Η επιστήμη των δεδομένων μέσα από τη γλώσσα R

Βασίλειος Σ. Βερύκιος, Βασίλειος Καγκλής,
Ηλίας Κ. Σταυρόπουλος

Ελληνικά Ακαδημαϊκά Ηλεκτρονικά Συγγράμματα και Βοηθήματα
www.kallipos.gr

HEALLINK
Σύνδεσμος Ελληνικών Ακαδημαϊκών Βιβλιοθηκών

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

ΕΣΠΑ 2007-2013
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

ΒΑΣΙΛΕΙΟΣ Σ. ΒΕΡΥΚΙΟΣ
PhD, Αναπληρωτής Καθηγητής ΕΑΠ

ΒΑΣΙΛΕΙΟΣ ΚΑΓΚΛΗΣ
MSc, Μηχανικός Η/Υ & Πληροφορικής

ΗΛΙΑΣ Κ. ΣΤΑΥΡΟΠΟΥΛΟΣ
PhD, Μαθηματικός

Η επιστήμη των δεδομένων μέσα από τη γλώσσα R

Ελληνικά Ακαδημαϊκά Ηλεκτρονικά Συγγράμματα και Βοηθήματα
www.kallipos.gr

Η επιστήμη των δεδομένων μέσα από τη γλώσσα R

Συγγραφή

Βασίλειος Σ. Βερύκιος, PhD, Αναπληρωτής Καθηγητής ΕΑΠ

Βασίλειος Καγκλής, MSc, Μηχανικός Η/Υ & Πληροφορικής

Ηλίας Κ. Σταυρόπουλος, PhD, Μαθηματικός

Κριτικός αναγνώστης

Δημήτριος Καλλές

Συντελεστές έκδοσης

Γλωσσική επιμέλεια: Αδαμαντία Σπανακά

Γραφιστική επιμέλεια: Βαλεντίνα Κλ.Φιλώνη

ISBN: 978-960-603-394-0

Copyright © ΣΕΑΒ, 2015

Το παρόν έργο αδειοδοτείται υπό τους όρους της άδειας Creative Commons Αναφορά Δημιουργού - Μη Εμπορική Χρήση - Όχι Παράγωγα Έργα 3.0. Για να δείτε ένα αντίγραφο της άδειας αυτής επισκεφτείτε τον ιστότοπο

https://creativecommons.org/licenses/by-nc-nd/3.0/gr/

Σύνδεσμος Ελληνικών Ακαδημαϊκών Βιβλιοθηκών

Εθνικό Μετσόβιο Πολυτεχνείο

Ηρώων Πολυτεχνείου 9, 15780 Ζωγράφου

www.kallipos.gr

… στην Τζένη και στον Άγγελο
Β.Σ.Β.

… στη Φεβρωνία και στον Γεώργιο
Β.Κ.

… στη Βασιλική και στη Θεοδώρα
Η.Κ.Σ.

Περιεχόμενα

Πίνακας Συντομεύσεων – Ακρωνυμίων

Αγγλικά

Ακρωνύμιο	Επεξήγηση αγγλική	Επεξήγηση ελληνική
API	*Application Programming Interface*	Διεπαφή Προγραμματισμού Εφαρμογών
CDH	*Cloudera Distribution for Hadoop*	Διανομή *Cloudera* για το *Hadoop*
CPI	*Consumer Price Index*	Δείκτης Τιμών Καταναλωτή
cv	*Coefficient of variation*	Συντελεστής μεταβλητότητας
EC2	*Elastic Compute Cloud*	Υπηρεσία Υπολογιστικού Νέφους της *Amazon*
FIFO	*First In First Out*	Πρώτο μέσα πρώτο έξω
FP-tree	*Frequent-Pattern tree*	Δένδρο Συχνών Μοτίβων
HA	*High-Availability*	Υψηλή Διαθεσιμότητα
HDFS	*Hadoop Distributed File System*	*Κατανεμημένο* Σύστημα Αρχείων του *Hadoop*
IRQ	*Interquartile range*	Ενδοτεταρτημοριακό εύρος
NaN	*Not a Number*	Μη ορισμένη τιμή
OLAP	*OnLine Analytical Processing*	*Εργαλεία* εξερεύνησης δεδομένων
OLTP	*OnLine Transaction Processing*	*Ηλεκτρονική* διεκπεραίωση συναλλαγών
POSIX	*Portable Operating System Interface for Unix*	Φορητή Διεπαφή Λειτουργικού Συστήματος για *Unix*
RPC	*Remote Procedure Call*	Απομακρυσμένη κλήση συνάρτησης
SPOF	*Single Point of Failure*	*Μοναδικό* σημείο αποτυχίας
S3	*Simple Storage Service*	Απλή Υπηρεσία Αποθήκευσης
STONITH	*Shoot The Other Node In The Head*	Πυροβόλησε τον άλλο κόμβο στο κεφάλι

Ελληνικά

Ακρωνύμιο	Επεξήγηση
ΑΓΒΔ	Ανακάλυψη Γνώσης από Βάσεις Δεδομένων
ΔΑ	Διαβητική αμφιβληστροειδοπάθεια
ΕΔ	Εξόρυξη Δεδομένων
ΣΔΒΔ	Συστήματα Διαχείρισης Βάσεων Δεδομένων

Κεφάλαιο 1: Εισαγωγή στην Εξόρυξη Δεδομένων

Σύνοψη

Στόχος αυτού του κεφαλαίου είναι να φέρει τον αναγνώστη σε μια πρώτη επαφή με την Εξόρυξη Δεδομένων (Data Mining) και την Ανακάλυψη Γνώσης από Βάσεις Δεδομένων (Knowledge Discovery in Databases). Αρχικά, ορίζονται οι βασικές, εισαγωγικές έννοιες και παρουσιάζονται οι λόγοι, για τους οποίους αναπτύχθηκε και διαδόθηκε το συγκεκριμένο επιστημονικό πεδίο. Στη συνέχεια, δίνονται πρακτικά παραδείγματα και αντιπαραδείγματα της Εξόρυξης Δεδομένων. Επιπλέον, αναφέρονται οι βασικότεροι τομείς πάνω στους οποίους βασίζεται η Εξόρυξη Δεδομένων. Τέλος, παρουσιάζεται η γλώσσα προγραμματισμού R και η γενική φιλοσοφία που τη διέπει. Δίνονται οδηγίες εγκατάστασης των απαραίτητων εργαλείων που θα χρησιμοποιηθούν, παρουσιάζεται ο τρόπος αναζήτησης βοήθειας από το ηλεκτρονικό εγχειρίδιο της γλώσσας, και ορίζονται οι βασικοί τύποι και συναρτήσεις της, καθώς και οι λειτουργίες τους.

Προαπαιτούμενη γνώση

Για την κατανόηση αυτού του κεφαλαίου δεν απαιτείται κάποια ιδιαίτερη προηγούμενη γνώση.

Εισαγωγή στην Εξόρυξη Δεδομένων

Η εξέλιξη της τεχνολογίας έδωσε τη δυνατότητα εξάπλωσης του Internet. Με το πέρασμα του χρόνου, η πρόσβαση στο Internet έγινε προσιτή σε ολοένα και περισσότερους ανθρώπους. Αυτό με τη σειρά του οδήγησε στο να αναπτυχθούν περισσότεροι ιστότοποι και να χρησιμοποιηθούν βάσεις δεδομένων για την αποθήκευση των δεδομένων. Με τη δημιουργία εμπορικών και κοινωνικών ιστοσελίδων υπήρξαν τα πρώτα άλματα στις απαιτήσεις και ανάγκες για αποθήκευση και διαχείριση μεγάλου όγκου δεδομένων.

Σήμερα, το πλήθος των διαθέσιμων δεδομένων είναι τεράστιο και αυξάνεται εκθετικά κάθε μέρα. Η μείωση στο κόστος συλλογής και της δυσκολίας στη συλλογή και αποθήκευση των δεδομένων συνετέλεσε σημαντικά στην ανάπτυξη του πεδίου αυτού.

Ο τεράστιος όγκος δεδομένων, που συσσωρεύεται στις βάσεις δεδομένων και στις αποθήκες δεδομένων (data warehouses), δεν μπορεί να αξιοποιηθεί όπως είναι. Πρέπει αρχικά να γίνουν κάποιες ενέργειες για να δομηθούν κατάλληλα τα δεδομένα, ώστε στη συνέχεια να μπορούν να αξιοποιηθούν. Στο κεφάλαιο αυτό θα δούμε ποια είναι τα θεμελιώδη στάδια, προκειμένου να μπορεί να εξαχθεί από τα δεδομένα χρήσιμη και αξιοποιήσιμη πληροφορία.

1.1 Η Επιστήμη των Δεδομένων

Η Επιστήμη των Δεδομένων (Data Science), είναι ένας καινούριος όρος, ο οποίος ήρθε να αντικαταστήσει προγενέστερους όρους, όπως Ανακάλυψη Γνώσης από Βάσεις Δεδομένων (*Knowledge Discovery in Database*) ή Εξόρυξη Δεδομένων (*Data Mining*).

Και οι τρεις αυτοί όροι χρησιμοποιούνται σχεδόν εναλλακτικά, για να περιγράψουν μία ημι-αυτοματοποιημένη διαδικασία, σκοπός της οποίας είναι να αναλύσει έναν μεγάλο όγκο δεδομένων που αφορούν ένα συγκεκριμένο πρόβλημα, συνήθως εμπορικού ή επιστημονικού ενδιαφέροντος, για την παραγωγή προτύπων (patterns), όπως συνηθίζεται να λέμε σε ορισμένους τομείς, όπως στη Στατιστική, στη Μηχανική Μάθηση (*Machine Learning*) και στην Αναγνώριση Προτύπων (*Pattern Recognition*).

Τα πρότυπα αυτά, τα οποία συναντάμε σε διάφορες μορφές, όπως συσχετίσεις, ανωμαλίες, συστάδες, κλάσεις κ.λπ., αποτελούν δομές ή περιστατικά, που εμφανίζονται στα δεδομένα και έχουν κάποια ιδιαίτερη σημασία από στατιστικής πλευράς.

Ένα από τα σημαντικότερα θέματα από πλευράς ουσίας στην Επιστήμη των Δεδομένων αφορά τόσο την εύρεση ή αλλιώς ανακάλυψη (η λέξη ανακάλυψη εμπεριέχει σε μεγάλο βαθμό το γεγονός ότι τα πρότυπα αυτά δεν ήταν αναμενόμενα εκ των προτέρων), όσο και τον χαρακτηρισμό αυτών των προτύπων. Πιο συγκεκριμένα, ένα πρότυπο αναφέρεται και ως μία διευθέτηση ή αλλιώς διάταξη, στην οποία θεωρείται ότι υπάρχει κάποιου είδους οργάνωση της υποκείμενης δομής. Τα πρότυπα αυτά, τα οποία είναι γνωστά και ως μοντέλα, ανιχνεύονται κατά κύριο λόγο με τη χρήση μετρήσιμων χαρακτηριστικών γνωρισμάτων ή ιδιοτήτων, τα οποία

έχουν εξαχθεί από τα δεδομένα.

Η Επιστήμη των Δεδομένων είναι ουσιαστικά μία καινούρια επιστήμη, η οποία άρχισε να εμφανίζεται σταδιακά, ξεκινώντας από τα τέλη της δεκαετίας του 1980. Την εποχή εκείνη μεσουρανούσαν τα σχεσιακά συστήματα βάσεων δεδομένων, τα οποία εξυπηρετούσαν τις αποθηκευτικού τύπου ανάγκες των δεδομένων-για τις επιχειρήσεις και τους οργανισμούς-με στόχο την καλύτερη οργάνωση και διαχείρισή τους, έτσι ώστε να ικανοποιούνται ταχύτερα μαζικά ερωτήματα, που αφορούσαν την καθημερινή τους λειτουργία.

Τέτοιου είδους Συστήματα Διαχείρισης Βάσεων Δεδομένων (ΣΔΒΔ) ακολουθούσαν το λεγόμενο *OLTP* (*OnLine Transaction Processing*) μοντέλο, το οποίο αφορά στην ηλεκτρονική διεκπεραίωση συναλλαγών ή αλλιώς δοσοληψιών. Τα εργαλεία αυτά επέτρεπαν ουσιαστικά στον χρήστη να βρει απαντήσεις (επιβεβαίωση) σε ερωτήματα (ειδικού τύπου πρότυπα), τα οποία ήδη ο χρήστης γνώριζε ή να δημιουργήσει κάποιες αναφορές.

Η ανάγκη για την καλύτερη εκμετάλλευση των δεδομένων που είχαν δημιουργηθεί μέσω των συστημάτων αυτών – συστήματα τα οποία κάλυπταν καθημερινά τη λειτουργία μιας εταιρίας – οδήγησε στην ανάπτυξη εργαλείων τύπου *OLAP* (*OnLine Analytical Processing*). Με αυτά εργαλεία τύπου OLAP μπορούσαν να απαντηθούν ερωτήματα και αναφορές πιο προηγμένης μορφής, που επέτρεπαν μεγάλες και πολυδιάστατες Βάσεις Δεδομένων να ερωτηθούν με μεγάλη ταχύτητα, ενώ παράλληλα προσέφεραν και οπτικοποίηση των αποτελεσμάτων.

Τα εργαλεία τύπου OLAP θα μπορούσαν να χαρακτηριστούν και ως εργαλεία εξερεύνησης δεδομένων, οδηγούμενα από την οπτικοποίηση των αποτελεσμάτων. Αυτού του τύπου τα εργαλεία επέτρεπαν στους χρήστες (διευθυντές πωλήσεων, προϊσταμένους τμημάτων προώθησης προϊόντων κ.λπ.) να βρίσκουν και να ανακαλύπτουν καινούρια πρότυπα, με τη διαφορά ότι η ανακάλυψη αυτή γινόταν από τον χρήστη.

Για παράδειγμα, ένας χρήστης θα μπορούσε να υποβάλει ερωτήματα για το ύψος των πωλήσεων των καταστημάτων μιας αλυσίδας καταστημάτων σε μία πόλη, έτσι ώστε να βρει τα καταστήματα με τις χαμηλότερες πωλήσεις.

Η αυτοματοποίηση της διαδικασίας ανακάλυψης προτύπων ή αλλιώς γνώσης έλαβε χώρα μέσω τεχνικών μεθοδολογιών και εργαλείων, τα οποία αναπτύχθηκαν στο πλαίσιο της Επιστήμης των Δεδομένων. Μέσω αυτών των λύσεων η ανακάλυψη νέων προτύπων ήταν οδηγούμενη (ή ίσως καλύτερα υποβοηθούμενη) από τον τελικό στόχο. Για παράδειγμα, αντί ο ίδιος ο χρήστης να ζητάει μία αναφορά για να δει ποιο είναι το κατάστημα με τις λιγότερες πωλήσεις τον προηγούμενο μήνα, θα μπορούσε να ζητάει από το σύστημα ενδιαφέροντα, από στατιστικής πλευράς, γεγονότα που αφορούν γενικότερα στις πωλήσεις των καταστημάτων.

Η μεγάλη άνθιση στον τομέα της Επιστήμης των Δεδομένων έλαβε χώρα σταδιακά και ήταν άμεσα εξαρτημένη από τη δυνατότητα που δόθηκε για συλλογή και καταγραφή τεράστιων ποσοτήτων δεδομένων, διαφορετικών μορφών και τύπων, μέσω της ανάπτυξης γρήγορων δικτυακών υποδομών, πάνω στις οποίες μπορούσαν να υποστηριχτούν αξιόπιστες εμπορικές εφαρμογές.

Στην κατηγορία των εταιριών που πρωτοστάτησαν σε αυτή τη νέα τάξη πραγμάτων ήταν και η Amazon, η οποία ξεκίνησε από την ηλεκτρονική πώληση βιβλίων και άλλων στοιχείων, δημιουργώντας στη συνέχεια ένα πολύ φιλικό προς τον χρήστη σύστημα συστάσεων. Το σύστημα αυτό χτιζόταν και ρυθμιζόταν με βάση τις αλληλεπιδράσεις των πελατών της, χρησιμοποιώντας μία τεχνική γνωστή ως Συνεργατικό Φιλτράρισμα (*Collaborative Filtering*). Το σύστημα αυτό αποτέλεσε τη βάση των Συστημάτων Συστάσεων (*Recommender Systems*).

Η άνευ όρων παραγωγή δεδομένων σε εικοσιτετράωρη βάση καλύπτει μια τεράστια γκάμα ανθρώπινων δραστηριοτήτων και όχι μόνο, όπως είναι τα δεδομένα από το καλάθι αγορών, τον ιατρικό φάκελο του ασθενούς, τις συζητήσεις ή και ανακοινώσεις στα κοινωνικά μέσα δικτύωσης, τις τραπεζικές ή και χρηματιστηριακές συναλλαγές, τα ίχνη κινούμενων οχημάτων, τα δεδομένα αισθητήρων από κινητήρες αεροσκαφών, η καταγραφή συνομιλιών σε κέντρα εξυπηρέτησης πελατών κ.λπ. Τα δεδομένα αυτά διαφέρουν πάρα πολύ μεταξύ τους τόσο σε μορφή (εικόνα, βίντεο, κείμενο, πολυδιάστατα ή πραγματικού χρόνου δεδομένα, ακολουθίες DNA και άλλα πολλά) όσο και στην ταχύτητα συλλογής. Εάν, μάλιστα, δεν υποστούν άμεση ανάλυση, ίσως να είναι ιδιαίτερα δύσκολο να αποθηκευτούν ή να τα επεξεργαστούν οι άνθρωποι, δημιουργώντας έτσι μία καινούρια ερευνητική δράση, γνωστή με τον όρο Μεγάλα Δεδομένα (*Big Data*). Η Επιστήμη των Δεδομένων στοχεύει σε αυτή τη φάση να καλύψει τις ανάγκες που δημιουργούνται από αυτόν τον νέο τομέα και να προσφέρει λύσεις για την κλιμακούμενη και αποτελεσματική επεξεργασία out-of core (εκτός μνήμης ή εξωτερικής μνήμης) δεδομένων.

Τεχνικές και εργαλεία που χρησιμοποιούνται γι' αυτόν τον σκοπό έχουν ήδη αρχίσει να μορφοποιούνται τόσο στα εργαστήρια όσο και στην αγορά και αποτυπώνονται με όρους όπως Map-Reduce, Hadoop, Hive, MongoDB, GraphPD κ.λπ. Τα δύο τελευταία συστήματα αναφέρονται σε μία ερευνητική περιοχή, που είναι

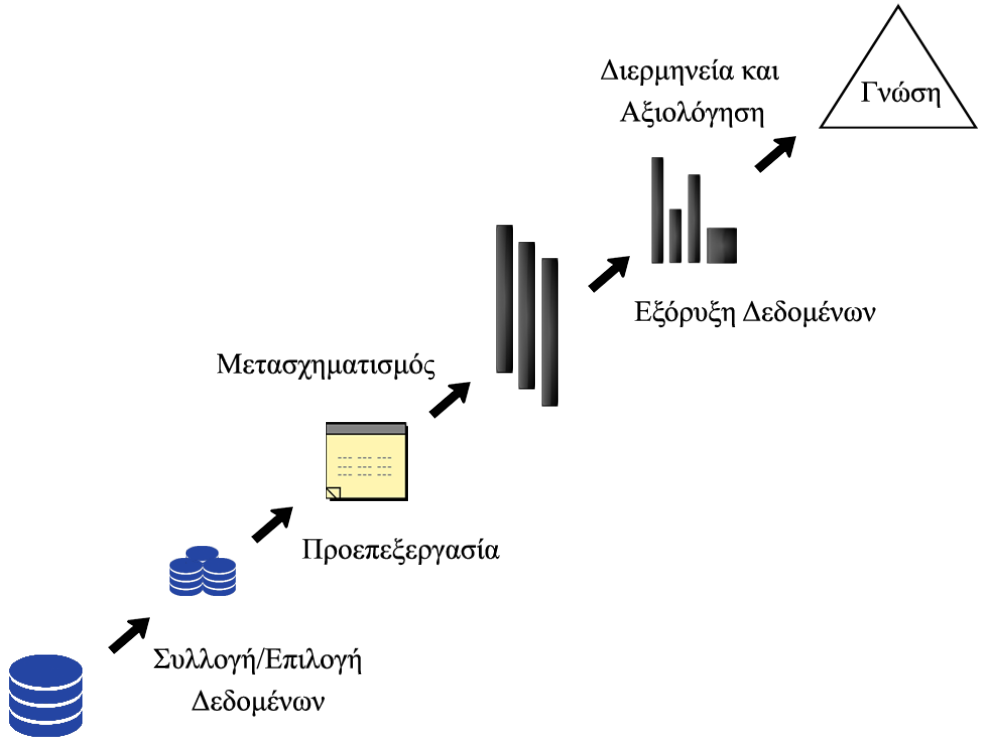
γνωστή και ως NoSQL.

Οι δύο πρωταρχικοί στόχοι στην πρακτική της Επιστήμης των Δεδομένων είναι η δημιουργία μοντέλων, τα οποία να μπορούν να χρησιμοποιηθούν τόσο για την πρόβλεψη, όσο και για την περιγραφή των δεδομένων. Η πρόβλεψη αφορά στη χρήση κάποιων μεταβλητών ή πεδίων μίας βάση δεδομένων, μέσω των τιμών των οποίων μπορεί να εκτιμηθεί η άγνωστη ή μελλοντική τιμή ενός άλλου γνωρίσματος. Η περιγραφή (σε μορφή σύνοψης ή περιληπτικής παρουσίασης) των δεδομένων εστιάζει στην εύρεση κατανοητών από τον άνθρωπο προτύπων, τα οποία περιγράφουν τα δεδομένα, όπως, δηλαδή, γίνεται κατά την εύρεση συστάδων ή ομάδων αντικειμένων με παρόμοια χαρακτηριστικά.

1.2 Ανακάλυψη Γνώσης από Βάσεις Δεδομένων

Η Ανακάλυψη Γνώσης από Βάσεις Δεδομένων (ΑΓΒΔ) αποτελείται από συγκεκριμένα στάδια. Πρόκειται για την αποκάλυψη ή παραγωγή λειτουργικής γνώσης μέσα από την ανάλυση των δεδομένων. Αναφέρεται σε ολόκληρη τη διαδικασία, από τη συλλογή δεδομένων μέχρι την αξιοποίηση των αποτελεσμάτων σε πιο πρακτικό επίπεδο. Τα βασικά στάδια της ΑΓΒΔ (Εικόνα 1.1) είναι:

1. Συλλογή Δεδομένων (Data Collection)
2. Προεπεξεργασία Δεδομένων (Preprocessing)
3. Μετασχηματισμός Δεδομένων (Transformation)
4. Εξόρυξη Δεδομένων (Data Mining)
5. Διερμηνεία και Αξιολόγηση (Interpretation/Evaluation)



Εικόνα *1.1* *Βασικά στάδια Ανακάλυψης Γνώσης από Βάσεις Δεδομένων.*

Ο όρος Ανακάλυψη Γνώσης από Βάσεις Δεδομένων συχνά ταυτίζεται με τον όρο Εξόρυξη Δεδομένων, που στην πραγματικότητα αποτελεί ένα μόνο επιμέρους βήμα της. Ο βασικός στόχος της Εξόρυξης Δεδομένων (ΕΔ) είναι η εξαγωγή μη τετριμμένης, προηγούμενα άγνωστης και πιθανά χρήσιμης πληροφορίας ή προτύπων από το σύνολο των δεδομένων. Ο όρος χρησιμοποιείται μάλλον καταχρηστικά αφού στην ουσία δεν γίνεται καμία εξαγωγή δεδομένων. Αντίθετα, χρησιμοποιούνται τα προεπεξεργασμένα και (ενδεχομένως) μετασχηματισμένα δεδομένα για την εξαγωγή χρήσιμης πληροφορίας, η οποία αξιοποιείται για την απόκτηση γνώσης σε σχέση με κάποιο πρόβλημα. Ακολουθεί σύντομη περιγραφή για το κάθε στάδιο της ΑΓΒΔ.

1.2.1 Συλλογή Δεδομένων

Το πρώτο βήμα της ΑΓΒΔ είναι η συλλογή και η αποθήκευση των δεδομένων. Η συλλογή των δεδομένων συνήθως γίνεται είτε αυτόματα, π.χ. με χρήση αισθητήρων, είτε μη αυτόματα, π.χ. με χρήση ερωτηματολογίων. Δυσλειτουργία στους αισθητήρες ή αδυναμία απάντησης κάποιας ερώτησης στα ερωτηματολόγια μπορεί να οδηγήσει σε θορυβώδη ή ελλιπή δεδομένα. Τα συγκεκριμένα προβλήματα, που ενδεχομένως να προκύψουν κατά τη συλλογή δεδομένων, αναλαμβάνει να τα αντιμετωπίσει το επόμενο στάδιο.

1.2.2 Προεπεξεργασία Δεδομένων

Το δεύτερο και πιο σημαντικό στάδιο της ΑΓΒΔ είναι η προεπεξεργασία του συνόλου δεδομένων, η οποία γίνεται με στόχο τον καθαρισμό τους, δηλαδή την τακτοποίηση εσφαλμένων, προβληματικών ή ελλιπόντων δεδομένων. Μπορεί να απαιτήσει έως και το 60% της συνολικής προσπάθειας και αυτό διότι, αν τα δεδομένα δεν είναι «καθαρά» και στην κατάλληλη μορφή, δεν έχει νόημα να μιλάμε για ποιότητα αποτελεσμάτων. Στο Κεφάλαιο 3 θα εξετάσουμε αναλυτικά από ποιες διεργασίες συνίσταται η προεπεξεργασία των δεδομένων και πότε χρησιμοποιείται η κάθε μια.

1.2.3 Μετασχηματισμός Δεδομένων

Ο μετασχηματισμός των δεδομένων αποτελεί το τρίτο στάδιο της ΑΓΒΔ. Ουσιαστικά, πρόκειται για τη μετατροπή των δεδομένων κάτω από ένα κοινό πλαίσιο, για επεξεργασία. Χρησιμοποιείται κυρίως για την εξομάλυνση των δεδομένων και απομάκρυνση θορύβου, για τη συνάθροιση των δεδομένων, δηλαδή για την παραγωγή σύνοψης τους, για την κανονικοποίηση τους, δηλαδή την κλιμάκωση των χαρακτηριστικών του συνόλου δεδομένων σε ένα συγκεκριμένο και περιορισμένο εύρος τιμών, ή τέλος για τη δημιουργία νέων χαρακτηριστικών από τα ήδη υπάρχοντα. Ειδικές μορφές μετασχηματισμού αποτελούν η διακριτοποίηση και η συμπίεση.

1.2.4. Εξόρυξη Δεδομένων

Σε αυτό το στάδιο της ΑΓΒΔ εφαρμόζεται κάποιος αλγόριθμος για την παραγωγή ενός μοντέλου. Έχοντας καθαρίσει και μετασχηματίσει τα δεδομένα, είναι έτοιμα να χρησιμοποιηθούν από κάποιον αλγόριθμο, ώστε να δημιουργηθεί κάποιο μοντέλο, συνήθως κατηγοριοποίησης ή πρόβλεψης. Θέλουμε να χρησιμοποιήσουμε το μοντέλο αυτό, το οποίο δημιουργήθηκε με βάση κάποια γνωστά δεδομένα, έτσι ώστε να μπορεί να μας δώσει απάντηση για την τιμή ενός χαρακτηριστικού-μεταβλητής στόχου για νέα, άγνωστα δεδομένα.

1.2.5 Διερμηνεία και Αξιολόγηση

Στο τελευταίο στάδιο της ΑΓΒΔ γίνεται η διερμηνεία και η αξιολόγηση των αποτελεσμάτων (όχι του μοντέλου) που παρήχθησαν από την όλη διαδικασία.

1.3 Τύποι Μοντέλων

Τα μοντέλα που παράγονται από το στάδιο της Εξόρυξης Δεδομένων διακρίνονται σε δυο βασικούς τύπους: τα μοντέλα πρόβλεψης (predictive) και τα περιγραφικά μοντέλα (descriptive). Στόχος ενός μοντέλου πρόβλεψης είναι να προβλέψει τιμές για ένα συγκεκριμένο χαρακτηριστικό που παρουσιάζει ενδιαφέρον και που πιθανώς βασίζεται στη συμπεριφορά άλλων χαρακτηριστικών. Για παράδειγμα, η πρόβλεψη μπορεί να βασίζεται στη χρονολογική κατάταξη των δεδομένων.

Ένα περιγραφικό μοντέλο βρίσκει πρότυπα (patterns) ή σχέσεις (relations) που ενυπάρχουν στα δεδομένα και μελετά τις ιδιότητες τους, ώστε να δοθεί μια αιτιολόγηση της συμπεριφοράς τους.

1.4 Παραδείγματα και Αντιπαραδείγματα

Αρκετά άτομα δυσκολεύονται να κατανοήσουν και να διαχωρίσουν τις έννοιες ΑΓΒΔ και ΕΔ. Για τον λόγο αυτό κρίνεται σκόπιμο να δοθούν μερικά πρακτικά παραδείγματα και αντιπαραδείγματα, ώστε να ξεκαθαριστεί το τι είναι ΕΔ και τι όχι. Μερικά παραδείγματα της Εξόρυξης Δεδομένων είναι τα ακόλουθα:

- Ο Bill Clinton δήλωσε πως μετά το τρομοκρατικό χτύπημα στις 11 Σεπτεμβρίου 2011, έπειτα από εξέταση πολλών βάσεων δεδομένων, πράκτορες του FBI ανακάλυψαν ότι 5 από τους αυτουργούς υπήρχαν καταχωρημένοι σε αυτές. Ένας από αυτούς κατείχε 30 πιστωτικές κάρτες με χρεωστικό υπόλοιπο της κατηγορία των $250,000 ενώ βρισκόταν στη χώρα για λιγότερο από δυο χρόνια.
- Εταιρίες τηλεπικοινωνιών επιβραβεύουν όχι μόνο όσους ξοδεύουν πολλά, αλλά και κάποιους από τους οικονομικότερους συνδρομητές. Οι λεγόμενοι «καθοδηγητές» συχνά πείθουν φίλους, συγγενικά πρόσωπα ή συνεργάτες να τους «ακολουθήσουν», όταν αλλάζουν πάροχο. Επομένως, στόχος είναι να βρεθούν αυτοί οι πελάτες και να μείνουν, προσφέροντας τους δελεαστικές εκπτώσεις και πακέτα προσφορών.
- Χρησιμοποιώντας τις καταγεγραμμένες θερμοκρασίες κατά τη θερινή σεζόν των προηγούμενων 15 ετών, γίνεται προσπάθεια πρόβλεψης του πώς θα κυμανθούν οι θερμοκρασίες το καλοκαίρι των επόμενων χρόνων.
- Τα τελευταία 7 χρόνια στο Σικάγο συλλέγονται δεδομένα ανά περιοχές για το πλήθος των κουνουπιών και τον ιό του δυτικού Νείλου. Χρησιμοποιώντας δεδομένα, όπως η τοποθεσία, το πλήθος των κουνουπιών και οι καιρικές συνθήκες, γίνεται προσπάθεια να προβλεφθούν οι περιοχές, στις οποίες είναι πιο πιθανό να υπάρξουν μολυσμένα κουνούπια, ώστε να ληφθούν τα απαραίτητα μέτρα προστασίας.

Εξόρυξη δεδομένων δεν είναι η απλή επεξεργασία ερωτημάτων, ούτε τα έμπειρα συστήματα ή τα μικρής κλίμακας στατιστικά προγράμματα. Μερικά αντιπαραδείγματα είναι τα ακόλουθα:

- Εύρεση αριθμού τηλεφώνου στον τηλεφωνικό κατάλογο.
- Εύρεση πληροφοριών για τον Αμαζόνιο στο Internet.
- Μέσος όρος βαθμολογίας μαθημάτων.
- Αναζήτηση του ιατρικού μητρώου ενός ασθενούς με κάποια ασθένεια, για την ανάλυση του ιστορικού του στο μητρώο.

1.5 Κατηγοριοποίηση Μεθόδων Εξόρυξης

Υπάρχει μια μεγάλη ποικιλία μεθόδων εξόρυξης δεδομένων. Ανάλογα με το είδος των δεδομένων και το είδος της γνώσης που εξάγεται, αυτές κατηγοριοποιούνται σε διαφορετική κατηγορία. Μερικές βασικές μέθοδοι της Εξόρυξης Δεδομένων παρουσιάζονται παρακάτω. Να σημειώσουμε εδώ ότι στην Επιστήμη των Δεδομένων η ειδοποιός διαφορά έγκειται στο σημείο της εκπαίδευσης στο οποίο γίνεται με τη χρήση δεδομένων, σε αντίθεση με άλλες μορφές εκπαίδευσης, όπου χρησιμοποιείται ένας δάσκαλος ή κάποιος ειδικός και η εκπαίδευση γίνεται με μεταφορά γνώσης από τον έναν στον άλλον.

1.5.1 Κατηγοριοποίηση

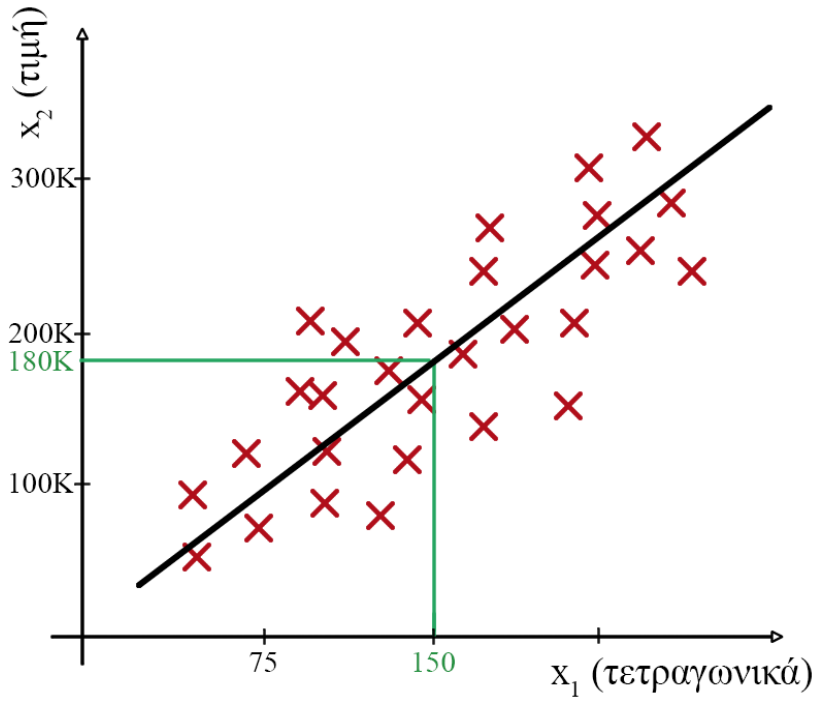
Πρόκειται για μια προγνωστική μέθοδο. Στόχος είναι η δημιουργία ενός μοντέλου – κατηγοριοποιητή (classifier) με βάση τα υπάρχοντα δεδομένα Ουσιαστικά, είναι η μάθηση μιας συνάρτησης, η οποία απεικονίζει ένα αντικείμενο (συνήθως αναπαρίσταται ως ένα διάνυσμα τιμών για τις χαρακτηριστικές του ιδιότητες) σε μία τιμή μιας κατηγορικής μεταβλητής, η οποία η οποία είναι γνωστή και ως κλάση (ή κατηγορία). Η μάθηση (learning), μία έννοια που ήδη αναφέρθηκε, αποτελεί συμπεριφορά των ευφυών συστημάτων, τα οποία μελετώνται από τομείς, όπως η Μηχανική Μάθηση ή η Τεχνητή Νοημοσύνη. Εξαιτίας αυτού, όλες αυτές οι περιοχές μελετούν παρόμοια προβλήματα, χωρίς αυτό να σημαίνει ότι δεν υπάρχουν και ξεχωριστά αντικείμενα, που μελετώνται από την κάθε μία ξεχωριστά.

Η κατηγοριοποίηση συχνά συγχέεται με το γενικό όρο της πρόβλεψης. Στην κατηγοριοποίηση, το αποτέλεσμα που θέλουμε να προβλέψουμε είναι η κλάση των δειγμάτων. Η κλάση μπορεί να πάρει διακριτές τιμές από ένα πεπερασμένο σύνολο. Αντίθετα, κατά την πρόβλεψη με χρήση τεχνικών όπως η παλινδρόμηση, η

μεταβλητή-στόχος μπορεί να είναι οποιοσδήποτε πραγματικός αριθμός.

1.5.2 Παλινδρόμηση

Μια σχετική διαδικασία με την κατηγοριοποίηση είναι η παλινδρόμηση (*regression*), στόχος της οποίας είναι η μάθηση ή αλλιώς η εκπαίδευση (*training*) μιας συνάρτησης, η οποία απεικονίζει ένα αντικείμενο σε μία πραγματική μεταβλητή. Πρόκειται για μια, επίσης, προγνωστική μέθοδο. Στόχος είναι με βάση κάποιες ανεξάρτητες μεταβλητές (independent variables) να προβλεφθούν οι τιμές μιας εξαρτημένης μεταβλητής (dependent variable).



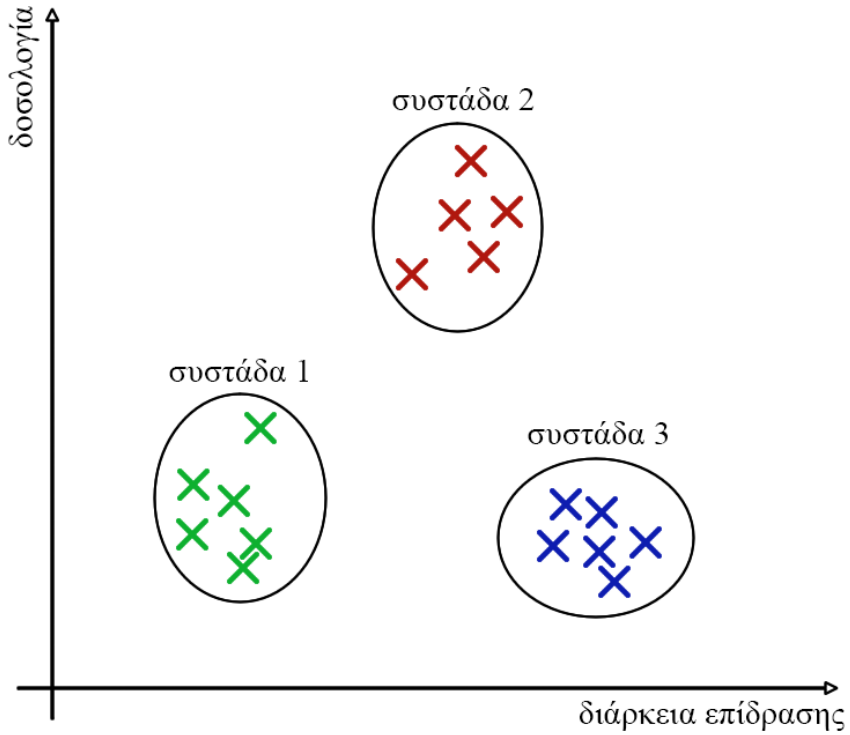
Εικόνα *1.2* *Παράδειγμα γραμμικής παλινδρόμησης.*

Στην Εικόνα 1.2 παρουσιάζουμε ένα απλό παράδειγμα γραμμικής παλινδρόμησης. Οι μεταβλητές είναι τα τετραγωνικά ενός σπιτιού και η τιμή πώλησης του σε χιλιάδες Ευρώ. Η γραμμική παλινδρόμηση προσαρμόζει μια ευθεία στα δείγματα του συνόλου δεδομένων, τα οποία σηματοδοτούνται με κόκκινο Χ. Η προσαρμογή γίνεται με βάση μια συνάρτηση απόστασης ή συνάρτηση κόστους, την τιμή της οποία θέλουμε να ελαχιστοποιήσουμε. Έχοντας τη βέλτιστη ευθεία, δηλαδή την ευθεία που ελαχιστοποιεί την τιμή της συνάρτησης κόστους, μπορούμε να δώσουμε μια προσεγγιστικά καλή απάντηση σε ερωτήματα της μορφής: «Σε τι τιμές πωλούνται σπίτια των 150 τετραγωνικών;».

1.5.3 Συσταδοποίηση

Η συσταδοποίηση (clustering) είναι μια περιγραφική μέθοδος. Έχοντας ένα σύνολο δεδομένων, στόχος της συσταδοποίησης είναι η δημιουργία συστάδων (clusters), δηλαδή ομάδων, οι οποίες θα περιέχουν όμοια ή παρεμφερή δείγματα. Ουσιαστικά αναζητείται ένα πεπερασμένο σύνολο κατηγοριών ή συστάδων, για να περιγράψει τα δεδομένα. Οι κατηγορίες μπορεί να είναι αμοιβαία αποκλειόμενες και εξαντλητικές ή να έχουν μία πιο σύνθετη αναπαράσταση, όπως για παράδειγμα ιεραρχικές και επικαλυπτόμενες.

Στο παρακάτω παράδειγμα (Εικόνα 1.3) βλέπουμε το αποτέλεσμα συσταδοποίησης φαρμακευτικών δεδομένων. Έχουν δημιουργηθεί 3 συστάδες με βάση τα χαρακτηριστικά «δοσολογία» και «διάρκεια επίδρασης».



Εικόνα *1.3* *Παράδειγμα συσταδοποίησης.*

1.5.4 Εξαγωγή και Ανάλυση Συσχετίσεων

Η εξαγωγή κανόνων συσχέτισης (Mining Association Rules) θεωρείται μια από τις σημαντικότερες διεργασίες εξόρυξης δεδομένων. Έχει προσελκύσει ιδιαίτερο ενδιαφέρον, καθώς οι κανόνες συσχέτισης παρέχουν έναν συνοπτικό τρόπο για να εκφραστούν οι ενδεχομένως χρήσιμες πληροφορίες που γίνονται εύκολα κατανοητές από τους τελικούς χρήστες. Οι κανόνες συσχέτισης ανακαλύπτουν κρυμμένες «συσχετίσεις» μεταξύ των γνωρισμάτων ενός συνόλου των δεδομένων. Αυτοί οι συσχετισμοί παρουσιάζονται στη μορφή Α → Β, όπου τα Α και Β αποτελούν σύνολα που αναφέρονται στα χαρακτηριστικά του συνόλου δεδομένων που αναλύουμε. Δεδομένου ενός συνόλου από δεδομένα, ένας κανόνας συσχέτισης Α →Β προβλέπει την εμφάνιση των χαρακτηριστικών του συνόλου Β δεδομένης της εμφάνισης των χαρακτηριστικών του συνόλου Α.

Κλασικό πεδίο εφαρμογής των κανόνων συσχέτισης είναι η ανάλυση του καλαθιού αγοράς (market basket), όπου τα δεδομένα αφορούν συναλλαγές πελατών σε ένα παντοπωλείο. Οι συναλλαγές μπορεί να είναι, για παράδειγμα: {ψωμί, γάλα}, {ψωμί, πάνες, μπύρα, αυγά}, {γάλα, πάνες, μπύρα, σόδα}, {ψωμί, γάλα, πάνες, μπύρα} και {ψωμί, γάλα, πάνες, σόδα}, και κάποιοι κανόνες συσχέτισης σε αυτές είναι {Πάνες} → {μπύρα}, {μπύρα, ψωμί}→{γάλα}, {γάλα, ψωμί}→{αυγά, σόδα}. Ο τελευταίος κανόνας, για παράδειγμα, φανερώνει ότι είναι πολύ πιθανό όποιος αγοράζει γάλα και ψωμί να αγοράσει, επίσης, αυγά και σόδα.

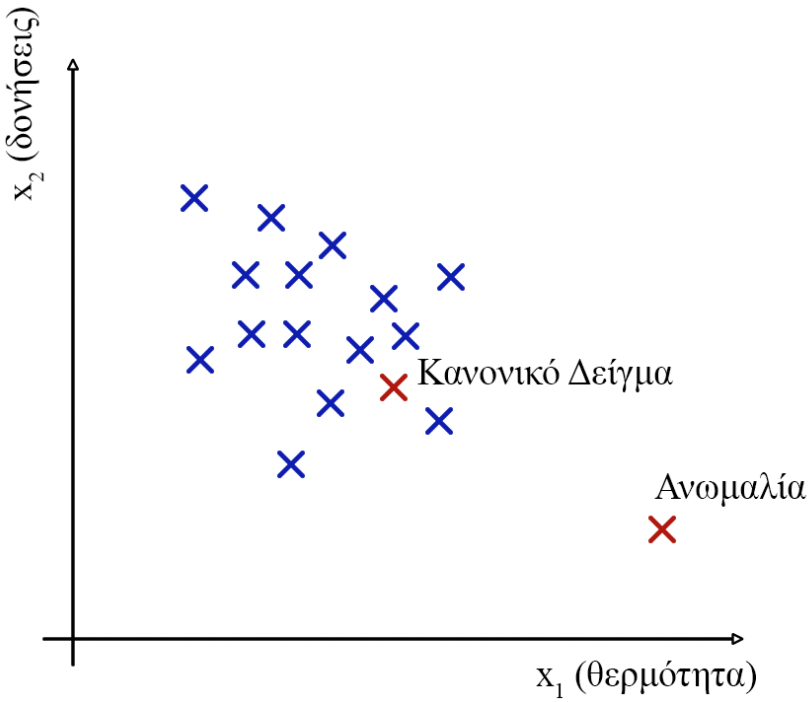
Εξάγοντας χρήσιμα συμπεράσματα μέσω των κανόνων συσχέτισης, το τμήμα προώθησης του παντοπωλείου μπορεί να τοποθετήσει κατάλληλα τα προϊόντα στα ράφια, να κάνει την κατάλληλη καμπάνια προώθησης τους και να διαχειριστεί πιο αποδοτικά τα αποθεματικά του.

1.5.5 Οπτικοποίηση

Η οπτικοποίηση των δεδομένων συχνά βοηθάει στην καλύτερη κατανόηση όχι μόνο των ίδιων των δεδομένων, αλλά και των συσχετίσεων που μπορεί να υπάρχουν μεταξύ τους. Σε επόμενο κεφάλαιο θα περιγράψουμε τους βασικούς τρόπους οπτικοποίησης στην R. Ωστόσο, οπτικοποίηση μπορεί να γίνει μόνο για συγκεκριμένο αριθμό διαστάσεων. Αυτό σημαίνει ότι για σύνολα δεδομένων με πολλά χαρακτηριστικά, η οπτικοποίηση τους είναι ανέφικτη ή εναλλακτικά αρκούμαστε στην οπτικοποίηση ενός μικρού μέρους αυτών. Σε κάθε περίπτωση, οι οπτικοποιήσεις θα πρέπει να συνοδεύονται και από τους αντίστοιχους στατιστικούς ελέγχους, προκειμένου να βεβαιωθούμε για την εγκυρότητα των συσχετίσεων που απεικονίζονται.

1.5.6 Ανίχνευση Ανωμαλιών

Η ανίχνευση ανωμαλιών εστιάζει στην ανακάλυψη αποκλίσεων στα δεδομένα σε σχέση με αντίστοιχα δεδομένα, τα οποία έχουν συλλεχθεί στο παρελθόν ή με τυπικές τιμές των δεδομένων αυτών. Η Εικόνα 1.4 παρουσιάζει ένα τέτοιο παράδειγμα, στο οποίο με κόκκινο φαίνονται ένα κανονικό δείγμα, κοντά στα υπόλοιπα με φυσιολογικές τιμές δείγματα, και ένα ανώμαλο δείγμα, του οποίου η τιμή απέχει αρκετά από τα υπόλοιπα.



Εικόνα ***1.4*** *Παράδειγμα ανίχνευσης ανωμαλίας.*

Μερικά ακόμα παραδείγματα χρήσης της ανίχνευσης ανωμαλιών είναι τα ακόλουθα:

- ανίχνευση απάτης με βάση το προφίλ κάποιου χρήστη (δείτε Κριτήριο Αξιολόγησης 2),
- εντοπισμός δυσλειτουργικών αντικειμένων στη βιομηχανική κατασκευή, και
- έλεγχος (monitoring) υπολογιστών σε ένα κέντρο δεδομένων.

1.6 Εφαρμογές

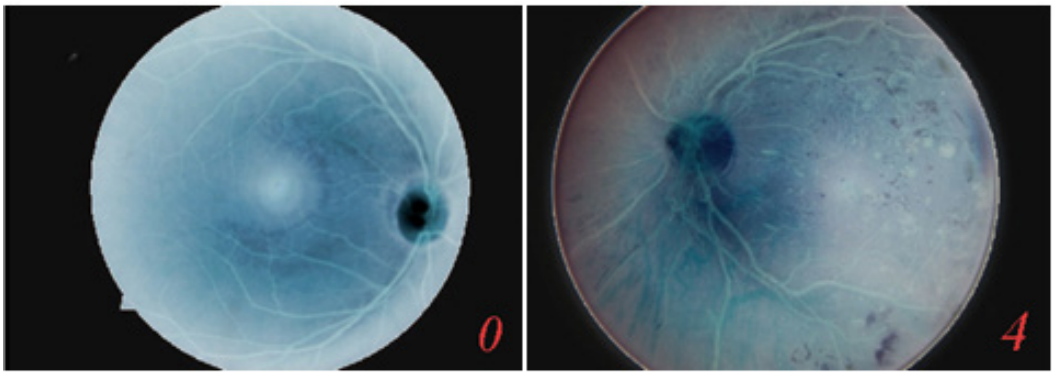
Η εξόρυξη δεδομένων χρησιμοποιείται σε ένα ευρύ φάσμα πεδίων, όπως π.χ. στην Ιατρική, στην Οικονομία, ακόμα και στις Τηλεπικοινωνίες. Παρακάτω παρουσιάζουμε μερικές εφαρμογές της ΕΔ σε μερικά από αυτά τα πεδία.

1.6.1 Ιατρική

Τα τελευταία χρόνια με την ανάπτυξη κλάδων της Ιατρικής, όπως η γενετική και η βιοϊατρική, αναδείχθηκε η χρησιμότητα της ΕΔ στην Ιατρική. Στον τομέα της γενετικής, στόχος είναι η κατανόηση και η χαρτογράφηση της σχέσης μεταξύ της μεταβολής των ακολουθιών του ανθρώπινου DNA και της προδιάθεσης κάποιας ασθένειας. Η ΕΔ είναι ένα εργαλείο, το οποίο μπορεί να βοηθήσει στη βελτίωση της διάγνωσης, της πρόληψης και κατ' επέκταση της θεραπείας ασθενειών.

Ένας από τους κύριους στόχους, που συνδέεται με την ανάλυση του DNA, είναι η σύγκριση ποικίλων ακολουθιών και η αναζήτηση ομοιοτήτων μεταξύ των δεδομένων του DNA. Η σύγκριση γίνεται μεταξύ της γονιδιακής ακολουθίας υγιών και βλαβερών ιστών, για να βρεθούν τυχόν διαφορές ανάμεσα τους.

Τα εργαλεία οπτικοποίησης παίζουν, επίσης, έναν σημαντικό ρόλο στην ΕΔ ως προς την βιοϊατρική. Τα εργαλεία αυτά μπορούν να παρουσιάσουν πολύπλοκες δομές γονιδίων σε γράφους, δένδρα και αλυσίδες. Η οπτική παρουσίαση συμβάλει στην καλύτερη κατανόηση αυτών των δομών και στην εξερεύνηση των δεδομένων.



Εικόνα *1.5* *Εικόνες υψηλής ανάλυσης αμφιβληστροειδούς. Εικόνες κλάσης 0 (αριστερά) και κλάσης 4 (δεξιά).*

Ένα παράδειγμα χρήσης της ΕΔ στην Ιατρική είναι η χρήση κατηγοριοποίησης για την ανίχνευση της διαβητικής αμφιβληστροειδοπάθειας (ΔΑ). Στο συγκεκριμένο πρόβλημα τα δεδομένα είναι εικόνες υψηλής ανάλυσης του αμφιβληστροειδούς των ασθενών (Εικόνα 1.5). Το πεδίο τιμών της κλάσης είναι από 0 έως και 4 (Πίνακας 1.1).

Κλάση	**Ερμηνεία**
0	Χωρίς ΔΑ
1	Ήπια μορφή ΔΑ
2	Μέτρια ΔΑ
3	Σοβαρή ΔΑ
4	Τελικό στάδιο ΔΑ (τύφλωση)

Πίνακας *1.1* *Τιμές κλάσης και η ερμηνεία τους για το παράδειγμα ανίχνευσης ΔΑ.*

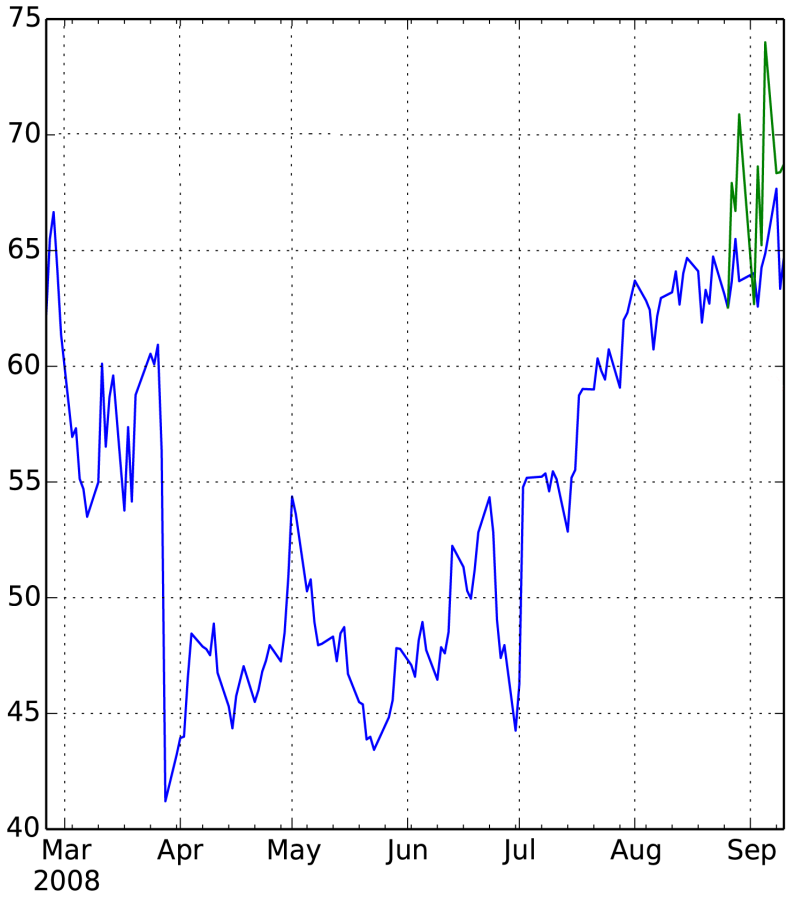
Επίσης, κλασικά παραδείγματα εφαρμογής της ΕΔ είναι η πρόβλεψη επιληπτικής κρίσης μέσα από την ανάλυση δεδομένων που συλλέχθηκαν από τη μαγνητική MRI των ασθενών και η κατηγοριοποίηση καρκίνου σε καλοήθη ή κακοήθη.

Τέλος, αναφορικά με τη δημόσια υγεία και την επιδημιολογία, η εφαρμογή τεχνικών της ΕΔ σε δεδομένα κοινωνικών δικτύων, όπως το Twitter, έδωσαν τη δυνατότητα ανακάλυψης ραγδαίας εξάπλωσης ιώσεων και ασθενειών, όπως γρίπη και HIV, αναφέροντας το περιστατικό σε πολύ λιγότερο χρόνο απ' ό,τι ο Δημόσιος Οργανισμός Υγείας, με αποτέλεσμα την εγκυρότερη πρόληψη της ασθένειας.

1.6.2 Οικονομία

Ακόμα ένας τομέας, στον οποίο εφαρμόζονται τεχνικές εξόρυξης δεδομένων, είναι η οικονομία. Οικονομικά δεδομένα συλλέγονται κατά κύριο λόγο από τράπεζες και άλλους οικονομικούς φορείς. Συνήθως πρόκειται για αξιόπιστα, ολοκληρωμένα και υψηλής ποιότητας δεδομένα. Η συνεισφορά της ΕΔ στην οικονομία συναντάται στη συλλογή και κατανόηση των δεδομένων, στον καθαρισμό και βελτίωση των δεδομένων, καθώς και στη δημιουργία ενός μοντέλου. Η ανάλυση των οικονομικών δεδομένων στοχεύει στη διευκόλυνση κατά τη λήψη αποφάσεων, ενεργώντας σύμφωνα με την ανάλυση που έχει γίνει.

Αρχικά, τα δεδομένα που συλλέγονται από διάφορους οικονομικούς οργανισμούς συγκεντρώνονται σε αποθήκες δεδομένων. Τεχνικές πολυδιάστατης ανάλυσης χρησιμοποιούνται για την ανάλυση τους. Επιπλέον, μια άλλη εφαρμογή της ΕΔ στην οικονομία σχετίζεται με την πρόβλεψη. Για παράδειγμα, η πρόβλεψη για το αν ένας πελάτης θα μπορέσει να αποπληρώσει το δάνειο που σκέφτεται να πάρει, βάσει των προηγούμενων συναλλαγών του με την τράπεζα και την τρέχουσα οικονομική του κατάσταση. Με την επεξεργασία τέτοιων δεδομένων, η τράπεζα μπορεί να αποφασίσει για τις πολιτικές δανειοδότησης, ώστε να υπάρχει χαμηλός κίνδυνος απώλειας κέρδους.



Εικόνα *1.6* *Παράδειγμα πρόβλεψης κίνησης τιμών μιας μετοχής.*

Μια ακόμα εφαρμογή της ΕΔ στην οικονομία είναι η προσπάθεια πρόβλεψης της κίνησης των μετοχών. Οι τιμές των μετοχών μοντελοποιούνται συνήθως ως χρονοσειρές (time series) και στόχος είναι η πρόβλεψη για το αν θα κινηθούν ανοδικά ή καθοδικά τιμές τους. Για παράδειγμα, στην Εικόνα 1.6, με βάση τις τιμές μιας μετοχής τους προηγούμενους μήνες, γίνεται πρόβλεψη για τις τιμές της το επόμενο δεκαπενθήμερο. Με πράσινο χρώμα φαίνεται η τιμή της πρόβλεψης, ενώ με μπλε η πραγματική τιμή της μετοχής. Η ακρίβεια των προβλέψεων ίσως να μην είναι όσο υψηλή θα ήθελε κανείς. Ωστόσο, η γενική τάση (ανοδική) της μετοχής προβλέφθηκε επιτυχώς.

Άλλο είδος τεχνικής που μπορεί να εφαρμοστεί είναι η συσταδοποίηση. Οι πελάτες ομαδοποιούνται σε συστάδες με βάσει κάποια χαρακτηριστικά. Η αποτελεσματική συσταδοποίηση βοηθάει τις τράπεζες να ταυτοποιούν μια ομάδα πελατών ή να συσχετίζουν έναν νέο πελάτη με κάποια υπάρχουσα ομάδα και να του προτείνουν λύσεις, οι οποίες ικανοποίησαν σε μεγάλο βαθμό τους πελάτες της συγκεκριμένης ομάδας.

Τέλος, με τη χρήση τεχνικών της ΕΔ μπορούν να εντοπιστούν πιθανές απάτες ή και παραποιημένα δεδομένα. Κάνοντας ανάλυση οικονομικών συναλλαγών και εξάγοντας κάποια μοτίβα, η καταγραφή ενός «ασυνήθιστου» γεγονότος μπορεί να ενεργοποιήσει μια διαδικασία ταυτοποίησης του πραγματικού πελάτη (βλ. Κριτήριο Αξιολόγησης 2).

1.6.3 Τηλεπικοινωνία

Η ΕΔ έχει συνεισφέρει και στις τηλεπικοινωνίες. Τα τηλεπικοινωνιακά δεδομένα, όπως ο τύπος κλήσης, η τοποθεσία του καλούντος και του κληθέντος, ο χρόνος κλήσης και η διάρκεια μπορούν να χρησιμοποιηθούν για την καλύτερη εξυπηρέτηση όχι μόνο καθενός πελάτη ξεχωριστά, αλλά καθολικά για όλους τους χρήστες του δικτύου. Τεχνικές ΕΔ χρησιμοποιούνται για την εξισορρόπηση του φορτίου του συστήματος και της κίνησης των δεδομένων.

Με χρήση των δεδομένων μπορεί να δημιουργηθεί ένα προφίλ των πελατών. Με βάση αυτά τα προφίλ και σε αυτόν τον τομέα μπορεί να γίνει μια συσταδοποίηση των πελατών και ανάλογα με την ομάδα, στην οποία

κατατάσσεται ένας υπάρχων ή νέος πελάτης, να του προσφέρονται τα αντίστοιχα πακέτα επικοινωνίας. Επιπλέον, μπορούν να εντοπιστούν ενδιαφέροντα μοτίβα και να αναλυθούν σε βάθος οι παράγοντες που οδηγούν σε αυτά. Για παράδειγμα, να προσδιοριστούν οι παράγοντες που οδηγούν σε μεγαλύτερη συχνότητα κλήσεων από τους χρήστες σε συγκεκριμένα χρονικά διαστήματα.

1.7 Προκλήσεις

Η Εξόρυξη Γνώσης είναι ένα πολλά υποσχόμενο επιστημονικό πεδίο. Όπως σε όλα τα επιστημονικά πεδία, έτσι και σε αυτό, υπάρχουν αρκετές προκλήσεις που καλούμαστε να αντιμετωπίσουμε. Οι προκλήσεις αυτές είναι κυρίως τεχνικές και κοινωνικές-ηθικές.

Η βασικότερη τεχνική πρόκληση είναι ο ολοένα αυξανόμενος όγκος των δεδομένων, καθώς και ο πολυδιάστατος και πολύπλοκος χαρακτήρας τους. Η υλοποίηση των λύσεων πρέπει να διέπετε από κλιμακωσιμότητα (scalability) και ευελιξία. Με άλλα λόγια, οι τεχνικές και οι αλγόριθμοι της ΕΔ θα πρέπει να μπορούν να χειριστούν τον μεγάλο όγκο δεδομένων εξίσου καλά με ένα μικρότερο σύνολο δεδομένων. Μια λύση για το μεγάλο όγκο των δεδομένων είναι η χρήση κατανεμημένων συστημάτων, όπως το Hadoop και το Map Reduce (Εικόνα 1.7), τα οποία θα μελετήσουμε στο Κεφάλαιο 8.

Εικόνα *1.7* *Hadoop & Map Reduce*

Όσον αφορά τις κοινωνικές προκλήσεις, οι κυριότερες είναι η εύρεση κατάλληλων περιοχών εφαρμογής της ΕΔ, καθώς και τα θέματα ιδιωτικότητας που μπορεί να προκύψουν από την επεξεργασία των δεδομένων κατά την ΕΔ.

1.8 Η Γλώσσα Προγραμματισμού R

Το βιβλίο αυτό αποτελεί ένα εγχειρίδιο εφαρμογής διάφορων τεχνικών και αλγορίθμων στην επίλυση πραγματικών προβλημάτων. Για να καταφέρει ο αναγνώστης να κατανοήσει πλήρως τη θεωρία, του δίνουμε ένα σύνολο από προβλήματα τόσο θεωρητικά όσο και πρακτικά. Πολλά από τα προβλήματα συνοδεύονται από μία ενδεικτική επίλυση. Τα πρακτικά προβλήματα απαιτούν τη χρήση μιας κατάλληλης γλώσσας προγραμματισμού, μέσω της οποίας ο αναγνώστης θα μπορέσει να ανταποκριθεί στις απαιτήσεις των προβλημάτων που του παρουσιάζονται. Στη θέση της γλώσσας αυτής έχουμε χρησιμοποιήσει τη γλώσσα R (Εικόνα 1.8).

Εικόνα *1.8* *Το λογότυπο της R.*

Ελπίζουμε να βρείτε αυτές τις επιλογές κατάλληλες για τα προβλήματα που έχετε να αντιμετωπίσετε και τελικά να τις υιοθετήσετε. Στη συνέχεια ακολουθεί μία περιγραφή των αντικειμένων, τα οποία καλύπτει κάθε ένα από τα κεφάλαια του βιβλίου.

Στο Κεφάλαιο 2 κάνουμε μια εισαγωγή στην R. Αν υπάρχει κάποιο κεφάλαιο, το οποίο δεν πρέπει να παραλειφθεί, είναι αυτό. Πιο συγκεκριμένα. παρουσιάζουμε τους τύπους δεδομένων που υποστηρίζει η R,

τις δομές ελέγχου, τον τρόπο δημιουργίας και κλήσης συναρτήσεων, τους κανόνες εμβέλειας που διέπουν τις μεταβλητές, τον τρόπο λειτουργίας βασικών και χρήσιμων συναρτήσεων, καθώς και τον τρόπο άντλησης βοήθειας σχετικά με τις συναρτήσεις οποιουδήποτε πακέτου. Το κεφάλαιο αυτό είναι προ-απαιτούμενο για όλα τα υπόλοιπα κεφάλαια.

Στο Κεφάλαιο 3 παρουσιάζουμε τους τύπους των δεδομένων, καθώς και τις βασικές ενέργειες προεπεξεργασίας των δεδομένων, προκειμένου να διασφαλιστεί η ποιότητά τους, και κατ' επέκταση η ποιότητα των αποτελεσμάτων. Έχοντας εξοικειωθεί με την R, παρουσιάζουμε τον τρόπο χρήσης συναρτήσεων από τα έτοιμα πακέτα `dplyr` και `tidyr`, για τη γρήγορη και αποδοτική διεκπεραίωση των διεργασιών της προεπεξεργασίας των δεδομένων.

Στο Κεφάλαιο 4 περιγράφουμε τη συνοπτική στατιστική και τρόπους οπτικοποίησης. Παρουσιάζουμε έννοιες, όπως τα μέτρα θέσης (μέση τιμή, διάμεσος), διασποράς (διασπορά, τυπική απόκλιση) και συσχέτισης, και τις συναρτήσεις που υλοποιούν αυτά τα μέτρα στην R. Στη συνέχεια, παρουσιάζουμε τρόπους οπτικοποίησης ποιοτικών δεδομένων, όπως είναι τα ιστογράμματα, τα ραβδογράμματα και τα διαγράμματα πίτας, με χρήση της γλώσσας R.

Στο Κεφάλαιο 5 μελετάμε τις έννοιες κατηγοριοποίηση και πρόβλεψη. Σε ό,τι αφορά την κατηγοριοποίηση, περιγράφουμε αναλυτικά τα δέντρα απόφασης. Αντίστοιχα, για την πρόβλεψη μελετάμε την τεχνική της γραμμικής παλινδρόμησης.

Στο Κεφάλαιο 6 παρουσιάζουμε τεχνικές συσταδοποίησης δεδομένων. Αφού ορίσουμε τη μη επιβλεπόμενη μάθηση και την έννοια της συστάδας, εξετάζουμε τρεις κατηγορίες τεχνικών συσταδοποίησης: τη διαμεριστική συσταδοποίηση, την ιεραρχική συσταδοποίηση και τη συσταδοποίηση που βασίζεται στην πυκνότητα. Στη συνέχεια γίνεται αναφορά σε συγκεκριμένους αλγορίθμους συσταδοποίησης, όπως ο αλγόριθμος k-means, ο συσσωρευτικός ιεραρχικός αλγόριθμος και ο αλγόριθμος DBSCAN.

Στο Κεφάλαιο 7 περιγράφουμε την εξόρυξη κανόνων συσχέτισης από συναλλαγματικές βάσεις δεδομένων. Αφού ορίσουμε τις βασικές έννοιες, παρουσιάζουμε τον αλγόριθμο Apriori για την εύρεση των συχνών στοιχειοσυνόλων. Έπειτα, δίνουμε ένα παράδειγμα εξόρυξης των κανόνων συσχέτισης από τα συχνά στοιχειοσύνολα. Τέλος, εξετάζουμε το πακέτο `arules`, στο οποίο υπάρχουν υλοποιημένα σε R όλα όσα περιγράφονται στο κεφάλαιο αυτό.

Στο Κεφάλαιο 8 μελετάμε υπολογιστικές μεθόδους για ανάλυση μεγάλου όγκου δεδομένων. Πιο συγκεκριμένα, εστιάζουμε στη λεπτομερή παρουσίαση δυο εργαλείων, του Hadoop και του MapReduce, καθώς και στον συνδυασμό τους για την επίλυση προβλημάτων.

Τέλος, το Κεφάλαιο 9 αποτελεί μια συλλογή λυμένων θεμάτων πάνω σε όλη την ύλη που καλύφθηκε στο παρόν σύγγραμμα.

1.9 Βασικές Έννοιες, Ορισμοί και Συμβολισμοί

Σε αυτό το σημείο θα ορίσουμε τις βασικότερες έννοιες και ορισμούς, καθώς και τους συμβολισμούς, που θα χρησιμοποιηθούν στα επόμενα κεφάλαια του βιβλίου. Αρχικά έχουμε τα δεδομένα μας, το λεγόμενο σύνολο δεδομένων. Το σύνολο δεδομένων είναι δομημένο σε γραμμές και στήλες. Κάθε στήλη αποτελεί ένα ξεχωριστό χαρακτηριστικό ή μεταβλητή του συνόλου δεδομένων. Το πλήθος των χαρακτηριστικών ενός συνόλου δεδομένων θα το συμβολίζουμε με m. Αντίστοιχα, κάθε γραμμή αποτελεί ένα δείγμα του συνόλου δεδομένων, το πλήθος των οποίων θα συμβολίσουμε με n. Κάθε δείγμα περιέχει m τιμές, μία για κάθε χαρακτηριστικό.

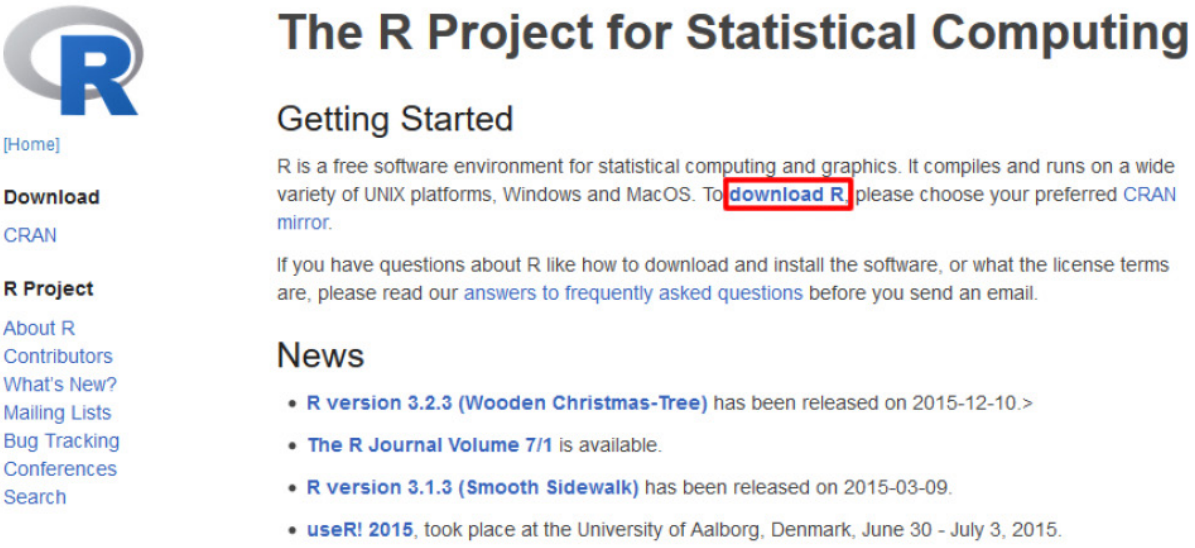
Χρησιμοποιούμε το σύνολο δεδομένων για τη δημιουργία ενός μοντέλου. Αυτό το μοντέλο θα χρησιμοποιηθεί στη συνέχεια για την αντιστοιχία νέων δειγμάτων σε ένα προκαθορισμένο σύνολο κατηγοριών ή κλάσεων (κατηγοριοποίηση) ή για την πρόβλεψη τιμών ενός σημαντικού χαρακτηριστικού, γνωστό και ως μεταβλητή στόχου (πρόβλεψη). Σε κάθε περίπτωση, το μοντέλο θα πρέπει να αξιολογηθεί. Για το σύνολο δεδομένων που έχουμε στη διάθεσή μας, η κλάση ή οι τιμές για τη μεταβλητή στόχου είναι γνωστές.

Για να κάνουμε έγκυρη αξιολόγηση του μοντέλου διαχωρίζουμε το σύνολο δεδομένων σε δύο υποσύνολα: το σύνολο εκπαίδευσης και το σύνολο ελέγχου. Συνήθως, το σύνολο εκπαίδευσης είναι τα 2/3 του αρχικού συνόλου δεδομένων, ενώ το σύνολο ελέγχου το υπόλοιπο 1/3. Όπως θα δούμε, υπάρχουν και άλλοι τρόποι διαχωρισμού του συνόλου δεδομένων.

Κατά τη φάση εκπαίδευσης χτίζεται το μοντέλο βάσει κάποιου αλγορίθμου και του συνόλου εκπαίδευσης. Η αξιολόγηση του μοντέλου είναι απαραίτητη πριν τη χρήση του και γίνεται χρησιμοποιώντας το σύνολο ελέγχου. Ουσιαστικά, χρησιμοποιούμε το μοντέλο πάνω σε δεδομένα, για τα οποία ήδη ξέρουμε την κλάση ή την τιμή της μεταβλητής στόχου, έτσι ώστε να μπορέσουμε να τη συγκρίνουμε με αυτή που θα παραχθεί από το μοντέλο.

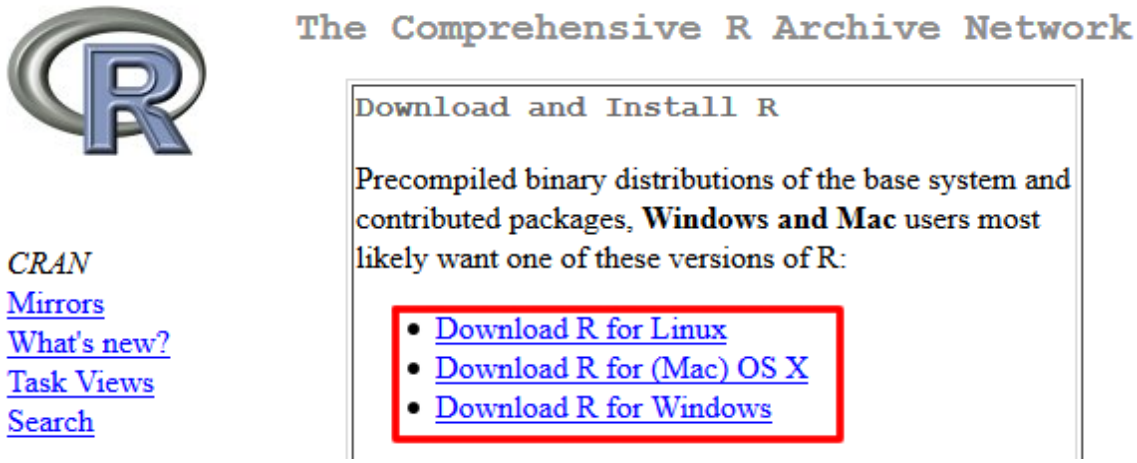
1.10 Εγκατάσταση Εργαλείων

Το βασικό εργαλείο που θα χρησιμοποιήσουμε είναι η γλώσσα προγραμματισμού R. Προαιρετικά, για τη διευκόλυνση του αναγνώστη, παρουσιάζουμε ένα ολοκληρωμένο περιβάλλον ανάπτυξης της R, το RStudio. Η απλή κονσόλα της R μαζί με τα βασικά έτοιμα πακέτα είναι ελεύθερα διαθέσιμη μέσω της επίσημης ιστοσελίδας (https://www.r-project.org/), όπως φαίνεται στην Εικόνα 1.9.



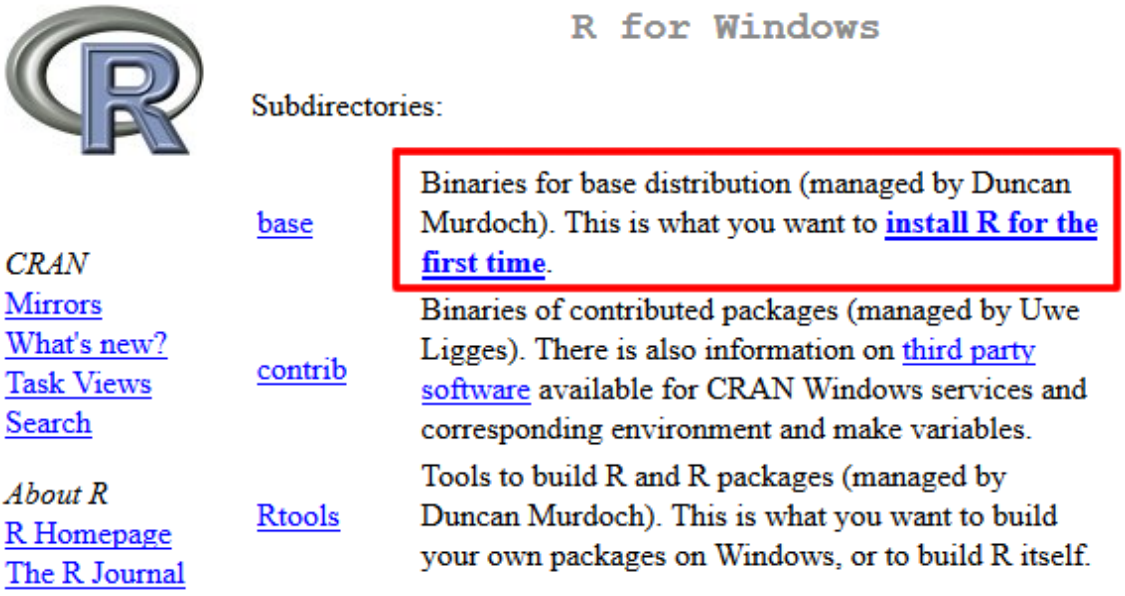
Εικόνα ***1.9*** *Περιεχόμενο αρχικής σελίδας της R.*

Από εκεί κάνοντας κλικ στο σύνδεσμο "download R" (Εικόνα 1.9, κόκκινο πλαίσιο), οδηγούμαστε σε μια οθόνη επιλογής τοποθεσίας των αρχείων από την οποία θα κατεβάσουμε τα αρχεία. Ο σύνδεσμος για την αντίστοιχη ελληνική τοποθεσία είναι: http://cran.cc.uoc.gr/mirrors/CRAN/ . Αν και δεν είναι υποχρεωτικό να κατεβάσετε από αυτό το σύνδεσμο την R, σας το προτείνουμε λόγω καλύτερης ταχύτητας λήψης των αρχείων. Έπειτα, θα πρέπει να επιλέξουμε το λειτουργικό στο οποίο δουλεύουμε, ώστε να κατεβάσουμε την κατάλληλη έκδοση της R (Εικόνα 1.10). Ενδεικτικά, θα δείξουμε την εγκατάσταση στο λειτουργικό σύστημα Windows.



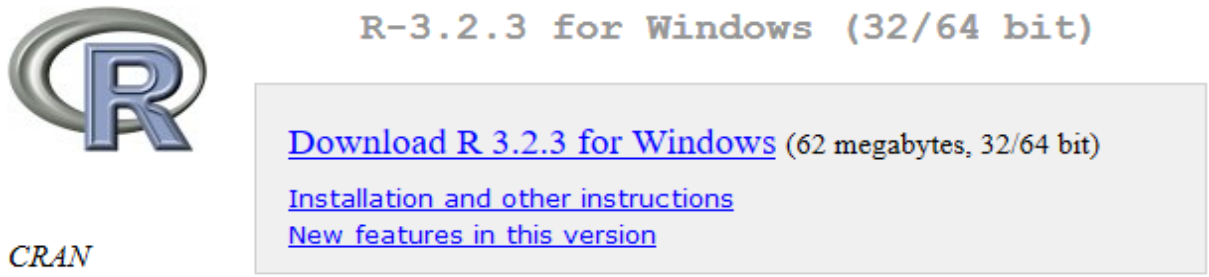
Εικόνα ***1.10*** *Επιλογή έκδοσης βάσει λειτουργικού συστήματος.*

Επιλέγοντας την έκδοση της R για Windows εμφανίζεται η οθόνη, όπως φαίνεται στην Εικόνα 1.11



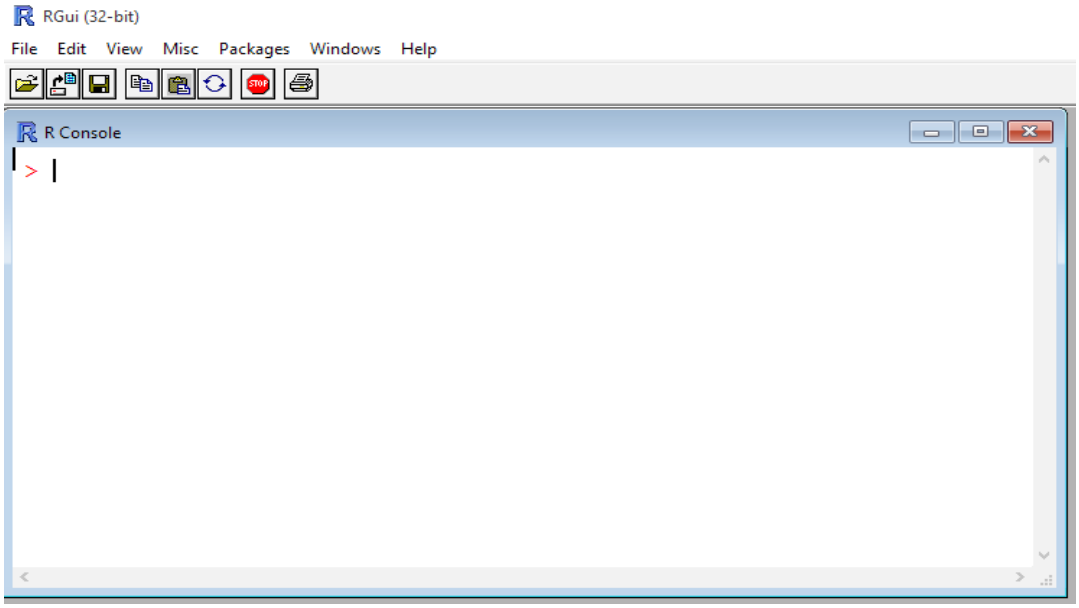
Εικόνα ***1.11*** *Οθόνη επιλογών εγκατάστασης.*

Εφόσον εγκαθιστούμε την R για πρώτη φορά, κάνουμε κλικ στο σύνδεσμο “install R for the first time”. Τέλος, εμφανίζεται ο σύνδεσμος λήψης του εκτελέσιμου της εγκατάστασης (Εικόνα 1.12).



Εικόνα ***1.12*** *Σύνδεσμος λήψης εκτελέσιμου εγκατάστασης της R.*

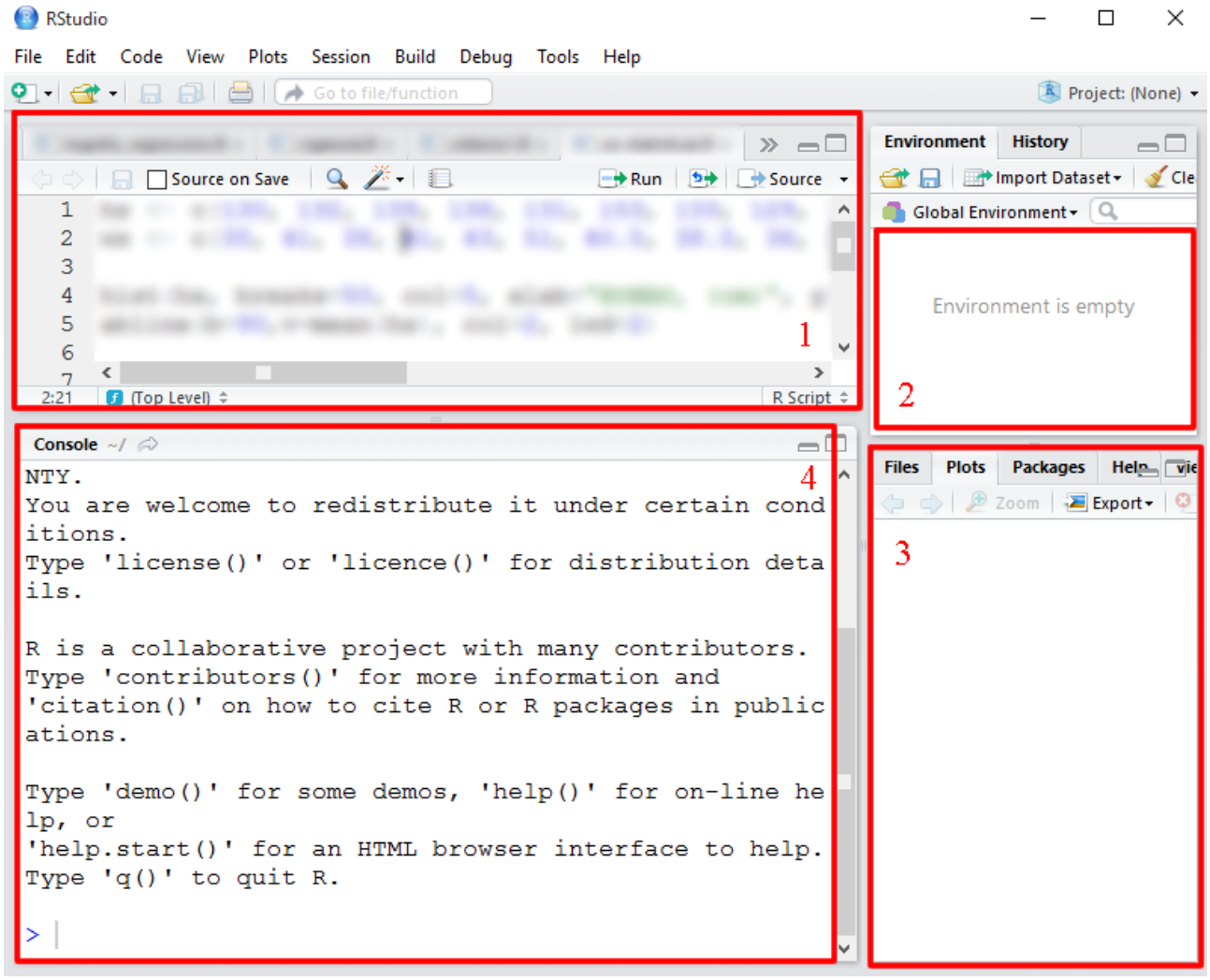
Η εγκατάσταση είναι πολύ απλή. Αρκεί να εκτελέσουμε το αρχείο και να ακολουθήσουμε τις οδηγίες του οδηγού εγκατάστασης. Στην Εικόνα 1.13 φαίνεται η κονσόλα της R.



