.
```

Μετατροπές πιο σύνθετων αντικειμένων είναι πιο σύνθετες και θα πρέπει να αποφεύγονται, εκτός αν ο χρήστης έχει εμπειρία.

1.9.3 Τύπος και κλάση αντικειμένων

Μπορούμε να εμφανίσουμε τον τύπο ενός αντικειμένου με την εντολή `mode`, την εσωτερική του δομή με την εντολή `str` και σε ποια κλάσης ή τάξης αντικειμένων ανήκει με την εντολή `class`.

```
> x <- 1:12
> mode(x)
[1] "numeric"
> str(x)
int [1:12] 1 2 3 4 5 6 7 8 9 10 ...
> class(x)
[1] "integer"                                                           ■.
```

Οι διαφορές μεταξύ της κλάσης (class) και του τύπου (mode) ενός αντικειμένου δεν είναι ευδιάκριτες. Ο τύπος του αντικειμένου είναι μια ταξινόμηση με βάση την κυρία δομή των αντικειμένων.

΄Ετσι αντικείμενα με στοιχεία ίδιου τύπου, όπως διανύσματα (vector) και πίνακες (matrices ή arrays), έχουν τύπο ίδιο με τον τύπο των στοιχείων τους (δηλαδή numeric, logical, character ή complex). Αντικείμενα που περιέχουν στοιχεία με διαφορετικού τύπου δεδομένα (όπως οι λίστες και τα πλαίσια δεδομένων) δίνουν ως αποτέλεσμα `list`. Αντίθετα, κλάση (class) είναι πιο αναλυτική. Για τα διανύσματα δίνει το ίδιο αποτέλεσμα με το `mode`. Για πιο σύνθετα αντικείμενα, όπως πίνακες, πολυδιάστατους πίνακες, λίστες και πλαίσια δεδομένων δίνει ως αποτελέσματα `matrix`, `array`, `list` και `data.frame`, αντίστοιχα. Συνεπώς, ο τύπος (mode) των δεδομένων αναφέρεται κυρίως στα στοιχεία ενός αντικειμένου και στον τρόπο αποθήκευσης, ενώ η κλάση (class) στο ίδιο το αντικείμενο συνολικά. Τέλος, να αναφέρουμε ότι η εντολή `class` καθορίζει πως μερικές εντολές/συναρτήσεις θα διαχειριστούν αυτά τα αντικείμενα, όπως για παράδειγμα η εντολή `summary`, που δίνει διαφορετικά αποτελέσματα, αν ένα διάνυσμα είναι αριθμητικό, και διαφορετικά, αν είναι ένα διάνυσμα χαρακτήρων ή στοιχείων λογικής.

Πιο συγκεκριμένα, τα αριθμητικά δεδομένα είναι πραγματικοί αριθμοί, όπως για παράδειγμα 3, 3.004 και 1.9×10^{-5}, οι οποίοι δίνονται στην R με τις ακόλουθες εντολές

```
x1<-3
mode(x1)
[1] "numeric"
x2<-3.004
x3<-1.9e-5
```
■.

Τα δεδομένα λογικής (logical) παίρνουν δύο τιμές `TRUE` και `FALSE` και υποδεικνύουν αν μια πρόταση ή συνθήκη είναι αληθής ή ψευδής, αντίστοιχα (δηλαδή αν ισχύει ή όχι). Για παράδειγμα, αν θέλουμε να αποθηκεύσουμε στο αντικείμενο `x2` την απάντηση του ερωτήματος «είναι το `x1` μεγαλύτερο του `4`;» και στο αντικείμενο `x3` την απάντηση του ερωτήματος «είναι το `x1` μικρότερο του `4`;» τότε γράφουμε τις ακόλουθες εντολές:

```
x1<-3
x2<- x1>4
x2
[1] FALSE
mode(x2)
[1] "logical"
x3<- x1<4
x3
[1] TRUE
```
■.

Τα δεδομένα χαρακτήρων (character) αποτελούνται από σειρές χαρακτήρων. Χρησιμοποιούνται για την αποθήκευση ονομάτων ή χρήσιμων αναλυτικών πληροφοριών. Οι χαρακτήρες πάντα περικλείονται από λατινικού τύπου εισαγωγικά (για παράδειγμα 'Ioannis' ή "Ioannis"). Για παράδειγμα:

```
x1<-'Ioannis'
```

```
x1
[1] "Ioannis"

mode(x1)
[1] "character"
```
■.

Οι μιγαδικοί αριθμοί (complex) έχουν τη μορφή $a + bi$, όπου $i = \sqrt{-1}$ και $a,\ b$ είναι πραγματικοί αριθμοί και δηλωμένοι ως αριθμητικού τύπου (numeric mode) δεδομένα στην R. Ο αριθμοί a και b ονομάζονται πραγματικό (real) και φανταστικό (imaginary) κομμάτι του μιγαδικού αριθμού αντίστοιχα και μπορούν να οριστούν με τη χρήση της εντολής `complex(real = a, imaginary = b)`. Έτσι μπορούμε να αποθηκεύσουμε τον μιγαδικό αριθμό $4 + 3i$ στο αντικείμενο `x1` με τις ακόλουθες εντολές

```
x1<-complex(real=4,imaginary=3)
x1
[1] 4+3i

mode(x1)
[1] "complex"
```
■.

Τέλος, η εντολή `typeof` είναι παρόμοια με την εντολή `mode`, αλλά λίγο πιο αναλυτική και κυρίως αναφέρεται στον τρόπο αποθήκευσης του αντικειμένου. Για παράδειγμα, διαχωρίζει τα αντικείμενα αριθμητικού τύπου σε ακέραια (`integer`) και πραγματικά (ή διπλής ακρίβειας με την ονομασία `double`), όπως βλέπουμε στο παράδειγμα που ακολουθεί.

```
[1] "logical"
> mode(1:10)
[1] "numeric"
> typeof(1:10)
[1] "integer"
> mode(1.4:10)
[1] "numeric"
> typeof(1.4:10)
[1] "double"
> mode(c(1.5,2.3))
[1] "numeric"
> typeof(c(1.5,2.3))
[1] "double"
```
■.

1.9.4 Ειδικές τιμές των στοιχείων των αντικειμένων

Όπως είδαμε στην προηγούμενη ενότητα, κάθε στοιχείο ενός αντικειμένου παίρνει μια τιμή. Στην R υπάρχουν μερικές ειδικές τιμές τις οποίες θα αναφέρουμε σε αυτή την ενότητα. Οι

σημαντικότερες ειδικές τιμές δεδομένων είναι οι ακόλουθες.

- `-Inf, Inf`: Οι τιμές αυτές δίνονται μόνο σε αριθμητικά δεδομένα και αντιστοιχούν στο $-\infty$ και το $+\infty$. Στην πράξη, η R θέτει αυτές τις τιμές για κάθε στοιχείο που έχει τιμή μεγαλύτερη (ή μικρότερη) από την ακρίβεια της (δηλαδή τη μικρότερη και τη μεγαλύτερη δυνατή τιμή που μπορεί να χειριστεί και να αποθηκεύσει). Έτσι, για παράδειγμα μπορούμε να υπολογίσουμε το e^{700} με την εντολή

```
exp(700)
[1] 1.014232e+304
```

(και το βρίσκουμε ίσο με 1.014232×10^{304}), ενώ αντίθετα, αν προσπαθήσουμε να υπολογίσουμε το e^{1700} παίρνουμε ως αποτέλεσμα

```
exp(1700)
[1] Inf
```

■.

- `NA`: Η τιμή αυτή είναι μια λογική σταθερά που χρησιμοποιείται για να υποδείξει μη διαθέσιμες τιμές (non available), οι οποίες ονομάζονται στη στατιστική αγνοούμενες τιμές (missing values). Η τιμή αυτή μπορεί να δοθεί σε ένα στοιχείο με οποιαδήποτε τύπο δεδομένων. Μπορούμε να ελέγξουμε αν μια τιμή είναι μη διαθέσιμη με την εντολή `is.na(x)`. Όταν το αποτέλεσμά της είναι `TRUE`, τότε σημαίνει ότι αυτή η τιμή είναι μη διαθέσιμη.

- `NULL`: Η τιμή αυτή αντιπροσωπεύει το «μηδενικό» αντικείμενο στην R. Χρησιμοποιείται για αντικείμενα που δεν έχουν καθόλου στοιχεία (και τιμές) και συνεπώς έχουν μηδενικό μήκος (δηλαδή η εντολή `length` θα μας επιστρέψει το μηδέν ως αποτέλεσμα). Επιστρέφεται συνήθως ως αποτέλεσμα σε εκφράσεις που δεν έχουν οριστεί επαρκώς. Μπορούμε να ελέγξουμε εάν ένα αντικείμενο έχει τιμές (άρα και στοιχεία) με την εντολή `is.null(x)`. Όταν το αποτέλεσμα είναι `TRUE`, τότε το αντικείμενο `x` δεν έχει καθόλου τιμές.

- `NaN`: Σημαίνει «Not a Number» και υποδηλώνει έναν αριθμό που δεν μπορεί να προσδιοριστεί. Χρησιμοποιείται για πραγματικούς και μιγαδικούς αριθμούς, αλλά όχι για ακέραιους. Συνήθως, από μία αριθμητική πράξη με απροσδιόριστο αποτέλεσμα, όπως για παράδειγμα αν διαιρέσουμε το πέντε με το μηδέν. Μπορούμε να ελέγξουμε αν μια αριθμητική τιμή δεν υφίσταται με την εντολή `is.nan(x)`. Όταν το αποτέλεσμά της είναι `TRUE`, τότε σημαίνει ότι αυτή η συγκεκριμένη αριθμητική τιμή είναι απροσδιόριστη.

Στην R η τιμή `Inf`, όπως προείπαμε, αντιστοιχεί στο άπειρο και αυτό έχει χρησιμότητα σε αρκετές περιπτώσεις. Για παράδειγμα αν γράψουμε `exp(-Inf)`, τότε θα πάρουμε αποτέλεσμα μηδέν και όχι `NaN`, αφού ο υπολογιστής θα υπολογίσει ουσιαστικά το όριο $\lim_{x\to\infty} e^{-x}$. Αυτή η ιδιότητα είναι πολύ χρήσιμη, καθώς συνήθως άλλα προγράμματα δίνουν ως αποτέλεσμα υπερχείλιση, δηλαδή η πράξη δεν μπορεί να εκτελεστεί.

Στο σημείο αυτό, θα θέλαμε να αναφέρουμε τις διαφορές μεταξύ NA και NaN. Η τιμή NA χρησιμοποιείται για κάθε τύπο αντικειμένου (αριθμητικό, χαρακτήρων ή λογικό) και σημαίνει ότι είναι μη διαθέσιμη (και συνεπώς άγνωστη). Αντίθετα, η τιμή NaN χρησιμοποιείται μόνο για αριθμητικά αντικείμενα με πραγματικούς ή μιγαδικούς αριθμούς και σημαίνει ότι είναι ένας απροσδιόριστος αριθμός (ή απλά ότι δεν είναι πραγματικός ή μιγαδικός αριθμός). Συνέπεια αυτής της διαφοράς είναι, ότι αν το στοιχείο ενός αντικειμένου είναι NaN, τότε είναι και NA, όμως το αντίθετο δεν ισχύει. Δηλαδή, αν το στοιχείο ενός αντικειμένου είναι NA, τότε δεν είναι NaN. Αυτή η διαφορά απεικονίζεται και στον κώδικα που ακολουθεί, όπου το x προκύπτει ως το αποτέλεσμα της διαίρεσης μηδέν δια μηδέν, της οποίας το αποτέλεσμα είναι απροσδιόριστο. Το μοναδικό στοιχείο του αντικείμενου αυτού είναι ταυτόχρονα και NaN και NA (αφού οι εντολές is.nan και is.na έχουν τιμές TRUE). Αντίθετα, το αντικείμενο y (που επίσης έχει οριστεί ως αριθμητικό διάνυσμα μήκους ενός στοιχείου) έχει οριστεί κατευθείαν ως μη διαθέσιμο. Συνεπώς, είναι NA, αλλά όχι NaN, αφού τα αποτελέσματα των αντίστοιχων συναρτήσεων είναι TRUE και FALSE.

```
> mode(x)
[1] "numeric"
> help(class)
> length(x)
[1] 1
> x<- 0/0
> x
[1] NaN
> is.na(x)
[1] TRUE
> is.nan(x)
> y<-numeric()
> y[1]<-NA
> y
[1] NA
> mode(y)
[1] "numeric"
> is.na(y)
[1] TRUE
> is.nan(y)
[1] FALSE                                                    ■.
```

Τέλος, μπορούμε να έχουμε ένα διάνυσμα το οποίο θα έχει και NA και NaN τιμές. Στον κώδικα που ακολουθεί, έχουμε ορίσει ένα αριθμητικό διάνυσμα μήκους τριών στοιχείων. Το πρώτο στοιχείο έχει 1.5, συνεπώς δεν είναι ούτε NA ούτε NaN. Το δεύτερο στοιχείο είναι NA και έχει τις τιμές TRUE και FALSE στις εντολές (που είναι λογικά ερωτήματα) is.na και is.nan,

αντίστοιχα. Και τέλος, το τρίτο στοιχείο είναι και NaN και NA (αφού και οι δύο εντολές is.na και is.nan έχουν τιμές TRUE).

```
> y<-c(1.5, NA, 0/0)
> mode(y)
[1] "numeric"
> is.na(y)
[1] FALSE  TRUE  TRUE
> is.nan(y)
[1] FALSE FALSE  TRUE
```

■.

1.10 Επιπλέον βιβλιογραφικό υλικό

Γενικά, υπάρχει άφθονο ελεύθερο υλικό για την R στο διαδίκτυο που μπορεί να βοηθήσουν επιπλέον τον αναγνώστη[2]. Ενδεικτικά να ξεχωρίσουμε τα βιβλία των Venables et al. (2015) και του Faraway (2002). Από τα εμπορικά συγγράμματα ξεχωρίζουν αυτά του Crawley (2012), και του Matloff (2011). Επίσης όσον αφορά την παλινδρόμηση και τα στατιστικά μοντέλα, το βιβλίο των Fox & Weisberg (2011) είναι πραγματικά ένα βιβλίο που αξίζει να έχουμε στη βιβλιοθήκη μας. Τέλος, το πρόσφατο βιβλίο του Forte (2015) δίνει μια πιο μοντέρνα σκοπιά της χρήσης της R μέσω της οπτικής της Επιστήμης των Δεδομένων. Όσον αφορά τα εισαγωγικά στοιχεία της R, παραπέμπουμε τον αναγνώστη στο πρώτο κεφάλαιο του βιβλίου των Venables et al. (2015).

Τέλος, όσον αφορά την ελληνική βιβλιογραφία, εκτός του παρόντος συγγράμματος, μπορείτε να ανατρέξετε στις Πανεπιστημιακές σημειώσεις των Φωκιανού & Χαραλάμπους (2010), στη διπλωματική εργασία του Καραβασίλη (2012) τις Πανεπιστημιακές σημειώσεις του Αντζουλάκου (2013) και το σύγγραμμα του Φουσκάκη (2014).

Βιβλιογραφικές Αναφορές Κεφαλαίου 1

Αντζουλάκος, Δ. (2013). Ανάλυση Δεδομένων με τη Χρήση Στατιστικών Πακέτων: Εισαγωγή στο R Σημειώσεις παραδόσεων, 2η Έκδοση. Τμήμα Στατιστικής και Ασφαλιστικής Επιστήμης, Πανεπιστήμιο Πειραιώς. Διαθέσιμες στην ιστοσελίδα http://www.unipi.gr/faculty/dantz/Introduction_to_R.pdf.

Φουσκάκης, Δ. (2014). *Ανάλυση Δεδομένων με Χρήση της R*. Εκδόσεις Τσότρας, Αθήνα.

Φωκιανός, Κ. & Χαραλάμπους, Χ. (2010). *Εισαγωγή στην R – Πρόχειρες Σημειώσεις*. 2η Έκδοση, Τμήμα Μαθηματικών & Στατιστικής, Πανεπιστήμιο Κύπρου. Διαθέσιμο στην ιστοσελίδα https://cran.r-project.org/doc/contrib/mainfokianos charalambous.pdf.

[2]βλ. για λεπτομέρειες στους ιστότοπους https://cran.r-project.org/other-docs.html και http://www.linuxlinks.com/article/20130216063859514/9oftheBestFreeRBooks-Part1.html

Καραβασίλης Γ. (2012). *Σχεδιασμός, Ανάπτυξη και Αξιολόγηση Εναλλακτικού Διδακτικού Υλικού: Ολοκληρωμένο Πληροφοριακό Σύστημα Επεξεργασίας και Ανάλυσης Ερευνητικών Δεδομένων με R*. Μεταπτυχιακή διπλωματική εργασία. Μεταπτυχιακό πρόγραμμα σπουδών "Σπουδές στην Εκπαίδευση", Ελληνικό Ανοικτό Πανεπιστήμιο. Διαθέσιμο στην ιστοσελίδα `http://users.sch.gr/gkaravasilis/index.html`.

Crawley, M. J. (2012). *The R Book*. 2nd Edition, Wiley and Sons.

Forte, R. (2015). *Mastering Predictive Analytics with R*. Packt Publishing.

Faraway, J. (2002). *Practical Regression and Anova using R*. available at `https://cran.r-project.org/doc/contrib/Faraway-PRA.pdf`.

Fox, J. & Weisberg, S. (2011). *An R Companion to Applied Regression*. Second Edition, Sage Publications.

Ihaka, R. & Gentlman, R. (1996). R: A Language for Data Analysis and Graphics. *Journal of Computational and Graphical Statistics*, **5**, 299−314.

Matloff, N. (2011). *The Art of R Programming: A Tour of Statistical Software Design*. No Starch Press.

Venables, W. N., Smith, D. M., & the R Core Team (2015). *An Introduction to R*. Version 3.2.3 (2015-12-10). Notes on R: A Programming Environment for Data Analysis and Graphics, The R Core Team.

ΚΕΦΑΛΑΙΟ 2

Διανύσματα και Τελεστές

2.1 Τελεστές στην R

Η R μας επιτρέπει να κάνουμε από τις πιο απλές μέχρι τις πιο πολύπλοκες μαθηματικές πράξεις, όπως π.χ. μεγιστοποίηση συναρτήσεων. Η πιο απλή εντολή έχει τη μορφή:

```
> 5+5 #this is my very first command
[1] 10                                                    ■.
```

Παρατηρείστε τα σύμβολα «> », «[1] », τα οποία όπως προαναφέραμε τα βάζει η ίδια η γλώσσα, αφενός για να μας προτρέψει να πληκτρολογήσουμε τις εντολές μας και αφετέρου για μας δείξει τα αποτελέσματα. Στην παραπάνω εντολή απλά ζητήσαμε να προσθέσει 2 αριθμούς. Οτιδήποτε ακολουθεί μετά τον χαρακτήρα # αγνοείται από την R, καθώς θεωρείται ότι είναι σχόλιο του χρήστη που βοηθάει την κατανόηση ή επεξήγηση των εντολών. Η παράσταση θα μπορούσε να είναι πιο πολύπλοκη περιέχοντας και συναρτήσεις, όπως για παράδειγμα η επόμενη,

```
> 3+sqrt(45)*exp(0.4)+8*cos(0.5)
[1] 20.02812                                              ■,
```

η οποία περιέχει υπάρχουσες μαθηματικές συναρτήσεις όπως η τετραγωνική ρίζα (sqrt) , η εκθετική συνάρτηση e (exp) και το συνημίτονο (cos)(βλ. 2.3).

Οι παραπάνω εντολές εμφανίζουν το αποτέλεσμα στην οθόνη χωρίς όμως να το καταχωρούν κάπου. Αν θέλουμε να καταχωρήσουμε το αποτέλεσμα σε κάποιο αντικείμενο, τότε πρέπει να δώσουμε την εντολή ως εξής:

```
> x <- 3+sqrt(45)*exp(0.4)+8*cos(0.5)
> x
[1] 20.02812                                              ■,
```

όπου οι χαρακτήρες «<− », είναι η εντολή εκχώρησης (ή ανάθεσης), όπως έχουμε αναφέρει στην Ενότητα 1.4 (βλ. σελ. 5).

2.2 Τελεστές πράξεων στην R

2.2.1 Αριθμητικοί τελεστές

Οι εντολές που ακολουθούν δίνουν αρχικές τιμές στα x και y και στη συνέχεια κάνουμε πράξεις με αυτά. Βασικά, τα ορίζουμε να έχουν ένα μόνο στοιχείο, αλλά αυτό για την R ισοδυναμεί με τον ορισμό ενός διανύσματος με ένα μόνο στοιχείο.

```
> 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
```
■.

Οι αριθμητικές πράξεις που μπορούμε να χρησιμοποιήσουμε στην R δίνονται στον Πίνακα 2.1.

Σύμβολο	**Πράξη**
+	Πρόσθεση
−	Αφαίρεση
*	Πολλαπλασιασμός
/	Διαίρεση
^	Ύψωση σε δύναμη
%/%	Ακέραια διαίρεση, επιστρέφει δηλαδή το ακέραιο μέρος της διαίρεσης
%%	Υπόλοιπο διαίρεσης

Πίνακας 2.1: Οι διάφοροι αριθμητικοί τελεστές της R

Γενικά, στις παραστάσεις η προτεραιότητα στις πράξεις είναι από αριστερά προς τα δεξιά, και οι πολλαπλασιασμοί και οι διαιρέσεις έχουν υψηλότερη προτεραιότητα από την πρόσθεση και την αφαίρεση, δηλαδή εκτελούνται πρώτα. Η ύψωση σε δύναμη έχει ακόμα μεγαλύτερη προτεραιότητα, εκτελείται δηλαδή πρώτη. Αν δεν είστε σίγουροι για την προτεραιότητα με την οποία θα εκτελεστούν οι πράξεις, ένας απλός τρόπος (ο οποίος και προτείνεται) για να αποφύγετε απροσδόκητα ή περίεργα αποτελέσματα είναι να χρησιμοποιήσετε παρενθέσεις. Οι εκφράσεις μέσα σε παρενθέσεις αποκτούν απόλυτη προτεραιότητα. Για να δείτε τη σημασία της χρήσης των

παρενθέσεων, παρατηρείστε τις επόμενες εντολές:

```
> 3+5* 8  %/% 3
[1] 13
> (3+5)* (8 %/% 3 )
[1] 16
> ((3+5)* 8) %/% 3
[1] 21
> 5/0
[1] Inf                                                    ■.
```

Ενδιαφέρουσα είναι η τελευταία εντολή όπου η διαίρεση με το 0 δίνει αποτέλεσμα άπειρο.

2.2.2 Τελεστές σύγκρισης

Εκτός από τους τελεστές αριθμητικών πράξεων, υπάρχουν και τελεστές συγκρίσεων. Για παράδειγμα, ο τελεστής == ελέγχει αν δύο στοιχεία είναι ίσα και επιστρέφει την τιμή `TRUE`, αν ισχύει η παράσταση και την τιμή `FALSE`, αν δεν ισχύει. Δηλαδή τα αποτελέσματα των τελεστών σύγκρισης είναι ένα δίτιμο αποτέλεσμα που μας δείχνει, αν ισχύει η όχι, η σχέση που ελέγχουμε και ονομάζεται λογική (logical) σταθερά ή τιμή. Άλλοι τελεστές συγκρίσεων δίνονται στον Πίνακα 2.2.

Τελεστής	**Ερμηνεία**
>	Μεγαλύτερο από
<	Μικρότερο από
>=	Μεγαλύτερο ή ίσο
<=	Μικρότερο ή ίσο
==	Ίσο
!=	Όχι ίσο

Πίνακας 2.2: Οι διάφοροι τελεστές συγκρίσεων της R

Μερικές χρήσιμες παρατηρήσεις είναι οι ακόλουθες:

- Όταν κάνουμε αναθέσεις ή ελέγχους, μπορούμε να χρησιμοποιήσουμε ισοδύναμα μόνο το πρώτο γράμμα των `FALSE` και `TRUE`. Δηλαδή μπορούμε να γράψουμε `F` αντί για `FALSE` και `T` αντί για `TRUE`.

- Η τιμή `FALSE` είναι μικρότερη της τιμής `TRUE` (είναι χρήσιμο όταν κάνουμε σύγκριση ανάμεσα σε δύο παραστάσεις).

- Μπορούμε να χρησιμοποιήσουμε συγκρίσεις στο δεξί μέλος μιας εντολής εκχώρησης και να κάνουμε αριθμητικές πράξεις. Στις πράξεις αυτές, οι λογικές τιμές μετατρέπονται σε

αριθμητικές με το FALSE να γίνεται μηδέν (0) και το TRUE να γίνεται ίσο με ένα (1). Αυτό προσφέρει τη δυνατότητα στο χρήστη να γράψει περιπλοκές κλαδωτές συναρτήσεις με μια απλή εντολή.

- Η χρήση της ισότητας σε συγκρίσεις αριθμητικών τιμών (ειδικά πραγματικών τιμών) είναι παρακινδυνευμένη, λόγω στρογγυλοποιήσεων, αλλά και συσσώρευσης μικρών λαθών μπορεί να πάρουμε τιμή FALSE σε περιπτώσεις που δεν θα έπρεπε και το αντίθετο. Συνήθως, για να ελέγξουμε την ύπαρξη μιας ισότητας όταν εμπλέκονται πραγματικοί αριθμοί χρησιμοποιούμε ανισότητα της μορφής, ότι η απόλυτη διαφορά των δύο αριθμών είναι μικρότερη από έναν πολύ μικρό αριθμό. Δείτε ένα παράδειγμα στις εντολές που ακολουθούν:

```
> 16.0000000000001 == 16
[1] FALSE
> (16.0000000000001 - 16) <  10^(-5)
[1] TRUE
> 16.00000000000001 == 16
[1] FALSE
> 16.000000000000001 == 16
[1] TRUE
```
■.

Στα ακόλουθα παραδείγματα γίνεται χρήση των αριθμητικών συγκρίσεων και στη συνέχεια ξανά σύγκριση των λογικών τιμών. Για παράδειγμα, στην πρώτη έκφραση ελέγχουμε αν το 3 είναι μικρότερο του 5 (αποτέλεσμα TRUE) και αν το 4 είναι μικρότερο του 3 (αποτέλεσμα FALSE). Μετά συγκρίνουμε τα δύο αποτελέσματα, δηλαδή αν το TRUE είναι μεγαλύτερο του FALSE που ισχύει (δηλαδή το τελικό αποτέλεσμα είναι TRUE), εφόσον στο πρώτο ορίζεται η τιμή 1 και στο δεύτερο η τιμή 0.

```
>    (3<5) > (4 < 3)
[1] TRUE
>    (3<5) > (4>3)
[1] FALSE
>    3 + 4 > 5
[1] TRUE
>    (3 + 4) > 5
[1] TRUE
>    3 + (4 > 5)
[1] 3
>    (4 > 5) + 3
[1] 3
>    (4 < 5) + 3
[1] 4
```
■.

Όπως αναφέραμε παραπάνω, μπορούμε να χρησιμοποιήσουμε τους τελεστές σύγκρισης και τις λογικές τιμές που προκύπτουν για να ορίζουμε κλαδωτές συναρτήσεις. Για παράδειγμα, έστω ότι έχουμε μια μεταβλητή `xnew` και θέλουμε να παίρνει την τιμή 100, αν μια άλλη μεταβλητή `x` είναι μεγαλύτερη από 10 και την τιμή -50, αν η `x` είναι μικρότερη ή ίση από 10, δηλαδή

$$xnew = \begin{cases} 100 & \text{if } x > 10 \\ -50 & \text{if } x \leq 10 \end{cases}.$$

Θα δούμε αργότερα πως υπάρχει μια εντολή στην R που μπορεί να βοηθήσει αρκετά σε τέτοιου είδους προβλήματα, η εντολή `if`. Προς το παρόν, όμως, ας χρησιμοποιήσουμε τις συγκρίσεις ως εξής

```
xnew <- (x>10) * 100  + (x<=10)* (-50)
```
■.

Αυτό που ουσιαστικά κάνει η εντολή αυτή είναι να εξετάζει αν η τιμή `x` μικρότερη από 10 ή όχι και για την αληθή πρόταση να δίνει την τιμή 1 και για την ψευδή την τιμή 0. Έτσι αν η τιμή είναι μικρότερη από το 10 έχουμε $1*100+0*(-50) = 100$, ενώ αν όχι $0*100+1*(-50) = -50$. Παρατηρήστε ότι δεν μπορούμε να έχουμε συγχρόνως και τις 2 παραστάσεις 0 ή 1, επομένως πάντα θα διαλέγουμε έναν από τους 2 κλάδους. Αυτό μπορεί να γενικευτεί για τη χρήση περισσότερων κλάδων. Βέβαια, με αυτόν τον τρόπο ουσιαστικά κάνουμε δύο συγκρίσεις (μια φορά για $x > 10$ και μία φορά για $x \leq 10$) και αυτό μπορεί να αποδειχτεί χρονοβόρο, όταν κάνουμε πολλαπλές τέτοιες συγκρίσεις σε ένα μεγάλο πρόγραμμα. Θα μπορούσαμε να αποθηκεύσουμε το λογικό αποτέλεσμα της της σύγκρισης $x > 10$ και έτσι να αποφύγουμε τις δύο συγκρίσεις με τον ακόλουθο κώδικα:

```
> x<- 0:20
> y <- (x>10)
> y
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
   FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
   TRUE
> as.numeric(y)
[1] 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1
> !(y)
[1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
   TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
> as.numeric(!y)
[1] 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0
> 1-y
[1] 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0
> xnew <- y * 100  + (1-y)* (-50)
> xnew
```

```
[1] -50 -50 -50 -50 -50 -50 -50 -50 -50 -50 -50 100 100 100 100 100
100 100 100 100 100                                                ■.
```

Στον παραπάνω κώδικα αποθηκεύουμε το αποτέλεσμα της σύγκρισης ($x > 10$) ως ένα λογικό διάνυσμα, με τιμές `TRUE`, όταν ισχύει η συνθήκη (δηλαδή όταν $x > 10$) και `FALSE` όταν δεν ισχύει (δηλαδή όταν $x \leq 10$). Όπως βλέπουμε και στον παραπάνω κώδικα και στα αντίστοιχα αποτελέσματα, η άρνηση μίας λογικής συνθήκης ή ενός λογικού διανύσματος y (που εκφράζεται με τον τελεστή του θαυμαστικού `!`) είναι ισοδύναμη με την έκφραση $1 - y$, όταν μετατρέπεται σε αριθμητικό διάνυσμα. Βάζοντας το λογικό διάνυσμα y και την άρνηση του (ή αλλιώς το συμπληρωματικό λογικό διάνυσμα) που δίνεται από το $1 - y$ σε μια αριθμητική πράξη, μας δίνει το επιθυμητό αποτέλεσμα.

2.2.3 Λογικοί Τελεστές

Μπορούμε να συνδυάσουμε δύο ή περισσότερες συγκρίσεις με τη χρήση των λογικών τελεστών `&` και `|`. Έτσι, χρησιμοποιώντας το `A&B` θα πάρουμε τιμή `TRUE`, αν ισχύουν και οι δύο συνθήκες που εμπλέκονται στην έκφραση (δηλαδή **και** η A **και** η B πρέπει να είναι αληθείς), ενώ με το `A|B` θα πάρουμε `TRUE`, αν ισχύει τουλάχιστον μία από αυτές (δηλαδή αν η A **ή** η B συνθήκη είναι αληθής).

```
> (x < -5) | (x >5)
[1] TRUE                                                           ■.
```

Ο τελεστής `!` υποδηλώνει άρνηση, δηλαδή το αντίθετο της λογικής έκφρασης ή της σύγκρισης που ακολουθεί. Έτσι, η παράσταση !(3>4) παίρνει την τιμή `TRUE`, επειδή η παράσταση (3>4) έχει την τιμή `FALSE`. Πρακτικά ο τελεστής `!` μετατρέπει τη λογική τιμή `TRUE` σε `FALSE` και το αντίθετο.

Ο Πίνακας 2.3 μας δίνει διαφόρους συνδυασμούς λογικών τελεστών σε συνδυασμό με τελεστές σύγκρισης. Ένας τέτοιος πίνακας ονομάζεται και πίνακας αληθείας. Από τον πίνακα μπορούμε να δούμε την τιμή διαφόρων εκφράσεων με βάση την τιμή των αρχικών εκφράσεων.

A	B	A&B	A|B	!(A&B)	!A	!B	!A &!B	!(A|B)	(!A) | (!B)
F	F	F	F	T	T	T	T	T	T
F	T	F	T	T	T	F	F	F	T
T	F	F	T	T	F	T	F	F	T
T	T	T	T	F	F	F	F	F	F

Πίνακας 2.3: Πίνακας αληθείας για τις παραστάσεις A και B

Ο παραπάνω πίνακας ουσιαστικά επιβεβαιώνει τους κανόνες που είναι γνωστοί και ως κανόνες De Morgan.

- Η άρνηση της τομής δύο ενδεχομένων είναι ίση με την ένωση των αρνήσεων των δύο ενδεχομένων ή σχηματικά χρησιμοποιώντας τελεστές της R

$$!(A\&B) = (!A) \mid (!B)$$

και

- Η άρνηση της ένωσης δύο ενδεχομένων είναι ίση με την τομή της άρνησης των δύο ενδεχομένων ή σχηματικά.

$$!(A \mid B) = (!A) \ \& \ (!B)$$

Παράδειγμα 2.1 *Ένα έτος είναι δίσεκτο όταν η χρονολογία του διαιρείται με το 4. Τα έτη όμως που διαιρούνται με το 100 δεν είναι δίσεκτα εκτός εάν διαιρούνται με το 400 (π.χ. το 2000 ήταν δίσεκτο, όχι όμως και το 1900). Να γράψετε μια έκφραση που να ελέγχει αν το έτος x είναι δίσεκτο.*

Ουσιαστικά θέλουμε να δούμε αν ισχύουν τα εξής 3 ενδεχόμενα:
Α: διαιρείται με το 4
Β: διαιρείται με το 100
Γ: διαιρείται με το 400
Προκύπτει ο παρακάτω λογικός πίνακας.

Α	Β	Γ	Είναι δίσεκτο	Παρατηρήσεις
T	T	T	T	
T	T	F	F	
T	F	T		Αδύνατο
T	F	F	T	
F	T	T		Αδύνατο
F	T	F		Αδύνατο
F	F	T		Αδύνατο
F	F	F	F	

Για να είναι ένα έτος δίσεκτο πρέπει να ισχύει το Α και να ισχύει είτε το Γ είτε η άρνηση του Β. Μπορούμε να εκφράσουμε τις 3 συγκρίσεις ως:

Α: `(x%%4==0)`

Β: `(x%%100==0)`

Γ: `(x%%400==0)`

Και επομένως η συνθήκη που θέλουμε είναι η

```
(x%%4==0) & ((x%%400==0) | !(x%%100==0))
```

Προφανώς, η συνθήκη αυτή μπορεί να γραφτεί και με άλλους ισοδύναμους τρόπους.

2.2.4 Σειρά προτεραιότητας τελεστών

Κλείνουμε την αναφορά μας στους τελεστές (αριθμητικούς, σύγκρισης και λογικούς) δίνοντας στον Πίνακα 2.4 τη σειρά προτεραιότητας τους μέσα σε μία έκφραση.

Τελεστής	Ερμηνεία
^	Ύψωση σε δύναμη
%/%, %%	Ακέραια διαίρεση, υπόλοιπο
/ , *	Διαίρεση , πολλαπλασιασμός
+,-	Πρόσθεση, αφαίρεση
<, > , <=, >=, ==, !=	Συγκρίσεις
!	Λογική άρνηση
&, |	Λογικοί τελεστές

Πίνακας 2.4: Σειρά προτεραιότητα διαφόρων τελεστών

2.3 Αριθμητικές συναρτήσεις

Στην R υπάρχει μια σειρά από μαθηματικές συναρτήσεις που είναι διαθέσιμες στο χρήστη και μπορεί να τις χρησιμοποιήσει. Αυτές αναφέρονται σε μαθηματικές συναρτήσεις που είναι χρήσιμες στη στατιστική και επιστρέφουν σαν αποτέλεσμα μια τιμή. Ο Πίνακας 2.5 περιέχει τις πιο συνηθισμένες αριθμητικές συναρτήσεις. Μπορείτε να παρατηρήσετε ότι σε αρκετές συναρτήσεις είναι διαθέσιμες, όχι μόνο οι απλές συναρτήσεις, αλλά και ο λογάριθμος τους. Αυτό οφείλεται στο γεγονός ότι αρκετές συναρτήσεις μπορεί να πάρουν υπερβολικά μεγάλες (ή μικρές) τιμές που ξεπερνούν την αριθμητική ακρίβεια της R. Λογαριθμώντας μπορούμε να τις διαχειριστούμε και να αποφύγουμε υπερχείλιση (overflow) ή υποχείλιση (underflow) των πράξεων και να πάρουμε ένα ακριβές αποτέλεσμα στο τέλος. Όταν λέμε υπερχείλιση (overflow) εννοούμε ότι ένα αριθμός γίνεται πολύ μεγάλος και ξεπερνάει τη μέγιστη ακρίβεια της R (η οποία είναι $\geq e^{709}$) και αυτόματα τίθεται ότι είναι ίσος με συν άπειρο (`+inf`), με αποτέλεσμα να μην μπορούν να γίνουν περαιτέρω πράξεις σωστά. Αντίστοιχα, όταν λέμε υπόχείλιση (underflow) εννοούμε ότι ένα αριθμός γίνεται πολύ μικρός και ξεπερνάει την ελάχιστη ακρίβεια της R (η οποία είναι $\leq e^{-740}$) και αυτόματα τίθεται ότι είναι ίσος με πλην άπειρο (`-inf`).

Χαρακτηριστικό παράδειγμα είναι αυτό των συνδυασμών, όπως εκφράζονται από τις εντολές `choose` και `lchoose` και υπολογίζεται από την έκφραση

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

Έτσι, εάν ζητήσουμε τους συνδυασμούς 2500 πραγμάτων ανά 1500, τότε ο αριθμός των συνδυασμών είναι τόσο μεγάλος που δεν μπορεί να αναπαρασταθεί στον υπολογιστή. Αντίθετα αν πάρουμε το λογάριθμο τότε μπορούμε να τον έχουμε σε αριθμητική μορφή. Για παράδειγμα,

```
> choose(2500, 1500)
[1] Inf
> lchoose(2500, 1500)
[1] 1678.412
> exp(lchoose(2500, 1500))
```

Εντολή	Συνάρτηση
`log`	νεπέριος λογάριθμος
`exp`	εκθετική συνάρτηση
`log2`	λογάριθμος με βάση 2
`log10`	λογάριθμος με βάση 10
`abs`	απόλυτη τιμή
`sqrt`	τετραγωνική ρίζα
`cos`	συνημίτονο
`sin`	ημίτονο
`tan`	εφαπτομένη
`atan`	τόξο εφαπτομένης
`gamma`	συνάρτηση Γάμμα: $\Gamma(a) = \int_0^\infty x^{a-1} exp(-x) dx$
`beta`	συνάρτηση Βήτα: $B(a,b) = \int_0^1 x^{a-1}(1-x)^{b-1} dx$
`lgamma`	λογάριθμος της συνάρτησης Γάμμα
`factorial`	παραγοντικό: $n! = \prod_{i=1}^{n} i$
`choose`	συνδυασμοί: $\binom{n}{x} = \frac{n!}{x!(n-x)!}$
`lchoose`	λογάριθμος συνδυασμών

Πίνακας 2.5: Διάφορες αριθμητικές συναρτήσεις

```
[1] Inf                                                    ■.
```

Θα μπορούσε να παρατηρήσει κανείς ότι οι συνδυασμοί μπορούν να υλοποιηθούν με τη χρήση της συνάρτησης `factorial`. Για παράδειγμα, για να υλοποιήσουμε τους συνδυασμούς 200 αντικειμένων ανά 30 τότε με τη χρήση της συνάρτησης `choose`, θα πάρουμε το αριθμητικό αποτέλεσμα, ενώ αν χρησιμοποιήσουμε τα παραγοντικά ο υπολογιστής δεν θα μπορέσει να μας δώσει αποτέλεσμα, γιατί τα παραγοντικά που εμπλέκονται, τόσο στον αριθμητή όσο και στον παρανομαστή δεν μπορούν να υπολογιστούν (ξεπερνούν την ακρίβεια της R)

```
> choose(200,30)
[1] 4.096817e+35
> factorial(200)
[1] Inf
Warning message:
In factorial(200) : value out of range in 'gammafn'
> factorial(30)
[1] 2.652529e+32
> factorial(170)
[1] 7.257416e+306
```

```
> factorial(30)*factorial(170)
[1] Inf
> factorial(200)/(factorial(30)*factorial(170))
[1] NaN
Warning message:
In factorial(200) : value out of range in 'gammafn'                    ■.
```

Ο λόγος είναι πως ο υπολογιστής κάνει τις πράξεις με τη σειρά προτεραιότητας, άρα στην περίπτωση μας προσπαθεί να υπολογίσει το 200! του αριθμητή που είναι τεράστιο νούμερο και άρα δεν τα καταφέρνει (καθώς το αποτέλεσμα είναι άπειρο γιατί είναι ένας αριθμός μεγαλύτερος από την ακρίβεια της R). Αντίστοιχα, ενώ μπορεί να υπολογίσει το 30! και το 170! τα οποία είναι ίσα με 2.652529×10^{32} και 7.257416×10^{306} αντίστοιχα, δεν μπορεί να υπολογίσει το γινόμενο τους που χρειάζεται στον παρανομαστή. Συνεπώς στο τελευταίο βήμα, η R προσπαθεί να κάνει την πράξη $+\inf / +\inf$ το οποίο είναι απροσδιόριστο και συνεπώς `NaN`. Γενικά, ο υπολογιστής δεν κάνει απλοποιήσεις αυτόματα, εκτός και αν του το πούμε. Μπορούμε να αποφύγουμε τέτοιου είδους προβλήματα δουλεύοντας σε λογαριθμική κλίμακα, δηλαδή χρησιμοποιώντας το `lfactorial` το οποίο είναι ίσο με

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

και αυτό μπορεί να επιβεβαιωθεί και με το ακόλουθο παράδειγμα στην R

```
> lfactorial(30)
[1] 74.65824
> sum(log(1:30))
[1] 74.65824                                                           ■.
```

Έτσι, δουλεύοντας σε λογαριθμική κλίμακα έχουμε

```
> lfactorial(200)
[1] 863.232
> lfactorial(30)
[1] 74.65824
> lfactorial(170)
[1] 706.5731
> lfactorial(200) - lfactorial(30)-lfactorial(170)
[1] 82.00069
> exp( lfactorial(200) - lfactorial(30)-lfactorial(170) )
[1] 4.096817e+35                                                       ■.
```

Από τον παραπάνω κώδικα βλέπουμε ότι $\log(200!) = 863.2$, $\log(30!) = 74.6$ και $\log(170!) = 706.5$, ενώ τα αντίστοιχα παραγοντικά είναι $e^{863.2}$, $e^{74.6}$ και $e^{706.5}$, αντίστοιχα, με τον πρώτο παραγοντικό να ξεπερνάει την ακρίβεια της R (που είναι περίπου e^{740}). Τέλος, να σημειώσουμε ότι η εντολή `choose` μας δίνει το τελικό αποτέλεσμα σωστά, διότι δουλεύει πρώτα σε λογαριθμική κλίμακα

και μετά υψώνει το e στο αποτέλεσμα που έχει βρει. Αυτό είναι μια γενική στρατηγική που έχει υιοθετηθεί στην R (και σε άλλες γλώσσες προγραμματισμού) και για το λόγο αυτό πολλές αριθμητικές συναρτήσεις (αν δεν έχουν άλλη συνάρτηση για το λογάριθμο) έχουν ως παράμετρο `log=FALSE` που ελέγχει αν θα δουλέψουμε σε λογαριθμική κλίμακα ή όχι.

Σημείωση, παρόλο που οι εντολές `choose` και `lchoose` χρησιμοποιούνται για τον υπολογισμό συνδυασμών, όπου οι εισαγόμενοι παράμετροι πρέπει να είναι ακέραιοι αριθμοί, στην R επιτρέπονται και πραγματικοί αριθμοί. Στην πραγματικότητα γίνεται οι ακόλουθος υπολογισμός

$$C(n,x) = \frac{(n-\lfloor x\rceil+1)(n-\lfloor x\rceil+2)\ldots n}{\lfloor x\rceil!} = \frac{\prod_{k=1}^{\lfloor x\rceil}(n-\lfloor x\rceil+k)}{\lfloor x\rceil!}$$

όπου $\lfloor x\rceil$ είναι ο ακέραιος αριθμός που είναι πιο κοντά στο x. Σε λογαριθμική κλίμακα έχουμε

$$\log C(n,x) = \sum_{k=1}^{\lfloor x\rceil}\log(n-\lfloor x\rceil+k) - \sum_{k=1}^{\lfloor x\rceil}\log k.$$

Στον κώδικα που ακολουθεί μπορείτε να δείτε μερικά σχετικά παραδείγματα με τη χρήση μη ακέραιων αριθμών στις εντολές `choose` και `lchoose`.

```
> choose( 7, 3 )
[1] 35
> choose( 7.4, 3 )
[1] 42.624
> exp(sum( log( (7.4-3+1):7.4 ) ) - sum( log(1:3)))
[1] 42.624
>
> choose( 7.4, 3.6 )
[1] 46.8864
Warning message:
In choose(7.4, 3.6) : 'k' (3.60) must be integer, rounded to 4
> choose( 7.4, 4 )
[1] 46.8864
> exp(sum( log( (7.4-round(3.6)+1):7.4 ) ) - sum( log(1:round(3.6)
   )))
[1] 46.8864                                                    ■.
```

Η R έχει αποθηκευμένες και κάποιες σταθερές, όπως το γνωστό σε όλους μας π, που στην R συμβολίζεται με `pi` και ισούται με 3.14. Συνεπώς, αν σε μια αριθμητική παράσταση χρησιμοποιήσουμε το συμβολισμό `pi`, τότε η R κατευθείαν θα κάνει την πράξη με το 3.14. Όπως φαίνεται και από τον κώδικα που ακολουθεί, εάν ορίσουμε ένα νέο αντικείμενο με το όνομα `pi`, τότε η R θα χρησιμοποιεί αυτό το όνομα για το νέο αντικείμενο, μέχρι να το σβήσουμε με την εντολή `rm`, οπότε και πάλι επανέρχεται η προκαθορισμένη τιμή του 3.14.

```
> pi
[1] 3.141593
> pi <- 1:10
> pi
 [1]  1  2  3  4  5  6  7  8  9 10
> rm(pi)
> pi
[1] 3.141593                                                    ■.
```

Τέλος, η εντολή `log`, υπολογίζει το λογάριθμο και έχει σαν προεπιλογή η R το φυσικό λογάριθμο, η εντολή `log10` υπολογίζει τον κοινό λογάριθμο με βάση το δέκα και η εντολή `log2` τον δυαδικό λογάριθμο με βάση το δύο. Αν θέλω να αλλάξω τη βάση του λογαρίθμου βάζω στην εντολή `log(x, base=p)` στην παράμετρο `base` την επιθυμητή τιμή, η προεπιλογή στην R είναι `p=exp(1)`.

Ο χρυσός κανόνας είναι να χρησιμοποιείτε τις συναρτήσεις που είναι διαθέσιμες, γιατί αυτές συνήθως, είναι βελτιστοποιημένες ως προς την αποφυγή προβλημάτων υπερχείλισης ή υποχείλισης. Αν χρησιμοποιείτε δικές σας συναρτήσεις ή κάνετε πολύπλοκες πράξεις, τότε πάντα να υπολογίζετε όλες τις εκφράσεις σε λογαριθμική κλίμακα και στο τέλος να υψώνετε στην κατάλληλη δύναμη, έτσι ώστε να πάρετε το αποτέλεσμα στην αρχική κλίμακα. Θα επανέλθουμε αργότερα με κόλπα για να αποφύγει κανείς προβλήματα υπερχείλισης ή υποχείλισης, δηλαδή μεγάλους ή μικρούς αριθμούς, που δεν μπορεί να χειριστεί ο υπολογιστής.

2.4 Διανύσματα

Η πιο βασική δομή της R είναι το **διάνυσμα (vector)**. Μάλιστα, όταν ένα μεμονωμένο στοιχείο καταχωρείται σε μία μεταβλητή, αυτή στην ουσία είναι ένα διάνυσμα με ένα στοιχείο.

Διάνυσμα είναι κάθε αντικείμενο της R, το οποίο περιέχει ένα σύνολο στοιχείων του ίδιου τύπου, καταχωρημένα με συγκεκριμένη σειρά, την οποία έχει ορίσει ο χρήστης. Δημιουργείται με την εντολή `c( )`, η οποία είναι το αρχικό της αγγλικής λέξης combine που σημαίνει συνδυάζω μοναδικές τιμές σε ένα διάνυσμα. Τα στοιχεία μέσα στην παρένθεση της εντολής `c()` διαχωρίζονται με κόμμα.

Για παράδειγμα:

```
> vec1<-c(4,-5.2,13,9.6,-0.17)
> vec1
[1]  4.00 -5.20 13.00  9.60 -0.17
> vec2<-c(5,6,7,8,9,10,11)
> vec2
[1] 5 6 7 8 9 10 11
> vec2<-c(1,3,0)
> vec2
```

```
[1] 1 3 0                                                    ■.
```

Με την εντολή `c()` μπορούμε να επικολλήσουμε στοιχεία σε διανύσματα ή και περισσότερα του ενός διανύσματα μεταξύ τους.

```
> vec3<-c(4,5,vec1)
> vec3
[1] 4.00 5.00 4.00 -5.20 13.00 9.60 -0.17
> vec4<-c(vec1,5,5)
> vec4
[1] 4.00 -5.20 13.00 9.60 -0.17 5.00 5.00
> vec5<-c(vec1,vec2)
> vec5
[1] 4.00 -5.20 13.00 9.60 -0.17 1.00 3.00 0.00               ■.
```

Στο παραπάνω πλαίσιο ορίσαμε το διάνυσμα `vec1` με στοιχεία τις τιμές 4, -5.2, 13, 9.6 και -0.17. Επίσης, ορίσαμε το `vec2` του οποίου, στη συνέχεια, αλλάξαμε τελείως τη σύνθεση. Προσέξτε ότι η R δεν μας δίνει καμία προειδοποίηση όταν μεταβάλλουμε κάποιο αντικείμενο.

2.4.1 Ιδιότητες και Χαρακτηριστικά Διανυσμάτων

Τα ονόματα των αντικειμένων της R μπορούν να αποτελούνται από οποιουδήποτε συνδυασμούς γραμμάτων, αριθμών και τελειών και πρέπει να αρχίζουν με γράμμα. Η R δεν θα εμποδίσει το χρήστη ακόμα και αν αυτός χρησιμοποιήσει για όνομα αντικειμένου μία δεσμευμένη λέξη (π.χ. το όνομα μίας έτοιμης συνάρτησης), γι' αυτό απαιτείται προσοχή στην ονοματοδοσία.

Επιπλέον τα διανύσματα έχουν τις ακόλουθες ιδιότητες:

- **Ονόματα στοιχείων**: Στην R έχουμε τη δυνατότητα να δώσουμε ονομασίες σε κάθε στοιχείο του διανύσματος με την εντολή `names(x)`. Συνήθως, όταν αναφερόμαστε σε στατιστικά δεδομένα, οι ονομασίες αυτές μπορεί να είναι τα ονόματα των ατόμων που συμπεριλαμβάνονται στο δείγμα.

- **Μήκος διανύσματος**: Ως μήκος ενός διανύσματος ορίζουμε το συνολικό αριθμό των στοιχείων του διανύσματος. Αν υπάρχουν τιμές που λείπουν (κωδικοποιημένες από την R ως `NA`) μετρώνται και αυτές. Μπορούμε αν ανακτήσουμε το μήκος ενός διανύσματος με την εντολή `length(x)`, καθώς επίσης και να το τροποποιήσουμε θέτοντας για παράδειγμα `length(x)<-10`.

- **Τύπος διανύσματος**: Όπως ήδη αναφέραμε, μπορούμε να το εξετάσουμε με την εντολή `mode(x)` και πιθανές τιμές να είναι οι «numeric», «logical», «character» και «complex».

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΔΙΑΝΥΣΜΑΤΩΝ

√ `length`: Μήκος (Αριθμός τιμών/δεδομένων στο διάνυσμα)

√ `mode`: Τύπος διανύσματος ("numeric", "logical", "character" ή "complex")

√ `names`: Ονόματα (όνομα για κάθε τιμή)

2.4.2 Ορισμών των ονομάτων των στοιχείων ενός διανύσματος

Σε κάθε διάνυσμα που ορίζουμε, αντιστοιχεί και ένα διάνυσμα το οποίο περιέχει τα ονόματα των στοιχείων του διανύσματος. Το διάνυσμα έχει το προκαθορισμένο όνομα `names` (όνομα διανύσματος). Εάν τα ονόματα των στοιχείων δεν οριστούν από το χρήστη, τότε δίνεται στο διάνυσμα αυτό η τιμή του μηδενικού διανύσματος (`NULL`), δηλαδή ενός διανύσματος χωρίς στοιχεία (μηδενικού μήκους). Για παράδειγμα:

```
> height<-c(1.75,1.84,1.81,1.63)
> names(height)
NULL                                                    ■.
```

Για να ορίζουμε τα ονόματα, τότε γράφουμε:

```
> names(height)<-c("Jim","George","John","Mary")
> names(height)
[1] "Jim"  "George" "John"  "Mary"
> height
Jim George John Mary
1.75  1.84  1.81  1.63                                  ■.
```

Στο παραπάνω παράδειγμα, ορίσαμε το διάνυσμα `height` και είδαμε ότι δεν υπάρχουν προκαθορισμένα ονόματα των στοιχείων του. Η ένδειξη `NULL` υποδηλώνει ότι το διάνυσμα `names(height)` είναι ακαθόριστο. Ύστερα, ορίσαμε στοιχεία του διανύσματος αυτού, δηλαδή ορίσαμε ονόματα για τα στοιχεία του `height`. Βλέπουμε ότι το `names(height)` είναι ένα διάνυσμα με στοιχεία χαρακτήρων (δηλαδή τύπου character). Τέλος, προσέξτε πως εμφανίζει η R τα στοιχεία ενός διανύσματος, τα οποία διαθέτουν ονόματα. Θα δούμε στη συνέχεια πως αν κάνουμε πράξεις με το διάνυσμα για το οποίο ορίσαμε ονόματα, τότε τα ονόματα ακολουθούν τις τιμές με τις οποίες τα συνδέσαμε. Έτσι, στο παράδειγμα μας, αν βάλουμε τα ύψη σε αύξουσα σειρά, τότε τα ονόματα που συσχετίζονται με κάθε ύψος θα ακολουθήσουν τις τιμές αυτόματα.

Προσοχή: Ονόματα πρέπει να δίνονται στα στοιχεία διανυσμάτων μόνο αν αυτό έχει νόημα. Στο παραπάνω παράδειγμα θεωρήσαμε ότι τα δεδομένα της `height` έδιναν το ύψος τεσσάρων ανθρώπων, τα ονόματα των οποίων δόθηκαν ως στοιχεία στο `names(height)`.

2.5 Αριθμητικά διανύσματα

Σε πολλές περιπτώσεις, χρειαζόμαστε να ορίσουμε διανύσματα, τα οποία αποτελούνται από τις ίδιες επαναλαμβανόμενες τιμές ή διανύσματα που παίρνουν συγκεκριμένες ισαπέχουσες τιμές σε ένα διάστημα. Για τη δημιουργία τέτοιων διανυσμάτων είναι χρήσιμες οι εντολές `rep` και `seq`.

2.5.1 Δημιουργία Αριθμητικών Ακολουθιών

Πολύ χρήσιμη εντολή είναι η κατασκευή μιας ακολουθίας αριθμών από το a στο b που απέχουν μεταξύ τους κατά μία μονάδα με τη σύνταξη `a:b`. Μερικά παραδείγματα είναι τα ακόλουθα:

```
> x<-1:10
> x
[1] 1 2 3 4 5 6 7 8 9 10
```

(παράγει ένα διάνυσμα με τους ακέραιους από το 1 έως το 10 σε αύξουσα σειρά),

```
> x<- -5:5
> x
[1] -5 -4 -3 -2 -1 0 1 2 3 4 5
```

(παράγει ένα διάνυσμα με τους ακέραιους από το -5 έως το 5 σε αύξουσα σειρά),

```
> x<-1.3:5.3
> x
[1] 1.3 2.3 3.3 4.3 5.3
```

(παράγει ένα διάνυσμα με τους αριθμούς από το 1.3 έως το 5.3 σε αύξουσα σειρά για τους οποίους 2 διαδοχικοί αριθμοί απέχουν μεταξύ τους κατά μία μονάδα).

Εάν θέλουμε να παράγουμε την ακολουθία σε φθίνουσα σειρά, απλά, αλλάζουμε τη σειρά των αριθμών, δηλαδή γράφουμε `b:a` για $b > a$, οπότε για παράδειγμα μπορούμε να έχουμε:

```
> x<-10:1
> x
[1] 10 9 8 7 6 5 4 3 2 1
```

(παράγει ένα διάνυσμα με τους ακέραιους από το 1 έως το 10 σε φθίνουσα σειρά).

Παρατηρούμε ότι η χρήση της άνω-κάτω τελείας (`:`) ορίζει μια ακολουθία τιμών από τον αριθμό που προηγείται του `:` μέχρι τον αριθμό που ακολουθεί το `:`, με βήμα 1. Όταν ο πρώτος αριθμός είναι μεγαλύτερος του δεύτερου, τότε η ακολουθία είναι φθίνουσα. Αν προσέξατε δεν είναι απαραίτητη η χρήση της εντολής `c()`.

Όταν η διαφορά των δύο αριθμών δεν είναι ακέραιος αριθμός, τότε η R δημιουργεί ακολουθία με βήμα 1, η οποία ξεκινάει από τον πρώτο αριθμό και σταματάει πριν ξεπεράσει το δεύτερο.

```
> x<-1.3:5.5
> x
[1] 1.3 2.3 3.3 4.3 5.3
```

```
> x<-2.7:-1.7
> x
[1] 2.7 1.7 0.7 -0.3 -1.3                                    ■.
```

Προσέξτε τι έγινε στη φθίνουσα ακολουθία, η οποία περιλαμβάνει και θετικούς και αρνητικούς αριθμούς.

Η παραπάνω σύνταξη αποτελεί συντομογραφία της εντολής `seq` (από το sequence, που σημαίνει ακολουθία, διαδοχή, αλληλουχία), η οποία είναι η κατεξοχήν συνάρτηση - εντολή για τη δημιουργία αριθμητικών ακολουθιών. Η συνάρτηση αυτή δέχεται μέχρι 5 παραμέτρους εκ των οποίων μόνο οι δύο είναι υποχρεωτικές. Οι παράμετροι που παραλείπονται παίρνουν προκαθορισμένες (default) τιμές. Συγκεκριμένα, οι παράμετροι `from`, `to` και `by` έχουν προκαθορισμένη (default) τιμή ίση με ένα (1).

΄Ετσι, λοιπόν, η σύνταξη της εντολής `seq` είναι η ακόλουθη:

`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.

Μερικά παραδείγματα είναι τα ακόλουθα:

```
> seq(from=1,to=10,by=1)
[1] 1 2 3 4 5 6 7 8 9 10  #  (αποτέλεσμα το ίδιο με 1:10)
> seq(from=10,to=1,by=-1)
[1] 10 9 8 7 6 5 4 3 2 1} #  (αποτέλεσμα το ίδιο με 10:1)
> seq(to=10,by=1)
[1] 1 2 3 4 5 6 7 8 9 10
> seq(from=10,to=1)
[1] 10 9 8 7 6 5 4 3 2 1
> seq(from=1,by=1)
[1] 1                                                        ■.
```

Παρατηρείστε τη χρήση των προκαθορισμένων (default) τιμών. Ειδικά στην τελευταία περίπτωση, όπου λείπει το όρισμα `to`, δημιουργείται ακολουθία από τον όρο 1 ως τον όρο 1 (προκαθορισμένη τιμή της παραμέτρου `to`). Ακολουθούν επιπλέον παραδείγματα:

```
> seq(from=1,by=1,length=20)
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
[20] 20
```

```
 > seq(from=10,by=2,length=12)
[1] 10 12 14 16 18 20 22 24 26 28 30 32
> seq(10)
[1] 1 2 3 4 5 6 7 8 9 10
> seq(to=15,length=5)
[1] 11 12 13 14 15                                                    ■.
```

Η `along` χρησιμοποιείται όπως φαίνεται στα ακόλουθα παραδείγματα:

```
> y<-1:20
> seq(from=10,by=10,along=y)
[1] 10 20 30 40 50 60 70 80 90 100 110 120 130 140
[15] 150 160 170 180 190 200
> seq(along=y)
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
[20]20
> seq(y)
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
[20] 20                                                               ■.
```

2.5.2 Επαναλήψεις Διανυσμάτων με τη χρήση της εντολής `rep`

Η εντολή `rep` μας δίνει ένα διάνυσμα πολλές φορές. Έτσι, αν έχουμε ένα διάνυσμα `x` με τιμές (1,4,10), το οποίο ορίζεται με τη σύνταξη

```
x <- c(1, 4, 10)                                                      ■,
```

τότε η εντολή `rep(x, 3)` θα μας δώσει ως αποτέλεσμα

```
1  4 10  1  4 10  1  4 10                                             ■.
```

Η R παρέχει την εντολή `rep()`, η οποία δέχεται ως παράμετρο ένα διάνυσμα, το οποίο και επαναλαμβάνει τόσες φορές και με τον τρόπο που καθορίζει ο χρήστης μέσω κατάλληλων παραμέτρων.

`rep(vector, times= ,each= )`	
`vector`	Το διάνυσμα το οποίο θα επαναληφθεί.
`times`	Αριθμός επαναλήψεων του διανύσματος.
`each`	Αριθμός επαναλήψεων κάθε στοιχείου του διανύσματος.

Πίνακας 2.6: Εντολή `rep` της R για δημιουργία επαναλήψεων

Η παράμετρος `times` μπορεί να είναι είτε ένας αριθμός είτε ένα διάνυσμα ίδιου μήκους με το διάνυσμα που θέλουμε να επαναλάβουμε και το δηλώνουμε στην παράμετρο `vector`. Στην περί-

πτωση που είναι διάνυσμα, τότε κάθε στοιχείο του `vector` επαναλαμβάνεται τόσες φορες, όσες το αντίστοιχο στοιχείο του `times` και μετά προχωράμε στην επανάληψη του επόμενου στοιχείου του `vector`. Δηλαδή, το πρώτο στοιχείο του `vector` επαναλαμβάνεται τόσες φορές, όσες το πρώτο στοιχείο της `times`, το δεύτερο στοιχείο του `vector` επαναλαμβάνεται τόσες φορές, όσες το δεύτερο στοιχείο της `times`, κ.ο.κ. Για παράδειγμα, η εντολή `rec( c(3,1,6), c(1,2,3))` θα έχει ως αποτέλεσμα το διάνυσμα `(3,1,1,6,6,6)`. Αντίθετα, αν η `times` είναι ένας αριθμός, τότε όλο το διάνυσμα της παραμέτρου `vector` επαναλαμβάνεται `times` φορές. Για παράδειγμα, η εντολή `rec( c(3,1,6), 2)` έχει ως αποτέλεσμα το διάνυσμα `(3,1,6,3,1,6)`, σε αντίθεση με την εντολή `rec( c(3,1,6), c(2,2,2))` έχει ως αποτέλεσμα το διάνυσμα `(3,3,1,1,6,6)`. Επιπλέον, παραδείγματα μπορείτε να δείτε στον κώδικα που ακολουθεί.

```
> rep(10,5)
[1] 10 10 10 10 10
> vec2
[1] 1 3 0
> rep(vec2,2)
[1] 1 3 0 1 3 0
>   rep(c(1,-2,0),times=2)
[1]  1 -2  0  1 -2  0
> rep(c(1,-2,0),2)
[1]  1 -2  0  1 -2  0
> rep(c(1,-2,0),c(2,2,2))
[1]  1  1 -2 -2  0  0
> rep(c(1,2,3),c(5,4,3))
[1] 1 1 1 1 1 2 2 2 2 3 3 3
```
■.

Προσέξτε τη διαφορά ανάμεσα στην παράμετρο `times` και `each`. Η παράμετρος `times` επαναλαμβάνει όλο το διάνυσμα `times` φορές τη μία μετά την άλλη, ενώ αντίθετα η παράμετρος `each` θα επαναλάβει κάθε στοιχείο του διανύσματος `(1 -2 0) each` φορές.

```
> rep(c(1,-2,0),times=2)
[1]  1 -2  0  1 -2  0
> rep(c(1,-2,0),each=2)
[1]  1  1 -2 -2  0  0
```
■.

Το `each` αντιστοιχεί στο να χρησιμοποιείς στο `times` ένα διάνυσμα με όλα τα στοιχεία του ίδια.

```
> rep(c(1,-2,0),c(2,2,2))
[1]  1  1 -2 -2  0  0
> rep(c(1,-2,0),each=2)
[1]  1  1 -2 -2  0  0
```
■.

Ενδιαφέρον παρουσιάζει ο συνδυασμός των δύο παραμέτρων. Στην ουσία πρώτα εφαρμόζεται η παράμετρος `each` και μετά, στο διάνυσμα που προκύπτει, η παράμετρος `times`.

```
> rep(c(1,-2,0),times=2,each=3)
[1]  1  1  1 -2 -2 -2  0  0  0  1  1  1 -2 -2 -2  0  0  0
> rep( c(1,-2,0),each=3 )
[1]  1  1  1 -2 -2 -2  0  0  0
> rep(rep( c(1,-2,0),each=3 ),2)
[1]  1  1  1 -2 -2 -2  0  0  0  1  1  1 -2 -2 -2  0  0  0
```

■.

Οι `seq` και `rep` μπορούν να χρησιμοποιηθούν συμπληρωματικά. Π.χ. η εντολή

```
> rep(1:4,3)
[1] 1 2 3 4 1 2 3 4 1 2 3 4
```

επαναλαμβάνει τρεις φορές την ακολουθία `1 2 3 4`.

2.5.3 Πράξεις Αριθμητικών Διανυσμάτων

Γενικά, όλοι οι αριθμητικοί τελεστές που περιγράψαμε στην Ενότητα 2.2.1 μπορούν να χρησιμοποιηθούν σε αριθμητικά διανύσματα, κάνοντας τις πράξεις στοιχείο-με-στοιχείο (elementwise), δηλαδή, μεταφέροντας την πράξη στα αντίστοιχα στοιχεία. Όταν δύο διανύσματα $\mathbf{x}$ και $\mathbf{y}$ έχουν το ίδιο μήκος n, τότε το τελικό αποτέλεσμα θα είναι ένα νέο διάνυσμα $\mathbf{z}$ μήκους n όπου το z_i (δηλαδή το i στοιχείο του $\mathbf{z}$) θα είναι ίσο με το αποτέλεσμα της πράξης μεταξύ x_i και y_i (δηλαδή των i στοιχείων του κάθε διανύσματος). Αν τα δύο διανύσματα έχουν διαφορετικό μήκος n_1 και n_2, τότε το αποτέλεσμα θα έχει μήκος $\max\{n_1, n_2\}$ και για το μικρότερο διάνυσμα εφαρμόζεται ανακύκλωση (recycling) των στοιχείων, δηλαδή τα στοιχεία συμπληρώνονται, ξεκινώντας από την αρχή του διανύσματος μέχρι να συμπληρωθεί το απαιτούμενο μήκος. Προσοχή, η ανακύκλωση στοιχείων μπορεί να είναι επικίνδυνη και να δώσει μη επιθυμητά αποτελέσματα.

Υπό αυτή την οπτική, στην R μπορούν να γίνουν πράξεις μεταξύ διανύσματος και αριθμού. Το αποτέλεσμα είναι, η πράξη να γίνει χωριστά σε κάθε στοιχείο του διανύσματος και να παραχθεί ένα νέο διάνυσμα με τα αποτελέσματα.

```
> x<-1:10
> x+2
[1] 3 4 5 6 7 8 9 10 11 12
> x-2
[1] -1 0 1 2 3 4 5 6 7 8
> x*2
[1] 2 4 6 8 10 12 14 16 18 20
> 2*x
[1] 2 4 6 8 10 12 14 16 18 20
> x/2
[1] 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
```

```
> x^2
[1] 1 4 9 16 25 36 49 64 81 100
 > 2^x
[1] 2 4 8 16 32 64 128 256 512 1024
>
> x/0
[1] Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
> -1*x/0
[1] -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf
[10] -Inf
> (x-1)/0
[1] NA Inf Inf Inf Inf Inf Inf Inf Inf Inf                    ■.
```

Στην τελευταία πράξη το πρώτο στοιχείο του `x-1` είναι μηδέν και διαιρείται με το μηδέν. Το αποτέλεσμα $0/0$ δεν ορίζεται.

Παρόμοια εφαρμόζονται σε κάθε στοιχείο ενός διανύσματος οι μαθηματικές συναρτήσεις που ζητούμε να εφαρμοστούν στο διάνυσμα:

```
> sqrt(x)
[1] 1.000000 1.414214 1.732051 2.000000 2.236068
[6] 2.449490 2.645751 2.828427 3.000000 3.162278
> log(x)
[1] 0.0000000 0.6931472 1.0986123 1.3862944
[5] 1.6094379 1.7917595 1.9459101 2.0794415
[9] 2.1972246 2.3025851
> log10(x)
[1] 0.0000000 0.3010300 0.4771213 0.6020600 0.6989700
[6] 0.7781513 0.8450980 0.9030900 0.9542425 1.0000000
> exp(x)
[1] 2.718282 7.389056 20.085537 54.598150 148.413159
[6] 403.428793 1096.633158 2980.957987 8103.083928
[10] 22026.465795
 > (2^x+exp(x)-5*x)/log10(x)
[1] -Inf 4.614344 27.426020 84.041708 222.345964
[6] 562.138522 1407.686567 3540.021518 8981.033482
[10] 23000.465795                                             ■.
```

Πράξεις γίνονται και μεταξύ διανυσμάτων ίδιου μήκους, οπότε και εκτελούνται μεταξύ των αντιστοίχων στοιχείων των διανυσμάτων. Το αποτέλεσμα είναι διάνυσμα ίδιου μήκους με τα αρχικά.

```
> x<-1:10
```

```
> y<-rep(c(2,3),each=5)
> x
[1] 1 2 3 4 5 6 7 8 9 10
> y
[1] 2 2 2 2 2 3 3 3 3 3
> x+y
[1] 3 4 5 6 7 9 10 11 12 13
> x-y
[1] -1 0 1 2 3 3 4 5 6 7
> x^y
[1] 1 4 9 16 25 216 343 512 729 1000
> x*y
[1] 2 4 6 8 10 18 21 24 27 30
> x/y
[1] 0.500000 1.000000 1.500000 2.000000 2.500000
[6] 2.000000 2.333333 2.666667 3.000000 3.333333
> (x^2)/(y-5)
[1] -0.3333333 -1.3333333 -3.0000000 -5.3333333
[5] -8.3333333 -18.0000000 -24.5000000 -32.0000000
[9] -40.5000000 -50.0000000
> x*y-2
[1] 0 2 4 6 8 16 19 22 25 28
> x*(y-2)
[1] 0 0 0 0 0 6 7 8 9 10
> exp(x-y)
[1] 0.3678794 1.0000000 2.7182818 7.3890561
[5] 20.0855369 20.0855369 54.5981500 148.4131591
[9] 403.4287935 1096.6331584
```

■.

Προσοχή χρειάζεται σε περιπτώσεις όπου τα διανύσματα δεν έχουν ίδιο μήκος. Τότε, το μήκος του αποτελέσματος είναι όσο και το μήκος του **μεγαλύτερου** από τα διανύσματα. Όσα διανύσματα έχουν μικρότερο μήκος, επαναλαμβάνονται από την αρχή τους τόσες φορές όσες χρειάζονται για να φτάσουν αυτό το μήκος.

```
> x<-1:10
> y<-rep(1:2,3)
> x
[1] 1 2 3 4 5 6 7 8 9 10
> y
[1] 1 2 1 2 1 2
> x+y
```

```
[1] 2 4 4 6 6 8 8 10 10 12
Warning messages:
In x + y : longer object length is not a multiple of shorter
 object length                                                  ■.
```

Αυτό που συνέβη είναι ότι επειδή το y έχει μήκος 6, ενώ το x έχει μήκος 10, στο τέλος του y επαναλήφθηκαν τα 4 πρώτα στοιχεία του (όσα χρειάζονταν για να αποκτήσει και αυτό μήκος 10) και στη συνέχεια έγινε η άθροιση. Σημειώνεται, ότι το μήκος του y αυξήθηκε μόνο για να γίνει η πράξη. Με το τέλος του υπολογισμού, το y είναι όπως ήταν πριν την πράξη. Επειδή δεν επαναλήφθηκε ολόκληρο το y, η R μας έδωσε ένα προειδοποιητικό μήνυμα.

```
> y<-1:2
> x+y
[1] 2 4 4 6 6 8 8 10 10 12                                      ■.
```

Στο παραπάνω παράδειγμα, το y χρειάζεται να επαναληφθεί ολόκληρο πέντε φορές πριν γίνει η πράξη και η R δεν δίνει καμία προειδοποίηση γι' αυτό. Η διαδικασία που μόλις περιγράψαμε λέγεται **ανακύκλωση** (recycling).

2.5.4 Συνθήκες και Λογικές Πράξεις σε Αριθμητικά Διανύσματα

Στα διανύσματα μπορούν να εφαρμοστούν και λογικές πράξεις. Ζητάμε να αξιολογηθεί, δηλαδή αν τα στοιχεία ενός διανύσματος ικανοποιούν μία συνθήκη. Η συνθήκη αξιολογείται σε κάθε στοιχείο του διανύσματος χωριστά και το αποτέλεσμα είναι ένα διάνυσμα από λογικές τιμές (TRUE και FALSE).

```
> height
Jim George John Mary
1.75 1.84 1.81 1.63
> height>1.8
Jim George John Mary
FALSE   TRUE   TRUE  FALSE
> height==1.75
Jim George John Mary
TRUE  FALSE  FALSE  FALSE
> height!=1.75
Jim George John Mary
FALSE   TRUE   TRUE   TRUE
> height>1.8 & height<1.83
Jim George John Mary
FALSE  FALSE   TRUE  FALSE
> height<1.70 | height>1.82
Jim George John Mary
```

```
 FALSE   TRUE  FALSE   TRUE                                          ■.
```

2.6 Λογικά διανύσματα

Λογικό είναι ένα διάνυσμα του οποίου τα στοιχεία είναι λογικές τιμές, δηλαδή `TRUE` ή `FALSE`. Συνήθως, τα λογικά διανύσματα προκύπτουν μετά από λογικές πράξεις ή συγκρίσεις μεταξύ διανυσμάτων, όπως αυτές της Ενότητας 2.5.4 ή ως το αποτέλεσμα λογικών συναρτήσεων. Αυτό δεν απορρίπτει τον απευθείας ορισμό λογικών διανυσμάτων αν και η αναγκαιότητα τέτοιων ορισμών είναι περιορισμένη στην πράξη. Έτσι. μπορούμε να ορίσουμε ένα λογικό διάνυσμα ως εξής:

```
> x<-logical(10)
> x
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
> x<-c(T,T,T,F,F,T,F)
> x
[1]  TRUE  TRUE  TRUE FALSE FALSE  TRUE FALSE                     ■.
```

Κατά αναλογία των αριθμητικών συναρτήσεων, υπάρχουν λογικές συναρτήσεις που ελέγχουν μια συνθήκη για κάθε στοιχείο ενός αριθμητικού (ή άλλου τύπου) διανύσματος. Μια τέτοια συνάρτηση είναι ή `is.na()` που μας επιστρέφει ένα λογικό διάνυσμα με τιμή `TRUE`, όπου η παράμετρος της έχει τιμή που λείπει.
Παραδείγματα:

```
> length(height)
[1] 4
> x
[1] 1 2 3 4 5 6 7 8 9 10
> x[3]<-NA
> x
[1] 1 2 NA 4 5 6 7 8 9 10
> is.na(x)
[1] F F T F F F F F F F
> x<-0:2
> x/0
[1] NA Inf Inf
> is.na(x/0)
[1] T F F                                                          ■.
```

Αντίστοιχες είναι οι εντολές: `is.finite(x)`, `is.infinite(x)`, `is.nan(x)`, που ελέγχουν αν κάποιες τιμές είναι πεπερασμένες (δηλαδή όχι ίσες με `inf` ή `-inf`) άπειρες (`inf` ή `-inf`), ή αν δεν είναι αριθμοί (`NaN`).

2.7 Επιλογή υποσυνόλων και στοιχείων διανυσμάτων

Ενδέχεται, ορισμένες φορές, να επιθυμούμε να εξετάσουμε ή να εργαστούμε με συγκεκριμένα και όχι με όλα τα στοιχεία ενός διανύσματος. Η R μας δίνει τον τρόπο να εξαγάγουμε συγκεκριμένα στοιχεία. Ο τρόπος είναι ο εξής: `vector[ ]`, όπου vector είναι το όνομα του διανύσματος, ενώ μέσα στις αγκύλες προσδιορίζουμε τα συγκεκριμένα στοιχεία. Ο προσδιορισμός γίνεται με τρεις τρόπους:

- δίνοντας αριθμούς που δηλώνουν τη θέση των στοιχείων στο διάνυσμα,
- δίνοντας τα ονόματα των στοιχείων που επιθυμούμε (εφόσον έχουν οριστεί ονόματα), και
- δίνοντας λογική συνθήκη την οποία ικανοποιούν τα στοιχεία του διανύσματος που θέλουμε.

Μερικά απλά παραδείγματα είναι τα ακόλουθα:

```
> height
 Jim George John Mary
1.75  1.84  1.81  1.63
> height[3]
John
1.81
> height[c(1,3)]
Jim John
1.75  1.81
> height[1:2]
Jim George
1.75  1.84
> height[c(4,1)]
Mary Jim
1.63  1.75
> height[c(4,1,4)]
Mary Jim Mary
1.63  1.75  1.63
> height[height>1.75]
George   John
1.84   1.81
> height[c(T,T,F,T)]
 Jim George   Mary
1.75   1.84   1.63
```

■.

Όταν χρησιμοποιούμε τις αγκύλες ζητάμε από την R να μας δώσει εκείνα τα στοιχεία που έχουν θέση ίση με τον αριθμό της αγκύλης. Πιο συγκεκριμένα, στο παράδειγμα `height[c(4,1)]`

ζητάμε το τέταρτο και το πρώτο στοιχείο του διανύσματος`height`. Επίσης, όταν στην αγκύλη βάλουμε μια λογική συνθήκη, όπως `height>1.75`, τότε παίρνουμε σαν αποτέλεσμα τα στοιχεία εκείνα που την πληρούν. Τέλος, αν στην αγκύλη βάλουμε ένα διάνυσμα με λογικούς τελεστές `TRUE` ή `FALSE`, τότε επιστρέφει εκείνα τα στοιχεία που τους επαληθεύουν. Πιο αναλυτικά, στο παράδειγμα `height[c(T,T,F,T)]`,ελέγχουμε αν το πρώτο του διανύσματος `height` πληρεί την συνθήκη `height>1.75`, δηλαδή έχει ως αποτέλεσμα `TRUE`, αν ισχύει το εκτυπώνει διαφορετικά πάει στο επόμενο και εκτυπώνει εκείνο το στοιχείο που πρώτο θα πληρεί την συνθήκη. Για παράδειγμα, το τρίτο στοιχείο είναι `FALSE`, άρα ζητάμε ένα στοιχείο που να μην πληρεί τη συνθήκη, το τρίτο στοιχείο του διανύσματος `heght` πληρεί την συνθήκη, άρα προχωράμε στο επόμενο που είναι η `Mary` και δεν την πληρεί, έτσι εκτυπώνεται το τέταρτο στοιχείο του διανύσματος `height`.

Μπορούμε να εξάγουμε το στοιχείο του διανύσματος `x` που βρίσκεται στην θέση `pos` πολύ απλά γράφοντας `x[pos]`. Για παράδειγμα:

```
> x<-c( 11,22,34,55 )
> x[3]
[1] 34
```
■.

δηλαδή, τυπώσαμε το 3ο στοιχείο του διανύσματος `x`. Ισοδύναμα, την παράμετρο της θέσης `pos` μπορούμε να αντικαταστήσουμε με την ονομασία της γραμμής που επιθυμούμε. ΄Ετσι, η εντολή

```
> height["Mary"]
Mary
1.63
```

θα μας δώσει το ύψος της Μαίρης, όπως είχε οριστεί παραπάνω.

Μπορούμε να αντικαταστήσουμε επιπλέον το `pos` με ένα διάνυσμα που θα υποδεικνύει ποια στοιχεία θέλουμε να αποσπάσουμε από το διάνυσμα. ΄Ετσι, η εντολή

```
> x[c(1,3)]
[1] 11 34
```

θα μας δώσει το 1ο και το 3ο στοιχείο του διανύσματος `x`. Παρόμοιο αποτέλεσμα έχουμε όταν εισάγουμε ένα διάνυσμα χαρακτήρων με τα ονόματα των στοιχείων. ΄Ετσι, η εντολή

```
> height[c("Mary","Jim")]
```

θα μας δώσει ως αποτέλεσμα

```
Mary Jim
1.63 1.75
```
■.

΄Ενας, επίσης, πολύ χρήσιμος τρόπος ορισμού των στοιχείων που θέλουμε να εξάγουμε είναι με τη χρήση λογικών συνθηκών ή διανυσμάτων. ΄Ετσι, για παράδειγμα, αν επιθυμούμε να κρατήσουμε τα άτομα με ύψος μεγαλύτερου του 1.80 μέτρων, τότε μπορούμε να γράψουμε:

```
> height[height<=1.8]

Jim Mary

1.75 1.63                                                    ■.
```

Αυτό είναι στην ουσία ισοδύναμο με το ακόλουθο:

```
> z<-height<=1.8
> z
   Jim George   John   Mary
  TRUE  FALSE  FALSE   TRUE
> height[z]
 Jim Mary
1.75 1.63                                                    ■.
```

Προφανώς, μπορούμε να χρησιμοποιήσουμε και σύνθετες συνθήκες, όπως να ζητήσουμε τα στοιχεία που αντιστοιχούν σε άτομα με ύψος μικρότερο ή ίσο του 1.80 και μεγαλύτερο ή ίσο του 1.82 :

```
> height[height<=1.8 | height>=1.82]
Jim George Mary
1.75 1.84 1.63                                               ■.
```

Οι λογικές συνθήκες, στα παραπάνω παραδείγματα, αναφέρονται στο ίδιο διάνυσμα. Μπορούν, όμως, να αναφέρονται και σε ένα άλλο, αρκεί βέβαια αυτό να συνδέεται εννοιολογικά με το διάνυσμα από το οποίο εξάγουμε στοιχεία. Για παράδειγμα, έστω ότι για τα άτομα των οποίων έχουμε καταγράψει το ύψος στο διάνυσμα `height`, έχουμε καταγράψει και το φύλο στο διάνυσμα `sex`. Μπορούμε να ζητήσουμε τα ύψη των αντρών, αλλά και ποιοι άνδρες έχουν ύψος μεγαλύτερο ή ίσο του 1.80.

```
> sex<-c("Male","Male","Male","Female")
> height[sex=="Male"]
Jim George John
1.75 1.84 1.81
> height[height>1.8 & sex=="Male"]
George John
1.84 1.81                                                    ■.
```

Γενικά, θα πρέπει το λογικό διάνυσμα που εισάγουμε ως δείκτη των διανυσμάτων να έχει ίδιο μήκος.

Επίσης, μπορούμε να δηλώσουμε ποια στοιχεία **δεν επιθυμούμε** να εξαχθούν, δίνοντας τη θέση τους με ένα «μείον»μπροστά τους. Έτσι, αν για παράδειγμα θέλουμε να κρατήσουμε όλα τα στοιχεία του διανύσματος `height` εκτός από το 2ο και το 4ο, τότε γράφουμε:

```
> height[-c(2,4)]
Jim John
1.75 1.81                                                    ■.
```

Τέλος, μπορούμε να επιλέξουμε μόνο τις τιμές που δεν έχουν missing values με την εντολή `is.na` που μας δίνει `TRUE` όταν έχουμε missing value. Για παράδειγμα:

```
> x<-c( 1, 5, NA, 4, 5, 10, NA, 3, NA )
> x
[1]  1  5 NA  4  5 10 NA  3 NA
> x[ is.na(x) ]
[1] NA NA NA
> x[ !is.na(x) ]
[1]  1  5  4  5 10  3
> index<-1:length(x)
> index
[1] 1 2 3 4 5 6 7 8 9
> index[ is.na(x) ]
[1] 3 7 9                                                    ■.
```

Η εντολή `x[ !is.na(x) ]` είναι που μας δίνει τα στοιχεία του `x` που δεν είναι missing. Αν θέλουμε τη θέση στο διάνυσμα που παρατηρούνται τα missing values τότε πρέπει να γράψουμε τις ακόλουθες εντολές.

2.8 Συναρτήσεις αριθμητικών διανυσμάτων

Στην ενότητα αυτή θα περιγράψουμε συναρτήσεις, όπου εισάγουμε ένα διάνυσμα και το αποτέλεσμα είναι μία τιμή σε αντίθεση με τις απλές αριθμητικές συναρτήσεις, όπου εισάγουμε διανύσματα και το αποτέλεσμα είναι ξανά ένα διάνυσμα ίδιου μήκους.

Η R παρέχει εντολές για την εύρεση της ελάχιστης και της μέγιστης τιμής ενός διανύσματος (`min` και `max` αντίστοιχα) και για την εύρεση του αθροίσματος και του γινομένου των τιμών του (`sum` και `prod` αντίστοιχα).

```
> min(height)
[1] 1.63}
> max(height)
[1] 1.84
> sum(height)
[1] 7.03
> prod(height)
[1] 9.499966                                                 ■.
```

Τέλος, υπάρχουν και αρκετές συναρτήσεις, οι οποίες όταν οριστούν για ένα διάνυσμα επι-

Εντολή	Περιγραφικό μέτρο
`mean`	μέση τιμή
`var`	διακύμανση (αμερόληπτη)
`median`	διάμεσος
`quantile`	ποσοστιαία σημεία
`cor`	συντελεστής συσχέτισης
`cov`	συνδιακύμανση

Πίνακας 2.7: Στατιστικές συναρτήσεις για τον υπολογισμό περιγραφικών μέτρων

στρέφουν περιγραφικά στατιστικά μέτρα για το διάνυσμα αυτό (βλ. Πίνακα 2.7).

Περισσότερα για τα περιγραφικά μέτρα θα δούμε αργότερα. Μπορεί κανείς να παρατηρήσει ότι πολλά από τα γνωστά περιγραφικά μέτρα δεν είναι άμεσα διαθέσιμα με μια εντολή, με αποτέλεσμα ο χρήστης να πρέπει να τα ορίσει. Για παράδειγμα, οι παρακάτω εντολές υπολογίζουν το συντελεστή ασυμμετρίας.

```
> m<-mean(x)
> m3<- sum ((x - m)^3 )/length(x)
> m3/var(x)^1.5                                                    ■.
```

2.9 Εντολές Ταξινόμησης

Η εντολή `sort` χρησιμοποιείται για την ταξινόμηση των τιμών ενός διανύσματος κατά αύξουσα τάξη μεγέθους. Αν στο αποτέλεσμά της εφαρμόσουμε την εντολή `rev`, οι τιμές του διανύσματος ταξινομούνται κατά φθίνουσα τάξη μεγέθους. Παρατηρείστε στο ακόλουθο παράδειγμα πώς συνδυάζουμε τις δύο εντολές για να ταξινομήσουμε απευθείας σε φθίνουσα σειρά ένα διάνυσμα.

```
> height
[1] 1.75 1.84 1.81 1.63
> rev(sort(height))
[1] 1.84 1.81 1.75 1.63
> sort(height)
[1] 1.63 1.75 1.81 1.84                                            ■.
```

Προφανώς, η εφαρμογή της `sort` ή της `rev` δεν επηρεάζει το διάνυσμα, εκτός από την περίπτωση που το αποτέλεσμα των εντολών αυτών έχουν εκχωρηθεί στο ίδιο το διάνυσμα. Οι εντολές αυτές μπορούν εύκολα να χρησιμοποιηθούν και για ταξινόμηση διανυσμάτων-χαρακτήρων.

Θα πρέπει εδώ να τονιστεί ότι η εντολή `sort` μπορεί να χρησιμοποιηθεί για να ταξινομήσει ένα διάνυσμα σε φθίνουσα σειρά χρησιμοποιώντας τη σύνταξη `sort(x, decreasing=T)`. Δηλαδή, το όρισμα `decreasing` είναι αυτό που καθορίζει αν η ταξινόμηση θα γίνει σε αύξουσα ή φθίνουσα σειρά. Επομένως, η χρησιμότητα της εντολής `rev` είναι περισσότερο στην αντιστροφή

της σειράς ενός διανύσματος.

2.9.1 Η εντολή `rank`

Η εντολή `rank` επιστρέφει τη σειρά κατάταξης κάθε παρατήρησης και έχει μεγάλη χρησιμότητα στη στατιστική και στη μη παραμετρική στατιστική, καθώς ένα μεγάλο κομμάτι τους βασίζεται στην ιδέα των ranks.

Για παράδειγμα, οι παρακάτω εντολή μας δίνει τη σειρά κατάταξης των στοιχείων του διανύσματος `x`

```
x<-c(32,43,65,19,90)
rank(x)
[1] 2 3 4 1 5                                                    ■.
```

δηλαδή η παρατήρηση 32 είναι η 2η μικρότερη παρατήρηση, ενώ η παρατήρηση 90 που έχει rank 5 είναι η μεγαλύτερη.

Σε περίπτωση που υπάρχουν όμως ισοπαλίες, δηλαδή παρατηρήσεις με τη ίδια τιμή, τότε η έννοια του rank περιπλέκεται ελαφρά. Η πιο συνηθισμένη προσέγγιση σε αυτήν την περίπτωση είναι να μοιράσουμε εξίσου στις παρατηρήσεις που είναι ίσες τα ranks στα οποία ισοβαθμούν. Για παράδειγμα

```
x<-c(23,65,29,23,50)
rank(x)
[1] 1.5 5.0 3.0 1.5 4.0                                          ■.
```

Στο διάνυσμα `x` υπάρχουν 2 ίσες παρατηρήσεις με την τιμή 23. Σε αυτή την περίπτωση, η σειρά κατάταξης τους είναι 1 και 2 καθώς το 23 είναι η μικρότερη τιμή. Επομένως, δίνουμε rank σε κάθε μια παρατήρηση την τιμή (1+2)/2 δηλαδή 1.5. Η εντολή `rank` στην R η επιτρέπει πέντε διαφορετικούς τρόπους διαχείρισης των ισοπαλιών (ties). Οι διαθέσιμες επιλογές είναι οι ακόλουθες:

- Average: η σειρά κατάταξης είναι η προκαθορισμένη τιμή του μέσου όρου των σειρών κατάταξης, που οι παρατηρήσεις καταλαμβάνουν.
- First: η πρώτη παρατήρηση σε σειρά εμφάνισης παίρνει το μικρότερο rank.
- Random: τα ranks δίνονται τυχαία (παρατήρηση: επομένως με τα ίδια δεδομένα 2 χρήστες μπορεί να έχουν διαφορετικά ranks!).
- Min: σε αντίθεση με την average που δίνει σε όλες τις ίδιες παρατηρήσεις το μέσο όρο τώρα όλες έχουν rank τη μικρότερη τιμή (στην ουσία θυμίζει βαθμολογία ποδοσφαιρικού πρωταθλήματος...).
- Max: όπως και η προηγούμενη αλλά όλες παίρνουν τη μεγαλύτερη τιμή.

Παρατηρήστε στα αποτελέσματα που ακολουθούν τις διαφορές των διαφόρων επιλογών.

```
> x<- c(10,14,13,15,10,20,18,10,23,21,18)
> rank(x, ties.method = "average")
 [1]  2.0  5.0  4.0  6.0  2.0  9.0  7.5  2.0 11.0 10.0  7.5
> rank(x, ties.method = "first")
 [1]  1  5  4  6  2  9  7  3 11 10  8
> rank(x, ties.method = "random")
 [1]  1  5  4  6  2  9  8  3 11 10  7
> rank(x, ties.method = "max")
 [1]  3  5  4  6  3  9  8  3 11 10  8
> rank(x, ties.method = "min")
 [1]  1  5  4  6  1  9  7  1 11 10  7                                    ■.
```

2.9.2 Η εντολή `order`

Η εντολή `order` επιστρέφει τη θέση που βρίσκεται η μικρότερη, η δεύτερη μικρότερη, κοκ παρατήρηση. Για παράδειγμα:

```
> x<-c(32,43,65,19,90)
> order(x)
[1] 4 1 2 3 5                                                           ■,
```

δηλαδή η θέση της μικρότερης παρατήρησης του διανύσματος είναι η τέταρτη (παρατηρείστε ότι όντως το 19 είναι η μικρότερη παρατήρηση, η δεύτερη μικρότερη παρατήρηση είναι στην πρώτη θέση κ.ο.κ.). Η εντολή `order` διαφέρει από την εντολή `rank` από την άποψη ότι αναφέρεται στη θέση της παρατήρησης και όχι στην παρατήρηση καθεαυτή.

Η εντολή `order` είναι αρκετά χρήσιμη όταν θέλουμε να ταξινομήσουμε περισσότερα από ένα διανύσματα ως προς ένα όμως και μόνο διάνυσμα. Για παράδειγμα, έστω 2 διανύσματα `math` και `phys` που περιέχουν τους βαθμούς φοιτητών στα μαθήματα των μαθηματικών και της φυσικής αντίστοιχα. Πώς μπορούμε να ταξινομήσουμε τους φοιτητές ανάλογα με το βαθμό των μαθηματικών και μόνο;

Αν χρησιμοποιήσουμε την εντολή `sort`, τότε ταξινομούμε κάθε διάνυσμα ξεχωριστά και άρα χάνουμε τα ζεύγη παρατηρήσεων. Συνεπώς, αυτό που θέλουμε να πετύχουμε είναι να ταξινομήσουμε ως προς το διάνυσμα `math`, αλλά να μπορέσουμε να βάλουμε τους βαθμούς της μεταβλητής `phys` στις αντίστοιχες θέσεις, ώστε τα ζευγάρια να παραμείνουν τα ίδια. Αυτό μας επιτρέπει να το κάνουμε η εντολή `order`. Έτσι, έχουμε τις παρακάτω εντολές:

```
> math<-c(30,56,78,54,90)
> phys<-c(65,98,45,67,87)
> t<-order(math)
> math<-sort(math)
> phys<-phys[t]
> math
```

```
[1] 30 54 56 78 90
> phys
[1] 65 67 98 45 87                                                    ■.
```

Προσοχή: αν ταξινομήσουμε πρώτα τα μαθηματικά και έπειτα πάρουμε το order τους, τότε θα χάσουμε τη σχέση με τους αντίστοιχους βαθμούς της φυσικής.

2.10 Συναρτήσεις `any` και `all`

Ο συναρτήσεις `any` και `all` εφαρμόζονται σε ένα διάνυσμα και επιστρέφουν μια τιμή τύπου Boolean (`TRUE` ή `FALSE`) ανάλογα με το αν κάποιο στοιχείο `any` ή όλα τα στοιχεία `all` του διανύσματος ικανοποιούν τη συνθήκη. Για παράδειγμα, η εντολή `any(x>0)` εξετάζει αν υπάρχει τουλάχιστον ένα στοιχείο του διανύσματος `x` που να είναι θετικό. Αν υπάρχει, τουλάχιστον ένα στοιχείο θετικό, θα πάρει την τιμή `TRUE`, ενώ αν όχι την τιμή `FALSE` . Αντίστοιχα, η εντολή `all(x>0)` θα γίνει `TRUE`, αν όλα τα στοιχεία του `x` είναι θετικά. Οι εντολές αυτές μας βοηθάνε να ελέγξουμε απλά και γρήγορα αν ικανοποιούνται κάποιες συνθήκες μέσα σε ένα διάνυσμα.

2.11 Παραδείγματα

2.11.1 Μέγιστος Κοινός Διαιρέτης

Να γραφούν εντολές, οι οποίες για δοθείσες τιμές των ακέραιων αριθμών `a` και `b` να υπολογίζουν το μέγιστο κοινό διαιρέτη.

Ουσιαστικά, αυτό που πρέπει να κάνουμε είναι να βρούμε όλους τους διαιρέτες των δύο αριθμών και επομένως να βρούμε το μέγιστο κοινό (αν υπάρχει κοινός).

Είναι λογικό ο μέγιστος κοινός διαιρέτης δύο αριθμών `a` και `b` να είναι ένας αριθμός από το 1 εώς και το μέγιστο των `a` και `b`. Επομένως, μια απλή ιδέα είναι να δούμε όλους τους αριθμούς από το 1 εως το το μέγιστο των `a` και `b`, αν είναι διαιρέτες και των δύο και στη συνέχεια από τους κοινούς διαιρέτες να βρούμε το μέγιστο. Αυτό κάνουν οι παρακάτω εντολές:

```
> a<-10
> b<-12
> c<-max(a,b)
> test<- 1:c
> y1<-  a%%  test==0
> y2<-  b%%  test ==0
> y3<- y1&y2
> max( test[y3])                                                      ■.
```

Ουσιαστικά, το `c` είναι το μέγιστο των `a` και `b`, το διάνυσμα `test` περιέχει όλους τους αριθμούς από το 1 έως και το `c` και τα διανύσματα `y1` και `y2` δείχνουν αν ο αριθμός διαιρεί το `a` και `b`. Το `y3` απλά δείχνει αν είναι κοινός διαιρέτης. Τέλος, η τελευταία εντολή μας επιστρέφει το μέγιστο κοινό διαιρέτη.

Στην παραπάνω λύση, μπορεί να παρατηρήσει κανείς ότι δεν υπάρχει λόγος να ελέγξουμε τα στοιχεία από το 1 ως και το μέγιστο αλλά ως και το ελάχιστο. Αυτό, ουσιαστικά, περιορίζει αρκετά τις συγκρίσεις που θέλουμε να κάνουμε. Επίσης, θα μπορούσε κανείς αν δει ότι, αφού έχουμε εξετάσει για τους διαιρέτες του `a`, στη συνέχεια μπορούμε να εξετάσουμε ως ενδιαφέροντες διαιρέτες για το `b` μόνο όσοι είναι ήδη διαιρέτες του `a` κλπ. Γενικά, για ένα πρόγραμμα συνήθως υπάρχουν μικρά πράγματα που μπορούν να το βελτιώσουν. Μερικές φορές, ο παραπάνω προγραμματιστικός κόπος δεν εξισορροπεί το κέρδος από αυτή τη βελτίωση!

2.11.2 Περικομμένος μέσος

Ο περικομμένος μέσος (trimmed mean) είναι ένα εναλλακτικό περιγραφικό μέτρο θέσης αντί για τον απλό αριθμητικό μέσο. Η χρήση του βασίζεται στην ιδέα να αγνοήσουμε κάποιες παρατηρήσεις και συγκεκριμένα παρατηρήσεις σε κάθε ουρά της κατανομής, όπου ουσιαστικά μπορεί να υπάρχουν outliers, τα οποία ως γνωστόν επιδρούν αρκετά στον αριθμητικό μέσο. Έτσι, στον περικομμένο μέσο αγνοούμε ένα ποσοστό $a\%$ από κάθε ουρά και στη συνέχεια υπολογίζουμε τη μέση τιμή μόνο με τις υπόλοιπες παρατηρήσεις. Να γράψετε εντολές στην R που να υπολογίζουν τον 5% περικομμένο μέσο για ένα διάνυσμα παρατηρήσεων `x`.

Υπάρχουν περισσότεροι από ένας τρόποι για να λύσουμε το πρόβλημα. Θα δούμε 2 προσεγγίσεις αν και θα μπορούσαμε να δοκιμάσουμε περισσότερες. Αλγοριθμικά αυτό που θέλουμε είναι:

1. Ταξινόμησε τις παρατηρήσεις σε αύξουσα σειρά,
2. Βρες ποιος είναι ο αριθμός των παρατηρήσεων που πρέπει να αγνοήσουμε σε κάθε ουρά,
3. Χρησιμοποίησε μόνο τις υπόλοιπες παρατηρήσεις για να υπολογίσεις τη μέση τιμή.

Έστω το διάνυσμα `x`, στο οποίο έχουμε αποθηκεύσει 20 τιμές και έστω ότι ενδιαφερόμαστε για τον 5% περικομμένο μέσο.

```
> x<-c(51, 46, 58, 30, 49, 49, 34, 49, 62, 53, 45, 50,
       46, 42, 46, 51, 48, 48, 45, 45)                    ■.
```

Χρησιμοποιώντας εντολές της R ο αλγόριθμος υλοποιείται ως εξής:

```
> y<-sort(x)
> n<-length(x)
> n1<- trunc(n*0.05)
> mean(y[(n1+1): (n-n1)])                                 ■.
```

Η εντολή `trunc` επιστρέφει το ακέραιο μέρος του αριθμού που ακολουθεί. Δηλαδή, για παράδειγμα, το `trunk(5.4)` είναι το 5. Επομένως, μετράει πόσες παρατηρήσεις πρέπει να αγνοήσω σε κάθε πλευρά. Άρα αν πρέπει να αγνοήσω `n1` παρατηρήσεις, πρέπει να λάβω υπόψη μου τις παρατηρήσεις από τη θέση `n1+1`, δηλαδή την επόμενη από αυτές που έχω περικόψει μέχρι

τη θέση `n-n1`, γιατί αυτό μου εξασφαλίζει ότι θα αγνοήσω τις `n1` μεγαλύτερες παρατηρήσεις. Προσοχή, θα πρέπει να χρησιμοποιήσω το ταξινομημένο διάνυσμα παρατηρήσεων και όχι το αρχικό, αφού εκεί οι παρατηρήσεις είναι τοποθετημένες κατά αύξουσα σειρά. Επίσης, ιδιαίτερη προσοχή χρειάζεται στις παρενθέσεις `(n1+1): (n-n1)`, γιατί αν δεν τις βάλω ο υπολογιστής θα λάβει υπόψη του άσχετες παρατηρήσεις. Δοκιμάστε το αποτέλεσμα της εντολής

```
> n1+1:n-n1
[1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

και για τα δεδομένα μας θα πάρετε τους αριθμούς 1 έως και 20 όπως φαίνεται παραπάνω.

Εναλλακτικά, μπορώ να χρησιμοποιήσω την εντολή `rank`, η οποία μου δίνει τη σειρά κατάταξης των παρατηρήσεων. Η ιδέα είναι να αγνοήσω παρατηρήσεις με πολύ μικρό ή πολύ μεγάλο rank, ανάλογα με το ποσοστό των παρατηρήσεων που θέλω. Οι εντολές είναι τώρα οι εξής:

```
> y<-rank(x)
> n<-length(x)
> n1<- trunc(n*0.05)
> t<- (y>n1)&(y<=n-n1)
> mean(x[t])                                                    ■,
```

δηλαδή για τα δεδομένα μου βρίσκω τη σειρά κατάταξης των παρατηρήσεων και ουσιαστικά βρίσκω τον μέσο όσων παρατηρήσεων έχουν rank μεγαλύτερο του 5 και μικρότερο ή ίσο του 95.

Η R επιτρέπει τον υπολογισμό του περικομμένου μέσου χρησιμοποιώντας την παράμετρο `trim` στην εντολή `mean`. Έτσι η σύνταξη `mean(x,trim=0.05)` μας δίνει το 5% περικομμένο μέσο. Η παράμετρος `trim` καθορίζει το ποσοστό των παρατηρήσεων που δε θα λάβουμε υπόψη μας. Με την επιλογή `trim=0.05` υπολογίζουμε το μέσο αφού αφαιρέσουμε το 5% των μικρότερων και το 5% των μεγαλύτερων τιμών.

2.11.3 Διάμεσος

Για τον υπολογισμό της διαμέσου ουσιαστικά χρειάζεται να ταξινομήσουμε τις παρατηρήσεις σε αύξουσα σειρά και στη συνέχεια να επιλέξουμε ανάλογα με το μέγεθος του δείγματος (άρτιος ή περιττός αριθμός) τις ταξινομημένες παρατηρήσεις που θα χρησιμοποιήσουμε. Δεδομένου ότι, προς το παρόν δεν έχουμε διαθέσιμες τις εντολές `if`, οι παρακάτω εντολές υπολογίζουν τη διάμεσο για ένα διάνυσμα `x`.

```
> y<-sort(x)
> n<-length(x)
> artios<- (n%%2 ==0)
> diamesos<-  artios * (y[ n/2] + y[n/2 +1])/2  + !artios * y[(n
 +1)/2]                                                         ■.
```

Θα πρέπει εδώ να τονίσουμε το εξής: στην τελευταία εντολή ουσιαστικά υλοποιούνται και τα 2 μέρη του αθροίσματος. Επομένως, σε κάθε περίπτωση υπολογίζουμε κάποια θέση το

`y[n/2]` ή το `y[(n+1)/2]` που ο δείκτης δεν είναι ακέραια τιμή. Σε αυτή την περίπτωση η R στρογγυλοποιεί χρησιμοποιώντας το ακέραιο μέρος. Για παράδειγμα, το `y[1.3]` στον υπολογιστή είναι ισοδύναμο με το `y[1]`.

2.12 Διανύσματα Χαρακτήρων

Ως διανύσματα χαρακτήρων ορίζουμε κάθε διάνυσμα, όπου κάθε στοιχείο του αποτελείται από μια σειρά χαρακτήρων (δηλαδή ονομάτων). Για παράδειγμα:

```
> x<-c( 'Yiannis', 'Yiorgos', 'Grigoris' )
> x
[1] "Yiannis"  "Yiorgos"  "Grigoris"
> x[2]
[1] "Yiorgos"                                                    ■.
```

Βασικές εντολές δημιουργίας και ελέγχου των διανυσμάτων χαρακτήρων είναι οι ακόλουθες:

- `character(length = 0)` : Δημιουργεί ένα άδειο διάνυσμα χαρακτήρων μήκους ίσο με `length`.
- `as.character(x, ...)` : Μετατρέπει το διάνυσμα `x` σε διάνυσμα χαρακτήρων.
- `is.character(x)` : Ελέγχει αν το διάνυσμα `x` είναι διάνυσμα χαρακτήρων.

Επίσης, χρήσιμες είναι και οι εντολές `noquote` και `nchar`. Η πρώτη εντολή απλά τυπώνει τις σειρές χαρακτήρων χωρίς τα εισαγωγικά, ενώ η δεύτερη δίνει ένα διάνυσμα ίδιου μήκους με το διάνυσμα χαρακτήρων που εισάγουμε, όπου κάθε στοιχείο έχει τον αριθμό των χαρακτήρων του αντίστοιχου διανύσματος εισαγωγής.
Παραδείγματα:

```
> xsurnames<-c('Papadopoulos','Kitsos','Ioannou', 'Tsapara',
+  'Grigoriadou')
> xsurnames
[1] "Papadopoulos" "Kitsos"     "Ioannou"     "Tsapara"
[5] "Grigoriadou"
> nchar(xsurnames)
[1] 12  6  7  7 11                                               ■.
```

2.12.1 Συνένωση Χαρακτήρων (εντολή `paste`).

Μια από τις βασικές εντολές διαχείρισης χαρακτήρων είναι η εντολή `paste`, η οποία μετατρέπει διανύσματα `vector1, ..., vectorn` σε διανύσματα και τα συνδέει μεταξύ τους. Η σύνταξή της είναι η ακόλουθη:

```
paste( vector1, ..., vectorn, sep = " ", collapse = NULL)       ■.
```

Το όρισμα `sep` ελέγχει τους χαρακτήρες τους οποίους τοποθετεί ενδιάμεσα των χαρακτήρων που συνενώνει η εντολή `paste` η default τιμή του είναι το κενό. Το όρισμα `collapse` ελέγχει αν όλα τα στοιχεία του διανύσματος πρέπει να συνενωθούν σε μία σειρά χαρακτήρων (δηλαδή σε διάνυσμα χαρακτήρων με μήκος ένα). Όταν το όρισμα δίνεται ίσο με ένα χαρακτήρα, τότε όλα τα στοιχεία των διανυσμάτων συνενώνονται σε μία σειρά χαρακτήρων και διαχωρίζονται μεταξύ τους με το χαρακτήρα που ορίζουμε εδώ. Αν αυτή το όρισμα είναι ίσο με `NULL` (προεπιλεγμένη τιμή), τότε η συνένωση γίνεται μόνο μεταξύ των αντίστοιχων στοιχείων του διανύσματος.

Οι γενικές χρήσεις της εντολής μπορεί να είναι οι ακόλουθες:

- **Συνένωση δύο ονομάτων (σειρών χαρακτήρων)**: Για παράδειγμα

```
> x1<-'Yiannis'
> x2<-'Nicolaou'
> paste(x1,x2)
[1] "Yiannis Nicolaou"                                    ■.
```

  Το παραπάνω αποτέλεσμα είναι ακριβώς ισοδύναμο με

```
> paste('Yiannis','Nicolaou')
[1] "Yiannis Nicolaou"                                    ■.
```

  Αν ο διαχωρισμός των χαρακτήρων θέλουμε να γίνει με διαφορετικούς χαρακτήρες, τότε μπορούμε να χρησιμοποιήσουμε το όρισμα `sep`. Έτσι, για παράδειγμα:

```
> x1<-5
> x2<-13
> paste(x1,x2, x1+x2, sep=' + ')
[1] "5 + 13 + 18"                                         ■.
```

  Αν θέλουμε όμως να έχουμε διαφορετικούς χαρακτήρες συνένωσης, τότε θα πρέπει να χρησιμοποιήσουμε δύο φορές την εντολή `paste`. Έτσι, η παρακάτω συνένωση είναι πιο ορθή:

```
> paste(paste(x1,x2, sep=' + '), x1+x2, sep=' = ')
[1] "5 + 13 = 18"                                         ■.
```

- **Συνένωση των στοιχείων δύο (ή περισσοτέρων) διανυσμάτων χαρακτήρων**: Για παράδειγμα, έχουμε δύο διανύσματα με τα μικρά ονόματα (`xnames`) και τα επώνυμα (`xsurnames`) από πέντε άτομα και επιθυμούμε να φτιάξουμε ένα καινούριο διάνυσμα με τα πλήρη ονοματεπώνυμα. Αυτό μπορεί να γίνει με τις ακόλουθες εντολές:

```
> xnames<-c('Yiannis','Yiorgos','Barbara','Aleka',
+  'Eugenia')
>  xsurnames<-c('Papadopoulos','Kitsos','Ioannou',
```

```
+  'Tsapara','Grigoriadou')
>  paste(xnames,xsurnames)
[1] "Yiannis Papadopoulos" "Yiorgos Kitsos"
[3] "Barbara Ioannou"      "Aleka Tsapara"
[5] "Eugenia Grigoriadou"                                    ■.
```

Εάν επιθυμούσαμε η συνένωση να γίνει με το επώνυμο πρώτα και μετά το μικρό όνομα διαχωρισμένα με κόμμα και μετά κενό, τότε πρέπει να γράψουμε:

```
>  paste(xsurnames,xnames,  sep=',  ')
[1] "Papadopoulos,  Yiannis" "Kitsos,  Yiorgos"
[3] "Ioannou,  Barbara"      "Tsapara,  Aleka"
[5] "Grigoriadou,  Eugenia"                                  ■.
```

Στην περίπτωση που τα διανύσματα δεν έχουν ίδιο μήκος, τότε το μικρότερο διάνυσμα ανακυκλώνει τα στοιχεία του μέχρι να φτάσει το μήκος του μεγαλύτερου. Έτσι, για παράδειγμα, αν στο παραπάνω παράδειγμα δώσουμε μόνο τα 3 επίθετα, τότε στο 4ο και 5ο στοιχείο θα χρησιμοποιηθούν τα δύο πρώτα επίθετα, π.χ.

```
> paste(xsurnames[1:3],xnames,  sep=',  ')
[1] "Papadopoulos,  Yiannis" "Kitsos,  Yiorgos"
[3] "Ioannou,  Barbara"      "Papadopoulos,  Aleka"
[5] "Kitsos,  Eugenia"                                       ■.
```

- **Συνένωση στοιχείων ενός διανύσματος χαρακτήρων σε μία σειρά χαρακτήρων**: Αν για οποιοδήποτε λόγο επιθυμούμε τη συνένωση των στοιχείων ενός διανύσματος σε μία σειρά χαρακτήρων, τότε πρέπει να ενεργοποιήσουμε την παράμετρο `collapse`, στην οποία πρέπει να δώσουμε το χαρακτήρα διαχωρισμού των στοιχείων. Έτσι, για παράδειγμα η εντολή:

```
> paste(1:10,  collapse='  ')
```

θα μας δώσει

```
[1] "1  2  3  4  5  6  7  8  9  10"                          ■.
```

Αντίστοιχα παραδείγματα είναι τα ακόλουθα:

```
> paste(1:10,  collapse='')
[1] "12345678910"
> paste(1:10,  collapse='+')
[1] "1+2+3+4+5+6+7+8+9+10"
> paste(xnames,  collapse=',  ')
[1] "Yiannis,  Yiorgos,  Barbara,  Aleka,  Eugenia"          ■.
```

- **Συνένωση στοιχείων δύο (ή περισσοτέρων) διανυσμάτων χαρακτήρων σε μία σειρά χαρακτήρων**: Αν για οποιοδήποτε λόγο επιθυμούμε τη συνένωση των στοιχείων πολλών διανυσμάτων, τότε θα πρέπει να θυμόμαστε ότι η συνάρτηση `paste` πρώτα ενώνει τα αντίστοιχα στοιχεία των διανυσμάτων και μετά τα ενώνει σε μία σειρά δεδομένων. Έτσι, η εντολή

```
> paste(xnames, xsurnames, collapse=',')
```

δίνει ως αποτέλεσμα

```
[1]  "Yiannis Papadopoulos,Yiorgos Kitsos,Barbara
Ioannou,Aleka Tsapara,Eugenia Grigoriadou"
```

το οποίο είναι ισοδύναμο με

```
> x<-paste(xnames, xsurnames)
> paste(x,collapse=',')
[1] "Yiannis Papadopoulos,Yiorgos Kitsos,Barbara
Ioannou,Aleka Tsapara,Eugenia Grigoriadou"
```
■.