Εικόνα *1.13* *Η κονσόλα της R.*

Το RStudio είναι ένα δωρεάν περιβάλλον ανάπτυξης της R, το οποίο διευκολύνει τον χρήστη με το απλή και εύχρηστη διεπαφή του. Το αρχείο εγκατάστασης μπορούμε να το κατεβάσουμε από τον σύνδεσμο: https://www.rstudio.com/products/rstudio/download/ . Θα πρέπει και πάλι να επιλέξουμε τη σωστή έκδοση ανάλογα με το λειτουργικό μας σύστημα.

Η αρχική οθόνη του Rstudio είναι όπως φαίνεται στην Εικόνα 1.14. Στο πλαίσιο 1 φαίνεται ο κώδικας των αρχείων που είναι ανοιγμένα. Κάθε καρτέλα αποτελεί και διαφορετικό αρχείο κώδικα. Στο πλαίσιο 2 εμφανίζονται οι μεταβλητές και οι συναρτήσεις. Στο πλαίσιο 3, στην καρτέλα "Plots", εκτυπώνονται οι γραφικές παραστάσεις, ενώ μπορούμε να δούμε και ποια πακέτα έχουμε κατεβάσει ή χρειάζονται ενημέρωση μέσω της καρτέλας "Packages". Επιπλέον, μέσω της καρτέλας "Help" (πλαίσιο 3) μπορούμε να βρούμε πληροφορίες και βοήθεια για κάποια συνάρτηση ή πακέτο. Τέλος, στο πλαίσιο 4 βρίσκεται η κλασσική κονσόλα της R.



Εικόνα ***1.14*** *Το RStudio περιβάλλον ανάπτυξης της R.*

Κριτήρια αξιολόγησης

Κριτήριο αξιολόγησης 1

Η εξόρυξη δεδομένων είναι μία περιοχή, η οποία στηρίζεται σε άλλες περιοχές, όπως η μηχανική μάθηση, η αναγνώριση προτύπων, η στατιστική, ο έλεγχος υποθέσεων, οι βάσεις δεδομένων και η ανάκτηση πληροφοριών. Μπορείτε να βρείτε δύο σημαντικές διαφορές που έχουν οι παραπάνω περιοχές από την εξόρυξη δεδομένων;

Απάντηση

Οι δύο πιο σημαντικές διαφορές ανάμεσα στις περιοχές, στις οποίες στηρίζεται η εξόρυξη των δεδομένων και στην εξόρυξη δεδομένων, είναι:

(1) η αυτοματοποίηση διαδικασίας δημιουργίας και αξιολόγησης υποθέσεων, και

(2) τα σύνολα δεδομένων, που αναλύονται στην εξόρυξη δεδομένων δεν αποτελούν συνήθως αποτέλεσμα ενός προσεκτικά σχεδιασμένου πειράματος, ενώ συχνά αναπαριστούν καιροσκοπικά, αντί για τυχαία, δείγματα δεδομένων.

Κριτήριο Αξιολόγησης 2

Επιλέξτε μία από τις ενέργειες που κάνατε στο πρόσφατο παρελθόν, η οποία θεωρείτε ότι είχε ως αποτέλεσμα τη συλλογή δεδομένων, και περιγράψτε σε μία παράγραφο ένα σενάριο χρήσης των δεδομένων αυτών στο πλαίσιο τεχνικών εξόρυξης δεδομένων.

Απάντηση

Την Κυριακή το μεσημέρι βρισκόμουν στο αεροδρόμιο της Λάρνακας, περιμένοντας να επιβιβαστώ στο αεροπλάνο, προκειμένου να επιστρέψω στην Ελλάδα. Στο αεροδρόμιο αυτό έκανα χρήση της πιστωτικής μου κάρτας, αγοράζοντας ένα προϊόν από τα καταστήματα αφορολογήτων ειδών. Πέντε λεπτά αργότερα με κάλεσαν στο κινητό μου τηλέφωνο με απόκρυψη. Στην άλλη γραμμή ήταν ένας υπάλληλος της εταιρείας από την οποία έχω βγάλει την πιστωτική μου κάρτα, ο οποίος, αφού μου συστήθηκε, με ρώτησε κατά πόσο την προηγούμενη ώρα είχα κάνει χρήση της κάρτας μου στο αεροδρόμιο της Λάρνακας. Η αγορά την οποία έκανα, δηλαδή η αγορά ενός προϊόντος σε μία ξένη χώρα που δεν είχα επισκεφτεί στο παρελθόν, δημιούργησε ένα γεγονός (alert) που πιθανόν δεν ταίριαζε στο προφίλ που είχε δημιουργήσει η εταιρεία για μένα, οπότε θεώρησε σωστό να εξακριβώσει, εάν πράγματι ήμουν εγώ αυτός που είχα κάνει τη συναλλαγή. Αυτό αποτελεί παράδειγμα των εργασιών ΕΔ, όπως της δημιουργίας προφίλ (profiling), δηλαδή της καταγραφής της αγοραστική συμπεριφοράς μου, και της ανίχνευσης ανωμαλιών.

Κριτήριο Αξιολόγησης 3

Ανατρέξτε στον παγκόσμιο ιστό, βρείτε και περιγράψτε δύο σύγχρονες (δηλαδή, με αναφορά στο διαδίκτυο εντός των 3 τελευταίων ετών) εφαρμογές της εξόρυξης δεδομένων.

Απάντηση

Διαβάστε το άρθρο «*Untangling the Social Web*» από το περιοδικό «*The Economist*» και δείτε εφαρμογές της εξόρυξης δεδομένων στην ανάλυση κοινωνικών δικτύων.

Κριτήριο Αξιολόγησης 4

Βρείτε δύο σύγχρονα (δηλαδή, με αναφορά στο διαδίκτυο εντός των 3 τελευταίων ετών) μη εμπορικά εργαλεία για εξόρυξη δεδομένων.

Απάντηση

R, Rattle, KNIME, WEKA, RapidMiner

Κριτήριο Αξιολόγησης 5

Υποθέστε ότι εργάζεστε σε μετεωρολογικό σταθμό. Ο συγκεκριμένος σταθμός κάνει μια από τις ακόλουθες προβλέψεις για τον καιρό κάθε μέρα: ηλιοφάνεια, βροχερός, συννεφιασμένος. Θέλετε να χρησιμοποιήσετε κάποιον αλγόριθμο μάθησης για την πρόβλεψη του καιρού της επόμενης ημέρας. Ως τι θα χαρακτηρίζατε το συγκεκριμένο πρόβλημα:

(1) πρόβλημα κατηγοριοποίηση

(2) πρόβλημα πρόβλεψης

Αιτιολογήστε την απάντησή σας.

Απάντηση

Η σωστή απάντηση είναι το (a). Στην κατηγοριοποίηση, η κλάση μπορεί να πάρει πεπερασμένο αριθμό τιμών, όπως για παράδειγμα ηλιοφάνεια, βροχερός, συννεφιασμένος στο πρόβλημα που αναφέρεται.

Κριτήριο Αξιολόγησης 6

Ποια από τα ακόλουθα προβλήματα μπορούν να λυθούν χρησιμοποιώντας κατηγοριοποίηση/πρόβλεψη (περισσότερες από μια σωστές απαντήσεις);

(1) Μελετάμε στατιστικά δυο ποδοσφαιρικών ομάδων και θέλουμε να προβλέψουμε το αποτέλεσμα του αυριανού αγώνα μεταξύ τους.

(2) Δοθέντος ενός μεγάλου συνόλου δεδομένων από ιατρικές εγγραφές ασθενών με καρδιακές ασθένειες, προσπαθούμε να ανακαλύψουμε διαφορετικές συστάδες ασθενών, οι οποίοι επιδέχονται διαφορετική θεραπευτική αγωγή.

(3) Δίνονται ιστορικά δεδομένα με την ηλικία και το ύψος παιδιών. Θέλουμε να προβλέψουμε το τελικό ύψος τους συναρτήσει της ηλικίας τους.

(4) Δίνεται μια συλλογή με 1000 εκθέσεις σχετικά με την παγκόσμια οικονομία. Στόχος είναι η εύρεση αυτόματης ομαδοποίησης σε σχετικές ομάδες.

Απάντηση

Οι σωστές απαντήσεις είναι οι (1) και (3).

Βιβλιογραφία

Fayyad, U, Piatetsky-Shapiro, G. & Smyth, P. (1996). From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, *17* (3), pp. 37-54.

The R Project for Statistical Computing. https://www.r-project.org/

RStudio. https://www.rstudio.com/

Brian, C. (2015). *Regression Models for Data Science in R*. Lean Publishing. Ανακτήθηκε στις 22 Νοεμβρίου 2015, από: https://leanpub.com/regmods

Wikipedia (2015). *Classification*. Ανακτήθηκε στις 22 Νοεμβρίου 2015, από: https://en.wikipedia.org/wiki/Classification

Data Mining Algorithms In R/Clustering/K-Means. Ανακτήθηκε στις 27 Νοεμβρίου 2015, από: https://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R/Clustering/K-Means

Wikipedia (2015). *Association Rule Learning*. Ανακτήθηκε στις 27 Νοεμβρίου 2015, από: https://en.wikipedia.org/wiki/Association_rule_learning

Hadoop. http://hadoop.apache.org

HDFS. http://hadoop.apache.org/docs/stable1/hdfsdesign.html

MapReduce. http://en.wikipedia.org/wiki/MapReduce

Κεφάλαιο 2: Εισαγωγή στην R

Σύνοψη

Μέσα από τη μελέτη αυτού του κεφαλαίου ο αναγνώστης μπορεί να εξοικειωθεί με τη γλώσσα R και να διακρίνει τους διαφορετικούς τύπους μεταβλητών που υποστηρίζει η R (λίστες, πλαίσια δεδομένων, πίνακες, παράγοντες, και διανύσματα). Στόχος μας στο παρόν κεφάλαιο είναι να μπορεί ο αναγνώστης να δημιουργεί τις δικές του συναρτήσεις στην R, να κατανοεί ορισμένες από τις βασικές της συναρτήσεις για φόρτωση, επεξεργασία και διερεύνηση πειραματικών δεδομένων και να είναι σε θέση να χρησιμοποιεί online πηγές, έτσι ώστε να μπορεί να προχωρήσει ανεξάρτητα στην περαιτέρω εκμάθηση της γλώσσας

Προαπαιτούμενη γνώση

Το τρέχον αποτελεί αυτόνομη ενότητα και δεν απαιτούνται προηγούμενες γνώσεις.

Εισαγωγή στην R

Η R δεν είναι απλά μια γλώσσα προγραμματισμού, αλλά και ένα περιβάλλον λογισμικού. Είναι ευρέως διαδεδομένη και χρησιμοποιείται κυρίως για στατιστικούς υπολογισμούς, για την παραγωγή γραφικών απεικονίσεων και για την επεξεργασία και ανάλυση των δεδομένων κατά την Εξόρυξη Δεδομένων.

Η υλοποίηση της R βασίστηκε στη γλώσσα προγραμματισμού S, την οποία δημιούργησε ο John Chambers, όσο βρισκόταν στα Bell Labs. Η R δημιουργήθηκε από τους Ross Ihaka και Robert Gentleman, στο πανεπιστήμιο Auckland στη Νέα Ζηλανδία. Τα τελευταία χρόνια έχει γίνει πολύ δημοφιλής, και πλέον αναπτύσσεται από μια ομάδα ανθρώπων, γνωστή ως R Development Core Team.

Οι βασικότεροι λόγοι για τους οποίους έγινε τόσο δημοφιλής η R είναι η ευκολία στην εκμάθησή της, η συμβατότητά της με τα πιο διαδεδομένα λειτουργικά συστήματα (Linux, Mac OS και Windows), το ότι διαθέτει έναν μεγάλο αριθμό έτοιμων πακέτων με καλογραμμένα εγχειρίδια χρήσης, και τέλος το γεγονός ότι είναι δωρεάν διαθέσιμη.

2.1 Τύποι Δεδομένων

2.1.1 Ορισμός και Κλάσεις Αντικειμένων

Στην κονσόλα της R, όπως φαίνεται στον Κώδικα 2.1, ο χρήστης μπορεί να πληκτρολογήσει διάφορες εκφράσεις. Στις περισσότερες γλώσσες προγραμματισμού υπάρχουν μεταβλητές και τύποι μεταβλητών. Ωστόσο, η R βλέπει τα πάντα ως αντικείμενα (object), τα οποία ανήκουν σε μια κλάση (class). Με απλά λόγια, για την R τα αντικείμενα είναι οι μεταβλητές, ενώ η κλάση είναι ο τύπος τους. Στην R δεν απαιτείται η ρητή δήλωση της κλάσης στην οποία ανήκουν τα αντικείμενα. Αυτή καθορίζεται αυτόματα από την τιμή που θα ανατεθεί στο αντικείμενο. Η ανάθεση τιμής γίνεται με τον τελεστή <− ή με τον τελεστή =. Η R έχει πέντε βασικές ή ατομικές (atomic) κλάσεις αντικειμένων:

- χαρακτήρας (character)
- αριθμητικός – πραγματικοί αριθμοί (numeric)
- ακέραιος (integer)
- σύνθετος (complex)
- λογικός (logical – True/False)

Η R χρησιμοποιεί, επίσης, βασικές δομές δεδομένων ως κλάσεις αντικειμένων. Η βασικότερη δομή είναι το διάνυσμα (vector). Ένα διάνυσμα μπορεί να περιέχει αντικείμενα του ίδιου μόνο τύπου. Η δημιουργία ενός διανύσματος είναι εφικτή, χρησιμοποιώντας είτε τη συνάρτηση `c`, είτε τη συνάρτηση `vector`.

Οι αριθμοί αντιμετωπίζονται γενικά ως αριθμητικά αντικείμενα, δηλαδή ως πραγματικοί αριθμοί. Στην περίπτωση που θέλουμε να ορίσουμε ρητά έναν αριθμό ως ακέραιο, θα πρέπει μετά τον αριθμό να ακολουθεί το επίθεμα L. Αξίζει να αναφέρουμε ότι υπάρχουν και ειδικές τιμές, όπως η τιμή Inf, η οποία αναπαριστά το

άπειρο, και η τιμή NaN (Not a Number) η οποία αναπαριστά μη ορισμένη τιμή.

```
> x <- "Hello World!"
> class(x)
[1] "character"
>
> y <- 3.14
> class(y)
[1] "numeric"
>
> z <- 15L
> class(z)
[1] "integer"
>
> c <- 5 + 2i
> class(c)
[1] "complex"
>
> t <- TRUE
> class(t)
[1] "logical"
```

Κώδικας *2.1* *Βασικές κλάσεις αντικειμένων της R.*

Κάθε αντικείμενο έχει συγκεκριμένες ιδιότητες, όπως:

- names,
- dim,
- class,
- length,
- και άλλες ιδιότητες ορισμένες από τον χρήστη.

```
> x <- list(age=c(10, 21 ,33), weight=c(30, 66, 80))
>
> names(x)
[1] "age"    "weight"
> length(x)
[1] 2
>
```

Κώδικας *2.2* *Παράδειγμα χρήσης των attributes.*

2.1.2 Διανύσματα και Λίστες

Όπως ήδη αναφέρθηκε στην Ενότητα 2.1.1, η R υποστηρίζει βασικές δομές δεδομένων ως κλάσεις αντικειμένων. Η βασικότερη δομή είναι το διάνυσμα. Ο πιο εύκολος τρόπος δημιουργίας ενός διανύσματος είναι με χρήση της συνάρτησης c, από την αγγλική λέξη concatenate (= συνένωση).

```
> x <- c("This", "is", "a", "character", "vector")
> x
[1] "This"      "is"        "a"         "character"
[5] "vector"
> y <- c(1, 2, 3, 5, 7)
> y
[1] 1 2 3 5 7
> class(x)
[1] «character»
> class(y)
[1] «numeric»
>
```

Κώδικας *2.3* *Παράδειγμα δημιουργίας διανύσματος με τη συνάρτηση c.*

Εναλλακτικά, μπορεί να χρησιμοποιηθεί η συνάρτηση `vector`. Γενικά, η δεικτοδότηση των διάφορων δομών στην R ξεκινούν από το 1 και όχι από το 0.

```
> # Αρχικοποίηση λογικού διανύσματος μήκους 5
> x <- vector(mode=»logical», length=5)
> x
[1] FALSE FALSE FALSE FALSE FALSE
> x[1] <- TRUE
> x
[1]  TRUE FALSE FALSE FALSE FALSE
>
```

Κώδικας *2.4* *Παράδειγμα δημιουργίας διανύσματος με τη συνάρτηση vector.*

Τα διανύσματα ανήκουν στα ατομικά (atomic) αντικείμενα. Αυτό σημαίνει ότι τα επιμέρους αντικείμενα του διανύσματος θα πρέπει να ανήκουν στην ίδια κλάση. Σε περίπτωση που αυτό δεν ισχύει, τότε η R δεν θα εκτυπώσει σφάλμα. Αντίθετα, θα κάνει τροποποίηση κλάσης, ώστε όλα τα αντικείμενα να ανήκουν στην ίδια κλάση. Για τις βασικές κλάσεις, η σειρά βαρύτητας είναι χαρακτήρας, αριθμός, λογικός.

```
> x <- c("Hello World!", 1, TRUE)
> x
[1] "Hello World!" "1"            "TRUE"
```

```
> y <- c(1, TRUE, FALSE)
> y
[1] 1 1 0
>
```

Κώδικας *2.5* *Παράδειγμα αυτόματης τροποποίησης κλάσης στοιχείων διανύσματος.*

Σε περίπτωση που θέλουμε να αποφύγουμε την αυτόματη μετατροπή, μπορούμε είτε να ορίσουμε ρητά το πώς αυτή θα γίνει με τις συναρτήσεις as.integer, as.numeric, as.logical κοκ, είτε να χρησιμοποιήσουμε μια άλλη δομή, τη λίστα (list). Μια λίστα, όπως και το διάνυσμα, είναι ένα σύνολο από αντικείμενα, που όμως μπορούν να ανήκουν σε διαφορετική κλάση. Για τη δημιουργία λίστας χρησιμοποιούμε τη συνάρτηση `list`.

```
> x <- list(«Hello World!», 2015, TRUE, 3.14)
> x
[[1]]
[1] "Hello World!"

[[2]]
[1] 2015

[[3]]
[1] TRUE

[[4]]
[1] 3.14

> class(x[[2]])
[1] «numeric»
>
```

Κώδικας *2.6* *Παράδειγμα δημιουργίας λίστας.*

2.1.3 Μητρώα

Ένα μητρώο (matrix) είναι ουσιαστικά πολλά διανύσματα ενωμένα. Δηλαδή, είναι μια ειδική δομή, η οποία έχει ως επιπλέον ιδιότητα (attribute) τη διάσταση (dimension). Με απλά λόγια, ένα μητρώο είναι ένα δισ-διάστατο διάνυσμα, και από τον ορισμό του είναι ατομικό, δηλαδή τόσο οι γραμμές όσο και οι στήλες του μητρώου θα πρέπει να περιέχουν αντικείμενα της ίδιας κλάσης. Υπάρχουν αρκετοί τρόποι δημιουργίας ενός μητρώου. Ένας από αυτούς είναι δημιουργώντας ένα διάνυσμα και στη συνέχεια θέτοντας τις διαστάσεις με τη συνάρτηση `dim`.

```
> mat <- c(1, 3, 2, 4)
> dim(mat) <- c(2, 2)
> mat
```

```
     [,1] [,2]
[1,]    1    2
[2,]    3    4
>
```

Κώδικας *2.7* *Παράδειγμα δημιουργίας μητρώου από διάνυσμα..*

Εναλλακτικά, μπορούμε να δημιουργήσουμε μητρώο μόνο με χρήση της συνάρτησης `matrix`, θέτοντας όμως κάποια επιπλέον ορίσματα.

```
> temp <- c(1, 2, 3, 7, 8, 9)
> mat <- matrix(temp, nrow=2, ncol=3, byrow=TRUE)
> mat
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    7    8    9


> # Default byrow=FALSE
> mat <- matrix(temp, nrow=2, ncol=3)
> mat
     [,1] [,2] [,3]
[1,]    1    3    8
[2,]    2    7    9
>
```

Κώδικας *2.8* *Παράδειγμα δημιουργίας μητρώου με χρήση της συνάρτησης matrix.*

Ένας ακόμα τρόπος είναι δένοντας υπάρχοντας διανύσματα είτε κατά γραμμές με χρήση της συνάρτησης `rbind` είτε κατά στήλες με χρήση της συνάρτησης `cbind`.

```
> t1 <- c(1, 2, 3)
> t2 <- c(7, 8, 9)
> # Κατά γραμμές
> rbind(t1, t2)
   [,1] [,2] [,3]
t1    1    2    3
t2    7    8    9
> # Κατά στήλες
> cbind(t1, t2)
```

```
     t1 t2
[1,]  1  7
[2,]  2  8
[3,]  3  9
>
```

Κώδικας *2.9* *Παράδειγμα δημιουργίας μητρώου με χρήση των συναρτήσεων rbind και cbind.*

2.1.4 Παράγοντες και Κατηγορικά Δεδομένα

Οι παράγοντες (factors) παρέχουν έναν εύκολο τρόπο αναπαράστασης και διαχείρισης κατηγορικών (nominal) δεδομένων. Διαθέτουν επίπεδα (levels), τα οποία ουσιαστικά είναι οι πιθανές τιμές που μπορούν να πάρουν. Η δημιουργία ενός παράγοντα γίνεται με χρήση της συνάρτησης `factor`. Η σειρά των επιπέδων σε κάποιες περιπτώσεις παίζει ρόλο. Επίσης, σε κάποιες περιπτώσεις είναι χρήσιμο να θέσουμε το όρισμα `levels` της συνάρτησης `factor`, περιορίζοντας έτσι τις επιτρεπτές τιμές. Σε αυτή την περίπτωση, τιμές, οι οποίες δεν αναφέρονται στο όρισμα levels, απορρίπτονται και θεωρούνται ελλιπείς τιμές.

```
> factor(c("Yes", "No", "No", "Yes"))
[1] Yes No  No  Yes
Levels: No Yes
> f <- factor(c("Yes", "No", "No", "Yes"), levels=c("Yes"))
> f
[1] Yes  <NA> <NA> Yes
Levels: Yes
>
```

Κώδικας *2.10* *Παράδειγμα δημιουργίας παράγοντα (factor).*

2.1.5 Ελλιπείς Τιμές

Υπάρχει ένας ειδικός τύπος δεδομένων για την αναπαράσταση των ελλιπών τιμών. Οι ελλιπείς τιμές στην R συμβολίζονται είτε ως NA, είτε ως NaN για μη προκαθορισμένες μαθηματικές πράξεις. Για τον έλεγχο ελλιπών τιμών υπάρχουν οι συναρτήσεις `is.na` και `is.nan`, αντίστοιχα. Η τιμή NA ανήκει στην αριθμητική κλάση, ενώ η τιμή NaN στη λογική κλάση.

```
> x <- NA
> is.na(x)
[1] TRUE
> y <- 0/0
> y
[1] NaN
> is.nan(y)
[1] TRUE
>
```

Κώδικας *2.11* *Παράδειγμα ελλιπών τιμών.*

2.1.6 Πλαίσια Δεδομένων

Τα πλαίσια δεδομένων (data frames) χρησιμοποιούνται για την αποθήκευση δεδομένων σε μορφή πίνακα. Έχουν παρόμοια δομή με τα μητρώα, καθώς είναι και αυτά δισδιάστατα. Ωστόσο, σε αντίθεση με τα μητρώα, μπορούν, όπως και οι λίστες, να έχουν διαφορετικό τύπο δεδομένων σε κάθε στήλη τους. Ένας λογικός περιορισμός που θέτουν είναι ότι κάθε στήλη μπορεί να περιέχει μόνο αντικείμενα της ίδιας κλάσης. Για τη δημιουργία ενός πλαισίου δεδομένων χρησιμοποιούμε τη συνάρτηση `data.frame`.

Τα στοιχεία σε ένα πλαίσιο δεδομένων μπορούν να προσπελαστούν με τον ίδιο τρόπο που γίνεται η δεικτοδότηση στα μητρώα. Μια ακόμα αξιοσημείωτη ιδιότητα τους είναι ότι σε κάθε στήλη μπορεί να ανατεθεί ένα όνομα. Από εκεί και έπειτα η αναφορά μπορεί να γίνεται και με το όνομα κάθε στήλης.

```
> x <- c("maria", "giwrgos", "giannhs")
> y <- c(15, 16, 20)
> z <- c(FALSE, FALSE, TRUE)
> dfr <- data.frame(username=x, age=y, adult=z)
> dfr
  username age adult
1    maria  15 FALSE
2  giwrgos  16 FALSE
3  giannhs  20  TRUE
>
> # Πρώτη γραμμή
> dfr[1,]
  username age adult
1    maria  15 FALSE
> # Πρώτη στήλη
> dfr[,1]
[1] maria   giwrgos giannhs
Levels: giannhs giwrgos maria
> # Στήλη ηλικίας
> dfr$age
[1] 15 16 20
>
```

Κώδικας *2.12* *Παράδειγμα δημιουργίας πλαισίου δεδομένων (data frame).*

2.2 Βασικές Εργασίες

2.2.1 Ανάγνωση Δεδομένων από Αρχείο

Μια από τις βασικότερες εργασίες είναι η ανάγνωση δεδομένων από κάποιο αρχείο. Οι πιο διαδεδομένοι τρόποι ανάγνωσης αρχείων είναι με χρήση των συναρτήσεων `read.table` και `read.csv`. Μερικά από τα βασικότερα ορίσματα τους είναι τα εξής:

- file, το όνομα του αρχείου,
- header, λογικό όρισμα, το οποίο σηματοδοτεί αν το αρχείο έχει γραμμή κεφαλίδας ή όχι,
- sep, αλφαριθμητικό, το οποίο ορίζει με τι διαχωρίζονται οι στήλες, π.χ. κόμμα, κενό κ.λπ.,
- colClasses, διάνυσμα χαρακτήρων με τις κλάσεις κάθε στήλης του συνόλου δεδομένων,
- nrows, αριθμός γραμμών, που θα διαβαστούν – η προεπιλογή είναι να διαβαστεί ολόκληρο το αρχείο,
- comment.char, αλφαριθμητικό, που ορίζει τον χαρακτήρα σχολίων στο αρχείο,
- skip, αριθμός γραμμών, που θα παραληφθούν από την αρχή του αρχείου,
- stringsAsFactors, λογικό όρισμα – τα αντικείμενα κλάσης χαρακτήρα να κωδικοποιηθούν ως παράγοντες. Η προεπιλεγμένη τιμή είναι TRUE.

```
> dat <- read.csv("./records.csv", stringsAsFactors=FALSE)
> dat
    ID age height weight gender
1  343  15    160     50 female
2  548  17    170     70   male
3  736  20    165     55 female
4  955  21    190     86   male
5  230  23    170     60 female
6  697  22    183     64   male
7  129  16    158     68 female
8  853  19    173     63   male
9  923  17    183    121   male
10 101  16    180     58   male
>
```

Κώδικας *2.13* *Παράδειγμα ανάγνωσης δεδομένων από αρχείο.*

2.2.2 Παραγωγή ακολουθιών

Η παραγωγή ακολουθιών είναι μια απλή, αλλά αρκετά σημαντική εργασία, αφού θέτει τα θεμέλια για πιο σημαντικές εργασίες, όπως η αναφορά σε υποσύνολο μιας δομής και η διανυσματοποίηση. Ο πιο απλός τρόπος παραγωγής ακολουθιών είναι με χρήση του τελεστή "**:**".

```
> x <- 1:10
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> y <- -5:5
> y
 [1] -5 -4 -3 -2 -1  0  1  2  3  4  5
>
```

Κώδικας *2.14* *Δημιουργία ακολουθίας με χρήση του τελεστή ":".*

Ένας άλλος τρόπος παραγωγής ακολουθιών είναι με χρήση της συνάρτησης `seq`. Η συνάρτηση `seq` δέχεται τα ακόλουθα ορίσματα:

- from, η αρχή της ακολουθίας,
- to, ο μέγιστος αριθμός, και συνεπώς το τέλος, της ακολουθίας,
- by, το βήμα αύξησης της ακολουθίας – προεπιλεγμένη τιμή το 1,
- length, αν δοθεί στη θέση του by, χωρίζει το διάστημα from-to σε τόσα διαστήματα και επιστρέφει τις τιμές των άκρων.

```
> x <- seq(from=2, to=10, by=3)
> x
[1] 2 5 8
> x <- seq(from=2, to=10, by=2)
> x
[1]  2  4  6  8 10
> y <- seq(from=2, to=10, length=4)
> y
[1]  2.000000  4.666667  7.333333 10.000000
>
```

Κώδικας *2.15* *Δημιουργία ακολουθίας με χρήση της συνάρτησης seq.*

Τέλος, μια ακόμα χρήσιμη συνάρτηση για παραγωγή ακολουθιών με συγκεκριμένο μοτίβο είναι και η συνάρτηση `rep`, η οποία δέχεται τα ακόλουθα ορίσματα:

- x, το αντικείμενο που θα χρησιμοποιηθεί για τη δημιουργία της ακολουθίας.
- times, φορές επανάληψης του αντικειμένου.

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

Κώδικας *2.16* *Δημιουργία ακολουθίας με χρήση της συνάρτησης rep.*

2.2.3 Αναφορά σε Υποσύνολα Δομών

Τα διανύσματα, τα μητρώα και οι λίστες παρέχουν έναν τρόπο ομαδοποίησης των δεδομένων. Ωστόσο, σε αρκετές περιπτώσεις ο χρήστης χρειάζεται να χρησιμοποιήσει ένα υποσύνολο μόνο αυτών των δεδομένων. Υπάρχουν 3 διαφορετικοί τελεστές για την αναφορά σε ένα υποσύνολο μιας δομής.

Ο πιο απλός τελεστής για αναφορά σε υποσύνολο δομής είναι η μονή τετραγωνική παρένθεση "`[`". Μπορεί να έχει ως περιεχόμενο έναν αριθμό ή μια ακολουθία αριθμών για την αναφορά ενός ή περισσότερων στοιχείων της δομής. Το αποτέλεσμα που επιστρέφεται ανήκει πάντα στην ίδια κλάση με αυτή στην οποία ανήκε το αρχικό αντικείμενο.

```
> x <- seq(1, 15, 2)
> x
[1]  1  3  5  7  9 11 13 15
> x[1:3]
[1] 1 3 5
> class(x[1:3])
[1] "numeric"
>
> y <- list("Hello", "Planet", "Earth!")
> y[c(1,3)]
[[1]]
[1] "Hello"

[[2]]
[1] "Earth!"

> class(y[c(1,3)])
[1] «list»
>
```

Κώδικας *2.17* *Παράδειγμα αναφοράς σε υποσύνολο δομής με τον τελεστή "[".*

Ένας άλλος τελεστής για αναφορά υποσυνόλου είναι οι διπλές τετραγωνικές παρενθέσεις "[[". Ο συγκεκριμένος τελεστής χρησιμοποιείται για λίστες και πλαίσια δεδομένων. Μπορεί να χρησιμοποιηθεί για αναφορά **ενός μόνο** στοιχείου της δομής και το αποτέλεσμα που επιστρέφεται ανήκει στην κλάση, στην οποία ανήκει το στοιχείο, όπου γίνεται η αναφορά.

```
> y <- list("Hello", "Planet", "Earth!")
> y[[1]]
[1] "Hello"
> class(y[[1]])
[1] "character"
> y[[c(1,3)]]
Error in y[[c(1, 3)]] : subscript out of bounds
>
```

Κώδικας *2.18* *Παράδειγμα αναφοράς σε υποσύνολο δομής με τον τελεστή "[[".*

Ο τρίτος τελεστής είναι το δολάριο "$" και μπορεί να γίνει αναφορά σε στοιχεία λίστας ή πλαισίου δεδομένων με ονόματα. Για την κλάση του αποτελέσματος που επιστρέφεται ισχύει η ίδια λογική με αυτή του τελεστή [[.

```
> y <- list(age=c(15, 16, 28), height=c(1.60, 1.68, 1.76))
> y
$age
[1] 15 16 28

$height
[1] 1.60 1.68 1.76

> class(y)
[1] "list"
> y$age
[1] 15 16 28
> class(y$age)
[1] «numeric»
```

Κώδικας *2.19* *Παράδειγμα αναφοράς σε υποσύνολο δομής με τον τελεστή "$".*

Ειδικά στα μητρώα, όταν γίνεται αναφορά σε ένα μόνο στοιχείο, το αποτέλεσμα θεωρείται ως ένα διάνυσμα μήκους 1, αντί για ένα μητρώο διαστάσεων 1x1. Με χρήση του ορίσματος `drop` μπορούμε να αλλάξουμε αυτή τη συμπεριφορά.

```
> x <- matrix(1:4, nrow=2, ncol=2, byrow=TRUE)
> x
     [,1] [,2]
[1,]    1    2
[2,]    3    4

> class(x[1,1])
[1] "integer"
> class(x[1,1, drop=FALSE])
[1] «matrix»
>
```

Κώδικας *2.20* *Παράδειγμα χρήσης του ορίσματος drop σε αναφορά στοιχείου μητρώου.*

Οι τελεστές [[και $ επιτρέπουν μερικό ταίριασμα ονομάτων. Αυτό είναι εφικτό με χρήση του ορίσματος exact.

```
> y <- list(age=c(15, 16, 28), height=c(1.60, 1.68, 1.76))
> y[["age"]]
[1] 15 16 28
> y[["a", exact=FALSE]]
[1] 15 16 28
>
```

Κώδικας *2.21* *Παράδειγμα χρήσης του ορίσματος exact για μερικό ταίριασμα ονόματος.*

Τέλος, μια από τις πιο χρήσιμες εφαρμογές της αναφοράς υποσυνόλου δομής είναι η εξαγωγή των θέσεων με ελλιπείς τιμές NA.

```
> y <- c(15, 20, 45, NA, NA, 50)
> y
[1] 15 20 45 NA NA 50
> # Εντοπίζουμε τις θέσεις με τιμή NA.
> i <- is.na(y)
> i
[1] FALSE FALSE FALSE  TRUE  TRUE FALSE
> # Αναφορά στο υποσύνολο του y, που δεν (!) περιέχει τιμές NA.
> y[!i]
[1] 15 20 45 50
>
```

Κώδικας *2.22* *Παράδειγμα αφαίρεσης ελλιπών τιμών.*

2.2.4 Διανυσματοποίηση

Ένα από τα χαρακτηριστικά της R που την κάνουν να ξεχωρίζει, είναι ότι επιτρέπει την εκτέλεση πράξεων μεταξύ διανυσμάτων και μητρώων. Η μετατροπή πράξεων σε πράξεις διανυσμάτων/μητρώων λέγεται διανυσματοποίηση (vectorization). Σε αρκετές περιπτώσεις, πράξεις μεταξύ διανυσμάτων ή/και μητρώων μπορούν να αντικαταστήσουν τη χρήση βρόχων επανάληψης. Έτσι, ο κώδικας μπορεί να γίνει συνοπτικός και αρκετά πιο ευανάγνωστος. Ο κυριότερος λόγος που θέλουμε να έχουμε όσο το δυνατόν περισσότερη διανυσματοποίηση είναι ότι οι πράξεις μεταξύ διανυσμάτων εκτελούνται πολύ πιο γρήγορα απ' ό,τι αν γίνονταν οι επιμέρους πράξεις μία-μία με βρόχο επανάληψης. Αυτό συμβαίνει, διότι κατά τις πράξεις μεταξύ διανυσμάτων, οι πράξεις εκτελούνται παράλληλα.

```
> x <- rnorm(10000000)
> y <- rnorm(10000000)
>
> z <- vector(mode="numeric", length=10000000)
>
> # Με χρήση βρόχου επανάληψης
> start <- proc.time()
> for (i in 1:10000000){
+    z[i] <- x[i] + y[i]
+ }
> proc.time()-start
   user  system elapsed
  15.23    0.03   15.29
> # Διανυσματοποίηση
> start <- proc.time()
> z <- x + y
> proc.time()-start
   user  system elapsed
   0.03    0.02    0.04
>
```

Κώδικας *2.23* *Σύγκριση χρόνου πρόσθεσης διανυσμάτων με βρόχο επανάληψης και διανυσματοποίηση.*

2.3 Δομές Ελέγχου

Η R διαθέτει βασικές δομές ελέγχου, όπως η υπό συνθήκη εκτέλεση και οι βρόχοι επανάληψης. Αυτές οι δομές ελέγχου είναι πολύ απλές, αλλά συγχρόνως χρήσιμες.

2.3.1 Εκτέλεση υπό συνθήκη: if-else

Αυτή είναι και η πιο βασική δομή ελέγχου. Η συντριπτική πλειοψηφία των προγραμμάτων σε κάποιο σημείο θα χρειαστεί να χρησιμοποιήσει υπό συνθήκη εκτέλεση. Ουσιαστικά, γίνεται έλεγχος μιας έκφρασης και αν αυτή αποτιμηθεί ως αληθής (True), τότε εκτελείται ένα κομμάτι κώδικα. Σε αντίθετη περίπτωση, υπάρχουν 3 πιθανά σενάρια. Είτε να συνεχιστεί η ροή του προγράμματος, είτε να ελεγχθεί κάποια άλλη συνθήκη, ή τέλος να εκτελεστεί ένα άλλο κομμάτι κώδικα, το οποίο θα θέλαμε να εκτελεστεί μόνο στην περίπτωση που η έκφραση είναι ψευδής (False).

```
> x <- 15
> if (x < 0) {
+    print("Negative!")
+ }else if (x < 10){
+    print("Positive, less than 10!")
+ }else{
+    print("Number larger than 10!")
+ }
[1] "Number larger than 10!"
>
```

Κώδικας *2.24* *Παράδειγμα δομής ελέγχου if-else.*

2.3.2 Εκτέλεση κατ' επανάληψη: for, repeat και while

Οι επαναληπτικές δομές έχουν ως στόχο την εκτέλεση ενός κομματιού κώδικα για προκαθορισμένο ή μη προκαθορισμένο αριθμό φορών. Στην R, ο βρόχος `for` αρκεί για να καλύψει το μεγαλύτερο μέρος των περιπτώσεων που απαιτούν επαναληπτική εκτέλεση. Ο βρόχος επανάληψης for μπορεί να χρησιμοποιηθεί με δυο τρόπους. Κατά τον κλασσικό τρόπο, μια μεταβλητή παίρνει τιμές από ένα καθορισμένο εύρος τιμών σε κάθε επανάληψη. Κατά τον εναλλακτικό τρόπο, μια μεταβλητή παίρνει τιμές από τα στοιχεία μιας συλλογής αντικειμένων.

```
> for(i in 1:10){
+    cat(i)
+    cat(" ")
+ }
1 2 3 4 5 6 7 8 9 10
> > letters
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n"
[15] "o" "p" "q" "r" "s" "t" "u" "v" "w" "x" "y" "z"
> for(x in letters){
+    cat(x)
+    cat(" ")
+ }
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
>
```

Κώδικας *2.25* *Παράδειγμα επαναληπτικού βρόχου for.*

Ένας άλλος τρόπος επανάληψης είναι με χρήση του βρόχου `while`. Σε έναν βρόχο `while` αρχικά ελέγχεται μια έκφραση. Όσο (while) η έκφραση είναι αληθής (true), τότε συνεχίζεται να εκτελείται το κομμάτι κώδικα μέσα στον βρόγο, μέχρι η έκφραση να γίνει ψευδής (false).

```
> x <- 1
> while (x < 5){
+    print(x)
+    x <- x+1
+ }
[1] 1
[1] 2
[1] 3
[1] 4
>
```

Κώδικας *2.26* *Παράδειγμα επαναληπτικού βρόχου while.*

Μια ακόμα επαναληπτική δομή είναι η `repeat`. Ουσιαστικά, πρόκειται για έναν ατέρμονο επαναληπτικό βρόχο. Ο μόνος τρόπος τερματισμού είναι με χρήση της εντολής `break`.

```
  > x <- 1
> repeat{
+    print(x)
+    if (x > 5){
+       break
+    }
+    x <- x+1
+ }
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
>
```

Κώδικας *2.27* *Παράδειγμα επαναληπτικού βρόχου repeat.*

2.3.3 Εντολές next και break

Οι εντολές `next` και `break` χρησιμοποιούνται σε συνδυασμό με τις επαναληπτικές δομές. Η εντολή next χρησιμοποιείται για την παράλειψη μιας επανάληψης ενός βρόχου. Ουσιαστικά, με την `next` ο βρόχος προχωράει στην επόμενη επανάληψη, μη εκτελώντας ότι ακολουθεί μετά την εντολή.

```
> for(i in 1:100){
+     # Προσπέρασε τις 20 πρώτες επαναλήψεις
+     if (i <= 20){
+        next
+     }
+ }
>
```

Κώδικας *2.28* *Παράδειγμα εντολής next..*

Η εντολή `break` χρησιμοποιείται για την άμεση έξοδο από έναν βρόχο επανάληψης. Στην περίπτωση εμφωλευμένων βρόχων, η `break` σταματάει μόνο τον πλησιέστερο βρόχο, μέσα στον οποίο και περιέχεται.

2.4 Συναρτήσεις

Οι συναρτήσεις είναι αναπόσπαστο κομμάτι της R. Σε κάθε ένα από τα πολλά έτοιμα πακέτα που είναι διαθέσιμα, υπάρχουν διαφορετικές συναρτήσεις, οι οποίες χρησιμοποιούνται για διαφορετική λειτουργία η κάθε μια. Πέρα από τις έτοιμες συναρτήσεις της, η R επιτρέπει στον χρήστη να ορίσει τις δικές του συναρτήσεις.

Οι συναρτήσεις ορίζονται, χρησιμοποιώντας την οδηγία ή δεσμευμένη λέξη `function` και αποθηκεύονται και αυτές ως αντικείμενα, όπως γίνεται σχεδόν με τα πάντα στην R. Πιο συγκεκριμένα, οι συναρτήσεις είναι αντικείμενα που ανήκουν στην κλάση "function". Παρακάτω θα δούμε ότι συναρτήσεις μπορούν να χρησιμοποιηθούν ως ορίσματα άλλων συναρτήσεων.

Στο παρακάτω παράδειγμα βλέπουμε τη δήλωση μιας απλής συνάρτησης, η οποία δέχεται έναν αριθμό και εκτυπώνει τόσες φορές ένα μήνυμα.

```
> myPrinter <- function(x){
+
+     for (i in seq_len(x)){
+        print("Hello World!")
+     }
+ }
>
> myPrinter(3)
[1] "Hello World!"
[1] «Hello World!»
[1] «Hello World!»
>
```

Κώδικας *2.29* *Ορισμός συνάρτησης από τον χρήστη.*

Κατά τη δήλωση μιας συνάρτησης μπορούμε να δώσουμε προεπιλεγμένες (default) τιμές, σε περίπτωση που ο χρήστης δεν δώσει τιμή για το αντίστοιχο όρισμα της συνάρτησης. Στο προηγούμενο παράδειγμα, αν ο χρήστης καλούσε τη συνάρτηση χωρίς όρισμα, τότε το μήνυμα θα εκτυπωνόταν 3 φορές, αφού η προεπιλεγμένη τιμή του ορίσματος είναι το 3.

Μπορούμε να καλέσουμε μια συνάρτηση, δίνοντας τα ορίσματα με διαφορετική σειρά, εφόσον όμως αναφέρουμε ρητά τα ονόματα τους.

```
> volume <- function(x=3, y=3, z=3){
+     print(x*y*z)
+ }
>
> volume(y=3, z=5, x=11)
[1] 165
> volume()
[1] 27
>
```

Κώδικας *2.30* *Πέρασμα ορισμάτων με διαφορετική σειρά.*

Στην περίπτωση που ο αριθμός των ορισμάτων μια συνάρτησης δεν είναι σταθερός, χρησιμοποιούμε το όρισμα "…", το οποίο μεταφράζεται ως 1 ή περισσότερα ορίσματα ακολουθούν. Το όρισμα "…" χρησιμοποιείται ως έχει μέσα στη συνάρτηση, όπως φαίνεται στο παρακάτω παράδειγμα. Ένα θέμα που προκύπτει από τη χρήση του ορίσματος "…" είναι ότι όσα ορίσματα ακολουθούν μετά από αυτό θα πρέπει να αναφερθούν ρητά κατά το πέρασμα τιμών στη συνάρτηση.

```
> myPrinter <- function(..., mes){
+     print(sum(...))
+     print(mes)
+ }
>
> myPrinter(3, 5, 11, mes = "Hi!")
[1] 19
[1] «Hi!»
>
```

Κώδικας *2.31* *Χρήση του ορίσματος ... σε συνάρτηση.*

2.5 Κανόνες Εμβέλειας

Οι κανόνες εμβέλειας είναι το βασικό χαρακτηριστικό της R που τη διαφοροποιεί από τον πρόγονό της, τη γλώσσα προγραμματισμού S. Αυτοί οι κανόνες χρησιμοποιούνται για να προκαθοριστεί ποια τιμή θα πάρει μια ελεύθερη μεταβλητή (free variable), η οποία ορίζεται και χρησιμοποιείται πρώτη φορά μέσα σε μια συνάρτηση. Η R χρησιμοποιεί λεξικολογική εμβέλεια (lexical scoping). Σύμφωνα με τους κανόνες της λεξικογραφικής εμβέλειας, οι τιμές των ελεύθερων μεταβλητών αναζητούνται στο ίδιο περιβάλλον, στο οποίο ορίστηκε και η συνάρτηση μέσα στην οποία ορίζονται.

Η R χρησιμοποιεί την έννοια του περιβάλλοντος (environment). Ένα περιβάλλον είναι μια συλλογή από ζεύγη (σύμβολο, τιμή), για παράδειγμα το σύμβολο x έχει τιμή 5. Κάθε περιβάλλον έχει ένα γονικό περιβάλ-

λον, με εξαίρεση το κενό περιβάλλον.

Για τον συσχετισμό τιμών και ελεύθερων μεταβλητών ακολουθείται η ακόλουθη διαδικασία αναζήτησης:

- Αν η τιμή ενός συμβόλου δεν βρεθεί στο περιβάλλον στο οποίο ορίστηκε η συνάρτηση, τότε η αναζήτηση συνεχίζεται στο γονικό περιβάλλον.
- Η αναζήτηση συνεχίζεται στη γονική ιεραρχία μέχρι το ανώτατο επίπεδο, στο οποίο βρίσκεται το καθολικό περιβάλλον (global environment), το οποίο ουσιαστικά είναι ο χώρος εργασίας (workspace) ή ο χώρος ονόματος του πακέτου (package namespace).
- Μετά το περιβάλλον στο ανώτατο επίπεδο, η αναζήτηση συνεχίζεται μέχρι να φτάσει στο άδειο περιβάλλον.
- Αν μια τιμή συμβόλου δεν έχει βρεθεί, πριν την άφιξη στο επίπεδο με το άδειο περιβάλλον, τότε η R παράγει σφάλμα.

2.6 Επαναληπτικές Συναρτήσεις

Η δημιουργία βρόχων for και while είναι χρήσιμη και εύκολη, όχι όμως όταν πρόκειται να έχουμε πολλά επίπεδα εμφωλευμένων βρόχων. Η R παρέχει κάποιες έτοιμες συναρτήσεις, οι οποίες υλοποιούν κατά κάποιο τρόπο τους βρόχους αυτούς με έναν πιο συμπαγή τρόπο.

2.6.1 lapply

Η `lapply` υπολογίζει το αποτέλεσμα μιας συνάρτησης πάνω στο κάθε στοιχείο μιας λίστας. Τα βασικά βήματα που εκτελεί η συγκεκριμένη συνάρτηση είναι τα εξής:

1. κάνει ένα πέρασμα της λίστας, στοιχείο προς στοιχείο,
2. εφαρμόζει τη συνάρτηση σε κάθε στοιχείο της λίστας, και
3. επιστρέφει μια λίστα.

Με την συνάρτηση `str`, μπορούμε να δούμε το πλήθος και τη σειρά των ορισμάτων οποιασδήποτε συνάρτησης. Όπως φαίνεται στον Κώδικα 2.32, η συνάρτηση `lapply` παίρνει 3 ορίσματα ως είσοδο:

- X, η λίστα πάνω στα στοιχεία της οποίας θα εφαρμοστεί η συνάρτηση FUN,
- FUN, η συνάρτηση που θα εφαρμοστεί ή το όνομα της συνάρτησης,
- …, περιέχει τα ορίσματα που θα περαστούν στη FUN.

Αν το X δεν είναι λίστα, τότε η R θα το μετατρέψει σε λίστα με χρήστη της συνάρτησης `as.list`.

```
> str(lapply)
function (X, FUN, ...)
> x <- list(a=rnorm(10), b=rnorm(20), c=rnorm(30))
> lapply(x, mean)
$a
[1] -0.1856258

$b
[1] -0.04310941

$c
[1] 0.3037476
>
```

Κώδικας *2.32* *Παράδειγμα χρήσης της lapply.*

2.6.2 sapply

Η `sapply` λειτουργεί όπως η `lapply`, αλλά επιχειρεί να απλοποιήσει το αποτέλεσμα που επιστρέφεται. Δηλαδή, η μόνη ουσιαστική διαφορά τους βρίσκεται στην επιστρεφόμενη τιμή. Πιο συγκεκριμένα, η `sapply` προσπαθεί να απλοποιήσει το επιστρεφόμενο αποτέλεσμα ως εξής:

- Αν το αποτέλεσμα είναι μια λίστα, της οποία τα στοιχεία έχουν όλα μήκος ίσο με 1, τότε επιστρέφεται ένα διάνυσμα.
- Αν το αποτέλεσμα είναι μια λίστα, της οποία τα στοιχεία είναι όλα διανύσματα ίδιου μήκους (> 1), τότε επιστρέφεται ένα μητρώο.
- Αν όλα τα υπόλοιπα αποτύχουν, τότε επιστρέφει μια λίστα.

Συγκρίνοντας τα αποτελέσματα του παρακάτω παραδείγματος με αυτά του παραδείγματος της `lapply` στην Ενότητα 2.6.1, γίνεται πιο ξεκάθαρη η χρησιμότητα και λόγος ύπαρξης της `sapply`.

```
> str(sapply)
function (X, FUN, ..., simplify = TRUE, USE.NAMES = TRUE)
> x <- list(a=rnorm(10), b=rnorm(20), c=rnorm(30))
> sapply(x, mean)
          a           b           c
 0.44342606 -0.05093783 -0.12495997
>
```

Κώδικας *2.33* *Παράδειγμα χρήσης της slapply.*

2.6.3 split

Η συνάρτηση `split` δεν ανήκει στις επαναληπτικές συναρτήσεις. Η συγκεκριμένη συνάρτηση παίρνει ως όρισμα ένα διάνυσμα ή κάποιο άλλο αντικείμενο και το χωρίζει σε ομάδες βάσει ενός παράγοντα (factor) ή μιας λίστα παραγόντων (list of factors). Ο λόγος, για τον οποίο αναφέρουμε αυτή τη συνάρτηση μαζί με τις επαναληπτικές συναρτήσεις, είναι ότι ο συνδυασμός της `split` μαζί με κάποια από τις `lapply` ή `sapply` αποτελεί κλασσικό παράδειγμα στην R.

Η βασική ιδέα είναι ότι μπορούμε να πάρουμε μια δομή δεδομένων, να τη χωρίσουμε σε υποσύνολα με βάση κάποια άλλη μεταβλητή, και στη συνέχεια να εφαρμόσουμε κάποια συνάρτηση σε αυτά τα υποσύνολα.

```
> dat <- data.frame(subject=1:6, age=c(15,17,16,20,21,23),
                    + adult=c(FALSE,FALSE,FALSE,TRUE,TRUE,TRUE))
> s <- split(dat, dat$adult)
> s
$`FALSE`
  subject age adult
1       1  15 FALSE
2       2  17 FALSE
3       3  16 FALSE
$`TRUE`
  subject age adult
4       4  20  TRUE
```

```
5        5  21  TRUE
6        6  23  TRUE

> sapply(s, function(x){
+    mean(x[["age"]])
+ })
   FALSE     TRUE
16.00000 21.33333
>
```

Κώδικας *2.34* *Παράδειγμα χρήσης της συνάρτησης split σε συνδυασμό με την sapply.*

2.6.4 tapply

Η συνάρτηση `tapply` χρησιμοποιείται για την εφαρμογή μιας συνάρτησης πάνω σε ένα υποσύνολο ενός διανύσματος. Μπορεί να θεωρηθεί ως συνδυασμός των συναρτήσεων `split` και `sapply`, αλλά μόνο για διανύσματα.

Τα ορίσματα της `tapply` είναι τα ακόλουθα:

- X, ένα διάνυσμα πάνω στο οποίο θα γίνει ο διαχωρισμός και η εφαρμογή της συνάρτησης,
- INDEX, παράγοντας (factor) ή λίστα παραγόντων (list of factors),
- FUN, η συνάρτηση που θα εφαρμοστεί,
- …, περιέχει τα υπόλοιπα ορίσματα που θα περαστούν στην FUN,
- simplify, λογικό όρισμα – να απλοποιηθούν τα αποτελέσματα ή όχι;

```
> str(tapply)
function (X, INDEX, FUN = NULL, ..., simplify = TRUE)
> x <- c(rnorm(10), rnorm(10), rnorm(10), rnorm(10))
>
> f <- gl(4, 10)
> f
 [1] 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3
[30] 3 4 4 4 4 4 4 4 4 4
Levels: 1 2 3 4
> tapply(x, f, mean)
         1          2          3          4
 0.2451280 -0.5042727 -0.1268080 -0.4878147
>
```

Κώδικας *2.35* *Παράδειγμα χρήσης της συνάρτησης tapply.*

2.7 Βοήθεια από την κονσόλα και Εγκατάσταση Πακέτων

Πέρα από το άπειρο υλικό, που μπορεί κανείς να βρει σε βιβλία και στο διαδίκτυο, η R παρέχει βοήθεια και μέσω της γραμμής εντολών της κονσόλας. Πιο συγκεκριμένα, ο χρήστης μπορεί να ζητήσει πληροφορίες για μια συνάρτηση, πληκτρολογώντας ένα αγγλικό ερωτηματικό μπροστά από το όνομα, π.χ. `?c`, `?vector`, `?tapply` κοκ. Η R φέρνει την αντίστοιχη σελίδα από το εγχειρίδιο, με την προϋπόθεση ότι υπάρχει σύνδεση στο διαδίκτυο.

Όπως έχει ήδη αναφερθεί, η R διαθέτει μια μεγάλη ποικιλία έτοιμων πακέτων. Η εγκατάσταση πακέτων γίνεται με τη συνάρτηση `install.packages`, ενώ το φόρτωμά τους στο περιβάλλον με τη συνάρτηση `library`. Ο Κώδικας 2.36 παρουσιάζει την εκτέλεση των δυο προαναφερφέντων εντολών για την εγκατάσταση και φόρτωση του πακέτου `rattle`.

```
> install.packages("rattle")
Installing package into 'C:/Users/User/Documents/R/win-library/3.1'
(as 'lib' is unspecified)
trying URL 'http://cran.rstudio.com/bin/windows/contrib/3.1/rattle_3.5.0.zip'
Content type 'application/zip' length 3649991 bytes (3.5 MB)
opened URL
downloaded 3.5 MB

package 'rattle' successfully unpacked and MD5 sums checked

> library(«rattle»)
>
```

Κώδικας *2.36* *Παράδειγμα εγκατάστασης του πακέτου rattle.*

Κριτήρια αξιολόγησης

Κριτήριο αξιολόγησης 1

Απαντήστε στις παρακάτω ερωτήσεις πολλαπλής επιλογής. Μόνο μία είναι η σωστή απάντηση σε κάθε ερώτημα.

1. **Ποιος από τους παρακάτω τύπος δεδομένων (κλάση αντικειμένου) ΔΕΝ είναι ατομικός;**

   (a) λίστα (list)
   (b) διάνυσμα (vector)
   (c) αριθμητικός (numeric)
   (d) λογικός (logical)

2. **Δοθέντων δυο διανυσμάτων x <− c(1, 2, 3) και y <− c(4, 5, 6), ποιο είναι το αποτέλεσμα της εκτέλεσης της εντολής rbind(x, y);**

   (a) ένα 2x2 μητρώο
   (b) ένα 2x3 μητρώο
   (c) ένα 3x2 μητρώο
   (d) ένα 3x3 μητρώο

3. **Ένα βασικό χαρακτηριστικό των διανυσμάτων στην R είναι ότι:**

   (a) έχουν όλα το ίδιο μήκος.
   (b) το μήκος τους πρέπει να είναι λιγότερο των 1024 στοιχείων.
   (c) ένα διάνυσμα δεν έχει ιδιότητες.
   (d) όλα τα στοιχεία ενός διανύσματος πρέπει να ανήκουν στην ίδια κλάση.

4. **Σας δίνεται η ακόλουθη συνάρτηση:**

```
t <- function(x) {
        g <- function(y) {
                y + z
        }
        z <- 5
        x + g(x)
}
```

και στη συνέχεια εκτελούμε:

```
z <- 8
t(2)
```

Ποια τιμή επιστρέφεται;

(a) 5
(b) 4
(c) 12
(d) 9

5. **Υποθέστε ότι έχετε το διάνυσμα x <− c(7, 8, 10, 2, 3, 0). Ποια εντολή θα αντικαταστήσει όλες τις τιμές μεγαλύτερες του 5 με 0;**

(a) x[x < 5] <− 0
(b) x[x == 5] <− 0
(c) x[x >= 5] <− 0
(d) x[x > 5] <− 0

Απαντήσεις

Οι σωστές απαντήσεις των παραπάνω ερωτημάτων είναι: 1a, 2b, 3d, 4d, 5d.

Βιβλιογραφία

Peng, R. D. (2015). *R Programming for Data Science*. Lean Publishing. Ανακτήθηκε στις 15 Ιουνίου 2015, από: https://leanpub.com/rprogramming

R Core Team (2015). *An Introduction to R (ver 3.2.2)*. Ανακτήθηκε στις 19 Νοεμβρίου 2015, από: http://cran.r-project.org/doc/manuals/r-release/R-intro.html

Wikipedia (2015). *R programming language.* Ανακτήθηκε στις 19 Νοεμβρίου 2015, από: https://en.wikipedia.org/wiki/R_%28programming_language%29

Κεφάλαιο 3: Τύποι, Ποιότητα και Προεπεξεργασία Δεδομένων

Σύνοψη

Μέσα από τη μελέτη αυτού του κεφαλαίου ο αναγνώστης κατανοεί ότι τα δεδομένα, οι τύποι και η ποιότητά τους συνιστούν ένα αναπόσπαστο κομμάτι στη διαδικασία της εξόρυξης δεδομένων. Γίνεται απόλυτα σαφές ότι η ποιότητα των δεδομένων καθορίζει σε μεγάλο βαθμό και την ποιότητα των αποτελεσμάτων της εξόρυξης δεδομένων. Οι παράμετροι εκείνοι των δεδομένων που επηρεάζουν την ποιότητά τους πρέπει να είναι σαφείς, έτσι ώστε να είναι σε θέση κάποιος να τις αξιολογήσει και να τις βελτιώσει. Η προεπεξεργασία των δεδομένων αποτελεί το πιο επίπονο και χρονοβόρο κομμάτι στη διαδικασία της ανακάλυψης γνώσης από τα δεδομένα. Στόχος, επίσης, του κεφαλαίου είναι να εξοικειωθεί ο φοιτητής με όλες τις διαφορετικές μορφές προεπεξεργασίας των δεδομένων και να είναι σε θέση να τις εφαρμόσει. Παράλληλα, να είναι σε θέση να εφαρμόζει τις τεχνικές αυτές μέσω ενός εργαλείου, όπως η γλώσσα προγραμματισμού R.

Προαπαιτούμενη γνώση

Πριν το τρέχον κεφάλαιο θα πρέπει να μελετηθεί τόσο το Κεφάλαιο 1 – Εισαγωγή στην Εξόρυξη Δεδομένων όσο και το Κεφάλαιο 2 – Εισαγωγή στην R.

Τύποι, Ποιότητα και Προεπεξεργασία Δεδομένων

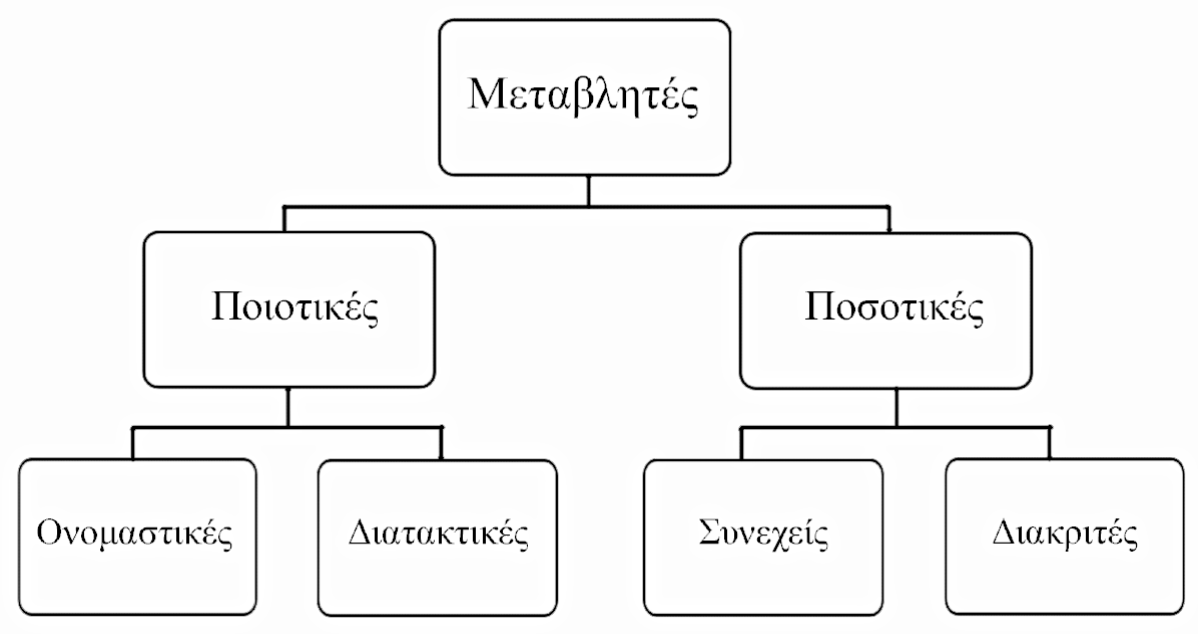
Η προεπεξεργασία των δεδομένων αποτελεί ένα από τα πιο σημαντικά βήματα της Ανακάλυψης Γνώσης σε Βάσεις Δεδομένων, το οποίο μπορεί να απαιτήσει έως και το 60% της συνολικής προσπάθειας. Αυτό συμβαίνει διότι, αν τα ίδια τα δεδομένα δεν είναι «καθαρά» και στην κατάλληλη μορφή, δεν έχει νόημα να μιλάμε για ποιότητα αποτελεσμάτων. Παρακάτω θα μελετήσουμε τις βασικές κατηγορίες και τύπους μεταβλητών, που μπορεί να έχει ένα σύνολο δεδομένων. Επιπλέον, θα συζητήσουμε από ποιες διεργασίες συνίσταται η προεπεξεργασία και πότε χρησιμοποιείται η κάθε μια, ενώ θα δούμε και πρακτικά παραδείγματά τους. Κλείνοντας, θα παρουσιάσουμε τα πακέτα `dplyr` και `tidyr` της R, τα οποία χρησιμοποιούνται για διαχείριση και καθαρισμό των δεδομένων, αντίστοιχα.

3.1 Κατηγορίες και Τύποι Μεταβλητών

Οι δυο βασικές κατηγορίες μεταβλητών είναι οι ποιοτικές και οι ποσοτικές. Οι ποιοτικές μεταβλητές αναφέρονται σε μεταβλητές, όπως για παράδειγμα το φύλο, το επίπεδο μόρφωσης, η περιοχή καταγωγής κ.ο.κ. Διαχωρίζονται σε ονομαστικές (nominal) και σε διατακτικές ή τακτικές. Οι ονομαστικές (nominal) μεταβλητές αναπαριστούν κατηγορίες, που η σειρά τους δεν έχει σημασία, π.χ. χρώμα[1], μέσο μεταφοράς. Αντίθετα, οι διατακτικές ή τακτικές μεταβλητές αναπαριστούν κατηγορίες, των οποίων η διάταξη έχει σημασία, π.χ. σοβαρότητα ασθένειας, γνώμη.

Οι ποσοτικές μεταβλητές είναι αριθμητικές τιμές, οι οποίες εκφράζονται σε μια μονάδα μέτρησης, π.χ. ηλικία. Διαχωρίζονται σε ασυνεχείς ή διακριτές και σε συνεχείς. Τα δεδομένα ανάλογα με την κλίμακα μέτρησης τους χαρακτηρίζονται ως κατηγορικά, δηλαδή σε επίπεδα ή κατηγορίες, και σε μετρήσεις. Στην Εικόνα 3.1 φαίνονται συνοπτικά όλες οι κατηγορίες και τύποι μεταβλητών.

1 Ανάλογα με την περιοχή εφαρμογής, το χρώμα θα μπορούσε να θεωρηθεί και ως τακτική μεταβλητή, αν λάβει κανείς υπόψη του τις διαφορετικές διαβαθμίσεις (φάσμα συχνοτήτων) που μπορεί ένα χρώμα να έχει.



Εικόνα ***3.1*** *Κατηγορίες και τύποι μεταβλητών*

3.2 Διεργασίες Προεπεξεργασίας

Όπως αναφέρθηκε ήδη, η προεπεξεργασία των δεδομένων είναι ίσως το σημαντικότερο βήμα για την ΑΓΒΔ. Γι' αυτό και θα πρέπει να έχει προηγηθεί κάποια προεπεξεργασία των δεδομένων που θα χρησιμοποιηθούν, ώστε να εξασφαλιστεί η ποιότητά τους. Παρακάτω παρουσιάζουμε τις βασικότερες διεργασίες που λαμβάνουν χώρα κατά το στάδιο της προεπεξεργασίας των δεδομένων.

3.2.1 Καθαρισμός Δεδομένων

Οι βασικότερες δραστηριότητες του καθαρισμού δεδομένων είναι:

- συμπλήρωση ελλιπών τιμών,
- αναγνώριση ακραίων τιμών (outliers) και εξομάλυνση, εφόσον περιέχουν θόρυβο, και τέλος
- διόρθωση τυχόν ασυνεπειών στα δεδομένα.

3.2.1.1 Ελλιπείς Τιμές

Τα δεδομένα δεν είναι πάντα διαθέσιμα. Πιο συγκεκριμένα, σε αρκετές πλειάδες (γραμμές) του συνόλου δεδομένων δεν υπάρχουν καταγεγραμμένες τιμές για ορισμένα από τα χαρακτηριστικά. Αυτό το φαινόμενο είναι γνωστό ως *ελλιπείς τιμές*. Μπορεί να οφείλεται σε διάφορους παράγοντες, όπως για παράδειγμα σε δυσλειτουργία του εξοπλισμού, σε ασυνέπειες με άλλα καταγεγραμμένα δεδομένα, που οδήγησε στη διαγραφή τους ή ακόμα και στη μη καταχώρηση τους. Σε κάθε περίπτωση, τα ελλιπή δεδομένα ενδεχομένως να πρέπει να τα συμπεράνουμε και να τα συμπληρώσουμε.

Το πρώτο βήμα στον χειρισμό των ελλιπών δεδομένων είναι η αναγνώριση των πλειάδων με ελλιπείς τιμές. Στη συνέχεια γίνεται η συμπλήρωση. Προφανώς, αν το σύνολο δεδομένων είναι μεγάλο, αυτό δεν μπορεί να γίνει χειρωνακτικά. Η πιο εύκολη λύση είναι να αγνοήσουμε τη συγκεκριμένη πλειάδα-γραμμή. Ωστόσο, αν έχουμε μεγάλο αριθμό ελλιπών τιμών αυτό δεν αποτελεί αποτελεσματική λύση. Μερικές από τις πιο αποτελεσματικές, αυτοματοποιημένες μεθόδους συμπλήρωσης ελλιπών τιμών είναι οι ακόλουθες:

- Χρήση καθολικής σταθεράς για τη συμπλήρωση των χαμένων τιμών, π.χ. -1, "unknown", νέα κλάση.
- Χρήση της μέσης τιμής του χαρακτηριστικού για τη συμπλήρωση των χαμένων τιμών.
- Χρήση της μέσης τιμής των δειγμάτων της ίδια κλάσης για τη συμπλήρωση των χαμένων τιμών.

- Χρήση της πιο πιθανής τιμής για τη συμπλήρωση των χαμένων τιμών, η οποία παράγεται από κάποια τεχνική, όπως η παλινδρόμηση, τα δένδρα απόφασης κ.ά.

3.2.1.2 Δεδομένα με Θόρυβο

Τα δεδομένα ενδεχομένως να είναι διαθέσιμα, αλλά να υπάρχει θόρυβος ή ακραίες τιμές σε αυτά. Για παράδειγμα, μπορεί να έχουμε λανθασμένες τιμές χαρακτηριστικών λόγω προβληματικού εξοπλισμού λήψης μετρήσεων ή λόγω κάποιου προβλήματος κατά την εγγραφή των δεδομένων.

Υπάρχουν αρκετές μέθοδοι για τον χειρισμό δεδομένων με θόρυβο. Θα εστιάσουμε στις μεθόδους ενδοχείασης (binning) και στη συσταδοποίηση. Οι μέθοδοι ενδοχείασης έχουν όλες ως πρώτο βήμα την ταξινόμηση των δεδομένων, έτσι ώστε στη συνέχεια να διαχωριστούν σε δοχεία (bins). Διακρίνονται με βάση τον διαμερισμό σε δοχεία, σε μεθόδους διαμερισμού ίσου πλάτους (απόσταση) και σε μεθόδους διαμερισμού ίσου βάθους (συχνότητα).

Κατά τον διαμερισμό ίσου πλάτους, το εύρος διαιρείται σε N διαστήματα ίσου μεγέθους. Ωστόσο, αυτός ο διαμερισμός είναι επιρρεπής σε ακραίες τιμές, καθώς τα ασύμμετρα δεδομένα δεν διαχειρίζονται σωστά. Κατά τον διαμερισμό ίσου βάθους, το εύρος διαιρείται σε N διαστήματα, τα οποία περιέχουν τον ίδιο αριθμό δειγμάτων. Σε αυτή την περίπτωση έχουμε καλύτερη κλιμάκωση των δεδομένων. Οι μέθοδοι ενδοχείασης χρησιμοποιούνται και για διακριτοποίηση. Οι πιο γνωστές είναι οι ακόλουθες:

- Ομαλοποίηση με βάση τη μέση τιμή του κάθε δοχείου: οι τιμές αντικαθίστανται με τη μέση τιμή κάθε δοχείου.
- Ομαλοποίηση με χρήση του μεσαίου (median) του κάθε δοχείου: οι τιμές αντικαθίστανται με τη μεσαία (median) τιμή κάθε δοχείου.
- Ομαλοποίηση με χρήση των ορίων του κάθε δοχείου: οι τιμές αντικαθίστανται με την τιμή των ορίων, ανάλογα σε ποιο από τα δυο είναι πιο κοντά.

Παράδειγμα - Εξομάλυνση δεδομένων με μεθόδους ενδοχείασης

Έστω ότι μας δίνονται κάποιες θερμοκρασίες (σε ^{o}C**) ταξινομημένες σε αύξουσα σειρά: 4, 9, 11, 16, 21, 2**3, 24, 24, 27, 30, 32, 35. Με διαμερισμό ίσου βάθους έχουμε τα εξής δοχεία:

- Δοχείο 1: 4, 9, 11, 16
- Δοχείο 2: 21, 23, 24, 24
- Δοχείο 3: 27, 30, 32, 35

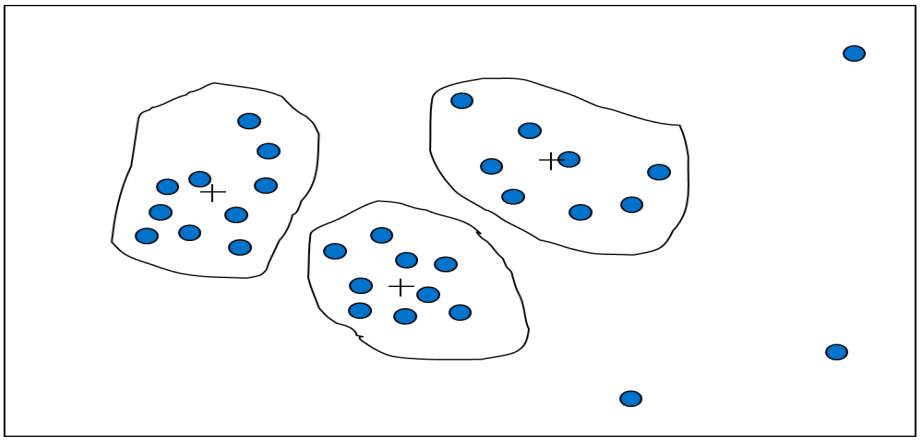
Χρησιμοποιώντας ομαλοποίηση με βάση τη μέση τιμή του κάθε δοχείου, **έχουμε:**

- Δοχείο 1: 10, 10, 10, 10
- Δοχείο 2: 23, 23, 23, 23
- Δοχείο 3: 31, 31, 31, 31

Χρησιμοποιώντας ομαλοποίηση με χρήση των ορίων του κάθε δοχείου, **έχουμε:**

- Δοχείο 1: 4, 4, 16, 16
- Δοχείο 2: 21, 24, 24, 24
- Δοχείο 3: 27, 27, 35, 35

Η χρήση συσταδοποίησης έχει ως στόχο την ομαδοποίηση των δεδομένων σε συστάδες (clusters), έτσι ώστε τα δεδομένα με θόρυβο να διαχωριστούν από τα καθαρά δεδομένα. Για παράδειγμα, στην Εικόνα 3.2 βλέπουμε ότι έχουν δημιουργηθεί 3 συστάδες και ότι οι ακραίες τιμές δεν ανήκουν σε καμία από αυτές.



Εικόνα *3.2* *Εφαρμογή συσταδοποίησης για ανίχνευση ακραίων τιμών.*

3.2.1.3 Ασυνεπή Δεδομένα

Ασυνέπεια στα δεδομένα έχουμε, όταν δυο ή περισσότερες διαφορετικές πηγές ή αρχεία έχουν διαφορετικές εκδόσεις αποθηκευμένων δεδομένων, τα οποία θα έπρεπε να είναι ίδια. Ασυνέπεια έχουμε, δηλαδή, όταν για την ίδια πραγματική οντότητα οι τιμές των χαρακτηριστικών από διαφορετικές πηγές διαφέρουν. Συνήθως, αυτό συμβαίνει όταν έχουμε πλεονασμό δεδομένων και χρειασθεί να γίνει κάποια αλλαγή. Τότε είναι πολύ πιθανό να γίνει διόρθωση μόνο σε κάποιο ή κάποια αρχεία και όχι σε όλα. Άλλη πιθανή αιτία είναι ο διαφορετικός τρόπος αναπαράστασης ή και η χρήση διαφορετικών κλιμάκων, π.χ. μονάδες μέτρησης, διαφορετικό νόμισμα. Για την επίλυση του προβλήματος των ασυνεπών δεδομένων μπορούμε να κάνουμε είτε χειρωνακτική διόρθωση, χρησιμοποιώντας εξωτερικές πηγές, είτε ημιαυτόματη διόρθωση, χρησιμοποιώντας εμπορικά εργαλεία καθαρισμού δεδομένων (data scrubbing tools) ή εργαλεία λογιστικού ελέγχου (data auditing tools).

3.2.2 Ενοποίηση Δεδομένων

Η ενοποίηση των δεδομένων έχει ως στόχο τον συνδυασμό δεδομένων από πολλαπλές πηγές σε μια συνεκτική έκδοση. Όταν τα δεδομένα είναι αποθηκευμένα σε βάσεις δεδομένων, πρέπει να υλοποιηθεί ενοποίηση σχήματος (schema integration) με χρήση των μεταδεδομένων που υπάρχουν από τις διάφορες πηγές. Κατά τη διαδικασία της ενοποίησης πρέπει να ανιχνευτούν και να αναλυθούν πιθανές συγκρούσεις ή ασυνέπειες μεταξύ των τιμών των δεδομένων.

Τα πλεονάζοντα δεδομένα εμφανίζονται συχνά, όταν συνενώνονται πολλαπλές βάσεις δεδομένων. Πιθανά προβλήματα, που μπορούν να προκύψουν στην προσπάθεια συνένωσης, είναι η χρήση διαφορετικού ονόματος σε διαφορετικές βάσεις δεδομένων ή όταν ένα γνώρισμα είναι παραγόμενο γνώρισμα σε άλλο πίνακα. Για να εντοπίσουμε τα πλεονάζοντα δεδομένα χρησιμοποιείται ανάλυση συσχετίσεων.

Τέλος, αξίζει να αναφέρουμε ότι με προσεκτική ενοποίηση μπορούν να αφαιρεθούν περιττές πληροφορίες και να αποφευχθούν ασυνέπειες, να βελτιωθεί σημαντικά η ταχύτητα της διαδικασίας εξόρυξης δεδομένων και να αυξηθεί η ποιότητα των αποτελεσμάτων της.

3.2.3 Μετασχηματισμός και Διακριτοποίηση Δεδομένων

Ο μετασχηματισμός των δεδομένων έχει ως βασικό στόχο τη δημιουργία συγκρίσιμων δεδομένων, τα οποία αρχικά είναι μη συγκρίσιμα. Με τον μετασχηματισμό των δεδομένων μπορούμε να πετύχουμε και άλλα θετικά αποτελέσματα, όπως μείωση του όγκου των δεδομένων, π.χ. με το μετασχηματισμό των τιμών ενός χαρακτηριστικού σε κάποιο υποσύνολο τους (πλήρης ημερομηνία/μόνο έτος), και μεγαλύτερη ακρίβεια των αποτελεσμάτων των αλγορίθμων εξόρυξης, π.χ. με το μετασχηματισμό των τιμών των χαρακτηριστικών σε κοινό εύρος (εύρος [0, 1]).

Η διακριτοποίηση μπορεί να θεωρηθεί ως μια ειδική μορφή μετασχηματισμού δεδομένων. Η βασική ιδέα είναι η μετατροπή ενός συνεχούς εύρους τιμών σε διακριτές τιμές ή ετικέτες. Θα δούμε παρακάτω ότι σε κάποιες περιπτώσεις, η διακριτοποίηση είναι αναγκαία για την εκτέλεση κάποιων τεχνικών εξόρυξης δεδομένων.

3.2.3.1 Μετασχηματισμός Δεδομένων

Ο μετασχηματισμός των δεδομένων χρησιμοποιείται κυρίως:

- για την εξομάλυνση των δεδομένων και την απομάκρυνση θορύβου,
- για τη συνάθροιση των δεδομένων, δηλαδή παραγωγή σύνοψης τους,
- για την κανονικοποίησή τους, δηλαδή την κλιμάκωση των χαρακτηριστικών του συνόλου δεδομένων σε ένα συγκεκριμένο και περιορισμένο εύρος τιμών, και
- για τη δημιουργία νέων χαρακτηριστικών από τα ήδη υπάρχοντα.

Η πιο συχνή εφαρμογή του μετασχηματισμού δεδομένων είναι η κανονικοποίηση και η δημιουργία νέων χαρακτηριστικών από τα ήδη υπάρχοντα. Η κανονικοποίηση είναι ιδιαίτερα χρήσιμη σε προβλήματα κατηγοριοποίησης, καθώς και όταν τα δεδομένα έχουν τελείως διαφορετικές κλίμακες και μονάδες μέτρησης. Υπάρχουν διαφορετικοί τρόποι κανονικοποίησης των δεδομένων. Οι πιο βασικοί είναι οι ακόλουθοι:

- Κανονικοποίηση min-max: οι τιμές κανονικοποιούνται, ώστε το εύρος τους να ανήκει σε ένα νέο, περιορισμένο εύρος, π.χ. [-1, 1], [0, 1] κοκ. Η νέα τιμή του χαρακτηριστικού σε κάθε πλειάδα υπολογίζεται χρησιμοποιώντας τον τύπο:

$$v_{new} = \frac{v - \min}{\max - \min}(\max_{new} - \min_{new}) + \min_{new}$$

- Κανονικοποίηση z-score: οι τιμές κανονικοποιούνται με χρήση της μέσης τιμής και της τυπικής απόκλισης, έτσι ώστε τα δεδομένα να έχουν μέση τιμή 0 και τυπική απόκλιση 1. Η κανονικοποίηση γίνεται με τον παρακάτω τύπο:

$$v_{new} = \frac{v - \mathrm{m}}{\mathrm{s}}$$

όπου μ είναι η μέση τιμή του χαρακτηριστικού και σ η τυπική του απόκλιση.

- Κανονικοποίηση με δεκαδική κλίμακα: οι τιμές κανονικοποιούνται με τάξεις μεγέθους του 10. Η κανονικοποίηση γίνεται με τον τύπο:

$$v_{new} = \frac{v}{10^j}$$

- όπου j είναι ο μικρότερος ακέραιος τέτοιος ώστε:

$$\max(v_{new}) < 1$$

Παράδειγμα - Κανονικοποίηση Δεδομένων
Έστω ότι δίνεται ένα πλαίσιο δεδομένων με ηλικίες και ύψη μαθητών. Θέλουμε να κανονικοποιήσουμε και τα δυο χαρακτηριστικά στο διάστημα [0, 1]. Ο Κώδικας 3.1 υλοποιεί σε R την παραπάνω διαδικασία.

```
> # Αρχικό σύνολο δεδομένων
> mydf
  age height
1  15    172
2  23    185
3  12    130
4  32    178
>
> # Εύρεση μέγιστου κάθε στήλης
> M <- sapply(mydf, max)
> M
   age height
    32    185
>
> # Εύρεση ελάχιστου κάθε στήλης
> m <- sapply(mydf, min)
> m
   age height
    12    130
>
> # Κανονικοποίηση σε νέο διάστημα [0, 1]
> mydf$age <-( (mydf$age - m[1])/(M[1] - m[1])
+             )* (1 - 0) + 0
> mydf$height <-( (mydf$height - m[2])/(M[2] - m[2])
+                ) * (1 - 0) + 0
>
> mydf
   age    height
1 0.15 0.7636364
2 0.55 1.0000000
3 0.00 0.0000000
4 1.00 0.8727273
>
```

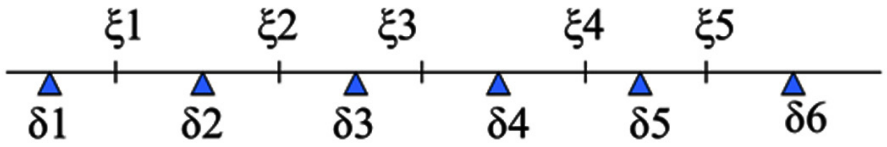
Κώδικας *3.1* *Παράδειγμα κανονικοποίησης min-max.*

3.2.3.2 Διακριτοποίηση Δεδομένων

Η διακριτοποίηση σχετίζεται με 3 τύπους χαρακτηριστικών:

- Ονομαστικά χαρακτηριστικά, όπου οι τιμές είναι ένα μη διατεταγμένο σύνολο.
- Διατακτικά χαρακτηριστικά, όπου οι τιμές είναι ένα διατεταγμένο σύνολο.
- Συνεχή χαρακτηριστικά, όπου οι τιμές είναι πραγματικοί αριθμοί.

Ένα παράδειγμα διακριτοποίησης είναι η δειγματοληψία από το εύρος ενός συνεχούς χαρακτηριστικού. Ο κύριος λόγος ύπαρξης της διακριτοποίησης είναι ότι κάποιοι αλγόριθμοι κατηγοριοποίησης δέχονται μόνο κατηγορικά χαρακτηριστικά. Ακόμα μπορεί να συντελέσει στη μείωση του αριθμού και συνεπώς του μεγέθους των δεδομένων. Για ένα δεδομένο συνεχές χαρακτηριστικό μπορούμε να διαχωρίσουμε το εύρος του σε διαστήματα και να αναθέσουμε ετικέτες στο κάθε διάστημα (Εικόνα 3.3). Για παράδειγμα, μια τιμή που ανήκει στο διάστημα [ξ1, ξ2) θα αντικατασταθεί από την ετικέτα δ2.



Εικόνα *3.3* *Παράδειγμα διακριτοποίησης.*

Όπως ήδη αναφέραμε, η ενδοχείαση (binning) μπορεί να χρησιμοποιηθεί για διακριτοποίηση. Μια ακόμα τεχνική διακριτοποίησης είναι η διακριτοποίηση με χρήση της εντροπίας. Έστω ένα σύνολο δειγμάτων S. Αν το S διαχωρίζεται σε δυο διαστήματα S_1 και S_2, χρησιμοποιώντας ένα κατώφλι T για τις τιμές του χαρακτηριστικού A, τότε το κέρδος πληροφορίας που προκύπτει από τον διαχωρισμό είναι:

$$I(S,T) = \frac{|S_1|}{|S|} E(S_1) + \frac{|S_2|}{|S|} E(S_2)$$

όπου η συνάρτηση εντροπίας E για ένα δεδομένο σύνολο υπολογίζεται με βάση την κατανομή της κλάσης των δειγμάτων στο σύνολο. Αν έχουμε m κλάσεις, η εντροπία για το διάστημα S_1 είναι:

$$E(S_1) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

όπου p_i είναι η πιθανότητα της κλάσης i στο S_1.

Η διαδικασία εφαρμόζεται αναδρομικά σε διαχωρισμούς, μέχρι να ικανοποιηθεί κάποιο κριτήριο τερματισμού, π.χ.

$$G(S,T) = E(S) - I(S,T) \le \delta$$

,

όπου δ είναι ένας αρκετά μικρός αριθμός. Με άλλα λόγια, η διαδικασία εφαρμόζεται αναδρομικά μέχρι να μην έχουμε ουσιαστικό κέρδος από περαιτέρω διαχωρισμούς. Πειράματα έχουν δείξει ότι η διακριτοποίηση μπορεί να μειώσει το μέγεθος των δεδομένων, βελτιώνοντας την ακρίβεια της κατηγοριοποίησης.

Παράδειγμα – Διακριτοποίηση βασισμένη στην εντροπία

Παρακάτω (Πίνακας 3.1) παρουσιάζουμε ένα σύνολο δεδομένων με τις ώρες μελέτης για την εξέταση ενός μαθήματος και το αν τελικά οι μαθητές πέτυχαν στην αντίστοιχη εξέταση του μαθήματος, δηλαδή αν πέρασαν το σχετικό μάθημα ή όχι (Ν = ΝΑΙ, Ο = ΟΧΙ).

Ώρες Μελέτης	**Επιτυχία Εξέτασης Μαθήματος**
4	Ο
5	Ν
8	Ο
12	Ν
15	Ν

Πίνακας *3.1* *Σύνολο δεδομένων για παράδειγμα διακριτοποίησης με χρήση εντροπίας*

Οι ώρες μελέτης είναι η συνεχής μεταβλητή. Θέλουμε να διακριτοποιήσουμε τα δεδομένα. Ξεκινάμε υπολογίζοντας την εντροπία του συνόλου δεδομένων. Για την επιτυχία εξέτασης μαθήματος έχουμε τρία Ν (ΝΑΙ), και δυο Ο (ΟΧΙ). Συνεπώς,

$$E(S) = -\left(\frac{3}{5}\log_2\left(\frac{3}{5}\right) + \frac{2}{5}\log_2\left(\frac{2}{5}\right)\right) = 0.529 + 0.442 = 0.971$$

Στη συνέχεια, θα πρέπει να βρούμε ποιος διαχωρισμός θα μας δώσει το μέγιστο κέρδος. Για να βρούμε έναν διαχωρισμό, υπολογίζουμε το ημιάθροισμα δυο γειτονικών τιμών. Για παράδειγμα, από τις δυο πρώτες τιμές έχουμε 5+4 = 9 και Τ= 9/2 = 4.5. Επομένως, ο πρώτος πιθανός διαχωρισμός είναι στο Τ = 4.5. Με βάση αυτό τον διαχωρισμό προκύπτουν οι τιμές που φαίνονται παρακάτω (Πίνακας 3.2).

	Επιτυχία Εξέτασης	**Αποτυχία Εξέτασης**
<= 4.5	0	1
> 4.5	3	1

Πίνακας *3.2* *Τιμές μετά τον πρώτο πιθανό διαχωρισμό στο Τ = 4.5..*

Υπολογίζουμε την εντροπία για κάθε περίπτωση και το κέρδος του συγκεκριμένου διαχωρισμού:

$$E(S_{\le 4.5}) = -\left(\frac{1}{1}\log_2(1) + 0\log_2(0)\right) = 0 + 0 = 0$$

$$E(S_{> 4.5}) = -\left(\frac{3}{4}\log_2\left(\frac{3}{4}\right) + \frac{1}{4}\log_2\left(\frac{1}{4}\right)\right) = 0.311 + 0.5 = 0.811$$

Επομένως, τώρα έχουμε:

$$I(S,4.5) = \frac{1}{5}(0) + \frac{4}{5}(0.811) = 0.6488$$

και το καθαρό κέρδος από τον διαχωρισμό είναι:

$$G(S,4.5) = E(S) - I(S,4.5) = 0.971 - 0.6488 = 0.322$$

Παίρνοντας τις δυο επόμενες διαδοχικές τιμές, έχουμε 6 + 8 = 13, και Τ = 13/2 = 6.5. Επομένως, ο δεύτερος

πιθανός διαχωρισμός είναι στο T = 6.5. Με βάση αυτό τον διαχωρισμό προκύπτουν οι τιμές που φαίνονται παρακάτω (Πίνακας 3.3).

	Επιτυχία Εξέτασης	**Αποτυχία Εξέτασης**
<= 6.5	1	1
> 6.5	2	1

Πίνακας *3.3* *Τιμές μετά τον δεύτερο πιθανό διαχωρισμό στο T = 6.5.*

Υπολογίζουμε την εντροπία για κάθε περίπτωση και το κέρδος του συγκεκριμένου διαχωρισμού:

$$E(S_{\leq 6.5}) = -\left(\frac{1}{2}\log_2\left(\frac{1}{2}\right) + \frac{1}{2}\log_2\left(\frac{1}{2}\right)\right) = 0.5 + 0.5 = 1$$

$$E(S_{>6.5}) = -\left(\frac{2}{3}\log_2\left(\frac{2}{3}\right) + \frac{1}{3}\log_2\left(\frac{1}{3}\right)\right) = 0.389 + 0.528 = 0.917$$

Επομένως, τώρα έχουμε:

$$I(S,6.5) = \frac{2}{5}(1) + \frac{3}{5}(0.917) = 0.95$$

και το καθαρό κέρδος από τον διαχωρισμό είναι:

$$G(S,6.5) = E(S) - I(S,6.5) = 0.971 - 0.95 = 0.021$$

Συνεχίζοντας το παράδειγμα, παίρνουμε τις δυο επόμενες διαδοχικές τιμές. Έχουμε 8 + 12 = 20, και T = 20/2 = 10. Επομένως, ο τρίτος πιθανός διαχωρισμός είναι στο T = 10. Με βάση αυτό τον διαχωρισμό προκύπτουν οι τιμές που φαίνονται παρακάτω (Πίνακας 3.4).

	Επιτυχία Εξέτασης	**Αποτυχία Εξέτασης**
<= 10	1	2
> 10	2	0

Πίνακας *3.4* *Τιμές μετά τον τρίτο πιθανό διαχωρισμό στο T = 10.*

Υπολογίζουμε την εντροπία για κάθε περίπτωση και το κέρδος του συγκεκριμένου διαχωρισμού:

$$E(S_{\leq 10}) = -\left(\frac{1}{3}\log_2\left(\frac{1}{3}\right) + \frac{2}{3}\log_2\left(\frac{2}{3}\right)\right) = 0.528 + 0.389 = 0.917$$

$$E(S_{>10}) = -\left(\frac{1}{1}\log_2(1) + 0\log_2(0)\right) = 0 + 0 = 0$$

Επομένως, τώρα έχουμε:

$$I(S,10) = \frac{2}{5}(0.) + \frac{3}{5}(0.917) = 0.55$$

και το καθαρό κέρδος από τον διαχωρισμό είναι:

$$G(S,10) = E(S) - I(S,10) = 0.971 - 0.55 = 0.421$$

Τέλος, παίρνουμε τις δυο τελευταίες διαδοχικές τιμές. Έχουμε 12 + 15 = 27, και T = 27/2 = 13.5. Επομένως, ο τέταρτος και τελευταίος πιθανός διαχωρισμός είναι στο T = 13.5. Με βάση αυτό τον διαχωρισμό προκύπτουν οι τιμές που φαίνονται παρακάτω (Πίνακας 3.5).

	Επιτυχία Εξέτασης	**Αποτυχία Εξέτασης**
<= 13.5	2	2
> 13.5	1	0

Πίνακας *3.5* *Τιμές μετά τον τέταρτο πιθανό διαχωρισμό στο T = 13.*

Υπολογίζουμε την εντροπία για κάθε περίπτωση και το κέρδος του συγκεκριμένου διαχωρισμού:

$$E(S_{\leq 13.5}) = -\left(\frac{2}{4}\log_2\left(\frac{2}{4}\right) + \frac{2}{4}\log_2\left(\frac{2}{4}\right)\right) = 0.5 + 0.5 = 1$$

$$E(S_{> 13.5}) = -\left(\frac{1}{1}\log_2(1) + 0\log_2(0)\right) = 0 + 0 = 0$$

Επομένως, τώρα έχουμε:

$$I(S,13.5) = \frac{1}{5}(0) + \frac{4}{5}(1) = 0.8$$

και το καθαρό κέρδος από τον διαχωρισμό είναι:

$$G(S,13.5) = E(S) - I(S,13.5) = 0.971 - 0.8 = 0.2$$

Από τα παραπάνω συμπεραίνουμε ότι ο τρίτος πιθανός διαχωρισμός, στο T = 10, είναι καλύτερος με το μέγιστο κέρδος (0.421). Μετά τον διαχωρισμό μπορούμε να συνεχίσουμε, εξετάζοντας νέες περιπτώσεις διαχωρισμού και επιλέγοντας πάλι τον καλύτερο. Η διαδικασία μπορεί να συνεχιστεί μέχρι να μην έχουμε κέρδος από περαιτέρω διαχωρισμούς, με βάση κάποια μικρή τιμή για το δ.

3.2.4 Μείωση Δεδομένων

Το πρόβλημα που προσπαθεί να αντιμετωπίσει η μείωση των δεδομένων είναι ο τεράστιος όγκος δεδομένων προς επεξεργασία, καθώς η ανάλυση σύνθετων δεδομένων ενδεχομένως να απαιτεί απαγορευτικά πολύ χρόνο για να εκτελεστεί σε ολόκληρο το σύνολο δεδομένων.

Η διαδικασία της μείωσης των δεδομένων έχει ως στόχο την παραγωγή μιας μειωμένης αναπαράστασης του συνόλου δεδομένων, η οποία είναι αρκετά μικρότερη σε μέγεθος, αλλά που να μπορεί να παράγει ίδια ή παραπλήσια αποτελέσματα.

3.2.4.1 Μείωση Διαστάσεων

Όσο περισσότερες διαστάσεις έχουμε, τόσο πιο δύσκολη είναι η διαχείριση των δεδομένων και τόσο πιο αραιά (sparse) είναι τα δεδομένα μας. Το τελευταίο φαινόμενο είναι γνωστό στη βιβλιογραφία ως η κατάρα της διαστατικότητας (curse of dimensionality). Η μείωση των διαστάσεων έχει ως στόχο την ευκολότερη διαχείριση, κατανόηση και οπτικοποίηση των δεδομένων, ενώ ταυτόχρονα μειώνει τις απαιτήσεις σε χώρο μνήμης και σε χρόνο εκτέλεσης των αλγορίθμων εξόρυξης δεδομένων και μηχανικής μάθησης. Δυο βασικές προσεγγίσεις, με τις οποίες μπορεί να επιτευχθεί μείωση των διαστάσεων, είναι η επιλογή χαρακτηριστικών και ο μετασχηματισμός των δεδομένων.

Με βάση την προσέγγιση επιλογής χαρακτηριστικών επιλέγουμε το ελάχιστο πλήθος χαρακτηριστικών, με τα οποία είναι εφικτό να παραχθούν ισοδύναμα ή όσο το δυνατόν κοντινότερα αποτελέσματα με αυτά που θα παίρναμε, αν χρησιμοποιούσαμε όλα τα χαρακτηριστικά για ανάλυση. Ιδανικά, ο αριθμός των χαρακτηριστικών που επιλέγονται είναι πολύ μικρότερος από τον αριθμό των αρχικών χαρακτηριστικών.

Ο πιο γνωστός μετασχηματισμός χαρακτηριστικών για μείωση των διαστάσεων είναι η Ανάλυση Βασικών Συνιστωσών (Principal Component Analysis, PCA). Ο μετασχηματισμός των χαρακτηριστικών δημιουργεί ένα νέο σύνολο χαρακτηριστικών, λιγότερων διαστάσεων από το αρχικό, αλλά χωρίς μείωση των βασικών διαστάσεων. Συχνά η PCA χρησιμοποιείται και για την οπτικοποίηση των δεδομένων.

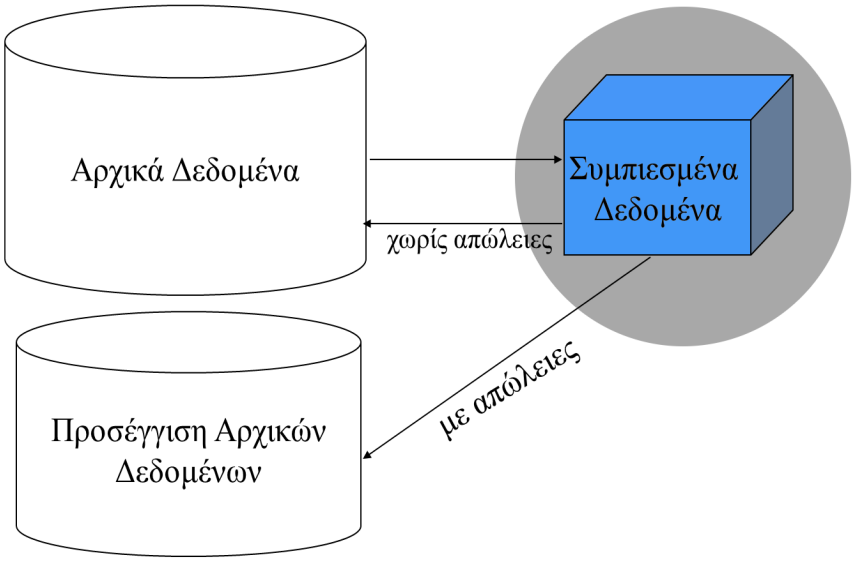
Η Ανάλυση Βασικών Συνιστωσών λειτουργεί ως εξής: Έχοντας N διανύσματα k-διαστάσεων βρίσκει $m \leq k$ ορθογώνια διανύσματα[2], τα οποία μπορούν να χρησιμοποιηθούν για τη βέλτιστη αναπαράσταση των δεδομένων. Έτσι, το αρχικό σύνολο δεδομένων μειώνεται, ουσιαστικά προβάλλεται, σε ένα νέο, το οποίο αποτελείται από N διανύσματα δεδομένων πάνω σε m βασικές συνιστώσες. Κάθε διάνυσμα δεδομένων είναι γραμμικός συνδυασμός των m διανυσμάτων βασικών συνιστωσών. Αυτή η τεχνική μπορεί να χρησιμοποιηθεί και με διατεταγμένα και με μη διατεταγμένα χαρακτηριστικά, ενώ χρησιμοποιείται κυρίως, όταν ο αριθμός των διαστάσεων είναι μεγάλος.

3.2.4.2 Συμπίεση Δεδομένων

Μια ακόμα επιλογή για τη μείωση των δεδομένων είναι η συμπίεση τους. Συμπίεση μπορούμε να κάνουμε σε διάφορες μορφές δεδομένων, όπως για παράδειγμα, σε αλφαριθμητικά. Εδώ υπάρχουν εκτενείς θεωρίες και αλγόριθμοι, ενώ συνήθως δεν έχουμε απώλεια πληροφορίας. Ωστόσο, εισάγονται περιορισμοί ως προς τη διαχείριση. Επίσης σε βίντεο, ήχο και εικόνα, όπου στις περισσότερες περιπτώσεις έχουμε απώλεια πληροφορίας κατά τη συμπίεση. Τέλος, σε χρονικές ακολουθίες, που δεν είναι ήχος, έχουν σύντομη διάρκεια και μεταβάλλονται αργά στον χρόνο.

Στην Εικόνα 3.4 φαίνονται οι δυο κατηγορίες συμπίεσης: με απώλειες και χωρίς. Στόχος είναι η μείωση των δεδομένων και η χρήση μιας προσέγγισης των αρχικών δεδομένων, που όμως θα δώσει όσο το δυνατόν πλησιέστερα αποτελέσματα με αυτά που θα παίρναμε, αν χρησιμοποιούσαμε τα αρχικά δεδομένα.

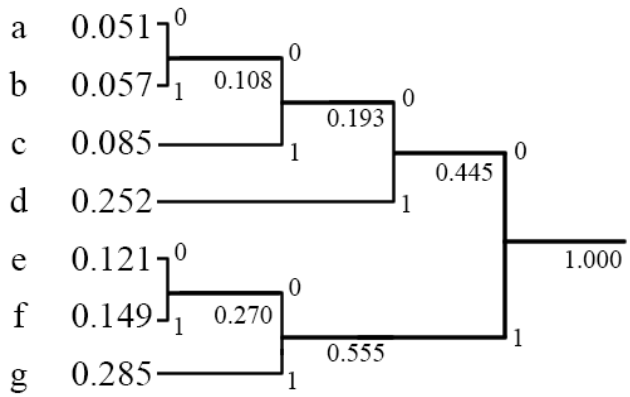
2 Δύο διανύσματα x, y καλούνται ορθογώνια όταν το εσωτερικό τους γινόμενο είναι ίσο με 0, δηλαδή όταν $x^T y = 0$.



Εικόνα *3.4* *Διαδικασία συμπίεσης δεδομένων.*

Μια τεχνική συμπίεσης είναι η κωδικοποίηση Huffman. Πρόκειται για έναν αλγόριθμο συμπίεσης χωρίς απώλειες. Ένας κωδικοποιητής Huffman παίρνει ως είσοδο χαρακτήρες προκαθορισμένου μήκους και παράγει ένα μπλοκ από δυαδικά ψηφία μεταβλητού μήκους στην έξοδο, είναι δηλαδή κωδικοποίηση από σταθερό σε μεταβλητό μήκος. Ο σχεδιασμός της κωδικοποίησης Huffman είναι βέλτιστος με την προϋπόθεση ότι τα στατιστικά της πηγής είναι γνωστά από πριν (a priori). Κατασκευάζεται με τη συγχώνευση των δυο λιγότερο πιθανών χαρακτήρων και η διαδικασία επαναλαμβάνεται, μέχρι να απομείνει μόνο ένας χαρακτήρας.

x	p(x)
a	0.051
b	0.057
c	0.085
d	0.252
e	0.121
f	0.149
g	0.285



x	code
a	0000
b	0001
c	001
d	01
e	100
f	101
g	11

Εικόνα *3.5* *Παράδειγμα κωδικοποίησης Huffman.*

Στο παράδειγμα στην Εικόνα 3.5, η διάταξη των χαρακτήρων δεν παίζει κανένα ρόλο, ούτε ο τρόπος με τον οποίο τοποθετούνται οι ετικέτες 0 και 1 στο τελικό δέντρο κώδικα. Για το συγκεκριμένο παράδειγμα και για λόγους ευανάγνωστης αναπαράστασης του δέντρου, οι πάνω κλάδοι του δέντρου έχουν ετικέτα 0, ενώ οι κάτω κλάδοι έχουν ετικέτα 1. Σε περίπτωση ισοπαλίας μεταξύ των δυο λιγότερο πιθανών χαρακτήρων, οποιοσδήποτε μηχανισμός είναι αποδεκτός για την επίλυση της ισοπαλίας. Τέλος, η κωδικοποίηση Huffman δεν είναι μοναδική, δηλαδή για κάποιο στιγμιότυπο μπορούμε να δημιουργήσουμε διαφορετικές κωδικοποιήσεις, ανάλογα και με τις υποθέσεις που θα κάνουμε.

Ακόμα μια τεχνική συμπίεσης χωρίς απώλειες, είναι η κωδικοποίηση Lempel-Ziv. Σε αντίθεση με την κωδικοποίηση Huffman, πρόκειται για μια κωδικοποίηση από μεταβλητό σε σταθερό μήκος. Ο αλγόριθμος για την παραγωγή της κωδικοποίησης αποτελείται από τα εξής βήματα:

1. Αρχικοποίησε ένα λεξικό με όλα τα μπλοκ μήκους ένα (D = {a, b}).
2. Αναζήτησε το μεγαλύτερο σε μήκος μπλοκ W, το οποίο εμφανίζεται στο λεξικό D.
3. Κωδικοποίησε το μπλοκ W με χρήση του δείκτη θέσης του στο λεξικό D.
4. Πρόσθεσε το μπλοκ W, ακολουθούμενο από το πρώτο σύμβολο του επόμενου μπλοκ, στο λεξικό D.
5. Πήγαινε στο βήμα 2.

Data: a b b a a b a b b a b a a b b b a b a a

0 1 1 0 2 2 4 4 7 8 5

Λεξικό			
Δείκτης	Εγγραφή	Δείκτης	Εγγραφή
0	a	6	a b a
1	b	7	a b b
2	a b	8	b a b
3	b b	9	b a a
4	b a	10	a b b b
5	a a	11	b a b a

Εικόνα *3.6* *Παράδειγμα κωδικοποίησης Lempel-Ziv.*

Η Εικόνα 3.6 παρουσιάζει ένα παράδειγμα κωδικοποίησης Lempel-Ziv. Στο παράδειγμα αυτό, το λεξικό έχει αρχικά μόνο τα μπλοκ μήκους ένα, δηλαδή D = {0: a, 1: b} (βήμα 1). Ξεκινάμε το σάρωμα της συμβολοσειράς. Το μεγαλύτερο μπλοκ της συμβολοσειράς που υπάρχει ήδη στο λεξικό είναι το a, αφού π.χ. το ab δεν υπάρχει ακόμα στο λεξικό (βήμα 2). Επομένως, το μπλοκ κωδικοποιείται με τον αντίστοιχο δείκτη, δηλαδή το 0 (βήμα 3). Έπειτα προσθέτουμε στο λεξικό το μπλοκ που κωδικοποιήσαμε (a) ακολουθούμενο από το αμέσως επόμενο μπλοκ (b). Δηλαδή, το λεξικό γίνεται D = {0: a, 1: b, 2: ab} (βήμα 4). Η διαδικασία επαναλαμβάνεται από το βήμα 2, μέχρις ότου να μην υπάρχει άλλο μπλόκ στη συμβολοσειρά.

Θεωρητικά, το μέγεθος του λεξικού D μπορεί να αυξάνεται επ' άπειρον. Πρακτικά όμως, υπάρχει περιορισμός για το μέγεθός του. Πιο συγκεκριμένα, αν το λεξικό φτάσει σε ένα προκαθορισμένο μέγεθος, τότε δεν γίνονται άλλες εισαγωγές. Το παράδειγμα στην Εικόνα 3.6, όπως και τα περισσότερα παραδείγματα στη βιβλιογραφία δεν καταλήγουν σε πραγματική συμπίεση των δεδομένων. Ουσιαστικά, χρησιμοποιούνται περισσότερα δυαδικά ψηφία για την αναπαράσταση των δεικτών θέσης απ' ό, τι τα αρχικά δεδομένα. Αυτό οφείλεται στο ότι το μήκος της εισόδου είναι πολύ μικρό. Στην πράξη, ο αλγόριθμος λειτουργεί καλά και οδηγεί σε πραγματική συμπίεση, υπό την προϋπόθεση ότι το μήκος εισόδου είναι αρκετά μεγάλο και ότι υπάρχει αρκετός πλεονασμός στα δεδομένα.

3.3 Πακέτα dplyr και tidyr

3.3.1 dplyr

Το πακέτο dplyr χρησιμοποιείται για τον εύκολο χειρισμό των δεδομένων. Αναπτύχθηκε από τους Hadley Wickham και Roman Francois και παρέχει έτοιμες συναρτήσεις για συνεπή και περιεκτική διαχείριση δεδομένων σε μορφή πινάκων. Η εγκατάσταση του πακέτου γίνεται με την εντολή `install.packages(dplyr)`, ενώ η φόρτωση του με την εντολή `library(dplyr)`.

Το πρώτο βήμα για τη χρήση του πακέτου dplyr είναι η μετατροπή των δεδομένων σε συμβατή μορφή με το πακέτο. Αυτό γίνεται εύκολα, καλώντας τη συνάρτηση `tbl_df` και δίνοντας ως όρισμα το αντικείμενο. Το βασικό πλεονέκτημα χρήσης της `tbl_df` είναι ότι κάνει την αναπαράσταση κατά την εκτύπωση πιο συμπαγή και ευανάγνωστη.

Δραστηριότητα: Εκτελέστε το κομμάτι κώδικα που δίνεται (Κώδικας 3.2). Εκτυπώστε το περιεχόμενο του αρχικού πλαισίου δεδομένων *airquality*, *το οποίο είναι ένα από τα έτοιμα σύνολα δεδομένων που παρέχονται από την* R. Τι διαφορές παρατηρείτε, ως προς τον τρόπο εκτύπωσης, σε σχέση με το νέο, `tbl_df`, αντικείμενο;

```
> library(dplyr)
> data(airquality)
> class(airquality)
[1] "data.frame"
> airquality <- tbl_df(airquality)
> class(airquality)
[1] "tbl_df"     "tbl"        "data.frame"
> airquality
Source: local data frame [153 x 6]

   Ozone Solar.R Wind Temp Month Day
1     41     190  7.4   67     5   1
2     36     118  8.0   72     5   2
3     12     149 12.6   74     5   3
4     18     313 11.5   62     5   4
5     NA      NA 14.3   56     5   5
6     28      NA 14.9   66     5   6
7     23     299  8.6   65     5   7
8     19      99 13.8   59     5   8
9      8      19 20.1   61     5   9
10    NA     194  8.6   69     5  10
..   ...     ...  ...  ...   ... ...
>
```

Κώδικας ***3.2*** *Φόρτωση πακέτου και μετατροπή δεδομένων σε συμβατή μορφή.*

Το πακέτο `dplyr` παρέχει 5 συναρτήσεις, οι οποίες καλύπτουν τις θεμελιώδεις εργασίες διαχείρισης δεδομένων. Αυτές είναι οι:

- `select`, για επιλογή-φιλτράρισμα στηλών του συνόλου δεδομένων,
- `filter`, για επιλογή-φιλτράρισμα γραμμών του συνόλου δεδομένων,
- `arrange`, για ταξινόμηση των γραμμών βάσει τιμών συγκεκριμένων στηλών,
- `mutate`, για δημιουργία νέων μεταβλητών από τις ήδη υπάρχουσες,
- `summarize`, για συνάθροιση δεδομένων – ιδιαίτερα χρήσιμη σε συνδυασμό με ομαδοποιημένα δεδομένα.

Σε αρκετές περιπτώσεις, κυρίως όταν το σύνολο δεδομένων είναι μεγάλο σε μέγεθος, μας ενδιαφέρει μόνο ένα υποσύνολο των χαρακτηριστικών (features) του συνόλου δεδομένων. Η συνάρτηση `select` μας επιτρέπει να επιλέξουμε συγκεκριμένες στήλες του συνόλου δεδομένων. Αρκεί να δώσουμε τα ονόματα των στηλών και η `select` θα μας επιστρέψει τις στήλες με τη σειρά που ορίσαμε.

```
> select(airquality, Ozone, Solar.R, Day)
Source: local data frame [153 x 3]


   Ozone Solar.R Day
1     41     190   1
2     36     118   2
3     12     149   3
4     18     313   4
5     NA      NA   5
6     28      NA   6
7     23     299   7
8     19      99   8
9      8      19   9
10    NA     194  10
..   ...     ... ...
>
```

Κώδικας *3.3* *Επιλογή στηλών από σύνολο δεδομένων με χρήση της συνάρτησης select.*

Επιπλέον, μπορούμε να επιλέξουμε πολλαπλές στήλες με χρήση του τελεστή ":", να επιλέξουμε ποια στήλη θέλουμε να παραλείψουμε με χρήση του "–" μπροστά από τα ονόματα στηλών ή να παραλείψουμε πολλαπλές στήλες με συνδυασμό των προηγούμενων τελεστών.

```
> select(airquality, -(Wind:Month))
Source: local data frame [153 x 3]


   Ozone Solar.R Day
1     41     190   1
2     36     118   2
3     12     149   3
4     18     313   4
```

```
5      NA       NA   5
6      28       NA   6
7      23      299   7
8      19       99   8
9       8       19   9
10     NA      194  10
..    ...      ... ...
>
```

Κώδικας *3.4* *Παράλειψη στηλών από σύνολο δεδομένων με χρήση της συνάρτησης select.*

Παρατηρείστε ότι οι δυο προηγούμενες κλήσεις της `select` είναι ισοδύναμες, δηλαδή επιστρέφουν το ίδιο σύνολο στηλών του συνόλου δεδομένων.

Αντίστοιχα, για το φιλτράρισμα γραμμών υπάρχει η συνάρτηση `filter`. Η διαφορά είναι ότι στο φιλτράρισμα γραμμών, ως δεύτερο όρισμα, πρέπει να δώσουμε μια συνθήκη πάνω στις στήλες. Η συνάρτηση θα επιστρέψει τις γραμμές που ικανοποιούν αυτή τη συνθήκη. Μπορούμε να ορίσουμε πολλαπλές συνθήκες, οι οποίες θέλουμε να ικανοποιούνται ταυτόχρονα (λογικό AND), χωρίζοντας τες με κόμμα (Κώδικας 3.5).

```
> filter(airquality, Month > 5, Month < 9, Day < 3)
Source: local data frame [6 x 6]
  Ozone Solar.R Wind Temp Month Day
1    NA     286  8.6   78     6   1
2    NA     287  9.7   74     6   2
3   135     269  4.1   84     7   1
4    49     248  9.2   85     7   2
5    39      83  6.9   81     8   1
6     9      24 13.8   81     8   2
>
```

Κώδικας *3.5* *Επιλογή γραμμών από σύνολο δεδομένων με χρήση της συνάρτησης filter.*

Σε περίπτωση που μας αρκεί οι γραμμές να ικανοποιούν μία από δύο συνθήκες, μπορούμε να χρησιμοποιήσουμε τον τελεστή "|" (λογικό OR) (Κώδικας 3.6). Γενικά, μπορούμε να χρησιμοποιήσουμε όλους του συγκριτικούς τελεστές με αριθμητικά δεδομένα.

```
> filter(airquality, Day == 1 | Day == 2)
Source: local data frame [10 x 6]
   Ozone Solar.R Wind Temp Month Day
1     41     190  7.4   67     5   1
2     36     118  8.0   72     5   2
3     NA     286  8.6   78     6   1
4     NA     287  9.7   74     6   2
5    135     269  4.1   84     7   1
6     49     248  9.2   85     7   2
```

```
7      39      83  6.9   81      8   1
8       9      24 13.8   81      8   2
9      96     167  6.9   91      9   1
10     78     197  5.1   92      9   2
>
```

Κώδικας *3.6* *Επιλογή γραμμών με λογική συνθήκη OR από σύνολο δεδομένων με χρήση της συνάρτησης filter.*

Για την ταξινόμηση των γραμμών με βάση την τιμή συγκεκριμένων στηλών χρησιμοποιούμε τη συνάρτηση `arrange`. Η συνάρτηση ταξινομεί τις γραμμές με βάση τη σειρά που θα δώσουμε τα ονόματα των στηλών ως ορίσματα. Η προεπιλεγμένη ταξινόμηση είναι κατά αύξουσα σειρά. Αν θέλουμε η ταξινόμηση να γίνει κατά φθίνουσα σειρά, θα πρέπει να το ορίσουμε ρητά, δίνοντας το όνομα της αντίστοιχης στήλης στη συνάρτηση `desc` (Κώδικας 3.7).

```
> arrange(airquality, Ozone, desc(Solar.R))
Source: local data frame [153 x 6]
   Ozone Solar.R Wind Temp Month Day
1      1       8  9.7   59      5  21
2      4      25  9.7   61      5  23
3      6      78 18.4   57      5  18
4      7      49 10.3   69      9  24
5      7      48 14.3   80      7  15
6      7      NA  6.9   74      5  11
7      8      19 20.1   61      5   9
8      9      36 14.3   72      8  22
9      9      24 13.8   81      8   2
10     9      24 10.9   71      9  14
..   ...     ...  ...  ...    ... ...
>
```

Κώδικας *3.7* *Ταξινόμηση γραμμών με χρήση της συνάρτησης arrange.*

Χρησιμοποιώντας τη συνάρτηση `mutate`, μπορούμε να δημιουργήσουμε νέες μεταβλητές-χαρακτηριστικά από τις ήδη υπάρχουσες. Η συγκεκριμένη συνάρτηση είναι ιδιαίτερα χρήσιμη, για παράδειγμα, όταν θέλουμε να κάνουμε μετατροπή μονάδων μέτρησης. Μπορούμε να δημιουργήσουμε πολλές νέες μεταβλητές με μόνο μια κλήση της συνάρτησης. Ένα πολύ χρήσιμο χαρακτηριστικό της συγκεκριμένης συνάρτησης είναι ότι μπορούμε να δώσουμε δικά μας ονόματα στις νέες μεταβλητές και να τα χρησιμοποιήσουμε άμεσα στην ίδια κλήση της συνάρτησης, για δημιουργία κι άλλων μεταβλητών (Κώδικας 3.8). Ουσιαστικά, το νέο χαρακτηριστικό δεν είναι παρά η μετατροπή του χαρακτηριστικού θερμοκρασίας `Temp` από Fahrenheit σε βαθμούς Κελσίου.

```
> mutate(airquality, Temp.C = round((Temp - 32)*5/9))
Source: local data frame [153 x 7]
   Ozone Solar.R Wind Temp Month Day Temp.C
1     41     190  7.4   67      5   1     19
2     36     118  8.0   72      5   2     22
```

```
3      12      149 12.6   74      5   3      23
4      18      313 11.5   62      5   4      17
5      NA       NA 14.3   56      5   5      13
6      28       NA 14.9   66      5   6      19
7      23      299  8.6   65      5   7      18
8      19       99 13.8   59      5   8      15
9       8       19 20.1   61      5   9      16
10     NA      194  8.6   69      5  10      21
..    ...      ...  ...   ...    ... ...     ...
>
```

Κώδικας *3.8* *Δημιουργία νέων μεταβλητών με χρήση της συνάρτησης mutate.*

Τέλος, η συνάρτηση summarize χρησιμοποιείται για συναθροιστικές λειτουργίες. Είναι ιδιαίτερα χρήσιμη, όταν δουλεύουμε με ομαδοποιημένα, βάσει τιμών, δεδομένα (Κώδικας 3.9).

```
> # Αφαίρεση γραμμών με ελλιπείς τιμές στο χαρακτηριστικό Ozone
> airquality <- filter(airquality, !is.na(Ozone))
> # Ομαδοποίηση κατά μήνα
> by_month <- group_by(airquality, Month)
> # Εύρεση ελάχιστης, μέσης και μέγιστης τιμής κατά μήνα
> summarize(by_month, min(Ozone), mean(Ozone), max(Ozone))
Source: local data frame [5 x 4]
  Month min(Ozone) mean(Ozone) max(Ozone)
1     5          1    23.61538        115
2     6         12    29.44444         71
3     7          7    59.11538        135
4     8          9    59.96154        168
5     9          7    31.44828         96
>
```

Κώδικας *3.9* *Χρήση της συνάρτησης summarize.*

3.3.2 `tidyr`

Το πακέτο tidyr αναπτύχθηκε από τον Hadley Wickham και χρησιμοποιείται για την εύκολη διαχείριση κατά τον καθαρισμό των δεδομένων, δηλαδή τον μετασχηματισμό των δεδομένων σε κατάλληλη μορφή ώστε να είναι κατάλληλα προς χρήση. Η εγκατάσταση του πακέτου γίνεται με την εντολή `install.packages(tidyr)`, ενώ η φόρτωση του με την εντολή `library(tidyr)`. Τα «καθαρά» δεδομένα πρέπει να πληρούν κάποιες συνθήκες, οι οποίες διευκολύνουν την εξερεύνηση και ανάλυση τους. Οι 3 θεμελιώδεις συνθήκες, οι οποίες πρέπει να ικανοποιούνται, είναι:

1. Κάθε μεταβλητή σχηματίζει μια στήλη στο σύνολο δεδομένων.
2. Κάθε παρατήρηση-μέτρηση σχηματίζει μια γραμμή στο σύνολο δεδομένων.

3. Κάθε μονάδα μέτρησης, που προκύπτει από τις εγγραφές των δεδομένων, σχηματίζει έναν ξεχωριστό πίνακα.

Η πρώτη προβληματική περίπτωση είναι όταν τα ονόματα στηλών είναι τιμές και όχι ονόματα μεταβλητών. Για την αντιμετώπιση αυτού του προβλήματος χρησιμοποιούμε τη συνάρτηση `gather`. Η συνάρτηση παίρνει ως ορίσματα ονόματα στηλών και τα συγκεντρώνει σε ζεύγη κλειδί-τιμή.

Το αρχικό σύνολο δεδομένων έχει ουσιαστικά 3 μεταβλητές: τον βαθμό, το φύλο και το πλήθος μαθητών (Κώδικας 3.10). Οι τιμές για το χαρακτηριστικό του φύλου των μαθητών εμφανίζονται ως ονόματα της δεύτερης και της τρίτης στήλης του συνόλου δεδομένων. Η τρίτη μεταβλητή είναι ο αριθμός των μαθητών για κάθε συνδυασμό βαθμού-φύλου.

```
> dat
Source: local data frame [3 x 3]
  grade male female
1     A    9     15
2     B   20     23
3     Γ   16     14
>
```

Κώδικας *3.10* *Σύνολο δεδομένων με τιμές μεταβλητών ως ονόματα στηλών πλαισίου δεδομένων.*

Για να καθαρίσουμε τα δεδομένα πρέπει κάθε μεταβλητή να βρίσκεται σε ξεχωριστή στήλη. Για τον σκοπό αυτό χρησιμοποιούμε τη συνάρτηση `gather`. Θέλουμε να ενοποιήσουμε τα δεδομένα ως προς το φύλο και το πλήθος, αφήνοντας τη στήλη του βαθμού απείραχτη. Για τον λόγο αυτό χρησιμοποιούμε τον τελεστή "–" μπροστά από τη μεταβλητή `grade` (Κώδικας 3.11).

```
> gather(dat, sex, count, -grade)
Source: local data frame [6 x 3]
  grade    sex count
1     A   male     9
2     B   male    20
3     Γ   male    16
4     A female    15
5     B female    23
6     Γ female    14
>
```

Κώδικας *3.11* *Χρήση της συνάρτησης gather.*

Η δεύτερη προβληματική περίπτωση που μπορεί να συναντήσουμε είναι όταν πολλαπλές μεταβλητές αποθηκεύονται σε μια στήλη. Σε αυτή την περίπτωση θα πρέπει να συνδυάσουμε τις συναρτήσεις `gather` και `separate`. Η συνάρτηση `separate` μετατρέπει μια στήλη σε πολλές βάσει ενός μοτίβου.

```
> dat
Source: local data frame [3 x 5]
  grade male_i male_ii female_i female_ii
1     A      6       3        8         7
2     B     13       7       16         7
3     Γ      8       8        9         5
>
```

Κώδικας *3.12* *Προβληματική περίπτωση συνόλου δεδομένων: αποθήκευση πολλαπλών μεταβλητών σε μια στήλη.*

Το σύνολο δεδομένων (Κώδικας 3.12) είναι παρόμοιο με αυτό που είδαμε προηγουμένως. Σε αυτή την περίπτωση έχουμε 2 διαφορετικά τμήματα μαθητών, `i` και `ii`, με το πλήθος των μαθητών για κάθε φύλο σε καθένα από τα τμήματα. Παρατηρούμε ότι έχουμε πολλαπλές μεταβλητές σε κάθε στήλη. Για τον καθαρισμό του συνόλου δεδομένων σε αυτή την περίπτωση πρέπει να γίνουν δυο ενέργειες.

Αρχικά με χρήση της `gather` συγκεντρώνουμε τα δεδομένα ως προς τη μεταβλητή που δηλώνει το φύλο και το τμήμα, και ως προς το πλήθος μαθητών (Κώδικας 3.13).

```
> dat <- gather(dat, sex_class, count, -grade)
> dat
Source: local data frame [12 x 3]
   grade sex_class count
1      A    male_i     6
2      B    male_i    13
3      Γ    male_i     8
4      A   male_ii     3
5      B   male_ii     7
6      Γ   male_ii     8
7      A  female_i     8
8      B  female_i    16
9      Γ  female_i     9
10     A female_ii     7
11     B female_ii     7
12     Γ female_ii     5
>
```

Κώδικας *3.13* *Συγκέντρωση δεδομένων ως προς φύλο-τμήμα και πλήθος των μαθημάτων.*

Στη συνέχεια, με χρήση της `separate` διαχωρίζουμε τη στήλη `sex_class` σε δυο διαφορετικές (Κώδικας 3.14). Για τη συγκεκριμένη περίπτωση, η συνάρτηση μπόρεσε από μόνη της να εντοπίσει τον χαρακτήρα διαχωρισμού.

```
  > separate(dat, sex_class, c("sex", "class"))Source: local data
frame [12 x 4]

   grade    sex class count
1      A   male     i     6
2      B   male     i    13
3      Γ   male     i     8
4      A   male    ii     3
5      B   male    ii     7
6      Γ   male    ii     8
7      A female     i     8
8      B female     i    16
9      Γ female     i     9
10     A female    ii     7
11     B female    ii     7
12     Γ female    ii     5
>
```

Κώδικας *3.14* *Χρήση της συνάρτησης separate για διαχωρισμό στήλης.*

Μια τρίτη περίπτωση προβληματικών δεδομένων είναι όταν μεταβλητές αποθηκεύονται και σε γραμμές και σε στήλες. Σε αυτή την περίπτωση θα πρέπει να συνδυάσουμε τις συναρτήσεις `gather` και `spread`. Η συνάρτηση `spread` υλοποιεί την αντίθετη λειτουργία της `gather`. Μετατρέπει ζεύγη κλειδιού-τιμής σε πολλαπλές στήλες. Έστω ένα σύνολο δεδομένων, στο οποίο μεταβλητές αποθηκεύονται τόσο σε γραμμές όσο και σε στήλες (Κώδικας 3.15). Πιο συγκεκριμένα, η πρώτη μεταβλητή είναι τα ονόματα (name) των μαθητών. Τα ονόματα των τεσσάρων τελευταίων στηλών αποτελούν τιμές για τη μεταβλητή μάθημα (lesson). Οι τιμές της μεταβλητής τριμήνου (quarter) θα πρέπει να καταχωρηθούν σε διαφορετικές μεταβλητές με τον αντίστοιχο βαθμό για κάθε μαθητή.

```
> dat
Source: local data frame [9 x 6]

        name quarter lesson1 lesson2 lesson3 lesson4
1     Κώστας       1       A      NA       Γ      NA
2     Κώστας       2       A      NA       B      NA
3     Κώστας       3       A      NA       B      NA
4      Μαρία       1       B       B      NA      NA
5      Μαρία       2       Γ       A      NA      NA
6      Μαρία       3       A       A      NA      NA
7 Παναγιώτης       1       Γ       B       A      NA
8 Παναγιώτης       2       B       B       B      NA
9 Παναγιώτης       3       A       A       B      NA
>
```

Κώδικας *3.15* *Προβληματική περίπτωση συνόλου δεδομένων: αποθήκευση πολλαπλών μεταβλητών σε μια στήλη.*

Αρχικά, συγκεντρώνουμε τα δεδομένα ως προς το μάθημα και τον βαθμό (Κώδικας 3.16).

```
> dat <- gather(dat, lesson, grade, lesson1:lesson4,
+               na.rm = TRUE)
> dat
Source: local data frame [21 x 4]
         name quarter  lesson grade
1      Κώστας       1 lesson1     A
2      Κώστας       2 lesson1     A
3      Κώστας       3 lesson1     A
4       Μαρία       1 lesson1     B
5       Μαρία       2 lesson1     Γ
6       Μαρία       3 lesson1     A
7  Παναγιώτης       1 lesson1     Γ
8  Παναγιώτης       2 lesson1     B
9  Παναγιώτης       3 lesson1     A
10      Μαρία       1 lesson2     B
..        ...     ...     ...   ...
>
```

Κώδικας *3.16* *Προβληματική περίπτωση συνόλου δεδομένων: αποθήκευση πολλαπλών μεταβλητών σε μια στήλη.*

Έπειτα, με χρήση της συνάρτησης `spread` δημιουργούμε τις 3 νέες μεταβλητές, που αντιστοιχούν στα τρίμηνα, για τα οποία έχουμε βαθμό σε όλα για κάποιο μάθημα.

```
> dat <- spread(dat, quarter, grade)
> dat
Source: local data frame [7 x 5]
        name  lesson 1 2 3
1     Κώστας lesson1 A A A
2     Κώστας lesson3 Γ B B
3      Μαρία lesson1 B Γ A
4      Μαρία lesson2 B A A
5 Παναγιώτης lesson1 Γ B A
6 Παναγιώτης lesson2 B B A
7 Παναγιώτης lesson3 A B B
>
```

Κώδικας *3.17* *Κατάτμηση των δεδομένων με χρήση της spread.*

Τέλος, μια ακόμα χρήσιμη συνάρτηση του πακέτου είναι και η `extract_numeric`, η οποία χρησιμοποιείται για την εξαγωγή αριθμητικής πληροφορίας από ένα αλφαριθμητικό. Παρατηρούμε ότι η στήλη μάθημα (lesson) μπορεί να απλοποιηθεί, κρατώντας μόνο τον αριθμό του μαθήματος (Κώδικας 3.17). Αυτό μπορεί να γίνει με χρήση της mutate και της extract_numeric (Κώδικας 3.18).

```
> mutate(dat, lesson = extract_numeric(lesson))
Source: local data frame [7 x 5]
        name lesson 1 2 3
1     Κώστας      1 A A A
2     Κώστας      3 Γ B B
3      Μαρία      1 B Γ A
4      Μαρία      2 B A A
5 Παναγιώτης      1 Γ B A
6 Παναγιώτης      2 B B A
7 Παναγιώτης      3 A B B
>
```

Κώδικας *3.18* *Χρήση των συναρτήσεων mutate και extract_numeric για απλοποίηση τιμών μεταβλητής.*

Κριτήρια αξιολόγησης

Κριτήριο αξιολόγησης 1

Δίνονται οι ακόλουθες ταχύτητες όπως μετρήθηκαν από αισθητήρα πάνω σε απλό επιβατικό αυτοκίνητο σε km/h: 6, 75, 100, 91, 45, 103, 55, 43, 87, 99, 73, 39, 23, 28, 101, 232.

(a) **Εφαρμόστε τη μέθοδο εξομάλυνσης ενδοχίασης (binning) με διαμερισμό ίσου βάθους και εξομάλυνση ως προς τη μέση τιμή του κάθε δοχείου.**
(b) **Εφαρμόστε τη μέθοδο εξομάλυνσης ενδοχίασης (binning) με διαμερισμό ίσου βάθους και εξομάλυνση ως προς τις τιμές ορίων του κάθε δοχείου.**
(c) **Ποια ή ποιες από αυτές τις τιμές θα μπορούσαν να θεωρηθούν ακραίες;**

Απάντηση

Αρχικά, ταξινομούμε τις τιμές κατά αύξουσα σειρά: 6, 23, 28, 39, 43, 45, 55, 73, 75, 87, 91, 99, 100, 101, 103, 232. Έχουμε 16 τιμές. Με βάση την εξομάλυνση ίσου βάθους πρέπει να γίνει διαχωρισμός σε τόσα δοχεία, ώστε όλα να έχουν τον ίδιο πλήθος τιμών. Συνεπώς, θα διαχωρίσουμε τα δεδομένα είτε σε 2 δοχεία των οχτώ τιμών είτε 4 δοχεία των τεσσάρων τιμών.

(a) Έστω ότι χωρίζουμε τις τιμές σε 4 δοχεία των τεσσάρων τιμών. Υπολογίζουμε τη μέση τιμή κάθε δοχείου και αντικαθιστούμε όλες τις τιμές με τη μέση τιμή. Έχουμε:

Δοχείο 1: 6, 23, 28, 39 → (6 + 23 + 28 + 39)/4 = 24 → Δοχείο 1: 24, 24, 24, 24
Δοχείο 2: 43, 45, 55, 73 → (43 + 45 + 55 + 73)/4 = 54 → Δοχείο 2: 54, 54, 54, 54
Δοχείο 3: 75, 87, 91, 99 → (75 + 87 + 91 + 99)/4 = 88 → Δοχείο 3: 88, 88, 88, 88
Δοχείο 4: 100, 101, 103, 232 → (100 + 101 + 103 + 232)/4 = 134 → Δοχείο 4: 134, 134, 134, 134

(b) Έστω ότι χωρίζουμε τις τιμές σε 4 δοχεία των τεσσάρων τιμών. Αντικαθιστούμε τις ενδιάμεσες τιμές με την τιμή του άκρου, στο οποίο βρίσκονται αριθμητικά πιο κοντά. Έχουμε:

Δοχείο 1: 6, 23, 28, 39 → 23-6=17, 28-6=22, 39-23=**16**, 39-28=**11** → Δοχείο 1: 6, 39, 39, 39
Δοχείο 2: 43, 45, 55, 73 → 45-43=**2**, 55-43=**12**, 73-45=28, 73-55=18 → Δοχείο 2: 43, 43, 43, 73
Δοχείο 3: 75, 87, 91, 99 → 87-75=**12**, 91-75=16, 99-87=**12**, 99-91=**8** → Δοχείο 3: 75, 75, 99, 99
Δοχείο 4: 100, 101, 103, 232→ 101-100=**1**, 103-100=**3**, 232-101=131, 232-103=129
→ Δοχείο 4: 100, 100, 100, 233

(c) Οι τιμές που μπορούν να θεωρηθούν ακραίες είναι οι 6 και 232.

Κριτήριο αξιολόγησης 2

Δίνονται οι τιμές του κριτηρίου αξιολόγησης 1: 6, 23, 28, 39, 43, 45, 55, 73, 75, 87, 91, 99, 100, 101, 103, 232. Αν χρησιμοποιούσαμε κανονικοποίηση min-max με στόχο το διάστημα [0, 1], ποια τιμή θα αντιστοιχούσε στο 0 και ποια στο 1; Προσπαθήστε να απαντήσετε, χωρίς να κάνετε υπολογισμούς. Στη συνέχεια, υπολογίστε σε ποια τιμή κανονικοποιείται η τιμή 100 στο [0, 1].

Απάντηση

Η τιμή 0 αντιστοιχεί στη μικρότερη αρχική τιμή, δηλαδή στο 6. Αντίστοιχα, η τιμή 1 αντιστοιχεί στη μεγαλύτερη αρχική τιμή, δηλαδή στο 232. Τέλος, χρησιμοποιώντας τον τύπο κανονικοποίησης min-max υπολογίζουμε σε ποια τιμή στο [0, 1] κανονικοποιείται το 100:

$$v_{new} = \frac{v - \min}{\max - \min}(\max_{new} - \min_{new}) + \min_{new} = \frac{100 - 6}{232 - 6}(1 - 0) + 0 = 0.42$$

Κριτήριο αξιολόγησης 3

(α) Δίνεται ο παρακάτω πίνακας με τα σύμβολα και τα στατιστικά τους. Υπολογίστε την κωδικοποίηση Huffman των συμβόλων.

x	p(x)
a	0.2
b	0.1
c	0.3
d	0.02
e	0.3
f	0.08

(β) Δίνονται οι παρακάτω συμβολοσειρές. Υπολογίστε μόνοι σας το λεξικό και τη συμπιεσμένη τους μορφή, χρησιμοποιώντας την τεχνική συμπίεσης Lempel-Ziv.

D_1 = a b c c a b c b a a b a d c b a b b c a

D_2 = a b a c b a c b b a c a c b b a c a

Απάντηση

Για βοήθεια μπορείτε να ανατρέξετε στην υποενότητα *3.2.4.2*.

α) Οι κωδικοποίηση για κάθε σύμβολο φαίνεται στον παρακάτω πίνακα:

x	Κώδικας
a	0010
b	0101
c	0
d	01101
e	11
f	11101

Να σημειωθεί ότι αυτή είναι μόνο μια ενδεικτική λύση. Για τα ίδια δεδομένα υπάρχουν αρκετές διαφορετικές, έγκυρες κωδικοποιήσεις.

β) Για την πρώτη συμβολοσειρά, D_1, η κωδικοποίηση είναι 0 1 2 2 4 2 1 0 4 0 3 9 4 5 0, ενώ το αντίστοιχο λεξικό φαίνεται στον παρακάτω πίνακα:

Λεξικό			
Δείκτης	Εγγραφή	Δείκτης	Εγγραφή
0	a	9	cb
1	b	10	ba
2	c	11	aa
3	d	12	aba
4	ab	13	ad
5	bc	14	dc
6	cc	15	cba
7	ca	16	abb
8	abc	17	bca

Για τη δεύτερη συμβολοσειρά, D_2, η κωδικοποίηση είναι 0 1 0 2 4 6 7 5 1 9, ενώ το αντίστοιχο λεξικό φαίνεται στον ακόλουθο πίνακα:

Λεξικό			
Δείκτης	Εγγραφή	Δείκτης	Εγγραφή
0	a	6	cb
1	b	7	bac
2	c	8	cbb
3	ab	9	baca
4	ba	10	acb
5	ac	11	bb

Για να επαληθεύσουμε την κωδικοποίηση, αντικαθιστούμε κάθε αριθμό της κωδικοποιημένης συμβολοσειράς με την αντίστοιχη εγγραφή. Στο τέλος αυτής της διαδικασίας, θα πρέπει να παραχθεί η αρχική συμβολοσειρά.

Κριτήριο αξιολόγησης 4

Στόχος αυτής της άσκησης είναι η περαιτέρω εξοικείωση με το πακέτο `dplyr`. Υλοποιείστε στην R τα παρακάτω ερωτήματα, χρησιμοποιώντας όπου χρειάζεται και μπορούν να χρησιμοποιηθούν οι έτοιμες συναρτήσεις του πακέτου `dplyr`.

α) Φορτώστε το έτοιμο σύνολο δεδομένων `mtcars`, που παρέχει η R, και μετατρέψτε το, ώστε η κλάση του να είναι `tbl_df`.
β) Επιλέξτε τις στήλες `mpg`, `cyl` και `wt`.
γ) Φιλτράρετε το επιλεγμένο υποσύνολο και κρατήστε μόνο τις εγγραφές με `cyl` ίσο με 8. Έπειτα ταξινομήστε τις εγγραφές κατά αύξουσα σειρά ως προς το `mpg` και κατά φθίνουσα ως προ το `wt`.

Απάντηση

α) Φορτώνουμε το πακέτο `dplyr`, το σύνολο δεδομένων και με χρήση της `tbl_df` μετατρέπουμε το σύνολο δεδομένο σε `tbl_df`:

```
> library(dplyr)
> data(mtcars)
> mtcars <- tbl_df(mtcars)
>
```

β) Με χρήση της συνάρτησης `select` του πακέτου `dplyr` επιλέγουμε τα ζητούμενα χαρακτηριστικά:

```
> mtcars <- select(mtcars, mpg, cyl, wt)
>
```

γ) Τέλος, με χρήση της συνάρτησης `filter` φιλτράρουμε τις εγγραφές και κρατάμε όσες έχουν τιμή `cyl` ίση με 8, ενώ για την ταξινόμηση χρησιμοποιούμε τη συνάρτηση `arrange`. Για τη φθίνουσα ταξινόμηση είναι αναγκαία και η χρήση της `desc`:

```
> mtcars <- filter(mtcars, cyl == 8)
> mtcars <- arrange(mtcars, mpg, desc(wt))
>
```

Βιβλιογραφία

Lossless Data Compression. Ανακτήθηκε στις 17 Νοεμβρίου 2015, από: http://www.data-compression.com/lossless.shtml

Gersho, A. & Gray, R. M. (1992). Vector Quantization and Signal Compression. *The Springer International Series in Engineering and Computer Science*. The Netherlands: Kluewer Academic Publishers Group.

Παράδειγμα Διακριτοποίησης με Εντροπία. URL: http://kevinmeurer.com/a-simple-guide-to-entropy-based-discretization/

Wickham, H. (2014). Tidy Data. *Journal of Statistical Sofrware, 59* (10). Ανακτήθηκε στις 17 Νοεμβρίου 2015, από: http://vita.had.co.nz/papers/tidy-data.pdf

Han, J., Kamber, M. & Pei, J. (2012). *Data Mining: Concepts and Techniques* (3rd edition). *Morgan Kaufmann, Elsevier.*

Κεφάλαιο 4: Συνοπτική Στατιστική και Οπτικοποίηση

Σύνοψη

Ο βασικός στόχος αυτού του κεφαλαίου είναι να εντρυφήσει ο φοιτητής στις διάφορες τεχνικές που διατίθενται για την εξερεύνηση των δεδομένων, έτσι ώστε να είναι σε θέση να τις γνωρίσει καλύτερα, για την πιο επιτυχή κατάληξη της εργασίας της Εξόρυξης Δεδομένων. Πιο συγκεκριμένα, ο φοιτητής θα γνωρίσει τεχνικές της Συνοπτικής Στατιστικής και της Οπτικοποίησης. Θα μελετήσει και θα εφαρμόσει μέτρα θέσης, διασποράς, και συσχέτισης, καθώς επίσης τεχνικές οπτικοποίησης, όπως ιστογράμματα, θηκογράμματα και διαγράμματα διασποράς. Θα μάθει να υπολογίζει τα διάφορα μέτρα θέσης, διασποράς και συσχέτισης, αλλά και να δημιουργεί ιστογράμματα, θηκογράμματα και διαγράμματα διασποράς με χρήση της γλώσσας R.

Προαπαιτούμενη γνώση

Για την καλύτερη κατανόηση αυτού του κεφαλαίου χρειάζεται να έχουν μελετηθεί ήδη τα εξής προηγούμενα Κεφάλαια: (α) Κεφάλαιο 1 – Εισαγωγή στην Εξόρυξη Δεδομένων, (β) Κεφάλαιο 2 – Εισαγωγή στην R, και (γ) Κεφάλαιο 3 – Τύποι, Ποιότητα και Προεπεξεργασία Δεδομένων.

Συνοπτική Στατιστική και Οπτικοποίηση

Η Συνοπτική ή Περιγραφική Στατιστική αποτελεί εκείνη την περιοχή της επιστήμης της Στατιστικής, που ασχολείται με τη συνοπτική και αποτελεσματική παρουσίαση των στατιστικών δεδομένων. Ανάλογα με την περιοχή εφαρμογής, τα στατιστικά δεδομένα μπορούν να παρουσιαστούν συνοπτικά είτε μέσω συγκεκριμένων αριθμητικών μέτρων, γνωστών ως μέτρων θέσης και διασποράς, είτε μέσω κατάλληλων διαγραμμάτων. Στην πιο αναλυτική, αλλά όχι τόσο κατάλληλη μορφή για την απόδοση συμπερασμάτων, τα στατιστικά δεδομένα μπορούν να παρουσιαστούν μέσω διανυσμάτων ή και πινάκων.

4.1 Μέτρα Θέσης

Τα μέτρα θέσης (ή μέτρα κεντρικής τάσης) περιγράφουν περιληπτικά τη θέση των δεδομένων πάνω στην ευθεία των πραγματικών αριθμών. Προσδιορίζουν ένα κεντρικό σημείο γύρω από το οποίο έχουν την τάση να συγκεντρώνονται τα δεδομένων. Τα κυριότερα μέτρα θέσης είναι η μέση τιμή (mean value) και η διάμεσος (median).

4.1.1 Μέση τιμή

Η μέση τιμή (mean value) ή αριθμητικός μέσος είναι το συνηθέστερο μέτρο κεντρικής θέσης. Αν n είναι το πλήθος των παρατηρήσεων x_i, i=1, …, n, η μέση τιμή ορίζεται ως

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Στην R, ο υπολογισμός της μέσης τιμής γίνεται με χρήση της συνάρτησης `mean()`.

Ας θεωρήσουμε, για παράδειγμα, τις παρακάτω τιμές, που αφορούν το πλήθος των ωρών χρήσης του διαδικτύου τον τελευταίο μήνα, ενός δείγματος 10 εφήβων: 22, 0, 7, 12, 5, 33, 14, 8, 0, 9. Εισάγουμε τα δεδομένα αυτά στην R, σε μια μεταβλητή με το όνομα `internet_usage`, και υπολογίζουμε τη μέση τιμή τους ως εξής:

```
> internet_usage = c(22, 0, 7, 12, 5, 33, 14, 8, 0, 9)
> internet_usage
 [1] 22  0  7  12  5  33  14  8  0  9
```

```
> mean(internet_usage)
[1] 11
```

Σε περίπτωση που τα διαθέσιμα δεδομένα έχουν ελλιπείς τιμές, τότε για τον υπολογισμό των μέτρων προσθέτουμε το όρισμα `na.rm = TRUE`. Για παράδειγμα:

```
> internet_usage = c(22, 0, 7, 12, 5, NA, 33, 14, 8, NA, 0, 9)
> mean(internet_usage, na.rm = TRUE)
[1] 11
```

Παρατηρήστε ότι η μέση τιμή του δείγματος, ακόμη και με την προσθήκη των μη διαθέσιμων παρατηρήσεων, δεν άλλαξε.

4.1.2 Διάμεσος

Η διάμεσος (median) είναι η τιμή της μεσαίας παρατήρησης, όταν οι παρατηρήσεις ταξινομηθούν με αύξουσα ή φθίνουσα διάταξη. Στην περίπτωση που το πλήθος n των παρατηρήσεων είναι περιττός αριθμός, τότε η μεσαία παρατήρηση είναι η (n+1)/2 – οστή, ενώ, όταν το πλήθος των παρατηρήσεων είναι άρτιος αριθμός, έχουμε δύο «μεσαίες» παρατηρήσεις, στις θέσεις n/2 και n/2+1, οπότε η η διάμεσος είναι το ημι-άθροισμα αυτών. Για παράδειγμα, για τις ταξινομημένες παρατηρήσεις (περιττού πλήθους):

2 8 16 17 21 33 33 35 37

η διάμεσος είναι η 5η παρατήρηση, δηλαδή είναι ίση με 21, ενώ για τις παρατηρήσεις (άρτιου πλήθους):

100 150 170 220 230 380

η διάμεσος είναι ίση με (170 + 220)/2 = 195.

Στην R, ο υπολογισμός της διαμέσου γίνεται με χρήση της συνάρτησης `median()`. Η εφαρμογή της συνάντησης στο προηγούμενο παράδειγμα με τις ώρες χρήσης διαδικτύου θα μας δώσει:

```
> median(internet_usage)
[1] 8.5
```

Ως μεσαία παρατήρηση, η διάμεσος έχει το χαρακτηριστικό να είναι μεγαλύτερη ή ίση από το 50% των παρατηρήσεων του δείγματος. Το χαρακτηριστικό αυτό θα το δούμε στη συνέχεια και σε άλλα αριθμητικά μέτρα.

4.2 Μέτρα Διασποράς

Τα μέτρα διασποράς ή μέτρα μεταβλητότητας περιγράφουν περιληπτικά τη διασπορά των δεδομένων πάνω στην ευθεία των πραγματικών αριθμών. Με άλλα λόγια, τα μέτρα διασποράς φανερώνουν τη μεταβλητότητα των παρατηρήσεων. Η μεταβλητότητα των παρατηρήσεων δεν είναι πάντα φανερή από τα μέτρα θέσης, για παράδειγμα, τη μέση τιμή. Αν τα δεδομένα είναι συγκεντρωμένα γύρω από τη μέση τιμή, δηλαδή αν η διασπορά είναι μικρή, τότε πράγματι η μέση τιμή αντιπροσωπεύει ικανοποιητικά τα δεδομένα. Σε διαφορετική περίπτωση, τα μέτρα θέσης δεν δίνουν καλή συνοπτική περιγραφή των παρατηρήσεων. Είναι δυνατό, επίσης, διαφορετικά δείγματα παρατηρήσεων από τον ίδιο πληθυσμό να έχουν το ίδιο μέτρο θέσης. Αυτό μπορεί να γίνει εύκολα κατανοητό, αν θεωρήσουμε τα σύνολα παρατηρήσεων A και B, όπου A = {33, 37, 48, 49, 52, 54, 62, 63, 64, 68, 71} και B = {1, 37, 38, 41, 45, 47, 48, 51, 56, 90, 147}. Η μέση τιμή και των δύο συνόλων είναι ίση με 54.636, ωστόσο, οι τιμές των δύο συνόλων έχουν διαφορετική μεταβλητότητα (διασπορά στην ευθεία των πραγματικών αριθμών). Τα κυριότερα μέτρα διασποράς, τα οποία και θα περιγράψουμε στη συνέχεια, είναι το εύρος, η διακύμανση, η τυπική απόκλιση, ο συντελεστής μεταβλητότητας και τα ποσοστιαία σημεία.

4.2.1 Ελάχιστη τιμή, Μέγιστη τιμή, Εύρος

Ας θεωρήσουμε το σύνολο παρατηρήσεων A = {49, 33, 37, 63, 48, 54, 62, 52, 64, 71, 68}. Η ελάχιστη (min) και η μέγιστη (max) παρατήρηση μπορεί να υπολογιστεί μέσω των συναρτήσεων `min()` και `max()`, αντίστοιχα.

```
> A = c(49, 33, 37, 63, 48, 54, 62, 52, 64, 71, 68)
> min(A)
[1] 33
> max(A)
[1] 71
```

Οι παρακάτω εντολές προσδιορίζουν τη θέση, όπου εμφανίζεται η ελάχιστη και η μέγιστη τιμή του A, αντίστοιχα.

```
> which.min(A)
[1] 2
> which.max(A)
[1] 10
```

Το εύρος (range) ορίζεται ως η διαφορά της μικρότερης παρατήρησης (min) από τη μεγαλύτερη παρατήρηση (max). Στην R μπορούμε εύκολα να υπολογίσουμε το εύρος με χρήση των αντίστοιχων συναρτήσεων.

```
> print(max(A) - min (A))
[1] 38
```

Η συνάρτηση `range()` επιστρέφει ένα διάνυσμα με τη μικρότερη και την ελάχιστη παρατήρηση του διανύσματος x.

```
> range(A)
[1] 33 71
```

Οπότε, μέσω της `range()` έχουμε έναν εναλλακτικό τρόπο υπολογισμού του εύρους εντός συνόλου παρατηρήσεων.

```
> print(range(A)[2] - range(A)[1])
[1] 38
```

4.2.2 Ποσοστιαία σημεία

Το p-ποσοστιαίο σημείο ενός δείγματος με n παρατηρήσεις, ορίζεται ως η παρατήρηση, για την οποία το πολύ p% των παρατηρήσεων είναι μικρότερες από αυτήν και το πολύ (1-p)% των παρατηρήσεων μεγαλύτερες από την αυτήν. Για την εύρεση του p-ποσοστιαίου σημείου, 1≤p≤99, οι παρατηρήσεις θα πρέπει να είναι διατεταγμένες σε αύξουσα διάταξη και τότε η παρατήρηση βρίσκεται στη θέση $\frac{n+1}{100}p$.

Στην R, για τον υπολογισμό του p-ποσοστιαίου σημείου χρησιμοποιούμε τη συνάρτηση `quantile(x,p,k)`, όπου `x` είναι το διάνυσμα με τις παρατηρήσεις μας και `p` είναι το p-ποσοστιαίο σημείο, $0.1 \leq p \leq 0.99$,και `k`, $1 \leq 9$, δηλαδή ένας αριθμός που προσδιορίζει τον αλγόριθμο υπολογισμού, τον οποίο θα χρησιμοποιήσουμε.

Ας θεωρήσουμε τις παρακάτω n=20 παρατηρήσεις, οι οποίες για διευκόλυνση είναι ήδη διατεταγμένες σε αύξουσα διάταξη: 3, 4, 5, 6, 7, 8, 10, 10, 11, 12, 14, 14, 14, 15, 16, 17, 21, 25, 27, 32.

Για παράδειγμα, το 80-ποσοστιαίο σημείο βρίσκεται στη θέση $\frac{20+1}{100}80 = 16.8$, άρα μεταξύ της 16ης και της 17ης παρατήρησης και συγκεκριμένα θα βρίσκεται δεξιότερα της 16ης παρατήρησης κατά 0.8 της διαφοράς μεταξύ των δύο παρατηρήσεων. Η 16η παρατήρηση είναι ίση με 17 και η 17η παρατήρησης είναι ίση με 21. Οπότε, η παρατήρηση που ψάχνουμε θα είναι ίση με $x_{16} + 0.8(x_{17}-x_{16}) = 17 + 0.8\ (21-17) = 20.2$. Ο τρόπος που χρησιμοποιήσαμε αντιστοιχεί στον υπ' αριθμό 7 αλγόριθμο υπολογισμού, ο οποίος είναι ενσωματωμένος στην R, οπότε, μέσω της εντολής `quantile()`, προκύπτει ότι:

```
> x= c(3, 4, 5, 6, 7, 8, 10, 10, 11, 12, 14, 14, 14, 15, 16,
17, 21, 25, 27, 32)
> quantile(x, 0.80, type = 7)
 80%
20.2
```

Ιδιαίτερο ενδιαφέρον παρουσιάσει το 25-ποσοστιαίο σημείο (η παρατήρηση που είναι μεγαλύτερη ή ίση από το 25% των παρατηρήσεων), που καλείται πρώτο τεταρτημόριο, το 50-ποσοστιαίο σημείο (η παρατήρηση που είναι μεγαλύτερη ή ίση από το 50% των παρατηρήσεων), που καλείται δεύτερο τεταρτημόριο και δεν είναι άλλο από τη διάμεσο, καθώς και το 75-ποσοστιαίο σημείο (η παρατήρηση που είναι μεγαλύτερη ή ίση από το 75% των παρατηρήσεων), που καλείται τρίτο τεταρτημόριο.

Για το προηγούμενο παράδειγμα, και με χρήση της R, έχουμε ότι:

```
> quantile(x, 0.25, type = 7)
 25%
7.75
> quantile(x, 0.50, type = 7)
50%
 13
> quantile(x, 0.75, type = 7)
  75%
16.25
```

Για περισσότερες πληροφορίες σχετικά με τη συνάρτηση `quantile()`, πληκτρολογήστε `help(«quantile»)`.

Η εντολή `summary()` συνοψίζει κάποια από τα αριθμητικά μέτρα, τα οποία έχουμε περιγράψει μέχρι τώρα.

```
> summary(x)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   3.00    7.75   13.00   13.55   16.25   32.00
```

4.2.3 Ενδοτεταρτημοριακό εύρος

Το ενδοτεταρτημοριακό εύρος (interquartile range, IRQ) ορίζεται ως η διαφορά τρίτου και πρώτου τεταρτημορίου. Στην R δεν υπάρχει ενσωματωμένη συνάρτηση που να το υπολογίζει, ωστόσο μπορούμε να δημιουργήσουμε τη δική μας:

```
> irq = function(x) (quantile(x,0.75) - quantile(x,0.25))
> irq(x)
75%
8.5
```

4.2.4 Διασπορά

Η διασπορά (variance) ή διακύμανση s^2 ενός δείγματος n παρατηρήσεων δίνεται από τον τύπο

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

Η διασπορά μπορεί να υπολογιστεί απλούστερα ως

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}x_i^2 - \frac{n}{n-1}\bar{x}^2$$

Όταν τα δεδομένα αποτελούν το σύνολο του πληθυσμού και όχι δείγμα αυτού, τότε η διακύμανση συμβολίζεται με σ^2 και δίνεται από τον τύπο

$$\mathsf{s}^2 = \frac{1}{\mathrm{N}}\sum_{i=1}^{n}(x_i - \mathsf{m})^2,$$

όπου N είναι το μέγεθος και μ η μέση τιμή του πληθυσμού.

Ο υπολογισμός της διασποράς στην R γίνεται μέσω της συνάρτησης `var()`. Για παράδειγμα, οι τιμές που ακολουθούν αντιστοιχούν στο πλήθος των μαθημάτων, που «οφείλει» ένα δείγμα 20 φοιτητών «επί πτυχίω»: 6, 2, 1, 9, 17, 4, 3, 2, 1, 5, 11, 4, 3, 1, 2, 2, 5, 4, 3, 6.

```
> courses = c(6, 2, 1, 9, 17, 4, 3, 2, 1, 5, 11, 4, 3, 1, 2,
2, 5, 4, 3, 6)
> var(courses)
[1] 15.41842
```

Δείτε, επίσης, πώς μπορούμε να υπολογίσουμε τη διασπορά χωρίς χρήση της συνάρτησης, αλλά μέσω του μαθηματικού ορισμού της.

```
> sum((courses - mean(courses))^2 / (length(courses)-1))
[1] 15.41842
```

Η συνάρτηση `sum()` υπολογίζει το άθροισμα της τετραγωνικής διαφοράς κάθε τιμής του διανύσματος `courses` από τη μέση τιμή `mean(courses)` αυτού, το οποίο στη συνέχεια διαιρείται με το πλήθος των παρατηρήσεων `length(courses)` μείον 1.

4.2.5 Τυπική Απόκλιση

Η τυπική απόκλιση (standard deviation) s ενός δείγματος παρατηρήσεων ορίζεται ως η τετραγωνική ρίζα της διακύμανσης των παρατηρήσεων. Στην R, η τυπική απόκλιση υπολογίζεται μέσω της συνάρτησης `sd()`.

```
> sd(courses)
[1] 3.92663
```

Ωστόσο, ο υπολογισμός της τυπικής απόκλισης μπορεί να γίνει εύκολα με χρήση των συναρτήσεων `var()` για τον υπολογισμό της διασποράς και της `sqrt()` για τον υπολογισμό της τετραγωνικής ρίζας. Για παράδειγμα:

```
> sqrt(var(courses))
[1] 3.92663
```

Μπορούμε, επίσης, να ορίσουμε εμείς τη συνάρτηση υπολογισμού της τυπικής απόκλισης, για παράδειγμα ως:

```
> std = function(x) sqrt(var(x))
```

και να την εφαρμόσουμε στο προηγούμενο παράδειγμα:

```
> std(courses)
[1] 3.92663
```

4.2.6 Συντελεστής μεταβλητότητας

Ο συντελεστής μεταβλητότητας (coefficient of variation, cv) ενός δείγματος παρατηρήσεων ορίζεται ως το πηλίκο της τυπικής απόκλισης προς τη μέση τιμή. Εκφράζει την τυπική απόκλιση ως ποσοστό της μέσης τιμής. Μπορούμε να δημιουργήσουμε την κατάλληλη συνάρτηση στην R για τον υπολογισμό του συντελεστή μεταβλητότητας ως εξής:

```
> cv = function(x) (sd(x) / mean(x))
```

Εφαρμόζοντας την στο προηγούμενο παράδειγμα με τη βαθμολογία μαθημάτων, έχουμε ότι:

```
> cv(courses)
[1] 0.8629955
```

4.3 Οπτικοποίηση Ποιοτικών Δεδομένων

Στην ενότητα που ακολουθεί θα παρουσιάσουμε τρόπους, με τους οποίους μπορούμε να οπτικοποιήσουμε τις παρατηρήσεις μας. Η οπτικοποίηση μπορεί να γίνει είτε μέσω διανυσμάτων και πινάκων αλλά είτε και μέσω διαγραμμάτων, όπως ιστογράμματα, ραβδογράμματα, διαγράμματα πίτας κ.ά. Σε κάθε περίπτωση, σημαντική είναι η διάκριση των παρατηρήσεων μας σε ποσοτικά ή ποιοτικά δεδομένα. Ξεκινάμε με τα ποιοτικά ή κατηγορικά δεδομένα, τα οποία συνήθως παρουσιάζουμε με πίνακες, ραβδογράμματα και διαγράμματα πίτας.

Ως ένα παράδειγμα εργασίας, ας υποθέσουμε ότι έχουμε στη διάθεση μας τις απαντήσεις που έδωσαν 20 άτομα αναφορικά με τη χρήση μέσων μεταφοράς σε καθημερινή βάση, προκειμένου να μεταβούν στον χώρο εργασίας τους. Οι ερωτηθέντες είχαν να επιλέξουν τη μετακίνηση τους με αυτοκίνητο (car), με λεωφορείο (bus), με μετρό (metro) και με τα πόδια (foot). Οι απαντήσεις που δόθηκαν ήταν οι εξής:

car, car, bus, metro, metro, car, metro, metro, foot, cat,
foot, bus. bus, metro, metro, car, car, car, metro, car

Αρχικά εισάγουμε τα δεδομένα, δημιουργώντας το διάνυσμα m. Τα δεδομένα μας είναι κατηγορικά, οπότε θα πρέπει να συμπεριλάβουμε τις τιμές τους σε διπλά εισαγωγικά.

```
> m = c("car", "car", "bus", "metro", "metro", "car", "metro", "met-
ro", "foot", "car", "foot", "bus", "bus", "metro", "metro", "car",
"car", "car", "metro", "car")
```

4.3.1 Πίνακες συχνοτήτων

Μπορούμε να απεικονίσουμε τις παρατηρήσεις μας σε μορφή πίνακα συχνοτήτων με χρήση της εντολής `table()`.

```
> table(m)
```

```
m
  bus   car  foot metro
    3     8     2     7
```

Ο παραπάνω πίνακας στην πρώτη γραμμή εμφανίζει τις διακεκριμένες τιμές των παρατηρήσεων και στη δεύτερη γραμμή τη συχνότητα εμφάνισης της κάθε τιμής. Μπορείτε εύκολα να επαληθεύσετε ότι, για παράδειγμα, 8 απάντησαν πως μετακινούνται στην εργασία τους με αυτοκίνητο (car).

Αν επιθυμούμε να δούμε τις σχετικές συχνότητας (την αναλογία συχνότητας, δηλαδή, κάθε παρατήρησης προς το πλήθος των παρατηρήσεων), τότε χρησιμοποιούμε την εντολή `prop.table(table())`.

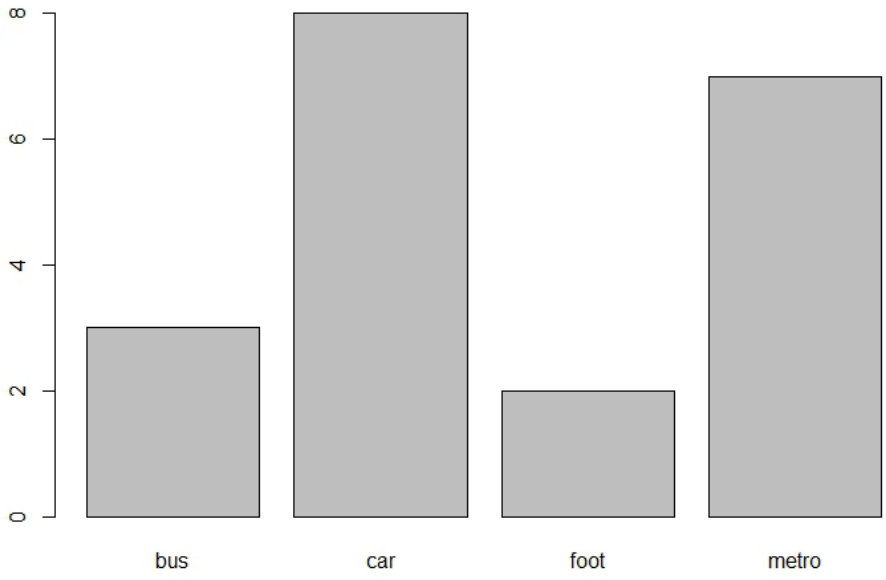
```
> prop.table(table(m))
```

```
m
  bus   car  foot metro
 0.15  0.40  0.10  0.35
```

4.3.2 Ραβδογράμματα

Το ραβδόγραμμα (bar chart) που ακολουθεί αφορά στις συχνότητες του δείγματος και προκύπτει μέσω της εντολής:

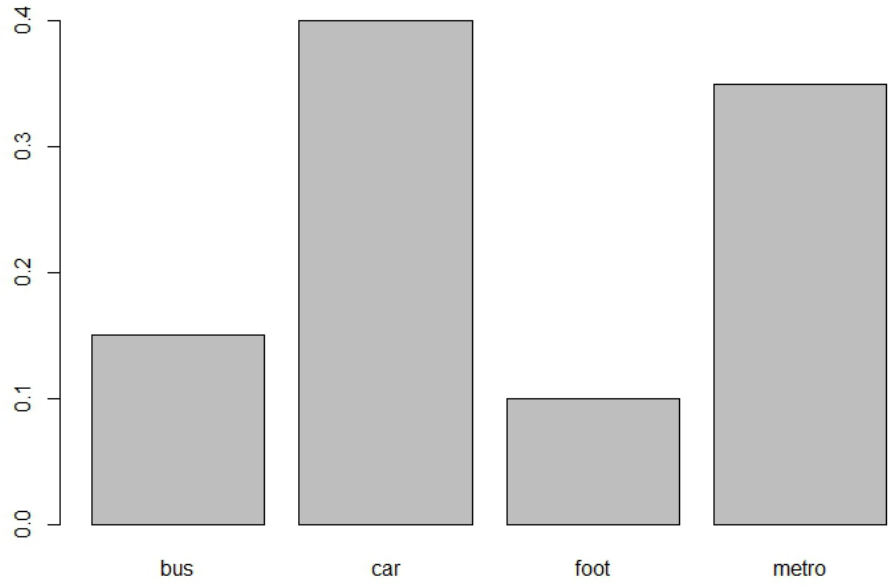
```
> barplot(table(m))
```



Εικόνα ***4.1*** *Παράδειγμα σχεδίασης ραβδογράμματος με χρήση συχνοτήτων.*

Αντίστοιχα, το ραβδόγραμμα σχετικών συχνοτήτων προκύπτει με την εντολή:

```
> barplot(prop.table(table(m)))
```

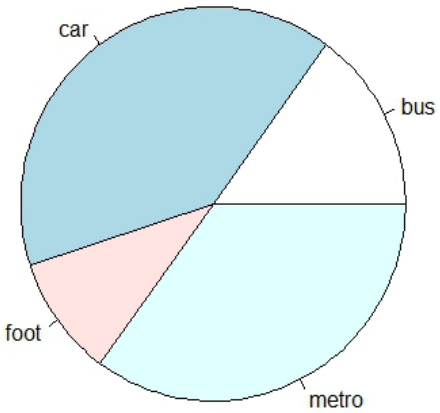


Εικόνα ***4.2*** *Παράδειγμα σχεδίασης ραβδοβράμματος με χρήση σχετικών συχνοτήτων.*

4.3.3 Διαγράμματα Πίτας

Μπορούμε να απεικονίσουμε τα δεδομένα μας σε μορφή διαγράμματος πίτας ή τομεογράφημα (pie chart), μέσω της εντολής:

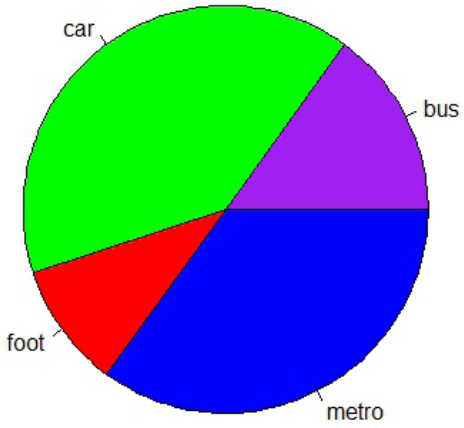
```
> pie(table(m))
```



Εικόνα ***4.3*** *Παράδειγμα διαγράμματος πίτας.*

ή ακόμα και να αποδώσουμε διαφορετικό χρωματισμό στους κυκλικούς τομείς του διαγράμματος πίτας:

```
> pie(prop.table(table(m)), col=c("purple", "green", "red",
"blue"))
```



Εικόνα ***4.4*** *Παράδειγμα ορισμού χρώματος στους κυκλικούς τομείς διαγράμματος πίτας.*

4.3.4 Πίνακες Συνάφειας

Ένας πίνακας συνάφειας (contingency matrix) αφορά δύο κατηγορηματικές μεταβλητές και απεικονίζει την κατανομή των συχνοτήτων τους. Η χρήση του θα γίνει κατανοητή με το ακόλουθο παράδειγμα.

Ας υποθέσουμε ότι στα δεδομένα της προηγούμενης έρευνας έχουμε συμπεριλάβει και το φύλο κάθε ατόμου. Έστω ότι, για απλότητα, τα 8 πρώτα άτομα ήταν άνδρες (M) και τα υπόλοιπα 12 ήταν γυναίκες (F). Δημιουργούμε το διάνυσμα g.

```
> g = c(rep(«M»,8),rep(«F»,12))
> g
 [1] "M" "M" "M" "M" "M" "M" "M" "M" "F" "F" "F" "F" "F" "F"
"F" "F" "F" "F" "F" "F"
```

Παρατηρήστε τον γρήγορο τρόπο αρχικοποίησης του διανύσματος g μέσω της συνάρτησης `rep()`, λόγω της υπόθεσης που κάναμε παραπάνω. Δημιουργούμε τον πίνακα συχνοτήτων διπλής εισόδου mg ως εξής:

```
> mg = table(m,g)
> mg
       g
m       F M
  bus   2 1
  car   5 3
  foot  2 0
  metro 3 4
```

Στη συνέχεια, με χρήση του πίνακα υπολογίζουμε τις περιθώριες (marginal) συχνότητες αναφορικά με το μέσο μετακίνησης και το φύλο, αντίστοιχα:

```
> margin.table(mg,1)
m
  bus   car  foot metro
    3     8     2     7
```

```
> margin.table(mg,2)
g
 F  M
12  8
```

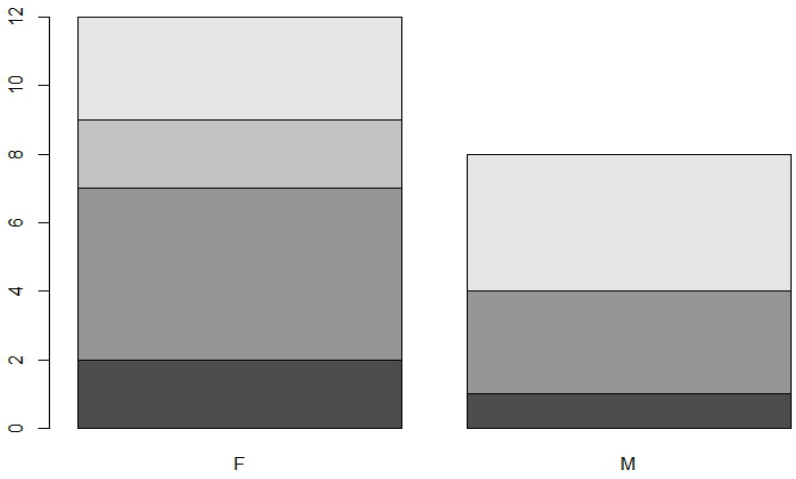
Μπορούμε να εργαστούμε με τον πίνακα σχετικών συχνοτήτων με τον ίδιο ακριβώς τρόπο.

```
> prop.table(mg)
       g
m          F    M
  bus   0.10 0.05
  car   0.25 0.15
  foot  0.10 0.00
  metro 0.15 0.20

> prop.table(mg,1)
       g
m                F         M
  bus   0.6666667 0.3333333
  car   0.6250000 0.3750000
  foot  1.0000000 0.0000000
  metro 0.4285714 0.5714286

> prop.table(mg,2)
       g
m               F         M
  bus   0.1666667 0.1250000
  car   0.4166667 0.3750000
  foot  0.1666667 0.0000000
  metro 0.2500000 0.500000
```

4.3.4 Στοιβαγμένα Ραβδογράμματα και Ομαδοποιημένα Ραβδογράμματα

Μπορούμε να αναπαραστήσουμε ποιοτικά δεδομένα, τα οποία προέρχονται από τις τιμές δύο μεταβλητών με χρήση ενός στοιβαγμένου ραβδογράμματος (stacked barplot) ή ενός ομαδοποιημένου ραβδογράμματος (grouped barplot).