Στην περίπτωση που επιθυμούμε να συνενώσουμε τα στοιχεία των δύο διανυσμάτων σε ένα διάνυσμα, τότε δημιουργούμε ένα καινούργιο διάνυσμα `c`, με ορίσματα τα ονόματα των δύο διανυσμάτων και το τοποθετούμε στην εντολή `paste`. Έτσι, θα έχουμε στο παραπάνω παράδειγμα:

```
> paste( c(xnames, xsurnames), collapse=',')
[1] "Yiannis,Yiorgos,Barbara,Aleka,Eugenia,
Papadopoulos,Kitsos,Ioannou,Tsapara,Grigoriadou"
```
■.

Εδώ πρέπει να αναφέρουμε ότι όταν χρησιμοποιήσουμε την εντολή `paste` με όρισμα ένα αριθμητικό μόνο διάνυσμα, τότε η μόνη επίδραση που θα έχει θα είναι να μετατρέψει το διάνυσμα αυτό σε διάνυσμα χαρακτήρων. Συνεπώς, στην περίπτωση αυτή θα έχει ακριβώς το ίδιο αποτέλεσμα όπως η εντολή `as.character`.
Παράδειγμα:

```
> paste(1:12)
 [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10"
[11] "11" "12"
> as.character(1:12)
 [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10"
[11] "11" "12"
```
■.

Επίσης, είναι δυνατή η χρήση μικτών δεδομένων στην εντολή `paste` π.χ. κάποια δεδομένα να είναι χαρακτήρες, κάποια αριθμητικά,ακόμα και έτοιμες συναρτήσεις στην R, εφόσον η εντολή τα μετατρέπει όλα πρώτα σε χαρακτήρες. Έτσι, για παράδειγμα έχουμε:

```
>  paste("A", 1:6, sep = "")
[1] "A1" "A2" "A3" "A4" "A5" "A6"
>  paste("Individual", 1:6, sep = " ")
[1] "Individual 1" "Individual 2" "Individual 3"
[4] "Individual 4" "Individual 5" "Individual 6"
> date()
[1] "Fri Oct 02 11:46:15 2015"
> paste("Today is", date())
[1] "Today is Fri Oct 02 11:46:15 2015"                    ■.
```

2.12.2 Διαχωρισμός Χαρακτήρων (εντολή `strsplit`).

Η εντολή αυτή διαχωρίζει τα στοιχεία ενός διανύσματος χαρακτήρων σε μικρότερες σειρές χαρακτήρων, ανάλογα με κάποιο διαχωριστικό χαρακτήρα. Στην πράξη, είναι η αντίστροφη συνάρτηση της εντολής `paste` που περιγράψαμε παραπάνω. Η σύνταξή της είναι η ακόλουθη:

```
strsplit(x, split, extended = TRUE, fixed = FALSE, perl = FALSE) ■.
```

Οι παράμετροι της συνάρτησης είναι οι ακόλουθες:

x : Διάνυσμα χαρακτήρων το οποίο θα σπάσει σε μικρότερες σειρές χαρακτήρων.

split : Διάνυσμα χαρακτήρων που περιέχει τους χαρακτήρες με τους οποίους γίνεται ο διαχωρισμός. Όταν αυτό έχει μήκος ένα, τότε διαχωρίζουμε κάθε στοιχείο του `x` σύμφωνα με το `split`. Αν το διάνυσμα αυτό έχει μήκος μεγαλύτερο του ένα, τότε ο διαχωρισμός κάθε στοιχείου του `x` γίνεται σύμφωνα με το αντίστοιχο στοιχείο του `split`. Αν στην περίπτωση αυτή το διάνυσμα `split` έχει μικρότερο μήκος από το `x`, τότε τα στοιχεία του `split` ανακυκλώνονται. Όταν σε μία σειρά χαρακτήρων εντοπιστεί ο χαρακτήρας διαχωρισμού αυτός σπάει στο σημείο αυτό σε δύο (ή περισσότερες) μικρότερες, π.χ. η εντολή

```
> strsplit('Yiannis','i')
```

θα δώσει ως αποτέλεσμα

```
[[1]]
[1] "Y"    "ann" "s"                                     ■.
```

Στην περίπτωση που το διάνυσμα αυτό το θέσουμε ίσο με το μηδενικό στοιχείο (δηλαδή `split=NULL`), τότε όλοι οι χαρακτήρες διαχωρίζονται μεταξύ τους, π.χ.

```
>  strsplit('Yiannis',NULL)
```

θα δώσει ως αποτέλεσμα

```
 [[1]]
 [1] "Y" "i" "a" "n" "n" "i" "s"                          ■.
```

Παρόμοιο αποτέλεσμα παράγει και η εντολή

```
> strsplit('Yiannis',split='')
```
■.

Ένα απλό παράδειγμα είναι το ακόλουθο:

```
> x<-paste( xnames, xsurnames, sep=', ')
> x
[1] "Yiannis, Papadopoulos" "Yiorgos, Kitsos"
[3] "Barbara, Ioannou"      "Aleka, Tsapara"
[5] "Eugenia, Grigoriadou"
> strsplit(x,split=',')
[[1]]
[1] "Yiannis"       " Papadopoulos"

[[2]]
[1] "Yiorgos" " Kitsos"

[[3]]
[1] "Barbara"   " Ioannou"

[[4]]
[1] "Aleka"    " Tsapara"

[[5]]
[1] "Eugenia"      " Grigoriadou"
```
■.

Να σημειώσουμε ότι το αποτέλεσμα της εντολής `strsplit` είναι τύπου λίστας, το οποίο θα περιγράψουμε αναλυτικά παρακάτω. Στην περίπτωση που θέλουμε το αποτέλεσμα να δίνεται ως διάνυσμα, τότε πρέπει να χρησιμοποιήσουμε την εντολή `unlist`, το οποίο θα μας δώσει

```
> unlist(strsplit(x,split=','))
 [1] "Yiannis"       " Papadopoulos"
 [3] "Yiorgos"       " Kitsos"
 [5] "Barbara"       " Ioannou"
 [7] "Aleka"         " Tsapara"
 [9] "Eugenia"       " Grigoriadou"
```
■.

Αν θέλουμε να τυπώσουμε ένα κείμενο με κενό μεταξύ των στοιχείων, τότε μπορούμε να χρησιμοποιήσουμε τις εντολές `noquote` και `strplit( ..., split=NULL)`. Έτσι, η ακόλουθη εντολή

```
> noquote(strsplit("A text I want to display with
+ spaces", split=NULL)[[1]])
```

(ένας τρόπος αναφοράς σε επιμέρους στοιχείο μίας λίστας, την οποία θα δούμε αναλυτικά στο Κεφάλαιο 3, είναι η χρήση της διπλής αγκύλης), δίνει ως αποτέλεσμα:

```
 [1] A   t e x t   I   w a n t   t o   d i s p l a y
[26] w i t h   s p a c e s                                    ■.
```

Ας υποθέσουμε ότι θέλουμε να χωρίσουμε τη σειρά χαρακτήρων `"a.b.c"`. Τότε, με βάση τα παραπάνω, αρκεί να γράψουμε

```
> strsplit("a.b.c", ".")
```

το οποίο όμως παράγει ως αποτέλεσμα

```
[[1]]
[1] "" "" "" "" ""                                           ■.
```

Ο λόγος είναι ότι μερικοί χαρακτήρες όταν χρησιμοποιούνται στην παράμετρο `split` είναι δεσμευμένοι για ειδικές λειτουργίες. Οι χαρακτήρες αυτοί ονομάζονται και regular expressions (βλέπε `help(regexp)` για λεπτομέρειες). Στην περίπτωση αυτή, το επιθυμητό αποτέλεσμα θα επιτευχθεί αν γράψουμε:

```
> strsplit("a.b.c", "\\.")
[[1]]
[1] "a" "b" "c"                                              ■,
```

ή εναλλακτικά με τη χρήση της παραμέτρου `fixed`:

```
> strsplit("a.b.c", ".", fixed=TRUE)
[[1]]
[1] "a" "b" "c"                                              ■.
```

2.12.3 Εξαγωγή και αντικατάσταση κομματιών μίας σειράς χαρακτήρων (εντολές *substr* και *substring*).

Στην παράγραφο αυτή, θα περιγράψουμε τις εντολές με τις οποίες αποσπούμε ή αντικαθιστούμε μέρη μιας σειράς δεδομένων χαρακτήρων με τις εντολές `substr` και `substring`. Η εντολή `substr` έχει τη σύνταξη

```
substr(x, start, stop)
```

με τις εξής παραμέτρους:

x : Ένα διάνυσμα χαρακτήρων.

start : Ακέραιος χαρακτήρας που υποδεικνύει τη θέση του χαρακτήρα που θα ξεκινήσει το κομμάτι της σειράς χαρακτήρων που θέλουμε να αποσπάσουμε.

stop : Ακέραιος χαρακτήρας που υποδεικνύει τη θέση του τελευταίου (σε σειρά) χαρακτήρα από το οποίο θα ξεκινήσει το κομμάτι της σειράς χαρακτήρων που θέλουμε να αποσπάσουμε.

Συνεπώς, η εντολή `substr('Yiannis', 2, 4)` θα μας δώσει ως αποτέλεσμα μια σειρά χαρακτήρων που θα ξεκινάει από το 2ο και θα σταματάει στο 4ο χαρακτήρα του ονόματος `'Yiannis'` και συνεπώς το αποτέλεσμα θα είναι `"ian"`. Παρόμοιο είναι το αποτέλεσμα και στην περίπτωση που εισάγουμε ένα διάνυσμα χαρακτήρων, π.χ.

```
> xnames<-c('Yiannis','Yiorgos','Barbara','Aleka',
+   'Eugenia')
> substr(xnames,3,7)
[1] "annis" "orgos" "rbara" "eka"   "genia"                    ■.
```

Παρόμοια είναι και εντολή `substring`, η οποία έχει σύνταξη

```
          substring(text, first, last = 1000000)               ■,
```

όπου οι παράμετροι `text`, `first` και `last` αντιστοιχούν στις παραμέτρους `x`, `start` και `stop` της συνάρτησης `substr`. Μία διαφορά είναι, ότι στην εντολή `substring` δεν χρειάζεται να ορίσουμε τη θέση του τελευταίου χαρακτήρα. Στην περίπτωση που δε δώσουμε τη θέση του χαρακτήρα που θα σταματήσει η εξαγωγή, τότε φτάνει μέχρι το τέλος της σειράς χαρακτήρων. Μερικά παραδείγματα είναι τα ακόλουθα:

```
> substring('Yiannis',2,4)
[1] "ian"
> substring('Yiannis',2)
[1] "iannis"
> substring(xnames,3,5)
[1] "ann" "org" "rba" "eka" "gen"
> substring(xnames,3)
[1] "annis" "orgos" "rbara" "eka"   "genia"                    ■.
```

Στην παραπάνω σύνταξη οι παράμετροι `first`, `last`, `start` και `stop` μπορεί να είναι και διανύσματα. Στην περίπτωση αυτή, μπορούμε να εξάγουμε διαφορετικά κομμάτια για κάθε στοιχείο του διανύσματος χαρακτήρων που εισάγουμε.

Στην περίπτωση που επιθυμούμε αντικατάσταση των κομματιών αυτών με άλλη σειρά χαρακτήρων, μπορούμε να χρησιμοποιήσουμε μια από τις ακόλουθες εντολές:

```
      substr(x, start, stop) <- value
      substring(text, first, last = 1000000) <- value          ■.
```

Για παράδειγμα, αν στο όνομα «Yiannis» θέλουμε να αντικαταστίσουμε τους χαρακτήρες 2 έως 4 με μια άλλη σειρά χαρακτήρων (π.χ. «test»), τότε αρκεί να γράψουμε τις ακόλουθες εντολές

```
> x<-'Yiannis'
> substr(x,2,4)<-'test'
```

```
> x
[1] "Ytesnis"                                                    ■.
```

Προσοχή, στο παραπάνω παράδειγμα, τελικά αντικαταστάθηκαν οι χαρακτήρες «ian» με τους 3 πρώτους χαρακτήρες του «test» (δηλαδή «tes») που χωρούσαν στη θέση των αρχικών χαρακτήρων που αντικαταστήσαμε.

Με παρόμοιο τρόπο, μπορούμε να αλλάξουμε το «Yiannis» σε «Yianna» με την εντολή

```
> x<-'Yiannis'
> substr(x,6,7)<-'a '
> x
[1] "Yianna "                                                    ■.
```

Αν στο παραπάνω θέλουμε να αφαιρέσουμε τον τελευταίο κενό χαρακτήρα, τότε θα πρέπει να ξαναχρησιμοποιήσουμε την εντολή `substr`, δηλαδή να γράψουμε

```
> x<-substr(x,1,6)
> x
[1] "Yianna"                                                     ■.
```

Όταν χρησιμοποιούμε διανύσματα, μπορούμε να αλλάξουμε κάθε στοιχείο του με διαφορετικούς χαρακτήρες ή σε διαφορετική σειρά, π.χ.

```
> test<-xnames
> substr(test,1,1)<-as.character(1:5)
> test
[1] "1iannis" "2iorgos" "3arbara" "4leka"   "5ugenia"
>
> test<-xnames
> substring(test,1:5)<-as.character(1:5)
> test
[1] "1iannis" "Y2orgos" "Ba3bara" "Ale4a"   "Euge5ia"            ■.
```

Μια σημαντική διαφορά μεταξύ των εντολών `substr` και `substring` είναι ότι η πρώτη δίνει πάντα ως αποτέλεσμα ένα διάνυσμα με μήκος ίσο με αυτό του διανύσματος χαρακτήρων `x` που εισάγουμε, ενώ το αποτέλεσμα της δεύτερης είναι ένα διάνυσμα χαρακτήρων με μήκος ίσο με το μεγαλύτερο μήκος όλων των παραμέτρων. Έτσι, για παράδειγμα έχουμε:

```
>   substr("abcdef",1:6,1:6)
[1] "a"
>   substring("abcdef",1:6,1:6)
[1] "a" "b" "c" "d" "e" "f"                                      ■.
```

2.12.4 Εύρεση και αντικατάσταση χαρακτήρων σε διανύσματα χαρακτήρων

Σε πολλές περιπτώσεις μας ενδιαφέρει να εξετάσουμε ποια στοιχεία ενός διανύσματος δεδομένων περιέχουν συγκεκριμένες σειρές χαρακτήρων. Για παράδειγμα, αν έχουμε ένα διάνυσμα με την ονομασία `fnames`, που περιέχει τα ονοματεπώνυμα των πελατών μιας εταιρείας, μπορεί να μας ενδιαφέρει να βρούμε τους πελάτες, των οποίων το όνομα είναι «Κώστας» ή «Κωνσταντίνος», έτσι ώστε να τους στείλουμε μια ευχετήρια κάρτα στην ονομαστική τους εορτή. Αυτό μπορεί να επιτευχθεί με τη χρήση των εντολών `grep` και `regexpr`, με τη δεύτερη να δίνει πιο αναλυτικά αποτελέσματα. Αντίστοιχα, οι εντολές `sub` και `gsub` αντικαθιστούν τα διανύσματα που ταιριάζουν με αυτά που δίνουμε με κάποια νέα.

Εντολές `grep` και `regexpr`.

Όπως προαναφέραμε, σκοπός των εντολών αυτών είναι ο έλεγχος και ο εντοπισμός συγκεκριμένων σειρών χαρακτήρων μέσα σε διανύσματα χαρακτήρων. Ο γενικός τρόπος σύνταξης των εντολών αυτών είναι ο ακόλουθος:

```
grep(pattern, x, ignore.case = FALSE, extended = TRUE,
     perl = FALSE, value = FALSE, fixed = FALSE)
regexpr(pattern, text,  extended = TRUE, perl = FALSE,
     fixed = FALSE)
```

με παραμέτρους

pattern : Σειρά χαρακτήρων που θέλουμε να ταιριάξουμε ή να ψάξουμε.

x, text : Διάνυσμα χαρακτήρων μέσα στο οποίο θα ψάξουμε για τη σειρά χαρακτήρων `pattern`.

ignore.case : Λογική παράμετρος με «ψευδή» (`FALSE`) προεπιλεγμένη τιμή. Όταν είναι ψευδής (`FALSE`), τότε κατά τη διαδικασία της εύρεσης λαμβάνονται υπόψη οι διαφοροποιήσεις μεταξύ μικρών και κεφαλαίων γραμμάτων, διαφορετικά (`TRUE`) δε λαμβάνονται υπόψη (δηλαδή για παράδειγμα «a» και «A» θεωρούνται ίδιοι χαρακτήρες).

extended : Λογική παράμετρος με «αληθή» (`TRUE`) προεπιλεγμένη τιμή. Όταν είναι αληθής (`TRUE`), τότε χρησιμοποιούνται εκτεταμένες εκφράσεις σύγκρισης (extended regular expression matching), διαφορετικά (`FALSE`), τότε βασικές εκφράσεις σύγκρισης (basic regular expressions) χρησιμοποιούνται.

perl : Λογική παράμετρος με «ψευδή» (`FALSE`) προεπιλεγμένη τιμή. Χρησιμοποιούνται εκφράσεις σύγκρισης συμβατές με το σύστημα perl. Η παράμετρος αυτή έχει προτεραιότητα επί της παραμέτρου `extended`.

value : Λογική παράμετρος με «ψευδή» (`FALSE`) προεπιλεγμένη τιμή. Όταν είναι ψευδής (`FALSE`), τότε ως αποτέλεσμα επιστρέφονται οι θέσεις των στοιχείων του διανύσματος, όπου εντοπίστηκε η σειρά χαρακτήρων pattern (π.χ. στο 1ο στοιχείο, στο 3ο κλπ.). Σε αντίθετη περίπτωση, δίνονται τα ίδια τα στοιχεία του διανύσματος `x`.

fixed : Λογική παράμετρος με «ψευδή» (`FALSE`) προεπιλεγμένη τιμή. Όταν η τιμή της παραμέτρου `fixed` είναι αληθής (`TRUE`), τότε στην παράμετρο `pattern` δίνονται μόνο χαρακτήρες και όχι λογικές ή βασικές εκφράσεις σύγκρισης. Αυτή η παράμετρος έχει προτεραιότητα επί των παραμέτρων `extended` και `perl`.

replacement : Σειρά χαρακτήρων, οι οποίοι θα αντικαταστήσουν τα σημεία όπου θα βρεθεί η σειρά χαρακτήρων pattern.

Η εντολή `grep`, όπως είπαμε παραπάνω δίνει ως αποτέλεσμα τις θέσεις των στοιχείων του διανύσματος, στα οποία εντοπίστηκε η επιθυμητή σειρά χαρακτήρων. Έτσι

```
> xnames
[1] "Yiannis" "Yiorgos" "Barbara" "Aleka"    "Eugenia"
> xsurnames
[1] "Papadopoulos" "Kitsos"       "Ioannou"
[4] "Tsapara"      "Grigoriadou"
> fnames<-paste(xnames,xsurnames)
> grep('Y',fnames)
[1] 1 2                                                    ■.
```

Το παραπάνω αποτέλεσμα μας πληροφορεί ότι το 1ο και 2ο στοιχείο του διανύσματος `fnames` έχει τον χαρακτήρα «Y». Αν επιθυμούμε να δούμε ποια είναι τα ονόματα αυτά, τότε δίνουμε την εντολή

```
> grep('Y',fnames,value=TRUE)
[1] "Yiannis Papadopoulos" "Yiorgos Kitsos"                ■.
```

Επίπλέον, με βάση τα προηγούμενα, η εντολή `grep` κάνει διαχωρισμό μεταξύ κεφαλαίων και πεζών γραμμάτων. Συνεπώς, με τις ακόλουθες εντολές παίρνουμε τα ονόματα που περιέχουν τον χαρακτήρα «i»

```
> grep('i',fnames,value=TRUE)
[1] "Yiannis Papadopoulos" "Yiorgos Kitsos"
[3] "Eugenia Grigoriadou"                                  ■,
```

ενώ τα ονόματα που περιέχουν τον χαρακτήρα «I» μπορούμε να τα εξάγουμε με τη σύνταξη

```
> grep('I',fnames,value=TRUE)
[1] "Barbara Ioannou"                                      ■.
```

Συνδυασμός και των δύο επιτυγχάνεται με την παράμετρο `ignore.case=TRUE`, δηλαδή

```
> grep('I',fnames,value=TRUE, ignore.case=TRUE)
[1] "Yiannis Papadopoulos" "Yiorgos Kitsos"
[3] "Barbara Ioannou"      "Eugenia Grigoriadou"           ■.
```

Όσον αφορά την εντολή `regexpr`, το αποτέλεσμα της είναι ένα διάνυσμα ακέραιων αριθμών με μήκος ίδιο με το διάνυσμα `text`, στο οποίο ψάχνουμε για τη σειρά χαρακτήρων (pattern). Οι ακέραιοι αριθμοί αντιστοιχούν στη θέση του χαρακτήρα που εντοπίστηκε η σειρά χαρακτήρων (pattern) για πρώτη φορά. Όταν δεν εντοπίζεται σε ένα στοιχείο η σειρά χαρακτήρων (pattern), τότε ως αποτέλεσμα δίνεται η τιμή -1. Για παράδειγμα:

```
>  regexpr('Y',fnames)
[1]  1  1 -1 -1 -1
attr(,"match.length")
[1]  1  1 -1 -1 -1                                                    ■,
```

δηλαδή μας λέει ότι στο 1ο και στο 2ο στοιχείο του `fnames` βρέθηκε ο χαρακτήρας «Y» και αυτός εντοπίστηκε στο 1ο χαρακτήρα του κάθε στοιχείου. Όμοια, αν ψάξουμε για το χαρακτήρα «a»

```
> regexpr('a',fnames)
[1]  3 -1  2  5  7
attr(,"match.length")
[1]  1 -1  1  1  1
```

βλέπουμε ότι εντοπίστηκε σε όλα τα στοιχεία εκτός από το 2ο. Συγκεκριμένα, η φορά που εντοπίστηκε ο χαρακτήρας «a» είναι στον 3ο, 2ο, 5ο και 7ο χαρακτήρα του 1ου, 3ου, 4ου και 5ου ονόματος, δηλαδή στοιχείου, του διανύσματος `fnames`. Μια απλοποιημένη μορφή είναι η επιλογή `match.length` που μας δίνει το μήκος του κειμένου που έχουμε εντοπίσει. Στο παραπάνω παράδειγμα είναι ένα, αφού ψάχνουμε για ένα χαρακτήρα.

```
>  attributes(regexpr('a',fnames))
$match.length
[1]  1 -1  1  1  1                                                    ■.
```

Δύο λίγο διαφορετικά παραδείγματα είναι τα ακόλουθα:

```
> attributes(regexpr('Yian',fnames))
$match.length
[1]  4 -1 -1 -1 -1
> attributes(regexpr('nn',fnames))
$match.length
[1]  2 -1  2 -1 -1                                                    ■.
```

Σε αυτή την περίπτωση χρήσιμη είναι η ακόλουθη σύνταξη:

```
> letters
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k"
[12] "l" "m" "n" "o" "p" "q" "r" "s" "t" "u" "v"
[23] "w" "x" "y" "z"
```

```
> grep("[a-z]", letters)
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
[16] 16 17 18 19 20 21 22 23 24 25 26                                    ■.
```

Η παραπάνω σύνταξη μας ζήτησε να βρούμε οποιοδήποτε χαρακτήρα από το «a» έως το «z» στο διάνυσμα `letters`. Άρα, αν γράψουμε

```
> grep("[a-c]", letters)
[1] 1 2 3
```

θα προκύψει ως αποτέλεσμα ότι βρέθηκαν μόνο στα 3 πρώτα στοιχεία του διανύσματος `letters`. Με παρόμοιο τρόπο μπορούμε να ζητήσουμε όλα τα ονοματεπώνυμα που περιέχουν τους χαρακτήρες «a» έως «b».

```
> grep("[a-b]", fnames)
[1] 1 3 4 5                                                              ■.
```

Παρόμοια εντολή με την `grep` είναι η `agrep`, η οποία ψάχνει για παρόμοιες εκφράσεις και όχι για ακριβώς όμοιες με τις ζητούμενες.

Εντολές αντικατάστασης `sub` και `gsub`.

Παρόμοια σύνταξη με τις εντολές `grep` και `regexpr` έχουν οι εντολές αντικατάστασης `sub` και `gsub`:

```
sub(pattern, replacement, x, ignore.case = FALSE,
          extended = TRUE, perl = FALSE)
gsub(pattern, replacement, x, ignore.case = FALSE,
         extended = TRUE, perl = FALSE)                                  ■.
```

Οι παραπάνω παράμετροι είναι αντίστοιχοι με τις παραμέτρους των εντολών `grep` και `regexpr`. Με την επιπλέον παράμετρο `replacement`, μπορούμε να ορίσουμε μια σειρά χαρακτήρων, οι οποίοι θα αντικαταστήσουν τα σημεία όπου θα βρεθεί η σειρά χαρακτήρων (pattern).

Να σημειώσουμε ότι οι εντολές αυτές διαφέρουν μεταξύ τους μόνο στο ότι η πρώτη εντολή αντικαθιστά μόνο την πρώτη σειρά χαρακτήρων (pattern) που θα εντοπιστεί, ενώ η δεύτερη εντολή αντικαθιστά όλες τις σειρές που θα εντοπίσει.

Απλά παραδείγματα χρήσης της εντολής `sub` είναι τα ακόλουθα:

```
> fnames  <- c( "Yiannis Papadopoulos",  "Yiorgos Kitsos",  "
   Barbara Ioannou",   "Aleka Tsapara" ,   "Eugenia Grigoriadou" )
> fnames
[1] "Yiannis Papadopoulos" "Yiorgos Kitsos"
[3] "Barbara Ioannou"      "Aleka Tsapara"
[5] "Eugenia Grigoriadou"
> sub('Y','G',fnames)
[1] "Giannis Papadopoulos" "Giorgos Kitsos"
```

```
[3] "Barbara Ioannou"       "Aleka Tsapara"
[5] "Eugenia Grigoriadou"
> sub('Yiannis','Ioannis',fnames)
[1] "Ioannis Papadopoulos" "Yiorgos Kitsos"
[3] "Barbara Ioannou"       "Aleka Tsapara"
[5] "Eugenia Grigoriadou"                                    ■,
```

ενώ για την διαφορά μεταξύ `sub` και `gsub` μπορούμε να δώσουμε τα ακόλουθα απλά παραδείγματα:

```
> sub('a','++',fnames)
[1] "Yi++nnis Papadopoulos" "Yiorgos Kitsos"
[3] "B++rbara Ioannou"       "Alek++ Tsapara"
[5] "Eugeni++ Grigoriadou"
>
> gsub('a','++',fnames)
[1] "Yi++nnis P++p++dopoulos"
[2] "Yiorgos Kitsos"
[3] "B++rb++r++ Io++nnou"
[4] "Alek++ Ts++p++r++"
[5] "Eugeni++ Grigori++dou"                                  ■.
```

Οι εντολές `match` και `%in%`.

Η εντολή `match` μας επιστρέφει ένα διάνυσμα με τις θέσεις των χαρακτήρων, όπου συναντάται για πρώτη φόρα η ζητούμενη σειρά χαρακτήρων (ταίριασμα). Η σύνταξη είναι η ακόλουθη:

```
  match(x, table, nomatch = NA, incomparables = FALSE)      ■.
```

Αναλυτικά οι παράμετροι της εντολής `match` είναι οι ακόλουθες:

x : διάνυσμα τιμών προς εύρεση - σύγκριση.

table : αντικείμενο μέσα στο οποίο θα γίνει η διερεύνηση για την ακριβή εύρεση του ταιριάσματος (`match`).

nomatch : η τιμή που επιστρέφεται όταν δεν βρεθεί κανένα στοιχείο που να ταιριάζει (`match`) στο αντικείμενο `table`.

incomparales : διάνυσμα τιμών το οποίο δεν μπορεί να ταιριάξει (`match`).

΄Ενα απλό παράδειγμα είναι το ακόλουθο:

```
> match( 'd', letters )
[1] 4                                                        ■,
```

το οποίο μας λέει ότι η σειρά χαρακτήρων μας «d» εντοπίστηκε για πρώτη φορά στο 4ο στοιχείο του διανύσματος `letters`. Προσοχή, η διαφορά είναι ότι εντολή `match` ψάχνει για ακριβές ταίριασμα των στοιχείων. Συνεπώς, η ακόλουθη εντολή θα μας δώσει μηδενικό εύρημα (σε αντίθεση με τις προηγούμενες εντολές εύρεσης):

```
> match( 'Y', fnames )
[1] NA                                                          ■.
```

Η εντολή, λοιπόν, `match` μας επιστρέφει ένα διάνυσμα με μήκος ίσο με το διάνυσμα `x` και μας λέει που συναντάμε στο αντικείμενο `table` για πρώτη φορά το αντίστοιχο στοιχείο, π.χ.

```
> match( c('z','d'), letters )
[1] 26  4
> match( c('d','z', 'a'), letters )
[1]  4 26  1                                                    ■.
```

Η εντολή αυτή, μπορεί να χρησιμοποιηθεί και για αριθμούς, για παράδειγμα:

```
> match(  c(3,4,7), rep(1:10,3) )
[1] 3 4 7
> match(  c(3,4,7), rep( c(0,1:10),3) )
[1] 4 5 8                                                       ■.
```

Αν δεν βρεθεί κάποιος αριθμός, τότε απλά δίνει την τιμή `NA` όπως βλέπουμε στο παράδειγμα που ακολουθεί.

```
> match(  c(3,4,70), rep( c(0,1:10),3) )
[1]  4  5 NA                                                    ■.
```

Τέλος η παράμετρος `incomparables` απλά αφαιρεί αυτά τα στοιχεία από τη σύγκρισή και βάζει στη θέση τους την τιμή `NA`.

```
> match(  c(3,4,70), rep( c(0,1:10),3), incomparables=c(3,7) )
[1] NA  5 NA
> match(  c(3,4,70), rep( c(0,1:10),3), incomparables=c(3,4) )
[1] NA NA NA                                                    ■.
```

Την ίδια δουλεία μπορεί να κάνει και η σύνταξη

```
x %in% table
```

όπως για παράδειγμα

```
> c(3,4,7) %in% rep(1:10,3)
[1] TRUE TRUE TRUE                                              ■.
```

Σε αυτή την περίπτωση όμως το αποτέλεσμα είναι ένα λογικό διάνυσμα που μας λέει αν τα στοιχεία του διανύσματος `x` όντως συμπεριλαμβάνονται στο αντικείμενο `table`.

Η εντολή `charmatch`**.**

Παραλλαγή της εντολής `match` είναι η εντολή `charmatch`, που χρησιμοποιείται για τον εντοπισμό και την εύρεση χαρακτήρων. Η σύνταξη είναι παρόμοια με αυτής της εντολής `match`, όμως το αποτέλεσμα είναι ένας αριθμός για κάθε στοιχείο του διανύσματος `match`. Αν υπάρχει μία και μοναδική εύρεση τότε δίνεται στο σημείο - στοιχείο του `table`, στο οποίο βρίσκουμε τη σειρά χαρακτήρων `match`. Εάν έχουμε πολλαπλές ευρέσεις, τότε δίνεται ως αποτέλεσμα μηδέν (0), ενώ όταν δεν εντοπίζεται καθόλου, τότε ως αποτέλεσμα δίνεται το `NA`.

Παραδείγματα:

```
> charmatch('Y', fnames)
[1] 0
> charmatch('Yiannis', fnames)
[1] 1
> charmatch('Yiorg', fnames)
[1] 2
> charmatch('barb', fnames)
[1] NA
> charmatch(c('Y','Barb', 'arba'), fnames)
[1]  0  3 NA                                                    ■.
```

΄Ενα βασικό μειονέκτημα της παραπάνω εντολής είναι ότι η σύγκριση ξεκινάει πάντα από τον πρώτο χαρακτήρα και δεν εντοπίζει μεμονωμένες ευρέσεις που βρίσκονται εσωτερικά στις σειρές χαρακτήρων που συγκρίνουμε.

2.12.5 Μετατροπές Χαρακτήρων

Οι εντολές `tolower` και `toupper` μετατρέπουν όλους τους χαρακτήρες σε πεζά ή κεφαλαία γράμματα αντίστοιχα. Για παράδειγμα:

```
> tolower(fnames)
[1] "yiannis papadopoulos" "yiorgos kitsos"
[3] "barbara ioannou"      "aleka tsapara"
[5] "eugenia grigoriadou"
> toupper(fnames)
[1] "YIANNIS PAPADOPOULOS" "YIORGOS KITSOS"
[3] "BARBARA IOANNOU"      "ALEKA TSAPARA"
[5] "EUGENIA GRIGORIADOU"                                       ■.
```

Επιπλέον, η εντολή `casefold(x, upper = FALSE)` είναι μια εντολή που συμπεριλαμβάνει τις δύο παραπάνω εντολές, ανάλογα τι τιμή θα βάλουμε στην παράμετρο `upper`. Αν η παράμετρος πάρει την τιμή `TRUE`, τότε μετατρέπει όλα τα στοιχεία του διανύσματος χαρακτήρων `x` σε πεζά γράμματα, ενώ αν πάρει την τιμή `FALSE` σε κεφαλαία γράμματα.

```
> x<-c("A text I want to display with spaces")
```

```
> casefold(x, upper = FALSE)
[1] "a text i want to display with spaces"

> casefold(x, upper = TRUE)
[1] "A TEXT I WANT TO DISPLAY WITH SPACES"                    ■.
```

ΚΕΦΑΛΑΙΟ 3

Πίνακες, Πλαίσια Δεδομένων και Λίστες

Όπως αναφέραμε και στην Ενότητα 1.9, η R εκτός από τα απλά διανύσματα έχει και πολλά διαφορετικά αντικείμενα. Σε αυτό το Κεφάλαιο θα περιγράψουμε τους κύριους τύπους αντικειμένων που διαφέρουν αρκετά τόσο στη μορφή όσο και στον τρόπο αποθήκευσης των δεδομένων τους. Έτσι, θα αναφερθούμε διεξοδικά στους πίνακες (ή αλλιώς μήτρες, matrices), πολυδιάστατους πίνακες (arrays), τα πλαίσια δεδομένων (data frames) και τις λίστες (lists). Αυτό αποτελεί μοναδικό χαρακτηριστικό της R που τη διακρίνει από τα άλλα πακέτα στα οποία ο μόνος τρόπος αποθήκευσης των δεδομένων είναι σε πινακοποιημένη μορφή ανάλογη των φύλλων εργασίας των λογιστικών εφαρμογών, όπως το Excel (που αντιστοιχεί σε ένα πλαίσιο δεδομένων στην R).

3.1 Δισδιάστατοι πίνακες

Οι πίνακες (**πίνακας - matrix**) είναι δομές δεδομένων, οι οποίες έχουν τα στοιχεία τους διατεταγμένα σε γραμμές και στήλες. Κάθε πίνακας περιέχει στοιχεία του ίδιου τύπου (αριθμούς, λογικές τιμές ή character).

Δημιουργούνται, εφαρμόζοντας την εντολή `matrix()` σε ένα διάνυσμα και δηλώνοντας πόσες γραμμές και στήλες επιθυμούμε να έχει ο πίνακας.

```
> x<-c(1:3,11:13,21:23,31:33)
> x
[1] 1 2 3 11 12 13 21 22 23 31 32 33
> y<-matrix(x,ncol=3)
> y
     [,1] [,2] [,3]
[1,]    1   12   23
[2,]    2   13   31
[3,]    3   21   32
```

```
[4,]   11   22   33
> y<-matrix(x,nrow=4)
> y
     [,1] [,2] [,3]
[1,]    1   12   23
[2,]    2   13   31
[3,]    3   21   32
[4,]   11   22   33                                                    ■.
```

Οι παράμετροι `ncol` και `nrow` δηλώνουν τον αριθμό των στηλών και τον αριθμό των γραμμών, αντίστοιχα. Μόνο η μία αρκεί. Η άλλη θα υπολογιστεί αυτόματα από την R.

Προσέξτε στα παραπάνω παραδείγματα ότι τα στοιχεία του διανύσματος γεμίζουν τον πίνακα στήλη-στήλη. Αν θέλουμε να τον γεμίσουν γραμμή-γραμμή πρέπει να δώσουμε στην εντολή `matrix()` την παράμετρο `byrow=T`.

```
> y<-matrix(x,ncol=3,byrow=T)
> y
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]   11   12   13
[3,]   21   22   23
[4,]   31   32   33                                                    ■.
```

Συγκρίνετε τα αποτελέσματα με τον προηγούμενο πίνακα `y`.

Σημείωση: Μπορούμε να θεωρήσουμε ότι υπάρχει ένας δείκτης γραμμών και ένας δείκτης στηλών που δείχνουν το στοιχείο του πίνακα στο οποίο αναφερόμαστε ανά πάσα στιγμή. Όταν ο πίνακας γεμίζει στήλη-στήλη, τα στοιχεία καταχωρούνται στη στήλη 1 μέχρι αυτή να γεμίσει, δηλαδή μέχρι ο δείκτης γραμμής να μεταβληθεί από 1 ως το πλήθος γραμμών. Ύστερα αρχίζει να γεμίζει η στήλη 2 ξανά, μέχρι ο δείκτης γραμμών να μετακινηθεί από το 1 ως τον αριθμό γραμμών. Για κάθε μεταβολή, δηλαδή της τιμής του δείκτη στηλών, ο δείκτης γραμμών παίρνει όλο το εύρος τιμών του. Αυτό το φαινόμενο περιγράφεται με τη φράση *«ο δείκτης γραμμών κινείται γρηγορότερα από το δείκτη στηλών»*. Θα ξανασυναντήσουμε την ορολογία όταν αναφερθούμε στους πολυδιάστατους πίνακες.

Μπορούμε να δώσουμε ονόματα στις γραμμές και στήλες ενός πίνακα με χρήση της εντολής `dimnames()`. Λαμβάνει ως παράμετρο δύο διανύσματα ονομάτων: το πρώτο αντιστοιχεί στις γραμμές και το δεύτερο στις στήλες.

```
> k<-matrix(c(2.3,1.4,0,-12,5.27,6),ncol=3)
> k
     [,1] [,2] [,3]
[1,]  2.3    0 5.27
[2,]  1.4  -12 6.00
```

```
> dimnames(k)<-list(c("Row 1","Row 2"),c("Col 1", "Col 2","Col 3"))
> k
      Col 1 Col 2 Col 3
Row 1   2.3     0  5.27
Row 2   1.4   -12  6.00
> dimnames(k)<-list(NULL,c("Col 1","Col 2","Col 3"))
> k
     Col 1 Col 2 Col 3
[1,]   2.3     0  5.27
[2,]   1.4   -12  6.00                                              ■.
```

Η δεσμευμένη λέξη `NULL` χρησιμοποιείται υποχρεωτικά για να δηλώσουμε ότι δεν επιθυμούμε να δώσουμε ή επιθυμούμε να σβήσουμε τα ονόματα μίας ή και των δύο διαστάσεων. Η εντολή `dimnames` μπορεί να δοθεί και ως παράμετρος της εντολής `matrix()` με τη μορφή

```
matrix(..., dimnames=list, ... )                                  ■.
```

Οι διαστάσεις ενός πίνακα επιστρέφονται με την εντολή `dim()`

```
> dim(k)
[1] 2 3                                                           ■.
```

Το αποτέλεσμα είναι ένα διάνυσμα μήκους δύο στοιχείων. Με την ίδια εντολή μπορούμε να δημιουργήσουμε πίνακες από διανύσματα. Έτσι, αν σε ένα διάνυσμα προσθέσουμε τη διάσταση, τότε θα τα στοιχεία του διανύσματος θα διαμορφώσουν τον πίνακα αντίστοιχής διάστασης, γεμίζοντας διαδοχικά τις στήλες του κάθε πίνακα. Για παράδειγμα, ο κώδικας

```
> dim(k)
[1] 2 3
> x<-1:6
> dim(x)<-c(3,2)                                                  ■,
```

δηλαδή ο πίνακας `x` θα είναι διάστασης 3×2 και θα έχει 3 γραμμές και 2 στήλες με την ακόλουθη διάταξη:

```
> x
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6                                                    ■.
```

Βλέπουμε ότι πρώτα γέμισε η πρώτη στήλη με τα τρία πρώτα στοιχεία του διανύσματος και μετά η δεύτερη στήλη.

Όταν ο πίνακας που ζητάμε να φτιαχτεί έχει περισσότερα στοιχεία από το διάνυσμα από το οποίο παράγεται, τότε η R κάνει ξανά ανακύκλωση και μας δίνει σχετική προειδοποίηση.

```
> matrix(c(12,-3,4,5,7,-1,0,4),ncol=3)
     [,1] [,2] [,3]
[1,]   12    5    0
[2,]   -3    7    4
[3,]    4   -1   12
Warning message:
In matrix(c(12, -3, 4, 5, 7, -1, 0, 4), ncol = 3) :
  data length [8] is not a sub-multiple or multiple of the number
   of rows [3]                                                    ■.
```

Πίνακες δημιουργούνται και με τη συνένωση διανυσμάτων. Τα διανύσματα τοποθετούνται το ένα δίπλα στο άλλο σαν στήλες, αν δοθούν ως παράμετροι της εντολής `cbind()` και το ένα κάτω από το άλλο σαν γραμμές, αν δοθούν ως παράμετροι της εντολής `rbind()`. Ανάλογα με την εντολή `cbind()` ή `rbind()` που θα χρησιμοποιήσουμε, τα ονόματα των διανυσμάτων γίνονται ονόματα στηλών ή γραμμών, αντίστοιχα. Αν τα διανύσματα δεν έχουν το ίδιο μήκος, η R εφαρμόζει ανακύκλωση,έτσι ώστε να τα φέρει όλα στο ίδιο μήκος.

```
> x<-c(5,3,4,7)
> y<-c(-2,-1,0,4)
> cbind(x,y)
     x  y
[1,] 5 -2
[2,] 3 -1
[3,] 4  0
[4,] 7  4
> cbind(y,x)
      y x
[1,] -2 5
[2,] -1 3
[3,]  0 4
[4,]  4 7
> rbind(x,y,x)
  [,1] [,2] [,3] [,4]
x    5    3    4    7
y   -2   -1    0    4
x    5    3    4    7
> x<-c(5,3,4)
> y<-c(-2,-1,0,4,-2)
> cbind(x,y)
     x  y
[1,] 5 -2
```

```
[2,] 3 -1
[3,] 4  0
[4,] 5  4
[5,] 3 -2
Warning message:
In cbind(x, y) :
  number of rows of result is not a multiple of vector length (arg
    1)
> cbind(y,x)
      y x
[1,] -2 5
[2,] -1 3
[3,]  0 4
[4,]  4 5
[5,] -2 3
Warning message:
In cbind(y, x) :
  number of rows of result is not a multiple of vector length (arg
    2)                                                              ■.
```

Στα δυο τελευταία παραδείγματα βλέπουμε ότι γίνεται ανακύκλωση.

Τέλος, μπορούμε να δημιουργήσουμε διαγώνιους πίνακες με την εντολή `diag()`. Αν η παράμετρος είναι διάνυσμα, τότε τα στοιχεία του τοποθετούνται στη διαγώνιο. Αν η παράμετρος είναι αριθμός, τότε δημιουργείται τετραγωνικός πίνακας με αριθμό γραμμών και στηλών όσο ο αριθμός και την τιμή 1 σε όλα τα διαγώνια στοιχεία.

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

Οι μαθηματικές και λογικές πράξεις που εφαρμόζονται σε πίνακες, εφαρμόζονται σε κάθε στοιχείο τους και το αποτέλεσμα είναι πίνακας με τις ίδιες διαστάσεις με τους αρχικούς.

```
> x<-matrix(1:9,ncol=3)
> x

     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

> x^2
     [,1] [,2] [,3]
[1,]    1   16   49
[2,]    4   25   64
[3,]    9   36   81

> sqrt(x)
         [,1]     [,2]     [,3]
[1,] 1.000000 2.000000 2.645751
[2,] 1.414214 2.236068 2.828427
[3,] 1.732051 2.449490 3.000000

> x/0

     [,1] [,2] [,3]
[1,]  Inf  Inf  Inf
[2,]  Inf  Inf  Inf
[3,]  Inf  Inf  Inf

> x==4
      [,1]  [,2]  [,3]
[1,] FALSE  TRUE FALSE
[2,] FALSE FALSE FALSE
[3,] FALSE FALSE FALSE

> x>3 & x<6
      [,1]  [,2]  [,3]
[1,] FALSE  TRUE FALSE
[2,] FALSE  TRUE FALSE
[3,] FALSE FALSE FALSE
```

```
> y<-matrix(-3:5,ncol=3)
> y
     [,1] [,2] [,3]
[1,]   -3    0    3
[2,]   -2    1    4
[3,]   -1    2    5

> x>2 & y>0
      [,1]  [,2] [,3]
[1,] FALSE FALSE TRUE
[2,] FALSE  TRUE TRUE
[3,] FALSE  TRUE TRUE
```
■.

Επιμέρους στοιχεία του πίνακα εξάγονται με τρόπο αντίστοιχο των διανυσμάτων, μόνο που μέσα στις αγκύλες πρέπει να αναφερόμαστε και σε γραμμές και σε στήλες, χωρίζοντας τις συνθήκες ή τους αριθμούς που δίνουμε με κόμματα.

```
> x <- matrix( -5:6, ncol=4, nrow=3 )
> x
     [,1] [,2] [,3] [,4]
[1,]   -5   -2    1    4
[2,]   -4   -1    2    5
[3,]   -3    0    3    6
> x[1,1]
[1] -5
> x[1:2,c(4,2)]
     [,1] [,2]
[1,]    4   -2
[2,]    5   -1
> x[x<=0]
[1] -5 -4 -3 -2 -1  0
> x[-2,-1]
     [,1] [,2] [,3]
[1,]   -2    1    4
[2,]    0    3    6
```
■.

Παρατηρείστε ότι όταν χρησιμοποιείται λογική συνθήκη, τα αποτελέσματα επιστρέφονται ως διάνυσμα. Με την τελευταία εντολή δηλώνουμε ότι δεν επιθυμούμε τη δεύτερη γραμμή, ούτε την πρώτη στήλη του πίνακα. Αν παραλείψουμε να δώσουμε αριθμό για κάποια διάσταση, είναι σαν να δηλώνουμε το πλήρες εύρος της διάστασης.

```
> x[2,]
[1] -4 -1  2  5
> x[,3]
[1] 1 2 3
> x[,]
     [,1] [,2] [,3] [,4]
[1,]   -5   -2    1    4
[2,]   -4   -1    2    5
[3,]   -3    0    3    6
```
■.

΄Ενας άλλος τρόπος να εξαγάγουμε στοιχεία πιο «ακανόνιστα» κατανεμημένα σε έναν πίνακα, είναι να κατασκευάσουμε έναν άλλο πίνακα με δύο στήλες. Κάθε γραμμή του δίνει τον αριθμό γραμμής και στήλης ενός στοιχείου. Στο παρακάτω παράδειγμα ζητάμε τα στοιχεία `[1,3]` (πρώτη γραμμή, τρίτη στήλη) και `[2,4]` (δεύτερη γραμμή, τέταρτη στήλη) του πίνακα `x`.

```
> z<-matrix(c(1,3,2,4),ncol=2,byrow=T)
> z
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```
■.

Είδαμε πώς μπορεί κανείς να χρησιμοποιήσει τους δείκτες για να αποκτήσει πρόσβαση σε συγκεκριμένα στοιχεία ή ακόμα και γραμμές (στήλες) ενός πίνακα. Αυτό που θα πρέπει να τονιστεί είναι πως αν από λάθος χρησιμοποιηθεί ένας μόνο δείκτης (χωρίς όμως κόμμα), τότε η R αντιλαμβάνεται τον πίνακα ως διάνυσμα, το οποίο κατασκευάζεται, βάζοντας σε σειρά τις στήλες (θυμηθείτε τον τρόπο με τον οποίο δημιουργούμε έναν πίνακα ανά στήλες). ΄Ετσι, έχουμε:

```
> x<-matrix(1:12,3,4)
> x
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> x[,4]
[1] 10 11 12
> x[2,]
[1]  2  5  8 11
> x[6]
[1] 6
```
■.

Δηλαδή όταν ζητήσουμε το `x[6]`, ο πίνακας `x` γίνεται διάνυσμα και η σειρά των στοιχειών καθορίζεται από τις στήλες, ουσιαστικά είναι ισοδύναμο με το να ορίσουμε `c(x[,1],x[,2],x[,3],x[,4])`

Τέλος, η εντολή `diag()` επιστρέφει τα στοιχεία της κύριας διαγωνίου του πίνακα. Προσέξτε ποια στοιχεία επιστρέφονται όταν ο πίνακας δεν είναι τετραγωνικός.

```
> x
     [,1] [,2] [,3] [,4]
[1,]   -5   -2    1    4
[2,]   -4   -1    2    5
[3,]   -3    0    3    6
> diag(x)
[1] -5 -1  3
> y<-matrix(1:9,ncol=3)
> y
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> diag(y)
[1] 1 5 9
```

■.

Πράξεις μεταξύ πινάκων δεν γίνονται με τη χρήση άλγεβρας πινάκων. Αντιθέτως, όπως και στα διανύσματα, γίνονται μεταξύ αντιστοίχων στοιχείων. Οι πίνακες όμως, πρέπει να έχουν τις ίδιες διαστάσεις, γιατί δεν πραγματοποιείται ανακύκλωση.

```
> x
     [,1] [,2] [,3] [,4]
[1,]   -5   -2    1    4
[2,]   -4   -1    2    5
[3,]   -3    0    3    6
> diag(x)
[1] -5 -1  3
> y<-matrix(1:9,ncol=3)
> y
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> diag(y)
[1] 1 5 9
> x<-matrix(1:9,ncol=3)
> y<-matrix(9:1,ncol=3)
> x+y
```

```
     [,1] [,2] [,3]
[1,]   10   10   10
[2,]   10   10   10
[3,]   10   10   10
> x^y
     [,1] [,2] [,3]
[1,]    1 4096  343
[2,]  256 3125   64
[3,] 2187 1296    9
> x>y
      [,1]  [,2] [,3]
[1,] FALSE FALSE TRUE
[2,] FALSE FALSE TRUE
[3,] FALSE  TRUE TRUE
>  x>y & x<8
      [,1]  [,2]  [,3]
[1,] FALSE FALSE  TRUE
[2,] FALSE FALSE FALSE
[3,] FALSE  TRUE FALSE
```

■.

Για να εφαρμοστεί άλγεβρα πινάκων υπάρχουν ειδικές συναρτήσεις στην R που δίνονται στον Πίνακα 3.1 που ακολουθεί.

Εντολή	**Ερμηνεία**
`t(Q)`	Ανάστροφος του πίνακα `Q`
`solve(Q)`	Αντίστροφος του πίνακα `Q` (αν υπάρχει)
`%*%`	Πολλαπλασιασμός πινάκων (αν οι διαστάσεις είναι συμβατές)

Πίνακας 3.1: Διάφορες συναρτήσεις και τελεστές για πράξεις με πίνακες

Παραδείγματα χρήσης των συναρτήσεων αυτών δίνονται στον ακόλουθο κώδικα της R.

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

```
[5,]    5   10
> y
     [,1] [,2]
[1,]    1    1
[2,]    3    2
> x %*% y
     [,1] [,2]
[1,]   19   13
[2,]   23   16
[3,]   27   19
[4,]   31   22
[5,]   35   25
> y%*%x
Error in y %*% x : non-conformable arguments
> x<-matrix(c(4,0,0,4),ncol=2)
> x
     [,1] [,2]
[1,]    4    0
[2,]    0    4
> solve(x)
     [,1] [,2]
[1,] 0.25 0.00
[2,] 0.00 0.25
> x<-matrix(-5:6,ncol=4)
> x
     [,1] [,2] [,3] [,4]
[1,]   -5   -2    1    4
[2,]   -4   -1    2    5
[3,]   -3    0    3    6
> t(x)
     [,1] [,2] [,3]
[1,]   -5   -4   -3
[2,]   -2   -1    0
[3,]    1    2    3
[4,]    4    5    6
```

■.

΄Ενα διάνυσμα μπορεί να θεωρηθεί είτε ως πίνακας-γραμμή, είτε ως πίνακας-στήλη. Η R θα το χειριστεί ανάλογα με τη θέση του σε μια πράξη.

```
> z<-1:3
> z%*%x
```

```
     [,1] [,2] [,3] [,4]
[1,]  -22   -4   14   32
> y<-matrix(1:12,nrow=4)
> y
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12
> y%*%z
     [,1]
[1,]   38
[2,]   44
[3,]   50
[4,]   56
```

■.

Στο παραπάνω παράδειγμα, στον πολλαπλασιασμό με το x, το z θεωρήθηκε πίνακας-γραμμή, ενώ στον πολλαπλασιασμό με το y θεωρήθηκε πίνακας-στήλη.

Παράδειγμα 3.1 *[Γραμμική Παλινδρόμηση]*

Ένας ερευνητής θέλει να μελετήσει τη σχέση ανάμεσα στις μεταβλητές X και Y με τη χρήση ενός γραμμικού μοντέλου, και συγκεκριμένα να προσαρμόσει ένα μοντέλο της μορφής

$$Y = \beta_1 + \beta_2 X + \varepsilon,\ \varepsilon \sim N(0, \sigma^2).$$

Η παραπάνω εξίσωση ισχύσει για ζευγάρι (X, Y) που παρατηρούμε. Οι 10 παρατηρήσεις που έχει είναι οι εξής:

X	4	6	7	12	4	3	2	4	1	8
Y	8.5	11	12	20	9	5.5	11	5	2.3	12

Πίνακας 3.2: Παρατηρήσεις Παραδείγματος 3.1

Λύση με R: Δεν θα χρησιμοποιήσουμε τη διαδικασία που η R έχει για γραμμική παλινδρόμηση, αλλά θα χρησιμοποιήσουμε άλγεβρα πινάκων για να βρούμε τον εκτιμητή

$$\widehat{\boldsymbol{\beta}} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{Y},$$

όπου $\boldsymbol{\beta} = (\beta_1, \beta_2)$ οι συντελεστές της παλινδρόμησης και $\widehat{\boldsymbol{\beta}} = (\widehat{\beta}_1, \widehat{\beta}_2)$ οι αντίστοιχες εκτιμήσεις. Επιπλέον, αν n είναι ο αριθμός των παρατηρήσεων τότε ο πίνακας $\boldsymbol{X}$ είναι διάστασης $n \times 2$ με την πρώτη στήλη να έχει μονάδες (για να εκτιμήσουμε τη σταθερά που συμβολίζεται εδώ ως β_1)

και η δεύτερη στήλη τις τιμές της μεταβλητής X (για να εκτιμήσουμε την παράμετρο β_2). Τέλος, το διάνυσμα $\boldsymbol{Y} = (y_1, d_2, \ldots, y_n)$ έχει τις τιμές που παρατηρήσαμε για τη μεταβλητή Y.

Μπορούμε να υπολογίσουμε το $\widehat{\boldsymbol{\beta}}$ με τις ακόλουθες εντολές

```
> x<-c(4,6,7,12,4,3,2,4,1,8)
> y<-c(8.5,11,12,20,9,5.5,11,5,2.3,12)
> n<-length(x)
> Y<-matrix(y,n,1)   # capital Y
> X<-cbind(1,x)      # capital X
> betahat<-solve(t(X)%*%X)%*%t(X)%*%Y
> betahat
     [,1]
   2.876396
x  1.324236                                                    ■.
```

Παρατηρείστε τα εξής: Η συνάρτηση `length`, που επιστρέφει το πλήθος των στοιχειών ενός διανύσματος, εδώ, μας χρησιμεύει να ορίσουμε κατάλληλα τους πίνακες. Οι πίνακες έχουν οριστεί με κεφαλαία γράμματα και διαφέρουν από τα δεδομένα που είναι με μικρά γράμματα. Χρησιμοποιώντας την εντολή `cbind` η R καταλαβαίνει πως δημιουργούμε πίνακα και πως η πρώτη στήλη περιέχει μόνο μονάδες.

3.1.1 Η εντολή `is.matrix`

Η εντολή `is.matrix` εξετάζει αν το αντικείμενο είναι τύπου matrix ή όχι και επιστρέφει `TRUE` ή `FALSE` αντίστοιχα. Με όμοιο τρόπο η εντολή `is.vector` εξετάζει αν ένα αντικείμενο είναι τύπου vector. Αυτές οι εντολές είναι πολύ χρήσιμες για να μπορέσουμε να ελέγξουμε τη μορφή που έχει το αντικείμενο και πιθανότατα να μπορέσουμε να επιλέξουμε την κατάλληλη μέθοδο.

```
> x<-c(1,4,7,10,12)
> y<-matrix(1:12,3,4)
> is.matrix(x)
[1] FALSE
> is.matrix(y)
[1] TRUE
> is.vector(x)
[1] TRUE
> is.vector(y)
[1] FALSE                                                      ■.
```

3.1.2 Πράξεις με πίνακες

Έστω ένας πίνακας `x`. Γράψτε μια εντολή που να επιστρέφει σε ένα διάνυσμα το άθροισμα κάθε γραμμής του πίνακα.

Θα μπορούσε κανείς να υπολογίσει ξεχωριστά το άθροισμα κάθε γραμμής, αλλά κάτι τέτοιο θα ήταν εξαιρετικά χρονοβόρο, ειδικά αν οι γραμμές ήταν πολλές. Εναλλακτικά, θα μπορούσε κανείς να χρησιμοποιήσει μια επαναληπτική εντολή, όπως θα δούμε παρακάτω. Μια απλή ιδέα είναι πως για να πάρουμε τα αθροίσματα των γραμμών, μπορούμε απλά να πολλαπλασιάσουμε τον πίνακα με ένα μοναδιαίο διάνυσμα, διάνυσμα με όλα τα στοιχεία ίσα με το 1. Για παράδειγμα, αν έχουμε έναν πίνακα με 3 γραμμές, τότε παίρνουμε:

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} a+b+c \\ d+e+f \\ g+h+i \end{bmatrix} .$$

Συνεπώς, οι εντολές που χρειαζόμαστε για ένα πίνακα `x` είναι:

```
> x<-matrix(-4:5,5,2)
> columns<- dim(x)[2]    #number of columns
> x %*%matrix(rep(1,columns),columns,1)
     [,1]
[1,]   -3
[2,]   -1
[3,]    1
[4,]    3
[5,]    5
```

■.

3.2 Πολυδιάστατοι πίνακες

Ένας **πολυδιάστατος πίνακας (array)** είναι ένας πίνακας με τρεις ή περισσότερες διαστάσεις. Ένας πίνακας τριών διαστάσεων για παράδειγμα, θα εμφανίζεται ως τόσοι δισδιάστατοι πίνακες, όσοι η τιμή της τρίτης διάστασης. Η R έχει όριο τις 8 διαστάσεις. Αυτοί οι πίνακες δημιουργούνται με την εντολή `array()` ή την εντολή `dim()`. Η `dim()` δέχεται ως παραμέτρους το μέγεθος της κάθε διάστασης. Η `array()`, για να δηλωθούν οι διαστάσεις, δέχεται ως παράμετρο την `dim=` με ένα διάνυσμα με τα μεγέθη των διαστάσεων.

```
 > a1<- array(1:24,dim=c(3,4,2))
> a1
, , 1

     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12

, , 2
```

```
     [,1] [,2] [,3] [,4]
[1,]   13   16   19   22
[2,]   14   17   20   23
[3,]   15   18   21   24                                    ■.
```

Παρατηρείστε πώς «γεμίζει» ο πολυδιάστατος πίνακας. «Κινείται» πιο γρήγορα η πρώτη διάσταση, μετά η δεύτερη και τελευταία η τρίτη. Δηλαδή «γεμίζει» πρώτα ο πρώτος πίνακας `[,,1]`, μέσα σε αυτόν «γεμίζουν» πρώτα οι στήλες και μετά οι γραμμές, ύστερα «γεμίζει» ο δεύτερος πίνακας, κ.ο.κ.

Με χρήση της `dim()` ο ίδιος πίνακας φτιάχνεται ως εξής:

```
> a1<-1:24
> dim(a1)<-c(3,4,2)
> a1
, , 1

     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12

, , 2

     [,1] [,2] [,3] [,4]
[1,]   13   16   19   22
[2,]   14   17   20   23
[3,]   15   18   21   24                                    ■.
```

Η R χειρίζεται τις δομές τύπου `array`, όπως και αυτές τύπου `matrix`. Μπορούν να δοθούν ονόματα στις διαστάσεις τους, οι πράξεις γίνονται στοιχείο-στοιχείο, επιμέρους στοιχεία εξάγονται με χρήση της αγκύλης, αριθμητικοί ή λογικοί δείκτες μπορούν να χρησιμοποιηθούν, κλπ.

3.3 Τύποι δεδομένων ως διανύσματα

Μέχρι τώρα, είδαμε, ότι τα διανύσματα μπορεί να είναι αριθμητικά, λογικά, σειρά χαρακτήρων και σύνθετα (ή μιγαδικά). Με εξαίρεση τον τελευταίο τύπο που είναι καθαρά μαθηματικός, οι υπόλοιποι τύποι των διανυσμάτων μπορούν χρησιμοποιηθούν για να αποθηκεύσουν αριθμητικά (συνεχή ή διακριτά) δεδομένα, δίτιμα δεδομένα ή διακριτά ονόματα. Υπάρχουν, όμως, και τα κατηγορικά δεδομένα που δεν μπορούν να εκφραστούν πλήρως από τους παραπάνω τύπους διανυσμάτων.

Στη Στατιστική, ως κατηγορικές μεταβλητές ονομάζουμε τις τυχαίες μεταβλητές που καταγράφουν τα χαρακτηριστικά της μονάδας που μελετάμε, των οποίων τα ενδεχόμενα μπορούν να περιγραφούν λεκτικά, δηλαδή με σειρές χαρακτήρων, και όχι αριθμητικά. Σε κάθε κατηγορική μεταβλητή, τα πιθανά αποτελέσματα (ενδεχόμενα) ονομάζονται και κατηγορίες ή τα επίπεδα (levels) και οι λεκτικές τους περιγραφές ως ετικέτες ή απλά περιγραφές. Οι κατηγορίες στις οποίες ανήκει κάθε μονάδα που μελετούμε μπορούν να καταγραφούν και ως ένα διάστημα χαρακτήρων (character). Επειδή, όμως, τα επίπεδα των κατηγορικών μεταβλητών είναι περιορισμένα σε πλήθος, χρησιμοποιούμε συμβατικούς κωδικούς (δηλαδή αριθμούς) για να διευκολύνουμε την εισαγωγή τους στον υπολογιστή και κατόπιν αναθέτουμε σε κάθε κωδικό την αντίστοιχη περιγραφή (ετικέτα). Ο τρόπος αυτός καταγραφής γίνεται στην R με τους παράγοντες (factors) που δεν είναι τίποτα άλλο από διανύσματα κατηγορικών μεταβλητών. Οι κατηγορικές μεταβλητές χωρίζονται σε δύο κατηγορίες: τις ονομαστικές (nominal) και τις διατάξιμες (ordinal) και στην R μπορούν να δηλωθούν με τις εντολές `factor` και `ordered`, αντίστοιχα. Στην R είναι πολύ σημαντικό να δηλώνονται οι κατηγορικές μεταβλητές σωστά, γιατί η εφαρμογή πολλών στατιστικών εντολών/συναρτήσεων αλλάζει ανάλογα με τον τύπο των μεταβλητών (διανυσμάτων).

3.3.1 Κατηγορικές μεταβλητές ως παράγοντες (factors)

Ο τύπος `factor` χρησιμοποιείται για να ορίσουμε στην R ότι τα δεδομένα είναι κατηγορικά και όχι αριθμητικά. Κάτι τέτοιο είναι σημαντικό γιατί:

- Προφυλάσσει από τη χρήση κατηγορικών μεταβλητών σε στατιστικές αναλύσεις ή μοντέλα που απαιτούν αριθμητικά δεδομένα. Για παράδειγμα, αν σε ένα μοντέλο παλινδρόμησης βάλουμε ως μεταβλητή απόκρισης μια κατηγορική μεταβλητή, τότε η R θα μας δώσει μήνυμα λάθους.•
- Για κάποια στατιστικά μοντέλα, η R αυτόματα χειρίζεται τις μεταβλητές αυτές σαν κατηγορικές με το σωστό τρόπο. Για παράδειγμα, σε γραμμικά μοντέλα αν μια μεταβλητή έχει οριστεί σαν `factor`, τότε η R αυτόματα χρησιμοποιεί ψευδομεταβλητές για να χρησιμοποιήσει τη μεταβλητή στο μοντέλο, χωρίς να χρειάζεται ο χρήστης να επέμβει.

Οι εντολές που ακολουθούν ορίζουν το αντικείμενο `classroom` να είναι τύπου `factor` και στη συνέχεια, παίρνουμε έναν πίνακα συχνοτήτων για τις τιμές του.

```
> classroom<-factor(c("A12","A13","A15","A17", "A12","A12"))
> classroom
[1] A12 A13 A15 A17 A12 A12
Levels: A12 A13 A15 A17
> classroom
A12 A13 A15 A17
  3   1   1   1                                                    ■.
```

Μερικές χρήσιμες ιδιότητες του τύπου `factor` είναι οι ακόλουθες.

- Μπορούμε να μετατρέψουμε ένα αντικείμενο σε τύπο `factor` με τη χρήση της συνάρτησης `as.factor`.
- Μπορούμε να μετατρέψουμε μια μεταβλητή τύπου `factor` σε αριθμητική με τη χρήση της συνάρτησης `as.numeric`.

Στα αντικείμενα τύπου `factor` δεν μπορούμε να εκτελέσουμε αριθμητικές πράξεις. Από τις λογικές πράξεις, νόημα έχουν μόνο οι συγκρίσεις ισότητας και ανισότητας και όχι συγκρίσεις με τη χρήση των τελεστών «ή».

```
> 4+classroom
[1] NA NA NA NA NA NA
Warning message:
In Ops.factor(4, classroom) : + not meaningful for factors
> as.numeric(classroom)
[1] 1 2 3 4 1 1
3+as.numeric(classroom)
[1] 4 5 6 7 4 4
> x<-c(1,1,2,3,1,2,1,2,1,2,2)
> as.factor(x)
[1] 1 1 2 3 1 2 1 2 1 2 2
Levels: 1 2 3
> 3+as.factor(x)
[1] NA NA NA NA NA NA NA NA NA NA NA
Warning message:
In Ops.factor(3, as.factor(x)) : + not meaningful for factors
> classroom[1]<classroom[2]
[1] NA
Warning message:
In Ops.factor(classroom[1], classroom[2]) : < not meaningful for
   factors
> classroom[1]==classroom[2]
[1] FALSE                                                          ■.
```

Θα πρέπει να είμαστε προσεκτικοί με τον ορισμό των επιπέδων μιας κατηγορικής μεταβλητής, καθώς αυτή καθορίζεται με τη σειρά που εμφανίζονται στο αντικείμενο. Αν θέλουμε να καθορίσουμε εμείς τη σειρά, θα πρέπει να χρησιμοποιήσουμε την παράμετρο `levels` για να δηλώσουμε τη σειρά των τιμών. Έτσι λοιπόν η εντολή

```
> satisf<-factor(c("Hi","Lo","Hi","Lo","Me","Me", "Lo"))
> satisf
[1] Hi Lo Hi Lo Me Me Lo
```

```
Levels: Hi Lo Me
```

θα δημιουργήσει μια κατηγορική μεταβλητή με κατηγορίες (επίπεδα) «Hi», «Lo» και «Me» και αντίστοιχους κωδικούς 1, 2 και 3. Την σειρά των επιπέδων μπορούμε να τη δούμε και με την εντολή

```
> levels(satisf)
[1] "Hi" "Lo" "Me"                                                    ■.
```

Εάν θέλουμε να ορίσουμε άλλη σειρά κωδικών θα πρέπει να χρησιμοποιήσουμε τις παραμέτρους `levels` (και `labels`, αν η αρχική μεταβλητή είναι ποσοτική) στον ορισμό του παράγοντα. Έτσι η εντολή

```
> satisf<-factor(c("Hi","Lo","Hi","Lo","Me","Me", "Lo"),
+ levels=c( 'Lo', 'Me', 'Hi' ))
> satisf
[1] Hi Lo Hi Lo Me Me Lo
Levels: Lo Me Hi
> as.numeric(satisf)
[1] 3 1 3 1 2 2 1
```

θα δημιουργήσει μια κατηγορική μεταβλητή με κατηγορίες (επίπεδα) «Lo», «Me» και «Hi» και αντίστοιχους κωδικούς 1, 2 και 3, που είναι πιο ορθολογική σειρά κωδικών για τη συγκεκριμένη μεταβλητή.

Η παραπάνω μεταβλητή δεν είναι απλά κατηγορική, αλλά διατάξιμη, δηλαδή τα επίπεδα της μπορούν να μπουν σε μία σειρά-κατάταξη. Αν θέλουμε, λοιπόν, να την ορίσουμε σωστά ως διατάξιμη μεταβλητή, τότε πρέπει να την ορίσουμε στην R ως `ordered`, όπως περιγράφεται στην Ενότητα 3.3.2 που ακολουθεί.

3.3.2 Διατάξιμοι παράγοντες (ordered factors)

Μπορούμε να ορίσουμε μεταβλητές κατάταξης ή διατάξιμες με την εντολή `ordered`. Οι μεταβλητές αυτές έχουν ίδια δομή αποθήκευσης στην R (δηλαδή κωδικοί για κάθε επίπεδο με τις αντίστοιχες ετικέτες-επεξηγήσεις), με τη διαφορά ότι οι κατηγορίες μπορούν να μπουν σε σειρα και η R το λαμβάνει αυτό υπόψη της. Έτσι, τα επίπεδα του παράγοντα μπορούν να συγκριθούν ως προς το ποιο είναι μεγαλύτερο και πιο είναι μικρότερο.

```
 > classroom2<-ordered(c("1","2","2","3", "1","2"))
> classroom2
[1] 1 2 2 3 1 2
Levels: 1 < 2 < 3
> classroom2[1]>classroom2[2]
 [1] FALSE                                                           ■.
```

Και εδώ, μπορούμε να χρησιμοποιήσουμε την παράμετρο `levels` στην εντολή `ordered`, για να ορίσουμε σωστά τη σειρά των επιπέδων.

```
> satisf<-ordered(c("Hi","Lo","Hi","Lo","Me","Me", "Lo"))
> satisf
[1] Hi Lo Hi Lo Me Me Lo
Hi < Lo < Me
> satisf[1]<satisf[2]
[1] TRUE
> satisf<-ordered(c("Hi","Lo","Hi","Lo","Me","Me", "Lo"),
+         levels=c("Lo","Me","Hi"))
> satisf[1]<satisf[2]
[1] FALSE                                                    ■.
```

Τέλος, με την εντολή `cut` μπορούμε να κατηγοριοποιήσουμε μια συνεχή μεταβλητή, κόβοντας την σε διάφορες τάξεις ή κλάσεις, ανάλογα με την τιμή της. Τέτοιου είδους διακριτοποιήσεις συνεχών μεταβλητών είναι συχνά χρήσιμες στη στατιστική. Ένα τέτοιο παράδειγμα είναι ο υπολογισμός των συχνοτήτων σε ομαδοποιημένα δεδομένα. Οι εντολές που ακολουθούν φτιάχνουν ένα διάνυσμα μεγέθους 1000, με τυχαίες τιμές από την τυποποιημένη κανονική κατανομή (με την εντολή `rnorm`). Στη συνέχεια διακριτοποιούμε τις τιμές αυτές σε διαστήματα (-∞-2], (-2,-1], (-1,0], (0,1], (1,2], (2, ∞) και βλέπουμε τις συχνότητες του κάθε διαστήματος.

```
x<-rnorm(1000)
> xdisc<-cut(x,c(-Inf,-2,-1,0,1,2,Inf))
> table(xdisc)
(-Inf,-2]   (-2,-1]    (-1,0]     (0,1]     (1,2]   (2, Inf]
       25       121       361       336       131        26    ■.
```

3.3.3 Η εντολή διαχωρισμού `split`

Τελειώνοντας την αναφορά μας στον τύπο `factor` θα πρέπει να αναφέρουμε πως είναι ιδιαίτερα χρήσιμος για να χωρίζουμε τα δεδομένα σε ομάδες. Για παράδειγμα, με την εντολή `split` χωρίζουμε τις παρακάτω 10 παρατηρήσεις σε 2 ομάδες, ανάλογα με την τιμή μιας μεταβλητής τύπου `factor`.

```
> x<-c(1,2,3,4,5,6,7,8,9,10)
> y<-factor(c(1,1,1,1,1,1,2,2,2,2))
> split(x,y)
$`1`
[1] 1 2 3 4 5 6

$`2`
[1]  7  8  9 10
```

```
> test<-split(x,y)

> test$"1"
[1] 1 2 3 4 5 6
>  mean(test$"1")
[1] 3.5                                                    ■.
```

3.4 Πλαίσια δεδομένων

Το **πλαίσιο δεδομένων (data frame)** είναι ένας διδιάστατος πίνακας, του οποίου, όμως, τα στοιχεία δεν είναι όλα του ίδιου τύπου. Τα στοιχεία της ίδιας στήλης είναι όλα του ίδιου τύπου. Ο στόχος της δομής data frame είναι να καταχωρούνται οι παρατηρήσεις που έχουν ληφθεί από ένα δείγμα για μία ομάδα μεταβλητών. Υπάρχει μία στήλη για κάθε μεταβλητή, ενώ κάθε γραμμή αναλογεί σε ένα μέλος του δείγματος.

Η δομή αυτή απαιτεί την ύπαρξη ονομάτων για τις γραμμές και τις στήλες (ονόματα μεταβλητών). Δημιουργείται με χρήση της εντολής `data.frame()`.

```
> height <- c (1.75, 1.84, 1.81, 1.63)
> names(height)<-c("Jim", "George", "John", "Mary")
> sex<-c("Male", "Male", "Male", "Female")
> income<-c(1500,900,1250,2300)
> sample1<-data.frame(height,income,sex)
> sample1
       height income    sex
Jim      1.75   1500   Male
George   1.84    900   Male
John     1.81   1250   Male
Mary     1.63   2300 Female                                  ■.
```

Παρατηρείστε στο παραπάνω παράδειγμα ότι τα ονόματα του διανύσματος `height` έγιναν ονόματα των γραμμών, ενώ τα ονόματα των διανυσμάτων έγιναν ονόματα των στηλών. Η R χρησιμοποιεί ως ονόματα γραμμών τα ονόματα του πρώτου διανύσματος της `data.frame()`, το οποίο δεν έχει επαναλήψεις ονομάτων. Όταν δεν υπάρχουν ονόματα, μπορούν να δοθούν με τη χρήση του ορίσματος `row.names` στην εντολή `data.frame()`.

```
> names(height)<-NULL
>sample1<-data.frame(height,income,sex,
+                     row.names=c("Jim","George","John","Mary"))
> sample1
       height income    sex
Jim      1.75   1500   Male
```

```
George   1.84    900   Male
John     1.81   1250   Male
Mary     1.63   2300 Female                                    ■.
```

Κατά τη δημιουργία του data frame μπορούν να δοθούν άλλα ονόματα στηλών με τον παρακάτω τρόπο:

```
> sample1<-data.frame(ipsos=height,isodima=income, filo=sex,
+                     row.names=c("Jim","George","John","Mary"))
> sample1
       ipsos isodima   filo
Jim     1.75    1500   Male
George  1.84     900   Male
John    1.81    1250   Male
Mary    1.63    2300 Female                                    ■.
```

Διανύσματα τύπου `character` μετατρέπονται αυτομάτως σε τύπου `factor` κατά την εισαγωγή τους στο data frame. Για να αποφευχθεί αυτό πρέπει το όνομα του διανύσματος να δηλωθεί ως `I(όνομα)`. Αν, δηλαδή, στο παραπάνω παράδειγμα, θέλαμε το διάνυσμα `sex` να παραμείνει τύπου `character` έπρεπε να δημιουργήσουμε το data frame με την εντολή:

```
sample1 <- data.frame( ipsos=height, isodima=income, filoI=(sex),
   row.names=c("Jim","George","John", "Mary"))                  ■.
```

Επιμέρους στοιχεία ενός data frame μπορούν να προκύψουν με χρήση δεικτών,

```
> sample1[2,1]
[1] 1.84                                                        ■,
```

και ονομάτων γραμμών και αριθμών στηλών.

```
> sample1["John",2:3]
     isodima filo
John    1250 Male                                              ■.
```

Δύο είναι οι τρόποι εξαγωγής μίας στήλης:

```
> sample1[,2]
[1] 1500  900 1250 2300
> sample1$isodima
[1] 1500  900 1250 2300                                         ■.
```

3.5 Λίστες

Η **λίστα (list)** είναι ένα διάνυσμα, του οποίου τα στοιχεία είναι άλλες δομές της R. Για παράδειγμα, μία λίστα μπορεί να έχει ως πρώτο στοιχείο ένα διάνυσμα, ως δεύτερο στοιχείο ένα data frame, ως τρίτο ένα διάνυσμα, κλπ.

Μία λίστα δημιουργείται με την εντολή `list()`. Αυτή παρουσιάζει ομοιότητες με την `data.frame()`. Πιο συγκεκριμένα δίνουμε ως παραμέτρους τα αντικείμενα που θέλουμε να την απαρτίσουν. Η δήλωσή τους πρέπει να συνοδεύεται από δήλωση των ονομάτων τους, ακόμα και αν αυτά είναι ίδια με τα ονόματα των αντικειμένων[1]. Ένα παράδειγμα θα κάνει τη δημιουργία λίστας πιο κατανοητή.

```
> list1<-list(deigma=sample1,x=x,vec2=1:10)
> list1
$deigma
       ipsos isodima   filo
Jim     1.75    1500   Male
George  1.84     900   Male
John    1.81    1250   Male
Mary    1.63    2300 Female

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

$vec2
 [1]  1  2  3  4  5  6  7  8  9 10                                    ■.
```

Παρατηρείστε πως τυπώνονται τα επιμέρους στοιχεία μίας λίστας και πως το όνομα καθενός έχει το σύμβολο του δολαρίου μπροστά.

Το δολάριο χρησιμοποιείται, όπως και στα data frames, για την αναφορά σε επιμέρους στοιχεία μίας λίστας. Αυτά τα επιμέρους στοιχεία θεωρούνται ως δομές δεδομένων του τύπου από τον οποίο προήλθαν κατά τη δημιουργία της λίστας.

```
> list1$deigma
       ipsos isodima   filo
Jim     1.75    1500   Male
George  1.84     900   Male
John    1.81    1250   Male
Mary    1.63    2300 Female

> list1$x
[1] 1 2 3 4 5 6 7 8 9 10 11 12
> list1$deigma[,2]
[1] 1500  900 1250 2300                                               ■.
```

Ένας δεύτερος τρόπος αναφοράς σε επιμέρους στοιχείο μίας λίστας είναι με διπλή αγκύλη

[1] Αυτό είναι μία διαφορά από την data.frame.

```
> list1[[2]]
 [1]  1  2  3  4  5  6  7  8  9 10
> list1[[1]]
       ipsos isodima   filo
Jim     1.75    1500   Male
George  1.84     900   Male
John    1.81    1250   Male
Mary    1.63    2300 Female
> list1[[1]][,2]
[1] 1500  900 1250 2300                                                    ■.
```

Λόγω της ύπαρξης των ονομάτων, η μέθοδος με το δολάριο είναι πιθανότατα πιο εύκολη στη χρήση.

Οι λίστες εμφανίζονται πολύ συχνά στην R. Όλες οι στατιστικές διαδικασίες επιστρέφουν πληθώρα αποτελεσμάτων, οργανωμένων σε μία λίστα και ο χρήστης μπορεί να δει επιμέρους αποτελέσματα ως μεμονωμένα στοιχεία της, με τους τρόπους που είδαμε πριν.

ΚΕΦΑΛΑΙΟ 4

Αλληλεπίδραση με το χρήστη

Στην ενότητα αυτή, θα ασχοληθούμε με τον τρόπο εισαγωγής δεδομένων στην R. Στην R, υπάρχουν πολλοί τρόποι να εισάγουμε τα δεδομένα μας. Έχουμε αναφέρει, ότι μπορούμε να εισάγουμε δεδομένα ορίζοντας άμεσα τα αντικείμενα που μας ενδιαφέρουν με εντολές, όπως οι `data.frame`, `list`, `vector` και άλλες αντίστοιχες. Στην ενότητα αυτή, θα δούμε επιπλέον εντολές και διαδικασίες για την εισαγωγή δεδομένων, με τη χρήση πληκτρολογίου, μέσω του συντάκτη δεδομένων (data editor) της R και τη χρήση της εντολής `scan` και από συγκεκριμένες επιλογές, με χρήση της εντολής `menu`. Θα συνεχίσουμε παρουσιάζοντας εντολές εισαγωγής και εξαγωγής δεδομένων, τόσο σε μορφές άμεσα προσπελάσιμες από την R, όσο και σε μορφές δεδομένων άλλων προγραμμάτων, όπως το Excel, το SPSS και το Stata.

4.1 Εισαγωγή δεδομένων μέσω πληκτρολογίου

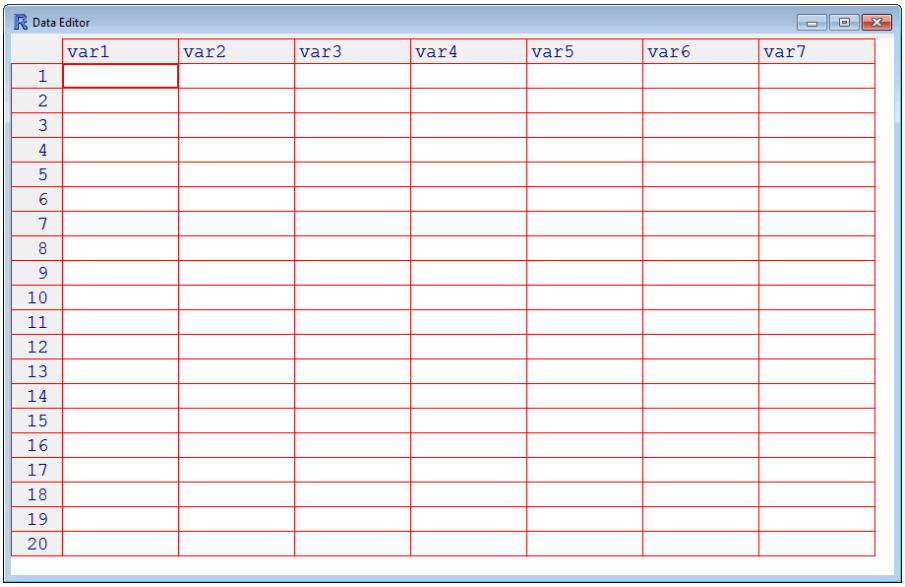
Στην ενότητα αυτή, θα ασχοληθούμε με την εισαγωγή δεδομένων, μέσω του πληκτρολογίου, με τη χρήση του συντάκτη δεδομένων (data editor) και των εντολών `scan` και `menu`.

4.1.1 Ο συντάκτης δεδομένων της R

Ο πιο εύκολος τρόπος εισαγωγής δεδομένων είναι η χρήση του data editor της R. Μπορούμε να τον καλέσουμε με την εντολή `edit(αντικείμενο)`. Αν το αντικείμενο που χρησιμοποιήσουμε είναι πλαίσιο δεδομένων, τότε ανοίγει το συντάκτη δεδομένων που έχει μορφή αντίστοιχη με όλα τα στατιστικά πακέτα (ένας πίνακας όπου κάθε γραμμή είναι μια παρατήρηση και κάθε στήλη μία μεταβλητή των δεδομένων). Μπορούμε να ανοίξουμε ένα νέο άδειο σετ δεδομένων και να το δούμε με τον data editor με τις ακόλουθες εντολές:

```
> x<-data.frame()
> x
data frame with 0 columns and 0 rows
> edit(x)                                                    ■,
```

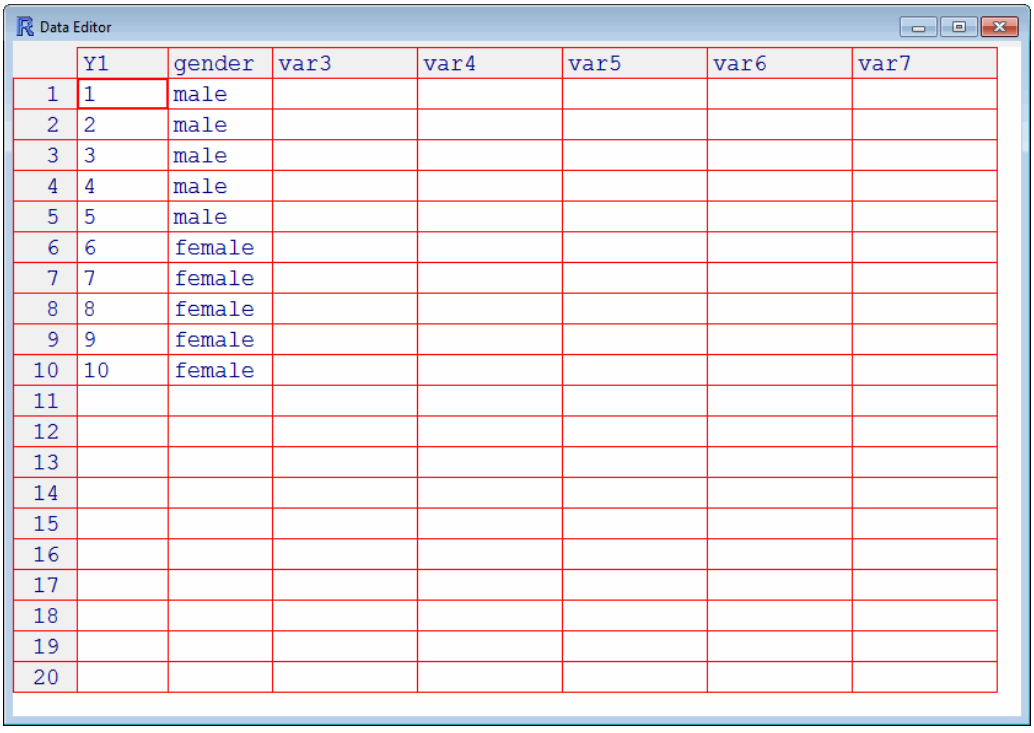
οι οποίες θα ανοίξουν το παράθυρο που ακολουθεί.



Προσοχή, επειδή δεν έγινε ανάθεση σε κάποιο αντικείμενο, ακόμα και αν κάνουμε αλλαγές στον data editor, αυτές δε θα αποθηκευτούν. Για να συντάξουμε και να αλλάξουμε ένα αντικείμενο, πρέπει να χρησιμοποιήσουμε την σύνταξη `x<-edit(x)`, η οποία θα ενημερώσει το αντικείμενο `x` με όποιες αλλαγές κάνουμε στον data editor. Έτσι λοιπόν, αν χρησιμοποιήσουμε τις εντολές:

```
> x
data frame with 0 columns and 0 rows
> x<-edit(x)                                                    ■,
```

και εισάγουμε τα ακόλουθα δεδομένα στον data editor,



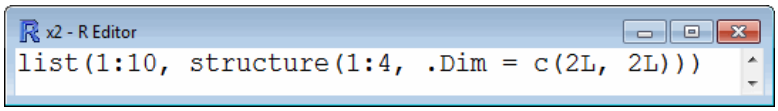
τότε θα πάρουμε το καινούριο ενημερωμένο αντικείμενο με τα ακόλουθα στοιχεία:

```
> x
   Y1 gender
1   1   male
2   2   male
3   3   male
4   4   male
5   5   male
6   6 female
7   7 female
8   8 female
9   9 female
10 10 female                                                    ■.
```

Στην περίπτωση που το υπό σύνταξη αντικείμενο δεν είναι πλαίσιο δεδομένων, τότε ανοίγει ένα νέο παράθυρο με τον R editor, στο οποίο εμφανίζεται η εντολή ορισμού του αντικειμένου, την οποία μπορούμε να αλλάξουμε ανάλογα με τις επιθυμίες μας. Έτσι λοιπόν, αν θέλουμε να συντάξουμε μία λίστα

```
> x2 <- list( x1<-1:10,  x2<-matrix(1:4,2,2))
> x2 <- edit(x2)
```

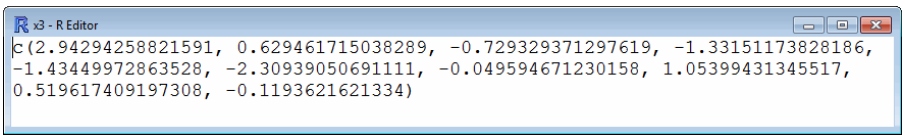
θα ανοίξει το ακόλουθο παράθυρο σύνταξης στην R,



ενώ για ένα απλό διάνυσμα

```
> x2 <- list( x1<-1:10,  x2<-matrix(1:4,2,2))
> x2 <- edit(x2)
```

θα έχουμε το ακόλουθο παράθυρο σύνταξης



4.1.2 Εισαγωγή δεδομένων με τη χρήση της εντολής `scan`

Η εντολή `scan` μπορεί να χρησιμοποιηθεί για την εισαγωγή δεδομένων, τόσο άμεσα (μέσω πληκτρολογίου), όσο και έμμεσα (από αποθηκευμένο αρχείο κειμένου). Στην ενότητα αυτή θα ασχοληθούμε με την περιγραφή της γενικής σύνταξης της συνάρτησης `scan` και με το πώς θα εισάγουμε δεδομένα μέσω πληκτρολογίου. Περαιτέρω λεπτομέρειες για το «διάβασμα» δεδομένων μέσω αρχείων θα βρείτε στην Ενότητα 4.3.

Η εντολή `scan` έχει πολλές παραμέτρους που μπορούμε να αλλάξουμε, αλλά μπορεί να πάρει και πολύ απλή μορφή. Αν για παράδειγμα θέλουμε να εισάγουμε 10 αριθμούς, τότε το μόνο που πρέπει να γράψουμε είναι `scan(n=10)`. Έτσι, με την εντολή

```
> x<-scan(n=10)
1:
```

θα αποθηκεύσουμε τις 10 τιμές που θα εισάγουμε σε ένα αντικείμενο με το όνομα `x`. Όπως βλέπουμε, αφού εκτελέσουμε την εντολή, η R περιμένει να δώσουμε την πρώτη τιμή δεδομένων. Αφού πληκτρολογήσουμε την τιμή, πατάμε το πλήκτρο ENTER

```
1: 3.4
2:                                                                    ■,
```

και συνεχίζουμε τη διαδικασία μέχρι να εισάγουμε όλα τα δεδομένα

```
1: 3.4
2: 3
3: 2
4: 45
5: 76
6: 2
7: 3
8: 5
9: 7
10: 8
Read 10 items
> x
 [1]  3.4  3.0  2.0 45.0 76.0  2.0  3.0  5.0  7.0  8.0
> is.vector(x)
[1] TRUE
> is.numeric(x)
[1] TRUE                                                              ■.
```

Όπως βλέπουμε, τα δεδομένα αποθηκεύτηκαν στο αντικείμενο `x`, το οποίο είναι ένα αριθμητικό διάνυσμα. Αν θέλουμε να εισάγουμε λιγότερα δεδομένα από αυτά που αρχικά ορίσαμε, τότε απλά πατάμε το ENTER, χωρίς να εισάγουμε δεδομένα, όπως στο παράδειγμα που ακολουθεί, όπου τελικά εισάγουμε μόνο τρεις τιμές και το διάνυσμα `x` έχει τελικά μήκος ίσο με τρία.

```
> x2<-scan(n=10)
1: 2
2: 2
3: 3
4:
```

```
Read 3 items
> x2
[1] 2 2 3
> length(x)
[1] 10
>                                                                    ■.
```

Μια άλλη παράμετρος που είναι χρήσιμη είναι η `what`, με την οποία μπορούμε να ορίσουμε τον τύπο δεδομένων που μπορούμε να να εισάγουμε. Έτσι, αν θέλουμε να εισάγουμε ένα όνομα (δηλαδή ένα αντικείμενο χαρακτήρων), τότε πρέπει να γράψουμε:

```
> x3<-scan(n=3, what=character())
1: Yiannis
2: f
3: 5456
Read 3 items
> x3
[1] "Yiannis" "f"       "5456"
> length(x3)
[1] 3
> is.vector(x3)
[1] TRUE
> is.character(x3)
[1] TRUE                                                             ■.
```

Στο παραπάνω παράδειγμα, πληκτρολογήσαμε τρία ονόματα (ή σειρές χαρακτήρων), τα οποία αποθηκεύτηκαν σε ένα διάνυσμα χαρακτήρων μήκους τρία.

Μπορείτε να δείτε τις επιπλέον επιλογές και παραμέτρους της εντολής `scan`, όπως συνήθως πληκτρολογώντας `help(scan)`. Γενικά, η εντολή αυτή χρησιμοποιείται κυρίως για διάβασμα δεδομένων από αρχεία (όπως θα δούμε στην Ενότητα 4.3), αλλά μπορεί να είναι εξαιρετικά χρήσιμη, όταν θέλουμε να πάρουμε γρήγορα δεδομένα από το χρήστη, κατά τη διάρκεια εκτέλεσης ενός μικρού προγράμματος ή συνάρτησης. Ένα χαρακτηριστικό παράδειγμα μικρού προγράμματος είναι το ακόλουθο:

```
> {
+ print('What is your name?')
+ yourname <- scan(n=1,what=character())
+ print(paste('Hello',yourname))
+ }
[1] "What is your name?"
1: John
Read 1 item
```

```
[1] "Hello John"                                                    ■.
```

Στο παραπάνω παράδειγμα, ζητείται το όνομα του χρήστη (η ερώτηση γίνεται μέσω της εντολής `print`). Η απάντηση από το χρήση δίνεται μέσω της εντολής `scan` και τέλος ο χαιρετισμός γίνεται ξανά με την εντολή `print`, αφού έχουμε χρησιμοποιήσει την εντολή `paste`, για να «κολλήσουμε» τον χαιρετισμό και το όνομα του χρήστη.

Τέλος, παρόμοια εντολή είναι η `readline`, η οποία διαβάζει ό,τι εισάγουμε σε μια γραμμή, ως αντικείμενο χαρακτήρων.

4.1.3 Εισαγωγή επιλογών με `menu`

Η εντολή `menu` μπορεί να χρησιμοποιηθεί όταν ο χρήστης καλείται να επιλέξει ανάμεσα σε προκαθορισμένες απαντήσεις. Είναι πολύ χρήσιμη για την κατασκευή ερωτηματολογίων ή για την επιλογή συγκεκριμένων ομάδων εντολών, σε συνδυασμό με τις υπο-συνθήκη εντολές `if` και `else` που θα συζητήσουμε στην Ενότητα 6.1.

Για παράδειγμα η εντολή:

```
> options<-menu(c('Ναι','Όχι'), title="Σας αρέσει το τρέξιμο;")
Σας αρέσει το τρέξιμο;

1: Ναι
2: Όχι

Selection: 1
> options
[1] 1
```

μας ρωτάει με μήνυμα στην οθόνη, αν μας αρέσει το τρέξιμο και μου δίνει δυνατότητα δύο επιλογών: Πατώντας το ένα (1) επιλέγω το ΝΑΙ, ενώ με το δύο (2) επιλέγω το ΟΧΙ. Μετά την επιλογή μας πρέπει να πατήσουμε το πλήκτρο ENTER και στις δύο περιπτώσεις. Η απάντησή μας (που εδώ είναι το ΝΑΙ), έχει αποθηκευτεί στο αντικείμενο `options`, ως ένα αριθμητικό διάνυσμα με τιμή ένα στο συγκεκριμένο παράδειγμα. Η παράμετρος `title`, ελέγχει το μήνυμα που θέλετε να εμφανίζεται στο μενού (ως τίτλος) και επεξηγεί τις απαντήσεις ή απλά θέτει ένα ερώτημα.

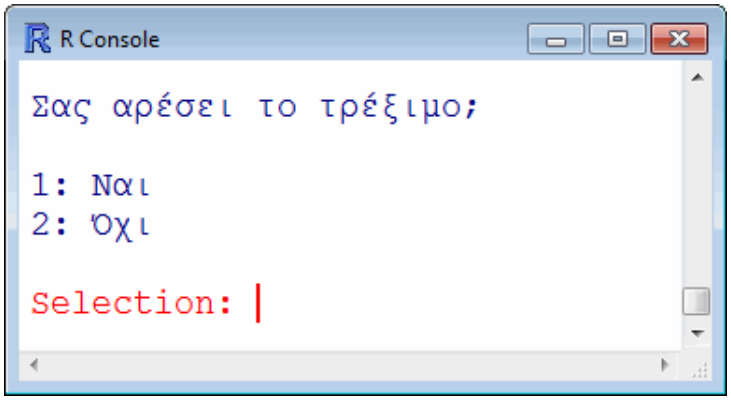
Οι επιλογές του μενού δίνονται ως ένα διάνυσμα χαρακτήρων. Αν δώσουμε άλλη τιμή από τις προκαθορισμένες (στο παράδειγμα αυτό οι τιμές 1 και 2), ή απλά πατήσουμε το ENTER, τότε η R θα μας προτρέψει να επιλέξουμε μία από τις προκαθορισμένες τιμές ή μηδέν αν θέλουμε να αποφύγουμε το μενού. Στην τελευταία περίπτωση, η τιμή του `options` θα είναι μηδέν - βλ. τις παρακάτω εντολές και το αντίστοιχο output της R.

```
> options<-menu(c('Ναι','Όχι'), title="Σας αρέσει το τρέξιμο;")
Σας αρέσει το τρέξιμο;
```

```
1: Ναι
2: Όχι

Selection: 3
Enter an item from the menu, or 0 to exit
Selection:
Enter an item from the menu, or 0 to exit
Selection: 0
> options
[1] 0                                                                    ■.
```

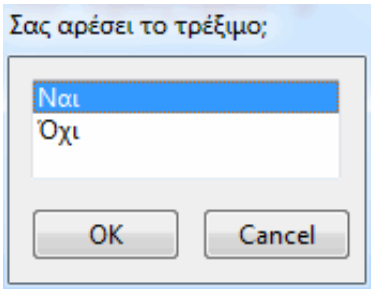
Τέλος, στην εντολή `scan` υπάρχει και η λογική παράμετρος `graphics`, η οποία μάς δίνει το μενού σε μορφή κειμένου, αν πάρει την τιμή `FALSE` (είναι και η προκαθορισμένη) και σε γραφικό περιβάλλον με τη μορφή παραθύρου των Windows να πάρει την τιμή `TRUE`. Η παραπάνω περιγραφή αναφερόταν στα μενού με τη μορφή κειμένου, τα οποία εμφανίζονται ως εξής στην οθόνη της R



ενώ αν επιλέξουμε την εντολή

```
> options<-menu(c('Ναι','Όχι'), title="Σας αρέσει το τρέξιμο;",
   graphics=T)
```

θα δούμε το μενού μας με την ακόλουθη μορφή:

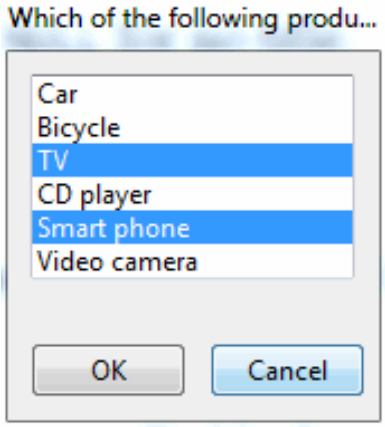


Προσοχή, στο γραφικό περιβάλλον έχουμε δύο διαφορές. Οι αριθμητικές τιμές δεν φαίνονται στην οθόνη και η επιλογή μας μπορεί να γίνει με το ποντίκι (πατώντας δύο φορές το αριστερό πλήκτρο) ή με τα βελάκια του πληκτρολογίου (και με το ENTER η οριστική επιλογή).

Παρόμοιες είναι και οι εντολές `select.list` και `tk_select.list`, οι οποίες επίσης μας ζητούν να επιλέξουμε από προεπιλεγμένες τιμές, με τη διαφορά ότι δίνεται και δυνατότητα πολλαπλών επιλογών. Έτσι, λοιπόν, για την πρώτη εντολή μπορούμε να χρησιμοποιήσουμε την σύνταξη:

```
> select.list( c('Car', 'Bicycle', 'TV', 'CD player', 'Smart phone'
+              , 'Video camera'),
+              title='Which of the following products do you own?',
+              multiple=TRUE)
[1] "Bicycle"   "CD player"                                          ■,
```

και θα λάβουμε ένα μενού όπως φαίνεται στην Εικόνα 4.1.



Εικόνα 4.1: Παράθυρο της εντολής `select.list`.

Τελικά, λαμβάνουμε ως αποτέλεσμα,

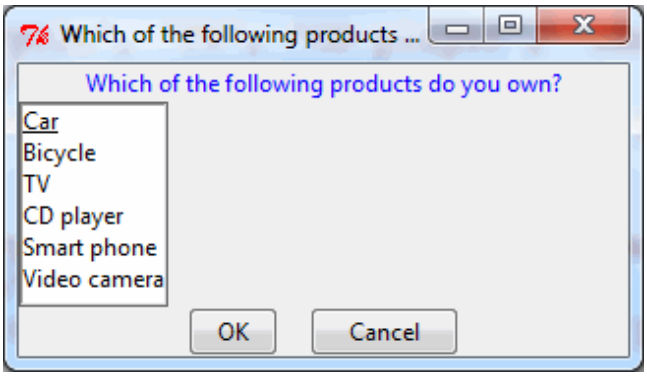
```
[1] "Bicycle"   "CD player"                                          ■,
```

δηλαδή ένα διάνυσμα χαρακτήρων, μήκους δύο, (όσο των απαντήσεων που επιλέξαμε). Σημείωση: Επιλέγουμε την πρώτη μας επιλογή και πατάμε το αριστερό πλήκτρο του ποντικιού. Για να επιλέξουμε και δεύτερη επιλογή, θα πρέπει να έχουμε πατημένο το κουμπί `Ctrl` στο πληκτρολόγιο, αλλιώς, απλά θα ορίσει τη νέα μας επιλογή ως μία και μοναδική. Αφού τελειώσουμε με τις επιλογές μας απλά πατάμε OK.

Για τη εντολή `tk_select.list` πρέπει πρώτα να φορτώσουμε το πακέτο `tcltk`. Έτσι, με τη σύνταξη:

```
> library(tcltk)
> tk_select.list(c('Car', 'Bicycle', 'TV', 'CD player',
+              'Smart phone', 'Video camera'),
+              title='Which of the following products do you own?',
+              multiple=TRUE)
```

παίρνουμε το μενού που δίνεται με το γραφικό περιβάλλον της Εικόνας 4.1



Εικόνα 4.2: Παράθυρο της εντολής `tk_select.list`.

Σημείωση: Οι πολλαπλές επιλογές σε αυτό το μενού γίνονται απλά, πατώντας το αριστερό πλήκτρο του ποντικιού. Η απο-επιλογή γίνεται ξανά πατώντας το αριστερό πλήκτρο σε μία από τις ήδη ενεργές μας επιλογές.

4.2 Εντολές για διαχείριση των καταλόγων και αρχείων

Πριν προχωρήσουμε στις εντολές εισαγωγής δεδομένων από αρχεία, θα αναφερθούμε σε κάποιες εντολές φακέλων (folders) στην ορολογία των Windows ή καταλόγων (directories) στην ορολογία του Unix και παλαιότερων εκδοχών των Windows. Ο όρος των καταλόγων χρησιμοποιείται και στην R, αντί των φακέλων που είναι πιο σύγχρονος. Οι όροι αυτοί θα χρησιμοποιούνται εναλλάξ στο σύγγραμμα αυτό για τις ίδιες έννοιες.

Στην R υπάρχει πάντα ένας κατάλογος εργασίας (working directory), ο οποίος αναφέρεται στον φάκελο (ή κατάλογο) που βρίσκεται το αρχείο του τρέχοντος χώρου εργασίας (δηλαδή του .Rdata). Αν απλά ανοίξουμε την R, τότε χρησιμοποιείται ο προκαθορισμένος χώρος εργασίας και ο αντίστοιχος φάκελος που είναι αυτός, όπου έχει γίνει η εγκατάσταση της R. Συνήθως, είναι ένας κατάλογος της μορφής `C: /Users/user/Documents`. Για να δείτε σε ποιο φάκελο δουλεύετε, μπορείτε να χρησιμοποιήσετε την εντολή `getwd` (get working directory):

```
> getwd()
[1] "C: /Users/user/Documents"                                    ■.
```

Όπως βλέπετε από το αποτέλεσμα της εντολής `getwd`, στις ονομασίες των φακέλων/καταλόγων δεν χρησιμοποιείται η ανάστροφη (ή αριστερόστροφη) κάθετος «\» (όπως έχουμε συνηθίσει στα Windows), αλλά η δεξιόστροφη κάθετος «/». Εναλλακτικά, μπορούμε να χρησιμοποιήσουμε δύο ανάστροφες καθέτους «\\», αλλά ο πρώτος τρόπος φαίνεται να επικρατεί στην κοινότητα των χρηστών της R.

Έτσι, αν θέλουμε να αλλάξουμε τον κατάλογο εργασίας, έτσι ώστε να δουλεύουμε στο φάκελο `c: \rfiles\`, τότε θα πρέπει να χρησιμοποιήσουμε την εντολή `setwd` με την ακόλουθη σύνταξη:

```
setwd('c: /rfiles')
```

ή εναλλακτικά

```
setwd('c:\\rfiles')                                    ■.
```

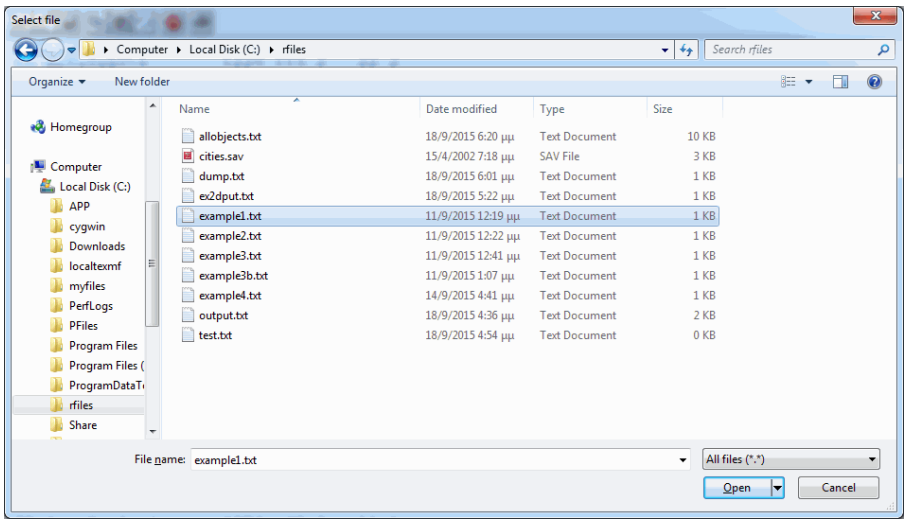
Προσοχή, για να επιτύχει η αλλαγή του καταλόγου εργασίας, θα πρέπει να υπάρχει στο νέο κατάλογο (εδώ το `c:\rfiles`) ένα αρχείο χώρου εργασίας της R (δηλαδή ένα αρχείο .Rdata).

Πριν ολοκληρώσουμε αυτή την μικρή ενότητα, να σημειώσουμε ότι μπορούμε να χρησιμοποιήσουμε την εντολή `paste` για να φτιάχνουμε διαδρομές (paths) καταλόγων, αλλά και ονόματα αρχείων. Για παράδειγμα, μπορούμε να γράψουμε:

```
> drive <- 'c:'
> path1 <- 'rfiles'
> path2 <- 'temp'
> newdir <- paste ( drive, path1, path2, sep='\\' )
> newdir
[1] "c:\\rfiles\\temp"
> setwd(newdir)                                        ■.
```

Με αυτό τον τρόπο μπορούμε να αλλάζουμε τα ονόματα αρχείων (με το πλήρες μονοπάτι τους), ή καταλόγων εύκολα και να τα χρησιμοποιούμε πολλαπλές φορές, χωρίς να γράφουμε τα πλήρη ονόματα τους, που συνήθως είναι μεγάλα.

Τέλος, εξαιρετικά χρήσιμη είναι η εντολή `file.choose`, με την οποία ανοίγει το παράθυρο της Εικόνας 4.3, το οποίο μας καλεί να επιλέξουμε ένα αρχείο. Με το που επιλέγουμε το επιθυμητό αρχείο, καταγράφεται η ονομασία του και η πλήρης διαδρομή του φακέλου του. Μπορούμε να το αποθηκεύσουμε μετά σε ένα αντικείμενο και να το χρησιμοποιούμε ως εισερχόμενο στις εντολές που περιγράφονται σε αυτή την ενότητα. Έτσι, λοιπόν, με την εντολή `file.choose` έχουμε



Εικόνα 4.3: Menu που ανοίγει με την εντολή `file_choose`

την ακόλουθη σύνταξη και αποτελέσματα:

```
> file.choose()
```

```
[1] "C:\\rfiles\\example1.txt"
> onoma_arxeiou <- file.choose()
> onoma_arxeiou
[1] "C:\\rfiles\\example1.txt"                                    ■.
```

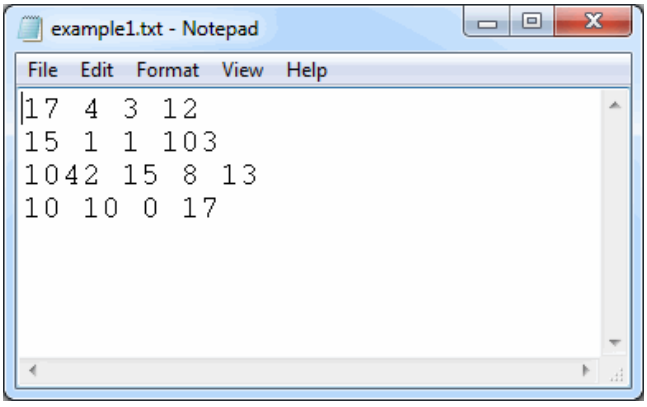
4.3 Εισαγωγή δεδομένων από αρχεία

Μέχρι τώρα είδαμε πώς μπορούμε να εισάγουμε τις τιμές στις μεταβλητές μας, είτε μέσω εντολών ορισμού των αντικειμένων, είτε μέσω πληκτρολογίου χρησιμοποιώντας το συντάκτη δεδομένων, ή την εντολή `scan`. Στην πράξη, αυτό είναι χρήσιμο μόνο σε περιορισμένες και ειδικές περιπτώσεις. Συνήθως, τα δεδομένα μάς δίνονται καταχωρημένα ήδη σε κάποιο αρχείο, συνήθως ως χαρακτήρες ASCII (δηλαδή ως απλό αρχείο κειμένου[1]). Για να μην ξανα-πληκτρολογούμε τα ήδη διαθέσιμα δεδομένα, η R παρέχει εντολές εισαγωγής τιμών από αρχεία κειμένου. Εμείς θα ασχοληθούμε με τρεις από αυτές: την `scan` (ξανά), την `read.table` και την `source`.

4.3.1 Εισαγωγή δεδομένων αρχείων με την εντολή `scan`

Σε συνέχεια της Ενότητας 4.1.2, όπου είδαμε πώς εισάγουμε δεδομένα μέσω πληκτρολογίου με την εντολή `scan`, εδώ θα δούμε πώς θα χρησιμοποιήσουμε αυτή την εντολή για να εισάγουμε δεδομένα από ένα αρχείο. Αν, στη θέση της πρώτης παραμέτρου, δώσουμε το όνομα ενός αρχείου κειμένου, τότε θα εισάγουμε όλα τα περιεχόμενα του σε ένα διάνυσμα. Αν το αρχείο δεν βρίσκεται στον ίδιο φάκελο, από τον οποίο εκτελείται η R (δηλαδή στον κατάλογο εργασίας), τότε θα πρέπει στο όνομα αρχείου να δοθεί και η πλήρης «διαδρομή» (path) του. Αυτό θα γίνει πιο κατανοητό με το ακόλουθο παράδειγμα:

Έστω, ότι στο φάκελο `c:\rfiles` βρίσκεται το αρχείο example1.txt με το περιεχόμενο, όπως φαίνεται στην Εικόνα 4.4.



Εικόνα 4.4: Περιεχόμενα αρχείου κειμένου `example1.txt`

Προσέξτε στη συνέχεια πώς δηλώνουμε το όνομα του αρχείου στην εντολή `scan()`.

[1] Απλά αρχεία κειμένου είναι για παράδειγμα αυτά που φτιάχνονται με το Notepad των Windows. Τα αρχεία με κατάληξη .doc ή .docx που δημιουργούνται με το Microsoft Word, δεν είναι απλά αρχεία κειμένου.