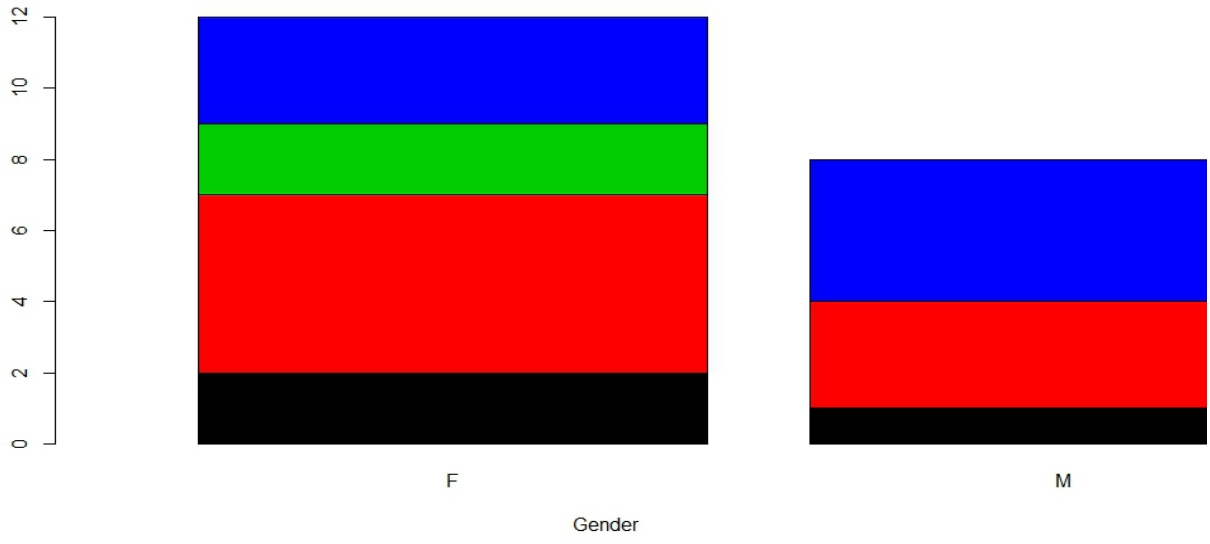
```
> barplot(mg)
```



Εικόνα ***4.5*** *Παράδειγμα στοιβαγμένου ραβδογράμματος.*

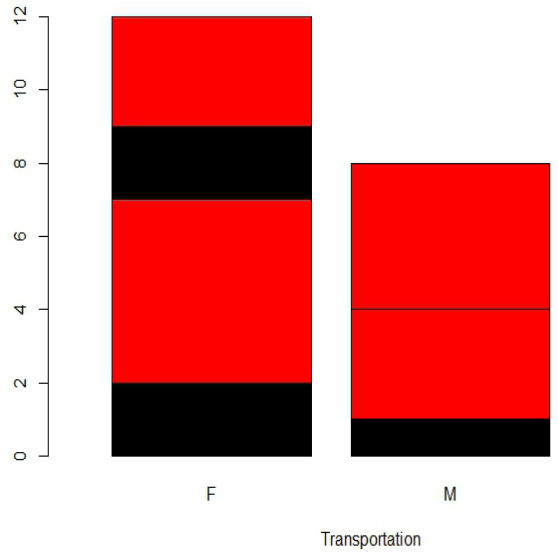
Χρησιμοποιώντας τις παραμέτρους της εντολής, το αποτέλεσμα είναι πολύ καλύτερο:

```
> barplot(mg, xlim = c(0,2), xlab="Gender", legent = levels(m),
col = 1:4)
```



Εικόνα ***4.6*** *Παράδειγμα σχεδίασης στοιβαγμένου ραβδογράμματος.*

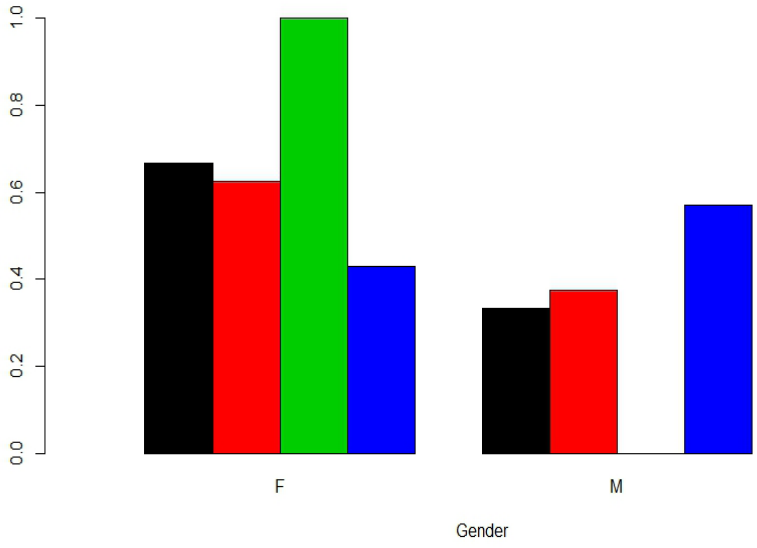
```
> barplot(mg, xlim = c(0,2), xlab="Transportation", legent =
levels(g), col=1:2)
```



Εικόνα ***4.7*** *Παράδειγμα σχεδίασης στοιβαγμένου ραβδογράμματος.*

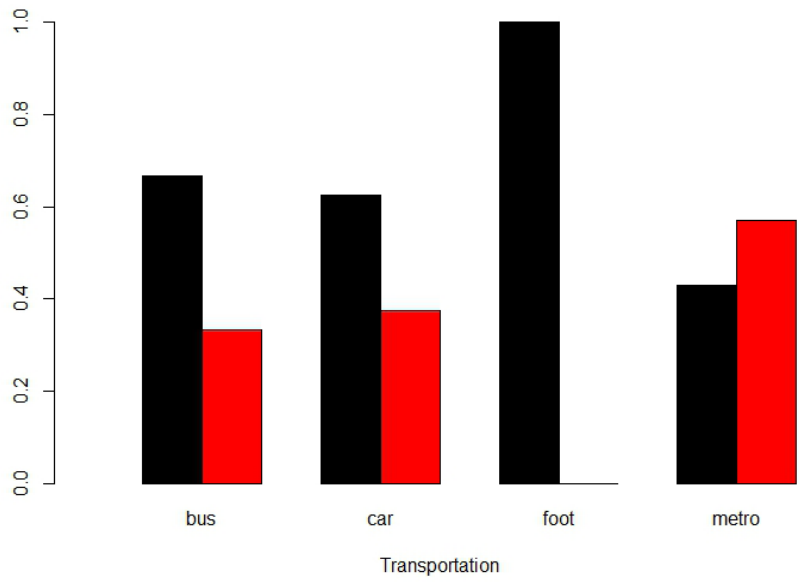
Οι εντολές που ακολουθούν δημιουργούν ομαδοποιημένα ραβδογράμματα.

```
> barplot(prop.table(mg,1), width=0.25, xlim = c(0,3), ylim =
c(0,1), xlab="Gender", legent = levels(m), beside=T, col = 1:4)
```



Εικόνα ***4.8*** *Παράδειγμα σχεδίασης ομαδοποιημένου ραβδογράμματος.*

```
> mg = table(g,m)
> barplot(prop.table(mg,2), width=0.25, xlim = c(0,3), ylim =
c(0,1), xlab="Transportation", legent = levels(a), beside=T,
col = 1:2)
```



Εικόνα *4.9* *Παράδειγμα σχεδίασης ομαδοποιημένου ραβδογράμματος.*

4.4 Οπτικοποίηση Ποσοτικών Δεδομένων

4.4.1 Πίνακες συχνοτήτων

Ας υποθέσουμε ότι οι παρακάτω τιμές αφορούν τη βαθμολογία ενός δείγματος 30 φοιτητών στο μάθημα «Εξόρυξη Δεδομένων»:

10, 10, 5, 9, 7, 6, 8, 6, 5, 8, 10, 7, 7, 8, 5, 6, 4, 7, 9, 7, 4, 8, 10, 10, 7, 4, 9, 5, 8, 9

Μπορούμε να παρουσιάσουμε τα δεδομένα μέσω ενός πίνακα συχνοτήτων, ως εξής:

```
> x = c(10, 10, 5, 9, 7, 6, 8, 6, 5, 8, 10, 7, 7, 8, 5, 6, 4,
7, 9, 7, 4, 8, 10, 10, 7, 4, 9, 5, 8, 9)
> table(x)
x
 4  5  6  7  8  9 10
 3  4  3  6  5  4  5
```

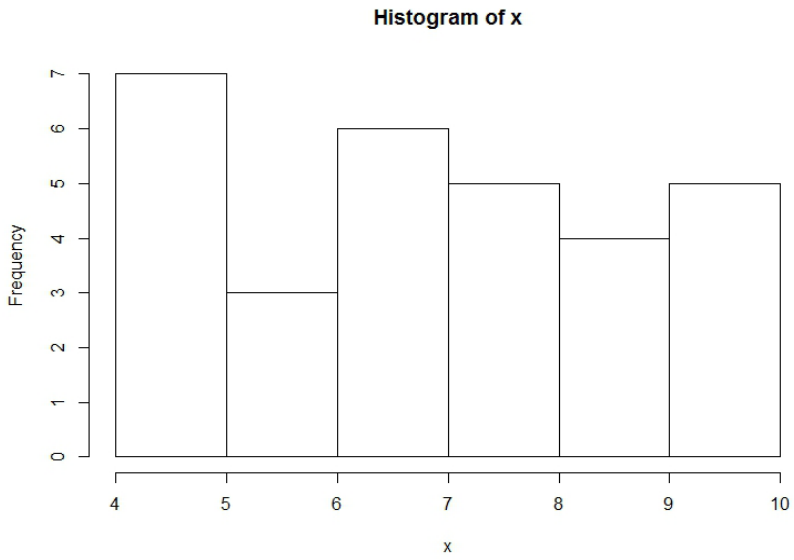
Ο παραπάνω πίνακας στην πρώτη γραμμή εμφανίζει τις διακεκριμένες τιμές των παρατηρήσεων και στη δεύτερη γραμμή τη συχνότητα εμφάνισης της κάθε τιμής. Μπορείτε εύκολα να επαληθεύσετε ότι για παράδειγμα η τιμή 7 εμφανίζεται 6 φορές στο δείγμα. Μπορούμε αντίστοιχα να εμφανίζουμε και τον πίνακα σχετικών συχνοτήτων:

```
> prop.table(table(x))
x
        4         5         6         7         8         9
10
0.1000000 0.1333333 0.1000000 0.2000000 0.1666667 0.1333333
0.1666667
```

4.4.2 Ιστογράμματα

Για την παρουσίαση ποσοτικών δεδομένων χρησιμοποιούμε τα ιστογράμματα (histograms), μέσω της εντολής `hist()`. Για το διάνυσμα x, το ιστόγραμμα συχνοτήτων προκύπτει με την εντολή:

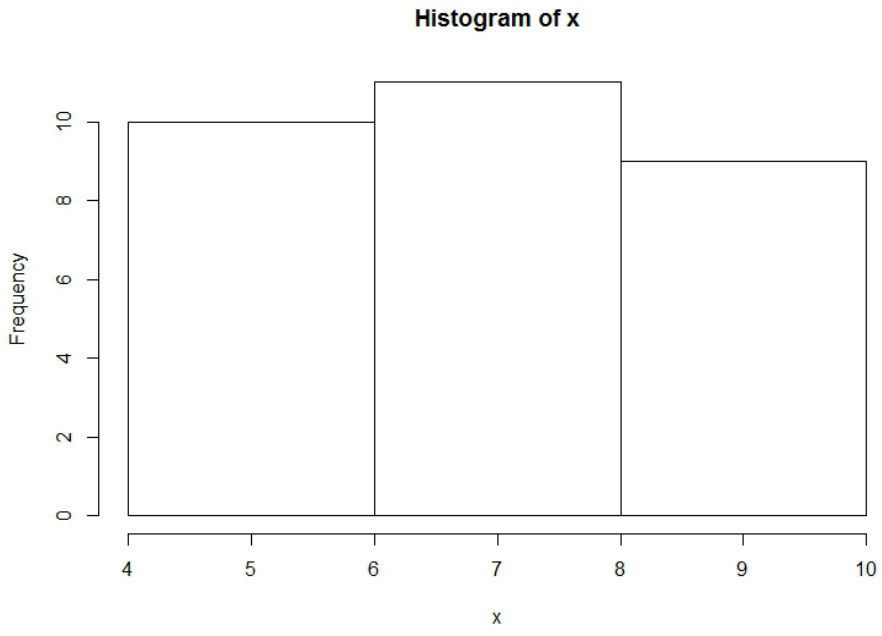
```
> hist(x)
```



Εικόνα ***4.10*** *Παράδειγμα ιστογράμματος.*

Τα δεδομένα ομαδοποιούνται σε κλάσεις, οι οποίες αναπαριστούνται με διαδοχικά ορθογώνια. Η βάση κάθε ορθογωνίου αντιστοιχεί στο εύρος της αντίστοιχης κλάσης, ενώ το ύψος αντιστοιχεί στη συχνότητα της αντίστοιχης παρατήρησης. Συνήθως δημιουργούμε κλάσεις ιδίου εύρους. Μπορούμε, αν θέλουμε, να προσδιορίσουμε εμείς το πλήθος των κλάσεων, ως εξής:

```
> hist(x, nclass=3)
```



Εικόνα ***4.11*** *Παράδειγμα σχεδίασης ιστογράμματος με προκαθορισμένο πλήθος κλάσεων.*

Η χρησιμότητα του ιστογράμματος θα γίνει πιο κατανοητή σε ένα παράδειγμα με μεγαλύτερο πλήθος παρατηρήσεων. Ας υποθέσουμε ότι έχουμε διαθέσιμα τα αποτελέσματα της μέτρησης του βάρους (σε γραμμάρια) γέννησης 60 βρεφών:

1950, 2090, 2700, 3350, 4200, 3720, 4400, 2980, 3850, 4550
3050, 2350, 1850, 2820, 3670, 2950, 3750, 1850, 2420, 3150
3000, 3470, 3920, 3100, 2400, 2900, 2650, 3450, 3650, 4020
4450, 3120, 3660, 3070, 3550, 2020, 3500, 2500, 3780, 3940
3540, 2800, 2850, 4450, 1950, 3020, 2800, 3500, 1480, 4495
2850, 3100, 2250, 3300, 4100, 3220, 3600, 2130, 4020, 4075

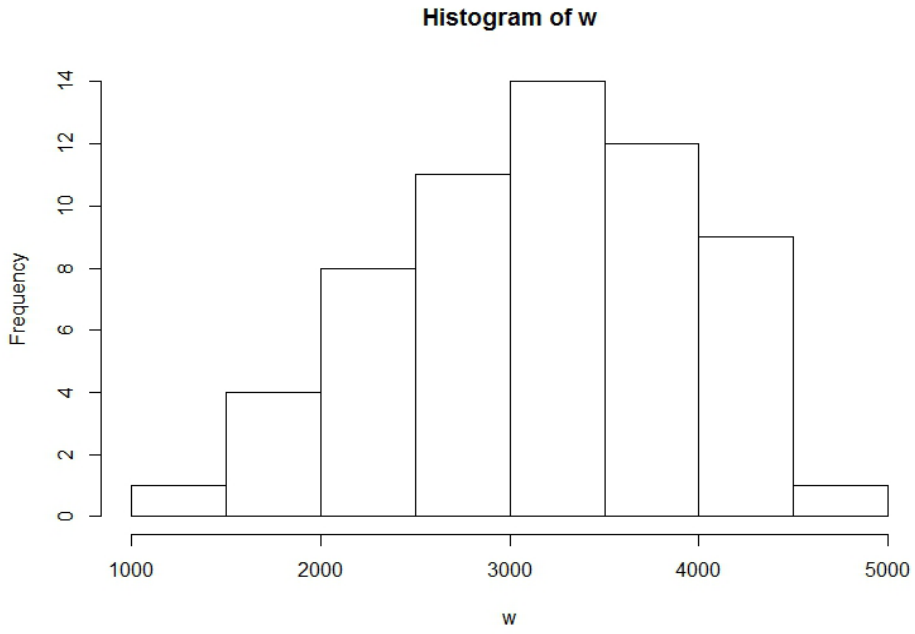
Αρχικά εισάγουμε τις παρατηρήσεις μας μέσω του διανύσματος w:

```
> w = c(1950, 2090, 2700, 3350, 4200, 3720, 4400, 2980, 3850, 4550,
+       3050, 2350, 1850, 2820, 3670, 2950, 3750, 1850, 2420, 3150,
+       3000, 3470, 3920, 3100, 2400, 2900, 2650, 3450, 3650, 4020,
+       4450, 3120, 3660, 3070, 3550, 2020, 3500, 2500, 3780, 3940,
+       3540, 2800, 2850, 4450, 1950, 3020, 2800, 3500, 1480, 4495,
+       2850, 3100, 2250, 3300, 4100, 3220, 3600, 2130, 4020, 4075)
```

Η χρήση της εντολής:

```
> hist(w)
```

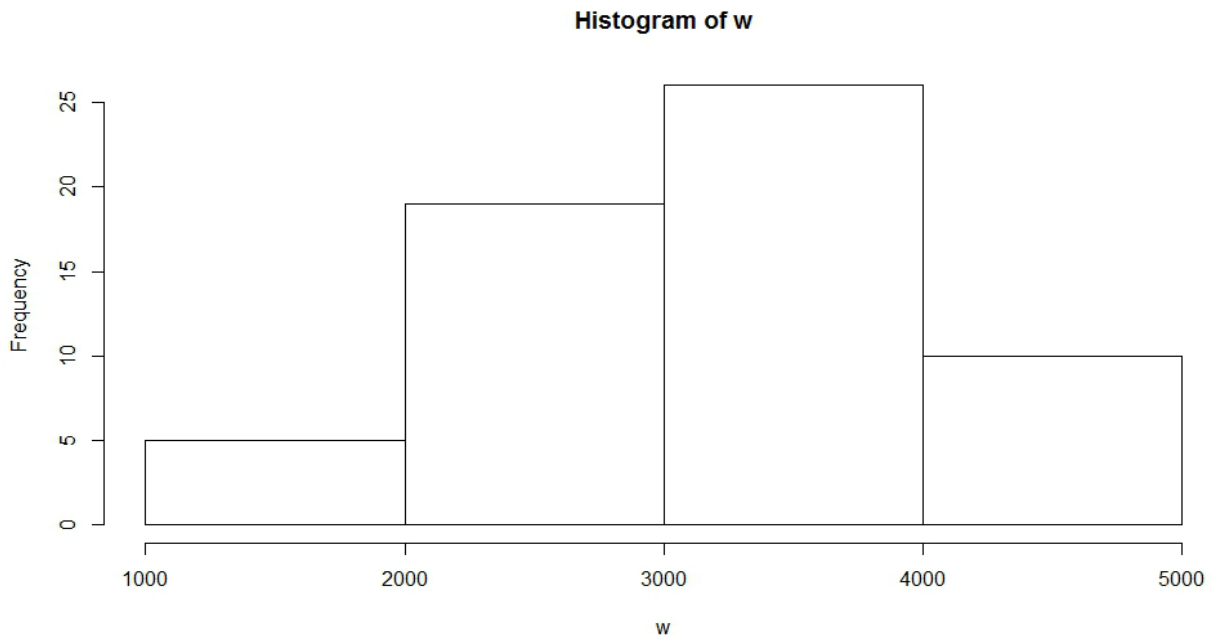
δίνει το ιστογράφημα που ακολουθεί:



Εικόνα *4.12* *Παράδειγμα σχεδίασης ιστογράμματος.*

Η R υπολογίζει αυτόματα το πλήθος και το εύρος των κλάσεων. Αν θέλουμε να δημιουργήσουμε ένα ιστόγραμμα, όπου οι παρατηρήσεις μας να είναι ομαδοποιημένες, για παράδειγμα, σε 4 κλάσεις, αυτό γίνεται μέσω της εντολής:

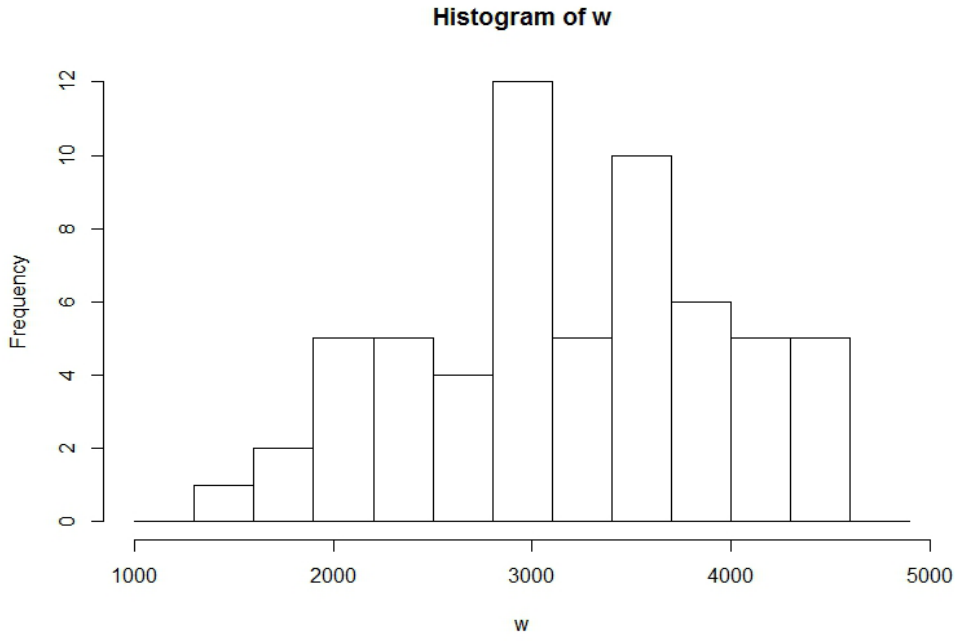
```
> hist(w, nclass = 4)
```



Εικόνα ***4.13*** *Σχεδίαση ιστογράμματος με προκαθορισμένο πλήθος κλάσεων.*

Μπορούμε, επίσης, να ορίσουμε την αρχή και το τέλος των κλάσεων μέσω μιας επαναληπτικής διαδικασίας.

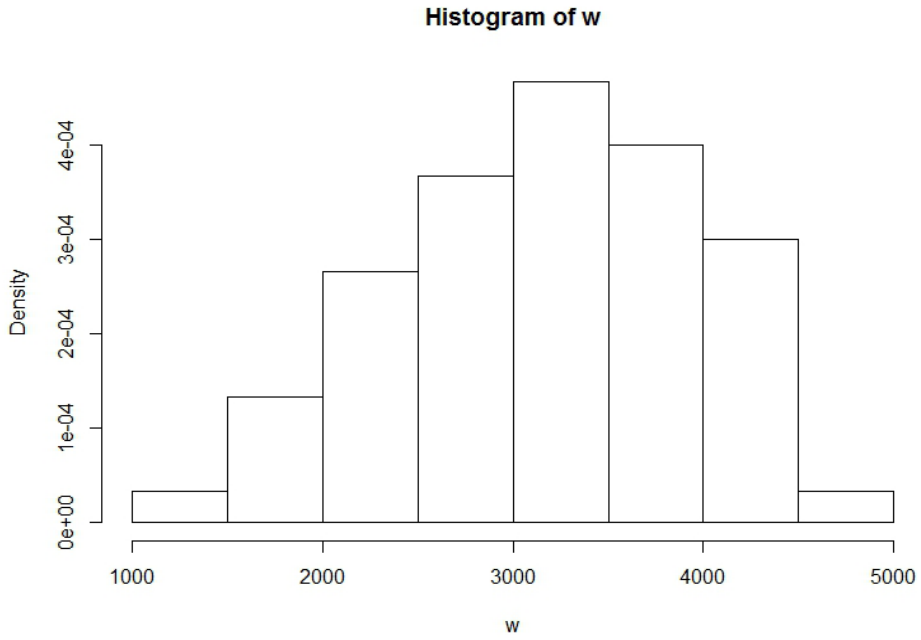
```
> hist(w, breaks = seq(from = 1000, to=5000, by=300))
```



Εικόνα ***4.14*** *Σχεδίαση ιστογράμματος με προκαθορισμένη αρχική και τελική τιμή και εύρος κλάσεων.*

Επιπλέον, μπορούμε στον κάθετο άξονα, αντί για συχνότητα, να έχουμε τη σχετική συχνότητα, δηλαδή την πυκνότητα πιθανότητας, ως εξής:

```
> hist(w, probability=T)
```

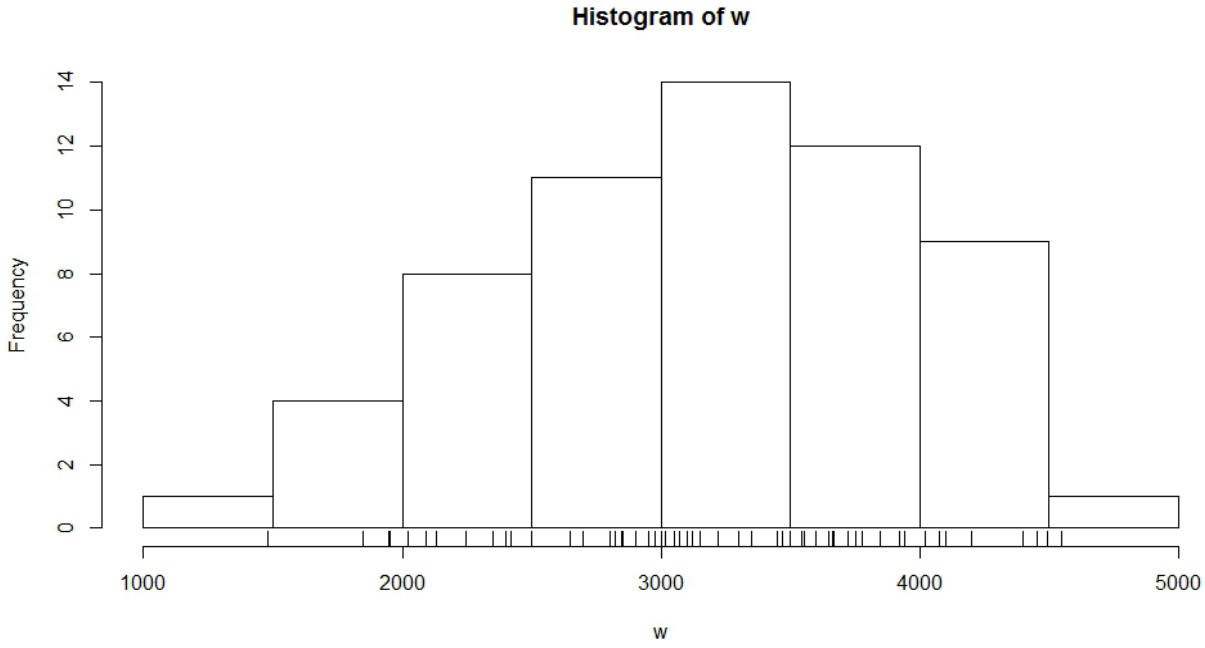


Εικόνα ***4.15*** *Σχεδίαση ιστογράμματος μέσω σχετικών συχνοτήτων.*

Οι εντολές:

```
> hist(w)
> rug(jitter(w))
```

εμφανίζουν, εντός από το ιστόγραμμα, και τις τιμές των παρατηρήσεων.



Εικόνα ***4.16*** *Εμφάνιση ιστογράμματος και των τιμών των παρατηρήσεων.*

4.4.3 Πολύγωνο Συχνοτήτων

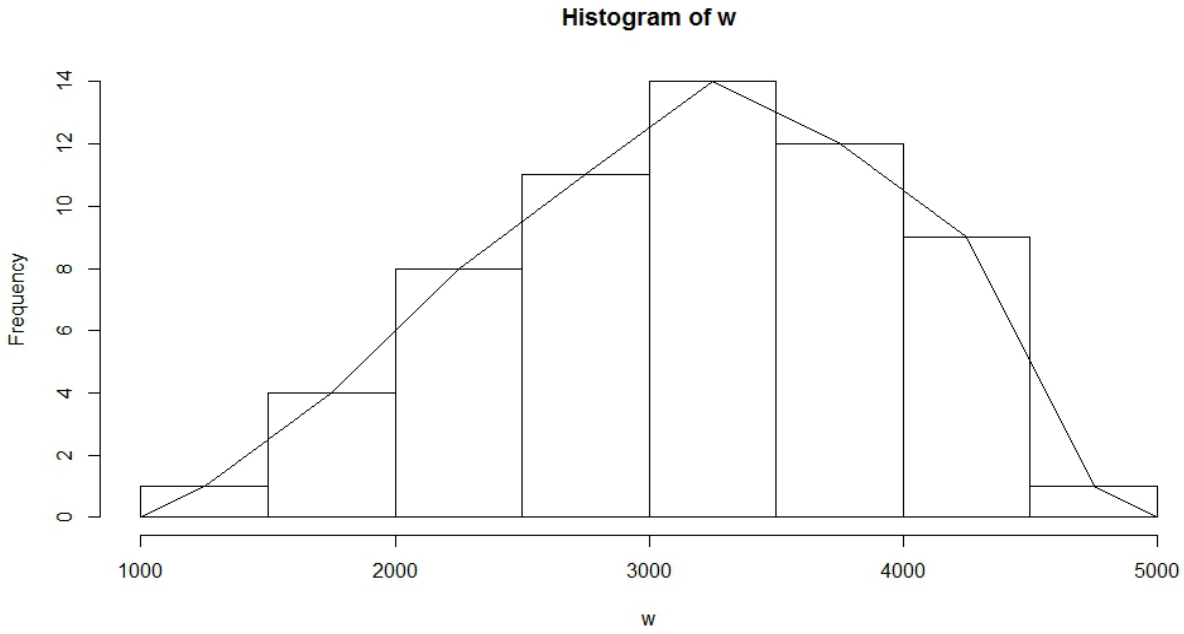
Μπορούμε να σχεδιάσουμε το πολύγωνο συχνοτήτων εύκολα, καθώς ουσιαστικά σχηματίζεται από την τεθλασμένη πολυγωνική γραμμή, η οποία ενώνει τα μέσα (mids) των κλάσεων, που δημιουργούν το ιστόγραμμα. Μπορούμε να διακρίνουμε τις τιμές των μέσων, καθώς και άλλων μεταβλητών, μέσω των εντολών:

```
> temp = hist(w)
> temp
$breaks
[1] 1000 1500 2000 2500 3000 3500 4000 4500 5000

$counts
[1]  1  4  8 11 14 12  9  1

$density
[1]  3.333333e-05  1.333333e-04  2.666667e-04  3.666667e-04
4.666667e-04 4.000000e-04 3.000000e-04 3.333333e-05

$mids
[1] 1250 1750 2250 2750 3250 3750 4250 4750

$xname
[1] "w"

$equidist
[1] TRUE

attr(,"class")
[1] «histogram»
```

Τώρα, αρκεί να ενώσουμε το κάτω αριστερό άκρο (την τιμή 1000), τα μέσα των κλάσεων, και το κάτω δεξιό άκρο (5000), από αριστερά προς τα δεξιά με μια τεθλασμένη γραμμή, μέσω της εντολής:

```
> lines(c(min(temp$breaks), temp$mids,max(temp$breaks)), c(0,-
temp$counts,0), type="l")
```

Οι συντεταγμένες ως προς τον κάθετο άξονα προκύπτουν από τη συχνότητα (counts) των παρατηρήσεων σε κάθε κλάση.

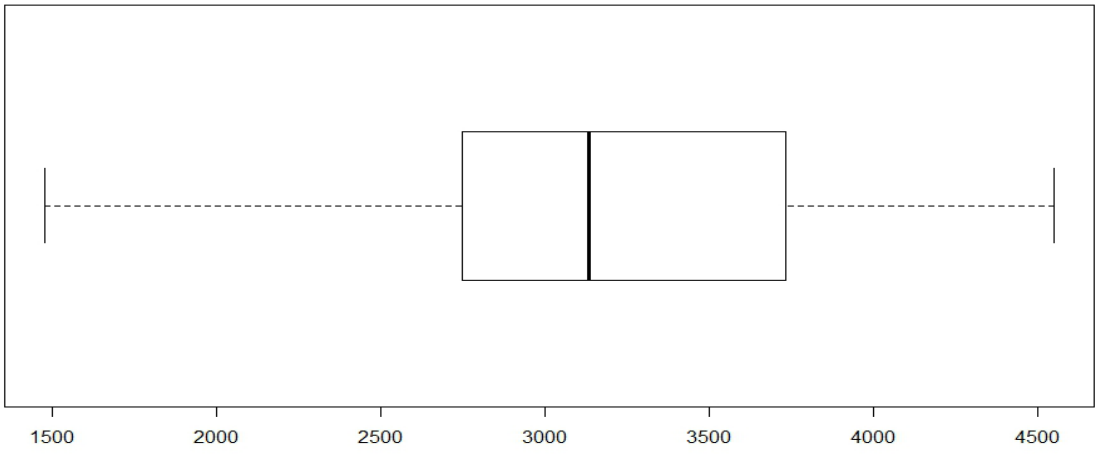


Εικόνα ***4.17*** *Παράδειγμα πολυγωνικού ιστογράμματος.*

4.4.4 Θηκόγραμμα

Το θηκόγραμμα (boxplot) είναι ένας κατάλληλος τρόπος, για να παρουσιάσουμε τα κυριότερα χαρακτηριστικά της κατανομής των παρατηρήσεων του δείγματος. Πρόκειται για ένα ορθογώνιο που βασίζεται στις τιμές του 1ου, του 2ου (της διαμέσου) και του 3ου τεταρτημορίου, ενώ οι μύστακες εκτείνονται από τη μικρότερη μέχρι τη μεγαλύτερη τιμή των παρατηρήσεων. Το θηκόγραμμα, για το παράδειγμα των βαρών γέννησης των βρεφών, μπορεί να σχεδιαστεί μέσω της εντολής:

```
> boxplot(w, horizontal = T)
```



Εικόνα ***4.18*** *Παράδειγμα οριζόντιας σχεδίασης θηκογράμματος.*

Αν θέλετε, μπορείτε να καταργήσετε την οριζόντια σχεδίαση του θηκογράμματος (θα προκύψει το ίδιο θηκόγραμμα στραμμένο 90 μοίρες δεξιά). Τις τιμές των πέντε στατιστικών μεγεθών, που χρησιμοποιούνται για την κατασκευή του θηκογράμματος, μπορούμε να τις δούμε μέσω της εντολής:

```
> fivenum(w)
[1] 1480 2750 3135 3735 4550
```

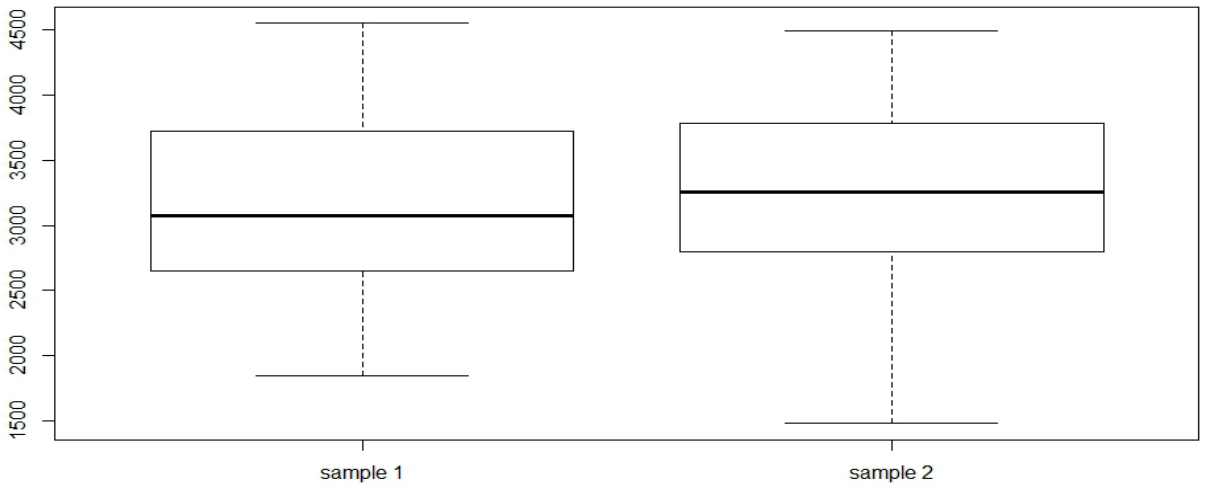
Τα θηκογράμματα είναι ένας ωραίος τρόπος, για να συγκρίνουμε δυο δείγματα μεταξύ τους. Ας υποθέσουμε, για παράδειγμα, πως το δείγμα w1 αποτελείται από τις τιμές:

1950, 2090, 2700, 3350, 4200, 3720, 4400, 2980, 3850, 4550
3050, 2350, 1850, 2820, 3670, 2950, 3750, 1850, 2420, 3150
3000, 3470, 3920, 3100, 2400, 2900, 2650, 3450, 3650, 4020

και το δείγμα w2 από τις τιμές:

4450, 3120, 3660, 3070, 3550, 2020, 3500, 2500, 3780, 3940
3540, 2800, 2850, 4450, 1950, 3020, 2800, 3500, 1480, 4495
2850, 3100, 2250, 3300, 4100, 3220, 3600, 2130, 4020, 4075

```
> w1 = c(1950, 2090, 2700, 3350, 4200, 3720, 4400, 2980, 3850, 4550,
+      3050, 2350, 1850, 2820, 3670, 2950, 3750, 1850, 2420, 3150,
+      3000, 3470, 3920, 3100, 2400, 2900, 2650, 3450, 3650, 4020)
> fivenum(w1)
[1] 1850 2650 3075 3720 4550

> w2 = c(4450, 3120, 3660, 3070, 3550, 2020, 3500, 2500, 3780, 3940,
+      3540, 2800, 2850, 4450, 1950, 3020, 2800, 3500, 1480, 4495,
+      2850, 3100, 2250, 3300, 4100, 3220, 3600, 2130, 4020, 4075)
> fivenum(w2)
[1] 1480 2800 3260 3780 4495

> boxplot(w1, w2, names = c("sample 1","sample 2"))
```



Εικόνα *4.19* *Σύγκριση διαφορετικών δειγμάτων μέσω θηκογραμμάτων.*

Κριτήρια αξιολόγησης

Κριτήριο αξιολόγησης 1

Υπολογίστε τη διάμεσο των παρατηρήσεων {2, 16, 33, 8, 17, 37, 21, 35, 33} και {150, 380, 100, 220, 230, 170}.

Απάντηση

```
> a = c(1, 16, 33, 8, 17, 37, 21, 35, 33)
> median(a)
[1] 21
> b = c(150, 380, 100, 220, 230, 170)
> median(b)
[1] 195
```

Κριτήριο αξιολόγησης 2

Επαληθεύστε ότι τα σύνολα τιμών A = {33, 37, 48, 49, 52, 54, 62, 63, 64, 68, 71} και B = {1, 37, 38, 41, 45, 47, 48, 51, 56, 90, 147} έχουν την ίδια μέση τιμή.

Απάντηση

```
> A = c(33, 37, 48, 49, 52, 54, 62, 63, 64, 68, 71)
> mean(A)
[1] 54.63636
> B = c(1, 37, 38, 41, 45, 47, 48, 51, 56, 90, 147)
> mean(B)
[1] 54.63636
```

Κριτήριο αξιολόγησης 3

Υπολογίστε τη μέση τιμή και την τυπική απόκλιση του δείγματος των παρατηρήσεων {74, 122, 235, 111, 292, 111, 211, 133, 156, 79}.

Απάντηση

```
> x = c(74, 122, 235, 111, 292, 111, 211, 133, 156, 79)
> mean(x)
[1] 152.4
> sd(x)
[1] 71.50789
```

Κριτήριο αξιολόγησης 4

Για το δείγμα παρατηρήσεων {46, 104, 94, 114, 35, 70, 120, 29, 19, 135, 200, 222, 89, 100, 55, 214, 15, 81, 118, 193} επαληθεύστε ότι ο συντελεστής μεταβλητότητας είναι ίσος με 0.624 και το ενδοτεταρτημοριακό εύρος είναι ίσο με 71.

Απάντηση

```
> cv = function(x) (sd(x) - mean(x))
> irq = function(x) (quantile(x,0.75) - quantile(x,0.25))
> y = c(46, 104, 94, 114, 35, 70, 120, 29, 19, 135, 200, 222,
89, 100, 55, 214, 15, 81, 118, 193)
> cv(y)
[1] 0.6240246
> irq(y)
75%
 71
```

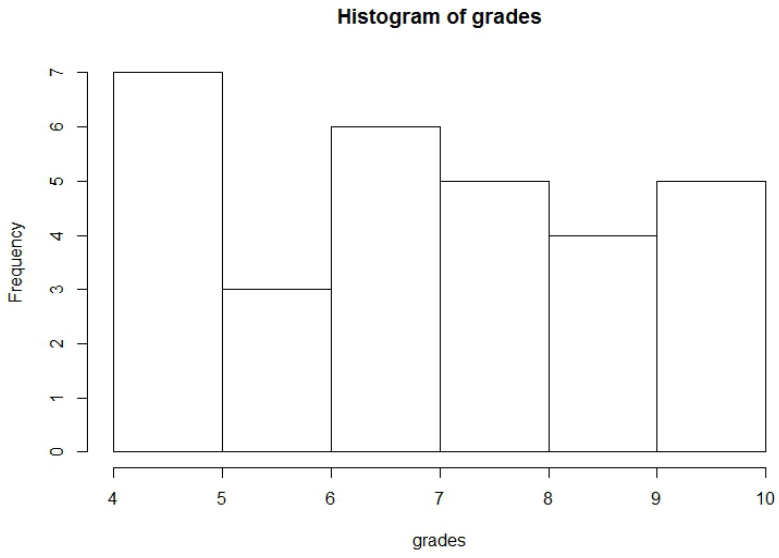
Κριτήριο αξιολόγησης 5

Θεωρήστε το παρακάτω δείγμα παρατηρήσεων, που αντιστοιχούν στη βαθμολογία 30 φοιτητών στο μάθημα «Επεξεργασία Δεδομένων»:10, 10, 5, 9, 7, 6, 8, 6, 5, 8, 10, 7, 7, 8, 5, 6, 4, 7, 9, 7, 4, 8, 10, 10, 7, 4, 9, 5, 8, 9.

Με χρήση των κατάλληλων εντολών, να σχεδιάσετε το ιστόγραμμα, όπου το αριστερό άκρο της 1ης κλάσης θα είναι ίσο με την τιμή της ελάχιστης παρατήρησης, το δεξιό άκρο της τελευταίας κλάσης θα είναι ίσο με την τιμή της μέγιστης παρατήρησης, και κάθε κλάση θα έχει εύρος 1.

Απάντηση

```
> grades = c(10, 10, 5, 9, 7, 6, 8, 6, 5, 8, 10, 7, 7, 8, 5,
6, 4, 7, 9, 7, 4, 8, 10, 10, 7, 4, 9, 5, 8, 9)
> hist(grades, breaks = seq(from = min(grades), to=max(grades),
by=1))
```



Εικόνα *4.20* *Το ιστόγραμμα του κριτηρίου αξιολόγησης 5*

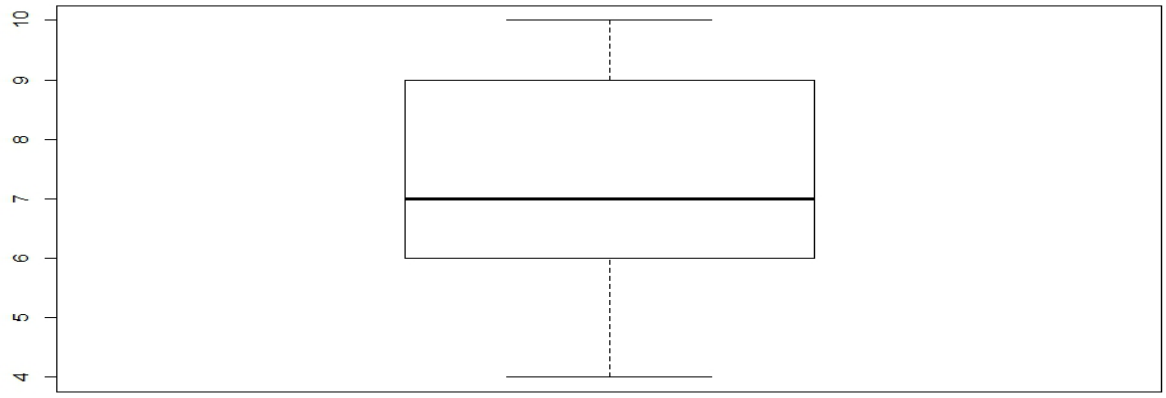
Κριτήριο αξιολόγησης 6

Θεωρήστε το παρακάτω δείγμα παρατηρήσεων, που αντιστοιχούν στη βαθμολογία 30 φοιτητών στο μάθημα «Επεξεργασία Δεδομένων»:

10, 10, 5, 9, 7, 6, 8, 6, 5, 8, 10, 7, 7, 8, 5, 6, 4, 7, 9, 7, 4, 8, 10, 10, 7, 4, 9, 5, 8, 9

Με χρήση των κατάλληλων εντολών, εμφανίστε το θηκόγραμμα, καθώς και τις τιμές των στατιστικών μεγεθών, στις οποίες βασίζεται.

Απάντηση



Εικόνα *4.21* *Το θηκόγραμμα του κριτηρίου αξιολόγησης 6*

```
> fivenum(grades)
[1]  4  6  7  9  10
```

Βιβλιογραφία

Peng, R. D. (2015). *R Programming for Data Science*. Lean Publishing. Ανακτήθηκε στις 19 Νοεμβρίου 2015, από: https://leanpub.com/rprogramming

R Core Team (2015). *An Introduction to R (ver 3.2.2)*. Ανακτήθηκε στις 19 Νοεμβρίου 2015, από: http://cran.r-project.org/doc/manuals/r-release/R-intro.html

Wikipedia (2015). *R programming language*. Ανακτήθηκε στις 25 Νοεμβρίου 2015, από: https://en.wikipedia.org/wiki/R_%28programming_language%29

Boslaugh, S. & Watters, P.A. (2008). *Statistics in a Nutshell.* Sebastopol, California: O'Reilly.

Κεφάλαιο 5: Κατηγοριοποίηση και Πρόβλεψη

Σύνοψη

Ο βασικός στόχος αυτού του κεφαλαίου είναι η εισαγωγή στις έννοιες της κατηγοριοποίησης και της πρόβλεψης. Η κατηγοριοποίηση έχει ως σκοπό τη δημιουργία ενός μοντέλου κατηγοριοποίησης με τη χρήση ενός συνόλου εκπαίδευσης και ενός αλγόριθμου μάθησης, μέσω του οποίου μπορεί να γίνει η ανάθεση τιμών στο γνώρισμα της κατηγορίας σε μη κατηγοριοποιημένες εγγραφές. Υπάρχουν διαφόρων ειδών μοντέλα κατηγοριοποίησης, όπως κανόνες, λίστες, δέντρα απόφασης, σύνολο υποδειγμάτων ή παραδειγμάτων δεδομένων, νευρωνικά δίκτυα, μέθοδοι ομάδων κ.λπ. Σε αυτό το κεφάλαιο θα ασχοληθούμε με την επαγωγή μοντέλων δέντρων απόφασης και θα δούμε τις τεχνικές διάσπασης, οι οποίες χρησιμοποιούνται για την ανάπτυξη των δέντρων αυτών. Στη συνέχεια, θα μελετηθεί η έννοια της πρόβλεψης και θα εξεταστεί η γραμμική παλινδρόμηση, ένα από τα πιο απλά μοντέλα πρόβλεψης για αριθμητικά δεδομένα. Τέλος, θα ασχοληθούμε με θέματα, που αφορούν τη γενίκευση των μοντέλων, όπως είναι η υπερπροσαρμογή ενός μοντέλου στα δεδομένα.

Προαπαιτούμενη γνώση

Η μελέτη αυτού του κεφαλαίου προϋποθέτει γνώσεις, οι οποίες στηρίζονται στο Κεφάλαιο 1, Ενότητα 1.8 – Βασικές Έννοιες, Ορισμοί και Συμβολισμοί, στο Κεφάλαιο 2 – Εισαγωγή στην R, καθώς και στο Κεφάλαιο 3 – Τύποι, Ποιότητα και Προεπεξεργασία Δεδομένων.

Κατηγοριοποίηση και Πρόβλεψη

5.1 Κατηγοριοποίηση

Η κατηγοριοποίηση αποτελεί μια από τις βασικές εργασίες στο στάδιο της Εξόρυξης Δεδομένων. Βασίζεται στην εξέταση των χαρακτηριστικών ενός αντικειμένου, το οποίο με βάση τα χαρακτηριστικά αυτά αντιστοιχίζεται σε ένα προκαθορισμένο σύνολο κλάσεων.

Η βασική ιδέα είναι η εξής: έχοντας ένα σύνολο από κατηγορίες (κλάσεις) και ένα σύνολο δεδομένων με δείγματα, για τα οποία ξέρουμε σε ποια κλάση ανήκουν, στόχος της κατηγοριοποίησης είναι η δημιουργία ενός μοντέλου, το οποίο θα μπορεί να κατηγοριοποιήσει αυτόματα σε αυτές τις κατηγορίες νέα, άγνωστα, μη-κατηγοριοποιημένα δείγματα.

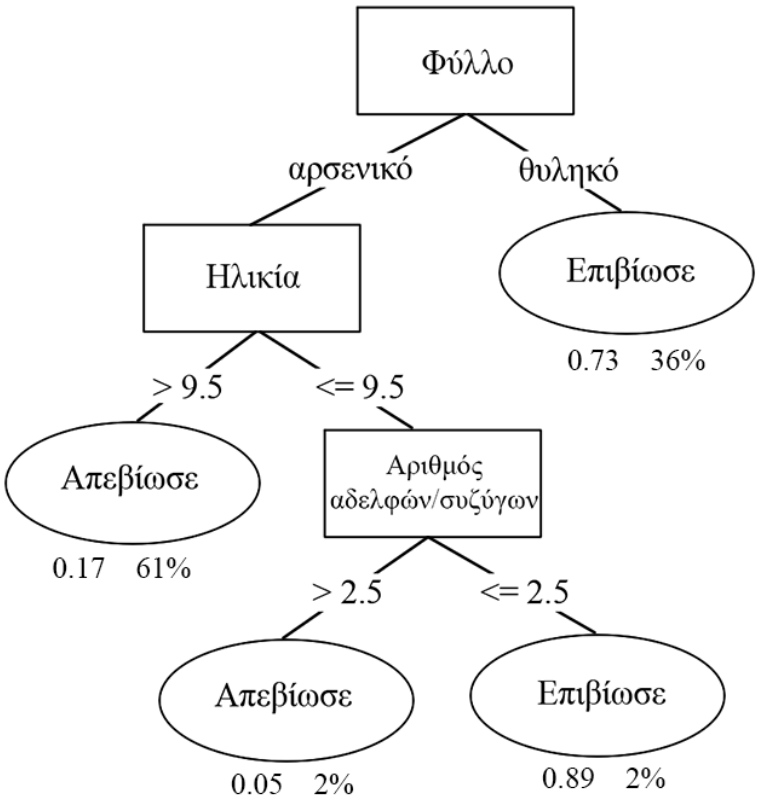
5.1.2 Δένδρα Απόφασης

Ένα από τα δημοφιλέστερα μοντέλα κατηγοριοποίησης είναι τα δένδρα απόφασης. Τα δέντρα απόφασης είναι μια απλή μορφή αναπαράστασης κανόνων και είναι ευρέως διαδεδομένα, επειδή είναι εύκολα κατανοητά από τον άνθρωπο.

5.1.2.1 Περιγραφή

Τα δένδρα απόφασης είναι το απλούστερο μοντέλο κατηγοριοποίησης. Γενικά, ένα δένδρο αποτελείται από εσωτερικούς κόμβους και φύλλα. Εσωτερικούς κόμβους λέμε τους κόμβους, οι οποίοι έχουν παιδιά, ενώ φύλλα λέμε τους κόμβους του κατώτερου επιπέδου, τα οποία δεν έχουν απογόνους. Ο τρόπος αναπαράστασης γίνεται ως εξής:

- κάθε εσωτερικός κόμβος του δένδρου ονοματίζεται με το όνομα ενός χαρακτηριστικού,
- κάθε κλαδί/σύνδεση δυο κόμβων ονοματίζεται με μια συνθήκη ή τιμή για το χαρακτηριστικό του γονικού κόμβου,
- κάθε φύλλο ονοματίζεται με το όνομα μιας κλάσης.



Εικόνα ***5.1*** *Παράδειγμα δένδρου απόφασης.*

Στην Εικόνα 5.1, βλέπουμε ένα παράδειγμα δένδρου απόφασης. Πρόκειται για ένα δένδρο απόφασης, το οποίο δημιουργήθηκε με βάση το σύνολο δεδομένων των επιβατών του Τιτανικού. Κάτω από τα φύλλα εμφανίζεται η πιθανότητα επιβίωσης και το ποσοστό δειγμάτων, που καταλήγουν στο συγκεκριμένο φύλλο. Είναι αναμενόμενο οι περισσότεροι άντρες να απεβίωσαν, αφού για τις σωστικές λέμβους δόθηκε προτεραιότητα στα παιδιά και στις γυναίκες.

Στο παραπάνω παράδειγμα χρησιμοποιήθηκαν οι μεταβλητές φύλο, ηλικία και αριθμός συνεπιβατών αδελφών/συζύγων, για να προσδιοριστεί η τιμή της κλάσης. Εφόσον έχουμε πεπερασμένο αριθμό τιμών (Επιβίωσε, Απεβίωσε), αναφερόμαστε σε ένα δένδρο απόφασης, το οποίο κάνει κατηγοριοποίηση.

5.1.2.2 Κατασκευή Δένδρου Απόφασης – Αλγόριθμος ID3

Ένας από τους δημοφιλέστερους αλγόριθμους κατασκευής δένδρων απόφασης είναι ο αλγόριθμος ID3. Ο συγκεκριμένος αλγόριθμος χρησιμοποιεί τις έννοιες της εντροπίας και του πληροφοριακού κέρδους για την επιλογή των κόμβων του δένδρου απόφασης. Υπενθυμίζουμε ότι το κέρδος πληροφορίας υπολογίζεται από τον τύπο:

$$G(S, A) = E(S) - I(S, A)$$

όπου:

$$I(S, A) = \sum_j \frac{|S_j|}{|S|} E(S_j),$$

όπου με S_j συμβολίζουμε τα δείγματα με τιμή j για το χαρακτηριστικό A, με $|S_j|$ το πλήθος τους, με S συμβολίζουμε όλα τα δείγματα και με |S| το πλήθος τους, ενώ με $E(S_j)$ συμβολίζουμε την εντροπία για το υποσύνολο δειγμάτων του συνόλου δεδομένων με τιμή j για το χαρακτηριστικό A. Η εντροπία E για ένα δεδομένο σύνολο υπολογίζεται με βάση την κατανομή της κλάσης των δειγμάτων στο σύνολο. Αν έχουμε k κλάσεις, η εντροπία για το σύνολο δεδομένων S είναι:

$$E(S) = -\sum_{i=1}^{k} p_i \log_2(p_i)$$

όπου p_i είναι η πιθανότητα της κλάσης i στο S.
Ο αλγόριθμος κατασκευής δένδρου απόφασης ID3 έχει τα εξής βήματα:

1. Υπολόγισε το πληροφοριακό κέρδος κάθε μεταβλητής.
2. Θέσε ως ρίζα του δένδρου τη μεταβλητή με το μεγαλύτερο πληροφοριακό κέρδος.
3. Δημιούργησε τόσα κλαδιά όσες και οι διακριτές τιμές της μεταβλητής.
4. Χώρισε το σύνολο δεδομένων σε τόσα υποσύνολα όσα και οι διακριτές τιμές της μεταβλητής που επιλέχθηκε.
5. Επέλεξε μια τιμή-υποσύνολο, που δεν έχει ήδη επιλεχθεί. Αν στην τρέχουσα τιμή – υποσύνολο αντιστοιχεί μόνο μια τιμή κλάσης, πήγαινε στο βήμα 6, αλλιώς στο βήμα 7.
6. Βάλε την τιμή κλάσης ως φύλλο και προχώρησε στην επόμενη τιμή μεταβλητής-υποσύνολο και πήγαινε στο βήμα 5.
7. Υπολόγισε το πληροφοριακό κέρδος των υπόλοιπων μεταβλητών για το συγκεκριμένο υποσύνολο.
8. Επέλεξε τη μεταβλητή με το μεγαλύτερο πληροφοριακό κέρδος και πρόσθεσε έναν νέο κόμβο στον κλάδο που αντιστοιχεί στην τρέχουσα τιμή-υποσύνολο.
9. Επανέλαβε από το βήμα 3, μέχρι να μην μπορούν να δημιουργηθούν νέα φύλλα.

Α/Α	**Θέα**	Θερμοκρασ**ία**	**Υγρασία**	**Αέρας**	**Κλάση**
1	Ηλιοφάνεια	Υψηλή	Υψηλή	Ασθενής	Μέσα
2	Ηλιοφάνεια	Υψηλή	Υψηλή	Δυνατός	Μέσα
3	Συννεφιά	Υψηλή	Υψηλή	Ασθενής	Έξω
4	Βροχή	Κανονική	Υψηλή	Ασθενής	Έξω
5	Βροχή	Χαμηλή	Κανονική	Δυνατός	Μέσα
6	Συννεφιά	Χαμηλή	Κανονική	Ασθενής	Έξω
7	Βροχή	Κανονική	Κανονική	Ασθενής	Έξω
8	Συννεφιά	Υψηλή	Κανονική	Ασθενής	Έξω

Πίνακας *5.1* *Σύνολο δεδομένων για το παράδειγμα κατασκευής δένδρου απόφασης με τον ID3.*

Ας δούμε ένα παράδειγμα κατασκευής δένδρου απόφασης με τον ID3, για το σύνολο δεδομένων που παρουσιάζει ο Πίνακας 5.1. Αρχικά υπολογίζουμε την εντροπία E(S). Για τη μεταβλητή κλάσης έχουμε 3 φορές την τιμή Μέσα και 5 φορές την τιμή Έξω. Συνεπώς:

$$E(S) = -\frac{3}{8}\log_2\left(\frac{3}{8}\right) - \frac{5}{8}\log_2\left(\frac{5}{8}\right) = 0.53 + 0.42 = 0.95$$

Στη συνέχεια υπολογίζουμε το πληροφοριακό κέρδος για κάθε μεταβλητή. Ξεκινάμε με τη μεταβλητή Θέα. Έχουμε συνολικά 8 δείγματα και η μεταβλητή Θέα παίρνει 2 φορές την τιμή Ηλιοφάνεια, και από 3 φορές τις τιμές Συννεφιά και Βροχή. Για τα 2 δείγματα με τιμή Θέα=Ηλιοφάνεια και τα 2 έχουν τιμή κλάσης Μέσα. Για τα 3 δείγματα με τιμή Θέα=Συννεφιά και τα 3 έχουν τιμή κλάσης Έξω. Για τα 3 δείγματα με τιμή Θέα=Βροχή, 1 έχει τιμή κλάσης Μέσα και 2 έχουν τιμή κλάσης Έξω. Συνεπώς, έχουμε:

$$G(S,\Theta\acute{\varepsilon}\alpha) = E(S) - I(S,\Theta\acute{\varepsilon}\alpha) = E(S) - \frac{2}{8}E(S_{\text{Ηλιοφάνεια}}) - \frac{3}{8}E(S_{\text{Συννεφιά}}) - \frac{3}{8}E(S_{\text{Βροχή}})$$

όπου:

$$E(S_{\text{Ηλιοφάνεια}}) = -\frac{2}{2}\log_2\left(\frac{2}{2}\right) - \frac{0}{2}\log_2\left(\frac{0}{2}\right) = 0$$

$$E(S_{\text{Συννεφιά}}) = -\frac{0}{3}\log_2\left(\frac{0}{3}\right) - \frac{3}{3}\log_2\left(\frac{3}{3}\right) = 0$$

$$E(S_{\text{Βροχή}}) = -\frac{1}{3}\log_2\left(\frac{1}{3}\right) - \frac{2}{3}\log_2\left(\frac{2}{3}\right) = 0.53 + 0.39 = 0.92$$

Επομένως, τελικά:

$$G(S, \Theta\acute{\varepsilon}\alpha) = 0.95 - \frac{2}{8}\cdot 0 - \frac{3}{8}\cdot 0 - \frac{3}{8}\cdot 0.92 = 0.345$$

Στη συνέχεια υπολογίζουμε το πληροφοριακό κέρδος για τη μεταβλητή Θερμοκρασία. Έχουμε συνολικά 8 δείγματα και η μεταβλητή Θερμοκρασία παίρνει 4 φορές την τιμή Υψηλή, 2 φορές την τιμή Κανονική και 2 φορές την τιμή Χαμηλή. Για τα 4 δείγματα με τιμή Θερμοκρασία=Υψηλή, 2 έχουν τιμή κλάσης Μέσα και 2 τιμή κλάσης Έξω. Και τα 2 δείγματα με τιμή Θερμοκρασία=Κανονική έχουν τιμή κλάσης Έξω. Για τα 2 δείγματα με τιμή Θερμοκρασία=Χαμηλή, 1 έχει τιμή κλάσης Μέσα και 1 έχει τιμή κλάσης Έξω. Συνεπώς, έχουμε:

$$G(S, \Theta\varepsilon\rho\mu\text{οκρασία}) = E(S) - I(S, \Theta\varepsilon\rho\mu\text{οκρασία}) =$$

$$= E(S) - \frac{4}{8}E(S_{\text{Υψηλή}}) - \frac{2}{8}E(S_{\text{Κανονική}}) - \frac{2}{8}E(S_{\text{Χαμηλή}})$$

όπου:

$$E(S_{\text{Υψηλή}}) = -\frac{2}{4}\log_2\left(\frac{2}{4}\right) - \frac{2}{4}\log_2\left(\frac{2}{4}\right) = 1$$

$$E(S_{\text{Κανονική}}) = -\frac{0}{2}\log_2\left(\frac{0}{2}\right) - \frac{2}{2}\log_2\left(\frac{2}{2}\right) = 0$$

$$E(S_{\text{Χαμηλή}}) = -\frac{1}{2}\log_2\left(\frac{1}{2}\right) - \frac{1}{2}\log_2\left(\frac{1}{2}\right) = 1$$

Επομένως, τελικά:

$$G(S, \Theta\varepsilon\rho\mu\text{οκρασία}) = 0.95 - \frac{4}{8}\cdot 1 - \frac{2}{8}\cdot 0 - \frac{2}{8}\cdot 1 = 0.2$$

Συνεχίζουμε με τη μεταβλητή Υγρασία. Έχουμε συνολικά 8 δείγματα και η μεταβλητή Υγρασία παίρνει 4 φορές την τιμή Υψηλή και 4 φορές την τιμή Κανονική. Για τα 4 δείγματα με τιμή Υγρασία=Υψηλή, 2 έχουν τιμή κλάσης Μέσα και 2 έχουν τιμή κλάσης Έξω. Για τα 2 δείγματα με τιμή Υγρασία=Κανονική, 1 έχει τιμή κλάσης Μέσα και 3 έχουν τιμή κλάσης Έξω. Συνεπώς, έχουμε:

$$G(S, \Upsilon\gamma\rho\alpha\sigma\acute{\iota}\alpha) = E(S) - I(S, \Upsilon\gamma\rho\alpha\sigma\acute{\iota}\alpha) = E(S) - \frac{4}{8}E(S_{\text{Υψηλή}}) - \frac{4}{8}E(S_{\text{Κανονική}})$$

όπου:

$$E(S_{\text{Υψηλή}}) = -\frac{2}{4}\log_2\left(\frac{2}{4}\right) - \frac{2}{4}\log_2\left(\frac{2}{4}\right) = 1$$

$$E(S_{\text{Κανονική}}) = -\frac{1}{4}\log_2\left(\frac{1}{4}\right) - \frac{3}{4}\log_2\left(\frac{3}{4}\right) = 0.81$$

Επομένως, τελικά:

$$G(S, \text{Υγρασία}) = 0.95 - \frac{4}{8}\cdot 1 - \frac{4}{8}\cdot 0.81 = 0.045$$

Τέλος, έχουμε τη μεταβλητή Αέρας. Έχουμε συνολικά 8 δείγματα και η μεταβλητή Αέρας παίρνει 6 φορές την τιμή Ασθενής και 2 φορές την τιμή Δυνατός. Για τα 6 δείγματα με τιμή Αέρας=Ασθενής, 1 έχει τιμή κλάσης Μέσα και 5 έχουν τιμή κλάσης Έξω. Για τα 2 δείγματα με τιμή Αέρας=Δυνατός, 1 έχει τιμή κλάσης Μέσα και 1 έχει τιμή κλάσης Έξω. Συνεπώς,, έχουμε:

$$G(S, \text{Αέρας}) = E(S) - I(S, \text{Αέρας}) = E(S) - \frac{6}{8}E(S_{\text{Ασθενής}}) - \frac{2}{8}E(S_{\text{Δυνατός}})$$

όπου:

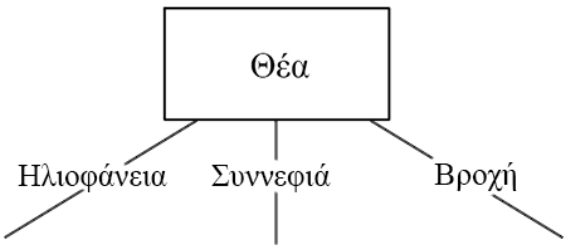
$$E(S_{\text{Ασθενής}}) = -\frac{1}{6}\log_2\left(\frac{1}{6}\right) - \frac{5}{6}\log_2\left(\frac{5}{6}\right) = 0.65$$

$$E(S_{\text{Δυνατόςή}}) = -\frac{1}{2}\log_2\left(\frac{1}{2}\right) - \frac{1}{2}\log_2\left(\frac{1}{2}\right) = 1$$

Επομένως, τελικά

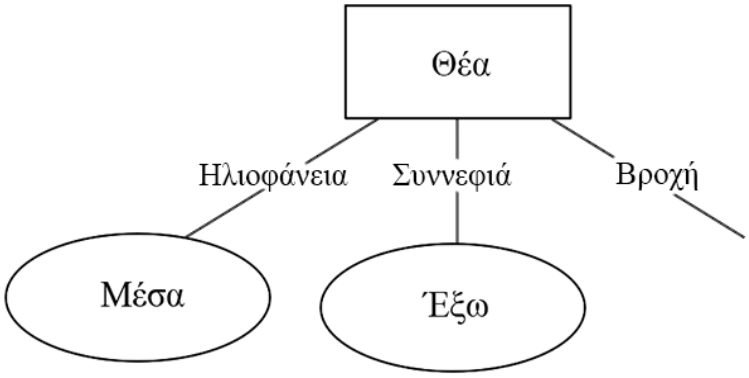
$$G(S, \text{Υγρασία}) = 0.95 - \frac{4}{8}\cdot 0.65 - \frac{4}{8}\cdot 1 = 0.125$$

Από τα παραπάνω, η μεταβλητή Θέα έχει το υψηλότερο πληροφοριακό κέρδος. Επομένως, την επιλέγουμε για ρίζα του δένδρου (Εικόνα 5.2).



Εικόνα ***5.2*** *Αρχικός κόμβος παραδείγματος κατασκευής δένδρου απόφασης με χρήση του αλγορίθμου ID3.*

Έπειτα πρέπει να εξετάσουμε πώς θα συνεχίσει το κάθε κλαδί του δένδρου. Για τις τιμές Ηλιοφάνεια και Συννεφιά παρατηρούμε ότι όλα τα δείγματα ανήκουν στην ίδια κλάση, Μέσα και Έξω, αντίστοιχα. Συνεπώς, οδηγούμαστε σε φύλλα (Εικόνα 5.3).



Εικόνα ***5.3*** *Παράδειγμα κατασκευής δένδρου απόφασης με χρήση του αλγορίθμου ID3 (συνέχεια).*

Μένει να εξετάσουμε τα δείγματα με τιμή Θέα=Βροχή (Πίνακας 5.2).

Α/Α	**Θέα**	Θερμοκρασ**ία**	**Υγρασία**	**Αέρας**	**Κλάση**
4	Βροχή	Κανονική	Υψηλή	Ασθενής	Έξω
7	Βροχή	Κανονική	Κανονική	Ασθενής	Έξω
5	Βροχή	Χαμηλή	Κανονική	Δυνατός	Μέσα

Πίνακας ***5.2*** *Διαχωρισμός βάσει της μεταβλητής Θέα.*

Αρχικά υπολογίζουμε το πληροφοριακό κέρδος των υπόλοιπων μεταβλητών. Για τη μεταβλητή Θερμοκρασία *(*Αέρας*)* έχουμε 2 με τιμή Κανονική (Ασθενής) και 1 με τιμή Χαμηλή (Δυνατός). Για την τιμή Θερμοκρασία=Κανονική (Αέρας=Ασθενής*)* έχουμε 2 φορές την τιμή κλάσης Έξω και 0 φορές την τιμή κλάσης Μέσα, ενώ για την τιμή Θερμοκρασία=Χαμηλή (Αέρας=Δυνατός*)* έχουμε 1 φορά την τιμή κλάσης Μέσα και 0 φορές τιμή κλάσης Έξω. Συνεπώς, έχουμε:

$$G(S_{\text{Βροχή}}, \Theta\varepsilon\rho\mu οκρασία) = G(S_{\text{Βροχή}}, \text{Αέρας})$$

$$G(S_{\text{Βροχή}}, \Theta\varepsilon\rho\mu οκρασία) = E(S_{\text{Βροχή}}) - I(S_{\text{Βροχή}}, \Theta\varepsilon\rho\mu οκρασία) =$$

$$E(S_{\text{Βροχή}}) - \frac{2}{3}E(S_{\text{Κανονική}}) - \frac{1}{3}E(S_{\text{Χαμηλή}})$$

όπου:

$$E(S_{\text{Κανονική}}) = -\frac{0}{2}\log_2\left(\frac{0}{2}\right) - \frac{2}{2}\log_2\left(\frac{2}{2}\right) = 0$$

$$E(S_{\text{Χαμηλή}}) = -\frac{1}{1}\log_2\left(\frac{1}{1}\right) - \frac{0}{1}\log_2\left(\frac{0}{1}\right) = 0$$

Επομένως, τελικά:

$$G(S_{\text{Βροχή}}, \Theta\varepsilon\rho\mu οκρασία) = 0.92 - \frac{2}{3}\cdot 0 - \frac{1}{3}\cdot 0 = 0.92$$

Τέλος, για τη μεταβλητή Υγρασία έχουμε 2 δείγματα με τιμή Κανονική και 1 με τιμή Υψηλή. Για το 1 δείγμα με τιμή Υγρασία=Υψηλή έχουμε 1 φορά τιμή κλάσης Έξω και 0 φορές τιμή κλάσης Μέσα. Για τα 2 δείγματα με τιμή Υγρασία=Κανονική έχουμε 1 φορά τιμή κλάσης Μέσα και 1 φορά τιμή κλάσης Έξω.

$$G(S_{\text{Βροχή}}, \text{Υγρασία}) = E(S_{\text{Βροχή}}) - I(S_{\text{Βροχή}}, \text{Υγρασία}) =$$
$$E(S_{\text{Βροχή}}) - \frac{2}{3}E(S_{\text{Κανονική}}) - \frac{1}{3}E(S_{\text{Υψηλή}})$$

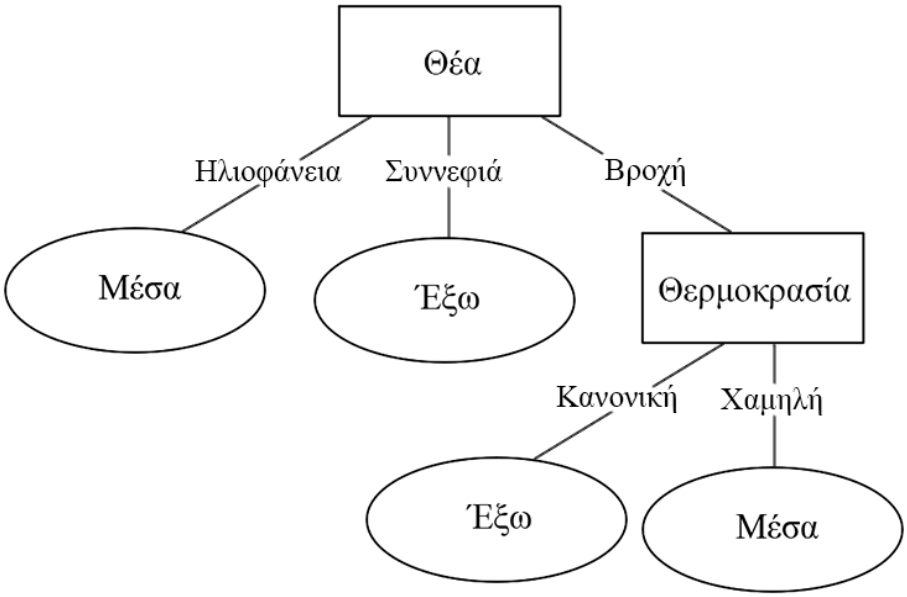
όπου:

$$E(S_{\text{Κανονική}}) = -\frac{1}{2}\log_2\left(\frac{1}{2}\right) - \frac{1}{2}\log_2\left(\frac{1}{2}\right) = 1$$
$$E(S_{\text{Υψηλή}}) = -\frac{1}{1}\log_2\left(\frac{1}{1}\right) - \frac{0}{1}\log_2\left(\frac{0}{1}\right) = 0$$

Επομένως, τελικά:

$$G(S_{\text{Βροχή}}, \text{Υγρασία}) = 0.92 - \frac{2}{3}\cdot 1 - \frac{1}{3}\cdot 0 = 0.25$$

Επιλέγουμε τη μεταβλητή με το μεγαλύτερο πληροφοριακό κέρδος, δηλαδή ή τη μεταβλητή Θερμοκρασία ή τη μεταβλητή Αέρας, αφού έχουν ίσο πληροφοριακό κέρδος. Στην Εικόνα 5.4 φαίνεται το τελικό δένδρο απόφασης με χρήση του αλγορίθμου ID3.



Εικόνα ***5.4*** *Δένδρο απόφασης παραδείγματος με χρήση του αλγορίθμου ID3.*

5.1.2.3 Κατασκευή Δένδρου Απόφασης – Gini Index

Ένας άλλος τρόπος κατασκευής δένδρων απόφασης γίνεται με τη χρήση του Gini Index για την επιλογή των κόμβων. Το Gini Index μετράει την ανισότητα μεταξύ τιμών μιας κατανομής συχνοτήτων. Οι τιμές του κυμαίνονται από 0 έως 1, με το 0 να δηλώνει πλήρη ισότητα και το 1 να δηλώνει πλήρη ανισότητα. Για ένα σύνολο δεδομένων S με m δείγματα και k κλάσεις το gini(S) υπολογίζεται με τον τύπο:

$$gini(S) = 1 - \sum_{j=1}^{k} p_j^2$$

όπου p_j είναι η πιθανότητα εμφάνισης της κλάσης j στο σύνολο δεδομένων S. Αν το S διαχωριστεί σε S_1 και S_2, τότε:

$$gini(S) = \frac{n_1}{n} gini(S_1) + \frac{n_2}{n} gini(S_2)$$

όπου n_1 και n_2 είναι το σύνολο των δειγμάτων στο S_1 και S_2 αντίστοιχα. Το πλεονέκτημα της μεθόδου αυτής είναι ότι για τον υπολογισμό απαιτείται μόνο ο διαχωρισμός των κλάσεων σε κάθε υποσύνολο. Το καλύτερο χαρακτηριστικό είναι εκείνο με τη μικρότερη τιμή Gini. Ας δούμε ένα παράδειγμα χρήσης του Gini Index για κατασκευή δένδρου απόφασης.

Έστω το σύνολο δεδομένων που φαίνεται παρακάτω (Πίνακας 5.3).

Α/Α	**Θέα**	Θερμοκρασία	**Υγρασία**	**Αέρας**	**Κλάση**
1	Ηλιοφάνεια	Υψηλή	Υψηλή	Ασθενής	Μέσα
2	Ηλιοφάνεια	Υψηλή	Υψηλή	Δυνατός	Μέσα
3	Συννεφιά	Υψηλή	Υψηλή	Ασθενής	Έξω
4	Βροχή	Κανονική	Υψηλή	Ασθενής	Έξω
5	Βροχή	Χαμηλή	Κανονική	Δυνατός	Μέσα
6	Συννεφιά	Χαμηλή	Κανονική	Ασθενής	Έξω

Πίνακας *5.3* *Σύνολο δεδομένων παραδείγματος κατασκευής δένδρου απόφασης με Gini Index.*

Α/Α	**Θέα**	Θερμοκρασία	**Υγρασία**	**Αέρας**	**Κλάση**
1	Ηλιοφάνεια	Υψηλή	Υψηλή	Ασθενής	Μέσα
2	Ηλιοφάνεια	Υψηλή	Υψηλή	Δυνατός	Μέσα
3	Συννεφιά	Υψηλή	Υψηλή	Ασθενής	Έξω
4	Βροχή	Κανονική	Υψηλή	Ασθενής	Έξω
5	Βροχή	Χαμηλή	Κανονική	Δυνατός	Μέσα
6	Συννεφιά	Χαμηλή	Κανονική	Ασθενής	Έξω

Πίνακας *5.4* *Διαχωρισμός βάσει της μεταβλητής Θέα.*

Ξεκινάμε από τη μεταβλητή Θέα. Αρχικά κάνουμε τον διαχωρισμό με βάση τις τιμές της μεταβλητής (Πίνακας 5.4), οπότε και έχουμε:

$$gini(\text{Ηλιοφάνεια}) = 1 - \left(p^2_{\text{Μέσα}} + p^2_{\text{Έξω}} \right) = 1 - \left(1^2 + 0\right) = 1 - 1 = 0 \qquad (\text{Μέσα})$$

$$gini(\text{Συννεφιά}) = 1 - \left(p^2_{\text{Μέσα}} + p^2_{\text{Έξω}} \right) = 1 - \left(0 + 1^2\right) = 1 - 1 = 0 \qquad (\text{Έξω})$$

$$gini(\text{Βροχή}) = 1 - \left(p^2_{\text{Μέσα}} + p^2_{\text{Έξω}} \right) = 1 - \left(\left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^2 \right) = 1 - \frac{1}{2} = 0.5 \quad (\text{Μέσα}, \text{Έξω})$$

Συνεπώς, για τη μεταβλητή Θέα καταλήγουμε:

$$gini(\text{Θέα}) = \frac{2}{6} gini(\text{Ηλιοφάνεια}) + \frac{2}{6} gini(\text{Συννεφιά}) + \frac{2}{6} gini(\text{Βροχή})$$
$$= \frac{2}{6} \cdot 0 + \frac{2}{6} \cdot 0 + \frac{2}{6} \cdot 0.5 = \frac{1}{6} = 0.16$$

Α/Α	Θέα	Θερμοκρασία	Υγρασία	Αέρας	Κλάση
1	Ηλιοφάνεια	Υψηλή	Υψηλή	Ασθενής	Μέσα
2	Ηλιοφάνεια	Υψηλή	Υψηλή	Δυνατός	Μέσα
3	Συννεφιά	Υψηλή	Υψηλή	Ασθενής	Έξω
4	Βροχή	Κανονική	Υψηλή	Ασθενής	Έξω
5	Βροχή	Χαμηλή	Κανονική	Δυνατός	Μέσα
6	Συννεφιά	Χαμηλή	Κανονική	Ασθενής	Έξω

Πίνακας 5.5 *Διαχωρισμός βάσει της μεταβλητής Θερμοκρασία.*

Συνεχίζουμε με τη μεταβλητή Θερμοκρασία. Κάνουμε τον διαχωρισμό με βάση τις τιμές της μεταβλητής (Πίνακας 5.5), οπότε και έχουμε:

$$gini(\text{Υψηλή}) = 1 - \left(p^2_{\text{Μέσα}} + p^2_{\text{Έξω}} \right) = 1 - \left(\left(\frac{2}{3}\right)^2 + \left(\frac{1}{3}\right)^2 \right) = 1 - \frac{5}{9} = \frac{4}{9} = 0.55 \quad (\text{Μέσα}, \text{Έξω})$$

$$gini(\text{Κανονική}) = 1 - \left(p^2_{\text{Μέσα}} + p^2_{\text{Έξω}} \right) = 1 - \left(0 + 1^2\right) = 1 - 1 = 0 \quad (\text{Έξω})$$

$$gini(\text{Χαμηλή}) = 1 - \left(p^2_{\text{Μέσα}} + p^2_{\text{Έξω}} \right) = 1 - \left(\left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^2 \right) = 1 - \frac{1}{2} = 0.5 \quad (\text{Μέσα}, \text{Έξω})$$

Συνεπώς, για τη μεταβλητή Θερμοκρασία καταλήγουμε:

$$gini(\text{Θερμοκρασία}) = \frac{3}{6} gini(\text{Υψηλή}) + \frac{1}{6} gini(\text{Κανονική}) + \frac{2}{6} gini(\text{Χαμηλή}) =$$

$$= \frac{2}{6} \cdot 0.55 + \frac{2}{6} \cdot 0 + \frac{2}{6} \cdot 0.5 = \frac{1}{6} = 0.35$$

Α/Α	Θέα	Θερμοκρασία	Υγρασία	Αέρας	Κλάση
1	Ηλιοφάνεια	Υψηλή	Υψηλή	Ασθενής	Μέσα
2	Ηλιοφάνεια	Υψηλή	Υψηλή	Δυνατός	Μέσα
3	Συννεφιά	Υψηλή	Υψηλή	Ασθενής	Έξω
4	Βροχή	Κανονική	Υψηλή	Ασθενής	Έξω
5	Βροχή	Χαμηλή	Κανονική	Δυνατός	Μέσα
6	Συννεφιά	Χαμηλή	Κανονική	Ασθενής	Έξω

Πίνακας 5.6 *Διαχωρισμός βάσει της μεταβλητής Υγρασία.*

Για τη μεταβλητή Υγρασία ο διαχωρισμός φαίνεται παραπάνω (Πίνακας 5.6). Έχουμε:

$$gini(\text{Υψηλή}) = 1 - \left(p^2_{\text{Μέσα}} + p^2_{\text{Έξω}} \right) = 1 - \left(\left(\frac{2}{4}\right)^2 + \left(\frac{2}{4}\right)^2 \right) = 1 - \frac{1}{2} = 0.5 \quad (\text{Μέσα}, \text{Έξω})$$

$$gini(\text{Κανονική}) = 1 - \left(p^2_{\text{Μέσα}} + p^2_{\text{Έξω}} \right) = 1 - \left(\left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^2 \right) = 1 - \frac{1}{2} = 0.5 \quad (\text{Μέσα}, \text{Έξω})$$

Συνεπώς, για τη μεταβλητή Υγρασία καταλήγουμε:

$$gini(\text{Υγρασία}) = \frac{4}{6} gini(\text{Υψηλή}) + \frac{2}{6} gini(\text{Κανονική}) = \frac{4}{6} \cdot 0.5 + \frac{2}{6} \cdot 0.5 = \frac{3}{6} = 0.5$$

Α/Α	**Θέα**	**Θερμοκρασία**	**Υγρασία**	**Αέρας**	**Κλάση**
1	Ηλιοφάνεια	Υψηλή	Υψηλή	Ασθενής	Μέσα
2	Ηλιοφάνεια	Υψηλή	Υψηλή	Δυνατός	Μέσα
3	Συννεφιά	Υψηλή	Υψηλή	Ασθενής	Έξω
4	Βροχή	Κανονική	Υψηλή	Ασθενής	Έξω
5	Βροχή	Χαμηλή	Κανονική	Δυνατός	Μέσα
6	Συννεφιά	Χαμηλή	Κανονική	Ασθενής	Έξω

Πίνακας ***5.7*** *Διαχωρισμός βάσει της μεταβλητής Αέρας.*

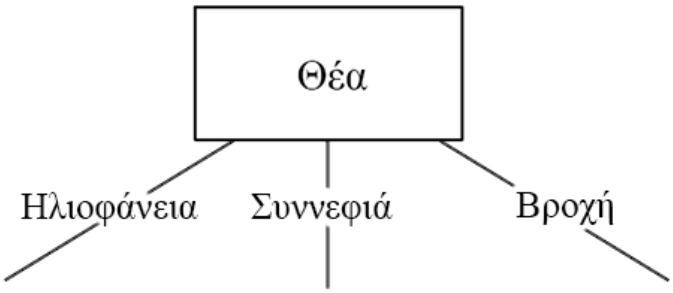
Τέλος, για τη μεταβλητή Αέρας ο διαχωρισμός φαίνεται στον παραπάνω πίνακα (Πίνακας 5.7). Έχουμε:

$$gini(\text{Ασθενής}) = 1 - \left(p^2_{\text{Μέσα}} + p^2_{\text{Έξω}} \right) = 1 - \left(\left(\frac{1}{4}\right)^2 + \left(\frac{3}{4}\right)^2 \right) = 1 - \frac{10}{16} = \frac{6}{16} = 0.375 \qquad (\text{Μέσα}, \text{Έξω})$$

$$gini(\text{Δυνατός}) = 1 - \left(p^2_{\text{Μέσα}} + p^2_{\text{Έξω}} \right) = 1 - \left(1^2 + 0\right) = 1 - 1 = 0 \qquad (\text{Μέσα}, \text{Έξω})$$

Συνεπώς, για τη μεταβλητή Αέρας καταλήγουμε:

$$gini(\text{Αέρας}) = \frac{4}{6} gini(\text{Ασθενής}) + \frac{2}{6} gini(\text{Δυνατός}) = \frac{4}{6} \cdot 0.375 + \frac{2}{6} \cdot 0 = 0.25$$



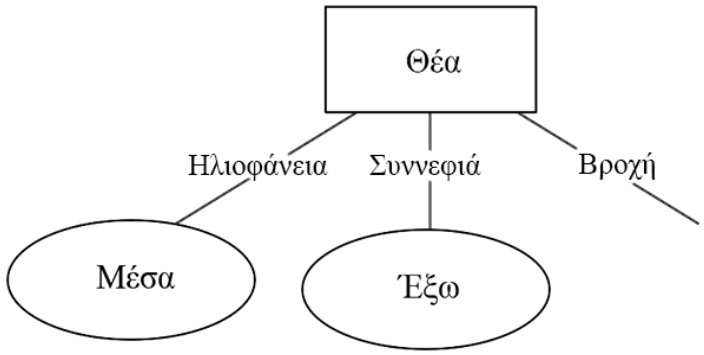
Εικόνα ***5.5*** *Αρχικός κόμβος παραδείγματος κατασκευής δένδρου απόφασης με χρήση Gini Index.*

Επιλέγουμε ως αρχικό κόμβο το χαρακτηριστικό με μικρότερη τιμή Gini, δηλαδή τη μεταβλητή Θέα (Εικόνα 5.5). Στη συνέχεια πρέπει να εξεταστούν οι τιμές Ηλιοφάνεια, Συννεφιά και Βροχή ξεχωριστά. Συνεπώς, ο αρχικός πίνακας πρέπει να διασπαστεί σε 3 τμήματα, όπως φαίνεται παρακάτω (Πίνακας 5.8).

Α/Α	Θέα	Θερμοκρασία	Υγρασία	Αέρας	Κλάση
1	Ηλιοφάνεια	Υψηλή	Υψηλή	Ασθενής	Μέσα
2	Ηλιοφάνεια	Υψηλή	Υψηλή	Δυνατός	Μέσα
3	Συννεφιά	Υψηλή	Υψηλή	Ασθενής	Έξω
6	Συννεφιά	Χαμηλή	Κανονική	Ασθενής	Έξω
4	Βροχή	Κανονική	Υψηλή	Ασθενής	Έξω
5	Βροχή	Χαμηλή	Κανονική	Δυνατός	Μέσα

Πίνακας ***5.8*** *Διάσπαση αρχικού πίνακα.*

Για τις τιμές Ηλιοφάνεια και Συννεφιά, παρατηρούμε ότι όλα τα δείγματα ανήκουν στην ίδια κλάση, Μέσα και Έξω, αντίστοιχα. Συνεπώς, οδηγούμαστε σε φύλλα (Εικόνα 5.6).



Εικόνα ***5.6*** *Παράδειγμα κατασκευής δένδρου απόφασης με χρήση Gini Index (συνέχεια).*

Για την τιμή Βροχή θα πρέπει να εξετάσουμε περαιτέρω τον διαχωρισμό. Αρκεί να εξετάσουμε μόνο τα δείγματα, για τα οποία η μεταβλητή Θέα έχει τιμή Βροχή. Ο Πίνακας 5.9 παρουσιάζει τα δείγματα που μας ενδιαφέρουν.