```
> scan("c:/rfiles/example1.txt")
 [1] 17 4 3 12 15 1 1 103 1042 15 8 13 10 10 0 17
> vec1<-scan("c:/rfiles/example1.txt")
> vec1
[1] 17 4 3 12 15 1 1 103 1042 15 8 13 10 10 0 17                    ■.
```

Το όνομα `c:/rfiles/example1.txt` είναι η πλήρης διαδρομή προς το αρχείο. Το αρχείο κειμένου πρέπει να τελειώνει με μία κενή γραμμή, δηλαδή μετά την τελευταία τιμή του πρέπει να έχει πατηθεί το πλήκτρο ENTER.

Σημειώνεται ότι, αφού το αποτέλεσμα της `scan()` είναι διάνυσμα, μπορούμε να το εκχωρήσουμε (όπως στο παραπάνω παράδειγμα) και να το χειριστούμε όπως όλα τα διανύσματα. Για παράδειγμα, μπορούμε το περιεχόμενο του αρχείου example1.txt να το εκχωρήσουμε σε ένα πίνακα με 4 στήλες, χρησιμοποιώντας και την παράμετρο `byrow=T`, αν θέλουμε κάθε γραμμή του αρχείου να αναλογεί σε μία γραμμή του πίνακα.

```
> mat1<-matrix(scan("c:/rfiles/example1.txt"), ncol=4,byrow=T)
Read 16 items
> mat1
     [,1] [,2] [,3] [,4]
[1,]   17    4    3   12
[2,]   15    1    1  103
[3,] 1042   15    8   13
[4,]   10   10    0   17                                            ■.
```

Το αρχείο κειμένου δεν είναι αναγκαίο να έχει τον ίδιο αριθμό τιμών ανά γραμμή. Μπορούν επίσης, μεταξύ των τιμών να παρεμβάλλονται πολλές κενές γραμμές. Στην πράξη, η εντολή `scan` σαρώνει το αρχείο ακολουθιακά και για αυτό κενοί χαρακτήρες δεν την επηρεάζουν.

Η `scan()` έχει και άλλες παραμέτρους και παραλλαγές, οι οποίες για παράδειγμα επιτρέπουν την εισαγωγή δεδομένων τύπου character, την εισαγωγή των δεδομένων σε λίστα με στοιχεία διαφορετικού τύπου κλπ., αλλά εδώ δε θα ασχοληθούμε παραπάνω.

Φυσικά, μπορούμε να χρησιμοποιήσουμε την εντολή `scan` (όπως και τις επόμενες αυτής της ενότητας) σε συνδυασμό με την εντολή `file.choose` που περιγράψαμε στην Ενότητα 4.2. Έτσι, λοιπόν, η σύνταξη:

```
> onoma_arxeiou <- file.choose()
> onoma_arxeiou
[1] "C:\\rfiles\\example1.txt"
> mat1<-matrix(scan(onoma_arxeiou), ncol=4,byrow=T)
Read 16 items
> mat1
     [,1] [,2] [,3] [,4]
[1,]   17    4    3   12
```

```
[2,]    15     1     1  103
[3,] 1042    15     8   13
[4,]   10    10     0   17
>
```

θα έχει ακριβώς τα ίδια αποτελέσματα, όπως η σύνταξη παραπάνω, όπου ορίσαμε πλήρως το όνομα του αρχείου με τη διαδρομή του.

Τέλος, να σημειώσουμε ότι η εντολή `scan` είναι ιδιαίτερα χρήσιμη όταν θέλουμε να διαβάσουμε μεγάλα σετ δεδομένων, που αποτελούνται από ίδιου τύπου τιμές (π.χ. αριθμητικές), γιατί είναι πολύ πιο γρήγορη από άλλες, όπως η `read.table` που περιγράφεται στην Ενότητα 4.3.2 που ακολουθεί.

4.3.2 Η εντολή `read.table`

Η δεύτερη εντολή εισαγωγής δεδομένων από αρχείο κειμένου που θα δούμε, είναι η εντολή `read.table()`. Και σε αυτή την εντολή, η πρώτη παράμετρος είναι ένα όνομα αρχείου (με την ίδια μορφή όπως η `scan`). Η διαφορά από την εντολή `scan` είναι ότι η εντολή `read.table` εισάγει τα δεδομένα στην R, ως πλαίσια δεδομένων (data frames), αντί για διανύσματα. Δηλαδή, μπορούμε να έχουμε διαφορετικούς τύπους δεδομένων ανά στήλη. Γενικά, τα δεδομένα που διαβάζονται με αυτή την εντολή πρέπει να έχουν δομή, που να είναι συμβατή με τα πλαίσια δεδομένων. Για παράδειγμα, όλες οι γραμμές θα πρέπει να έχουν τον ίδιο αριθμό δεδομένων (αντίστοιχα, το ίδιο ισχύει και για τις στήλες), ενώ τυχόν κενές ή χαμένες τιμές (missing values) θα πρέπει να είναι εγγεγραμμένες ως `NA`. Οι παράμετροι της εντολής ελέγχουν τον τρόπο που τα δεδομένα πρέπει να διαβαστούν. Οι σημαντικότερες παράμετροι επεξηγούνται στον Πίνακα 4.1 και στη συνέχεια αναλύονται με παραδείγματα.

`header`	Λογική παράμετρος. Τιμή `TRUE` δηλώνει ότι η πρώτη γραμμή του αρχείου δίνει ονόματα στηλών.
`sep`	Δηλώνει το χαρακτήρα που χωρίζει τιμές στο αρχείο. Καλό είναι να μην δίνεται τιμή παρά μόνο αν υπάρχει λόγος.
`col.names`	Διάνυσμα με ονόματα για τις στήλες του data frame.
`row.names`	Διάνυσμα με ονόματα για τις γραμμές του data frame.
`dec = "."`	Σημείο που ορίζει τα δεκαδικά ψηφία
`na.strings="NA"`	Τρόπος που θα διαβάσει τα missing values
`nrows = -1`	Μέγιστος αριθμός γραμμών. Η προκαθορισμένη τιμή του -1 δε θέτει όριο.
`skip = 0`	Ο αριθμός των αρχικών γραμμών που δε θα διαβαστούν.

Πίνακας 4.1: Παράμετροι της εντολής `read.table()` της R

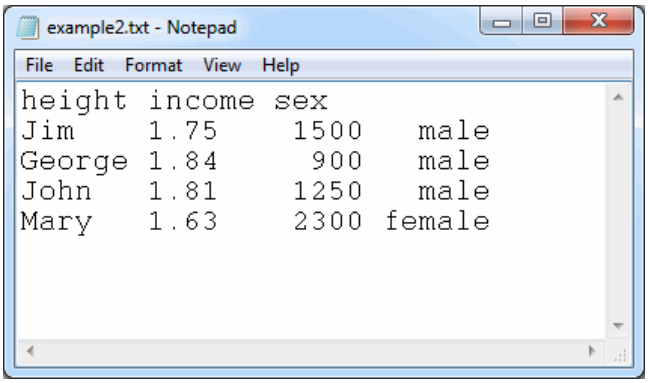
Καλό είναι, κάθε γραμμή του αρχείου να αναλογεί σε μία γραμμή του data frame και, ενδεχομένως, η πρώτη γραμμή να δίνει ονόματα στηλών και η πρώτη στήλη τα ονόματα των γραμμών. Η γραμμή με τα ονόματα στηλών δεν χρειάζεται να περιέχει όνομα για τη στήλη που έχει ονόματα γραμμών. Ανάλογα με τη μορφή του αρχείου, η R κάποιες φορές «αποφασίζει» για τις τιμές κάποιων από τις παραπάνω παραμέτρους.

Για παράδειγμα, αν η πρώτη γραμμή έχει ένα στοιχείο λιγότερο από τις επόμενες, η R τη θεωρεί ως γραμμή, που περιέχει τα ονόματα των στηλών και θεωρεί ότι στις επόμενες γραμμές η πρώτη στήλη δίνει τα ονόματα των γραμμών (εφόσον δεν δηλωθούν ονόματα γραμμών). Εναλλακτικά, η R αν δεν δηλωθεί τίποτα, θα θεωρήσει ως στήλη με ονόματα γραμμών την πρώτη στήλη, που περιέχει τιμές τύπου χαρακτήρων (character), οι οποίες δεν επαναλαμβάνονται. Αν δοθεί η παράμετρος `row.names=NULL`, τότε οι γραμμές του πλαισίου δεδομένων (data frame) δεν θα λάβουν ονόματα, αλλά αρίθμηση. Αν δεν υπάρχει πρώτη γραμμή με ένα στοιχείο λιγότερο και δεν δοθούν ονόματα στηλών, τότε η R δίνει στις στήλες ονόματα της μορφής: `V1, V2, ...` κλπ.

```
> read.table("c:/rfiles/example1.txt")
    V1 V2 V3  V4
1   17  4  3  12
2   15  1  1 103
3 1042 15  8  13
4   10 10  0  17                                                    ■.
```

Στο πρώτο παράδειγμα, το αρχείο είχε ίδιο αριθμό στοιχείων σε όλες τις γραμμές, αλλά δεν δηλώθηκαν ονόματα γραμμών ή στηλών και δεν υπήρχε στήλη με στοιχεία τύπου character. Έτσι, οι στήλες έλαβαν τα τυπικά ονόματα και οι γραμμές αρίθμηση.

Στο επόμενο παράδειγμα, χρησιμοποιούμε το αρχείο `example2.txt` που βρίσκεται στον ίδιο κατάλογο με το `example1.txt` και έχει περιεχόμενο όπως στην Εικόνα 4.5.



Εικόνα 4.5: Περιεχόμενα αρχείου κειμένου `example2.txt`

```
> read.table("c:/rfiles/example2.txt")
       height income    sex
Jim      1.75   1500   male
George   1.84    900   male
```

```
John      1.81   1250   male
Mary      1.63   2300 female                                    ■.
```

Η πρώτη γραμμή περιείχε ένα στοιχείο λιγότερο και έτσι εκλήφθηκε ως γραμμή ονομάτων στηλών. Το πρώτο στοιχείο κάθε επόμενης γραμμής εκλήφθηκε ως όνομα γραμμής. Γενικά, καλό είναι τα αρχεία κειμένου να έχουν εξαρχής μία σχετικά δομημένη μορφή (με ονόματα μεταβλητών), ώστε με απλή εφαρμογή της `read.table()` να εισάγονται ορθά τα αποτελέσματα.

Προσοχή, με την προηγούμενη σύνταξη διαβάσαμε τα δεδομένα μας και εμφανίστηκαν στην οθόνη, αλλά δεν αποθηκεύτηκαν σε κάποιο αντικείμενο στο χώρο εργασίας μας στην R. Για να το επιτύχουμε αυτό, πρέπει να κάνουμε ανάθεση του `read.table` σε ένα αντικείμενο. Έτσι η εντολή

```
> ex2 <- read.table("c:/rfiles/example2.txt")
```

θα αποθηκεύσει τα στοιχεία του αρχείου `example2.txt`, στο αντικείμενο `ex2` ως ένα πλαίσιο δεδομένων. Αυτό μπορούμε να το ελέγξουμε με την εντολή `class`, ή να το επιβεβαιώσουμε με την εντολή `is.data.frame`:

```
> class(ex2)
[1] "data.frame"
> is.data.frame(ex2)
[1] TRUE                                                        ■.
```

Προσοχή, κάνοντας ανάθεση στο αντικείμενο `ex2`, τα περιεχόμενα του αρχείου δεν εμφανίζονται άμεσα στην οθόνη, όπως προηγουμένως, αλλά μόνο αν ζητήσουμε να δούμε το αντικείμενο `ex2` απλά πληκτρολογώντας το όνομα του, δηλαδή:

```
> ex2
       height income    sex
Jim      1.75   1500   male
George   1.84    900   male
John     1.81   1250   male
Mary     1.63   2300 female                                    ■.
```

Χρήσιμη είναι επίσης, η παράμετρος `stringsAsFactors`, η οποία αναφέρεται στον τρόπο διαχείρισης των μεταβλητών με χαρακτήρες. Η προκαθορισμένη τιμή μετατρέπει τις μεταβλητές χαρακτήρων σε παράγοντες (factors).

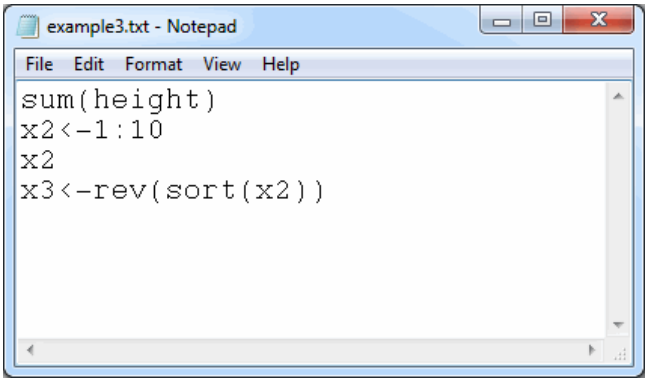
Σημείωση: η R μπορεί να εισάγει δεδομένα και από αρχεία που δημιουργούνται με βάση άλλα προγράμματα, όπως π.χ. Excel, SPSS, κλπ. Θα αναφερθούμε σε αυτό στην Ενότητα 4.6.

4.3.3 Διάβασμα πηγαίου κώδικα με την εντολή `source`

Μία συναφής εντολή με τις παραπάνω είναι η `source()`, η οποία δέχεται ως παράμετρο ένα αρχείο με εντολές της R, τις οποίες και εκτελεί σαν να τις πληκτρολογούσαμε τη στιγμή που

εκτελείται. Τα αποτελέσματα της εκτέλεσης δεν φαίνονται στην οθόνη, αλλά τυχόν αντικείμενα που δημιουργούνται παραμένουν στη μνήμη για μετέπειτα χρήση.

Έστω, ότι το αρχείο `example3.txt` έχει το περιεχόμενο όπως στην Εικόνα 4.6.



Εικόνα 4.6: Περιεχόμενα αρχείου εντολών `example3.txt`

Πριν «τρέξουμε» τις εντολές αυτές με τη συνάρτηση `source`, δημιουργούμε ένα αντικείμενο με το όνομα `height` (που χρησιμοποιείται στο αρχείο `example3.txt`) και ελέγχουμε αν στο χώρο εργασίας μας υπάρχουν τα αντικείμενα `x2` και `x3`.

```
> height
Error: object 'height' not found
> height<-ex2$height
> x2
Error: object 'x2' not found
> x3
Error: object 'x3' not found
```
■.

Μετά, τρέχουμε τις εντολές του αρχείου `example3.txt` με την ακόλουθη σύνταξη:

```
> source("c:/rfiles/example3.txt")
> x2
 [1]  1  2  3  4  5  6  7  8  9 10
> x3
 [1] 10  9  8  7  6  5  4  3  2  1
```
■.

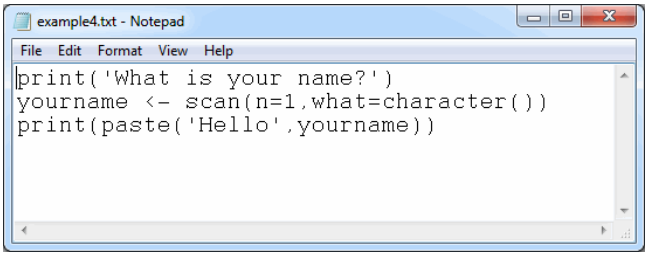
Όπως βλέπουμε, τα αντικείμενα `x2` και `x3` δημιουργήθηκαν και είναι διαθέσιμα μετά την εκτέλεση της εντολής, `source`. Επιπλέον, παρόλο που η πρώτη εντολή (`sum(height)`) εκτελέστηκε στην R και υπολογίστηκε το σχετικό άθροισμα χωρίς όμως να αποθηκευτεί σε κάποιο αντικείμενο, το αποτέλεσμα της δεν εμφανίζεται στην οθόνη, όπως όταν δουλεύουμε κατευθείαν στην κονσόλα. Όμοια, το αντικείμενο `x2` δεν εμφανίζεται στην οθόνη όταν το καλούμε απλά με το όνομα του. Έτσι, λοιπόν, αν θέλουμε με τη χρήση της εντολής `source` να δούμε στην οθόνη κάποιο αποτέλεσμα ή αντικείμενο, τότε θα πρέπει να χρησιμοποιήσουμε την εντολή `print`. Για το λόγο αυτό, συχνά χρησιμοποιείται η ορολογία «εκτύπωση στην οθόνη». Έτσι, το νέο αρχείο με το όνομα `example3.txt` και οι εντολές

```
print(sum(height))
x2<-1:10
print(x2)
x3<-rev(sort(x2))
```

θα μας δώσουν ως αποτέλεσμα:

```
> source("c:/rfiles/example3b.txt")
[1] 7.03
[1]  1  2  3  4  5  6  7  8  9 10
```

όταν διαβαστούν με την εντολή `source`.



Εικόνα 4.7: Περιεχόμενα αρχείου εντολών `example4.txt`

Τέλος, επιστρέφοντας στο παράδειγμα της Ενότητας 4.1.2 για την εντολή `scan`, αν έχουμε το αρχείο `example4.txt`, τα περιεχόμενα του οποίου φαίνονται στην Εικόνα 4.7, τότε θα πάρουμε το εξής αποτέλεσμα:

```
> source('c:/rfiles/example4.txt')
[1] "What is your name?"
1: John
Read 1 item
[1] "Hello John"                                                    ■.
```

4.4 Εντολές αποθήκευσης και ανάκτησης δεδομένων

4.4.1 Οι εντολές `save` και `load`

Στην R, είναι πολύ εύκολο να μεταφέρουμε τα δεδομένα μας σε άλλο υπολογιστή. Η πιο εύκολη λύση είναι να βρούμε το αρχείο εργασίας (δηλαδή το αρχείο .Rdata) και να το αντιγράψουμε σε ένα φορητό αποθηκευτικό μέσο, μέσω των εντολών των Windows. Με αυτό τον τρόπο αντιγράφουμε όλα τα δεδομένα του χώρου εργασίας.

Μπορούμε να συνεχίσουμε την εργασία μας σε ένα χώρο εργασίας που έχουμε αποθηκεύσει σε προηγούμενη μας συνεδρία, επιλέγοντας το αρχείο στο περιβάλλον Windows (πατώντας δύο φορές το αριστερό πλήκτρο του ποντικιού). Με αυτό τον τρόπο, η R θα ανοίξει φορτώνοντας το συγκεκριμένο χώρο εργασίας (και όλα τα αντικείμενα του), θέτοντας ως φάκελο εργασίας τον αντίστοιχο που βρίσκεται το αρχείο «.Rdata ». Γενικά, προτείνουμε για κάθε ανάλυση, εργασία,

ή ερευνητικό πρόγραμμα, να διατηρείται σε ξεχωριστό φάκελο με αντίστοιχο .Rdata αρχείο, που θα περιέχει μόνο τα σχετικά αντικείμενα, έτσι ώστε να μπορείτε εύκολα να τα εντοπίσετε και να τα διαχειριστείτε.

Εναλλακτικά, στην R υπάρχει η εντολή `save`, που αποθηκεύει πολλά αντικείμενα ταυτόχρονα. Για παράδειγμα, τα αντικείμενα `x` και `x2`

```
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> x2
     [,1] [,2]
[1,]    1    6
[2,]    2    7
[3,]    3    8
[4,]    4    9
[5,]    5   10
```

μπορούν να αποθηκευτούν στο αρχείο `test.txt` με την εντολή:

```
> save( x,x2,file='c:/rfiles/test.txt')
> rm(x)
> rm(x2)                                                    ■,
```

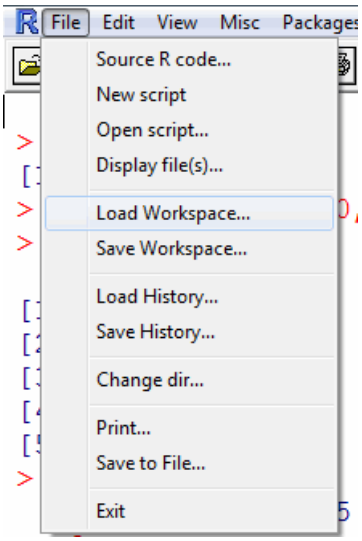
ενώ η ανάκτηση των δεδομένων του αρχείου `test.txt` γίνεται με την εντολή:

```
> load(file='c:/rfiles/test.txt')
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> x2
     [,1] [,2]
[1,]    1    6
[2,]    2    7
[3,]    3    8
[4,]    4    9
[5,]    5   10                                              ■.
```

Η αποθήκευση όλων των δεδομένων (αντικειμένων) του χώρου εργασίας επιτυγχάνεται με την εντολή `save.image`, που δεν είναι τίποτα άλλο, από συντομογραφία για την εντολή

```
save(list = ls(all = TRUE), file = ".RData")                ■.
```

Εναλλακτικά, μπορούμε να αποθηκεύσουμε το .Rdata, ή να το ανακτήσουμε αρχείο, χρησιμοποιώντας τις λειτουργίες `Load Workspace` ή `Save Workspace`, αντίστοιχα, που είναι διαθέσιμες στο μενού `file` της R (βλ. Εικόνα 4.8).



Εικόνα 4.8: Απεικόνιση της λειτουργίας `load Workspace` στο μενού `file` της R

4.4.2 Η εντολές `dput` και `dget`

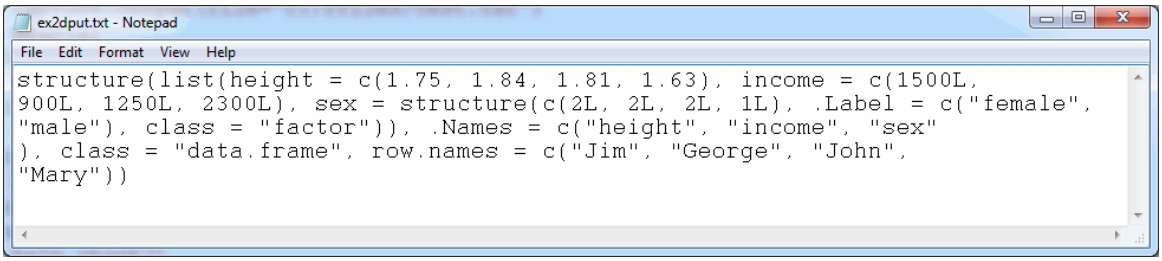
Η εντολή `dput` αποθηκεύει ένα αντικείμενο ως την εντολή ορισμού της στην R. Έτσι, λοιπόν, το αρχείο, που αποθηκεύει τα δεδομένα, είναι ένα απλό αρχείο κειμένου που μπορεί να ανοίξει από το notepad των Windows, αλλά μπορεί να διαβαστεί μόνο στην R. Η εντολή ανάκτησης είναι `dget`. Για παράδειγμα, έστω το αντικείμενο `ex2` που είχαμε διαβάσει από το αρχείο `example2.txt` στην Ενότητα 4.3.2.

```
> ex2 <- read.table("c:/rfiles/example2.txt")
> ex2
      height income    sex
Jim     1.75   1500   male
George  1.84    900   male
John    1.81   1250   male
Mary    1.63   2300 female
> dput(ex2, 'c:/rfiles/ex2dput.txt')                    ■.
```

Με την παραπάνω εντολή αποθηκεύσαμε το αντικείμενο `ex2` στο αρχείο `ex2dput.txt` με διαδρομή φακέλου `c:\rfiles`. Να σημειώσουμε ότι, αν δεν βάλουμε διαδρομή φακέλου, αλλά γράψουμε μόνο το όνομα του αρχείου, τότε αυτό θα αποθηκευτεί στο φάκελο εργασίας. Όπως βλέπουμε από την Εικόνα 4.9, το αρχείο είναι μια εντολή της R. Μπορούμε να μεταφέρουμε τα δεδομένα πίσω στην R με ένα απλό `copy` και `paste`.

Επίσης, πρέπει να υπογραμμίσουμε ότι, ενώ έχουν αποθηκευτεί το όνομα των μεταβλητών, των παρατηρήσεων και ποιες μεταβλητές είναι κατηγορικές, δεν έχει αποθηκευτεί το όνομα του αντικειμένου (το οποίο πρέπει να οριστεί όταν ξανακαλέσουμε το αντικείμενο) με την εντολή `dget`. Έτσι, η εντολή:

```
> dget('c:/rfiles/ex2dput.txt')
```



Εικόνα 4.9: Απεικόνιση του αρχείου `ex2dput.txt` που δημιουργήθηκε με την εντολή `dput`

```
       height income    sex
Jim      1.75   1500   male
George   1.84    900   male
John     1.81   1250   male
Mary     1.63   2300 female
```

εισάγει το αντικείμενο στην R χωρίς να το αποθηκεύει στο χώρο εργασίας, αλλά απλά το εμφανίζει στην οθόνη, ενώ η εντολή:

```
> x<-dget('c:/rfiles/ex2dput.txt')
> x
       height income    sex
Jim      1.75   1500   male
George   1.84    900   male
John     1.81   1250   male
Mary     1.63   2300 female
```

το εισάγει και το αποθηκεύει στο αντικείμενο `x`. Μια άλλη χρήσιμη λειτουργία της εντολής `dput` είναι ότι, αν δεν ορίσουμε κάποιο αρχείο αποθήκευσης, τότε εμφανίζει την εντολή ορισμού στην οθόνη, όπως φαίνεται παρακάτω.

```
> dput(ex2)
structure(list(height = c(1.75, 1.84, 1.81, 1.63), income = c(1500L
   , 900L, 1250L, 2300L), sex = structure(c(2L, 2L, 2L, 1L),
   .Label = c("female", "male"), class = "factor")), .Names = c("
   height", "income", "sex"), class = "data.frame", row.names = c("
   Jim", "George", "John", "Mary"))                                  ■.
```

Αυτό μπορεί να είναι εξαιρετικά χρήσιμο, όταν θέλουμε να συμπεριλάβουμε με ένα απλό `copy` και `paste` τα δεδομένα μας σε ένα πρόγραμμα εντολών, τα οποία θα μπορούν να διαβαστούν με την εντολή `source`.

Τέλος, η παράμετρος `control` ελέγχει τον τρόπο εμφάνισης των εντολών ορισμού. Για παράδειγμα, η εντολή:

```
> dput(ex2, control = c("showAttributes"))
```

```
structure(list(height = c(1.75, 1.84, 1.81, 1.63), income =  c
   (1500, 900, 1250, 2300), sex = structure(c(2, 2, 2, 1), .Label
    = c("female", "male"), class = "factor")), .Names = c("height"
   , "income", "sex"), class = "data.frame", row.names = c("Jim", "
   George", "John", "Mary"))
```

ξεφορτώνεται τα ενοχλητικά «L» που εμφανίζονται μετά από τους ακέραιους αριθμούς, ενώ η εντολή:

```
> dput(ex2, control ='useSource')
list(height = c(1.75, 1.84, 1.81, 1.63), income = c(1500, 900,
1250, 2300), sex = c(2, 2, 2, 1))
```

δεν αποθηκεύει τις ιδιότητες του αρχείου, πάρα μόνο τις τιμές σε μια απλή λίστα, όπως φαίνεται παρακάτω:

```
> list(height = c(1.75, 1.84, 1.81, 1.63), income = c(1500, 900,
+ 1250, 2300), sex = c(2, 2, 2, 1))
$height
[1] 1.75 1.84 1.81 1.63

$income
[1] 1500  900 1250 2300

$sex
[1] 2 2 2 1
```

■.

4.4.3 Η εντολή `dump`

Η εντολή `dump` είναι παρόμοια με τη `dput`, μόνο που μπορεί να αποθηκεύσει πολλά αρχεία ταυτόχρονα. Η αποθήκευση γίνεται πάλι σε αρχείο κειμένου, χρησιμοποιώντας εντολές ορισμού των αντικειμένων, αλλά τώρα αυτά αποθηκεύονται με το πλήρες όνομά τους. Αφού τα αρχεία αυτά αποτελούνται από εντολές της R, μπορούμε να ανακτήσουμε τα αντικείμενα που έχουν αποθηκευτεί σε ένα αρχείο με την εντολή `dump`, χρησιμοποιώντας την εντολή `source`. Έτσι, η εντολή

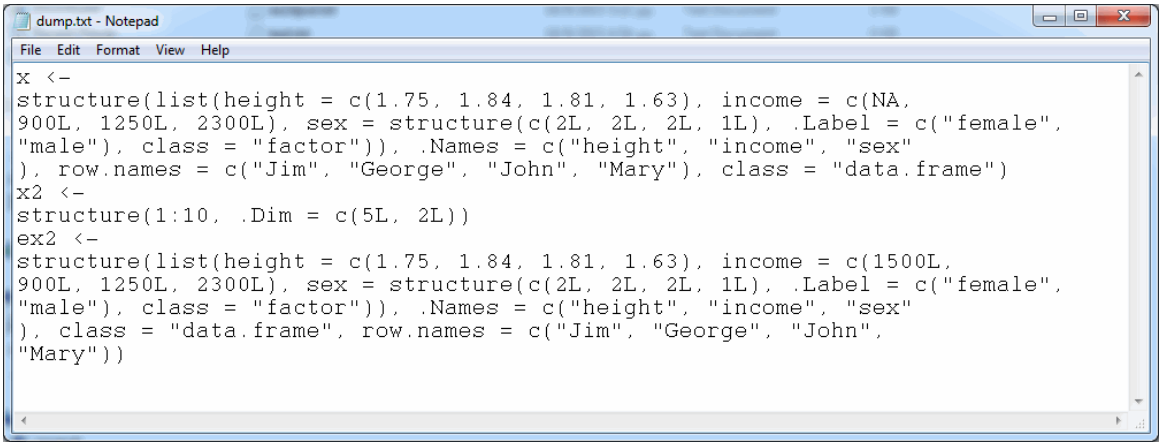
```
dump(c( 'x', 'x2', 'ex2' ), 'c:/rfiles/dump.txt')
```

θα μας δώσει το αρχείο της εικόνας 4.10 αποθηκευμένο στο φάκελο `c:\rfiles` και με όνομα `dump.txt`, το οποίο θα περιέχει τις εντολές ορισμού των αντικειμένων `x`, `x2` και `ex2`. Η ανάκτηση τους μπορεί να γίνει με την εντολή:

```
source('c:/rfiles/dump.txt')
```

■.

Προσοχή, αντικείμενα στο χώρο εργασίας με τα ίδια ονόματα με τα αντικείμενα που ορίζονται στο αρχείο `dump.txt`, θα αντικατασταθούν από τα δεύτερα, χωρίς κάποιο μήνυμα.



Εικόνα 4.10: Απεικόνιση του αρχείου `dump.txt` που δημιουργήθηκε με την εντολή `dump`

Αν δεν ορίσουμε διαδρομή αρχείου, τότε το αρχείο θα αποθηκευτεί στο φάκελο εργασίας, ενώ αν δεν ορίσουμε όνομα αρχείου, τότε τα αντικείμενα θα αποθηκευτούν στο αρχείο `dumpdata.R` στο φάκελο εργασίας. Αν θέλουμε να εμφανίσουμε τις εντολές ορισμού στην οθόνη, τότε γράφουμε:

```
> dump(c( 'x', 'x2', 'ex2' ), '')
x <-
structure(list(height = c(1.75, 1.84, 1.81, 1.63), income = c(NA
   , 900L, 1250L, 2300L), sex = structure(c(2L, 2L, 2L, 1L), .
   Label = c("female", "male"), class = "factor")), .Names = c("
   height", "income", "sex" ), row.names = c("Jim", "George", "John
   ", "Mary"), class = "data.frame")
x2 <-
structure(1:10, .Dim = c(5L, 2L))
ex2 <-
structure(list(height = c(1.75, 1.84, 1.81, 1.63), income = c(1500L
   , 900L, 1250L, 2300L), sex = structure(c(2L, 2L, 2L, 1L), .
   Label = c("female", "male"), class = "factor")), .Names = c("
   height", "income", "sex" ), class = "data.frame", row.names = c(
   "Jim", "George", "John", "Mary"))                                    ■.
```

Να σημειώσουμε επίσης, ότι υπάρχει η παράμετρος `control` η οποία λειτουργεί με τον ίδιο τρόπο, όπως στην εντολή `source`, βλ. Ενότητα 4.4.2. Επιπλέον, η παράμετρος `append` ελέγχει αν θέλουμε να αρχίσουμε νέο αρχείο (με την προκαθορισμένη τιμή `FALSE`), σβήνοντας το τυχόν προϋπάρχον ή αν θέλουμε να προσθέσουμε νέα αντικείμενα (με την τιμή `TRUE`) σε ένα αρχείο που ήδη υπάρχει.

Τέλος, μπορούμε να αποθηκεύσουμε όλα τα αρχεία του χώρου εργασίας με την εντολή

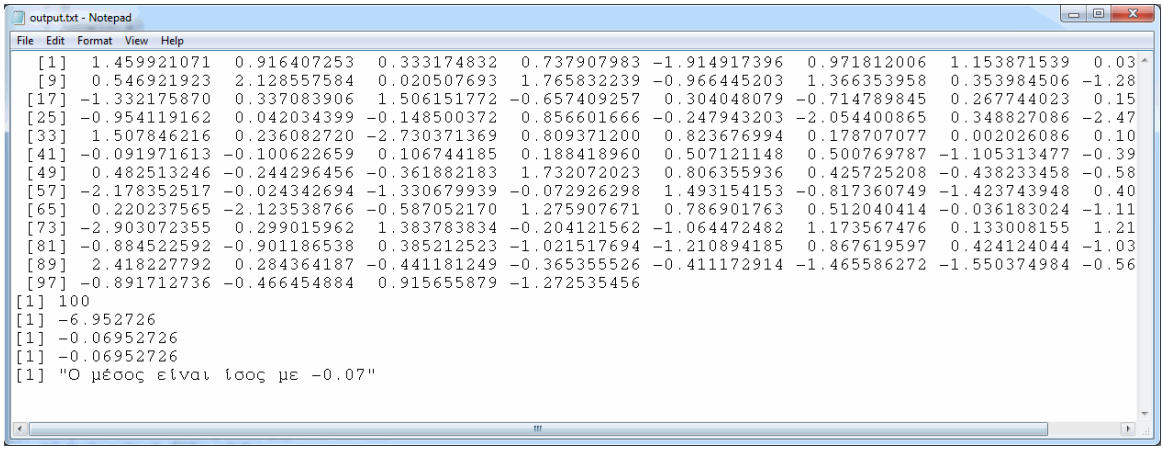
```
dump(ls(all = TRUE), 'c:/rfiles/allobjects.txt')                        ■.
```

4.5 Αποθήκευση των αποτελεσμάτων σε αρχείο

Σε αρκετές περιπτώσεις ενδιαφερόμαστε να αποθηκεύσουμε τα αποτελέσματα της ανάλυσης μας σε ένα αρχείο, έτσι ώστε να το φυλάξουμε για αργότερα, ή να το μελετήσουμε στο τέλος. Αυτό γίνεται εύκολα με την εντολή `sink`, η οποία προωθεί όλα τα αποτελέσματα σε ένα αρχείο (ή σε άλλο μέσο), αντί για την οθόνη. Για παράδειγμα, οι εντολές:

```
> sink('c:/rfiles/output.txt')
> x <- rnorm(100)
> x
> length(x)
> sum(x)
> mean(x)
> sum(x)/length(x)
> print( paste( 'Ο μέσος είναι ίσος με',  round(mean(x),2) ) )
> sink()
>
> mean(x)
[1] -0.06952726
```

θα δημιουργήσουν το αρχείο `output` στον κατάλογο `rfiles` του σκληρού δίσκου `c:`, όπως αυτό φαίνεται στην Εικόνα 4.11. Εάν δεν ορίσουμε το πλήρες μονοπάτι του φακέλου και απλά γράψουμε το όνομα του, τότε αυτό θα αποθηκευτεί στο τρέχων φάκελο εργασίας (υπενθυμίζουμε ότι μπορούμε να δούμε ποιος είναι με την εντολή `getwd`). Όπως βλέπουμε και στον παραπάνω κώδικα, επιστρέφουμε στην επιλογή της εμφάνισης των αποτελεσμάτων στην οθόνη με την εντολή `sink()`, δηλαδή χωρίς να ορίσουμε κάποιο αρχείο. Όπως βλέπουμε από την Εικόνα 4.11, οι εντολές δεν εμφανίζονται στο αρχείο εξόδου. Για το λόγο αυτό είναι σημαντικό να βάζουμε επεξηγήσεις χρησιμοποιώντας την εντολή `print`.



Εικόνα 4.11: Απεικόνιση του αρχείου `output.txt` όπως δημιουργήθηκε με την εντολή `sink`

Αν στην εντολή `sink` βάλουμε το όνομα ενός αρχείου που ήδη υπάρχει, τότε θα σβηστούν τα περιεχόμενα του και θα αρχίσουμε να αποθηκεύουμε τα καινούρια. Πολλές φορές όμως, θέλουμε να συνεχίσουμε την αποθήκευση νέων αποτελεσμάτων, χωρίς να σβήσουμε αυτά που ήδη έχουμε στο αρχείο μας. Αυτό γίνεται με την παράμετρο `append=TRUE`. Τέλος, η παράμετρος `split=TRUE` μας επιτρέπει να βλέπουμε τα αποτελέσματα και στην οθόνη, αλλά και στο αρχείο που έχουμε καθορίσει.

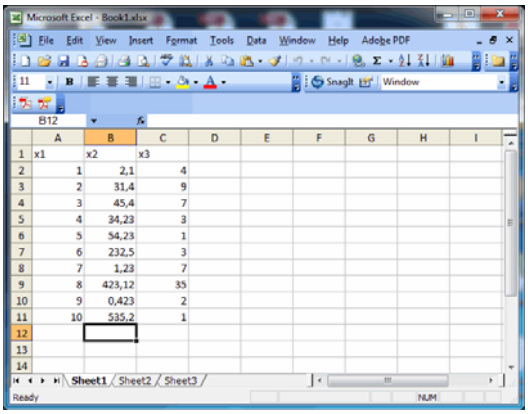
Η εντολή `sink` αποθηκεύει τα αποτελέσματα σε απλά αρχεία κειμένου, χωρίς τη δυνατότητα να ενσωματωθούν σε αυτά διαγράμματα. Στην R υπάρχουν σύγχρονες βιβλιοθήκες που μας επιτρέπουν να εξάγουμε τα αποτελέσματα τα μας σε πολύ καλύτερη μορφή, συμπεριλαμβάνοντας και διαγράμματα. Μερικές από τις βιβλιοθήκες αυτές είναι η R2wd και η R2DOCX για εξαγωγή σε Microsoft Word, η R2PPT για εξαγωγή σε Microsoft Powerpoint, η R2HTML για εξαγωγή σε αρχεία τύπου html (διαβάζονται από internet browsers και το Word), η ReporteRs για εξαγωγή σε όλους του προηγούμενους τύπους, η Sweave και knitr για εξαγωγή σε latex και τέλος, η sjPlot, η οποία μετατρέπει τους πίνακες σε υψηλής ποιότητας πίνακες και τους εξάγει σε html αρχείο. Δε θα ασχοληθούμε παραπάνω με αυτές τις βιβλιοθήκες εδώ, αλλά τις αναφέρουμε για όποιον χρήστες ενδιαφέρεται περαιτέρω.

4.6 Εισαγωγή δεδομένων από άλλα προγράμματα

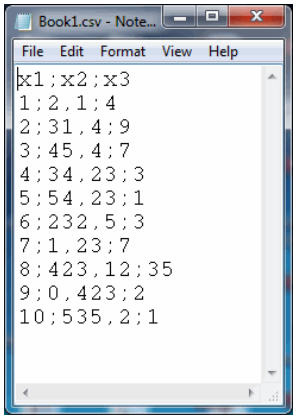
Γενικά στην R υπάρχει η δυνατότητα και ευελιξία να εισάγουμε αρχεία δεδομένων που έχουν δημιουργηθεί με άλλα στατιστικά (και μη) προγράμματα. Εδώ, θα αναφέρουμε τα πιο σημαντικά, όπως μετατροπές από Excel, SPSS, Stata και SAS. Επίσης, πληθώρα εντολών είναι διαθέσιμες από την βιβλιοθήκη foreign.

4.6.1 Εισαγωγή από Excel

Γενικά, υπάρχουν πολλοί τρόποι να διαβάσουμε δεδομένα που έχουμε στο Excel. Οι τρόποι που προτείνουμε είναι μέσω των εντολών `read.csv` και `read.csv2`, αφού όμως, αποθηκεύσουμε το φύλο εργασίας (worksheet) μας ως αρχείο `csv`. Η διαφορά των δύο εντολών είναι πως διαχωρίζονται οι στήλες (με κόμμα και Ελληνικό ερωτηματικό αντίστοιχα) και τον τρόπο ορισμού της υποδιαστολής στους αριθμούς (με τελεία και κόμμα αντίστοιχα).



Εικόνα 4.12: Απεικόνιση του αρχείου Excel με την ονομασία `Book1.xlxs`



Εικόνα 4.13: Απεικόνιση του αρχείου `Book1.csv` όπως αποθηκεύτηκε από το Excel με την επιλογή csv file (MSDOS)

΄Ετσι, το αρχείο της Εικόνας 4.12 αν αποθηκευτεί ως csv θα έχει την Εικόνα 4.13 και θα μπορεί να διαβαστεί από με την εντολή:

```
> read.csv2('c:/rfiles/Book1.csv')
   x1       x2 x3
1   1    2.100  4
2   2   31.400  9
3   3   45.400  7
4   4   34.230  3
5   5   54.230  1
6   6  232.500  3
7   7    1.230  7
8   8  423.120 35
9   9    0.423  2
10 10  535.200  1
```

■.

Ο δεύτερος τρόπος είναι μέσω της εντολής `read.xls` της βιβλιοθήκης `gdata`. Αφού εγκαταστήσουμε τη βιβλιοθήκη `gdata`, μπορούμε να καλέσουμε την εντολή `read.xls` με την ακόλουθη εντολή:

```
> require(gdata)
> myDf <- read.xls ('c:/rfiles/Book1.xlsx', sheet = 1, header =
   TRUE)
Error in findPerl(verbose = verbose) :
  perl executable not found. Use perl= argument to specify the
   correct path.
```

```
Error in file.exists(tfn) : invalid 'file' argument ■.
```

Όπως βλέπετε από το παραπάνω μήνυμα λάθους, η λειτουργία της εντολής αυτής προϋποθέτει την εγκατάσταση του λογισμικού `Perl` και μετά να ορίσουμε και την πλήρη διαδρομή του, μέσω της παραμέτρου `perl`. Έτσι, αφού εγκαταστήσουμε την τελευταία ελεύθερη έκδοση του `Active perl`, γράφουμε:

```
> myDf <- read.xls ('c:/rfiles/Book1.xlsx', sheet = 1, header =
   TRUE, perl='C:/Perl/bin/Perl.exe')
> myDf
   x1      x2 x3
1   1   2.100  4
2   2  31.400  9
3   3  45.400  7
4   4  34.230  3
5   5  54.230  1
6   6 232.500  3
7   7   1.230  7
8   8 423.120 35
9   9   0.423  2
10 10 535.200  1
```

το οποίο οδηγεί στην επιτυχή εκτέλεση της εισαγωγής του αρχείου Excel στην R. Για να τρέξουμε την εντολή χρειάστηκε να ορίσουμε ποιο φύλλο εργασίας θα διαβάσουμε (με την παράμετρο `sheet`) και αν έχουμε επικεφαλίδες με τα ονόματα των μεταβλητών (`header=TRUE`) ή όχι (`header=FALSE`).

4.6.2 Εισαγωγή από SPSS

Η εισαγωγή δεδομένων από το SPSS μπορεί να γίνει με την εντολή `read.spss` της βιβλιοθήκης `foreign`. Έτσι, λοιπόν, μπορούμε να εισάγουμε ένα αρχείο με το όνομα `cities.sav` (βλ. Εικόνα 4.14) από τον κατάλογο `c:\rfiles` με τη σύνταξη:

```
> library(foreign)
> ex5_cities <- read.spss('c:/rfiles/cities.sav', to.data.frame =
   TRUE)
> ex5_cities
              CITY WORK PRICE SALARY
1  Amsterdam\t       1714  65.6   49.0
2  Athens\t\t         1792  53.8   30.4
3  Bogota\t\t         2152  37.9   11.5 ■.
```

Στην παραπάνω σύνταξη, πρώτα καλούμε και «φορτώνουμε» τη βιβλιοθήκη `foreign` (χρειάζεται μόνο την πρώτη φορά κάθε συνεδρίας) και μετά διαβάζουμε με την εντολή `read.spss` το

αρχείο και το εκχωρούμε σε ένα αντικείμενο με το όνομα `ex5_cities`. Επίσης, χρησιμοποιήσαμε την παράμετρο `to.data.frame` με τιμή `TRUE` για να αποθηκευτούν τα δεδομένα μας, ως πλαίσιο δεδομένων, καθώς (για κάποιο περίεργο λόγο) η προκαθορισμένη τιμή είναι `FALSE` και η αποθήκευση σε αυτή την περίπτωση γίνεται σαν λίστα.

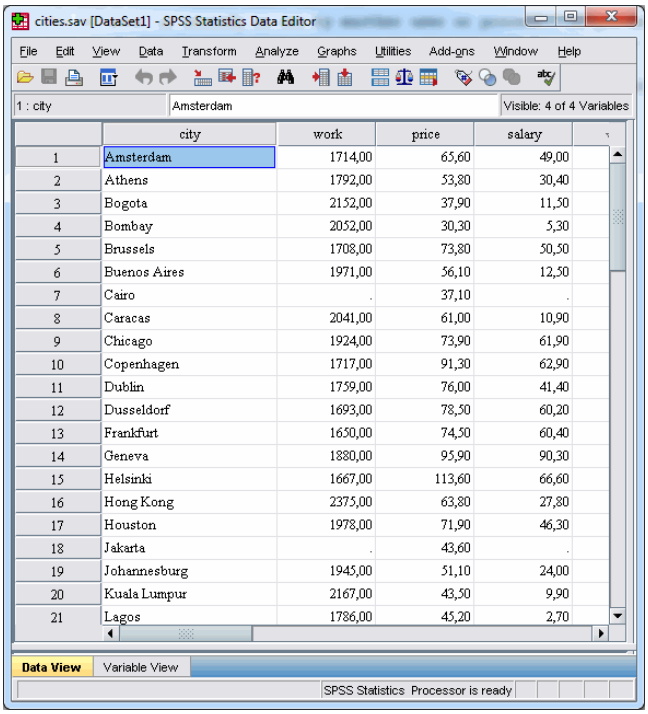
Εναλλακτικά, μπορούμε να χρησιμοποιήσουμε την εντολή `spss.get`, που είναι διαθέσιμη στη βιβλιοθήκη `Hmisc`. Η μόνη διαφορά είναι ότι η εντολή `spss.get` διαβάζει τα δεδομένα ως πλαίσια δεδομένων, χωρίς να πρέπει να αλλάξουμε κάποιες παραμέτρους. Έτσι, η σύνταξη

```
> library(Hmisc)
Loading required package: grid
Loading required package: lattice
Loading required package: survival
Loading required package: splines
Loading required package: Formula

Attaching package: Hmisc

The following objects are masked from package:base:

    format.pval, round.POSIXt, trunc.POSIXt, units
> spss.get(file.choose())
```



Εικόνα 4.14: Απεικόνιση του αρχείου `cities.sav` sto SPSS

```
              CITY WORK PRICE SALARY
1  Amsterdam\t      1714  65.6   49.0
2  Athens\t\t        1792  53.8   30.4
3  Bogota\t\t        2152  37.9   11.5
```

θα διαβάσει το αρχείο που θα του καθορίσουμε στο menu της εντολής `file.choose` και θα εμφανίσει τα περιεχόμενα του στην οθόνη, χωρίς να αποθηκευτεί στο χώρο εργασίας της R, αφού δεν το καταχωρίσαμε σε κάποιο αντικείμενο.

Να σημειώσουμε ότι οι εντολές αυτές διαβάζουν επιτυχημένα μέχρι σήμερα τα αρχεία δεδομένων του SPSS (όπου η τρέχουσα έκδοση του SPSS είναι η 22) με κατάληξη `sav`. Σε περίπτωση που αντιμετωπίσετε προβλήματα ή στο μέλλον η SPSS αλλάξει τον τρόπο αποθήκευσης, τότε προτείνουμε να αποθηκεύετε τα αρχεία σας ως παλαιότερου τύπου SPSS ή ως portable (με κατάληξη por). Εναλλακτικά, μπορείτε να τα αποθηκεύσετε ως αρχεία κειμένου (tab delimited κατά προτίμηση) και μετά να τα διαβάζετε με την εντολή `read.table`. Το πρόβλημα με αυτή την προσέγγιση είναι ότι χάνονται τα μετα-δεδομένα που συνοδεύουν τα αρχεία μας, όπως ο τύπος της μεταβλητής και οι ετικέτες τους. Επίσης, αρκετά αποτελεσματικός τρόπος αποθήκευσης του αρχείου μας στο SPSS είναι να αποθηκευτεί ως αρχείο Stata και μετά να το διαβάσουμε με την εντολή `read.dta` της βιβλιοθήκης `foreign` • βλ. Ενότητα 4.6.3 που ακολουθεί.

4.6.3 Εισαγωγή από άλλα στατιστικά πακέτα

Αν θέλουμε να μεταφέρουμε ένα αρχείο από το SAS στην R, τότε μπορούμε να το αποθηκεύσουμε στο το SAS με τις εντολές

```
# save SAS dataset in trasport format
libname out xport 'c:/rfiles/example6.xpt';
data out.mydata;
set sasuser.mydata;
run;                                                    ■,
```

και μετά να το εισάγουμε μέσω της εντολής `sasxport.get` της βιβλιοθήκης `Hmisc`, χρησιμοποιώντας την ακόλουθη σύνταξη:

```
library(Hmisc)
mydata <- sasxport.get("c:/rfiles/example6.xpt")
```

ή μέσω της εντολή `read.xport` της βιβλιοθήκης `foreign`

```
library(foreign)
mydata <- read.xport("c:/rfiles/example6.xpt")          ■,
```

ενώ δεδομένα πού είναι αποθηκευμένα από το πακέτο Stata, διαβάζονται χρησιμοποιώντας την εντολή `read.dta` (βιβλιοθήκη `foreign`) και την ακόλουθη σύνταξη:

```
library(foreign)
mydata <- read.dta("c:/rfiles/example7.dta")            ■.
```

Γενικά, η βιβλιοθήκη `foreign` μας δίνει μια μεγάλη ποικιλία από εντολές για εισαγωγή δεδομένων, που έχουν αποθηκευτεί από άλλα προγράμματα. Στον Πίνακα 4.2 δίνονται περιληπτικά οι πιο σημαντικές εντολές (συμπεριλαμβάνοντας και αυτές για διάβασμα από αρχεία κειμένου της R).

Τύπος αρχείου	**Κατάληξη**	**Εντολή**	**Βιβλιοθήκη**	**Σύντομη περιγραφή**
Κειμένου	txt, dat, R	`scan`	base	Διαβάζει δεδομένα σε ένα διάνυσμα ή σε μία λίστα από το πληκτρολόγιο ή από έαν αρχείο κειμένου.
		`read.table`	utils	Διαβάζει πινακοποιημένα δεδομένα από ένα αρχείο κειμένου και τα αποθηκεύει ως πλαίσιο δεδομένων.
		`read.csv`	utils	Παρόμοια με τη `read.table` με συγκεκριμένες προεπιλογές (*header = TRUE, sep = ",", dec="."*).
		`read.csv2`	utils	Παρόμοια με τη `read.table` με συγκεκριμένες προεπιλογές (*header = TRUE, sep = ";", dec=","*).
R	Rdata	`load`	base	Φορτώνει δεδομένα (και αντικείμενα) που έχουν αποθηκευτεί με την εντολή `save`.
	R, txt	`dget`	base	Φορτώνει ένα αντικείμενο που έχει αποθηκευτεί με την εντολή `dput`.
	R, txt	`source`	base	Φορτώνει τα αντικείμενα που έχουν αποθηκευτεί με την εντολή `dump`. Επίσης, εκτελεί εντολές της R από ένα αρχείο.
Excel	xls, xlsx	`read.xls`	`gdata`	Διαβάζει αρχεία Microsoft Excel, αφού πρώτα τα μετατρέψει σε csv με την εντολή xls2csv.
		`read.csv` `read.csv2`	utils	Διαβάζει φύλα εργασίας που έχουν αποθηκευτεί ως csv. Η επιλογή της `read.csv` ή `read.csv2` εξαρτάται από τη χώρα και τη κωδικοποίηση που χρησιμοποιείται. Στην Ελλάδα (όπου το κόμμα «,» χρησιμοποιείται ως υποδιαστολή στους πραγματικούς αριθμούς) συνήθως χρησιμοποιείται η δεύτερη εντολή
SPSS	sav	`read.spss`	`foreign`	Διαβάζει αρχεία αποθηκευμένα από το SPSS με την κατάληξη sav και τα αποθηκεύει ως λίστα. Η αποθήκευση σε πλαίσιο δεδομένων γίνεται με την παράμετρο *to.data.frame=TRUE*.
		`spss.get`	`Hmisc`	Διαβάζει αρχεία αποθηκευμένα από το SPSS με την κατάληξη sav και τα αποθηκεύει ως πλαίσιο δεδομένων.
SAS	xpt	`read.xport`	`foreign`	Διαβάζει ένα αρχείο a SAS XPORT και το επιστρέφει ως λίστα ή πλαίσιο αντικειμένου.
		`sasxport.get`	`Hmisc`	Όμοια όπως η `read.xport`.
		`lookup.xport`	`foreign`	Εξαγωγή πληροφοριών του περιεχομένου ενός αρχείου SAS XPORT.
		`read.ssd`	`foreign`	Δημιουργεί κώδικα SAS για τη μετατροπή ενός αρχείου ssd σε SAS XPORT, το οποίο μετά μπορεί να διαβαστεί με την εντολή `read.xport`.
Stata	dta	`read.dta`	`foreign`	Διαβάζει ένα αρχείο Stata (version 5–11) και το αποθηκεύει ως πλαίσιο δεδομένων.
S/Splus	S, txt	`data.restore`	`foreign`	Διαβάζει αρχεία δεδομένων που έχουν αποθηκευτεί με την εντολή `data.dump` στη γλώσσα S version 3.
Βάσεις δεδομένων	dbf	`read.dbf`	`foreign`	Διαβάζει αρχεία βάσεων δεδομένων όπως έχουν αποθηκευτει από προγράμματα όπως dBase, Clipper, FoxPro και Visual Objects.
Minitab	mtp	`read.mtp`	`foreign`	Διαβάζει ένα αρχείο minitab

(Οι βιβλιοθήκες `base` και `utils` είναι διαθέσιμες στη βασική έκδοση της R)

Πίνακας 4.2: Εντολές εισαγωγής δεδομένων από διαφορετικούς τύπους αρχείων

4.7 Εντολές για εμφάνιση στην οθόνη

Η κύρια εντολή για εμφάνιση ενός μηνύματος στην οθόνη είναι η συνάρτηση `print`. Κυρίως χρησιμοποιείται όταν τρέχουμε ένα πρόγραμμα (δηλαδή μια σειρά εντολών), χρησιμοποιώντας την εντολή `source`, ή όταν δουλεύουμε σε δικές μας συναρτήσεις, όπως θα δούμε στο Κεφάλαιο 7. Η εντολή είναι πολύ απλή στη σύνταξη, όπως βλέπουμε στα ακόλουθα παραδείγματα:

```
> print('Ποιο είναι το όνομα σου;')
[1] "Ποιο είναι το όνομα σου;"
>
> print(letters)
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p
   " "q" "r" "s" "t" "u" "v" "w" "x" "y" "z"
>
> print(1:10)
 [1]  1  2  3  4  5  6  7  8  9 10                                  ■.
```

Όπως βλέπουμε από τα παραπάνω παραδείγματα, τα αντικείμενα χαρακτήρων εμφανίζονται με εισαγωγικά. Αν θέλουμε να εμφανίζονται χωρίς αυτά, τότε μπορούμε να χρησιμοποιήσουμε την παράμετρο `quote=FALSE` ή την εντολή `noquote`.

```
>  print('Ποιο είναι το όνομα σου;', quote=F)
[1] Ποιο είναι το όνομα σου;
>  print(letters, quote=F)
 [1] a b c d e f g h i j k l m n o p q r s t u v w x y z
>  noquote('Ποιο είναι το όνομα σου;')
[1] Ποιο είναι το όνομα σου;
>  noquote(letters)
 [1] a b c d e f g h i j k l m n o p q r s t u v w x y z             ■.
```

4.8 Εντολές για αποθήκευση σε αρχείο

Από τις εντολές αποθήκευσης σε αρχείο, ήδη έχουμε περιγράψει τις εντολές `save`, `dput`, `dump` και `sink` • βλ. Ενότητες 4.4.1, 4.4.2, 4.4.3 και 4.5 αντίστοιχα.

Στην ενότητα αυτή θα περιγράψουμε κυρίως την εντολή `write` και τις παραλλαγές της, όπως `write.matrix`, `write.table` και `write.csv`. Όσων αφορά την εντολή `write`, η γενική σύνταξη είναι η ακόλουθη:

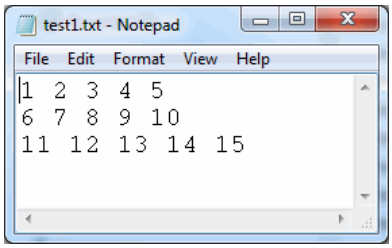
```
write(x, file = "data",
      ncolumns = if(is.character(x)) 1 else 5,
      append = FALSE, sep = " ")                                    ■,
```

όπου `x` είναι το αντικείμενο που θέλουμε να αποθηκεύσουμε (συνήθως είναι διάνυσμα ή πίνακας), `file` είναι το όνομα του αρχείου που θα γίνει η εγγραφή, `ncolumns` είναι ο αριθμός των

στηλών με προκαθορισμένη τιμή τη στήλη, αν είναι το x είναι αντικείμενο χαρακτήρων, ή πέντε στήλες διαφορετικά. Τέλος, η παράμετρος append είναι ίδια με την αντίστοιχη στο sink και ελέγχει αν το αρχείο θα ξεκινήσει από την αρχή (με την προκαθορισμένη τιμή FALSE), σβήνοντας τα προηγούμενα περιεχόμενα του αρχείου, ή αν θα συνεχίσει προσθέτοντας τα νέα στοιχεία, χωρίς να σβήσει τα προηγούμενα περιεχόμενα του αρχείου (με την τιμή TRUE). Για παράδειγμα, με τη σύνταξη :

```
> x<-matrix(1:15,nrow=5)
> x
     [,1] [,2] [,3]
[1,]    1    6   11
[2,]    2    7   12
[3,]    3    8   13
[4,]    4    9   14
[5,]    5   10   15
> write(x, 'c:/rfiles/test1.txt')
```

γίνεται η εγγραφή των στοιχείων του πίνακα x στο αρχείο test1.txt στον κατάλογο c:/rfiles με την μορφή που φαίνεται στην Εικόνα 4.15.



Εικόνα 4.15: Αρχείο test1.txt που έχει δημιουργηθεί με την εντολή write χωρίς μορφοποίηση

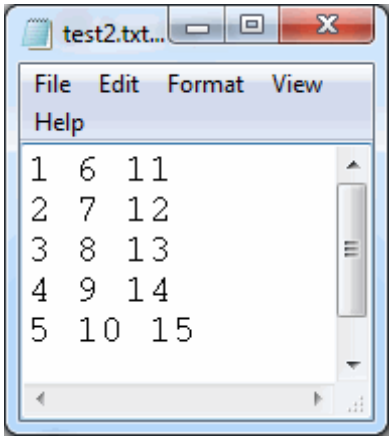
Αν θέλουμε τη σωστή μορφή, θα πρέπει να εισάγουμε τον ανάστροφο πίνακα και να ορίσουμε ότι θέλουμε τόσες στήλες, όσες οι στήλες του πίνακα. Έτσι, η σύνταξη:

```
> write(t(x), 'c:/rfiles/test2.txt', ncol=3)
```

θα μας αποθηκεύσει τον πίνακα x σωστά στο αρχείο test2.txt στον κατάλογο c:/rfiles, όπως φαίνεται στην Εικόνα 4.16.

Να σημειώσουμε ότι αν δε δηλώσουμε όνομα αρχείου στην παράμετρο file, τότε τα περιεχόμενα του αντικειμένου μας αποθηκεύονται σε ένα αρχείο με το όνομα data στον τρέχοντα κατάλογο εργασίας, ενώ αν γράψουμε file='', τότε απλά θα γίνει εμφάνιση του αντικειμένου στην οθόνη (μορφοποιημένου σύμφωνα με τους κανόνες της εντολής write), όπως βλέπουμε και στο ακόλουθο κομμάτι κώδικα:

```
> write(x)
> write(x, file='')
1 2 3 4 5
```



Εικόνα 4.16: Αρχείο test2.txt που έχει δημιουργηθεί με την εντολή `write` με την κατάλληλη μορφοποίηση

```
6 7 8 9 10
11 12 13 14 15                                                    ■.
```

Γενικά, αν θέλουμε να αποθηκεύσουμε πίνακες αντί για διανύσματα, θα πρέπει να προτιμούμε την εντολή `write.matrix` (του πακέτου MASS) και την εντολή `write.table`, αν θέλουμε να αποθηκεύσουμε πλαίσια δεδομένων. Η εντολή είναι αντίστοιχη της εντολής `read.table`. Υπάρχουν επίσης και οι εντολές `write.csv` και `write.csv2` που είναι αντίστοιχες των `read.csv` και `read.csv2`. Γράφουν ένα πλαίσιο δεδομένων με συγκεκριμένη μορφοποίηση, η οποία είναι και άμεσα προσπελάσιμη και από το Microsoft Excel. Τέλος, σε όλες αυτές εντολές αν δεν ορίσουμε αρχείο, τότε το αντικείμενο μας τυπώνεται στην οθόνη με την μορφοποίηση που επιβάλλει η αντιστοιχεί εντολή, όπως βλέπουμε στα ακόλουθα παραδείγματα:

```
> write.matrix(x)
 1  6 11
 2  7 12
 3  8 13
 4  9 14
 5 10 15
> write.table(x)
"V1" "V2" "V3"
"1" 1 6 11
"2" 2 7 12
"3" 3 8 13
"4" 4 9 14
"5" 5 10 15
> write.csv(x)
"","V1","V2","V3"
"1",1,6,11
```

```
"2",2,7,12
"3",3,8,13
"4",4,9,14
"5",5,10,15
> write.csv2(x)
"";"V1";"V2";"V3"
"1";1;6;11
"2";2;7;12
"3";3;8;13
"4";4;9;14
"5";5;10;15
```
■.

Οι περισσότερες εντολές εισαγωγής δεδομένων έχουν αντίστοιχες εντολές αποθήκευσης, όπως οι εντολές `write.dta` (για Stata) και `write.dbf` (για αρχεία βάσεων δεδομένων). Επίσης, υπάρχει και η εντολή `write.foreign` (για Stata, SPSS και SAS). Και οι τρεις αυτές εντολές είναι διαθέσιμες μέσα από το πακέτο foreign. Όλες οι εντολές εγγραφής (με τις αντίστοιχες εντολές εισαγωγής/διαβάσματος), δίνονται στον Πίνακα 4.3.

Τύπος αρχείου	**Εντολή εγγραφής**	**Εντολή διαβάσματος**	**Βιβλιοθήκη**	**Σύντομη περιγραφή**
Κειμένου	`write`	`scan`	base	Αποθηκεύει δεδομένα σε ένα διάνυσμα σειριακά.
	`read.matrix`	`scan`	MASS	Αποθηκεύει τα δεδομένα ενός πίνακα με την κατάλληλη μορφοποίηση.
	`write.table`	`read.table`	utils	Αποθηκεύει τα δεδομένα ενός πλαισίου δεδομένων.
	`write.csv`	`read.csv`	utils	Παρόμοια με τη `write.table` με συγκεκριμένες προεπιλογές (*sep* = ",", *dec*=".").
	`write.csv2`	`read.csv2`	utils	Παρόμοια με τη `read.table` με συγκεκριμένες προεπιλογές (*sep* = ";", *dec*=".").
R	`save`	`load`	base	Αποθηκεύει δεδομένα (και αντικείμενα) με την μορφή `.Rdata`.
	`save.image`	`load`	base	Αποθηκεύει όλα τα δεδομένα του χώρου εργασίας (και αντικείμενα) με την μορφή `.Rdata`.
	`dput`	`dget`	base	Αποθηκεύει ένα αντικείμενο ως εντολή ορισμού (χωρίς το όνομά του).
	`dump`	`source`	base	Αποθηκεύει πολλά αντικείμενα ως εντολές ορισμού (με τα ονόματά τους).

Excel	`write.csv`
	
	`write.csv2`
		`read.csv`
		
		`read.csv2`
			utils	Αποθηκεύει ένα πλαίσιο δεδομένων σε μορφή csv που εισάγεται άμεσα στο Excel.
				Η επιλογή μεταξύ των `write.csv` ή `write.csv2` εξαρτάται από τη χώρα και τη κωδικοποίηση που χρησιμοποιείται. Στην Ελλάδα (όπου το κόμμα «,» χρησιμοποιείται ως υποδιαστολή στους πραγματικούς αριθμούς) συνήθως, χρησιμοποιείται η δεύτερη εντολή

SPSS	`write.foreign( ... package = "SPSS")`	`read.spss`	`foreign`	Αποθηκεύει δεδομένα σε ένα αρχείο με τη μορφή SPSS (κατάληξη sav).
SAS	`write.foreign( ... package = "SAS")`	`read.xport`	`foreign`	Αποθηκεύει δεδομένα σε ένα αρχείο SAS.
Stata	`write.foreign( ... package = "Stata")`	`read.dta`	`foreign`	Αποθηκεύει δεδομένα σε ένα αρχείο Stata.
	`write.dta`	`read.dta`	`foreign`	Αποθηκεύει δεδομένα σε ένα αρχείο Stata. Η παράμετρος `version` ελέγχει την έκδοση Stata. Συγκεκριμένα, μπορούμε να δώσουμε τις τιμές 6, 7 (προκαθορισμένη), 8 (για τις εκδόσεις 8 και 9) και 10 (για τις εκδόσεις 10 και 11).

(Οι βιβλιοθήκες `base` και `utils` είναι άμεσα διαθέσιμες στη βασική έκδοση της R. Η βιβλιοθήκη `MASS` είναι επίσης εγκατεστημένη στη βασική έκδοση της R, αλλά χρειάζεται να «φορτωθεί» πριν τη χρησιμοποιήσουμε για πρώτη φορά.)

Πίνακας 4.3: Εντολές αποθήκευσης δεδομένων σε διαφορετικούς τύπους αρχείων

ΚΕΦΑΛΑΙΟ 5

Απεικόνιση δεδομένων

΄Ενα από τα ισχυρά σημεία της R είναι τα πολύ καλής ποιότητας διαγράμματα που μπορεί να δημιουργήσει. Πληθώρα διαφορετικών τύπων διαγραμμάτων είναι διαθέσιμα από τα μενού του προγράμματος. Εντούτοις, η γραμμή εντολών δεν χάνει τη χρησιμότητά της, διότι τα διαγράμματα παραμετροποιούνται σε πολύ μεγάλο βαθμό και η πλήρης ποικιλία εντολών παραμετροποίησης είναι διαθέσιμη μόνο από τη γραμμή εντολών.

Σε αυτές τις σημειώσεις, θα ασχοληθούμε με ιστογράμματα, διαγράμματα πλαισίου-απολήξεων ή θηκογράμματα (boxplots) και διαγράμματα διασποράς.

5.1 Ιστόγραμμα

Η εντολή δημιουργίας ιστογράμματος είναι η `hist()`, με απαραίτητη παράμετρο ένα διάνυσμα με τις τιμές που θα αναπαρασταθούν στο ιστόγραμμα. Αν το διάνυσμα περιέχει n τιμές, ο αριθμός των ράβδων του ιστογράμματος είναι ίσος με το ακέραιο μέρος του λογαρίθμου του n συν 1.

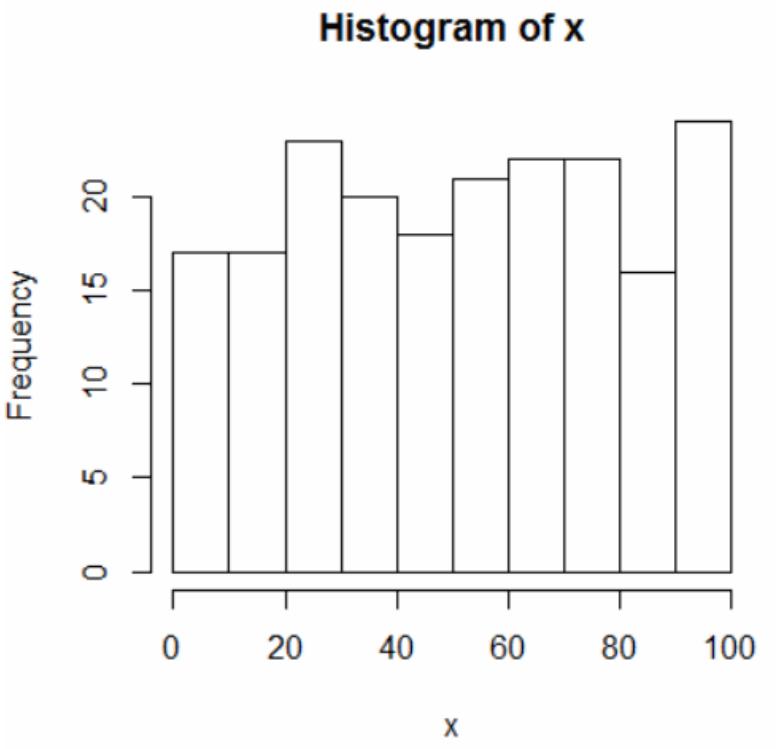
```
> x<-sample(0:100,size=200,replace=T)
> hist(x)                                                    ■.
```

Με την εντολή `sample()` πήραμε δείγμα τιμών από το διάνυσμα `1:100`, μεγέθους 200 (δηλαδή `size=200`) με επανατοποθέτηση (`replace=T`). Το αποτέλεσμα των παραπάνω εντολών δίνεται στο Διάγραμμα 5.1.

Αν θέλουμε να ορίσουμε εμείς τον αριθμό κλάσεων, τότε πρέπει να δώσουμε στην `hist()` την παράμετρο `nclass` με τον αριθμό των κλάσεων που επιθυμούμε. Δυστυχώς, η R δεν θα τηρήσει πάντα την επιλογή μας. Δώστε την εντολή `hist(x,nclass=20)` για να γίνει ιστόγραμμα του ίδιου με πριν διανύσματος με 20 κλάσεις.

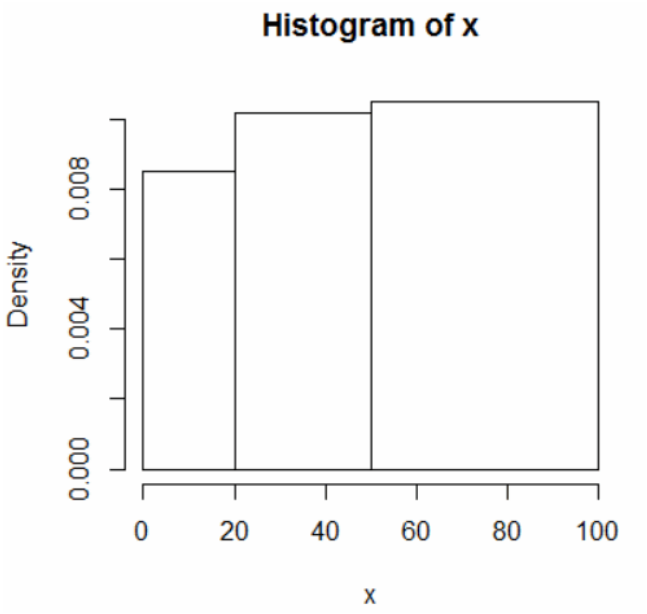
Μία δεύτερη χρήσιμη παράμετρος είναι η `breaks`, η οποία μας επιτρέπει να ορίσουμε εμείς τα όρια των κλάσεων. Για παράδειγμα, για να κάνουμε ιστόγραμμα της `x` παραπάνω, αλλά με τρεις κλάσεις, τις `[0,20]`, `(21,50]`, `(51,100]`, γράφουμε

```
hist(x,breaks=c(0,20,50,100))
```



Διάγραμμα 5.1: Ιστόγραμμα 200 προσομοιωμένων τιμών από την διακριτή ομοιόμορφη κατανομή με τιμές 0-100.

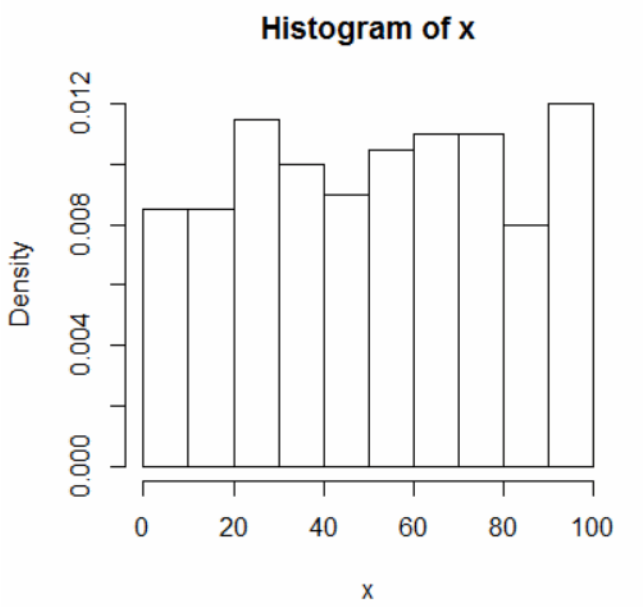
με αποτέλεσμα το Διάγραμμα 5.2.



Διάγραμμα 5.2: Ιστόγραμμα 200 προσομοιωμένων τιμών από την διακριτή ομοιόμορφη κατανομή με τιμές 0-100 με κλάσεις διαφορετικού εύρους

Προσοχή πρέπει να δοθεί, στο ότι όλες οι κλάσεις είναι ανοικτές αριστερά και κλειστές δεξιά, εκτός από την πρώτη που είναι κλειστή και στα δύο άκρα.

Με χρήση της παραμέτρου `probability=T`, το ύψος κάθε ράβδου είναι τέτοιο, ώστε αν πολλαπλασιαστεί με το εύρος της κάθε κλάσης (εύρος = δεξί άκρο - αριστερό άκρο) και τα γινόμενα αθροιστούν για όλες τις κλάσεις, το σύνολο είναι 1. Το ιστόγραμμα που προκύπτει είναι δηλαδή μία «εκτίμηση» της συνάρτησης πιθανότητας ή πυκνότητας πιθανότητας του πληθυσμού από τον οποίο προέρχεται το δείγμα. Έτσι, η εντολή `hist(x,probability=T)` μας δίνει το Διάγραμμα 5.3.



Διάγραμμα 5.3: Ιστόγραμμα 200 προσομοιωμένων τιμών από την διακριτή ομοιόμορφη κατανομή με τιμές 0-100 με μοναδιαίο εμβαδόν (`probability=TRUE`)

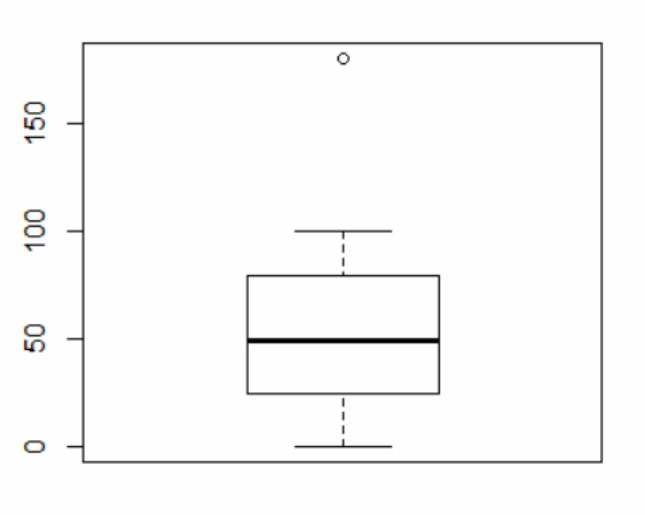
Χρησιμοποιώντας την παράμετρο `plot=F`, το ιστόγραμμα δεν σχεδιάζεται, αλλά παράγει μία λίστα με στοιχεία δύο διανύσματα. Το πρώτο δίνει τα όρια των κλάσεων και το δεύτερο το ύψος κάθε ράβδου του ιστογράμματος.

```
> hist(x,plot=F)
$breaks:
[1]  0  10  20  30  40  50  60  70  80  90 100
$counts:
[1] 19 16 17 21 19 23 24 14 24 23                    ■.
```

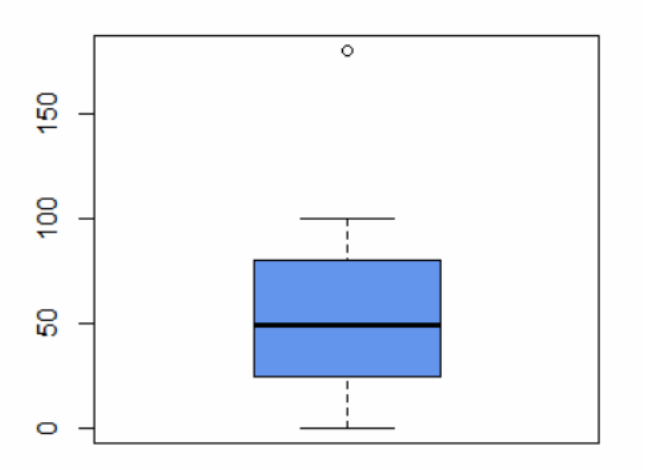
Το χρώμα των ράβδων του ιστογράμματος καθορίζεται με χρήση της παραμέτρου `col=`. Για μαύρες ράβδους δίνουμε `col=1`, για άσπρες `col=0`. Στα χρώματα θα αναφερθούμε στη συνέχεια.

5.2 Διάγραμμα πλαισίου-απολήξεων ή θηκογράμμα (Boxplot)

Η εντολή `boxplot` δημιουργεί ένα διάγραμμα πλαισίου-απολήξεων ή θηκόγραμμα. Η εντολή έχει βασικό όρισμα ένα διάνυσμα με τις τιμές που θα αναπαρασταθούν στο boxplot. Αν αντικαταστήσουμε την πρώτη γραμμή του `x` (βλ. Ενότητα 5.1)με την τιμή 180 για να εισαχθεί μία ακραία τιμή στο δείγμα και δώσουμε την εντολή `Boxplot(x)`, το αποτέλεσμα απεικονίζεται στο διάγραμμα 5.4. Επιπλέον, μπορούμε να αλλάξουμε το χρώμα του πλαισίου ορίζοντας την παράμετρο `col`. Για παράδειγμα, η σύνταξη `boxplot(x, col=5)` ή `boxplot(x, col="cyan")` θα μας δώσει το Διάγραμμα 5.5.



Διάγραμμα 5.4: Διάγραμμα πλαισίου απολήξεων για ένα δείγμα 200 προσομοιωμένων τιμών

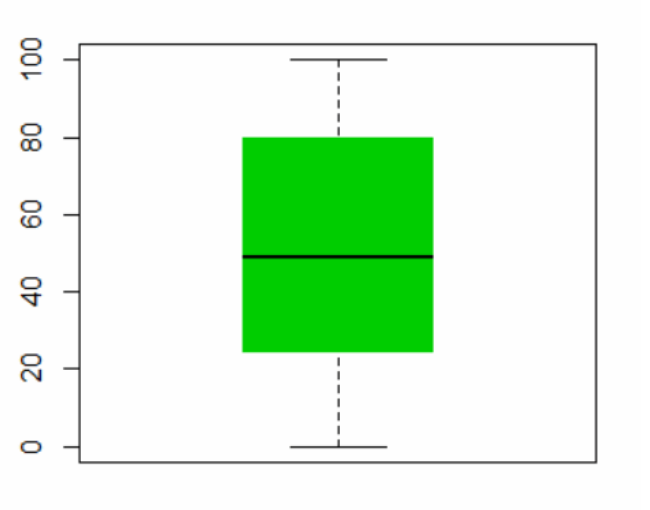


Διάγραμμα 5.5: Διάγραμμα πλαισίου απολήξεων για ένα δείγμα 200 προσομοιωμένων τιμών με γαλάζιο χρώμα πλαισίου

Η κεντρική γραμμή δείχνει τη διάμεσο και τα όρια του κουτιού τοποθετούνται στα δύο τεταρτημόρια. Οι εξωτερικές γραμμές τοποθετούνται στη θέση των παρατηρήσεων που βρί-

σκονται επάνω και κάτω από το αντίστοιχο τεταρτημόριο σε απόσταση 1.5 φορά το ενδοτεταρτημοριακό εύρος του δείγματος. Αν δεν υπάρχουν παρατηρήσεις τόσο μακριά, οι γραμμές τοποθετούνται σε παρατηρήσεις που είναι πιο κοντά στο τεταρτημόριο, αλλά και όσο το δυνατό πιο κοντινές σε αυτήν την απόσταση. Τέλος, ακόμα πιο ακραίες παρατηρήσεις παρουσιάζονται με κουκκίδες, οι οποίες δεν συνδέονται με το κουτί.

Οι πολύ ακραίες αυτές παρατηρήσεις εμφανίζονται με σημείο, αντί για γραμμή, αν δοθούν οι παράμετροι `outline=F` και `outchar=T`. Η πρώτη ορίζει να μην εμφανίζονται με γραμμή και η δεύτερη να εμφανίζονται με σημείο. Το είδος του σημείου που θα εμφανίζεται, το ορίζει η παράμετρος `outpch=αριθμός`. Για παράδειγμα, `outpch=1` ορίζει λευκό σημείο με μαύρο περίγραμμα (άδειος κύκλος), ενώ `outpch=16` ορίζει μαύρο συμπαγή κύκλο. Με `boxcol=1` ζητάμε το κουτί του boxplot να έχει μαύρο χρώμα, ενώ με `boxcol=0` ζητάμε να έχει λευκό χρώμα. Προσέξτε ότι οι αριθμοί των χρωμάτων είναι ίδιοι με αυτούς που χρησιμοποιήσαμε στην παράμετρο `col` της εντολής `hist()`. Όταν το boxplot γίνει με λευκό χρώμα, η διάμεσος δεν θα φαίνεται. Για να γίνει η διάμεσος με μαύρο χρώμα, δίνουμε την παράμετρο `medcol=1`. Αν, τέλος, η γραμμή της διαμέσου μας φαίνεται παχιά δίνουμε `medlwd=2`. Η R χρησιμοποιεί εξ ορισμού `medlwd=5`. Έτσι, η εντολή `boxplot(x,outchar=T, outline=F, boxcol=0, medcol=1, medlwd=2, col=3)`
δίνει το Διάγραμμα 5.2 με άσπρο χρώμα για το περίγραμμα (`boxcol`), με μαύρο χρώμα γραμμή διαμέσου (`medcol`) και με διπλάσιο πάχος (`medlwd=2`), και με κόκκινο χρώμα για το περίγραμμα (`boxcol`).



Διάγραμμα 5.6: Διάγραμμα πλαισίου απολήξεων χωρίς το περίγραμμα του πλαισίου για ένα δείγμα 200 προσομοιωμένων τιμών

Αν δώσουμε την παράμετρο `range=0`, τότε τα δύο όρια πάνω και κάτω από το κουτί απλώνονται ως τη μικρότερη και μεγαλύτερη τιμή του δείγματος και καμία τιμή δεν εμφανίζεται ως ακραία.

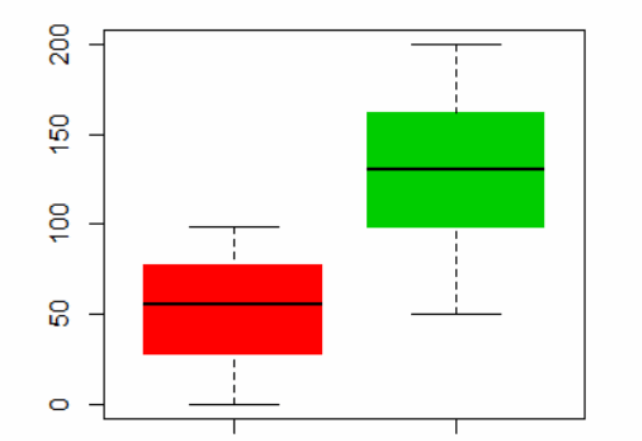
Τα boxplot είναι χρήσιμα για να συγκρίνουμε το εύρος των διαφόρων δειγμάτων καθώς και των

παρατηρήσεων από τα διάφορα δείγματα. Αν δώσουμε δύο ή περισσότερα διανύσματα ως παραμέτρους της εντολής `boxplot()`, θα δημιουργηθούν τα boxplot τους δίπλα - δίπλα. Παράμετροι χρωμάτων, μορφής σημείων, κλπ, επηρεάζουν όλα τα boxplot το ίδιο. Για παράδειγμα, δείτε το αποτέλεσμα των εντολών

```
> y<-sample(50:200,size=200,replace=T)}
> boxplot(x,y,outchar=T,outline=F,boxcol=0,medcol=1,medlwd=2, col
   =2:3)
```

που δίνει ως αποτέλεσμα το Διάγραμμα 5.7. Τα χρώματα των πλαισίων έχουν καθοριστεί από την παράμετρο `col=2:3`, που σημαίνει ότι το πρώτο πλαίσιο θα έχει χρώμα κόκκινο (κωδικός 2) και το δεύτερο χρώμα πράσινο (κωδικός 3).Τα χρώματα εκτός από τους κωδικούς τους μπορούμε να τα δηλώσουμε με τα ονόματά τους. Πιο συγκεκριμένα για το κόκκινο χρώμα αντί να γράψουμε `col=2` το αντικαθιστούμε με `col="red"`. Αντίστοιχα, το πράσινο αντί να το δηλώσουμε με τον κωδικό του `col=3`, το αντικαθιστούμε με `col="green"`. Αν έχουμε περισσότερα από ένα αντικείμενα που θέλουμε να χρωματίσουμε, τότε βάζουμε τα ονόματά τους σε ένα διάνυσμα, όπως φαίνεται παρακάτω:

```
> boxplot(x,y,outchar=T,outline=F,boxcol=0,medcol=1, medlwd=2,
  col=c("red","green"))                                          ■.
```



Διάγραμμα 5.7: Διάγραμμα πλαισίου απολήξεων χωρίς το περίγραμμα του πλαισίου για δύο δείγματα προσομοιωμένων τιμών

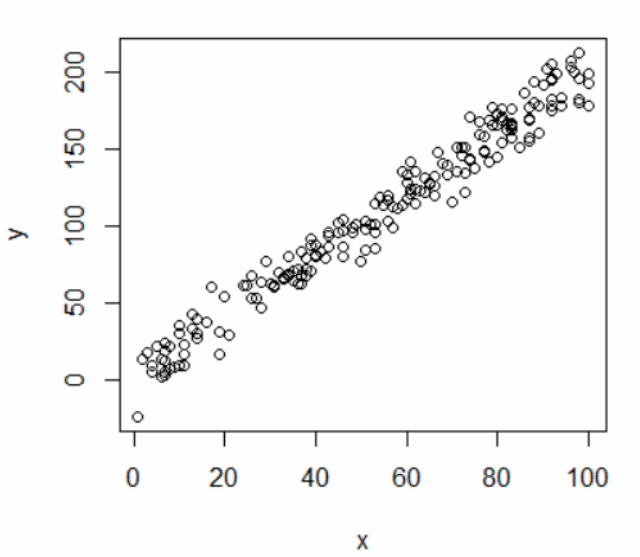
Αν θέλουμε κάτω από κάθε boxplot να εμφανιστεί ένα επεξηγηματικό όνομα, π.χ. X και Y αντίστοιχα, δίνουμε την παράμετρο `names=c("X","Y")`.

5.3 Διάγραμμα διασποράς

Η εντολή δημιουργίας διαγράμματος διασποράς (scatter plot) είναι `plot()`, με απαραίτητες παραμέτρους δύο διανύσματα με τις τιμές που θα αναπαρασταθούν στο διάγραμμα. Το πρώτο δίνει τις συντεταγμένες ως προς τον οριζόντιο άξονα και το δεύτερο τις συντεταγμένες ως προς

τον κατακόρυφο. Στο παράδειγμα που ακολουθεί, δημιουργούμε ένα διάνυσμα x με 200 τυχαίους αριθμούς από τις ακέραιες τιμές από το 0 έως το 100. Στη συνέχεια δημιουργούμε διάνυσμα y, το οποίο είναι ίσο με δύο φορές το διάνυσμα x συν μία τυχαία ποσότητα που ακολουθεί κανονική κατανομή με μέσο μηδέν και τυπική απόκλιση 10 (χρησιμοποιούμε την εντολή `rnorm` για να παράγουμε 200 τυχαίους αριθμούς από αυτή την κατανομή). Κατόπιν, κάνουμε διάγραμμα διασποράς των `x` και `y` • βλ. Διάγραμμα 5.8.

```
x<-sample(0:100,size=200,replace=T)
e<-rnorm(200,0,10)
y<-2*x+e
plot(x,y)                                                    ■.
```



Διάγραμμα 5.8: Διάγραμμα διασποράς (scatter plot)

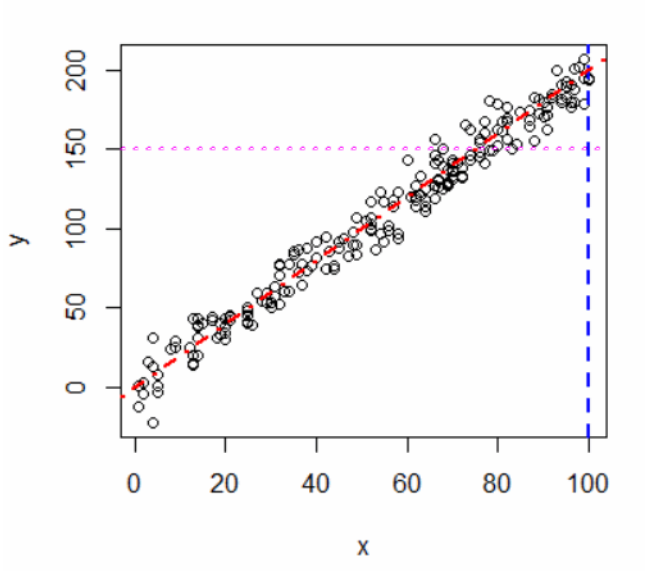
Η R τοποθετεί τα ονόματα των διανυσμάτων στους άξονες. Επίσης, αν και δεν φαίνεται λόγω του μικρού μεγέθους του διαγράμματος στη σελίδα, κάθε σημείο αναπαρίσταται με έναν άδειο κύκλο. Αν θέλουμε συμπαγείς μαύρους κύκλους πρέπει να προσθέσουμε την παράμετρο `pch=16`.

Μία σημαντική εντολή είναι η `abline()`. Αυτή προσθέτει σε ένα ανοικτό διάγραμμα διασποράς μία ευθεία με παραμέτρους που ορίζει ο χρήστης. Με διαδοχικές κλήσεις της εντολής μπορούν να προστεθούν περισσότερες της μίας γραμμής στο ίδιο διάγραμμα. Δείχνουμε τη χρήση της με παραδείγματα. Οι εντολές

```
> abline(v=100, col="blue", lty=2, lwd=2)
> abline(h=150, col="magenta", lty=3, lwd=2)
> abline(0,2, col="red", lty=4, lwd=2)}
```

δημιουργούν η πρώτη μία κάθετη γραμμή στη συντεταγμένη `x=100` (χρώμα κόκκινο, δεύτερου τύπου γραμμή), η δεύτερη μία οριζόντια γραμμή στη συντεταγμένη `y=150` (χρώμα πράσινο και

τρίτου τύπου γραμμή) και η τρίτη μία ευθεία με εξίσωση `y=0+2*x` (χρώμα μπλε και τέταρτου τύπου γραμμή), ενώ όλες οι γραμμές έχουν διπλάσιο πάχος από το default (`lwd=2`), (βλ. Διάγραμμα 5.9).



Διάγραμμα 5.9: Διάγραμμα διασποράς με γραμμές αναφοράς μέσω της εντολής `abline`

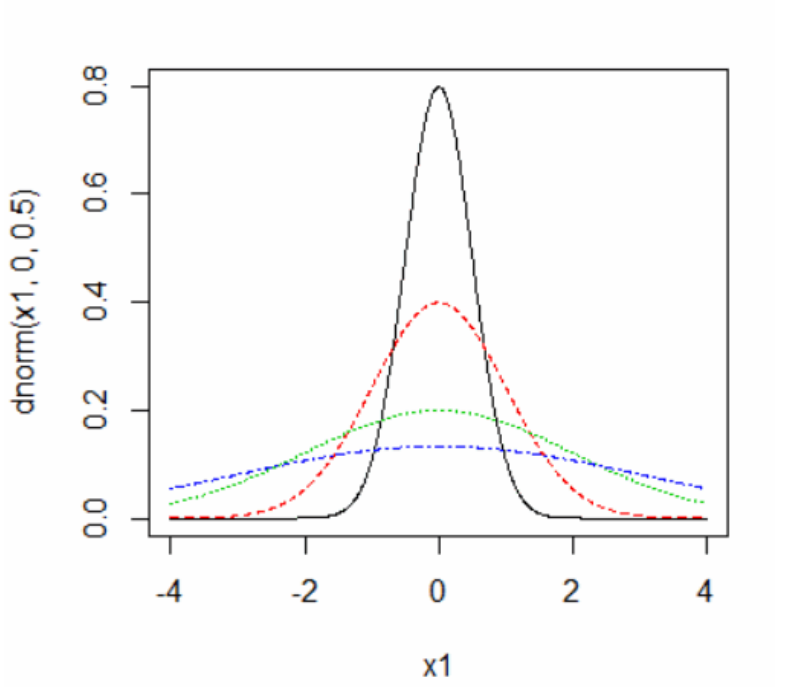
Μία χρήσιμη παραλλαγή της εντολής `plot()` είναι η παράμετρος `type="l"` αφενός και η εντολή `lines()` αφετέρου.Με τη διαφορά ότι η πρώτη δημιουργεί διάγραμμα διασποράς, αλλά ενώνει κάθε σημείο με το επόμενο με μία γραμμή (το επόμενο είναι αυτό που το ακολουθεί στο διάνυσμα που δίνει τις συντεταγμένες), ενώ η δεύτερη δημιουργεί επάνω σε ήδη ανοικτό διάγραμμα ένα νέο διάγραμμα διασποράς με τα σημεία ενωμένα με γραμμές.

Γενικά, οι δυνατές τιμές που μπορεί να πάρει το όρισμα type είναι οι εξής:

- `"p"` τυπώνονται σύμβολα στα σημεία,
- `"l"` συνδέονται τα σημεία με ευθύγραμμα τμήματα,
- `"b"` σημεία και γραμμές όχι ενωμένα,
- `"c"` το τμήμα γραμμών μόνο,όταν έχω την επιλογή των δυο "b",
- `"o"` (overstruck) σημεία και γραμμές ενωμένα,
- `"h"` (high-density) κατακόρυφες γραμμές μέχρι τον οριζόντιο άξονα,
- `"s"` για βήματα με τη μορφή σκαλοπατιών,
- `"S"` για άλλα είδη βήματος,
- `"n"` δεν τυπώνει διάγραμμα.

Οι εντολές είναι χρήσιμες για την απεικόνιση γραφικών παραστάσεων συναρτήσεων. Με τις παρακάτω εντολές απεικονίζουμε τις συναρτήσεις (μάζας) πυκνότητας πιθανότητας τεσσάρων Κανονικών κατανομών, των N(0,0.25), N(0,1), N(0,2) και N(0,3). Με χρήση της παραμέτρου `lty` αλλάζουμε και το είδος της κάθε γραμμής. Η τυπική γραμμή αντιστοιχεί σε `lty=1`.

```
> x1<-seq(from=-4,to=4,length=1000)
> plot(x1,dnorm(x1,0,0.5),type="l",lty=1,col=1)
> lines(x1,dnorm(x1,0,1), lty=2,col=2, lwd=2)
> lines(x1,dnorm(x1,0,2), lty=3,col=3, lwd=2)
> lines(x1,dnorm(x1,0,3), lty=4,col=4, lwd=2)                    ■.
```



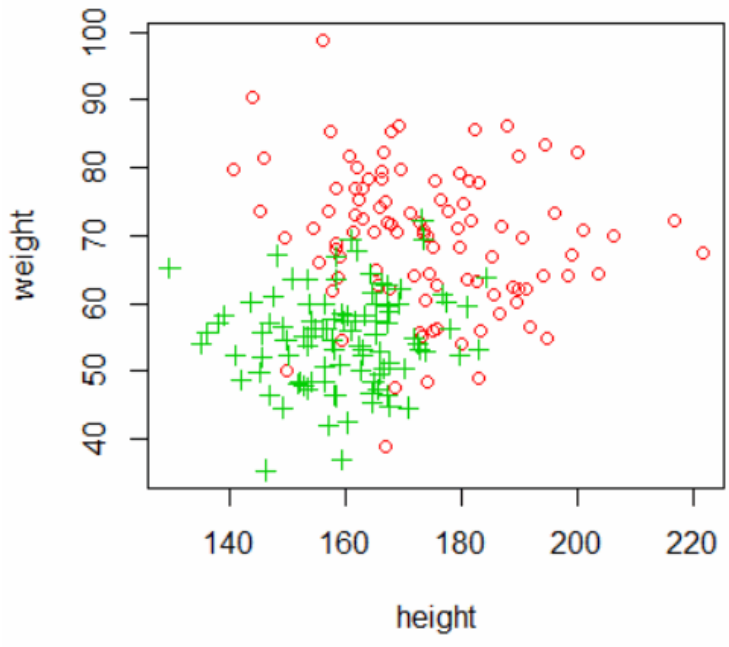
Διάγραμμα 5.10: Συναρτήσεις (μάζας) πυκνότητας πιθανότητας τεσσάρων Κανονικών κατανομών, των N(0,0.25), N(0,1), N(0,2) και N(0,3)

Η λογική δημιουργίας γραφικών παραστάσεων είναι πάντα η ίδια: δημιουργούμε μια ακολουθία τιμών `x` με εύρος που καλύπτει το εύρος τιμών που μας ενδιαφέρει και με τις τιμές της πολύ κοντά η μία στην άλλη. Για κάθε μέλος της `x` υπολογίζουμε την τιμή `y` της συνάρτησης και αποθηκεύουμε τις τιμές σε ένα δεύτερο διάνυσμα. Έπειτα, κάνουμε διάγραμμα διασποράς επιλέγοντας `type="l"` και η γραφική παράσταση είναι έτοιμη.

Αντίστοιχη της εντολής `lines()` είναι η εντολή `points()`, η οποία προσθέτει ένα κλασικό διάγραμμα διασποράς σε ήδη ανοικτό διάγραμμα. Στο επόμενο παράδειγμα προσομοιώνουμε 200 παρατηρήσεις ύψους και βάρους για 100 άντρες και 100 γυναίκες. Κατόπιν, δημιουργούμε ένα άδειο διάγραμμα διασποράς (με χρήση της παραμέτρου `type="n"`, η οποία ετοιμάζει τους άξονες), στο οποίο προσθέτουμε διάγραμμα διασποράς ύψους και βάρους (οριζόντιος και

κατακόρυφος άξονας αντίστοιχα) με άλλο σύμβολο για τους άντρες (συμπαγής κύκλος: `pch=1`) και άλλο για τις γυναίκες (σταυρός: `pch=3`). Η προσθήκη γίνεται με χρήση της `points()`. Έτσι, λοιπόν, οι ακόλουθες εντολές μας δίνουν το Διάγραμμα 5.3.

```
> weight<-c(rnorm(100,mean=70,12),rnorm(100,mean=55,7))
> height<-c(rnorm(100,mean=175,14),rnorm(100,mean=158,10))
> sex<-rep(c("M","F"),each=100)
> plot(height,weight,type="n")
> points(height[sex=="M"],weight[sex=="M"],pch=1, col=2)
> points(height[sex=="F"],weight[sex=="F"],pch=3, col=3)          ■.
```

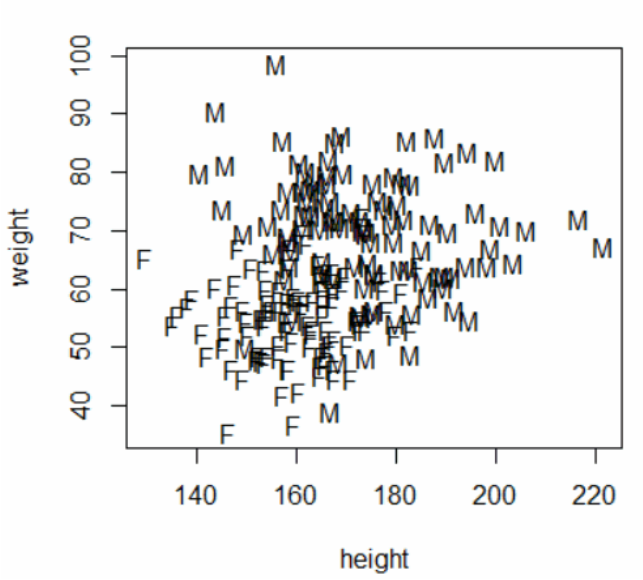


Διάγραμμα 5.11: Διάγραμμα διασποράς με διαφορετικό τύπο σημείων ανά επίπεδο κατηγορικής μεταβλητής

Τέλος, μπορούμε να χρησιμοποιήσουμε και την εντολή `text()`, η οποία φτιάχνει διάγραμμα διασποράς, όπου αντί για σημεία εμφανίζει χαρακτήρες. Αυτοί δίνονται από ένα διάνυσμα, το οποίο δηλώνεται μέσω της παραμέτρου `labels`. Για παράδειγμα, οι εντολές

```
> plot(height,weight,type="n")
> text(height,weight,labels=sex)
```

εμφανίζουν σε κάθε σημείο το σύμβολο που υπάρχει στην αντίστοιχη θέση του διανύσματος `sex` (βλ. Διάγραμμα 5.3).



Διάγραμμα 5.12: Διάγραμμα διασποράς με διαφορετικό γράμμα ανά επίπεδο κατηγορικής μεταβλητής

5.4 Πολλαπλά διαγράμματα διασποράς

Αν έχουμε έναν πίνακα, κάθε στήλη του οποίου αντιστοιχεί στις τιμές μιας μεταβλητής που έχουμε συλλέξει στο δείγμα μας, η εντολή `pairs()` με παράμετρο τον πίνακα εμφανίζει τα διαγράμματα διασποράς όλων των δειγμάτων ανά δύο. Έτσι, λοιπόν, οι ακόλουθες εντολές θα μας δώσουν την απεικόνιση του Διαγράμματος 5.4.