Α/Α	Θέα	Θερμοκρασία	Υγρασία	Αέρας	Κλάση
4	Βροχή	Κανονική	Υψηλή	Ασθενής	Έξω
5	Βροχή	Χαμηλή	Κανονική	Δυνατός	Μέσα

Πίνακας ***5.9*** *Διαχωρισμός βάσει της μεταβλητής Θέα.*

Ξεκινάμε και πάλι από τη μεταβλητή Θέα. Ο Πίνακας 5.9 παρουσιάζει τον διαχωρισμό με βάση τις τιμές της μεταβλητής, οπότε και έχουμε:

$$gini(\text{Βροχή}) = 1 - \left(p^2_{\text{Μέσα}} + p^2_{\text{Έξω}} \right) = 1 - \left(\left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^2 \right) = 1 - \frac{1}{2} = 0.5 \quad (\text{Μέσα}, \text{Έξω})$$

Συνεπώς, για τη μεταβλητή Θέα καταλήγουμε:

$$gini(\text{Θέα}) = \frac{2}{2} gini(\text{Βροχή}) = 1 \cdot 0.5 = 0.5$$

Παρατηρούμε ότι για τις μεταβλητές Θερμοκρασία, Υγρασία και Αέρας έχουμε παρόμοιο διαχωρισμό, δηλαδή αντιστοιχία διαφορετικής τιμής μεταβλητής και τιμής κλάσης. Συνεπώς, ο υπολογισμός γίνεται με τον ίδιο τρόπο και οι τιμές που θα προκύψουν θα είναι ίσες. Αρκεί, λοιπόν, να υπολογίσουμε για μια από αυτές τις μεταβλητές το Gini Index. Έστω για τη μεταβλητή Θερμοκρασία.

Α/Α	**Θέα**	Θερμοκρασία	**Υγρασία**	**Αέρας**	**Κλάση**
4	Βροχή	Κανονική	Υψηλή	Ασθενής	Έξω
5	Βροχή	Χαμηλή	Κανονική	Δυνατός	Μέσα

Πίνακας *5.10* *Διαχωρισμός βάσει της μεταβλητής Θερμοκρασίας.*

Ο Πίνακας 5.10 παρουσιάζει τον διαχωρισμό με βάση τις τιμές της μεταβλητής, οπότε και έχουμε:

$$gini(\text{Κανονική}) = 1 - \left(p^2_{\text{Μέσα}} + p^2_{\text{Έξω}} \right) = 1 - \left(0 + 1^2\right) = 1 - 1 = 0 \qquad (\text{Έξω})$$

$$gini(\text{Χαμηλή}) = 1 - \left(p^2_{\text{Μέσα}} + p^2_{\text{Έξω}} \right) = 1 - \left(1^2 + 0\right) = 1 - 1 = 0 \qquad (\text{Μέσα})$$

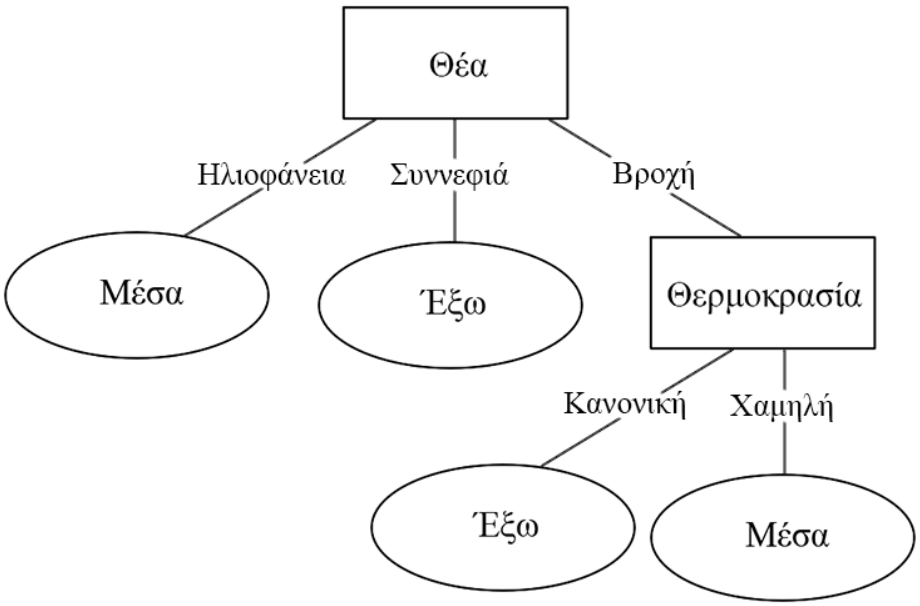
Συνεπώς, για τη μεταβλητή Θερμοκρασία καταλήγουμε:

$$gini(\text{Θερμοκρασία}) = \frac{1}{2} gini(\text{Κανονική}) + \frac{1}{2} gini(\text{Χαμηλή}) = \frac{1}{2} \cdot 0 + \frac{1}{2} \cdot 0 = 0$$

Αν υπολογίσουμε το Gini Index και για τις μεταβλητές Υγρασία και Αέρας, θα καταλήξουμε:

$$gini(\text{Θερμοκρασία}) = gini(\text{Υγρασία}) = gini(\text{Αέρας}) = 0$$

Υπάρχει ισοπαλία μεταξύ τους και επιλέγουμε τυχαία τη μεταβλητή Θερμοκρασία. Επομένως, τελικά το δένδρο απόφασης, που προκύπτει, φαίνεται στην Εικόνα 5.7.



Εικόνα *5.7* *Δένδρο απόφασης παραδείγματος με χρήση Gini Index.*

Να σημειώσουμε ότι τα δένδρα απόφασης, τα οποία κατασκευάσαμε με τον αλγόριθμο ID3 και το Gini Index, τυχαία βγήκαν ίδια.

5.2 Πρόβλεψη

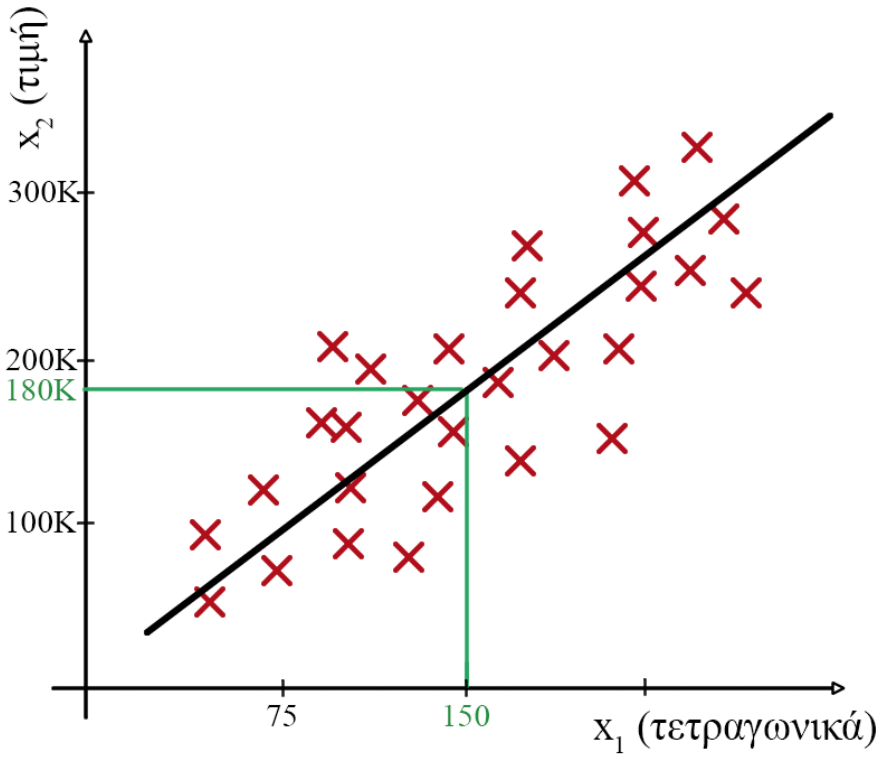
5.2.1 Διαφορά Κατηγοριοποίησης και Πρόβλεψης

Με μια πιο αφαιρετική ματιά, η κατηγοριοποίηση και η πρόβλεψη δεν φαίνονται να διαφέρουν και πολύ. Η βασική διαφορά κατηγοριοποίησης και πρόβλεψης είναι ότι στην κατηγοριοποίηση υπάρχει ένα πεπερασμένο σύνολο διακριτών κλάσεων. Τα δείγματα χρησιμοποιούνται με σκοπό τη δημιουργία ενός μοντέλου, το οποίο να είναι σε θέση να κατηγοριοποιήσει νέα δείγματα. Στην πρόβλεψη η τιμή που προβλέπει το μοντέλο είναι συνεχής και δεν ανήκει σε κάποιο προκαθορισμένο, πεπερασμένο σύνολο. Όπως αναφέραμε και προηγουμένως, στο παράδειγμα στην Εικόνα 5.1, έχουμε πεπερασμένο αριθμό τιμών κλάσης (Επιβίωσε, Απεβίωσε) και, συνεπώς, έχουμε δένδρο απόφασης, το οποίο κάνει κατηγοριοποίηση. Αν οι τιμές της μεταβλητής στόχου δεν ήταν πεπερασμένες, τότε θα είχαμε ένα δένδρο παλινδρόμησης, το οποίο θα έκανε πρόβλεψη.

5.2.2 Γραμμική Παλινδρόμηση

5.2.2.1 Περιγραφή, Ορισμοί και Συμβολισμοί

Η γραμμική παλινδρόμηση αποτελεί την απλούστερη μορφή παλινδρόμησης. Όπως περιγράψαμε και στο Κεφάλαιο 1, η παλινδρόμηση (*regression*) έχει ως στόχο την εκπαίδευση (*training*) μιας συνάρτησης, η οποία απεικονίζει ένα αντικείμενο σε μία πραγματική μεταβλητή.



Εικόνα *5.8* *Παράδειγμα (γραμμικής) παλινδρόμησης.*

Στην Εικόνα 5.8 παρουσιάζουμε ένα απλό παράδειγμα γραμμικής παλινδρόμησης. Οι μεταβλητές είναι τα τετραγωνικά ενός σπιτιού και η τιμή πώλησής του σε χιλιάδες Ευρώ. Η γραμμική παλινδρόμηση προσαρμόζει μια ευθεία στα δείγματα του συνόλου δεδομένων, τα οποία σηματοδοτούνται με κόκκινο Χ. Η προσαρμογή γίνεται με βάση μια συνάρτηση κόστους, την τιμή της οποία θέλουμε να ελαχιστοποιήσουμε. Έχοντας τη βέλτιστη ευθεία, δηλαδή την ευθεία που ελαχιστοποιεί την τιμή της συνάρτησης κόστους, μπορούμε να δώσουμε μια προσεγγιστικά καλή απάντηση σε ερωτήματα της μορφής: «Σε τι τιμές πωλούνται σπίτια των 150 τετραγωνικών;». Δηλαδή, δοθέντων των τιμών της μεταβλητής στόχου (στην περίπτωσή μας η τιμή πώλησης) για κάθε δείγμα, επιδιώκουμε να προβλέψουμε τις τιμές της μεταβλητής στόχου για νέα δείγματα.

Στο πλαίσιο παρουσίασης της γραμμικής παλινδρόμησης θα ορίσουμε κάποιους συμβολισμούς. Θα συμ-

βολίσουμε με *m* το πλήθος των δειγμάτων του συνόλου εκπαίδευσης. Με X θα συμβολίσουμε τις μεταβλητές εισόδου, ενώ με *y* τη μεταβλητή στόχου. Με β θα αναφερόμαστε στις παραμέτρους του μοντέλου.

5.2.2.2 Συνάρτηση Κόστους

Ορίζουμε τη συνάρτηση κόστους *F*, η οποία δίνεται από τον παρακάτω τύπο:

$$F(\beta_0, \beta_1, ..., \beta_n) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\beta (x^{(i)}) - y^{(i)} \right)^2$$

Η βασική ιδέα είναι ότι θέλουμε να ελαχιστοποιήσουμε τη συνάρτηση κόστους ως προς τα β_j, δηλαδή θέλουμε:

$$\underset{\beta_0, \beta_1, ..., \beta_n}{minimize} F(\beta_0, \beta_1, ..., \beta_n)$$

έτσι ώστε η τιμή της υπόθεσης h_β, δηλαδή η πρόβλεψη, να είναι όσο πιο κοντά στην τιμή της πραγματικής μεταβλητής στόχου *y*. Η παραπάνω συνάρτηση κόστους είναι η πιο διαδεδομένη και είναι γνωστή ως συνάρτηση τετραγωνικού σφάλματος.

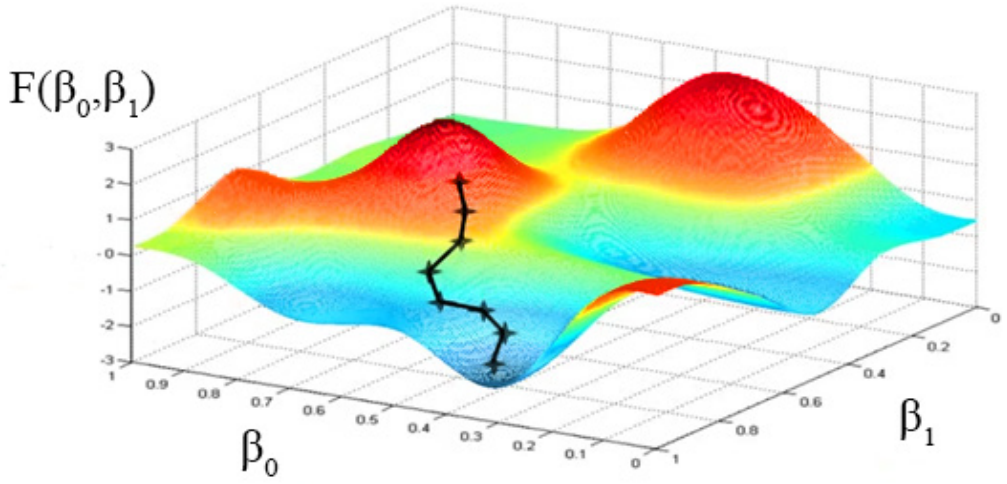
5.2.2.3 Αλγόριθμος Σταδιακής Καθόδου (Gradient Descent)

Στόχος μας είναι να ελαχιστοποιήσουμε την τιμή της συνάρτησης κόστους *F*. Αυτό μπορεί να επιτευχθεί με τη σωστή επιλογή τιμών για τις παραμέτρους β_j. Η εξαντλητική, χειρωνακτική αναζήτηση είναι απαγορευτικά χρονοβόρα. Ο αλγόριθμος σταδιακής καθόδου έχει ως στόχο την επιλογή των κατάλληλων β_j, ώστε η τιμή της συνάρτησης κόστους να ελαχιστοποιηθεί. Ο αλγόριθμος συνοπτικά λειτουργεί ως εξής:

- επιλέγονται τυχαίες τιμές για τα β_j,
- οι τιμές τους μεταβάλλονται επαναληπτικά και με προκαθορισμένο τρόπο, ώστε η συνάρτηση σε κάθε βήμα να ελαχιστοποιείται.

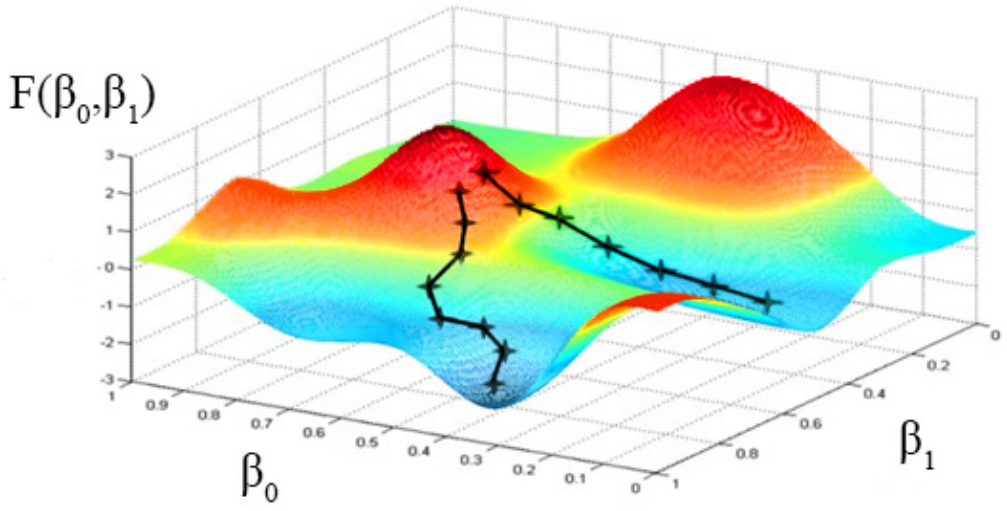
Ας δούμε πρώτα τη γενική εικόνα του πώς λειτουργεί ο αλγόριθμος, πριν προχωρήσουμε στους τύπους. Για τον σκοπό αυτό θα χρησιμοποιήσουμε ένα απλό παράδειγμα με μία μόνο μεταβλητή εισόδου και, συνεπώς, δυο παραμέτρους β_0 και β_1. Φανταστείτε ότι βρισκόμαστε σε ένα συγκεκριμένο σημείο στο παρακάτω γράφημα (Εικόνα 5.9), έστω σε κάποιον από τους δυο κόκκινους λόφους, και ότι στόχος μας είναι να προχωρήσουμε προς κάποιο χαμηλότερο σημείο.

Το πρώτο πράγμα που πρέπει να κάνουμε είναι να σκεφτούμε και να αναρωτηθούμε: αν μπορούσαμε να κάνουμε ένα μικρό βήμα, προς ποια κατεύθυνση θα ήταν, ώστε να μας οδηγήσει σε χαμηλότερο σημείο; Με παρόμοια λογική λειτουργεί και ο αλγόριθμος σταδιακής καθόδου, ως προς την τιμή της συνάρτησης κόστους (Εικόνα 5.9). Όπως θα δούμε παρακάτω, η λογική αυτή υλοποιείται μέσα από τις μερικές παραγώγους των β_0 και β_1. Να υπενθυμίσουμε ότι η τιμή της παραγώγου δηλώνει την κλίση μιας ευθείας, και, συνεπώς, στην περίπτωσή μας την κατεύθυνση της διαδρομής που θα ακολουθήσει ο αλγόριθμος σε κάθε βήμα του.



Εικόνα ***5.9*** *Παράδειγμα παρουσίασης της βασικής ιδέας του αλγορίθμου σταδιακής καθόδου.*

Ο αλγόριθμος σταδιακής καθόδου έχει μια ενδιαφέρουσα ιδιότητα. Από διαφορετικό σημείο εκκίνησης, είναι δυνατό να προκύψει διαφορετικό τελικό σημείο (ένα τέτοιο παράδειγμα βλέπουμε στην Εικόνα 5.10).



Εικόνα ***5.10*** *Παράδειγμα παρουσίασης της βασικής ιδέας του αλγορίθμου σταδιακής καθόδου (συνέχεια).*

Ο αλγόριθμος είναι ο ακόλουθος:

επανέλαβε μέχρι να υπάρξει σύγκλιση{

$$\beta_j \longleftarrow \beta_j - a\frac{\partial}{\partial \beta_j}F(\beta_0,\beta_1,\ldots,\beta_n)$$

ενημέρωσε τα β_j ταυτόχρονα (στο τέλος)

}

Η παράμετρος α ονομάζεται παράμετρος μάθησης και δηλώνει το πόσο μεγάλο θα είναι το βήμα που θα κάνει ο αλγόριθμος σε κάθε επανάληψη. Συνήθως, η παράμετρος α έχει σταθερή τιμή και δεν μεταβάλλεται κατά τη διάρκεια εκτέλεσης του αλγορίθμου. Η μερική παράγωγος ως προς β_j καθορίζει την κατεύθυνση προς την οποία ο αλγόριθμος θα προχωρήσει στο τρέχον βήμα. Τέλος, η ενημέρωση των παραμέτρων β_j γίνεται στο τέλος κάθε επανάληψης. Ο αντίστοιχος ψευδοκώδικας για τον τρόπο ενημέρωσης για δυο παραμέτρους β_0 και β_1 είναι ο ακόλουθος:

$$tmp_0 \longleftarrow \beta_0 - a\frac{\partial}{\partial \beta_0}F(\beta_0, \beta_1)$$

$$tmp_1 \longleftarrow \beta_1 - a\frac{\partial}{\partial \beta_1}F(\beta_0, \beta_1)$$

$$\beta_0 \longleftarrow tmp_0$$

$$\beta_1 \longleftarrow tmp_1$$

Δηλαδή, αφού υπολογίσουμε τη νέα τιμή του β_0, έστω tmp_0, χρησιμοποιούμε το β_0 για τον υπολογισμό της νέας τιμής του β_1, έστω tmp_1, και όχι το tmp_0. Οι νέες τιμές θα χρησιμοποιηθούν στην επόμενη επανάληψη.

5.2.2.4 Σταδιακή Κάθοδος στη Γραμμική Παλινδρόμηση

Στις προηγούμενες υποενότητες παρουσιάσαμε ξεχωριστά τη γραμμική παλινδρόμηση και τον αλγόριθμο σταδιακής καθόδου. Ας δούμε τώρα πώς εφαρμόζεται ο αλγόριθμος σταδιακής καθόδου σε συνδυασμό με τη γραμμική παλινδρόμηση. Έστω ένα μοντέλο γραμμικής παλινδρόμησης με δυο παραμέτρους β_0 και β_1, και η υπόθεση δίνεται ως:

$$h_{\mathsf{b}}(x) = \mathsf{b}_0 + \mathsf{b}_1 x,$$

η οποία ορίζει μια ευθεία $y = ax + b$ με κλίση $a = \beta_1$ και σταθερό όρο $b = \beta_0$. Για το συγκεκριμένο μοντέλο γραμμικής παλινδρόμησης, η συνάρτηση κόστους είναι:

$$F(\beta_0, \beta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\beta(x^{(i)}) - y^{(i)}\right)^2$$

Ουσιαστικά, με χρήση του αλγορίθμου σταδιακής καθόδου θα ελαχιστοποιήσουμε τη συνάρτηση κόστους F. Αρχικά, πρέπει να υπολογίσουμε τις μερικές παραγώγους. Έχουμε:

$$\frac{\partial}{\partial \beta_j}F(\beta_0, \beta_1) = \frac{\partial}{\partial \beta_j}\frac{1}{2m}\sum_{i=1}^{m}\left(h_\beta(x^{(i)}) - y^{(i)}\right)^2 = \frac{\partial}{\partial \beta_j}\frac{1}{2m}\sum_{i=1}^{m}\left(\beta_0 + \beta_1 x^{(i)} - y^{(i)}\right)^2$$

$$j = 0 : \frac{\partial}{\partial \beta_0}F(\beta_0, \beta_1) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\beta(x^{(i)}) - y^{(i)}\right)$$

$$j = 1 : \frac{\partial}{\partial \beta_1}F(\beta_0, \beta_1) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\beta(x^{(i)}) - y^{(i)}\right) \cdot x^{(i)}$$

Οπότε, με βάση τους παραπάνω υπολογισμούς, ο αλγόριθμος γίνεται ως εξής:

επανέλαβε μέχρι να υπάρξει σύγκλιση{

$$\beta_0 \longleftarrow \beta_0 - a\frac{1}{m}\sum_{i=1}^{m}\left(h_\beta(x^{(i)}) - y^{(i)}\right)$$

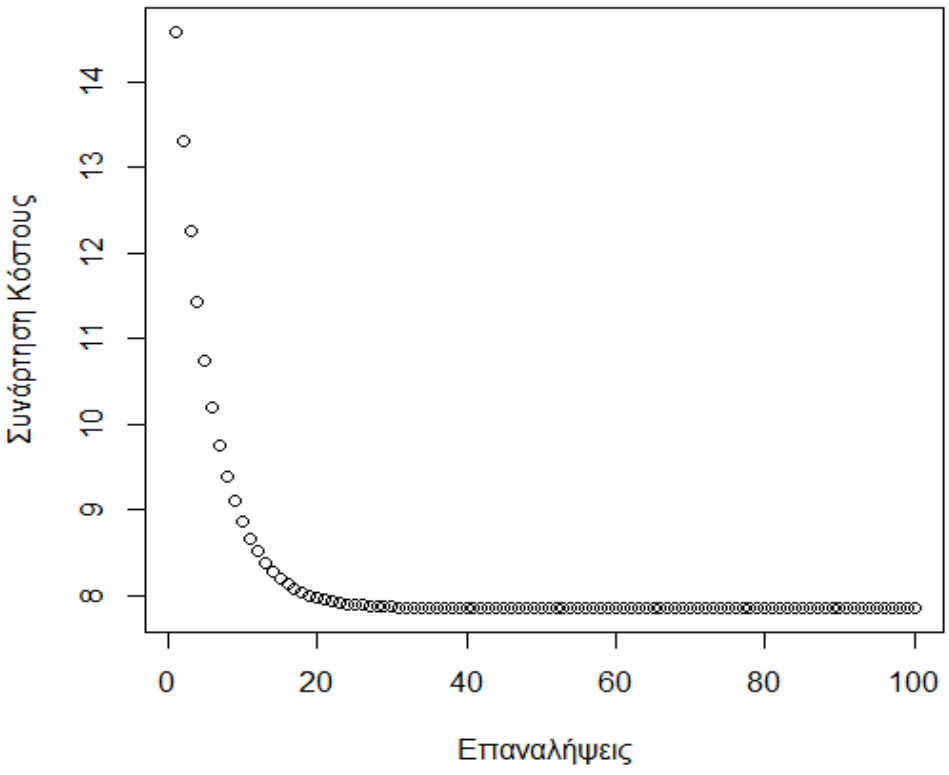
$$\beta_1 \longleftarrow \beta_1 - a\frac{1}{m}\sum_{i=1}^{m}\left(h_\beta(x^{(i)}) - y^{(i)}\right)\cdot x^{(i)}$$

ενημέρωσε τα β_j ταυτόχρονα (στο τέλος)

}

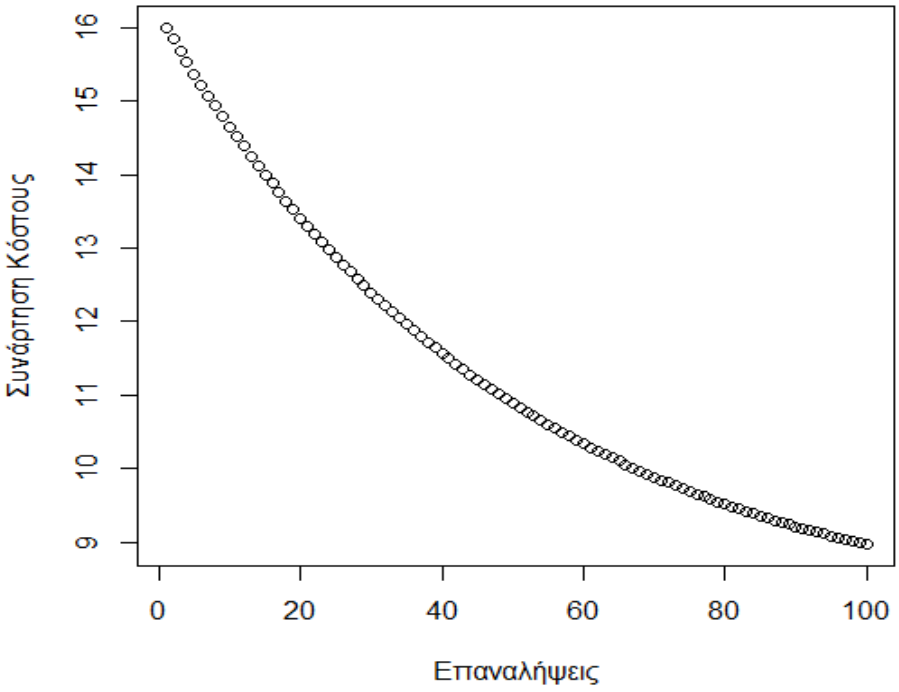
5.2.2.5 Παράμετρος Μάθησης

Η παράμετρος μάθησης είναι η παράμετρος α, που είδαμε στον αλγόριθμο σταδιακής καθόδου. Το βασικό ερώτημα σε αυτό το σημείο είναι με τι κριτήρια επιλέγουμε την τιμή αυτής της παραμέτρου. Αρχικά, ας δούμε πώς μπορούμε να βεβαιωθούμε ότι ο αλγόριθμός μας λειτουργεί όντως σωστά. Θα πρέπει να απεικονίσουμε τη συνάρτηση κόστους *F* ως προς τον αριθμό επαναλήψεων του αλγορίθμου. Καθώς το μέγεθος των επαναλήψεων μεγαλώνει, περιμένουμε η συνάρτηση κόστους να ακολουθεί φθίνουσα πορεία (Εικόνα 5.11).

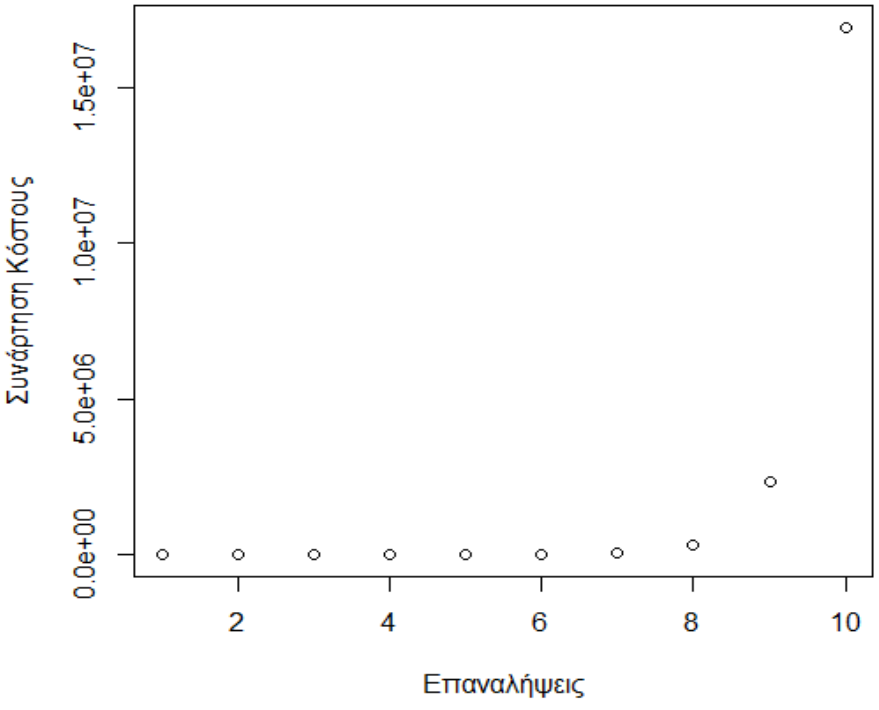


Εικόνα *5.11* *Παράδειγμα σωστής εκτέλεσης αλγορίθμου σταδιακής καθόδου.*

Αντίθετα, αν έχουμε μια γραφική απεικόνιση, όπως στην Εικόνα 5.12, τότε ο αλγόριθμος δεν λειτουργεί σωστά. Μια πιθανή αιτία μπορεί να είναι και η τιμή της παραμέτρου μάθησης. Στη γραφική αναπαράσταση του αλγορίθμου, η παράμετρος μάθησης καθορίζει το πόσο μεγάλο θα είναι το βήμα που θα κάνουμε. Αν η τιμή είναι πολύ μικρή, τότε ο αλγόριθμος θα αργήσει πολύ να βρει κάποιο ελάχιστο (Εικόνα 5.12). Αντίθετα, αν είναι πολύ μεγάλη, υπάρχει περίπτωση να υπερπηδήσει το ελάχιστο και, μάλιστα, να αρχίσει να κινείται προς υψηλότερες τιμές της συνάρτησης κόστους (Εικόνα 5.13).



Εικόνα ***5.12*** *Παράδειγμα σταδιακής καθόδου με πολύ μικρή τιμή παραμέτρου μάθησης.*



Εικόνα ***5.13*** *Παράδειγμα σταδιακής καθόδου με πολύ μεγάλη τιμή παραμέτρου μάθησης.*

Δυστυχώς, δεν υπάρχει κάποιος κανόνας για την επιλογή της παραμέτρου μάθησης. Ο μόνος τρόπος είναι μέσα από δοκιμές, προσέχοντας ταυτόχρονα η γραφική παράσταση της συνάρτησης κόστους σε σχέση με το πλήθος επαναλήψεων να παραμένει μια φθίνουσα καμπύλη.

5.3 Υπερπροσαρμογή και Κανονικοποίηση

5.3.1 Υπερπροσαρμογή

Στην προηγούμενη ενότητα, εξετάσαμε τη γραμμική παλινδρόμηση. Όπως είδαμε, το μοντέλο που παράγεται προσπαθεί να ταιριάξει όσο περισσότερο γίνεται με τα δεδομένα. Υπάρχουν τρεις πιθανές περιπτώσεις για το μοντέλο μας:

1. το μοντέλο να μην προσεγγίζει καλά τα δεδομένα και να έχουμε υποπροσαρμογή,
2. το μοντέλο να προσεγγίζει καλά τα δεδομένα και να γενικεύει σωστά, δηλαδή να κατηγοριοποιεί σωστά νέα δείγματα, και
3. το μοντέλο να προσεγγίζει τέλεια τα δεδομένα, αλλά να μην μπορεί να γενικεύσει.

Η τρίτη περίπτωση είναι γνωστή ως υπερπροσαρμογή. Το μοντέλο έχει υπερκεπευδευτεί στο να βγάζει τέλεια αποτελέσματα για το σύνολο εκπαίδευσης, αλλά αδυνατεί να γενικεύσει και να πετύχει εξίσου καλά αποτελέσματα για τα νέα δεδομένα. Αν έχουμε πολλά χαρακτηριστικά, αλλά το πλήθος εγγραφών του συνόλου δεδομένων είναι μικρό, τότε το πιο πιθανό είναι να έχουμε πρόβλημα υπερπροσαρμογής.

Για την αντιμετώπιση αυτού του προβλήματος υπάρχουν δυο λύσεις. Η πρώτη λύση είναι να μειώσουμε τον αριθμό των χαρακτηριστικών, είτε επιλέγοντας χειρωνακτικά τα χαρακτηριστικά που θα χρησιμοποιήσουμε είτε χρησιμοποιώντας κάποιον αλγόριθμο επιλογής. Η δεύτερη λύση είναι να κάνουμε κανονικοποίηση (regularization) του μοντέλου. Δηλαδή, κρατάμε όλα τα χαρακτηριστικά, αλλά μειώνουμε την αντίστοιχη παράμετρο β_j, δηλαδή τη βαρύτητα που έχει το αντίστοιχο χαρακτηριστικό κατά την εκπαίδευση και δημιουργία του μοντέλου. Η κανονικοποίηση δίνει καλά αποτελέσματα, όταν καθένα από τα πολλά χαρακτηριστικά συνεισφέρει από λίγο.

5.3.2 Κανονικοποίηση Μοντέλου

Η βασική ιδέα της κανονικοποίησης του μοντέλου είναι ότι μικρές τιμές στις παραμέτρους β_1, β_2, …, β_n οδηγούν σε απλούστερες υποθέσεις και, κατά συνέπεια, μειώνονται οι πιθανότητες να εμφανιστεί το φαινόμενο της υπερπροσαρμογής. Στην περίπτωση της γραμμικής παλινδρόμησης, αρκεί να προσθέσουμε έναν επιπλέον όρο στη συνάρτηση κόστους, όπως φαίνεται στον παρακάτω τύπο:

$$F(\beta_0, \beta_1, ..., \beta_n) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_\beta(x^{(i)}) - y^{(i)}\right)^2 + \lambda \cdot \sum_{j=1}^{n}\beta_j^2\right]$$

Ουσιαστικά, ο επιπλέον όρος επιβάλει τη μείωση των παραμέτρων β_j, ώστε να μειωθεί γενικά η τιμή της συνάρτησης. Η παράμετρος κανονικοποίησης λ ρυθμίζει το πόσο καλά θα προσεγγίσει το μοντέλο τα δεδομένα και το τι τάξη μεγέθους θα είναι οι τιμές των παραμέτρων β_j, ώστε να αποφευχθεί η υπερπροσαρμογή. Ωστόσο, αν το λ πάρει πολύ μεγάλες τιμές (π.χ. $\lambda = 10^{10}$), τότε οι παράμετροι β_j θα γίνουν τόσο μικρές που θα πλησιάζουν το 0, και, συνεπώς, θα έχουμε υποπροσαρμογή.

Στην υπενότητα 5.3.3 παρουσιάζουμε την υλοποίηση των βασικών συναρτήσεων του μοντέλου γραμμικής παλινδρόμησης με ενσωματωμένη κανονικοποίηση.

5.3.3 Υλοποίηση Γραμμικής Παλινδρόμησης με Κανονικοποίηση

Στον παρακάτω κώδικα υλοποιούνται 2 συναρτήσεις. Η συνάρτηση `grad_desc` υλοποιεί τον αλγόριθμο σταδιακής καθόδου.

```
grad_desc <- function(X, y, theta, alpha, lambda, num_iters){
    m <- length(y)
    F_history <- c(rep(0, num_iters))

    for(iter in c(1:num_iters)){
       temp <- vector()
       temp <- theta*(1 - ((alpha*lambda)/m)) -
                    alpha*(1/m)*(t(X) %*% (X %*% theta - y))
       theta <- temp
       F_history[iter] <- computeCost(X, y, theta)
    }
    print(F_history[num_iters])
    return(list("theta"=theta, "F_history"=F_history))
}
```

Τα ορίσματα της συνάρτησης είναι:

- X: το σύνολο δεδομένων, δηλαδή όλα τα δείγματα, χωρίς τη μεταβλητή στόχου.
- y: οι τιμές της μεταβλητή στόχου για όλα τα δείγματα, αντίστοιχα.
- theta: οι παράμετροι β_j.
- alpha: η παράμετρος μάθησης α.
- lambda: η παράμετρος κανονικοποίησης.
- num_iters: το πλήθος επαναλήψεων, το οποίο θα τρέξει ο αλγόριθμος σταδιακής καθόδου.

Στο διάνυσμα `F_history` ουσιαστικά αποθηκεύουμε την τιμή της συνάρτησης κόστους σε κάθε βήμα. Μέσα στον βρόχο επανάληψης `for` υλοποιείται ο ψευδοκώδικας για τη σταδιακή κάθοδο σε γραμμική παλινδρόμηση που είδαμε στην υποενότητα 5.2.2.4. Η συνάρτηση εκτυπώνει την τελική τιμή της συνάρτησης κόστους και στη συνέχεια επιστρέφει τις παραμέτρους β_j και το διάνυσμα `F_history`.
Η συνάρτηση `computeCost` είναι ουσιαστικά η συνάρτηση κόστους *F* και η υλοποίηση της παρουσιάζεται παρακάτω:

```
computeCost <- function(X, y, th){

     m <- length(y)
     return(1/(2*m) * sum((X%*%th - y)^2))

}
```

Να σημειώσουμε ότι τα δεδομένα (χαρακτηριστικά και μεταβλητή στόχου) θα πρέπει να είναι αριθμητικά, προκειμένου να τρέξει ο παραπάνω κώδικας, χωρίς σφάλματα.

Κριτήρια αξιολόγησης

Κριτήριο αξιολόγησης 1

Στόχος αυτού του κριτηρίου αξιολόγησης είναι η εξοικείωση με τα δέντρα απόφασης, με τη χρήση του πακέτου `rpart` της R. Επειδή το πακέτο αυτό δεν είναι προεγκατεστημένο στην R, θα πρέπει να εγκατασταθεί με τη χρήση της εντολής `install.packages()` της R. Το σύνολο δεδομένων, που θα χρησιμοποιηθεί για την επαγωγή του δέντρου απόφασης, είναι το `spam` της βιβλιοθήκης `kernlab`. Ομοίως, θα πρέπει να το εγκαταστήσετε, καθώς δεν είναι προεγκατεστημένο. Αφού φορτώσετε το αρχείο των δεδομένων στην R, ενημερωθείτε για τα περιεχόμενά του (πλήθος μεταβλητών/γνωρισμάτων, πλήθος γραμμών, τύποι γνωρισμάτων, ύπαρξη άγνωστων τιμών κ.λπ.).

α) Χρησιμοποιήστε τη συνάρτηση `rpart()` με τις default τιμές για τις παραμέτρους που αυτή δέχεται, για να κτίσετε ένα δέντρο κατηγοριοποίησης για την πρόβλεψη του κατηγορικού γνωρίσματος `Type` με τη χρήση όλων των άλλων γνωρισμάτων του συνόλου δεδομένων (Προσοχή! Εξαιρουμένου φυσικά και του ίδιου του `Type`). Για την εκπαίδευση και τον έλεγχο του μοντέλου θα πρέπει να δημιουργήσετε δύο τυχαία δείγματα μεγέθους ίσου με το 70% και το 30% αντίστοιχα του μεγέθους του αρχικού συνόλου δεδομένων. Γράψτε κώδικα, που να κάνει αυτό τον διαχωρισμό. Αν χρησιμοποιήσετε έτοιμες συναρτήσεις, χρησιμοποιήστε seed = 70. Αν επιλέξετε να γράψετε δικό σας κώδικα, επιλέξτε ως σύνολο ελέγχου τις γραμμές, των οποίων το υπόλοιπο της διαίρεσης με το 3 θα ισούται με 0 (π.χ. γραμμή 3, 6, 9, κ.ο.κ). Σε κάθε περίπτωση, φροντίστε, ώστε ο διαχωρισμός να είναι ο ίδιος σε κάθε εκτέλεση του κώδικα και τα δύο σύνολα να μην περιέχουν κοινές εγγραφές.

β) Χρησιμοποιήστε το μοντέλο που δημιουργήσατε για το ερώτημα (α), για να κατηγοριοποιήσετε τις 10 πρώτες γραμμές του αρχικού συνόλου δεδομένων.

γ) Δημιουργήστε μία γραφική αναπαράσταση του μοντέλου (του δέντρου απόφασης) που δημιουργήσατε στο ερώτημα (α), κάνοντας χρήση των συναρτήσεων `plot()`, `text()` και `par()`. Αναφέρετε 2 μειονεκτήματα των δένδρων απόφασης. Τι είναι η υπερπροσαρμογή (overfitting) και πώς μπορούμε να την αποφύγουμε;

Απάντηση

α) Ο κώδικας για την επίλυση του ερωτήματος παρατίθεται παρακάτω:

```
> rm(list=ls())
> library(rpart)
> library(kernlab)
> data(spam)
> vars=c(1:57,58)
> sum(spam$type=="nonspam")
[1] 2788
> sum(spam$type=="spam")
[1] 1813
> No.sub=c(1:nrow(spam))[spam$type=="nonspam"]
> Yes.sub=c(1:nrow(spam))[spam$type=="spam"]
> set.seed(70)
```

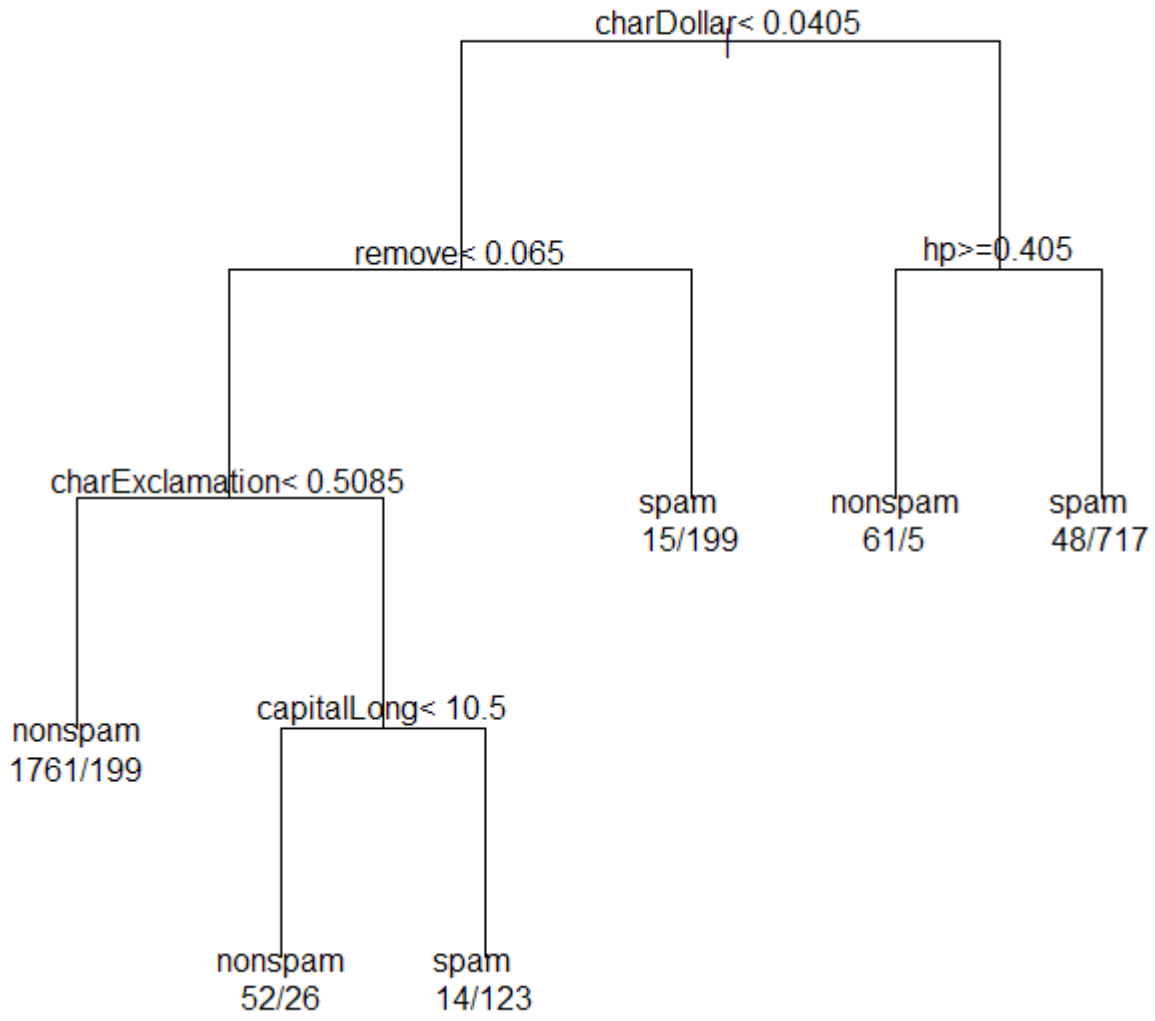
```
> train.No=sample(No.sub,floor(length(No.sub)*70/100))
> set.seed(70)
> train.Yes=sample(Yes.sub,floor(length(Yes.sub)*70/100))
> train=c(train.No,train.Yes)
> train.set=spam[train,vars]
> test.set=spam[-train,vars]
> model=rpart(type ~ ., data=train.set)
> print(model)
```

β) Ο κώδικας για την κατηγοριοποίηση των 10 πρώτων γραμμών παρατίθεται παρακάτω:

```
> manual=c(1,2,3,4,5,6,7,8,9,10)
> manual.set=spam[manual,vars]
> prediction0=predict(model,manual.set,type="class")
> print(prediction0)
      1       2       3       4       5       6       7
   spam    spam    spam    spam    spam nonspam    spam
      8       9      10
nonspam    spam    spam
Levels: nonspam spam
>    print(table(spam[manual,"type"],prediction0,dnn=c("Actu-
al","Predicted")))
         Predicted
Actual    nonspam spam
  nonspam       0    0
  spam          2    8
```

γ) Ο κώδικας για τη δημιουργία της ζητούμενης γραφικής αναπαράστασης παρατίθεται παρακάτω:

```
> prediction=predict(model,test.set,type="class")
> mypar=par(xpd=TRUE)
> plot(model,uniform=T,compress=F,branch=1)
> text(model,use.n=T)
```



Εικόνα ***5.14*** *Οπτικοποίηση μοντέλου δένδρου απόφασης για το κριτήριο αξιολόγησης 1.*

Κριτήριο αξιολόγησης 2

Σε προηγούμενη ενότητα του κεφαλαίου περιγράψαμε πώς μπορεί να υλοποιηθεί η γραμμική παλινδρόμηση στην R. Στόχος αυτού του κριτηρίου αξιολόγησης είναι η εξοικείωση με την ήδη υλοποιημένη έκδοση της γραμμικής παλινδρόμησης, η οποία συμπεριλαμβάνεται στη βιβλιοθήκη πακέτων της R.

α) Χρησιμοποιήστε τις παρακάτω εντολές, για να φορτώσετε τα δεδομένα που θα χρησιμοποιήσουμε. Τα δεδομένα είναι ο δείκτης CPI (Consumer Price Index), για κάθε τρίμηνο από το 2013 μέχρι και το 2015. Στη συνέχεια, δημιουργήστε μια απλή γραφική απεικόνισή τους με χρήση των συναρτήσεων `plot` και `axis`.

```
> year <- rep(2013:2015, each=4)
> quarter <- rep(1:4, 3)
> cpi <- c(163.3, 165.7, 166.5, 168, 168.4, 169, 167.5, 170.1,
173, 172.1, 174.5, 174)
```

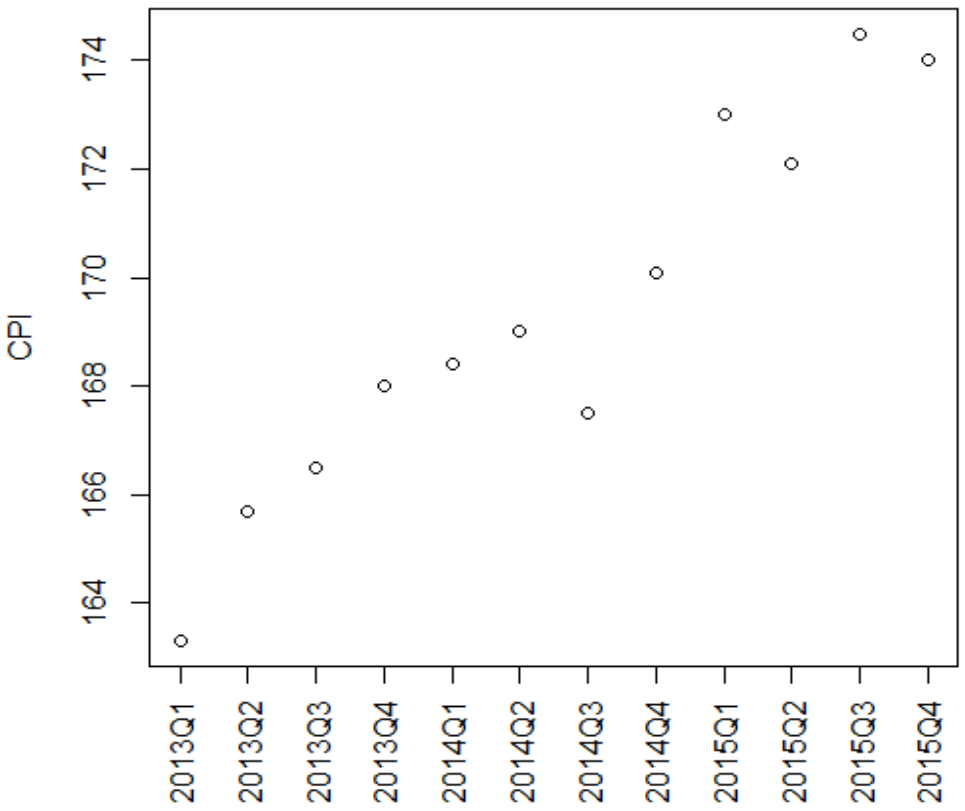
β) Χρησιμοποιώντας τη συνάρτηση `lm` και τη σχετική βοήθεια από την εντολή `?help`, δημιουργήστε ένα μοντέλο παλινδρόμησης με όνομα fit.
γ) Χρησιμοποιώντας το μοντέλο που δημιουργήσατε στο ερώτημα (β) και τη συνάρτηση `predict`, προβλέψτε τις τιμές CPI για τα τέσσερα τρίμηνα του έτους 2016.

```
> data2016 <- data.frame(year = 2016, quarter=1:4)
```

Απάντηση

α) Αρχικά φορτώνουμε τα δεδομένα και στη συνέχεια δημιουργούμε τη γραφική απεικόνιση. Ο κώδικας επίλυσης του ερωτήματος και η αντίστοιχη γραφική παράσταση παρατίθενται παρακάτω:

```
> year <- rep(2013:2015, each=4)
> quarter <- rep(1:4, 3)
> cpi <- c(163.3, 165.7, 166.5, 168, 168.4, 169, 167.5, 170.1,
173, 172.1, 174.5, 174)
>
> plot(cpi, xaxt='n', ylab='CPI', xlab="")
>
> axis(1, labels=paste(year, quarter, sep="Q"), at = 1:12,
las=3)
```



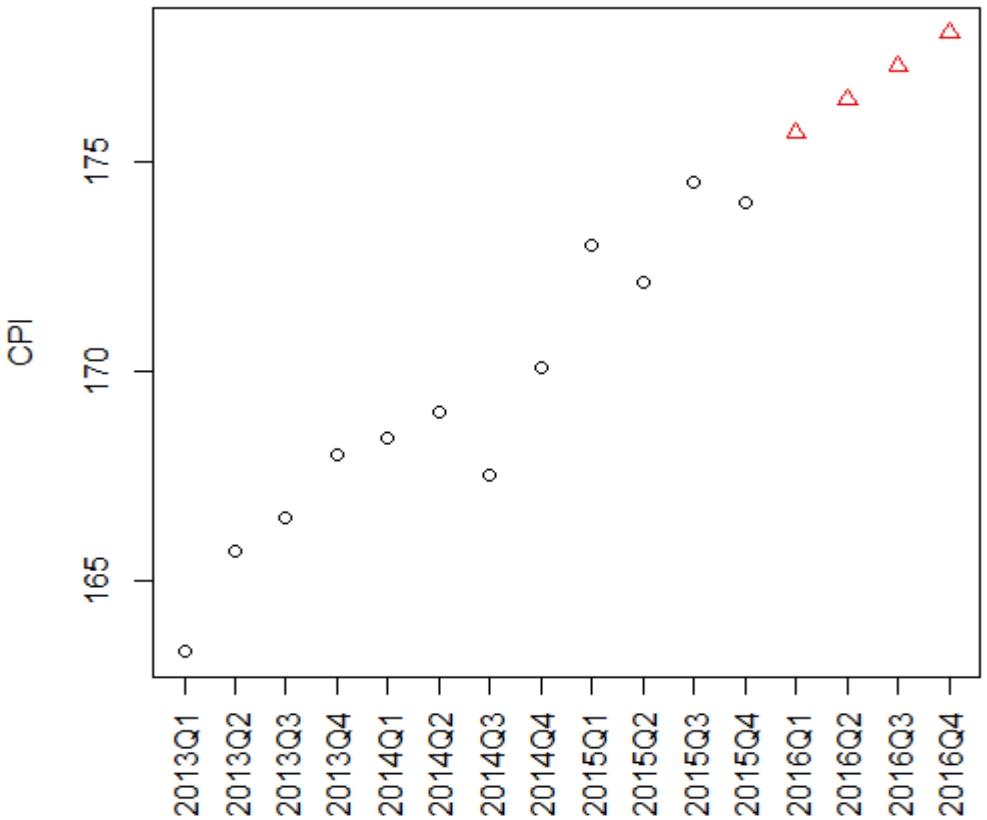
Εικόνα *5.15* *Οπτικοποίηση δεδομένων για το κριτήριο αξιολόγησης 2.*

β) Χρησιμοποιώντας τη συνάρτηση `lm`, δημιουργούμε το μοντέλο παλινδρόμησης:

```
> fit <- lm(cpi ~ year + quarter)
```

γ) Τέλος, αφού φορτώσουμε τα δεδομένα, χρησιμοποιούμε το μοντέλο που δημιουργήσαμε στο ερώτημα (β) για την πρόβλεψη των νέων τιμών. Η πρόβλεψη γίνεται, χρησιμοποιώντας τη συνάρτηση `predict`. Ο κώδικας επίλυσης του ερωτήματος και η αντίστοιχη γραφική παράσταση παρατίθενται παρακάτω:

```
> data2016 <- data.frame(year = 2016, quarter=1:4)
> cpi2016 <- predict(fit, newdata=data2016)
> style <- c(rep(1, 12), rep(2, 4))
> plot(c(cpi, cpi2016), xaxt="n", ylab="CPI", xlab="", pch=-
style, col=style)
>  axis(1,  at=1:16,  las=3,  labels=c(paste(year,  quarter,
sep="Q"), "2016Q1", "2016Q2", "2016Q3", "2016Q4"))
```



Εικόνα ***5.16*** *Οπτικοποίηση δεδομένων και πρόβλεψης για το κριτήριο αξιολόγησης 2.*

Βιβλιογραφία

Peng, R. D. (2015). *R Programming for Data Science*. Lean Publishing. Ανακτήθηκε στις 19 Νοεμβρίου 2015, από: https://leanpub.com/rprogramming

Brian, C. (2015). *Regression Models for Data Science in R*. Lean Publishing. Ανακτήθηκε στις 22 Νοεμβρίου 2015, από: https://leanpub.com/regmods

Nilsson, N. J. (1998). *Introduction to Machine Learning*. Ανακτήθηκε στις 22 Νοεμβρίου 2015, από: http://robotics.stanford.edu/~nilsson/MLBOOK.pdf

Wikipedia (2015). *Classification*. Ανακτήθηκε στις 22 Νοεμβρίου 2015, από: https://en.wikipedia.org/wiki/Classification

Yanchang, Z. (2015). *Regression and Classification with R*. Ανακτήθηκε στις 25 Νοεμβρίου 2015, από: http://www.rdatamining.com/docs/regression-and-classification-with-r

Κεφάλαιο 6: Συσταδοποίηση

Σύνοψη

Ο βασικός στόχος αυτού του κεφαλαίου είναι η εξοικείωση με θέματα που αφορούν την τρίτη σημαντική εργασία της εξόρυξης δεδομένων, δηλαδή την ανάλυση των συστάδων. Πιο συγκεκριμένα, παρουσιάζεται μία σειρά από βασικούς ορισμούς αναφορικά με την ανάλυση συστάδων και τη συσταδοποίηση, και εξετάζονται με λεπτομέρεια τρεις κατηγορίες τεχνικών συσταδοποίησης: η διαμεριστική συσταδοποίηση, η ιεραρχική συσταδοποίηση, και συσταδοποίηση που βασίζεται στην πυκνότητα. Στη συνέχεια γίνεται αναφορά σε συγκεκριμένους αλγορίθμους συσταδοποίησης, όπως ο αλγόριθμος k-means, ο συσσωρευτικός ιεραρχικός αλγόριθμος και ο αλγόριθμος DBSCAN. Παρουσιάζονται, επίσης, διαφορετικές τεχνικές εφαρμογής της ιεραρχικής συσταδοποίησης, όπως είναι η τεχνική του απλού συνδέσμου (ή της ελάχιστης απόστασης), η τεχνική του πλήρους συνδέσμου (ή της μέγιστης απόστασης), η τεχνική του μέσου όρου ομάδας και η μέθοδος Ward.

Προαπαιτούμενη γνώση

Πριν το τρέχον κεφάλαιο θα πρέπει να μελετηθεί τόσο το Κεφάλαιο 1 – Εισαγωγή στην Εξόρυξη Δεδομένων όσο και το Κεφάλαιο 2 – Εισαγωγή στην R.

Συσταδοποίηση

6.1 Μη Επιβλεπόμενη Μάθηση (Unsupervised Learning)

Στην επιβλεπόμενη μάθηση (Supervised Learning) μας δίνεται ένα σύνολο δεδομένων με τις αντίστοιχες κλάσεις-ετικέτες κάθε εγγραφής. Στόχος είναι η δημιουργία ενός μοντέλου, το οποίο να μπορεί να κατηγοριοποιήσει νέα δεδομένα σε κάποια από τις προϋπάρχουσες κλάσεις. Αντίθετα, στη μη επιβλεπομένη μάθηση (Unsupervised Learning) μας δίνεται ένα σύνολο δεδομένων, χωρίς τις αντίστοιχες κλάσεις-ετικέτες κάθε εγγραφής και στόχος είναι η χρήση κάποιου αλγορίθμου, ώστε αυτόματα να ανακαλύψουμε κάποια ενδεχομένως ενδιαφέρουσα δομή των δεδομένων. Για παράδειγμα, η συσταδοποίηση είναι μια από τις τεχνικές μη επιβλεπόμενης μάθησης. Δοθέντων κάποιων δεδομένων χωρίς κλάσεις, οι αλγόριθμοι συσταδοποίησης ομαδοποιούν τα δεδομένα σε συστάδες, έτσι ώστε εγγραφές, οι οποίες ανήκουν στην ίδια συστάδα, να έχουν όμοια ή παραπλήσια χαρακτηριστικά.

6.2 Έννοια της Συστάδας

Στο πρόβλημα της συσταδοποίησης μας δίνεται ένα σύνολο δεδομένων, χωρίς τις αντίστοιχες κλάσεις ή ετικέτες και χρειαζόμαστε κάποιον αλγόριθμο, ο οποίος θα ομαδοποιήσει αυτόματα τα δεδομένα σε συστάδες. Οι συστάδες που δημιουργούνται θέλουμε να διαχωρίζουν ορθά τα δεδομένα. Αυτό πρακτικά σημαίνει ότι μια συστάδα θέλουμε να απαρτίζεται από αντικείμενα, όπου κάθε αντικείμενο είναι πιο κοντά σε κάθε άλλο αντικείμενο της ίδιας συστάδας απ' ότι σε κάποιο άλλο αντικείμενο διαφορετικής συστάδας.

6.3 Αλγόριθμος k-means

6.3.1 Περιγραφή Αλγορίθμου

Ο αλγόριθμος k-means ξεκινάει με k τυχαία σημεία, τα οποία ονομάζονται κεντροειδή της συστάδας και δηλώνουν το κέντρο βάρους της συστάδας. Το k υποδηλώνει σε πόσες συστάδες θέλουμε ο αλγόριθμος να δημιουργήσει. Ο αλγόριθμος εκτελεί επαναληπτικά δύο βήματα. Το πρώτο βήμα αφορά την ανάθεση σε κάποια συστάδα, ενώ το δεύτερο βήμα αφορά τον επαναπροσδιορισμό και τη μετατόπιση του κεντροειδούς κάθε συστάδας.

Πιο αναλυτικά, όσον αφορά στο πρώτο βήμα, δηλαδή την ανάθεση σε κάποια συστάδα, ο αλγόριθμος εξε-

τάζει κάθε δείγμα σε σχέση με τα κεντροειδή των συστάδων. Με χρήση κάποιου μέτρου απόστασης, αναθέτει το εξεταζόμενο δείγμα στη συστάδα, της οποίας το κεντροειδές είναι το πλησιέστερο ως προς το συγκεκριμένο δείγμα. Στο δεύτερο βήμα, παίρνοντας τον μέσο όρο των δειγμάτων κάθε συστάδας, επανυπολογίζονται τα κεντροειδή της κάθε συστάδας, ώστε το κεντροειδές να είναι πιο αντιπροσωπευτικό στην πρόσφατα διαμορφωμένη συστάδα.

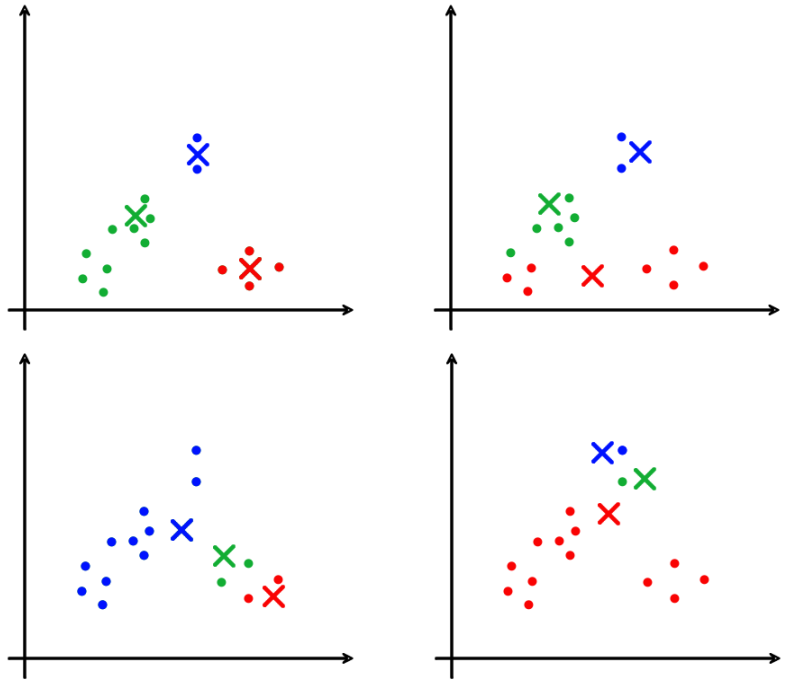
Ο αλγόριθμος εκτελεί επαναληπτικά αυτά τα δύο βήματα, μέχρις ότου τα κεντροειδή των συστάδων να μετατοπίζονται ελάχιστα και σε απόσταση μικρότερη από κάποια δοθείσα τιμή κατωφλίου. Ως εναλλακτικό κριτήριο τερματισμού του αλγορίθμου μπορεί να χρησιμοποιηθεί και ο αριθμός επαναλήψεων του αλγορίθμου.

```
Αρχικοποίησε τυχαία τα k κεντροειδή των συστάδων μ1,μ2,…,μk.
Επανέλαβε{
      Εξέτασε κάθε δείγμα και ανέθεσε το στη συστάδα με το
      πλησιέστερο κεντροειδές ( min |x(i) – μk|2 )
      Επανυπολόγισε τα κεντροειδή υπολογίζοντας το μέσο όρο των
      δειγμάτων της συστάδας
}
```

6.3.2 Τυχαία Αρχικοποίηση Κεντροειδών

Το πρώτο βήμα του αλγορίθμου k-means είναι η τυχαία αρχικοποίηση των k κεντροειδών των συστάδων. Παρόλο που το συγκεκριμένο βήμα φαίνεται απλό και ασήμαντο, αρκετές φορές μια «κακή» αρχικοποίηση μπορεί να οδηγήσει σε κακής ποιότητας συστάδες στην πορεία. Στην Εικόνα 6.1 βλέπουμε ένα παράδειγμα τεσσάρων τυχαίων αρχικοποιήσεων των κεντροειδών, ενώ με χρώμα υποδεικνύεται το πώς τελικά καταλήγουν να είναι οι συστάδες που δημιουργεί ο αλγόριθμος.

Πάνω αριστερά έχουμε την καλύτερη περίπτωση. Ακολουθεί μια λιγότερο ποιοτικά καλή συσταδοποίηση πάνω δεξιά. Στις δυο τελευταίες περιπτώσεις είναι προφανές ότι η αρχικοποίηση επηρεάζει αρνητικά τη διαδικασία συσταδοποίησης. Σε αυτές τις δυο περιπτώσεις, οι δύο συστάδες περιέχουν πολύ λίγα δείγματα, ενώ η μία περιέχει όλα τα υπόλοιπα δείγματα.

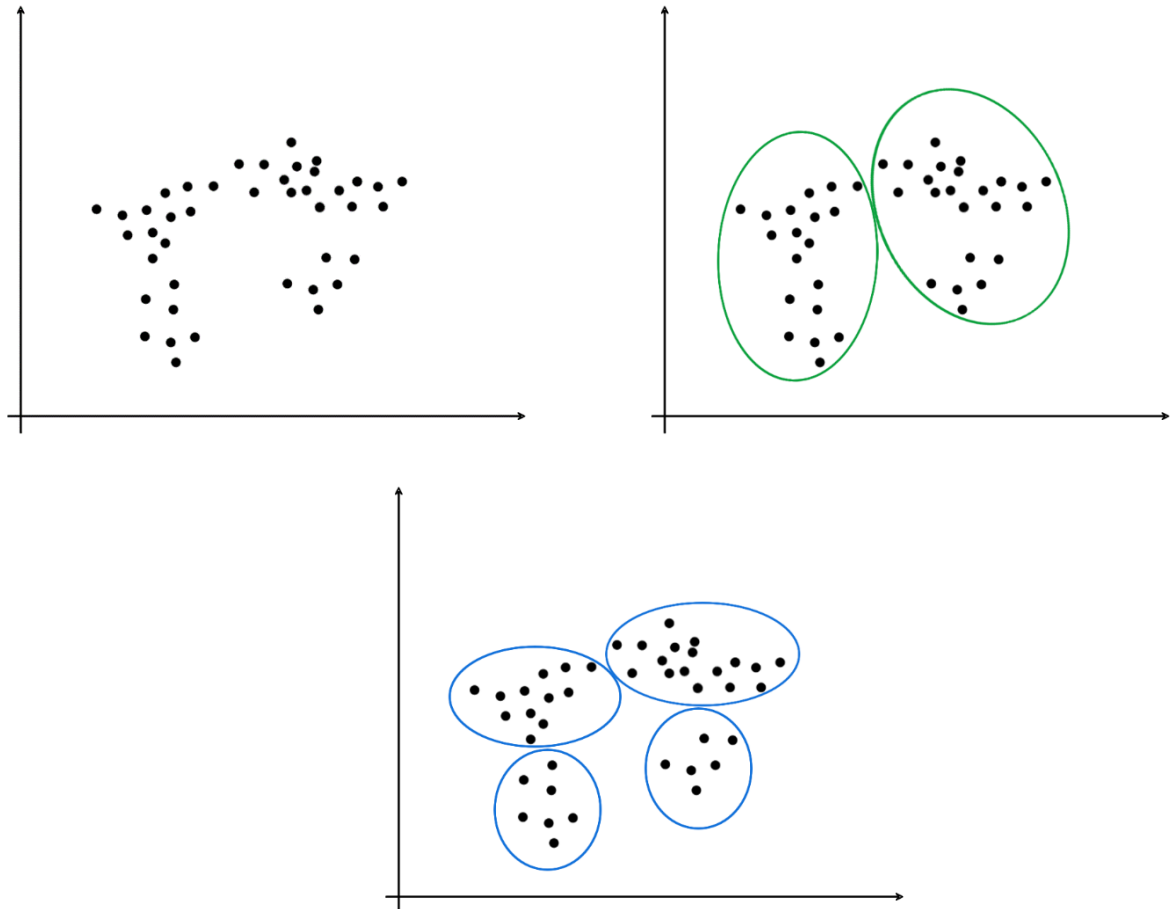


Εικόνα *6.1* *Τυχαία αρχικοποίηση κέντροειδών.*

6.3.3 Επιλογή του Αριθμού Συστάδων

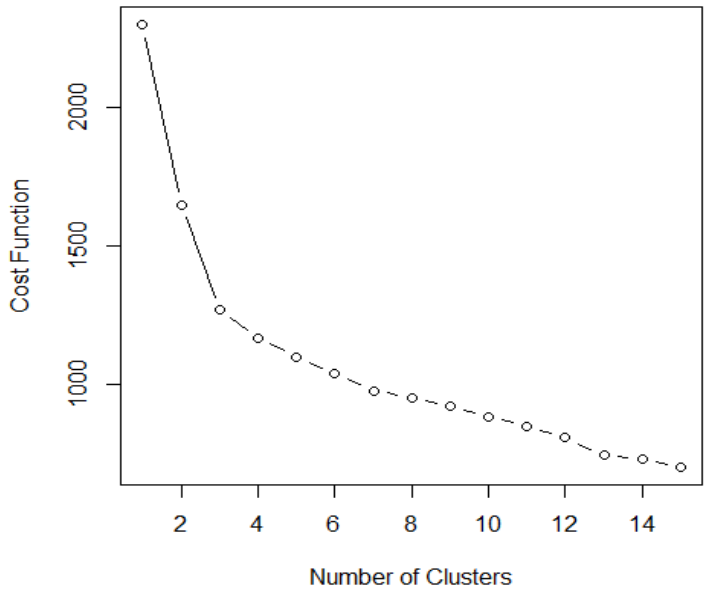
Ένα από τα μειονεκτήματα του αλγορίθμου k-means είναι το γεγονός ότι δεν υπάρχει κάποιος αυτοματοποιημένος τρόπος επιλογής του k, δηλαδή του αριθμού των συστάδων. Ο αριθμός των συστάδων δίνεται ως είσοδος από τον χρήστη και η επιλογή του σωστού αριθμού επαφίεται στη δική του γνώση και εμπειρία. Να υπενθυμίσουμε ότι κατά τη συσταδοποίηση δεν δίνεται το επιπλέον χαρακτηριστικό κλάσης των δειγμάτων. Συνεπώς, η διαδικασία επιλογής του αριθμού συστάδων, ενδεχομένως, να απαιτήσει την εξερεύνηση και μελέτη των δεδομένων, για παράδειγμα, μέσα από οπτικοποιήσεις, προκειμένου να καταλήξουμε στον σωστό αριθμό συστάδων.

Αρκετές φορές τα ίδια τα δεδομένα είναι διφορούμενα. Για παράδειγμα, η οπτικοποίηση των δεδομένων, όπως φαίνονται στην Εικόνα 6.2, δείχνει ότι τα δεδομένα δεν είναι εύκολα διαχωρίσιμα (πάνω αριστερά). Σε πόσες συστάδες θα πρέπει να διαχωριστούν; Σε δύο (πάνω δεξιά) ή σε τέσσερις (κάτω) συστάδες.



Εικόνα *6.2* *Διφορούμενα δεδομένα ως προς τον προσδιορισμού του αριθμού διακριτών συστάδων.*

Δυστυχώς, για την επιλογή του αριθμού των συστάδων δεν υπάρχει κάποιος γενικός κανόνας, ο οποίος να λειτουργεί εγγυημένα και για όλες τις περιπτώσεις. Ένα απλό και πρακτικό τέχνασμα, το οποίο μπορεί να βοηθήσει σε ορισμένες περιπτώσεις, είναι «ο κανόνας του αγκώνα» (the elbow rule). Στην Εικόνα 6.3 ο κανόνας του αγκώνα υποδυκνύει ότι η επιλογή k=3 είναι αρκετά καλή. Ωστόσο, υπάρχουν περιπτώσεις, όπου η γραφική είναι πιο ομαλή και δεν έχει τον τύπο σχήματος του αγκώνα, με αποτέλεσμα η επιλογή και πάλι να μην είναι ξεκάθαρη.



Εικόνα *6.3* *Ο κανόνας του αγκώνα.*

6.3.4 Υλοποίηση k-Means στην R

Στο ακόλουθο παράδειγμα θα παρουσιάσουμε την υλοποίηση του αλγορίθμου k-means στην R και τις σχετικές συναρτήσεις (Κώδικας 6.1). Το σύνολο δεδομένων *iris* περιέχει 50 μετρήσεις για καθένα από τα τρία διαφορετικά είδη λουλουδιών: setosa, versicolor και virginica (συνολικά 150 δείγματα). Οι μετρήσεις αφορούν το μήκος και το πλάτος (σε cm) των πετάλων και των σέπαλων των λουλουδιών κάθε είδους. Στόχος του παραδείγματος είναι να ελέγξουμε την ποιότητα της συσταδοποίησης, αφού αφαιρέσουμε το χαρακτηριστικό Species, το οποίο δηλώνει το είδος στο οποίο ανήκει το λουλούδι.

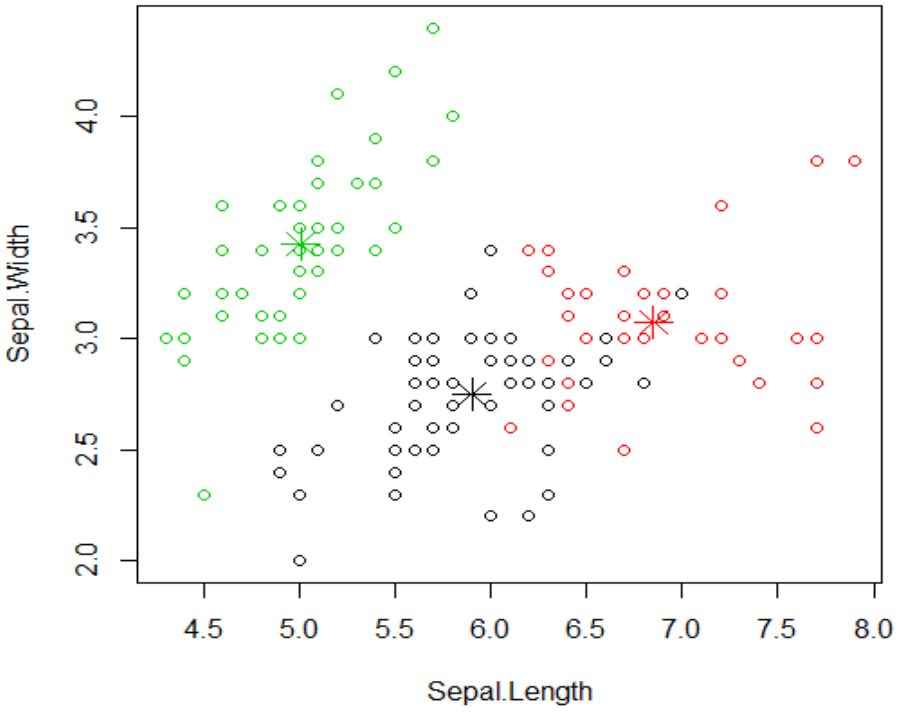
Αρχικά φορτώνουμε τα δεδομένα, τα οποία υπάρχουν στα έτοιμα και ήδη φορτωμένα πακέτα. Αποθηκεύουμε τα δεδομένα με όνομα μεταβλητής iris_new και διαγράφουμε το χαρακτηριστικό Species. Αυτή η πληροφορία πρέπει να μείνει κρυφή στον αλγόριθμο, ώστε να εξετάσουμε στην πορεία, πόσο καλό διαχωρισμό έκανε η συσταδοποίηση που υλοποιήσαμε.

Στη συνέχεια εφαρμόζουμε τον αλγόριθμο k-means με όρισμα *k=3*. Τελικά, για την πληρότητα του παραδείγματος ελέγχουμε πόσα από τα δείγματα έχουν μπει στη σωστή συστάδα, σε σχέση και με όλα τα υπόλοιπα δείγματα. Η απεικόνιση της συσταδοποίησης παρουσιάζεται στην Εικόνα 6.4. Το ιδανικό θα ήταν αν όλα τα δείγματα από κάθε είδος (Species) τοποθετούνταν στην ίδια συστάδα.

```
> iris_new <- iris
> iris_new$Species <- NULL
> kc <- kmeans(iris_new, 3)
> table(iris$Species, kc$cluster)

             1  2  3
  setosa     0  0 50
  versicolor 48  2  0
  virginica  14 36  0
```

```
> plot(iris_new[c("Sepal.Length", "Sepal.Width")], col=kc$-
cluster)
> points(kc$centers[,c("Sepal.Length", "Sepal.Width")],
col=1:3, pch=8, cex=2)
```

Κώδικας *6.1* *Συσταδοποίηση k-means πάνω στο σύνολο δεδομένων iris.*



Εικόνα *6.4* *Απεικόνιση συσταδοποίησης με k-means, k=3.*

6.4 Ιεραρχικοί Αλγόριθμοι Συσταδοποίησης

Οι ιεραρχικοί αλγόριθμοι συσταδοποίησης, όπως δηλώνει και το όνομά τους, δημιουργούν μια ιεραρχία εμφωλιασμένων συσταδοποιήσεων. Δηλαδή, συστάδες περιέχουν μεμονωμένα στοιχεία και άλλες συστάδες, οι οποίες με τη σειρά τους μπορεί να περιέχουν και αυτές άλλες, μικρότερες συστάδες, δημιουργώντας έτσι τα επίπεδα της ιεραρχίας.

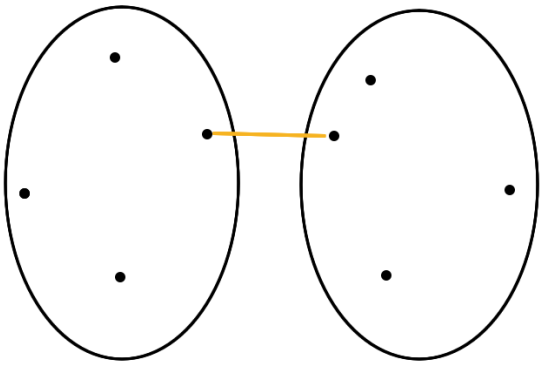
Οι ιεραρχικοί αλγόριθμοι διακρίνονται σε δύο υποκατηγορίες: τους συσσωρευτικούς και τους διαιρετικούς. Οι αλγόριθμοι μπορούν να αναπαρασταθούν πλήρως με δενδρογράμματα, δηλαδή με δενδρικά διαγράμματα, τα οποία παρουσιάζουν τη διάταξη των συστάδων που δημιουργήθηκαν από την ιεραρχική συσταδοποίηση. Ουσιαστικά, κάθε επίπεδο ενός δενδρογράμματος ορίζει ένα βήμα του αλγορίθμου. Το βασικό πλεονέκτημα των ιεραρχικών αλγορίθμων είναι ότι δεν χρειάζεται να υποθέσουμε ένα συγκεκριμένο αριθμό συστάδων, αφού οποιοσδήποτε αριθμός μπορεί να επιτευχθεί, απλά κόβοντας το δενδρόγραμμα στο κατάλληλο επίπεδο.

6.4.1 Ορισμός Απόστασης Συστάδων

Πριν προχωρήσουμε στην ανάλυση των συσσωρευτικών ιεραρχικών αλγορίθμων, κρίνεται σκόπιμο να ορίσουμε μερικούς τρόπους προσδιορισμού της απόστασης μεταξύ δύο συστάδων. Οι κυριότεροι είναι οι εξής:

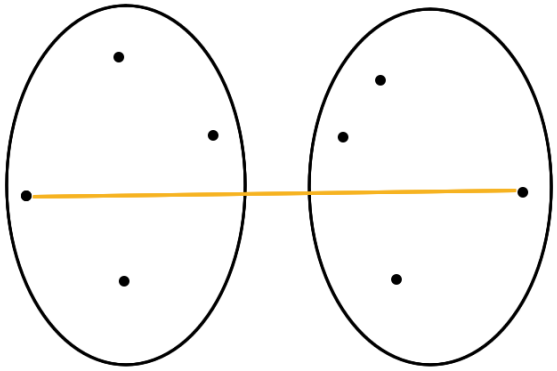
- Ελάχιστης απόστασης ή απλού συνδέσμου (single link).
- Μέγιστης απόστασης ή πλήρους συνδέσμου (complete link).
- Μέσου όρου της συστάδας (group average).
- Απόσταση κεντρικών σημείων.
- Μέθοδος του Ward.

Με βάση το κριτήριο απλού συνδέσμου, η ομοιότητα μεταξύ δύο συστάδων βασίζεται στα δύο πιο όμοια (πιο γειτονικά) σημεία στις διαφορετικές συστάδες (Εικόνα 6.5), δηλαδή στα σημεία με την ελάχιστη απόσταση μεταξύ τους. Είναι γνωστή και ως μέθοδος συσταδοποίησης κοντινότερου γείτονα. Τα προτερήματα αυτής της μεθόδου είναι ότι δημιουργούνται συνεχόμενες συστάδες, ενώ μπορεί να χειριστεί μη ελλειπτικά σχήματα. Το βασικό μειονέκτημα είναι η ευαισθησία στον θόρυβο και στις ακραίες τιμές (outliers).



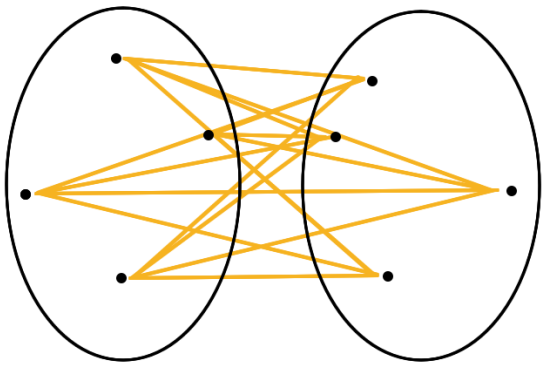
Εικόνα 6.5 *Ομοιότητα συστάδων βάσει ελάχιστης απόστασης ή κριτηρίου απλού συνδέσμου.*

Με βάση το κριτήριο πλήρους συνδέσμου, η ομοιότητα μεταξύ δύο συστάδων βασίζεται στα δύο πιο ανόμοια (πιο απόμακρα) σημεία στις διαφορετικές συστάδες (Εικόνα 6.6), δηλαδή στα σημεία με τη μέγιστη απόσταση μεταξύ τους. Το βασικό πλεονέκτημα αυτού του τρόπου σύνδεσης είναι η μικρή ευαισθησία στον θόρυβο και στις ακραίες τιμές (outliers). Τα μειονεκτήματα που έχει είναι ότι τείνει να διασπά μεγάλες συστάδες και ότι οδηγεί, συνήθως, σε κυκλικά σχήματα.



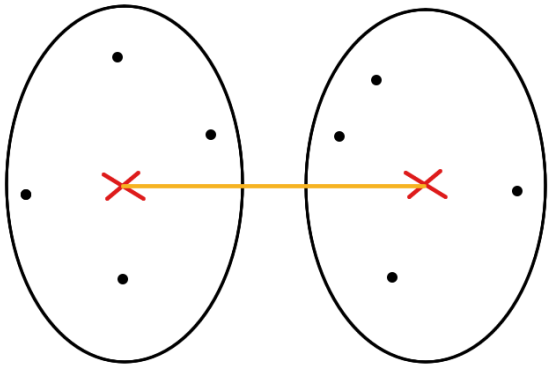
Εικόνα 6.6 *Ομοιότητα συστάδων βάσει μέγιστης απόστασης ή κριτηρίου πλήρους συνδέσμου.*

Ο μέσος όρος συστάδων είναι ουσιαστικά η μέση τιμή των αποστάσεων μεταξύ κάθε πιθανού ζεύγους μεταξύ των σημείων των δύο συστάδων (Εικόνα 6.7). Βρίσκεται κάπου ανάμεσα στην ελάχιστη και τη μέγιστη απόσταση. Έχει μικρότερη ευαισθησία σε θόρυβο και σε ακραίες τιμές (outliers), αλλά ευνοεί τις συστάδες με κυκλικό σχήμα.



Εικόνα 6.7 *Ομοιότητα συστάδων βάσει μέσου όρου συστάδας.*

Η απόσταση κεντρικών σημείων είναι η απόσταση μεταξύ των κέντρων των συστάδων. Το πρόβλημα με αυτή την απόσταση είναι ότι δεν έχει μονότονη αύξηση. Έτσι, δύο συστάδες που συγχωνεύονται μπορεί να έχουν μικρότερη απόσταση από συστάδες, οι οποίες έχουν συγχωνευτεί σε προηγούμενα βήματα.