```
> e<-rnorm(200,0,15)
> z<-30-3*x+e
> matrix1<-cbind(x,y,z)
> dimnames(matrix1)<-list(NULL,c("X","Y","Z"))
> pairs(matrix1)                                    ■.
```

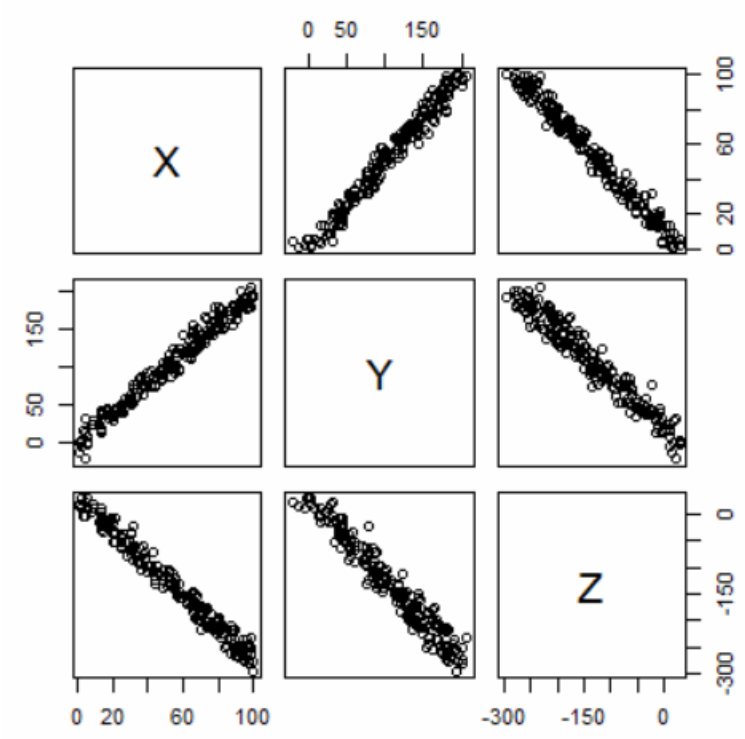
Στις παραπάνω εντολές, τα διανύσματα `x`, `y` και `z` ενώθηκαν σε ένα πίνακα με την εντολή `cbind`, όπου η κάθε στήλη του πίνακα είναι ένα από τα διανύσματα που ενώσαμε.

Όσον αφορά το διάγραμμα, ποια μεταβλητή αναλογεί σε κάθε άξονα φαίνεται από τη σειρά χαρακτήρων που βρίσκεται πάνω ή κάτω του (αν είναι οριζόντιος) και αριστερά ή δεξιά του (αν είναι κατακόρυφος) και δεν είναι τίποτα άλλο από το όνομα του αντίστοιχου διανύσματος. Για παράδειγμα, το διάγραμμα κάτω αριστερά έχει την `X` στον οριζόντιο και την `Z` στον κατακόρυφο.

5.5 Περαιτέρω παραμετροποίηση διαγραμμάτων

Σε κάθε εντολή δημιουργίας διαγράμματος μπορούν να δοθούν και παράμετροι, οι οποίες βελτιώνουν την εμφάνιση του. Συγκεκριμένα, οι παράμετροι:

- `main="κείμενο"` ορίζει τίτλο για το διάγραμμα,



Διάγραμμα 5.13: Πολλαπλά διαγράμματα διασποράς

- `sub="κείμενο"` ορίζει υπότιτλο (τοποθετείται κάτω από το διάγραμμα),
- `xlab="κείμενο"` ορίζει όνομα για τον οριζόντιο άξονα και
- `ylab="κείμενο"` ορίζει όνομα για τον κατακόρυφο άξονα.

Για παράδειγμα, η εντολή

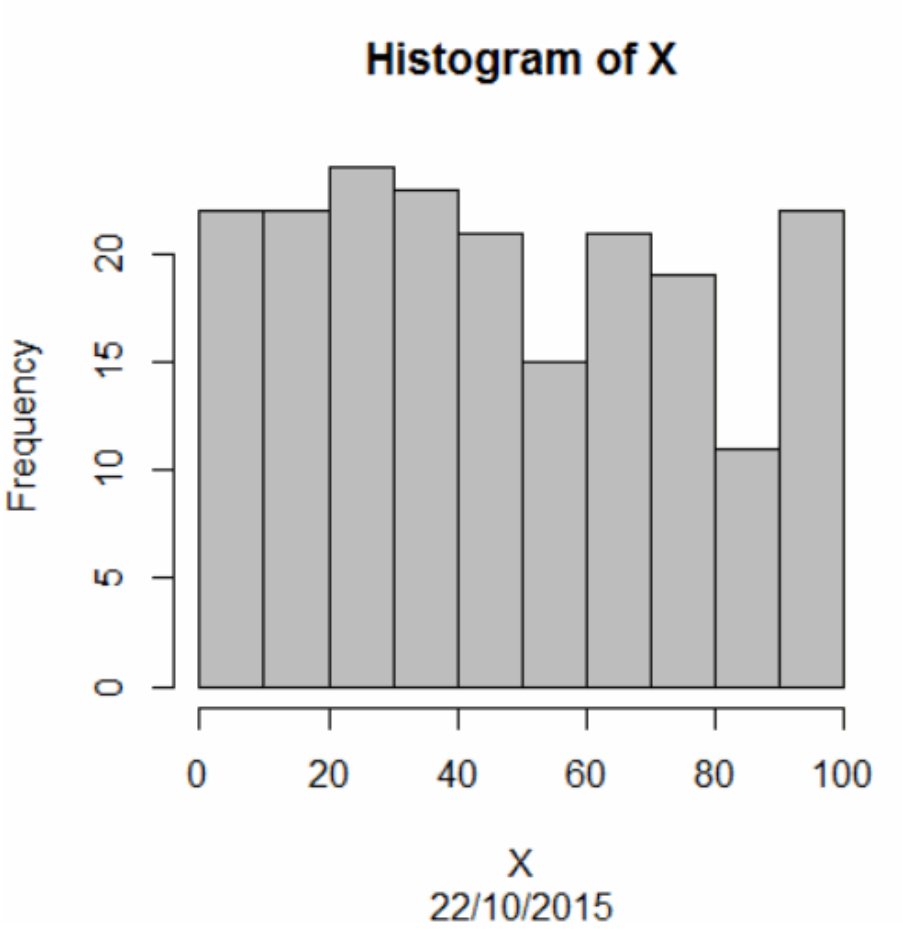
```
> hist(x,main="Histogram of X",sub="22/10/2015",xlab="X",ylab="
   Frequency", col="gray")
```

δίνει το Διάγραμμα 5.5.

Οι παραπάνω παράμετροι μπορούν να εισαχθούν στην εντολή `title()`, οπότε και εισάγονται σε ήδη ανοικτό διάγραμμα. Τότε, όμως, δεν σβήνουν τυχόν προηγούμενους τίτλους.

Δύο άλλες χρήσιμες παράμετροι της εντολής `plot()` ή των παραλλαγών της είναι οι `xlim` και `ylim`, οι οποίες ορίζονται από ένα διάνυσμα, οι τιμές του οποίου καθορίζουν τα όρια του οριζόντιου και του κατακόρυφου άξονα, αντίστοιχα. Για παράδειγμα, μπορείτε να δείτε το αποτέλεσμα της εντολής `plot(x, y, xlim=c(-50, 300), ylim=c(-50, 500))` στο Διάγραμμα 5.5.

Η εντολή `legend()` προσθέτει λεζάντα σε ένα διάγραμμα. Η λεζάντα εμφανίζει κάποια χαρακτηριστικά του διαγράμματος που ορίζει ο χρήστης (π.χ. τύπο γραμμής ή είδος σημείου) και ένα επεξηγηματικό κείμενο για το κάθε χαρακτηριστικό.



Διάγραμμα 5.14: Ιστόγραμμα

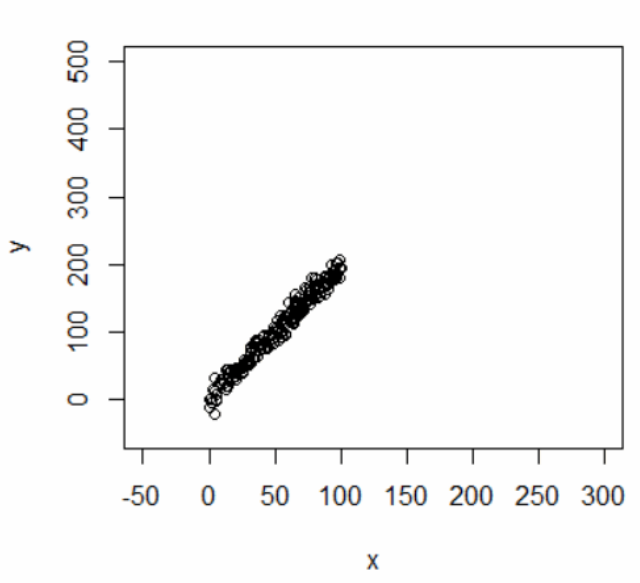
Στο διάγραμμα του ύψους-βάρους (ξανατρέξτε τις αντίστοιχες εντολές για να το ξανακάνετε) μπορούμε να προσθέσουμε μία λεζάντα (βλ. Διάγραμμα 5.5) που επεξηγεί τα σημεία με την εντολή

```
> legend( 138, 98 ,pch=c(1,3),legend=c("Males","Females"), col=2:3)
```

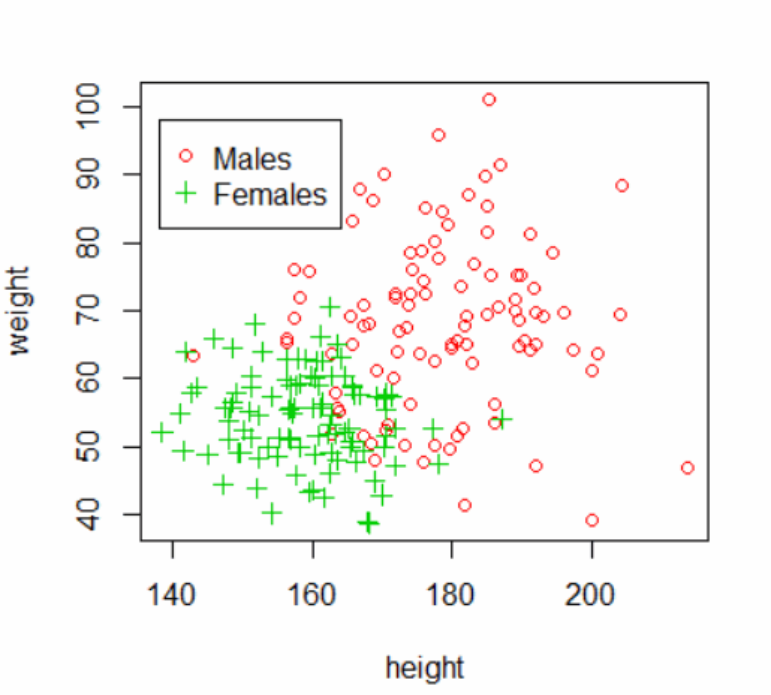
στην οποία τα δύο πρώτα νούμερα δίνουν τις συντεταγμένες (σε τιμές του οριζόντιου και κάθετου άξονα) της πάνω δεξιάς γωνίας του πλαισίου της λεζάντας, το η παράμετρος `pch` ορίζει ότι έχουμε δύο τύπους σημείων (με κωδικούς 1 και 3). Η παράμετρος `col` ορίζει ότι έχουμε δύο χρώματα στα σημεία (με κωδικούς 2 και 3) και τέλος η παράμετρος `legend` καθορίζει τη λεκτική περιγραφή των σημείων.

Στην παραπάνω εντολή μπορούμε να προκαθορίσουμε τις συντεταγμένες με μία από τις ακόλουθες σειρές χαρακτήρων: `"bottomright"`, `"bottom"`, `"bottomleft"`, `"left"`, `"topleft"`, `"top"`, `"topright"`, `"right"` και `"center"`. Αυτές μας ορίζουν προκαθορισμένες θέσεις για τη λεζάντα στο διάγραμμα. Για παράδειγμα, η εντολή

```
> legend( 'topleft' ,pch=c(1,3),legend=c("Males","Females"),
  col=2:3)
```

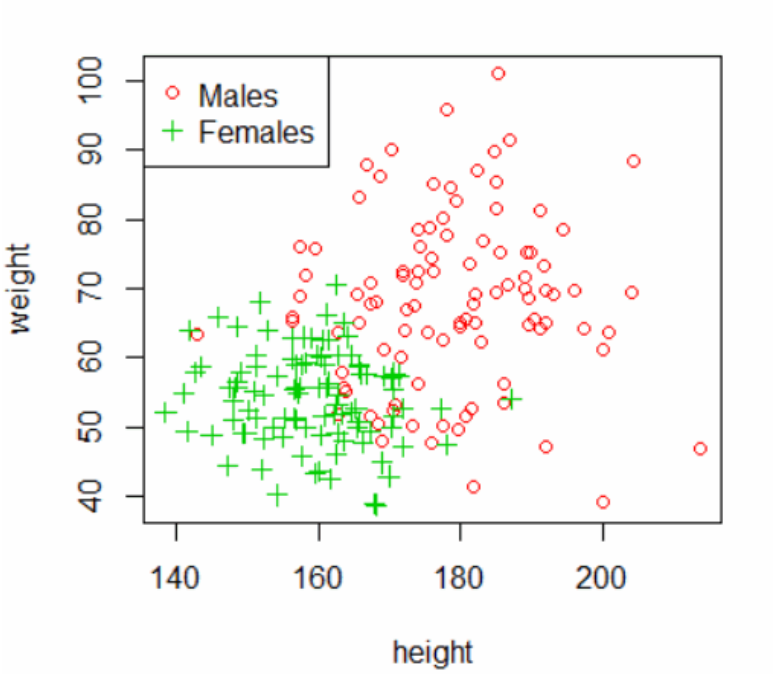


Διάγραμμα 5.15: Διάγραμμα διασποράς



Διάγραμμα 5.16: Διάγραμμα διασποράς με λεζάντα στην πάνω δεξιά γωνιά χρησιμοποιώντας συντεταγμένες

θα μας δώσει το Διάγραμμα 5.5, όπου η λεζάντα τυπώθηκε στην πάνω δεξιά γωνία του διαγράμματος.



Διάγραμμα 5.17: Διάγραμμα διασποράς με λεζάντα στην πάνω δεξιά γωνιά

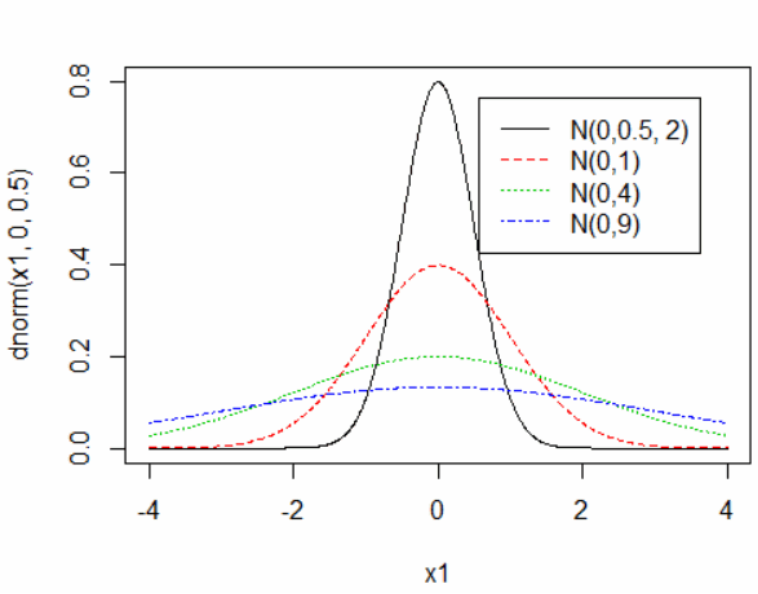
Τέλος, μπορείτε να τυπώσετε τη λεζάντα σε όποιο σημείο θέλετε διαδραστικά, βάζοντας αντί των συντεταγμένων την τιμή-παράμετρο `locator(1)`. Η παράμετρος αυτή θα σας εμφανίσει ένα «σταυρό» στην οθόνη, τον οποίο μπορείτε να τον μετακινήσετε με το ποντίκι. Οδηγήστε με το ποντίκι το σταυρό στο επιθυμητό σημείο εκτύπωσης της πάνω δεξιάς γωνίας του πλαισίου της λεζάντας και πατάτε το αριστερό πλήκτρο του ποντικιού. Η χρήση της παραμέτρου `locator(1)` σημαίνει ότι εσείς θα δείξετε, πατώντας το αριστερό πλήκτρο του ποντικιού, που θα τοποθετηθεί η λεζάντα.

Στο διάγραμμα των τεσσάρων κανονικών κατανομών μπορούμε να προσθέσουμε μία λεζάντα που επεξηγεί τις γραμμές με την εντολή

```
>legend(locator(1),lty=1:4,legend=c('N(0,0.5, 2)', 'N(0,1)','N(0,4)
  ','N(0,9)'),col=1:4)                                          ■.
```

Η εντολή `identify()` ζητάει να εμφανιστεί πληροφορία (ορίζουμε το είδος της με την παράμετρο `labels`) για κάθε σημείο ενός διαγράμματος διασποράς κοντά στο οποίο «κλικάρουμε». Για παράδειγμα, παρακάτω δημιουργούμε διάγραμμα ύψους και βάρους χωρίς να διαφοροποιούμε τα σημεία ανάλογα με το φύλο. Με την `identify()` που χρησιμοποιούμε μας εμφανίζεται το διάγραμμα και κλικάροντας κοντά σε σημεία εμφανίζεται ένα γράμμα που δηλώνει το φύλο. Αυτό συνεχίζεται μέχρι να πατήσουμε το πλήκτρο ESC.

```
> plot(height,weight)
```



Διάγραμμα 5.18: Διάγραμμα των συναρτήσεων (μάζας) πυκνότητας πιθανότητας των τεσσάρων Κανονικών κατανομών με λεζάντα, που καθορίζει ο χρήστης που θα τοποθετηθεί

```
> identify(height,weight,labels=sex)                                    ■.
```

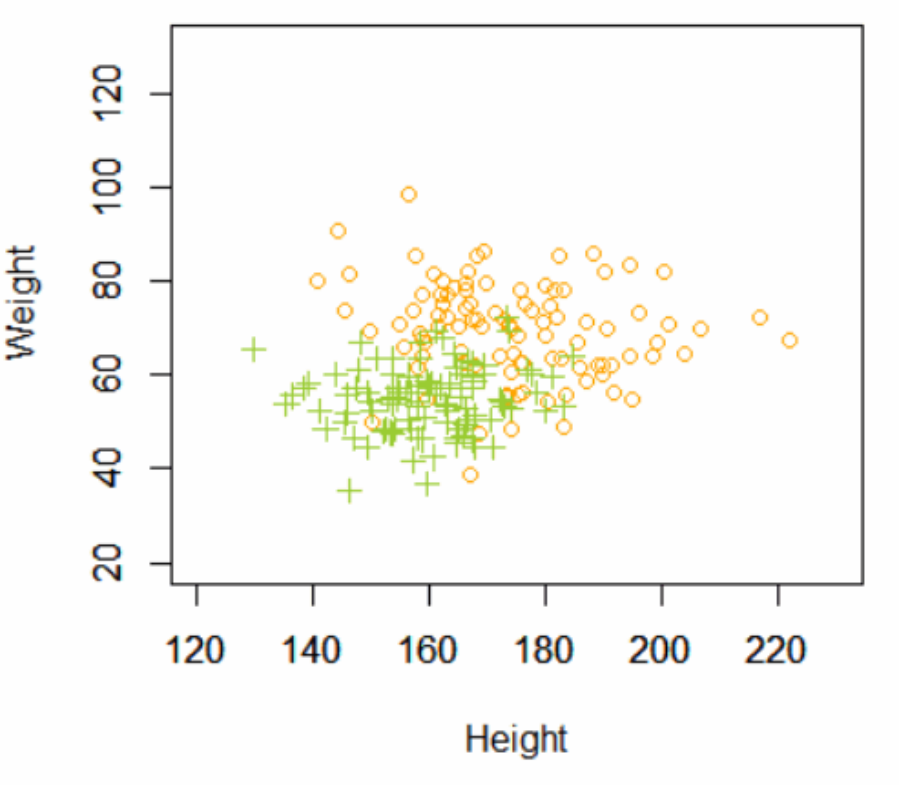
Δοκιμάστε την, και κατόπιν δοκιμάστε και με `labels=weight`. Δίπλα σε κάθε σημείο που κλικάρετε θα εμφανίζεται το βάρος. Αν δεν δώσετε καθόλου `labels`, τότε εμφανίζεται η θέση του σημείου στο διάνυσμα.

5.6 Πολλά διαγράμματα στο ίδιο πλαίσιο

Μόλις δημιουργήσουμε ένα διάγραμμα, τότε αυτό σβήνει το προηγούμενο. Αν δεν θέλουμε να χαθεί το προηγούμενο, αλλά το νέο να μπει στο ίδιο πλαίσιο, πρέπει πριν δώσουμε την εντολή δημιουργίας του δεύτερου διαγράμματος να δώσουμε την εντολή `par(new=T)`.

Απαιτείται προσοχή όμως, γιατί η R θα δημιουργήσει το νέο διάγραμμα με τους δικούς του άξονες και τίτλους, οι οποίοι μπορεί να μην έχουν καμία σχέση με του προηγούμενου (π.χ. άλλα όρια αξόνων, ένας τίτλος πάνω στον άλλο, κλπ). Γι' αυτόν τον λόγο, πρέπει να γίνεται προσεκτική χρήση των παραμέτρων `xlim`, `ylim`, `xlab`, `ylab`, κλπ. Δείτε ένα παράδειγμα:

```
> plot(height[sex=="M"],weight[sex=="M"],pch=1, xlab="Height",
  ylab="Weight",xlim=c(120,230),ylim=c(20,130))
> par(new=T)
> plot(height[sex=="F"],weight[sex=="F"],pch=3,xlab="",ylab="",
  xlim=c(120,230),ylim=c(20,130))                                       ■.
```



Διάγραμμα 5.19: Διάγραμμα διασποράς (7)

5.7 Πολλά διαγράμματα στο ίδιο παράθυρο

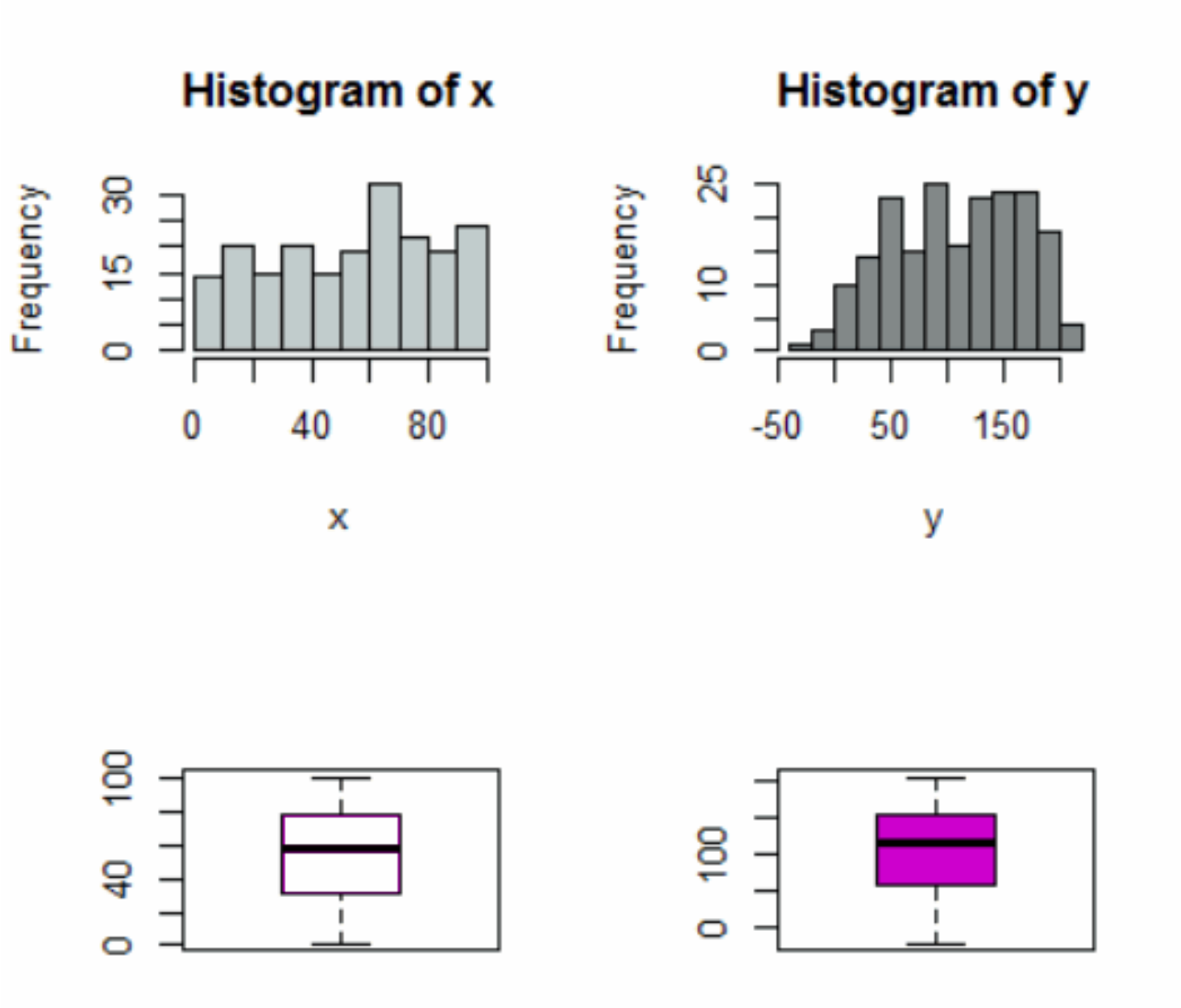
Μπορούμε στο ίδιο παράθυρο να έχουμε πολλά διαγράμματα σε μία διάταξη με γραμμές και στήλες.

Με την εντολή `par(mfrow=c(a,b))`, όπου στη θέση των `a` και `b` δηλώνουμε αριθμούς, το παράθυρο χωρίζεται σε `a` γραμμές και `b` στήλες και στην κάθε θέση (a*b συνολικά) μπορεί να δεχτεί ένα διάγραμμα. Το πρώτο διάγραμμα τοποθετείται στη θέση πάνω αριστερά, το δεύτερο δίπλα του, κ.ο.κ. Μόλις γεμίσει μία γραμμή, συνεχίζουμε στην επόμενη. Όταν γεμίσουν όλες οι γραμμές, τότε το επόμενο διάγραμμα μπαίνει πάλι επάνω αριστερά και τα υπόλοιπα σβήνονται. Αντίστοιχα, η εντολή `par(mfcol=c(a,b))`, τοποθετεί τα διαγράμματα κατά στήλες.

Ο χρήστης δεν μπορεί να ελέγξει που θα τοποθετηθεί το επόμενο διάγραμμα. Θα πάει πάντα δίπλα στο προηγούμενο. Δείτε το ακόλουθο παράδειγμα:

```
> par(mfrow=c(2,2))
> hist(x)
> hist(y)
> boxplot(x)
> boxplot(y)                                                    ■.
```

Με την εντολή `par(mfrow=c(1,1))` ή κλείνοντας το παράθυρο τελείως και ανοίγοντας



Διάγραμμα 5.20: Πολλά διαγράμματα στο ίδιο παράθυρο (1)

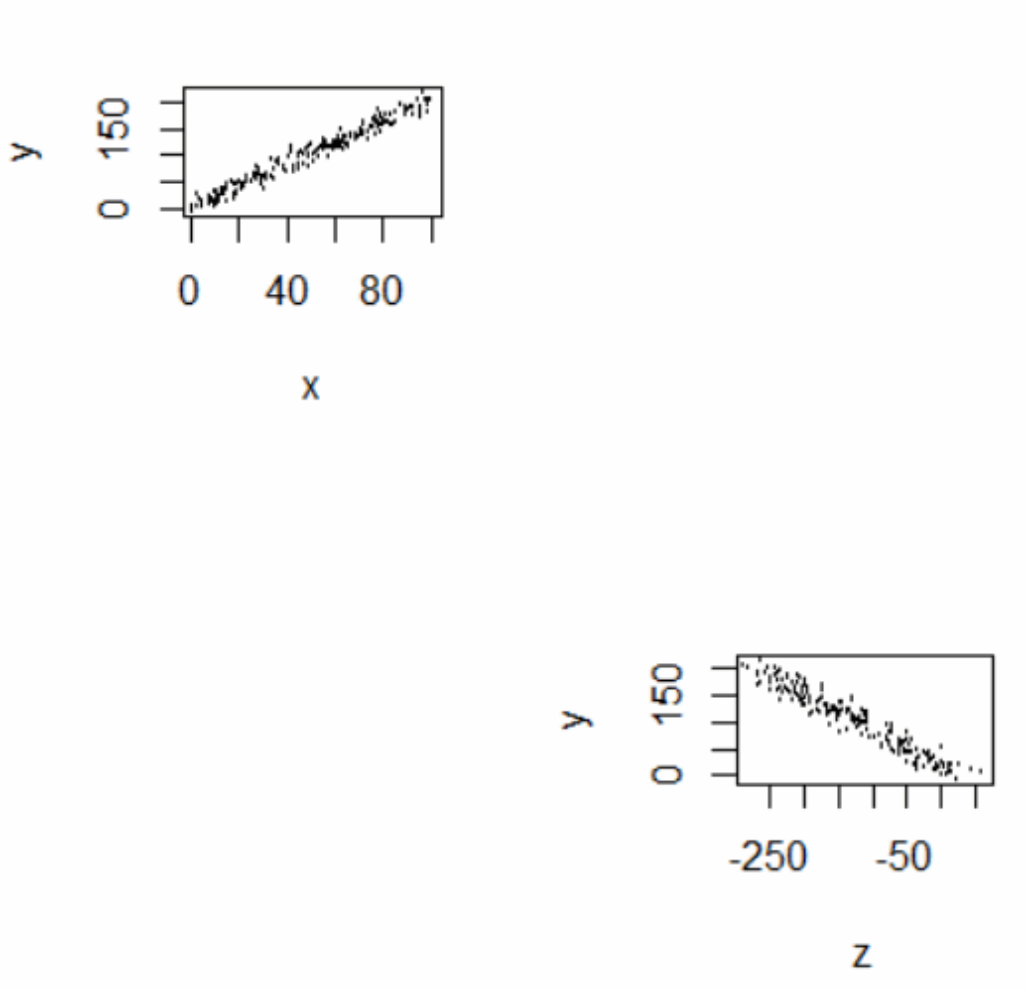
νέο επανερχόμαστε σε ένα ενιαίο παράθυρο.

Η εντολή `split.screen(a,b)`, επίσης, χωρίζει το παράθυρο σε `a` γραμμές και `b` στήλες με την κάθε θέση να μπορεί να δεχτεί ένα διάγραμμα. Τώρα, όμως, κάθε θέση έχει έναν αριθμό που της αναλογεί: η επάνω αριστερά έχει αριθμό 1, η δίπλα από αυτήν αριθμό 2, κ.ο.κ. Ο χρήστης μπορεί να ορίσει σε ποια θέση θα μπει το επόμενο διάγραμμα, δίνοντας την εντολή `screen` (αριθμός θέσης). Όσο δεν ορίζει νέα θέση, τα νέα διαγράμματα που ζητάει θα μπαίνουν στην ίδια σβήνοντας τα προηγούμενα. Όμως, αν σβήνεται κάποιο προηγούμενο, σβήνονται και όσα υπάρχουν σε άλλες θέσεις. Δίνουμε ένα παράδειγμα:

```
> split.screen(c(2,3))
[1] 1 2 3 4 5 6
```

```
> screen(1)
> plot(x,y,cex=0.3)
> screen(5)
> plot(z,y,cex=0.3)
```

το οποίο έχει ως αποτέλεσμα το Διάγραμμα 5.7.



Διάγραμμα 5.21: Πολλά διαγράμματα στο ίδιο παράθυρο (2)

Αν τώρα ζητήσουμε διάγραμμα στη θέση 1 ή στην 5, θα σβηστούν όλα όσα υπάρχουν στην οθόνη.

Η οθόνη ξαναγίνεται ενιαία αν δώσουμε την εντολή `close.screen(all=T)` ή φυσικά αν κλείσουμε το παράθυρο και ανοίξουμε νέο.

Και με τους δύο τρόπους μοιράσματος μπορούμε να χρησιμοποιούμε την `par(new=T)`. Το επόμενο διάγραμμα θα μπαίνει στην ίδια θέση με το προηγούμενο χωρίς να το σβήνει.

5.8 Πολλά παράθυρα γραφικών

Όπως είδαμε, μόλις δώσουμε εντολή δημιουργίας ενός διαγράμματος, αυτόματα ανοίγει ένα παράθυρο γραφικών στο οποίο εμφανίζεται το διάγραμμα. Κάθε νέο διάγραμμα που δημιουργείται, σβήνει το προηγούμενο και μπαίνει στη θέση του.

Όταν ένα μοναδικό παράθυρο γραφικών είναι ανοικτό, αυτό έχει τον αριθμό 2. Μπορούμε να ανοίξουμε περισσότερα παράθυρα με διαδοχικές εκτελέσεις της εντολής `graphsheet()`. Κάθε εκτέλεση της ανοίγει ένα νέο κενό παράθυρο γραφικών. Ανά πάσα στιγμή μόνο ένα παράθυρο είναι ενεργό και όποιο διάγραμμα δημιουργούμε τοποθετείται εκεί, σβήνοντας τυχόν προηγούμενο. Για να δούμε ποια παράθυρα είναι ανοικτά και ποιος ο αριθμός του καθενός, εκτελούμε την εντολή `dev.list()`. Ο αριθμός κάθε παραθύρου εμφανίζεται επίσης στην κορυφή του. Ο αριθμός του ενεργού παραθύρου εμφανίζεται με την εντολή `dev.cur()`. Για να ορίσουμε ένα συγκεκριμένο παράθυρο ως το ενεργό, π.χ. το παράθυρο 4, δίνουμε την εντολή `dev.set(4)`. Μπορούμε, τέλος, να κλείσουμε ένα παράθυρο με την εντολή `dev.off(##)`, όπου `##` είναι ο αριθμός παραθύρου. Η αρίθμηση των άλλων παραθύρων δεν αλλάζει, και αν ανοίξουμε νέο παράθυρο θα πάρει τον αριθμό εκείνου που σβήστηκε. Τέλος, με την εντολή `graphics.off()` κλείνουμε όλα τα παράθυρα γραφικών.

Θέτοντας ένα παράθυρο ως το ενεργό μπορούμε να το χωρίσουμε σε μικρότερα παράθυρα με χρήση των `par(mfrow=. . . )` και `split.screen()`, χωρίς να επηρεαστούν τα υπόλοιπα.

5.9 Επιπλέον βιβλιογραφικό υλικό

Στην ενότητα αυτή είδαμε μόνο κάποιες βασικές γραφικές λειτουργίες της R. Οι δυνατότητες της είναι πολύ μεγάλες ειδικά με χρήση προχωρημένων βιβλιοθηκών όπως η `ggplot` που είναι και η πιο δημοφιλής. Για τις επιπλέον γραφικές δυνατότητες της R, παραπέμπουμε στο βιβλία των Keen (2010), Murrell (2011) και Chang (2013).

Βιβλιογραφικές Αναφορές Κεφαλαίου 5

Chang, W. (2013). *R Graphics Cookbook*. Oreilly and Associate Series. O'Reilly Media, Incorporated.

Keen, K. (2010). *Graphics for Statistics and Data Analysis with R*. Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis.

Murrell, P. (2011). *R Graphics, Second Edition*. Second Edition, Chapman & Hall/CRC The R Series. CRC Press.

ΚΕΦΑΛΑΙΟ 6

Βρόχοι και υπο-συνθήκη εκτέλεση εντολών

Η R επιτρέπει τη χρήση δομημένων εντολών και, συνεπώς, προσφέρει τη δυνατότητα πολυπλοκότερων υπολογισμών. Στις σημειώσεις αυτές, θα μας απασχολήσουν μόνο κάποιες από τις πολλές τέτοιες δυνατότητες και συγκεκριμένα οι εντολές `for`, `if` και `while`. Όλες αυτές οι εντολές είναι εντολές ελέγχου και στη R μπορείτε να καλέσετε το αντίστοιχο αρχείο βοήθειας με την εντολή `help(Control)`.

6.1 Υπο-συνθήκη εκτέλεση εντολών

6.1.1 Σύνταξη `if`

Η εντολή `if` επιτρέπει να εκτελέσουμε κάποια ή κάποιες εντολές, μόνο στη περίπτωση που κάποια συνθήκη ισχύει. Η γενική της σύνταξη έχει τη μορφή:

```
if (συνθήκη) εντολή
```

ή

```
if (συνθήκη) { εντολή #1; εντολή #2; . . . εντολή #n }
```

ή

```
if (συνθήκη) {
      εντολή #1
      εντολή #2
      ⋮
      εντολή #n
      }                                                    ■.
```

Η λειτουργία της είναι πως πρώτα ελέγχεται η συνθήκη και αν είναι αληθής (`TRUE`), τότε εκτελείται η εντολή ή οι εντολές που ακολουθούν. Αν οι εντολές είναι περισσότερες από μία, τότε

πρέπει να τις συμπεριλάβουμε μέσα σε άγκιστρα. Μερικά παραδείγματα στη χρήση της εντολής είναι τα ακόλουθα:

```
> x<-5
> y<-4
> if (x>=5) y<-0
> y
[1] 0
> print(c(x,y))
[1] 5 0
> if (x<5)y<-10
> y
[1] 0
> print(c(x,y))
[1] 5 0
> x <- 5                                                    ■.
```

Παρατηρείστε πως κατά την εκτέλεση μιας εντολής `if`, η R εμφανίζει το αποτέλεσμα. Στη περίπτωση που θέλουμε να χρησιμοποιήσουμε το `if`, με περισσότερες από μια εντολές, τότε η σύνταξη έχει την ακόλουθη μορφή:

```
> x<-5
> y<-4
> if (x<=5) {
+     y<-10
+     x<-x^2
+ }
> print(c(x,y))
[1] 25 10
> if(x > 0){
+    print("Θετικός αριθμός")
+ }
[1] "Θετικός αριθμός"                                       ■.
```

Στις περιπτώσεις που δεν ισχύει η συνθήκη, αλλά θα θέλαμε να εκτελεστεί κάποια άλλη εντολή, τότε πρέπει να χρησιμοποιήσουμε την σύνταξη:

```
     if (συνθήκη) { εντολές #1 }  else  { εντολές #2 }      ■.
```

Η εντολή αυτή ελέγχει αν ισχύει η συνθήκη. Αν ισχύει, εκτελείται η πρώτη ομάδα εντολών «`εντολές #1`», αλλιώς εκτελείται η δεύτερη ομάδα εντολών «`εντολές #2`». Και πάλι, αν οι «`εντολές #1`» ή οι «`εντολές #2`» είναι περισσότερες από μια, πρέπει να τις περιλάβουμε μέσα σε άγκιστρα. Επίσης, η εντολή `else` πρέπει να είναι στην ίδια γραμμή που

τελειώνει το κομμάτι της `if`, είτε αυτή είναι μια εντολή, είτε το άγκιστρο που δηλώνει το τέλος των εντολών της ομάδας «`εντολές #1`», γιατί αλλιώς η R θα καταλάβει πως πρόκειται για μια απλή `if` εντολή. Ουσιαστικά, μια τέτοια εντολή είναι ισοδύναμη με δύο απλά `if`. Η βασική διαφορά είναι ότι η πραγματοποίηση συγκρίσεων από τον υπολογιστή είναι πολύ χρονοβόρα διαδικασία. Γι' αυτό το λόγο, καλό είναι να αποφεύγουμε τις πολλές συγκρίσεις και η χρήση της εντολής `if else` είναι πιο αποτελεσματική.

΄Ενα παράδειγμα της χρήσης της εντολής είναι το ακόλουθο:

```
> x<-4
> y<-5
> if (x<5) { y<-0
+           x<-x^2
+ } else {
+              x<-0
+              y<-10
+            }
> print(c(x,y))
[1] 16  0                                                    ■.
```

Διανύσματα δεν μπορούν να χρησιμοποιηθούν μέσα στη συνθήκη, όμως η συνθήκη μπορεί να περιέχει παράσταση. ΄Ετσι, συνθήκη της μορφής `x^2+5+sqrt(x+4)<0` είναι αποδεκτή αν το `x` είναι αριθμός, όχι όμως αν είναι διάνυσμα.

Για παράδειγμα, ας δούμε την περίπτωση της διαμέσου. Γνωρίζουμε καλά πως η διάμεσος, στην περίπτωση άρτιου μεγέθους δείγματος, είναι ο αριθμητικός μέσος των δύο κεντρικών παρατηρήσεων, ενώ αν το μέγεθος του δείγματος είναι περιττός αριθμός, η διάμεσος είναι απλά η κεντρική παρατήρηση που έχει τόσες παρατηρήσεις μικρότερες της, όσο και μεγαλύτερες της.

Οι παρακάτω εντολές υλοποιούν τον ορισμό της διαμέσου για οποιοδήποτε διάνυσμα x. Στο παράδειγμα αυτό έχουμε χρησιμοποιήσει 100 τυχαίες τιμές από την τυποποιημένη κανονική κατανομή που παράγαμε μέσω της εντολής `rnorm`. Παρατηρείστε τον τρόπο με τον οποίο ελέγχουμε αν ένας αριθμός είναι άρτιος ή όχι. Κοιτάζουμε αν το ακέραιο υπόλοιπο μετά τη διαίρεση με το δύο είναι μηδέν ή όχι.

```
> x<-rnorm(100)
> y<-sort(x)
> n<-length(x)
> if (n%%2 ==0) M<-(y[n/2] + y [n/2 +1] )/2
> if (n%%2 !=0) M<- y[(n+1)/2]
>  M
[1] -0.007235315                                             ■.
```

Η παραπάνω υλοποίηση έχει το μειονέκτημα ότι χρησιμοποιεί δύο `if` και αυτό, όπως είπαμε, είναι χρονοβόρο. Μπορούμε να χρησιμοποιήσουμε μια εντολή `if else` για τον ίδιο σκοπό.

Οι εντολές θα είναι:

```
> x<-rnorm(100)
> y<-sort(x)
> n<-length(x)
> if (n%%2==0) {
+    M<-(y[n/2]+y [n/2 +1])/2
+ }else {
+    M<-y[(n+1)/2]
+ }
> M
[1] -0.2292315                                                    ■.
```

Είναι σημαντικό να αναφέρουμε πως μέσα στην παράσταση ελέγχου της εντολής `if`, θα πρέπει να υπάρχει αριθμητική παράσταση και όχι διάνυσμα. Αν υπάρχει διάνυσμα, τότε η R θα χρησιμοποιήσει για τη σύγκριση μόνο το πρώτο στοιχείο του. Σ' αυτές τις περιπτώσεις υπάρχουν δύο εναλλακτικές. Η πρώτη αφορά τη χρήση κάποιας επαναληπτικής εντολής, όπως θα δούμε σε λίγο. Η δεύτερη απαιτεί τη χρήση εκφράσεων τύπου Boolean, που προαναφέραμε. Για παράδειγμα, αν έχουμε ένα διάνυσμα με τιμές στο διάστημα (0,1) και θέλουμε να φτιάξουμε ένα άλλο διάνυσμα με τιμές -1 και 1 ανάλογα με το αν η τιμή του πρώτου διανύσματος είναι μικρότερη ή όχι του 0.5, μπορούμε να γράψουμε την εξής εντολή:

```
> u
 [1] 0.1148585 0.7938276 0.5504808 0.8947566 0.5662130
 [6] 0.5470632 0.9398213 0.2855965 0.7676873 0.7300850
> t<- (u<0.5)*(-1) + (u>=0.5)
> t
 [1] -1  1  1  1  1  1  1 -1  1  1                                ■.
```

Τέτοιες εκφράσεις έχουν την ενδιαφέρουσα ιδιότητα, ότι είναι πολύ πιο γρήγορες από επαναληπτικές εντολές, και για αυτό θα πρέπει να προτιμούνται, όταν είναι αυτό εφικτό.

Τελειώνοντας τη σύντομη περιγραφή της εντολής `if`, θα πρέπει να τονίσουμε ότι υπάρχει η δυνατότητα για φωλιασμένες εντολές `if`, δηλαδή εντολές που εμφανίζονται η μία μέσα στην άλλη. Για πολλαπλούς ελέγχους χρησιμοποιούμε τη κλιμακωτή `if – else if` εντολή ή να συνδυάσουμε συνεχόμενα `else if`. Παράδειγμα:

```
> x <- 5
> if (x ==10){
+     print('same')
+ } else if (x > 1){
+            print('bigger')
+        } else {
+            print('smaller')
```

```
+ }
[1] "bigger"                                                    ■.
```

Στην παραπάνω σύνταξη, όταν $x = 10$ τυπώνεται η λέξη "same", αλλιώς ελέγχεται άν $x > 1$ (και $x \neq 1$) και τυπώνεται η λέξη "bigger' διαφορετικά (για $x \leq 1$ και $x \neq 10$) τυπώνεται η λέξη "smaller". Το ίδιο αποτέλεσμα μπορεί να επιτευχθεί και με τη χρήση λογικών τελεστών• π.χ. & και | - βλ. Ενότητα 2.2.3 για λεπτομέρειες. Έτσι λοιπόν η παραπάνω σύνταξη μπορεί να γραφτεί ισοδύναμα

```
> x <- 5
> if (x ==10) print('same')
> if ( (x>1)&(x!=10) ) print('same')
> if ( (x<=1)  ) print('same')                                  ■.
```

6.1.2 Εντολή `ifelse`

Η εντολή `ifelse` επιστρέφει μια τιμή ίδιας κλάσης με το εισερχόμενο αντικείμενο με στοιχεία που γεμίζουν με τις αντίστοιχες τιμές που θα του δώσουμε εδώ ως εισερχόμενα. Γενικά η σύνταξη της είναι η ακόλουθη

```
ifelse( test, yes, no )                                         ■,
```

όπου `test` είναι ένα λογικό αντικείμενο (για διανύσματα, πίνακες και arrays), `yes` είναι οι τιμές που θα αντιστοιχηθούν στις αληθείς συνθήκες (δηλαδή στις τιμές `TRUE` του λογικού διανύσματος) και `no` είναι οι τιμές που θα αντιστοιχηθούν στις ψευδείς συνθήκες (δηλαδή στις τιμές `FALSE` του λογικού διανύσματος). Για παράδειγμα, η σύνταξη

```
> x<-c(T,F,F,T)
> ifelse(x,5,99)
[1]  5 99 99  5
```

μας δίνει σαν αποτέλεσμα ένα διάνυσμα μήκους ίσο με το x αλλά για κάθε στοιχείο έχουμε την τιμή 5 αντί για `TRUE` και την τιμή 99 αντί για `FALSE`. Αντίστοιχα είναι και τα ακόλουθα παραδείγματα για πίνακες:

```
> x
      [,1]  [,2]  [,3]  [,4]  [,5]
[1,]  TRUE  TRUE FALSE  TRUE  TRUE
[2,] FALSE  TRUE FALSE  TRUE  TRUE
[3,]  TRUE  TRUE FALSE FALSE  TRUE
[4,] FALSE FALSE  TRUE FALSE FALSE
> ifelse(x,5,99)
     [,1] [,2] [,3] [,4] [,5]
[1,]    5    5   99    5    5
[2,]   99    5   99    5    5
```

```
[3,]    5    5   99   99    5
[4,]   99   99    5   99   99                    ■.                    ■.
```

Σημείωση: Μερικές φορές είναι προτιμότερο να χρησιμοποιούμε τη σύνταξη

```
> x<-c(T,F,F,T,F,F)
> yes <- 5
> no <- 99
> z<-rep(yes, length(x))
> z[!x]<-no
> z
> z
[1]  5 99 99  5 99 99
> x
[1]  TRUE FALSE FALSE  TRUE FALSE FALSE                                    ■.
```

6.2 Επαναληπτικές διαδικασίες - Βρόχοι

6.2.1 Σύνταξη `for`

Η εντολή `for`, όπως και η εντολή `while`, μας επιτρέπει να επαναλάβουμε κάποιες εντολές περισσότερες από μια φορές. Η βασική και ουσιαστική τους διαφορά είναι πως με την εντολή `for` ξέρουμε εκ των προτέρων και καθορίζουμε εμείς πόσες φορές θα εκτελεστεί, ενώ με την εντολή `while` πρέπει να συμπεριλάβουμε μια συνθήκη ελέγχου, η οποία, όσο είναι αληθής, συνεχίζονται οι επαναλήψεις και αν δεν είναι αληθής, σταματάνε οι επαναλήψεις.

Η γενική σύνταξη της εντολής `for` είναι η ακόλουθη:

`for (name in values)` εντολές

Η μεταβλητή `name` θα πάρει όλες τις τιμές που έχουμε ορίσει στο σύνολο `values`. Συνήθως, με την εντολή `for` ενδιαφερόμαστε για διαδοχικές ακέραιες τιμές, οπότε το σύνολο αυτό έχει για παράδειγμα τη μορφή `1:100`. Μερικά παραδείγματα είναι τα ακόλουθα:

```
> t<-c(1,2,4,5)
> x<-c(10,15,25,34,55)
> for (i in t) x[i]<- -x[i]
> x
[1] -10 -15  25 -34 -55                                                   ■.
```

Στον παραπάνω κώδικα, αρχικά ορίζουμε το διάνυσμα `t` μήκους τέσσερα με τιμές (1,2,4,5) και το διάνυσμα `x` μήκους πέντε με τιμές (10,15,25,34,55). Έπειτα στο πρώτο βρόχο (for-loop) η μεταβλητή ελέγχου `i` παίρνει τιμές από αυτό το διάνυσμα `t`, δηλαδή $i \in (1,2,4,5)$. Συνεπώς, θα έχουμε για $i = 1$ το x_1 να αντικατασταθεί με το $-x_1$ (δηλαδή θα αλλάξει πρόσημο) και η ίδια αντικατάσταση θα γίνει για τα x_2, x_4 και x_5 (δηλαδή για $i = 2, 4$ και 5 αντίστοιχα). Καθώς η τιμή

3 δεν περιέχεται στο διάνυσμα `t`, από όπου παίρνει τιμές ο δείκτης του for-loop i, το x_3 δε θα υποστεί καμία αλλαγή.

Στο επόμενο παράδειγμα, ο δείκτης ελέγχου του for-loop παίρνει όλες τις τιμές από το ένα έως το πέντε. Παρατηρείστε ότι στο z δώσαμε μηδενικές αρχικές τιμές, πολλαπλασιάζοντας όλα τα στοιχεία του διανύσματος $(1, 2, 3, 4, 5)$ με το μηδέν. Στην εντολή `for` κάθε φορά επεξεργαζόμαστε ένα στοιχείο κάθε διανύσματος. Ο δείκτης ελέγχου του for-loop μπορεί να είναι και αυτή μέρος μιας αριθμητικής παράστασης ή συνάρτησης. Στην πρώτη παράσταση, κάθε στοιχείο x_i υψώνεται εις το τετράγωνο, ενώ στη δεύτερη παράσταση υπολογίζουμε ότι $y_i + z_i + i$ και μετά το αποθηκεύουμε ως το i στοιχείο του διανύσματος y. Έτσι, το y_1 θα πάρει την τιμή $y_1 + z_1 + 1 = 2 + 0 + 1 = 3$, το y_2 θα πάρει την τιμή $y_2 + z_2 + 1 = 2 + 0 + 2 = 4$, κ.ο.κ.

```
> x<-c(10,15,25,34,55)
> y<-rep(2,5)
> z<-(1:5)*0
> z
[1] 0 0 0 0 0
> for (i in 1:5) {
+   x[i]<-x[i]^2
+   y[i]<-y[i]+z[i]+i
+ }
> print(c(x,y,z))
 [1]  100  225  625 1156 3025    3    4    5    6    7    0    0
 [13]   0    0    0                                              ■.
```

Σε αντίθεση με τον παραπάνω κώδικα, στο παράδειγμα που ακολουθεί, αλλάζουμε όλο το διάνυσμα `x` μέσα στο for-loop, χωρίς ο δείκτης να παίζει ρόλο. Έτσι, λοιπόν, για πέντε διαδοχικές φορές ($i = 1, 2, 3, 4, 5$) γίνεται η εκχώρηση `x <- x^2`, δηλαδή όλα τα στοιχεία του διανύσματος αλλάζουν σαν να εφαρμόσουμε την εντολή `x[k] <- x[k]^2` για όλα τα $k = 1, 2, 3, 4, 5$ (εδώ το πέντε αναφέρεται στο μήκος του διανύσματος `x`). Για παράδειγμα, για το x_1 έχουμε τιμή 10 πριν ξεκινήσει το for-loop. Για $i = 1$, το x_1 παίρνει την τιμη $10^2 = 100$, και μετά διαδοχικά τις τιμές $100^2 = 10^4$, $(10^4)^2 = 10^8$, $(10^8)^2 = 10^{16}$, καταληγοντας να πάρει τιμή $(100^{16})^2 = 10^{32}$ για $i = 5$, δηλαδή με τον τερματισμό του βρόχου.

```
> x<-c(10,15,25,34,55)
> for (i in 1:5) x<-x^2
> x
[1] 1.000000e+32 4.314399e+37 5.421011e+44 1.017010e+49
[5] 4.915934e+55                                        ■.
```

Η εντολή `for` είναι πολύ χρήσιμη σε συνδυασμό με διανύσματα, καθώς επιτρέπει απευθείας έλεγχο σε κάθε στοιχείο του διανύσματος.

Μερικά χρήσιμα σημεία είναι τα εξής:

- Οι επαναλήψεις στην R είναι πολύ χρονοβόρες, και αν μπορούμε να τις αποφύγουμε, καλό είναι να το κάνουμε. Για δοκιμή, τρέξτε τις εντολές που ακολουθούν. Και οι δύο εντολές κάνουν το ίδιο πράγμα, αλλά η μια με τη χρήση επαναλήψεων και η άλλη με απλή εντολή, για να δείτε τη διαφορά στην ταχύτητα.

```
> x<-rnorm(10000,0,1)
> x<-x+1
> for (i in 1:10000) x[i]<-x[i]+1                                    ■.
```

- Μπορούμε να χρησιμοποιήσουμε πολλά φωλιασμένα `for`. Αυτό είναι πολύ χρήσιμο για να δουλέψουμε με πίνακες. Για παράδειγμα:

```
> x<-matrix(1:16,4,4)
> for (i in 1:4) {
+ for (j in 1:4) {
+     x[i,j]<--x[i,j]
+ }
+ }
> x
     [,1] [,2] [,3] [,4]
[1,]   -1   -5   -9  -13
[2,]   -2   -6  -10  -14
[3,]   -3   -7  -11  -15
[4,]   -4   -8  -12  -16                                             ■.
```

Παράδειγμα 6.1 *Ένα στατιστικό περιγραφικό μέτρο μεταβλητότητας είναι και το ακόλουθο*

$$\frac{\sum_{i=1}^{n}\sum_{j=1}^{n}(X_i - X_j)^2}{2n^2}.$$

Το μέτρο αυτό ορίζεται με διπλό άθροισμα, επομένως χρειαζόμαστε ένα διπλό `for` για να το υλοποιήσουμε. Οι παρακάτω εντολές υπολογίζουν αυτό το μέτρο μαζί με τη μεροληπτική διακύμανση.

```
> x<-c(18,9,15,15,19,14,6,5,20,19,20,15,18,15,9,9,12,19,14,17)
>  sum1<-0
>  n<-length(x)
>  for (i in 1:n) {
+  for (j in 1:n) {
+     sum1<-sum1+ (x[i]-x[j])^2
+  }
```

```
+   }
>   sum1<-sum1/(2*n^2)
>   print(c(sum1,(n-1)/n*var(x)))
[1] 20.64 20.64                                                    ■.
```

Βλέπουμε, και όχι τυχαία, πως το μέτρο που ορίσαμε πριν από λίγο ταυτίζεται με τη μεροληπτική διακύμανση. Μπορεί κανείς να αποδείξει πως, όντως, τα δύο μέτρα ταυτίζονται και πως η παραπάνω έκφραση είναι ένας εναλλακτικός τρόπος να ορίσουμε τη διακύμανση.

- Η μεταβλητή που χρησιμοποιούμε για να ελέγχουμε τον αριθμό των επαναλήψεων, ονομάζεται μεταβλητή ελέγχου. Καλό είναι, αυτή να μην αλλάζει τιμές κατά τη διάρκεια της εντολής `for`. Η R, σε αντίθεση με άλλες γλώσσες προγραμματισμού, δεν επηρεάζεται αν η μεταβλητή ελέγχου αλλάξει τιμή ξαφνικά. Έτσι, οι εντολές που ακολουθούν θα εκτελεστούν ακριβώς 10 φορές, άσχετα αν αλλάζουμε τιμή στο `i` μέσα στην εντολή `for`.

```
> for (i in 1:10) {
+   i<-i+10
+   print(i)
+   }
[1] 11
[1] 12
[1] 13
[1] 14
[1] 15
[1] 16
[1] 17
[1] 18
[1] 19
[1] 20                                                             ■.
```

Άλλες γλώσσες προγραμματισμού θα είχαν πρόβλημα με μια τέτοια εντολή!

6.2.2 Σύνταξη `while`

Η εντολή `while` είναι και αυτή μια επαναληπτική εντολή, αλλά ο αριθμός των επαναλήψεων δεν είναι προκαθορισμένος και εξαρτάται από κάποια συνθήκη ελέγχου. Η γενική σύνταξη είναι η ακόλουθη:

`while(συνθήκη) εντολές`

Θα πρέπει να έχουμε στο νου μας τα εξής:

- Η συνθήκη πρέπει με κάποια εντολή να μπορεί να αλλάξει τιμή, αλλιώς θα συνεχιστούν οι επαναλήψεις χωρίς να σταματήσουν ποτέ.

- Αν η συνθήκη είναι από την αρχή λανθασμένη, τότε δεν θα εκτελεστεί καμιά επανάληψη.
- Στη συνθήκη while δεν υπάρχει ένα δείκτης ορισμένος (όπως στη σύνταξη for) που να συνδέεται με τις επαναλήψεις του βρόχου. Για το λόγο αυτό πρέπει εμείς να ορίσουμε ένα δείκτη (μετρητή) που θα μετράει τις επαναλήψεις.
- Καλό θα είναι να ενσωματώνουμε πάντα μια συνθήκη η οποία θα εξασφαλίζει το τερματισμό του βρόχου όταν φτάσουμε ένα πολύ μεγάλο αριθμό επαναλήψεων έτσι να αποφύγουμε περιπτώσεις που εγκλωβιζόμαστε σε μια ατέρμονη επαναληπτική διαδικασία.

΄Ετσι ένα απλό παράδειγμα βρόχου με τη χρήση της σύνταξης while είναι το ακόλουθο

```
i<-1
while (i <100) {
       print (i)
       i<-i+1
}
```
■.

Στην παραπάνω σύνταξη, θα τυπώνεται το αντικείμενο i και μετά θα αυξάνει η τιμή του κατά μία μονάδα. ΄Ετσι λοιπόν το αντικείμενο αυτό έχει το ρόλο του δείκτη που μετράει τις επαναλήψεις του βρόχου. ΄Οταν το i φτάσει στην τιμή 100 τότε η συνθήκη του while θα σταματήσει να ισχύει και συνεπώς θα τερματιστεί ο βρόχος. Προσοχή, αν στη συνθήκη είχαμε `i>100` με αρχική τιμή `i=101` (ή μεγαλύτερη), η τιμή του δείκτη θα ενημερωνόταν αλλά η συνθήκη θα ίσχυε πάντα και συνεπώς ο βρόχος θα συνεχιζόταν για πάντα τυπώνοντας πάντα την τρέχουσα τιμή του i. Αντίστοιχα, αν αφαιρέσουμε τη σύνταξη ενημέρωσης του δείκτη (δηλαδή τη σύνταξη `i<-i+1`) τότε ο βρόχος θα τυπώνει επ' άπειρο την αρχική τιμή (στην παραπάνω σύνταξη την τιμή ένα) εφόσον η συνθήκη του while θα ισχύει συνέχεια και δε θα αλλάζει.

Στο παρακάτω παράδειγμα θα υπολογίσουμε (προσεγγιστικά) το 70% ποσοστιαίο σημείο της τυποποιημένης κανονικής κατανομής με τη βοήθεια της εντολής `pnorm`. Η εντολή `pnorm(x)` υπολογίζει την πιθανότητα μια τυχαία μεταβλητή Z που προέρχεται από την τυποποιημένη κανονική κατανομή να πάρει τιμή μικρότερη ή ίση το x, δηλαδή $\texttt{pnorm(x)} = P(Z \leq x)$ για $Z \sim N(0,1)$. Όπως θα δούμε αργότερα, η R παρέχει την συνάρτηση `qnorm` για την εύρεσή ποσοστιαίων σημείων από την κανονική κατανομή, αλλά εδώ θα χρησιμοποιήσουμε τη σύνταξη `while` για να δείξουμε πως λειτουργεί και πως μπορεί να χρησιμοποιηθεί για την επίλυση λύσεων απλών εξισώσεων• για περισσότερες λεπτομέρειες για τις σχετικές εντολές για κατανομές πιθανοτήτων βλ. Ενότητα 8.2.

΄Ετσι λοιπόν εδώ θέλουμε να λύσουμε την εξίσωση $\texttt{pnorm(x)} = 0.70$. Η τακτική που θα ακολουθήσουμε είναι να ξεκινήσουμε από μια τιμή, έστω -5 και θα προχωράμε με βήμα 0.001 αυξάνοντας την τιμή, μέχρις ότου το τρέχον σημείο να έχει συνάρτηση κατανομής μεγαλύτερη του 0.70. Παρατηρείστε πώς μεταχειριζόμαστε τη μεταβλητή ελέγχου. Προφανώς, οι εντολές μας είναι πολύ πιο αργές από τη χρήση της ήδη υπάρχουσας συνάρτησης. Το αποτέλεσμα είναι πολύ κοντά και οι διαφορές οφείλονται, στο ότι χρησιμοποιήσαμε μικρότερη ακρίβεια διαλέγοντας βήμα

0.001. Επίσης, τέτοια προβλήματα, που είναι ουσιαστικά προβλήματα επίλυσης μη γραμμικών εξισώσεων, μπορούν να λυθούν πολύ πιο εύκολα και γρήγορα με τη χρήση αριθμητικών μεθόδων.

```
> t <- -5
> testvar <- 0
> while(testvar != 10) {
+   t <- t + 0.001
+     if(pnorm(t, 0, 1) >= 0.70)
+        testvar <- 10
+ }
> t
[1] 0.525
> qnorm(0.7,0,1)
[1] 0.5244005                                                    ■.
```

Στο παραπάνω παράδειγμα, δεν θα μπορούσαμε να χρησιμοποιήσουμε εντολή `for`, καθώς δεν γνωρίζουμε εκ των προτέρων πόσες επαναλήψεις θα κάνουμε. Τέλος, ένας έμπειρος προγραμματιστής θα μπορούσε να ενσωματώσει τη μεταβλητή ελέγχου και να γράψει πιο απλά την εντολή ως:

```
> t <- -5
> while(pnorm(t, 0, 1) <= 0.70) {
+       t <- t + 0.001
+ }
> t
[1] 0.525                                                        ■.
```

Είναι ευνόητο πως μπορεί κανείς να συνδυάσει τις εντολές `for` και `while`, ή ότι μπορεί να χρησιμοποιήσει δύο ή περισσότερα διαδοχικά `while`. Η εντολή `while` έχει το πλεονέκτημα από την εντολή `for`, καθώς ο αριθμός των φορών που θα εκτελεστεί δεν είναι προδιαγεγραμμένος. Παρ' όλα αυτά, η εντολή `while` θα μπορούσε να χρησιμοποιηθεί ακριβώς, όπως η `for`, με κατάλληλη επιλογή της μεταβλητής ελέγχου.

Οι παρακάτω εντολές οδηγούν την εντολή `while` σε άπειρες επαναλήψεις, καθώς ποτέ δεν γίνεται `FALSE` η παράσταση ελέγχου και, επομένως, οι εντολές εκτελούνται επ' άπειρο (infinite loop). Σ' αυτή την περίπτωση, μπορεί να κολλήσει η R και να χάσετε ό,τι έχετε κάνει. Γι' αυτό, πρέπει να σιγουρευτείτε πως η παράσταση ελέγχου μπορεί να αλλάξει τιμή κατά την εκτέλεση της εντολής.

```
> t<-0
> i<-1
> while (t==0) {
+        i<-i+1
+ }                                                               ■.
```

6.3 Αποφυγή βρόχων με την εντολή `apply`

Όπως προαναφέραμε, οι επαναληπτικές εντολές είναι ιδιαίτερα αργές όταν υλοποιούνται στην R. Ένας τρόπος για να τις κάνουμε πιο γρήγορες, είναι με τη χρήση της εντολής `apply`. Η εντολή αυτή, έχει τη γενική σύνταξη:

```
apply( X, MARGIN, FUN )
```

όπου `X` είναι κάποιο αντικείμενο, συνήθως πίνακας ή array, `MARGIN` είναι η μεταβλητή που μας δείχνει ως προς ποια διάσταση του αντικειμένου θα επαναλάβουμε τη διαδικασία. Αν το `X` είναι πίνακας (matrix), τότε το 1 αντιστοιχεί στις γραμμές και το 2 στις στήλες. Δηλαδή, αν η παράμετρος `MARGIN` πάρει την τιμή 1, λέμε στην R να επαναλάβει τη συνάρτηση ως προς όλες τις γραμμές (π.χ. να υπολογίσει το μέσο κάθε γραμμής ξεχωριστά).

Η παράμετρος `FUN` είναι η συνάρτηση την οποία θα επαναλάβουμε και η οποία μπορεί να είναι, είτε μια από τις συναρτήσεις της R (π.χ. mean, max κλπ), είτε κάποια συνάρτηση που έχει ορίσει ο χρήστης.

Στην πραγματικότητα μπορούν να υπάρξουν και άλλες παράμετροι εισόδου, αλλά θα περιοριστούμε σε αυτή την απλή μορφή της εντολής:

```
> testmatrix<-matrix(1:12,3,4)
> testmatrix
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> apply(testmatrix,1,mean)
[1] 5.5 6.5 7.5
> apply(testmatrix,2,mean)
[1]  2  5  8 11
```

■.

Οι παραπάνω εντολές δημιουργούν έναν πίνακα και στη συνέχεια υπολογίζουν τη μέση τιμή ως προς κάθε γραμμή (όταν η παράμετρος `MARGIN` είναι 1), ή ως προς κάθε στήλη του πίνακα (όταν η παράμετρος `MARGIN` είναι 2) . Μπορεί κάποιος να επιβεβαιώσει ότι για τον συγκεκριμένο πίνακα ο μέσος της πρώτης γραμμής είναι 5.5, της δεύτερης 6.5 κλπ.

Στην πράξη, μπορούμε να χρησιμοποιήσουμε και δικές μας συναρτήσεις, για τις οποίες θα μιλήσουμε σε επόμενο κεφάλαιο. Η χρήση της εντολής `apply` προσφέρει πολύ μεγάλη ταχύτητα, και όταν είναι εφικτό θα πρέπει να την προτιμάμε από την εντολή `for`, ή όποια άλλη επαναληπτική εντολή.

6.4 Επιπλέον βιβλιογραφικό υλικό

Για επιπλέον υλικό σχετικά με τις εντολές ελέγχου στην R (και γενικότερα για προγραμματισμό) παραπέμπουμε στα βιβλία των Braun & Murdoch (2007), Chambers (2008), Matloff (2011) και

Pace (2015).

Βιβλιογραφικές Αναφορές Κεφαλαίου 6

Braun, W. & Murdoch, D. (2007). *A First Course in Statistical Programming with R*. Cambridge University Press.

Chambers, J. (2008). *Software for Data Analysis: Programming with R*. Statistics and Computing. Springer New York.

Matloff, N. (2011). *The Art of R Programming: A Tour of Statistical Software Design*. No Starch Press.

Pace, L. (2015). *Beginning R: An Introduction to Statistical Programming*. Apress.

ΚΕΦΑΛΑΙΟ 7

Συναρτήσεις

7.1 Εισαγωγή

Η R διαθέτει από μόνη της μια τεράστια ποικιλία συναρτήσεων που αφορούν, τόσο απλές μαθηματικές συναρτήσεις (π.χ. τετραγωνική ρίζα, συνημίτονο), όσο και στατιστικές συναρτήσεις (π.χ. μέση τιμή, διακύμανση), αλλά και πιο πολύπλοκες μαθηματικές διαδικασίες (π.χ. επίλυση εξισώσεων). Ένα από τα τεράστια πλεονεκτήματα της R είναι, πως ο χρήστης μπορεί να γράψει και να χρησιμοποιήσει τις δικές του συναρτήσεις που ταιριάζουν και αφορούν το συγκεκριμένο πρόβλημα, με το οποίο ασχολείται, αλλά και να τροποποιήσει ήδη υπάρχουσες συναρτήσεις. Στην ενότητα αυτή θα αναφερθούμε στον τρόπο, με τον οποίο μπορεί κανείς να γράψει και να χρησιμοποιήσει δικές του συναρτήσεις. Το να χρησιμοποιεί κανείς συναρτήσεις, είναι πάρα πολύ βολικό, όταν πρόκειται να επαναλάβει κάποιες εντολές πολλές φορές. Οι συναρτήσεις είναι γνωστές και ως υποπρογράμματα, με την έννοια πως κάθε συνάρτηση είναι στην ουσία ένα σύνολο εντολών που λύνει, ή υπολογίζει κάποια (ή κάποιες) συγκεκριμένα προβλήματα ή ποσότητες.

Η χρήση υποπρογραμμάτων - συναρτήσεων είναι πολύ χρήσιμη από προγραμματιστική άποψη, καθώς διευκολύνει πάρα πολύ τη δημιουργία προγραμμάτων μεσαίου ή μεγάλου μεγέθους. Η βασική ιδέα, για τη χρήση συναρτήσεων, είναι το σπάσιμο ενός προβλήματος σε πολλά μικρότερα προβλήματα. Μερικά από τα πλεονεκτήματα της χρήσης συναρτήσεων είναι τα εξής:

- Οι συναρτήσεις είναι συνήθως μικρές και αυτόνομες. Έτσι μπορούν να επαληθευτούν και να διορθωθούν πιο εύκολα γιατί είναι πιο εύκολο να ακολουθήσουμε τη ροή ενός μικρού κομματιού εντολών, που εκτελεί κάτι πολύ συγκεκριμένο, παρά ένα μεγαλύτερο και πιο πολύπλοκο.

- Με τη χρήση συναρτήσεων αποφεύγουμε την επανάληψη της ίδιας ομάδας εντολών πολλές φορές. Έτσι, το πρόγραμμά μας είναι πιο ευέλικτο και πιο εύκολο να διορθωθεί, αλλά και να γίνει κατανοητό. Αν, μάλιστα, έχουμε χρησιμοποιήσει κατάλληλα ονόματα, τότε το πρόγραμμα μοιάζει σαν να είναι σχεδόν σε φυσική γλώσσα.

- Με τη χρήση συναρτήσεων αρκεί να βελτιώσουμε τη συνάρτηση μια φορά και όχι να ψάχνουμε μέσα στο πρόγραμμα τις ομάδες εντολών και να κάνουμε πολλές διορθώσεις. Γενικά, είναι απλό να αλλάξουμε κάτι σε μια συνάρτηση και αυτόματα να ενημερωθεί και το πρόγραμμα που απλά, καλεί αυτή τη συνάρτηση.

- Τέλος, η χρήση συναρτήσεων βοηθάει στην καλύτερη διαχείριση μνήμης. Οι μεταβλητές που χρησιμοποιούμε μέσα στις συναρτήσεις και ονομάζονται τοπικές μεταβλητές, χρησιμοποιούν τη μνήμη μόνο, όσο χρόνο εκτελείται η συνάρτηση και παύουν να απασχολούν μνήμη όταν τελειώσει η συνάρτηση. Έτσι, επιτυγχάνεται πολύ καλύτερη διαχείριση της μνήμης του υπολογιστή.

Τελειώνοντας αυτή τη μικρή εισαγωγή στη δημιουργία συναρτήσεων της R, ας δούμε το παρακάτω παράδειγμα. Η εντολή `range` μας δίνει ένα διάνυσμα με την ελάχιστη και τη μέγιστη τιμή των τιμών ενός διανύσματος. Εμείς όμως μπορεί να θέλουμε κατασκευάσουμε μια δική μας συνάρτηση που να μας δίνει το πραγματικό εύρος (δηλαδή τις διαφορές της μέγιστης και της ελάχιστης τιμής) κατευθείαν. Το να φτιάξουμε μια τέτοια συνάρτηση, είναι σχετικά απλό και αρκούν οι επόμενες εντολές:

```
> Range<-function(x){
+         z<-max(x)-min(x)
+         return(z)
+ }
> y<-rnorm(100,0,1)
> Range(y)
[1] 4.406891
```
■.

Παρατήρηση: Σε αντίθεση με άλλες γλώσσες προγραμματισμούς στην R δεν χρειάζεται να δηλώσουμε τον τύπο του ορίσματος μιας συνάρτησης.

7.2 Βασικά χαρακτηριστικά

Τα βασικά χαρακτηριστικά της συνάρτησης που γράψαμε είναι τα εξής:

- Το όνομα της συνάρτησης είναι προσωπική επιλογή του κάθε χρήστη. Στο προηγούμενο εισαγωγικό παράδειαγμα, η συνάρτηση ονομάστηκε `Range`. Άλλα ονόματα, όπως `R`, `evros` ή ακόμα και `range`, είναι εξίσου αποδεκτά. Στη τελευταία περίπτωση, θα γίνει προσωρινή αντικατάσταση της συνάρτησης `range` της R από αυτή που ορίσαμε εμείς. Θα μπορούμε να χρησιμοποιήσουμε τη συνάρτηση μας χωρίς πρόβλημα άλλα όχι την προκαθορισμένη της R. Για να γίνει εφικτή ξανά η χρήση της προκαθορισμένης εντολής της R, θα πρέπει να απομακρύνουμε τη δικιά μας συνάρτηση χρησιμοποιώντας την εντολή `rm(range)`. Καλό είναι να αποφεύγουμε ονόματα που ήδη χρησιμοποιούνται στην R, είτε ως ονόματα συναρτήσεων, είτε ως αντικείμενα γιατί μπορεί να δημιουργήσουν σύγχιση και

προβλήματα ειδικά αν ξεχαστούν στο χώρο εργασίας και προσπαθήσουμε να ανακαλέσουμε αυτές τις εντολές μετά από αρκετό χρόνο. Γενικά, το όνομα μιας συνάρτησης θα πρέπει είναι σύντομο αλλά να μας δίνει κάποια πληροφορία για τη χρησιμότητα της συνάρτησης. Σε συνδυασμό με το πληροφοριακό όνομα της συνάρτησης, χρήσιμο είναι να υπάρχει στην αρχή της συνάρτησης κάποιο σχόλιο σχετικά με το τι ακριβώς κάνει η συνάρτηση.

- Μετά το όνομα, ακολουθεί ο χαρακτήρας εκχώρησης «<- » και η εντολή `function`. Στην ουσία, συνδέουμε το όνομα που είχαμε δώσει πριν με μια συνάρτηση.

- Ακολουθούν μέσα σε παρένθεση οι μεταβλητές που θα χρησιμοποιηθούν μέσα στη συνάρτηση, δηλαδή οι μεταβλητές εισόδου. Πρέπει να λάβετε υπόψη σας, πως αυτές είναι τοπικές μεταβλητές, δηλαδή αφορούν μόνο τη συγκεκριμένη συνάρτηση και, επομένως, δεν έχουν φυσική αντιστοιχία με κάποια άλλη μεταβλητή που έχετε με το ίδιο όνομα. Εδώ, χρησιμοποιούμε ως τοπική μεταβλητή το διάνυσμα `x`, που αφορά οποιοδήποτε διάνυσμα μας ενδιαφέρει. Παρατηρείστε πως μετά χρησιμοποιούμε τη συνάρτηση με όρισμα το `y`. Όταν αργότερα καλέσουμε τη συνάρτηση, θα πρέπει να χρησιμοποιήσουμε στη θέση αυτών των μεταβλητών, κάποιες μεταβλητές που ήδη υπάρχουν στην R, οι οποίες, όμως, να είναι του ίδιου τύπου. Για παράδειγμα, το `x` που ορίσαμε πριν, είναι διάνυσμα και, επομένως, όταν καλέσουμε τη συνάρτηση, πρέπει σε εκείνη τη θέση να υπάρχει ένα διάνυσμα.

- Αφού ορίσουμε τις μεταβλητές, που μπορεί να είναι περισσότερες από μια και αφού κλείσει η παρένθεση, ακολουθούν μέσα σε άγκιστρα οι εντολές που αποτελούν την συνάρτηση. Μέσα σε αυτές τις εντολές, μπορεί να χρησιμοποιηθούν διάφορες μεταβλητές, αλλά καλό είναι να δίνουμε αρχικές τιμές στις μεταβλητές που θα χρησιμοποιήσουμε (στο παράδειγμα δεν έχουμε τέτοιες μεταβλητές), για να αποφύγουμε πλάγια αποτελέσματα, δηλαδή κάποιες από τις μεταβλητές που δεν είναι τοπικές και τις χρησιμοποιούμε και σε άλλες συναρτήσεις να αλλάξουν τιμή.

- Τέλος, θα πρέπει το αποτέλεσμα που μας ενδιαφέρει, να επιστρέφεται με κάποιον τρόπο, ο οποίος είναι συνήθως, είτε μια εντολή `list` για αποτέλεσμα που αποτελείται από πολλά αντικείμενα, είτε για την περίπτωση ενός μόνο αντικειμένου, το ίδιο αντικείμενο που θέλουμε να κρατήσουμε ως αποτέλεσμα, να εμφανίζεται στην τελευταία γραμμή, πριν το άγκιστρο που ολοκληρώνεται η συνάρτηση. Το αποτέλεσμα μπορεί να είναι οποιοδήποτε αντικείμενο, δηλαδή μπορεί να είναι αριθμός, διάνυσμα, λίστα κλπ. Μπορούμε να ορίσουμε συναρτήσεις, που απλά κάνουν κάτι (π.χ. ένα διάγραμμα) και δεν επιστρέφουν κανένα αποτέλεσμα.

Είναι βασικό να θυμόμαστε πως, όταν καλούμε μια συνάρτηση, πρέπει να υπάρχει μια πλήρης αντιστοιχία των μεταβλητών εισόδου που εμφανίζονται μέσα στην παρένθεση, μετά τη λέξη `function` και των μεταβλητών που χρησιμοποιούμε για να καλέσουμε τη συνάρτηση. Για παράδειγμα, έστω μια συνάρτηση που η πρώτη της γραμμή είναι η εξής:

```
> testfunc<-function(x,n) {
+ ...
+ ...
+ }                                                    ■.
```

και η οποία παίρνει σαν είσοδο ένα διάνυσμα `x` και έναν αριθμό `n`. Τότε, όταν καλέσουμε τη συνάρτηση, έστω με μια εντολή `testfunc(y,nnew)`, θα πρέπει το `y` να είναι διάνυσμα και το `nnew` να είναι αριθμός. Φυσικά, αν καλέσουμε τη συνάρτηση ως `testfunc(nnew,y)` ή `testfunc(y)`, θα πάρουμε μήνυμα λάθους. Για την ακρίβεια, η R ενδέχεται να μη μας δώσει μήνυμα λάθους, αλλά να χρησιμοποιήσει λανθασμένα τη συνάρτηση. Και αυτό, γιατί, το μήνυμα λάθους θα εμφανιστεί, μόνο, όταν μέσα στη συνάρτηση προσπαθήσουμε να κάνουμε κάποια πράξη που δεν είναι σωστή. Για παράδειγμα, επειδή ένας αριθμός είναι στην ουσία διάνυσμα μήκους 1, στο παράδειγμα μας η συνάρτηση θα εκτελεστεί χρησιμοποιώντας τον αριθμό `nnew` ως διάνυσμα μήκους 1. Επειδή θα βρει το διάνυσμα `y` στη θέση ενός αριθμού, θα πάρουμε μήνυμα λάθους, μόνο, αν εκτελείται μέσα στη συνάρτηση κάποια μη επιτρεπτή για διανύσματα πράξη (π.χ. κάποια σύγκριση). Το παραπάνω σχόλιο είναι σημαντικό, γιατί ενδέχεται να μην πάρουμε μήνυμα λάθους, αλλά απλά λανθασμένα αποτελέσματα.

Επειδή οι συναρτήσεις, συνήθως, αποτελούνται από αρκετές εντολές, δίνουμε τη συμβουλή να χρησιμοποιείτε κάποιον απλό επεξεργαστή κειμένου για τη συγγραφή τους και να μεταφέρετε τις εντολές στην R με απλή αντιγραφή και επικόλληση (copy-paste). Εναλλακτικά, μπορεί κανείς μόνο στην R, να χρησιμοποιήσει το παράθυρο `Script`. Για να ανοίξετε ένα τέτοιο παράθυρο, πηγαίνετε `File > New > Script`. Εκεί, λειτουργώντας σαν επεξεργαστής κειμένου, μπορεί κανείς να γράψει όσες εντολές θέλει. Για να εκτελεστούν οι εντολές που επιθυμεί, χρειάζεται απλά να τις μαρκάρει και στη συνέχεια να πατήσει F10. Τα αποτελέσματα και τα μηνύματα λάθους θα εμφανιστούν στο κάτω μέρος του παραθύρου που ανοίξατε.

7.3 Τοπικές και καθολικές μεταβλητές/αντικείμενα

Πριν δούμε και συζητήσουμε για την κατασκευή συναρτήσεων, θα πρέπει να ξεχωρίσουμε τις εξής κατηγορίες μεταβλητών:

- Τοπικές μεταβλητές (local variables), οι οποίες ορίζονται μέσα σε μια συνάρτηση και ισχύουν μόνο μέσα στη συνάρτηση (για την ακρίβεια καταλαμβάνουν χώρο στη μνήμη του υπολογιστή, μόνο, όταν εκτελείται η συνάρτηση), ενώ οι τιμές τους παύουν να ισχύουν μετά.

- Καθολικές μεταβλητές (global variables), οι οποίες αφορούν μεταβλητές κάθε είδους και ισχύουν γενικά και όχι μόνο μέσα στη συνάρτηση που τις χρησιμοποιεί.

Η διαφορά ανάμεσα σε αυτές τις δύο κατηγορίες μεταβλητών είναι πολύ σημαντική και ουσιαστική για το σωστό προγραμματισμό. Μερικά πολύ σημαντικά θέματα, που αφορούν τις μεταβλητές αυτές, είναι τα εξής:

- Μπορούμε να χρησιμοποιήσουμε, ως ονόματα τοπικών μεταβλητών, ονόματα που έχουμε δώσει και σε καθολικές μεταβλητές. Θα πρέπει να έχουμε, βέβαια, στο μυαλό μας πως, ουσιαστικά, πρόκειται για άλλες μεταβλητές. Όταν τελειώσει η συνάρτηση, οι καθολικές μεταβλητές θα έχουν κρατήσει τις τιμές τους, ενώ οι τοπικές θα χαθούν.

- Είναι αρκετά επικίνδυνο, για την παραγωγή λανθασμένων αποτελεσμάτων, να χρησιμοποιούμε μέσα σε μια συνάρτηση καθολικές μεταβλητές. Αν, μέσα στη συνάρτηση για κάποια μεταβλητή, δεν δώσουμε τιμές με κάποια εντολή εκχώρησης, η R χρησιμοποιεί την τιμή της καθολικής μεταβλητής και έτσι μπορεί να πάρουμε περίεργα αποτελέσματα (χωρίς, βέβαια, καμιά προειδοποίηση γι'αυτό). Συνεπώς, χρειάζεται μεγάλη προσοχή, ώστε όλες οι μεταβλητές που χρησιμοποιούνται μέσα στις συναρτήσεις, να τους δίνονται αρχικές τιμές, είτε να είναι μεταβλητές εισόδου, που σημαίνει πως παίρνουν αρχικές τιμές αυτόματα. Επομένως, μια βασική αρχή στο στάδιο του προγραμματισμού συναρτήσεων είναι να αποφεύγεται η χρήση καθολικών μεταβλητών μέσα σε συναρτήσεις, εκτός αν είναι αναγκαίο να υπάρχουν σαν παράμετροι εισόδου. Αυτό εξασφαλίζει, αφενός την αυτονομία των συναρτήσεων και αφετέρου περιορίζει την πιθανότητα πλάγιων αποτελεσμάτων. Θα δούμε σε λίγο κάποια παραδείγματα.

7.4 Επιστροφή αποτελεσμάτων

Αναφέρθηκε, προηγουμένως, πως μια συνάρτηση πρέπει, με την τελευταία της εντολή, να επιστρέφει τα αποτελέσματα. Θα δούμε αργότερα συναρτήσεις, όπου δεν μας ενδιαφέρουν τα αποτελέσματα, αλλά, για παράδειγμα, η κατασκευή ενός διαγράμματος. Ας δούμε παραδείγματα για το πώς μπορούμε να επιστρέψουμε τα αποτελέσματα που μας ενδιαφέρουν.

Ας δούμε την παρακάτω συνάρτηση. Πρόκειται για μια συνάρτηση που δέχεται σαν δεδομένα ένα διάνυσμα και επιστρέφει τη μέση τιμή των παρατηρήσεων, που βρίσκονται εντός 2 τυπικών αποκλίσεων από τη μέση τιμή, υπολογίζει δηλαδή μια μορφή περικομμένου μέσου. Η συνάρτηση είναι η εξής:

```
> trimmedmean<-function(x) {
+              mesitimi<-mean(x)
+              n<-length(x)
+              std<-sd(x) #or std<-sqrt(var(x))
+              sum1<-0
+              metr<-0
+              for (i in 1:n) {
+                   if (abs((x[i]-mesitimi)/std) <=2) {
+                       sum1<-sum1+x[i]
+                       metr<-metr+1
+                   }
+              } # end of for
```

```
+                 sum1/metr
+ }                                                                    ■.
```

Μέσα στη συνάρτηση αυτή χρησιμοποιούμε τόσο εντολή `if`, όσο και εντολή `for`. Η τελευταία εντολή, πριν το άγκιστρο, μας επιστρέφει το αποτέλεσμα που θέλουμε. Έχοντας αυτή τη συνάρτηση, μπορούμε για οποιοδήποτε διάνυσμα να υπολογίζουμε αυτή τη μέση τιμή, όπως στα παραδείγματα που ακολουθούν:

```
> x<-rnorm(1000,0,1)
> trimmedmean(x)
[1] -0.03998702
> y<-rgamma(150,1,1)
> trimmedmean(y)
[1] 0.8579935                                                        ■.
```

Στην παραπάνω σύνταξη τα αντικείμενα `x` και `y` είναι διανύσματα με 1000 και 150 τυχαίες τιμές από την κανονική και τη γάμμα κατανομή αντίστοιχά• βλ. για περισσότερες λεπτομέρειες στην Ενότητα 8.2.

Είναι βασικό να υπενθυμίσουμε, πως μια συνάρτηση που γράψαμε εμείς, μπορεί να χρησιμοποιηθεί όπως κάθε συνάρτηση που έχει η R. Έτσι, όπως στο παραπάνω παράδειγμα, μπορεί να εμφανιστεί στο δεξί μέλος εντολής εκχώρησης, αλλά και για να δημιουργήσουμε περίπλοκες παραστάσεις με αυτήν, όπως στα παραδείγματα που ακολουθούν:

```
> test<-5*trimmedmean(x) - trimmedmean(x[12:100])+ mean(x)
> test
[1] 0.2227619
> if (abs(trimmedmean(x) - mean(x))<0.01) {
+         test<-mean(x)
+ }else {
+         test<-trimmedmean(x)
+ }
> test
[1] 0.02705253
> trimmedmean(c(rnorm(100),3,3,2,seq(-1,1,length=20)))
[1] 0.0455615                                                        ■.
```

Στα παραδείγματα αυτά η συνάρτηση χρησιμοποιήθηκε, σε συνδυασμό με άλλες συναρτήσεις, ως μέλος μιας άλλης συνάρτησης (`abs`), αλλά και ως παράσταση σε εντολή `if`. Επίσης, παρατηρείστε πώς χρησιμοποιήσαμε μέρος από τα δεδομένα για να ορίσουμε ένα διάνυσμα και να το χρησιμοποιήσουμε ως μεταβλητή εισόδους στη συνάρτηση που ορίσαμε.

Η παραπάνω συνάρτηση επιστρέφει μόνο την τιμή του περικομμένου μέσου. Μερικές φορές, όμως, ενδιαφερόμαστε για περισσότερα από ένα αποτελέσματα. Στο συγκεκριμένο παράδειγμα μας ενδιαφέρει, επίσης, και το πόσες παρατηρήσεις έχουν ληφθεί υπόψη. Για να επιτευχθεί αυτό,

ένας εύκολος τρόπος, είναι να ορίσουμε μια λίστα που περιέχει τα αποτελέσματα που θέλουμε να επιστραφούν από τη συνάρτηση. Για να το επιτύχουμε αυτό, θα πρέπει να κάνουμε κάποιες λίγες αλλαγές στη συνάρτηση μας. Κατ' αρχάς, θα πρέπει να μετρήσουμε το πλήθος αυτών των τιμών, που ουσιαστικά, αφορούν τη μεταβλητή `metr` που έχουμε μέσα στη συνάρτηση, και άρα, η μόνη αλλαγή είναι στον τρόπο που εξάγουμε τα αποτελέσματα. Η συνάρτηση παίρνει την εξής μορφή:

```
> trimmedmean<-function(x) {
+    mesitimi<-mean(x)
+    n<-length(x)
+    std<-sd(x) #or std<-sqrt(var(x))
+    sum1<-0
+    metr<-0
+    for (i in 1:n) {
+         if (abs((x[i]-mesitimi)/std) <=2) {
+             sum1<-sum1+x[i]
+             metr<-metr+1
+         }
+     }# end of for
+     list(result=sum1/metr,numberofdata=metr)
+ }                                                                    ■.
```

Αν τώρα χρησιμοποιήσουμε τα ίδια δεδομένα με πριν, το αποτέλεσμα που θα πάρουμε είναι το εξής:

```
> trimmedmean(x)
$result
[1] 0.02747271

$numberofdata
[1] 953

> trimmedmean(y)
$result
[1] 0.904833

$numberofdata
[1] 144                                                              ■.
```

Παρατηρείστε, πως η R χρησιμοποιεί, για να ονομάσει τα αποτελέσματα, τα ονόματα που δώσαμε μέσα από την εντολή `list`. Είναι πολύ ενδιαφέρον, πώς μπορούμε να αποθηκεύσουμε τα αποτελέσματα μιας συνάρτησης σε μια μεταβλητή, η οποία αυτόματα παίρνει τον τύπο του

αποτελέσματος. Στην περίπτωση μας, λοιπόν, δημιουργείται μια λίστα και μπορούμε να αναφερθούμε στα αποτελέσματα χρησιμοποιώντας το σύμβολο $ και το όνομα του μέλους της λίστας που μας ενδιαφέρει. Για παράδειγμα:

```
> xres<-trimmedmean(x)
> xres$numberofdata
[1] 953
> xres$result-mean(x)
[1] 0.004946959
> mean(x)
[1] 0.02252575                                                    ■.
```

Παρατηρείστε, πως χρησιμοποιούμε τη μεταβλητή `xres$result` κανονικά, σαν να πρόκειται για κάποια μεταβλητή που εμείς ορίσαμε.

Παρατήρηση: εναλλακτικά θα μπορούσαμε να επιστρέψουμε περισσότερα από ένα αποτελέσματα, βάζοντάς τα μέσα σε ένα διάνυσμα. Δηλαδή, αντικαθιστώντας την εντολή `list` με μια εντολή της μορφής:

```
                              c(sum1/metr,metr)                   ■,
```

η οποία δημιουργεί ένα διάνυσμα μεγέθους 2 με τα αποτελέσματα που θέλουμε. Κάτι τέτοιο, όμως, δεν είναι εύχρηστο για πολλούς λόγους. Ο βασικότερος είναι πως δεν είναι εύκολο να θυμόμαστε τη σειρά που παίρνουμε τα αποτελέσματα, ειδικά, αν αυτά είναι περισσότερα από δύο, όπως στην περίπτωση μας. Επίσης, στο παράδειγμά μας έχουμε ως αποτελέσματα δύο αριθμούς οι οποίοι μπορούν να αποθηκευτούν εύκολα σε ένα διάνυσμα (μήκους δύο στοιχείων). Αν, όμως, το ένα από τα αποτελέσματα ήταν ένας ολόκληρος πίνακας, τότε τα αποτελέσματα δε θα μπορούσαμε να τα αποθηκεύσουμε σε ένα διάνυσμα (η ακόμα και σε πίνακα) και, συνεπώς, η χρήση της λίστας είναι η μόνη μας επιλογή.

7.5 Προκαθορισμένες τιμές

Ας υποθέσουμε, πως θέλουμε να κάνουμε πιο ευέλικτη τη συνάρτησή μας και θέλουμε ο χρήστης να ορίζει το μέγεθος του διαστήματος που τον ενδιαφέρει, δηλαδή αντί για δύο τυπικές αποκλίσεις, να μπορεί να ορίζει πόσες τυπικές αποκλίσεις θέλει. Για να το επιτύχουμε αυτό, μπορούμε να χρησιμοποιήσουμε αυτή την πληροφορία (πόσες τυπικές αποκλίσεις) ως όρισμα στη συνάρτηση μας. Η συνάρτηση τώρα γίνεται:

```
trimmedmean<-function(x,nst) {
    mesitimi<-mean(x)
    n<-length(x)
    std<-sd(x) #or std<-sqrt(var(x))
    sum1<-0
    metr<-0
```

```
    for (i in 1:n) {
         if (abs((x[i]-mesitimi)/std) <=nst){
             sum1<-sum1+x[i]
             metr<-metr+1
         }
    } # end of for
    list(result=sum1/metr,numberofdata=metr)
}                                                                    ■.
```

Στην επικεφαλίδα υπάρχουν δύο ορίσματα, ενώ τώρα καλούμε τη συνάρτηση ως

```
> trimmedmean(x,1.5)
$result
[1] -0.0004802955

$numberofdata
[1] 864                                                              ■.
```

Τι θα συμβεί όμως, αν δεν δώσουμε τις μεταβλητές εισόδου σωστά; Αν, δηλαδή, εκεί που πρέπει να δώσουμε το διάνυσμα, ορίσουμε έναν αριθμό και το ανάποδο. Ας δούμε τις παρακάτω εντολές:

```
> data<-c(-1.529, 0.148, -2.302, 0.319, -0.361, -1.204, -0.601,
+ -0.72,  -0.726, 0.916, -1.199, 1.913, 2.299, 0.641, -2.168,
+ 1.423, -0.455, -1.673, 0.89, 0.008, 0.627, 0.335, -0.544, -1.573,
+ 0.733, 0.908, -0.575, -0.673, -0.295, 0.93)
> trimmedmean(data,1.5)
$result
[1] -0.1634615

$numberofdata
[1] 26

$result
[1] -0.1502667

$numberofdata
[1] 30

> trimmedmean(2.5,data)
Error in if (abs((x[i] - mesitimi)/std) <= nst) { :
  missing value where TRUE/FALSE needed
```

```
In addition: Warning message:
In if (abs((x[i] - mesitimi)/std) <= nst) { :
  the condition has length > 1 and only the first element will be
   used                                                          ■.
   ■.
```

Κατ' αρχάς, για τα δεδομένα που είχαμε στο διάνυσμα data, υπολογίσαμε τον περικομμένο μέσο για 1.5 και 2.5 τυπικές αποκλίσεις. Στην τελευταία κλήση της συνάρτησης αλλάξαμε τη σειρά των μεταβλητών εισόδου. Έτσι, ο υπολογιστής καταλαβαίνει πως τα δεδομένα είναι στο διάνυσμα μεγέθους 1 με τιμή 2.5, ενώ το πόσες τυπικές αποκλίσεις θα πρέπει να λάβει υπόψη του περιλαμβάνεται στο διάνυσμα data. Ευτυχώς, στο παράδειγμα αυτό, παίρνουμε μήνυμα λάθους και δεν εκτελείται η εντολή. Όμως, πώς ακριβώς δουλεύει η R σε αυτή την περίπτωση; Η R ταιριάζει τα αντικείμενα με τις μεταβλητές εισόδου και προσπαθεί να εκτελέσει κανονικά τη συνάρτηση. Έτσι, έχουμε ένα διάνυσμα μεγέθους ένα για δεδομένα και πολλές τιμές που περιλαμβάνονται σε ένα διάνυσμα για τον αριθμό των τυπικών αποκλίσεων. Η συνάρτηση μάς βγάζει μήνυμα λάθους, απλά, γιατί η τυπική απόκλιση που υπολογίζει για το διάνυσμα μεγέθους ένα είναι μηδέν και δεν μπορεί να κάνει διαίρεση με το μηδέν.

Για να γίνει πιο κατανοητός ο τρόπος που λειτουργεί η R, δείτε το παράδειγμα που ακολουθεί:

```
> trimmedmean(data, c(1.4,2))
$result
[1] -0.1634615

$numberofdata
[1] 26

There were 30 warnings (use warnings() to see them)


> warnings()
Warning messages:
1: In if (abs((x[i] - mesitimi)/std) <= nst) { ... :
  the condition has length > 1 and only the first element will be
   used
.....................................................................

30: In if (abs((x[i] - mesitimi)/std) <= nst) { ... :
  the condition has length > 1 and only the first element will be
   used
                                                                 ■.
```

Στο παραπάνω παράδειγμα, καλούμε τη συνάρτηση με ένα διάνυσμα, `c(1.4,2)`, που να υποδεικνύει πόσες τυπικές αποκλίσεις θα χρησιμοποιήσουμε. Η συνάρτηση εκτελείται κανονικά και παίρνουμε αποτέλεσμα. Όμως, για ποιο στοιχείο του διανύσματος; Δείτε, πως υπάρχουν και μερικές προειδοποιήσεις (warnings). Για να τις δούμε, αρκεί να πληκτρολογήσουμε την εντολή `warnings()`. Εκεί βλέπουμε, πως έχει χρησιμοποιηθεί μόνο το πρώτο στοιχείο του διανύσματος το 1.4.

Συνεπώς, θα πρέπει να είμαστε αρκετά προσεκτικοί, ώστε στην κλήση μιας συνάρτησης να χρησιμοποιούμε με ακρίβεια τις μεταβλητές εισόδου, με στόχο να εφοδιάσουμε τη συνάρτηση με σωστά δεδομένα για να εκτελεστεί, αλλιώς τα αποτελέσματα μπορεί να μην είναι σωστά.

Τελειώνοντας αυτή την παράγραφο, θα πρέπει να πούμε πως αν οι μεταβλητές εισόδου είναι λιγότερες ή περισσότερες από αυτές που πρέπει, η συνάρτηση δεν θα εκτελεστεί και θα πάρουμε μηνύματα λάθους, όπως αυτά που ακολουθούν:

```
> trimmedmean(data, )
Error: argument "nst" is missing, with no default
> trimmedmean(data,1.2,3 )
Error in trimmedmean(data, 1.2, 3) : unused argument (3)          ■.
```

Υπάρχει, όμως, η δυνατότητα να το αποφύγουμε αυτό. Μπορούμε στην R να ορίσουμε τιμές για κάποιες από τις μεταβλητές εισόδου, οι οποίες θα ισχύουν στην περίπτωση που κατά την κλήση τους δεν ορίσουμε εμείς την τιμή αυτή. Αυτές οι τιμές ονομάζονται προκαθορισμένες (default). Για παράδειγμα, στη συνάρτηση που χρησιμοποιήσαμε θα μπορούσαμε να αλλάξουμε μόνο την πρώτη γραμμή ως εξής:

```
> trimmedmean<-function(x,nst=2) {                                  ■.
```

και η υπόλοιπη συνάρτηση μένει ως έχει. Αυτό που προσθέσαμε είναι το `nst=2`, που πριν δεν είχε τιμή. Αυτό που καταλαβαίνει η R είναι, πως η προκαθορισμένη τιμή είναι 2 για το πλήθος των τυπικών αποκλίσεων και, επομένως, αν καλέσουμε τη συνάρτηση χωρίς τιμή εισόδου, θα χρησιμοποιήσει αυτή την τιμή. Για παράδειγμα:

```
> trimmedmean(data,2)
$result
[1] -0.2347241

$numberofdata
[1] 29

> trimmedmean(data)
$result
[1] -0.2347241

$numberofdata
```

```
[1] 29                                                    ■.
```

Παρατηρήσεις

Αν έχουμε πολλές μεταβλητές εισόδου, προφανώς, μπορούμε να ορίσουμε προκαθορισμένες τιμές για όλες αυτές. Αν κατά την κλήση, θέλουμε να χρησιμοποιήσουμε τις προκαθορισμένες αυτές τιμές, αρκεί να αφήσουμε κενό το πεδίο που τούς αντιστοιχεί. Αν για παράδειγμα, η επικεφαλίδα μιας συνάρτησης είναι η

```
Prokathor<-function (a=2,b=1,c=12) {
 ......
```

τότε μπορούμε να καλέσουμε τη συνάρτηση αυτή ως:

```
Prokathor(a, ,)
Prokathor(a,b)
Prokathor(,,c)
Prokathor()
Prokathor(a=5,b)                                          ■.
```

7.6 Πλάγια αποτελέσματα

Ας δούμε λίγο τα προβλήματα που δημιουργούνται, όταν μπερδέψουμε τοπικές και καθολικές μεταβλητές κατά τη δημιουργία συναρτήσεων. Για τα παραδείγματα, θα χρησιμοποιήσουμε διάφορες συναρτήσεις που σκοπό έχουν να υπολογίζουν την μέση απόλυτη απόκλιση των δεδομένων, δηλαδή τη συνάρτηση:

$$MAD = \frac{\sum_{i=1}^{n} |X_i - \bar{X}|}{n} .$$

΄Εστω οι επόμενες εντολές:

```
> data <- rnorm(1000)
> m<- 74
> m
[1] 74
> madstat<-function(x) {
+ n<-length(x)
+ m<-m+1
+ mad<-sum(abs(x-mean(x)))/n
+ list(MAD= mad, valueofm = m)
+ }
> madstat(data)
$MAD
[1] 0.8058891
```

```
$valueofm
[1] 75

> m
[1] 74                                                    ■.
```

Η βασική ιδέα είναι, πως η μεταβλητή / αντικείμενο `m` υπάρχει, τόσο ως καθολική μεταβλητή, όσο και ως τοπική μεταβλητή μέσα στη συνάρτηση. Ας δούμε, όμως, και δύο λόγια για τη συνάρτηση. Κατ' αρχάς, η μεταβλητή εισόδου είναι το διάνυσμα με τα δεδομένα. Τόσο η μέση τιμή, όσο και το πλήθος των παρατηρήσεων που χρειαζόμαστε, είναι καλό να υπολογίζονται μέσα στη συνάρτηση και να μην είναι μεταβλητές εισόδου, ώστε να δίνουν στη συνάρτηση μεγαλύτερη ευελιξία και αυτονομία. Παρατηρείστε, επίσης, πως στην εντολή `m<-m+1` η μεταβλητή `m` εμφανίζεται στο δεξί μέλος της εντολής εκχώρησης. Δεδομένου πως μέσα στη συνάρτηση δεν έχει πάρει πουθενά τιμή, η R θα χρησιμοποιήσει την τιμή από την καθολική μεταβλητή. Για να το δούμε αυτό, ζητάμε, όχι μόνο την τιμή της μέσης απόλυτης απόκλισης ως αποτέλεσμα, αλλά και την τιμή του `m` ως αποτέλεσμα της συνάρτησης, δηλαδή ζητάμε την τιμή της τοπικής μεταβλητής.

```
> madstat(data)
$MAD
[1] 0.8058891

$valueofm
[1] 75

> m
[1] 74                                                    ■.
```

Τα αποτελέσματα δείχνουν πως η καθολική μεταβλητή κράτησε την τιμή της, δηλαδή παρ' όλο που ορίσαμε μια τοπική μεταβλητή με το ίδιο όνομα, αυτή δεν έχασε την τιμή της (σε μερικές γλώσσες προγραμματισμού αυτό δεν ισχύει). Δείτε, όμως, την τιμή της μέσα στη συνάρτηση πήρε τιμή από την ομώνυμη καθολική μεταβλητή. Η προειδοποίηση είναι, πως, αν σε κάποια τοπική μεταβλητή δώσουμε τιμή από κάποια μεταβλητή, που δεν έχουμε εμείς ορίσει, υπάρχει ο κίνδυνος να πάρει τιμές από μεταβλητές ξεχασμένες στην R από άλλους ή άλλες εργασίες και έτσι να πάρουμε λανθασμένα αποτελέσματα.

Ας δούμε το επόμενο παράδειγμα. Η βασική αλλαγή από την προηγούμενη συνάρτηση είναι πως το πλήθος των παρατηρήσεων ξεχνάμε να το υπολογίσουμε. Για κακή μας τύχη, υπάρχει μια άλλη μεταβλητή με το ίδιο όνομα την οποία χρησιμοποιεί η R στη θέση της. Επομένως, αυτή η συνάρτηση για οποιοδήποτε μέγεθος δείγματος θα διαιρεί με 100. Η συνάρτηση `madstat3` είναι η σωστή συνάρτηση.

```
> n<-100
```

```
> madstat2<-function(x) {
+           sum(abs(x-mean(x)))/n
+ }
> madstat3<-function(x) {
+           n<-length(x)
+           sum(abs(x-mean(x)))/n
+ }
>  madstat2(x)
[1] 7.806224
>  madstat3(x)
[1] 0.7806224                                                    ■.
```

Συνεπώς, θα πρέπει να είμαστε πολύ προσεκτικοί στο να μην μπερδεύουμε τοπικές με καθολικές μεταβλητές, καθώς υπάρχει περίπτωση να πάρουμε λάθος αποτελέσματα, ενώ όλα δείχνουν ότι έχουν εκτελεστεί σωστά. Μια σωστή πρακτική είναι, να εξασφαλίζουμε ότι όλες οι μεταβλητές που χρησιμοποιούμε μέσα σε μια συνάρτηση, έχουν οριστεί ως τοπικές μεταβλητές και, επομένως, ακόμα και καθολική μεταβλητή με το ίδιο όνομα να υπάρχει δεν θα μάς επηρεάσει, ή ως μεταβλητές εισόδου.

7.7 Ορισμός συναρτήσεων μέσα σε συναρτήσεις

Η R μάς δίνει τη δυνατότητα να ορίζουμε συναρτήσεις μέσα σε άλλες συναρτήσεις.

Για παράδειγμα, δείτε την επόμενη συνάρτηση:

```
> madstat4<-function(x) {
+           n<-length(x)
+           mestim<-function(x) {
+                   sum(x)/length(x)
+           }
+           sum(abs(x-mestim(x)))/n
+ }                                                               ■.
```

Μέσα στη συνάρτηση, που υπολογίζει τη μέση απόλυτη απόκλιση, έχουμε γράψει τη δική μας συνάρτηση για να υπολογίζει τη μέση τιμή. Κάτι τέτοιο, δεν απαγορεύεται, αλλά είναι συζητήσιμο, αν και κατά πόσο κάτι τέτοιο είναι χρήσιμο και προτιμότερο, από το να γράψουμε τη δεύτερη συνάρτηση έξω από την κύρια συνάρτηση. Ο μόνος λόγος, για να έχουμε συναρτήσεις μέσα σε συναρτήσεις, είναι όταν αυτές είναι αρκετά μεγάλες και δεν θέλουμε να χρησιμοποιούν τη μνήμη του υπολογιστή. Συναρτήσεις που ορίζονται μέσα σε συναρτήσεις, είναι διαθέσιμες, μόνο κατά τη διάρκεια εκτέλεσης της κύριας συνάρτησης και σίγουρα, όχι, μετά το τέλος της συνάρτησης. Δείτε το επόμενο παράδειγμα, όπου προσπαθούμε να καλέσουμε τη συνάρτηση `mestim` που ορίσαμε προηγουμένως μέσα στη συνάρτηση. Η R δεν αναγνωρίζει την ύπαρξη αυτής της συνάρτησης.

```
> data<-rnorm(1000)
```

```
> madstat4(data)
[1] 0.849113
> mestim(data)
Error: could not find function "mestim"                    ■.
```

Φυσικά, συναρτήσεις που ορίζονται διαδοχικά, είναι αναγνωρίσιμες με τη σειρά που αυτές ορίζονται. Αν, δηλαδή ορίσουμε τις παρακάτω συναρτήσεις:

```
mestim5<-function(x) {
         sum(x)/length(x)
}
madstat5<-function(x) {
          n<-length(x)
          sum(abs(x-mestim5(x)))/n
}                                                          ■,
```

τότε η συνάρτηση `mestim5` είναι διαθέσιμη για την επόμενη συνάρτηση που ακολουθεί, αλλά και γενικότερα για κάθε άλλη συνάρτηση, σε αντίθεση με προηγουμένως.

Αυτό, που επίσης είναι σημαντικό να θυμόμαστε, είναι πως η R αναγνωρίζει τις συναρτήσεις αυτές από τη στιγμή που θα δηλωθούν και μετά. Κάτι το οποίο σημαίνει, πως μπορούμε να ορίσουμε συναρτήσεις σε διάφορα σημεία του προγράμματος. Έτσι, η παρακάτω δήλωση συναρτήσεων είναι καθόλα αποδεκτή, αφού τη στιγμή που τελικά καλούμε τη συνάρτηση `madstat6` έχει οριστεί και η συνάρτηση που αυτή καλεί.

```
> madstat6<-function(x) {
+           n<-length(x)
+           sum(abs(x-mestim6(x)))/n
+ }
> mestim6<-function(x) {
+          sum(x)/length(x)
+ }
> data<-rnorm(1200)
> madstat6(data)
[1] 0.7861857                                              ■.
```

7.8 Διάφορα άλλα θέματα

Είναι προφανές, ότι μπορούμε να εξειδικεύουμε ολοένα και περισσότερο στις συναρτήσεις μας. Επειδή ο σκοπός των σημειώσεων αυτών είναι καθαρά εισαγωγικός, δεν θα αναφέρουμε περισσότερα σχετικά με τις συναρτήσεις. Μερικά, όμως, πράγματα που πρέπει να έχετε κατά νου είναι τα εξής:

Μέσα στις συναρτήσεις μπορούμε να έχουμε διαγράμματα, ακόμα και το αποτέλεσμα μπορεί να είναι απλά ένα διάγραμμα. Για παράδειγμα η επόμενη συνάρτηση

```
> plotnormal<-function(m=0,s=1) {
+              xkato<-qnorm(0.001,m,s)
+              xano<-qnorm(0.999,m,s)
+              x<-seq(xkato,xano,length=1000)
+              plot(x,dnorm(x,m,s),ylab="density",type="l")
+ }
```

απλά, δημιουργεί το διάγραμμα της συνάρτησης πυκνότητας πιθανότητας μιας κανονικής κατανομής, με μέση τιμή `m` και τυπική απόκλιση `s`. Οι προκαθορισμένες τιμές της συνάρτησης είναι μηδέν και ένα, αντίστοιχα. Παρατηρείστε, πως δεν υπάρχει κάποια εντολή που να επιστρέφει κάποιο αποτέλεσμα, η συνάρτηση απλά δημιουργεί το διάγραμμα και τίποτα άλλο. Επίσης, παρατηρείστε πώς κατασκευάζει το διάγραμμα. Προκειμένου να βρούμε τα δύο άκρα του διαστήματος, που θα χρησιμοποιηθούν για το διάγραμμα, παίρνουμε με την συνάρτηση της R `qnorm(0.001,m,s)` τα ποσοστιαία σημεία 0.1% και 99.9%. Στη συνέχεια, δημιουργούμε 1000 τιμές ανάμεσα στα δύο αυτά άκρα του διαστήματος και χρησιμοποιούμε αυτές τις τιμές για να κατασκευάσουμε το διάγραμμα. Θα μπορούσαμε να γράψουμε τη συνάρτηση, έτσι ώστε να μας επιστρέφει και αποτελέσματα, δηλαδή μέσα σε μια συνάρτηση μπορεί να κατασκευάσουμε, απλά, ένα διάγραμμα σαν μέρος της συνάρτησης. Για παράδειγμα, η επόμενη συνάρτηση, εκτός από τη συνάρτηση πυκνότητας πιθανότητας, υπολογίζει και το ενδοτεταρτημοριακό εύρος της κανονικής κατανομής.

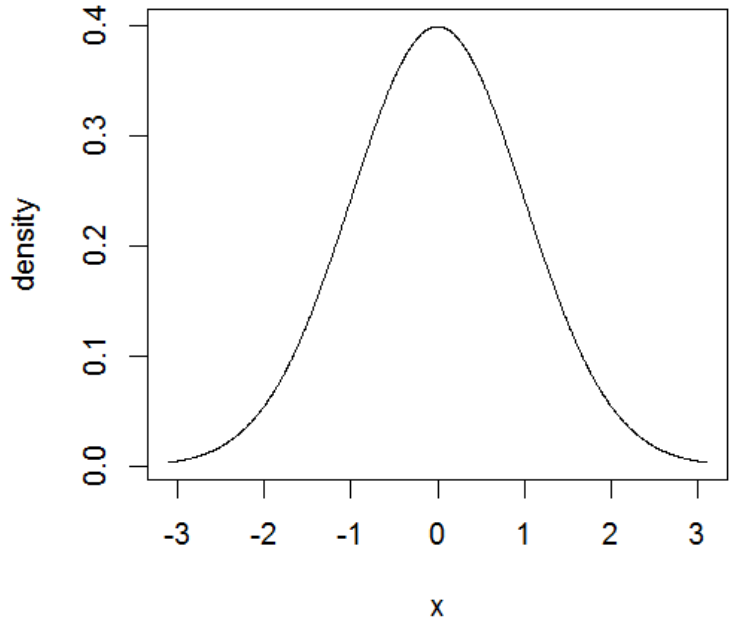
```
plotnormal2<-function(m=0,s=1) {
             xkato<-qnorm(0.001,m,s)
             xano<-qnorm(0.999,m,s)
             x<-seq(xkato,xano,length=1000)
             plot(x,dnorm(x,m,s),ylab="density",type="l")
             qnorm(0.75,m,s) - qnorm(0.25,m,s)
}                                                              ■.
```

΄Ετσι, η κλήση τώρα της συνάρτησης θα έχει ως αποτέλεσμα και την κατασκευή του διαγράμματος 7.1, αλλά και τον υπολογισμό του ενδοτεταρτημοριακού εύρους.

```
> plotnormal2()
[1] 1.34898                                                    ■.
```

- Αν γράψουμε μια νέα συνάρτηση με το ίδιο όνομα, η παλιότερη χάνεται. Για παράδειγμα, πριν λίγο συνεχώς ανανεώναμε τη συνάρτηση `trimmedmean`, με αποτέλεσμα οι παλιότερες εκδόσεις να χάνονται. Συνεπώς, πρέπει να είναι κανείς προσεκτικός στα ονόματα που δίνει στις καινούριες συναρτήσεις, ώστε να μην χάσει ήδη υπάρχουσες.

- Αν πληκτρολογήσουμε το όνομα μιας συνάρτησης, θα δούμε τις εντολές που την αποτελούν. Αυτό είναι πολύ χρήσιμο, όταν θέλουμε να δούμε τι κάνει μια συνάρτηση, αλλά και όταν



Διάγραμμα 7.1: Η συνάρτηση (μάζας) πυκνότητας πιθανότητας της τυποποιημένης κανονικής κατανομής

θέλουμε να πάρουμε μια συνάρτηση για να τις κάνουμε κάποιες αλλαγές. Το ίδιο ισχύει και για συναρτήσεις που έχει ήδη η R. Μπορείτε να το δείτε, απλά, πληκτρολογώντας `mean`, όπου θα εμφανιστεί στην οθόνη ο πλήρης κώδικας της συνάρτησης αυτής. Για παράδειγμα, αν πληκτρολογήσουμε απλά `plotnormal2`, τότε θα εμφανιστούν οι εντολές της συνάρτησης αυτής.

```
> plotnormal2
function(m = 0, s = 1){
        xkato <- qnorm(0.001, m, s)
        xano <- qnorm(0.999, m, s)
        x <- seq(xkato, xano, length = 1000)
        plot(x, dnorm(x, m, s), ylab = "density",type= "l")
        qnorm(0.75, m, s) - qnorm(0.25, m, s)
}                                    ■.                                    ■.
```

7.9 Επιπλέον βιβλιογραφικό υλικό

Για επιπλέον υλικό σχετικά με τις συναρτήσεις (και γενικότερα για προγραμματισμό) στην R παραπέμπουμε στα βιβλία των Braun & Murdoch (2007), Chambers (2008), Matloff (2011) και

Pace (2015). Επιπλέον το βιβλίο του Grolemund (2014) είναι επικεντρωμένο αποκλειστικά στη συγγραφή συναρτήσεων στην R.

Βιβλιογραφικές Αναφορές Κεφαλαίου 7

Braun, W. & Murdoch, D. (2007). *A First Course in Statistical Programming with R*. Cambridge University Press.

Chambers, J. (2008). *Software for Data Analysis: Programming with R*. Statistics and Computing. Springer New York.

Grolemund, G. (2014). *Hands-On Programming with R: Write Your Own Functions and Simulations*. O'Reilly Media.

Matloff, N. (2011). *The Art of R Programming: A Tour of Statistical Software Design*. No Starch Press.

Pace, L. (2015). *Beginning R: An Introduction to Statistical Programming*. Apress.

ΚΕΦΑΛΑΙΟ 8

Βασικές εντολές ανάλυσης δεδομένων

Στην ενότητα αυτή, θα παρουσιάσουμε μερικές από τις βασικές εντολές ανάλυσης δεδομένων. Οι εντολές αυτές θα δοθούν περιληπτικά και όχι εκτεταμένα, καθώς το βιβλίο που κρατάτε στα χέρια σας, κυρίως επικεντρώνεται στην εκμάθηση της R και όχι στην ανάλυση δεδομένων.

8.1 Απλές στατιστικές συναρτήσεις- Περιγραφικά μέτρα

Όπως είδαμε στα προηγούμενα κεφάλαια, η R προσφέρει υψηλής ποιότητας διαγράμματα για την περιγραφή των δεδομένων. Επίσης προσφέρει μια ποικιλία από περιγραφικά μέτρα, μέσα από μια σειρά από στατιστικές συναρτήσεις, που μπορεί να χρησιμοποιήσει ο χρήστης. Μερικές από αυτές μπορείτε να δείτε στον Πίνακα 8.1, όπου x είναι ένα αριθμητικό διάνυσμα με παρατηρήσεις μιας ποσοτικής μεταβλητής. Επίσης, προσφέρει μια σειρά από άλλα περιγραφικά μέτρα, με πιο ανθεκτικές (robust) ιδιότητες, που επηρεάζονται λιγότερο από ακραίες τιμές. Ένα τέτοιο παράδειγμα είναι η συνάρτηση `mad`, η οποία υπολογίζει ένα ανθεκτικό μέτρο μεταβλητότητας και περιλαμβάνει σαν ειδική περίπτωση και τη μέση απόλυτη απόκλιση.

Στο παράδειγμα που ακολουθεί, προσομοιώνουμε 50 παρατηρήσεις από την κανονική κατανομή με μέσο μηδέν και τυπική απόκλιση έξι. Στις τιμές αυτές υπολογίζουμε τα περιγραφικά μέτρα του Πίνακα 8.1.

```
> x<-rnorm(50,0,6)
> x
 [1]  -4.1607447  -4.2830320   0.6836461   2.7697539
 [5]  -8.8928930   0.4519978  -6.0040847  -5.6057633
 [9]  -6.8827595  -7.3827094   2.4763824  -8.9399582
[13]   6.6392061   5.1252083  -4.2151924   9.3845574
[17]   0.2168974   6.1154202  -0.4507548   4.0587800
[21]  -1.8860930   6.0474796  -2.3157290   1.8209788
[25]  10.8970168   2.2159570   2.5437735   0.6180315
```

```
[29] -10.0819013  14.7447170   7.1700254   6.3038185
[33]  -3.3482184   2.8842158  -6.9532273 -12.8407598
[37]   2.7469375  -5.9224643  -1.9748813  -2.2895899
[41]   0.3893093   7.6768122   9.5453959  -6.5928187
[45]   0.6668843  -0.3630957  -6.1072677  -3.7903136
[49]   4.9749856  -3.6629836
> mean(x)
[1] -0.1155809
> mean(x, 0.05)
[1] -0.1847417
> median(x)
[1] 0.3031033
> min(x)
[1] -12.84076                                                    ■.
```

Έτσι, βλέπουμε ότι και οι μέσες τιμές ($\overline{x} = -0.116$ και ο περικομμένος μέσος είναι ίσος με -0.18) και η διάμεσος ($M = 0.30$) είναι πολύ κοντά στη πραγματική τιμή του μηδενός, όπως αναμενόταν.

Όσον αφορά τα μέτρα σχετικής θέσης, μπορούμε να χρησιμοποιήσουμε τι ακόλουθες εντολές:

```
> max(x)
[1] 14.74472
> range(x)
[1] -12.84076  14.74472
> quantile(x)
         0%          25%          50%          75%         100%
-12.8407598  -4.2660721   0.3031033   3.7651390  14.7447170
> quantile(x,0.975)
   97.5%
10.5929                                                          ■.
```

Επιπλέον, μια συνολική εικόνα (των μέτρων σχετικής και κεντρικής θέσης) μπορούμε να πάρουμε με την εντολή `summary`:

```
> summary(x)
    Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-12.8400  -4.2660   0.3031  -0.1156   3.7650  14.7400              ■.
```

Από τα παραπάνω βλέπουμε, ότι οι τιμές κυμαίνονται από το -12.84 έως το 14.74, ενώ το 25% των τιμών είναι μικρότερο του -4.26, ενώ το 75% μικρότερο του 3.76 (ή ισοδύναμα το 25% των τιμών του x είναι μεγαλύτερες της τιμής 3.76). Τέλος, μόνο το 2.5% των τιμών του διανύσματος x ξεπερνούν την τιμή 10.59.

Όσον αφορά τα μέτρα διακύμανσης, με την ακόλουθη σύνταξη

```
> var(x)
[1] 36.04576
> sd(x)
[1] 6.003812
> sqrt(var(x))
[1] 6.003812
```

βλέπουμε ότι η εκτίμηση της διακύμανση είναι ίση με 36 και η τυπική απόκλιση ίση με 6, που είναι και οι πραγματικές τιμές. Εναλλακτικά μέτρα διασποράς μπορούμε να πάρουμε έμμεσα από την εντολή `range`,

```
> range(x)
[1] -12.84076  14.74472
```
■,

η οποία δε δίνει το εύρος (που είναι ένα μέτρο διασποράς), αλλά την ελάχιστη και τη μέγιστη τιμή. Για να υπολογίσουμε το εύρος, απλά γράφουμε:

```
> R<-range(x)
> R[2]-R[1]
[1] 27.58548
```
■,

και διαπιστώνουμε ότι είναι ίσο με 27.6 μονάδες. Με παρόμοιο τρόπο μπορούμε να υπολογίσουμε και το ενδοτεταρτομοριακό εύρος, χρησιμοποιώντας την ακόλουθη σύνταξη:

```
> IQR<- quantile(x, probs=c(0.25, 0.75))
> IRQ[2] - IRQ[1]
     75%
8.031211
```
■,

το οποίο μας δίνει περίπου 8 μονάδες εύρος για το 50% των κεντρικών τιμών του x.

Μπορούμε να υπολογίσουμε το άθροισμα και το γινόμενο των στοιχείων του x εύκολα με τις εντολές:

```
> sum(x)
[1] 51.52361
> prod(x)
[1] 3.930711e+24
```
■. ■.

Πολλά περιγραφικά μέτρα, που δεν προσφέρονται άμεσα από την R, μπορούν πολύ εύκολα να δημιουργηθούν ως συναρτήσεις της R, όπως έχουμε δει ως παραδείγματα σε προηγούμενες ενότητες αυτού του συγγράμματος.

Συνάρτηση της R	**Επεξήγηση**
`length(x)`	Μέγεθος δείγματος n, δηλαδή ο αριθμός των παρατηρήσεων (ή, μαθηματικά, το μήκος του διανύσματος).
`mean(x)`	Δειγματικός μέσος $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$.
`mean(x, p)`	Περικομμένος μέσος. `p` δηλώνει το ποσοστό των παρατηρήσεων που θα περικοπούν από κάθε πλευρά και παίρνει τιμές από το 0 έως και 0.5.
`min(x)`	Μικρότερη τιμή.
`max(x)`	Μεγαλύτερη τιμή.
`median(x)`	Διάμεσος.
`var(x)`	Αμερόληπτη δειγματική διακύμανση $s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2$. Αν το x δεν είναι διάνυσμα, αλλά πίνακας, δηλαδή έχει πολλές στήλες, τότε παίρνουμε τον πίνακα διακύμανσης του πίνακα δεδομένων x.
`sd(x)`	Τυπική απόκλιση $s = \sqrt{s^2}$ βασισμένη στην αμερόληπτη δειγματική διακύμανση.
`range(x)`	Ελάχιστη και μέγιστη τιμή σε ένα διάνυσμα.
`sum(x)`	Άθροισμα $\sum_{i=1}^{n} x_i$.
`prod(x)`	Γινόμενο $\prod_{i=1}^{n} x_i$ (χρήσιμο για το γεωμετρικό μέσο).
`summary(x)`	Επιστρέφει τα περιληπτικά μέτρα του x (μέση τιμή, διάμεσο, το πρώτο και τρίτο τεταρτημόριο και μέγιστη και ελάχιστη τιμή).
`quantile(x)`	Επιστρέφει τα τεταρτημόρια (ελάχιστη τιμή, Q_1, διάμεσο, Q_3 και τη μέγιστη τιμή).
`quantile(x,p)`	Επιστρέφει το p ποσοστιαίο σημείο. Το p μπορεί να έχει τιμή ανάμεσα στο 0 και το 1. Αν είναι διάνυσμα, τότε θα επιστρέψει τα αντίστοιχα ποσοστιαία σημεία ως διάνυσμα.
`cor(x1,x2)` `cor(x)`	Δείκτης γραμμικής συσχέτισης του Pearson ανάμεσα στα διανύσματα $x1$ και $x2$, ή τον πίνακα συσχέτισης, αν το x είναι ένας πίνακας (βλ. Ενότητα 8.7.1.2 για λεπτομέρειες).

Πίνακας 8.1: Βασικά περιγραφικά μέτρα στην R

8.2 Κατανομές

Η R περιέχει προκαθορισμένες συναρτήσεις για τον υπολογισμό ποσοτήτων από αρκετές κατανομές (διακριτές και συνεχείς) και, κυρίως, όλες τις βασικές κατανομές που χρειάζεται κανείς. Συγκεκριμένα, για πολλές κατανομές πιθανότητας είναι εφικτή η παραγωγή τυχαίων αριθμών και ο υπολογισμός των συναρτήσεων:

- (πυκνότητας) πιθανότητας $f(x)$,
- κατανομής $F(x)$, και
- των ποσοστιαίων σημείων $Q(p) = F^{-1}(p)$.

Ο τρόπος που χρησιμοποιούμε αυτές τις συναρτήσεις, είναι παρόμοιος για όλες τις κατανομές. Κάθε κατανομή έχει ένα όνομα, και βάζοντας το κατάλληλο πρόθεμα: `r` για τις γεννήτριες τυχαίων αριθμών, `d` για τη συνάρτηση (μάζας) πυκνότητας πιθανότητας, `p` για τη συνάρτηση κατανομής και `q` για τα ποσοστιαία σημεία• βλ. και Πίνακα 8.2, χρησιμοποιώντας ως παράδειγμα την κανονική κατανομή (norm).

Γράμμα	**Αποτέλεσμα**	**Παράδειγμα με τη χρήση της τυποποιημένης κανονικής κατανομής**
`r`	Τυχαίους αριθμούς	`rnorm(n,0,1)`
`d`	Πυκνότητα πιθανότητας	`dnorm(x,0,1)`
`p`	Αθροιστική συνάρτηση κατανομής	`pnorm(x,0,1)`
`q`	Ποσοστιαία σημεία	`qnorm(p,0,1)`

Πίνακας 8.2: Τύποι συναρτήσεων σχετικών με κατανομές

Παρόλο που το πρόθεμα `d` αναφέρεται στο αρχικό γράμμα της αγγλικής λέξης density, που σημαίνει πυκνότητα, οι σχετικές συναρτήσεις μας δίνουν τη συνάρτηση πιθανότητας, όταν η τυχαία μεταβλητή είναι διακριτή, δηλαδή $f(x) = P(X = x)$ και τη συνάρτηση (μάζας) πυκνότητας πιθανότητας, μόνο στην περίπτωση των συνεχών τυχαίων μεταβλητών. Να υπενθυμίσουμε ότι οι συναρτήσεις κατανομών $F(x)$, δίνουν την πιθανότητα μέχρι και την τιμή x, δηλαδή $F(x) = P(X \leq x)$ και υπολογίζεται ως

$$F(x) = \sum_{k \leq x} P(X = k)$$

στην περίπτωση μιας διακριτής τυχαίας μεταβλητής και από τον τύπο:

$$F(x) = \int_{-\infty}^{x} f(t)dt$$

στην περίπτωση μιας συνεχούς τυχαίας μεταβλητής. Τέλος, η συνάρτηση των ποσοστιαίων σημείων είναι η αντίστροφη συνάρτηση της συνάρτησης κατανομής, δηλαδή $Q(p) = F^{-1}(p)$ και μας δίνει την τιμή της τυχαίας μεταβλητής με αθροιστική πιθανότητα ίση με p, δηλαδή $Q(p) = x$ τέτοιο, ώστε $F(x) = p$. Για το λόγο αυτό, στις συναρτήσεις των ποσοστιαίων σημείων εισάγουμε μια τιμή πιθανότητας p και όχι πιθανές τιμές για το x, όπως στους άλλους δύο τύπους συναρτήσεων. Όσον αφορά τις γεννήτριες τυχαίων αριθμών (με το πρόθεμα r), αυτές έχουν ως κύρια παράμετρο εισαγωγής τον αριθμό των τυχαίων αριθμών n, που επιθυμούμε να παράγουμε. Τέλος, σχετικά υποχρεωτικές παράμετροι εισαγωγής, είναι οι παράμετροι των κατανομών, οι οποίες έχουν προκαθορισμένες τιμές, στις περισσότερες περιπτώσεις, παραπέμποντας στις πιο συνηθισμένες περιπτώσεις τους. Για παράδειγμα, η κανονική κατανομή έχει παραμέτρους το μέσο mu και την τυπική απόκλιση sd, με προκαθορισμένες τιμές μηδέν και ένα αντίστοιχα, παραπέμποντας στην τυποποιημένη κανονική κατανομή. Συνεπώς, για κάθε κατανομή θα πρέπει να ορίσουμε και τις παραμέτρους της, οι οποίες διαφέρουν για κάθε κατανομή ως προς τον αριθμό τους, το πεδίο τιμών τους και την ερμηνεία τους. Για την ερμηνεία των παραμέτρων συμβουλευτείτε τη βοήθεια της R, καθώς πολλές κατανομές έχουν ποικίλους τρόπους ορισμού και παραμετροποίησης.

Οι κατανομές που είναι διαθέσιμες, περιλαμβάνουν την κανονική (τυποποιημένη ή μη), τη διωνυμική, την κατανομή Poisson, την εκθετική, τη Student-t, τη Γάμμα, τη Βήτα, την ομοιόμορφη, τη χ^2 , την F και αρκετές ακόμα, λιγότερο γνωστές, που ίσως δεν έχετε ακόμα συναντήσει. Μια σχετικά πλήρη λίστα μπορείτε να δείτε στον Πίνακα 8.3. Γενικά, υπάρχει μεγάλη ποικιλία συναρτήσεων πιθανοτήτων στην R σε διάφορα πακέτα. Μια εκτεταμένη λίστα μπορείτε να βρείτε στο σχετικό ιστότοπο του CRAN[1].

Την ερμηνεία των παραμέτρων για κάθε κατανομή, μπορείτε να την βρείτε μέσα από το `help`. Για την κανονική κατανομή που χρησιμοποιήσαμε στον Πίνακα 8.2 και στα παραδείγματα που ακολουθούν, η πρώτη παράμετρος είναι η μέση τιμή και η δεύτερη η τυπική απόκλιση της κατανομής. Μερικά παραδείγματα ακολουθούν:

```
> dbinom(5, size=10, prob=0.4)
[1] 0.2006581
> dnorm( 180, 170, 5)
[1] 0.01079819                                                    ■.
```

Στο πρώτο από τα δύο παραπάνω παραδείγματα, υπολογίσαμε την πιθανότητα να έχουμε πέντε επιτυχίες σε ένα διωνυμικό πείραμα με 10 επαναλήψεις και πιθανότητα επιτυχίας 40% (και τη βρήκαμε ίση με 20%). Με την επόμενη εντολή, βρήκαμε ότι η πυκνότητα της πιθανότητας της τιμής 180 είναι ίση με 0.0107 για την κανονική κατανομή, με μέσο 170 και τυπική απόκλιση 5. Με τις ακόλουθες εντολές:

```
> pbinom(5, size=10, prob=0.4)
[1] 0.8337614
> pnorm( 180, 170, 5)
```

[1] https://cran.r-project.org/web/views/Distributions.html

Διακριτές κατανομές	**Εντολή R**
Αρνητική διωνυμική	`dnbinom`
Γεωμετρική	`dgeom`
Διωνυμική	`dbinom`
Υπεργεωμετρική	`dhyper`
Poisson	`dpois`
Συνεχείς κατανομές	**Εντολή R**
Βήτα	`beta`
Γάμμα	`gamma`
Εκθετική	`exp`
Κανονική	`norm`
Λογαριθμοκανονική	`lnorm`
Λογιστική	`logis`
Ομοιόμορφη	`unif`
Πολυμεταβλητή κανονική	`mvnorm`
χ^2	`chisq`
Cauchy	`cauchy`
Student t	`t`
Weibull	`weibull`

Πίνακας 8.3: Οι διαθέσιμες κατανομές στην R

```
[1] 0.9772499
```

υπολογίζουμε τις αντίστοιχες τιμές των συναρτήσεων κατανομής. Έτσι, στην πρώτη περίπτωση υπολογίζουμε ότι, η πιθανότητα να έχουμε μέχρι και πέντε επιτυχίες στη διωνυμική 10 επαναλήψεων και πιθανότητα επιτυχίας 40%, είναι ίση με 83.4%, ενώ στη δεύτερη περίπτωση η πιθανότητα να έχουμε τιμή μικρότερη (ή ίση) του 180 σε μια κανονική κατανομή, με μέσο 170 και τυπική απόκλιση 5, είναι ίση με 0.977. Αν επιθυμούμε να υπολογίσουμε τις πιθανότητες για ενδεχόμενα μεγαλύτερα από κάποια τιμή, τότε μπορούμε να χρησιμοποιήσουμε την παράμετρο `lower.tail = FALSE`,

```
> pbinom(5, size=10, prob=0.4, lower.tail=FALSE)
[1] 0.1662386
> pnorm( 180, 170, 5, lower.tail=FALSE)
[1] 0.02275013
```

ή απλά να αφαιρέσουμε τις αρχικές τιμές από τη μονάδα, δηλαδή:

```
> 1-pbinom(5, size=10, prob=0.4)
[1] 0.1662386
```

```
> 1-pnorm( 180, 170, 5)
[1] 0.02275013                                                    ■.
```

Όσον αφορά τα ποσοστιαία σημεία, μπορούμε για παράδειγμα να δώσουμε τις εντολές:

```
> qbinom(0.8337, size=10, prob=0.4)
[1] 6
> qbinom(0.8337, size=10, prob=0.4)
[1] 5
> qnorm( 0.977, 170, 5)
[1] 179.977                                                       ■,
```

και να πάρουμε πίσω τις τιμές (ή έστω κοντινές), που είχαμε δώσει ως παραμέτρους εισαγωγής στις εντολές `pbinom` και `pnorm`, αντίστοιχα (λόγω της αντίστροφης σχέσης μεταξύ των δύο συναρτήσεων). Για λόγους ερμηνείας, ας υποθέσουμε ότι έχουμε ένα τεστ με 100 ερωτήματα και πιθανότητα σωστής απάντησης 70%. Η παρακάτω εντολή:

```
> qbinom(0.90, size=100, prob=0.7)
[1] 76
```

θα μας πει πόσες (τουλάχιστον) ερωτήσεις πρέπει να απαντήσουμε για να είμαστε στο 10% των καλύτερων επιδόσεων (εδώ πρέπει να απαντήσουμε σε 76 ερωτήσεις και πάνω).

Να σημειώσουμε, ότι και στις τρεις συναρτήσεις μπορούμε να χρησιμοποιήσουμε διάνυσμα στη θέση των κύριων τιμών εισαγωγής (`x`, `x` και `p`, αντίστοιχα). Ως αποτέλεσμα θα πάρουμε ένα διάνυσμα y, με κάθε στοιχείο y_i να είναι ίσο με την τιμή της συνάρτησης, υπολογισμένο στο αντίστοιχο στοιχείο του διανύσματος εισαγωγής (x_i ή p_i) και τις ίδιες παραμέτρους της κατανομής. Έτσι για παράδειγμα, η εντολή:

```
> dpois(0:10,5)
 [1] 0.006737947 0.033689735 0.084224337 0.140373896
 [5] 0.175467370 0.175467370 0.146222808 0.104444863
 [9] 0.065278039 0.036265577 0.018132789
```

μας δίνει τις πιθανότητες για τους ακέραιους αριθμούς από το μηδέν έως το 10 ($x \in \{0, 1, 2, \ldots, 10\}$) για μια κατανομή Poisson με μέσο πέντε. Αντίστοιχα, η εντολή:

```
> pnorm(c(1.645,1.96),0,1)
[1] 0.9500151 0.9750021
```

θα μας δώσει τις πιθανότητες να έχουμε τιμές κάτω από 1.645 και 1.96 στην τυποποιημένη κανονική κατανομή, ενώ η εντολή:

```
> qnorm(c(0.025,0.05,0.50,0.90,0.95,0.975),0,1)
[1] -1.959964 -1.644854  0.000000  1.281552  1.644854  1.959964
```

θα μας δώσει τα 2.5%, 5%, 50%, 90%, 95% και 97.5% ποσοστιαία σημεία της τυποποιημένης κανονικής κατανομής.

Αν έχουμε διανύσματα και στις ίδιες τις παραμέτρους εισαγωγής, τότε η αντιστοίχιση γίνεται στοιχείο με στοιχείο. Καλό θα είναι, τα διανύσματα αυτά να έχουν το ίδιο μήκος. Αν έχουν διαφορετικό μήκος, τότε γίνεται ανακύκλωση των στοιχείων (χωρίς κάποιο μήνυμα), μέχρι να φτάσουν στο ίδιο μήκος, το οποίο μπορεί να έχει τα μη αναμενόμενα αποτελέσματα και καλό θα είναι να αποφεύγεται. Μερικά παραδείγματα ακολουθούν.

```
> dpois( 0:3, c(1.1, 4, 3, 2) )
[1] 0.33287108 0.07326256 0.22404181 0.18044704
>
> dpois( 0:3, c(1.1, 4, 3) )
[1] 0.33287108 0.07326256 0.22404181 0.07384190
>
> dpois( 0, c(1.1, 4, 3, 2) )
[1] 0.33287108 0.01831564 0.04978707 0.13533528
>
> dpois( 2:3, c(1.1, 4, 3) )
[1] 0.2013870 0.1953668 0.2240418                                    ■.
```

Στο πρώτο παράδειγμα έγινε ο υπολογισμός των πιθανοτήτων $P(X = 0)$, $P(X = 1)$, $P(X = 2)$ και $P(X = 3)$ για τις κατανομές Poisson με μέσους 1.1, 4, 3 και 2, αντίστοιχα. Στο δεύτερο παράδειγμα έχουμε σχεδόν την ίδια σύνταξη, με τη διαφορά ότι το διάνυσμα των μέσων έχει μήκος 3. Λόγω ανακύκλωσης, το διάνυσμα των μέσων γίνεται `c(1.1, 4, 3, 1.1)` και έτσι στο αποτέλεσμα έχουμε διαφορά (σε σχέση με το πρώτο παράδειγμα), μόνο στο τελευταίο στοιχείο, όπου υπολογίζεται η πιθανότητα $P(X = 3)$ για την κατανομή Poisson με μέσο 1.1 αντί για 2. Στο τρίτο παράδειγμα υπολογίζεται η πιθανότητα $P(X = 0)$ για τις κατανομές Poisson με τις τέσσερις διαφορετικές τιμές του μέσου. Τέλος, στο τέταρτο παράδειγμα έχουμε εφαρμογή της ανακύκλωσης για το διάνυσμα `x`, το οποίο γίνεται `2, 3, 2` και ως αποτέλεσμα παίρνουμε τις πιθανότητες $P(X = 2)$, $P(X = 3)$ και $P(X = 2)$ για τις κατανομές Poisson με μέσους 1.1, 4 και 3, αντίστοιχα.

Στο παράδειγμα που ακολουθεί, παράγουμε 10 τυχαίους αριθμούς από την κανονική κατανομή με μέση τιμή 0 και τυπική απόκλιση 5.

```
> rnorm(10,0,5)
 [1]  5.1298738  3.0018368 -5.0527777  8.5999875 -7.0608498
 [6]  1.3271696  1.9620653  0.3388949 -2.2597581  4.1290557      ■.
```

Το αποτέλεσμα εμφανίζεται ως ένα διάνυσμα μήκους $n = 10$. Προφανώς, εφόσον πρόκειται για τυχαίους αριθμούς, κάθε φορά που θα καλούμε τη συνάρτηση θα παίρνουμε και άλλες τιμές. Στο επόμενο παράδειγμα παράγουμε 15 τυχαίους αριθμούς από την κατανομή Poisson, με μέση τιμή 1.

```
> rpois(15,1)
 [1] 0 2 1 1 1 0 0 3 2 1 1 0 2 0 0                                  ■.
```

Να σημειώσουμε, ότι το πλήθος των τυχαίων αριθμών δε μπορεί να είναι διάνυσμα, όπως στις περιπτώσεις των τριών άλλων συναρτήσεων (πυκνότητας, κατανομής και ποσοστιαίων σημείων). Στην περίπτωση που βάλουμε διάνυσμα στη θέση του n, τότε θα λάβει ως μήκος του διανύσματος το επιθυμητό αριθμό των αριθμών που θέλουμε να παράγουμε, όπως βλέπουμε στο παράδειγμα που ακολουθεί:

```
> rpois( c(1,4,5,1) ,1)
[1] 0 3 2 0
```
■.

Να σημειώσουμε, επίσης, την ύπαρξη της συνάρτησης `sample` που παράγει τυχαίες αναδιατάξεις από ένα διάνυσμα και τυχαίους αριθμούς με επανάθεση (επανατοποθέτηση), δηλαδή αλλαγή των πιθανοτήτων επιτυχίας, γιατί κάθε αριθμός εμφανίζεται μία φορά στο δείγμα, ή χωρίς επανάθεση (δηλαδή ένας αριθμός μπορεί να εμφανιστεί πολλές φορές στο δείγμα και οι πιθανότητες εμφάνισης κάθε στοιχείου του διανύσματος δεν αλλάζουν). Στα παραδείγματα που ακολουθούν, παίρνουμε το διάνυσμα $x = (5, 10, 20, 1, 2)$ και παράγουμε μία τυχαία αναδιάταξή του, παίρνουμε 3 τυχαίους αριθμούς (χωρίς επανάθεση), παίρνουμε 3 τυχαίους αριθμούς (με επανάθεση), όπου βλέπουμε ότι το 10 επαναλαμβάνεται 2 φορές, και τέλος, 30 τυχαίους αριθμούς (με επανάθεση).

```
> sample( c(5,10,20,1,2) )
[1]  2 20 10  1  5
> sample( c(5,10,20,1,2),3 )
[1]  5 20 10
> sample( c(5,10,20,1,2),3, replace=TRUE )
[1] 10  1 10
> sample( c(5,10,20,1,2),30, replace=TRUE )
[1]  37  64  94   5   1  79  64  84  51  82  72  20
[13]  42  60  98  72  30  85  69  51   8  65  88  41
[25] 100  38  45  33  29  88
```
■.

Στα παραπάνω παραδείγματα εισάγουμε ένα διάνυσμα με τα πιθανά ενδεχόμενα, τα οποία θα μπορούσαν να είναι και μια σειρά από χαρακτήρες. Αν θέλουμε να παίρνουμε τυχαίους αριθμούς από το ένα έως το n, τότε είτε εισάγουμε το διάνυσμα `1:n` ως εισαγώμενη παράμετρο στην συνάρτηση `sample`, είτε χρησιμοποιούμε την εντολή `sample.int`. Μερικά απλά και αντίστοιχα παραδείγματα ακολουθούν.

```
> sample.int(10)
 [1]  3  5  9  4  8  1  7 10  6  2
> sample.int(10,5)
[1] 3 7 1 8 5
> sample.int(10,5, replace=TRUE)
[1] 8 6 5 4 6
> sample.int(10)
```

```
 [1]  1  8  2  7  4  9  3 10  6  5                                    ■.
```

Πριν κλείσουμε την ενότητα αυτή, θα θέλαμε να τονίσουμε τη σημαντικότητα των λογικών παραμέτρων `log` και `log.p`, των συναρτήσεων (μάζας) πυκνότητας πιθανότητας και κατανομής αντίστοιχα. Και οι δύο υπολογίζουν τις αντίστοιχες συναρτήσεις σε λογαριθμική κλίμακα και είναι εξαιρετικά χρήσιμες για την αποφυγή υπερχείλιση (underflows) και υποχείλιση (overflows).

```
> dnorm(1.96)
[1] 0.05844094
> dnorm(1.96,log=T)
[1] -2.839739
> exp(dnorm(1.96,log=T))
[1] 0.05844094
>  pnorm(1.96,log.p=TRUE)
[1] -0.02531565
> pnorm(1.96,log=TRUE)
[1] -0.02531565
> pnorm(1.96,log.p=F)
[1] 0.9750021
> exp(pnorm(1.96,log.p=TRUE))
[1] 0.9750021
> qnorm(0.975)
[1] 1.959964                                                          ■.
```

Θα πρέπει επίσης να τονιστεί, πως αρκετές άλλες κατανομές προσφέρονται μέσα από διάφορες βιβλιοθήκες που υπάρχουν στο διαδίκτυο. Συνεπώς, είναι καλή ιδέα όταν χρειάζεστε κάποια άλλη κατανομή να καταφεύγετε σε έτοιμες συναρτήσεις, που με μεγάλη πιθανότητα υπάρχουν σε άλλες βιβλιοθήκες. Οι παραπάνω συναρτήσεις, συνήθως, έχουν γραφεί με βέλτιστο τρόπο για να αποφεύγονται προβλήματα και συνεπώς, είναι αξιόπιστες. Βασικά, όσο πιο δημοφιλές είναι ένα πακέτο, τόσο πιο μεγάλη αξιοπιστία περιμένουμε να έχει. Στην περίπτωση που είστε αρχάριος χρήστης, καλό θα είναι να προτιμάτε τις έτοιμες συναρτήσεις, ενώ για πιο έμπειρους χρήστες προτείνουμε να δοκιμάζουν να φτιάχνουν τις δικές τους, για να γνωρίζουν τι ακριβώς υπολογίζουν.

8.3 ΄Ελεγχοι υποθέσεων για ένα και δύο δείγματα

Στην ενότητα αυτή θα αναφερθούμε περιληπτικά στους βασικούς ελέγχους υποθέσεων. Στους ελέγχους περιλαμβάνονται, τόσο παραμετρικοί, όσο και μη παραμετρικοί έλεγχοι. Η θεωρία που οδηγεί σε αυτούς τους ελέγχους, μπορεί να βρεθεί σε βασικά βιβλία στατιστικής και για αυτό περιοριζόμαστε σε μια συνοπτική αναφορά τους.

Πριν ξεκινήσουμε για τους ελέγχους υποθέσεων, θα αναφερθούμε σε κάποια γενικά στοιχεία για την ομάδα αυτή των συναρτήσεων στην R.

8.3.1 Γενικά για τους ελέγχους υποθέσεων στην R

Με τους ελέγχους υποθέσεων ελέγχουμε την ορθότητα μιας μηδενικής υπόθεσης H_0 (όπου, συνήθως, μία ή περισσότερες παράμετροι θ είναι ίσες με κάποιες τιμές θ_0), έναντι μιας εναλλακτικής H_1, για παράδειγμα:

$$H_0 : \theta = \theta_0 \text{ έναντι της } H_1 : \theta \neq \theta_0 \ ,$$

όπου θ μπορεί να είναι ο μέσος, η διάμεσος, η διακύμανση ή οποιοδήποτε άλλη παράμετρος του πληθυσμού ή της κατανομής που υποθέτουμε.

Η εναλλακτική υπόθεση.
Η εναλλακτική υπόθεση H_1 μπορεί να πάρει τρεις εναλλακτικές μορφές:

1. Δίπλευρος έλεγχος όπου $H_1 : \theta \neq \theta_0$,
2. Μονόπλευρος μικρότερος του θ_0 όπου $H_1 : \theta < \theta_0$, και
3. Μονόπλευρος μεγαλύτερος του θ_0 όπου $H_1 : \theta > \theta_0$.

Στις συναρτήσεις ελέγχων υποθέσεων, η επιλογή της εναλλακτικής υπόθεσης γίνεται από το όρισμα `alternative` και τις επιλογές: `"two.sided"` (προκαθορισμένος), `"less"`, και `"greater"`.

Ορισμός των δεδομένων.
Σε όλες τις συναρτήσεις ελέγχων υποθέσεων πρέπει να εισάγουμε κάποια δεδομένα. Αυτό γίνεται υποχρεωτικά από τα ορίσματα δεδομένων, που συνήθως, είναι διανύσματα. Εναλλακτικά, σε όλα μπορούμε να χρησιμοποιήσουμε τη σύνταξη `formula`, που χρησιμοποιείται κυρίως στα στατιστικά μοντέλα και στις τελευταίες εκδόσεις πλέον χρησιμοποιείται και στους ελέγχους υποθέσεων. Για τον εναλλακτικό αυτό τρόπο θα επανέλθουμε στην Ενότητα 8.6, όπου περιγράφουμε τον τρόπο αυτό σύνταξης.

Διαστήματα εμπιστοσύνης.
Σε όλες τις συναρτήσεις/εντολές ελέγχων υποθέσεων υπάρχει το όρισμα `conf.level`, που καθορίζει το επίπεδο εμπιστοσύνης για το διάστημα εμπιστοσύνης της παραμέτρου θ, για την οποία γίνεται ο έλεγχος. Με τον τρόπο αυτόν δίνουμε το επίπεδο σημαντικότητας του ελέγχου που είναι η συμπληρωματική πιθανότητα που δίνουμε. Η προκαθορισμένη τιμή είναι ίση με 0.95, έτσι ώστε να υπολογίζονται τα 95% διαστήματα εμπιστοσύνης.

Αποτελέσματα και αντικείμενα ελέγχων υποθέσεων.
Οι εντολές/συναρτήσεις ελέγχων υποθέσεων δίνουν πολλαπλά αποτελέσματα, κάποια εκ των οποίων είναι κοινά για τις διαφορετικές συναρτήσεις αυτού του τύπου. Αν απλά ζητήσουμε την εφαρμογή ενός ελέγχου υπόθεσης, τότε θα εμφανιστεί ένα περιληπτικό κείμενο με τα κυριότερα αποτελέσματα. Για παράδειγμα, η εντολή:

```
> t.test(x)
```

θα εφαρμόσει ένα $t-test$ για τον έλεγχο της $H_0 : \mu = 0$ έναντι της εναλλακτικής $H_1 : \mu \neq 0$ με αποτελέσματα:

```
        One Sample t-test

data:  x
t = 0.3838, df = 99, p-value = 0.7019
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 -0.1619135  0.2395827
sample estimates:
 mean of x
0.03883459
```

που θα εμφανιστούν άμεσα στην κονσόλα της R.

Όμως, όπως έχουμε αναφέρει, τα αποτελέσματα των εντολών/συναρτήσεων της R είναι και αυτά αντικείμενα, συνήθως της μορφής λίστας και ως συνέπεια μπορούν να αποθηκευτούν τα περιεχόμενά τους (όλα ή κομμάτια τους), να ανακληθούν αργότερα και να χρησιμοποιηθούν σε άλλα προγράμματα, συναρτήσεις ή αναλύσεις. Έτσι, λοιπόν, με την εντολή:

```
> z<-t.test(x)
```

αποθηκεύουμε τα παραπάνω αποτελέσματα στο αντικείμενο `z`. Μπορούμε να δούμε τον τύπο και την κλάση του `z` με τις εντολές:

```
> mode(z)
[1] "list"
> class(z)
[1] "htest"
```
■,

όπου βλέπουμε ότι είναι μια λίστα κλάσης `htest` (δηλαδή hypothesis test). Μπορούμε να δούμε τα περιεχόμενα του `z` με την εντολή `names` και να αποσπάσουμε τα αντικείμενά της με τη χρήση του συμβόλου του δολαρίου ($), όπως σε μια συνηθισμένη λίστα:

```
> names(z)
[1] "statistic"   "parameter"   "p.value"     "conf.int"
[5] "estimate"    "null.value"  "alternative" "method"
[9] "data.name"
> z$p.value
[1] 0.7019161
```
■.

Αντίθετα, αν απλά γράψουμε το όνομα του αντικειμένου, αντί να εμφανιστούν όλα τα περιεχόμενα, εμφανίζεται ένα πιο φιλικό περιληπτικό output των αποτελέσμάτων των ελέγχων.

```
> z
```

```
        One Sample t-test

data:  x
t = 0.3838, df = 99, p-value = 0.7019
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 -0.1619135  0.2395827
sample estimates:
 mean of x
0.03883459                                                    ■.
```

Τα ονόματα του εξερχόμενου αντικειμένου κλάσης `htest` είναι (σχεδόν) σε όλες τις συναρτήσεις ελέγχων υποθέσεων ίδια, για λόγους ομοιομορφίας και εύκολης ανάκλησης τους από το χρήστη. Στην λίστα που ακολουθεί, περιγράφουμε σε τι αναφέρεται το κάθε στοιχείο ενός αντικειμένου κλάσης `htest`, συνοδευόμενο με με το αντίστοιχο αντικείμενο του αποτελέσματος της εντολής `t.test(x)` (δηλαδή ενός t-test για ένα δείγμα):

`statistic` : Η δειγματική τιμή της ελεγχο-συνάρτησης (test statistic), τιμή κάτω από τη μηδενική υπόθεση, που χρησιμοποιείται για τον έλεγχο.

```
> z$statistic
        t
0.3838455                                                     ■.
```

`parameter` : Η τιμή της παραμέτρου (ή των παραμέτρων) της κατανομής της ελεγχο-συνάρτησης που χρησιμοποιείται.

```
> z$parameter
df
99                                                            ■.
```

`p.value` : Η τιμή p (p-value).

```
> z$p.value
[1] 0.7019161                                                 ■.
```

`conf.int` : Διάστημα εμπιστοσύνης.

```
> z$conf.int
[1] -0.1619135  0.2395827
attr(,"conf.level")
[1] 0.95                                                      ■.
```

`estimate` : Η δειγματική εκτίμηση της παραμέτρου που χρησιμοποιείται στις υποθέσεις του ελέγχου.

```
> z$estimate
 mean of x
0.03883459                                                    ■.
```

`null.value` : Η τιμή της παραμέτρου κάτω από την H_0.

```
> z$null.value
mean
   0                                                          ■.
```

`alternative` : Ο τύπος της εναλλακτικής υπόθεσης H_1, που έχει χρησιμοποιηθεί ($\neq, <, >$), είναι ένα διάνυσμα χαρακτήρων μήκους ένα, όπως έχει οριστεί από το όρισμα `alternative` (βλ. στην αρχή αυτής της ενότητας για λεπτομέρειες).

```
> z$alternative
[1] "two.sided"                                               ■.
```

`method` : Δίνεται η ονομασία του ελέγχου, που έχει εφαρμοστεί (διάνυσμα χαρακτήρων με ένα στοιχείο).

```
> z$method
[1] "One Sample t-test"                                       ■.
```

`data.name` : Περιγράφονται λεκτικά τα δεδομένα, που έχουν χρησιμοποιηθεί (διάνυσμα χαρακτήρων με ένα στοιχείο).

```
> z$data.name
[1] "x"                                                       ■.
```

8.3.2 ΄Ελεγχος για ένα ποσοστό.

Ο έλεγχος για μία αναλογία ή ένα ποσοστό, γίνεται με την εντολή `binom.test` και τη γενική σύνταξη:

```
binom.test(x, n, p=0.5, alternative="two.sided",conf.level=0.95) ■,
```

όπου `x` είναι ο αριθμός των επιτυχιών σε `n` δοκιμές Bernoulli με πιθανότητα επιτυχίας `p`. Με την εντολή αυτή, ελέγχουμε για την ορθότητα της υπόθεσης

$$H_0 : p = p_0 \text{ έναντι της εναλλακτικής } H_1 : p \neq p_0 \ .$$

Ο τύπος της εναλλακτικής υπόθεση καθορίζεται από τις επιλογές `"two.sided"`, `"less"` ή `"greater"`, όπως περιγράψαμε στην Ενότητα 8.3.1.

Στο ακόλουθο παράδειγμα ελέγχουμε, αν ένα δείγμα με 120 επιτυχίες σε σύνολο 350 ανεξάρτητων δοκιμών Bernoulli (δηλαδή δειγματικό ποσοστό επιτυχίας 34.3%), προέρχεται από ένα πληθυσμό με ποσοστό επιτυχίας 35%.

```
> binom.test(120,350,p=0.35)

        Exact binomial test

data:  120 and 350
number of successes = 120, number of trials = 350,
p-value = 0.8227
alternative hypothesis: true probability of success is not equal to 0.35
95 percent confidence interval:
 0.2932165 0.3951830
sample estimates:
probability of success
             0.3428571                                                    ■.
```

Στο παράδειγμα βλέπουμε ότι p-value=$0.8227 > 0.05$, συνεπώς, δεν απορρίπτεται η μηδενική υπόθεση ότι $p = 0.35$, έναντι της εναλλακτικής $H_1 : p \neq 0.35$ σε επίπεδο σημαντικότητας $\alpha = 5\%$ (το ίδιο ισχύει για $\alpha = 10\%$ ή $\alpha = 1\%$). Επιπλέον, βλέπουμε ότι το 95% διάστημα εμπιστοσύνης για την αναλογία είναι $(0.293,\ 0.395)$, δηλαδή το διάστημα εμπιστοσύνης περιλαμβάνειτην τιμή 0.35 κατω από την H_0, ενώ η σημειακή εκτίμηση είναι 0.3429.

Με την εντολή `prop.test`, γίνεται ο έλεγχος για ποσοστό `p` λαμβάνοντας υπόψη τη διόρθωση συνέχειας (continuity correction), δηλαδή γίνεται προσέγγιση διακριτής κατανομής από συνεχή βάση του ΚΟΘ για μεγάλα δείγματα. Ο έλεγχος αυτός είναι προσεγγιστικός και ισχύει για μεγάλα δείγματα. Στο παρακάτω παράδειγμα βλέπουμε ότι για 100 τυχαίους αριθμούς από την Bernoulli (διωνυμική με $= 1$) δεν απορρίπτεται η μηδενική υπόθεση ότι $p = 0.5$ εφόσον το p-value$= 0.368 > 0.05$ για επίπεδο σημαντικότητας $\alpha = 5\%$.

```
> heads <- rbinom(1, size = 100, prob = .5)
> prop.test(heads, 100)          # continuity correction TRUE by
   default

        1-sample proportions test with continuity correction

data:  heads out of 100, null probability 0.5
X-squared = 0.81, df = 1, p-value = 0.3681
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
 0.3514281 0.5524574
sample estimates:
   p
0.45
```

```
> prop.test(heads, 100, correct = FALSE)

        1-sample proportions test without continuity correction

data:  heads out of 100, null probability 0.5
X-squared = 1, df = 1, p-value = 0.3173
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
 0.3561454 0.5475540
sample estimates:
   p
0.45

> ## Data from Fleiss (1981), p. 139.
> ## H0: The null hypothesis is that the four populations from
   which
> ##     the patients were drawn have the same true proportion of
   smokers.
> ## A:  The alternative is that this proportion is different in at
> ##     least one of the populations.
>
> smokers  <- c( 83, 90, 129, 70 )
> patients <- c( 86, 93, 136, 82 )
> prop.test(smokers, patients)

        4-sample test for equality of proportions without
   continuity
        correction

data:  smokers out of patients
X-squared = 12.6004, df = 3, p-value = 0.005585
alternative hypothesis: two.sided
sample estimates:
   prop 1    prop 2    prop 3    prop 4
0.9651163 0.9677419 0.9485294 0.8536585
```

■.

8.3.3 Έλεγχοι t για ένα ή δύο μέσους.

Με τη συνάρτηση `t.test` υπάρχει η δυνατότητα να γίνουν όλοι οι έλεγχοι για έναν ή δύο μέσες τιμές, βασισμένοι στην κατανομή t του Student. Η γενική σύνταξή της εντολής έχει την ακόλουθη μορφή:

```
t.test(x, y=NULL, alternative="two.sided", mu=0, paired=F,
                  var.equal=T, conf.level=.95)                    ■.
```

Πιο συγκεκριμένα, μπορούμε να εφαρμόσουμε τους ελέγχους:

- για έναν μέσο (One sample t-test): $H_0 : \mu =$ mu

  ```
  t.test(x, mu=0)                                                 ■.
  ```

- για τη διαφορά των μέσων σε δύο ανεξάρτητα δείγματα (independent samples t-test): $H_0 : \mu_x - \mu_y =$ mu.

  ```
  t.test(x, y=NULL, mu=0)                                         ■.
  ```

- για την ισότητα μέσων σε δύο εξαρτημένα δείγματα (paired t-test): $H_0 : \mu_{x-y} =$ mu, όπου μ_{x-y} ο μέσος της διαφοράς των τιμών των ζευγαριών.

  ```
  t.test(x, mu=0, paired=TRUE)                                    ■.
  ```

Η επιλογή, αν θα έχουμε ελέγχους για ένα ή δύο δείγματα, γίνεται μέσα από τα ορίσματα `x` και `y`, όπου εισάγουμε τις τιμές των δύο δειγμάτων. Αν δώσουμε ως εισερχόμενα δεδομένα μόνο ένα διάνυσμα `x`, τότε εφαρμόζεται (αυτόματα) το $t-test$ για ένα δείγμα. Το $t-test$ για δύο δείγματα εφαρμόζεται, αν δώσουμε ως εισερχόμενα δεδομένα ένα διάνυσμα ως `x` και ένα ως `y`. Η εφαρμογή ενός t-test για ζευγάρια τιμών γίνεται με το όρισμα `paired=TRUE` και με προαπαιτούμενο ότι τα εισερχόμενα διανύσματα `x` και `y` έχουν το ίδιο μήκος.

Οι τιμές της μηδενικής υπόθεσης H_0 καθορίζονται από το όρισμα μ , το οποίο μας δίνει τις τιμές μ_0 και μ_Δ για ένα ή δύο δείγματα, αντίστοιχα. Όσον αφορά το έλεγχο για δύο ανεξάρτητα δείγματα, μπορούμε να επιλέξουμε την ισότητα ή όχι των διακυμάνσεων με το όρισμα `var.equal`, το οποίο έχει προκαθορισμένη τιμή `FALSE`.

8.3.3.1 Έλεγχος t για ένα δείγμα

Όπως είπαμε και παραπάνω, για να εφαρμόσουμε ένα t-test για τον έλεγχο μέσου, πρέπει να δώσουμε ως εισερχόμενα δεδομένα μόνο ένα αριθμητικό διάνυσμα. Η γενική σύνταξη σε αυτή την περίπτωση είναι:

```
t.test(x, mu=0, alternative="two.sided")                          ■,
```

και εκτελεί τον έλεγχο

$$H_0 : \mu = \text{mu} \text{ έναντι της εναλλακτικής } H_1 : \mu \neq \text{mu} \ .$$

Έτσι, στο παράδειγμα:

```
> x0<-c(12,10,6,7,5,13,14,11,18)
> mean(x0)
[1] 10.66667
> t.test(x0, mu=7)

        One Sample t-test

data:  x0
t = 2.6295, df = 8, p-value = 0.0302
alternative hypothesis: true mean is not equal to 7
95 percent confidence interval:
  7.451098 13.882236
sample estimates:
mean of x
 10.66667                                                    ■,
```

εφαρμόζουμε τον έλεγχο $H_0 : \mu = 7$ έναντι της εναλλακτικής $H_1 : \mu \neq 7$. Η τιμή της ελεγχοσυνάρτησης $t = \sqrt{n}(x - \overline{x})/S_x$ στο δείγμα είναι ίση με 2.63 ενώ οι βαθμοί ελευθερίας είναι 8 ($df = n-1 = 9-1 = 8$). Το p-value$= 0.03 < 0.05$, άρα απορρίπτουμε την μηδενική υπόθεση για 5% επίπεδο σημαντικότητας. Επιπλέον δίνεται η τιμή του δειγματικού μέσου, ίση εδώ με 10.67, και το 95% διάστημα εμπιστοσύνης $-$ $(7.45, 13.88)$ στο συγκεκριμένο παράδειγμα. Αντίθετα, δεν απορρίπτουμε την μηδενική υπόθεση $H_1 : \mu = 10$ έναντι της εναλλακτικής $H_1 : \mu \neq 10$ σύμφωνα με τα ακόλουθα αποτελέσματα:

```
> t.test(x0, mu=10)

        One Sample t-test

data:  x0
t = 0.4781, df = 8, p-value = 0.6454
alternative hypothesis: true mean is not equal to 10
95 percent confidence interval:
  7.451098 13.882236
sample estimates:
mean of x
 10.66667                                                    ■.
```

8.3.3.2 Έλεγχος t για δύο ανεξάρτητα δείγματα με άνισες διακυμάνσεις

Από τη στιγμή που θα ορίσουμε δύο διανύσματα ως εισερχόμενα δεδομένα, τότε η συνάρτηση `t.test` εκτελεί ένα t-test για τη διαφορά των μέσων σε δύο δείγματα, υποθέτοντας διαφορετικές διακυμάνσεις για κάθε δείγμα. Η γενική σύνταξη σε αυτή την περίπτωση είναι:

```
t.test(x, y, mu=0, alternative="two.sided")
```
■,

και εκτελεί τον έλεγχο:

$$H_0 : \mu_x - \mu_y = \text{mu} \text{ έναντι της εναλλακτικής } H_1 : \mu_x - \mu_y \neq \text{mu} \ . \tag{8.1}$$

Ο πιο συνηθισμένος έλεγχος είναι αυτός της ισότητας των δύο μέσων (δηλαδή για την προκαθορισμένη τιμή `mu=0`), που γίνεται έλεγχος

$$H_0 : \mu_x = \mu_y \text{ έναντι της εναλλακτικής } H_1 : \mu_x \neq \mu_y \ .$$

γράφοντας απλά:

```
t.test(x, y)
```
■.

΄Ετσι, στο ακόλουθο παράδειγμα:

```
> x<-c(12,10,6,7,5,13,14,11,18)
> y<-c(9,8,5,4,3,6,7,12,14,15,17,7,8)
> length(x);length(y)
[1] 9
[1] 13
> t.test(x,y)

        Welch Two Sample t-test

data:  x and y
t = 0.9849, df = 17.871, p-value = 0.3378
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -2.064979  5.706004
sample estimates:
mean of x mean of y
10.666667  8.846154
```

έχουμε δύο δείγματα με μέγεθος δείγματος 9 και 13 παρατηρήσεις. Οι δειγματικοί μέσοι είναι ίσοι με 10.67 και 8.84, αντίστοιχα, ενώ το 95% διάστημα εμπιστοσύνης είναι ίσο με $(-2.065, 5.706)$ (συμπεριλαμβάνει την υπο-έλεγχο μηδενική τιμή). Η συνάρτηση εφαρμόζει τον Welch t-test για δύο ανεξάρτητα δείγματα με άνισες διακυμάνσεις, με παρατηρούμενη τιμή της ελεγχοσυνάρτησης ίση με 0.985, βαθμούς ελευθερίας 17.87 και p-value$= 0.338$, με βάση το οποίο δεν μπορούμε να απορρίψουμε την H_0 για επίπεδο 5% ή 10%.

Αντίθετα, η εντολή:

```
> t.test(x,y,alternative="less",conf.level=0.90)
```

```
        Welch Two Sample t-test

data:  x and y
t = 0.9849, df = 17.871, p-value = 0.8311
alternative hypothesis: true difference in means is less than 0
90 percent confidence interval:
     -Inf 4.280372
sample estimates:
mean of x mean of y
10.666667  8.846154
```

ελέγχει ξανά για την ισότητα των δύο μέσων, αλλά με την εναλλακτική $H_1 : \mu_x < \mu_y$, ενώ επίσης, δίνει το μονόπλευρο 90% διάστημα εμπιστοσύνης, το οποίο είναι ίσο με $(-\infty, 4.28)$.

8.3.3.3 Έλεγχος t για δύο ανεξάρτητα δείγματα με ίσες διακυμάνσεις

Ο έλεγχος που, συνήθως, διδασκόμαστε στα αρχικά μαθήματα στατιστικής συμπερασματολογίας, αναφέρεται στο $t - test$ για δύο ανεξάρτητα δείγματα, αλλά με την προϋπόθεση των ίσων διακυμάνσεων για τα δύο δείγματα. Ο λόγος είναι, ότι η κατανομή αυτής της ελεγχο-συνάρτησης είναι εύκολα υπολογίσιμη, ενώ υπάρχει άμεση ερμηνεία των βαθμών ελευθερίας σε σχέση με τις παραμέτρους που εκτιμούμε. Η γενική σύνταξη σε αυτή την περίπτωση είναι:

```
t.test(x, y, mu=0, alternative="two.sided", var.equal=TRUE)
```
■,

και εκτελεί τον έλεγχο (8.1), υπό την προϋπόθεση ότι εκτιμάται μια κοινή διακύμανση σ^2 από τον εκτιμητή

$$S^2 = \frac{(n_1-1)S_x^2+(n_2-1)S_y^2}{n_1+n_2-2}$$

ο οποίος ονομάζεται και κοινός εκτιμητής διακύμανσης (pooled variance estimator)• όπου S_x^2 και S_y^2 οι δειγματικές διασπορές των τυχαίων μεταβλητών X και Y, αντίστοιχα. Όπως είναι κατανοητό, η σύνταξη είναι παρόμοια με την περίπτωση των άνισων διακυμάνσεων, με την επιπλέον δήλωση των ίσων διακυμάνσεων, μέσω του ορίσματος `var.equal=TRUE`. Αντίστοιχα, ο συνηθισμένος έλεγχος είναι αυτός της ισότητας των δύο μέσων για δύο δείγματα με ίσες διακυμάνσεις, δίνεται γράφοντας απλά:

```
t.test(x, y, var.equal=TRUE )
```
■.

Έτσι, στο προηγούμενο παράδειγμα βλέπουμε ότι:

```
> var(x)
[1] 17.5
> var(y)
[1] 19.14103

> t.test(x,y,var.equal=T)
```

```
        Two Sample t-test

data:  x and y
t = 0.9765, df = 20, p-value = 0.3405
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -2.068419  5.709444
sample estimates:
mean of x mean of y
10.666667  8.846154                                                    ■,
```

δηλαδή εφαρμόζοντας το t-test με ίσες διακυμάνσεις, οι βαθμοί ελευθερίας τώρα είναι ίσοι με 20 ($n_1 + n_2 - 2 = 9 + 13 - 2$) και το p-value είναι τώρα ίσο με 0.34, αντί για 0.338 που ήταν στην περίπτωση των άνισων διακυμάνσεων. Το τελικό συμπέρασμα παραμένει το ίδιο, ότι δεν απορρίπτουμε τη μηδενική υπόθεση της ισότητας των μέσων για επίπεδο $\alpha = 5\%$ ή $\alpha = 10\%$.

8.3.3.4 Έλεγχος t για δύο εξαρτημένα δείγματα (paired t-test)

Στην περίπτωση που έχουμε δύο εξαρτημένα δείγματα, δηλαδή ζευγάρια τιμών (x_i, y_i) που έχουν κάποια μορφή εξάρτησης, τότε πρέπει να εφαρμόσουμε τον έλεγχο t για εξαρτημένα δείγματα (paired t-test). Εφόσον αναφερόμαστε σε ζευγάρια τιμών, προϋπόθεση για εφαρμόσουμε τον έλεγχο είναι, να έχουμε δύο διανύσματα με ίδιο μήκος (μέγεθος δείγματος) n. Επιπλέον, κάθε στοιχείο του `x` πρέπει να ταιριάζει με το αντίστοιχο στοιχείο του `y`, δηλαδή να έχουμε ζευγάρια τιμών (x_i, y_i) (`x[i], y[i]`, στην R), που να αντιστοιχούν στο ίδιο άτομο (ή γενικότερα στη μονάδα που μελετάμε).

Η σύνταξη τώρα είναι παρόμοια με αυτή για δύο ανεξάρτητα δείγματα, με την προσθήκη του ορίσματος `paired=TRUE`, που προσδιορίζει αν θα γίνει ο έλεγχος για εξαρτημένα ή ανεξάρτητα δείγματα:

```
    t.test(x, y, mu=0, alternative="two.sided", paired=TRUE)    ■.
```

Όσον αφορά τον έλεγχο, ενώ έχει τη μορφή 8.1 στην πραγματικότητα για να λάβουμε υπόψη μας την εξάρτηση και να την αφαιρέσουμε, υπολογίζουμε τη διαφορά $\Delta = X - Y$ και εφαρμόζουμε ένα $t - test$ για ένα δείγμα για το μέσο της διαφοράς μ_Δ, δηλαδή:

$$H_0 : \mu_\Delta = \text{mu} \text{ έναντι της εναλλακτικής } H_1 : \mu_\Delta \neq \text{mu} .$$

Έτσι, λοιπόν, η σύνταξη:

```
> t.test(x,y,paired=T)
Error in complete.cases(x, y) : not all arguments have the same
   length
```

μας δίνει μήνυμα λάθους, λόγω του διαφορετικού μεγέθους δείγματος (μήκους) των δύο δειγμάτων (διανυσμάτων που εισάγουμε). Αντίθετα, στο παρακάτω παράδειγμα προσομοιώσαμε δύο τυχαία διανύσματα `x1` και `y1` μήκους 30 ($n = 30$), με συσχέτιση ίση με 0.97 και ελέγξαμε αν οι μέσοι τους είναι ίσοι.

```
> x1<- round( rnorm(10), 2)
> y1<-2+3*x1 + round(rnorm(10,0,1),2 )
> dput(x1)
c(-1.58, 0.86, 0.33, 1.25, -0.97, -0.57, -1.17, 0.83, -0.22, 0.96)
> dput(y1)
c(-2.7, 4.77, 3.62, 6.21, -1.85, -1.21, -3.35, 3.06, 1.03, 5.74)
> cor(x1,y1)
[1] 0.97216
>  t.test(x1,y1,paired=T)

        Paired t-test

data:  x1 and y1
t = -1.8653, df = 9, p-value = 0.09501
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -3.4519413  0.3319413
sample estimates:
mean of the differences
                  -1.56                                              ■.
```

Όπως βλέπουμε, η μηδενική υπόθεση δεν απορρίπτεται ($p - value = 0.095 > 0.05$) για $\alpha = 5\%$ και, συνεπώς, δεν υπάρχει διαφορά στους μέσους μεταξύ των δύο μετρήσεων. Αν τώρα τρέξουμε το t-test για ένα δείγμα για την διαφορά `x1-y1`, βλέπουμε ότι τα αποτελέσματα είναι ακριβώς ίδια.

```
> t.test(x1-y1)

        One Sample t-test

data:  x1 - y1
t = -1.8653, df = 9, p-value = 0.09501
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 -3.4519413  0.3319413
sample estimates:
```

```
mean of x
    -1.56                                                    ■.
```

Τέλος, αν στα συγκεκριμένα δεδομένα τρέξουμε (λανθασμένα) ένα t-test για ανεξάρτητα δείγματα:

```
> t.test(x1,y1, paired=F)

        Welch Two Sample t-test

data:  x1 and y1
t = -1.3137, df = 10.397, p-value = 0.2172
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -4.192295  1.072295
sample estimates:
mean of x mean of y
   -0.028     1.532
```

τότε βλέπουμε ότι η διαφορά των μέσων μένει ίδια ($-0.028 - 1.532 = -1.56$), όμως όλες οι υπόλοιπες τιμές αλλάζουν, λόγω της ανεξαρτησίας που υποθέτουμε και της συσχέτισης που παραβλέπουμε. Έτσι, τόσο τα τυπικά σφάλματα των εκτιμητών, όσο και η ελεγχοσυνάρτηση, η αντίστοιχη κατανομή της και το p-value (που τώρα είναι 0.217 αντί για 0.095) είναι διαφορετικά.

8.3.4 Έλεγχος για το λόγο δύο διακυμάνσεων

Ο έλεγχος ισότητας των διακυμάνσεων δύο δειγμάτων (ή ομάδων) γίνεται με την εντολή `var.test`. Η γενική σύνταξη είναι της μορφής:

```
var.test(x, y, ratio = 1, alternative = "two.sided",
                          conf.level = 0.95, ...)                ■,
```

όπου `x` και `y` είναι τα δεδομένα των δύο διαφορετικών δειγμάτων. Με την εντολή αυτή (και με τις προκαθορισμένες τιμές) ελέγχουμε την υπόθεση

$$H_0 : \sigma_x^2 = \sigma_y^2 \text{ έναντι της εναλλακτικής } H_1 : \sigma_x^2 \neq \sigma_y^2$$

αλλά μπορούμε γενικότερα να κάνουμε τον έλεγχο

$$H_0 : \frac{\sigma_x^2}{\sigma_y^2} = \texttt{ratio} \text{ έναντι της εναλλακτικής } H_1 : \frac{\sigma_x^2}{\sigma_y^2} \neq \texttt{ratio}$$

αλλάζοντας την τιμή του ορίσματος `ratio`.

Έτσι λοιπόν, για το απλό παράδειγμα της Ενότητας 8.3.3.2 έχουμε:

```
> x<-c(12, 10, 6, 7, 5, 13, 14, 11, 18)
> y<-c(9, 8, 5, 4, 3, 6, 7, 12, 14, 15, 17, 7, 8)
```

```
> var(x)
[1] 17.5
> var(y)
[1] 19.14103
> var.test(x,y)

        F test to compare two variances

data:  x and y
F = 0.9143, num df = 8, denom df = 12, p-value = 0.9272
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.260343 3.839616
sample estimates:
ratio of variances
         0.9142666                                                ■,
```

δηλαδή p-values = $0.927 > 0.05$, άρα δεν απορρίπτουμε τη μηδενική υπόθεση για την ισότητα των δύο διακυμάνσεων σε επίπεδο σημαντικότητας 5% (το ίδιο ισχύει και για 10%). Επιπλέον δίνονται η δειγματική τιμή της ελεγχοσυνάρτησης $F = 0.91$, οι αντίστοιχοι βαθμοί ελευθερίας (8 και 12) το 95% διάστημα εμπιστοσύνης για το λόγο των διακυμάνσεων $(0.26, 3.84)$ και η σημειακή εκτίμηση του λόγου των διακυμάνσεων (ίση με 0.914).

Να σημειώσουμε, ότι ο έλεγχος αυτός χρησιμοποιείται βοηθητικά για να αποφασίσουμε πιο t-test θα χρησιμοποιήσουμε: με άνισες (Ενότητα 8.3.3.3) ή ίσες διακυμάνσεις (Ενότητα 8.3.3.2).

8.3.5 Έλεγχοι κανονικότητας

Στην R υπάρχει μεγάλη ποικιλία συναρτήσεων που μπορούμε να εφαρμόσουμε για την υλοποίηση των ελέγχων υποθέσεων. Οι πιο συνηθισμένοι έλεγχοι είναι το Shapiro-Wilk τεστ (Shapiro & Wilk, 1965) και το Kolmogorov-Smirnov τεστ (Kolmogorov, 1933, Smirnov, 1948) με τη διόρθωση του Lilliefors ή απλά Lilliefors τεστ κανονικότητας (Lilliefors, 1967).

Ο έλεγχος Shapiro-Wilk μπορεί να εφαρμοστεί με την σύνταξη:

```
     shapiro.test(x)                                              ■,
```

η οποία δίνει ως εξερχόμενες τιμές τα αντικείμενα `statistic`, `p.value`, `method` και `data.name`. Το p-value υπολογίζεται ασυμπτωτικά, το οποίο σύμφωνα με τον Royston (1995) είναι αρκετά ικανοποιητικό για τιμές μικρότερες του 0.1. Σύμφωνα με το αρχείο βοηθείας της R, η συνάρτηση είναι μετάφραση του προγράμματος γλώσσας C του Royston (1995), που είναι διαθέσιμο στην ιστοσελίδα `http://lib.stat.cmu.edu/apstat/R94`. Έτσι για παράδειγμα, το παρακάτω διάνυσμα:

```
> z<-rnorm(20)
```

```
> z<-round(z,1)
 [1]  0.2  1.9  1.0 -0.6  0.7 -0.6 -1.7 -1.4 -1.9  0.2 -0.4 -1.0
    -2.2  0.0 -1.0 -2.3  1.7 -1.7  0.3  0.9
> dput(round(z,1))
c(0.2, 1.9, 1, -0.6, 0.7, -0.6, -1.7, -1.4, -1.9, 0.2, -0.4,
-1, -2.2, 0, -1, -2.3, 1.7, -1.7, 0.3, 0.9)                    ■,
```

το οποίο έχει 20 παρατηρήσεις από την τυποποιημένη κανονική κατανομή στρογγυλοποιημένες με ένα δεκαδικό ψηφίο, εντοπίζεται σωστά από το Shapiro-Wilk test, εφόσον δεν απορρίπτεται η κανονικότητα με p-value$= 0.6$:

```
> shapiro.test(z)

        Shapiro-Wilk normality test

data:  z
W = 0.9629, p-value = 0.6025                                   ■.
```

Όσον αφορά τον έλεγχο Lilliefors, αυτός είναι διαθέσιμος στη βιβλιοθήκη nortest που περιέχει ελέγχους κανονικότητας. Μπορούμε να τρέξουμε τον συγκεκριμένο έλεγχο με τη σύνταξη:

```
                lillie.test(x)                                 ■,
```

και παίρνουμε ως αποτέλεσμα ίδιου τύπου αντικείμενα με αυτά της συνάρτησης `shapiro.test`. Πρέπει να προσέξουμε και να προτιμήσουμε αυτή τη συνάρτηση, αντί της εντολής
`ks.test(x, "pgamma", mu=0, sd=1)`, γιατί αυτή εφαρμόζει το αρχικό Kolmogorov-Smirnov έλεγχο, ο οποίος υποθέτει ότι η μέση τιμή και η διακύμανση είναι γνωστές και δεν εκτιμώνται από το δείγμα, σε αντίθεση με τον έλεγχο του Lilliefors (1967) που διόρθωσε την κατανομή της ελεγχο-συνάρτησης για τις περιπτώσεις που εκτιμούμε τις παραμέτρους από το δείγμα. Να σημειώσουμε, ότι ο έλεγχος Kolmogorov–Smirnov και η ασυμπτωτική κατανομή της ελεγχο-συνάρτησης του βρέθηκε από τον Kolmogorov (1933), ενώ αργότερα ο Smirnov (1948) δημοσίευσε την πινακοποίηση της κατανομής της ελεγχο-συναρτησης.

Τέλος, να αναφέρουμε ότι στη βιβλιοθήκη nortest είναι διαθέσιμοι και οι έλεγχοι Anderson-Darling με την εντολή `ad.test`, Cramer-von Mises με την εντολή `cvm.test`, χ^2 του Pearson με την εντολή `pearson.test`, και Shapiro-Francia με την εντολή `sf.test`.

Σύμφωνα με την μελέτη προσομοίωσης των Razali & Wah (2011), ο έλεγχος Shapiro–Wilk έχει μεγαλύτερη ισχύ για δεδομένο επίπεδο σημαντικότητας, ακολουθούμενος από το Anderson–Darling test, ενώ χειρότερος είναι τόσο ο έλεγχος του Lilliefors, όσο και ο αρχικός των Kolmogorov-Smirnov. Ο έλεγχος του Lilliefors (Kolomorov-Smirnov) είναι ο πιο γνωστός και ευρέως χρησιμοποιούμενος έλεγχος κανονικότητας, αλλά είναι χειρότερος από τους Shapiro–Wilk, Anderson–Darling και Cramer-von Mises. Ο έλεγχος Anderson-Darling έχει αναφερθεί ως καλύτερος του Cramer-von Mises από τον Stephens (1986). Ο έλεγχος του χ^2 του Pearson είναι ο

αρχαιότερος, αλλά δεν προτείνεται λόγω της μικρής του ισχύος (ειδικά σε σχέση με τους άλλου, νεότερους, έλεγχους) και λόγω της περίφημης ευαισθησίας του στην επιλογή των διαστημάτων, στα οποία θα γίνουν οι συγκρίσεις μεταξύ αναμενόμενων και παρατηρούμενων συχνοτήτων. Τέλος, ο έλεγχος Shapiro-Francia έχει αναφερθεί ως ένας έλεγχος με καλή συμπεριφορά από το Royston (1993), αλλά δεν έχει αναφερθεί ιεράρχηση όσον αφορά την ισχύ του, σε σχέση με τους άλλους ελέγχους κανονικότητας.

8.3.6 Μη-παραμετρικός έλεγχος του Wilcoxon για διάμεσους

Το τεστ του Wilcoxon (1945) (συνάρτηση `wilcox.test`), ο αντίστοιχος μη παραμετρικός έλεγχος των t-test (άρα και της εντολής `t.test`) και έχει γενική σύνταξη:

```
wilcox.test(x, y = NULL, alternative = "two.sided",
            mu = 0, paired = FALSE, exact = NULL, correct = TRUE,
            conf.int = FALSE, conf.level = 0.95, ...)          ■.
```

Όπως βλέπουμε, η σύνταξη της συνάρτησης `wilcox.test` είναι ανάλογη του `t.test`. Αν δώσουμε ως εισερχόμενο μόνο ένα διάνυσμα `x`, τότε εφαρμόζεται ο έλεγχος για ένα δείγμα, ενώ αν δώσουμε δύο διανύσματα (`x` και `y`), τότε θα εφαρμόσουμε τους ελέγχους για δύο δείγματα. Στη δεύτερη περίπτωση μπορούμε να κάνουμε έλεγχο για δύο ανεξάρτητα δείγματα (προκαθορισμένη επιλογή) ή για δύο εξαρτημένα δείγματα (ζευγάρια τιμών), ορίζοντας `paired=TRUE`. Όλοι αυτοί οι έλεγχοι αφορούν τη διάμεσο, σε αντίθεση με τα t-test, όπου οι έλεγχοι αφορούσαν το μέσο.

΄Ενα γενικότερο σχόλιο για τους μη παραμετρικούς ελέγχους είναι ότι χτίζονται, χρησιμοποιώντας τη διάταξη των παρατηρήσεων (δηλαδή τις τάξεις - ranks). Για το λόγο αυτό, χρησιμοποιούν λιγότερη πληροφορία από τα παραμετρικά, που χρησιμοποιούν τις πραγματικές τιμές των δεδομένων, καταλήγοντας σε ελέγχους με μικρότερη ισχύ, που στην πράξη σημαίνει, ότι οι πραγματικές διαφορές (αν υπάρχουν) μπορούν να εντοπιστούν δυσκολότερα (με μικρότερη συχνότητα). Φυσικά, το πλεονέκτημα αυτών των ελέγχων είναι ότι δε χρειάζονται καμία υπόθεση για την αρχική κατανομή των δεδομένων (π.χ. κανονικότητα).

Γυρίζοντας στη συνάρτηση `wilcox.test`, το όρισμα `mu` καθορίζει την υποθετική τιμή της διαμέσου (για τον έλεγχο για ένα δείγμα) ή την υποθετική τιμή της διαφοράς των διαμέσων (για τον έλεγχο για δύο δείγματα) κάτω από την H_0. Το λογικό όρισμα `correct` ελέγχει αν θα εφαρμοστεί διόρθωση κανονικότητας (προκαθορισμένη τιμή) ή όχι, ενώ το όρισμα `exact` ορίζει αν θα γίνει ασυμπτωτικός ή ακριβής έλεγχος. Αν δεν ορίσουμε τιμή για το όρισμα `exact`, τότε η συνάρτηση εφαρμόζει ακριβή έλεγχο για δείγματα με μικρότερες από 50 διαφορετικές τιμές και ασυμπτωτικό διαφορετικά.

Πιο συγκεκριμένα, για τον έλεγχο για ένα δείγμα η σύνταξη είναι:

```
    wilcox.test(x, mu = 0, ...)                                ■,
```

όπου εφαρμόζεται το Wilcoxon signed rank test, με μηδενική υπόθεση ότι η κατανομή του x είναι συμμετρική γύρω από την τιμή του mu. Γενικότερα, ελέγχουμε ότι

$$H_0 : M_x = \text{mu} \text{ έναντι της εναλλακτικής } H_1 : M_x \neq \text{mu}$$

με την προϋπόθεση ότι η κατανομή του x είναι συμμετρική, όπου M_x είναι η διάμεσός του X.

Αντίστοιχη είναι η περίπτωση για τον έλεγχο για εξαρτημένα δείγματα, ο οποίος μπορεί να εφαρμοστεί με την εντολή:

```
wilcox.test(x, y, mu = 0, paired=TRUE, ...)
```

ή ισοδύναμα με την εντολή:

```
wilcox.test(x-y, mu = 0, ...)
```
■.

Στην περίπτωση αυτή, ο έλεγχος γίνεται

$$H_0 : M_{x-y} = \text{mu} \text{ έναντι της εναλλακτικής } H_1 : M_{x-y} \neq \text{mu}$$

δηλαδή ελέγχουμε για την τιμή της διαμέσου της διαφοράς $X - Y$ (και έμμεσα και για τη διαφορά των δύο διαμέσων).

Τέλος, ο έλεγχος για τη διαφορά δύο διαμέσων γίνεται με τη σύνταξη:

```
wilcox.test(x, y, mu = 0, ...)
```
■,

η οποία ελέγχει κατά πόσο οι κατανομές των X και Y είναι ίδιες (η γενικότερα με μια μετατόπιση θέσης ίση με mu), το οποίο μεταφράζεται στην υπόθεση

$$H_0 : M_x - M_y = \text{mu} \text{ έναντι της εναλλακτικής } H_1 : M_x - M_y \neq \text{mu}$$

εφαρμόζοντας τη δοκιμασία Wilcoxon rank sum (Wilcoxon, 1945), το οποίο είναι ισοδύναμο με το Mann-Whitney test (Mann & Whitney, 1947).

Έτσι, λοιπόν, για το παράδειγμα της Ενότητας 8.3.3.1 (9 παρατηρήσεις με μέσο 10.67 και διάμεσο 11) έχουμε:

```
> x0<-c(12, 10, 6, 7, 5, 13, 14, 11, 18)
> mean(x0)
[1] 10.66667
> median(x0)
[1] 11
> wilcox.test(x0, mu=7)

        Wilcoxon signed rank test with continuity correction

data:  x0
V = 33, p-value = 0.04232
```

```
alternative hypothesis: true location is not equal to 7

Warning message:
In wilcox.test.default(x0, mu = 7) :
  cannot compute exact p-value with zeroes                    ■,
```

δηλαδή p-value$= 0.042 < 0.05$, συνεπώς, απορρίπτουμε τη μηδενική υπόθεση ότι η διάμεσος τιμή είναι ίση με επτά για επίπεδο σημαντικότητας 5%. Αντίστοιχα, για το παράδειγμα της Ενότητας 8.3.3.4 έχουμε:

```
>x1<-c(-1.58, 0.86, 0.33, 1.25, -0.97, -0.57, -1.17, 0.83, -0.22,
        0.96)
>y1<-c(-2.7, 4.77, 3.62, 6.21, -1.85, -1.21, -3.35, 3.06, 1.03,
        5.74)
> wilcox.test(x1,y1, paired=T)

        Wilcoxon signed rank test

data:  x1 and y1
V = 11, p-value = 0.1055
alternative hypothesis: true location shift is not equal to 0

> wilcox.test(x1-y1)

        Wilcoxon signed rank test

data:  x1 - y1
V = 11, p-value = 0.1055
alternative hypothesis: true location is not equal to 0          ■,
```

δηλαδή δεν απορρίπτεται $(p - value = 0.105 > 0.05)$ η υπόθεση ότι οι δύο κατανομές (και διάμεσοι) είναι ίδιες για $\alpha = 0.05$.

Τέλος, για τα δεδομένα του παραδείγματος της Ενότητας 8.3.3.2 έχουμε:

```
> x<-c(12,10,6,7,5,13,14,11,18)
> y<-c(9,8,5,4,3,6,7,12,14,15,17,7,8)
> median(x)
[1] 11
> median(y)
[1] 8
> wilcox.test(x,y)
```

```
        Wilcoxon rank sum test with continuity correction

data:  x and y
W = 72, p-value = 0.3841
alternative hypothesis: true location shift is not equal to 0

Warning message:
In wilcox.test.default(x, y) : cannot compute exact p-value with
   ties                                                          ■,
```

συνεπώς, δεν απορρίπτεται η μηδενική υπόθεση (p-value$= 0.384 > 0.05$), ότι οι δύο κατανομές (και διάμεσοι) είναι ίδιες για $\alpha = 0.05$.

Παρατήρηση: Στο παραπάνω παράδειγμα παίρνουμε σαν προειδοποιητικό μήνυμα

```
In wilcox.test.default(x, y) : cannot compute exact p-value with
   ties                                                          ■.
```

Αυτό σημαίνει ότι η R μας προειδοποιεί ότι έχουμε δύο ίδιες τιμές στα δεδομένα μας, οι οποίες λέγονται ισοπαλίες (ties). Συνεπώς τα ranks δεν είναι πλέον μοναδικά και δεν μπορεί να υπολογιστεί το ακριβές p-values. Το πως εξαλείφουμε αυτές τις ισοπαλίες, υπάρχουν διάφοροι τρόποι, με τους οποίους δε θα ασχοληθούμε περαιτέρω εδώ.

Γενικά, οι μη-παραμετρικοί έλεγχοι του Wilcoxon χρησιμοποιούνται εναλλακτικά των αντίστοιχων t-test, στην περίπτωση που η υπόθεση της κανονικότητας παραβιάζεται.

8.4 Μονο-παραγοντική Ανάλυση Διακύμανσης: ΄Ελεγχος ισότητας πολλών μέσων

Η ανάλυση διακύμανσης, κατά ένα παράγοντα, (one-way analysis of variance - ANOVA) χρησιμοποιείται για τον έλεγχο της ισότητας πολλών μέσων και θεωρείται γενίκευση του t-test που αναφέρεται σε δύο δείγματα. Η γενική σύνταξη είναι η ακόλουθη:

```
    oneway.test(y~x, var.equal = FALSE)                            ■,
```

όπου `y` είναι μια ποσοτική μεταβλητή και `x` είναι η αντίστοιχη κατηγορική μεταβλητή (με κ επίπεδα), με βάση την οποία διαχωρίζονται οι ομάδες (ή τα δείγματα). Η σύνταξη `y~x` ονομάζεται formula στην R και χρησιμοποιείται για τον ορισμό στατιστικών μοντέλων, κυρίως και δευτερεύοντος για κάποιους ελέγχους υποθέσεων• βλ. Ενότητα 8.6 για λεπτομέρειες.

Γενικά, η ανάλυση διακύμανσης ελέγχει την υπόθεση

$$H_0 : \mu_1 = \mu_2 = \cdots = \mu_\kappa \text{ έναντι της εναλλακτικής } H_1 : \mu_i \neq \mu_j \text{ για κάποια } i \neq j \ .$$

Ο κλασσικός έλεγχος της ανάλυσης διακύμανσης προϋποθέτει ίσες διακυμάνσεις (πρέπει να ορίσουμε `var.equal = TRUE`), ενώ εδώ δίνεται η δυνατότητα ελέγχου χωρίς ίσες διακυμάνσεις

(προκαθορισμένη επιλογή). Επίσης, η εφαρμογή αυτού του ελέγχου προϋποθέτει κανονικότητα για τα δεδομένα κάθε ομάδας, ή εναλλακτικά, (και συνήθως αυτό ελέγχουμε) κανονικότητα των κατάλοιπων ϵ_i, που υπολογίζονται ως τη διαφορά της κάθε τιμης y_i από τη μέση τιμή της ομάδας στην οποία ανήκουν.

΄Ετσι, στο ακόλουθο παράδειγμα παίρνουμε τα δεδομένα του πλαισίου δεδομένων `PlantGrowth` της βιβλιοθήκης `datasets`, το οποίο έχει δύο μεταβλητές: το βάρος (weight) φυτών και τη μέθοδο καλλιέργειας που ακολουθήθηκε (η ομάδα ελέγχου και δύο διαφορετικές καταστάσεις θεραπείας με λιπάσματα)

```
> head(PlantGrowth )
  weight group
1   4.17  ctrl
2   5.58  ctrl
3   5.18  ctrl
4   6.11  ctrl
5   4.50  ctrl
6   4.61  ctrl
> str(PlantGrowth )
'data.frame':   30 obs.  of  2 variables:
 $ weight: num
    4.17 5.58 5.18 6.11 4.5 4.61 5.17 4.53 5.33 5.14 ...
 $ group : Factor w/ 3 levels "ctrl","trt1"
  ,..: 1 1 1 1 1 1 1 1 1 1 ...
> summary(PlantGrowth )
     weight       group
 Min.   :3.590   ctrl:10
 1st Qu.:4.550   trt1:10
 Median :5.155   trt2:10
 Mean   :5.073
 3rd Qu.:5.530
 Max.   :6.310                                                    ■.
```

Άρα, λοιπόν, εδώ μας ενδιαφέρει να ελέγξουμε την υπόθεση

$$H_0 : \mu_1 = \mu_2 = \mu_3 \text{ έναντι της εναλλακτικής } H_1 : \mu_i \neq \mu_j \text{ για κάποια } i \neq j \in \{1, 2, 3\} \ ,$$

όπου μ_1 είναι το μέσο βάρος στην ομάδα ελέγχου και μ_2 και μ_3 είναι το μέσο βάρος στις ομάδες λιπασμάτων 1 και 2.

Πριν προχωρήσουμε, ας δούμε αν οι διακυμάνσεις είναι παρόμοιες στις τρεις ομάδες (χωρίς όμως να εφαρμόσουμε κάποιο έλεγχο). Αυτό μπορούμε να το κάνουμε εύκολα με την εντολή `by` (ή εναλλακτικά με συνδιασμό των εντολών `split` και `lapply`). Συνεπώς, έχουμε:

```
> by(PlantGrowth$weight,PlantGrowth$group, sd)
PlantGrowth$group: ctrl
[1] 0.5830914
--------------------------
PlantGrowth$group: trt1
[1] 0.7936757
--------------------------
PlantGrowth$group: trt2
[1] 0.4425733                                                    ■,
```

από το οποίο προκύπτει ότι οι διακυμάνσεις είναι αρκετά διαφοροποιημένες (πιο «σωστό» τρόπο με χρήση στατιστικού ελέγχου σημαντικότητας θα δούμε στην Ενότητα 8.4.1). Εφαρμόζοντας τον έλεγχο για άνισες διακυμάνσεις, έχουμε τα ακόλουθα αποτελέσματα:

```
> oneway.test(PlantGrowth$weight~PlantGrowth$group)

        One-way analysis of means (not assuming equal variances)

data:  PlantGrowth$weight and PlantGrowth$group
F = 5.181, num df = 2.000, denom df = 17.128, p-value = 0.01739  ■,
```

δηλαδή απορρίπτουμε τη μηδενική υπόθεση για ισότητα του μέσου βάρους για τις τρεις διαφορετικές προσεγγίσεις/μεθόδους καλλιέργειας (p-value$= 0.017 < 0.05$) για επίπεδο σημαντικότητας 5%. Συνεπώς, υπάρχει τουλάχιστον μία μέθοδος που διαφοροποιείται από τις άλλες. Χρησιμοποιώντας ξανά την εντολή `by`, μπορούμε να δούμε τους μέσους ανά ομάδα:

```
> by(PlantGrowth$weight,PlantGrowth$group, mean)
PlantGrowth$group: ctrl
[1] 5.032
-----------------------------
PlantGrowth$group: trt1
[1] 4.661
-----------------------------
PlantGrowth$group: trt2
[1] 5.526                                                        ■,
```

και να ιεραρχήσουμε τις ομάδες. Άρα, η καλύτερη μεθοδολογία φαίνεται να είναι η δεύτερη θεραπεία, ενώ η χειρότερη είναι η πρώτη θεραπεία. Προσοχή, απορρίπτοντας τη μηδενική υπόθεση στον παραπάνω έλεγχο, δεν έχουμε πληροφορία για το ποια ομάδα ή ποιες ομάδες διαφέρουν. Και φυσικά, ούτε με την παραπάνω απλοϊκή (περιγραφική) ανάλυση υπολογισμού των μέσων δεν μπορούμε να καταλήξουμε σε ανάλογο συμπέρασμα. Για να το πετύχουμε αυτό, θα πρέπει να κάνουμε π.χ. ελέγχους πολλαπλών συγκρίσεων (βλ. Ενότητα 8.4.2), οι οποίοι διορθώνουν τα p-value ανάλογα με τον αριθμό των ανά-δύο συγκρίσεων των ομάδων. Χρήσιμη

είναι και η απεικόνιση των αντίστοιχων διαστημάτων εμπιστοσύνης των διαφορών των μέσων για όλες τις δυνατές συγκρίσεις των ομάδων.

Εδώ να σημειώσουμε ότι, αντί να γράψουμε `oneway.test(PlantGrowth$weight ~PlantGrowth$group)`, μπορούμε να χρησιμοποιήσουμε τη σύνταξη:

```
      oneway.test(weight~group, data=PlantGrowth)                ■,
```

δηλαδή να ορίσουμε το πλαίσιο δεδομένων με το όρισμα `data` και στον ορισμό των μεταβλητών να παραλείψουμε το όνομα του πλαισίου δεδομένων. Αυτό είναι χαρακτηριστικό της σύνταξης formula (δηλαδή ο συνδυασμός τους με το όρισμα `data`) και θα το εξηγήσουμε πιο αναλυτικά στην Ενότητα 8.6.

Τέλος να αναφέρουμε, ότι με την εντολή `oneway.test` δεν μπορούμε να πάρουμε τον παραδοσιακό πίνακα της ανάλυσης διακύμανσης. Αυτό μπορεί να γίνει με την εντολή `aov`, που είναι πιο γενική και επιτρέπει πολυ-παραγοντικά μοντέλα ανάλυσης διακύμανσης (τα οποία όμως δε θα εξηγήσουμε εδώ), σε συνδυασμό με την εντολή `summary`. Έτσι, αν γράψουμε:

```
> summary(aov(PlantGrowth$weight~PlantGrowth$group))
                   Df Sum Sq Mean Sq F value Pr(>F)
PlantGrowth$group   2  3.766  1.8832   4.846 0.0159 *
Residuals          27 10.492  0.3886
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

θα πάρουμε τον παραδοσιακό πίνακα ανάλυσης διακύμανσης, που αντιστοιχεί στα αποτελέσματα της εντολή `oneway.test`, αλλά με ίσες διακυμάνσεις (`var.equal=TRUE`), όπως βλέπουμε στα ακόλουθα αποτελέσματα:

```
> oneway.test(PlantGrowth$weight~PlantGrowth$group, var.equal=T)

        One-way analysis of means

data:  PlantGrowth$weight and PlantGrowth$group
F = 4.8461, num df = 2, denom df = 27, p-value = 0.01591          ■.
```

8.4.1 Έλεγχοι των προϋποθέσεων της ανάλυσης διακύμανσης

Οι προϋποθέσεις της ανάλυσης διακύμανσης είναι τρεις:

1. η ομοσκεδαστικότητα (ή ομοιογένεια ή ισότητα διακυμάνσεων),
2. η κανονικότητα των καταλοίπων, και
3. η ανεξαρτησία των παρατηρήσεων.

Στην ανάλυση των προϋποθέσεων επικεντρωνόμαστε στον έλεγχο της ομοσκεδαστικότητας και της κανονικότητας καθώς η ανεξαρτησία των παρατηρήσεων είναι (συνήθως) γνωστή από το σχεδιασμό του πειράματος.

8.4.1.1 Έλεγχοι ομοσκεδαστικότητας

Την πρώτη προϋπόθεση μπορούμε να την ελέγξουμε με την εντολή `bartlett.test` και τον αντίστοιχο έλεγχο (Bartlett, 1937) για την ισότητα των διακυμάνσεων. Πιο συγκεκριμένα, ελέγχουμε για την ορθότητα της μηδενικής υπόθεσης

$$H_0 : \sigma_1^2 = \sigma_2^2 = \cdots = \sigma_\kappa^2 \text{ έναντι της εναλλακτικής } H_1 : \sigma_i^2 \neq \sigma_j^2 \text{ για κάποια } i \neq j \ .$$

Η γενική σύνταξη είναι:

```
bartlett.test(x, g, ...)
```

ή, εναλλακτικά, με τη χρήση formula:

```
bartlett.test(x~g, data, ...)                                    ■,
```

όπου `x` είναι η ποσοτική μεταβλητή και `g` είναι η κατηγορική μας μεταβλητή, με βάση την οποία χωρίζονται οι υπό-σύγκριση ομάδες.

Έτσι, για τα δεδομένα `PlantGrowth` έχουμε τα ακόλουθα αποτελέσματα:

```
> bartlett.test(PlantGrowth$weight, PlantGrowth$group)

        Bartlett test of homogeneity of variances

data:  PlantGrowth$weight and PlantGrowth$group
Bartlett's K-squared = 2.8786, df = 2, p-value = 0.2371          ■,
```

δηλαδή δεν απορρίπτεται η υπόθεση των ίσων διακυμάνσεων (για $\alpha = 5\%$) και, συνεπώς, μπορούμε να υποθέσουμε ίσες διακυμάνσεις (p-value=$0.237 > 0.05$).

Ποιο συνηθισμένος έλεγχος είναι αυτός του Levene (1960), ο οποίος μπορεί να εφαρμοστεί με την εντολή `leveneTest`, που είναι διαθέσιμη στο πακέτο `car`. Η σύνταξη της είναι ακόλουθη:

```
leveneTest(y, group, center=mean, ...)                           ■,
```

ενώ αν στο όρισμα `center` αφήσουμε την προκαθορισμένη τιμή `center=median`

```
leveneTest(y, group, ...)                                        ■,
```

τότε θα εφαρμόσουμε το έλεγχο των Brown & Forsythe (1974), ο οποίος απλά αντικαθιστά το μέσο με τη διάμεσο στην ελεγχο-συνάρτηση του Levene (1960). Και οι δύο αυτοί έλεγχοι θεωρούνται πιο εύρωστοι από τον έλεγχο του Bartlett σε αποκλίσεις από την κανονικότητα. Ο έλεγχος Brown-Forsyth προτείνεται για μη κανονικά δεδομένα, καθώς φαίνεται να διατηρεί καλή στατιστική ισχύ. Να σημειώσουμε ότι, με την εντολή αυτή μπορούμε τόσο να χρησιμοποιήσουμε σύνταξη formula, όσο και να εισάγουμε και κατευθείαν ένα αντικείμενο `aov` (δηλαδή το αποτέλεσμα μιας ανάλυσης

διακύμανσης). Γυρίζοντας στα δεδομένα `PlantGrowth`, έχουμε τα ακόλουθα αποτελέσματα για τους δύο έλεγχους:

```
> leveneTest(PlantGrowth$weight, PlantGrowth$group)
Levene's Test for Homogeneity of Variance (center = median)
      Df F value Pr(>F)
group  2  1.1192 0.3412
      27
> leveneTest(PlantGrowth$weight, PlantGrowth$group, center=mean)
Levene's Test for Homogeneity of Variance (center = mean)
      Df F value Pr(>F)
group  2   1.237 0.3062
      27                                                    ■,
```

όπου βλέπουμε ότι δεν απορρίπτεται η υπόθεση της ομοσκεδαστικότητας, ούτε με τον έλεγχο Brown-Forsyth (p-value$= 0.34 > 0.05$), ούτε με τον έλεγχο Leven (p-value$= 0.306 > 0.05$) για επιπεδο σημαντικότητας 5%.

Τέλος, στη βιβλιοθήκη `stats`, που είναι άμεσα διαθέσιμη στην R, υπάρχουν οι εντολές `fligner.test`, `ansari.test` και `mood.test`, που εφαρμόζουν αντίστοιχους μη-παραμετρικούς ελέγχους για την ομοιογένεια των διακυμάνσεων (ο πρώτος) και γενικότερα της κλίμακας/μεταβλητότητας (οι άλλοι δύο).

8.4.1.2 Έλεγχοι κανονικότητας των καταλοίπων

Οι έλεγχοι κανονικότητας των καταλοίπων μπορούν να γίνουν με τους ελέγχους που περιγράψαμε στην Ενότητα 8.3.5. Ως κατάλοιπα, εδώ, ορίζουμε τις διαφορές των παρατηρούμενων τιμών y_i από τους μέσους της ομάδας που ανήκουν, δηλαδή $\epsilon_i = y_i - \overline{y}_{G_i}$, όπου G_i είναι η ομάδα (ή δείγμα) όπου ανήκει η παρατήρηση i, ενώ $\overline{y}_k$ για $k = 1, 2, \ldots, \kappa$ είναι ο δειγματικός μέσος της k ομάδας.

Τα κατάλοιπα είναι άμεσα διαθέσιμα μέσω της εντολής `aov`. Έτσι λοιπόν, γενικά αποθηκεύουμε τα κατάλοιπα μιας μονο-παραγοντικής ανάλυσης διακύμανσης με τη σύνταξη:

```
resid <- aov( y ~ x, ... )$residuals                    ■,
```

και μετά εφαρμόζουμε κανονικά τους ελέγχους κανονικότητας στο διάνυσμα `resid`. Για το παράδειγμα `PlantGrowth` έχουμε:

```
> resid <- aov(PlantGrowth$weight~PlantGrowth$group)$residuals
> shapiro.test(resid)

        Shapiro-Wilk normality test

data:  resid
W = 0.9661, p-value = 0.4379
> library(nortest)
```

```
> lillie.test(resid)

        Lilliefors (Kolmogorov-Smirnov) normality test

data:  resid
D = 0.1101, p-value = 0.4653
```

από όπου βλέπουμε ότι δεν απορρίπτεται η υπόθεση της κανονικότητας και με τους δύο ελέγχους (p-values ίσα με 0.438 και 0.465 για τον έλεγχο Shapiro-Wilk και Lilliefors, αντίστοιχα).

8.4.2 Έλεγχοι πολλαπλών συγκρίσεων

Όπως ήδη είπαμε, όταν απορρίπτουμε τη μηδενική υπόθεση ισότητας των μέσων στην ανάλυση διακύμανσης (ή των διαμέσων στον έλεγχο Kruskal-Wallis), τότε θέλουμε να δούμε ποιες ομάδες είναι αυτές που διαφοροποιούνται. Αυτό γίνεται εφαρμόζοντας ελέγχους ανά ζεύγη για την ισότητα των μέσων (ή των διαμέσων). Σε αυτή την περίπτωση, επειδή εφαρμόζουμε πολλαπλούς ελέγχους (για παράδειγμα για πέντε ομάδες έχουμε δέκα συγκρίσεις ανά δύο ομάδες), πρέπει να λάβουμε υπόψη μας το πλήθος των ελέγχων και να διορθώσουμε το τελικό p-value.

Ο πρώτος έλεγχος είναι το HSD (Honest Significant Differences) τεστ του Tukey (1949). Σε αυτό το τεστ, για κάθε σύγκριση ζευγαριού μέσων, χρησιμοποιείται η κατανομή τυποποιημένου εύρους (studentized range). Γενικά, η μέθοδος θεωρείται η καλύτερη για τον υπολογισμό διαστημάτων εμπιστοσύνης των διαφορών ή για μεγέθη δειγμάτων που διαφέρουν πολύ από ομάδα σε ομάδα. Στην περίπτωση που τα μεγέθη δείγματος στις υπό-σύγκριση ομάδες είναι ίσα ή σχετικά κοντά, τότε ο έλεγχος αυτός είναι λιγότερο ισχυρός από τις μεθόδους, που διορθώνουν τα p-value και εφαρμόζονται από την εντολή `pairwise.t.test`. Παρ'όλα αυτά, η απώλεια ισχύος είναι σχετικά μικρή ειδικά για μεγάλα δείγματα. Η σύνταξη του ελέγχου αυτού είναι:

```
      TukeyHSD(x, which, ordered = FALSE, conf.level = 0.95, ...)
```

■,

όπου `x` είναι τώρα ένα αντικείμενο `aov`, δηλαδή το αποτέλεσμα της ανάλυσης διακύμανσης, ενώ στο όρισμα `which` δηλώνουμε ποιες συγκρίσεις μας ενδιαφέρουν (όλες οι συγκρίσεις θα γίνουν αν δε εισάγουμε κάποια τιμή στο όρισμα αυτό).

Στο παράδειγμα `PlantGrowth` έχουμε:

```
> plant_anova<-aov( PlantGrowth$weight~PlantGrowth$group )
> TukeyHSD(plant_anova)
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = PlantGrowth$weight ~ PlantGrowth$group)

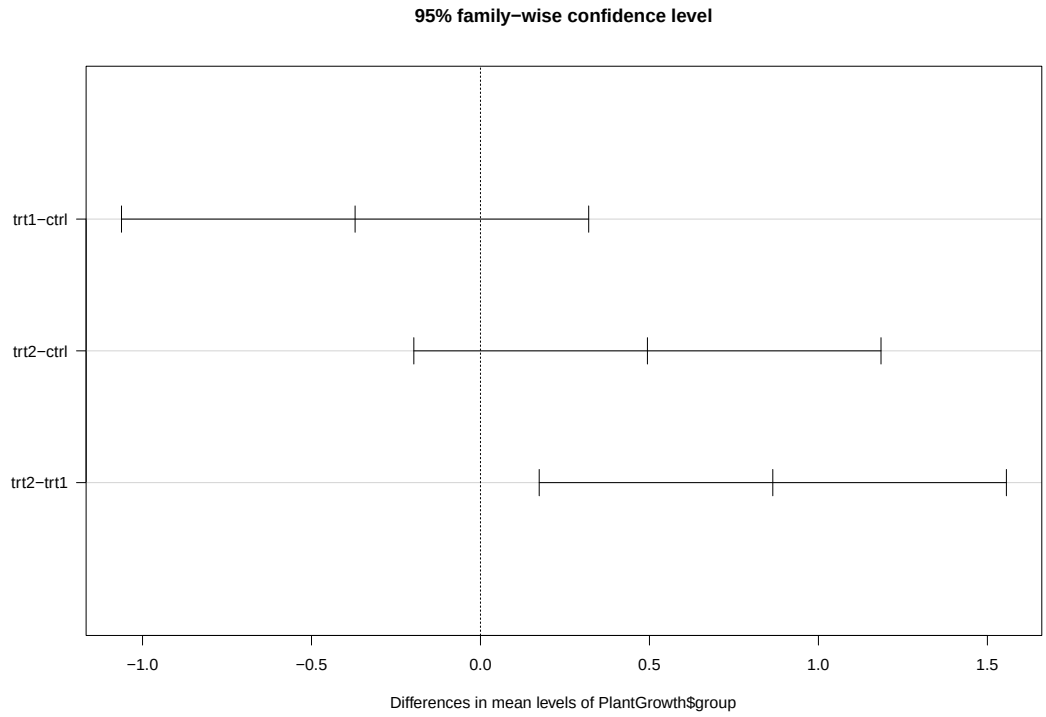
$`PlantGrowth$group`
```

```
             diff        lwr       upr     p adj
trt1-ctrl -0.371 -1.0622161 0.3202161 0.3908711
trt2-ctrl  0.494 -0.1972161 1.1852161 0.1979960
trt2-trt1  0.865  0.1737839 1.5562161 0.0120064                    ■,
```

και, συνεπώς, εντοπίζουμε μόνο μία στατιστικά σημαντική διαφορά σε 5% επίπεδο σημαντικότητας, μεταξύ της πρώτης και της δεύτερης αγωγής (p-value$= 0.012 < 0.05$). Μπορούμε επίσης, να αποθηκεύσουμε τα αποτελέσματα του ελέγχου HSD του Tukey σε ένα αντικείμενο, το οποίο με τη σειρά του μπορεί να χρησιμοποιηθεί ως εισερχόμενο στην εντολή `plot`, όπως βλέπουμε εδώ:

```
> res_tuk <- TukeyHSD(plant_anova)
> plot(res_tuk, bty='l', ylim=c(0,4), las=1)                         ■.
```

Η σύνταξη αυτή παράγει το Διάγραμμα 8.1, όπου απεικονίζονται τα διαστήματα εμπιστοσύνης για όλες τις ανά δύο συγκρίσεις. Για την τελευταία σύγκριση (treatment 2 σε σχέση με την 1), βλέπουμε ότι το αντίστοιχο διάστημα εμπιστοσύνης δεν συμπεριλαμβάνει το μηδέν, συνεπώς, οι μέσες τιμές των δύο αυτών ομάδων διαφέρουν σημαντικά.



Διάγραμμα 8.1: Διαστήματα εμπιστοσύνης για τις διαφορές των μέσων ανά δύο με τη μέθοδο HSD του Tukey

Εναλλακτικά, μπορούμε να χρησιμοποιήσουμε την εντολή `pairwise.t.test`, η οποία εφαρμόζει μια σειρά από ζευγαρωτούς ελέγχους με διόρθωση των p-values. Η σύνταξη είναι η ακόλουθη:

```
pairwise.t.test(x, g, p.adjust.method = p.adjust.methods,
                pool.sd = !paired, paired = FALSE,
                alternative = c("two.sided", "less", "greater")
  , ...)                                                    ■,
```

όπου x είναι η αριθμητική μας μεταβλητή, ενώ g είναι η κατηγορική μας μεταβλητή που καθορίζει τις υπό σύγκριση ομάδες και δίνεται έτσι η δυνατότητα να έχουμε εξαρτημένα ή ανεξάρτητα δείγματα. Το όρισμα p.adjust.method καθορίζει τη μέθοδο διόρθωσης των p-value με πιθανές επιλογές

```
c("holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr",
  "none")                                                   ■.
```

Στις παραπάνω επιλογές, η διόρθωση Bonferroni ("bonferroni") είναι αρκετά συνηθισμένη, αν και σχετικά απλή, αφού τα p-values απλά πολλαπλασιάζονται με τον αριθμό των συγκρίσεων. Η διόρθωση Bonferroni, για το παράδειγμα PlantGrowth μας δίνει

```
> pairwise.t.test(PlantGrowth$weight, PlantGrowth$group, p.adjust='
  bonferroni')

        Pairwise comparisons using t tests with pooled SD

data:  PlantGrowth$weight and PlantGrowth$group

     ctrl  trt1
trt1 0.583 -
trt2 0.263 0.013

P value adjustment method: bonferroni
>                                   ■,                      ■,
```

δηλαδή βλέπουμε ότι έχουμε διαφοροποίηση μεταξύ των δύο θεραπειών (p-value$= 0.013 < 0.05$) και όχι μεταξύ των θεραπειών και της ομάδας ελέγχου (p-values ίσα με 0.58 και 0.26 για τις δύο θεραπείες αντίστοιχα). Εδώ βλέπουμε ότι τα αποτελέσματα με τις προσεγγίσεις του Tukey και του Bonferroni συμφωνούν.

Οι υπόλοιπες διορθώσεις που είναι διαθέσιμες στην συνάρτηση pairwise.t.test είναι πιο συντηρητικές και περιλαμβάνουν τις προσεγγίσεις των Holm (1979) ("holm"), Hochberg (1988) ("hochberg"), Hommel (1988) ("hommel"), Benjamini & Hochberg (1995) ("BH" or its alias "fdr"), και Benjamini & Yekutieli (2001) ("BY"). Τέλος, η επιλογή p.adjust.method="none" δίνει τα απλά p-values του αντίστοιχου t-test, χωρίς καμία διόρθωση για την πολλαπλότητα των ελέγχων. Γενικά, η μέθοδος του Holm θεωρείται καλύτερη από την προσέγγιση του Bonferroni. Οι διορθώσεις των Hochberg και Hommel έχουν μικρές διαφορές, όμως στην πρώτη προσέγγιση το p-value υπολογίζεται πιο γρήγορα. Οι άλλες δύο μέθοδοι έχουν πιο μεγάλη ισχύ, καθώς

επικεντρώνονται στο αναμενόμενο ποσοστό λανθασμένων απορρίψεων της μηδενικής υπόθεσης (false discovery rate).

8.4.3 ΄Ελεγχος Kruskal-Wallis για την ισότητα διαμέσων

Όταν δεν ισχύει η κανονικότητα, μπορούμε να χρησιμοποιήσουμε τον μη-παραμετρικό έλεγχο των Kruskal & Wallis (1952). Το τεστ γενικά, ελέγχει κατά πόσο τιμές από κ δείγματα προέρχονται από την ίδια κατανομή ή διαφορετική, χωρίς, όμως, να υποθέτουμε κανονικότητα. Αν οι κατανομές στις διαφορετικές ομάδες είναι ίδιες, όσον αφορά το σχήμα και τη διακύμανση, τότε η μηδενική υπόθεση μπορείς να γραφτεί ως ισότητα των διαμέσων των κ ομάδων. Η σύνταξη της εντολής είναι:

```
kruskal.test(x, g, ...)                                    ■.
```

Για τα δεδομένα `PlantGrowth`, έχουμε:

```
> kruskal.test(PlantGrowth$weight, PlantGrowth$group)

        Kruskal-Wallis rank sum test

data:  PlantGrowth$weight and PlantGrowth$group
Kruskal-Wallis chi-squared = 7.9882, df = 2, p-value = 0.01842   ■,
```

που σημαίνει ότι οι διάμεσοι διαφέρουν για τις τρεις ομάδες. Επιπλέον με τις εντολές

```
> by(PlantGrowth$weight, PlantGrowth$group,median)
PlantGrowth$group: ctrl
[1] 5.155
--------------------------
PlantGrowth$group: trt1
[1] 4.55
--------------------------
PlantGrowth$group: trt2
[1] 5.435
```

θα πάρουμε τις δειγματικές διάμεσους οι οποίες είναι ίσες με 5.1, 4.55 και 5.43, και είναι παρόμοιες με τους αντίστοιχους δειγματικούς μέσους.

8.5 Πίνακες συνάφειας κατηγορικών μεταβλητών

Στην Ενότητα 8.3 ασχοληθήκαμε, κυρίως, με ελέγχους που εξετάζουν τις σχέσεις μεταξύ μίας ποσοτικής και μιας κατηγορικής μεταβλητής (π.χ. t-tests, ανάλυση διακύμανσης, Wilcoxon test, Kruskal-Wallis test), ή μεταξύ δύο μεταβλητών (μέσω του ελέγχου για τις διακυμάνσεις). Εδώ, θα ασχοληθούμε με τον έλεγχο για την ανεξαρτησία μεταξύ κατηγορικών μεταβλητών.

Όταν έχουμε κατηγορικές μεταβλητές, δε μπορούμε να υπολογίσουμε περιληπτικά μέτρα, όπως στις ποσοτικές μεταβλητές π.χ. ο μέσος ή η διακύμανση. Στις περιπτώσεις αυτές, μπορούμε να εμφανίσουμε μόνο την κατανομή συχνοτήτων της κατηγορικής μεταβλητής και την επικρατούσα κατηγορία (mode), δηλαδή το επίπεδο με τη μεγαλύτερη συχνότητα εμφάνισης. Οι συχνότητες και οι σχετικές συχνότητες υπολογίζονται από τις εντολές `table` και `prop.table`, όπως βλέπουμε και παρακάτω:

```
> library(MASS)
> table(bacteria$trt)

placebo    drug   drug+
     96      62      62
> prop.table(table(bacteria$trt))

  placebo      drug     drug+
0.4363636 0.2818182 0.2818182                                    ■.
```

Έτσι, για τη μεταβλητή `trt` του σετ δεδομένων `bacteria` (της βιβλιοθήκης `MASS`), έχουμε 96 άτομα στην ομάδα ελέγχου (δηλαδή στην ομάδα που λαμβάνουν ψευδο-φάρμακο), και από 62 άτομα στις δύο ομάδες θεραπείας (αντίστοιχα ποσοστά 43.6%, 28.2%, και 28.2%). Μια πιο καλή παρουσίαση μπορούμε να πετύχουμε με τη βιβλιοθήκη `sjPlot` και την σύνταξη:

```
sjt.frq( as.factor(bacteria$trt),
         variableLabels=list("Κατανομή Θεραπείας"),
         encoding="Greek", showSummary=F)                        ■,
```

όπου η κατανομή συχνοτήτων δίνεται στον Πίνακα 8.4.

Κατανομή Θεραπείας

value	*N*	*raw %*	*valid %*	*cumulative %*
placebo	96	43.64	43.64	43.64
drug	62	28.18	28.18	71.82
drug+	62	28.18	28.18	100.00
missings	0	0.00		

Πίνακας 8.4: Κατανομή συχνοτήτων της μεταβλητής `bacteria$trt` όπως εμφανίζεται από την εντολή `sjt.frq` της βιβλιοθήκης `sjPlot`.

8.5.1 Πίνακες διπλής εισόδου

Όμως, ακόμα πιο πολύ ενδιαφέρον έχουν οι πίνακες κατανομής συχνοτήτων, που αφορούν δύο (η περισσότερες) μεταβλητές. Έτσι λοιπόν, ένας πίνακας συνάφειας (contingency table) διπλής εισόδου (two-way) $I \times J$, είναι ένας πίνακας με τις συχνότητες n_{ij} όλων των συνδυασμών των επιπέδων δύο κατηγορικών μεταβλητών X και Y με I και J επίπεδα (κατηγορίες), αντίστοιχα. Χρησιμοποιούνται ευρέως στις κοινωνικές επιστήμες και γενικότερα στις μελέτες με ερωτηματολόγια. Οι πίνακες αυτοί μας δίνουν μια γενική εικόνα της σχέσης (συνάφειας) μεταξύ των δύο κατηγορικών μεταβλητών. Ο όρος πίνακας συνάφειας (contingency table) χρησιμοποιήθηκε για πρώτη φορά από τον Karl Pearson το 1904 (Pearson, 1904).

Μπορούμε να κατασκευάσουμε ένα πίνακα συνάφειας από δεδομένα ενός data.frame με την εντολή `table`. Για παράδειγμα, ο 3×2 πίνακας συνάφειας μεταξύ `y` (αν ένα παιδάκι έχει ωτίτιδα ή όχι) και `trt` (θεραπείας) του πλαίσιου δεδομένων `bacteria` (βλ. βιβλιοθήκη `MASS`) μάς δίνει το ακόλουθο αποτέλεσμα:

```
> table( bacteria$trt, bacteria$y )
          n  y
  placebo 12 84
  drug    18 44
  drug+   13 49                                                    ■.
```

Σε κάποιες περιπτώσεις μπορεί να μας ενδιαφέρουν οι απο-κοινού πιθανότητες $\pi_{ij} = P(X = i, Y = j)$, οι οποίες εκτιμούνται από τις δειγματικές αναλογίες $p_{ij} = n_{ij}/n$ (όταν το δείγμα είναι αντιπροσωπευτικό του πληθυσμού) και τις οποίες μπορούμε να υπολογίσουμε με την εντολή `prop.table`.

```
> prop.table(table( bacteria$trt, bacteria$y))
                    n          y
  placebo 0.05454545 0.38181818
  drug    0.08181818 0.20000000
  drug+   0.05909091 0.22272727                                    ■.
```

Προσέξτε, ότι στην εντολή `prop.table` εισάγουμε τις συχνότητες αποθηκευμένες σε ένα πίνακα (ή σε ένα διάνυσμα) και το μόνο που κάνει, είναι να διαιρεί το κάθε στοιχείο με το άθροισμά τους.

Τέλος, με χρήση της εντολής `sjt.xtab` της βιβλιοθήκης `sjPlot` μπορούμε να έχουμε ένα πίνακα καλύτερης οπτικής αισθητικής και με περισσότερες λεπτομέρειες. Έτσι, η σύνταξη:

```
sjt.xtab(bacteria$trt, bacteria$y,
         variableLabels=c('Θεραπευτική Αγωγή', 'Ασθένεια'),
         showCellPerc=FALSE, showRowPerc=TRUE, showColPerc=FALSE,
         showExpected=TRUE, encoding = "Greek" )
```

θα δώσει το αποτελέσματα του Πίνακα 8.5 όπου εμφανίζονται και οι υπο-συνθήκη πιθανότητες

$P(Y|X)$, $P(\texttt{y}|\texttt{trt})$ στο συγκεκριμένο παράδειγμα, ως πιθανότητες ανά γραμμή (row probabilities) με μπλε χρώμα, αλλά και οι περιθώριες κατανομές των δύο μεταβλητών $\pi_{i\cdot} = P(X = i)$ και $\pi_{\cdot j} = P(Y = j)$, δηλαδή οι περιθώριες κατανομές των `trt` και `Y`, αντίστοιχα, για το συγκεκριμένο παράδειγμα.

Θεραπευτική Αγωγή	*Ασθένεια* n	y	***Total***
placebo	12	84	96
	19	77	96
	12.5 %	87.5 %	100.0 %
drug	18	44	62
	12	50	62
	29 %	71 %	100.0 %
drug+	13	49	62
	12	50	62
	21 %	79 %	100.0 %
Total	43	177	220
	43	177	220
	19.5 %	80.5 %	100.0 %

X^2=6.659 · *df*=2 · Φ_c=0.174 · *p*=0.036

Πίνακας 8.5: Κατανομή συνάφειας της νόσου (`bacteria$y`) ανά θεραπεία (`bacteria$trt`), όπως εμφανίζεται από την εντολή `sjt.xtab` της βιβλιοθήκης `sjPlot`.

8.5.2 Έλεγχοι ανεξαρτησίας για δύο κατηγορικές μεταβλητές

Στην περίπτωση των πινάκων συνάφειας διπλής εισόδου, συνήθως, μας ενδιαφέρει να ελέγξουμε για ανεξαρτησία μεταξύ των δύο κατηγορικών μεταβλητών, δηλαδή ελέγχουμε για τη μηδενική υπόθεση

H_0 : Ανεξαρτησία μεταξύ X και Y έναντι της εναλλακτικής H_1 : εξάρτηση μεταξύ X και Y

Η ανεξαρτησία ορίζεται στατιστικά ως $P(X = i, Y = j) = P(X = i)P(Y = j)$, ή $\pi_{ij} = \pi_{i\cdot}\pi_{\cdot j}$ με συμβολισμούς πινάκων συνάφειας, δηλαδή ο έλεγχος:

$$H_0 : \pi_{ij} = \pi_{i\cdot}\pi_{\cdot j} \text{ έναντι της } H_1 : \pi_{ij} \neq \pi_{i\cdot}\pi_{\cdot j}$$

για όλα τα τα $i = 1, \ldots, I$ και $j = 1, \ldots, J$.

Ένα άλλος τρόπος να δούμε την ανεξαρτησία, είναι να εξετάσουμε τις υπό συνθήκη κατανομή της Y δεδομένου της X. Αυτό πρακτικά σημαίνει ότι μας ενδιαφέρει να ελέγξουμε την ορθότητα

της υπόθεσης $P(Y = j|X = i) = P(Y = j)$ για κάθε $i = 1, \ldots, I$ και $j = 1, \ldots, J$. Δηλαδή, μας ενδιαφέρει να δούμε, αν η πιθανότητα του Y διαφοροποιείται ή όχι, για διαφορετικές τιμές (επίπεδα) του X. Αν δε διαφοροποιείται, τότε σημαίνει ότι η έκβαση του X δεν επηρεάζει την έκβαση του Y και, συνεπώς, οι δύο μεταβλητές είναι ανεξάρτητες.