Εικόνα *6.8* *Ομοιότητα συστάδων βάσει απόστασης μεταξύ κέντρων.*

Τέλος, η βασική ιδέα πίσω από τη μέθοδο του *Ward* είναι ότι η απόσταση μεταξύ δύο συστάδων, C_i και C_j, είναι ίση με το πόσο θα αυξηθεί το άθροισμα των τετραγώνων της απόστασης των στοιχείων της κάθε συστάδας από το αντίστοιχο κεντροειδές (της κάθε συστάδας) μετά τη συγχώνευση τους, C_{ij}, δηλαδή:

$$D_W(C_i, C_j) = \sum_{x \in C_i} (x - r_i)^2 + \sum_{x \in C_j} (x - r_j)^2 - \sum_{x \in C_{ij}} (x - r_{ij})^2 ,$$

όπου r_i είναι το κεντροειδές της συστάδας C_i, r_j είναι το κεντροειδές της συστάδας C_j, και r_{ij} είναι το κεντροειδές της συστάδας C_{ij}, που προκύπτει από τη συγχώνευσή τους. Πρόκειται για το ιεραρχικό ανάλογο του k-means.

6.4.2 Συσσωρευτικοί Αλγόριθμοι (Agglomerative Algorithms)

Οι συσσωρευτικοί αλγόριθμοι ξεκινάνε με κάθε ένα από τα *n* δείγματα να ανήκει σε μια ξεχωριστή συστάδα, δηλαδή ξεκινάνε με *n* συστάδες. Σε κάθε βήμα, συγχωνεύονται οι δύο πιο κοντινές συστάδες, δηλαδή το πλήθος των συστάδων μειώνεται κατά ένα. Αυτή η διαδικασία επαναλαμβάνεται, μέχρις ότου ο αλγόριθμος καταλήξει σε μια μοναδική συστάδα, η οποία θα εμπεριέχει όλα τα *n* δείγματα. Η όλη διαδικασία του αλγορίθμου μπορεί να αναπαρασταθεί με δενδρόγραμμα ανομοιότητας. Το δενδρόγραμμα περιέχει *n-1* επίπεδα και το κάθε επίπεδο αντιστοιχεί σε ένα βήμα του αλγορίθμου.

6.4.3 Διαιρετικοί Αλγόριθμοι (Divisive Algorithms)

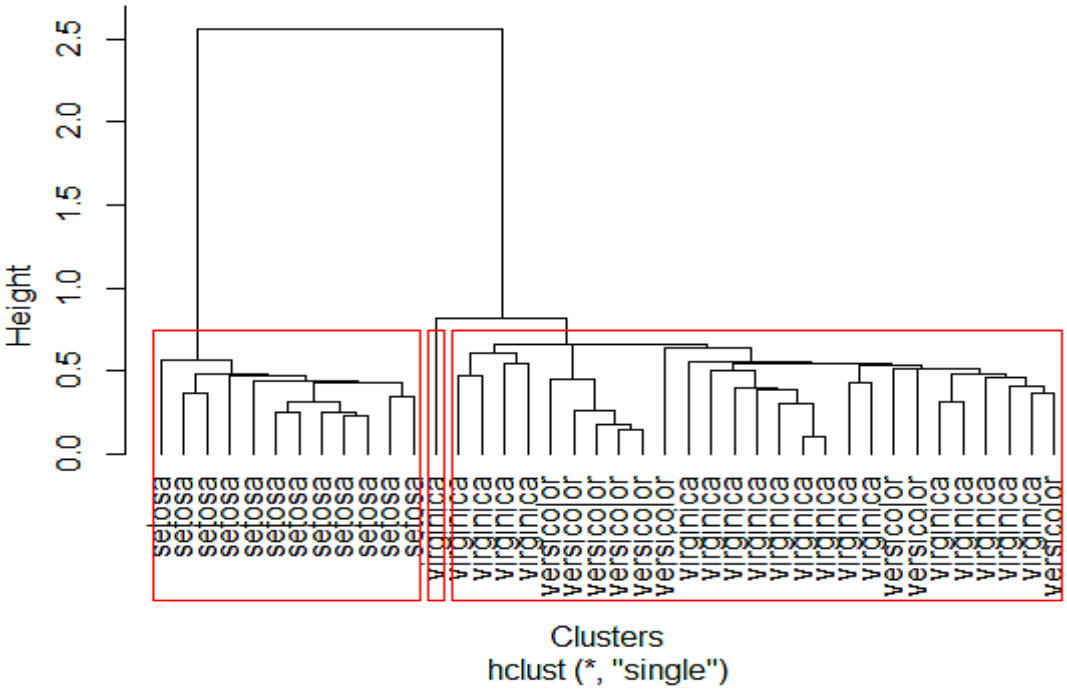
Οι διαιρετικοί αλγόριθμοι ξεκινάνε με όλα τα δείγματα να ανήκουν σε μια ενιαία συστάδα. Σε κάθε βήμα, μια ομάδα διασπάται σε δύο. Αυτό γίνεται επαναληπτικά, μέχρι να καταλήξουμε σε *n* ομάδες. Η πολυπλοκότητα των διαιρετικών αλγορίθμων είναι μεγαλύτερη από αυτή των συσσωρευτικών, αφού η διάσπαση μιας ομάδας σε δύο μπορεί να γίνει κατά 2^{n-1}-1 τρόπους. Η επιλογή της βέλτιστης διάσπασης πρακτικά είναι αδύνατη ακόμα και για μικρό *n*. Στην πράξη η διάσπαση γίνεται, αλλά όχι κατά τον βέλτιστο τρόπο. Η όλη διαδικασία του αλγορίθμου μπορεί να αναπαρασταθεί, όπως και στους συσσωρευτικούς, με δενδρόγραμμα.

6.4.4 Υλοποίηση Ιεραρχικής Συσταδοποίησης στην R

Με το παρακάτω τμήμα κώδικα (Κώδικας 6.2) φορτώνουμε το σύνολο δεδομένων iris, δειγματοληπτούμε 40 από τις εγγραφές του και εφαρμόζουμε ιεραρχική συσταδοποίηση με τη μέθοδο απλού και πλήρους συνδέσμου αντίστοιχα. Να σημειώσουμε ότι αφαιρούμε το χαρακτηριστικό Species, με στόχο να ελέγξουμε κατά πόσο τα στοιχεία των συστάδων βρίσκονται στη σωστή συστάδα. Υπάρχουν τρεις διακριτές τιμές για το συγκεκριμένο χαρακτηριστικό, οπότε στο τέλος κόβουμε το δενδρόγραμμα, έτσι ώστε να σχηματιστούν τρεις συστάδες.

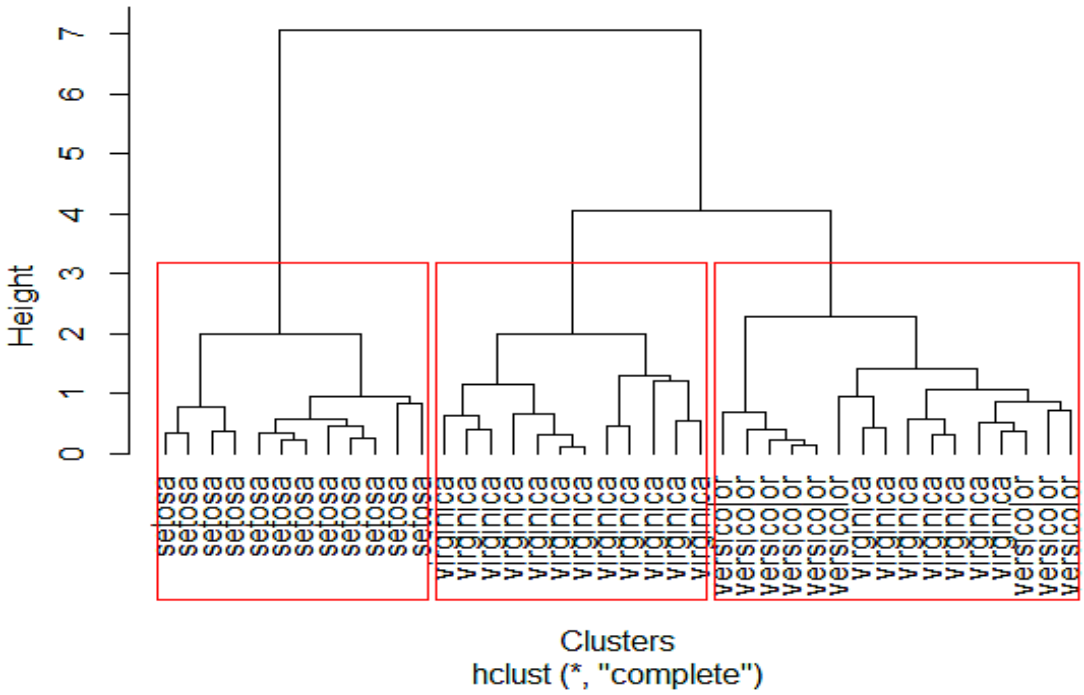
```
> data(iris)
> set.seed(500)
> idx <- sample(1:dim(iris)[1], 40)
> irisSample <- iris[idx,]
> irisSample$Species <- NULL
> hc <- hclust(dist(irisSample), method="single")
> plot(hc, hang = -1, labels=iris$Species[idx], xlab="Clus-
ters")
> rect.hclust(hc, 3,)
> hc <- hclust(dist(irisSample), method="complete")
> plot(hc, hang = -1, labels=iris$Species[idx], xlab="Clus-
ters")
> rect.hclust(hc, 3,)
```

Κώδικας *6.2* *Ιεραρχική συσταδοποίηση πάνω στο σύνολο δεδομένων iris.*



Εικόνα *6.9* *Ιεραρχική συσταδοποίηση με χρήση κριτηρίου απλού συνδέσμου (single link).*

Στην Εικόνα 6.9 παρουσιάζεται το δενδρόγραμμα της ιεραρχικής συσταδοποίησης με χρήση κριτηρίου απλού συνδέσμου. Αντίστοιχα, στην Εικόνα 6.10 παρουσιάζεται το δενδρόγραμμα της ιεραρχικής συσταδοποίησης με χρήση κριτηρίου πλήρους συνδέσμου. Συγκρίνοντας τις δύο εικόνες, μπορούμε να διαπιστώσουμε στην πράξη το μειονέκτημα της περίπτωσής του με τον απλό σύνδεσμο. Ο απλός σύνδεσμος είναι ευαίσθητος στον θόρυβο και τις ακραίες τιμές, ενώ τείνει να δημιουργεί συνεχόμενες συστάδες. Αυτός είναι ο λόγος που η μεσαία και τελευταία συστάδα, από αριστερά προς δεξιά, δεν είναι όπως θα θέλαμε να είναι, δηλαδή κάθε συστάδα να έχει στοιχεία με την ίδια τιμή για το χαρακτηριστικό Species. Αντίθετα, με χρήση πλήρους συνδέσμου έχουμε καλύτερης ποιότητας συστάδες. Ωστόσο, ούτε αυτή η περίπτωση είναι η ιδανική, αφού στην τελευταία συστάδα από αριστερά προς τα δεξιά (Εικόνα 6.10) υπάρχουν στοιχεία με διαφορετικές τιμές (versicolor και virginica). Η ιδανική συσταδοποίηση θα δημιουργούσε συστάδες με τα στοιχεία να έχουν την ίδια τιμή στο χαρακτηριστικό Species, δηλαδή μόνο setosa, virginica ή versicolor.



Εικόνα *6.10* *Ιεραρχική συσταδοποίηση με χρήση κριτηρίου πλήρους συνδέσμου (complete link).*

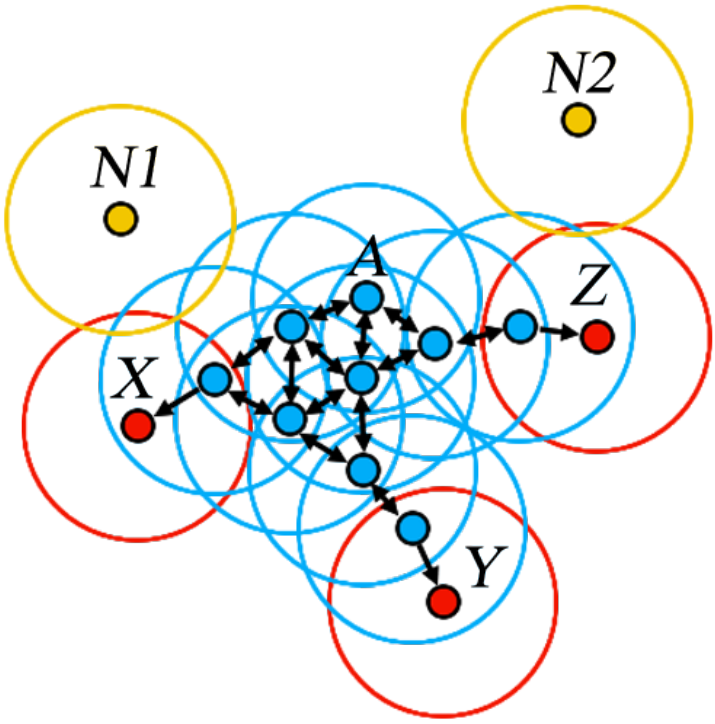
6.5 Αλγόριθμος DBSCAN

6.5.1 Βασικές Έννοιες

Έστω ένα σύνολο από σημεία στον χώρο, τα οποία θέλουμε να συσταδοποιήσουμε. Στο πλαίσιο της συσταδοποίησης με τον αλγόριθμο DBSCAN, τα σημεία αυτά κατηγοριοποιούνται σε κεντρικά (core points), προσεγγίσιμα ή πυκνά-προσεγγίσιμα (density-reachable points) ή ακραία (outliers) με βάση τους παρακάτω κανόνες:

1. Ένα σημείο ***p*** είναι κεντρικό σημείο, αν τουλάχιστον *MinPts* σημεία βρίσκονται σε απόσταση ε από αυτό και αυτά τα σημεία είναι άμεσα προσεγγίσιμα από το ***p*** (Εικόνα 6.11, σημεία με γαλάζιο χρώμα). Κανένα σημείο δεν είναι προσεγγίσιμο από μη κεντρικό σημείο.
2. Ένα σημείο ***q*** είναι πυκνά-προσεγγίσιμο από ένα σημείο ***p***, αν υπάρχει μονοπάτι $p_1, \ldots, p_n$, με $p_1 = \boldsymbol{p}$ και $p_n = \boldsymbol{q}$, όπου κάθε p_{i+1} είναι άμεσα προσεγγίσιμο από το p_i, δηλαδή όλα τα σημεία του μονοπατιού πρέπει να είναι κεντρικά, με πιθανή εξαίρεση το ***q*** (Εικόνα 6.11, σημεία X, Y και Z).
3. Σημεία, τα οποία δεν είναι προσεγγίσιμα από κανένα άλλο σημείο, είναι ακραία (Εικόνα 6.11, σημεία N1 και N2).

Αν το σημείο ***p*** είναι κεντρικό σημείο, τότε σχηματίζει μια συστάδα μαζί με όλα τα σημεία (κεντρικά ή μη), τα οποία είναι προσεγγίσιμα από αυτό. Κάθε συστάδα περιέχει τουλάχιστον ένα κεντρικό σημείο. Η προσεγγισιμότητα δεν είναι συμμετρική σχέση, καθώς εξ ορισμού μπορεί κανένα σημείο να μην είναι προσεγγίσιμο από ένα μη κεντρικό σημείο, ανεξάρτητα από την απόσταση μεταξύ τους. Δηλαδή, ένα μη κεντρικό σημείο μπορεί να είναι προσεγγίσιμο, αλλά κανένα σημείο να μην μπορεί να προσεγγιστεί από αυτό. Συνεπώς, μια επιπλέον έννοια διασύνδεσης απαιτείται για τον ορισμό των συστάδων που δημιουργεί ο DBSCAN. Δυο σημεία ***p*** και ***q*** είναι πυκνά-συνδεδεμένα, αν υπάρχει σημείο ***s,*** τέτοιο ώστε τα ***p*** και ***q*** να είναι (πυκνά-)προσεγγίσιμα από το σημείο ***s***. Η πυκνή-συνδεσιμότητα είναι συμμετρική σχέση. Έτσι, μια συστάδα εμφανίζει τις δυο ακόλουθες ιδιότητες



Εικόνα *6.11* *Παράδειγμα δημιουργίας συστάδων με τον DBSCAN, για MinPts=3.*

1. Όλα τα σημεία μιας συστάδας είναι αμοιβαία πυκνά-συνδεδεμένα μεταξύ τους.
2. Αν ένα σημείο είναι πυκνά-προσεγγίσιμο από κάποιο σημείο της συστάδας, τότε είναι και αυτό, επίσης, σημείο της συστάδας.

6.5.2 Περιγραφή Αλγορίθμου

Ο αλγόριθμος DBSCAN χρησιμοποιεί δυο παραμέτρους, το ε και το *MinPts*, δηλαδή τον ελάχιστο αριθμό σημείων που απαιτούνται για τη δημιουργία μιας πυκνής περιοχής. Ξεκινάει από ένα τυχαίο σημείο, το οποίο δεν έχει επισκεφθεί. Για το επιλεγμένο σημείο ανακτώνται τα σημεία στην ε-γειτονιά, δηλαδή σημεία που απέχουν το πολύ απόσταση ε από το επιλεγμένο σημείο. Αν υπάρχει επαρκής αριθμός σημείων, δηλαδή μεγαλύτερος του MinPts, ο αλγόριθμος δημιουργεί μια συστάδα. Σε αντίθετη περίπτωση, το σημείο μαρκάρεται προσωρινά ως θόρυβος. Μαρκάρονται προσωρινά, διότι σημεία που έχουν μαρκαριστεί ως θόρυβος, μπορεί στην πορεία να βρεθούν σε κάποια ε-γειτονιά άλλου επιλεγμένου σημείου και έτσι τελικά να γίνουν μέρος κάποιας συστάδας.

Αν ένα σημείο αποτελεί πυκνό τμήμα μιας συστάδας, τότε σίγουρα η ε-γειτονιά του είναι υποσύνολο της συστάδας αυτής. Έτσι, όλα τα σημεία μέσα στην ε-γειτονιά προστίθενται στη συστάδα, καθώς επίσης και τα σημεία στην ε-γειτονιά καθενός από αυτά τα σημεία. Αυτή η διαδικασία συνεχίζεται μέχρι να ολοκληρωθεί η εύρεση της πυκνά-συνδεδεμένης συστάδας. Έπειτα ένα νέο σημείο επιλέγεται και ακολουθείται η παραπάνω διαδικασία, για την ανακάλυψη επιπλέον συστάδας ή θορύβου. Σημεία, τα οποία μάρκαρε ο αλγόριθμος ως θόρυβο και που τελικά δεν έγιναν μέρος κάποιας συστάδας αποτελούν ακραία σημεία. Ο Κώδικας 6.3 παρουσιάζει τον ψευδοκώδικα του αλγορίθμου DBSCAN.

```
DBSCAN(D, eps, MinPts) {
   C = 0
   για κάθε σημείο P στη βάση D {
      αν το P είναι μαρκαρισμένο
         συνέχισε με το επόμενο σημείο
      μαρκάρισε το P
```

```
      NeighborPts = ερώτημαΠεριοχής(P, eps)
      αν πλήθος(NeighborPts) < MinPts
         μαρκάρισε το P ως θόρυβο
      αλλιώς {
         C = επόμενη συστάδα
         επέκτασηΣυστάδας(P, NeighborPts, C, eps, MinPts)
      }
   }
}

επέκτασηΣυστάδας(P, NeighborPts, C, eps, MinPts) {
   πρόσθεσε το P στη συστάδα C
   για κάθε σημείο P' στο σύνολο NeighborPts {
      αν το P δεν είναι μαρκαρισμένο {
         μαρκάρισε το P'
         NeighborPts' = ερώτημαΠεριοχής(P', eps)
         αν πλήθος(NeighborPts') >= MinPts
            NeighborPts = NeighborPts U NeighborPts'
      }
      αν το P' δεν ανήκει ήδη σε κάποια συστάδα
         πρόσθεσε το P' στη συστάδα C
   }
}

ερώτημαΠεριοχής(P, eps)
 επέστρεψε όλα τα σημεία στην ε-γειτονιά του P (συμπεριλαμβανομένου
και του P)
```

Κώδικας 6.3 *Ψευδοκώδικας αλγορίθμου DBSCAN.*

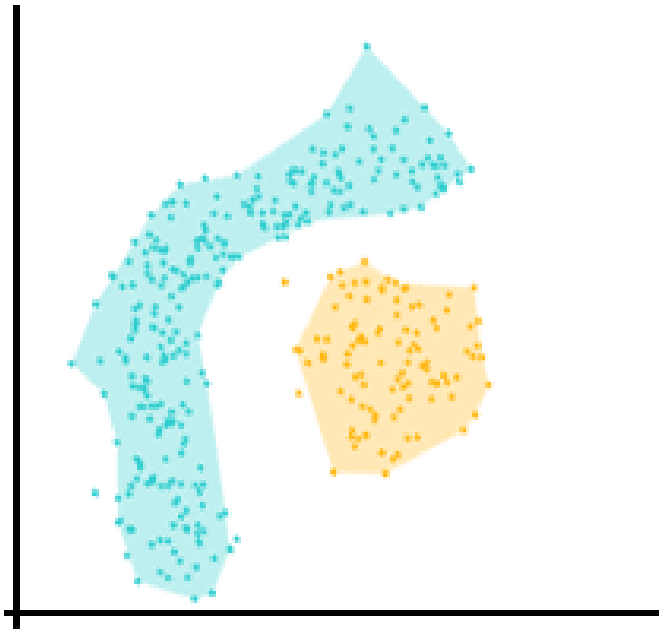
6.5.3 Πολυπλοκότητα Αλγορίθμου

Ο αλγόριθμος DBSCAN επισκέπτεται κάθε σημείο της βάσης, πιθανόν περισσότερες από μία φορές. Ωστόσο, το μεγαλύτερο μέρος της πολυπλοκότητας πηγάζει από τον αριθμό κλήσεων της συνάρτησης ερώτημαΠεριοχής. Η συνάρτηση καλείται ακριβώς μια φορά για κάθε σημείο. Με χρήση ειδικής δομής δεικτοδότησης μπορεί να εκτελεστεί σε O(log n) για n σημεία, και επομένως συνολικά O(n log n). Αν δεν χρησιμοποιηθεί ειδική δομή ή τα δεδομένα έχουν κακή διάταξη (όλα τα σημεία σε απόσταση μικρότερη του ε), τότε έχουμε τη χειρότερη περίπτωση και πολυπλοκότητα $O(n^2)$. Οι απαιτήσεις σε μνήμη είναι O(n), αν χρησιμοποιηθεί υλοποίηση χωρίς μητρώα, ενώ, σε αντίθετη περίπτωση, είναι $O(n^2)$.

6.5.4 Πλεονεκτήματα

Τα βασικότερα πλεονεκτήματα του αλγορίθμου DBSCAN είναι τα ακόλουθα:

1. Σε αντίθεση με αλγορίθμους, όπως ο k-means, ο DBSCAN δεν απαιτεί τον εκ των προτέρων προσδιορισμό του αριθμού των συστάδων.
2. Μπορεί να καταλήξει σε αυθαίρετα σχήματα συστάδων. Μπορεί να εντοπίσει ακόμα και μια συστάδα, η οποία βρίσκεται γύρω από κάποια άλλη (Εικόνα 6.12). Αυτό συμβαίνει λόγω της παραμέτρου MinPts, η οποία ελαττώνει την εμφάνιση του φαινομένου της αλυσίδας συστάδων. Το φαινόμενο της αλυσίδας συστάδων συμβαίνει, όταν διαφορετικές συστάδες συνδέονται με μια λεπτή γραμμή σημείων-αντικειμένων.
3. Έχει καλή ευαισθησία στον θόρυβο και δεν επηρεάζεται από ακραίες τιμές.
4. Χρειάζεται μόνο δυο παραμέτρους και έχει μικρή ευαισθησία ως προς τη σειρά εμφάνισης των δεδομένων στη βάση.
5. Εφόσον έχουν μελετηθεί τα δεδομένα και έχουν γίνει κατανοητά, ο προσδιορισμός των παραμέτρων MinPts και ε δεν είναι δύσκολος.



Εικόνα *6.12* *Συστάδα με αυθαίρετο σχήμα.*

6.5.5 Μειονεκτήματα

Παρόλο που ο αλγόριθμος DBSCAN έχει αρκετά πλεονεκτήματα, διαθέτει και αυτός κάποια μειονεκτήματα, τα βασικότερα από τα οποία είναι τα ακόλουθα:

1. Δεν είναι απόλυτα ντετερμινιστικός, υπό την έννοια ότι τα περιθωριακά σημεία μιας συστάδας μπορούν να ανήκουν είτε σε αυτή είτε σε κάποια γειτονική, ανάλογα με τη σειρά επεξεργασίας. Ευτυχώς, αυτό δεν συμβαίνει συχνά και έχει μικρό αντίκτυπο στα αποτελέσματα.
2. Η ποιότητα των αποτελεσμάτων εξαρτάται από τη μετρική απόστασης που θα χρησιμοποιηθεί. Η πιο κοινή μετρική απόστασης είναι η Ευκλείδεια απόσταση. Ειδικά για πολυδιάστατα δεδομένα, η συγκεκριμένη μετρική είναι σχεδόν άχρηστη, λόγω της λεγόμενης «κατάρας της διαστατικότητας», κάνοντας έτσι δύσκολη την επιλογή της παραμέτρου ε. Ωστόσο, αυτό μπορεί να συμβεί σε οποιονδήποτε αλγόριθμο χρησιμοποιεί την Ευκλείδεια απόσταση.
3. Δεν μπορεί να συσταδοποιήσει καλά σύνολα από δεδομένα με μεγάλες διαφορές πυκνότητας, καθώς δεν μπορεί να εντοπιστεί κάποιος συνδυασμός MinPts-ε, που να είναι κατάλληλος για όλες τις συστάδες.
4. Αν τα δεδομένα δεν έχουν γίνει κατανοητά, η επιλογή ενός κατωφλίου ε που να έχει νόημα μπορεί να είναι δύσκολη.

Κριτήρια αξιολόγησης

Κριτήριο αξιολόγησης 1

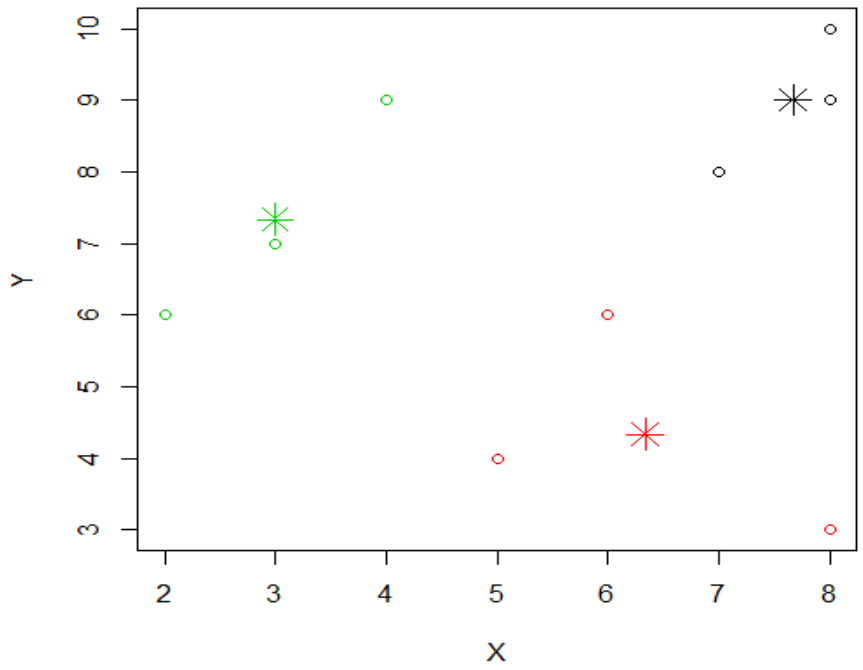
Να γραφεί κώδικας που να χρησιμοποιεί τον αλγόριθμο k-means και την Ευκλείδεια απόσταση για τη συσταδοποίηση των παρακάτω 9 δειγμάτων σε 3 συστάδες: $P_1 = (4, 10)$, $P_2 = (3, 7)$, $P_3 = (8, 3)$, $P_4 = (4, 9)$, $P_5 = (6, 6)$, $P_6 = (6, 4)$, $P_7 = (2, 6)$, $P_8 = (4, 7)$, $P_9 = (7, 8)$. Να γίνει απεικόνιση των αποτελεσμάτων, δηλαδή των σημείων και των κεντροειδών κάθε συστάδας. Να εξεταστεί με τον κανόνα του αγκώνα, αν ο αριθμός συστάδων που προτείνεται από την εκφώνηση είναι καλός ή όχι.

Απάντηση

Παραθέτουμε τον κώδικα επίλυσης του ερωτήματος και την αντίστοιχη απεικόνιση που παράγεται από αυτόν:

```
> # Δεδομένα
> data <- data.frame(X=c(8, 3, 8, 4, 6, 5, 2, 8, 7),Y=c(10, 7,
3, 9, 6, 4, 6, 9, 8))
>
> # Χρήση k-means
> kc <- kmeans(data, 3)
> # Απεικόνιση
> plot(data, col=kc$cluster)
> points(kc$centers[,c("X", "Y")], col=1:3, pch=8, cex=2)
>
```

Κώδικας 6.*4* *Κώδικας επίλυσης κριτηρίου αξιολόγησης 1.*



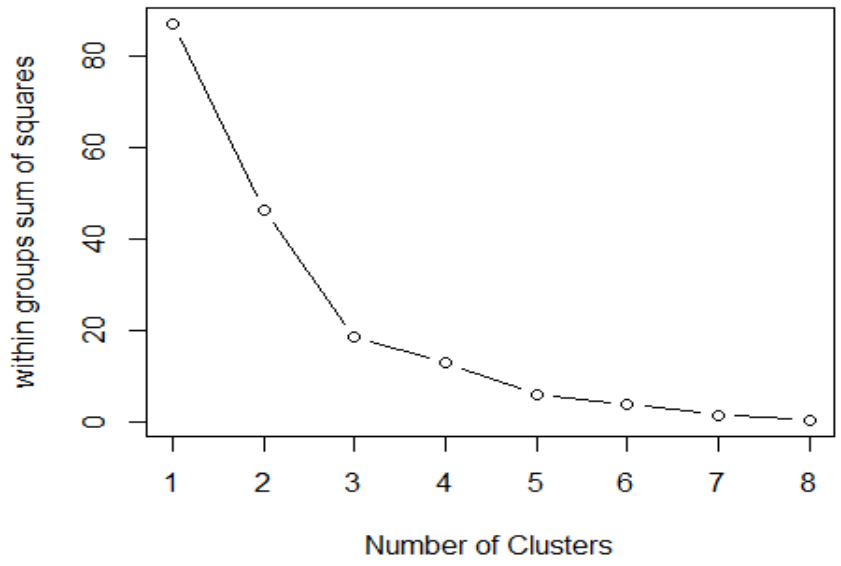
Εικόνα *6.13* *Απεικόνιση σημείων και των κεντροειδών κάθε συστάδας.*

Στη συνέχεια, πρέπει να υλοποιήσουμε μια συνάρτηση για την απεικόνιση του σφάλματος βάσει επιλογής του αριθμού συστάδων. Η συνάρτηση στον Κώδικα 6.4 δέχεται ως όρισμα τα δεδομένα και τον μέγιστο αριθμό συστάδων `nc` που θα μπορούσε να προκύψει από αυτά. Εκτελεί τον αλγόριθμο για k = nc, ..., 2, 1 και απεικονίζει τα αποτελέσματα.

```
> wssplot <- function(data, nc){
+    wss <- (nrow(data)-1) * sum(apply(data,2,var))
+    for(i in 2:nc){
+       wss[i] <- sum(kmeans(data,centers=i)$withinss)
+    }
+    plot(1:nc, wss, type='b',xlab='Number of Clusters',
+         ylab='within groups sum of squares')
+ }
> wssplot(data, nrow(data)-1)
>
```

Κώδικας *6.5* *Κώδικας υλοποιήσης του κάνονα του αγκώνα.*

Η γραφική απεικόνιση που προκύπτει από τον Κώδικα 6.4 φαίνεται παρακάτω (Εικόνα 6.14).



Εικόνα *6.14* *Ο κανόνας του αγκώνα για τα δεδομένα του κριτηρίου αξιολόγησης 1.*

Παρατηρούμε ότι ο αγκώνας σχηματίζεται στον αριθμό συστάδων k = 3. Συνεπώς, ο προτεινόμενος αριθμός από την εκφώνηση είναι πολύ καλός.

Κριτήριο αξιολόγησης 2

Χρησιμοποιώντας το σύνολο δεδομένων `mtcars` της R, καθώς και τις συναρτήσεις `dist()` και `hclust()`, υλοποιήστε ιεραρχική συσταδοποίηση πάνω στο σύνολο δεδομένων. Στη συνέχεια, εκτυπώστε το δενδρόγραμμα.

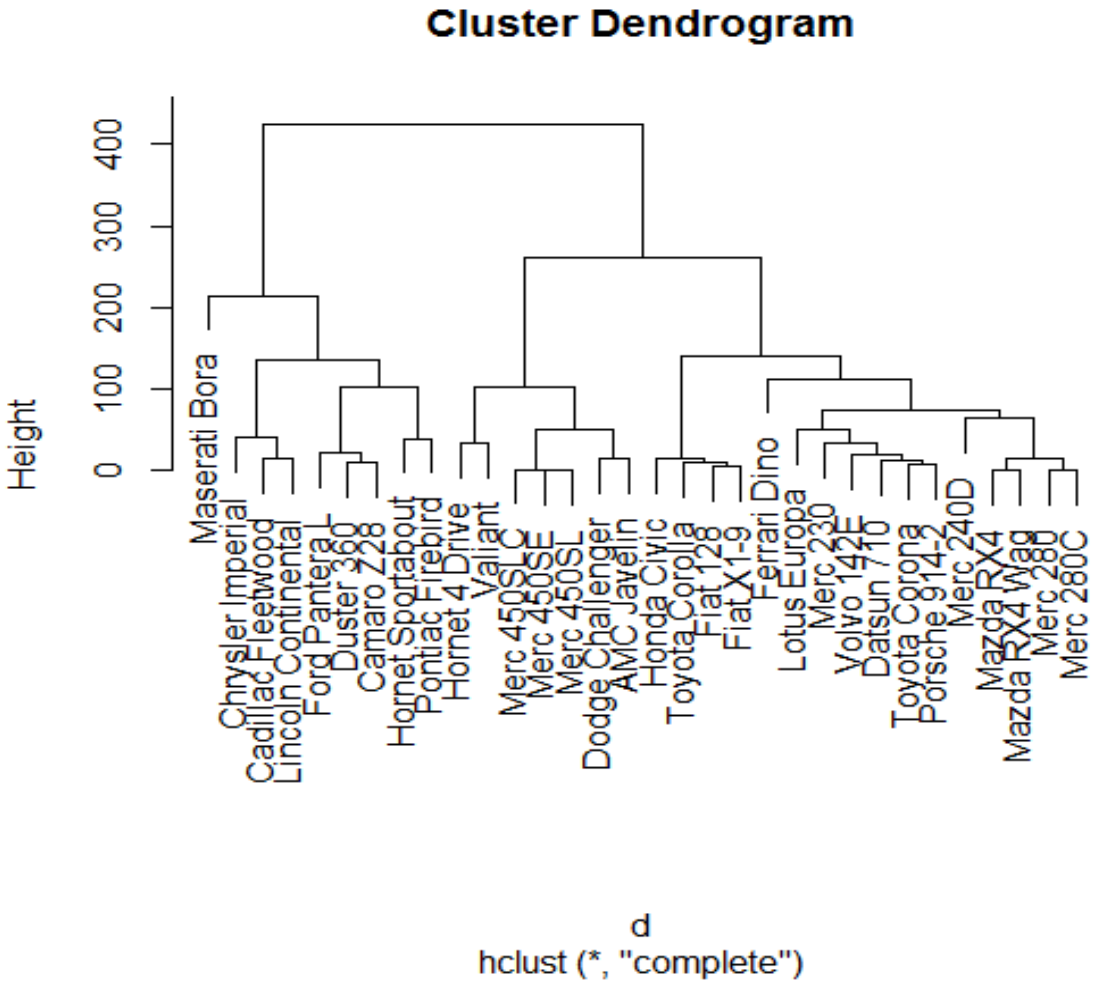
Απάντηση

Ο κώδικας επίλυσης του ερωτήματος παρατίθεται παρακάτω (Κώδικας 6.6).

```
> d <- dist(as.matrix(mtcars))   # Υπολογισμός μητρώου αποστάσεων
> hc <- hclust(d)                # Εφαρμογή Ιεραρχικής Συσταδοποίησης
> plot(hc)                       # Εκτύπωση δενδρογράμματος
>
```

Κώδικας 6.6 *Κώδικας επίλυσης κριτηρίου αξιολόγησης 2.*

Ουσιαστικά, πρώτα υπολογίζουμε την απόσταση των δειγμάτων για τη δημιουργία των συστάδων και στη συνέχεια, χρησιμοποιώντας την hclust(), εφαρμόζουμε την ιεραρχική συσταδοποίηση. Το δενδρόγραμμα, που παράγει ο αλγόριθμος, απεικονίζεται στην παρακάτω εικόνα (Εικόνα 6.15). Ο προεπιλεγμένος τρόπος σύνδεσης είναι ο πλήρης σύνδεσμος. Πειραματηστείτε δοκιμάζοντας διαφορετικά κριτήρια συνδέσμου.



Εικόνα ***6.15*** *Ιεραρχική συσταδοποίηση με χρήση του προεπιλεγμένου κριτηρίου συνδέσμου.*

Κριτήριο αξιολόγησης 3

Για τα δεδομένα του κριτηρίου αξιολόγησης 1 να χρησιμοποιηθεί ο αλγόριθμος DBSCAN. Ο αλγόριθμος είναι υλοποιημένος από το πακέτο `fpc` της R. Δοκιμάστε διαφορετικούς συνδυασμούς παραμέτρων. Τι παρατηρείτε; Συγκρίνετε τις απεικονίσεις του DBSCAN με αυτές του k-means.

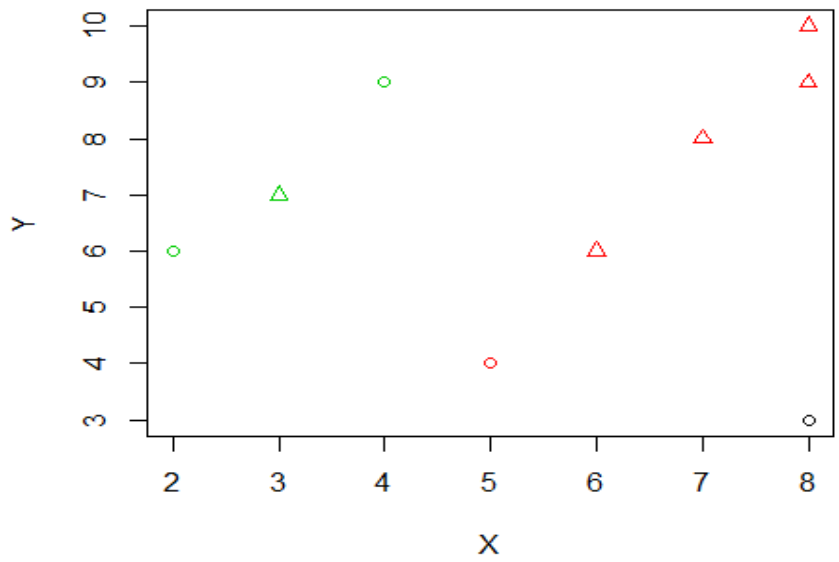
Απάντηση

Ο κώδικας επίλυσης του ερωτήματος παρατίθεται παρακάτω (Κώδικας 6.7).

```
> library(fpc)
>
> # Δεδομένα
> data <- data.frame(X=c(8, 3, 8, 4, 6, 5, 2, 8, 7),Y=c(10, 7,
3, 9, 6, 4, 6, 9, 8))
>
> clusters <- dbscan(data, eps=3, MinPts=3, showplot=TRUE)
>
```

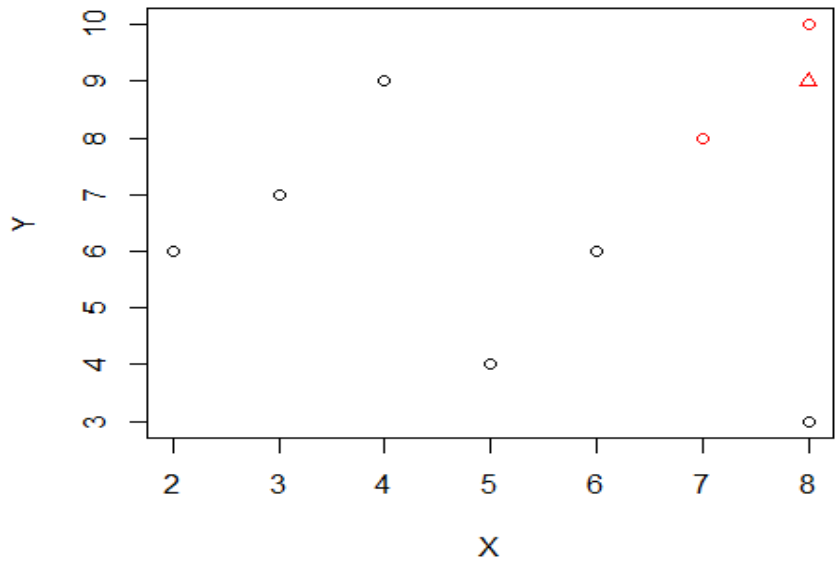
Κώδικας 6.7 *Κώδικας επίλυσης κριτηρίου αξιολόγησης 3.*

Η απεικόνιση που προκύπτει από τον Κώδικα 6.6 φαίνεται στην Εικόνα 6.16.

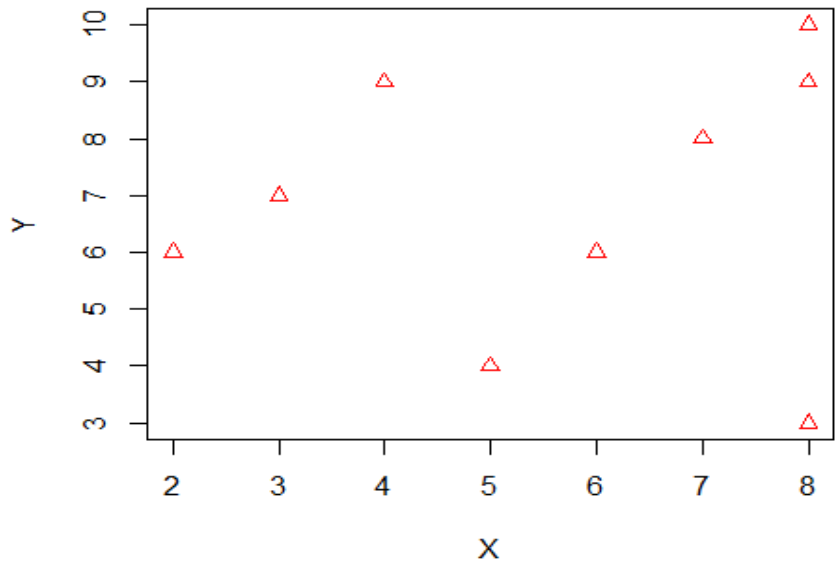


Εικόνα 6.16 *Δημιουργία συστάδων με τον DBSCAN για ε = MinPts = 3.*

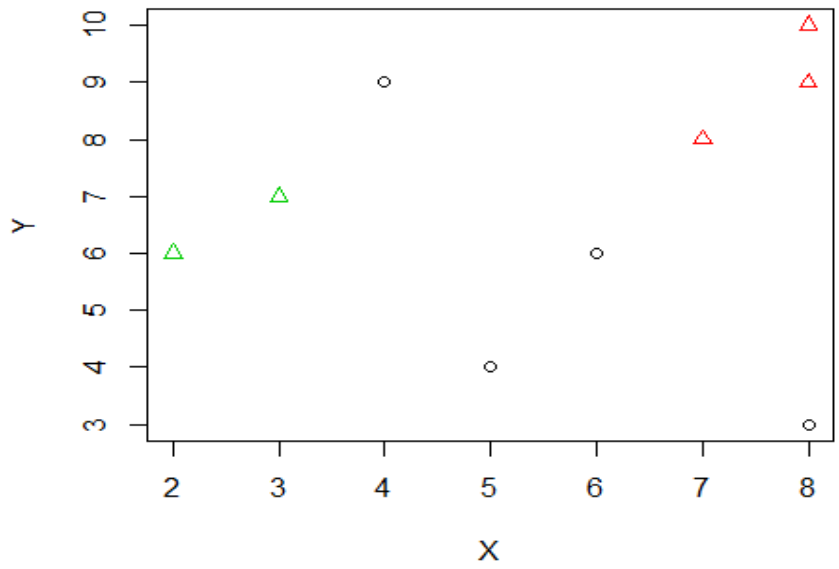
Με τριγωνικό σχήμα εμφανίζονται τα κεντρικά σημεία, ενώ με κυκλικό τα μη κεντρικά. Με διαφορετικό χρώμα, πράσινο και κόκκινο, διαχωρίζονται οι συστάδες. Με μαύρο χρώμα δηλώνονται σημεία τα οποία είναι ακραία, δηλαδή που ο αλγόριθμος τα χαρακτήρισε ως θόρυβο. Παρατηρούμε ότι ο αλγόριθμος DBSCAN βρήκε δυο συστάδες, ενώ ο k-means τρεις. Δοκιμάζοντας διαφορετικές τιμές για το ε και το MinPts, καταλήγουμε στο ότι ο συνδυασμός (3, 3) είναι ο καλύτερος. Χρησιμοποιώντας ε = 2 και MinPts = 3, καταλήγουμε σε πολλές ακραίες τιμές (Εικόνα 6.17). Χρησιμοποιώντας ε = 4 και MinPts = 3 ή MinPts = 4, καταλήγουμε σε 1 συστάδα μόνο (Εικόνα 6.18). Τέλος, για ε = MinPts = 2 δημιουργούνται δυο συστάδες, αλλά και πάλι έχουμε πολλές ακραίες τιμές (Εικόνα 6.19).



Εικόνα *6.17* *Δημιουργία συστάδων με τον DBSCAN για ε = 2, MinPts = 3.*



Εικόνα *6.18* *Δημιουργία συστάδων με τον DBSCAN για ε =4, MinPts = 3, 4.*



Εικόνα *6.19* *Δημιουργία συστάδων με τον DBSCAN για ε = 2, MinPts = 2.*

Παρατηρούμε, λοιπόν, ότι η σωστή επιλογή των παραμέτρων ε και MinPts αποτελεί βασική προϋπόθεση για την ορθή συσταδοποίηση των δεδομένων από τον αλγόριθμο.

Βιβλιογραφία

Tan, P. N., Steinbach, M. & Kumar, V. (2006). *Introduction to Data Mining.* Boston, MA: Pearson/Addison-Wesley.

Dunham, M. H. (2003). Data Mining: Introductory and Advanced Topics. *Pearson Education*, Upper Saddle River, N. J. Prentice Hall.

k-means clustering. Ανακτήθηκε στις 27 Νοεμβρίου 2015, από: http://www.rdatamining.com/examples/kmeans-clustering

Hierarchical Cluster Analysis. Ανακτήθηκε στις 27 Νοεμβρίου 2015, από: http://www.r-tutor.com/gpu-computing/clustering/hierarchical-cluster-analysis

Data Mining Algorithms In R/Clustering/K-Means. Ανακτήθηκε στις 27 Νοεμβρίου 2015, από: https://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R/Clustering/K-Means

DBSCAN. Ανακτήθηκε στις 29 Νοεμβρίου 2015, από: https://en.wikipedia.org/wiki/DBSCAN

Κεφάλαιο 7: Εξόρυξη Συχνών Στοιχειοσυνόλων και Κανόνων Συσχέτισης

Σύνοψη

Ο βασικός στόχος αυτού του κεφαλαίου είναι η εισαγωγή σε θέματα που αφορούν στην εξόρυξη συχνών στοιχειοσυνόλων και κανόνων συσχέτισης. Πιο συγκεκριμένα, παρουσιάζει στον αναγνώστη βασικές έννοιες των στοιχειοσυνόλων και των κανόνων συσχέτισης, των μέτρων της υποστήριξης και της εμπιστοσύνης, ενώ περιγράφει σε βάθος τον βασικό αλγόριθμο εξόρυξης συχνών στοιχειοσυνόλων, τον Apriori. Τέλος, παρουσιάζουμε το πακέτο arules της R, στο οποίο υπάρχουν υλοποιημένες αρκετές συναρτήσεις σχετικά με τα συχνά στοιχειοσύνολα και την εξόρυξη κανόνων συσχέτισης.

Προαπαιτούμενη γνώση

Πριν το τρέχον κεφάλαιο θα πρέπει να μελετηθεί τόσο το Κεφάλαιο 1 – Εισαγωγή στην Εξόρυξη Δεδομένων όσο και το Κεφάλαιο 2 – Εισαγωγή στην R.

Εξόρυξη Συχνών Στοιχειοσυνόλων και Κανόνων Συσχέτισης

7.1 Εισαγωγή

Όπως αναφέραμε και στην εισαγωγή του βιβλίου, μία από τις πιο διαδεδομένες εργασίες της εξόρυξης δεδομένων είναι και η εξόρυξη συχνών στοιχειοσυνόλων και κανόνων συσχέτισης. Ουσιαστικά, θα λέγαμε ότι η εργασία αυτή αποτέλεσε την κινητήρια δύναμη για την εξάπλωση του αντικειμένου της Εξόρυξης Δεδομένων με την ανακάλυψη του γνωστού πλέον εμπορικού αποτελέσματος, που αφορούσε στη μεγάλη συχνότητα, με την οποία νεαροί Αμερικανοί, που αγόραζαν πάνες για μωρά, εμφάνιζαν την τάση να αγοράζουν ταυτόχρονα και μπύρες. Στην πραγματικότητα, η ίδια αυτή εργασία είναι και το πρώτο "πράγμα" που έρχεται στο μυαλό μας, όταν ακούμε τον όρο Εξόρυξη Δεδομένων.

Πράγματι, τα συχνά στοιχειοσύνολα και οι κανόνες συσχέτισης αποτελούσαν νέα αντικείμενα, τα οποία δεν είχαν αναλυθεί στο παρελθόν στο πλαίσιο συγγενικών επιστημών με την Εξόρυξη Δεδομένων, όπως ήταν η Στατιστική και η Μηχανική Μάθηση. Επίσης, αποτελούσαν και ένα πολύ καλό παράδειγμα της καινοτόμου προσέγγισης που εφαρμοζόταν για την ανακάλυψη των προτύπων, η οποία δεν αφορούσε πλέον τη διατύπωση ενός ερωτήματος του ερευνητή, αλλά τη μεθοδολογική ανάλυση των δεδομένων με διαφορετικές τεχνικές για την ανακάλυψη κάτι "καινούριου". Μπορούμε να αναφέρουμε εδώ, και πριν υπεισέλθουμε σε λεπτομέρειες, ότι τόσο τα συχνά στοιχειοσύνολα όσο και οι κανόνες συσχέτισης αποτελούν, όπως λέμε, τοπικά πρότυπα τα οποία περιγράφουν διαφορετικά κομμάτια των δεδομένων, σε αντίθεση με τα μοντέλα, που αποτελούν σφαιρικά ή γενικά πρότυπα, που προσπαθούν να καταγράψουν ολόκληρο το σύνολο των δεδομένων.

Η εξόρυξη στοιχειοσυνόλων και κανόνων, σε κάποιο βαθμό, έγινε συνώνυμη με την ανάλυση του καλαθιού των αγορών, εξαιτίας του πεδίου, όπου η εφαρμογή έδωσε το προηγούμενο σημαντικό και αναπάντεχο αποτέλεσμα, το οποίο με τη σειρά του οδήγησε στο συμπέρασμα ότι υπάρχουν αναξιοποίητα πολλά μυστικά, κρυμμένα μέσα στα δεδομένα που συσσωρεύονται σε αποθετήρες δεδομένων (data warehouses). Τα μυστικά αυτά θα μπορούσαν να δώσουν λύσεις σε πολλαπλά προβλήματα, όπως είναι η προώθηση των προϊόντων, οι πωλήσεις, η τοποθέτηση προϊόντων στα ράφια και ο εφοδιασμός.

Αλλά, ας δούμε στη συνέχεια τι ακριβώς είναι το καλάθι αγορών και πώς αυτό σχετίζεται με το αντικείμενό μας. Όπως μάλλον εύκολα μπορεί να φανταστεί κανείς, το καλάθι αγορών είναι η συλλογή των προϊόντων που αγοράζει ένας πελάτης κατά την επίσκεψή του σε ένα κατάστημα λιανικής πώλησης. Η επίσκεψη αυτή εισάγεται στη βάση δεδομένων του καταστήματος κατά την τιμολόγηση των προϊόντων από τις ταμειακές μηχανές. Επομένως, η επίσκεψη ενός πελάτη σε ένα κατάστημα λιανικής πώλησης λέμε ότι έχει ως αποτέλεσμα την εισαγωγή μιας συναλλαγής ή δοσοληψίας μέσα στη βάση δεδομένων των πωλήσεων. Γι' αυτόν τον λόγο οι βάσεις δεδομένων αυτού του τύπου είναι γνωστές και ως βάσεις δοσοληψιών.

Με τον τρόπο αυτό τα καταστήματα λιανικής πώλησης συσσωρεύουν τεράστιες ποσότητες τέτοιων δεδομένων, καταγράφοντας όλες τις δοσοληψίες σε συνεχόμενη βάση καθ' όλη τη διάρκεια λειτουργίας των καταστημάτων αυτών. Ας δούμε, όμως, σε αυτό το σημείο τι ακριβώς είναι η ανάλυση καλαθιού αγοράς. Στό-

χος της ανάλυσης αυτής είναι η αναζήτηση προϊόντων, τα οποία συνήθως αγοράζονται ταυτόχρονα από τους πελάτες. Η πληροφορία αυτή μπορεί να οδηγήσει, όπως αναφέρθηκε προηγουμένως, σε πολλές λύσεις. Στην ορολογία της εξόρυξης κανόνων συσχέτισης, το σύνολο των προϊόντων ή γενικότερα το σύνολο των στοιχείων, που αγοράζονται μαζί, είναι γνωστό και ως στοιχειοσύνολο. Το πλήθος των στοιχείων ενός στοιχειοσυνόλου καθορίζει το μήκος του και συνήθως χρησιμοποιείται ο όρος i-στοιχειοσύνολο, για να αναπαραστήσει ένα στοιχειοσύνολο μήκους i, το οποίο αποτελείται, δηλαδή, από i στοιχεία.

Θα πρέπει να αναφέρουμε σε αυτό το σημείο ότι με την ανάλυση δεν μπορούν όλα τα στοιχειοσύνολα να βρεθούν ως ενδιαφέροντα από "εμπορική" άποψη. Πέρα από το αντικειμενικό κριτήριο του πόσο αναπάντεχη θα ήταν η εύρεση ενός στοιχειοσυνόλου (για παράδειγμα, δεν θα ήταν καθόλου αναπάντεχη η ανακάλυψη ενός στοιχειοσυνόλου, το οποίο θα αποτελείται από το ποτό τζιν και από τη σόδα), ένα σημαντικό κριτήριο, που καθορίζει την "αξία" ενός στοιχειοσυνόλου, σχετίζεται με τη συχνότητα εμφάνισης του στοιχειοσυνόλου αυτού στη βάση δεδομένων με τις πωλήσεις. Το κριτήριο αυτό αναφέρεται ως υποστήριξη του στοιχειοσυνόλου.

Συνήθως, οι ειδικοί επί των πωλήσεων θέτουν συγκεκριμένα όρια στη συχνότητα εμφάνισης ή αλλιώς στην υποστήριξη, πάνω από τα οποία ένα στοιχειοσύνολο θα ήταν, όπως λέμε, "συχνό". Είναι προφανές ότι το μέτρο της υποστήριξης θα πρέπει να έχει διαφορετικά όρια για διαφορετικά στοιχειοσύνολα.

Στη συνέχεια θα δούμε γιατί είναι σημαντικό το πρόβλημα, στο οποίο έχουμε εστιάσει. Αρχικά θα πρέπει να κατανοήσουμε ότι η επίλυση ενός τέτοιου προβλήματος, στο πλαίσιο της Εξόρυξης Δεδομένων, θα είναι αλγοριθμικής φύσεως. Θα χρειαστεί, δηλαδή, να σχεδιάσουμε και να υλοποιήσουμε έναν αλγόριθμο, ο οποίος θα είναι σε θέση να βρίσκει για εμάς όλα τα ενδιαφέροντα ή αλλιώς συχνά στοιχειοσύνολα.

Όμως, ένας τέτοιος αλγόριθμος έχει να αντιμετωπίσει δύο πολύ σοβαρά προβλήματα. Το πρώτο έχει να κάνει με το γεγονός ότι πολύ συχνά οι βάσεις δοσοληψιών έχουν τεράστια μεγέθη, που συνήθως δεν μπορούν να φορτωθούν συνολικά στην κεντρική μνήμη του υπολογιστή. Το παραπάνω γεγονός αυξάνει τη δυσκολία σχεδιασμού του αλγορίθμου. Από την άλλη μεριά, το πλήθος των στοιχειοσυνόλων, που μπορούν να παραχθούν, αυξάνει εκθετικά με το πλήθος των στοιχείων που ο αλγόριθμος δέχεται ως είσοδο, έστω και αν τελικά το πλήθος των συχνών στοιχειοσυνόλων, τα οποία στην ουσία μας ενδιαφέρουν, είναι στην πραγματικότητα πάρα πολύ μικρότερο.

Για να ξεπεράσουμε τα δύο παραπάνω προβλήματα, θα πρέπει να σχεδιάσουμε αλγορίθμους εκτός μνήμης (out of core), οι οποίοι έχουν γραμμική κλιμάκωση τόσο σε σχέση με το πλήθος των δοσοληψιών όσο και σε σχέση με το πλήθος των στοιχείων. Στη συνέχεια δίνεται το θεωρητικό υπόβαθρο, πριν συνεχίσουμε με την περιγραφή ενός αλγορίθμου, ο οποίος αποτέλεσε τη βάση για την επίλυση του προβλήματος αυτού.

7.2 Θεωρητικό Υπόβαθρο

Έστω μία βάση δοσοληψιών $T=\{T_1,T_2,\ldots T_n\}$, από την οποία θέλουμε να βρούμε τα στοιχειοσύνολα που εμφανίζονται συχνά και έστω $I=\{i_1,i_2,\ldots i_m\}$ ένα σύνολο στοιχείων (items). Για κάθε μία από τις δοσοληψίες T_i της βάσης θεωρούμε ότι το σύνολο των στοιχείων, που εμφανίζονται στη δοσοληψία, αποτελεί υποσύνολο του I, δηλαδή $T_i \subseteq I$. Σε κάθε δοσοληψία αντιστοιχεί ένα μοναδικό αναγνωριστικό με όνομα TID. Ένα παράδειγμα μιας βάσης δοσοληψιών δίνεται στον Πίνακα 7.1.

Tid	**Items**
1	ABDE
2	BCE
3	DE
4	ACDE

Πίνακας ***7.1*** *Βάση Δοσοληψιών.*

Έστω τώρα X και Y δύο στοιχειοσύνολα. Μία δοσοληψία T_i λέμε ότι περιέχει το στοιχειοσύνολο X, εάν και μόνο εάν ισχύει ότι $X \subseteq T_i$. Με βάση τον ορισμό των στοιχειοσυνόλων μπορούμε να ορίσουμε στη συνέχεια έναν κανόνα συσχέτισης (association rule), ο οποίος είναι στην ουσία ένας κανόνας συμπερασμού της μορφής $X \Rightarrow Y$, όπου $X \subseteq I$ και $Y \subseteq I$, ενώ θα πρέπει ταυτόχρονα να ισχύει και ότι $X \cap Y = \emptyset$. Για κάθε κανόνα συσχέτισης ορίζουμε ένα μέτρο που ονομάζεται εμπιστοσύνη (confidence), το οποίο φανερώνει τη δύναμη με την οποία συσχετίζονται τα δύο μέλη (δεξιό και αριστερό) του κανόνα. Η εμπιστοσύνη ενός κανόνα $X \Rightarrow Y$ ορίζεται με βάση την υποστήριξη των στοιχειοσυνόλων X και $X \cup Y$ και προκύπτει από τον τύπο conf = sup-

p(X∪Y)/supp(X), όπου supp(·) είναι το μέτρο της υποστήριξης.

Για παράδειγμα, εάν χρησιμοποιήσουμε τα δεδομένα του Πίνακα 7.1 και θεωρήσουμε τον κανόνα Β ⇒ Ε, τότε θα ισχύει supp(B) = 2/4, supp(B∪E) = 2/4, ενώ conf(B⇒E) = 1. Η τιμή της εμπιστοσύνης για τον κανόνα Β ⇒ Ε, που ισούται με το 1, η οποία αποτελεί και τη μέγιστη τιμή του μέτρου της εμπιστοσύνης ενός κανόνα συσχέτισης, υποδηλώνει ότι όποτε εμφανίζεται το Β σε μία δοσοληψία, τότε θα εμφανίζεται και το Ε με πιθανότητα ίση με 1. Παρατηρήστε σε αυτό το σημείο ότι:

$$\text{conf}(\text{E}\Rightarrow\text{B}) = \frac{\text{supp}(\text{E}\cup\text{B})}{\text{supp}(\text{E})} = \frac{\frac{2}{4}}{\frac{4}{4}} = 0.5$$

επειδή στις δοσοληψίες εκείνες που εμφανίζεται το Ε, μόνο στις μισές εμφανίζεται και το Β.

Πολλές φορές είναι σημαντικό, προκειμένου να βρούμε τους κανόνες συσχέτισης με υψηλή τιμή του μέτρου εμπιστοσύνης, να εστιάσουμε σε εκείνους τους κανόνες, που έχουν υψηλή υποστήριξη. Οι κανόνες εκείνοι, που έχουν υψηλή υποστήριξη και υψηλή εμπιστοσύνη, ονομάζονται ισχυροί κανόνες. Ο στόχος ενός αλγόριθμου εξόρυξης κανόνων συσχέτισης είναι να βρει τους ισχυρούς κανόνες σε μία μεγάλη βάση δοσοληψιών με τον πλέον αποτελεσματικό τρόπο. Πιο συγκεκριμένα, το πρόβλημα της εξόρυξης κανόνων συσχέτισης μπορεί να αναλυθεί σε δύο φάσεις:

(a) εύρεση των μεγάλων (ή συχνών) στοιχειοσυνόλων, που έχουν ελάχιστη υποστήριξη ίση με supp_{min}, και
(b) εύρεση των κανόνων συσχέτισης, που έχουν ελάχιστη εμπιστοσύνη ίση με conf_{min}, κάνοντας χρήση των μεγάλων στοιχειοσυνόλων, που δημιουργήθηκαν στο προηγούμενο βήμα.

Η συνολική πολυπλοκότητα ενός αλγορίθμου εξόρυξης κανόνων συσχέτισης κυριαρχείται από το πρώτο βήμα, που είναι και το πιο "βαρύ" υπολογιστικά. Μετά την εύρεση των μεγάλων στοιχειοσυνόλων, οι αντίστοιχοι κανόνες συσχέτισης μπορούν να βρεθούν με έναν άμεσο τρόπο.

Στη συνέχεια παρουσιάζουμε έναν πρωτοπόρο αλγόριθμο, ο οποίος έλυσε το πρόβλημα της αποτελεσματικής μέτρησης των μεγάλων στοιχειοσυνόλων. Ο αλγόριθμος αυτός είναι γνωστός ως Αλγόριθμος Apriori και κάνει χρήση της ακόλουθης ιδιότητας των στοιχειοσυνόλων: εάν $X \subseteq Y \Rightarrow \text{supp}(X) \geq \text{supp}(Y)$, δηλαδή, εάν ένα στοιχειοσύνολο X είναι υποσύνολο ενός στοιχειοσυνόλου Y, τότε η υποστήριξη του στοιχειοσυνόλου X είναι τουλάχιστον ίση με αυτή του στοιχειοσυνόλου Y. Μία σειρά άλλων τεχνικών, οι οποίες βελτίωσαν τον πρώτο αλγόριθμο, θα αναφερθούν, επίσης, μετά την παρουσίαση του αλγορίθμου Apriori.

7.3 Ο Αλγόριθμος Apriori

Ο αλγόριθμος Apriori είναι ένας από τους πιο γνωστούς και δημοφιλείς αλγορίθμους στην Εξόρυξη Δεδομένων, ενώ έχει ευρέως χρησιμοποιηθεί στο παρελθόν για τον υπολογισμό των μεγάλων στοιχειοσυνόλων σε μία βάση δοσοληψιών. Ο Apriori δουλεύει σε επίπεδα (levelwise), τα οποία αντιστοιχούν στο πλήθος των στοιχείων ενός στοιχειοσυνόλου που εξετάζεται σε κάθε επίπεδο, δουλεύοντας με επαναλήψεις από το ένα επίπεδο στο επόμενο, ξεκινώντας από το πρώτο επίπεδο και προχωρώντας μέχρι το επίπεδο εκείνο, όπου για πρώτη φορά δεν θα εμφανιστούν μεγάλα στοιχειοσύνολα. Η διαδικασία αυτή θα παρουσιαστεί στη συνέχεια λεπτομερώς, οπότε και θα δοθεί συγκεκριμένο παράδειγμα.

Σε κάθε επανάληψη του αλγορίθμου λαμβάνουν χώρα δύο φάσεις. Κατά τη διάρκεια της πρώτης φάσης δημιουργούνται τα υποψήφια στοιχειοσύνολα, ενώ στη δεύτερη γίνεται καταμέτρηση της υποστήριξης των υποψηφίων στοιχειοσυνόλων και επιλογή των μεγάλων. Στην πρώτη φάση της πρώτης επανάληψης το σύνολο των υποψηφίων i-στοιχειοσυνόλων περιλαμβάνει όλα τα στοιχεία που εμφανίζονται στη βάση δοσοληψιών. Κατά τη διάρκεια της φάσης της καταμέτρησης ο αλγόριθμος μετράει την υποστήριξη όλων των στοιχείων, κάνοντας ένα πέρασμα μέσα από τη βάση, δηλαδή ουσιαστικά το πρώτο πέρασμα. Στη συνέχεια, από τα στοιχεία, των οποίων η υποστήριξη καταμετρήθηκε, διατηρούνται για περαιτέρω επεξεργασία μόνο αυτά, των οποίων η υποστήριξη βρέθηκε μεγαλύτερη ή ίση από το κατώφλι της ελάχιστης υποστήριξης, που έχει αρχικά καθοριστεί. Με αυτόν τον τρόπο, και μετά το τέλος της πρώτης επανάληψης, έχουν βρεθεί όλα τα μεγάλα στοιχειοσύνολα μήκους 1.

Να αναφέρουμε σε αυτό το σημείο ότι ο αλγόριθμος Apriori σε κάθε πέρασμα κάνει χρήση δύο συνόλων στοιχειοσυνόλων με όνομα αντίστοιχο C και L για την αποθήκευση των υποψηφίων (Candidates) και των μεγάλων στοιχειοσυνόλων (Large). Τελειώνοντας με τις δύο φάσεις της πρώτης επανάληψης, ο αλγόριθμος

Apriori προχωράει στη δεύτερη επανάληψη, εφόσον φυσικά το σύνολο των μεγάλων στοιχειοσυνόλων δεν είναι κενό. Κατά τη διάρκεια της πρώτης φάσης της δεύτερης επανάληψης, ο αλγόριθμος θα πρέπει να δημιουργήσει τα υποψήφια στοιχειοσύνολα μεγέθους δύο, δηλαδή να καθορίσει το σύνολο C_2. Για να το κάνει αυτό ο Apriori χρησιμοποιεί μία ρουτίνα, η οποία, συνδυάζοντας δύο μεγάλα στοιχειοσύνολα μεγέθους 1, δημιουργεί ένα υποψήφιο στοιχειοσύνολο μεγέθους 2. Αυτό βέβαια σε επόμενες επαναλήψεις εφαρμόζεται γενικότερα, οπότε για κάθε δύο μεγάλα στοιχειοσύνολα μεγέθους i-1 κατάλληλα επιλεγμένα προκύπτει ένα υποψήφιο στοιχειοσύνολο μεγέθους i.

Στο σημείο της παραγωγής των υποψηφίων στοιχειοσυνόλων και για την περικοπή κάποιων, που "εκ των προτέρων" γνωρίζουμε ότι δεν μπορεί να είναι μεγάλα, εφαρμόζεται, επίσης, η αρχή του Apriori. Η αρχή αυτή βασίζεται στην αντιμονότονη ιδιότητα της υποστήριξης, που λέει ότι η υποστήριξη ενός στοιχειοσυνόλου δεν μπορεί να υπερβαίνει την υποστήριξη των υποσυνόλων του. Πιο συγκεκριμένα και με βάση την αρχή Apriori, ένα υποψήφιο στοιχειοσύνολο, που περιέχει ένα όχι μεγάλο υποσύνολο, θα πρέπει να εξαιρεθεί από την καταμέτρηση της δεύτερης φάσης, αφού δεν μπορεί να έχει υποστήριξη μεγαλύτερη από αυτή του μη συχνού υποσυνόλου του και, επομένως, δεν θα μπει στα μεγάλα στοιχειοσύνολα του αντίστοιχου επιπέδου. Ο Κώδικας 7.1 παρουσιάζει μία υψηλού επιπέδου περιγραφή των βημάτων του Apriori αλγορίθμου.

```
1. Έστω k=1
2. Δημιουργία συχνών στοιχειοσυνόλων (ΣΣ) μήκους 1
3. Επανέλαβε μέχρις ότου δεν ανιχνεύονται νέα συχνά ΣΣ
   i.  Δημιούργησε μήκους (k+1) υποψήφια ΣΣ από τα μεγέθους k
       συχνά ΣΣ
   ii. Κλάδεμα υποψηφίων ΣΣ που περιέχουν υποσύνολα μεγέθους k,
       τα οποία είναι μη συχνά
   iii.   Μέτρηση της υποστήριξης κάθε υποψηφίου με πέρασμα
       της βάσης δοσοληψιών
   iv. Απαλοιφή υποψηφίων που είναι μη συχνά, αφήνοντας μόνο
       εκείνα που είναι συχνά
```

Κώδικας ***7.1*** *Βήματα του Αλγόριθμου Apriori.*

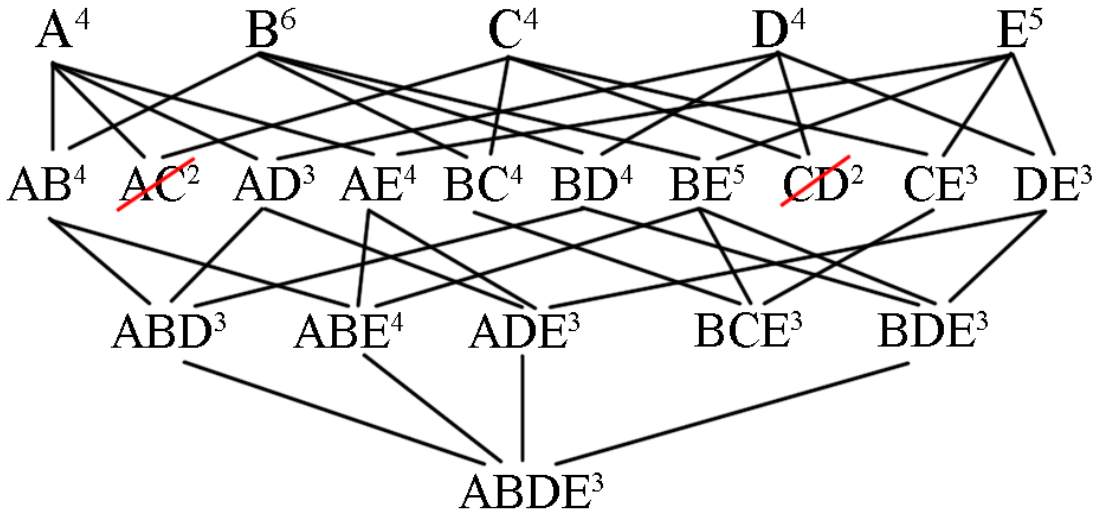
Στη συνέχεια, δίνεται ένα συγκεκριμένο παράδειγμα εφαρμογής του αλγορίθμου Apriori, κάνοντας χρήση της βάσης δοσοληψιών του Πίνακα 7.2.

Tid	Items
1	ABDE
2	BCE
3	ABDE
4	ABCE
5	ABCDE
6	BCD

Πίνακας ***7.2*** *Η βάση δεδομένων δοσοληψιών για το παράδειγμα του αλγορίθμου Apriori.*

Εφαρμόζουμε τον αλγόριθμο Apriori στη βάση δοσοληψιών του Πίνακα 7.2. Υπολογίζουμε το σύνολο C1 των υποψηφίων στοιχειοσυνόλων (ΣΣ) μεγέθους 1, C1={A, B, C, D, E} και μετράμε τις υποστηρίξεις των ΣΣ του, οπότε C1={A:4, B:6, C:4, D:4, E:5}. Αφού το κατώφλι του μετρητή υποστήριξης είναι ίσο με 3, αυτό σημαίνει ότι το σύνολο των συχνών ΣΣ μεγέθους 1 είναι το L1={A, B, C, D, E}.

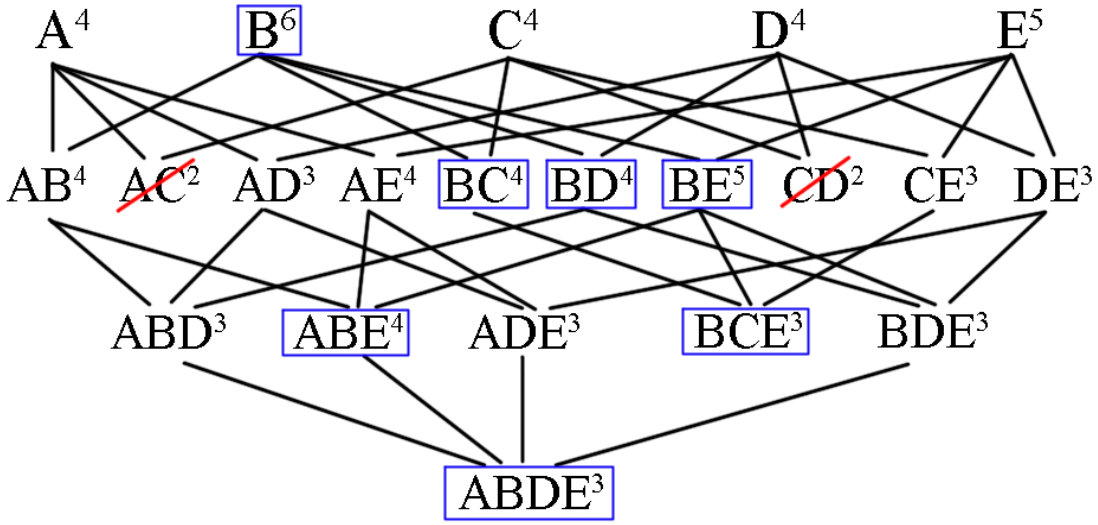
Στη συνέχεια, δημιουργούμε το C2={AB, AC, AD, AE, BC, BD, BE, CD, CE, DE} και μετράμε τις υποστηρίξεις, οπότε C2={AB:4, AC:2, AD:3, AE:4, BC:4, BD:4, BE:5, CD:2, CE:3, DE:3} και L2={AB, AD, AE, BC, BD, BE, CE, DE}. Στη συνέχεια, δημιουργούμε το C3={ABD, ABE, ADE, BCE, BDE}, το οποίο με τις υποστηρίξεις γίνεται C3={ABD:3, ABE:4, ADE:3, BCE:3, BDE:3}. Από τις υποστηρίξεις των στοιχείων του C3 βρίσκουμε το L3={ABD, ABE, ADE, BCE, BDE}. Από το L3 υπολογίζουμε το C4={ABDE}, απ' όπου βρίσκουμε το C4={ABDE:3} και το L4={ABDE}. Παρακάτω φαίνεται το πλέγμα όλων των συχνών στοιχειοσυνόλων που βρέθηκαν από τον αλγόριθμο (Εικόνα 7.1).



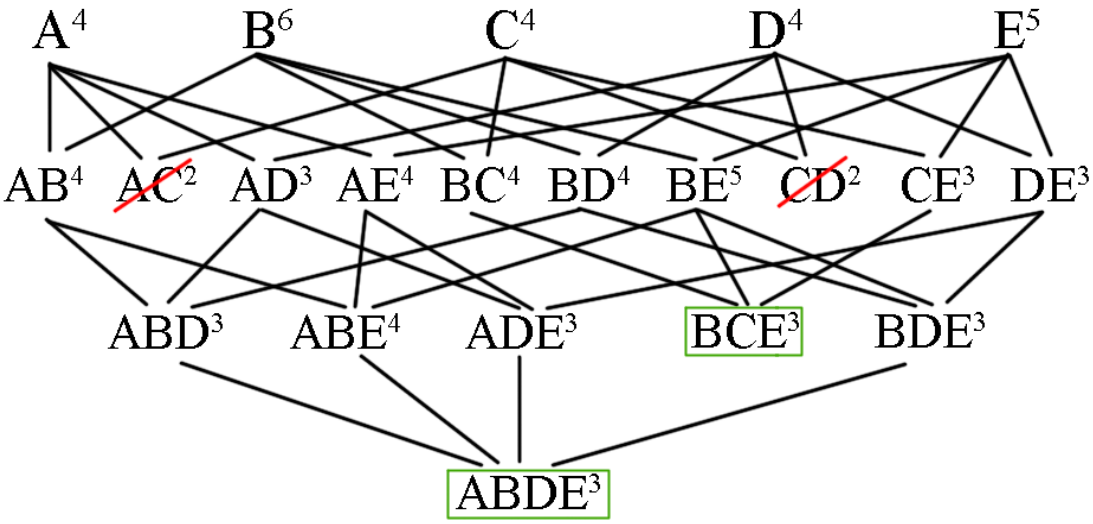
Εικόνα *7.1 Πλέγμα συχνών στοιχειοσυνόλων του παραδείγματος.*

7.4 Είδη Συχνών Στοιχειοσυνόλω

Όπως ήδη ορίσαμε, συχνά καλούνται όλα τα στοιχειοσύνολα που υποστηρίζονται από τόσες δοσοληψίες μιας βάσης, ώστε η υποστήριξή τους να είναι μεγαλύτερη από ή ίση με το δοθέν κατώφλι. Ένα υποσύνολο των συχνών στοιχειοσυνόλων είναι τα κλειστά στοιχειοσύνολα. Κλειστά λέμε τα στοιχειοσύνολα, τα οποία είναι συχνά και δεν υπάρχει κανένα υπερσύνολο με την ίδια υποστήριξη (Εικόνα 7.2). Για παράδειγμα, αν τα AB και ABC είναι συχνά και έχουν την ίδια υποστήριξη, τότε το AB δεν είναι κλειστό συχνό στοιχειοσύνολο. Ένα υποσύνολο των συχνών κλειστών στοιχειοσυνόλων είναι τα μέγιστα στοιχειοσύνολα. Μέγιστα λέμε τα στοιχειοσύνολα, τα οποία είναι συχνά και κανένα υπερσύνολό τους δεν είναι συχνό (Εικόνα 7.3).



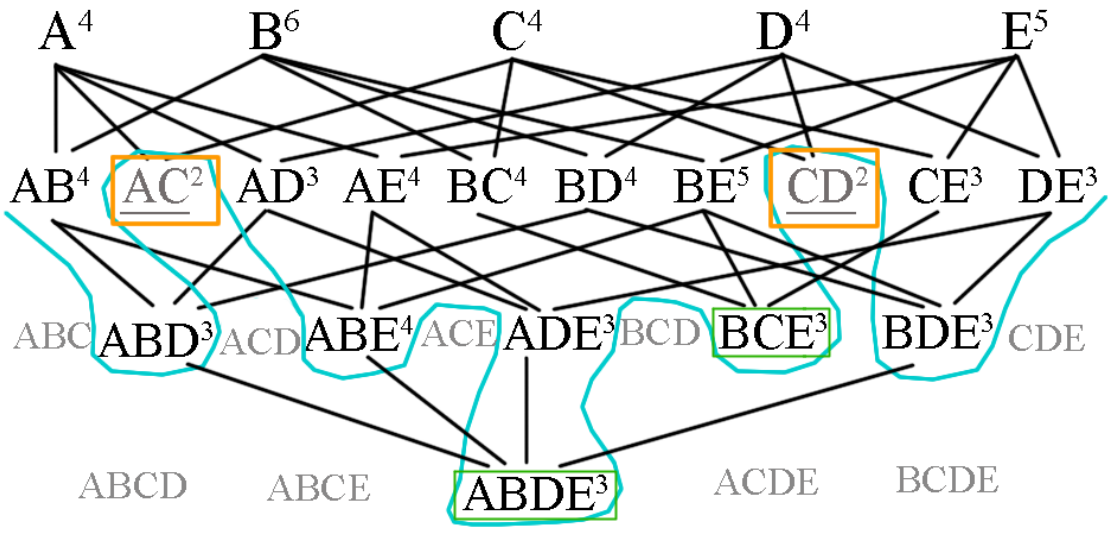
Εικόνα *7.2 Κλειστά συχνά στοιχειοσύνολα (με μπλε πλαίσιο) για το παράδειγμα της υποενότητας 7.2.*



Εικόνα *7.3 Μέγιστα συχνά στοιχειοσύνολα (με πράσινο πλαίσιο) για το παράδειγμα της υποενότητας 7.2.*

7.4 Θετικό και Αρνητικό Περιθώριο Συχνών Στοιχειοσυνόλων

Το θετικό και το αρνητικό περιθώριο ορίζουν μια νοητή γραμμή στο πλέγμα των στοιχειοσυνόλων, η οποία χωρίζει τα συχνά από τα μη συχνά στοιχειοσύνολα (Εικόνα 7.4, γαλάζια γραμμή). Το θετικό περιθώριο είναι τα μέγιστα συχνά στοιχειοσύνολα και συμβολίζεται ως $BD^{+}(L)$. Συνεπώς, για το παράδειγμά μας το θετικό περιθώριο αποτελείται από τα ABDE και BCE (Εικόνα 7.3). Το αρνητικό περιθώριο αποτελείται από όλα τα μη συχνά στοιχειοσύνολα με το ελάχιστο μήκος, των οποίων τα υποσύνολά τους είναι συχνά, και συμβολίζεται με $BD^{-}(L)$. Στην Εικόνα 7.4 τα μη συχνά στοιχειοσύνολα εμφανίζονται με γκρι χρώμα. Τα AC και CD είναι τα μοναδικά μη συχνά στοιχειοσύνολα με όλα τα υποσύνολά τους να είναι συχνά. Επομένως, για το παράδειγμά μας το αρνητικό περιθώριο αποτελείται από τα AC και CD (Εικόνα 7.4).



Εικόνα *7.4 Θετικό περιθώριο (με πράσινο πλαίσιο) και αρνητικό περιθώριο (με πορτοκαλί πλαίσιο) για το παράδειγμα της υποενότητας 7.2.*

7.5 Εξόρυξη Κανόνων Συσχέτισης

Όταν τα συχνά στοιχειοσύνολα έχουν βρεθεί, η εξόρυξη των κανόνων συσχέτισης είναι μία αρκετά απλή διαδικασία. Πιο συγκεκριμένα, για κάθε ένα από τα συχνά στοιχειοσύνολα L υπολογίζονται όλα τα μη κενά και γνήσια υποσύνολα τους, έστω $X_1, X_2, ..., X_v$. Στη συνέχεια, γίνεται υπολογισμός του μέτρου της εμπιστοσύνης για κάθε συνδυασμό $X_i \Rightarrow X_j$, όπου i, j = 1,...,v και i≠ j, που προκύπτουν από τα υποσύνολα αυτά. Εάν το μέτρο της εμπιστοσύνης

$$conf(\mathrm{Xi} \Rightarrow \mathrm{Xj}) = \frac{supp(Xj \cup Xi)}{supp(Xi)}$$

είναι μεγαλύτερο από το κατώφλι της ελάχιστης εμπιστοσύνης, τότε ο κανόνας $X_i \Rightarrow X_j$ είναι ένας ισχυρός κανόνας, οπότε και παρουσιάζεται στην έξοδο.

Για παράδειγμα, για τη βάση δεδομένων του προηγούμενου παραδείγματος (Πίνακας 7.3) βρήκαμε ότι το AB είναι συχνό στοιχειοσύνολο με υποστήριξη 4. Από αυτό το στοιχειοσύνολο προκύπτουν δύο πιθανά υποσύνολα: το A και το B. Επομένως, υπάρχουν δύο πιθανοί συνδυασμοί κανόνων συσχέτισης: A ⇒ B και B ⇒ A. Για τον κανόνα συσχέτισης A ⇒ B έχουμε ότι το A εμφανίζεται σε 4 δοσοληψίες, supp(A) = 4, ενώ από αυτές τις 4 το B εμφανίζεται σε όλες, supp(A∪B) = 4. Επομένως, η εμπιστοσύνη (confidence) του κανόνα συσχέτισης είναι:

$$conf(\mathrm{A} \Rightarrow \mathrm{B}) = \frac{supp(A \cup B)}{supp(A)} = \frac{4}{4} = 1.$$

Αντίστοιχα, για τον κανόνα συσχέτισης B ⇒ A έχουμε ότι το B εμφανίζεται σε 6 δοσοληψίες, supp(B) = 6. Από αυτές τις 6 δοσοληψίες μόνο οι 4 περιέχουν και το A, supp(A∪B) = 4. Συνεπώς, η εμπιστοσύνη (confidence) του κανόνα συσχέτισης είναι:

$$conf(\mathrm{B} \Rightarrow \mathrm{A}) = \frac{supp(A \cup B)}{supp(B)} = \frac{4}{6} = 0.667.$$

Προφανώς, για ένα συχνό στοιχειοσύνολο μεγαλύτερου μήκους θα είχαμε περισσότερους συνδυασμούς υποσυνόλων και, επομένως, περισσότερους κανόνες συσχέτισης. Για παράδειγμα, από το συχνό στοιχειοσύνολο ABD προκύπτουν 6 κανόνες συσχέτισης, όπως φαίνονται στον Πίνακα 7.3 μαζί με τον υπολογισμό του μέτρου εμπιστοσύνης για τον καθένα.

Κανόνας Συσχέτισης	**Αριθμητής**	**Παρονομαστής**	**Εμπιστοσύνη**
AB ⇒ D	supp(AB ∪ D) = 3	supp(D) = 4	0.75
AD ⇒ B	supp(AB ∪ B) = 3	supp(B) = 6	0.5
BD ⇒ A	supp(AB ∪ A) = 3	supp(A) = 4	0.75
A ⇒ BD	supp(A ∪ BD) = 3	supp(BD) = 4	0.75
B ⇒ AD	supp(B ∪ AD) = 3	supp(AD) = 3	1
D ⇒ AB	supp(D ∪ AB) = 3	supp(AB) = 4	0.75

Πίνακας 7.*3* *Κανόνες συσχέτισης, που προκύπτουν από το συχνό στοιχειοσύνολο ABD.*

7.6 Εναλλακτικές Τεχνικές Παραγωγής Μεγάλων Στοιχειοσυνόλων

ίναι πολύ σημαντικό οι τεχνικές, οι οποίες προτείνονται για την επίλυση του προβλήματος της εξόρυξης των μεγάλων στοιχειοσυνόλων, να είναι όσο γίνεται πιο αποτελεσματικές από πλευράς χώρου και χρόνου που απαιτούν, γιατί το πλήθος των στοιχειοσυνόλων, που μπορεί να προκύψει ακόμα και για ένα μικρό προς μεσαίο πλήθος στοιχείων, μπορεί να είναι υπερβολικά μεγάλο.