Στην R για τον έλεγχο ανεξαρτησίας δύο κατηγορικών μεταβλητών υπάρχουν διαθέσιμοι δύο έλεγχοι με αντίστοιχες εντολές: `chisq.test` και `fisher.test`. Η πρώτη εντολή εφαρμόζει τον (ασυμπτωτικό) έλεγχο ανεξαρτησίας του Pearson (1900), ενώ η δεύτερη τον ακριβή έλεγχο του Fisher (1922). Και στις δύο συναρτήσεις, τα εισαγόμενα δεδομένα μπορούν να είναι ένας πίνακας συνάφειας ή δύο κατηγορικές μεταβλητές (factors).

8.5.2.1 Ο έλεγχος ανεξαρτησίας του Pearson

Η γενική σύνταξη της συνάρτησης `chisq.test` είναι:

```
chisq.test(x, y = NULL, correct = TRUE,
           p = rep(1/length(x), length(x)), rescale.p = FALSE,
           simulate.p.value = FALSE, B = 2000)                    ■,
```

ενώ, γενικά, αν έχουμε δύο κατηγορικά διανύσματα `x` και `y`, αρκεί να γράψουμε:

```
chisq.test(x, y)
```

ή

```
tab <- table(x,y)
chisq.test(tab)
```

Όσον αφορά τα υπόλοιπα δυνητικά εισερχόμενα αντικείμενα, το όρισμα `correct` ελέγχει αν θα εφαρμοστεί η διόρθωση του Yates σε πίνακες 2×2, η παράμετρος `simulate.p.value` ελέγχει αν θα υπολογιστεί το p-value ασυμπτωτικά, μέσω της χ^2 κατανομής (μέσω της προκαθορισμένης τιμής `FALSE`) ή αν θα την εκτιμήσει μέσω Monte Carlo προσομοίωσης, ενώ η τιμή του `B` καθορίζει πόσους πίνακες υπό την υπόθεση της ανεξαρτηρίας H_0 θα παράγουμε για τον έλεγχο Monte Carlo. Να θυμίσουμε ότι η κατανομή της ελεγχοσυνάρτησης χ^2 του Pearson, προσεγγίζεται ικανοποιητικά από τη κατανομή χ^2, όταν οι αναμενόμενες συχνότητες $e_{ij} = n_{i.}n_{.j}/n > 5$. Τα άλλα δύο ορίσματα δε χρησιμοποιούνται στην περίπτωση των πινάκων συνάφειας , αλλά μόνο, αν η εντολή χρησιμοποιηθεί με εισερχόμενα αριθμητικά δεδομένα, οπότε και εφαρμόζεται ο έλεγχος καλής προσαρμογής του Pearson.

Για τα δεδομένα του πλαισίου δεδομένων `bacteria` (βλ. τον Πίνακα 8.5), έχουμε:

```
> tab.bacteria <- table( bacteria$trt, bacteria$y )
> chisq.test(tab.bacteria)

        Pearson's Chi-squared test

data:  tab.bacteria
```

```
X-squared = 6.6585, df = 2, p-value = 0.03582                    ■,
```

δηλαδή απορρίπτεται η υπόθεση ανεξαρτησίας για $\alpha = 5\%$ (ασυμπτωτικό p-value=0.0358<0.05) και η ύπαρξη της ωτίτιδας εξαρτάται από τη θεραπεία που ακολούθησε το κάθε παιδί.

Τα αποτελέσματα της συνάρτησης `chisq.test`

Όπως και οι προηγούμενες συναρτήσεις ελέγχων υποθέσεων στην R, το αποτέλεσμα της συνάρτησης `chisq.test` είναι μία λίστα κλάσης `htest`.

```
> chisq.bacteria<-chisq.test(tab.bacteria)
> class(chisq.bacteria)
[1] "htest"
> mode(chisq.bacteria)
[1] "list"
> names(chisq.bacteria)
[1] "statistic" "parameter" "p.value"   "method"    "data.name" "
   observed"  "expected"  "residuals" "stdres"                    ■.
```

Τα περιεχόμενα του αποτελέσματος της `chisq.test` είναι η τιμή της ελεγχοσυνάρτησης:

```
> chisq.bacteria$statistic
X-squared
 6.658529                                                        ■,
```

η τιμή του p-value:

```
> chisq.bacteria$p.value
[1] 0.03581944                                                   ■,
```

οι βαθμοί ελευθερίας της χ^2 κατανομής (`parameter`):

```
> chisq.bacteria$parameter
df
 2                                                               ■,
```

ο έλεγχος που εφαρμόστηκε (`method`):

```
> chisq.bacteria$method
[1] "Pearson's Chi-squared test"                                 ■,
```

και οι αναμενόμενες τιμές e_{ij} κάτω από την υπόθεση της ανεξαρτησίας (`expected`):

```
> chisq.bacteria$expected

                n        y
  placebo 18.76364 77.23636
  drug    12.11818 49.88182
  drug+   12.11818 49.88182                                      ■.
```

Επιπλέον μπορούμε να εξάγουμε τις παρατηρούμενες συχνότητες n_{ij}, δηλαδή τον πίνακα συνάφεια (`observed`), τα ονόματα των δεδομένων (πίνακας ή παράγοντες) όπου έχει εφαρμοστεί ο έλεγχος (`data. name`), τα κατάλοιπα (`residuals`), που δίνονται από τον τύπο $\epsilon_{ij} = n_{ij} - e_{ij}$, και τα τυποποιημένα κατάλοιπα (`stdres`), που ορίζονται ως $\epsilon_{ij}/\sqrt{Var(\epsilon_{ij})}$. Τα αντικείμενα `observed`, `expected`, `residuals` και `stdres` είναι $I \times J$ πίνακες, ενώ τα υπόλοιπα είναι διανύσματα μοναδιαίου μήκους.

Να σημειώσουμε, ότι οι αναμενόμενες τιμές (`chisq. bacteria$expected` στο παράδειγμα) χρησιμοποιούνται για να αποφασίσουμε, αν θα εφαρμόσουμε τον ασυμπτωτικό έλεγχο, ή θα προχωρήσουμε στην προσέγγιση Monte Carlo, που είναι πιο κατάλληλη, όταν κάποιες αναμενόμενες τιμές είναι μικρότερες από πέντε (δηλαδή $e_{ij} < 5$ για κάποιο i και j). Στο παραπάνω παράδειγμα, έχουμε:

```
> min( chisq.bacteria$expected)
[1] 12.11818                                                    ■,
```

δηλαδή η ελάχιστη αναμενόμη τιμή είναι ίση με $12.1 > 5$, άρα μπορούμε να χρησιμοποιήσουμε το ασυμπτωτικό p-value χωρίς κανένα πρόβλημα.

Εκτίμηση του p-value μέσω της προσέγγισης Monte Carlo

Αν τώρα υπολογίσουμε το p-value με τη χρήση της προσέγγισης Monte Carlo, θα πάρουμε αποτελέσματα παρόμοια με τα ακόλουθα:

```
> chisq.test(bacteria$trt, bacteria$y, simulate.p.value=TRUE)

        Pearson's Chi-squared test with simulated p-value (based on
    2000 replicates)

data:  bacteria$trt and bacteria$y
X-squared = 6.6585, df = NA, p-value = 0.04348

> chisq.test(bacteria$trt, bacteria$y,
                            simulate.p.value=TRUE, B=100000)

        Pearson's Chi-squared test with simulated p-value (based on
    1e+05 replicates)

data:  bacteria$trt and bacteria$y
X-squared = 6.6585, df = NA, p-value = 0.03676                  ■.
```

Βλέπουμε ότι στις δύο φορές που υπολογίσαμε το p-value με αυτή την προσέγγιση, πήραμε δύο διαφορετικές τιμές (p-value=0.04348 και 0.03676). Αυτό συμβαίνει, γιατί τα αποτελέσματα βασίζονται σε προσομοίωση, δηλαδή τυχαία δειγματοληψία πινάκων συνάφειας κάτω από την

υπόθεση της ανεξαρτησίας. Συνεπώς, κάθε φορά που καλούμε την εντολή, παράγονται άλλοι τυχαίοι πίνακες και για το λόγο αυτό, παίρνουμε διαφορετική εκτίμηση για το p-value. Αν τώρα τρέξουμε τον έλεγχο πολλές φορές, με $B = 2000$, δηλαδή την προκαθορισμένη τιμή, θα δείτε ότι οι εκτιμήσεις του p-value έχουν μεγάλη μεταβλητότητα (δηλαδή τυπικό σφάλμα). Αντίθετα, αν αυξήσουμε την παράμετρο `B` (τους τυχαίους πίνακες που δειγματοληπτούμε), τότε η εκτίμηση του p-value έχει πολύ μικρότερη μεταβλητότητα. Στο παραπάνω παράδειγμα, μπορούμε να εμπιστευτούμε πολύ περισσότερο την τιμή p-value=0.03676, που έχει εκτιμηθεί με 100000 πίνακες, σε σχέση με την πρώτη εκτίμηση που έχει προέλθει από 2000 πίνακες, καθώς τα τυπικά σφάλματα είναι περίπου 0.0006 και 0.0043, αντίστοιχα (αναλογία περίπου $1 : \sqrt{50}$ περίπου). Να σημειώσουμε ότι τα τυπικά σφάλματα των p-values, μπορούν να εκτιμηθούν ως η τυπική απόκλιση ενός αριθμού Monte Carlo εκτιμήσεων των p-value και τον ακόλουθο κώδικα:

```
pval1<-numeric(0)
for (k in 1:100){
pval1[k] <- chisq.test(bacteria$trt, bacteria$y, simulate.p.value=
  TRUE, B=100000)$p.value
print(k)
}

pval2<-numeric(0)
for (k in 1:100){
pval2[k] <- chisq.test(bacteria$trt, bacteria$y, simulate.p.value=
  TRUE, B=2000)$p.value
print(k)
}
sd(pval1); sd(pval2)                                              ■,
```

όπου τα διανύσματα `pval1` και `pval2` είναι διανύσματα με 100 εκτιμήσεις p-values με 100000 και 2000 προσομοιωμένους πίνακες, αντίστοιχα, ενώ οι τυπικές αποκλίσεις τους θα μας δώσουν εκτιμήσεις των τυπικών σφαλμάτων.

8.5.2.2 Ο ακριβής έλεγχος του Fisher

Ο ακριβής έλεγχος του Fisher (1922), χρησιμοποιείται για τον έλεγχο της ανεξαρτησίας, κυρίως σε μικρά δείγματα, που δεν μπορεί να εφαρμοστεί ο ασυμπτωτικός έλεγχος του Pearson. Η διαφορά είναι ότι αυτός ο έλεγχος υπολογίζει το p-value από την ακριβή κατανομή και για το λόγο αυτό έχει το προσωνύμιο ακριβής (exact). Αν και χρησιμοποιείται κυρίως για μικρά δείγματα, φυσικά, μπορεί να χρησιμοποιηθεί και για μεγάλα δείγματα χωρίς πρόβλημα, με τη διαφορά ότι απαιτεί πολύ περισσότερους υπολογισμούς σε σχέση με τον ασυμπτωτικό έλεγχο του Pearson, που υπολογίζεται εύκολα (αυτό βέβαια δεν αποτελεί τόσο μεγάλο πρόβλημα πλέον, λόγω της τεράστιας υπολογιστικής ισχύος που έχουν οι σύγχρονοι υπολογιστές).

Ο έλεγχος μπορεί να εφαρμοστεί στην R με τη γενική σύνταξη:

```
fisher.test(x, y = NULL, workspace = 200000, hybrid = FALSE,
            control = list(), or = 1, alternative = "two.sided",
            conf.int = TRUE, conf.level = 0.95,
            simulate.p.value = FALSE, B = 2000)                    ■,
```

όπου `x` μπορεί να είναι ένας πίνακας συνάφειας ή διαφορετικά `x` και `y` είναι κατηγορικά διανύσματα (factors), όπως και στον έλεγχο του Pearson. Τώρα, όσον αφορά τις παραμέτρους εισόδου, το όρισμα `hybrid` καθορίζει αν θα υπολογιστεί ο ακριβής έλεγχος ή μια υβριδική προσεγγιστική λύση σε πίνακες διάστασης μεγαλύτερης από 2×2. Με το όρισμα `or` ελέγχουμε τη τιμή της μηδενικής υπόθεσης για το odds ratio, με προκαθορισμένη τιμή ίση με ένα, που αντιστοιχεί στην ανεξαρτησία και το `alternative` προσδιορίζει, ως συνήθως, τον τύπο της εναλλακτικής υπόθεσης. Τα ορίσματα `conf.int` και `conf.level` καθορίζουν κατά πόσο θα υπολογιστεί το διάστημα εμπιστοσύνης για το odds ratio και το αντίστοιχο επίπεδο εμπιστοσύνης (ως προκαθορισμένη επιλογή είναι ο υπολογισμός του 95% διαστήματος εμπιστοσύνης). Τέλος, τα ορίσματα `simulate.p.value` και `B` είναι ίδια όπως στον έλεγχο του Pearson και καθορίζουν, αν θα υπολογιστεί ο ακριβής έλεγχος, ή θα εκτιμηθεί το p-value με τη χρήση προσομοίωσης Monte Carlo και ισχύει μόνο για πίνακες με διάσταση μεγαλύτερη από 2×2.

Στο παράδειγμα των δεδομένων `bacteria`, παίρνουμε τα ακόλουθα αποτελέσματα:

```
> fisher.test(tab.bacteria)

        Fisher's Exact Test for Count Data

data:  tab.bacteria
p-value = 0.03299
alternative hypothesis: two.sided

> fres <- fisher.test(tab.bacteria)
> names(fres)
[1] "p.value"     "alternative" "method"      "data.name"          ■.
```

Όπως βλέπουμε, το p-value είναι ίσο με $0.033 < 0.05$ και, συνεπώς, απορρίπτουμε την υπόθεση της ανεξαρτησίας για $\alpha = 5\%$. Πριν κλείσουμε την ενότητα αυτή, να σημειώσουμε ότι αυτό το p-value είναι πολύ κοντά στο αντίστοιχο του ασυμπτωτικού ελέγχου του Pearson (p-value = 0.03582), όπως και περιμέναμε, εφόσον όλες οι αναμενόμενες τιμές των κελιών e_{ij} ήταν όλες μεγαλύτερες από την τιμή πέντε.

8.5.2.3 Ο έλεγχος McNemar για εξαρτημένες τιμές

Ο έλεγχος McNemar (1947) είναι για εξαρτημένες κατηγορικές μεταβλητές. Συνήθως, έχουμε την ίδια κατηγορική μεταβλητή καταγεγραμμένη σε διαφορετικές χρονικές στιγμές ή σε άτομα ή μονάδες που μελετάμε, που σχετίζονται μεταξύ τους (για παράδειγμα μέτρηση του ίδιου χαρακτηριστικού σε αδέρφια ή στο ίδιο άτομο αλλά σε διαφορετικές χρονικές στιγμές). Εφόσον

μετράμε την ίδια μεταβλητή, ο έλεγχος εφαρμόζεται μόνο σε τετραγωνικούς πίνακες διάστασης $I \times I$ όπου σε κάθε κελί έχουμε τις συχνότητες n_{ij} των ατόμων (ή γενικότερα παρατηρήσεων) όπου στην πρώτη μέτρηση έχει καταγραφεί το επίπεδο i ενώ στη δεύτερη το επίπεδο j. Συνεπώς τα διαγώνια στοιχεία n_{ii} δίνουν τα συμφωνούντα ζευγάρια δηλαδή αυτά στα οποία δεν υπάρχει μεταβολή και έχουν μικρότερο ενδιαφέρον για τον παρακάτω έλεγχο. Γενικά, το τεστ ελέγχει η μηδενική υπόθεση

$$H_0 : n_{ij} = n_{ji} \text{ έναντι της εναλλακτικής } H_1 : n_{ij} \neq n_{ji}$$

η οποία αντιστοιχεί στον έλεγχο συμμετρίας του πίνακα.

Για δίτιμες μεταβλητές, δηλαδή για πίνακες 2×2, ο έλεγχος αποτελεί τον αντίστοιχο του paired t-test, αφού κυρίως βασίζεται στις διαφορές μεταξύ των δύο μετρήσεων. Σε αυτή την περίπτωση οι υπό σύγκριση υποθέσεις είναι ισοδύναμες με:

$$H_0 : \pi_{i\cdot} = \pi_{\cdot i} \text{ έναντι της εναλλακτικής } H_1 : \pi_{i\cdot} \neq \pi_{\cdot i} \tag{8.2}$$

δηλαδή ότι οι περιθώριες κατανομές της X και Y είναι ίδιες. Γενικά, σε εξαρτημένα δείγματα μας ενδιαφέρει να ελέγξουμε για τις υποθέσεις (8.2), ενώ ο έλεγχος συμμετρίας που εφαρμόζεται για πίνακες μεγαλύτερους από 2×2 από το τεστ McNemar, είναι ειδική περίπτωση της σύγκρισης (8.2), αλλά πιο περιοριστικός, αφού αν ένας πίνακας είναι συμμετρικός, τότε και οι περιθώριες θα είναι ίδιες, αλλά δεν ισχύει το αντίθετο.

Η γενική σύνταξη του ελέγχου είναι σχετικά απλή, όπως βλέπουμε παρακάτω:

```
mcnemar.test(x, y = NULL, correct = TRUE)
```
■,

όπου τα δεδομένα δίνονται, όπως και στους δύο προηγούμενου ελέγχους, ως ένας πίνακας συνάφειας `x`, ή ως διανύσματα-παράγοντες `x` και `y`, με τον περιορισμό ότι ο πίνακας πρέπει να είναι τετραγωνικός $I \times I$ και ίσος ο αριθμός των επιπέδων των δύο παραγόντων, αντίστοιχα. Με το όρισμα `correct` επιλέγουμε μια διόρθωση συνέχειας στην περίπτωση των 2×2 πινάκων συνάφειας.

Στο ακόλουθο παράδειγμα, καρκινοπαθείς που λαμβάνουν θεραπεία έχουν συνταιριαστεί (ή εξομοιωθεί - matched) ως προς την ηλικία και τη σοβαρότητα της νόσου. Με τυποποίηση, ο ένας ασθενής του κάθε ζευγαριού έλαβε χημειοθεραπεία, ενώ ο άλλος εναλλακτική θεραπεία για μια περίοδο έξι μηνών. Ύστερα καταγράφτηκε η επιβίωση τους (ή όχι) μετά από πέντε έτη. Μάς ενδιαφέρει να ελέγξουμε κατά πόσο η πιθανότητα επιβίωσης ήταν ίδια για τις δύο θεραπευτικές αγωγές. Τα δεδομένα δίνονται στον Πίνακα 8.6.

Εφόσον τα δεδομένα δίνονται πινακοποιημένα, μπορούμε εδώ να ορίσουμε κατευθείαν τον πίνακα συνάφειας με την εντολή:

```
> matched.clinical.trial <- matrix( c(532,6,22,75), ncol=2 )
> matched.clinical.trial
     [,1] [,2]
[1,]  532   22
[2,]    6   75
```
■,

Επιβίωση μετά από 5 έτη για τους ασθενείς που έλαβαν χημειοθεραπεία	Ασθενείς με εναλλακτική θεραπεία Επιβίωση μετά από 5 έτη	
	NAI	OXI
NAI	532	22
OXI	6	75

Πίνακας 8.6: Δεδομενα παραδείγματος μίας εξομοιωμένης κλινικής δοκιμής της Ενότητας 8.5.2.3

και μετά να τρέξουμε τον έλεγχο McNemar με τη σύνταξη:

```
> mcnemar.test(matched.clinical.trial )

        McNemar's Chi-squared test with continuity correction

data:  matched.clinical.trial
McNemar's chi-squared = 8.0357, df = 1, p-value = 0.004586
```
■,

όπου απορρίπτεται για $\alpha = 1\%$ η μηδενική υπόθεση, ότι οι δύο θεραπείες είναι το ίδιο αποτελεσματικές (p-value=0.0046 < 0.01). Μπορούμε να υπολογίσουμε τις περιθώριες κατανομές με την εντολή `margin.table` και την ακόλουθη σύνταξη:

```
> prop.table(margin.table(matched.clinical.trial, 1))
[1] 0.8724409 0.1275591
> prop.table(margin.table(matched.clinical.trial, 2))
[1] 0.8472441 0.1527559
```
■,

όπου βλέπουμε ότι το 87.2% των ασθενών ,που έλαβαν χημειοθεραπεία, επιβίωσε μετά από 5 έτη, ενώ το 84.7% των ασθενών, που ακολούθησαν την εναλλακτική θεραπεία, επιβίωσαν μετά από 5 έτη.

8.6 Σύνταξη και αντικείμενα τύπου formula

Κεντρικό ρόλο στην R για τον ορισμό των στατιστικών μοντέλων, με βασικότερο αυτό της γραμμικής παλινδρόμησης (βλ. Ενότητα 8.7 για λεπτομέρειες), έχει η χρήση των αντικειμένων τύπου formula (και της αντίστοιχης σύνταξης). Ως στατιστικό μοντέλο εννοούμε ένα πιθανοθεωρητικό υπόδειγμα, που χαρακτηρίζεται από μία (τουλάχιστον) τυχαία μεταβλητή απόκρισης και πολλές επεξηγηματικές μεταβλητές $X_1, X_2, \ldots, X_n$, που χρησιμοποιούνται για να εξηγήσουν και να καθορίσουν κάποια χαρακτηριστικά της . Στα αντικείμενα formula καταγράφονται ακριβώς αυτά τα χαρακτηριστικά. Για παράδειγμα, αν θα χρησιμοποιηθεί η μεταβλητή X ή η X^2 ως επεξηγηματική. ΄Ενα σημαντικό χαρακτηριστικό της R, μέσω της σύνταξης formula, είναι ότι αναγνωρίζει (αν φυσικά έχει οριστεί σωστα) τις κατηγορικές μεταβλητές και δημιουργεί τις κατάλληλες ψευδομεταβλητές (dummies), κάτι που δεν γίνεται αυτόματα σε άλλα στατιστικά πακέτα.

8.6.1 Γενική σύνταξη της formula και τελεστές

Η R, λοιπόν, παρέχει τη δυνατότητα στο χρήστη, μέσω των αντικείμενων formula, να ορίσει (αλλά και να τροποποιήσει) κανείς μια ποικιλία μοντέλων με έναν αρκετά απλό και άμεσο τρόπο, περιλαμβάνοντας σε ένα γενικό τρόπο σύνταξης (σχεδόν) όλα τα μοντέλα που χρησιμοποιούνται στην πράξη. Η βασική σύνταξη των αντικειμένων formula είναι η ακόλουθη:

```
Y ~ X1 + X2 + ... + Xn
```

όπου βλέπουμε ότι υπάρχει αριστερό μέρος της εξίσωσης, στο οποίο ορίζεται η μεταβλητή απόκρισης (response variable) και το δεξί μέρος, όπου ορίζονται οι επεξηγηματικοί όροι (term). Τα δύο μέρη χωρίζονται με το σύμβολο ∼, που χρησιμοποιείται για να υποδηλώσει μια στοχαστική (ή τυχαία) σχέση. Στον παραπάνω ορισμό είπαμε ότι στο δεξί μέρος μιας στατιστικής εξίσωσης (formula) στην R, ορίζουμε ποιοι θα είναι οι στοχαστικοί όροι και όχι μεταβλητές, γιατί μπορούμε να εισάγουμε και συναρτήσεις μεταβλητών, πολλαπλασιαστικούς όρους και αλληλεπιδράσεις μεταξύ άλλων. Σε μοντέλα που δεν υπάρχει εξαρτημένη μεταβλητή (όπως για παράδειγμα στην ανάλυση κυρίων συνιστωσών ή στα λογαριθμο-γραμμικά Poisson μοντέλα για πίνακες συνάφειας), τότε το αριστερό μέλος μένει κενό.

Στα αντικείμενα και στη σύνταξη formula, ο πιο συνηθισμένος τελεστής είναι αυτός της πρόσθεσης (+), που υποδηλώνει ότι αυτή η μεταβλητή που ακολουθεί εισάγεται στο μοντέλο ως επεξηγηματική μεταβλητή. Αντίστοιχα, ο τελεστής της αφαίρεσης υποδηλώνει ότι η μεταβλητή που ακολουθεί, αφαιρείται από το μοντέλο και δεν είναι πλέον επεξηγηματική μεταβλητή. Αντίστοιχα, υπάρχει ο τελεστής της αλληλεπίδρασης που συμβολίζεται από την άνω-κάτω τελεία (:) και ο τελεστής των αλληλεπιδράσεων, που συμπεριλαμβάνει όλους τους υπο-όρους και δίνεται με τον αστερίσκο (*). Για περισσότερους τελεστές ορισμού μοντέλων και επεξηγηματικών μεταβλητών παραπέμπουμε στον Πίνακα 8.7, ενώ ολοκληρωμένα παραδείγματα σύνταξης δίνονται στον Πίνακα 8.8.

Πριν προχωρήσουμε, να σημειώσουμε ότι τα αντικείμενα και η σύνταξη formula συνοδεύονται, συνήθως, με το όρισμα `data`, όπου ορίζουμε από ποιο πλαίσιο δεδομένων θα πάρουμε τις μεταβλητές του μοντέλου (και τα δεδομένα). Στην περίπτωση αυτή η σύνταξη `Y ~ .` ορίζει το πλήρες μοντέλο, δηλαδή ένα μοντέλο με την μεταβλητή Y από το πλαίσιο δεδομένων `data` ως απόκριση και όλες τις υπόλοιπες μεταβλητές του πλαισίου δεδομένων `data` ως επεξηγηματικές. Αντίστοιχα, μπορούμε να ορίσουμε το σταθερό μοντέλο με την σύνταξη `Y ~ 1` . Το σταθερό μοντέλο είναι αυτό χωρίς καμία επεξηγηματική μεταβλητή, όπου στην ουσία χρησιμοποιείται ο συνολικός μέσος ως τιμή πρόβλεψης για όλες τις παρατηρήσεις. Η σταθερά γενικότερα μπορεί να αφαιρεθεί από ένα παλινδρομικό μοντέλο, χρησιμοποιώντας τη σύνταξη `-1` ή `+0`, ενώ μπορεί να ξαναπροστεθεί με τη σύνταξη `+1`.

Γενικά, υπάρχουν και άλλες συναρτήσεις που μπορούν να χρησιμοποιηθούν στο δεξί μέρος μιας formula, όπως:

- Πολυωνυμικοί όροι, (ορθογώνια πολυώνυμα ή απλά πολυώνυμα της μορφής x^p) με την εντολή `poly(x,p)`.

Τελεστής	Επεξήγηση
`.`	Υποδηλώνει ότι θα πάρουμε το πλήρες μοντέλο, δηλαδή με όλες τις μεταβλητές του πλαισίου δεδομένων που θα δηλώσουμε (υποχρεωτικά μαζί) με το όρισμα `data`.
`~1`	Υποδηλώνει ότι θα πάρουμε το σταθερό μοντέλο, δηλαδή χωρίς καμία επεξηγηματική μεταβλητή.
`~0`	Υποδηλώνει ότι θα πάρουμε το μοντέλο χωρίς καμία παράμετρο.
`+0`	Υποδηλώνει ότι θα πάρουμε το μοντέλο χωρίς τη σταθερά.
`-1`	Υποδηλώνει ότι θα αφαιρεθεί η σταθερά από το μοντέλο μας (όμοιο με το προηγούμενο).
`+`	Προσθέτει επεξηγηματικές μεταβλητές στην εξίσωση του μοντέλου.
`-`	Αφαιρεί επεξηγηματικές μεταβλητές από την εξίσωση του μοντέλου.
`:`	Ορίζει την αλληλεπίδραση μεταξύ δύο ή περισσοτέρων μεταβλητών.
`*`	Λαμβάνει υπόψη της όλες τις αλληλεπιδράσεις μεταξύ δύο ή περισσοτέρων μεταβλητών (και τις αλληλεπιδράσεις μικρότερης τάξης).
`^k`	Λαμβάνει υπόψη της όλες τις αλληλεπιδράσεις, μέχρι αυτές της k τάξης μεταξύ δύο ή περισσοτέρων μεταβλητών.
`/`	Ορίζει φωλιασμένες μεταβλητές.
`%in%`	Όμοια με «/».
`(A)`	Δημιουργεί δείκτριες συμμεταβλητές με βάση τη λογική συνθήκη .
`I(···)`	Η σύνταξη αυτή δημιοργεί μια νέα επεξηγηματική μεταβλητή από την αριθμητική σχέση, που θα δώσουμε μέσα στις παρενθέσεις.
`Error(A)`	Ο παράγοντας A είναι τυχαία επίδραση (random effect).

Πίνακας 8.7: Τελεστές ορισμού όρων στα αντικείμενα formula στην R

Σύνταξη	Ερμηνεία του μοντέλου
`Y ~ X + Z`	Το μοντέλο έχει τις μεταβλητές X και Z ως επεξηγηματικές.
`Y ~ .`	Το πλήρες μοντέλο (με όλες τις μεταβλητές, εκτός της Y, του πλαισίου δεδομένων που έχουμε ορίσει υποχρεωτικά στο όρισμα `data`.
`Y ~ 1`	Το σταθερό μοντέλο (χωρίς επεξηγηματικές).
`Y ~ .-1`	Το πλήρες μοντέλο χωρίς τη σταθερά.
`Y ~ .-X`	Το πλήρες μοντέλο χωρίς τη μεταβλητή X.
`Y ~ X -1`	Το μοντέλο δεν έχει σταθερά, αλλά μόνο τη μεταβλητή X.
`Y ~ 0 +X`	Όμοιο με το προηγούμενο.
`Y ~ X + Z-1`	Το μοντέλο με τις μεταβλητές X και Z, αλλά χωρίς σταθερά.
`Y ~ X + Z +0`	Όμοιο με το προηγούμενο.
`Y ~ X*Z`	Το μοντέλο έχει τις μεταβλητές X και Z και την αλληλεπίδραση τους.
`Y ~ (X+Z+W)^2`	Το μοντέλο έχει τις μεταβλητές X, Z και W και τις αλληλεπιδράσεις τους, μέχρι και αυτές της δευτέρας ταξης (two-way interaction terms).
`Y ~ (X>0)`	Το μοντέλο έχει ως επεξηγηματική μια δίτιμη δείκτρια μεταβλητή, που έχει τιμή ένα, αν το X είναι μεγαλύτερο του μηδενός και μηδέν διαφορετικά.
`Y ~ X + X/Z`	Το μοντέλο έχει ως επεξηγηματική την X και την Z φωλιασμένη ως προς την X.
`Y ~ X + X%in%Z`	Όμοιο με το προηγούμενο
`Y ~ X + X:Z`	Το σύμβολο : ορίζει τις αλληλεπιδράσεις, άρα το μοντέλο έχει τη μεταβλητή X και την αλληλεπίδραση της X με τη Z. Είναι ισοδύναμο με το μοντέλο των δύο προηγούμενων formula.
`Y ~ X + I(X*Z)`	Το μοντέλο αυτό έχει ως επεξηγηματική μεταβλητή την X και το γινόμενο $X \times Y$.
`Y ~ X*Z*W - X:Z:W`	Το μοντέλο υποθέτει τη χρήση όλων των αλληλεπιδράσεων μεταξύ των μεταβλητών X, W και Z, εκτός την αλληλεπίδραση τρίτης τάξεως $X.W.Z$.

(Σε όλα τα μοντέλα η μεταβλητή Y έχει χρησιμοποιηθεί ως μεταβλητή απόκρισης.)

Πίνακας 8.8: Παραδείγματα σύνταξης αντικειμένων formula και η ερμηνεία τους

- B-splines με p κόμβους, με την εντολή `bs(x,p)`.
- Κατηγοριοποιημένες μεταβλητές από μια αριθμητική μεταβλητή x, σπάζοντάς την σε p κατηγορίες, με τη σύνταξη `cut(x,p)`.

Τέτοιες προχωρημένες συναρτήσεις, μέσα σε μια formula, είναι πέρα από τους σκοπούς αυτού του συγγράμματος. Επιπλέον, με την σύνταξη `offset(variable)` μπορούμε να προσθέσουμε στο μοντέλο μια μεταβλητή με σταθερό συντελεστή ίσο με τη μονάδα. Συνήθως, τέτοιος όρος χρειάζεται στα μοντέλα Poisson (για συχνότητες), για να λάβουμε υπόψη μας το μέγεθος του συνολικού πληθυσμού. Τέλος, με ειδικούς τελεστές μπορούμε να ορίσουμε τυχαίες επιδράσεις• βλ. Πίνακες 8.7 και 8.8 για λεπτομέρειες και παραδείγματα.

8.6.2 Μερικά παραδείγματα σύνταξης formula

Από τα παραπάνω είναι σαφές ότι ένα μοντέλο μπορεί να γραφεί με αρκετούς διαφορετικούς τρόπους. Οι ακόλουθες συντάξεις formula:

$$\begin{aligned} Y &\sim X + Z + W + X:Z + X:W + Z:W \\ Y &\sim X * Z * W - X:Z:W \\ Y &\sim (X + Z + W)^2 \end{aligned}$$

αντιστοιχούν στο ίδιο μοντέλο με γενικό τύπο:

$$g\big[E(Y_i|X_i, Z_i, W_i)\big] = \beta_0 + X_i\beta_1 + Z_i\beta_2 + W_i\beta_3 + X_iZ_i\beta_4 + X_iW_i\beta_5 + Z_iW_i\beta_6 \ .$$

΄Εστω, λοιπόν, το πλαίσιο δεδομένων `my_data_file` με στήλες (μεταβλητές) `Y`, `X`, `Z`, και `W`, τότε η σύνταξη της μορφής:

```
fit <- r_model_command(Y ~ ., data = my_data_file)
```

είναι ισοδύναμη με τη σύνταξη:

```
fit <- r_model_command(Y ~ X + Z + W, data = my_data_file)   ■,
```

όπου `r_model_command` είναι οποιαδήποτε εντολή μοντέλου της R, όπως, για παράδειγμα, `lm` (γραμμική παλινδρόμηση), `aov` (ανάλυση διακύμανσης) και `glm` (γενικευμένο γραμμικό μοντέλο). Ομοίως, η σύνταξη:

```
fit <- lm(Y ~ .-W, data = my_data_file)
```

είναι ισοδύναμη με

```
fit <- lm(Y ~ X + Z)                                          ■,
```

ενώ η σύνταξη:

```
fit <- lm(Y ~ .*W, data = my_data_file)
```

είναι ισοδύναμη με τη σύνταξη:

```
fit <- lm(Y ~ X + Z + W + X:W + Z:W, data=my_data_file)       ■.
```

8.6.3 Αντικείμενα τύπου formula

Συνήθως στην R, το αποτέλεσμα μιας συνάρτησης formula είναι ένα αντικείμενο τύπου formula. Συνεπώς, να εκχωρηθεί σε ένα νέο αντικείμενο, το οποίο μπορεί να χρησιμοποιηθεί για τον ορισμό ενός μοντέλου.

Έτσι, λοιπόν, η εντολή:

```
lm(Y ~ X)                                                    ■,
```

η οποία ορίζει ένα απλό παλινδρομικό μοντέλο, με μεταβλητή απόκρισης την Y και επεξηγηματική μεταβλητή τη X, μπορεί να γραφτεί ισοδύναμα ως

```
formula.ex <- Y ~ X
lm(formula.ex)                                               ■.
```

Έτσι, λοιπόν, το αντικείμενο `formula.ex` είναι ένα αντικείμενο κλάσης formula και τύπου «γλώσσας προγραμματισμούς» ή «σύνταξης», όπως βλέπουμε και στο ακόλουθο παράδειγμα:

```
> formula.ex1<- y ~ x1+x2
> class(formula.ex1)
[1] "formula"
> typeof(formula.ex1)
[1] "language"                                               ■.
```

Σε κάθε αντικείμενο formula μπορούμε να δούμε λεπτομέρειες για τους όρους με την εντολή: `terms`,

```
> terms(formula.ex1)
y ~ x1 + x2
attr(,"variables")
list(y, x1, x2)
attr(,"factors")
   x1 x2
y   0  0
x1  1  0
x2  0  1
attr(,"term.labels")
[1] "x1" "x2"
attr(,"order")
[1] 1 1
attr(,"intercept")
[1] 1
attr(,"response")
[1] 1
attr(,".Environment")
```

```
<environment: R_GlobalEnv>                                    ■,
```

ενώ μπορούμε να δούμε όλες τις μεταβλητές που συμμετέχουν με τη συνάρτηση `all.vars`

```
> all.vars(formula.ex1)
[1] "y"  "x1" "x2"
```

με τη μεταβλητή απόκρισης να δίνεται ως το πρώτο στοιχείο του διανύσματος χαρακτήρων των αποτελεσμάτων. Τέλος, μπορούμε να ορίσουμε μια formula μέσω ενός διανύσματος χαρακτήρων με την εντολή `as.formula` (ή εναλλακτικά τη `formula`). Για παράδειγμα:

```
# This returns a string:
"y ~ x1 + x2" #
> [1] "y ~ x1 + x2"
# This returns a formula:
as.formula("y ~ x1 + x2") #
> y ~ x1 + x2                                                 ■,
```

ενώ μπορούμε να χρησιμοποιήσουμε την εντολή `paste` για να χτίσουμε αντικείμενα formula, όπως στο παράδειγμα που ακολουθεί:

```
m <- paste("x", 1:25, sep="")
> xnam
 [1] "x1"  "x2"  "x3"  "x4"  "x5"  "x6"  "x7"  "x8"  "x9"  "x10"
[11] "x11" "x12" "x13" "x14" "x15" "x16" "x17" "x18" "x19" "x20"
[21] "x21" "x22" "x23" "x24" "x25"
> (fmla <- as.formula(paste("y ~ ", paste(xnam, collapse= "+"))))
y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 + x10 + x11 +
    x12 + x13 + x14 + x15 + x16 + x17 + x18 + x19 + x20 + x21 +
    x22 + x23 + x24 + x25                                      ■,
```

όπου με την πρώτη εντολή φτιάχνουμε ένα διάνυσμα με τις ονομασίες `x1` έως `x25`, ενώ με τη δεύτερη εντολή φτιάχνουμε τη σύνταξη της formula σε χαρακτήρες, η οποία μετατρέπεται σε αντικείμενο formula με την συνάρτηση `as.formula`.

8.6.4 Έλεγχοι υποθέσεων με τη χρήση formula

Τα τελευταία χρόνια, η σύνταξη formula υιοθετήθηκε από τη R και την κοινότητά της ως εναλλακτικός τρόπος για την εισαγωγή των δεδομένων στις συναρτήσεις των ελέγχων υποθέσεων.

Έτσι, στην εντολή `t.test` μπορούμε να χρησιμοποιήσουμε τη σύνταξη:

```
> t.test(my.response~groups, data=mydata)

        Welch Two Sample t-test

data:  my.response by groups
```

```
t = 0.9849, df = 17.871, p-value = 0.3378
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -2.064979  5.706004
sample estimates:
mean in group 1 mean in group 2
      10.666667        8.846154
```

αντί για τη σύνταξη που χρησιμοποιήσαμε στην Ενότητα 8.3.3.

```
> x<-mydata$my.response[mydata$groups=="1"]
> y<-mydata$my.response[mydata$groups=="2"]
> t.test(x,y)

        Welch Two Sample t-test

data:  x and y
t = 0.9849, df = 17.871, p-value = 0.3378
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -2.064979  5.706004
sample estimates:
mean of x mean of y
10.666667  8.846154
```

■.

Να σημειώσουμε ότι στο παράδειγμα αυτό, τα δεδομένα μας είναι αποθηκευμένα σε ένα πλαίσιο δεδομένων, με την ονομασία `mydata`, με 22 παρατηρήσεις και δύο μεταβλητές. Η πρώτη μεταβλητή είναι η ποσοτική με την ονομασία `my.response`, ενώ η δεύτερη είναι η κατηγορική μεταβλητή, που ορίζει τις δύο ομάδες (με ονομασία `groups`) ή τα δύο δείγματα, όπως ορίζεται στην κλασσική ορολογία της στατιστικής συμπερασματολογίας. Να σημειώσουμε ότι, αυτός είναι ο συνηθισμένος τρόπος αποθήκευσης των δεδομένων και για το λόγο αυτό η σύνταξη formula είναι πιο άμεση και ευκολόχρηστη, σε αντίθεση με τον παραδοσιακό τρόπο, όπου πρέπει πρώτα να χωρίσεις τα δεδομένα σε δύο διανύσματα (ένα για κάθε κατηγορία της κατηγορική μεταβλητής).

Γενικά, για τη συνάρτηση `t.test` η σύνταξη formula είναι της μορφής:

```
Y ~ X, data=mydata
```

για ένα πλαίσιο δεδομένων με το όνομα `mydata`, με την αριθμητική μεταβλητή Y να διαχωρίζεται με τα επίπεδα της κατηγορικής μεταβλητής X (η οποία πρέπει να είναι δηλωμένη ως factor στην R). Γενικά, η μηδενική υπόθεση σε αυτή την περίπτωση θα είναι $H_0 : \mu_1 = \mu_2$, όπου $\mu_k = E(Y|X = k)$ για $k = 1, 2$. Παρόμοια είναι η σύνταξη formula για τις εντολές `wilcox.test`. Για τις εντολές `var.test`, `oneway.test`, `kruskall.test`, `bartlett.test` και

`leveneTest` η σύνταξη είναι πάλι ίδια, με τη διαφορά είναι ότι η X μεταβλητή μπορεί να έχει $k \geq 2$ επίπεδα.

Τέλος, η εντολή `cor. test`, επίσης, να γραφεί με τη σύνταξη formula ως εξής:

```
cor.test( ~ X1+X2+X3, data=mydata)
```

όπου δεν εισάγουμε καμία μεταβλητή ως μεταβλητή απόκρισης. Στην παραπάνω σύνταξη, το αποτέλεσμα θα είναι ένας 3×3 πίνακας συσχετίσεων των μεταβλητών `X1`, `X2` και `X3` του πλαισίου δεδομένων `mydata`.

8.7 Γραμμική συσχέτιση και παλινδρομικά μοντέλα

Στην ενότητα αυτή θα ασχοληθούμε κυρίως με τα μοντέλα γραμμικής παλινδρόμησης τα οποία χρησιμοποιούνται για να περιγράψουν ή να προβλέψουν τη συμπεριφορά μιας μεταβλητής απόκρισης μέσω άλλων μεταβλητών ή χαρακτηριστικών που ονομάζονται επεξηγηματικές μεταβλητές. Πριν προχωρήσουμε όμως στα παλινδρομικά μοντέλα, θα δώσουμε λίγες λεπτομέρειες για τους δείκτες συσχέτισης και πως μπορούμε να τους υπολογίσουμε στην R.

8.7.1 Συσχέτιση μεταξύ δύο μεταβλητών

8.7.1.1 Συνδιακύμανση δύο μεταβλητών

Εκτός από το συντελεστή διακύμανσης, στη στατιστική μπορούμε να υπολογίσουμε τη συνδιακύμανση μεταξύ δύο τυχαίων μεταβλητών ως $Cov(X,Y) = E\Big[\big(X - E(X)\big)\big(Y - E(Y)\big)\Big]$ που μετράει τον τρόπο που μεταβάλλονται μαζί οι X και Y, όπου $E(Z)$ είναι η αναμενόμενη τιμή της τυχαίας μεταβλητής Z. Μηδενικές τιμές της συνδιακύμανσης υποδεικνύουν ότι δεν υπάρχει γραμμική συσχέτιση μεταξύ των δύο μεταβλητών, ενώ το πρόσημο του δείκτη υποδεικνύει θετική (αύξουσα) ή αρνητική (φθίνουσα) σχέση μεταξύ των δύο μεταβλητών. Ο αντίστοιχος δειγματικός δείκτης δίνεται από τον τύπο

$$S_{XY} = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})$$

και στην R υπολογίζεται με τις εντολές `var` και `cov`.

΄Ετσι, λοιπόν, για τα διανύσματα

```
> x1<-rnorm(100,4,10)
> x2<-5*x1+rnorm(100,0,10)
```

έχουμε

```
> var(x1,x2)
[1] 420.5153
> cov(x1,x2)
[1] 420.5153
> sum((x1-mean(x1))*(x2-mean(x2)))/(length(x1)-1)
[1] 420.5153
```

Αν στις εντολές `var` ή `cor` δώσουμε ως εισερχόμενο όρισμα ένα πίνακα διάστασης $n \times p$ (αντί για δύο διανύσματα), τότε το αποτέλεσμα θα είναι είναι ο πίνακας διακύμανσης-συνδιακύμανσης $\mathbf{\Sigma}$ διάστασης $p \times p$, με στοιχεία Σ_{ij} να αντιστοιχούν στις δειγματικές συνδιακυμάνσεις των στηλών i και j του πίνακα $\boldsymbol{A}$. Προφανώς, για τα διαγώνια στοιχεία A_{ii} δίνεται η δειγματική διακύμανση της i στήλης του πίνακα $\boldsymbol{A}$, ενώ ο πίνακας είναι συμμετρικός αφού $Cov(X, Y) = Cov(Y, Y)$. Για παράδειγμα:

```
> x3<-cbind(x1,x2,rnorm(100))
> cov(x3)
           x1          x2
x1  86.182405  420.515286 0.9534440
x2 420.515286 2165.421598 5.6080451
    0.953444    5.608045 0.8998306
> var(x3)
           x1          x2
x1  86.182405  420.515286 0.9534440
x2 420.515286 2165.421598 5.6080451
    0.953444    5.608045 0.8998306                                    ■,
```

όπου βλέπουμε ότι οι διακυμάνσεις είναι ίσες με 86.2, 2165.4 και 0.899 για τις τρεις μεταβλητές του πίνακα `x3`. Αντίστοιχα, οι συνδιακυμάνσεις S_{12}, S_{13} και S_{23} είναι ίσες με 420.5, 0.953 και 5.608.

8.7.1.2 Δείκτες συσχέτισης

΄Ενα πρόβλημα με το δείκτη συνδιακύμανσης είναι ότι δε γνωρίζουμε την κλίμακα του και έτσι δεν μπορούμε να αξιολογήσουμε το βαθμό συσχέτισης μεταξύ των δύο μεταβλητών. Αυτό λύνεται με το δείκτη γραμμικής συσχέτισης του Pearson, ο οποίος υπολογίζει το βαθμό της γραμμικής σχέσης μεταξύ δύο τυχαίων μεταβλητών, δίνεται από τον τύπο

$$\rho_{XY} = \frac{Cov(X, Y)}{\sqrt{Var(X)\, Var(Y)}},$$

ενώ ο αντίστοιχος δειγματικός συντελεστής είναι υπολογίζεται από τον τύπο

$$r_{XY} = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \overline{X})^2 \sum_{i=1}^{n}(Y_i - \overline{Y})^2}}.$$

Ο δείκτης του Pearson παίρνει τιμές από το -1 έως το 1 και μετράει το βαθμό γραμμικής συσχέτισης μεταξύ δύο τυχαίων μεταβλητών. Μηδενικές τιμές υποδεικνύουν ότι οι δύο μεταβλητές είναι μη συσχετισμένες γραμμικά (και όχι ανεξαρτησία), τιμές ίσες με τη μονάδα δείχνουν τέλεια θετική γραμμική συσχέτιση (και αντίστοιχα κοντά στο μείον ένα δείχνουν τέλεια αρνητική γραμμική σχέση).

Τέλος, υπογραμμίζουμε για άλλη μια φορά, η ανεξαρτησία μεταξύ δύο τυχαίων μεταβλητών συνεπάγεται μη γραμμική συσχέτιση, αλλά δεν ισχύει και το αντίθετο. Μόνο στην περίπτωση των κανονικών τυχαίων μεταβλητών, μη-γραμμική συσχέτιση (δηλάδη $\rho = 0$) συνεπάγεται και ανεξαρτησία.

Στην R μπορούμε να υπολογίσουμε τις δειγματικές συσχετίσεις με τις συνάρτηση `cor`. Για παράδειγμα, για τις προσομοιωμένες τιμές:

```
> x1<-rnorm(100)
> x2<-5*x1+rnorm(100)
```

έχουμε

```
> cor(x1,x2)
[1] 0.9864612                                                    ■.
```

Τα διανύσματα x_1 και x_2 έχουν δημιουργηθεί έτσι ώστε να έχουμε γραμμική συσχέτιση μεταξύ των δύο διανυσμάτων. Στα παραπάνω αποτελέσματα βλέπουμε ότι η συσχέτιση μεταξύ των δύο διανυσμάτων είναι πολύ ισχυρή, με το δείκτη του Pearson να έχει τιμή 0.986 (πολύ κοντά στην τέλεια γραμμική σχέση). Αν τώρα κατασκευάσουμε ένα πίνακα x_3 με την εντολή `cbind`, τότε το αποτέλεσμα θα είναι ένας πίνακας συσχετίσεων, όπως βλέπουμε στον κώδικα που ακολουθεί:

```
> x3<-cbind(x1,x2,rnorm(100))
> cor(x3)
              x1           x2
x1  1.000000000  0.98646115 -0.006586435
x2  0.986461155  1.00000000 -0.006363080
   -0.006586435 -0.00636308  1.000000000                          ■.
```

Ο πίνακας `x3` έχει 100 γραμμές και 3 στήλες (πρώτη στήλη x_1, δεύτερη στήλη x_2 και τελευταία στήλη επιπλέον 100 τυχαίους αριθμούς από την τυποποιημένη κανονική κατανομή). Το αποτέλεσμα είναι ένας 3×3 πίνακας (δηλαδή τετραγωνικός και συμμετρικός, με διάσταση ίση με τον αριθμό των στηλών του πίνακα x_3 που εισάγουμε στην εντολή `cor`). Όλα τα στοιχεία της διαγωνίου είναι ένα, εφόσον κάθε μεταβλητή συσχετίζεται τέλεια με τον εαυτό της. Στο κελί $(1, 2)$ βλέπουμε τη συσχέτιση των μεταβλητών x_1 και x_2, που είναι τοποθετημένες στην πρώτη και δεύτερη στήλη του πίνακα που εισάγαμε στην εντολή `cor` (και αντίστοιχα στο στοιχείο (2,1), λόγω συμμετρίας). Αντίθετα, βλέπουμε ότι η τρίτη μεταβλήτη δε σχετίζεται ισχυρά με καμία από τις άλλες δύο μεταβλητές, αφού οι τιμές γραμμικής συσχέτισης είναι ίσες με -0.094 και -0.0919, αντίστοιχα.

Ο συντελεστής συσχέτισης του Spearman είναι ένας εναλλακτικός συντελεστής συσχέτισης, συμπληρωματικός του γραμμικού συντελεστή συσχέτισης του Pearson, ο οποίος μελετάει αυστηρά γραμμική συσχέτιση. Ουσιαστικά, ο συντελεστής του Spearman είναι ένας συντελεστής που μπορεί να δείξει την ύπαρξη μονότονης (όχι δηλαδή απαραίτητα γραμμικής) συσχέτισης. Μπορεί εύκολα κάποιος να αποδείξει ότι ο συντελεστής συσχέτισης του Spearman είναι ισοδύναμος με το να

υπολογίσει κανείς το συντελεστή συσχέτισης του Pearson, όχι όμως στα αρχικά δεδομένα αλλά στα rank τους. Στην R, οι συντελεστές συσχέτισης μπορούν να υπολογιστούν με τη χρήση της εντολής `cor(x,y)`, όπου `x,y` είναι διανύσματα με παρατηρήσεις. Η παραπάνω εντολή επιστρέφει το συντελεστή συσχέτισης του Pearson, ενώ για το συντελεστή του Spearman χρειαζόμαστε την εντολή `cor(x,y, method="spearman")`. Στις εντολές που ακολουθούν, μπορείτε να επαληθεύσετε ότι ο συντελεστής του Spearman αντιστοιχεί στο συντελεστή Pearson, υπολογισμένο όμως στις τάξεις (ranks).

```
> math<-c(30,56,78,54,90)
> phys<-c(65,98,45,67,87)
> cor(math,phys)                        #Pearson coefficient
[1] 0.05043061
> cor(math,phys,method="spearman")      #Spearman coefficient
[1] 0.2
> cor(rank(math),rank(phys))            #Pearson based on ranks
[1] 0.2                                                          ■.
```

Τέλος, η εντολή `cor` μας δίνει την επιλογή να υπολογίσουμε το μη-παραμετρικό δείκτη τ του Kendall, ο οποίος δίνεται από τον τύπο

$$\tau_{xy} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} sgn(x_i - x_j)sgn(y_i - y_j)}{n(n-1)} \tag{8.3}$$

όπου $sgn(x)$ είναι το πρόσημο του x. Ουσιαστικά ο δείκτης τ_{xy} μετράει τη διαφορά μεταξύ των ζευγαριών που μεταβάλλονται με την ίδια φορά (συμφωνούντα ζευγάρια n_c), και των ζευγαριών που μεταβάλλονται αντίστροφα (διαφωνούντα ζευγάρια n_d), δηλαδή $\tau_{xy} = 2\frac{(n_c - n_d)}{n(n-1)}$. Ο δείκτης παίρνει τιμές από το -1 έως το 1. Οι τιμές στα άκρα δείχνουν πλήρη διαφωνία (δηλαδή γνησίως φθίνουσα σχέση) ή πλήρη συμφωνία (δηλαδή γνησίως αύξουσα σχέση). Αντίθετα, ανεξαρτησία μεταξύ των δύο μεταβλητών θα δίνει μηδενικές τιμές του δείκτη, ενώ το ανάποδο γενικά δεν ισχύει. Μηδενικές τιμές δείχνουν ισορροπία μεταξύ των συμφωνούντων και διαφωνούντων ζευγαριών.

Μπορούμε, λοιπόν, να φτιάξουμε μια απλή συνάρτηση που θα εφαρμόζει τον τύπο (8.3) με την ακόλουθη σύνταξη:

```
mykendall<-function( x, y ) {
    d<-0
    n<-length(x)
    for (i in 1:n){
        for (j in 1:n){
            d<-d + sign( x[i]-x[j] ) * sign( y[i]-y[j] )
    }}
    return( d/(n*(n-1)) )
}                                                                ■,
```

και να συγκρίνουμε τα αποτελέσματα της με αυτά της `cor`. Όντως βλέπουμε ότι οι τιμές συμπίπτουν, σύμφωνα με τα αποτελέσματα που ακολουθούν:

```
>  math<-c(30,56,78,54,90)
>  phys<-c(65,98,45,67,87)
>  cor(math,phys,method='kendall')
[1] 0.2
>  mykendall(math,phys)
[1] 0.2
>  x1<-rnorm(100)
>  x2<-5*x1+rnorm(100)
>  cor(x1, x2, method='kendall'); mykendall(x1, x2)
[1] 0.8464646
[1] 0.8464646                                                    ■.
```

Συνοψίζοντας, ο δείκτης του Pearson μετράει τη γραμμική σχέση μεταξύ δύο μεταβλητών, ενώ οι δείκτες του Spearman και του Kendall μετράνε μονότονες σχέσεις. Όλοι οι δείκτες έχουν νόημα να υπολογιστούν ακόμα και αν δεν ισχύει η κανονικότητα. Μηδενικές τιμές των δεικτών δεν σημαίνει απαραίτητα και την ύπαρξη ανεξαρτησίας, αλλά μη γραμμική ή μη μονότονη σχέση μεταξύ των μεταβλητών. Ο δείκτης του Kendall χρησιμοποιεί λιγότερη πληροφορία από το δείκτη του Spearman, ενώ αν έχουμε διατάξιμες μεταβλητές, ο δείκτης του Spearman συμπίπτει με αυτό του Pearson.

8.7.1.3 Έλεγχος για τη συσχέτιση μεταξύ δύο μεταβλητών

Η εντολή `cor.test` εφαρμόζει έναν έλεγχο της μορφής:

$$H_0 : \rho_{x,y} = 0 \text{ έναντι της εναλλακτικής } H_0 : \rho_{x,y} \neq 0,$$

όπου $\rho_{x,y}$ είναι ένας δείκτης συσχέτισης (του Pearson, του Kendall ή του Spearman) μεταξύ των μεταβλητών X και Y. Στην ουσία, εδώ ελέγχουμε για το αν οι δύο μεταβλητές X και Y είναι ασυσχέτιστες, έναντι της εναλλακτικής, αν δηλαδή υπάρχει κάποια μορφή συσχέτισης. Η εφαρμογή του ελέγχου για το συντελεστή συσχέτισης του Pearson βασίζεται σε μία κατανομή Student με $n-2$ βαθμούς ελευθερίας υπό την υπόθεση ότι οι τιμές των X και Y προέρχονται από την κανονική κατανομή. Στην περίπτωση που δεν ισχύει η κανονικότητα, τότε μπορούμε να προχωρήσουμε στον αντίστοιχο έλεγχο για τους μη-παραμετρικούς δείκτες συσχέτισης του Spearman ή του Kendall.

Η γενική σύνταξη της εντολής είναι:

```
cor.test(x, y, alternative = c("two.sided", "less", "greater"),
         method = c("pearson", "kendall", "spearman"),
         exact=NULL, conf.level=0.95, continuity=FALSE, ...)     ■,
```

όπου `x` και `y` δύο αριθμητικά διανύσματα ίσου μήκους. Τα ορίσματα `method` και `exact` ελέγχουν το δείκτη συσχέτισης και αν θα γίνει ακριβής ή ασυμπτωτικός έλεγχος, αντίστοιχα.

Στο ακόλουθο παράδειγμα, έχουμε δύο διανύσματα με τυχαίους αριθμούς από την τυποποιημένη κανονική χωρίς καμία συσχέτιση.

```
> x<-rnorm(100)
> y<-rnorm(100)
> cor.test(x,y)

        Pearson's product-moment correlation

data:  x and y
t = 0.5784, df = 98, p-value = 0.5643
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.1396873  0.2518637
sample estimates:
       cor
0.05833125
```

■.

΄Ετσι, λοιπόν, βρίσκουμε ότι η δειγματική συσχέτιση του Pearson είναι 0.058, για την οποία δεν απορρίπτουμε τη μηδενική υπόθεση (p-value$= 0.56 > 0.05$), ότι οι δυο μεταβλητές δε συσχετίζονται γραμμικά για $\alpha = 0.05$.

Αντίθετα, στο παρακάτω παράδειγμα το διάνυσμα y κατασκευάστηκε, ώστε να σχετίζεται γραμμικά με το x, με δειγματική γραμμική συσχέτιση του Pearson ίση με 0.754.

```
> y<-x+rnorm(100)
> cor(x,y)
[1] 0.7541664
> cor.test(x,y, method='kendall')

        Kendall's rank correlation tau

data:  x and y
z = 7.6299, p-value = 2.354e-14
alternative hypothesis: true tau is not equal to 0
sample estimates:
      tau
0.5175758
```

■.

Από τον παραπάνω έλεγχο, βλέπουμε ότι ο δείκτης του Kendall δε μπορεί να θεωρηθεί μηδέν, αφού απορρίπτεται η μηδενική υπόθεση (p-value$= 2.35 \times 10^{-14} < 0.01$), ακόμα και για $\alpha = 1\%$.

8.7.2 Απλή Παλινδρόμηση

Το μοντέλο της γραμμικής παλινδρόμησης αποτελεί ένα από τα πιο ευρέως γνωστά μοντέλα της στατιστικής. Η βασική ιδέα του μοντέλου είναι πως μπορούμε να περιγράψουμε τη σχέση ανάμεσα σε μια εξαρτημένη μεταβλητή (ή μεταβλητή απόκρισης) Y και μια ανεξάρτητη μεταβλητή (ή επεξηγηματική ή συμμεταβλητή) X με μια γραμμική σχέση και συγκεκριμένα πως για την i παρατήρηση των δεδομένων μας έχουμε:

$$Y_i = \alpha + \beta X_i + \varepsilon_i, \quad \varepsilon_i \sim N(0, \sigma^2) \text{ για } i = 1, \ldots, n \ .$$

Το μοντέλο παλινδρόμησης βασίζεται σε κάποιες βασικές προϋποθέσεις που συνοψίζονται από την μαθηματική έκφρασή $\varepsilon_i \sim N(0, \sigma^2)$. Ποιο συγκεκριμένα, και με βάση αυτή την έκφραση, οι προϋποθέσεις για τα σφάλματα είναι οι ακόλουθες:

1. κανονικότητα σφαλμάτων,

2. ομοσκεδαστικότητα σφαλμάτων,

3. ανεξαρτησία σφαλμάτων.

Επιπλέον, έχουμε άλλη μια προϋπόθεση:

4. γραμμικότητα μεταξύ Y και X.

Αυτή η τελευταία προϋπόθεση απορρέει από τη γραμμική σχέση μεταξύ Y και X όπως ορίζεται στον τύπο της παλινδρόμησης. Η τελευταία προϋπόθεση δεν αναφέρεται άμεσα στα σφάλματα αλλά είναι δομική του μοντέλου και για το λόγο αυτό επηρεάζει και τις υπόλοιπες προϋποθέσεις. Σε κάποιες περιπτώσεις, η διόρθωση της μη γραμμικότητας μπορεί διορθώσει και τις άλλες προϋποθέσεις. Περισσότερες λεπτομέρειες για τον έλεγχο των προϋποθέσεων της παλινδρόμησης δίνονται στην Ενότητα 8.7.3 που ακολουθεί.

Επιπλέον, εναλλακτικά μπορούμε να γράψουμε το μοντέλο της παλινδρόμησης με τον ακόλουθο τρόπο

$$Y_i \sim N(\alpha + \beta X_i, \sigma^2) \text{ για } i = 1, \ldots, n \ .$$

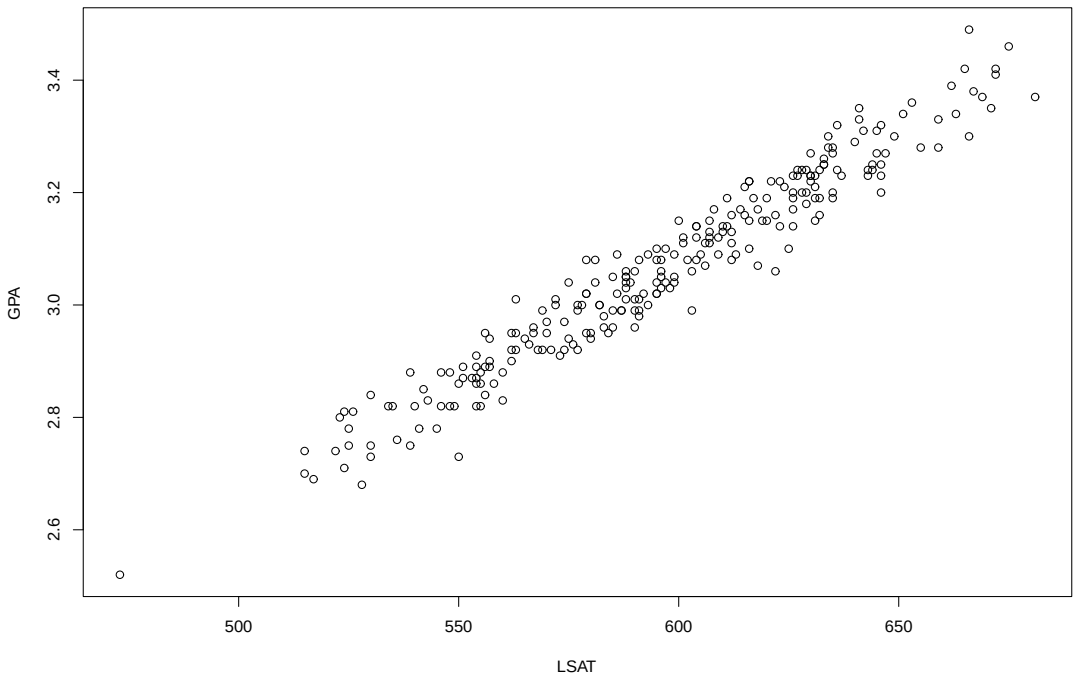
Στον παραπάνω τύπο, το σφάλμα (δηλαδή το τυχαίο κομμάτι του μοντέλου) έχει ενσωματωθεί στη μέση τιμή της Y. Ο παραπάνω τύπος μας επιτρέπει να ορίσουμε (και να αλλάξουμε) κατευθείαν την κατανομή του Y, ενώ οι συμμεταβλητές συνδέονται με τη μέση τιμή της Y γραμμικά, δηλαδή

$$E(Y|X = x) = \alpha + \beta x \ .$$

Τέλος, σε όλα τα προβλεπτικά μοντέλα η συμμεταβλητή (ή, γενικότερα, οι συμμεταβλητές $X_1, \ldots, X_p$) θεωρούνται γνωστές και ότι προηγούνται της Y, αλλιώς το συγκεκριμένο μοντέλο δεν έχει ούτε ερμηνεία, αλλά ούτε μπορεί να χρησιμοποιηθεί για πρόβλεψη.

Παράδειγμα: Τα δεδομένα αφορούν την επίδοση στο τεστ εισαγωγής στο πανεπιστήμιο για 250 φοιτητές, καθώς και τον μέσο όρο GPA που τελικά πέτυχαν. Στο Διάγραμμα 8.2 μπορεί κανείς να δει τα δεδομένα. Τα δεδομένα μπορείτε να τα βρείτε και να τα διαβάσετε και να κάνετε το διάγραμμα με τη χρήση των εντολών:

```
> newdata<-read.table("http://www.stat-athens.aueb.gr/~karlis/lsat
   .txt",header=TRUE)
> head(newdata)
  LSAT  GPA
1  635 3.27
2  631 3.21
3  640 3.29
4  626 3.20
5  585 2.99
6  675 3.46
> plot(newdata$LSAT,newdata$GPA, xlab="LSAT", ylab="GPA")          ■.
```



Διάγραμμα 8.2: Τα δεδομένα.

Μπορεί κανείς να παρατηρήσει, ότι τα δεδομένα μοιάζουν να έχουν μια έντονη γραμμική σχέση, καθώς φαίνεται να κινούνται πάνω σε μια ευθεία γραμμή. Συνεπώς, είναι λογικό να χρησιμοποιήσουμε ένα γραμμικό μοντέλο για να περιγράψουμε τη σχέση του GPA με τη χρήση του LSAT score. Μάλιστα, αν υπολογίσουμε την τιμή του συντελεστή συσχέτισης του Pearson,

βρίσκουμε ότι αυτή είναι εξαιρετικά υψηλή (0.975), κάτι που υπονοεί πολύ έντονη γραμμική σχέση ανάμεσα στις δυο μεταβλητές.

```
> cor(newdata$LSAT,newdata$GPA)
[1] 0.9715196                                                    ■.
```

Μπορούμε να προσαρμόσουμε ένα γραμμικό μοντέλο με τη χρήση της μεθόδου των ελάχιστων τετραγώνων (OLS). Η ιδέα είναι πως, αν ξέρουμε τις τιμές των α και β, μπορούμε να βρούμε και την τιμή του Y που περιμένουμε και, συνεπώς, να δούμε πόσο κοντά είμαστε στην πραγματική. Η μέθοδος OLS ελαχιστοποιεί την ποσότητα

$$\sum_{i=1}^{n}(y_i - \hat{y}_i)^2,$$

όπου $\hat{y}_i = \hat{\alpha} + \hat{\beta}x_i$ είναι οι προσαρμοσμένες (fitted) τιμές του μοντέλου, που μας δείχνουν τι αναμένουμε ως τιμή για τη μεταβλητή απόκρισης , όταν έχουμε παρατηρήσει την τιμή x_i για τη συμμεταβλητή X. Τις αποκλίσεις $e_i = y_i - \hat{y}_i$ των πραγματικών τιμών y_i από τις προσαρμοσμένες $\hat{y}_i$ τις ονομάζουμε κατάλοιπα (residuals) και αποτελούν τις δειγματικές τιμές των σφαλμάτων ε_i. Αν όλα τα σημεία που παρατηρήσουμε, είναι πάνω στην ευθεία, τότε η ποσότητα αυτή μηδενίζεται. Στη πράξη, δεν περιμένουμε κάτι τέτοιο να συμβεί με πραγματικά δεδομένα, αλλά είναι σαφές ότι, όσο πιο μικρή είναι αυτή η ποσότητα (δηλαδή τόσο πιο κοντά στα σημεία θα περνάει η γραμμή), τόσο καλύτερο θα είναι το μοντέλο που προσαρμόσαμε στα δεδομένα.

Στην R, για να κάνουμε γραμμική παλινδρόμηση, χρειαζόμαστε την εντολή `lm` και συγκεκριμένα μπορούμε να πάρουμε τις εκτιμήσεις του μοντέλου

$$GPA_i = \alpha + \beta\ LSAT_i + \varepsilon_i$$

με τη σύνταξη:

```
> lm(GPA ~ LSAT,data=newdata)

Call:
lm(formula = GPA ~ LSAT, data = newdata)

Coefficients:
(Intercept)         LSAT
   0.420459     0.004433                                         ■,
```

δηλαδή εδώ έχουμε ότι:

$$GPA_i = 0.4205 + 0.0044\ LSAT_i + \varepsilon_i\ .$$

Αποθηκεύοντας όμως τα αποτελέσματα σε ένα αντικείμενο (π.χ. με το όνομα `mymodel`) με τη σύνταξη:

```
mymodel<-lm(GPA~LSAT,data=newdata)
```

έχουμε διαθέσιμα πολύ περισσότερα αποτελέσματα. Όταν λέμε αποτελέσματα, εννοούμε μια σειρά από ποσότητες που είναι χρήσιμες για τη συμπερασματολογία, αλλά και την καλύτερη μελέτη του μοντέλου. Μπορεί να δει κανείς, ότι για να ορίσουμε στην R το μοντέλο που θα προσαρμόσουμε, χρειαζόμαστε ένα αντικείμενο τύπου `formula`. Είδαμε προηγούμενα πως μπορούμε στην R να γράψουμε τα μοντέλα που μας ενδιαφέρουν με αντίστοιχο τρόπο.

Για να δούμε τι περιέχει το αντικείμενο τύπου `lm`, μπορούμε να χρησιμοποιήσουμε την εντολή `attributes(mymodel)`

```
> attributes(mymodel)
$names
 [1] "coefficients"  "residuals"     "effects"       "rank"
 [5] "fitted.values" "assign"        "qr"            "df.residual"
 [9] "xlevels"       "call"          "terms"         "model"

$class
[1] "lm"                                                          ■.
```

Στον Πίνακα 8.9 μπορεί να δει κανείς τα πιο σημαντικά περιεχόμενα του αντικειμένου `mymodel`. Δεν θα μιλήσουμε για όλα από αυτά, καθώς αφορούν πιο προχωρημένα θέματα της γραμμικής παλινδρόμησης, κάτι που είναι πέρα από τους σκοπούς αυτού του συγγράμματος.

Αντικείμενο	**Περιγραφή**
`coefficients`	Εκτιμήσεις των συντελεστών του μοντέλου.
`residuals`	Κατάλοιπα του μοντέλου.
`fitted.values`	Προσαρμοσμένες τιμές $\hat{y}_i$.
`df.residual`	Βαθμοί ελευθερίας των καταλοίπων ($n-k-1$ για ένα μοντέλο με k επεξηγηματικές μεταβλητές).
`model`	Πίνακας με τα δεδομένα που χρησιμοποιήθηκαν για να προσαρμόσουμε το μοντέλο.

Πίνακας 8.9: Τα σημαντικότερα περιεχόμενα ενός αντικειμένου κλάσης γραμμικού μοντέλου (`lm`)

Τα επιμέρους αυτά μέρη του αντικειμένου, μπορούμε να τα χρησιμοποιήσουμε όπως θέλουμε. Στις εντολές που ακολουθούν μπορούμε να δούμε τα πρώτα 5 στοιχεία των αντικειμένων με τις προσαρμοσμένες τιμές, τα κατάλοιπα και τα δεδομένα, όπως επίσης, και τους εκτιμημένους συντελεστές:

```
> mymodel$residuals[1:5]
           1            2            3            4            5
 0.034404067 -0.007862784  0.032237631  0.004303652 -0.023931574
> mymodel$fitted[1:5]
```

```
       1        2        3        4        5
3.235596 3.217863 3.257762 3.195696 3.013932
> mymodel$coef
(Intercept)        LSAT
0.420458578 0.004433287
> mymodel$model[1:5,]
   GPA LSAT
1 3.27  635
2 3.21  631
3 3.29  640
4 3.20  626
5 2.99  585                                                    ■.
```

Για να δούμε αναλυτικα τα αποτελέσματα, αρκεί να ζητήσουμε το summary του αντικειμένου που κατασκευάσαμε, δηλαδή:

```
> summary(mymodel, cor=T)

Call:
lm(formula = GPA ~ LSAT, data = newdata)

Residuals:
      Min        1Q    Median        3Q       Max
-0.128767 -0.031665  0.002302  0.032220  0.116972

Coefficients:
             Estimate Std. Error t value Pr(>t)
(Intercept) 4.205e-01  4.101e-02   10.25   <2e-16 ***
LSAT        4.433e-03  6.866e-05   64.57   <2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 0.04183 on 248 degrees of freedom
Multiple R-squared:  0.9439,    Adjusted R-squared:  0.9436
F-statistic:  4169 on 1 and 248 DF,  p-value: < 2.2e-16          ■.
```

Στην οθόνη εμφανίζονται τα πιο σημαντικά αποτελέσματα που μας ενδιαφέρουν σε μια απλή παλινδρόμηση. Για παράδειγμα, μπορούμε να δούμε τις εκτιμήσεις των συντελεστών (`Estimate`), τα τυπικά τους σφάλματα (`Std. Error`), την τιμή της ελεγχοσυνάρτησης για να ελέγξουμε αν η τιμή στον πληθυσμό είναι 0 (`t value`) και το αντίστοιχο p-value (που εμφανίζεται ως `Pr(>|t|)`). Στην πρώτη γραμμή του πίνακα με τους συντελεστές της παλινδρόμησης

εμφανίζεται η σταθερά (`Intercept`) του μοντέλου και οι σχετικές λεπτομέρειες. Άρα, το μοντέλο που βρήκαμε είναι το

$$GPA = 0.4205 + 0.004433\ LSAT + \varepsilon_i,$$

όπως γράψαμε και παραπάνω.

Τα σύμβολα «.», «*», «**» και «***», δίπλα από τα p-value, επισημαίνουν τις μεταβλητές με τους στατιστικά σημαντικούς συντελεστές για $\alpha = 0.10, 0.05, 0.01$ και 0.001, σύμφωνα με την επεξήγηση κάτω από τον πίνακα με τους συντελεστές του μοντέλου (μπορούν να παραληφθούν, αν κάποιος δεν θέλει να εμφανίζονται). Από τα αποτελέσματα βλέπουμε ότι η μεταβλητή LSAT είναι στατιστικά σημαντική, ακόμα και για $\alpha = 0.001$.

Στο δεύτερο μέρος των αποτελεσμάτων, βλέπουμε κάποιες τιμές που είναι χρήσιμες για να καταλάβουμε πόσο καλό είναι το μοντέλο. Συγκεκριμένα, μπορούμε να βρούμε:

`Residual standard error`**:** Είναι η εκτίμηση της τυπικής απόκλισης των σφαλμάτων, δηλαδή του σ. Στο μοντέλο μας η εκτίμηση είναι 0.04183, και μάς δείχνει την ακρίβεια των εκτιμήσεων του μοντέλου στις μονάδες μέτρησης της μεταβλητής απόκρισης Y. Επιπλέον, μπορεί να χρησιμοποιηθεί για τη σύγκριση μοντέλων και για σχετικούς ελέγχους υποθέσεων. Οι 248 βαθμοί ελευθερίας αναφέρονται στα κατάλοιπα και είναι το πλήθος των παρατηρήσεων (250 στην περίπτωσή μας) μείον τους συντελεστές που εκτιμήσαμε (δύο στην περίπτωσή μας).

`Multiple R-squared`**:** Είναι ο συντελεστής προσδιορισμού (coefficient of determination), γνωστός και ως R^2. Τιμές κοντά στη μονάδα δείχνουν ένα μοντέλο με πολύ καλή προσαρμογή (goodness of fit), δηλαδή οι αναμενόμενες τιμές του μοντέλου $\hat{y}_i$ είναι κοντά στις παρατηρούμενες y_i. Επίσης, ο συντελεστής αυτός μπορεί να ερμηνευτεί ως το ποσοστό της διακύμανσης της μεταβλητής απόκρισης (εδώ το GPA), που ερμηνεύεται από τη συμμεταβλητή (εδώ LSAT) ή, πιο σωστά, από το παλινδρομικό μοντέλο με τη συμμεταβλητή LSAT. Εδώ βλέπουμε ότι το 94.4% της GPA εξηγείται από το παλινδρομικό μοντέλο με συμμεταβλητή την LSAT.

Γενικά, ένας εμπειρικός κανόνας είναι ότι τιμές του R^2 μεγαλύτερες του 70% υποδεικνύουν ικανοποιητικά μοντέλα και τιμές πάνω από 90% υποδεικνύουν μοντέλα με πολύ καλή προσαρμοστικότητα. Το R^2 δεν είναι τίποτα άλλο από το τυπικό σφάλμα του μοντέλου σ^2, γραμμένο ως ποσοστό βελτίωσης σε σχέση με το σταθερό μοντέλο (δηλαδή πόσο βελτιώνουμε την εκτίμηση μας με το μοντέλο σε σχέση με το να χρησιμοποιήσουμε το δειγματικό μέσο για πρόβλεψη). Σε μερικές περιπτώσεις, παρόλο που το R^2 μπορεί να είναι πολύ υψηλό υποδεικνύοντας μεγάλη βελτίωση σε σχέση με το σταθερό, το σφάλμα στις αρχικές μονάδες μέτρησης και για πρακτικούς λόγους μπορεί να είναι ακόμα μεγάλο. Συνεπώς, για να εξηγήσουμε τελικά την ποιότητα και την πρακτική χρησιμότητα των εκτιμήσεων και προβλέψεων του μοντέλου θα πρέπει να κοιτάμε και το σ πριν αποφασίσουμε ότι είναι αρκετά ακριβείς για να χρησιμοποιηθούν.

Το R^2 παίρνει τιμές αυστηρά στο διάστημα $[0, 1]$, με την τιμή μηδέν να αντιστοιχεί στο σταθερό μοντέλο και την τιμή ένα σε ένα μοντέλο χωρίς σφάλμα, δηλαδή σε μία σχέση που μπορεί να εξηγηθεί πλήρως από μια γραμμή. Επίσης, το R^2 είναι μια αυστηρά αύξουσα συνάρτηση του αριθμού των συμμεταβλητών. Αυτό σημαίνει ότι, αν σε ένα μοντέλα προσθέσουμε (ή αφαιρέσουμε) μια συμμεταβλητή, τότε αυτό θα αυξηθεί (ή αντίστοιχα θα μειωθεί). Για το λόγο αυτό, σε καμία περίπτωση, δεν πρέπει να χρησιμοποιείται ως δείκτης επιλογής μοντέλου, γιατί πάντα θα μάς δίνει ως καλύτερο το πλήρες μοντέλο (δηλαδή το μοντέλο με όλες τις διαθέσιμες μεταβλητές).

Τέλος, το R^2 μας δείχνει την καλή προσαρμογή του μοντέλου και όχι πόσο καλή πρόβλεψη κάνει το μοντέλο, καθώς υπερ-εκτιμάει την προβλεπτικότητα του μοντέλου, λόγω της διπλής χρήσης των δεδομένων (μεγιστοποιούμε τις παραμέτρους του μοντέλου και μετά μετράμε πόσο κοντά πλησιάζουμε τις παρατηρούμενες τιμές, δηλαδή υπολογίζουμε τη μικρότερη δυνατή απόσταση). Για να υπολογίσουμε σωστά ένα δείκτη πρόβλεψης, θα πρέπει να χρησιμοποιήσουμε τεχνικές cross-validation, δηλαδή να χωρίσουμε το δείγμα μας σε δύο μέρη (training και test datasets) και να εκτιμήσουμε τις παραμέτρους στα δεδομένα εκπαίδευσης (training dataset) και να υπολογίσουμε ένα δείκτη πρόβλεψης (π.χ. το R^2) στα δεδομένα ελέγχου (test dataset).

`Adjusted R-squared`**:** Είναι ο συντελεστής προσδιορισμού διορθωμένος για τον αριθμό των συμμεταβλητών του μοντέλου. Στον τύπο του χρησιμοποιούνται οι αμερόληπτοι εκτιμητές του σ^2 (για το τρέχων και το σταθερό μοντέλο), σε αντίθεση με το απλό R^2, όπου χρησιμοποιούνται οι μεροληπτικοί εκτιμητές. Επειδή αυτός ο δείκτης λαμβάνει υπόψη το πλήθος των μεταβλητών, δεν είναι αυστηρά αύξουσα συνάρτηση του αριθμού των μεταβλητών, όπως το απλό R^2, και, συνεπώς, δεν αυξάνεται υποχρεωτικά όταν προσθέσουμε μια νέα μεταβλητή. Για το λόγο αυτό, μπορεί να χρησιμοποιηθεί ως μέτρο σύγκρισης μοντέλων και επιλογής μεταβλητών, αν και δεν προτείνεται, αφού υπάρχουν πιο καθιερωμένοι δείκτες επιλογής μοντέλων, όπως το κριτήριο πληροφορίας του Akaike (1973, Akaike information criterion, AIC) ή κριτήριο πληροφορίας κατά Bayes (Schwarz, 1978, Bayesian information criterion, BIC).

`F-statistic`**:** Είναι η τιμή της ελεγχοσυνάρτησης για τον έλεγχο ότι $H_0 : \beta = 0$ έναντι $H_1 : \beta \neq 0$ με τους αντίστοιχους βαθμούς ελευθερίας. Στην περίπτωση της απλής παλινδρόμησης ο έλεγχος ταυτίζεται με αυτό που είδαμε για τον συντελεστή παραπάνω (t-value =64.17) και συγκεκριμένα ισχύει ότι $F = t^2$ και στο παράδειγμα μας $64.17^2 = 4169$.

Όλα τα αποτελέσματα της εντολής `summary` μπορούν να καταχωρηθούν σε ένα νέο αντικείμενο και μέσω αυτού μπορούμε να τα ανακαλέσουμε και να τα χρησιμοποιήσουμε για επιπλέον ανάλυση. Για παράδειγμα, οι εντολές:

```
> attributes(summary(mymodel))
$names
```

```
 [1] "call"          "terms"         "residuals"     "coefficients"
 [5] "aliased"       "sigma"         "df"            "r.squared"
 [9] "adj.r.squared" "fstatistic"    "cov.unscaled"

$class
[1] "summary.lm"

> summary(mymodel)$r.squared
[1] 0.9438503
> summary(mymodel)$sigma
[1] 0.04183453
> summary(mymodel)$cov.unscaled
             (Intercept)          LSAT
(Intercept)  0.960888153 -1.605528e-03
LSAT        -0.001605528  2.693857e-06
```

μας επιστρέφουν αντικείμενα της `summary` (το ποιά, μπορούμε να τα δούμε με την εντολή `attributes` ή `names`). Έτσι, στο παραπάνω κομμάτι κώδικα επιστρέφουμε το R^2, το $\hat{\sigma}$ και τον πίνακα διακύμανσης-συνδιακύμανσης των συντελεστών $\hat{\alpha}$ και $\hat{\beta}$.

Για να δει κανείς τον πίνακα ανάλυσης διακύμανσης, που συνοδεύει μια απλή παλινδρόμηση, αρκεί να χρησιμοποιήσει την εντολή `anova(mymodel)`

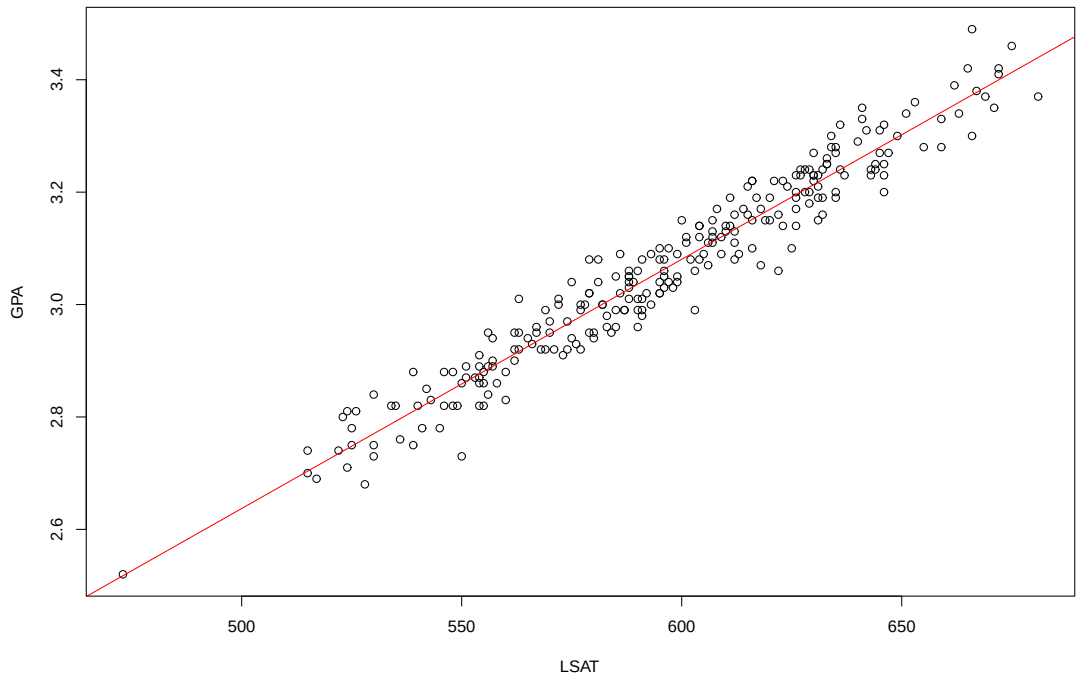
```
> anova(mymodel)
Analysis of Variance Table
Response: GPA
           Df Sum Sq Mean Sq F value    Pr(>F)
LSAT        1 7.2959  7.2959  4168.8 < 2.2e-16 ***
Residuals 248 0.4340  0.0018
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1                    ■.
```

Μπορεί κανείς να προσθέσει την εκτιμηθείσα γραμμή στο scatterplot με τη χρήση της εντολής `abline`. Συγκεκριμένα, οι εντολές που ακολουθούν προσθέτουν στο scatterplot μια κόκκινη γραμμή με την εκτιμηθείσα ευθεία. Το scatterplot αυτό μπορείτε να το δείτε στο Διάγραμμα 8.3.

```
> plot(newdata$LSAT,newdata$GPA, xlab="LSAT", ylab="GPA")
> abline(mymodel,col=2)                                                  ■.
```

Μερικές ακόμα παράμετροι, που θα μπορούσαν να χρησιμοποιηθούν με την εντολή `lm`, είναι οι παρακάτω:

- `subset`: μπορούμε να επιλέξουμε τα δεδομένα με τα οποία θα δουλέψουμε, χρησιμοποιώντας μια λογική συνάρτηση ή τους δείκτες των παρατηρήσεων που θέλουμε να λάβουμε υπόψη.



Διάγραμμα 8.3: Τα δεδομένα και η εκτιμηθείσα γραμμή παλινδρόμησης

```
> model1<- lm(GPA ~ LSAT , data=newdata, subset= (GPA < 3) )
> summary(model1)

Call:
lm(formula = GPA ~ LSAT, data = newdata, subset = (GPA < 3))

Residuals:
      Min        1Q    Median        3Q       Max
-0.121710 -0.024183  0.000692  0.024072  0.075976

Coefficients:
             Estimate Std. Error t value Pr(>t)
(Intercept) 0.8063476  0.0992975   8.121 2.81e-12 ***
LSAT        0.0037188  0.0001785  20.831  < 2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 0.03742 on 87 degrees of freedom
Multiple R-squared:  0.833,     Adjusted R-squared:  0.8311
F-statistic: 433.9 on 1 and 87 DF,  p-value: < 2.2e-16
```

```
> model2<- lm(GPA ~ LSAT , data=newdata,
                            subset= sample(1:250, 100) )
> summary(model2)

Call:
lm(formula = GPA ~ LSAT, data = newdata,
                          subset = sample(1:250, 100))

Residuals:
      Min         1Q     Median         3Q        Max
-0.111317 -0.031239  0.003168  0.028942  0.128168

Coefficients:
             Estimate Std. Error t value Pr(>t)
(Intercept) 0.4781179  0.0656465   7.283 8.29e-11 ***
LSAT        0.0043299  0.0001098  39.444  < 2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 0.04314 on 98 degrees of freedom
Multiple R-squared:  0.9407,    Adjusted R-squared:  0.9401
F-statistic:  1556 on 1 and 98 DF,  p-value: < 2.2e-16      ■.
```

Στην πρώτη περίπτωση επιλέγουμε να δουλέψουμε με τις παρατηρήσεις που έχουν τιμή για το GPA μικρότερη από 3, ενώ στη δεύτερη με ένα τυχαίο δείγμα 100 παρατηρήσεων από τις 250 που έχουμε.

- weights: με τη χρήση αυτής της υπο-εντολής, μπορούμε να δώσουμε βάρη σε κάθε παρατήρηση και, επομένως, να προσαρμόσουμε ένα μοντέλο χρησιμοποιώντας τη μέθοδο των σταθμισμένων ελαχίστων τετράγωνων (weighted least squares). Ένα τέτοιο μοντέλο είναι για παράδειγμα χρήσιμο για να αντιμετωπίσουμε προβλήματα ετεροσκεδαστικότητας. Επίσης, θέτοντας κάποια βάρη (weights) ίσα με μηδέν, ουσιαστικά, αφαιρούμε τις αντίστοιχες παρατηρήσεις (δηλαδή σε αυτή την περίπτωση, θα δώσει το ίδιο αποτέλεσμα με το όρισμα `subset`).

8.7.3 Βασικοί διαγνωστικοί έλεγχοι των προϋποθέσεων της παλινδρόμησης

Όπως είπαμε και στην αρχής της Ενότητας 8.7.2, ένα μοντέλο γραμμικής παλινδρόμησης βασίζεται σε κάποιες προϋποθέσεις, τις οποίες θα θέλαμε να δούμε αν ισχύουν. Αυτό μπορούμε, είτε να το ελέγξουμε με μια πιο αυστηρή στατιστική διαδικασία (για παράδειγμα έναν έλεγχο υποθέσεων),

είτε με κάποιο διαγνωστικό διάγραμμα. Για παράδειγμα, αν το μοντέλο μας είναι πολύ καλό, τότε θα περιμέναμε οι προσαρμοσμένες (`fitted`) τιμές $\hat{y}_i$ να είναι πολύ κοντά στις πραγματικές y_i και άρα, αν τις απεικονίσουμε διαγραμματικά θα πρέπει να είναι σχεδόν πάνω στη διαγώνιο. Το διάγραμμα θα μοιάζει πολύ με το Διάγραμμα 8.2, ουσιαστικά, επειδή οι προσαρμοσμένες τιμές είναι απλά ένας γραμμικός μετασχηματισμός της συμμεταβλητής LSAT. Συνεπώς, το διάγραμμα θα είναι παρόμοιο, αλλά με μια αλλαγή της κλίμακας στον οριζόντιο άξονα των X.

Στους διαγνωστικούς ελέγχους δουλεύουμε αρκετά με τα τυποποιημένα κατάλοιπα, τα οποία ορίζονται ως

$$r_i = \frac{e_i}{s(e_i)} = \frac{y_i - \hat{y}_i}{\hat{\sigma}\sqrt{1-h_i}},$$

όπου $\hat{\sigma}^2$ είναι η εκτιμώμενη διακύμανση των σφαλμάτων και $h_i = H_{ii}$ είναι η i τιμή στη διαγώνιο του πίνακα $\boldsymbol{H} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T$ που ονομάζεται hat-matrix. Οι τιμές h_i ονομάζονται δείκτες επιρροής ή μόχλευσης (leverages) ή απλά hat-values – για περισσότερες λεπτομέρειες βλ. Ενότητα 8.7.5.

Δεδομένου ότι οι προϋποθέσεις του μοντέλου αφορούν τα σφάλματα ε_i, μας ενδιαφέρει να τις ελέγξουμε μέσω των κατάλοιπων e_i, που αποτελούν το δειγματικό τους ανάλογο. Μπορούμε να ελέγξουμε μερικές από τις βασικές προϋποθέσεις του μοντέλου με την εντολή `plot(mymodel)`, η οποία μας δίνει μια σειρά από διαγνωστικά διαγράμματα των καταλοίπων:

```
> par(mfrow=c(2,2))
> plot(mymodel)                                                    ■.
```

Με την εντολή αυτή παίρνουμε τις παρακάτω γραφικές παραστάσεις (βλ. Διάγραμμα 8.4 για τα δεδομένα του παραδείγματος μας):

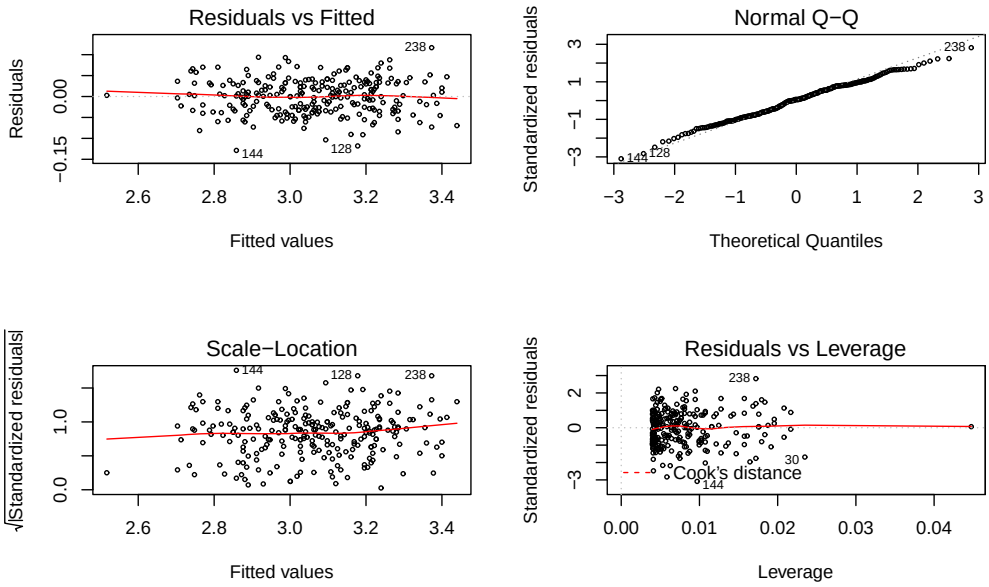
Διάγραμμα 1 (πάνω αριστερά) με τις προσαρμοσμένες τιμές $\hat{y}_i$ σε σχέση με τα απλά, μη τυποποιημένα, κατάλοιπα e_i.

Διάγραμμα 2 (πάνω δεξιά) QQ-plot των τυποποιημένων καταλοίπων r_i.

Διάγραμμα 3 (κάτω αριστερά) με τις προσαρμοσμένες τιμές $\hat{y}_i$ σε σχέση με τη ρίζα των απόλυτων τιμών των τυποποιημένων κατάλοιπων r_i.

Διάγραμμα 4 (κάτω δεξιά) με του δείκτες επιρροής (leverages) σε σχέση με τα τυποποιημένα κατάλοιπα.

Με το πρώτο διάγραμμα ($\hat{y}_i$ με e_i) ελέγχουμε αν τα κατάλοιπα είναι ομοσκεδαστικά, καθώς επίσης και προβλήματα μη-γραμμικότητας. Αν οι προϋποθέσεις ισχύουν, τότε περιμένουμε να δούμε ένα νέφος σημείων χωρίς συστηματικές διαφορές στο άπλωμα των σημείων (δηλαδή στη διακύμανση τους), ή την ύπαρξη κάποιου μοτίβου (pattern). Αλλαγές στο άπλωμα των σημείων υποδεικνύει πρόβλημα ετεροσκεδατικότητας, ενώ η ύπαρξη κάποιου συστηματικού μοτίβου είναι ένδειξη μη γραμμικότητας. Μπορούμε να εντοπίσουμε καλύτερα την ύπαρξη της μη γραμμικότητας μέσω της λείας κόκκινης γραμμής που εμφανίζεται στο διάγραμμα και είναι μια εκτίμηση της μέσης



Διάγραμμα 8.4: Διαγνωστικά διαγράμματα καταλοίπων χρησιμοποιώντας την εντολή `plot(mymodel)`.

τιμής των καταλοίπων σε μια μικρή περιοχή γύρω από κάθε σημείο. Όταν ισχύει η γραμμικότητα, τότε αυτή η γραμμή θα βρίσκεται πολύ κοντά κινούμενη γύρω από τη γραμμή αναφοράς του μηδέν, που είναι η μέση τιμή των σφαλμάτων. Αντίθετα, συστηματικό μοτίβο (δηλαδή ευθεία που υποδεικνύει αύξουσα ή φθίνουσα ή τετραγωνική συνάρτηση) είναι ένδειξη μη γραμμικότητας.

Το τρίτο διάγραμμα ($\hat{y}_i$ με $\sqrt{|r_i|}$) ονομάζεται διάγραμμα κλίμακας-θέσης (Scale-Location) καταλοίπων. Με αυτό το διάγραμμα μπορούμε να ελέγξουμε, επίσης, τις προϋποθέσεις της ομοσκεδαστικότητας και της γραμμικότητας. Επιπλέον, μπορούμε να εντοπίσουμε την ύπαρξη ακραίων τιμών ελέγχοντας τις παρατηρήσεις που ξεπερνούν ενδεικτικά τις τιμές $\sqrt{2} = 1.41$ και $\sqrt{3} = 1.73$ και έχοντας στο μυαλό μας, αν το μοντέλο μας είναι σωστά ορισμένο και οι προϋποθέσεις ισχύουν, ότι περιμένουμε περίπου το 5%, και 1%, αντίστοιχα, των τιμών να ξεπερνούν αυτά τα όρια. Γενικά, προτιμούμε αυτό το διάγραμμα από το αντίστοιχο με τα μη-τυποποιημένα κατάλοιπα, γιατί γίνεται διόρθωση ως προς την κλίμακα των καταλοίπων, έτσι ώστε όλα να έχουν την ίδια διακύμανση (η οποία είναι ασυμπτωτικά ίση με τη μονάδα). Η χρήση της ρίζας γίνεται εδώ για να αφαιρεθεί η ασυμμετρία των απόλυτων τιμών των καταλοίπων. Εναλλακτικά, μπορούμε να χρησιμοποιήσουμε διαγράμματα με τα τυποποιημένα κατάλοιπα r_i ή με τα τετράγωνα τους r_i^2. Η χρήση των r_i^2 θα μεγαλώσει τις διαφορές μεταξύ των καταλοίπων και θα κάνει πιο ευδιάκριτα τυχόν προβλήματα ετεροσκεδαστικότητας.

Το δεύτερο διάγραμμα απεικονίζει τα ποσοστιαία σημεία των καταλοίπων σε σχέση με τα ποσοστιαία σημεία της κανονικής κατανομής. Σημεία κοντά στη γραμμή μας υποδεικνύουν κανονικά κατάλοιπα. Οι αποκλίσεις θα μπορούσαν να μας υποδείξουν προβλήματα που τυχόν υπάρχουν. Στο συγκεκριμένο διάγραμμα παρατηρούμε κάποιες μικρές αποκλίσεις. Επιπλέον,

μπορούμε να κάνουμε έλεγχους κανονικότητας (π.χ. Lilliefors-Kolmogorov-Smirnov test ή Shapiro-Wilks test), χρησιμοποιώντας οποιοδήποτε τύπο καταλοίπων, αν το δείγμα είναι αρκετά μεγάλο.

```
> r <- rstandard(mymodel)
> e<- mymodel$residuals
> library(nortest)
> lillie.test(e)

        Lilliefors (Kolmogorov-Smirnov) normality test

data:  e
D = 0.041, p-value = 0.387

> shapiro.test(e)

        Shapiro-Wilk normality test

data:  e
W = 0.9948, p-value = 0.5481

>
>
> r <- rstandard(mymodel)
> lillie.test(r)

        Lilliefors (Kolmogorov-Smirnov) normality test

data:  r
D = 0.0407, p-value = 0.4

> shapiro.test(r)

        Shapiro-Wilk normality test

data:  r
W = 0.9948, p-value = 0.5562                                    ■.
```

Σε όλες τους παραπάνω ελέγχους δεν απορρίπτουμε την προϋπόθεση της κανονικότητας.

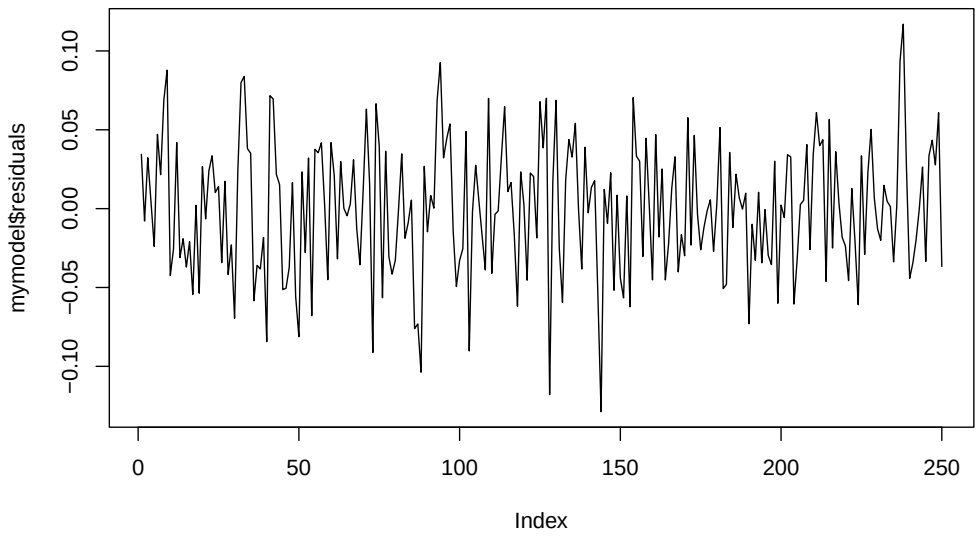
Τέλος, το τελευταίο διάγραμμα απεικονίζει τα τυποποιημένα κατάλοιπα σε σχέση τις τιμές επιρροής h_i. Οι δείκτες επιρροής h_i παίρνουν τιμές από το μηδέν έως το ένα και τιμές κοντά στη μονάδα υποδεικνύουν σημεία επιρροής.

Γενικά, τα παραπάνω διαγράμματα μπορούμε να τα εμφανίσουμε και επιλεκτικά χρησιμοποιώντας την παράμετρο `which`. Έτσι λοιπόν η σύνταξη `plot(mymodel, which=1)` θα μας δώσει το πρώτο διάγραμμα που περιγράψαμε παραπάνω, δηλαδή των προσαρμοσμένων τιμών σε σχέση με τα απλά κατάλοιπα. Οι τιμές $2, 3$, και 5, αντίστοιχα, θα μας δώσουν τα διαγράμματα $2-4$ που περιγράψαμε παραπάνω. Να σημειώσουμε ότι ότι έχουμε τη δυνατότητα να πάρουμε δυο ακόμα διαγράμματα δίνοντας τις τιμές 4 και 6 στην παράμετρο `which`. Αυτά τα διαγράμματα θα μας δώσουν επιπλέον ανάλυση των σημείων επιρροής, απεικονίζοντας τις αποστάσεις Cook για κάθε παρατήρηση και τις αποστάσεις Cook σε σχέση με τους δείκτες επιρροής (leverage)• λεπτομέρειες για τις αποστάσεις Cook δίνονται στην Ενότητα 8.7.5 που ακολουθεί.

Τέλος, όσον αφορά την προϋπόθεση της ανεξαρτησίας αυτή έχει νόημα να την ελέγξουμε μόνο στην περίπτωση που τα δεδομένα έχουν κάποια έννοια χρονικής αλληλουχίας ή σειράς (π.χ. σειρά συλλογής των δεδομένων). Στην περίπτωση που έχουμε στα δεδομένα μας την έννοια της χρονικής σειράς ή αλληλουχίας, τότε μπορούμε να κάνουμε ένα απλό διάγραμμα χρονικής σειράς (time sequence plot) ή να εφαρμόσουμε τον έλεγχο Durbin-Watson, που ελέγχει για την ύπαρξη αυτοσυσχέτισης μίας χρονικής υστέρησης (Durbin & Watson, 1950). Έτσι, λοιπόν για το παράδειγμα μας, μπορούμε να κάνουμε το διάγραμμα χρονικής σειράς (βλ. Διάγραμμα 8.5) με την σύνταξή:

```
plot(mymodel$residuals, type='l')
```
■.

Όταν ισχύει η προϋπόθεση της ανεξαρτησίας τότε περιμένουμε τυχαία σκαμπανεβάσματα όπως στο Διάγραμμα 8.5 χωρίς κάποιο ευδιάκριτο μοτίβο.



Διάγραμμα 8.5: Διάγραμμα χρονικής σειράς των καταλοίπων.

Επιπλέον μπορούμε να υπολογίσουμε το δείκτη των Durbin & Watson (1950)

$$d = \frac{\sum_{i=2}^{n} (e_i - e_{i-1})^2}{\sum_{i=2}^{n} e_i^2},$$

ο οποίος, όπως φαίνεται και από τον τύπο, μετράει αυτοσυσχετίσεις μίας χρονικής υστέρησης δηλαδή μεταξύ των καταλοίπων i και $i-1$. Ο δείκτης παίρνει τιμές από το μηδέν έως το τέσσερα με τιμές κοντά στο δύο να υποδεικνύουν την μη ύπαρξη αυτοσυσχέτισης. Στην R μπορούμε να υπολογίσουμε τον έλεγχο των Durbin-Watson με την συνάρτηση `dwtest` της βιβλιοθήκης `lmtest` με τη σύνταξη:

```
> library(lmtest)
> dwtest(mymodel)

        Durbin-Watson test

data:  mymodel
DW = 1.8707, p-value = 0.1531
alternative hypothesis: true autocorrelation is greater than 0
```

■.

Στα παραπάνω αποτελέσματα βλέπουμε ότι ο δείκτης Durbin-Watson είναι ίσος με 1.87 (σχετικά κοντά στο δύο), με το σχετικό έλεγχο να μην απορρίπτει τη μηδενική υπόθεση $H_0 : d = 0$ σε επίπεδο $\alpha = 5\%$, εφόσον p-value$= 0.15 > 0.05$ και, συνεπώς, να μην εντοπίζεται κάποιο πρόβλημα παραβίασης της προϋπόθεσης της ανεξαρτησίας των σφαλμάτων.

Πιο ενδελεχείς έλεγχοι και διαγνωστικά διαγράμματα δίνονται στη βιβλιοθήκη `car` – βλ. για λεπτομέρειες στο βιβλίο των Fox & Weisberg (2011).

8.7.4 Διόρθωση των αποκλίσεων από τις προϋποθέσεις παλινδρόμησης

Γενικά, είναι πολύ δύσκολο να περιγράψουμε πώς διορθώνονται προβλήματα που οφείλονται σε αποκλίσεις από τις προϋποθέσεις του παλινδρομικού μοντέλου. Στην ενότητα αυτή απλά θα δώσουμε μερικούς ενδεικτικούς τρόπους για να ξεπεράσουμε σχετικά προβλήματα.

Μη γραμμικότητα. Αποκλίσεις από την προϋπόθεση της γραμμικότητας μπορεί να λυθούν με

1. Μετασχηματισμό της μεταβλητής απόκρισης.
2. Μετασχηματισμό των συμμεταβλητών.
3. Εισάγοντας πολυωνυμικούς όρους των ποσοτικών συμμεταβλητών στο μοντέλο.

Όσον αφορά το μετασχηματισμό των μεταβλητών μας, συνήθως ξεκινάμε μετασχηματίζοντας τη μεταβλητή απόκρισης χρησιμοποιώντας το λογάριθμο. Αν αυτός ο μετασχηματισμός δεν

είναι αποτελεσματικός, τότε προχωράμε σε μετασχηματισμούς τύπου Box-Cox (βλ. συνάρτηση `boxcox` στη βιβλιοθήκη `MASS` και συναρτήσεις `boxCox` και `powertransform` στη βιβλιοθήκη `car`). Μπορούμε να προσθέσουμε πολυωνυμικούς όρους στο μοντέλο μας με την εντολή `poly(x, raw=TRUE)`.

Εάν οι παραπάνω προσεγγίσεις δεν είναι αποτελεσματικές, τότε μπορούμε να προχωρήσουμε σε τεχνικές μη-παραμετρικής παλινδρόμησης. Γενικά, διορθώνοντας τη μη-γραμμικότητα, θα βελτιωθούν και οι διαγνωστικοί έλεγχοι για τις υπόλοιπες προϋποθέσεις.

Αποκλίσεις από την κανονικότητα. Αποκλίσεις από την κανονικότητα συνήθως λύνονται με:

1. Μετασχηματισμό της μεταβλητής απόκρισης.
2. Χρήση άλλων κατανομών σφαλμάτων (π.χ. κατανομή Student με πιο παχιές ουρές).
3. Χρησιμοποιώντας άλλη κατανομή για τη μεταβλητή απόκρισης (π.χ. Γάμμα ή Weibull), δηλαδή χρησιμοποιώντας γενικευμένα γραμμικά μοντέλα ή πιο γενικές προσεγγίσεις.

Ετεροσκεδαστικότητα. Αποκλίσεις από την προϋπόθεση της ομοσκεδαστικότητας μπορούν να λυθούν με:

1. Μετασχηματισμό της μεταβλητής απόκρισης.
2. Μικτά μοντέλα με δομή στις διακυμάνσεις των σφαλμάτων.

Χρονική εξάρτηση των παρατηρήσεων. Αν τα δεδομένα μας έχουν χρονική εξάρτηση, τότε πρέπει να χρησιμοποιήσουμε

1. Τεχνικές χρονολογικών σειρών με αυτοσυσχετισμένα σφάλματα.
2. Μικτά μοντέλα με δομή στις συνδιακυμάνσεις των σφαλμάτων (ειδικά σε δεδομένα με επαναλαμβανόμενες μετρήσεις ή longitudinal δεδομένα).

8.7.5 Επιπλέον εντολές σχετικές με τα κατάλοιπα

Στην ενότητα αυτή παραθέτουμε μερικές εντολές σχετικές με τα κατάλοιπα, οι οποίες χρησιμοποιούνται κυρίως για διαγνωστικούς ελέγχους. Οι εντολές αυτές δίνονται περιληπτικά και χωρίς περαιτέρω σχολιασμό της χρησιμότητας τους, καθώς το σύγγραμμα αυτό δεν επικεντρώνεται στην ανάλυση δεδομένων.

Τυποποιημένα κατάλοιπα (`rstandard`)

Τα συνηθισμένα κατάλοιπα υπολογίζονται απλά ως $e_i = y_i - \hat{y}_i$. Με την εντολή `rstandard` παίρνουμε τα τυποποιημένα κατάλοιπα (internally studentized residuals)

$$r_i = \frac{e_i}{s(e_i)} = \frac{e_i}{\hat{\sigma}\sqrt{1-h_i}},$$

όπου $\hat{\sigma}$ είναι η εκτιμώμενη τυπική απόκλιση των σφαλμάτων και h_i είναι οι δείκτες επιρροής ή μόχλευσης (leverages) ή απλά οι τιμές hat οι οποίες δίνονται από τα στοιχεία της διαγώνιου του πίνακα hat, $\boldsymbol{H} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T$.

Τυποποιημένα κατάλοιπα κατά Student (`rstudent`)

Με την εντολή `rstudent` παίρνουμε τα τυποποιημένα κατάλοιπα, έχοντας αφαιρέσει την παρατήρηση (externally studentized residuals). Αυτά μοιάζουν αρκετά με τα προηγούμενα, αλλά διαφέρει το $\hat{\sigma}$. Υπολογίζονται ως

$$r_i = \frac{e_i}{\hat{\sigma}_{(i)}\sqrt{1-h_i}},$$

όπου $\hat{\sigma}_{(i)}$ είναι η εκτίμηση της διακύμανσης, όταν η i παρατήρηση έχει αφαιρεθεί.