Ουσιαστικά, το μεγάλο πρόβλημα του Apriori αλγορίθμου είναι το πλήθος των περασμάτων που κάνει στη βάση των δοσοληψιών. Δύο από τους αλγορίθμους, που βελτιώνουν τη συμπεριφορά του Apriori ως προς το πλήθος των περασμάτων, είναι ο αλγόριθμος της διαμέρισης και ο αλγόριθμος της δειγματοληψίας. Και οι δύο αυτοί αλγόριθμοι περιορίζουν τα περάσματα της βάσης σε δύο.

7.6.1 Αλγόριθμος Δειγματοληψίας

Ο αλγόριθμος δειγματοληψίας χρησιμοποιείται για την αντιμετώπιση προβλημάτων που σχετίζονται με το μεγάλο μέγεθος βάσεων δεδομένων. Ουσιαστικά, γίνεται δειγματοληψία της βάσης δεδομένων και εφαρμογή του Apriori στο επιλεγμένο δείγμα. Ο αλγόριθμος χρησιμοποιεί τις έννοιες των ενδεχομένως συχνών στοιχειοσυνόλων, έστω PL, και του αρνητικού περιθωρίου του, $BD^{-}(PL)$. Τα ενδεχομένως συχνά στοιχειοσύνολα είναι τα συχνά στοιχειοσύνολα, τα οποία εντοπίζει ο αλγόριθμος για το επιλεγμένο δείγμα. Κατ' επέκταση το αρνητικό του περιθώριο, $BD^{-}(PL)$, είναι όλα τα μη συχνά στοιχειοσύνολα, των οποίων τα υποσύνολά τους είναι ενδεχομένως συχνά στοιχειοσύνολα. Ο ψευδοκώδικας του αλγορίθμου φαίνεται παρακάτω (Κώδικας 7.2).

```
Ds = δειγματοληψία της βάσης D;
PL = συχνά στοιχειοσύνολα του Ds;
C = PL ∪ BD⁻(PL);
L = καταμέτρηση υποστήριξης C στη βάση D;
ML = στοιχειοσύνολα στο BD⁻(PL) τα οποία βρέθηκαν ότι είναι
συχνά;
ΑΝ ML = Ø ΤΟΤΕ
ΤΕΛΟΣ
ΑΛΛΙΩΣ
C = L;
ΕΠΑΝΕΛΑΒΕ
C = C ∪ BD⁻(C);
ΜΕΧΡΙΣ ΟΤΟΥ BD⁻(C) = Ø
L = καταμέτρηση υποστήριξης C στη βάση D;
```

Κώδικας *7.2 Ψευδοκώδικας αλγορίθμου δειγματοληψίας.*

Αρχικά, γίνεται η δειγματοληψία της βάσης και επιλέγεται το δείγμα D_s. Στη συνέχεια, υπολογίζονται τα ενδεχομένως συχνά στοιχειοσύνολα στο D_s και τα υποψήφια τελικά συχνά στοιχειοσύνολα C, οπότε γίνεται το πρώτο πέρασμα στη βάση D, για την καταμέτρηση υποστήριξης των υποψήφιων στο σύνολο C. Από αυτή την καταμέτρηση, υπάρχει περίπτωση να σχηματιστεί ένα νέο σύνολο, ML, που περιέχει στοιχειοσύνολα του $BD^{-}(PL)$, τα οποία βρέθηκαν τελικά ότι είναι συχνά στην αρχική βάση D. Αν το σύνολο ML είναι κενό, τότε ο αλγόριθμος τερματίζει, αφού βρέθηκαν όλα τα τελικά συχνά στοιχειοσύνολα. Σε αντίθετη περίπτωση, γίνεται επέκταση του συνόλου των υποψηφίων C, προσθέτοντας επαναληπτικά το αρνητικό περιθώριο τους, $BD^{-}(C)$, μέχρις ότου το $BD^{-}(C)$ να είναι κενό. Τότε γίνεται το τελικό (δεύτερο) πέρασμα της αρχικής βάσης D, για την καταμέτρηση της υποστήριξης των υποψήφιων στοιχειοσυνόλων C. Όσα έχουν υποστήριξη μεγαλύτερη από το κατώφλι υποστήριξης είναι και τα τελικά συχνά στοιχειοσύνολα.

Ο αλγόριθμος μειώνει το πλήθος των σαρώσεων της βάσης δεδομένων σε ένα, στην καλύτερη περίπτωση,

ή σε δύο στη χειρότερη. Επιπλέον, έχει καλύτερη κλιμάκωση σε σχέση με τον Apriori, αφού είναι αποδοτικός τόσο σε μικρές όσο και σε μεγάλες βάσεις δεδομένων. Το κύριο μειονέκτημά του είναι η πιθανή παραγωγή πολλών υποψήφιων στοιχειοσυνόλων στη δεύτερη σάρωση της βάσης, εξαιτίας του επαναληπτικού υπολογισμού του αρνητικού περιθωρίου των στοιχειοσυνόλων στο C.

7.6.2 Αλγόριθμος Διαμέρισης

Ο αλγόριθμος διαμέρισης ξεκινάει με τον διαχωρισμό της αρχικής βάσης D σε διαμερίσεις, έστω D_1, D_2, …, D_p. Η βασική ιδέα πίσω από αυτόν τον αλγόριθμο είναι ότι κάθε συχνό στοιχειοσύνολο πρέπει να είναι συχνό σε έναν τουλάχιστον από τους διαχωρισμούς. Στη συνέχεια, εφαρμόζεται ο Apriori αλγόριθμος σε κάθε διαμέριση D_j και προκύπτουν τα αντίστοιχα συχνά στοιχειοσύνολα, έστω L_1, L_2, … L_p. Τα υποψήφια συχνά στοιχειοσύνολα είναι $C = L_1 \cup L_2 \cup ... \cup L_p$ Τέλος, γίνεται μια τελική καταμέτρηση της υποστήριξης των υποψήφιων συχνών στοιχειοσυνόλων C στη βάση D, για να βρεθούν τα τελικά συχνά στοιχειοσύνολα L.

Τα βασικά πλεονεκτήματα αυτού του αλγορίθμου είναι τα ακόλουθα:

- είναι προσαρμόσιμος στη διαθέσιμη κύρια μνήμη,
- υλοποιείται εύκολα σε παράλληλη μορφή, δηλαδή σχεδιασμός υλοποίησης για τον ταυτόχρονο υπολογισμό των υποψήφιων συχνών στοιχειοσυνόλων, και τέλος
- ο μέγιστος αριθμός σαρωμάτων ισούται με δύο.

Το μοναδικό μειονέκτημα του αλγορίθμου διαμέρισης είναι η πιθανή παραγωγή πολλών υποψήφιων στο δεύτερο σάρωμα.

7.7 Ο Αλγόριθμος FP-Growth

Ο αλγόριθμος FP-Growth είναι ένας διαφορετικός τρόπος εύρεσης των συχνών στοιχειοσυνόλων. Ο αλγόριθμος δεν χρησιμοποιεί παραγωγή υποψήφιων στοιχειοσυνόλων, βελτιώνοντας αρκετά την απόδοσή του. Ο αλγόριθμος βασίζεται σε μια ειδική δενδρική δομή, γνωστή ως *frequent-pattern tree* (FP-tree).

Με απλά λόγια, ο αλγόριθμος λειτουργεί ως εξής: αρχικά, σαρώνει τη βάση δεδομένων για πρώτη φορά, βρίσκει τα συχνά στοιχεία (στοιχειοσύνολα μήκους ένα) και δημιουργεί μια λίστα, ταξινομημένη κατά φθίνουσα σειρά υποστήριξης. Στη συνέχεια, με βάση αυτή τη λίστα ταξινομεί τα στοιχεία κάθε δοσοληψίας, ενώ τα μη συχνά (στοιχεία) τα διαγράφει. Τέλος, κάνει ένα δεύτερο σάρωμα και δημιουργεί τη δενδρική δομή, FP-tree, τοποθετώντας κάθε ταξινομημένη (κατά συχνότητα στοιχείων) δοσοληψία στη δομή αυτή.

Ας δούμε ένα παράδειγμα εύρεσης συχνών στοιχειοσυνόλων με τον αλγόριθμο FP-Growth. Έστω ότι έχουμε την παρακάτω βάση δεδομένων (Πίνακας 7.4) και κατώφλι υποστήριξης ίσο με 3.

TID	Items
1	FACDGLH
2	ABCFLHN
3	BFHJM
4	ACEFMN
5	BCKLI

Πίνακας 7.*4* *Η βάση δεδομένων δοσοληψιών για το παράδειγμα του αλγορίθμου FP-Growth.*

Στο πρώτο πέρασμα γίνεται καταμέτρηση της υποστήριξης κάθε στοιχείου και δημιουργείται η λίστα με τα ταξινομημένα στοιχεία κατά φθίνουσα σειρά υποστήριξης (Πίνακας 7.5).

Items	Support
F	4
C	4
A	3
B	3
H	3
L	3

Πίνακας 7.5 *Λίστα με ταξινομημένα στοιχεία σε φθίνουσα σειρά ως προς την υποστήριξή τους.*

Στη συνέχεια, γίνεται η ταξινόμηση των δοσοληψιών με βάση τη λίστα, όπως φαίνεται στον Πίνακα 7.5, και διαγράφονται τα μη συχνά (Πίνακας 7.6).

TID	Items	(Ordered) Frequent Items
1	FACDGLH	FCAHL
2	ABCFLHN	FCABH
3	BFHJM	FBH
4	ACEFMN	FCA
5	BCKLI	CBL

Πίνακας 7.6 *Οι ταξινομημένες δοσοληψίες.*

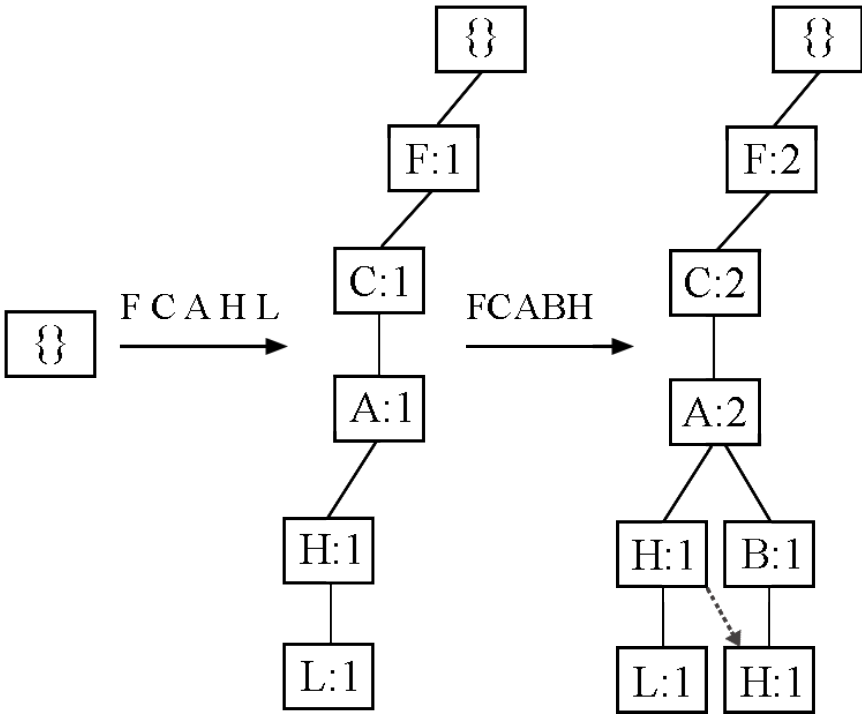
Τέλος, γίνεται το δεύτερο σάρωμα και δημιουργείται η δενδρική δομή, FP-tree, τοποθετώντας κάθε ταξινομημένη (κατά συχνότητα στοιχείων) δοσοληψία στη δομή αυτή, όπως φαίνεται παρακάτω (Εικόνα 7.5 και Εικόνα 7.6). Τα διακεκομμένα βέλη μεταξύ όμοιων στοιχείων αναπαριστούν δείκτες, οι οποίοι είναι χρήσιμοι κατά τη φάση της καταμέτρησης της υποστήριξης των στοιχειοσυνόλων.

Αρχικά, η δενδρική δομή έχει έναν κενό κόμβο, ο οποίος είναι η ρίζα του δένδρου και συμβολίζεται ως "{}" στο παράδειγμά μας (Εικόνα 7.5 και Εικόνα 7.6). Έπειτα, σαρώνονται με τη σειρά οι ταξινομημένες δοσοληψίες (Πίνακας 7.6, τελευταία στήλη) και προστίθενται νέοι κόμβοι, όποτε αυτό είναι απαραίτητο. Οι κόμβοι περιέχουν τα στοιχεία των δοσοληψιών και τον τρέχοντα αριθμό εμφάνισής τους στις δοσοληψίες, οι οποίες έχουν ήδη προσπελαστεί.

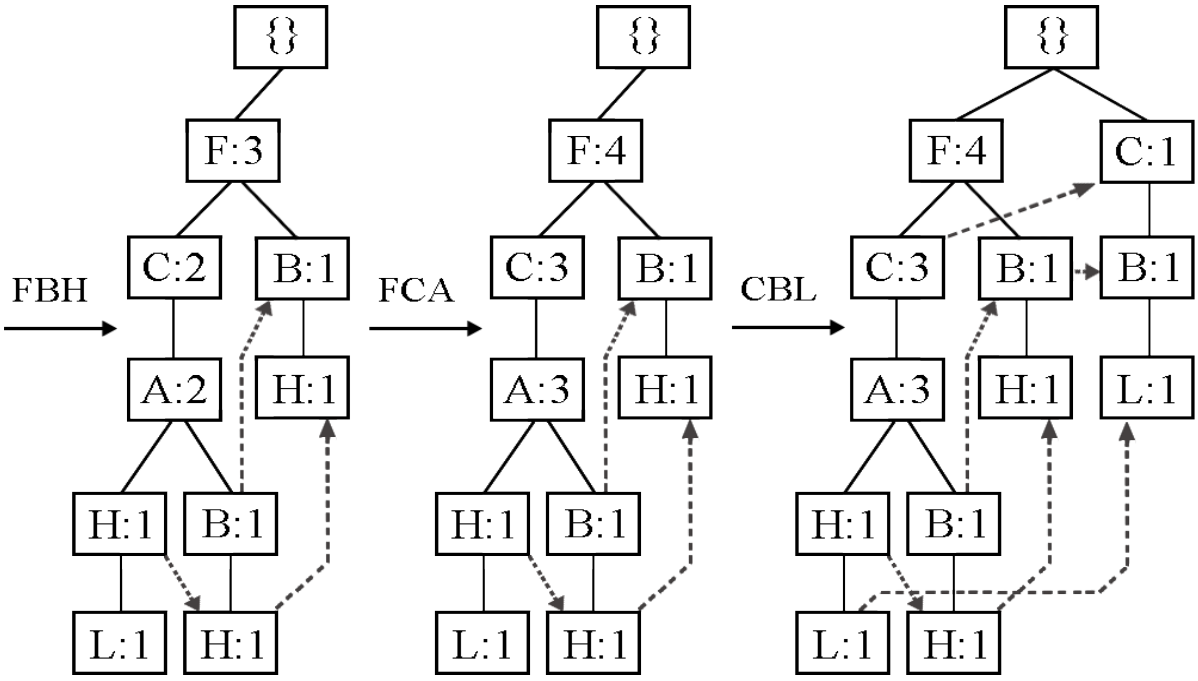
Αν υπάρχει κοινό πρόθεμα μεταξύ δύο ταξινομημένων δοσοληψιών, π.χ. FCAHL, FCABH, τότε δεν εισάγονται νέοι κόμβοι για το κοινό πρόθεμα, αλλά αυξάνεται ο μετρητής κάθε κόμβου, που συμμετέχει στο κοινό πρόθεμα κατά ένα (Εικόνα 7.5). Στην περίπτωση που μια δοσοληψία είναι υποσύνολο κάποιας προηγούμενης δοσοληψίας, τότε δεν προστίθεται κανένας νέος κόμβος, όπως, για παράδειγμα, συμβαίνει με την τελευταία δοσοληψία του παραδείγματός μας (FCA $\subset$ FCABH).

Αφού δημιουργηθεί η δενδρική δομή, FP-Tree, η εύρεση των συχνών στοιχειοσυνόλων γίνεται με προσπέλαση του δέντρου από κάτω προς τα πάνω, δηλαδή από τα φύλλα προς τη ρίζα. Για την εύρεση των συχνών στοιχειοσυνόλων εντοπίζονται τα προθεματικά υποδέντρα που καταλήγουν σε κάθε απλό στοιχειοσύνολο (στοιχείο), χρησιμοποιώντας τους δείκτες μεταξύ των ίδιων στοιχείων που υπάρχουν στη δενδρική δομή. Κάθε υποδέντρο προσπελαύνεται αναδρομικά για την εξαγωγή όλων των συχνών στοιχειοσυνόλων.

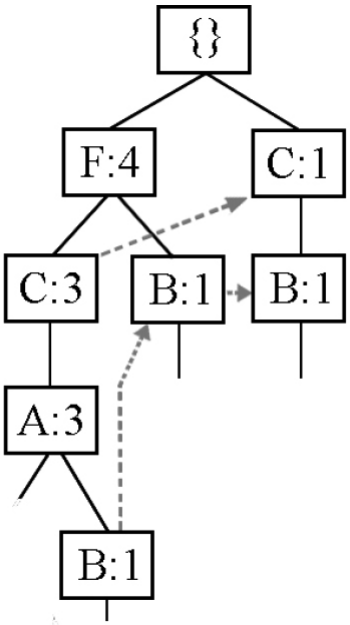
Για παράδειγμα, έστω ότι έχουμε το προθεματικό υποδέντρο για το στοιχείο B (Εικόνα 7.7). Η προσπέλαση θα γίνει αναδρομικά, αναζητώντας στοιχειοσύνολα που τελειώνουν σε B, έπειτα σε FB, CB, AB, έπειτα σε FCB, FAB, CAB κοκ. Παράλληλα, γίνεται η καταμέτρηση υποστήριξης για κάθε στοιχειοσύνολο. Στο τέλος της εκτέλεσης του αλγορίθμου γίνεται συνένωση των συνόλων για κάθε προθεματικό υποδέντρο και προκύπτει το σύνολο όλων των συχνών στοιχειοσυνόλων, μαζί με τις αντίστοιχες υποστηρίξεις τους.



Εικόνα *7.5 Δημιουργία δενδρικής δομής FP-tree.*



Εικόνα *7.6 Δημιουργία δενδρικής δομής FP-tree (συνέχεια).*



Εικόνα *7.7 Προθεματικό υποδέντρο για την εύρεση συχνών στοιχειοσυνόλων, που τελειώνουν με το στοιχείο B.*

7.8 Το πακέτο `arules`

Το πακέτο `arules` παρέχει στους χρήστες της R έτοιμες συναρτήσεις για την εξόρυξη συχνών στοιχειοσυνόλων και κανόνων συσχέτισης. Η βασικότερη συνάρτηση του πακέτου είναι η `apriori`, η οποία δέχεται 4 ορίσματα:

- data, τα δεδομένα από τα οποία θέλουμε να εξάγουμε τα στοιχειοσύνολα ή τους κανόνες συσχέτισης,
- parameter, μια λίστα παραμέτρων, όπως το κατώφλι υποστήριξης και εμπιστοσύνης (προεπιλεγμένες τιμές 0.1 και 0.8 αντίστοιχα),
- appearance, μια λίστα παραμέτρων, που καθορίζει περιορισμούς πάνω σε στοιχεία ή και μέλη των κανόνων,
- control, μια λίστα παραμέτρων, που αφορά περιορισμούς της επίδοσης του αλγορίθμου.

Το πρώτο όρισμα, data, μπορεί να είναι δομή οποιασδήποτε κλάσης (π.χ. λίστα, μητρώο) της R, αρκεί να μπορεί να μετατραπεί σε κλάση δοσοληψίας (transaction). Η κλάση δοσοληψίας (transaction) ορίζεται στο πακέτο `arules`. Σε κάθε περίπτωση, η συνάρτηση ελέγχει τον τύπο και κάνει την μετατροπή σε κλάση δοσοληψίας, όποτε αυτό είναι απαραίτητο.

Το δεύτερο όρισμα, parameter, είναι μια λίστα παραμέτρων. Σε αυτή τη λίστα μπορούμε να καθορίσουμε το κατώφλι υποστήριξης (supp) και το κατώφλι εμπιστοσύνης (conf) για την εύρεση των συχνών στοιχειοσυνόλων και την εξόρυξη κανόνων συσχέτισης. Επιπλέον, μπορούμε να καθορίσουμε τι θα επιστρέψει η συνάρτηση ως έξοδο (παράμετρος target). Μπορούμε να θέσουμε περιορισμούς ακόμα και για το ελάχιστο και μέγιστο μήκος των στοιχειοσυνόλων μέσω των minlen και maxlen, αντίστοιχα.

Ο Κώδικας 7.3 υλοποιεί την εξόρυξη των συχνών στοιχειοσυνόλων για τη βάση δεδομένων του Πίνακα 7.2. Χρησιμοποιήσαμε κατώφλι υποστήριξης ίσο με 3/6 = 0.5, ισοδύναμα με το παράδειγμα της υποενότητας 7.2.

```
> library(arules)
>
> db <- list(
```

```
+     c("A","B","D","E"),
+     c("B","C","E"),
+     c("A","B","D","E"),
+     c("A","B","C","E"),
+     c("A","B","C","D","E"),
+     c("B","C","D")
+     )
>
> frequent <- apriori(db, parameter=list(supp=0.5, conf=1,
target="frequent itemsets")
```

Κώδικας ***7.3*** *Εξόρυξη συχνών στοιχειοσυνόλων με χρήση της συνάρτησης apriori του πακέτου arules.*

Χρησιμοποιώντας τη συνάρτηση inspect, μπορούμε να δούμε αναλυτικά τα συχνά στοιχειοσύνολα και την υποστήριξη του καθενός (Κώδικας 7.4). Αυτό επαληθεύει ότι σωστά βρήκαμε 19 συχνά στοιχειοσύνολα στο προηγούμενό μας παράδειγμα.

```
> inspect(frequent)
   items     support
1  {C}       0.6666667
2  {D}       0.6666667
3  {A}       0.6666667
4  {E}       0.8333333
5  {B}       1.0000000
6  {C,E}     0.5000000
7  {B,C}     0.6666667
8  {A,D}     0.5000000
9  {D,E}     0.5000000
10 {B,D}     0.6666667
11 {A,E}     0.6666667
12 {A,B}     0.6666667
13 {B,E}     0.8333333
14 {B,C,E}   0.5000000
15 {A,D,E}   0.5000000
16 {A,B,D}   0.5000000
17 {B,D,E}   0.5000000
18 {A,B,E}   0.6666667
19 {A,B,D,E} 0.5000000
```

Κώδικας ***7.4*** *Επισκόπηση των συχνών στοιχειοσυνόλων.*

Αν θέλαμε μόνο τα κλειστά συχνά στοιχειοσύνολα, τότε ως `target` θα έπρεπε να δώσουμε "`closed`" (Κώδικας 7.5). Ο Κώδικας 7.5 επαληθεύει ότι στην Εικόνα 7.2 φαίνονται σωστά τα κλειστά συχνά στοιχειοσύνολα με μπλε πλαίσιο.

```
> cl<-apriori(db,parameter=list(supp=0.5, conf=1, target="-
closed"))
> inspect(cl)
  items     support
1 {B}       1.0000000
2 {B,C}     0.6666667
3 {B,D}     0.6666667
4 {B,E}     0.8333333
5 {B,C,E}   0.5000000
6 {A,B,E}   0.6666667
7 {A,B,D,E} 0.5000000
```

Κώδικας ***7.5*** *Εξόρυξη κλειστών συχνών στοιχειοσυνόλων.*

Αντίστοιχα, αν θέλαμε μόνο τα μέγιστα συχνά στοιχειοσύνολα, τότε ως `target` θα έπρεπε να δώσουμε "`maximal`" (Κώδικας 7.6). Ο Κώδικας 7.6 επαληθεύει ότι στην Εικόνα 7.3 φαίνονται σωστά τα μέγιστα συχνά στοιχειοσύνολα με πράσινο πλαίσιο.

```
> mx<-apriori(db,parameter=list(supp=0.5, conf=1, target="max-
imal"))
> inspect(mx)
  items     support
1 {B,C,E}   0.5
2 {A,B,D,E} 0.5
```

Κώδικας ***7.6*** *Εξόρυξη μέγιστων συχνών στοιχειοσυνόλων.*

Αν θέλαμε τους κανόνες συσχέτισης, τότε ως `target` θα έπρεπε να δώσουμε "`rules`" (Κώδικας 7.7). Το `lhs` συμβολίζει το αριστερό μέλος, ενώ το `rhs` το δεξιό μέλος του κανόνα συσχέτισης.

```
> rules<-apriori(db, parameter=list(supp=0.5, conf=1, tar-
get="rules"))
> inspect(rules)
   lhs       rhs support   confidence lift
1  {}     => {B} 1.0000000 1          1.0
2  {C}    => {B} 0.6666667 1          1.0
3  {D}    => {B} 0.6666667 1          1.0
4  {A}    => {E} 0.6666667 1          1.2
5  {A}    => {B} 0.6666667 1          1.0
6  {E}    => {B} 0.8333333 1          1.0
7  {C,E}  => {B} 0.5000000 1          1.0
```

```
8  {A,D}   => {E} 0.5000000 1          1.2
9  {D,E}   => {A} 0.5000000 1          1.5
10 {A,D}   => {B} 0.5000000 1          1.0
11 {D,E}   => {B} 0.5000000 1          1.0
12 {A,E}   => {B} 0.6666667 1          1.0
13 {A,B}   => {E} 0.6666667 1          1.2
14 {A,D,E} => {B} 0.5000000 1          1.0
15 {A,B,D} => {E} 0.5000000 1          1.2
16 {B,D,E} => {A} 0.5000000 1          1.5
```

Κώδικας *7.7 Εξόρυξη κανόνων συσχέτισης.*

Το τρίτο όρισμα, appearance, είναι και αυτό μια λίστα παραμέτρων. Μέσω αυτού του ορίσματος μπορούμε να καθορίσουμε ποια στοιχεία επιτρέπονται στους κανόνες, φιλτράροντας έτσι τους κανόνες που θα επιστρέψει η συνάρτηση. Αυτή η λίστα μπορεί να περιέχει τις παραμέτρους `lhs`, `rhs`, `both`, `items` ή `none`. Στις παραμέτρους πρέπει να δοθεί ως τιμή ένα διάνυσμα χαρακτήρων, το οποίο καθορίζει ποια στοιχεία μπορούν να εμφανιστούν. Πιο συγκεκριμένα, τα `rhs`, `lhs` και `both` χρησιμοποιούνται κατά την εξόρυξη κανόνων συσχέτισης, ενώ τα `items` και `none` κατά την εύρεση συχνών στοιχειοσυνόλων.

Μια ακόμα παράμετρος που πρέπει να τεθεί σε συνδυασμό με τις προηγούμενες είναι η default, η οποία παίρνει τιμές `rhs`, `lhs`, `both` ή `none`. Αυτή καθορίζει τη συμπεριφορά όλων των υπόλοιπων μελών του κανόνα, για τα οποία δεν έχουν οριστεί ρητά περιορισμοί.

Για παράδειγμα, για το σύνολο δεδομένων Adult και με κατώφλι ίσο με 3/4, η συνάρτηση `apriori` (Κώδικας 7.8) μας επιστρέφει 19 κανόνες συσχέτισης (Εικόνα 7.8).

```
> library(arules)
> data(Adult)
> inspect(apriori(Adult, parameter=list(supp=0.75)))
```

Κώδικας *7.8 Εξόρυξη κανόνων συσχέτισης από το σύνολο δεδομένων Adult.*

Με τις παραμέτρους που αναφέραμε, μπορούμε να θέσουμε περιορισμούς, π.χ. για το δεξί μέλος του κανόνα, επιτρέποντας μόνο το στοιχείο “capital-gain=None” (Κώδικας 7.9). Οπότε, τώρα επιστρέφονται μόνο 5 από τους 19 κανόνες συσχέτισης, όπως φαίνεται στην Εικόνα 7.9.

```
> inspect(apriori(Adult, parameter=list(supp=0.75),
+ appearance=list(rhs=”capital-gain=None”, default=”lhs”)))
```

Κώδικας *7.9 Εξόρυξη κανόνων συσχέτισης από το σύνολο δεδομένων Adult με περιορισμούς για το δεξί μέλος.*

```
    lhs                                                    rhs
1   {}                                                  => {race=White}
2   {}                                                  => {native-country=United-States}
3   {}                                                  => {capital-gain=None}
4   {}                                                  => {capital-loss=None}
5   {race=White}                                        => {native-country=United-States}
6   {native-country=United-States}                      => {race=White}
7   {race=White}                                        => {capital-gain=None}
8   {capital-gain=None}                                 => {race=White}
9   {race=White}                                        => {capital-loss=None}
10  {capital-loss=None}                                 => {race=White}
11  {native-country=United-States}                      => {capital-gain=None}
12  {capital-gain=None}                                 => {native-country=United-States}
13  {native-country=United-States}                      => {capital-loss=None}
14  {capital-loss=None}                                 => {native-country=United-States}
15  {capital-gain=None}                                 => {capital-loss=None}
16  {capital-loss=None}                                 => {capital-gain=None}
17  {capital-gain=None,native-country=United-States}    => {capital-loss=None}
18  {capital-loss=None,native-country=United-States}    => {capital-gain=None}
19  {capital-gain=None,capital-loss=None}               => {native-country=United-States}
```

Εικόνα ***7.8*** *Κανόνες συσχέτισης από τον Κώδικα 7.8.*

```
  lhs                                                  rhs
1 {}                                                => {capital-gain=None}
2 {race=White}                                      => {capital-gain=None}
3 {native-country=United-States}                    => {capital-gain=None}
4 {capital-loss=None}                               => {capital-gain=None}
5 {capital-loss=None,native-country=United-States} => {capital-gain=None}
```

Εικόνα ***7.9*** *Κανόνες συσχέτισης από τον Κώδικα 7.9.*

Κριτήρια αξιολόγησης

Κριτήριο αξιολόγησης 1

Θεωρήστε το ακόλουθο σύνολο των συχνών 3-στοιχειοσυνόλων: {1,2,3}, {1,2,4}, {1,2,5}, {1,3,4}, {1,3,5}, {2,3,4}, {2,3,5}, {3,4,5} μιας βάσης δεδομένων, που περιέχει τα στοιχεία 1, 2, 3, 4 και 5. Βρείτε ποια από τα παρακάτω υποψήφια 4-στοιχειοσύνολα επιβιώνουν από το βήμα περικοπής (κλαδέματος) υποψηφίων του Apriori αλγόριθμου.

(a) **{1,2,3,4} και {1,2,4,5}**
(b) **{1,2,3,4} μόνο**
(c) **{1,2,3,4} και {1,2,3,5}**
(d) κανένα

Κριτήριο αξιολόγησης 2

Θεωρήστε τη δυαδική βάση δεδομένων στον παρακάτω πίνακα. Ποιο από τα παρακάτω είναι το σύνολο των κλειστών συχνών στοιχειοσυνόλων, που παράγονται από την εξόρυξη συχνών στοιχειοσυνόλων της παραπάνω βάσης, με κατώφλι του μετρητή υποστήριξης ίσο με 3;

Tid	Items
1	ABDE
2	BCE
3	ABDE
4	ABCE
5	ABCDE
6	BCD

Πίνακας *7.7 Πίνακας κριτηρίου αξιολόγησης 2.*

(a) **{B, BC, BD, BE, ABE, BCE}**
(b) **{BC, BD, BE, ABE, BCE, ABDE}**
(c) **{B, BC, BD, BE, ABE, ABDE}**
(d) **{B, BC, BD, BE, ABE, BCE, ABDE}**

Κριτήριο αξιολόγησης 3

Θεωρήστε τη δυαδική βάση δεδομένων στον παρακάτω πίνακα. Ποιο από τα παρακάτω είναι το σύνολο των μέγιστων συχνών στοιχειοσυνόλων, που παράγονται από την εξόρυξη της παραπάνω βάσης, με κατώφλι του μετρητή υποστήριξης ίσο με 3;

Tid	Items
1	ABDE
2	BCE
3	ABDE
4	ABCE
5	ABCDE
6	BCD

Πίνακας ***7.8*** *Πίνακας κριτηρίου αξιολόγησης 3.*

(a) **{ABE, BCE, ABDE}**
(b) **{BCE, ABDE}**
(c) **{B, BC, BD, BE, ABE, ABDE}**
(d) **{B, BC, BD, BE, ABE, BCE, ABDE}**

Κριτήριο αξιολόγησης 4

Εάν το σύνολο των κλειστών συχνών στοιχειοσυνόλων με τους αντίστοιχους μετρητές υποστήριξης είναι το {B:5, C:5, D:3, AC:4, BC:4, BD:2, BE:4, ABC:3, ACD:2, BCE:3, ABCE:2}, βρείτε τον μετρητή υποστήριξης του στοιχειοσυνόλου AB.

(a) **1**
(b) **2**
(c) **3**
(d) **4**

Κριτήριο αξιολόγησης 5

Η εμπιστοσύνη του κανόνα συσχέτισης {X}→{Y,Z} βρέθηκε ότι είναι 0.85. Τι σημαίνει η τιμή 0.85;

(a) **85% των δοσοληψιών περιέχουν τα X, Y και Z.**
(b) **85% των δοσοληψιών περιέχουν τουλάχιστον ένα από τα X, Y και Z.**
(c) **85% των δοσοληψιών που περιέχουν το X, περιέχουν,** επίσης, και τα Y και Z.
(d) **85% των δοσοληψιών που περιέχουν τα Y και Z, περιέχουν,** επίσης, και το X.

Κριτήριο αξιολόγησης 6

Θεωρήστε το ακόλουθο σύνολο των συχνών 3-στοιχειοσυνόλων: {1,2,3}, {1,2,4}, {1,2,5}, {1,3,4}, {1,3,5}, {2,3,4}, {2,3,5}, {3,4,5} μιας βάσης δεδομένων, που περιέχει τα στοιχεία 1, 2, 3, 4 και 5. Βρείτε ποια από τα παρακάτω υποψήφια 4-στοιχειοσύνολα επιβιώνουν από το βήμα περικοπής (κλαδέματος) υποψηφίων του Apriori αλγόριθμου.

(a) **{1,2,3,4} και {1,2,3,5}**
(b) **{1,2,3,4} μόνο**
(c) **{1,2,3,4} και {1,2,4,5}**
(d) κανένα

Κριτήριο αξιολόγησης 7

Θεωρήστε τη δυαδική βάση δεδομένων στον παρακάτω πίνακα. Ποιο από τα παρακάτω είναι το σύνολο των κλειστών συχνών στοιχειοσυνόλων που παράγονται από την εξόρυξη συχνών στοιχειοσυνόλων της παραπάνω βάσης με κατώφλι του μετρητή υποστήριξης ίσο με 3;

Tid	Items
1	ABDE
2	BCE
3	ABDE
4	ABCE
5	ABCDE
6	BCD

Πίνακας 7.*9* *Πίνακας κριτηρίου αξιολόγησης 7.*

(a) **{B, BC, BD, BE, ABE, BCE}**
(b) **{BC, BD, BE, ABE, BCE, ABDE}**
(c) **{B, BC, BD, BE, ABE, ABDE}**
(d) **{B, BC, BD, BE, ABE, BCE, ABDE}**

Κριτήριο αξιολόγησης 8

Θεωρήστε τη βάση δεδομένων στον παρακάτω πίνακα. Ποιο από τα παρακάτω είναι το σύνολο των μέγιστων συχνών στοιχειοσυνόλων που παράγονται από την εξόρυξη της παραπάνω βάσης με κατώφλι του μετρητή υποστήριξης ίσο με 3;

Tid	Items
1	ABDE
2	BCE
3	ABDE
4	ABCE
5	ABCDE
6	BCD

Πίνακας 7.*10* *Πίνακας κριτηρίου αξιολόγησης 8.*

(a) **{ABE, BCE, ABDE}**
(b) **{B, BC, BD, BE, ABE, BCE, ABDE}**
(c) **{B, BC, BD, BE, ABE, ABDE}**
(d) **{BCE, ABDE}**

Κριτήριο αξιολόγησης 9

Θεωρήστε τη βάση δεδομένων στον Πίνακα 7.10. Ποιο από τα παρακάτω είναι το αρνητικό περιθώριο των συχνών στοιχειοσυνόλων, δεδομένου ότι η εξόρυξη των συχνών στοιχειοσυνόλων έγινε με κατώφλι του μετρητή υποστήριξης ίσο με 3;

(a) **{AC, CD}**
(e) **{AC, ABC, ACE, BCD}**
(f) **{ABC, BCD, ACD, ACE}**
(g) **{BCE, ABDE}**

Κριτήριο αξιολόγησης 10

Έστω ότι για μια βάση συναλλαγών, το σύνολο των κλειστών συχνών στοιχειοσυνόλων με τους αντίστοιχους μετρητές υποστήριξης είναι το {B:5, C:5, D:3, AC:4, BC:4, BD:2, BE:4, ABC:3, ACD:2, BCE:3, ABCE:2}. Ποιος είναι ο μετρητή υποστήριξης του στοιχειοσυνόλου AB;
(*Υπόδειξη: διαβάστε το Κριτήριο Αξιολόγησης 1*1* και την επίλυσή του*).

(a) **1**
(c) **2**
(d) **3**
(e) **4**

Κριτήριο αξιολόγησης 11

Τα μέγιστα συχνά στοιχειοσύνολα είναι ικανά για να καθορίσουν όλα τα συχνά στοιχειοσύνολα και τις υποστηρίξεις αυτών;

(a) Σωστό
(b) Λάθος

Απαντήσεις

Οι σωστές απαντήσεις των παραπάνω ερωτημάτων είναι: 1c, 2d, 3b, 4c, 5c, 6a, 7d, 8d, 9a, 10c, 11b.

Κριτήριο αξιολόγησης 12

Αποδείξτε με τρόπο απλό (με τρεις-τέσσερεις προτάσεις το πολύ!) ότι η συλλογή όλων των μέγιστων συχνών στοιχειοσυνόλων M, αποτελεί υποσύνολο της συλλογής των κλειστών συχνών στοιχειοσυνόλων C, δηλαδή ότι M⊆C.

Απάντηση

Έστω ότι το στοιχειοσύνολο X ανήκει στο M. Το γεγονός ότι το X ανήκει στο M συνεπάγεται ότι το X δεν έχει συχνά υπερσύνολα. Το ότι το X δεν έχει συχνά υπερσύνολα σημαίνει ότι δεν έχει συχνό υπερσύνολο με την ίδια υποστήριξη με αυτό. Αυτό σημαίνει ότι το X είναι κλειστό ή αλλιώς ότι X ανήκει στο C. Συνεπώς, ισχύει ότι M⊆C.

Βιβλιογραφία

Wikipedia (2015). *Association Rule Learning.* Ανακτήθηκε στις 27 Νοεμβρίου 2015, από: https://en.wikipedia.org/wiki/Association_rule_learning

Dunham, M. H. (2003). Data Mining: Introductory and Advanced Topics. *Pearson Education*, Upper Saddle River, N. J. Prentice Hall.

Zhao, Q. & Bhowmick, S. S. (2003). *Association Rule Mining: A survey* (Technical report: No 2003116). Singapore: CAIS, Nanyang Technological University

Vercellis, C. (2009). *Business Intelligence: Data Mining and Optimization for Decision Making*. John Wiley & Sons Ltd.

Κεφάλαιο 8: Υπολογιστικές Μέθοδοι για Ανάλυση Μεγάλων Δεδομένων (Hadoop και MapReduce)

Σύνοψη

Η λύση του Hadoop και MapReduce και των συνεργαζόμενων εφαρμογών τους λαμβάνει μεταξύ άλλων μία απλή, αλλά πολύ ισχυρή, μέθοδο για την επεξεργασία και την ανάλυση των εξαιρετικά μεγάλων συνόλων δεδομένων, ακόμα και μέχρι το επίπεδο πολλαπλών Petabyte. Κατά βάση, το MapReduce είναι μία διαδικασία συνδυασμού των δεδομένων από πολλαπλές εισόδους (δημιουργώντας έτσι το "map") και αναγωγής (reduce), όπου εκεί χρησιμοποιείται μία λειτουργία που θα διυλίσει τα επιθυμητά αποτελέσματα. Στο κεφάλαιο αυτό παρουσιάζονται πολλαπλές περιπτώσεις χρήσεις των Hadoop & MapReduce και των εφαρμογών τους για περιπτώσεις πολλών TB ακόμα και αρκετών PB. Το Hadoop & MapReduce χρησιμοποιεί ένα κατανεμημένο σύστημα αρχείων, το HDFS. Το σύστημα του Hadoop & MapReduce είναι χρήσιμο για δεδομένα, τα οποία είναι λιγότερο δομημένα, όπως, για παράδειγμα, οι σελίδες Διαδικτύου ή τα πολλαπλά δεδομένα εγγράφων ή εικόνων, τα οποία δεν είναι πλήρως οργανωμένα/δομημένα. Το Κεφάλαιο αυτό έχει ως σκοπό την εισαγωγική παρουσίαση στο πλαίσιο λογισμικού Hadoop, καθώς και την παρουσίαση της προγραμματιστικής διεπαφής του (Hadoop Application Programming Interface-API) σε Java.

Προαπαιτούμενη γνώση

Για τη μελέτη αυτού του κεφαλαίου απαιτείται πολύ καλή γνώση της γλώσσας προγραμματισμού Java και κάποια σχετική εξοικείωση με την εκτέλεση κατανεμημένων τύπων προγραμμάτων.

Υπολογιστικές Μέθοδοι για Ανάλυση Μεγάλων Δεδομένων

8.1 Εισαγωγή

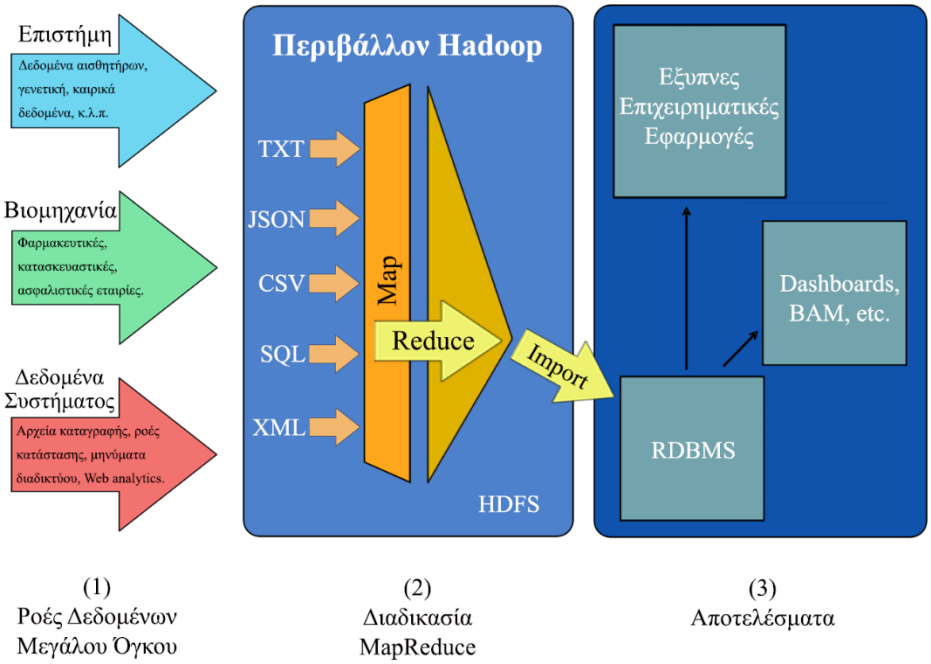
Το Hadoop δεν είναι μία ακόμη βάση δεδομένων. Είναι λογισμικό υποδομής ή, όπως θα μπορούσε να πει κανείς, είναι σχεδόν ένα λειτουργικό σύστημα. Είναι ένα πλαίσιο λογισμικού γραμμένο σε Java για να "τρέχει" εφαρμογές σε μεγάλα clusters τυπικών υπολογιστών και να ενσωματώνει χαρακτηριστικά παρόμοια με αυτά του Google File System και του MapReduce. Δημιουργήθηκε για να διαχειριστεί μεγάλες ποσότητες δεδομένων τα οποία, λόγω του μεγάλου όγκου τους, δεν είναι δυνατόν να τοποθετηθούν στον δίσκο ενός υπολογιστή και έτσι τόσο τα δεδομένα όσο και η ανάλυσή τους χρειάζεται να κατανεμηθούν σε μεγάλες συστάδες υπολογιστών.

Η χρήση του Hadoop περιλαμβάνει τα ακόλουθα στάδια: (α) εγκατάσταση του Hadoop σε μεγάλο αριθμό υπολογιστών και χρήση των δίσκων τους για αποθήκευση δεδομένων, και (β) χρησιμοποίηση των CPU των υπολογιστών για την επεξεργασία των δεδομένων. Η αρχιτεκτονική μιας εφαρμογής Hadoop είναι μια αρχιτεκτονική share-nothing, με τυπικούς (συνήθεις) υπολογιστές. Το Hadoop δημιουργήθηκε από τον Doug Cutting, τον δημιουργό της Apache Lucene (διαδικτυακή βιβλιοθήκη αναζήτησης κειμένων).

Στην προσπάθειά του να φτιάξει μια μηχανή αναζήτησης στο διαδίκτυο, ανοιχτού κώδικα (Apache Nutch), είχε πρόβλημα στη διαχείριση υπολογισμών που έτρεχαν σε μικρό αριθμό υπολογιστών. Στη λύση αυτού του προβλήματος βοήθησε η δημοσίευση του Google File System και MapReduce. Με τη βοήθεια της Yahoo ξεχώρισε τη διαδικασία της εφαρμογής Nutch από τη κατανομή των υπολογισμών και την ονόμασε Hadoop. Έτσι, κατάφερε να υλοποιήσει την τεχνολογία που έτρεχε πλέον στο διαδίκτυο. Στη συνέχεια, χρησιμοποιήθηκε στις υπηρεσίες Amazon Elastic Compute Cloud (EC2) και Amazon Simple Storage Service (S3). Σήμερα το Hadoop είναι μια συλλογή σχετικών υποέργων, που αφορούν στην υποδομή των κατανεμημένων συστημάτων πληροφορικής. Αυτά τα έργα φιλοξενούνται από την Apache Software Foundation, η οποία παρέχει υποστήριξη για έργα ανοιχτού κώδικα. Το Hadoop είναι, επίσης, γνωστό και από το MapReduce και από το κατανεμημένο σύστημα αρχείων του Hadoop (Distributed File System, HDFS).

Το όνομα Hadoop δεν είναι ένα ακρωνύμιο. Είναι ένα κατασκευασμένο όνομα. Ο Doug Cutting εξηγεί: «Είναι το όνομα που έδωσε ο γιος μου σε ένα κίτρινο ελεφαντάκι. Είναι μικρό, σχετικά εύκολο να το προφέρεις και δεν χρησιμοποιείται πουθενά. Αυτά ήταν τα κριτήριά μου για την επιλογή του ονόματος. Τα παιδιά τα

καταφέρνουν σε αυτά. Π.χ. Googol είναι ένας παιδικός όρος...» Επίσης, διάφορα υποέργα και τμήματα του Hadoop έχουν ονόματα άσχετα με τη λειτουργία τους, συχνά με έναν ελέφαντα ή κάποιο άλλο ζώο (pig, για παράδειγμα). Μικρότερα συστατικά έχουν πιο περιγραφικά ονόματα για τη λειτουργία τους. Αυτή είναι μια καλή αρχή για την κατανόηση της λειτουργίας του. Για παράδειγμα, ο Jobtracker (ανιχνευτή εργασίας) για τις MapReduce εργασίες.



Εικόνα *8.1* *Γενική αρχιτεκτονική χρήσης του Hadoop με επιχειρηματικά δεδομένα.*

Γενικά, οι σχεσιακές βάσεις δεδομένων έχουν αποδειχθεί ότι είναι εξαιρετικά ευέλικτες και ότι συνιστούν το κατάλληλο εργαλείο για την πλειονότητα των αναγκών των επιχειρήσεων μέχρι σήμερα. Ωστόσο, παρουσιάζονται συνεχώς νέα δεδομένα, στα οποία η χρήση RDBMS δεν είναι συχνά η καλύτερη επιλογή.

Η εναλλακτική λύση του Hadoop & MapReduce, όπως παρουσιάζεται στην Εικόνα 8.1, και των συνεργαζόμενων εφαρμογών τους αποτελεί μια καλή εναλλακτική λύση. Περιλαμβάνει μεταξύ άλλων μια απλή, αλλά πολύ ισχυρή μέθοδο για την επεξεργασία και την ανάλυση των εξαιρετικά μεγάλων συνόλων δεδομένων, ακόμη και μέχρι το επίπεδο πολλαπλών Petabyte (Εικόνα 8.1, Σημείο 1 Ροές Δεδομένων Μεγάλου Όγκου).

Τέτοιας διάστασης δεδομένα παλαιότερα ήταν πολύ δύσκολο και ακριβό να επεξεργαστούν μέσα από ένα παραδοσιακό RDBMS. Για να αντιμετωπίσει τη διαχείριση τεράστιου όγκου δεδομένων, που δεν χωρούν συχνά ακόμη και σε δεκάδες μηχανές, το Hadoop χρησιμοποιεί ένα κατανεμημένο σύστημα αρχείων, το HDFS (Εικόνα 8.1, Σημείο 2 Διαδικασία MapReduce & HDFS).

Η συγκεκριμένη προσέγγιση είναι ιδιαίτερα χρήσιμη, όταν απαιτούνται λειτουργίες επεξεργασίας πολύ μεγάλου όγκου, στις οποίες τα αποτελέσματα δεν απαιτούνται σε πραγματικό χρόνο (Εικόνα 8.1, Σημείο 3 Αποτελέσματα). Επιπλέον, είναι χρήσιμο, όταν έχουμε να διαχειριστούμε δεδομένα που γράφονται, χωρίς μελλοντικές ενημερώσεις, ενώ παράλληλα απαιτείται να αναγνωστούν πολλαπλές φορές. Αντίστοιχα, το σύστημα του Hadoop/MapReduce είναι χρήσιμο για δεδομένα, τα οποία είναι λιγότερο δομημένα, όπως, για παράδειγμα, οι σελίδες Διαδικτύου ή πολλαπλά δεδομένα εγγράφων ή εικόνων, που δεν είναι πλήρως οργανωμένα-δομημένα.

8.2 Ιστορική Αναδρομή

Η χρονική εξέλιξη των Hadoop και MapReduce έχει ως εξής:

- 2002 – 2004: Έναρξη ανάπτυξής του στο πλαίσιο του Nutch, ενός web-crawler, open source project, με συμμετοχή πολλών προγραμματιστών, κατανεμημένο σύστημα, χρήση sort/merge επεξεργασίας.
- 2004- 2006: Δημοσίευση άρθρων σχετικά με Google File System και τεχνικής MapReduce,

με άμεση αναφορά στα προβλήματα του Nutch, ενσωμάτωση κατανεμημένου File System και MapReduce στο Nutch.
- 2006 -2008: Αποκοπή του Hadoop από το Nutch, χρήση από την Yahoo!. Σε cluster 1000 υπολογιστών, πλήρης εγκατάσταση από την Yahoo!.

8.3 Πλεονεκτήματα του Κατανεμημένου Συστήματος Αρχείων Hadoop

Το κατανεμημένο Σύστημα Αρχείων του Hadoop (HDFS) είναι ένα σύστημα σχεδιασμένο να τρέχει σε τυπικά PC. Το HDFS έχει πολλές ομοιότητες, αλλά και σημαντικές διαφορές με άλλα κατανεμημένα συστήματα αρχείων που ήδη υπάρχουν. Στη συνέχεια, παρατίθενται τα προβλήματα που επιτυχώς επιλύει το HDFS, πλεονεκτώντας με αυτόν τον τρόπο έναντι των ήδη υπαρχόντων κατανεμημένων συστημάτων αρχείων, καθώς και τα σημεία στα οποία το HDFS υπερτερεί.

1. Αποτυχία υλικού. Ένα στιγμιότυπο HDFS μπορεί να συνίσταται από εκατοντάδες ή χιλιάδες μηχανήματα, όπου το καθένα αποθηκεύει το δικό του μερίδιο δεδομένων. Είναι πιθανό να προκύψουν αποτυχίες στο hardware, χωρίς αυτό να σημαίνει ότι κάποια από τα συστατικά του HDFS δεν είναι λειτουργικά. Το HDFS συνεχώς ερευνά για λάθη και αυτόματα ανακτά τα δεδομένα από αυτά. Ο τρόπος αυτός λειτουργίας του HDFS αποτελεί ένα πλεονέκτημα αρχιτεκτονικής του πυρήνα του.
2. Συνεχής ροή πρόσβασης δεδομένων. Οι εφαρμογές που τρέχουν στο HDFS χρειάζονται συνεχή πρόσβαση στα δεδομένα τους. Το HDFS είναι σχεδιασμένο περισσότερο για batch processing παρά για διαδραστική χρήση από τους χρήστες. Το POSIX (Portable Operating System Interface for Unix) θέτει πολλές απαιτήσεις που δεν χρειάζονται στις εφαρμογές που εκτελούνται στο HDFS.
3. Μεγάλο σύνολο δεδομένων. Οι εφαρμογές που τρέχουν σε HDFS χειρίζονται δεδομένα πολύ μεγάλου όγκου. Ένα συνηθισμένο αρχείο στο HDFS έχει μέγεθος της τάξης Gigabyte έως και Terabyte. Το HDFS δίνει τη δυνατότητα (α) παροχής υπηρεσιών με υψηλό όγκο πληροφορίας, που μπορεί να μεταδοθεί μέσω διαδικτύου σε δεδομένο χρόνο, (bandwidth), και (β) ύπαρξης εκατοντάδων κόμβων σε ένα ενιαίο cluster. Το HDFS θα μπορούσε να υποστηρίξει δεκάδες εκατομμύρια αρχεία σε ένα ενιαίο στιγμιότυπο.
4. Απλό-συνεπές μοντέλο. Το HDFS χρησιμοποιεί για τα αρχεία το μοντέλο πρόσβασης «γράφω μία φορά, διαβάζω πολλές». Ένα αρχείο δημιουργείται, γράφεται και κλείνει, χωρίς να χρειάζεται αλλαγές. Έτσι, απλοποιείται η κατάσταση των δεδομένων και επιτυγχάνεται υψηλή παραγωγικότητα (throughput), με άλλα λόγια παράγεται μεγάλη ποσότητα έργου από έναν σταθμό εξυπηρέτησης, στη μονάδα του χρόνου.
5. Μετακίνηση των εφαρμογών κοντά στα δεδομένα. Όταν απαιτείται μία επεξεργασία από την εφαρμογή, η επεξεργασία αυτή είναι αποτελεσματικότερη, όταν εκτελείται κοντά στα δεδομένα που αυτή χρειάζεται, κυρίως, όταν το σύνολο των δεδομένων είναι μεγάλο. Κάτι τέτοιο ελαχιστοποιεί τη δικτυακή συμφόρηση και αυξάνει τη παραγωγικότητα του συστήματος. Συχνά είναι προτιμότερο να μετακινείται η επεξεργασία κοντά στα δεδομένα, παρά να μετακινούνται τα δεδομένα εκεί που τρέχει η εφαρμογή. Το HDFS παρέχει διεπαφές για τις εφαρμογές, ώστε να μπορούν οι τελευταίες να μετακινούνται κοντά στα δεδομένα.
6. Φορητότητα μεταξύ διαφορετικού υλικού και λογισμικού. Το HDFS σχεδιάστηκε για να είναι εύκολα φορητό από μία πλατφόρμα σε μία άλλη.

8.4 Χρήστες του Hadoop

Το Hadoop δεν χρησιμοποιείται μόνο για εκπαιδευτικούς και ερευνητικούς σκοπούς. Χρησιμοποιείται ευρέως για την αντιμετώπιση πραγματικών προβλημάτων, που συναντώνται σε διάφορους τομείς του πραγματικού κόσμου. Στη συνέχεια, αναφέρονται ορισμένες χαρακτηριστικές περιπτώσεις χρήσης του Hadoop, όπως έχουν καταγραφεί από τους ίδιους τους χρήστες του.

- **Amazon/A9** - Η New York Times χρησιμοποίησε cluster 100-servers με το Hadoop φιλοξενούμενο από την Amazon για να μετασχηματίσει 4TB παλιών φωτογραφιών σε 11.000.000 PDF αρχεία. Το έκανε σε 24 ώρες με συνολικό κόστος 240$.
- **Adobe** - Χρησιμοποιεί το Hadoop για τις υπηρεσίες social media και για επεξεργασία δεδομένων για εσωτερική χρήση από το 2008.
- **Facebook** - Με το Hadoop βελτίωσε την τοποθέτηση των διαφημίσεών του, μελέτησε τη συ-

μπεριφορά των χρηστών μέσω των δεδομένων, με αποτέλεσμα να συμβάλει στη λήψη σωστών αποφάσεων εμπορίας των προϊόντων. Περιλαμβάνει >1000 μηχανήματα με >10000 πυρήνες και >15PB δεδομένων προς επεξεργασία.

- Δίκτυο FOX - Κάνει χρήση του Hadoop για την ανάλυση καταγραφών πρόσβασης στους ιστοχώρους της και για εξόρυξη δεδομένων.
- **Microsoft - Powerset / Microsoft** – Συμβάλει στο HBase κυρίως με σκοπό τη βελτίωση της επεξεργασίας δεδομένων φυσικής γλώσσας με χρήση >400 κόμβων του Amazon EC2 και αποθηκευτικούς χώρους του Amazon S3.
- **LinkedIn** - Χρησιμοποιεί το Hadoop, για να εντοπίζει μέσα στο κοινωνικό δίκτυο της νέες διασυνδέσεις των μελών της με χρήση >4000 μηχανών.
- EBay - Κάνει ευρεία χρήση, με σκοπό τη βελτίωση της υπηρεσίας αναζήτησης που προσφέρει με >500 μηχανές και >4000 πυρήνες.
- **NAVTEQ Media Solutions** - Χρησιμοποιεί το Hadoop, για να επεξεργάζεται τις καταγραφές χρήσης διαφημίσεων και να βελτιστοποιεί τη στοχευμένη παρουσίαση διαφημίσεων.
- Yahoo! - Χρησιμοποιεί >40000 μηχανές για να εκτελεί το Hadoop, για να υποστηρίζει το σύστημα προβολής διαφημίσεων, καθώς επίσης και τις αναζητήσεις.

8.5 Εργαλεία του Hadoop

Στη συνέχεια, παρουσιάζονται τα επιμέρους εργαλεία, που συναντά κανείς κατά την ενασχόλησή του με το Hadoop.

- **Hadoop** - Το Hadoop της Apache αναπτύσσει λογισμικό ανοιχτού κώδικα για κατανεμημένα συστήματα πληροφορικής.
- **HDFS** - Ένα κατανεμημένο σύστημα αρχείων, που παρέχει υψηλό βαθμό εξυπηρέτησης από κάθε μονάδα του cluster σε ένα μεγάλο σύνολο δεδομένων.
- **MapReduce** - Ένα λογισμικό/προγραμματιστικό παράδειγμα για κατανεμημένη επεξεργασία σε ένα μεγάλο σύνολο δεδομένων ενός cluster.
- **Amazon Elastic MapReduce** - Είναι μια web υπηρεσία, η οποία επιτρέπει σε επιχειρηματίες, ερευνητές, αναλυτές δεδομένων και προγραμματιστές να επεξεργαστούν μεγάλες ποσότητες δεδομένων εύκολα και αποδοτικά.
- **Cloudera Distribution for Hadoop (CDH)** - Είναι μια συλλογή εφαρμογών λογισμικού που σχεδιάστηκαν, έτσι ώστε ο χρήσης του Hadoop να είναι σε θέση να το αξιοποιήσει στο έπακρο. Ο χρήστης αποκτά πλήρη έλεγχο στο Hadoop, αφού μπορεί έξυπνα και με ακρίβεια να χειριστεί το σύστημά του.
- **ZooKeeper** - Μία υψηλής απόδοσης υπηρεσία για κατανεμημένες εφαρμογές. Μια κεντρική υπηρεσία για τη διατήρηση πληροφοριών διαμόρφωσης, ονομασιών, παροχής κατανεμημένου συγχρονισμού και παροχής ομάδων.
- **HBase** - Μία κατανεμημένη βάση δεδομένων, που υποστηρίζει την αποθήκευση δομημένων δεδομένων για μεγάλους πίνακες.
- **Avro** - Ένα σύστημα, που μπορεί να διαμορφώσει τα δεδομένα, έτσι ώστε τα τελευταία να μπορούν να υποθηκευθούν. Παρόμοια συστήματα είναι και τα Thrift, Protocol Buffers.
- **Sqoop** - SQL στο Hadoop (Sqoop) είναι ένα εργαλείο μιας απλής γραμμής εντολών, που εισάγει μεμονωμένους πίνακες ή ολόκληρη βάση δεδομένων στο HDFS, δημιουργεί Java κλάσεις, για να μπορεί ο χρήστης να επέμβει στα δεδομένα του. Τέλος, παρέχει τη δυνατότητα στον χρήστη να εισάγει βάσεις δεδομένων SQL στο σύστημα αποθήκευσης του Hive.
- **Flume** - Είναι μία κατανεμημένη και αξιόπιστη υπηρεσία για τη μετακίνηση μεγάλων ποσοτήτων δεδομένων.
- **Hive** - Το Facebook χρησιμοποιεί τη γλώσσα Hive. Είναι μία γλώσσα υψηλότερου επίπεδου ερωτημάτων, χτισμένη πάνω στο MapReduce. Είναι η υποδομή αποθήκης δεδομένων που παρέχει εργαλεία για την περιληπτική παρουσίαση δεδομένων, την ad-hoc αναζήτηση (έρευνα – εντοπισμό – ανάκτηση δεδομένων με μεγάλη ταχύτητα) και την ανάλυση μεγάλων συνόλων δεδομένων, που είναι αποθηκευμένα σε αρχεία Hadoop. Παρέχει μηχανισμό, ο οποίος αλλάζει τα δεδομένα σε δομημένα δεδομένα και, επίσης, παρέχει μια απλή γλώσσα ερωτημάτων, την Hive QL, η οποία βασίζεται στην SQL, όμως, επιτρέπει και map/reduce συναρτήσεις.

- **Pig** - Μία γλώσσα υψηλού επιπέδου ερωτημάτων, δεδομένων ροής (Pig Latin) και εκτέλεσης σε παράλληλα υπολογιστικά συστήματα. Αναλύει μεγάλα σύνολα δεδομένων και πετυχαίνει και συγχώνευσή τους. Η Pig είναι ελαστική γλώσσα. Παρότι δημιουργεί το σχήμα των δεδομένων και κατά την ώρα της εκτέλεσης, όχι απαραίτητα από πριν, όπως γίνεται στις παραδοσιακές DBMS, το κάνει με τον ίδιο κατάλληλο και σωστό τρόπο.
- **Oozie** - Είναι μία υπηρεσία που διαχειρίζεται εργασίες επεξεργασίας δεδομένων και συντονίζει τις εξαρτήσεις μεταξύ των εργασιών που εκτελούνται στο Hadoop (και των HDFS, Pig, MapReduce).
- **Cascading** - Είναι μία διεπαφή προγραμματισμού εφαρμογών με ερωτήματα. Ένας Query Planner, ο οποίος βελτιώνει τα ερωτήματα προς τη βάση δεδομένων, ορίζει και εκτελεί σύνθετες επεξεργασίες δεδομένων με υψηλή ανοχή σε σφάλματα, στο Hadoop cluster.
- **Cascalog** - Είναι ένα εργαλείο για την επεξεργασία των δεδομένων στο Hadoop με τη γλώσσα προγραμματισμού Clojure. Το Cascalog συνδυάζει δύο σημαντικές τεχνολογίες (Clojure, Hadoop) και μία παλαιότερη, το Datalog (γλώσσα ερωτήσεων και κανόνων για λογικές βάσεις δεδομένων, η οποία συντακτικά είναι υποσύνολο της Prolog). Είναι εύρωστη, ευέλικτη και αποδοτική.
- **Hue** - Είναι μια γραφική διεπαφή χρήστη, η οποία λειτουργεί και αναπτύσσει εφαρμογές για Hadoop. Οι εφαρμογές συλλέγονται σε ένα περιβάλλον desktop και αποστέλλονται ως μια web εφαρμογή, η οποία δεν απαιτεί εγκατάσταση από τους χρήστες.
- **Chukwa** - Είναι ένα σύστημα συλλογής δεδομένων για την εποπτεία μεγάλων κατανεμημένων συστημάτων. Το Chukwa βρίσκεται πάνω από το HDFS και MapReduce framework και κληρονομεί την ευρωστία του Hadoop. Η Chukwa περιέχει, επίσης, ένα ισχυρό και ευέλικτο toolkit για την εμφάνιση, εποπτεία και ανάλυση αποτελεσμάτων, ώστε ο χρήστης να χρησιμοποιήσει με τον καλύτερο τρόπο τα αποτελέσματά του.
- **Mahout** - Χρησιμοποιεί το MapReduce και πετυχαίνει την εξόρυξη δεδομένων από το σύνολο των δεδομένων μας. Μελετά συμπεριφορές χρηστών και προσπαθεί να εντοπίσει παρόμοιους χρήστες, ομαδοποιεί έγγραφα σχετικά κ.ά.

8.6 Αρχιτεκτονική Hadoop

Ένα δίκτυο εκτέλεσης Hadoop μπορεί να αποτελείται από μια μέχρι και αρκετές δεκάδες χιλιάδες δικτυωμένες μηχανές. Επίσης, δεν απαιτείται να εγκατασταθεί σε μηχανές συγκεκριμένης αρχιτεκτονικής, λειτουργικού συστήματος ή προδιαγραφών, αφού μπορεί να τρέξει ακόμα και σε οικιακούς υπολογιστές, φτάνει να μπορούν οι τελευταίοι να υποστηρίζουν την πλατφόρμα Java.

8.6.1 Hadoop Distributed File System (HDFS)

Όταν ένα σύνολο δεδομένων ξεπερνά τη χωρητικότητα αποθήκευσης ενός και μόνο υπολογιστή, καθίσταται αναγκαίο να κατακερματιστεί σε ένα σύνολο από υπολογιστές. Τα συστήματα αρχείων, που διαχειρίζονται την αποθήκευση σε ένα δίκτυο υπολογιστών, ονομάζονται κατανεμημένα συστήματα αρχείων. Δεδομένου ότι αυτά βασίζονται σε κάποιο δίκτυο δεδομένων, υπεισέρχονται όλες οι επιπλοκές του προγραμματισμού μέσω δικτύου, καθιστώντας έτσι τα κατανεμημένα συστήματα αρχείων πιο περίπλοκα απ' ό,τι τα συστήματα αρχείων σε έναν μόνο σκληρό δίσκο. Για παράδειγμα, μία από τις μεγαλύτερες προκλήσεις είναι το κατανεμημένο σύστημα αρχείων να αντιμετωπίζει βλάβες σε υπολογιστικούς κόμβους, χωρίς να υποστεί απώλεια δεδομένων.

Το Hadoop έρχεται με ένα κατανεμημένο σύστημα αρχείων, που ονομάζεται HDFS (Hadoop Distributed File System). Μπορεί μερικές φορές να δείτε να αναφέρονται σε αυτό και ως «DFS» – ιδιαίτερα σε άτυπες αναφορές ή σε παλαιότερα έγγραφα. Το HDFS είναι η ναυαρχίδα των συστημάτων αρχείων του Hadoop και είναι το επίκεντρο αυτού του κεφαλαίου. Ωστόσο, το Hadoop προσφέρει και μια διεπαφή για γενικού σκοπού συστήματα αρχείων.

8.6.2 Η Αρχιτεκτονική του HDFS

Το HDFS είναι ένα σύστημα αρχείων σχεδιασμένο για την αποθήκευση πολύ μεγάλων αρχείων, με υποστήριξη για πρότυπα προσπέλασης σε δεδομένα συνεχούς ροής *(streaming data access patterns), σε συστάδες τυπικών υπολογιστών*. Προκειμένου να αναλυθεί η παραπάνω πρόταση παραθέτουμε τα παρακάτω στοιχεία:

«…πολύ μεγάλων αρχείων…»

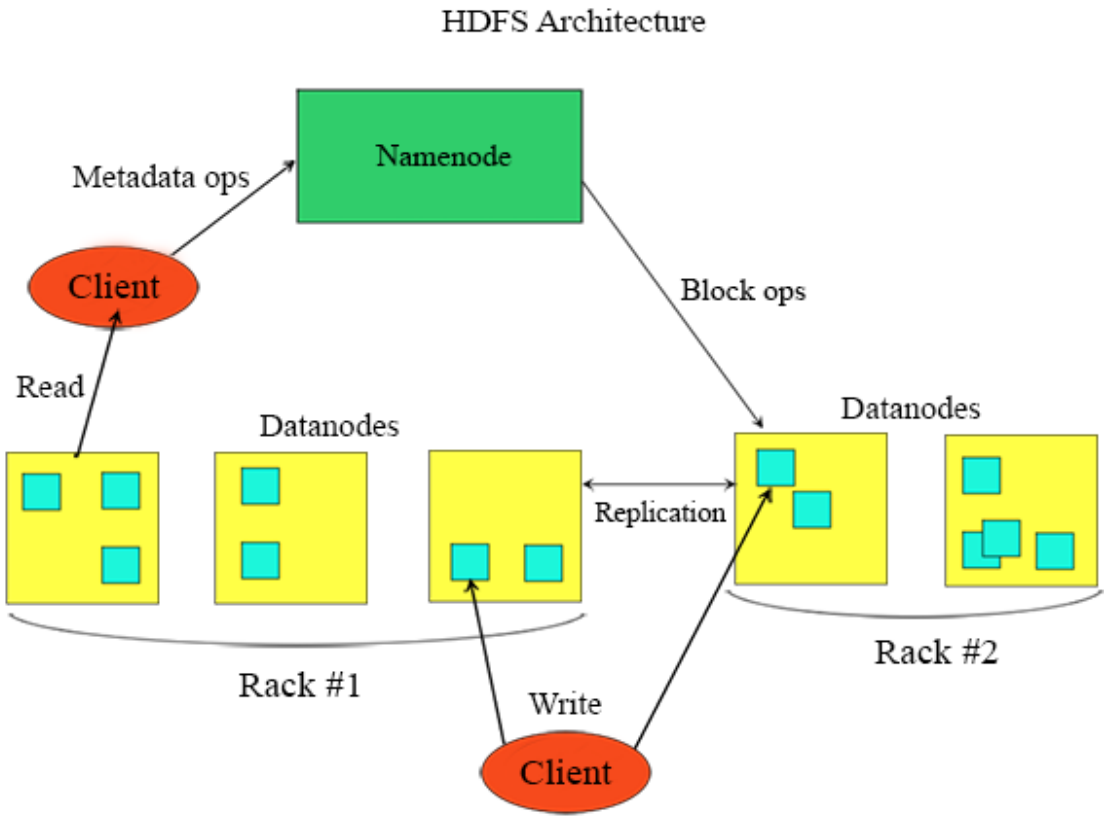
Στο πλαίσιο του Hadoop κάτι τέτοιο σημαίνει ότι τα αρχεία μπορεί να έχουν μέγεθος που είναι εκατοντάδες Megabytes, Gigabytes ή ακόμα και Terabytes. Υπάρχουν σήμερα συστάδες Hadoop σε λειτουργία, που αποθηκεύουν Petabytes δεδομένων.

«…προσπέλασης σε δεδομένα συνεχούς ροής…»

Το HDFS είναι χτισμένο γύρω από την ιδέα ότι το πιο αποτελεσματική πρότυπο επεξεργασίας δεδομένων είναι αυτό της «εγγραφής-μια φορά, ανάγνωσης-πολλές φορές» (write-once, read-multiple times). Τυπικά, ένα σύνολο δεδομένων παράγεται ή αντιγράφεται από προϋπάρχουσα πηγή, και, στη συνέχεια, πραγματοποιούνται διάφορες αναλύσεις σε αυτό το σύνολο δεδομένων. Κάθε ανάλυση περιλαμβάνει ένα μεγάλο μέρος, αν όχι όλο, από το σύνολο των δεδομένων. Κατά συνέπεια, ο χρόνος για την ανάγνωση ολόκληρου του συνόλου δεδομένων είναι πιο σημαντικός από τον χρόνο ανάγνωσης της πρώτης εγγραφής.

«…τυπικών υπολογιστών…»

Το Hadoop δεν απαιτεί ακριβό ή εξαιρετικά αξιόπιστο υλικό. Έχει σχεδιαστεί, ώστε να εκτελείται σε συστάδες που αποτελούνται από τυπικούς υπολογιστές, το υλικό των οποίων μπορεί να προμηθευτεί κάποιος από περισσότερους προμηθευτές. Αυτό καθιστά την πιθανότητα αποτυχίας ενός κόμβου στη συστάδα υψηλή, ιδιαίτερα για συστάδες μεγάλου μεγέθους. Το HDFS έχει σχεδιαστεί να συνεχίζει να δουλεύει αξιόπιστα και χωρίς εμφανείς διακοπές λειτουργίας προς τον χρήστη σε περίπτωση τέτοιων προβλημάτων.



Εικόνα ***8.2*** *Η Αρχιτεκτονική του HDFS.*

8.6.3 HDFS – Τομείς Χαμηλής Απόδοσης

Στη συνέχεια, παρουσιάζονται μερικοί τύποι εφαρμογών, στους οποίους η χρήση του HDFS δεν έχει την επιθυμητή απόδοση.

8.6.3.1 Πρόσβαση σε δεδομένα με μικρό χρόνο προσπέλασης

Οι εφαρμογές, που απαιτούν πρόσβαση σε δεδομένα με μικρό χρόνο προσπέλασης, της τάξης κάποιων δεκάδων χιλιοστών του δευτερολέπτου, δεν έχουν καλή επίδοση, όταν χρησιμοποιείται το HDFS. Όπως προαναφέρθηκε, το HDFS είναι βελτιστοποιημένο για την επίτευξη υψηλού ρυθμού διαβίβασης δεδομένων (throughput), το οποίο τυπικά έρχεται σε σύγκρουση με την επίτευξη μικρού χρόνου προσπέλασης. Το εργαλείο HBase αποτελεί προς το παρόν την καλύτερη επιλογή για εφαρμογές, που απαιτούν μικρούς χρόνους προσπέλασης.

8.6.3.2 Πολλά μικρά αρχεία

Καθώς το namenode ενός συστήματος Hadoop αποθηκεύει τα μεταδεδομένα του συστήματος αρχείων στη μνήμη, το πλήθος των αρχείων, που μπορεί να αποθηκευτεί με χρήση του HDFS, περιορίζεται από τη διαθέσιμη μνήμη στο namenode. Προσεγγιστικά, η πληροφορία για κάθε αρχείο, κατάλογο και block δεδομένων απαιτεί περίπου 150 byte. Έτσι, για παράδειγμα, αν είχατε ένα εκατομμύριο αρχεία, καθένα εκ των οποίων καταλάμβανε ένα block, θα χρειαζόσασταν τουλάχιστον 300 MB μνήμης. Αν και η αποθήκευση κάποιων εκατομμύριων αρχείων είναι εφικτή, η αποθήκευση μεγαλύτερου πλήθους είναι προς το παρόν πέρα από τις δυνατότητες του υπάρχοντος υλικού.

8.6.3.3 Πολλαπλοί κόμβοι εγγραφής δεδομένων, αυθαίρετες τροποποιήσεις αρχείων

Τα αρχεία στο HDFS μπορούν να γράφονται μόνο από έναν κόμβο. Νέες εγγραφές προστίθενται πάντα στο τέλος του αρχείου. Δεν υπάρχει υποστήριξη για πολλαπλούς κόμβους, που θέλουν να γράψουν δεδομένα, ή για τροποποιήσεις σε αυθαίρετες θέσεις εντός του αρχείου. Τα παραπάνω ίσως υποστηρίζονται σε μελλοντικές εκδόσεις του HDFS, αλλά ακόμα και έτσι είναι εξαιρετικά πιθανό να είναι σχετικά αργές λειτουργίες.

8.6.4 Βασικές Έννοιες του HDFS

8.6.4.1 Η έννοια των block

Κάθε σκληρός δίσκος έχει ένα προκαθορισμένο μέγεθος block στο επίπεδο του υλικού, το οποίο είναι το ελάχιστο πλήθος δεδομένων, που μπορεί να διαβάσει ή να γράψει σε κάθε αντίστοιχη λειτουργία. Τα τυπικά συστήματα αρχείων για ένα μόνο υπολογιστικό σύστημα διαχειρίζονται τα δεδομένα, επίσης, σε μεγέθη block, τα οποία πρέπει να είναι πολλαπλάσια του μεγέθους του block του σκληρού δίσκου. Συνήθως τα block του συστήματος αρχείων είναι μερικά kilobytes σε μέγεθος (τυπικά 1KB, 2KB ή 4KB), ενώ τα block του σκληρού δίσκου είναι συνήθως 512 byte. Η ύπαρξη των block κρύβεται από τον χρήστη του συστήματος αρχείων, ο οποίος απλά διαβάζει ή γράφει ένα αρχείο, το οποίο μπορεί να έχει οποιοδήποτε μέγεθος. Ωστόσο, υπάρχουν εργαλεία για τη συντήρηση του συστήματος αρχείων, όπως τα df και fsck, τα οποία λειτουργούν σε επίπεδο block του συστήματος αρχείων.

Το HDFS, επίσης, έχει την έννοια block, ωστόσο αυτό είναι μια πολύ μεγαλύτερη μονάδα – το προεπιλεγμένο μέγεθος είναι 128 MB. Όπως και σε ένα σύστημα αρχείων για ένα μόνο υπολογιστικό σύστημα, τα αρχεία στο HDFS χωρίζονται σε μικρότερα τμήματα σε μέγεθος block, τα οποία, στη συνέχεια, αποθηκεύονται ως ανεξάρτητες μονάδες. Σε αντίθεση με ένα σύστημα αρχείων για ένα μόνο υπολογιστικό σύστημα, ένα αρχείο στο HDFS, που έχει μέγεθος μικρότερο από ένα block, δεν καταλαμβάνει ένα ολόκληρο block στο αποθηκευτικό μέσο (ωστόσο, στο πλαίσιο του HDFS συνεχίζει να θεωρείται ότι καταλαμβάνει ένα ολόκληρο HDFS block).

Το μέγεθος του block στο HDFS είναι μεγάλο σε σχέση με το block ενός σκληρού δίσκου και ο λόγος είναι, για να ελαχιστοποιηθεί το κόστος της αναζήτησης των δεδομένων. Κάνοντας ένα block αρκετά μεγάλο, ο χρόνος για τη μεταφορά των δεδομένων από τον δίσκο γίνεται σημαντικά μεγαλύτερος από τον χρόνο για την αναζήτηση της αρχής του block. Έτσι, ο χρόνος για τη μεταφορά ενός μεγάλου αρχείου, που αποτελείται από πολλά block, μπορεί να πραγματοποιηθεί με την ταχύτητα μεταφοράς των δεδομένων από τον σκληρό δίσκο.

Για παράδειγμα, αν θεωρήσουμε πως ο χρόνος αναζήτησης (seek time) είναι 10 ms και ο ρυθμός μεταφοράς είναι 100 MB/s, τότε, για να καταστεί ο χρόνος αναζήτησης το 1% του χρόνου μεταφοράς, πρέπει να ορίσουμε το μέγεθος του block περίπου στα 100 MB. Το εξ ορισμού μέγεθος ενός block, όπως προαναφέρθηκε, είναι 128 MB, δηλαδή πολύ κοντά στο μέγεθος που υπολογίσαμε στο παράδειγμά μας. Το μέγεθος αυτό θα συνεχίσει να αναθεωρείται προς τα πάνω, καθώς οι ταχύτητες μεταφοράς θα αυξάνονται στις νέες γενιές σκληρών δίσκων.

Ωστόσο, θα πρέπει να γίνει αντιληπτό ότι υπάρχει και κάποιο όριο στο παραπάνω επιχείρημα. Οι λειτουργίες απεικόνισης ("Map") στο μοντέλο MapReduce υπό κανονικές συνθήκες λειτουργούν σε ένα μόνο block. Κατά συνέπεια, με πολύ μεγάλα μεγέθη block θα μπορούσαν να δημιουργηθούν λιγότερες εργασίες απεικόνισης απ' ό,τι το πλήθος των διαθέσιμων κόμβων και τελικά το πρόγραμμα μας να εκτελεστεί πιο αργά. Η αφαιρετική έννοια του block προσφέρει πολλά οφέλη σε ένα κατανεμημένο σύστημα αρχείων. Το πρώτο πλεονέκτημα είναι και το πιο προφανές: ένα αρχείο μπορεί να είναι μεγαλύτερο από οποιονδήποτε σκληρό δίσκο, που είναι διαθέσιμος στο δίκτυο. Δεν υπάρχει κανένας περιορισμός, που να απαιτεί τα block ενός αρχείου να είναι αποθηκευμένα στον ίδιο σκληρό δίσκο. Κατά συνέπεια, μπορούμε να αξιοποιήσουμε οποιονδήποτε από τους διαθέσιμους δίσκους της συστάδας. Στην πραγματικότητα θα ήταν δυνατό, αν και βέβαια ασυνήθιστο, να αποθηκεύσουμε ένα και μόνο μεγάλο αρχείο σε μια συστάδα HDFS, του οποίου τα blocks θα καταλαμβάνουν όλους τους διαθέσιμους σκληρούς δίσκους της συστάδας.

Σύμφωνα με το δεύτερο πλεονέκτημα, καθιστώντας ως μονάδα διαχείρισης ένα block και όχι ένα αρχείο, απλοποιείται το υποσύστημα αποθήκευσης. Η απλότητα είναι ένα χαρακτηριστικό, το οποίο ούτως ή άλλως αγωνίζονται όλα τα συστήματα να διαθέτουν, αλλά είναι ιδιαίτερα σημαντικό για ένα κατανεμημένο σύστημα, στο οποίο οι πιθανοί λόγοι αποτυχίας κόμβων είναι τόσο ποικίλοι. Το υποσύστημα αποθήκευσης διαχειρίζεται block, καθιστώντας τη διαχείριση του αποθηκευτικού χώρου απλούστερη (κάθε block έχει σταθερό μέγεθος και, επομένως, είναι εύκολο να υπολογιστεί πόσα από αυτά μπορούν να αποθηκευτούν σε έναν δεδομένο σκληρό δίσκο), εξαλείφοντας ταυτόχρονα την ανάγκη για τη διατήρηση μεταδεδομένων (κάθε block είναι απλά μια ακολουθία δεδομένων, που πρέπει να αποθηκευτεί, και έτσι τα μεταδεδομένα ενός αρχείου, όπως τα δικαιώματα προσπέλασης, δεν είναι απαραίτητο να αποθηκεύονται με το block, αλλά μπορούν να διαχειρίζονται αυτόνομα από κάποιο άλλο υποσύστημα).

Επιπλέον, η έννοια του block ταιριάζει καλά με την έννοια της αντιγραφής για την παροχή ανοχής σφαλμάτων και τη διαθεσιμότητα των δεδομένων. Για να εξασφαλιστεί η αδιάκοπη λειτουργία του συστήματος και να προστατευθούν τα δεδομένα από κατεστραμμένα blocks, κατεστραμμένους σκληρούς δίσκους ή κατεστραμμένους κόμβους της συστάδας, κάθε block αντιγράφεται σε ένα μικρό σύνολο διαφορετικών κόμβων της συστάδας (τυπικά σε τρεις). Αν ένα block σταματήσει να είναι διαθέσιμο από έναν κόμβο, τότε μπορεί να αναγνωστεί από άλλον κόμβο με εντελώς διάφανο τρόπο προς τον χρήστη. Ένα block, που δεν είναι πια διαθέσιμο, μπορεί να αντιγραφεί από τα υπόλοιπα διαθέσιμα αντίγραφα σε άλλον κόμβο της συστάδας, ώστε συνολικά να υπάρχει πάλι το προκαθορισμένο πλήθος αντιγράφων του block αυτού στη συστάδα. Επιπλέον, μια εφαρμογή θα μπορούσε να ζητήσει ένα μεγαλύτερο πλήθος αντιγράφων για κάθε block ενός αρχείου που προσπελαύνεται συχνά, ώστε να μειώσει τον φόρτο εργασίας κάθε κόμβου της συστάδας για την ανάγνωση του αρχείου αυτού.

8.6.4.2 Namenodes και Datanodes

Μια συστάδα με HDFS έχει δύο τύπους κόμβων, που λειτουργούν κάτω από το μοντέλο «αφέντη-σκλάβου» (master-slave): ένα namenode (αφέντη) και μια σειρά από datanodes (σκλάβοι). Το namenode διαχειρίζεται τον χώρο ονομάτων αρχείων. Με λίγα λόγια, διατηρεί το δέντρο του συστήματος αρχείων και τα μεταδεδομένα για όλα τα αρχεία και τους καταλόγους στο δέντρο. Αυτές οι πληροφορίες αποθηκεύονται μόνιμα στον τοπικό δίσκο του namenode με τη μορφή δύο αρχείων: την «εικόνα ονομάτων» (namespace image) και το αρχείο καταγραφής των λειτουργιών επεξεργασίας (edit log). Το namenode γνωρίζει, επίσης, σε ποια datanodes βρίσκονται τα blocks κάθε αρχείου. Η πληροφορία αυτή, όμως, δεν αποθηκεύεται μόνιμα, γιατί ανακατασκευάζεται από τα datanodes κατά την εκκίνηση του συστήματος.

Μια εφαρμογή-πελάτης έχει πρόσβαση στο σύστημα αρχείων για λογαριασμό του χρήστη, επικοινωνώντας με το namenode και τα datanodes. Η εφαρμογή-πελάτης χρησιμοποιεί μια προγραμματιστική διεπαφή (Application Programming Interface ή API), παρόμοια με αυτή που ορίζει το πρότυπο Portable Operating System Interface for Unix (POSIX) για τα κοινά συστήματα αρχείων, έτσι ώστε ο κώδικας του χρήστη να μην χρειάζεται να γνωρίζει την ύπαρξη του namenode και των datanodes, για να λειτουργήσει σωστά.

Τα datanodes αποτελούν τον ακρογωνιαίο λίθο του συστήματος αρχείων HDFS. Αποθηκεύουν και ανα-

κτούν block όποτε τους ζητείται (από της εφαρμογές-πελάτες ή το namenode), ενώ ενημερώνουν περιοδικά το namenode με το σύνολο των blocks, που έχουν αποθηκευμένα.