Δείκτες επιρροής ή μόχλευσης (leverages)

Όπως γράψαμε και παραπάνω, οι δείκτες επιρροής ή μόχλευσης (leverages) h_i δίνονται από τα στοιχεία της διαγώνιου του πίνακα hat $\boldsymbol{H} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T$. Εναλλακτικά λέγονται και τιμές hat και μετράνε κατά πόσο μία παρατήρηση είναι σημείο επιρροής με την έννοια ότι αλλάζει σημαντικά τους συντελεστές της παλινδρόμησης και τις αντίστοιχες προσαρμοσμένες τιμές. Οι δείκτες επιρροής παίρνουν τιμές στο διάστημα μηδέν έως ένα. Τιμές κοντά στη μονάδα υποδεικνύουν σημεία επιρροής. Μπορούμε να εξάγουμε τους δείκτες επιρροής με την εντολή `hatvalues(mymodel)`.

Δείκτης μεταβολή πρόβλεψης `dffits`

Τα dffits ορίζονται ως

$$df_i = \frac{e_{(i)}}{\hat{\sigma}_{(i)}\sqrt{H_{ii}}}.$$

Μετράει κατά πόσο αλλάζει η παλινδρόμηση, αν αφαιρέσουμε την i παρατήρηση. Για μικρά δείγματα, τιμές μεγαλύτερες από 1 είναι ύποπτες. Για μεγάλα δείγματα, ψάχνουμε για τιμές μεγαλύτερες από $2\sqrt{(p+1)/n}$.

Αποστάσεις Cook

Η απόσταση του Cook υπολογίζεται για κάθε παρατήρηση ως

$$D_i = \frac{\sum_{j=1}^{n}(\hat{y}_j - \hat{y}_j(i))^2}{(p+1)\hat{\sigma}^2},$$

όπου $\hat{y}_j(i)$ είναι η εκτίμηση που παίρνουμε για την j παρατήρηση, όταν έχουμε αφαιρέσει από το μοντέλο την i παρατήρηση. Ουσιαστικά μετράει πόσο επηρεάζει κάθε παρατήρηση τους εκτιμηθέντες συντελεστές.

Δείκτης επίδρασης των συντελεστών `dfbeta`

Τα dfbeta μετράνε επίσης, πόσο επηρεάζει κάθε παρατήρηση την εκτίμηση των συντελεστών.

$$dfb_i = \frac{\hat{\beta}_j - \hat{\beta}_{j(i)}}{\sqrt{\hat{\sigma}^2_{(i)}(X'X)^{-1}_{jj}}}\ .$$

Η ποσότητα μετράει για κάθε παρατήρηση πόσο αλλάζει ο συντελεστής που εκτιμάμε για μια μεταβλητή, όταν η εκτίμηση βασιστεί σε όλα τα δεδομένα, ή όταν αφαιρέσουμε την παρατήρηση i. Ο παρονομαστής είναι η τυπική απόκλιση του συντελεστή, όταν αφαιρέσουμε την παρατήρηση αυτή. Για παρατηρήσεις που παίζουν μεγάλο ρόλο περιμένουμε μεγάλη διάφορα και, άρα, μεγάλη τιμή. Για μικρά/μεσαία σετ δεδομένων, τιμές σε απόλυτη τιμή μεγαλύτερες από το 1 είναι ύποπτες. Για μεγάλα δείγματα, κοιτάμε για τιμές μεγαλύτερες από $2/\sqrt{n}$.

Λόγος συνδιακυμάνσεων (covariance ratio)

Μετράει για κάθε παρατήρηση την επίδραση της στις τυπικές αποκλίσεις των συντελεστών της παλινδρόμησης και γενικότερα σε ολόκληρο τον πίνακα διακύμανσης. Τιμές εκτός του διαστήματος $1 \pm 3p/n$ θεωρούνται ότι έχουν μεγάλη επίδραση.

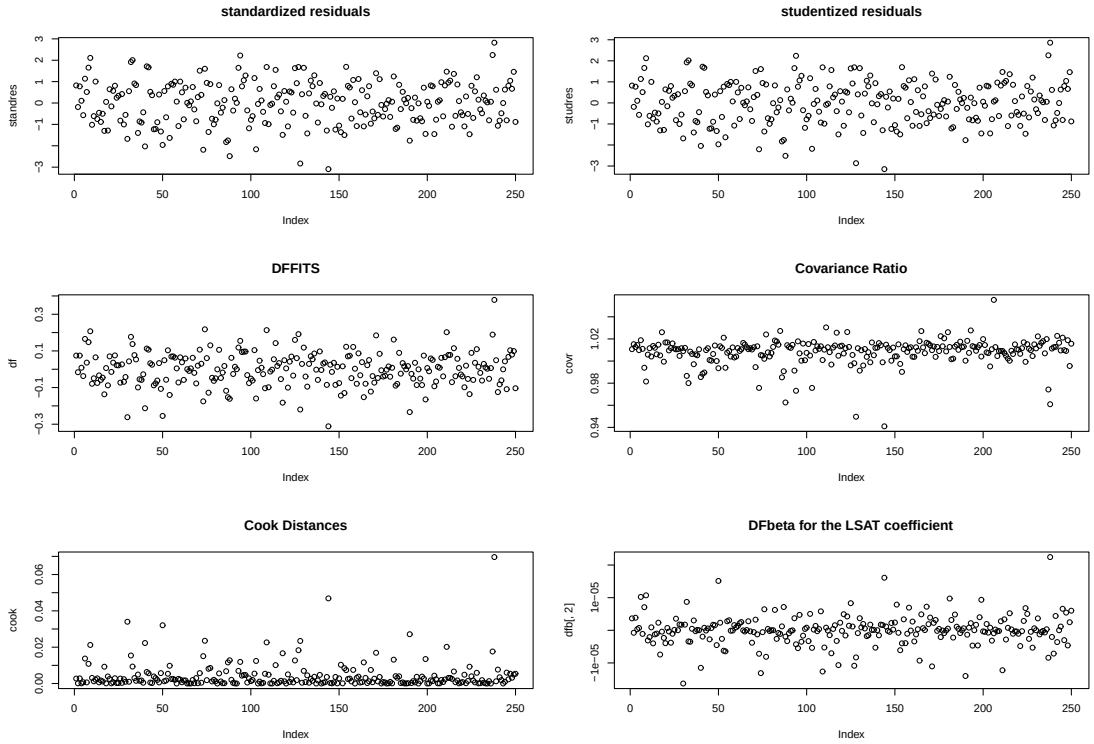
Παράδειγμα χρήσης των επιπλέον διαγνωστικών

Όλα αυτά τα διαγνωστικά τεστ, μπορούμε να τα πάρουμε χρησιμοποιώντας τις αντίστοιχες συναρτήσεις στην R. Τα αντικείμενα που επιστρέφονται είναι διανύσματα ή πίνακες, αντίστοιχα. Τα αποτελέσματα μπορείτε να τα δείτε παρακάτω:

```
> standres[1:5]
         1          2          3          4          5
 0.8257346 -0.1886395  0.7741743  0.1032053 -0.5732944
> studres<-rstudent(mymodel)
> studres[1:5]
         1          2          3          4          5
 0.8252033 -0.1882723  0.7735472  0.1029992 -0.5725169
> df<-dffits(mymodel)
> df[1:5]
           1            2            3            4            5
 0.074562557 -0.016145839  0.074606145  0.008282741 -0.037736257
> dfb<-dfbeta(mymodel)
> dfb[1:5,]
    (Intercept)          LSAT
1 -0.0020333015  3.644389e-06
2  0.0004134550 -7.468800e-07
3 -0.0021686132  3.857011e-06
4 -0.0001913313  3.500984e-07
5 -0.0005204749  7.119725e-07
> covr<-covratio(mymodel)
> covr[1:5]
       1        2        3        4        5
1.010763 1.015236 1.012579 1.014546 1.009811
```

```
> cook<-cooks.distance(mymodel)
> cook[1:5]
           1            2            3            4            5
2.783368e-03 1.308530e-04 2.787553e-03 3.443929e-05 7.139478e-04
> hatv<- hatvalues(mymodel)
> hatv[1:5]
          1           2           3           4           5
0.008098197 0.007300729 0.009216255 0.006425118 0.004325720
```

Στο Διάγραμμα 8.6 δίνεται η γραφική απεικόνιση των διαγνωστικών δεικτών αυτής της ενότητας, για κάθε παρατήρηση.



Διάγραμμα 8.6: Διαγνωστικοί δείκτες για κάθε παρατήρηση.

Εναλλακτικά, η εντολή `influence.measures` επιστρέφει μια σειρά, από τα παραπάνω, στη μορφή ενός πίνακας. Για την ακρίβεια, η εντολή επιστρέφει ένα αντικείμενο με έναν πίνακα με τα διαγνωστικά (το αντικείμενο `infmat` της λίστας) και έναν δεύτερο πίνακα με στοιχεία TRUE και FALSE, αν για κάποιο από τα διαγνωστικά τεστ θεωρείτε ότι είναι έξω από τα όρια. Έτσι, η εντολή `which(apply(infl$is.inf, 1, any))` επιστρέφει τις παρατηρήσεις (με τον αύξοντα αριθμό ή το όνομα που έχουμε ορίσει), αν για τουλάχιστον ένα διαγνωστικό υπάρχει ένδειξη ότι η παρατήρηση είναι ενδιαφέρουσα.

```
> infl<-influence.measures(mymodel)
> head(infl$infmat)
```

```
        dfb.1_      dfb.LSAT         dffit    cov.r        cook.d
            hat
1 -0.049550826  0.053042383  0.074562557 1.010763 2.783368e
   -03 0.008098197
2  0.010062612 -0.010856315 -0.016145839 1.015236 1.308530e
   -04 0.007300729
3 -0.052839511  0.056127637  0.074606145 1.012579 2.787553e
   -03 0.009216255
4 -0.004656362  0.005088619  0.008282741 1.014546 3.443929e
   -05 0.006425118
5 -0.012674740  0.010355036 -0.037736257 1.009811 7.139478e
   -04 0.004325720
6 -0.144092062  0.149094617  0.165883833 1.018835 1.374240e
   -02 0.020814062
> head(infl$is.inf)
  dfb.1_ dfb.LSAT dffit cov.r cook.d   hat
1  FALSE    FALSE FALSE FALSE  FALSE FALSE
2  FALSE    FALSE FALSE FALSE  FALSE FALSE
3  FALSE    FALSE FALSE FALSE  FALSE FALSE
4  FALSE    FALSE FALSE FALSE  FALSE FALSE
5  FALSE    FALSE FALSE FALSE  FALSE FALSE
6  FALSE    FALSE FALSE FALSE  FALSE FALSE
>
> which(apply(infl$is.inf,1,any))
 18  73  85  88  94 103 111 117 123 128 144 165 180 193 206 237 238
 18  73  85  88  94 103 111 117 123 128 144 165 180 193 206 237 238
                                                                  ■.
```

8.7.6 Πολλαπλή Παλινδρόμηση

Η ιδέα της απλής γραμμικής παλινδρόμησης μπορεί να γενικευτεί με πολλούς τρόπους και σε πολλές κατευθύνσεις. Η πιο απλή γενίκευση είναι να υποθέσουμε ότι έχουμε πολλές υποψήφιες επεξηγηματικές μεταβλητές. Τότε το μοντέλο γίνεται:

$$Y_i = \beta_0 + \beta_1 X_{i1} + \ldots + \beta_k X_{ik} + \varepsilon_i, i = 1, \ldots, n \ ,$$

δηλαδή τώρα έχουμε k επεξηγηματικές μεταβλητές. Το μοντέλο αυτό βασίζεται στις ίδιες προϋποθέσεις για τα σφάλματα, όπως είδαμε και προηγουμένως. Το μοντέλο μπορεί να γραφτεί και σε μορφή πινάκων ως

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \varepsilon,$$

όπου $\mathbf{Y}$ είναι ένας πίνακας $n \times 1$ (δηλαδή ένα διάνυσμα στήλη) με τις τιμές της εξαρτημένης μεταβλητής, $\mathbf{X}$ είναι ο πίνακας σχεδιασμού με τις τιμές των επεξηγηματικών μεταβλητών μεγέθους $n \times (k+1)$, η πρώτη του στήλη είναι μονάδες και αντιστοιχεί στην σταθερά, $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_k)$ είναι ένα $k+1$ διάνυσμα στήλη με τους συντελεστές της παλινδρόμησης και ε είναι ένα $n \times 1$ διάνυσμα με τα σφάλματα. Αυτή η αναπαράσταση με τη χρήση πινάκων βοηθάει, καθώς η εκτιμήτρια ελαχίστων τετραγώνων είναι η

$$\widehat{\boldsymbol{\beta}} = (\mathbf{X}\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y},$$

Και πάλι βρίσκουμε τις προσαρμοσμένες τιμές ως $\hat{y}_i = \hat{\beta}_0 + \sum_{j=1}^{k} \hat{\beta}_j X_{ij}$ και τα κατάλοιπα ως $y_i - \hat{y}_i$. Αν οι επεξηγηματικές μεταβλητές είναι συσχετισμένες, τότε έχουμε το πρόβλημα της πολυσυγγραμμικότητας. Άλλα προβλήματα που θα δούμε, αφορούν πολλαπλούς έλεγχους για πολλά β συγχρόνως, προβλήματα επιλογής μεταβλητών και άλλα.

Παράδειγμα: Τα δεδομένα αυτού του παραδείγματος αφορούν 211 διαμερίσματα στην περιοχή της Βαλτιμόρης το έτος 1978. Είναι κομμάτι ενός μεγαλύτερου σετ δεδομένων, αλλά εδώ θα περιοριστούμε σε αυτές τις 211 παρατηρήσεις. Για κάθε σπίτι έχουμε την τιμή του και αρκετές πληροφορίες σχετικά με τα χαρακτηριστικά του, η περιγραφή των μεταβλητών υπάρχει στον Πίνακα 8.10. Σκοπός μας είναι να δούμε ποιες επεξηγηματικές μεταβλητές επιδρούν στην τιμή του σπιτιού και με ποιον τρόπο. Παράλληλα, θα θέλαμε να έχουμε έναν τρόπο να επιλέξουμε τις μεταβλητές.

Όνομα Μεταβλητής	**Περιγραφή**
PRICE	Τιμή του διαμερίσματος
NROOM	Αριθμός δωματίων
NBATH	Αριθμός μπάνιων
PATIO	Αν υπάρχει ή όχι υπαίθριος χώρος
FIREPL	Αν υπάρχει ή όχι τζάκι
AC	Αν υπάρχει ή όχι air condition
BMENT	Αν υπάρχει ή όχι υπόγειο
NSTOR	Αριθμός αποθηκευτικών χώρων
GAR	Αν υπάρχει ή όχι χώρος πάρκινγκ
AGE	Ηλικία του κτιρίου
SQFT	Έκταση σπιτιού σε τετραγωνικά πόδια

Πίνακας 8.10: Διαθέσιμες μεταβλητές στα δεδομένα σπιτιών της Βαλτιμόρης

Τρέχοντας το πλήρες μοντέλο, δηλαδή αυτό με όλες τις υποψήφιες επεξηγηματικές μεταβλητές, η εικόνα είναι η ακόλουθη:

```
> balt<- read.table("http://stat-athens.aueb.gr/~karlis/baltimore.
   txt", sep=";",header=TRUE)
> mymodel2<- lm(PRICE ~ ., data=balt)
> summary(mymodel2)

Call:
lm(formula = PRICE ~ ., data = balt)

Residuals:
    Min      1Q  Median      3Q     Max
-51.856  -8.321  -0.617   6.492  69.876
Coefficients:
             Estimate Std. Error t value Pr(>t)
(Intercept)  39.52317    5.99729   6.590 3.81e-10 ***
NROOM         0.28225    1.23550   0.228  0.81953
NBATH         3.19655    2.11447   1.512  0.13218
PATIO        12.12830    3.10780   3.903  0.00013 ***
FIREPL        9.96394    2.78269   3.581  0.00043 ***
AC            8.67794    2.70530   3.208  0.00156 **
BMENT         2.82308    1.14134   2.473  0.01421 *
NSTOR       -14.51252    2.62350  -5.532 9.85e-08 ***
GAR           8.15603    1.86840   4.365 2.03e-05 ***
AGE          -0.16325    0.05921  -2.757  0.00637 **
SQFT          1.03003    0.21881   4.708 4.68e-06 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 14.32 on 200 degrees of freedom
Multiple R-squared:  0.6493,    Adjusted R-squared:  0.6318
F-statistic: 37.03 on 10 and 200 DF,  p-value: < 2.2e-16          ■.
```

Οι μεταβλητές NROOM και NBATH δεν είναι στατιστικά σημαντικές. Το R^2 είναι ικανοποιητικό κοντά στο 0.65. Ο πίνακας διακύμανσης για το μοντέλο αυτό είναι ο ακόλουθος:

```
> aov(mymodel2)
Call:
   aov(formula = mymodel2)

Terms:
                    NROOM    NBATH    PATIO   FIREPL      AC  BMENT
Sum of Squares  20581.78 11228.60 12385.28 10467.54 5622.92 424.53
```

```
Deg. of Freedom       1         1         1         1         1       1

                  NSTOR     GAR
                4279.18 4696.13
                      1       1

                     AGE     SQFT Residuals
Sum of Squares   1749.64  4547.29  41038.96
Deg. of Freedom        1        1       200

Residual standard error: 14.32462
Estimated effects may be unbalanced                                    ■.
```

Η εντολή `confint(mymodel2)` μας δίνει $100(1-\alpha)\%$ διαστήματα εμπιστοσύνης για τους συντελεστές ενός παλινδρομικού μοντέλου. Το επίπεδο εμπιστοσύνης διαστήματος ελέγχεται από το όρισμα `level` που θέτουμε ίσο με το $1-\alpha$. Στα παρακάτω παραδείγματα μπορούμε να δούμε τα 90% και 95% διαστήματα εμπιστοσύνης για τους συντελεστές του δεύτερου μοντέλου (`model2`).

```
> cbind(coef(mymodel2),confint(mymodel2, level=0.90))
                                5 %          95 %
(Intercept)  39.5231660  29.6125955  49.43373644
NROOM         0.2822488  -1.7594296   2.32392715
NBATH         3.1965484  -0.2976244   6.69072112
PATIO        12.1282995   6.9926305  17.26396852
FIREPL        9.9639448   5.3655253  14.56236429
AC            8.6779436   4.2074215  13.14846563
BMENT         2.8230846   0.9370184   4.70915085
NSTOR       -14.5125173 -18.8478764 -10.17715824
GAR           8.1560280   5.0684889  11.24356706
AGE          -0.1632477  -0.2610965  -0.06539886
SQFT          1.0300341   0.6684559   1.39161234

> cbind(coef(mymodel2),confint(mymodel2, level=0.95))
                              2.5 %       97.5 %
(Intercept)  39.5231660  27.6971323 51.34919966
NROOM         0.2822488  -2.1540345  2.71853207
NBATH         3.1965484  -0.9729598  7.36605655
PATIO        12.1282995   6.0000352 18.25656377
FIREPL        9.9639448   4.4767668 15.45112276
AC            8.6779436   3.3433824 14.01250475
```

```
BMENT          2.8230846   0.5724894  5.07367987
NSTOR        -14.5125173 -19.6857919 -9.33924272
GAR            8.1560280   4.4717455 11.84031046
AGE           -0.1632477  -0.2800082 -0.04648715
SQFT           1.0300341   0.5985720  1.46149629                    ■.
```

Από τα παραπάνω αποτελέσματα, μπορούμε να παρατηρήσουμε ότι η τιμή του μηδενός δε συμπεριλαμβάνεται στα $100(1-\alpha)\%$ διαστήματα εμπιστοσύνης για τις μεταβλητές που δεν βρέθηκαν στατιστικά σημαντικές στο αντίστοιχο επίπεδο σημαντικότητας α.

Γενικά, ισχύουν όλα όσα είπαμε και στην απλή παλινδρόμηση. Δηλαδή οι αντίστοιχες εντολές θα δώσουν αντίστοιχα αποτελέσματα. Επομένως, θα ασχοληθούμε με θέματα που πριν, επειδή είχαμε μόνο μια επεξηγηματική μεταβλητή, δεν αναφέραμε. Ένα τέτοιο πρόβλημα αφορά την επιλογή των μεταβλητών για το μοντέλο, μέσα από ένα μεγαλύτερο σετ δεδομένων.

8.7.7 Μερικός Έλεγχος F

Πολλές φορές σε ένα πρόβλημα παλινδρόμησης, αυτό που θέλουμε να ελέγξουμε δεν είναι απλά η σημαντικότητα μιας και μόνο μεταβλητής, αλλά για περισσότερες από μια. Αυτό κατά κάποιο βαθμό μπορούμε να το επιτύχουμε με το F-statistic. Στις περιπτώσεις όμως που μας ενδιαφέρει ένα υποσύνολο από τις μεταβλητές, χρειάζεται να χρησιμοποιήσουμε το γενικευμένο F-test. Στα δεδομένα μας, έστω ότι έχουμε το μοντέλο:

$$PRICE = \beta_0 + \beta_1 NBATH + \beta_2 FIREPL + \beta_3 PATIO + \beta_4 NROOM,$$

και θέλουμε να ελέγξουμε

$$H_0 : \beta_1 = \beta_2 = 0 \text{ έναντι της εναλλακτικής } H_1 : \beta_j \neq 0 \text{ για τουλάχιστον ένα } j = 1, 2,$$

δηλαδή η εναλλακτική υπόθεση ελέγχει ότι τουλάχιστον ένας από τους συντελεστές β_1 και β_2 είναι διαφορετικός από το μηδέν.
Στην R μπορούμε να κάνουμε αυτό τον έλεγχο σχετικά εύκολα. Προσαρμόζουμε και τα δυο μοντέλα (με και χωρίς τις μεταβλητές) και χρησιμοποιούμε την εντολή `anova` για τον έλεγχο

```
> model1<- lm(PRICE~NBATH + FIREPL + PATIO +NROOM, data=balt)
> model2<- lm(PRICE~PATIO +NROOM, data=balt)
> anova(model2,model1)
Analysis of Variance Table

Model 1: PRICE ~ PATIO + NROOM
Model 2: PRICE ~ NBATH + FIREPL + PATIO + NROOM
  Res.Df   RSS Df Sum of Sq      F    Pr(>F)
1    208 82153
2    206 62359  2     19794 32.695 4.658e-13 ***
```

```
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1                    ■.
```

Αυτό που βλέπουμε, είναι το άθροισμα τετραγώνων των καταλοίπων (residual sum of squares) για τα δυο μοντέλα και τους αντίστοιχους βαθμούς ελευθερίας. Το γενικευμένο F-test βασίζεται στη διαφορά τους και στην περίπτωση μας η τιμή του είναι 32695, που είναι πολύ μεγάλη και απορρίπτουμε τη μηδενική υπόθεση.

Αν τα δυο μοντέλα διαφέρουν κατά μια μόνο μεταβλητή, ο έλεγχος είναι ισοδύναμος με το να ελέγξουμε τη στατιστική σημαντικότητα του συντελεστή της μεταβλητής με το t-test. Εξαιτίας της σχέσης της κατανομής t με την κατανομή F, μπορεί να δει κανείς αμέσως την ισοδυναμία. Δείτε το παρακάτω παράδειγμα:

```
> model1<- lm(PRICE~NBATH + FIREPL + PATIO +NROOM, data=balt)
> model2<- lm(PRICE~PATIO + FIREPL + NROOM, data=balt)
> anova(model2,model1)
Analysis of Variance Table

Model 1: PRICE ~ PATIO + FIREPL + NROOM
Model 2: PRICE ~ NBATH + FIREPL + PATIO + NROOM
  Res.Df   RSS Df Sum of Sq      F    Pr(>F)
1    207 68433
2    206 62359  1    6074.8 20.068 1.238e-05 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
> summary(model1)$coefficient[2,3]^2
[1] 20.06781                                                           ■.
```

Ο συγκεκριμένος έλεγχος είναι ιδιαίτερα χρήσιμος όταν θέλουμε να ελέγξουμε τη σημαντικότητα μιας κατηγορικής συμμεταβλητής (με > 2 επίπεδα) συνολικά καθώς θα εισάγει στο μοντέλο μας $k-1$ ψευδομεταβλητές και αντίστοιχα παραμέτρους.

8.7.8 Επιλογή μεταβλητών με χρήση κλιμακωτών διαδικασιών

΄Ενα από τα σημαντικότερα προβλήματα στην πολλαπλή παλινδρόμηση είναι η επιλογή των μεταβλητών. Στην R η επιλογή των μεταβλητών γίνεται με τη συνάρτηση `step` η οποία εφαρμόζει κλιμακωτές ή βηματικές διαδικασίες (stepwise methods) επιλογής μεταβλητών με χρήση διαφορετικών κριτηρίων επιλογής μοντέλων. Πιο συγκεκριμένα σε κάθε βήμα ελέγχεται η πρόσθεση ή/και η αφαίρεση όλων των μεταβλητών και επιλέγεται το μοντέλο με τη βέλτιστη τιμή του κριτηρίου που έχουμε επιλέξει. Η διαδικασία επαναλαμβάνεται μέχρι να μην μπορούμε να προσθέσουμε ή να αφαιρέσουμε άλλη μεταβλητή. ΄Εχουμε τρεις δυνατές επιλογές διερεύνησης του χώρου των πιθανών μοντέλων: την κλιμακωτή διαδικασία (stepwise procedure) που ελέγχει και την πρόσθεση και την αφαίρεση μεταβλητών σε κάθε βήμα, την πρόσθια διαδικασία (forward pro-

cedure) που ελέγχει σε κάθε βήμα την πρόσθεση μεταβλητών, και την οπισθοδρομική απαλοιφή (backward elimination) που ελέγχει σε κάθε βήμα την αφαίρεση μεταβλητών. Η επιλογή του τύπου της διερεύνησης των μοντέλων γίνεται μέσω της παραμέτρου `direction` με επιλογές `both`, `backward` και `forward` για τις αντίστοιχες διαδικασίες.

Ως μοντέλο εκκίνησης συνήθως λαμβάνουμε το σταθερό μοντέλο (`y 1`) για την forward procedure και το πλήρες μοντέλο (`y .`) για την backward procedure. Για τη stepwise procedure μπορούμε να χρησιμοποιήσουμε οποιοδήποτε μοντέλο για να εκκινήσουμε τον αλγόριθμο αν και συνήθως περιοριζόμαστε στις δύο αυτές επιλογές. Γενικά προτιμάμε την stepwise procedure γιατί εφαρμόζει πιο αναλυτικούς ελέγχους και μπορεί να εισάγει ή να αφαιρέσει μεταβλητές που σε προηγούμενα βήματα αξιολογήθηκαν διαφορετικά και η σημαντικότητα τους άλλαξε λόγω αφαίρεσης άλλων μεταβλητών. Ως σημείο εκκίνησης συνήθως επιλέγουμε το πλήρες μοντέλο όταν είναι εφικτό για να μη χάσουμε μεταβλητές που είναι σημαντικές. Όταν όμως ο αριθμός των μεταβλητών είναι μεγάλος ή προτιμούμε απλούστερα μοντέλα τότε καλό σημείο εκκίνησης είναι το σταθερό.

Όσον αφορά τα κριτήρια επιλογής μοντέλων, η συνάρτηση `step` χρησιμοποιεί κριτήρια πληροφορίας του τύπου

$$IC = -2\log f(\boldsymbol{y}|\widehat{\boldsymbol{\beta}}, \widehat{\sigma}^2) + dk$$

όπου $f(\boldsymbol{y}|\widehat{\boldsymbol{\beta}}, \widehat{\sigma}^2)$ είναι η μέγιστη πιθανοφάνεια του μοντέλου, d είναι ο αριθμός των παραμέτρων του μοντέλου και είναι ίσος με $p+2$ όταν η σταθερά συμπεριλαμβάνεται στο μοντέλο ή $p+1$ όταν η σταθερά δεν συμπεριλαμβάνεται στο μοντέλο και p είναι ο αριθμός των συμμεταβλητών στο μοντέλο. Το παραπάνω κριτήριο είναι ένα κριτήριο ποινικοποιημένης πιθανοφάνειας (penalized likelihood) όπου το πρώτο μέρος μετράει την καλή προσαρμογή του μοντέλου (και έμμεσα την πρόβλεψη) ενώ το δεύτερο εφαρμόζει μια ποινή για κάθε εξτρά παράμετρο που εκτιμάμε υποστηρίζοντας με αυτό τον τρόπο την αρχή της απλότητας (parsimony principle). Η τιμή της ποινής k ελέγχει το βάρος που δίνουμε στις δύο αυτές αρχές που διέπουν τις διαδικασίες επιλογής μοντέλων. Η τιμή της ποινής k ελέγχεται από την παράμετρο `k` με προκαθορισμένη τιμή $k=2$ που παραπέμπει στο κριτήριο πληροφορίας του Akaike (AIC) ενώ το κριτήριο πληροφορίας του Bayes (BIC) μπορεί να εφαρμοστεί για `k=log(n)` όπου n είναι το μέγεθος του δείγματος. Εναλλακτικά μπορούμε να χρησιμοποιήσουμε ελέγχους σημαντικότητας F μέσω της παραμέτρου `test='F'`.

Η γενική σύνταξη της συνάρτησης `step` είναι η ακόλουθη

```
step(object, scope, scale = 0,
     direction = c("both", "backward", "forward"),
     steps = 1000, k = 2, ...)                                    ■,
```

όπου

object : είναι ένα αντικείμενο που προέκυψε από την εντολή `lm`,

scope : Η παράμετρος αυτή ορίζει το εύρος των μοντέλων που εξετάζονται με τις διαδικασίες επιλογής. Το αντικείμενο εισόδου πρέπει να είναι μια λίστα με το μικρότερο και το μεγα-

λύτερο μοντέλο με τη μορφή formula. Αν ορίσουμε ένα μόνο μοντέλο, τότε ορίζεται αυτό ως το μεγαλύτερο και το μικρότερο το θέτουμε ίσο με το σταθερό. Αν δεν ορίσουμε καμία τιμή για την παράμετρο αυτή τότε επιπλέον λαμβάνει το τρέχον μοντέλο ως το μέγιστο.

direction : Ορισμός της διαδικασίας επιλογής: `both` προσθέτει και αφαιρεί, `backward` αφαιρεί και `forward` προσθέτει μεταβλητές σε κάθε βήμα.

steps : Μεγιστος αριθμός βημάτων της διαδικασίας.

k : Επιλογή ποινής για κάθε επιπλέον τιμή. Η προκαθορισμένη τιμή $k = 2$ εφαρμόζει το AIC.

test='F' : Στην περίπτωση αυτή εφαρμόζονται οι κλιμακωτές διαδικασίες χρησιμοποιώντας τους ελέγχους σημαντικότητας F για την επιλογή.

΄Ετσι λοιπόν η σύνταξη

```
step(min.model, direction='forward', scope=biggest)
```

θα εφαρμόσει μια πρόσθια (forward) διαδικασία όπου θα ξεκινήσει από το μοντέλο `min.model` (μικρότερο μοντέλο) και θα προσθέσει μεταβλητές ψάχνοντας το πολύ μέχρι το μεγαλύτερο μοντέλο που έχει ορίστε από το αντικείμενο `biggest`. Τα μοντέλα αυτά (δηλαδή το μικρότερο και το μεγαλύτερο) μπορούμε να τα ορίσουμε με τη χρήση `formula`.

Το παράδειγμα που ακολουθεί αφορά τα δεδομένα της Βαλτιμόρης και, ουσιαστικά, ψάχνουμε για το καλύτερο μοντέλο σύμφωνα με το AIC, ξεκινώντας από το μοντέλο με μόνο τη σταθερά (`PRICE ~1`) και έχοντας να επιλέξουμε ανάμεσα στις μεταβλητές FIREPL, NROOM, NBATH, PATIO (για λόγους ευκολίας). Στα αποτελέσματα βλέπουμε ότι το αρχικό μοντέλο με μόνο τη σταθερά είχε τιμές AIC =1335.15. Από τις 4 υποψήφιες μεταβλητές, αυτή που έδωσε το καλύτερο AIC ήταν η μεταβλητή FIREPL και, άρα, είναι η πρώτη που μπήκε στο μοντέλο. Στη συνέχεια προστέθηκαν με σειρά οι PATIO, NBATH, ενώ η μεταβλητή NROOM δεν κρίθηκε ότι πρέπει να μπει στο μοντέλο.

```
>  min.model = lm(PRICE ~ 1, data=balt)
>  biggest <- "PRICE ~ FIREPL + NROOM +NBATH + PATIO"
>  biggest
[1] "PRICE ~ FIREPL + NROOM +NBATH + PATIO"
>  fwd.model = step(min.model, direction='forward', scope=biggest)
Start:  AIC=1335.15
PRICE ~ 1
          Df Sum of Sq    RSS    AIC
+ FIREPL   1     32303  84719 1269.0
+ NBATH    1     28079  88943 1279.3
+ PATIO    1     24157  92865 1288.4
+ NROOM    1     20582  96440 1296.3
<none>                117022 1335.2
```

```
Step:  AIC=1268.99
PRICE ~ FIREPL
        Df Sum of Sq   RSS    AIC
+ PATIO  1   13569.9 71149 1234.2
+ NBATH  1   12024.3 72695 1238.7
+ NROOM  1    5712.9 79006 1256.3
<none>               84719 1269.0

Step:  AIC=1234.16
PRICE ~ FIREPL + PATIO
        Df Sum of Sq   RSS    AIC
+ NBATH  1    8674.6 62474 1208.7
+ NROOM  1    2715.6 68433 1228.0
<none>               71149 1234.2
Step:  AIC=1208.73
PRICE ~ FIREPL + PATIO + NBATH
        Df Sum of Sq   RSS    AIC
<none>               62474 1208.7
+ NROOM  1    115.74 62359 1210.3                                    ■.
```

Τα αποτελέσματα του μοντέλου που επιλέγουμε, έχουν αποθηκευτεί στο αντικείμενο `fwd.model` και μπορούμε να τα δούμε:

```
> summary(fwd.model)

Call:
lm(formula = PRICE ~ FIREPL + PATIO + NBATH, data = balt)

Residuals:
    Min      1Q  Median      3Q     Max
-48.581 -11.812  -0.916   9.473  78.842

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   19.750      3.176   6.218 2.74e-09 ***
FIREPL        19.007      3.043   6.245 2.36e-09 ***
PATIO         20.458      3.516   5.819 2.22e-08 ***
NBATH         10.777      2.010   5.361 2.20e-07 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

```
Residual standard error: 17.37 on 207 degrees of freedom
Multiple R-squared:  0.4661,    Adjusted R-squared:  0.4584
F-statistic: 60.25 on 3 and 207 DF,  p-value: < 2.2e-16
                                                          ■.
```

8.7.9 Πρόβλεψη παλινδρόμησης στην R

Η ιδέα της πρόβλεψης αφορά, τόσο την πρόβλεψη των τιμών που έχουμε δει (δηλαδή του $\hat{y}$), όσο και πολύ περισσότερο νέων τιμών, που μπορεί να είναι διαθέσιμες. Στην R είναι διαθέσιμη η εντολή `predict`, η οποία δουλεύει με παρόμοιο τρόπο σε μια ευρεία γκάμα μοντέλων και επιτρέπει την πρόβλεψη. Στην πιο απλή της μορφή:

```
> predictions<- predict(mymodel2)
> head(predictions)
         1          2          3          4          5          6
 -6.615448 101.513015  95.123746  99.281910  64.607355  81.726995
> length(predictions)
[1] 211
```

η εντολή προβλέπει τις τιμές που βρίσκει στο αντικείμενο τύπου `lm`, στην περίπτωση μας τις 211 διαθέσιμες τιμές. Αν θέλουμε πρόβλεψη σε άλλες τιμές, θα πρέπει αυτό να οριστεί κατάλληλα. Οι εντολές που ακολουθούν, ουσιαστικά, χωρίζουν τα δεδομένα σε δυο μέρη. Το πρώτο μέρος `balt2` αποτελείται από 150 παρατηρήσεις και το χρησιμοποιούμε για να φτιάξουμε το μοντέλο `mymodel3`, ενώ οι υπόλοιπες 61 θα χρησιμοποιηθούν μόνο για πρόβλεψη. Έτσι τώρα, η εντολή `predict` χρειάζεται να ορίσουμε στο `newdata` σε ποιο σετ δεδομένων θα κάνουμε πρόβλεψη. Οι τιμές αφορούν τώρα πια τις παρατηρήσεις 151-211 του αρχικού σετ δεδομένων.

```
> balt2<- balt[1:150,]
> balt3<- balt[151:211,]
> mymodel3<- lm(PRICE~., data=balt2)
> predictions<- predict(mymodel3, newdata=balt3)
> length(predictions)
[1] 61
> predictions[1:10]
     151      152      153      154      155      156
19.85106 53.35726 91.54055 41.54340 41.01075 79.15745
     157      158      159      160
41.32286 36.57357 30.08071 68.32414                   ■.
```

Αν θέλουμε να μετρήσουμε την αβεβαιότητα γύρω από την πρόβλεψη, μπορούμε να καθορίσουμε διαστήματα εμπιστοσύνης. Για την πρόβλεψη έχουμε δυο διαφορετικά επίπεδα, εξαρτάται τι θέλουμε να προβλέψουμε: α) μια μεμονωμένη τιμή και β) τη μέση πρόβλεψη. Στη δεύτερη

περίπτωση είναι εύκολο να καταλάβει κανείς, ότι η αβεβαιότητα είναι μικρότερη, αφού πρόκειται για μια μέση τιμή. Για να πάρουμε διαστήματα εμπιστοσύνης χρειαζόμαστε την υποεντολή `interval`. Έτσι, τώρα έχουμε:

```
> mypred2<-predict(mymodel3, newdata=balt3, level=0.95,
+                          interval="prediction")
>
> mypred3<-predict(mymodel3, newdata=balt3, level=0.95,
+                          interval="confidence")
>
> cbind(mypred2[1:5,],mypred3[1:5,])
         fit        lwr       upr      fit      lwr       upr
151 19.85106 -12.214667  51.91679 19.85106 12.50948  27.19263
152 53.35726  21.008059  85.70646 53.35726 44.86263  61.85189
153 91.54055  59.095605 123.98550 91.54055 82.68829 100.39282
154 41.54340   8.299928  74.78687 41.54340 30.10591  52.98089
155 41.01075   9.035819  72.98568 41.01075 34.07648  47.94503    ■.
```

Όπως βλέπουμε και στα παραπάνω αποτελέσματα, για το όρισμα `inteval` έχουμε δύο επιλογές: `confidence` και `prediction`. Η πρώτη επιλογή (`confidence`) δίνει το διάστημα εμπιστοσύνης για την αναμενόμενη τιμή ενώ για τη δεύτερη επιλογή (`prediction`) παίρνουμε το διάστημα εμπιστοσύνης για μια μεμονωμένη τιμή. Για μια μεμονωμένη τιμή, το διάστημα εμπιστοσύνης έχει μεγαλύτερο εύρος αφού η διακύμανσή του προέρχεται κυρίως από την τυχαιότητα του σφάλματος (δηλαδή το σ^2) και από το σφάλμα εκτίμησης της αναμενόμενης τιμής (δηλαδή το $Var(X\widehat{\boldsymbol{\beta}})$).

Σημείωση: θα πρέπει να προσέξουμε να μη χρησιμοποιούμε το μοντέλο μας (ή έστω να το χρησιμοποιούμε πολύ επιφυλακτικά) για πρόβλεψη εκτός του εύρους των επεξηγηματικών μεταβλητών. Η διαδικασία πρόβλεψης εκτός του εύρους των παρατηρούμενων τιμών ονομάζεται extrapolation (παρέκταση). Το βασικό πρόβλημα είναι ότι υποθέτουμε ότι τα δεδομένα συνεχίζουν να έχουν την ίδια συμπεριφορά ακόμα και για τις τιμές που δεν έχουμε παρατηρήσει.

8.7.10 Ορισμός παραμετροποίησης και ψευδομεταβλητών

Ιδιαίτερη προσοχή δώστε στο γεγονός ότι αν μία επεξηγηματική μεταβλητή X είναι κατηγορική και έχει δηλωθεί ως factor, η R θα δημιουργήσει αυτόματα τις ψευδομεταβλητές (dummies). Ο τύπος των ψευδομεταβλητών και οι περιορισμοί που θα επιβάλουμε στις παραμέτρους, που αντιστοιχούν στις επιδράσεις της μεταβλητής αυτής, καθορίζονται με τη συνάρτηση `contrasts` που συνοδεύει τον ορισμό κάθε παράγοντα. Με το που ορίζουμε ένα παράγοντα, προκαθορίζεται η χρήση των γωνιακών περιορισμών (corner constraints), που επιτυγχάνεται με τη χρήση των συνηθισμένων δείκτριων δίτιμων ψευδο-μεταβλητών, που απλά παίρνουν την τιμή ένα, αν βρισκόμαστε σε ένα επίπεδο και μηδέν διαφορετικά. Για να γίνει πιο κατανοητό, ας λάβουμε υπόψη το ακόλουθο απλό παράδειγμα:

```
> x<-factor( rep(1:4,5))
> x
 [1] 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4
Levels: 1 2 3 4
> contrasts(x)
  2 3 4
1 0 0 0
2 1 0 0
3 0 1 0
4 0 0 1
```
■,

όπου ορίζουμε ένα απλό κατηγορικό διάνυσμα μήκους 20 με τρία επίπεδα/κατηγορίες ($\kappa = 4$). Γράφοντας `contrasts(x)`, παίρνουμε ως αποτέλεσμα έναν πίνακα διάστασης 4×3 (ή γενικότερα $\kappa \times (\kappa - 1)$ για παράγοντες με κ κατηγορίες). Ας ονομάσουμε τις τρεις ψευδομεταβλητές D_2, D_3 και D_4, έτσι όπως αντιστοιχούν στις τρεις στήλες του πίνακα. Έτσι, λοιπόν, βλέπουμε ότι το πρώτο επίπεδο (επίπεδο αναφοράς με βάση αυτή την παραμετροποίηση) δίνεται, αν όλες οι ψευδομεταβλητές είναι μηδέν ($D_2 = D_3 = D_4 = 0$). Το δεύτερο αν πάρουμε την τιμή ένα για την D_2 και μηδέν για τις άλλες, και γενικότερα, για κάθε επίπεδο k εκτός της αναφοράς ($1 < k \leq \kappa$) θα πρέπει η αντίστοιχη ψευδομεταβλητή να πάρει την τιμή ένα ($D_k = 1$) και οι υπόλοιπες την τιμη μηδέν. Για το λόγο αυτό λέγονται και δείκτριες (η κάθε ψευδομεταβλητή D_k επισημαίνει αν βρισκόμαστε στο k επίπεδο παίρνοντας την τιμή ένα). Τον τύπο των ψευδομεταβλητών μπορούμε να τον ορίσουμε, τόσο φτιάχνοντας μόνοι μας τον πίνακα των ψευδομεταβλητών (προσοχή θέλει εμπειρία και προσοχή), όσο και με τις έτοιμες εντολές:

- `contr.treatment` (οι συνηθισμένοι γωνιακοί περιορισμοί και προκαθορισμένος τρόπος για ένα νέο παράγοντα),
- `contr.sum` (περιορισμοί μηδενικού αθροίσματος),
- `contr.SAS` (γωνιακοί περιορισμοί με το τελευταίο επίπεδο ως αναφορά),
- `contr.helmert` (περιορισμοί Helmert, όπου κάθε επίπεδο συγκρίνεται με το μέσο όλων των υπολοίπων), και
- `contr.poly` (παραμετροποίηση βασισμένοι σε ορθογώνια πολυώνυμα).

Έτσι, λοιπόν, η παραμετροποίηση με μηδενικού αθροίσματος περιορισμούς για μια μεταβλητή με τέσσερα επίπεδα θα μας δώσει:

```
> contr.sum(4)
  [,1] [,2] [,3]
1    1    0    0
2    0    1    0
```

```
3    0    0    1
4   -1   -1   -1                                                    ■,
```

ενώ για να αλλάξουμε την παραμετροποίηση και τις ψευδομεταβλητές που θα χρησιμοποιούνται σε ένα παράγοντα `x` γράφουμε:

```
> contrasts(x) <- contr.sum(4)
> contrasts(x)
  [,1] [,2] [,3]
1    1    0    0
2    0    1    0
3    0    0    1
4   -1   -1   -1                                                    ■.
```

Τέλος, ενδεικτικά δίνουμε τις ψευδομεταβλητές των προκαθορισμένων contrasts για μια κατηγορική μεταβλητή με 4 επίπεδα για τις υπόλοιπες προκαθορισμένες παραμετροποιήσεις.

```
> contr.SAS(4)
  1 2 3
1 1 0 0
2 0 1 0
3 0 0 1
4 0 0 0
> contr.helmert(4)
  [,1] [,2] [,3]
1   -1   -1   -1
2    1   -1   -1
3    0    2   -1
4    0    0    3
> contr.poly(4)
              .L   .Q         .C
[1,] -0.6708204  0.5 -0.2236068
[2,] -0.2236068 -0.5  0.6708204
[3,]  0.2236068 -0.5 -0.6708204
[4,]  0.6708204  0.5  0.2236068                                     ■.
```

8.7.11 Τελικές παρατηρήσεις για την παλινδρόμηση

Σε αυτή την ενότητα, είδαμε εν τάχει το πώς μπορεί κανείς να κάνει γραμμική (απλή και πολλαπλή) παλινδρόμηση με τη χρήση της R. Ένα από τα βασικά πλεονεκτήματα της R είναι, ότι υπάρχουν πολλές βιβλιοθήκες, οι οποίες μπορούν να πραγματοποιήσουν πιο εξειδικευμένες μεθοδολογίες σχετικές με την παλινδρόμηση, όπως για παράδειγμα έλεγχοι ετεροσκεδαστικότητας, επιλογή μεταβλητών, περισσότερα διαγνωστικά ή διαγράμματα και άλλα πολλά). Για παράδειγμα, η

βιβλιοθήκη `lmtest` παρέχει τη δυνατότητα για έλεγχο Durbin-Watson. Η βιβλιοθήκη `car` μια σειρά από άλλα διαγνωστικά. Άλλες σχετικές βιβλιοθήκες, από τις πολλές διαθέσιμες, είναι οι βιβλιοθηκες `leaps`, `vif` και πολλές ακόμα.

Συνεπώς, αν κάποιος θέλει να εμβαθύνει και να χρησιμοποιήσει την R για προβλήματα παλινδρόμησης, σαφώς και μπορεί να χρησιμοποιήσει πολύ περισσότερες μεθοδολογίες από αυτές που περιγράψαμε σε αυτή την ενότητα.

Βιβλιογραφικές Αναφορές Κεφαλαίου 8

Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle. In *2nd International Symposium on Information Theory, Tsahkadsor, Armenia, USSR (September 2-8, 1971)*, Petrov, B.N.; Csáki, F. (eds.), pages 267−281. Budapest: Akadémiai Kiadó.

Bartlett, M. S. (1937). Properties of sufficiency and statistical tests. *Proceedings of the Royal Society of London Series A*, **160**, 268−282.

Benjamini, Y. & Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society Series B*, **57**, 289−300.

Benjamini, Y. & Yekutieli, D. (2001). The control of the false discovery rate in multiple testing under dependency. *Annals of Statistics*, **29**, 1165−1188.

Brown, M. B. & Forsythe, A. B. (1974). Robust tests for equality of variances. *Journal of the American Statistical Association*, **69**, 364−367.

Durbin, J. & Watson, G. S. (1950). Testing for serial correlation in least squares regression, I. *Biometrika*, **37**, 409−428.

Fisher, R. A. (1922). On the interpretation of χ^2 from contingency tables, and the calculation of p. *Journal of the Royal Statistical Society*, **85**, 87−94.

Fox, J. & Weisberg, S. (2011). *An R Companion to Applied Regression*. Second Edition, Sage Publications.

Hochberg, Y. (1988). A sharper Bonferroni procedure for multiple tests of significance . *Biometrika*, **75**, 800−803.

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, **6**, 65−70.

Hommel, G. (1988). A stagewise rejective multiple test procedure based on a modified Bonferroni test. *Biometrika*, **75**, 383−386.

Kolmogorov, A. (1933). Sulla determinazione empirica di una legge di distribuzione. *G. Ist. Ital. Attuari*, **4**, 83−91.

Kruskal, W. H. & Wallis, W. A. (1952). Use of Ranks in One-Criterion Variance Analysis. *Journal of the American Statistical Association*, **47**(260), 583−621.

Levene, H. (1960). Robust tests for equality of variances. In *Contributions to Probability and Statistics: Essays in Honor of Harold Hotelling*, Ingram Olkin, Harold Hotelling, et alia (eds.), pages 278−292. Stanford University Press.

Lilliefors, H. (1967). On the Kolmogorov-Smirnov test for normality with mean and variance unknown. *Journal of the American Statistical Association*, **62**, 399−402.

Mann, H. B. & Whitney, D. R. (1947). On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *Annals of Mathematical Statistics*, **18**, 50−60.

McNemar, Q. (1947). Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, **12**, 153−157.

Pearson, K. F. (1900). On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *Philosophical Magazine Series 5*, **50**, 157−175.

Pearson, K. F. (1904). *Mathematical contributions to the theory of evolution*. Dulau and Co.

Razali, N. & Wah, Y. B. (2011). Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling tests. *Journal of Statistical Modeling and Analytics*, **2**, 21−33.

Royston, P. (1993). A pocket-calculator algorithm for the Shapiro-Francia test for non-normality: an application to medicine. *Statistics in Medicine*, **12**, 181−184.

Royston, P. (1995). Remark AS R94: A remark on Algorithm AS 181: The W test for normality. *Applied Statistics*, **44**, 547−551.

Schwarz, G. E. (1978). Estimating the dimension of a model. *Annals of Statistics*, **6**, 461−464.

Shapiro, S. S. & Wilk, M. B. (1965). An analysis of variance test for normality (complete samples). *Biometrika*, **52**, 591−611.

Smirnov, N. (1948). Table for estimating the goodness of fit of empirical distributions. *Annals of Mathematical Statistics*, **19**, 279−281.

Stephens, M. (1986). Tests based on edf statistics. In *Goodness-of-Fit Techniques*, D'Agostino, R.B. and Stephens, M.A. (eds.). Marcel Dekker, New York.

Tukey, J. (1949). Comparing Individual Means in the Analysis of Variance. *Biometrics*, **5**, 99−114.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, **1**, 80−83.

ΚΕΦΑΛΑΙΟ 9

Στατιστικές εφαρμογές

Στο κεφάλαιο αυτό θα δούμε κάποια πιο εξειδικευμένα θέματα Στατιστικής και πώς μπορούμε να χρησιμοποιήσουμε την R για να εκτελέσουμε αλγόριθμους, που θα μας επιλύσουν πιο σύνθετα προβλήματα.

9.1 Εκτιμητική

Μια από τις πιο διαδεδομένες μεθόδους εκτίμησης των παραμέτρων στη στατιστική είναι η μέθοδος μεγίστης πιθανοφάνειας. Η ιδέα είναι να υποθέσουμε, πως το τυχαίο δείγμα που έχουμε στα χέρια μας, αποτελείται από ανεξάρτητες τυχαίες μεταβλητές, τις οποίες συνήθως συμβολίζουμε ως $X_1, X_2, ..., X_n$, προέρχεται από έναν πληθυσμό με συνάρτηση (πυκνότητας) πιθανότητας $f(x;\theta)$, όπου θ είναι η άγνωστη παράμετρος, η οποία μπορεί να μην είναι μόνο μια, αλλά να είναι ένα διάνυσμα από παραμέτρους. Για παράδειγμα, στην περίπτωση της κανονικής κατανομής οι παράμετροι είναι $\theta = (\mu, \sigma)$. Η πιθανοφάνεια του δείγματος μας δίνεται ως εξής

$$L(\theta) = \prod_{i=1}^{n} f(x_i;\theta),$$

η οποία είναι συνάρτηση του θ και μας δείχνει την πιθανότητα το δείγμα μας να έχει προέλθει από την κατανομή που υποθέσαμε με τη συγκεκριμένη τιμή του θ. Συνεπώς, θέλουμε να βρούμε την τιμή του θ, η οποία κάνει αυτή την πιθανότητα μεγαλύτερη.

Στην πράξη και κυρίως για λόγους ευκολίας δουλεύουμε με τη λογαριθμική πιθανοφάνεια, δηλαδή την

$$\ell(\theta) = \log L(\theta) = \sum_{i=1}^{n} \log f(x_i;\theta),$$

καθώς τόσο ή απλή όσο και η λογαριθμο-πιθανοφάνεια μεγιστοποιούνται στο ίδιο σημείο.

Η μεγιστοποίηση μπορεί να γίνει με τον κλασικό αλγεβρικό τρόπο, να πάρουμε δηλαδή τις πρώτες παραγώγους και να τις εξισώσουμε με το μηδέν και να λύσουμε το σύστημα που προκύπτει.

Μετά από τις πιθανές λύσεις, εντοπίζουμε αυτές που ικανοποιούν το κριτήριο που χρησιμοποιεί τις δεύτερες παραγώγους έτσι ώστε να έχουμε τοπικό μέγιστο. Σε κάποιες περιπτώσεις, όπως στην ανάλυση παλινδρόμησης, η πιθανοφάνεια μεγιστοποιείται μαθηματικά χωρίς ιδιαίτερα προβλήματα. Υπάρχουν, όμως, αρκετές περιπτώσεις που η εκτιμητές μέγιστης πιθανοφάνειας δεν μπορούν να βρεθούν σε κλειστή μορφή. Στις περιπτώσεις αυτές χρειαζόμαστε υπολογιστικές μεθόδους για την εύρεση τους. Στη συνέχεια, θα δούμε κάποια παραδείγματα όπου μπορούμε να χρησιμοποιήσουμε την R για να μελετήσουμε τέτοια προβλήματα.

9.1.1 Κατανομή Poisson

Μια εύκολη, σχετικά, περίπτωση είναι η κατανομή Poisson. Μπορεί κανείς εύκολα να δείξει πως η εκτιμήτρια μέγιστης πιθανοφάνειας είναι απλά η μέση τιμή του δείγματος. Η κατανομή Poisson έχει συνάρτηση πιθανότητας που δίνεται από τον τύπο

$$P(X=x)=\frac{e^{-\theta}\theta^{x}}{x!},\ x=0,1,...,\theta>0\ ,$$

συνεπώς, οι συναρτήσεις της πιθανοφάνειας και της λογαριθμικής πιθανοφάνειας είναι οι εξής:

$$L(\theta)=\prod_{i=1}^{n}\frac{e^{-\theta}\theta^{x_i}}{x_i!}=e^{-n\theta}\theta^{\Sigma x_i}\left(\prod_{i=1}^{n}x_i!\right)^{-1},$$

$$\ell(\theta)=-n\theta+\sum_{i=1}^{n}x_i\log\theta-\log\left(\prod_{i=1}^{n}x_i!\right)\ .$$

Η συνάρτηση που ακολουθεί, υπολογίζει τη λογαριθμική πιθανοφάνεια για ένα δείγμα για δοθείσα τιμή του θ.

```
poisson.loglikelihood<-function(data,theta) {
        temp<-log(dpois(data,theta))
        sum(temp)
}                                                                    ■.
```

Χρησιμοποιούμε τη συνάρτηση που έχει η R για τη συνάρτηση πιθανότητας της κατανομής Poisson. Έχοντας αυτή τη συνάρτηση στα χέρια μας, μπορούμε να υπολογίσουμε για ένα πλήθος από διαφορετικές τιμές του θ την πιθανοφάνεια και είτε να φτιάξουμε ένα διάγραμμα της συνάρτησης πιθανοφάνειας (ή του λογαρίθμου της), είτε να βρούμε εμπειρικά πού η συνάρτηση αυτή μεγιστοποιείται. Η συνάρτηση που ακολουθεί κάνει αυτά τα δυο:

```
plotloglik<-function(data){
            k<-1000
            theta<-seq(0.01,5,length=k)
             results<-rep(NA,k)
             for (i in 1:k) {
                  results[i]<-poisson.loglikelihood(data,
```

```
                    theta[i])
                }
                plot(theta,results,xlab="theta",
                ylab="loglikelihood", type="l")
                list(theta=theta,loglik=results,
                maxim=theta[order(results)[k]])
}                                                    ■.
```

Στη συνάρτηση αυτή, δημιουργούμε με την εντολή `seq` ένα διάνυσμα με τις διαφορετικές τιμές που θέλουμε για το θ. Στο παράδειγμα μας παίρνουμε 1000 τιμές στο διάστημα 0.01 έως 5. Θα πρέπει να σημειώσουμε πως το εύρος των τιμών θα μπορούσε να είναι παράμετρος εισόδου στη συνάρτηση ώστε να την κάνουμε πολύ γενική. Στη συνέχεια, υπολογίζουμε τη λογαριθμική πιθανοφάνεια για καθένα από αυτά τα σημεία και κατασκευάζουμε το διάγραμμα με την εντολή `plot`. Τι ακριβώς, όμως, κάνει η τελευταία εντολή της συνάρτησης; Υπολογίζει το μέγιστο, δηλαδή βρίσκει την τιμή από τις 1000 τιμές που έχουμε χρησιμοποιήσει στο διάνυσμα `theta` που έχει μεγαλύτερη λογαριθμική πιθανοφάνεια.

Εδώ, είναι χρήσιμο να εξηγήσουμε λίγο τη συνάρτηση `order`. Η συνάρτηση αυτή παίρνει σαν είσοδο ένα διάνυσμα και επιστρέφει ως αποτέλεσμα ένα διάνυσμα με ακέραιους αριθμούς, έτσι ώστε στην i θέση του διανύσματος να υπάρχει η ακέραια τιμή που δηλώνει τη θέση στο αρχικό διάνυσμα. Το παρακάτω παράδειγμα είναι κατατοπιστικό:

```
> test<-c(10,12,4,23,18)
> order(test)
[1] 3 1 2 5 4
>                                                    ■.
```

Η πρώτη τιμή του διανύσματος `order`, η οποία είναι το 3, υποδηλώνει πως η μικρότερη παρατήρηση του διανύσματος `test` είναι στη 3η θέση του διανύσματος `test`. Όντως, μπορεί κανείς εύκολα να επαληθεύσει πως η τιμή 4 είναι η μικρότερη. Στη θέση 2 του διανύσματος `order` υπάρχει η τιμή 1, που υποδηλώνει πως η 2η μικρότερη παρατήρηση για το διάνυσμα `test` υπάρχει στην 1η θέση του διανύσματος και ούτω καθεξής. Αν γυρίζουμε πίσω στη συνάρτηση μας, βρίσκουμε την τιμή

```
   order(results)[k]
```

που σύμφωνα με τα προηγούμενα θα μας επιστρέψει τη θέση που υπάρχει η μεγαλύτερη τιμή, δηλαδή η μεγαλύτερη πιθανοφάνεια. Συνεπώς,

```
   theta[order(results)[k]]
```

είναι το θ με τη μεγαλύτερη τιμή για τη λογαριθμική πιθανοφάνεια. Θυμηθείτε, ότι αυτό δεν σημαίνει πως είναι ακριβώς το μέγιστο, καθώς εμείς έχουμε πάρει μόνο ένα μικρό αριθμό σημείων, αλλά σίγουρα είναι μια τιμή κοντά στο μέγιστο.

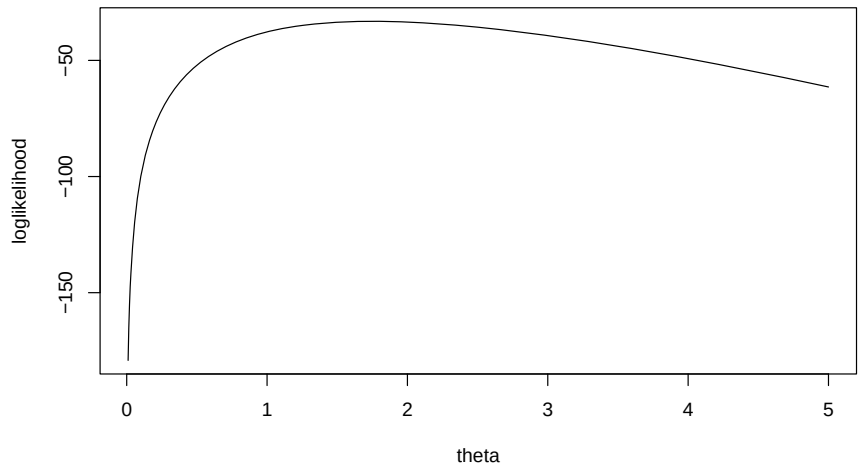
Παράδειγμα 9.1 *Τα δεδομένα που ακολουθούν αναφέρονται στον αριθμό ατυχημάτων σε 20 διασταυρώσεις μιας πόλης για ένα έτος.*

1, 1, 0, 3, 1, 1, 3, 2, 1, 1, 1, 1, 1, 3, 7, 2, 3, 2, 0, 1

Αν υποθέσουμε πως η κατανομή των δεδομένων είναι η κατανομή Poisson, από τη θεωρία ξέρουμε πως η εκτιμήτρια μεγίστης πιθανοφάνειας (ΕΜΠ) της είναι $\hat{\theta} = \bar{x} = 1.75$. Χρησιμοποιώντας τις συναρτήσεις που ορίσαμε πριν, παίρνουμε τα εξής αποτελέσματα:

```
> x<-c(1, 1, 0, 3, 1, 1, 3, 2, 1, 1, 1, 1, 1, 3, 7, 2, 3, 2, 0, 1)
> solution<-plotloglik(x)
> solution$maxim
[1] 1.748258
> plot(solution$theta, solution$loglik, xlab="theta", ylab="
   loglikelihood", type="l")                                    ■.
```

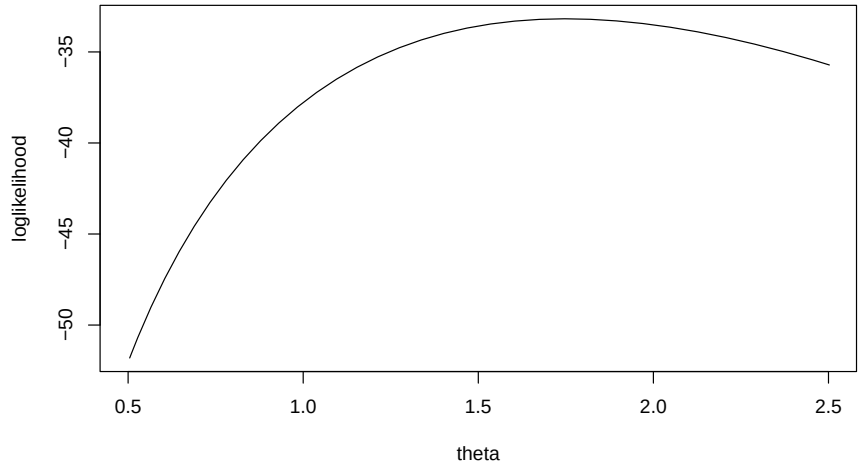
Το Διάγραμμα 9.1 παράγει η συνάρτηση που έχουμε ορίσει. Η μέγιστη τιμή βρέθηκε στο 1.748, πολύ κοντά στη μέση τιμή, η απόκλιση οφείλεται αποκλειστικά και μόνο στο γεγονός πως έχουμε πάρει ένα μικρό πλήθος σημείων.



Διάγραμμα 9.1: Η λογαριθμική συνάρτηση πιθανοφάνειας Poisson για το διάνυσμα x

Από το Διάγραμμα 9.1 βλέπει κανείς τη μορφή της λογαριθμικής πιθανοφάνειας ως συνάρτηση του θ. Η συνάρτηση έχει ξεκάθαρα ένα μέγιστο. Το Διάγραμμα 9.2 εμφανίζει την ίδια πιθανοφάνεια, αλλά σε μικρότερο εύρος τιμών και διαφορετική κλίμακα, χρησιμοποιώντας την ακόλουθη σύνταξη:

```
> plot(solution$theta[100:500], solution$loglik[100:500], xlab="
   theta", ylab="loglikelihood", type="l")                      ■.
```



Διάγραμμα 9.2: Η λογαριθμική συνάρτηση πιθανοφάνειας Poisson για το διάνυσμα x (για περιορισμένο εύρος τιμών θ)

Από την παραπάνω ανάλυση, καταλαβαίνει κανείς πως τέτοιου είδους διαγράμματα προσφέρουν μια καλή εποπτική δυνατότητα για την εύρεση εκτιμητών μεγίστης πιθανοφάνειας, ειδικά σε περιπτώσεις που αυτοί δεν δίνονται σε κλειστή μορφή. Μια τέτοια περίπτωση θα δούμε αμέσως τώρα.

9.1.2 Ομοιόμορφη κατανομή

Ας υποθέσουμε πως τα δεδομένα μας προέρχονται από μια ομοιόμορφη κατανομή στο διάστημα $(0, \theta)$ και πως θέλουμε να εκτιμήσουμε την άγνωστη παράμετρο θ. Σε αυτό το μοντέλο η συνάρτηση (μάζας) πυκνότητας πιθανότητας είναι η

$$f(x) = \frac{1}{\theta} I(0 < x < \theta)$$

για οποιοδήποτε πραγματικό αριθμό $x > 0$• όπου $I(A)$ είναι μια δείκτρια συνάρτηση που παίρνει την τιμή ένα όταν η συνθήκη A είναι αληθής και τη τιμή μηδέν διαφορετικά.

Συνεπώς, για ένα τυχαίο δείγμα $X_1, X_2, ..., X_n$ η πιθανοφάνεια και η λογαριθμική πιθανοφάνεια γίνονται

$$\begin{aligned} L(\theta) &= \prod_{i=1}^{n} \left\{ \frac{1}{\theta} I(0 < x_i < \theta) \right\} = \left(\frac{1}{\theta} \right)^n \prod_{i=1}^{n} I(0 < x_i < \theta) \\ &= \left(\frac{1}{\theta} \right)^n I(\max(\boldsymbol{x}) < \theta) \prod_{i=1}^{n} I(x_i > 0) \\ \ell(\theta) &= n \log \theta \text{ όταν } \max(\boldsymbol{x}) < \theta \text{ και } x_i > 0 \text{ για όλα τα } i \in \{1, \ldots, n\}, \end{aligned}$$

όπου $\max(\boldsymbol{x}) = \max(x_1, x_2, \ldots, x_n)$.

Μπορούμε λοιπόν να δούμε ότι η λογαριθμο-πιθανοφάνεια ορίζεται μόνο όταν η μέγιστη τιμή όλων των θετικών x_i είναι μικρότερη από την παράμετρο θ. Όταν αυτός ο περιορισμός ισχύει, τότε η παράγωγος της λογαριθμο-πιθανοφάνειας δεν γίνεται ποτέ μηδέν και, επομένως, δεν μπορούμε να μεγιστοποιήσουμε με τον κλασικό τρόπο. Η μελέτη της συνάρτησης πιθανοφάνειας μπορεί να βοηθήσει στην περίπτωση μας. Οι συναρτήσεις που ακολουθούν υπολογίζουν, αφενός την λογαριθμική συνάρτηση πιθανοφάνειας για ένα δείγμα και αφετέρου φτιάχνουν ένα διάγραμμα της συνάρτησης αυτής. Για την υλοποίηση δεν έχουμε χρησιμοποιήσει την εντολή `dunif`, που υπολογίζει αυτόματα τη συνάρτηση (μάζας) πυκνότητας πιθανότητας, αλλά τη συνάρτηση $1/\theta$. Αυτό εμπεριέχει ένα σημαντικό λάθος. Αν δούμε τον ορισμό της σ.π.π. της ομοιόμορφης κατανομής, τότε $f(x) = 0$, αν $x > \theta$, οπότε ο τρόπος που υπολογίζουμε την λογαριθμική πιθανοφάνεια δεν είναι σωστός. Αυτό το σχόλιο είναι σημαντικό για να ξεχωρίσουμε τη στατιστική σημασία της μεγιστοποίησης, που δεν είναι ακριβώς ισοδύναμη με τη μαθηματική. Στη δεύτερη συνάρτηση χρησιμοποιούμε τη σωστή συνάρτηση:

```
uniform.loglikelihood <- function(data,theta) {
       temp<-log(1/theta)
       sum(temp)
}
#
uniform.loglikelihood2 <- function(data,theta) {
       temp<-log(dunif(data,0,theta))
       sum(temp)
}
#
plotloglikunif<-function(data) {
       k<-1000
       theta<-seq(0.01,2,length=k)
       results<-rep(NA,k)
       for (i in 1:k) {
            results[i]<-uniform.loglikelihood2(data, theta[i])
       }
       plot(theta,results,xlab="theta",
      ylab="loglikelihood", type="l")
      list(theta=theta,loglik=results)
}
```
■.

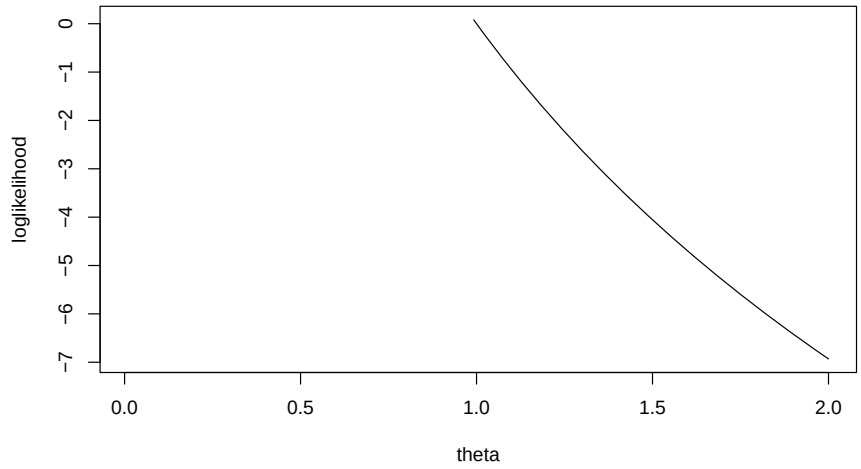
Όταν εκτελέσουμε τη συνάρτηση με τη χρήση της `uniform.loglikelihood2`, τότε θα πάρουμε μια σειρά από τιμές `inf`, καθώς όταν $f(x) = 0$, ο λογάριθμος δεν υπολογίζεται. Συνεπώς, η λογαριθμική πιθανοφάνεια υπάρχει μόνο όταν $\theta > X_i$, για κάθε i.

Έστω, δείγμα μεγέθους 10 από την ομοιόμορφη κατανομή που συζητάμε. Οι τιμές που πήραμε είναι οι

```
x<-c(0.54394762, 0.97010427, 0.75747115, 0.72676165, 0.37344721,
     0.99134905, 0.87447262, 0.06558424, 0.67115599, 0.17419927)
```

■.

Το διάγραμμα της λογαριθμικής συνάρτησης πιθανοφάνειας εμφανίζεται στο Διάγραμμα 9.3. Είναι μια φθίνουσα συνάρτηση του θ. Συνεπώς, το μέγιστο είναι στο αριστερό όριο. Όμως, με βάση αυτά που είπαμε πρίν, η λογαριθμική συνάρτηση πιθανοφάνειας ορίζεται μόνο για $\theta > X_i$ για κάθε i. Επομένως, $\hat{\theta} = X_{\max}$, δηλαδή η μεγαλύτερη παρατήρηση.



Διάγραμμα 9.3: Η λογαριθμική συνάρτηση πιθανοφάνειας της ομοιόμορφης κατανομής για το διάνυσμα x της Ενότητας 9.1.2

Είναι ενδιαφέρον πως για το συγκεκριμένο πρόβλημα, η δημιουργία του διαγράμματος, είναι ιδιαίτερα χρήσιμη.

9.1.3 Μέθοδος Newton-Raphson

Σε πολλές περιπτώσεις, η λύση της εξίσωσης $f(x) = 0$ δεν είναι τόσο εύκολη. Για παράδειγμα, ενώ για μια πολυωνυμική εξίσωση η λύση μπορεί να βρεθεί σχετικά εύκολα, η εξίσωση

$$x + \exp(x + \log x)x - 5 = 0$$

δεν είναι εύκολο να λυθεί.

Σε τέτοιες περιπτώσεις καταφεύγουμε σε αριθμητική επίλυση των εξισώσεων. Στη βιβλιογραφία υπάρχουν πολλές τέτοιες μέθοδοι. Στις σημειώσεις αυτές θα ασχοληθούμε με τη μέθοδο Newton-Raphson και μάλιστα μόνο με την περίπτωση εξισώσεων μιας μεταβλητής.

Η θεωρία σχετικά με τη μέθοδο Newton-Raphson είναι τεράστια και δεν θα μας απασχολήσουν εδώ οι λεπτομέρειές της. Η ιδέα είναι πως η λύση της εξίσωσης προκύπτει επαναληπτικά. Δηλαδή,

αν μετά από r βήματα η λύση είναι x_r, τότε στο επόμενο βήμα υπολογίζουμε την νέα τιμή

$$x_{r+1} = x_r - \frac{f(x_r)}{f'(x_r)} \ .$$

όπου
$f'(x)$ είναι η παράγωγος της συνάρτησης.

Παράδειγμα 9.2 . *Έστω η εξίσωση* $x^3 + 8x^2 = 6$.

Μπορούμε να τη γράψουμε στη μορφή $f(x) = x^3 + 8x^2 - 6 = 0$ (πρέπει πάντα να την εκφράζουμε έτσι ώστε στο δεξί μέλος να έχουμε μηδέν). Τότε $f'(x) = 3x^2 + 16x$, οπότε το σχήμα γίνεται

$$x_{r+1} = x_r - \frac{x_r^3 + 8x_r^2 - 6}{3x_r^2 + 16x_r} \ .$$

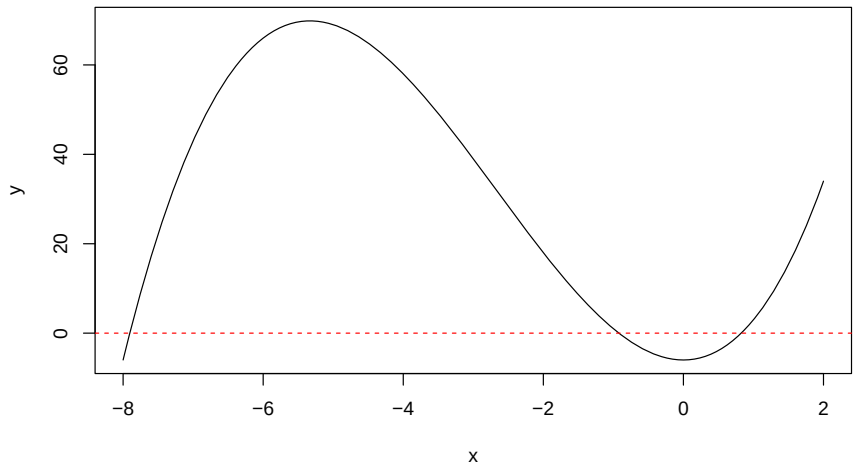
Συνεπώς, χρειαζόμαστε μια αρχική τιμή, έστω $x_0 = 1$. Με τη χρήση του παραπάνω τύπου βρίσκουμε

$$\begin{aligned} x_1 &= 0.8421053 \\ x_2 &= 0.8247795 \\ x_3 &= 0.8245724 \\ x_4 &= 0.8245724 \end{aligned}$$

Στη συνέχεια η τιμή δεν αλλάζει. Μπορεί κάποιος να επαληθεύσει πως όντως η τιμή 0.8245724 είναι μια λύση της εξίσωσης. Οι άλλες δύο λύσεις της εξίσωσης μπορούν να βρεθούν, αν ξεκινήσουμε από διαφορετικές αρχικές τιμές. Για παράδειγμα, αν $x_0 = -1$ οδηγούμαστε στη λύση -0.9206146, ενώ αν η αρχική τιμή είναι $x_0 = -10$ οδηγούμαστε στη λύση -7.903958. Γενικά, σε τέτοιους αλγορίθμους αριθμητικής επίλυσης εξισώσεων η αρχική τιμή παίζει πολύ σημαντικό ρόλο και, επομένως, είναι χρήσιμο να δοκιμάζει κανείς με διάφορες αρχικές τιμές, ώστε να σιγουρέψει την εύρεση όλων των ριζών της εξίσωσης. Στο Διάγραμμα 9.4 μπορεί κανείς να δει τη συνάρτηση $f(x) = x^3 + 8x^2 - 6$, που όντως συναντά το μηδέν στις ρίζες που βρήκαμε.

Οι εντολές που χρειαστήκαμε για την εύρεση των ριζών είναι οι εξής:

```
> x<-1
> crit<-10
> while (crit>10^(-6)) {
        xnew<-x - (x^3 + 8*x^2 -6)/(3*x^2 + 16*x)
        crit<-abs(xnew-x)
        x<-xnew
        print(c(crit,x))
}
```



Διάγραμμα 9.4: Διάγραμμα της συνάρτησης $f(x) = x^3 + 8x^2 - 6$

```
[1] 0.1578947 0.8421053
[1] 0.01732581 0.82477946
[1] 0.0002070338 0.8245724220
[1] 2.947253e-008 8.245724e-001
```
■.

Η μεταβλητή `crit`, ουσιαστικά, χρησιμοποιείται σαν μεταβλητή ελέγχου για να σταματήσουμε τις επαναλήψεις. Όταν σε δύο διαδοχικές επαναλήψεις η λύση δεν βελτιωθεί περισσότερο από 10^{-6}, τότε ο αλγόριθμος σταματά. Η μεταβλητή `xnew` χρειάζεται ώστε να μπορούμε να συγκρίνουμε κάθε φορά την καινούρια λύση με την προηγούμενη. Στο παράδειγμά μας δεν ξέρουμε πόσες επαναλήψεις θα χρειαστούμε ώστε να τελειώσει ο αλγόριθμος και για αυτό χρησιμοποιούμε την εντολή `while` αντί της εντολής `for`.

Εφαρμογή στη Στατιστική

Πολλές φορές στη Στατιστική, όταν θέλουμε να βρούμε την ΕΜΠ, δεν μπορούμε να τη βρούμε σε κλειστή μορφή. Αν πάρουμε την παράγωγο της λογαριθμικής πιθανοφάνειας, τότε η εξίσωση δεν λύνεται σε κλειστή μορφή και πρέπει να λυθεί αριθμητικά. Η μέθοδος Newton-Raphson που είδαμε, μπορεί να δώσει λύση. Για παράδειγμα, έστω ότι η κατανομή του πληθυσμού έχει συνάρτηση πιθανότητας

$$P(X = x) = \frac{e^{-\theta}\theta^x}{(1 - e^{-\theta})x!},\ x = 1, 2, ...,\ \theta > 0\ .$$

Συνεπώς, η πιθανοφάνεια και η λογαριθμική πιθανοφάνεια είναι οι εξής:

$$\begin{aligned} L(\theta) &= \prod_{i=1}^{n} \frac{e^{-\theta}\theta^{x_i}}{(1-e^{-\theta})x_i!} = \left(\frac{e^{-\theta}}{1-e^{-\theta}}\right)^n \theta^{\Sigma x_i} \left(\prod_{i=1}^{n} x_i!\right)^{-1}, \\ \ell(\theta) &= -n\theta - n\log(1 - e^{-\theta}) + \sum_{i=1}^{n} x_i \log\theta - \log\left(\prod_{i=1}^{n} x_i!\right) \end{aligned}\ .$$

και η ΕΜΠ είναι η λύση της εξίσωσης

$$\frac{d\ell(\theta)}{d\theta} = -n - \frac{ne^{-\theta}}{1-e^{-\theta}} + \frac{\sum_{i=1}^{n} x_i}{\theta} = 0 \Leftrightarrow$$

$$\theta - \bar{x}(1-e^{-\theta}) = 0\ .$$

Η εξίσωση αυτή δεν μπορεί να λυθεί σε κλειστή μορφή και άρα καταφεύγουμε στη μέθοδο Newton-Raphson. Για τη συγκεκριμένη συνάρτηση έχουμε πως

$$\begin{array}{l} f(\theta) = \theta - \bar{x}(1-e^{-\theta}) = 0 \\ f'(\theta) = 1 - \bar{x}e^{-\theta} \end{array},$$

και, επομένως,

$$\theta_{r+1} = \theta_r - \frac{\theta_r - \bar{x}(1-e^{-\theta_r})}{1-\bar{x}e^{-\theta_r}}$$

για κάποια αρχική τιμή θ_0.

Οι συναρτήσεις που ακολουθούν υπολογίζουν η πρώτη την πιθανοφάνεια (έχει παραλειφθεί η σταθερά $-\log\left(\prod_{i=1}^{n} x_i!\right)$) και η δεύτερη λύνει την εξίσωση για δοθείσα αρχική τιμή

```
likelihood<-function(y, theta){
     # the constant has been removed!!
     n <- length(y)
     loglike <- -n *theta + sum(y) * log(theta) -  n * log(1-exp(-
   theta))
     return(loglike)
}
solveequation<-function(y,theta=mean(y)) {
     thetaold<-mean(y)
     vrethike<-10
     while (vrethike==10) {
           thetanew<-thetaold - (thetaold - mean(y)*(1-exp(-
   thetaold)))/(1- mean(y)* exp(-thetaold))
          if (abs(thetanew-thetaold)<10^-6) {
                vrethike<-20
          }
          thetaold<-thetanew
     }
     thetaold
}
```

■.

Παράδειγμα 9.3 *Τα δεδομένα αφορούν τον αριθμό των ατυχημάτων και πάλι, αλλά μόνο διασταυρώσεις, όπου υπήρξε τουλάχιστον ένα ατύχημα κατά τη χρονιά που μελετάμε. Συνεπώς, δεν μπορούμε να έχουμε μηδενικές παρατηρήσεις. Τα δεδομένα είχαν τις τιμές*

1 2 1 1 2 2 6 2 1 1 2 1 1 2 1 1 2 2 2 1

Να βρεθεί η ΕΜΠ του υποθέτοντας πως η κατανομή του πληθυσμού είναι η

$$P(X = x) = \frac{e^{-\theta}\theta^x}{(1 - e^{-\theta})x!},\ x = 1, 2, ...,\ \theta > 0\ .$$

Σύμφωνα με όσα είπαμε, και με τη χρήση της συνάρτησης που ορίσαμε πιο πάνω, βρίσκουμε πως $\hat{\theta} = 1.17$. Παρατηρείστε πως έχουμε χρησιμοποιήσει πολλές αρχικές τιμές και πάντα το αποτέλεσμα ήταν το ίδιο.

```
> x<-c(1, 2, 1, 1, 2, 2, 6, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 2, 2, 1)
> solveequation(x)
[1] 1.175016
> solveequation(x, 5)
[1] 1.175016
> solveequation(x, 0.1)
[1] 1.175016
> solveequation(x, 2)
[1] 1.175016}
```
■.

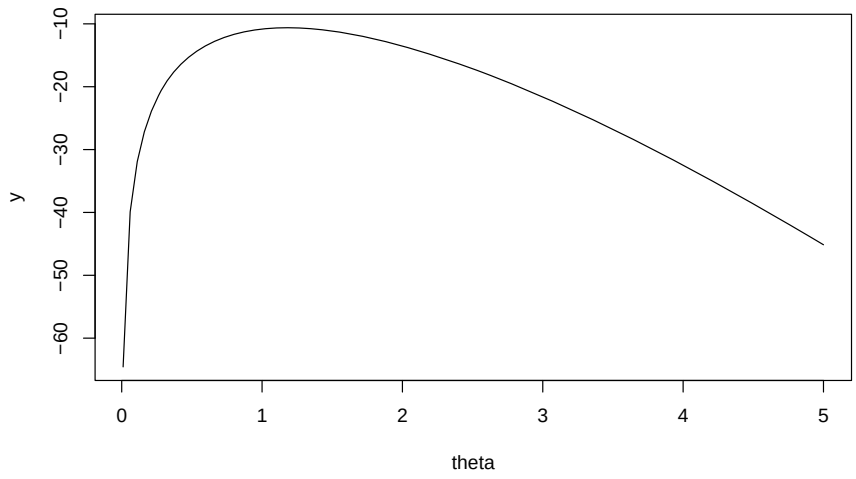
Μερικές παρατηρήσεις σχετικά με τις συναρτήσεις που χρησιμοποιήσαμε:

- Στην συνάρτηση για τη λογαριθμική πιθανοφάνεια δεν έχουμε χρησιμοποιήσει τον όρο $\log\left(\prod_{i=1}^{n} x_i!\right)$. Αφενός δεν δημιουργεί κάποιο πρόβλημα στην μεγιστοποίηση, καθώς δεν περιέχει την παράμετρο, αφετέρου πρέπει να έχετε υπόψη σας πως το παραγοντικό μπορεί να οδηγήσει τον υπολογιστή σε υπερχείλιση (overflow), βλ. Ενότητα 9.4.1, δηλαδή σε έναν πολύ μεγάλο αριθμό που ο υπολογιστής δεν μπορεί να υπολογίσει. Τέτοια προβλήματα μπορούν να ξεπεραστούν με διάφορα τρικ. Για παράδειγμα, στην περίπτωσή μας, αντί να υλοποιήσουμε το παραγοντικό με τη συνάρτηση της R `factorial`, μπορούμε να δούμε πως ο λογάριθμος του γινομένου είναι ίσος με το άθροισμα των λογαρίθμων. Έτσι, ο λογάριθμος ενός παραγοντικού είναι και πάλι ένα άθροισμα λογαρίθμων, που είναι πολύ πιο εύκολο να υλοποιηθεί χωρίς σφάλμα.

- Σε αυτή τη συνάρτηση χρησιμοποιούμε εναλλακτικά το κριτήριο τερματισμού με τη χρήση μιας μεταβλητής που αλλάζει τιμή, αν και μόνο αν το κριτήριο τερματισμού ικανοποιείται.

- Παρατηρείστε πως ορίζουμε μια προκαθορισμένη αρχική τιμή για την περίπτωση που ο χρήστης δεν ξέρει τι αρχική τιμή να δώσει. Αυτή η αρχική τιμή είναι η μέση τιμή του δείγματος. Επομένως, η προκαθορισμένη τιμή για την παράμετρο εισόδου μιας συνάρτησης, μπορεί να είναι μια συνάρτηση άλλων παραμέτρων εισόδου.

- ΄Ενα διάγραμμα της λογαριθμικής συνάρτησης πιθανοφάνειας, χωρίς τη σταθερά που αναφέραμε, μπορείτε να δείτε στο Διάγραμμα 9.5.



Διάγραμμα 9.5: Η λογαριθμική συνάρτηση πιθανοφάνειας για το Παράδειγμα 9.3

9.2 Εμπειρική συνάρτηση κατανομής

Μια σημαντική ποσότητα στην ανάλυση δεδομένων και την περιγραφική στατιστική είναι η εμπειρική συνάρτηση κατανομής (ε.σ.κ.). Εκτός των άλλων, μας επιτρέπει μια οπτική σύγκριση των δεδομένων με την κατανομή που έχουμε υποθέσει, αλλά και εναλλακτικές κατανομές στην περίπτωση που δεν ταιριάζει με την κατανομή που έχουμε υποθέσει. Εκτός από αυτά, η εμπειρική συνάρτηση κατανομής έχει ποικίλες εφαρμογές σε πολλές άλλες στατιστικές μεθόδους, όπως σε ελέγχους υποθέσεων.

Η ε.σ.κ. των δεδομένων ορίζεται ως:

$$S(x) = \frac{\sum_{i=1}^{n} I(X_i \leq x)}{n},$$

δηλαδή η ε.σ.κ. στο σημείο x είναι το ποσοστό των παρατηρήσεων που είναι μικρότερες ή ίσες από αυτή την τιμή.

Η συνάρτηση που ακολουθεί υλοποιεί αυτόν τον τύπο:

```
#function for calculating the
#empirical distribution function
cumul<-function(x) {
       fx<-rep(NA,100)
```

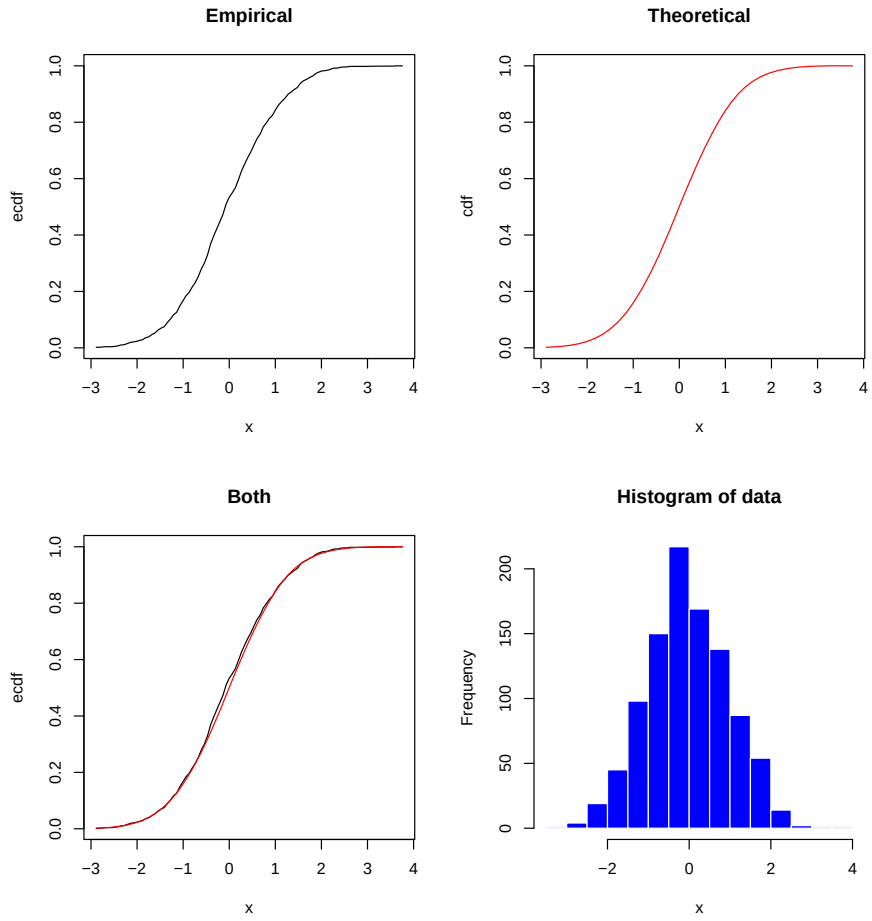
```
        ll<-0.95*min(x)
        ul<-1.05*max(x)
        n<-length(x)
        y<-seq(ll,ul,length=100)
        for (i in 1:100) {
            temp<-0
            for (j in 1:n) {
                if (y[i]>=x[j]) temp<-temp+1/n
            }
            fx[i]<-temp
        }
        list(points=y,empir=fx)
}
```
■.

Παρατηρήσεις: Κατά αρχάς πρέπει να βρούμε τα άκρα του διαστήματος μέσα στο οποίο θα υλοποιήσουμε την ε.σ.κ. Διαλέξαμε αυτά τα άκρα να είναι 0.95 φορές η μικρότερη παρατήρηση και 1.05 φορές η μεγαλύτερη, έτσι είμαστε σίγουροι πως περιλαμβάνουμε όλες τις παρατηρήσεις. Στη συνέχεια παίρνουμε πολλές τιμές μέσα σε αυτό το διάστημα, τις οποίες χρειαζόμαστε για να δημιουργήσουμε το διάγραμμα. Τέλος, για κάθε μια από αυτές τις τιμές μετράμε πόσες παρατηρήσεις είναι μικρότερες ή ίσες. Η συνάρτηση επιστρέφει ως αποτελέσματα, τόσο τα σημεία που υπολογίσαμε την ε.σ.κ., όσο και τις τιμές της ώστε να κατασκευαστεί εκ των υστέρων το διάγραμμά της.

Στο παράδειγμα που ακολουθεί, προσομοιώνουμε 10009 τιμές από τυποποιημένη κανονική κατανομή και δημιουργούμε, τόσο την εμπειρική κατανομή της, όσο και τη θεωρητική της κατανομή.

Οι εντολές που χρησιμοποιήθηκαν για το Διάγραμμα 9.6 δίνονται αμέσως πιο κάτω. Αυτό που είναι ενδιαφέρον είναι πως, αν χρησιμοποιήσουμε τις συναρτήσεις κατανομών, η ομοιότητα της εμπειρικής με την θεωρητική είναι προφανής. Κάτι τέτοιο δεν είναι εύκολο να το δούμε σε ένα απλό ιστόγραμμα. Παρατηρείστε πως το ιστόγραμμα παρουσιάζει μια ασυμμετρία. Αυτός είναι και ένας λόγος για τη χρήση της συνάρτησης κατανομής, αντί του ιστογράμματος.

```
data<-rnorm(1000)
cumuldata<-cumul(data)
par(mfrow=c(2,2))
plot(cumuldata$points,cumuldata$empir,xlab="x", ylab="ecdf",type="l
   ",main="Empirical")
plot(cumuldata$points,pnorm(cumuldata$points), xlab="x",ylab="cdf",
   type="l",main="Theoretical", col=2)
plot(cumuldata$points,cumuldata$empir,xlab="x", ylab="ecdf", type="
   l",main="Both")
lines(cumuldata$points,pnorm(cumuldata$points), col=2)
```



Διάγραμμα 9.6: Η ε.σ.κ., η πραγματική συνάρτηση κατανομής και η σύγκριση τους.

```
hist(data,xlab="x", col='blue', border='white')
```
■.

Θα πρέπει να σημειωθεί πως η συνάρτηση, επειδή χρησιμοποιεί την επαναληπτική εντολή `for`, μπορεί να γίνει πολύ αργή, όταν το μέγεθος του δείγματος είναι πολύ μεγάλο.

9.3 Κεντρικό Οριακό Θεώρημα

Το κεντρικό οριακό θεώρημα (ΚΟΘ) είναι θεμελιώδες στη στατιστική επιστήμη και πάνω σε αυτό βασίζονται πολλά αποτελέσματα που χρησιμοποιούμε κατά κόρον στη στατιστική.

΄Εστω ανεξάρτητες τυχαίες μεταβλητές $X_1, X_2, ..., X_n$, κάθε μια ακολουθεί κάποια κατανομή με αναμενόμενη τιμή $E(X_i) = \mu_i$ και διακύμανση $Var(X_i) = \sigma_i^2$. Τότε, η τυχαία μεταβλητή

$$S_n = \frac{\sum_{i=1}^{n} X_i - \sum_{i=1}^{n} \mu_i}{\sqrt{\sum_{i=1}^{n} \sigma_i^2}}$$

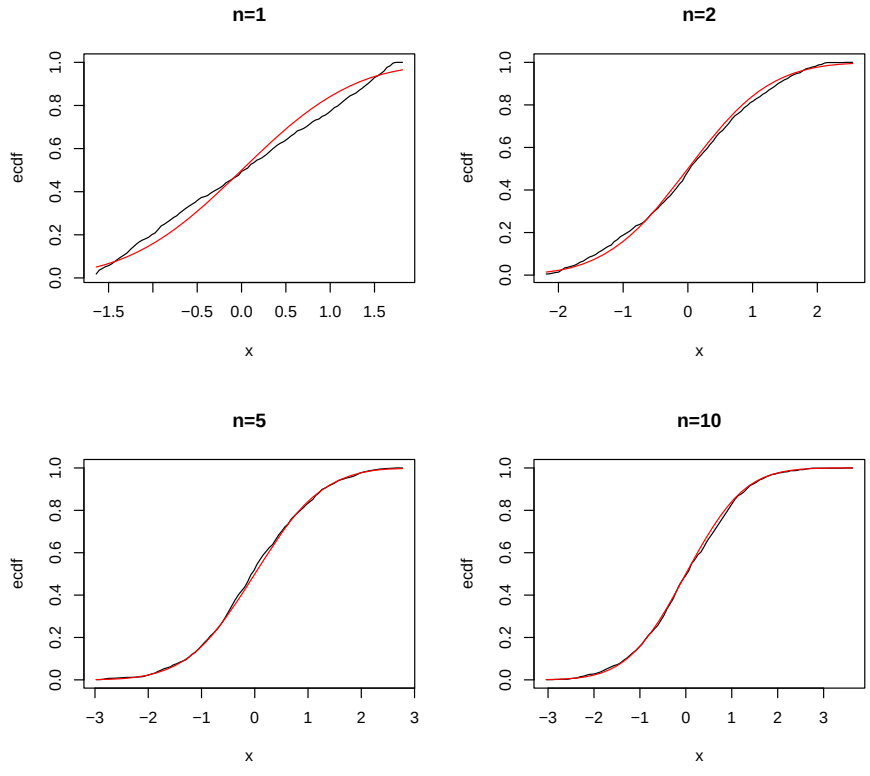
ακολουθεί, ασυμπτωτικά (για μεγάλες τιμές του n), την τυποποιημένη κανονική κατανομή.

Η σημασία του ΚΟΘ είναι πολύ μεγάλη, γιατί ουσιαστικά λέει πως κάθε άθροισμα ανεξάρτητων τυχαίων μεταβλητών που έχουν μέση τιμή και διακύμανση, όσο μεγαλώνει το πλήθος των όρων του αθροίσματος, προσεγγίζει την κανονική κατανομή.

Ας δούμε στην πράξη λοιπόν το κεντρικό οριακό θεώρημα. Θα προσομοιώσουμε δείγματα από κάποιες κατανομές και θα δούμε κατά πόσον και πόσο γρήγορα η κατανομή του αθροίσματος προσεγγίζει την κανονική κατανομή.

Ξεκινάμε με την ομοιόμορφη κατανομή. Προσομοιώνουμε n ομοιόμορφες στο $(0, 1)$ τυχαίες μεταβλητές και παίρνουμε το άθροισμά τους. Επαναλαμβάνουμε αυτή τη διαδικασία 1000 φορές και, ουσιαστικά, έχουμε ένα δείγμα από την κατανομή που ακολουθεί το άθροισμα n ομοιόμορφων τυχαίων μεταβλητών. Στο Διάγραμμα 9.7 έχουμε σχεδιάσει την εμπειρική συνάρτηση κατανομής του δείγματος των 1000 τιμών μαζί με την αντίστοιχη της τυποποιημένης κανονικής κατανομής, που από το κεντρικό οριακό θεώρημα περιμένουμε πως ασυμπτωτικά ισχύει. Διαλέξαμε 4 τιμές του n, δηλαδή $1, 2, 5,$ και 10. Παρατηρείστε πόσο μικρές είναι αυτές οι τιμές.

Από το διάγραμμα παρατηρούμε πως ήδη για $n = 2$, η κατανομή αρχίζει να μοιάζει πολύ με την κανονική, ενώ για $n = 10$ δεν είναι εύκολο να ξεχωρίσουμε τις δύο κατανομές.



Διάγραμμα 9.7: Το κεντρικό οριακό θεώρημα για ομοιόμορφες τυχαίες μεταβλητές

Οι εντολές στην R, που χρησιμοποιήθηκαν για την κατασκευή του πρώτου διαγράμματος (τα υπόλοιπα δημιουργήθηκαν με απόλυτα όμοιο τρόπο), είναι οι εξής:

```
a<-1
```

```
b<-1
EX<-a/(a+b)
VX<-EX*(1-EX)/(a+b+1)

n<-1

par(mfrow=c(2,2))
  x<-rep(NA,1000)
  for (i in 1:1000) {
        x[i]<-(sum(rbeta(n,a,b)) - n*EX)/sqrt(n*VX)
  }
  cumuldata<-cumul(x)
  plot(cumuldata$points,cumuldata$empir,xlab="x",   ylab="ecdf",
   type="l",main=paste("n=",n,sep='') )
  lines(cumuldata$points,pnorm(cumuldata$points), col=2)          ■.
```

Στη μεταβλητή `n` αποθηκεύουμε το πλήθος των τυχαίων μεταβλητών που θέλουμε να αθροίσουμε. Η ομοιόμορφη στο $(0,1)$ κατανομή είναι ειδική περίπτωση της συνάρτησης κατανομής βήτα με παραμέτρους $a=b=1$. Για το λόγο αυτό χρησιμοποιήσαμε εναλλακτικά τη συνάρτηση `rbeta`, αντί για την `runif` (έτσι ώστε ο κώδικας να είναι πιο γενικός). Η μέση τιμή της είναι ίση με $1/2$ και η διακύμανση $1/12$, ενώ στη βήτα κατανομή η μέση τιμή είναι ίση με $a/(a+b)$ και η διακύμανση ίση με $ab/[(a+b)^2(a+b+1)]$, αντίστοιχα. Συνεπώς, εδώ παίρνουμε n τυχαίες μεταβλητές, τις αθροίζουμε και από το άθροισμα αφαιρούμε τη μέση τιμή και διαιρούμε με τη ρίζα της διακύμανσης που προβλέπει το ΚΟΘ. Επαναλαμβάνουμε τη διαδικασία 1000 φορές με την εντολή `for`. Οι δύο τελευταίες εντολές δημιουργούν το διάγραμμα, η πρώτη της εμπειρικής συνάρτησης κατανομής των 1000 τιμών και η δεύτερη προσθέτει στο πρώτο διάγραμμα τη θεωρητική συνάρτηση κατανομής της τυποποιημένης κανονικής κατανομής.

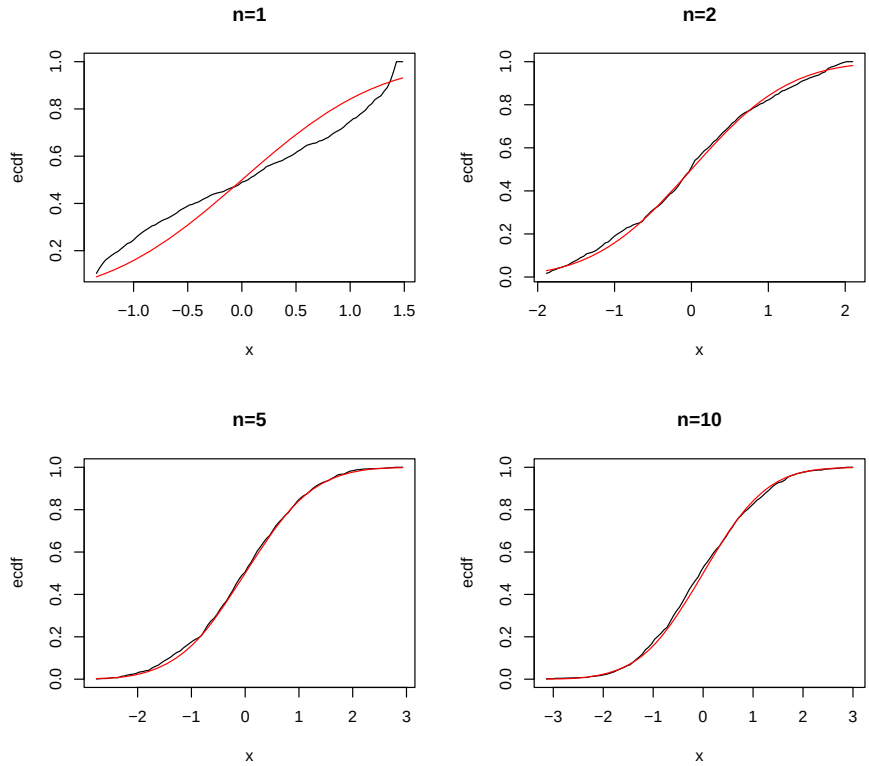
Στη συνέχεια, ας δούμε την περίπτωση τυχαίων μεταβλητών από μια κατανομή Βήτα με παραμέτρους $1/2$ και $1/2$, αντίστοιχα (βλ. Διάγραμμα 9.8). Η κατανομή αυτή έχει σχήμα σαν U με την περισσότερη πιθανότητα να υπάρχει στις άκρες κοντά στο μηδέν και το ένα. Επαναλαμβάνουμε την ίδια διαδικασία και παρατηρείστε πως πολύ γρήγορα και η βήτα κατανομή παρά τη U μορφή της, προσεγγίζει πολύ γρήγορα την κανονική κατανομή. Η σύνταξη της R είναι ίδια όπως στην ομοιόμορφη, απλά αλλάζουμε τις τιμές των παραμέτρων σε `a=1/2` και `b=1/2`.

Ουσιαστικά, αυτό που είδαμε για την Ομοιόμορφη και τη Βήτα κατανομή, μπορούμε να το δούμε για οποιαδήποτε κατανομή, όσο περίεργη μορφή και αν έχει. Φανταστείτε την εξής διακριτή κατανομή. Η κατανομή μας παίρνει την τιμή 0 με πιθανότητα 0.9 και την τιμή 20 με πιθανότητα 0.1. Επομένως, η μέση της τιμή είναι 2 και η διακύμανση της είναι 36. Για να προσομοιώσουμε από αυτή την κατανομή, αρκεί να προσομοιώσουμε μια Ομοιόμορφη τυχαία μεταβλητή, αν αυτή είναι μικρότερη από 0.9, τότε παίρνω την τιμή 0, ενώ αν είναι μεγαλύτερη την τιμή 20. Ας δούμε τι συμβαίνει με το ΚΟΘ για αυτή την κατανομή.

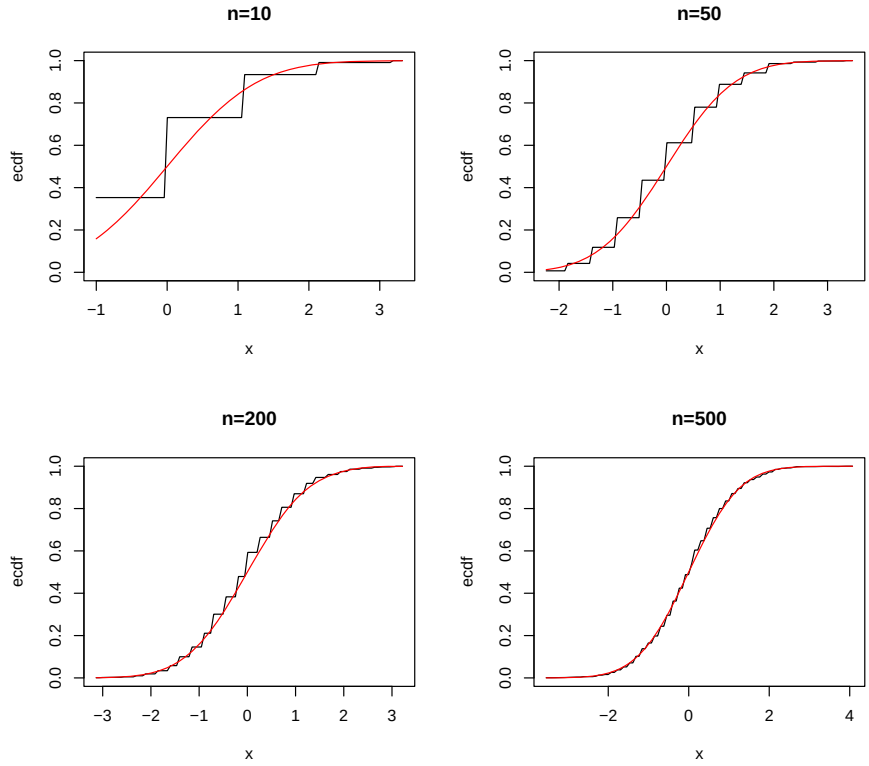
Τώρα, ο αριθμός των τυχαίων μεταβλητών στο άθροισμα είναι $10, 50, 200$ και 500, αντίστοιχα. Στο Διάγραμμα 9.9, μπορεί να δείτε ξεκάθαρα τις αποκλίσεις για $n = 10$ ή 50, Παρατηρείστε, όμως, πως τελικά η κατανομή του αθροίσματος για $n = 500$ μοιάζει πάρα πολύ με την κανονική, όπως το ΚΟΘ ορίζει, παρά το γεγονός ότι η κατανομή από όπου πήραμε τις τυχαίες μεταβλητές δεν μοιάζει καθόλου με την κανονική κατανομή.

Η σύνταξη της R που χρησιμοποιήσαμε για το Διάγραμμα 9.9 είναι παρόμοια με αυτή που χρησιμοποιήσαμε για τη συνάρτηση βήτα.

```
n<-500
x<-rep(NA,1000)
EX <- 2
VX <- 36
for (i in 1:1000) {
    u<-runif(n)
    test<-(u>0.9)*20
    x[i]<-(sum(test) - n*EX)/sqrt(n*VX)
}
cumuldata<-cumul(x)
```



Διάγραμμα 9.8: Το κεντρικό οριακό θεώρημα για τυχαίες μεταβλητές πού ακολουθούν βήτα κατανομή



Διάγραμμα 9.9: Το κεντρικό οριακό θεώρημα για δίτιμες τυχαίες μεταβλητές (με τιμές 0 και 20 και πιθανότητες εμφάνισης 0.9 και 0.1, αντίστοιχα,

```
plot(cumuldata$points,cumuldata$empir,xlab="x", ylab="ecdf",type="l
   ",main="n=500",ylim=c(0,1))
lines(cumuldata$points,pnorm(cumuldata$points), col=2)          ■.
```

Στον παραπάνω κώδικα της R, έχει ενδιαφέρον να παρατηρήσουμε τον τρόπο με τον οποίο γεννάμε τις τυχαίες μεταβλητές, από την κατανομή που μας ενδιαφέρει. Καθώς η παράσταση `(u>0.9)` παίρνει τιμή 0, όταν είναι `FALSE` και 1, αν είναι `TRUE`, τότε θα επιστρέφει την τιμή 0, αν `(u < 0.9)`. Έτσι, θα πάρουμε το ζητούμενο αποτέλεσμα, καθώς η πιθανότητα μια τυχαία μεταβλητή από την ομοιόμορφη κατανομή να είναι μικρότερη από το 0.9 είναι ακριβώς 0.9, επομένως με πιθανότητα 0.9 θα παίρνουμε τιμή 0 και με πιθανότητα 0.1 τιμή 20, όπως ακριβώς θέλουμε.

Τα παραπάνω αποτελέσματα μπορούν να γενικευθούν και με άλλες κατανομές, για να δούμε πόσο γρήγορα ισχύει το κεντρικό οριακό θεώρημα.

9.4 Διάφορα θέματα

9.4.1 Αντιμετώπιση προβλημάτων υπερχείλισης (overflow)

΄Ενα βασικό πρόβλημα στον προγραμματισμό των Η/Υ προκύπτει εξαιτίας του γεγονότος, ότι ο υπολογιστής εκτελεί τις πράξεις με της σειρά που του ορίζονται και σε καμιά περίπτωση δεν παίρνει πρωτοβουλίες για να κάνει απλοποιήσεις στις πράξεις και να αποφύγει προβλήματα υπερχείλισης (overflow). Αυτό σημαίνει, πως πολλές φορές μπορεί σε πράξεις που στο ανθρώπινο μάτι φαίνονται απλοϊκές, ο υπολογιστής να μην μπορεί αν τις εκτελέσει. Για παράδειγμα, φανταστείτε την απλή περίπτωση $\frac{200!}{199!}$. Είναι εξαιρετικά απλό να δει κανείς πως το αποτέλεσμα είναι 200. Αν όμως, δώσουμε αυτή την εντολή στον υπολογιστή χωρίς να γίνει απλοποίηση, τότε παίρνουμε το εξής μήνυμα:

```
> factorial(200)/factorial(199)
[1] NaN Warning messages: 1: value out of range in 'gammafn' 2:
value out of range in 'gammafn'                                  ■.
```

Συνεπώς, ο υπολογιστής δεν μπορεί να υπολογίσει το 200!, αφού είναι ένα τεράστιο μέγεθος για αυτόν. Θα πρέπει να τονίσουμε ότι το πρόβλημα ενδέχεται να μην εμφανιστεί όταν γράψουμε ένα πρόγραμμα, απλά γιατί οι πράξεις που θα κάνουμε θα είναι εφικτές στον υπολογιστή. Στη γενική περίπτωση του υπολογισμού του $n!/(n-1)!$ για κάθε $n > 170$, θα παίρνουμε μήνυμα λάθους. Προγραμματιστικά, για να αποφύγουμε το πρόβλημα αυτό, υπάρχουν διάφορα τρικ τα οποία μπορούμε να κάνουμε. Ας δούμε μερικούς τρόπους λοιπόν, που θα μας επιτρέψουν να αποφύγουμε τέτοιου είδους προβλήματα.

Κατά αρχάς, είναι σε όλους σαφές ότι, όταν μπορούμε να απλοποιήσουμε μια μαθηματική έκφραση, καλό είναι να το κάνουμε. ΄Ετσι, στο παράδειγμα που συζητάμε, το αποτέλεσμα είναι πάντα n και, συνεπώς, δεν υπάρχει λόγος να προσπαθούμε να το υλοποιήσουμε με παραγοντικά. ΄Εστω ο υπολογισμός της ποσότητας $\frac{200!}{100!100!}$. Για τους λόγους που είδαμε παραπάνω, δεν είναι εφικτός ο υπολογισμός χρησιμοποιώντας παραγοντικά. Αν όμως κάνει κανείς τις πράξεις, βρίσκει πως

$$\frac{101 \cdot 102 \cdot 103 \ldots 200}{1 \cdot 2 \cdot 3 \ldots 100}.$$

Και, συνεπώς, θα μπορούσε να υλοποιηθεί κατασκευάζοντας τα δύο γινόμενα ως:

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

Βέβαια, και αυτή η λύση δεν είναι η καλύτερη δυνατή, αφού σε πολλές περιπτώσεις τα γινόμενα μπορούν να γίνουν τεράστια νούμερα και να έχουμε πρόβλημα υπερχείλισης (overflow). Μια λύση είναι να δουλέψει κανείς ως εξής: η παραπάνω παράσταση γράφεται ως

$$\frac{101}{1} \cdot \frac{102}{2} \cdot \frac{103}{3} \cdots \frac{200}{100}.$$

Μπορεί να παρατηρήσει κανείς πως τώρα τα γινόμενα είναι σχετικά μικρότερα και όσο προσθέτουμε όρους, αυτοί μικραίνουν, άρα ο κίνδυνος υπερχείλισης είναι αρκετά μικρότερος.

Επομένως, μπορούμε να υλοποιήσουμε τη ζητούμενη ποσότητα απλά ως:

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

΄Ενας ακόμα τρόπος να αποφεύγουμε υπερχείλιση (overflow) όταν χρησιμοποιούμε μεγάλους αριθμούς, είναι να τους μικρύνουμε (με κάποιο τρόπο), ώστε να μπορεί να τους διαχειριστεί η R. ΄Ενας κλασσικός τέτοιος τρόπος είναι η χρήση της λογαριθμικής κλίμακας, δηλαδή να χρησιμοποιούμε του λογάριθμους των πραγματικών αριθμών. Μπορεί κανείς να δει εύκολα πως

$$\log\left(\frac{200!}{100!100!}\right) = \log(200!) - 2.\log(100!)\ .$$

Και, επιπλέον, εύκολα μπορεί κανείς να δει πως γενικά

$$\log(n!) = \log\left(\prod_{i=1}^{n} i\right) = \sum_{i=1}^{n} \log(i)\ .$$

Άρα, αρκεί να προσθέσουμε τους λογαρίθμους των αριθμών και βέβαια στο τέλος να αντιλογαριθμήσουμε, παίρνοντας την εκθετική συνάρτηση. Επομένως,

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

Θα μας δώσει τους απαιτούμενους συνδυασμούς, μειώνοντας τον κίνδυνο υπερχείλισης (overflow).

Παράδειγμα να υπολογιστεί ο γεωμετρικός μέσος $G = \left(\prod_{i=1}^{n} X_i\right)^{1/n}$.

Επειδή έχουμε γινόμενο, που μπορεί να αφορά πολλούς αριθμός, ειδικά αν το μέγεθος του δείγματος είναι πολύ μεγάλο ή οι αριθμοί είναι είτε πολύ μικροί, είτε πολύ μεγάλοι, η ιδέα είναι να μετασχηματίσουμε τους υπολογισμούς ως εξής:

$$\begin{aligned} G &= \left(\prod_{i=1}^{n} X_i\right)^{1/n} \Longrightarrow \\ \log G &= \frac{1}{n}\sum_{i=1}^{n} \log X_i \Longrightarrow \\ G &= \exp\left(\frac{1}{n}\sum_{i=1}^{n} \log X_i\right) \end{aligned}$$

Επομένως, για την υλοποίηση αρκεί αν χρησιμοποιήσουμε τις εντολές:

```
n<-length(x)
G<-exp( mean(log(x)))                                               ■.
```

Βέβαια, θα πρέπει να έχουμε εξασφαλίσει ότι δεν υπάρχουν παρατηρήσεις με αρνητική τιμή.

Παράδειγμα να υπολογίσετε τη συνάρτηση πιθανότητας της κατανομής Poisson. Η συνάρτηση πιθανότητας της κατανομής Poisson δίνεται από τον τύπο:

$$P(X = x) = \frac{e^{-\lambda}\lambda^x}{x!}, \quad x = 0, 1, \ldots, \lambda > 0 \ .$$

Το βασικό πρόβλημα με την υλοποίηση της παραπάνω συνάρτησης είναι κυρίως οι όροι λ^x και $x!$. Η ποσότητα $e^{-\lambda}$ υπολογίζεται εύκολα για $\lambda < 200$, συνεπώς για τιμές του λ που η κατανομή Poisson θα μπορούσε να ενδιαφέρει. Θυμηθείτε πως για μεγάλο λ, η κατανομή Poisson μπορεί να προσεγγιστεί μέσω της κανονικής κατανομής. Μπορεί λοιπόν να παρατηρήσει κανείς ότι οι όροι λ^x και $x!$ είναι ο καθένας γινόμενο από ακριβώς x όρους και, συνεπώς, μπορούμε να το γράψουμε ως

$$\frac{\lambda^x}{x!} = \frac{\lambda \cdot \lambda \cdot \lambda \ldots \lambda}{1 \cdot 2 \cdot 3 \ldots x} = \frac{\lambda}{1} \cdot \frac{\lambda}{2} \cdots \frac{\lambda}{x},$$

και, επομένως, μπορεί να υλοποιηθεί υπολογίζοντας το γινόμενο

$$\prod_{i=1}^{x} \frac{\lambda}{i}.$$

Οι εντολές που ακολουθούν, δείχνουν πως αυτή η υλοποίηση δίνει το σωστό αποτέλεσμα, ενώ ο απευθείας υπολογισμός αποτυγχάνει εξαιτίας υπερχείλισης.

```
> lambda<-150
> x<-200
> exp(-lambda)*lambda^x/factorial(x)
[1] NaN Warning message: value out of range in 'gammafn'
>
> x1<-rep(lambda,x)
> x2<-1:x
> exp(-lambda)*prod(x1/x2)
[1] 1.503803e-05
> dpois(x,lambda)
[1] 1.503803e-05
```

■.

9.4.2 Γραφικά με κίνηση

Μπορούμε να χρησιμοποιήσουμε την R για να κατασκευάσουμε διαγράμματα με κίνηση. Στην πραγματικότητα, κατασκευάζουμε μια σειρά από διαγράμματα που δίνουν στο ανθρώπινο μάτι την ψευδαίσθηση ότι κινούνται, χρησιμοποιώντας την απλή αρχή του κινηματογράφου, δηλαδή, απλά τοποθετούμε διαδοχικά εικόνες που διαφέρουν λίγο και δίνουν την αίσθηση της κίνησης. Αυτό είναι αρκετά χρήσιμο για να κατανοήσει κανείς κάποιες βασικές ιδέες στην Στατιστική.

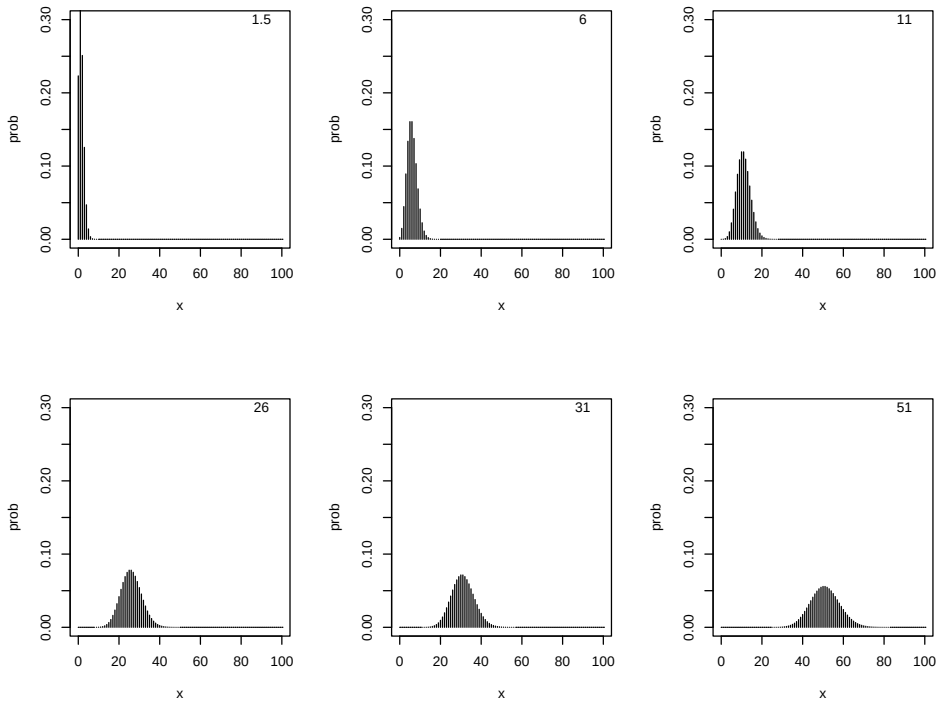
Για παράδειγμα, οι εντολές που ακολουθούν, κατασκευάζουν διάγραμμα της συνάρτησης πιθανότητας της κατανομής Poisson για τιμές της παραμέτρου λ, δηλαδή της μέσης τιμής από 1

εως και 70. Για κάθε διάγραμμα χρησιμοποιούμε ακριβώς τις ίδιες κλίμακες, ώστε να δίνεται η αίσθηση της κίνησης και στην πραγματικότητα ζωγραφίζουμε κάθε ένα διάγραμμα (συνολικά 691 διαγράμματα, όσες και οι διαφορετικές τιμές του y) πάνω στο προηγούμενο. Αυτό δίνει την αίσθηση της κίνησης. Αυτό που είναι ενδιαφέρον στο διάγραμμα, είναι πως πολύ γρήγορα αποκτάμε την εικόνα της κανονικής κατανομής, δηλαδή το διάγραμμα μοιάζει με μια καμπάνα. Ξέρουμε από τη θεωρία πως για μεγάλες τιμές του λ, η κατανομή Poisson προσεγγίζεται πολύ καλά από την κανονική κατανομή. Ο κώδικας μάς επιτρέπει να δούμε πως αυτό επιτυγχάνεται σιγά σιγά με διαδοχικές τιμές του λ. Η τιμή του λ εμφανίζεται πάνω δεξιά στο διάγραμμα.

```
y<-seq(1,70,by=0.1)
for (i in 1:length(y) ) {
lambda<-y[i]
plot(c(0,100),c(0,0.3),type="n",xlab="x",ylab="prob")
    for ( j in 0:100) {
    segments(j,0,j,dpois(j,lambda))
    }
text(90,0.3,lambda)
}
```

■.

Στο Διάγραμμα 9.10 μπορεί να δει κανείς στατικά έξι διαγράμματα από τα 691 που φτιάξαμε διαδοχικά. Δοκιμάστε τον παραπάνω κώδικα για να κατανοήσετε την κίνηση.



Διάγραμμα 9.10: Η συνάρτηση πιθανότητας της κατανομής Poisson για διάφορες τιμές του λ.

9.5 Επιπλέον βιβλιογραφικό υλικό

Όπως είδαμε το κεφάλαιο αυτό ήταν αφιερωμένο σε πιο προχωρημένα παραδείγματα χρήσης της R στη στατιστική και κυρίως σε προβλήματα υπολογιστικής Στατιστικής. Επιπλέον υλικό για εξειδικευμένες εφαρμογές σχετικές με υπολογιστική στατιστική μπορείτε να βρείτε, ενδεικτικά, στα βιβλία των Rizzo (2007) και Robert & Casella (2009). Θέματα βελτιστοποίησης περιγράφονται στο βιβλίο του Cortez (2014) Μπευζιανές υπολογιστικές τεχνικές περιγράφονται στο βιβλίο του Albert (2009) – κυρίως με τη χρήση του πακέτου `MCMCpack` – και στο βιβλίο των Marin & Robert (2013). Τέλος ένας οδηγός για τα πιο δημοφιλή πακέτα της R δίνεται από το Wickham (2015).

Βιβλιογραφικές Αναφορές Κεφαλαίου 9

Albert, J. (2009). *Bayesian Computation with R*. Use R! Springer New York.

Cortez, P. (2014). *Modern Optimization with R*. Use R! Springer International Publishing.

Marin, J. & Robert, C. (2013). *Bayesian Essentials with R*. Springer Texts in Statistics. Springer New York.

Rizzo, M. (2007). *Statistical Computing with R*. Chapman & Hall/CRC The R Series. Taylor & Francis.

Robert, C. & Casella, G. (2009). *Introducing Monte Carlo Methods with R*. Use R! Springer.

Wickham, H. (2015). *R Packages*. O'Reilly Media.

Ασκήσεις

Για την επίλυση ορισμένων από τις ασκήσεις που ακολουθούν, είναι απαραίτητη η χρήση αρχείων που είναι διαθέσιμα από τη ιστο-σελίδα τηλε-εκπαίδευσης (`eclass.aueb.gr`) του μαθήματος. Οι ασκήσεις είναι ομαδοποιημένες ανά θεματική ενότητα. Η ομαδοποίηση αυτή έγινε για είναι πιο συνεκτική η παρουσίασή τους.

Απλές Πράξεις

1. Έχουμε δημιουργήσει τα διανύσματα `x<-c(4,-2,6,8)` και `y<-c(1,1,3,-1)`. Σημειώστε το αποτέλεσμα κάθε μίας από τις παρακάτω πράξεις:

   - α) `x/2` β) `x*y`
     γ) `2*x+y` δ) `3*y+x/2`
   - Για κάθε μία από τις ακόλουθες εκφράσεις αναφέρετε τι είδους δομή δεδομένων παράγει (πίνακα, διάνυσμα, λίστα, data frame, κλπ) και σημειώστε τις τιμές της:
     α) `z<-c(y,x,y)` β) `k<-cbind(x,y)`

2. Έχουμε δημιουργήσει τα διανύσματα `x<-c(-4,1,0,0,5,-3,2)` και `y<-c(1,1,-5,4,3,-2` και εξηγήστε τα διανύσματα που προκύπτουν ως αποτέλεσμα των ακόλουθων εκφράσεων:

   $i)$ $\texttt{x[c(1,2,6)]}$ $ii)$ $\texttt{y[1:4]}$
   $iii)$ $\texttt{x[x<0]}$ $iv)$ $\texttt{x[}-3\texttt{]}$
   $v)$ $\texttt{x[x<0\&y!=1]}$ $vi)$ $\texttt{y[y<0|x<=0]}$
   $vii)$ $\texttt{y[rep(1:2,2)]}$

3. Για τα διανύσματα που ορίζονται με τις εντολές `x<-10:-4` και `y<-rep(c(12,3,-2,4,0),3)` σημειώστε τα αποτελέσματα των ακόλουθων πράξεων:

   1) $\texttt{x+y}$ 2) $\texttt{x}-\texttt{y/2}$ 3) $\texttt{sum(y[y<x])}$
   4) $\texttt{sum(y<x)}$ 5) $\texttt{1+y}-\texttt{x}$

Επιπλέον εξηγήστε ποιες πράξεις εκτελούνται σε κάθε περίπτωση, έτσι ώστε να προκύψουν τα αποτελέσματα που σημειώσατε παραπάνω (π.χ. στο (α) ερώτημα αθροίζουμε τα αντίστοιχα στοιχεία των διανυσμάτων `x` και `y`).

Επαναλάβετε την ίδια διαδικασία για `x<-10:1`.

4. Έστω ένα διάνυσμα `x` που περιέχει ακέραιους αριθμούς. Περιγράψτε τις εντολές του R που χρειάζονται για να ολοκληρώσετε επιτυχώς τις ακόλουθες ενέργειες:

   - Υπολογίστε το άθροισμα των αρνητικών ακέραιων που περιέχει το διάνυσμα.
   - Υπολογίστε το άθροισμα των στοιχείων μέχρι να βρουν για πρώτη φορά αρνητική τιμή.
   - Δημιουργείστε ένα καινούριο διάνυσμα όπου το i στοιχείο περιέχει το άθροισμα των μη αρνητικών αριθμών του `x`) μέχρι τη θέση i.
   - Δημιουργείστε ένα καινούριο διάνυσμα `x2`, το οποίο έχει ως στοιχεία τον αριθμό των στοιχειών ενός διανύσματος `x`, μέχρι να παρατηρηθεί εναλλαγή του πρόσημου μεταξύ 2 διαδοχικών στοιχείων `x[i]` και `x[i+1]`. Για παράδειγμα, αν το διάνυσμα είναι το `x<-c(1,2,-3,-4,2,3,-5,1,-1)`, το αποτέλεσμα θα είναι `x2<-c(2,2,2,1,1)`.

5. Υπολογίστε τις παρακάτω μαθηματικές εκφράσεις για διάφορες τιμές των μεταβλητών:

   (a) $e^{-t^{1/\theta}}$

   (b) $(1+t)^{-1/\theta}$

   (c) $1-(1-e^{-t})^{1/\theta}$

   (d) $-\frac{\log\left(1-(1-e^{-\theta})e^{-t}\right)}{\theta}$

   (e) $e^{-(x^\theta+y^\theta)^{1/\theta}}$

   (f) $(x^{-\theta}+y^{-\theta}-1)^{-1/\theta}$

   (g) $1-(x^\theta+y^\theta-x^\theta y^\theta)^{1/\theta}$

   (h) $-\frac{1}{\theta}\log\left\{1+\frac{(e^{-\theta x}-1)(e^{-\theta y}-1)}{e^{-\theta}-1}\right\}$

   (i) $xye^{(x^{-\theta}+y^{-\theta})^{-1/\theta}}$

   (j) $\theta\cos(x)^{\theta+1}\sin\left\{(\theta+1)(y+\theta\frac{\pi}{2})\right\}$

6. Ένας φοιτητής περνάει στο επόμενο εξάμηνο εάν:

   (a) Έχει σε όλα τα μαθήματα πάνω από την βάση (50).

   (b) Εάν έχει σε ένα κάτω από τη βάση, να σημειώνει μέσο όρο βαθμολογίας πάνω από 60.

   (c) Εάν έχει δυο κάτω από την βάση, αλλά πάνω από 40, να σημειώνει μέσο όρο βαθμολογίας πάνω από 70.

Να γράψετε μια έκφραση που να ελέγχει, αν η βαθμολογία ενός φοιτητή σε 4 μαθήματα τον προάγει στο επόμενο εξάμηνο.

7. Έστω ότι το στο `x` έχετε αποθηκεύσει μια τιμή. Να γράψετε εντολές που να υπολογίζουν μια νέα τιμή `y` ως εξής:

   (a) $y = \begin{cases} 12 & x > 0 \\ 6 & x \leq 0 \end{cases}$

   (b) $y = \begin{cases} 3 & x = 5 \\ 4 & x > 5 \\ 6 & x < 5 \end{cases}$

   (c) $y = \begin{cases} 5 & 0 < x < 1 \\ -12 & x \leq 0, x \geq 1 \end{cases}$

8. Έστω ότι στα `x` και `y` αντίστοιχα, είναι αποθηκευμένοι οι βαθμοί στις εργασίες και τη γραπτή εξέταση ενός μαθήματος για έναν συγκεκριμένο φοιτητή. Να υπολογίσετε την τελική βαθμολογία του φοιτητή αν

   (a) ο τελικός βαθμός είναι 0.5 εργασία + 0.5 γραπτή εξέταση

   (b) για να περάσει κάποιος το μάθημα πρέπει να έχει τουλάχιστον 7 στις εργασίες και τότε ο τελικός βαθμός είναι ο βαθμός της εξέτασης, προσαυξημένος με μια μονάδα, αν οι εργασίες ήταν πάνω από το 8, ενώ ο τελικός βαθμός είναι 4 αν ο φοιτητής δεν συμπλήρωσε τις 7 μονάδες από την εργασία.

   (c) Ο τελικός βαθμός είναι ο μικρότερος από τους βαθμούς της εργασίας και του γραπτού

   (d) ο τελικός βαθμός είναι:

   - αν οι εργασίες είχαν πάνω από 5, ο βαθμός του γραπτού και της εργασίας με ίσο βάρος, ενώ
   - αν οι εργασίες ήταν κάτω από 5 μόνο ο βαθμός του γραπτού.

9. Να γράψετε στην R τις ακόλουθες εκφράσεις:

   (a) $\frac{1}{2(3+x)^2} - \frac{2}{1+\frac{1}{4x}-5}x$

   (b) $\log(x + 5^{2(x+1)}) + 18x\exp(-x)$

   (c) $\cos\left(x + 4x\frac{1}{x+7}\right) + 32$

   (d) $\log\left(\frac{x}{1-x}\right)$

   (e) $\frac{\exp(-x)x^x}{x!}$

10. Δημιουργήστε με τα δεδομένα του πίνακα 9.1 κατάλληλα διανύσματα. Στη συνέχεια για αυτά τα διανύσματα:

(a) Υπολογίστε τη μέση τιμή για κάθε μάθημα ως προς το φύλο.

(b) Βρείτε τη μέγιστη βαθμολογία για κάθε μάθημα ξεχωριστά.

(c) Βρείτε τη μέγιστη βαθμολογία που πήρε φοιτητής σε οποιοδήποτε μάθημα.

(d) Βρείτε το μέσο όρο της βαθμολογίας των φοιτητών και την κατάταξή τους, με βάση αυτό το μέσο όρο στο σύνολο των φοιτητών, αλλά και στο σύνολο των φοιτητών από το ίδιο έτος.

(e) Ποιος είναι ο καλύτερος φοιτητής; Τυποποιήστε τους βαθμούς, ώστε να είναι συγκρίσιμοι και υπολογίστε το μέσο όρο με τη χρήση των τυποποιημένων βαθμών.

(f) Ποιό ποσοστό φοιτητών πέρασε όλα τα μαθήματα;

(g) Ποιος είναι ο μέσος βαθμός και η διακύμανση για αυτούς που πέρασαν κάθε μάθημα;

(h) Κατασκευάστε από τα δεδομένα έναν πίνακα. Πώς θα υπολογίστε τώρα τη μέση τιμή κάθε μαθήματος ως προς το φύλο, αλλά και τη μέση τιμή για κάθε μάθημα ξεχωριστά;

(i) Για κάθε μάθημα υπολογίστε το λόγο της τυπικής απόκλισης με τη μέση τιμή. Τι μας δείχνει αυτός ο λόγος;

11. Έστω δύο διανύσματα `x` και `y`, τα οποία μπορεί και να περιέχουν Missing values (NA). Πώς μπορώ να συγκρίνω αν αυτά τα δύο vectors είναι ακριβώς ίδια; (Παρατήρηση: Το πρόβλημα που πρέπει να ξεπεράσω είναι πως αν έχω `NA` τιμές η όποια σύγκριση θα επιστρέψει αποτέλεσμα `NA` και όχι `TRUE` ή `FALSE`. Η εντολή `identical` μπορεί να λύσει το πρόβλημα αλλά υποθέστε πως δεν είναι διαθέσιμη).

12. Έστω πως στο διάνυσμα `x` έχουμε αποθηκεύσει τις τιμές στο μάθημα των Μαθηματικών και στο διάνυσμα `y` τις τιμές στο μάθημα της Φυσικής. Ουσιαστικά, στην i θέση καθενός από τα δύο διανύσματα έχουμε τη βαθμολογία του i μαθητή. Έστω ότι θέλουμε να ταξινομήσουμε τους φοιτητές σε φθίνουσα σειρά με βάση τη βαθμολογία στα μαθηματικά.

(a) Περιγράψτε εντολές που να το κάνουν και δοκιμάστε τα αποτελέσματά σας δίνοντας υποθετικές τιμές στα `x` και `y`.

(b) Ας προσπαθήσουμε τώρα να γενικεύσουμε το πρόβλημα. Τα δεδομένα μας είναι ένας πίνακας $n \times p$, δηλαδή έχουμε n παρατηρήσεις και p μεταβλητές και θέλουμε να ταξινομήσουμε τα δεδομένα μας ως προς μια στήλη του πίνακα. Γράψτε ένα πρόγραμμα που να λύνει αυτό το γενικό πρόβλημα

13. Έστω τα διανύσματα

```
x<- c(3,2,8,6,-1,-2,-5) y<- c(-4,3,5,3,2,7,6)
```

Ποια θα είναι τα αποτελέσματα των επόμενων πράξεων;

1. `x[y>0]`

Χημεία	Φυσική	Μαθηματικά	Αρχαία	Φύλο	Έτος
93	42	98	34	Α	1
71	67	68	33	Α	1
77	59	36	24	Α	1
78	70	92	24	Α	1
77	59	44	31	Α	1
81	50	45	22	Α	2
88	50	58	23	Γ	2
74	51	31	32	Γ	2
67	45	70	31	Γ	2
78	64	46	26	Γ	2
77	49	41	75	Α	1
67	49	46	81	Α	1
63	48	65	87	Γ	1
83	51	62	100	Γ	1
73	56	20	81	Γ	1
70	47	22	100	Γ	2
78	53	92	77	Α	2
95	56	56	89	Α	2
88	49	28	100	Α	2
75	71	94	77	Α	2

Πίνακας 9.1: Δεδομένα Άσκησης 10

2. `rbind(x,y)`
3. `x[ -(y>0) ]`
4. `x^2 + y[1]/10 - 2`
5. `matrix(c(x,y),2,8,byrow=T)`
6. `dim(cbind(x))[1]*length(y)`
7. `sum(x<y)`
8. ```
   for (i in 1:2){
   x <- x^2}
   print(x)
   ```
9. ```
   for (i in 1:4){
   i<-x[i] + i}
   print(i)
   ```
10. ```
    t<-(x>0)*(y<0)
    x[-t]
    ```

(Παρατήρηση: κάθε πράξη είναι ανεξάρτητη από τις προηγούμενες, δηλαδή δεν χρησιμοποιεί τα αποτελέσματα από τυχόν προηγούμενες πράξεις/ εντολές)

14. Να δώσετε τις εντολές (με σύντομη περιγραφή) για την κατασκευή μιας λίστας που θα περιέχει:

    (a) ένα διάνυσμα με τους φυσικούς αριθμούς από το 1 έως το 1000.

    (b) τον πίνακα $A = \begin{pmatrix} 1 & 2 & 6 \\ 5 & 2 & 5 \\ 6 & 1 & 3 \end{pmatrix}$

    (c) το ακόλουθο data.frame

aa	age	Weight
1	21	70
2	23	65
3	27	66
4	20	78

15. Δώστε τις εντολές εισαγωγής ενός τρισδιάστατου πίνακα (διάστασης 3×2×2) στην παραπάνω λίστα με στοιχεία $B[,,1] = \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{pmatrix}$ και $B[,,2] = \begin{pmatrix} 1 & 10 \\ 1 & 20 \\ 1 & 30 \end{pmatrix}$.
    ```

16. ΄Εστω ότι έχετε ένα dataframe με το όνομα football με δεδομένα από το Ελληνικό ποδοσφαιρικό πρωτάθλημα της προηγούμενης χρονιάς. Το dataframe έχει τις ακόλουθες μεταβλητές

    goals1: Γκολ που πέτυχε η ομάδα που αγωνιζόταν στο γήπεδό της.

    goals2: Γκολ που πέτυχε η ομάδα που αγωνιζόταν εκτός έδρας (φιλοξενούμενη ομάδα).

    team1: Εντός έδρας ομάδα.

    team2: Εκτός έδρας ομάδα (φιλοξενούμενη ομάδα).

    Οι μεταβλητές team1 και team2 είναι κατηγορικές με 18 επίπεδα (κατηγορίες), ένα για κάθε ομάδα. Τα επίπεδα των δύο αυτών μεταβλητών είναι τα ίδια, αφού κάθε ομάδα παίζει με όλες τις ομάδες εντός και εκτός έδρας. Οι μεταβλητές goals1 και goals2 είναι διακριτές.

    (a) Δώστε τις εντολές για τον υπολογισμό κάποιων συνολικών περιγραφικών δεικτών των διαθέσιμων μεταβλητών.

    (b) Να δώσετε τις εντολές που χρειάζονται για να υπολογίσουμε το συνολικό αριθμό γκολ που δέχεται και πετυχαίνει κάθε ομάδα, τόσο εντός, όσο και εκτός έδρας καθώς επίσης και τη συνολική τους επίδοση. Κάντε το ίδιο και με τη διαφορά των γκολ. Βάλτε τα αποτελέσματα καθώς επίσης και το μέσο αριθμό γκολ σε ένα πίνακα, που θα είναι εύκολος να διαβαστεί.

    (c) Για να ελέγξουμε την ύπαρξη ή όχι σημαντικής επίδρασης της έδρας, κάνουμε τον ακόλουθο έλεγχο υπόθεσης.

```
> t.test(ex4.ita91$g1, ex4.ita91$g2, mu = 0,
paired = TRUE, conf.level = 0.95)

        Paired t-test

data: ex4.ita91$g1 and ex4.ita91$g2 t = 4.7374, df = 305,
p-value = 3.323e-06 alternative hypothesis: true difference
    in
means
  is not equal to 0
95 percent confidence interval: 0.2388191 0.5781743 sample
estimates: mean of the differences
    0.4084967
```

    Να περιγράψετε τι κάνει η εντολή και να ερμηνεύσετε τα αποτελέσματα. ■.

Διαγράμματα

17. Δημιουργήστε στο ίδιο παράθυρο μια σειρά από διαγράμματα της συνάρτησης (μάζας) πυκνότητας πιθανότητας της κανονικής κατανομής με μέση τιμή 0, αλλά διακύμανση διαφορετική

κάθε φορά και συγκεκριμένα τιμές $\sigma^2 = 0.5, 1, 1.5, 2$. Τι παρατηρείτε; Στη συνέχεια κρατήστε τη διακύμανση σταθερή και ίση με 1 αλλά αφήστε τη μέση τιμή να αλλάζει και να παίρνει τιμές $\mu = -0.5, 1, 0.5, 1$. Αυτά τα διαγράμματα θα σας βοηθήσουν να κατανοήσετε τη σημασία των παραμέτρων της κατανομής. Επαναλάβετε το ίδιο με άλλες κατανομές, αφήνοντας τις παραμέτρους να αλλάζουν για να κατανοήσετε τη σημασία τους.

18. Ας υποθέσουμε ότι δεν είναι διαθέσιμη η συνάρτηση `boxplot` για να κατασκευάσουμε το Boxplot. Χρησιμοποιήστε εντολές για διαγράμματα της R και κατάλληλες συναρτήσεις για να βρείτε τις ποσότητες που χρειάζεστε για να κατασκευάσετε το Boxplot και φτιάξτε το δικό σας Boxplot.

19. Η εξίσωση της έλλειψης δίνεται από τον τύπο:

$$\frac{(x-x_0)^2}{\alpha} + \frac{(y-y_0)}{\beta} = 1 \, .$$

Κατασκευάστε μια συνάρτηση που να παίρνει σαν input τις τιμές των παραμέτρων α, β, x_0, y_0 και να κατασκευάζει την έλλειψη αυτή. Θα πρέπει να τονιστεί ότι τέτοιες ελλείψεις έχουν μεγάλη χρησιμότητα στη στατιστική και ειδικότερα στην Πολυμεταβλητή Στατιστική Ανάλυση.

20. **Καμπύλες του Andrews:** Οι καμπύλες του Andrews (Andrews' curves) εισήχθηκαν από τον Andrews με σκοπό να απεικονιστούν πολυμεταβλητά δεδομένα. Ας υποθέσουμε πως έχουμε n παρατηρήσεις και για κάθε παρατήρηση έχουμε ένα διάνυσμα από k τυχαίες μεταβλητές, έστω $X_1, X_2, \ldots, X_k$. Για κάθε παρατήρηση σχηματίζουμε την καμπύλη της συνάρτησης

$$f_X(t) = \frac{X_1}{\sqrt{2}} + X_2 \sin t + X_3 \cos t + X_4 \sin(2t) + X_5 cos(2t) + \ldots$$

για τιμές του $t \in (-\pi, \pi)$. Έτσι για κάθε τιμή έχουμε μια καμπύλη που βασίζεται σε ημίτονα και συνημίτονα και συνεπώς διαφορές στις παρατηρήσεις θα φανούν εύκολα, εξαιτίας της διαφορετικής μορφής της καμπύλης. Προσοχή όμως, καθώς η X_1 είναι η μεταβλητή με τη μεγαλύτερη διακύμανση, η X_2 με τη δεύτερη μεγαλύτερη και λοιπά. Δηλαδή οι μεταβλητές είναι σε φθίνουσα τάξη διακύμανσης. Υποθέστε πως έχετε 6 μεταβλητές.

(a) Γράψτε ένα πρόγραμμα σε R που να δέχεται ως εισερχόμενο έναν πίνακα δεδομένων με 20 γραμμές (παρατηρήσεις) και 6 στήλες (μεταβλητές) και να κατασκευάζει τις καμπύλες Andrews για τις 20 παρατηρήσεις σε ένα διάγραμμα. Προσοχή, οι μεταβλητές δεν είναι απαραίτητα σε φθίνουσα τάξη διακύμανσης.

(b) Προσπαθήστε να γενικεύσετε τον κώδικά σας, ώστε να μπορεί να δουλέψει για κάθε αριθμό μεταβλητών

21. • Να απεικονίσετε γραφικά σε ένα κοινό διάγραμμα τις συναρτήσεις $f(x) = -x^3 + 4x^2 + 200x + 3$ και $g(x) = 10x^2 - 10x - 1000$. Βρείτε προσεγγιστικά ποιες τιμές

λύνουν τις εξισώσεις $f(x) = 0$, $g(x) = 0$, $g(x) = f(x)$ και απεικονίστε τα και στο διάγραμμα.

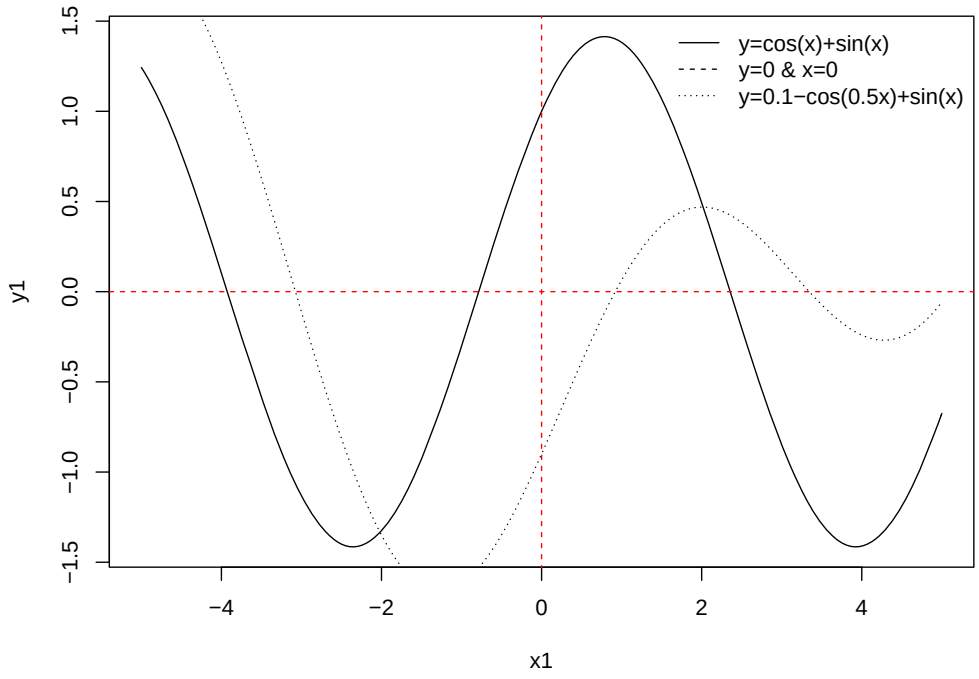
- Να απεικονίσετε γραφικά σε ένα κοινό διάγραμμα τις συναρτήσεις $f(x) = |sin(x)|$ για $-2\pi < x < 2\pi$ και $g(x) = cos(x)$, όταν $0 < x < \pi$ και $-2\pi < x < -\pi$ και 0 για όλες τις υπόλοιπες τιμές.
- Να απεικονίσετε διαγραμματικά τη διμεταβλητή συνάρτηση $f(x, y) = 2x^2 - 2y^2 + 3x + y + 4$
- Να απεικονίσετε γραφικά σε ένα κοινό διάγραμμα τις συναρτήσεις αθροιστικής πιθανότητας των κατανομών Poisson(1.5) και Poisson(2.0).
- Να απεικονίσετε γραφικά σε ένα κοινό διάγραμμα τις συναρτήσεις πιθανότητας των διωνυμικών κατανομών με $n = 10$, $p = 0.5$ και $n = 10$, $p = 0.7$.
- Να απεικονίσετε γραφικά σε ένα κοινό διάγραμμα τις συναρτήσεις (μάζας) πυκνότητας πιθανότητας των κατανομών $N(0, \sigma^2 = 1)$ και $N(0.5, \sigma^2 = 0.25)$.

22. Να κατασκευάσετε το ακόλουθο διάγραμμα των συναρτήσεων:

$y = \cos(x) + \sin(x)$

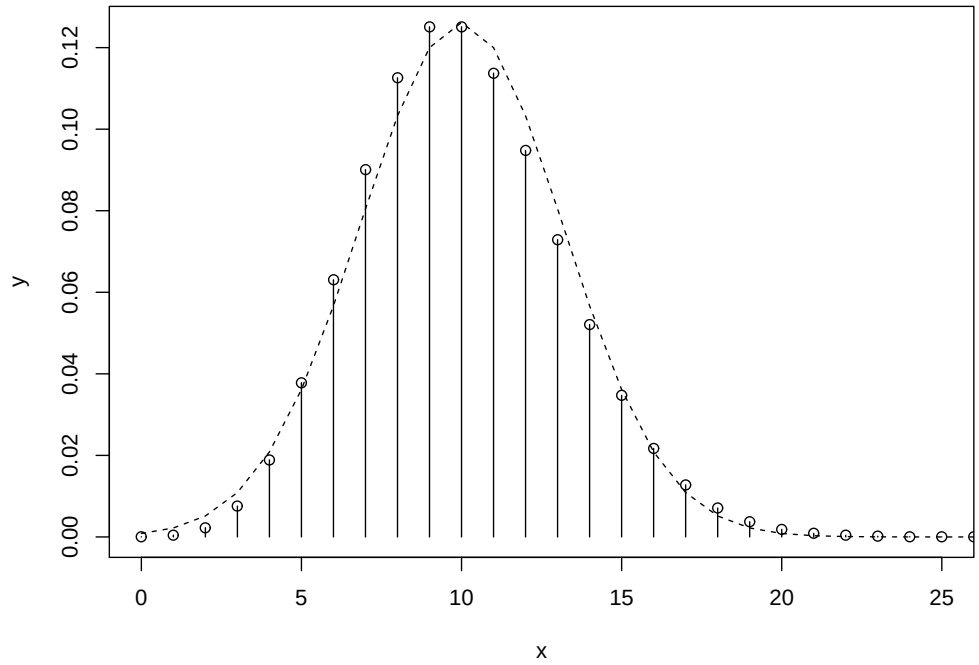
$y = 0.1 - \cos(x/2) + \sin(x)$

και να προσθέσετε το σύστημα των κάθετων αξόνων κόκκινο χρώμα, όπως φαίνεται στο Διάγραμμα 9.11.



Διάγραμμα 9.11: Απεικόνιση των συναρτήσεων της Άσκησης 22

23. Να φτιάξετε ένα διάγραμμα που θα συγκρίνει τη συνάρτηση πιθανότητας της κατανομής Poisson, με μέσο 10 και της αντίστοιχης προσέγγισης της μέσω της κανονικής κατανομής. Το διάγραμμα θα πρέπει να μοιάζει με το ακόλουθο:



Διάγραμμα 9.12: Απεικόνιση της συνάρτησης πιθανότητας της Poisson(10) για την Άσκηση 23

Περιγραφική Στατιστική

24. Ένα περιγραφικό μέτρο που σκοπό έχει να αποφύγει την επίδραση ακραίων τιμών, είναι η χρήση του περικομμένου μέσου, ο οποίος ορίζεται ως

$$y_{tr} = \frac{\sum_{i=1}^{k} y_i^*}{k},$$

όπου y_i^* είναι οι παρατηρήσεις y_i που δεν αφορούν όλο το δείγμα, αλλά μόνο τις τιμές που μένουν, αφού αφαιρέσουμε το πάνω 5% και το κάτω 5% των παρατηρήσεων και k είναι το αντίστοιχο πλήθος των y_i^*.

Γράψτε τις εντολές που χρειάζονται για το υπολογισμό του παραπάνω περικκομμένου μέσου.

Μετά γράψτε μια συνάρτηση που να επιστρέφει τον περικομμένο μέσο αφού αφαιρεθεί το 100q% των συνολικών παρατηρήσεων (50q% από κάθε πλευρά).

25. Ένα άλλο περιγραφικό μέτρο, που σκοπό έχει να αποφύγει την επίδραση ακραίων τιμών,

είναι η χρήση του Windsorized μέσου, ο οποίος ορίζεται ως

$$\bar{y}_w = \frac{\sum_{i=1}^{n} y_i^*}{n} \quad ,$$

όπου

$$y_i^* = \begin{cases} y_i & \text{if } \left|\frac{y_i - \bar{y}}{s}\right| \leq 2 \\ \bar{y} - 2s & \text{if } y_i < \bar{y} - 2s \\ \bar{y} + 2s & \text{if } y_i > \bar{y} + 2s \end{cases}$$

Να γράψετε μια συνάρτηση που να υπολογίζει το Windsorized μέσο για ένα διάνυσμα παρατηρήσεων

26. Ο συντελεστής Gini ορίζεται ως

$$G = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} |x_i - x_j|}{2n^2 \bar{x}} \ .$$

- Γράψτε μια συνάρτηση που να υπολογίζει για ένα διάνυσμα x το συντελεστή Gini.
- Γράψτε μια συνάρτηση που να επιστρέφει ως αποτελέσματα, τόσο το Windsorized μέσο της προηγούμενης άσκησης, όσο και το συντελεστή Gini.

27. ΄Εστω το εξής μέτρο μεταβλητότητας

$$T = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} (x_i - x_j)^2}{2n^2} \ .$$

- Γράψτε μια συνάρτηση που να υπολογίζει για ένα διάνυσμα x το μέτρο αυτό.
- Γράψτε μια συνάρτηση που να υπολογίζει το μέτρο αυτό και τη διακύμανση. Τι παρατηρείτε; Πώς σχετίζεται το με τη διακύμανση;

28. Γράψτε μια συνάρτηση η οποία θα παίρνει ως Input ένα διάνυσμα θετικών παρατηρήσεων και θα επιστρέφει την ποσότητα

$$d = \sum_{i=1}^{n} c_i X_{(i)},$$

όπου $c_i = \left(\frac{n-i}{n}\right)^{1/2} - \left(\frac{n-i+1}{n}\right)^{1/2} - \frac{1}{n}$, n είναι το πλήθος των παρατηρήσεων και $X_{(i)}$είναι η i διατεταγμένη παρατήρηση, δηλαδή $X_{(1)}$ είναι η μικρότερη και $X_{(n)}$ η μεγαλύτερη παρατήρηση, αντίστοιχα.

Προσοχή: μη θετικές παρατηρήσεις θα πρέπει να τις αγνοήσετε για τους υπολογισμούς, δηλαδή το πλήθος των παρατηρήσεων θα μειωθεί.

29. Ο συντελεστής συσχέτισης του Pearson δίνεται ως

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2 \sum_{i=1}^{n}(X_i - \bar{X})^2}}.$$

Να γράψετε μια συνάρτηση που να παίρνει σαν παράμετρο εισόδου έναν πίνακα με δύο στήλες (κάθε στήλη είναι μια μεταβλητή) και να επιστρέφει το συντελεστή συσχέτισης του Pearson, τις μέσες τιμές και τις διακυμάνσεις των μεταβλητών.

(Προσοχή δεν μπορείτε να χρησιμοποιήσετε τις εντολές mean, var, cov, cor)

Συναρτήσεις

30. Έστω μια ακολουθία από ακέραιους αριθμούς. Γράψτε μια συνάρτηση που να παίρνει σαν παράμετρο εισόδου μια τέτοια ακολουθία και να μετράει πόσες φορές ο επόμενος αριθμός είναι μικρότερος από τον προηγούμενο, δηλαδή πόσες φορές ο αριθμός στη θέση $\nu + 1$ είναι μικρότερος από τον αριθμό στη θέση ν.

Παράδειγμα: για την ακολουθία (2, 5, 3, 4, 2, 7, 7, 1, 9), έχουμε πως το 3 είναι μικρότερο του 5, το 2 από το 4, το 1 από το 7 κι επομένως η απάντηση είναι 3.

31. Χωρίς να χρησιμοποιήσετε την εντολή table, να κατασκευάσετε μια συνάρτηση η οποία να υπολογίζει για ένα δοθέν διάνυσμα από μονοψήφιος αριθμούς 0,1,2,...,9 , πόσες φορές εμφανίζεται ο καθένας από αυτούς. Το αποτέλεσμα θα πρέπει να είναι ένα διάνυσμα. Δώστε 3 διανύσματα με δεδομένα και γράψτε ποιο ακριβώς θα είναι το αποτέλεσμα.

32. Γράψτε μία συνάρτηση η οποία παίρνει τρεις παραμέτρους εισόδου: έναν πίνακα (έστω `x`), έναν αριθμό (έστω `n`) και μία λογική παράμετρο (έστω `col`), δηλαδή μία παράμετρο που παίρνει τιμές `TRUE` ή `FALSE`. Αν η παράμετρος `col` έχει την τιμή `TRUE`, η συνάρτηση πρέπει να επιστρέφει την n-στή στήλη του πίνακα. Διαφορετικά πρέπει να επιστρέφει την n-στή γραμμή.

33. Οι αριθμοί του Fibonacci έχουν ιδιαίτερο ενδιαφέρον στα μαθηματικά, αλλά και σε άλλες επιστήμες, καθώς δημιουργούν ενδιαφέροντες σχηματισμούς. Οι αριθμοί ορίζονται αναδρομικά, δηλαδή ισχύει

$$F_n = F_{n-2} + F_{n-1},$$

δηλαδή κάθε αριθμός προκύπτει ως το άθροισμα των δύο προηγούμενων του. Ισχύει επίσης πως $F_0 = 0, F_1 = 1, F_2 = 1$. Έτσι, για παράδειγμα, σύμφωνα με τον ορισμό: $F_3 = F_1 + F_2 = 1 + 1 = 2$. Οι πρώτοι αριθμοί Fibonacci είναι 0, 1, 1, 2, 3, 5, 8, 13,.... Μπορείτε εύκολα να επιβεβαιώσετε ότι κάθε αριθμός προκύπτει ως άθροισμα των 2 προηγούμενων του. Γράψτε μια συνάρτηση η οποία θα παίρνει σαν input το n (υποθέστε ότι είναι ακέραιος)

και θα επιστρέφει όλη τη σειρά μέχρι και το F_n. Δηλαδή αν ονομάσετε τη συνάρτηση fib, περιμένουμε όταν καλέσετε την fib(4) να επιστραφεί η ακολουθία 0,1,1,2,3 ενώ για fib(9) θα επιστραφεί το διάνυσμα 0, 1, 1, 2, 3, 5, 8, 13, 21, 34.

34. Γράψτε μια συνάρτηση η οποία θα παίρνει ως input δύο αριθμούς x,y και αν το άθροισμα τους είναι μεγαλύτερο ή ίσο του 100 θα ανταλλάσσει τους αριθμούς (δηλαδή το x θα πάρει την τιμή του y και αντίστροφα) ενώ αλλιώς θα επιστρέφει (χωρίς ανταλλαγή) τα 1/x και 1/y. Επίσης θα επιστρέφει το άθροισμα τους.

35. Ας υποθέσουμε πως οι συναρτήσεις dpois και ppois δεν είναι διαθέσιμες. Η συνάρτηση πιθανότητας της κατανομή Poisson είναι η

$$P(X = x) = \frac{e^{-\lambda}\lambda^x}{x!}, x = 0, 1, \ldots, \ \lambda > 0.$$

Επειδή όμως χρειάζεται να υπολογιστεί το παραγοντικό στον παρονομαστή, ο παραπάνω ορισμός δεν είναι ιδιαίτερα εύχρηστος. Για αυτό το λόγο στην πράξη χρησιμοποιούμε αναδρομικό τύπο και συγκεκριμένα, υπολογίζουμε την

$$P(X = 0) = e^{-\lambda},$$

και στη συνέχεια οι υπόλοιπες προκύπτουν ως

$$P(X = k) = P(X = k - 1)\frac{\lambda}{k}, \ \ k = 1, 2, \ldots \ .$$

Για παράδειγμα $P(X = 1) = P(X = 0)\frac{\lambda}{1}$, $P(X = 2) = P(X = 1)\frac{\lambda}{2}$ και ούτω καθεξής. Να γράψετε μια συνάρτηση που να έχει ως παραμέτρους εισόδου την τιμή της παραμέτρου και την τιμή του x για το οποίο θέλετε ν υπολογίσετε την πιθανότητα και η οποία να επιστρέφει ως αποτελέσματα την συνάρτηση πιθανότητας και τη συνάρτηση κατανομής στο σημείο x, και να κατασκευάζει τα διαγράμματα της συνάρτησης πιθανότητας και της συνάρτησης κατανομής από κοινού σε ένα διάγραμμα.

36. Ας υποθέσουμε πως οι συναρτήσεις `dbinom` και `pbinom` δεν είναι διαθέσιμες. Η συνάρτηση πιθανότητας της διωνυμικής είναι

$$P(X = x) = \left(\begin{array}{c} n \\ x \end{array} \right) p^x (1 - p)^{n-x}, x = 0, 1, \ldots, n \ p > 0.$$

Επειδή όμως, χρειάζεται να υπολογιστεί το παραγοντικό στον παρονομαστή, ο παραπάνω ορισμός δεν είναι ιδιαίτερα εύχρηστος. Γράψτε μια συνάρτηση που να υπολογίζει τη συνάρτηση πιθανότητας και μια άλλη που να υπολογίζει τη συνάρτηση κατανομής για δοθέν x και τιμές των παραμέτρων χρησιμοποιώντας αναδρομικούς τύπους (δηλαδή κάθε πιθανότητα θα ορίζεται από την προηγούμενη της).

37. Να γράψετε εντολές οι οποίες:

(a) Για δοθέν ακέραιο αριθμό x να βρίσκουν όλους τους ακέραιους που τον διαιρούν ακριβώς.

(b) Για κάθε ζεύγος αριθμών x και y να βρίσκουν το μέγιστο κοινό διαιρέτη.

(c) Να εξετάζουν αν ο αριθμός είναι πρώτος. Πρώτος είναι κάθε ακέραιος αριθμός που διαιρείται μόνο από το ένα (1) και τον εαυτό του (π.χ. 3,5,7,11 κλπ).

΄Ελεγχοι Υποθέσεων

38. Ο στατιστικός έλεγχος Kolmogorov-Smirnov για το αν ένα τυχαίο δείγμα $X = (X_1, X_2, \ldots, X_n)$ προέρχεται από συγκεκριμένη κατανομή στηρίζεται σε ελεγχοσυνάρτηση T, η οποία υπολογίζεται ως εξής:

    Για κάθε παρατήρηση X_i, υπολογίζεται αφενός η τιμή $F(X_i)$, της αθροιστικής συνάρτησης κατανομής της κατανομής από την οποία ελέγχουμε αν προέρχεται το δείγμα, και αφετέρου η τιμή $S(X_i)$ που ισούται με το ποσοστό των παρατηρήσεων του δείγματος (συμπεριλαμβανόμενης της X_i) οι οποίες είναι μικρότερες ή ίσες της X_i. Η T ισούται με τη μέγιστη τιμή που παίρνει η απόλυτη τιμή της διαφοράς $|F(X_i) - S(X_i)|$ στις παρατηρήσεις του δείγματος.

    Εκτιμήστε τα ποσοστιαία σημεία της κατανομής της ελεγχοσυνάρτησης όταν τα δεδομένα έχουν προέρχονται από εκθετική κατανομή με μέση τιμή 10 και μέγεθος δείγματος $n = 50$ και $n = 500$.

39. Στους ελέγχους υποθέσεων το επίπεδο στατιστικής σημαντικότητας υποδηλώνει την πιθανότητα τύπου I, δηλαδή την πιθανότητα να απορρίψουμε τη μηδενική υπόθεση ενώ αυτή είναι αληθής. ΄Εστω ο απλός έλεγχος για μια μέση τιμή από κανονικό πληθυσμό, με γνωστή διακύμανση σ^2. Αν $\bar{x}$ είναι ο δειγματικός μέσος από ένα δείγμα μεγέθους n τότε απορρίπτουμε σε α=5% τη μηδενική υπόθεση $H_0 : \mu = \mu_0$ έναντι της εναλλακτικής $H_1 : \mu \neq \mu_0$ όταν $Z = \left|\frac{\bar{x}-\mu_0}{\sigma/\sqrt{n}}\right|$>1.96.

    Στην πράξη αυτό σημαίνει πως αν το δείγμα πραγματικά προέρχεται από την κατανομή που υποθέτει η μηδενική υπόθεση τότε περίπου το 5% των φορών θα απορρίπτουμε λανθασμένα την υπόθεση αυτή. Να κατασκευάσετε μια συνάρτηση η οποία να έχει ως παραμέτρους εισόδου το μέγεθος του δείγματος n, την τιμή μ_0 και τον αριθμό B των επαναλήψεων που θα πάρετε και να επιστρέφει ως αποτέλεσμα το ποσοστό των φορών (από τα B δείγματα που πήρατε) που απορρίψατε τη μηδενική υπόθεση.

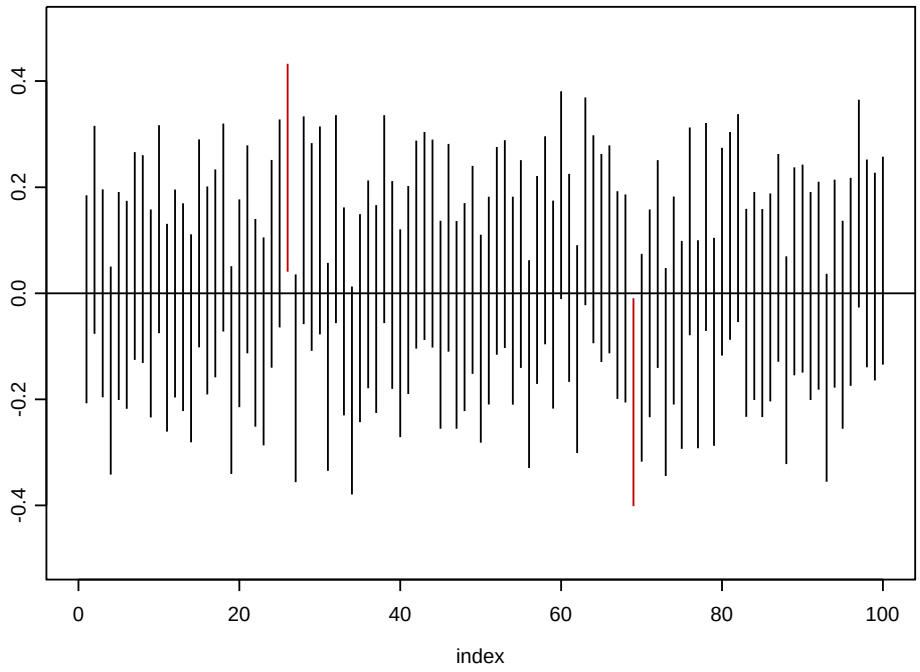
40. ΄Οταν κατασκευάζουμε ένα 95% διάστημα εμπιστοσύνης η ερμηνεία του είναι πως η πραγματική τιμή της παραμέτρου που εξετάζουμε θα ανήκει στο διάστημα με πιθανότητα 95%. Ας χρησιμοποιήσουμε προσομοίωση για να μελετήσουμε τα διαστήματα εμπιστοσύνης. Είναι γνωστό πως ένα 95% διάστημα εμπιστοσύνης για τη μέση τιμή όταν η διακύμανση είναι άγνωστη είναι το

$$\left[\bar{x} - t_{1-a/2,n-1}\frac{s}{\sqrt{n}},\ \bar{x} + t_{1-a/2,n-1}\frac{s}{\sqrt{n}}\right].$$

Γράψτε ένα πρόγραμμα το οποίο να προσομοιώνει B δείγματα μεγέθους n από κανονική κατανομή με μέση τιμή μ και διακύμανση $\sigma^2 = 1$. Για κάθε δείγμα κατασκευάστε το αντίστοιχο διάστημα εμπιστοσύνης και βρείτε το ποσοστό των διαστημάτων εμπιστοσύνης που περιέχουν τη μέση τιμή του πληθυσμού από όπου προσομοιώσατε τα δεδομένα. Τι παρατηρείτε;

Επαναλάβετε το ίδιο όμως προσομοιώστε τα δεδομένα από εκθετική κατανομή με μέση τιμή 1 και μεγέθη δείγματος $n = 10, 100, 1000$. Τι παρατηρείτε; Πως το ερμηνεύετε;

Χρησιμοποιήστε 100 από τα διαστήματα εμπιστοσύνης για να κατασκευάσετε μια γραφική απεικόνιση όπως αυτή στο Διάγραμμα 9.13. Κάθε κάθετη γραμμή είναι ένα διάστημα εμπιστοσύνης ενώ η οριζόντια γραμμή είναι η πραγματική τιμή του πληθυσμού. Κάποια διαστήματα είναι ολόκληρα πάνω ή κάτω από τη γραμμή.



Διάγραμμα 9.13: Διαστήματα εμπιστοσύνης δειγματικού μέσου για 100 διαφορετικά δείγματα

41. **Ελεγχος ανεξαρτησίας** χ^2 . Έστω ένας πίνακας συνάφειας (πίνακας διπλής εισόδου) με κελιά X_{ij}, $i = 1, \ldots, r$ και $j = 1, \ldots, c$, όπου X_{ij} είναι η συχνότητα της γραμμής i και της στήλης j. Ο έλεγχος ανεξαρτησίας γραμμών και στηλών γίνεται με την ελεγχοσυνάρτηση

$$X^2 = \sum_{i=1}^{r} \sum_{j=1}^{c} \frac{(X_{ij} - E_{ij})^2}{E_{ij}},$$

όπου $E_{ij} = Np_{i\cdot}p_{\cdot j}$, $p_{i\cdot} = \sum_{j=1}^{c} X_{ij}/N$ και $p_{\cdot j} = \sum_{i=1}^{r} X_{ij}/N$, δηλαδή οι σχετικές συχνότητες κάθε γραμμής και κάθε στήλης αντίστοιχα, όπου N είναι το συνολικό μέγεθος του δείγματος. Γράψτε ένα πρόγραμμα που από έναν πίνακα συνάφειας A να υπολογίζει την τιμή της ελεγχοσυνάρτησης. Χρησιμοποιήστε τον πίνακα που αναφέρεται στο πόσο εύκολο βρήκαν ένα διαγώνισμα 110 φοιτητές και φοιτήτριες.

	Καθόλου	Λίγο	Πολύ
Φοιτητές	23	18	12
Φοιτήτριες	12	21	24

.

Στη συνέχεια γράψτε μια συνάρτηση η οποία να παίρνει ως παραμέτρους εισόδου δύο διανύσματα με παρατηρήσεις και από αυτά να κατασκευάζει τον πίνακα συνάφειας. Πως μπορείτε να συνδυάσετε τις 2 συναρτήσεις (μια που να μετατρέπει τα δεδομένα σε πίνακα συνάφειας και μια που να κάνει τον έλεγχο);

42. Έστω ότι έχουμε 2 δείγματα από διωνυμικές τυχαίες μεταβλητές και θέλουμε να ελέγξουμε την ισότητα των πιθανοτήτων επιτυχίας στα δύο δείγματα δηλαδή επιθυμούμε να ελέγξουμε την υπόθεση $H_0 : \pi_1 = \pi_2$ έναντι της εναλλακτικής $H_1 : \pi_1 \neq \pi_2$. Το πρόβλημα είναι τι μέγεθος δείγματος χρειαζόμαστε να πάρουμε από τους δύο πληθυσμούς έτσι ώστε να μην απορρίπτουμε την H_0 επιτυχώς (δηλαδή όταν είναι αληθής) με πιθανότητα $1 - \alpha$ και ταυτόχρονα να μπορούμε να εντοπίσουμε επιτυχώς (απορρίπτοντας την H_0) διαφορές μεταξύ των πιθανοτήτων ίσες με δ με πιθανότητα $1-\beta$. Οι πιθανότητες α και β ονομάζονται και σφάλματα τύπου I και II. Χρησιμοποιώντας αυτές τις συνθήκες μπορούμε να υπολογίσουμε το μέγεθος δείγματος το οποίο ικανοποιεί τη συνθήκη

$$P(\text{Απόρριψη }_0 | \ |\pi_1 - \pi_2| = \delta) = 1 - \beta$$

καταλήγοντας στον τύπο

$$n_1 = \frac{1}{\delta^2}\left(z_{1-\beta}\sqrt{\pi_1(1-\pi_1) + \pi_2(1-\pi_2)/k} + z_{1-\alpha/2}\sqrt{\pi(1-\pi)(1+1/k)}\right)^2$$

όπου $n_2 = kn_1$, π_1 και π_2 είναι οι πιθανότητες επιτυχίας σε κάθε δείγμα. Οι ποσότητες αυτές συνήθως βρίσκονται από τη βιβλιογραφία ή από μικρές πιλοτικές έρευνες. Επιπλέον το κοινό ποσοστό π δίνεται από τον τύπο $\pi = (\pi_1 + k\pi_2)/(1 + k)$.

α) Να κατασκευάσετε μία συνάρτηση που να δίνονται ως γνωστά το επίπεδο σημαντικότητας α (σφάλμα τύπου I) το σφάλμα τύπου II (δηλαδή το β) όταν η πραγματική διαφορά είναι ίση με δ και θα υπολογίζει το επιθυμητό μέγεθος δείγματος

β) Έστω ότι το ποσοστό εμφάνισης μίας νόσου είναι 150 ανά 100000 άτομα και θέλουμε να εντοπίσουμε μια μείωση του κινδύνου κατά $\delta = 20$ ποσοστιαίες μονάδες. Να βρεθεί το μέγεθος δείγματος αν υποθέσουμε ίσο αριθμό ατόμων στις 2 ομάδες.

γ) Στο παραπάνω παράδειγμα να βρεθεί το μέγεθος δείγματος αν η πρώτη ομάδα έχει το μισό αριθμό ατόμων από ότι η δεύτερη.

Πίνακες

43. Έστω οι πίνακες:

$$X = \begin{bmatrix} 1 & 2 & -2 \\ 1 & 3 & -4 \\ 1 & 4 & 3 \\ 1 & 0 & 5 \\ 1 & 6 & 7 \\ 1 & 4 & 8 \\ 1 & 3 & 9 \\ 1 & 7 & 11 \end{bmatrix} \text{ και } Y = \begin{bmatrix} 2 \\ 8 \\ 7 \\ 6 \\ 4 \\ 3 \\ 2 \\ 1 \end{bmatrix}$$

Να υπολογίσετε τις ποσότητες

(a) $(X'X)^{-1}$

(b) $(X'X)^{-1}X'Y$

(c) $X(X'X)^{-1}X'$

(d) $(I - (X'X)^{-1})X'$

(e) Να βρείτε τις ιδιοτιμές του πίνακα $(X'X)^{-1}$ και να επαληθεύσετε τη φασματική του ανάλυση.

44. Η εντολή `rnorm(n, m, s)` προσομοιώνει n τιμές από κανονική κατανομή με μέση τιμή m και τυπική απόκλιση s.

- Φτιάξτε έναν πίνακα με 10 στήλες όπου κάθε στήλη θα περιέχει 100 τιμές από κανονική κατανομή με την παραπάνω εντολή
- Στη συνέχεια φτιάξτε έναν καινούριο πίνακα ο οποίος να έχει ως πρώτη στήλη τη στήλη του προηγούμενου πίνακα με τη μικρότερη διακύμανση, σαν δεύτερη στήλη τη στήλη του προηγούμενου πίνακα με τη δεύτερη μικρότερη και ούτω καθεξής

45. Βρείτε τις εντολές για να κατασκευάσετε τον εξής πίνακα

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 6 & 6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 7 & 0 & 0 & 7 & 0 & 0 & 0 \\ 0 & 0 & 8 & 0 & 0 & 0 & 0 & 8 & 0 & 0 \\ 0 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 9 & 0 \\ 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 9 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 & 8 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 7 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 & 8 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 9 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 \end{bmatrix}$$

46. Έστω A ένας συμμετρικός και θετικά ορισμένος πίνακας διαστάσεων $p \times p$. Έστω αριθμός $q < p$ ο οποίος μπορεί να χρησιμοποιηθεί για να διαμερίσει τον πίνακα A ως εξής

$$A = \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix},$$

όπου A_1 είναι ένας $q \times q$ πίνακας που περιέχει το πάνω αριστερά μέρος του αρχικού πίνακα A και ομοίως A_2 είναι πίνακας $q \times (p-q)$ με το πάνω αριστερά μέρος και αντίστοιχα οι πίνακες A_3 διαστάσεων $(p-q) \times q$ και A_4 διαστάσεων $(p-q) \times (p-q)$.

Να γράψετε εντολές οι οποίες να εισάγουν έναν πίνακα , να ελέγχουν αν είναι συμμετρικός και θετικά ορισμένος και στη συνέχεια να τον διαμερίζουν και να υπολογίζουν τον πίνακα $A_1 - A_2 A_4^{-1} A_3$ και την ορίζουσα του για δοθείσα τιμή $q < p$.

Παράδειγμα διαμέρισης. Αν ο πίνακας A είναι ο εξής

$$A = \begin{bmatrix} 1 & 0 & 0.5 & -0.3 & 0.2 \\ 0 & 1 & 0.1 & 0 & 0 \\ 0.5 & 0.1 & 1 & 0.3 & 0.7 \\ -0.3 & 0 & 0.3 & 1 & 0.4 \\ 0.2 & 0 & 0.7 & 0.4 & 1 \end{bmatrix}$$

τότε βρίσκουμε πως

$$A_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, A_2 = \begin{bmatrix} 0.5 & -0.3 & 0.2 \\ 0.1 & 0 & 0 \end{bmatrix},$$

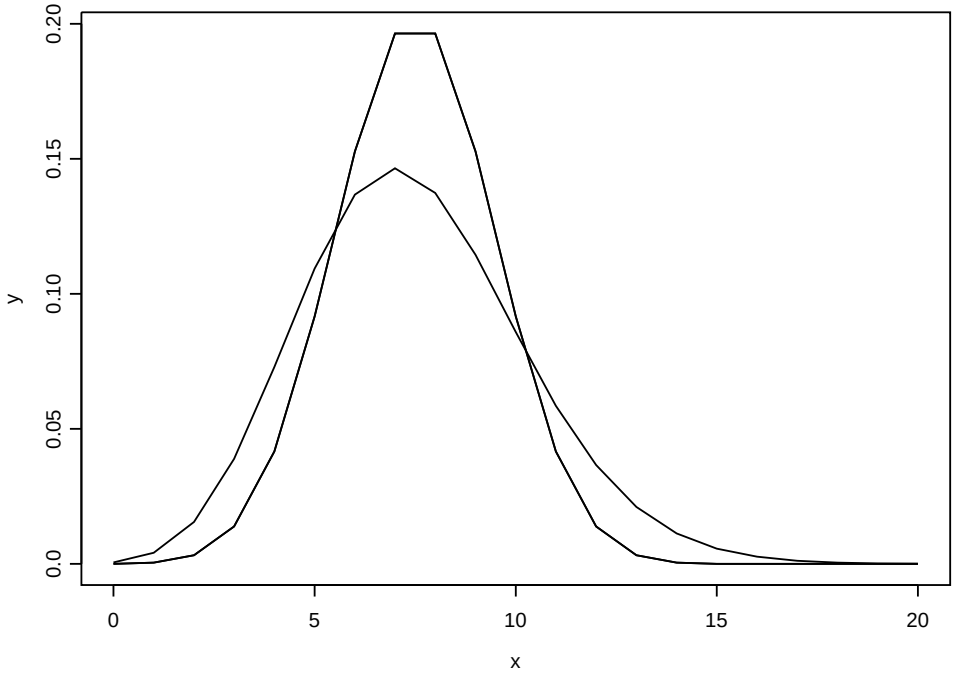
$$A_3 = \begin{bmatrix} 0.5 & 0.1 \\ -0.3 & 0 \\ 0.2 & 0 \end{bmatrix}, A_4 = \begin{bmatrix} 1 & 0.3 & 0.7 \\ 0.3 & 1 & 0.4 \\ 0.7 & 0.4 & 1 \end{bmatrix}.$$

47. Από τη θεωρία κατανομών γνωρίζουμε πως δύο βασικές διακριτές κατανομές είναι η διωνυμική κατανομή και η κατανομή Poisson. Θέλουμε να κατασκευάσουμε ένα διάγραμμα το οποίο να συγκρίνει τις δύο αυτές κατανομές. Επομένως για δοθείσες τιμές των παραμέτρων της διωνυμικής κατανομής βρείτε την τιμή της παραμέτρου της κατανομής Poisson που οδηγεί στην ίδια αναμενόμενη τιμή και στη συνέχεια σε ένα διάγραμμα τοποθετήστε τις συναρτήσεις πυκνότητας πιθανότητας των δύο κατανομών με τις μορφές που δίνονται στα Διαγράμματα 9.14 και 9.15.

Ποιο από τα δύο διαγράμματα νομίζετε ότι απεικονίζει πιο σωστά τις κατανομές; Αιτιολογήστε την απάντηση σας.

Να κατασκευάσετε τα ίδια διαγράμματα για τις συναρτήσεις αθροιστικής πιθανότητας των παραπάνω κατανομών.

48. Ας υποθέσουμε πως η R δεν διέθετε συγκεκριμένο τελεστή για να κάνει πολλαπλασιασμό πινάκων (δηλαδή την πράξη %*%) Δημιουργήστε μια συνάρτηση οι οποία να παίρνει ως



Διάγραμμα 9.14:

input δύο πίνακες $\mathbf{A}$ και $\mathbf{B}$ και να επιστρέφει ως αποτέλεσμα το γινόμενο τους αν βέβαια η πράξη μπορεί να γίνει. Στην περίπτωση που οι διαστάσεις των πινάκων δεν επιτρέπουν τον πολλαπλασιασμό η συνάρτηση θα πρέπει να επιστρέφει έναν πίνακα με στοιχεία `NA` διαστάσεων $m \times n$ όπου m το πλήθος των γραμμών του $\mathbf{A}$ και n το πλήθος των γραμμών του $\mathbf{B}$.

49. Έστω ότι έχουμε τους πίνακες $\mathbf{A}$ και $\mathbf{B}$ οι οποίοι είναι οι ακόλουθοι:

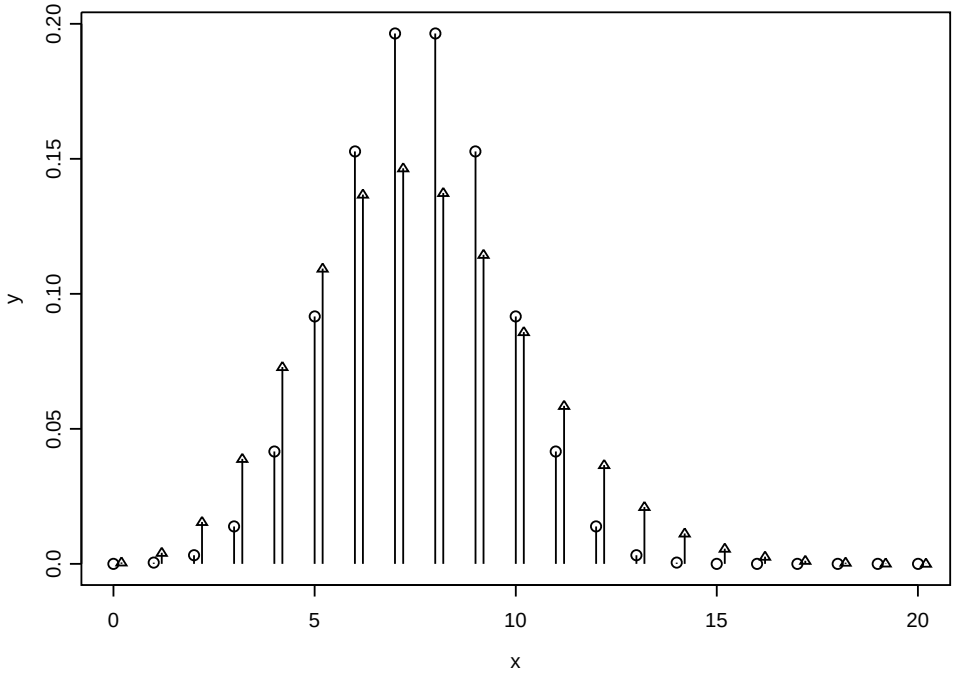
$$\mathbf{A} = \begin{pmatrix} 1 & 2 & 6 \\ 5 & 2 & 5 \\ 6 & 1 & 3 \end{pmatrix} \text{και ο } \mathbf{B} = \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{pmatrix}.$$

   (a) Ορίστε τους παραπάνω πίνακες στο R.

   (b) Με ποια εντολή θα προκύψει ο πίνακας $\mathbf{\Gamma} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{pmatrix}$ από τον πίνακα $\mathbf{A}$.

   (c) Με ποιες εντολές θα υπολογίσετε την ακόλουθη ποσότητα

$$5\mathbf{A}^{-1} + 3\mathbf{A}\mathbf{A}^T - 2(\mathbf{B}\mathbf{B}^T)^{-1} + \mathbf{I}_3 + \begin{pmatrix} 5 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 3 \end{pmatrix},$$



Διάγραμμα 9.15:

όπου $_3$ είναι ο μοναδιαίος πίνακας διάστασης 3×3.

Στατιστική

50. Ο συντελεστής συσχέτισης του Spearman είναι ένας εναλλακτικός τρόπος μέτρησης της συσχέτισης μεταξύ 2 μεταβλητών. Μάλιστα η συσχέτιση που μετράει ο δείκτης αυτός δεν είναι απαραίτητα γραμμική αλλά απλά μονότονη. Ο συντελεστής αυτός είναι εύκολα διαθέσιμος στην R αλλά όχι στην R. ΄Ενας έμμεσος (αλλά ισοδύναμος) τρόπος για τον υπολογισμό του είναι να υπολογίσουμε κλασσικό συντελεστή συσχέτισης του Pearson όχι όμως στα αρχικά δεδομένα αλλά όταν αντικαταστήσουμε τις τιμές με τη σειρά κατάταξης τους. Δημιουργήστε μια συνάρτηση που να υπολογίζει το συντελεστή του Spearman χρησιμοποιώντας αυτή την προσέγγιση.

51. P-P plot: Το P-P plot είναι ένας γρήγορος τρόπος να δούμε αν τα δεδομένα μας ακολουθούν μια κατανομή που έχουμε υποθέσει. Το διάγραμμα στην πιο απλή του μορφή έχει στον άξονα των x την θεωρητική τιμή της αθροιστικής κατανομής που θέλουμε να ελέγξουμε και στον άξονα των y την αντίστοιχη εμπειρική κατανομή, δηλαδή το ποσοστό των τιμών του δείγματος που είναι μικρότερες ή ίσες από το x που ελέγχουμε. Αν η κατανομή που έχουμε υποθέσει είναι σωστή τότε τα σημεία πρέπει να είναι πάνω στη διαγώνιο. Φτιάξτε μια συνά-

ρτηση που να έχει ως input ένα διάνυσμα με δεδομένα και να κατασκευάζει ένα διάγραμμα για να ελέγξουμε μια κανονική κατανομή με μέση τιμή και διακύμανση τη δειγματική μέση τιμή και διακύμανση αντίστοιχα. Στο διάγραμμα τοποθετήστε και τη διαγώνιο για να είναι πιο εύκολη η σύγκριση.

52. Σε συνέχεια της προηγούμενης άσκησης

    (a) προσομοιώστε δεδομένα από τις εξής κατανομές: Student-t , Γάμμα, βήτα, εκθετική, Weibull με διάφορες παραμέτρους και μελετήστε τον τρόπο που τα στοιχεία σας αποκλίνουν από τη διαγώνιο. Τέτοιου είδους αποκλίσεις ουσιαστικά σας δίνουν πολύτιμες πληροφορίες για να διαγνώσετε ποιά κατανομή θα μπορούσε να χρησιμοποιηθεί

    (b) Τέλος κατασκευάστε τα P-P plots αλλά με χρήση των κατανομών από όπου προσομοιώσατε τα δεδομένα κάθε φορά. Πόσο αποκλίνουν από τη διαγώνιο;

53. Από τη θεωρία κατανομών έχουμε μάθει πως για κάποιες κατανομές δεν υπάρχουν οι ροπές συγκεκριμένης τάξης. Τι σημαίνει όμως αυτό στην πράξη; Για να δούμε λοιπόν τι ακριβώς σημαίνει χρησιμοποιήστε την εντολή `rcauchy(1000,0,1)` που προσομοιώνει 1000 τιμές από την κατανομή Cauchy με διάμεσο 0 και παράμετρο κλίμακας 1. Υπενθυμίζουμε ότι η κατανομή Cauchy έχει συνάρτηση (μάζας) πυκνότητας πιθανότητας

$$f(x;\mu,\beta) = \frac{1}{\pi}\frac{\beta}{(x-\mu)^2+\beta^2},$$

    όπου μ είναι η διάμεσος και β η παράμετρος κλίμακας. Για αυτή την κατανομή δεν υπάρχει η μέση τιμή δηλαδή το ολοκλήρωμα $\int xf(x)dx = \infty$.

    (a) Πάρτε διαδοχικά δείγματα μεγέθους 1000 από αυτή την κατανομή και για κάθε δείγμα υπολογίστε τη δειγματική μέση τιμή. Τι παρατηρείτε;

    (b) Στη συνέχεια κάντε το εξής. Πάρτε ένα δείγμα μεγέθους 100 και υπολογίστε τη μέση του τιμή. Στη συνέχεια προσθέστε άλλες 100 παρατηρήσεις και υπολογίστε τη μέση τιμή των 200 παρατηρήσεων και διαδοχικά να αυξάνετε το μέγεθος του δείγματος κατά 100 μέχρι να φτάσετε τις 10000 παρατηρήσεις αποθηκεύοντας κάθε φορά τη μέση τιμή του συνολικού δείγματος. Αν κάνετε ένα διάγραμμα σε αυτή τη μέση τιμή (στη στατιστική ονομάζεται και εργοδικός μέσος) τι παρατηρείτε;

    (c) Επαναλάβετε τη διαδικασία αλλά τώρα προσομοιώστε από την τυποποιημένη κανονική κατανομή. Συγκρίνετε τα 2 διαγράμματα. τι βλέπετε; Γιατί συμβαίνει αυτό;

54. Γράψτε εντολές στην R για να παίξετε το εξής παιχνίδι.

    (a) Ξεκινάτε με 10 Ευρώ. Στρίβετε ένα νόμισμα, αν έρθει κορώνα κερδίζετε ένα Ευρώ ενώ αν έρθει γράμματα χάνετε ένα Ευρώ. Το παιχνίδι σταματά όταν χάσετε όλα σας

τα χρήματα. το πρόγραμμα σας θα πρέπει να καταγράφει τον αριθμό της ρίψης του νομίσματος μέχρι να τελειώσει το παιχνίδι

(b) Επαναλάβετε την παραπάνω διαδικασία 100 φορές και καταγράψτε για τις 100 αυτές επαναλήψεις τον αριθμό των φορών που χρειάστηκε να στρίψετε το νόμισμα.

(c) Στο παραπάνω πείραμα υποθέστε ότι έχετε 100 Ευρώ αρχικά. Τι αλλάζει στο παιχνίδι αν και ξεκινάτε με 100 Ευρώ; πόσο πιθανό είναι να χρεοκοπήσετε κάποια στιγμή;

(Βοήθεια: Για να στρίψετε ένα νόμισμα στον υπολογιστή αρκεί να γεννήσετε μια τυχαία μεταβλητή με την εντολή `runif(1,0,1)` και αν η τιμή είναι μικρότερη από 0.5 υποθέστε πως έχετε κορώνα ενώ αν είναι μεγαλύτερη από 0.5 γράμματα).

55. Σε πολλές έρευνες νοικοκυριών τα δεδομένα που αφορούν τις ηλικίες των μελών του νοικοκυριού δίνονται με τη μορφή ενός διανύσματος που περιέχει τις ηλικίες των μελών αλλά για να μπορέσει ο ερευνητής να περιλάβει όλα τα μέλη υπάρχουν αρκετά κενά κελιά. Για παράδειγμα αν έχει προβλεφθεί να υπάρχουν 10 στήλες ώστε να περιληφθούν οι ηλικίες για νοικοκυριά μέχρι 10 ατόμων τα δεδομένα έχουν τη μορφή

$$\begin{array}{cccccccccc} 28 & 26 & 2 & 1 & NA & NA & NA & NA & NA & NA \\ 54 & 48 & 24 & 18 & 16 & NA & NA & NA & NA & NA \\ 21 & NA & NA & NA & NA & NA & NA & NA & NA & NA \\ \ldots & & & & & & & & & \end{array}.$$

Όμως σε πολλές εφαρμογές μας ενδιαφέρει να μετρήσουμε τον αριθμό των μελών του νοικοκυριού που ανήκουν σε μια συγκεκριμένη ηλικιακή ομάδα οπότε τα αρχικά δεδομένα πρέπει να μετασχηματιστούν. Για παράδειγμα αν μας ενδιαφέρουν οι ηλικιακές ομάδες $< 18, 19-45, 46-65, > 65$ τότε τα δεδομένα μας αντιστοιχούν

$$\begin{array}{cccc} 2 & 2 & 0 & 0 \\ 1 & 3 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ \ldots & & & \end{array}.$$

Γράψτε ένα πρόγραμμα το οποίο να διαβάζει από ένα αρχείο δεδομένα στην αρχική μορφή και να τα μετασχηματίζει στη μορφή που θέλουμε. Υποθέστε πως δεν μπορούμε να έχουμε στα δεδομένα νοικοκυριό χωρίς μέλη.

56. Πολλές φορές όταν θέλουμε να κάνουμε ANOVA τα δεδομένα μας δίνονται σε μορφή πινάκων κάτι που δεν μας διευκολύνει σε κανένα στατιστικό πακέτο να κάνουμε την ανάλυση. Σε αυτή την περίπτωση θα πρέπει να τα μετασχηματίσουμε έτσι ώστε κάθε μεταβλητή να είναι μια στήλη. Έστω ο παρακάτω πίνακας από ανάλυση διακύμανσης με δύο

παράγοντες και 4 παρατηρήσεις για κάθε κελί

		Παράγοντας 1		
		A	B	C
Παραγοντας 2	A	$15, 12, 18, 21$	$16, 14, 12, 9$	$17, 14, 25, 13$
	B	$8, 11, 10, 9$	$9, 15, 11, 14$	$11, 15, 16, 21$
	C	$12, 14, 15, 11$	$15, 12, 9, 9$	$17, 21, 23, 21$

.

Γράψτε ένα πρόγραμμα το οποίο να δημιουργεί για τον παραπάνω πίνακα τα δεδομένα που χρειαζόμαστε σε κάθε στατιστικό πακέτο για να δουλέψουμε με ANOVA. Δηλαδή τα δεδομένα θα πρέπει να έχουν 3 στήλες και να μοιάζουν ως εξής

$$\begin{array}{ccc} A & A & 15 \\ A & A & 12 \\ A & A & 18 \\ A & A & 21 \\ A & B & 8 \\ \ldots & & \end{array}$$

.

57. Ας υποθέσουμε πως δεν υπάρχει διαθέσιμη εντολή για να δίνει τον ανάστροφο ενός πίνακα. Γράψτε εντολές οι οποίες για δεδομένο πίνακα **A** να βρίσκουν τον αντίστροφο του χρησιμοποιώντας φωλιασμένες εντολές `for`.

    Γράψτε εντολές που να βρίσκουν τον αντίστροφο αλλά τώρα χρησιμοποιώντας μόνο μια εντολή `for`.

58. Σε ένα αρχείο μια ασφαλιστική εταιρεία κρατάει τα εξής δεδομένα: η πρώτη στήλη αφορά τον αριθμό του συμβολαίου και η δεύτερη στήλη το ύψος της ζημιάς που πλήρωσε. Δεδομένου πως για κάθε συμβόλαιο μπορεί να υπάρχουν περισσότερες από μια εγγραφές (καθώς κάποιος μπορεί να έχει περισσότερα από ένα ατυχήματα μέσα στην εξεταζόμενη χρονική περίοδο) στο αρχείο υπάρχουν περισσότερες από μια εγγραφές για κάποια συμβόλαια. Γράψτε ένα πρόγραμμα το οποίο να διαβάζει ένα τέτοιο αρχείο και να κατασκευάζει στη συνέχεια ένα αντίστοιχο όπου τώρα για κάθε συμβόλαιο να υπάρχει μόνο μια εγγραφή με το συνολικό ποσό που πλήρωσε η εταιρεία.

Αριθμητικές μέθοδοι

59. Ένας προσεγγιστικός τρόπος να υπολογιστούν ολοκληρώματα είναι η *μέθοδος Monte Carlo*. Την παρουσιάζουμε με ένα παράδειγμα:

    Το ολοκλήρωμα $\int_0^1 x^2 dx$ υπολογίζεται ως εξής: παράγουμε ένα μεγάλο πλήθος (έστω n) τιμών x, από Ομοιόμορφη κατανομή στο διάστημα [0,1]. Κατόπιν παράγουμε ένα ίσο αριθμό

τιμών y, επίσης από Ομοιόμορφη κατανομή στο διάστημα [0,1]. ΄Ετσι σχηματίζονται n ζεύγη τιμών (x, y). Δηλαδή η πρώτη x τιμή με την πρώτη y, η δεύτερη x τιμή με την δεύτερη y, κλπ. ΄Εστω z το πλήθος των ζευγών στα οποία ισχύει $y < x^2$. Τότε μία προσέγγιση του παραπάνω ολοκληρώματος είναι το πηλίκο z/n.

Γράψτε μία συνάρτηση η οποία παίρνει ως παράμετρο εισόδου το πλήθος n των ζευγών (x, y) που θέλουμε να παράγουμε και υπολογίζει προσεγγιστικά το παραπάνω ολοκλήρωμα.

60. Ξέρουμε πως η αναμενόμενη τιμή μιας συνεχούς τυχαίας μεταβλητής με συνάρτηση (μάζας) πυκνότητας πιθανότητας $f(x)$ δίνεται ως

$$E(X) = \int xf(x)dx \ .$$

Στην πράξη για να εκτιμήσουμε αυτή την ποσότητα χρησιμοποιούμε το δειγματικό μέσο $\bar{x} = \sum X_i/n$ όπου οι παρατηρήσεις X_i, $i = 1, \ldots, n$ είναι τυχαίες μεταβλητές από την κατανομή $f(x)$. Αυτό μας οδηγεί στην ιδέα πως αν θέλουμε να εκτιμήσουμε ένα ολοκλήρωμα της μορφής

$$\int \phi(x)dx$$

αρκεί να το γράψουμε ως μια αναμενόμενη τιμή διαλέγοντας μια κατάλληλη πυκνότητα. Για παράδειγμα αν $h(x)$ είναι μια σππ τότε

$$\int \phi(x)dx = \int \frac{\phi(x)}{h(x)}h(x)dx = E\left[\frac{\phi(x)}{h(x)}\right],$$

και επομένως αν πάρουμε ένα δείγμα (προσομοιώσουμε) από την $h(x)$ τότε μπορούμε να εκτιμήσουμε με τη χρήση του δειγματικού μέσου το ζητούμενο ολοκλήρωμα. Στο παράδειγμα μας αν πάρουμε δείγμα $X_1, \ldots, X_n$ από την $h(x)$ και ορίσουμε

$$Y_i = \frac{\phi(X_i)}{h(X_i)}$$

τότε το παραπάνω ολοκλήρωμα μπορεί να προσεγγιστεί με τη χρήση της μέσης τιμής των Y_i δηλαδή με τη χρήση του $\bar{y} = \sum Y_i/n$.

Χρησιμοποιήστε αυτή την προσέγγιση για να υπολογίσετε τα παρακάτω ολοκληρώματα.

$$\int_0^1 (x^2 + 4)dx, \qquad \int_0^1 x(1 - x)dx \ .$$

Υποθέστε πως η $h(x)$ που χρησιμοποιήσαμε πριν είναι η ομοιόμορφη κατανομή στο $(0, 1)$. Ελέγξτε την ακρίβεια της προσέγγισης σε σχέση με την πραγματική τιμή. Παρατηρήστε επίσης πως βελτιώνεται η εκτίμηση αυτή ανα αυξήσουμε το μέγεθος του δείγματος που θα πάρουμε.

61. **Η μέθοδος Newton-Raphson** Έστω ότι θέλουμε να λύσουμε την εξίσωση

$$F(\theta) = 0 \ .$$

Η μέθοδος Newton-Raphson (N-R) είναι μια επαναληπτική μέθοδος επίλυσης εξισώσεων, δηλαδή σε κάθε επανάληψη βελτιώνουμε την ήδη υπάρχουσα λύση. Η μέθοδος ξεκινά με μια αρχική λύση $\theta^{(0)}$ (η οποία μπορεί να επιλεγεί είτε τυχαία είτε με βάση κάποια λογική επιλογή) και σε κάθε επανάληψη ανανεώνουμε τη λύση μας. Αν είμαστε στην r επανάληψη τότε η καινούρια λύση προκύπτει χρησιμοποιώντας τον παρακάτω τύπο

$$\theta^{(r+1)} = \theta^{(r)} - \frac{F(\theta^{(r)})}{F'(\theta^{(r)})} \ .$$

Οι επαναλήψεις συνεχίζονται μέχρι κάποιο κριτήριο τερματισμού να ικανοποιηθεί, για παράδειγμα σταματάμε αν η διαφορά ανάμεσα σε δύο διαδοχικές λύσεις είναι μικρότερη από 10^{-4}, δηλαδή αν $|\theta^{(r+1)} - \theta^{(r)}| \leq 10^{-4}$.

Να χρησιμοποιήσετε τη μέθοδο N-R για να επιλύσετε την εξίσωση $x^3 - 2x^2 = -8$.

62. Η περικομμένη κατανομή Poisson (truncated Poisson distribution) προκύπτει από την απλή κατανομή Poisson όταν η τιμή 0 δεν μπορεί να παρατηρηθεί. Η συνάρτηση πιθανότητας της κατανομής δίνεται από τον τύπο

$$P(X = x) = \frac{exp(-\lambda)\lambda^x}{x!(1 - exp(-\lambda))} = \frac{\lambda^x}{x!(e^\lambda - 1)}, \quad \lambda > 0, x = 1, 2, \dots \ .$$

Από ένα τυχαίο δείγμα $(x_1, x_2, \dots, x_n)$ η λογαριθμική συνάρτηση πιθανοφάνειας δίνεται ως

$$\ell(\lambda) = \log\lambda \sum_{i=1}^{n} x_i - n \log\left(e^\lambda - 1\right) - \sum_{i=1}^{n} (\log x_i!) \ .$$

Η μεγιστοποίηση της πιθανοφάνειας δεν μπορεί να γίνει σε κλειστή μορφή και για αυτό θα καταφύγουμε στη μέθοδο N-R.

Τα δεδομένα που ακολουθούν αναφέρονται στην συχνότητα εμφάνισης των λέξεων σε ένα κείμενο. Έτσι 797 λέξεις εμφανίζονται μόνο μια φορά στο κείμενο, 301 λέξεις εμφανίζονται 2 φορές κλπ.

x	1	2	3	4	5	6	7
	797	301	77	17	6	1	1

.

a) Κατασκευάστε το διάγραμμα της λογαριθμικής πιθανοφάνειας ως συνάρτηση της παραμέτρου.

b) Εκτιμήστε την τιμή της παραμέτρου χρησιμοποιώντας τη μέθοδο μεγίστης πιθανοφάνειας. Ξεκινήστε από διαφορετικές αρχικές τιμές.

63. **Η μέθοδος της διχοτόμησης**. Η μέθοδος της διχοτόμησης είναι ένας απλός τρόπος επίλυσης εξισώσεων όταν ξέρουμε πως η συνάρτηση είναι μονότονη. Η ιδέα είναι η εξής: Έχουμε δύο σημεία x_L, x_U και ξέρουμε πως η λύση βρίσκεται ανάμεσα τους. Τότε κόβουμε το διάστημα (x_L, x_U) σε δύο ίσα μέρη $(x_L, (x_U + x_L)/2)$ και $((x_U + x_L)/2, x_U)$ και συνεχίζουμε κάθε φορά να ψάχνουμε στο κομμάτι που ξέρουμε πως υπάρχει η λύση επαναπροσδιορίζοντας κατάλληλα τα άκρα του διαστήματος, μέχρι να έχουμε διαστήματα πολύ μικρά οπότε και βρίσκουμε τη λύση με όση ακρίβεια θέλουμε.

    Χρησιμοποιήστε τη μέθοδο της διχοτόμησης για να βρείτε το 0.70 ποσοστιαίο σημείο της κανονικής κατανομής (υποθέτουμε πως δεν είναι διαθέσιμη η συνάρτηση qnorm).

Εργασίες

64. Η ΕΥΔΑΠ χρησιμοποιεί για τις χρεώσεις της κλιμακωτό τιμολόγιο, δηλαδή τα πρώτα κυβικά κατανάλωσης έχουν διαφορετική χρέωση από τα επόμενα κλπ. Συγκεκριμένα το τιμολόγιο (Φεβρ 07) είναι το εξής

Κυβικά	τιμή (σε ευρώ)
$0-5$	0.39
$5-20$	0.61
$20-27$	1.75
$27-35$	2.45
>35	3.05

    .

    Για παράδειγμα κάποιος με κατανάλωση 22 κυβικά μέτρα, θα πληρώσει για τα πρώτα 5 κυβικά από 0.39 ευρώ, για τα επομενα 15 προς 0.61 ευρώ και για τα εναπομειναντα 2 κυβικά προς 1.75 ευρώ.

    Γράψτε ένα προγραμμα το οποίο θα διαβάζει από ένα αρχείο τις καταναλώσεις των νοικοκυριών και θα υπολογίζει το ποσό που πρέπει να πληρώσει κάθε νοικοκυριό.

65. Σας δίνεται ο παρακάτω $2 \times 2 \times 2 \times 2$ πίνακας συνάφειας που δείχνει τη συχνότητα για το κάθε συνδυασμό των τεσσάρων μεταβλητών.

    - Να εισάγετε τα παραπάνω δεδομένα με τη μορφή array .
    - Να εισάγετε τα παραπάνω δεδομένα με τη μορφή matrix.
    - Να εισάγετε τα παραπάνω δεδομένα με τη μορφή data.frame.
    - Σε κάθε διαδικασία να γράψετε και να εξηγήσετε τις εντολές που χρησιμοποιήσατε.
    - Ποια τα πλεονεκτήματα τις κάθε προσέγγισης; Γιατί θα ήταν βολικό να χρησιμοποιήσουμε array;
    - Να εξάγετε τον αριθμό των γυναικών με θετική απόκριση (response) όταν έχουμε με low density και γυναίκες που παραβίασαν το προσωπικό χώρο του συνάνθρωπου τους

(1) Density	(2) Sex of subject	(3) Sex of Intruder	(4) Response Yes	No
Low	Male	Male	18	1
		Female	15	8
	Female	Male	17	5
		Female	12	7
High	Male	Male	13	6
		Female	16	4
	Female	Male	10	9
		Female	14	6

(μεταβλητή sex of intruder) από το κάθε τύπο αντικειμένων. Σε ποιο τύπο ήταν πιο άμεσο;

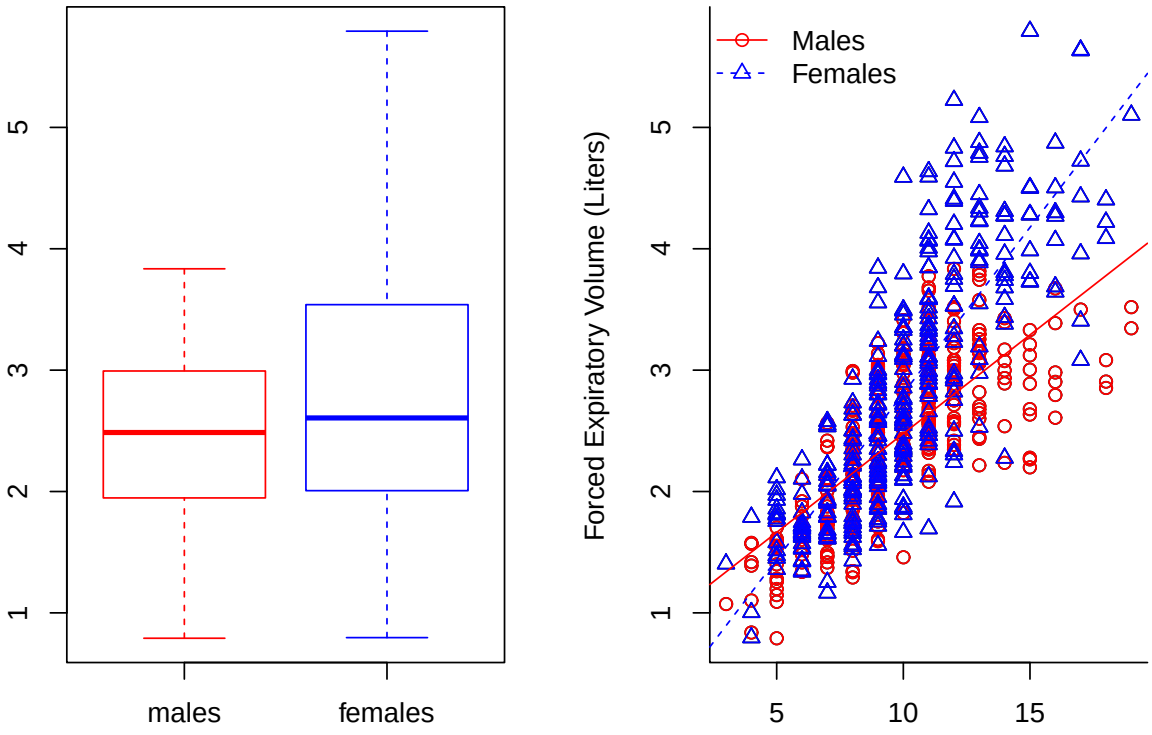
- Τέλος να επιλέξετε να δείτε τα χαρακτηριστικά όλων των ανδρών σε κάθε τύπο αντικειμένων.

66. Να διαβάσετε τα 4 σετ δεδομένων που βρίσκονται στην ιστοσελίδα του συγγράμματος. Όσον αφορά το αρχείο `data10\_SMOKE.DAT` (που είναι και το πιο δύσκολο) να διαβάσετε το αντίστοιχο `readme` αρχείο για δείτε τις λεπτομέρειες των μεταβλητών.

    - Σε κάθε διαδικασία να γράψετε και να εξηγήσετε τις εντολές που χρησιμοποιήσατε καθώς επίσης και πιθανές μετατροπές που κάνατε στα αρχεία των δεδομένων.
    - Να ορίσετε σωστά την κάθε μεταβλητή.
    - Να χρησιμοποιήσετε την εντολή `summary` για να εξάγετε κάποιους βασικούς περιγραφικούς δείκτες.

67. Ο δείκτης FEV (Forced expiratory volume) χρησιμοποιείται για τη μέτρηση της λειτουργίας του πνεύμονα και μετράει τον όγκο του αέρα που εκπνέει ένα άτομο μετά από ένα δευτερόλεπτο συγκράτησης της αναπνοής. Να υποθέσετε ότι τα δεδομένα βρίσκονται data.frame που ονομάζεται «data1» το οποίο περιέχει τις ακόλουθες μετρήσεις:

    - id : Αύξων Αριθμός
    - age : Ηλικία
    - fev : Δείκτης FEV (σε λίτρα)
    - height: Ύψος (σε ίντσες)
    - gender: Φύλο (0=γυναίκα/1=άνδρας)
    - smoke: Κάπνισμα (0=μη τρέχων καπνιστής/1=τρέχων καπνιστής).



Να δώσετε τις εντολές με τις οποίες προκύπτει το παρακάτω διάγραμμα.

68. Ο δείκτης FEV (Forced Expiratory Volume - Όγκος Εκπνοής) είναι ένας μέτρο της αναπνευστικής λειτουργίας που καταγράφει τον όγκο του αέρα που εκπνέεται μετά από ένα δευτερόλεπτο εισπνοής. Τα δεδομένα αυτού του εργαστηρίου δίδονται στα αρχεία FEV.txt (απλό κειμένου) και FEV.sav (SPSS) και περιέχουν 684 μετρήσεις FEV από παιδιά ηλικίας 3-19 ετών που έγιναν το 1980 στα πλαίσια της μελέτης της Παιδικών Αναπνευστικών Παθήσεων (CRD study) στην περιοχή της Ανατολικής Βοστόνης. Τα δεδομένα αυτά είναι κομμάτι μιας διαχρονικής μελέτης με σκοπό την παρακολούθηση της αναπνευστικής λειτουργίας των παιδιών στο χρόνο (για λεπτομέρειες βλ. Tager *et al.*, 1979, *American Journal of Epidemiology*, 15-26). Στη διάθεση σας είναι οι ακόλουθες μεταβλητές:

- ID : Κωδικός ασθενή
- age : Ηλικία (σε χρόνια)
- FEV : Δείκτης Όγκου Εκπνοής (σε λίτρα)
- Height : Ύψος (σε ίντσες)
- sex : Φύλο (0=κορίτσι, 1=αγόρι)
- smoking : Κάπνισμα (0=δεν είναι καπνιστής, 1=είναι καπνιστής)

Να γίνουν οι ακόλουθες διαδικασίες στο S-PLUS:

(a) Να διαβαστούν τα δεδομένα από το αρχείο FEV.sav και να αποθηκευτούν σε ένα αρχείο με το όνομα fevspss.

(b) Να διαβαστούν τα δεδομένα από το αρχείο FEV.txt και να αποθηκευτούν σε ένα αρχείο με το όνομα fevtxt.

(c) Συγκρίνετε τα δύο αρχεία. Είναι τα ίδια; Να αποθηκεύσετε το σωστό αρχείο και να το αποθηκεύσετε στο αντικείμενο fev και μετά να σβήσετε τα αντικείμενα fevtxt και fevspss.

(d) Να μορφοποιήσετε το αρχείο fev (να ορίσετε ονόματα μεταβλητών αν δεν υπάρχουν και τα επίπεδα μεταβλητών κλπ.)

(e) Για κάθε μεταβλητή να βγάλετε τους βασικούς στατιστικούς δείκτες με την εντολή summary

(f) Για κάθε μεταβλητή να φτιάξετε τα κατάλληλα διαγράμματα για κάθε μεταβλητή.

(g) Να ελέγξετε γραφικά τη σχέση κάθε μεταβλητής με το FEV.

(h) Να κάνετε το ίδιο για κάθε φύλλο.

(i) Να δείτε γραφικά τις σχέσεις FEV-ύψος, FEV-ηλικία σημειώνοντας με διαφορετικό χρώμα το κάθε φύλο.

(j) Να δείτε γραφικά τις σχέσεις FEV-ύψος, FEV-ηλικία σημειώνοντας με διαφορετικό χρώμα τους καπνιστές και τους μη καπνιστές.

(k) Να φτιάξετε την κατανομή συχνοτήτων των ηλικιών (στα διαστήματα 0,4,8,12,16,20) και να την αποθηκεύσετε σε ένα data.frame με 3 στήλες (κάτω όριο τάξης, άνω όριο τάξης, συχνότητα).

69. Το 1975 δημοσιεύθηκαν τα αποτελέσματα μιας μεγάλης μελέτης που σκοπό είχε να διερευνήσει την επίδραση της παρατεταμένης έκθεσης στο μόλυβδο στην ψυχολογική και νευρολογική κατάσταση παιδιών (για λεπτομέρειες βλ. Landigram *et al.*, 1975, *Lancet*, 1, σελ. 708 - 715) . Τα δεδομένα είναι διαθέσιμα στο αρχείο `LEAD2.DAT`.

Περιληπτικά να αναφέρουμε ότι η μελέτη εξετάζει μια ομάδα παιδιών που διέμεναν κοντά σε ένα εργοστάσιο τήξεως μολύβδου στο El Paso του Texas. Στα παιδιά αυτά καταγράφηκε το επίπεδο μολύβδου στο αίμα τους το 1972 και 1973. Μια ομάδα έκθεσης αποτελείται από 46 παιδιά με επίπεδο μολύβδου $\geq$ 40 μg/ml στο αίμα το 1972 (ή σε μερικές περιπτώσεις το 1973). Αυτή ομάδα αναφέρεται ως GROUP = 2 ή 3. Μια ομάδα ελέγχου (μαρτύρων) αποτελείται από 78 παιδιά με επίπεδο μολύβδου $<$ 40 μg/ml στο αίμα και τις 2 χρονιές (1972 & 1973). Σημαντικές μεταβλητές για τη μελέτη είναι και ο αριθμός κτυπημάτων δακτύλου - καρπού (finger-wrist taps test) του κυρίως χεριού το οποίο αποτελεί ένα σύνηθες τεστ καλή νευρολογικής λειτουργίας όπως επίσης και η κλίμακα ευφυΐας του Wechsler .

- Να εισάγετε τα δεδομένα στο R από το αρχείο κειμένου `lead2.dat`

- Με βάση το αρχείο `lead.txt` όπου δίνεται τίτλος και επεξήγηση της κάθε μεταβλητής, να ορίσετε σωστά την κάθε μεταβλητή και τα missing values.
- Να επιλέξετε τις κατάλληλες μεταβλητές και να κάνετε από ένα Ιστόγραμμα, Διάγραμμα πλαισίου − απολήξεων, ραβδόγραμμα, κυκλικό διάγραμμα (πίτας), scatterplot και density plot.
- Συνοδέψτε τα παραπάνω διαγράμματα με τους αντίστοιχους περιγραφικούς δείκτες που ταιριάζουν στο κάθε διάγραμμα και μεταβλητή
- Να γράψετε ένα σχόλιο 1-3 γραμμές για κάθε διάγραμμα και τους αντίστοιχους περιγραφικούς δείκτες.
- Να επιλέξετε 5 μεταβλητές από το παραπάνω παράδειγμα για να κάνετε από ένα starplot, faceplot, correlation matrix plot. Να σχολιάστε με 2-4 γραμμές το κάθε διάγραμμα.
- Να υπολογίσετε το δείκτη συσχέτισης του Pearson και να σχολιάσετε ποιες μεταβλητές σχετίζονται μεταξύ τους
- Επιλέξτε ένα (κατάλληλο) ζευγάρι μεταβλητών και εφαρμόστε ανάλυση παλινδρόμησης. Απεικονίστε το διαγραμματικά. Ερμηνεύστε τα αποτελέσματα.
- Να ελέγξετε τη σχέση των ακόλουθων ζευγαριών μεταβλητών κάνοντας χρήση των κατάλληλων ελέγχων και εντολών R. Σχολιάστε τα αποτελέσματα που αφορούν

  α) AREA + GROUP,

  β) GROUP + IQF,

  γ) IQF + FINGER-WRIST TAPPING TEST και

  δ) SEX+ IQF

70. Τα δεδομένα αφορούν έναν υποθετικό πληθυσμό μιας χώρας που καλείται να πληρώσει φόρο ανάλογα με το εισόδημα και την οικογενειακή κατάσταση του νοικοκυριού. Τα δεδομένα είναι υποθετικά. Για κάθε νοικοκυριό υπάρχουν οι εξής πληροφορίες: ηλικία του άντρα και της γυναίκας, το εισόδημα του καθενός, το ύψος του ενοικίου που τυχόν πληρώνουν καθώς και οι ηλικίες των παιδιών που έχουν (αν έχουν). Ο φόρος καθορίζεται ως εξής

    Για κάθε έναν από τους δύο (σε μερικά νοικοκυριά δεν υπάρχουν 2 άτομα αλλά μόνο ένα) το συνολικό εισόδημα υπολογίζεται ως εξής:

    - Από το εισόδημα που δηλώνει το κάθε άτομο αφαιρείται το ενοίκιο αν υπάρχει (αν υπάρχουν και οι 2 αφαιρείται το μισό ενοίκιο από τον καθένα).
    - Αν το άτομο είναι πάνω από 67 χρονών αφαιρείται και ένα ακόμα ποσό της τάξης των 1000 ευρώ, ενώ για κάθε παιδί ηλικίας κάτω των 18 ετών αφαιρούνται 500 ευρώ. Τα ποσά για τα παιδιά αφαιρούνται και από τους 2 γονείς αν υπάρχουν.

- Στη συνέχεια ο φόρος υπολογίζεται με βάση κλίμακες που είναι

Μέχρι 13500 Ευρώ	0%
Από 13500 εως 16000 ευρώ	15%
Από 16000 εώς 18000 ευρώ	25%
Από 18000 εώς 22000 ευρώ	40%
Πάνω από 22000 Ευρώ	45%

.

Παράδειγμα: Έστω ότι στο νοικοκυριό υπάρχουν δύο ενήλικες, ο άντρας έχει εισόδημα 20000 και ηλικία 54 ετών, η γυναίκα 19000 και ηλικία 47 ετών αντίστοιχα. Επίσης έχουν δύο παιδιά ηλικίας 16 και 19 ετών και ενοίκιο 5000 ευρώ. Για τον άντρα το εισόδημα είναι 20000 μείον 500 ευρώ για το ανήλικο παιδί, μείον 2500 ευρώ από το μισό ενοίκιο. Δηλαδή εισόδημα 17000 ευρώ. Για αυτό το εισόδημα πληρώνει φόρο σύμφωνα με τον παρακάτω πίνακα

	Φόρος	Ποσό φόρου
Ποσό μέχρι τα 13500	0%	0
Ποσό από τις 13500 μέχρι τις 16000, δηλαδή ποσό 2500 ευρώ	15%	375
Ποσό από τις 16000 μέχρι και τις 17000, δηλαδή ποσό 1000 ευρώ	25%	250
Σύνολο		625

.

Δηλαδή συνολικά ο φόρος που θα πληρώσει ο άντρας είναι 625 Ευρώ. Ομοίως για τη γυναίκα το συνολικό εισόδημα είναι 16000 και άρα θα πληρώσει 375 ευρώ φόρο. Συνολικά το νοικοκυριό πληρώνει 1000 Ευρώ φόρο.

Σκοπός της εργασίας είναι:

- Να διαβάσετε τα δεδομένα σας και να υπολογίσετε για κάθε ένα νοικοκυριό το φόρο που του αναλογεί, και στη συνέχεια να περιγράψετε στατιστικά το φόρο που θα πληρώσουν τα νοικοκυριά ανάλογα με τη σύνθεση τους. Σε περίπτωση που υπάρχουν «λανθασμένα» δεδομένα να εξαιρέσετε τις παρατηρήσεις αυτές από την ανάλυση.
- Να περιγράψετε πλήρως και σε φυσική γλώσσα (όχι R) τον αλγόριθμο που θα χρησιμοποιήσετε.
- Να κατασκευάσετε μια συνάρτηση που να λαμβάνει ως input έναν πίνακα με τα δεδομένα αλλά και τις κλίμακες εισοδήματος και να επιστρέφει σαν αποτέλεσμα τη μέση τιμή, τη διακύμανση και το συντελεστή Gini για τα δεδομένα εισόδου.
- Ας υποθέσουμε πως κάποιος προτείνει το εξής. Να αλλάξουν οι κλίμακες σύμφωνα

με τον παρακάτω πίνακα

Μέχρι 15000 Ευρώ	0%
Από 15000 εως 18000 ευρώ	25%
Από 16000 εώς 20000 ευρώ	35%
Πάνω από 20000 Ευρώ	45%

,

ενώ όσοι έχουν πάνω από δύο ανήλικα παιδιά δεν πληρώνουν καθόλου φόρο ανεξάρτητα εισοδήματος.

Συγκρίνετε τα αποτελέσματα με τον καινούριο τρόπο με τα προηγούμενα. Τι αποτέλεσμα έχει στο συνολικό ποσό που θα πληρώσουν οι κάτοικοι της χώρας αυτής;

Κάθε γραμμή αντιστοιχεί σε ένα νοικοκυριό. Η σειρά των μεταβλητών είναι ακόλουθη

- Εισόδημα άντρα
- Εισόδημα γυναίκας
- Ενοίκιο
- Ηλικία άντρα
- Ηλικία γυναίκας
- Επτά αριθμοί που αντιστοιχούν στις ηλικίες των παιδιών• `NA` σημαίνει πως δεν υπάρχει το αντίστοιχο παιδί. Επομένως αν υπάρχουν δύο παιδιά τότε θα υπάρχουν δύο ηλικίες και πέντε τιμές `NA`. Υποθέτουμε πως δεν υπάρχει νοικοκυριό με περισσότερα από επτά παιδιά.