Από την άλλη, χωρίς το namenode δεν μπορεί να χρησιμοποιηθεί το σύστημα αρχείων. Για την ακρίβεια, αν για κάποιον λόγο καταστραφεί ο κόμβος της συστάδας που λειτουργεί ως namenode, όλα τα αρχεία στο σύστημα αρχείων θα χαθούν, καθώς δεν θα υπάρχει τρόπος να ανακατασκευαστούν τα αρχεία από τα blocks που υπάρχουν στα datanodes. Για τον λόγο αυτό είναι σημαντικό το namenode να είναι ανθεκτικό σε τυχόν αστοχία του υλικού και το Hadoop παρέχει δύο μηχανισμούς για να επιτευχθεί αυτό.

Ο πρώτος τρόπος είναι να δημιουργούνται αντίγραφα ασφαλείας των αρχείων, τα οποία απαρτίζουν τα μεταδεδομένα των αρχείων που υπάρχουν στο HDFS. Το Hadoop μπορεί να ρυθμιστεί, έτσι ώστε το namenode να αντιγράφει τα αρχεία αυτά σε πολλά συστήματα αρχείων. Οι παραπάνω αντιγραφές είναι σύγχρονες και ατομικές. Για παράδειγμα, μια συχνή επιλογή σε μια συστάδα με HDFS είναι να αντιγράφονται τα αρχεία των μεταδεδομένων στον τοπικό σκληρό δίσκο του namenode, καθώς και σε μια απομακρυσμένη προσάρτηση NFS (remote NFS mount).

Είναι, επίσης, δυνατόν να χρησιμοποιηθεί ένα δευτερεύον namenode, ένας κόμβος, που παρά την ονομασία του, δεν ενεργεί στην πραγματικότητα ως namenode. Ο κύριος ρόλος του είναι να συγχωνεύει σε τακτά χρονικά διαστήματα την «εικόνα ονομάτων» (namespace image) και το αρχείο καταγραφής των λειτουργιών επεξεργασίας (edit log), ώστε να αποφευχθεί το δεύτερο να γίνει πάρα πολύ μεγάλο σε μέγεθος. Το δευτερεύον namenode τρέχει συνήθως σε έναν ξεχωριστό κόμβο της συστάδας, διότι απαιτεί σημαντική επεξεργαστική ισχύ και τόση μνήμη όση και το namenode, για να πραγματοποιήσει τη συγχώνευση. Κρατά ένα αντίγραφο της νέας «εικόνας ονομάτων», η οποία μπορεί να χρησιμοποιηθεί σε περίπτωση βλάβης του namenode. Ωστόσο, η κατάσταση που απεικονίζεται στο δευτερεύον namenode για το σύστημα αρχείων συνήθως υστερεί της κατάστασης που υπάρχει στο namenode, καθώς η συγχώνευση της πληροφορίας γίνεται σε διακριτές χρονικές στιγμές. Έτσι, σε περίπτωση βλάβης του namenode είναι σχεδόν βέβαιη η απώλεια κάποιων δεδομένων. Η συνήθης πορεία δράσης στην περίπτωση αυτή είναι η αντιγραφή των αρχείων μεταδεδομένων του namenode, τα οποία βρίσκονται στην απομακρυσμένη προσάρτηση NFS στο δευτερεύον namenode και η μετατροπή του τελευταίου σε κυρίως namenode.

8.6.4.3 HDFS Federation

Το namenode διατηρεί στην κύρια μνήμη του μια αναφορά προς κάθε αρχείο και block του κατανεμημένου συστήματος αρχείων. Κατά συνέπεια, σε πολύ μεγάλες συστάδες με πολλά αρχεία η κύρια μνήμη είναι αυτή που γίνεται ο βασικός παράγοντας, που δεν επιτρέπει την κλιμάκωση του μεγέθους του συστήματος αρχείων. Το HDFS Federation, το οποίο εισήχθη στη σειρά εκδόσεων 2.x του Hadoop, επιτρέπει σε μια συστάδα κόμβων να κλιμακώσει το κατανεμημένο σύστημα αρχείων της μέσω της προσθήκης περισσότερων namenode. Καθένα από τα namenode διαχειρίζεται στην περίπτωση αυτή ένα μέρος του χώρου ονομάτων αρχείων. Για παράδειγμα, ένα namenode μπορεί να διαχειρίζεται όλα τα αρχεία κάτω από τον κατάλογο "/user", ενώ ένα δεύτερο namenode θα μπορούσε να διαχειρίζεται τα αρχεία κάτω από τον κατάλογο "/share".

Κάτω από την επίβλεψη του Federation, κάθε namenode διαχειρίζεται ένα σύνολο ονομάτων, το οποίο αποτελείται από τα μεταδεδομένα για τα ονόματα αυτά, καθώς και από ένα σύνολο blocks, που περιέχει τα blocks των αρχείων που ανήκουν στο σύνολο ονομάτων. Κάθε σύνολο ονομάτων είναι ανεξάρτητο από τα άλλο, που σημαίνει πως τα namenodes δεν επικοινωνούν μεταξύ τους, και, επιπλέον, η βλάβη σε ένα namenode δεν επηρεάζει τη διαθεσιμότητα των ονομάτων, που διαχειρίζονται άλλα namenode. Ωστόσο, το σύνολο των blocks δεν είναι διαμοιρασμένο, δηλαδή κάθε datanode επικοινωνεί με κάθε namenode της συστάδας και μπορεί να αποθηκεύει block από διαφορετικά σύνολα αρχείων.

8.6.4.4 HDFS High-Availability

Ο συνδυασμός της αντιγραφής των μεταδεδομένων, που διατηρεί ένα namenode σε πολλαπλά συστήματα αρχείων, και η αξιοποίηση ενός δευτερεύοντος namenode για τη δημιουργία σημείων ελέγχου και αποκατάστασης προστατεύει από την απώλεια δεδομένων, αλλά δεν παρέχει «υψηλή διαθεσιμότητα» (high availability) του συστήματος αρχείων. Το namenode εξακολουθεί να είναι ένα μοναδικό σημείο αποτυχίας (Single Point of Failure ή SPOF). Αν για οποιονδήποτε λόγο υποστεί βλάβη, όλοι οι πελάτες του συστήματος αρχείων – συμπεριλαμβανομένων των εργασιών MapReduce, που μπορεί να εκτελούνται εκείνη τη χρονική στιγμή– δεν θα είναι σε θέση να διαβάσουν, να γράψουν ή ακόμα και να δουν έναν κατάλογο με τα διαθέσιμα αρχεία, καθώς το namenode είναι το μοναδικό αποθετήριο των μεταδεδομένων των αρχείων και του χάρτη απεικόνισης των

αρχείων σε block. Σε μια τέτοια περίπτωση όλο το σύστημα Hadoop θα είναι πρακτικά εκτός λειτουργίας, έως ότου ένα νέο namenode τεθεί σε κατάσταση λειτουργίας.

Για να επαναφέρει ένας διαχειριστής μια συστάδα Hadoop σε περίπτωση βλάβης του namenode, θα πρέπει να ξεκινήσει ένα νέο πρωτεύων namenode, να μεταφέρει σε αυτόν ένα από τα αντίγραφα των αρχείων μεταδεδομένων και να ενημερώσει τα datanodes και τους πελάτες, ώστε να χρησιμοποιούν πλέον το νέο namenode. Το νέο namenode δεν είναι σε θέση να εξυπηρετήσει αιτήματα για χρήση αρχείων, μέχρις ότου i) φορτωθεί η «εικόνα ονομάτων» στη μνήμη του, ii) επανεκτελέσει τις εντολές επεξεργασίας αρχείων, που πραγματοποιήθηκαν από την τελευταία δημιουργία αντιγράφου «εικόνας ονομάτων» και έχουν καταγραφεί στο edit log, και iii) λάβει αρκετές ενημερώσεις από τα datanodes για το ποια block διατηρεί το καθένα, ώστε να μπορέσει να βγει από την κατάσταση ασφαλούς λειτουργίας και να μπει σε κατάσταση κανονικής λειτουργίας. Σε μεγάλες συστάδες με πολλά αρχεία και blocks, ο χρόνος που χρειάζεται για ένα namenode να ξεκινήσει μπορεί να είναι 30 λεπτά ή και περισσότερο.

Ο μεγάλος χρόνος αποκατάστασης αποτελεί σημαντικό πρόβλημα και για τις διαδικασίες συντήρησης ρουτίνας της συστάδας. Στην πράξη, επειδή η απροσδόκητη αποτυχία του namenode είναι εξαιρετικά σπάνια, η περίπτωση προγραμματισμένης διακοπής λειτουργίας και συντήρησης είναι πιο σημαντική.

Στη σειρά εκδόσεων 2.x του Hadoop γίνεται προσπάθεια βελτίωσης της κατάστασης μέσω της προσθήκης του HDFS High-Availability (HA). Σε αυτή την προσέγγιση υπάρχει ένα ζευγάρι από namenodes, εκ των οποίων το ένα namenode είναι «ενεργό», ενώ το δεύτερο είναι σε «αναμονή». Σε περίπτωση βλάβης του ενεργού namenode, το δεύτερο namenode αναλαμβάνει καθήκοντα «ενεργού» namenode, ώστε να συνεχιστεί η απρόσκοπτη εξυπηρέτηση των αιτημάτων των πελατών, χωρίς σημαντικό χρόνο διακοπής. Μερικές αλλαγές στην αρχιτεκτονική του συστήματος απαιτούνται, για να μπορέσει να υποστηριχθεί η λειτουργία αυτή:

- Τα namenodes πρέπει να χρησιμοποιούν υψηλής διαθεσιμότητας και κοινής προσπέλασης χώρο αποθήκευσης, ώστε να μπορούν να διαμοιράζονται το αρχείο καταγραφής επεξεργασίας (edit log). (Στην αρχική έκδοση του HA αυτό απαιτεί τον ορισμό ενός "NFS filer", αλλά σε μελλοντικές εκδόσεις θα παρέχονται περισσότερες επιλογές, όπως ένα σύστημα που βασίζεται στο BookKeeper, το οποίο έχει υλοποιηθεί μέσω του ZooKeeper.) Όταν ένα namenode μεταβαίνει σε ενεργή κατάσταση, τότε διαβάζει όλο το αρχείο καταγραφής επεξεργασίας και εκτελεί πάλι όλες τις καταγεγραμμένες πράξεις σε αρχεία, ώστε να συγχρονίσει την κατάσταση του με το namenode που υπέστη τη βλάβη και, στη συνέχεια, διαβάζει νέες καταχωρήσεις, όπως κάνει κανονικά το ενεργό namenode.
- Τα datanodes πρέπει να αποστέλλουν τα σύνολα των blocks που διαχειρίζονται σε δύο namenodes, καθώς η πληροφορία αυτή είναι αποθηκευμένη στη μνήμη του κάθε namenode του και όχι στον σκληρό δίσκο.
- Οι πελάτες του συστήματος αρχείων θα πρέπει να ρυθμιστούν, ώστε να μπορούν να χειριστούν τη μετάβαση σε διαφορετικό namenode, χρησιμοποιώντας έναν μηχανισμό που θα είναι διαφανής προς τους χρήστες.

Όταν το ενεργό namenode βγει εκτός λειτουργίας, το εν αναμονή namenode μπορεί να αναλάβει πολύ γρήγορα (μέσα σε λίγα δέκατα του δευτερολέπτου), επειδή έχει την τελευταία κατάσταση του συστήματος αρχείων διαθέσιμη στη μνήμη: τόσο τις τελευταίες καταχωρήσεις του edit log, όσο και τα τελευταία δεδομένα για την κατανομή των blocks στα datanodes. Στην πράξη, όμως, ο πραγματικός χρόνος, που απαιτείται για τη μετάβαση στο νέο namenode, είναι μεγαλύτερος (περίπου ένα λεπτό), επειδή το σύστημα πρέπει να είναι συντηρητικό ως προς την απόφαση που παίρνει για το πότε πραγματικά το ενεργό namenode έχει τεθεί εκτός λειτουργίας.

Στην απίθανη περίπτωση που και το εν αναμονή namenode είναι εκτός λειτουργίας, ο διαχειριστής μπορεί να ακολουθήσει τη διαδικασία που περιγράψαμε νωρίτερα για το απλό HDFS σύστημα αρχείων (χωρίς το Federation).

Η μετάβαση από τον ενεργό στο εν αναμονή namenode διαχειρίζεται από μια νέα οντότητα του συστήματος, που ονομάζεται «ελεγκτής ανακατεύθυνσης» (failover controller). Οι ελεγκτές ανακατεύθυνσης μπορούν να οριστούν δυναμικά κατά τον χρόνο λειτουργίας του συστήματος (pluggable). Η τρέχουσα υλοποίηση χρησιμοποιεί το ZooKeeper, για να εξασφαλίσει ότι κάθε χρονική στιγμή μόνο ένα namenode είναι ενεργό. Κάθε namenode τρέχει μια πολύ ελαφριά διεργασία ελέγχου, η οποία παρακολουθεί το namenode για τυχόν βλάβη και ενεργοποιεί την ανακατεύθυνση προς το νέο namenode, μόλις γίνει αντιληπτή μια βλάβη. Ο μηχανισμός, που χρησιμοποιείται για την παρακολούθηση, είναι ένας απλός «χτύπος καρδιάς» (heartbeat). Η μετάβαση από το ενεργό στο εν αναμονή namenode διαχειρίζεται από μια νέα οντότητα του συστήματος, που ονομά-

ζεται «ελεγκτής ανακατεύθυνσης» (failover controller). Οι ελεγκτές ανακατεύθυνσης μπορούν να οριστούν δυναμικά κατά τον χρόνο λειτουργίας του συστήματος (pluggable). Η τρέχουσα υλοποίηση χρησιμοποιεί το ZooKeeper, για να εξασφαλίσει ότι κάθε χρονική στιγμή μόνο ένα namenode είναι ενεργό. Κάθε namenode τρέχει μια πολύ ελαφριά διεργασία ελέγχου, η οποία παρακολουθεί το namenode για τυχόν βλάβη και ενεργοποιεί την ανακατεύθυνση προς το νέο namenode, μόλις γίνει αντιληπτή μια βλάβη. Ο μηχανισμός, που χρησιμοποιείται για την παρακολούθηση, είναι ένας απλός «χτύπος καρδιάς» (heartbeat), δηλαδή ένα απλό μήνυμα προς τον ελεγκτή ανακατεύθυνσης ανά τακτά χρονικά διαστήματα, ώστε ο τελευταίος να γνωρίζει ότι το namenode είναι σε λειτουργία.

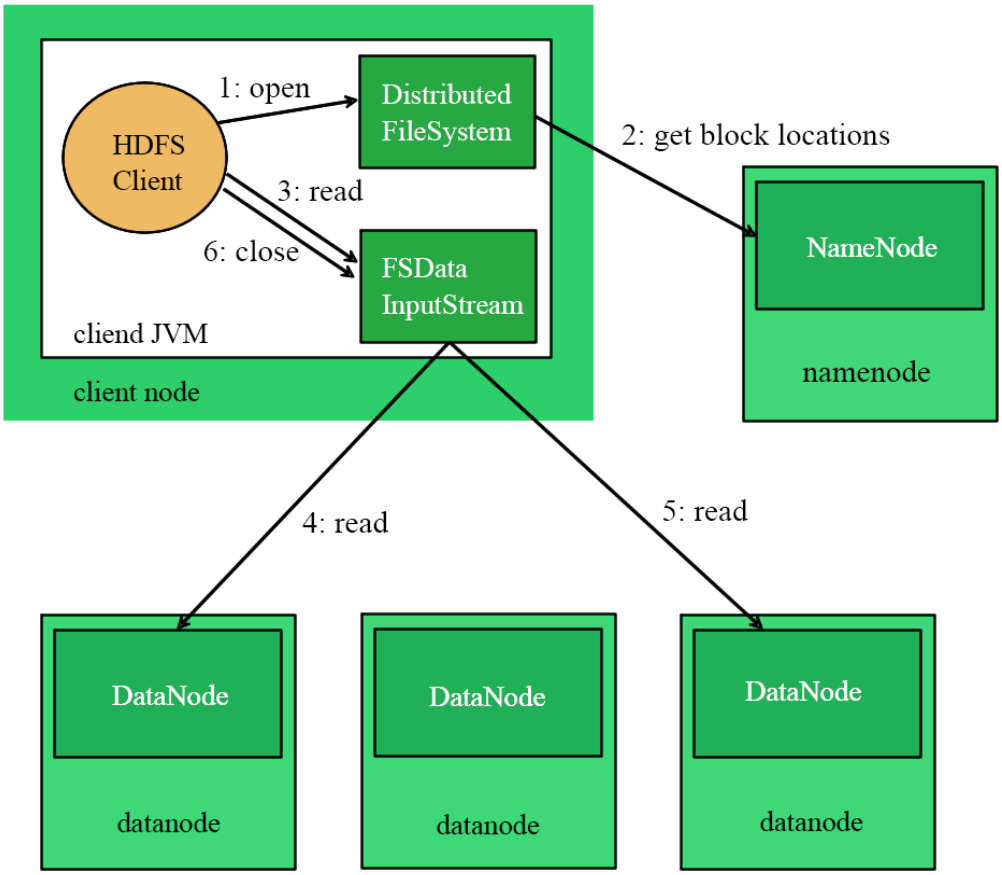
Ο μηχανισμός ανακατεύθυνσης προς νέο namenode μπορεί να τεθεί σε λειτουργία και «χειροκίνητα» από τον διαχειριστή του συστήματος, π.χ., στην περίπτωση που το ενεργό namenode πρέπει να κλείσει για προγραμματισμένη συντήρηση. Η διαδικασία αυτή είναι γνωστή και ως "graceful failover", καθώς ο ελεγκτής ανακατεύθυνσης οργανώνει μια ομαλή εναλλαγή ρόλων μεταξύ των δύο namenodes.

Στην περίπτωση της ενεργοποίησης του μηχανισμού ανακατεύθυνσης λόγω βλάβης, υπάρχει πάντα ένα ποσοστό αβεβαιότητας ως προς το αν πραγματικά το namenode έχει σταματήσει να λειτουργεί. Για παράδειγμα, ένα αργό δίκτυο ή τμήμα του συνολικού δικτύου μπορεί να ενεργοποιήσει τον μηχανισμό ανακατεύθυνσης, παρά το ότι το προηγούμενο ενεργό namenode εξακολουθεί να λειτουργεί και πιστεύει πως συνεχίζει να είναι το ενεργό namenode. Το HDFS HA καταβάλει μεγάλη προσπάθεια, για να εξασφαλίσει ότι το προηγούμενο ενεργό namenode δε θα κάνει οποιαδήποτε ζημιά, που θα προκαλέσει δυσλειτουργία του συστήματος. Η μέθοδος, που ακολουθείται στην περίπτωση αυτή, είναι γνωστή ως «περίφραξη». Το σύστημα χρησιμοποιεί μια σειρά από μηχανισμούς περίφραξης, συμπεριλαμβανομένου του τερματισμού της διαδικασίας που τρέχει το namenode, την ανάκληση της πρόσβασης της στον κοινό κατάλογο αποθήκευσης (συνήθως με τη χρήση συγκεκριμένων εντολών του NFS) και απενεργοποίηση της θύρας δικτύου της διεργασίας αυτής μέσω μιας απομακρυσμένης εντολής διαχείρισης. Ως έσχατη λύση, το προηγούμενο ενεργό namenode μπορεί να χρησιμοποιήσει μια τεχνική περιφραγής με τη μάλλον γραφική ονομασία STONITH (Shoot The Other Node In The Head ή πυροβόλησε τον άλλο κόμβο στο κεφάλι), η οποία χρησιμοποιεί μια εξειδικευμένη μονάδα διανομής ρεύματος, για να απενεργοποιήσει βίαια το άλλο namenode.

Η ανακατεύθυνση διαχειρίζεται τις εφαρμογές-πελάτες με διάφανο τρόπο. Ο απλούστερος τρόπος αντιμετώπισης χρησιμοποιεί ένα αρχείο διαμόρφωσης (configuration file) από την πλευρά της εφαρμογής-πελάτη για τον έλεγχο της ανακατεύθυνσης. Στη συνέχεια, το HDFS χρησιμοποιεί ένα λογικό όνομα για το namenode, το οποίο έχει αντιστοιχιστεί στο ζευγάρι των διευθύνσεων των δύο namenodes. Όταν η εφαρμογή-πελάτης προσπαθήσει να προσπελάσει ένα αρχείο, τότε δοκιμάζεται κάθε διεύθυνση namenode στο αρχείο διαμόρφωσης, μέχρι να επιτύχει η λειτουργία.

8.6.5 Ροή Δεδομένων – Ανάγνωση Δεδομένων

Για να γίνει καλύτερα αντιληπτός ο τρόπος με τον οποίο ρέουν τα δεδομένα μεταξύ μιας εφαρμογής-πελάτη, που αλληλοεπιδρά με το HDFS, το namenode και τα datanodes, παραθέτουμε την Εικόνα 8.3, στην οποία φαίνεται η κύρια ακολουθία των γεγονότων, τα οποία λαμβάνουν χώρα κατά την ανάγνωση ενός αρχείου.



Εικόνα *8.3* *Ανάγνωση δεδομένων από αρχείο HDFS.*

Ο πελάτης ανοίγει το αρχείο που επιθυμεί να διαβάσει, καλώντας τη μέθοδο open() για ένα στιγμιότυπο της κλάσης FileSystem. Συγκεκριμένα, στην περίπτωση του HDFS, το στιγμιότυπο αυτό είναι του τύπου DistributedFileSystem, με την τελευταία κλάση να είναι υποκλάση της FileSystem (βήμα 1 στην Εικόνα 8.3). Μέσω του στιγμιότυπου τύπου DistributedFileSystem πραγματοποιείται επικοινωνία με το namenode, χρησιμοποιώντας μια τεχνική γνωστή ως «απομακρυσμένη κλήση συνάρτησης» (Remote Procedure Call ή RPC), η οποία επιτρέπει την εκτέλεση μιας μεθόδου σε διαφορετικό υπολογιστικό σύστημα από αυτό στο οποίο απαιτούνται τα αποτελέσματα της κλήσης. Σκοπός της απομακρυσμένης κλήσης είναι να προσδιοριστούν τα datanodes, οι οποίοι περιέχουν αντίγραφα των πρώτων blocks του αρχείου (βήμα 2). Για κάθε block, το namenode επιστρέφει τις διευθύνσεις των datanodes που έχουν αντίγραφο αυτών των blocks. Επιπλέον, τα datanodes ταξινομούνται ανάλογα με την εγγύτητά τους στον πελάτη, σύμφωνα με την τοπολογία του δικτύου της συστάδας. Αν ο πελάτης εκτελείται σε κάποιο datanode (για παράδειγμα, είναι μια εργασία MapReduce), τότε ο πελάτης θα διαβάσει από το τοπικό datanode το αντίγραφο του block, εφόσον το block φιλοξενείται στο συγκεκριμένο datanode.

Η κλήση της μεθόδου open() με χρήση του στιγμιότυπου τύπου DistributedFileSystem επιστρέφει στον πελάτη μια αναφορά τύπου FSDataInputStream (ένα ρεύμα εισόδου (input stream), το οποίο υποστηρίζει μετακινήσεις σε τυχαίες θέσεις του αρχείου), για να διαβάσει δεδομένα από το αρχείο. Το στιγμιότυπο τύπου FSDataInputStream με τη σειρά του εμπεριέχει (wraps) ένα στιγμιότυπο τύπου DFSInputStream, το οποίο διαχειρίζεται την επικοινωνία με τα datanodes και το namenode.

Ο πελάτης, στη συνέχεια, καλεί τη μέθοδο read() στο ρεύμα εισόδου (βήμα 3). Τότε το στιγμιότυπο τύπου DFSInputStream, το οποίο έχει αποθηκεύσει τις διευθύνσεις των datanodes, όπου είναι αποθηκευμένα τα πρώτα blocks του αρχείου, συνδέεται με το πρώτο (πιο κοντινό) datanode για το πρώτο block του αρχείου. Τα δεδομένα μεταφέρονται από το datanode στον πελάτη, ο οποίος μπορεί, στη συνέχεια, να καλέσει επανει-

λημμένα τη μέθοδο read() για το ρεύμα εισόδου (βήμα 4). Όταν χρησιμοποιηθούν όλα τα δεδομένα του block από τον πελάτη, το στιγμιότυπο τύπου DFSInputStream θα κλείσει τη σύνδεση με το datanode και έπειτα θα βρει το καλύτερο datanode για τη μεταφορά του επόμενου block (βήμα 5). Όλες αυτές οι λειτουργίες είναι εντελώς διαφανείς προς τον πελάτη, ο οποίος αντιλαμβάνεται απλά ότι διαβάζει μια συνεχή ροή δεδομένων από το αρχείο.

Τα blocks διαβάζονται με τη σειρά, με το στιγμιότυπο τύπου DFSInputStream να δημιουργεί τις κατάλληλες συνδέσεις με τα datanodes, καθώς ο πελάτης διαβάζει δεδομένα από το ρεύμα εισόδου. Επιπλέον, όταν απαιτείται, θα επικοινωνήσει με το namenode, για να ανακτήσει τις διευθύνσεις των datanodes, που έχουν διαθέσιμα τα επόμενα blocks του αρχείου. Όταν ο πελάτης τελειώσει με την ανάγνωση, καλεί τη μέθοδο close() για το στιγμιότυπο τύπου FSDataInputStream (βήμα 6).

Κατά τη διάρκεια της ανάγνωσης, αν το στιγμιότυπο τύπου DFSInputStream αντιμετωπίσει ένα σφάλμα κατά την επικοινωνία με κάποιο datanode, τότε θα προσπαθήσει να επικοινωνήσει με το αμέσως επόμενο κοντινότερο datanode, που διαθέτει το ζητούμενο block. Επιπλέον, καταγράφει τα datanodes, με τα οποία απέτυχε να επικοινωνήσει, ώστε να μην προσπαθήσει ξανά να επικοινωνήσει μαζί τους μελλοντικά για τα επόμενα blocks. Το στιγμιότυπο τύπου DFSInputStream επαληθεύει, επίσης, την ορθότητα των δεδομένων που μεταφέρονται από το datanode με χρήση «αθροισμάτων ελέγχου» (checksums). Αν βρεθεί ένα κατεστραμμένο block, τότε αυτό αναφέρεται στο namenode, πριν γίνει προσπάθεια να διαβαστεί το ίδιο block από άλλο datanode.

Μια σημαντική πτυχή αυτού του σχεδιασμού είναι ότι ο πελάτης επικοινωνεί άμεσα με τα datanodes για την ανάκτηση δεδομένων και καθοδηγείται από το namenode στην εύρεση του κοντινότερου datanode για κάθε block. Ο σχεδιασμός αυτός επιτρέπει στο HDFS να κλιμακώνεται σε έναν μεγάλο αριθμό πελατών, που επιθυμούν ταυτόχρονα να προσπελάσουν αρχεία, επειδή η κίνηση των δεδομένων διαμοιράζεται σε όλα τα datanodes της συστάδας. Εν τω μεταξύ, το namenode πρέπει απλώς να εξυπηρετεί τις αιτήσεις για την εύρεση της τοποθεσίας των blocks (κάτι που μπορεί να κάνει εξαιρετικά αποτελεσματικά, καθώς έχει όλη αυτή τη πληροφορία αποθηκευμένη στη μνήμη του) και δεν χρειάζεται να μεταφέρει δεδομένα, γεγονός που θα προκαλούσε συμφόρηση στο namenode όσο το πλήθος των πελατών αυξανόταν.

8.6.6 Τοπολογία Δικτύου στο Hadoop

Τι σημαίνει δύο κόμβοι σε ένα τοπικό δίκτυο να βρίσκονται «κοντά» ο ένας με τον άλλον; Στο πλαίσιο της επεξεργασίας δεδομένων μεγάλου όγκου, ο παράγοντας που περιορίζει την ταχύτητα επεξεργασίας είναι ο ρυθμός, με τον οποίο μπορούμε να μεταφέρουμε δεδομένα μεταξύ των κόμβων. Με λίγα λόγια, το εύρος ζώνης (bandwidth) του δικτύου θα πρέπει να αξιοποιηθεί προσεκτικά για την επίτευξη καλής απόδοσης. Με βάση την παραπάνω παρατήρηση, η ιδέα είναι να χρησιμοποιηθεί το εύρος ζώνης μεταξύ δύο κόμβων ως μέτρο της απόστασης μεταξύ τους.

Αντί να μετρηθεί το εύρος ζώνης μεταξύ των κόμβων, κάτι που είναι δύσκολο να γίνει στην πράξη (απαιτεί μια συστάδα, στην οποία δεν εκτελούνται εργασίες και, επιπλέον, το πλήθος των ζευγών κόμβων σε μια συστάδα αυξάνεται με το τετράγωνο του αριθμού των κόμβων), το Hadoop ακολουθεί μια άλλη απλή προσέγγιση. Το δίκτυο αναπαρίσταται ως ένα δέντρο και η απόσταση μεταξύ δύο κόμβων είναι το άθροισμα των αποστάσεων τους έως τον πλησιέστερο κοινό πρόγονο τους. Σε ένα τέτοιο δέντρο δεν υπάρχουν προκαθορισμένα επίπεδα, αλλά είναι σύνηθες να ορίζονται επίπεδα, τα οποία αντιστοιχούν στο κέντρο δεδομένων (data center), το ράφι (rack) και τον κόμβο (node) στον οποίο εκτελείται μια διεργασία. Το εύρος ζώνης μεταξύ δύο διεργασιών, που εκτελούνται, θεωρείται ότι μειώνεται για κάθε ένα από τα παρακάτω σενάρια:

- Διεργασίες στον ίδιο κόμβο (μέγιστο εύρος ζώνης).
- Διαφορετικοί κόμβοι στο ίδιο ράφι.
- Κόμβοι σε διαφορετικά ράφια στο ίδιο κέντρο δεδομένων.
- Κόμβοι σε διαφορετικά κέντρα δεδομένων (ελάχιστο εύρος ζώνης).

Για παράδειγμα, φανταστείτε έναν κόμβο n1 στο ράφι r1 στο κέντρο δεδομένων d1. Αν αναπαραστήσουμε την πληροφορία αυτή ως /d1/r1/n1, τότε μπορούμε να υπολογίσουμε την απόσταση για τα παρακάτω τέσσερα σενάρια (βλ. Εικόνα 8.4):

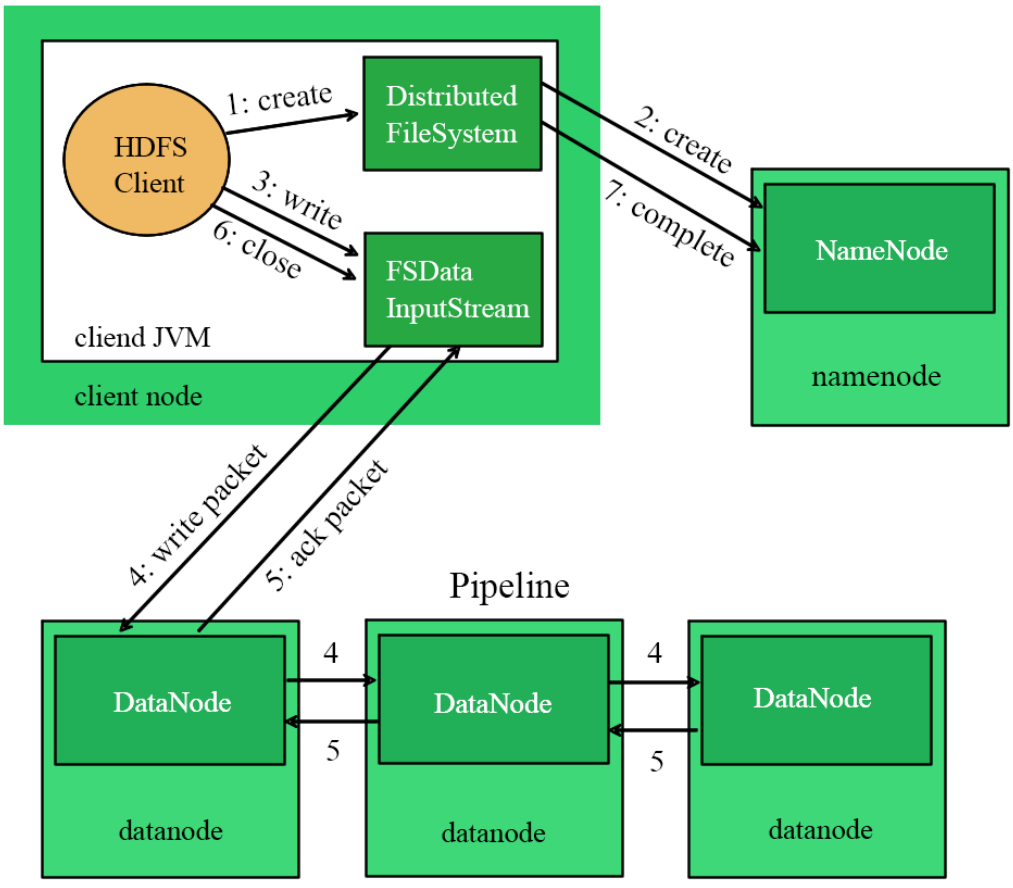
- απόσταση/d1/r1/n1, /d1/r1/n1) = 0 (διεργασίες στον ίδιο κόμβο),
- απόσταση (/d1/r1/n1, /d1/r1/n2) = 2 (διαφορετικοί κόμβοι στο ίδιο ράφι),
- απόσταση /d1/r1/n1, /d1/r2/n3) = 4 (κόμβοι σε διαφορετικά ράφια στο ίδιο κέντρο δεδομένων),

- απόσταση (/d1/r1/n1, /d2/r3/n4) = 6 (κόμβοι σε διαφορετικά κέντρα δεδομένων).

Τέλος, είναι σημαντικό να συνειδητοποιήσουμε ότι το Hadoop δεν μπορεί να εξάγει με κάποιον αυτόματο τρόπο την τοπολογία του δικτύου σας. Εξ ορισμού υποθέτει ότι το δίκτυο είναι επίπεδο, με άλλα λόγια ότι όλοι οι κόμβοι βρίσκονται σε ένα ράφι, ενός κέντρου δεδομένων. Για μικρές συστάδες αυτό μπορεί πράγματι να ισχύει και να μην απαιτείται περαιτέρω διαμόρφωση.

8.6.7 Εγγραφή Αρχείου

Στη συνέχεια, θα μελετήσουμε πώς γίνεται εγγραφή δεδομένων σε αρχεία του HDFS. Αν και η περιγραφή είναι αρκετά λεπτομερής, είναι χρήσιμη για να κατανοήσουμε τη ροή των δεδομένων, καθώς αυτό θα μας βοηθήσει στη συνέχεια να κατανοήσουμε το μοντέλο συνέπειας του HDFS. Συγκεκριμένα, θα μελετήσουμε την περίπτωση της δημιουργίας ενός νέου αρχείου, την εγγραφή δεδομένων σε αυτό και τέλος το κλείσιμο του αρχείου. Θα βασιστούμε στην Εικόνα 8.4 για την περιγραφή.
Ο πελάτης δημιουργεί το αρχείο, καλώντας τη μέθοδο create() για ένα στιγμιότυπο της κλάσης DistributedFileSystem (βήμα 1 στην Εικόνα 8.4). Το στιγμιότυπο τύπου DistributedFileSystem κάνει μια κλήση RPC στο namenode, για να δημιουργήσει ένα νέο αρχείο στον χώρο ονομάτων του συστήματος αρχείων. Το αρχείο αυτό δεν καταλαμβάνει ακόμα κάποιο block (βήμα 2). Το namenode θα προχωρήσει στην πραγματοποίηση διαφόρων ελέγχων, για να βεβαιωθεί ότι το αρχείο δεν υπάρχει ήδη και ότι ο πελάτης έχει τα κατάλληλα δικαιώματα, για να δημιουργήσει το αρχείο. Αν οι έλεγχοι αυτοί είναι επιτυχείς, το namenode δημιουργεί μια εγγραφή για το νέο αρχείο. Διαφορετικά, η δημιουργία του αρχείου αποτυγχάνει και στον πελάτη υψώνεται μια εξαίρεση (raise of an exception) τύπου IOException. Το στιγμιότυπο τύπου DistributedFileSystem επιστρέφει στον πελάτη μια αναφορά για ένα ρεύμα εξόδου (output stream) τύπου FSDataOutputStream, με χρήση του οποίου ο πελάτης μπορεί να αρχίσει να γράφει δεδομένα. Ακριβώς όπως και στην περίπτωση της ανάγνωσης δεδομένων, το FSDataOutputStream εμπεριέχει (wraps) ένα DFSOutputStream, το οποίο διαχειρίζεται την επικοινωνία με τα datanodes και το namenode.



Εικόνα ***8.4*** *Εγγραφή δεδομένων σε αρχείο στο HDFS.*

Καθώς ο πελάτης αρχίζει να γράφει δεδομένα (βήμα 3), το DFSOutputStream χωρίζει τα δεδομένα αυτά σε πακέτα, τα οποία τα καταχωρεί σε μια εσωτερική ουρά, που ονομάζεται «ουρά δεδομένων» (data queue). Οι καταχωρήσεις στην ουρά δεδομένων καταναλώνονται από το DataStreamer, το οποίο είναι υπεύθυνο να ζητάει από το namenode να διαθέσει νέα block για την αποθήκευση των δεδομένων, επιλέγοντας ένα σύνολο από κατάλληλα datanodes για την αποθήκευση των δεδομένων και τη δημιουργία αντιγράφων. Για τη δημιουργία των αντιγράφων, θεωρούμε πως το σύνολο των επιλεγμένων datanodes σχηματίζει έναν αγωγό (pipeline). Θεωρούμε, επίσης, στο παράδειγμα μας ότι τα επίπεδα αντιγραφής είναι τρία, οπότε υπάρχουν τρεις κόμβοι στον αγωγό. Το DataStreamer αποστέλλει τα πακέτα δεδομένων στο πρώτο datanode του αγωγού, το οποίο αποθηκεύει το πακέτο και το προωθεί στο δεύτερο datanode στον αγωγό. Ομοίως, το δεύτερο datanode αποθηκεύει το πακέτο και το προωθεί στο τρίτο (και τελευταίο) datanode στον αγωγό (στάδιο 4).

Το DFSOutputStream διατηρεί, επίσης, μια εσωτερική ουρά των πακέτων, που περιμένουν να γίνει επιβεβαίωση της λήψης τους από τα datanodes. Η ουρά αυτή ονομάζεται «ουρά επιβεβαίωσης» (ack queue). Ένα πακέτο αφαιρείται από την ουρά, μόνο όταν επιβεβαιωθεί η λήψη του από όλα τα datanodes στον αγωγό (βήμα 5).

Εάν ένα datanode υποστεί βλάβη την ώρα που γίνεται εγγραφή δεδομένων σε αυτό, τότε λαμβάνονται οι ακόλουθες δράσεις, οι οποίες είναι διαφανείς προς τον πελάτη, που ζήτησε την εγγραφή των δεδομένων. Πρώτον, ο αγωγός κλείνει και τα πακέτα, που βρίσκονται στην ουρά επιβεβαίωσης, προστίθεται στο μπροστινό μέρος της ουράς δεδομένων. Στόχος είναι τα datanodes, που βρίσκονται στον αγωγό μετά το datanode που υπέστη τη βλάβη, να μην χάσουν πακέτα δεδομένων. Στη συνέχεια, δίνεται μια νέα ταυτότητα (identity) στο τρέχον block στα datanodes που συνεχίζουν να λειτουργούν. Η ταυτότητα αυτή κοινοποιείται, επίσης, στο namenode, έτσι ώστε το τρέχον block στο κατεστραμμένο datanode θα διαγραφεί, εάν το datanode αυτό επανέλθει στη συνέχεια. Το κατεστραμμένο datanode αφαιρείται από τον αγωγό και τα υπόλοιπα δεδομένα του block γράφονται στα υπόλοιπα δύο datanodes, που συνεχίζουν να βρίσκονται στον αγωγό. Τέλος, το namenode θα παρατηρήσει ότι το block δεν έχει αντιγραφεί σε όσα datanodes απαιτείται και θα φροντίσει να δημιουργηθεί ένα ακόμη αντίγραφο σε άλλον κόμβο της συστάδας. Blocks, τα οποία θα δημιουργηθούν στη συνέχεια, διαχειρίζονται κανονικά.

Αν και εξαιρετικά απίθανο, υπάρχει και η περίπτωση να αποτύχουν ταυτόχρονα περισσότερα datanodes, που πρέπει να αποθηκεύσουν ένα block, την ώρα ακριβώς που γράφονται ακόμα δεδομένα σε αυτά. Η καταχώρηση dfs.replication.min στο αρχείο διαμόρφωσης ορίζει το ελάχιστο πλήθος αντιγράφων που πρέπει να υπάρχουν για κάθε block (εξ ορισμού ένα). Όσο υπάρχουν τουλάχιστον τόσα αντίγραφα, το block θα αντιγραφεί κάποια στιγμή ασύγχρονα σε άλλους κόμβους της συστάδας, έως ότου φτάσει το επιθυμητό πλήθος αντιγράφων (εξ ορισμού 3. Καθορίζεται από την εγγραφή dfs.replication στο αρχείο διαμόρφωσης).

Όταν ο πελάτης ολοκληρώσει την εγγραφή δεδομένων, καλεί τη μέθοδο close() για το ρεύμα εξόδου FSDataOutputStream (βήμα 6). Η λειτουργία στέλνει όλα τα υπόλοιπα πακέτα από την ουρά δεδομένων στα datanodes και περιμένει για επιβεβαίωση ότι όλα παρελήφθησαν από τα datanodes. Στη συνέχεια, επικοινωνεί με το namenode, για να το ενημερώσει ότι το αρχείο είναι έτοιμο (βήμα 7). Το namenode ήδη γνωρίζει από ποια blocks αποτελείται το αρχείο (μέσω του DataStreamer, ο οποίος ζητούσε ενδιάμεσα τη δέσμευση νέων blocks). Έτσι, το namenode χρειάζεται μόνο να περιμένει όλα τα blocks να έχουν το ελάχιστο απαιτούμενο πλήθος αντιγράφων, πριν επιστρέψει με επιτυχία.

8.6.8 Τοποθέτηση Αντιγράφων

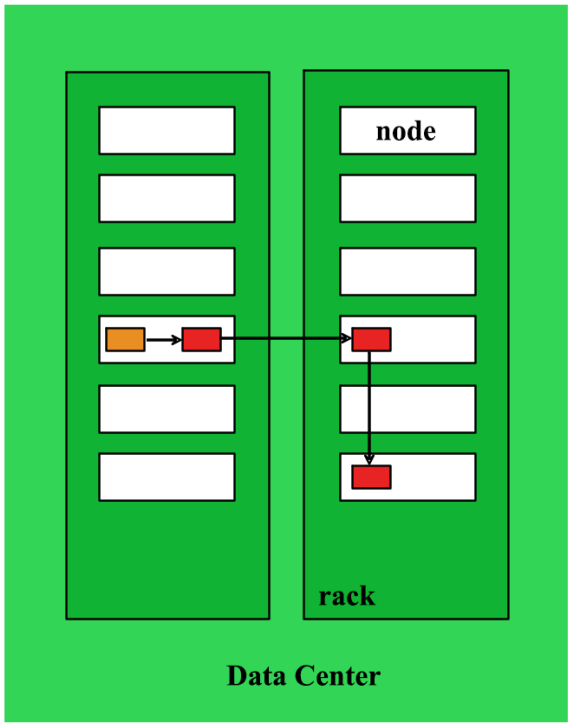
Πώς επιλέγει το namenode τα datanodes, στα οποία θα αποθηκευτούν τα αντίγραφα των blocks; Προφανώς, πρέπει να σταθμιστεί το κόστος ανάμεσα στην αξιοπιστία του συστήματος και το εύρος ζώνης για εγγραφή και ανάγνωση δεδομένων. Για παράδειγμα, τοποθετώντας όλα τα αντίγραφα σε έναν μόνο κόμβο επιτυγχάνεται το ελάχιστο κόστος στο εύρος ζώνης για την εγγραφή δεδομένων, αφού ο αγωγός αντιγραφής εκτελείται σε έναν μόνο κόμβο. Ωστόσο, η προσέγγιση αυτή προφανώς δεν προσφέρει καμία αξιοπιστία (αν ο κόμβος υποστεί βλάβη, τα δεδομένα για το συγκεκριμένο block θα χαθούν). Επίσης, το εύρος ζώνης για την ανάγνωση δεδομένων είναι υψηλό, όταν η ανάγνωση γίνεται από άλλο ράφι. Περνώντας στο άλλο άκρο, δηλαδή τοποθετώντας αντίγραφα σε διαφορετικά κέντρα δεδομένων, μπορούμε να μεγιστοποιήσουμε την αξιοπιστία του συστήματος, με το εύρος ζώνης, όμως, να μειώνεται δραματικά. Ακόμα και στο ίδιο κέντρο δεδομένων (εξάλλου όλες οι σύγχρονες συστάδες Hadoop μέχρι σήμερα αυτή τη διαμόρφωση χρησιμοποιούν) υπάρχει μια ποικιλία από στρατηγικές τοποθέτησης αντιγράφων, που μπορούμε να ακολουθήσουμε. Πράγματι, το Hadoop άλλαξε τη στρατηγική τοποθέτηση αντιγράφων στην έκδοση 0.17.0, ώστε να διατηρεί μια αρκετά ομοιόμορφη κατανομή των blocks στη συστάδα. Επιπλέον, στις εκδόσεις μετά την 1.x, οι πολιτικές τοποθέτησης block είναι δυναμι-

κές (pluggable).

Η εξ ορισμού χρησιμοποιούμενη στρατηγική του Hadoop είναι να τοποθετείται το πρώτο αντίγραφο στον ίδιο κόμβο με τον πελάτη (για τους πελάτες που εκτελούνται εκτός της συστάδας επιλέγεται ένας τυχαίος κόμβος, αν και το σύστημα κάνει μια προσπάθεια να μην επιλεγεί κόμβος, που είναι υπερβολικά απασχολημένος ή ήδη διατηρεί υπερβολικά πολλά δεδομένα). Το δεύτερο αντίγραφο τοποθετείται σε διαφορετικό ράφι από το πρώτο (off-rack), το οποίο επιλέγεται τυχαία. Το τρίτο αντίγραφο τοποθετείται στο ίδιο ράφι με το δεύτερο, αλλά σε διαφορετικό κόμβο που, επίσης, επιλέγεται τυχαία. Περαιτέρω αντίγραφα τοποθετούνται σε τυχαίους κόμβους της συστάδας, με το σύστημα να προσπαθεί να αποφύγει την τοποθέτηση υπερβολικά πολλών αντιγράφων στο ίδιο ράφι.

Μόλις οι θέσεις των αντιγράφων έχουν επιλεγεί, δημιουργείται ο αγωγός, λαμβάνοντας υπόψη την τοπολογία του δικτύου. Για μια διαμόρφωση του συστήματος, όπου απαιτούνται 3 αντίγραφα, ο αγωγός θα μπορούσε να είναι, όπως αυτός στην Εικόνα 8.5.

Συνολικά, η στρατηγική αυτή προσφέρει καλή ισορροπία μεταξύ αξιοπιστίας (τα blocks αποθηκεύονται σε δύο ράφια), εύρος ζώνης εγγραφής (οι εγγραφές πρέπει να περάσουν μόνο από έναν διακόπτη [network switch]), εύρος ζώνης ανάγνωσης (υπάρχει επιλογή μεταξύ δύο ραφιών για ανάγνωση) και την κατανομή block στη συστάδα (οι πελάτες γράφουν μόνο ένα block στο τοπικό rack).



Εικόνα *8.5* *Ένας τυπικός αγωγός δημιουργίας αντιγράφων.*

8.6.9 Μοντέλο Συνέπειας

Ένα μοντέλο συνέπειας για ένα σύστημα αρχείων περιγράφει την ορατότητα των δεδομένων, που διαβάζονται από ή γράφονται σε ένα αρχείο. Το HDFS ανταλλάσσει μερικές από τις αυστηρές απαιτήσεις του POSIX με την επίτευξη καλύτερης απόδοσης σε ένα κατανεμημένο σύστημα αρχείων. Κατά συνέπεια, ορισμένες λειτουργίες πάνω σε αρχεία ενδέχεται να συμπεριφέρονται διαφορετικά απ' ό,τι θα περιμέναμε.

Μετά τη δημιουργία ενός αρχείου, αυτό είναι ορατό στον χώρο ονομάτων αρχείων, όπως και αναμένεται. Ωστόσο, οποιαδήποτε δεδομένα εγγραφούν στο αρχείο δεν είναι εγγυημένο ότι θα είναι άμεσα ορατά, ακόμα και αν γίνει άδειασμα (flush) του ρεύματος εξόδου. Έτσι, το αρχείο φαίνεται να έχει μηδενικό μέγεθος.

Μόλις εγγραφούν τα δεδομένα ενός ολόκληρου block, τότε αυτό θα είναι ορατό για ανάγνωση από τους πελάτες-αναγνώστες του αρχείου. Το ίδιο ισχύει και για τα επόμενα blocks: πάντα το τρέχον block, στο οποίο γίνονται εγγραφές, δεν είναι ορατό από τους πελάτες-αναγνώστες του αρχείου.

Το HDFS παρέχει, ωστόσο, τη μέθοδο sync(), η οποία εκτελείται για το ρεύμα εξόδου FSDataOutputStream και υποχρεώνει όλα τα δεδομένα να συγχρονιστούν σε όλα τα datanodes. Μετά από μια επιτυχημένη επιστρο-

φή από τη μέθοδο αυτή το HDFS εγγυάται ότι τα δεδομένα, που έχουν εγγραφεί μέχρι εκείνο το χρονικό σημείο στο αρχείο, έχουν φτάσει σε όλα τα datanodes στον αγωγό εγγραφής και είναι ορατά σε όλους τους πελάτες-αναγνώστες. Κατά το κλείσιμο ενός αρχείου, επίσης, εκτελείται έμμεσα η μέθοδος sync() για το συγκεκριμένο αρχείο. Η συμπεριφορά αυτή είναι παρόμοια με την κλήση συστήματος fsync() του προτύπου POSIX, η οποία υποχρεώνει να εγγραφούν στον σκληρό δίσκο τα δεδομένα που διατηρούνται σε τμήματα μνήμης προσωρινής αποθήκευσης (buffer) για έναν περιγραφέα (descriptor) αρχείου.

Το παραπάνω μοντέλο συνοχής έχει σημαντικές συνέπειες στον τρόπο που σχεδιάζονται εφαρμογές. Χωρίς κλήση της μεθόδου sync() θα πρέπει να είμαστε προετοιμασμένοι να χάσουμε μέχρι ένα block δεδομένων σε περίπτωση αποτυχίας ενός πελάτη ή ενός κόμβου. Για πολλές εφαρμογές κάτι τέτοιο θα ήταν απαράδεκτο, οπότε θα πρέπει να καλείται η sync() σε κατάλληλα σημεία στον κώδικα της εφαρμογής, όπως μετά την εγγραφή ενός ορισμένου πλήθους εγγραφών ή πλήθους bytes. Αν και η μέθοδος sync() έχει σχεδιαστεί, έτσι ώστε να μην έχει υπερβολικά μεγάλο κόστος, ωστόσο απαιτεί αρκετό χρόνο για να ολοκληρωθεί. Κατά συνέπεια, θα πρέπει να υπάρχει μια στάθμιση μεταξύ της ορθότητας των εγγραφόμενων δεδομένων και της απόδοσης της εφαρμογής. Τι συνιστά μια αποδεκτή στάθμιση εξαρτάται προφανώς από το είδος της εφαρμογής και κατάλληλες τιμές μπορούν να επιλεγούν μετά από μέτρηση της απόδοσης της εφαρμογής με διαφορετικές συχνότητες χρήσης της sync().

8.7 Η Αρχιτεκτονική Hadoop Cluster

Πάνω από το σύστημα αρχείων του Hadoop, βρίσκεται η μηχανή MapReduce, η οποία αποτελείται από έναν master Jobtracker (ανιχνευτής εργασίας), στον οποίο οι εφαρμογές υποβάλλουν MapReduce εργασίες. Ο Jobtracker ωθεί τις εργασίες στους διαθέσιμους Tasktrackers του cluster, όσο το δυνατόν πιο κοντά στα δεδομένα.

Σε ένα Hadoop cluster συνήθως υπάρχει ένας κόμβος που τρέχει το namenode, ένας κόμβος που τρέχει τον Jobtracker και αρκετές μηχανές, οι οποίες τρέχουν ένα datanode και έναν Tasktracker.

Είναι δυνατόν η εγκατάσταση των datanodes και των tasktrackers να γίνει σε διαφορετικές μηχανές, αλλά με τέτοιον τρόπο, ώστε να μη χάσουμε τη βελτιστοποίηση rack awareness (η οποία λαμβάνει υπόψη τη γεωγραφική συγκέντρωση των κεντρικών υπολογιστών), που μας παρέχει το Hadoop, όπου ο Jobtracker ξέρει ποιος κόμβος κατέχει τα δεδομένα προς επεξεργασία και ποιοι άλλοι κόμβοι είναι κοντά του και έτσι στέλνει την επεξεργασία σε αυτούς. Αυτή η βελτιστοποίηση βασίζεται στην αρχή εκείνη που θεωρεί πιο συμφέρουσα, από άποψη χρόνου εκτέλεσης, τη μετακίνηση της επεξεργασίας παρά τη μετακίνηση των δεδομένων.

Ο χρήστης στέλνει τις εντολές του στον jobtracker, ο οποίος τις τοποθετεί σε μια σειρά από εργασίες προς εκτέλεση και τις εξυπηρετεί με την πολιτική FIFO (First In First Out) στέλνοντας τις στους tasktrackers για εκτέλεση. Ο κάθε tasktracker εκτελεί απλώς τις εργασίες που του αναθέτει ο Jobtracker. Με ένα rack-awareness σύστημα αρχείων, ο Jobtracker γνωρίζει ποιος κόμβος περιέχει τα δεδομένα και ποιες άλλες μηχανές είναι κοντά.

Εάν ένας Tasktracker αποτύχει, το μέρος εκείνο της εργασίας επαναπρογραμματίζεται. Εάν ο Jobtracker αποτύχει, όλη η εργασία χάνεται και πρέπει να υποβληθεί εκ νέου. Δεν υπάρχει κανένα σημείο ελέγχου ή αποκατάστασης σε μια MapReduce εργασία. Εάν ένας Tasktracker είναι πολύ αργός, μπορεί να καθυστερήσει ολόκληρη τη λειτουργία.

8.8 API-Προγραμματιστική Διεπαφή του Hadoop σε Java

Για να είναι σε θέση ένα πρόγραμμα να χρησιμοποιεί το Hadoop, θα πρέπει να χρησιμοποιηθεί κάποια γλώσσα προγραμματισμού. Το Hadoop προσφέρει προγραμματιστικές διεπαφές (Application Programming Interface-API) για την ανάπτυξη προγραμμάτων σε διάφορες γλώσσες προγραμματισμού. Ωστόσο, η πλέον διαδεδομένη χρήση είναι μέσω της γλώσσας Java.

Χρησιμοποιώντας ένα κλασσικό παράδειγμα για την εισαγωγή στον προγραμματισμό σε Hadoop, θα αναπτύξουμε τις δυνατότητες που μας προσφέρει η προγραμματιστική διεπαφή του Hadoop σε Java. Πιο συγκεκριμένα, θα χρησιμοποιήσουμε το παράδειγμα της Εικόνας 8.1. Στο παράδειγμα αυτό δίνεται ως είσοδος ένα αρχείο κειμένου και παράγεται ως έξοδος ένα νέο αρχείο κειμένου, το οποίο περιέχει πληροφορία για το πόσες φορές υπάρχει κάθε λέξη στο αρχείο εισόδου. Επομένως, το αρχείο εξόδου του προγράμματος περιέχει γραμμές της μορφής:

<Λέξη><Αριθμός εμφανίσεων>

Θα πρέπει, επίσης, να διευκρινιστεί ότι λέξη θεωρείται οποιοδήποτε αλφαριθμητικό μεταξύ χαρακτήρων διαχωρισμού, οι οποίοι περιλαμβάνουν το κενό, τους στηλοθέτες (tab), τον χαρακτήρα αλλαγής γραμμής κ.λπ. Για παράδειγμα, στο κείμενο "this is an example (show me)", λέξεις θεωρούνται τα αλφαριθμητικά "this", "is", "an", "example", "(show" και "me)". Προφανώς μια πιο ακριβής καταμέτρηση (π.χ. να μην λαμβάνουμε υπ' όψη μας τις παρενθέσεις) θα περιέπλεκε αρκετά το παράδειγμα μας και θα μας αποσπούσε την προσοχή από τα βασικά σημεία του προγραμματισμού με Hadoop. Κατά συνέπεια, δεν συμπεριλαμβάνονται στο παράδειγμα πιο λεπτομερείς έλεγχοι.

Στο παράδειγμα παρατηρούμε πως ο κώδικας αποτελείται από ένα σύνολο διακριτών μερών. Το πρώτο μέρος (γραμμές 1-11) ενημερώνει τον μεταγλωττιστή της Java για το ποιες κλάσεις θα χρησιμοποιηθούν στο πρόγραμμα. Οι κλάσεις αυτές αποτελούν μέρος της βιβλιοθήκης της Java και του Hadoop. Σκοπός των κλάσεων αυτών είναι να προσφέρουν έτοιμη λειτουργικότητα στον προγραμματιστή. Έτσι, δεν είναι απαραίτητο ο προγραμματιστής να γράφει εκ νέου κώδικα για κάθε λειτουργία που απαιτεί το πρόγραμμα του. Στη συνέχεια (γραμμή 13), ορίζεται μια νέα κλάση για το πρόγραμμά μας. Υπενθυμίζεται πως το όνομα του αρχείου, στο οποίο σώζεται ένα πρόγραμμα Java, πρέπει να έχει το ίδιο όνομα με την public κλάση που ορίζει. Κατά συνέπεια, το πρόγραμμά μας θα πρέπει να αποθηκευτεί σε ένα αρχείο με όνομα "WordCount.java" και δεν μπορεί να περιέχει περισσότερους ορισμούς public κλάσεων.

Αυτό, ωστόσο, δεν απαγορεύει τον ορισμό μη public κλάσεων μέσα στο ίδιο αρχείο. Παρατηρούμε πως στο παράδειγμά μας ορίζονται οι μη public κλάσεις "Map" και "Reduce" (γραμμές 15-28 και 30-39 αντίστοιχα). Είναι προφανές πως οι κλάσεις αυτές περιλαμβάνουν τα πεδία και τις μεθόδους που πραγματοποιούν την κύρια εργασία του προγράμματος. Κατά συνέπεια, θα επανέλθουμε σε αυτές.

```
import java.io.IOException;
import java.util.*;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import  org.apache.hadoop.mapreduce.lib.output.FileOutputFor-
mat;
import  org.apache.hadoop.mapreduce.lib.output.TextOutputFor-
mat;

public class WordCount {

  public static class Map extends Mapper<LongWritable, Text,
Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(LongWritable key, Text value, Context con-
text)
    throws IOException, InterruptedException {
```

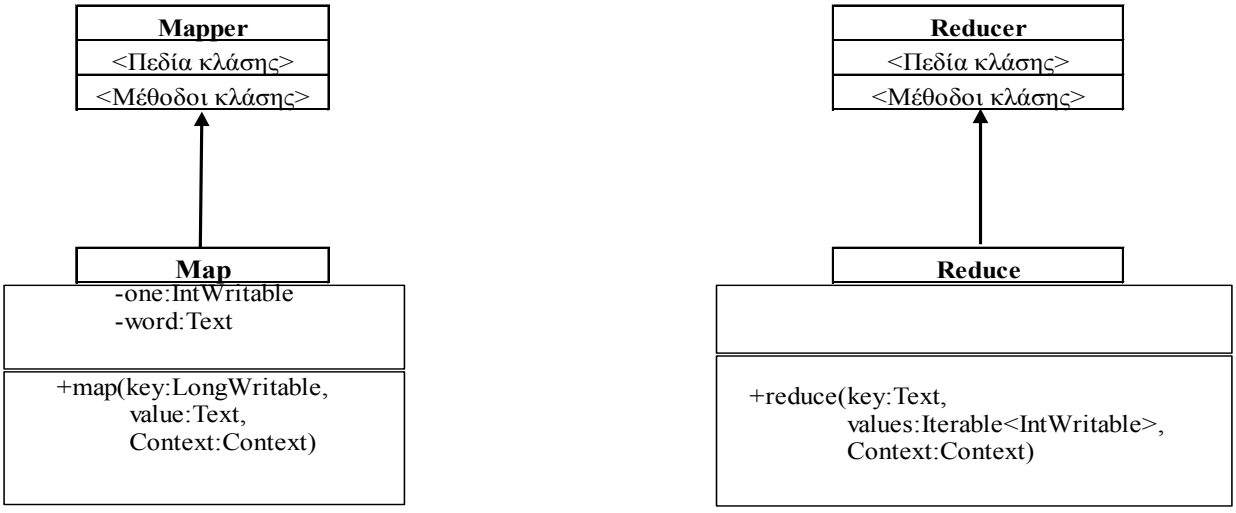
```
      String line = value.toString();
      StringTokenizer tokenizer = new StringTokenizer(line);
      while (tokenizer.hasMoreTokens()) {
          word.set(tokenizer.nextToken());
          context.write(word, one);
      }
    }
  }

  public static class Reduce extends Reducer<Text, IntWritable,
Text, IntWritable> {
    public void reduce(Text key, Iterable<IntWritable> values,
Context context)
    throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
          sum += val.get();
      }
      context.write(key, new IntWritable(sum));
    }
  }

 public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();

    Job job = new Job(conf, "wordcount");
    job.setJarByClass(WordCount.class);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

    job.setMapperClass(Map.class);
    job.setReducerClass(Reduce.class);

    job.setInputFormatClass(TextInputFormat.class);
    job.setOutputFormatClass(TextOutputFormat.class);
```

```
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        job.waitForCompletion(true);
    }
}
```

Κώδικας *8.1* *Κώδικας παραδείγματος σε Java.*

Το τελευταίο τμήμα του προγράμματος μας (γραμμές 41-60) αποτελείται από τη μέθοδο "main()", από την οποία ξεκινάει η εκτέλεση του προγράμματος. Στη μέθοδο αυτή γίνεται η αρχικοποίηση και η παραμετροποίηση των εργασιών που θα πρέπει να εκτελέσει το Hadoop. Θα επανέλθουμε και σε αυτήν τη μέθοδο με περισσότερες λεπτομέρειες.

Οι δύο βασικές κλάσεις, που παρέχει η προγραμματιστική διεπαφή του Hadoop για Java, είναι οι abstract κλάσεις "Mapper" και "Reducer". Υπενθυμίζεται πως δεν επιτρέπεται να δημιουργούνται στιγμιότυπα από abstract κλάσεις. Επομένως, οι παραπάνω κλάσεις μπορούν να χρησιμοποιηθούν μόνο σε μια ιεραρχία κλάσεων. Δηλαδή, χρησιμοποιούμε τις κλάσεις αυτές μόνο για να τις επεκτείνουμε και να δημιουργήσουμε νέες κλάσεις. Αυτό φαίνεται στο παράδειγμα μας, όπου στη γραμμή 15 δημιουργούμε την κλάση "Map", επεκτείνοντας ("extends") την κλάση "Mapper". Αντίστοιχα, στη γραμμή 30 δημιουργούμε την κλάση "Reduce", επεκτείνοντας την κλάση "Reducer". Σχηματικά, αυτό μπορούμε να το αναπαραστήσουμε, όπως στην Εικόνα 8.6 που ακολουθεί.



Εικόνα 8.6 *Δημιουργία των κλάσεων "Map" και "Reduce" μέσω επέκτασης των κλάσεων "Mapper" και "Reducer" αντίστοιχα.*

Η κλάση "Mapper", όπως εξάλλου και η "Reducer", είναι παραμετρική. Μέσω αυτής της ιδιότητας μας δίνεται η δυνατότητα να ορίσουμε εμείς τους τύπους δεδομένων των πεδίων και των τιμών των εκάστοτε κλάσεων που δημιουργούμε. Στο παράδειγμα της Εικόνας 8.4 από τη γραμμή 15 φαίνεται πως η κλάση "Mapper" δέχεται 4 παραμέτρους. Οι παράμετροι βρίσκονται μεταξύ των συμβόλων "<" και ">" και διαχωρίζονται με κόμμα. Οι δύο πρώτες παράμετροι υποδεικνύουν τον τύπο δεδομένων του κλειδιού εισόδου (input key) και της τιμής εισόδου (input value). Οι δύο τελευταίες παράμετροι υποδεικνύουν τον τύπο δεδομένων του κλειδιού αποτελέσματος (output key) και της τιμής αποτελέσματος (output value). Παρατηρήστε πως το Hadoop

ορίζει τους δικούς του τύπους δεδομένων, οι οποίοι αντιστοιχούν σε τύπους δεδομένων της Java. Για παράδειγμα, οι τύπου δεδομένων LongWritable, IntWritable και Text αντιστοιχούν στους τύπους δεδομένων long, int και String της Java. Ωστόσο, οι τύποι δεδομένων του Hadoop είναι βελτιστοποιημένοι για την αποδοτική μεταφορά τους μέσω δικτύου. Σε κάθε περίπτωση, θα πρέπει να χρησιμοποιούνται οι τύποι δεδομένων του Hadoop. Οι τύποι δεδομένων του Hadoop είναι ορισμένοι σε κλάσεις, που περιέχονται στο πακέτο (package) org.apache.hadoop.io. Αυτός είναι ο λόγος που στη γραμμή 6 ενημερώνεται ο μεταγλωττιστής ότι πρέπει να χρησιμοποιηθούν οι κλάσεις του συγκεκριμένου πακέτου.

Στην περίπτωσή μας, οι παράμετροι εισόδου έχουν οριστεί να είναι τύπου LongWritable και Text για το κλειδί και την τιμή εισόδου αντίστοιχα. Όπως θα δούμε και στη συνέχεια, το κλειδί εισόδου δεν χρησιμοποιείται στο παράδειγμά μας και κατά συνέπεια θα μπορούσε να είναι οποιουδήποτε τύπου. Ωστόσο, η τιμή εισόδου διαβάζεται από το αρχείο κειμένου που δίνουμε στο πρόγραμμα ως είσοδο. Άρα, πρέπει να είναι τύπου Text.

Σε ό,τι αφορά το κλειδί και την τιμή αποτελέσματος χρησιμοποιούνται οι τύποι Text και IntWritable. Αυτό είναι λογικό καθώς, όπως προαναφέρθηκε, η έξοδος θέλουμε να είναι της μορφής <Λέξη> <Αριθμός εμφανίσεων>, που είναι κείμενο και αριθμός αντίστοιχα.

Για την κλάση "Reducer" ισχύουν τα αντίστοιχα σε ό,τι αφορά τις παραμέτρους. Θα πρέπει, όμως, να δοθεί έμφαση στο εξής σημείο: η είσοδος στις εργασίες Συγχώνευσης (Reducers) είναι η έξοδος των εργασιών Αντιστοίχισης (Mappers). Κατά συνέπεια, οι παράμετροι εισόδου στην κλάση "Reducer" (δηλαδή, οι δύο πρώτες παράμετροι της "Reducer") θα πρέπει να είναι του ιδίου τύπου με αυτούς των παραμέτρων εξόδου της κλάσης "Mapper" (δηλαδή, των δύο τελευταίων παραμέτρων της "Mapper"). Πράγματι, στο παράδειγμά μας φαίνεται πως υπάρχει αυτή η αντιστοιχία. Από την άλλη, οι παράμετροι εξόδου της κλάσης "Reducer" μπορούν να είναι οποιουδήποτε τύπου, ανάλογα με τις ανάγκες του προβλήματος που λύνουμε. Στην περίπτωσή μας θέλουμε απλά να συνδυάσουμε/προσθέσουμε το πλήθος εμφανίσεων κάθε λέξης, που υπολόγισε κάθε εργασία Αντιστοίχισης. Επομένως, οι παράμετροι εξόδου θα πρέπει να είναι Text και IntWritable.

Μετά τον ορισμό των κατάλληλων κλάσεων ως επεκτάσεις των κλάσεων "Mapper" και "Reducer", το επόμενο βήμα είναι ο ορισμός των κατάλληλων μεθόδων εντός των κλάσεων αυτών. Οι μέθοδοι αυτές είναι υπεύθυνες για την εκτέλεση των λειτουργιών, οι οποίες απαιτούνται για την επίλυση του προβλήματος που διαπραγματευόμαστε κάθε φορά. Πιο συγκεκριμένα, η κλάση "Mapper" απαιτεί την ύπαρξη μιας μεθόδου "map()". Στην κλάση "Map" θα πρέπει, επομένως, να υπερκαλύψουμε (override) τον ορισμό της μεθόδου "map()" στην κλάση "Mapper". Αυτό φαίνεται στην γραμμή 19, όπου ορίζουμε τη μέθοδο "map()" εντός της κλάσης "Map". Παρατηρήστε πως οι τύποι δεδομένων των παραμέτρων key και value της "map()" αντιστοιχούν στους τύπους δεδομένων των δύο πρώτων παραμέτρων της κλάσης "Mapper" (LongWritable και Text αντίστοιχα). Η παράμετρος context παρέχεται από το Hadoop και χρησιμεύει για να γράφουμε τα δεδομένα που παράγει κάθε εργασία Αντιστοίχισης, ώστε αυτά να μπορούν να χρησιμοποιηθούν, στη συνέχεια, από τις εργασίες Συγχώνευσης.

Παρατηρήστε πως για τη μέθοδο "map()" δεν γνωρίζουμε εκ των προτέρων ποια δεδομένα από το αρχείο εισόδου θα ανατεθούν σε κάθε εργασία Αντιστοίχισης. Γνωρίζουμε μόνο πως αυτά θα είναι κάποιο τμήμα του κειμένου εισόδου, το οποίο παρέχεται ως είσοδος στη μέθοδο, δηλαδή είναι η παράμετρος value.

Στο παράδειγμα καταμέτρησης λέξεων εκμεταλλευόμαστε την κλάση "StringTokenizer" που μας παρέχει η Java. Η λειτουργία της είναι να παίρνει μια μεταβλητή τύπου String και να εξάγει από αυτήν μία προς μία τις λέξεις που την αποτελούν. Για τον λόγο αυτό, μετατρέπουμε πρώτα τα δεδομένα εισόδου που είναι τύπου Text σε String (γραμμή 21) και στη συνέχεια επεξεργαζόμαστε τα δεδομένα εισόδου με τη νέα τους μορφή.

Η επανάληψη τύπου "while", που ακολουθεί, ελέγχει, αν υπάρχουν ακόμα δεδομένα εισόδου στη μεταβλητή τύπου String, που δημιουργήσαμε στη γραμμή 21. Όσο υπάρχουν ακόμα λέξεις, εξάγει την επόμενη, τη μετατρέπει ξανά σε τύπο Text (αφού το Hadoop με αυτόν τον τύπο μπορεί να επεξεργαστεί τα δεδομένα) και στέλνει το αποτέλεσμα στην παράμετρο context, ώστε να αποθηκευτεί η εμφάνιση της συγκεκριμένης λέξης. Παρατηρήστε πως τα δεδομένα, που στέλνονται στην context, έχουν τύπους δεδομένων αντίστοιχους των δύο τελευταίων παραμέτρων της κλάσης "Mapper". Τέλος, αξίζει να αναφερθεί πως κάθε εργασία Αντιστοίχισης θα εκτελέσει τις ίδιες εντολές στο τμήμα των δεδομένων που (αυτόματα) της έχει ανατεθεί από το Hadoop.

Αντίστοιχα, πρέπει να υπερκαλύψουμε τη μέθοδο "reduce()" στην κλάση "Reduce". Πάλι οι παράμετροι key και values έχουν τύπους δεδομένων αντίστοιχους των δύο πρώτων παραμέτρων της κλάσης "Reducer". Μια σημαντική διαφορά σε σχέση με την "map()" είναι το γεγονός πως η παράμετρος values είναι πιο συγκεκριμένα τύπου "Iterable<IntWritable>". Ο τύπος "Iterable" είναι ένας τύπος πίνακα ή αλλιώς λίστας, που παρέχει το Hadoop και στο παράδειγμά μας ο πίνακας αυτός περιέχει τιμές τύπου IntWritable. Ο ορισμός αυτός είναι λογικός για τον εξής λόγο: Φανταστείτε πως κάθε εργασία Αντιστοίχισης βρήκε τη λέξη "the" στο

τμήμα του κειμένου που τις ανατέθηκε. Αυτό σημαίνει πως κάθε τέτοια εργασία έχει δημιουργήσει ένα ζεύγος του τύπου <"the", 1> για κάθε εμφάνιση της λέξης "the". Αν μια εργασία Αντιστοίχισης έχει βρει 5 φορές τη λέξη αυτή στο κείμενο που της δόθηκε προς επεξεργασία, θα έχει δημιουργήσει 5 τέτοια ζεύγη. Όμως, σε κάθε εργασία Συγχώνευσης μπορεί να δοθεί προς επεξεργασία το αποτέλεσμα περισσότερων εργασιών Αντιστοίχισης με το ίδιο κλειδί (key), που κάθε μια μπορεί να έχει δημιουργήσει πολλαπλά ζεύγη κλειδιού-τιμής. Σε κάθε περίπτωση, το κλειδί είναι κοινό. Όμως, θα πρέπει με κάποιον τρόπο να διατηρηθούν όλες οι τιμές που δίνονται στην εργασία Συγχώνευσης, ώστε αυτή να αποφασίσει τι θα κάνει με τις τιμές αυτές. Στο παράδειγμά μας, περνάμε από όλα τα στοιχεία του πίνακα values και προσθέτουμε σε μια μεταβλητή όλες τις τιμές, δηλαδή βρίσκουμε τελικά πόσες φορές εμφανίστηκε το συγκεκριμένο κλειδί.

Θα πρέπει, επίσης, να αναφερθούμε στο τμήμα του κώδικα που αρχικοποιεί και παραμετροποιεί την εργασία και τη στέλνει στον Ιχνηλάτη Εργασιών (JobTracker). Η περιγραφή της εργασίας γίνεται μέσω μιας ακόμα κλάσης που προσφέρει το Hadoop και συγκεκριμένα την "Job". Για τη δημιουργία ενός στιγμιότυπου της κλάσης "Job" (γραμμή 44) χρησιμοποιούμε έναν δημιουργό (constructor) με δύο παραμέτρους. Η πρώτη παράμετρος είναι ένα στιγμιότυπο της κλάσης "Configuration" (γραμμή 42), ενώ η δεύτερη παράμετρος είναι το όνομα που θέλουμε να δώσουμε στην εργασία. Το στιγμιότυπο της κλάσης "Configuration" περιέχει πληροφορίες από την παραμετροποίηση που έχει γίνει στο Hadoop από τον διαχειριστή του συστήματος και πιθανόν από τον ίδιο τον χρήστη του Hadoop με τη δημιουργία κατάλληλων αρχείων. Ωστόσο, το θέμα αυτό δεν θα μας απασχολήσει περαιτέρω εδώ.

Στη συνέχεια, καλούνται ορισμένες μέθοδοι της κλάσης "Job", οι οποίες παραμετροποιούν την εργασία που θα υποβληθεί στον Ιχνηλάτη Εργασιών. Πιο συγκεκριμένα, οι "setOutputKeyClass()" και "setOutputValueClass()" ορίζουν τους τύπους δεδομένων για τα ζεύγη κλειδιού-τιμής του τελικού αποτελέσματος. Οι "setMapperClass()" και "setReducerClass()" ορίζουν ποιες κλάσεις του προγράμματός μας θα χρησιμοποιηθούν για τις εργασίες Αντιστοίχισης και Συγχώνευσης αντίστοιχα. Οι "setInputFormatClass()" και "setOutputFormatClass()" ορίζουν τον τύπο δεδομένων για τα στοιχεία εισόδου και εξόδου αντίστοιχα. Στο παράδειγμά μας τόσο η είσοδος όσο και η έξοδος του προγράμματος μας είναι ένα αρχείο κειμένου, επομένως, χρησιμοποιείται η κλάση "TextInputFormat.class" και "TextOutputFormat.class" αντίστοιχα. Τέλος, η "waitForCompletion()" υποβάλλει την εργασία στον Ιχνηλάτη Εργασιών και περιμένει την ολοκλήρωση της εργασίας.

Για να εξηγήσουμε τις εναπομείνασες μεθόδους (γραμμές 56-57) θα πρέπει πρώτα να εξηγήσουμε με ποιον τρόπο μεταγλωττίζουμε και εκτελούμε ένα πρόγραμμα Hadoop. Η παρακάτω μέθοδος είναι η πλέον συνηθισμένη που ακολουθείται:

1. Δημιουργούμε έναν υποκατάλογο, στον οποίο σώζουμε το αρχείο "WordCount.java". Ο κατάλογος αυτός μπορεί να έχει οποιοδήποτε όνομα, αλλά συνίσταται να έχει το ίδιο όνομα με το αρχείο:

```
mkdir WordCount
```

2. Μεταφερόμαστε στον υποκατάλογο "WordCount":

```
cd WordCount
```

3. Μέσα στον υποκατάλογο "WordCount" δημιουργούμε δύο ακόμα υποκαταλόγους:

```
mkdir classes
```

```
mkdir jar
```

Στον υποκατάλογο "classes" θα τοποθετούνται τα μεταγλωττισμένα αρχεία της εφαρμογής μας. Στον υποκατάλογο "jar" θα τοποθετείται μια βιβλιοθήκη χρόνου εκτέλεσης (run-time library), που θα φτιάχνουμε από τα αρχεία του υποκαταλόγου "classes".

4. Η μεταγλώττιση του προγράμματός μας γίνεται δίνοντας την παρακάτω εντολή:

```
javac -cp `hadoop classpath` -d classes WordCount.java
```

5. Η δημιουργία της βιβλιοθήκης χρόνου εκτέλεσης γίνεται δίνοντας την παρακάτω εντολή:

```
jar –cvf jar/WordCount.jar –C classes/.
```

6. Η εκτέλεση του προγράμματος μας γίνεται δίνοντας την παρακάτω εντολή:

```
hadoop jar jar/WordCount.jar WordCount <InputFile> <OutputDir>
```

Παρατηρούμε ότι για την εκτέλεση του προγράμματός μας πρέπει να δώσουμε δύο επιπλέον παραμέτρους, το αρχείο εισόδου και έναν κατάλογο στο σύστημα αρχείων HDFS, στον οποίο θα τοποθετηθούν τα αποτελέσματα της εκτέλεσης του προγράμματός μας.

Οι παράμετροι αυτοί περνάνε στο πρόγραμμα Java στον πίνακα "args" (γραμμή 41) και μπορούμε να τις διαχειριστούμε μέσω αυτού του πίνακα. Επομένως, το τι είναι η κάθε παράμετρος καθορίζεται από το ίδιο το πρόγραμμα. Αφού η πρώτη παράμετρος (θέση 0 στον πίνακα "args") θέλουμε να είναι αρχείο εισόδου, θα πρέπει να το δηλώσουμε:

```
FileInputFormat.addInputPath(job, new Path(args[0]));
```

Αντίστοιχα, αφού θέλουμε η δεύτερη παράμετρος (θέση 1 στον πίνακα "args") να είναι ο κατάλογος εξόδου, θα πρέπει και αυτό να το δηλώσουμε:

```
FileOutputFormat.setOutputPath(job, new Path(args[1]));
```

Σε ένα πρόγραμμα Hadoop επιτρέπεται να θέσουμε μόνο έναν κατάλογο εξόδου. Ωστόσο, επιτρέπεται να έχουμε περισσότερα αρχεία εισόδου. Αν, για παράδειγμα, θέλαμε να έχουμε τρία αρχεία εισόδου στο πρόγραμμα μας και έναν κατάλογο εξόδου θα μπορούσαμε να καλέσουμε το πρόγραμμα μας ως εξής:

```
hadoop jar jar/WordCount.jar WordCount <In1> <In2> <In3> <Out-
putDir>
```

και ο αντίστοιχος κώδικας Java στη μέθοδο "main()" θα έπρεπε να είναι:

```
FileInputFormat.addInputPath(job, new Path(args[0]));
FileInputFormat.addInputPath(job, new Path(args[1]));
FileInputFormat.addInputPath(job, new Path(args[2]));
FileOutputFormat.setOutputPath(job, new Path(args[3]));
```

Φυσικά, μπορούμε να αλλάξουμε τη σειρά των παραμέτρων, όπως εμείς επιθυμούμε. Αρκεί να υπάρχει αντιστοιχία στον κώδικα Java. Αν, για παράδειγμα, θέλαμε ο κατάλογος εξόδου να είναι η πρώτη παράμετρος, τότε θα έπρεπε να καλέσουμε το πρόγραμμα μας ως εξής:

```
hadoop jar jar/WordCount.jar WordCount <OutputDir> <In1> <In2>
<In3>
```

και ο κώδικας Java θα έπρεπε να τροποποιηθεί ως εξής:

```
FileOutputFormat.setOutputPath(job, new Path(args[0]));
FileInputFormat.addInputPath(job, new Path(args[1]));
FileInputFormat.addInputPath(job, new Path(args[2]));
FileInputFormat.addInputPath(job, new Path(args[3]));
```

8.9 Βρόχοι για Λίστες & Generic Κλάσεις και Μέθοδοι

Στην παρούσα ενότητα παρουσιάζονται φαινόμενα που ήδη συναντήσατε νωρίτερα στο παράδειγμα της Εικόνας 8.1. Σκοπός της παραγράφου αυτής είναι να δώσει περισσότερες λεπτομέρειες για τα επιμέρους θέματα.

Είδαμε νωρίτερα έναν δυναμικό πίνακα ή αλλιώς μία λίστα τύπου "Iterable<IntWritable>". Ο τύπος "Iterable" είναι ένας τύπος λίστας που παρέχει το Hadoop και στο παράδειγμα περιέχει τιμές τύπου IntWritable. Γενικά, τέτοιες λίστες είναι της γενικής Iterable<T>. Στις λίστες αυτές αποθηκεύονται αντικείμενα τύπου T.

Η μέθοδος iterator(), που ορίζεται στη διεπαφή Iterable<T>, επιστρέφει ένα αντικείμενο τύπου Iterator, το οποίο διατρέχει τη λίστα από την αρχή έως το τέλος της. Φανταστείτε τον iterator ως έναν δείκτη μεταξύ δύο στοιχείων της λίστας, που μπορεί να τη διαπερνά και προς τα μπροστά και προς τα πίσω. Η Iterator<T> περιέχει μεθόδους, όπως τα:

- hasNext()- Επιστρέφει true, αν υπάρχει επόμενο στοιχείο στη λίστα.
- next()- Επιστρέφει, αν υπάρχει, το επόμενο στοιχείο στη λίστα.
- remove()- Αφαιρεί το στοιχείο, που μόλις πέρασε ο iterator.

Η υλοποίηση της διαπέρασης μιας συλλογής μπορεί να γίνει με τη χρήση του Iterator ως εξής:

```
void MyMethod(Collection<IntWritable> c) {
for(Iterator<IntWritable> i = c.iterator(); i.hasNext();)
        i.next().cancel();
}
```

Ωστόσο, η χρήση του Iterator απλώς παρεμβάλει «θόρυβο» και μάλιστα χρησιμοποιείται τρεις φορές μέσα στον βρόχο, με αποτέλεσμα αυτό να αυξάνει τους κινδύνους να προκληθεί κάποιο σφάλμα. Ο παραπάνω βρόχος μπορεί να γραφεί με σύμπτυξη, χωρίς τον Iterator ως εξής:

```
void MyMethod(Collection<IntWritable> c) {
    for (IntWritable t : c)
        t.cancel();
}
```

Όταν βλέπετε το σημείο στίξης της άνω-κάτω τελείας (:) σε έναν βρόχο, όπως παραπάνω, μπορείτε να τις διαβάζετε ως «μέσα στο». Ο παραπάνω βρόχος διαβάζεται ως «για κάθε IntWritable t μέσα στο c». Χωρίς τον Iterator η δομή του FOR είναι πιο απλή και χωρίς κινδύνους για πολλαπλά σφάλματα.

8.9.1 Generic Κλάσεις και Μέθοδοι

Οι κλάσεις Mapper & Reducer θα παρατηρήσατε ότι έχουν ως παραμέτρους τύπους δεδομένων, δηλαδή να μπορούν να λειτουργούν με αντικείμενα διαφορετικών τύπων, παρέχοντας ταυτόχρονα ασφάλεια κατά τη μεταγλώττιση (από την έκδοση 5.0 και μετά). Αυτή η δυνατότητα επιτρέπει τη συλλογή αντικειμένων σε μία οντότητα – Java Collection.
Μια generic κλάση ή παραμετρική κλάση ορίζεται, όπως και οι υπόλοιπες κλάσεις, με τη διαφορά ότι μετά το όνομά της πρέπει να ακολουθεί τουλάχιστον μια τυπική παράμετρος ανάμεσα στα σύμβολα <>. Μπορεί να υπάρχουν περισσότερες τυπικοί παράμετροι διαχωρισμένες με το σύμβολο (,). Για παράδειγμα:

```
class name<T1, T2,..., Tn>
{ /* ... */ }
```

Αντίστοιχα, για την περίπτωση των μεθόδων μπορείτε να γράφετε μια generic μέθοδο, που θα κληθεί με παραμέτρους διαφορετικών τύπων. Ανάλογα με τον τύπο της παραμέτρου ο μεταγλωττιστής χειρίζεται τις κλήσεις. Στην υπογραφή της μεθόδου θα υπάρχει, μέσα στα σύμβολα <>, ο τύπος του επιστρεφόμενου αποτελέσματος. Η μέθοδος μπορεί να δεχτεί μία ή περισσότερες παραμέτρους, χωρισμένες με κόμμα (,).

8.9.2 Το Αντικείμενο Class

Στις γραμμές 45-54 του παραδείγματος (Κώδικας 8.1) χρησιμοποιούνται μέθοδοι απαραίτητοι για την εκτέλεση της εργασίας σε Hadoop. Παρατηρείστε ότι μέθοδοι, που δίνονται στις προαναφερόμενες γραμμές, λαμβάνουν ως παράμετρο τον τύπο της κλάσης που μοντελοποιείται από το αντικείμενο Class. Για παράδειγμα, ο τύπος του Text.class είναι Class<Text>.

Κριτήρια αξιολόγησης

Κριτήριο αξιολόγησης 1

Απαντήστε στις παρακάτω δέκα (10) ερωτήσεις πολλαπλών επιλογών. Μόνο μία είναι η σωστή απάντηση σε κάθε ερώτημα.

1. Η μέθοδος Απεικόνισης/Συγχώνευσης (MapReduce) μπορεί να χρησιμοποιηθεί για την επίλυση:

(a) Οποιουδήποτε υπολογιστικού προβλήματος.
(b) Υπολογιστικών προβλημάτων, χωρίς εξαρτήσεις δεδομένων.
(c) Υπολογιστικών προβλημάτων, όπου οι εξαρτήσεις δεδομένων δεν είναι κυκλικές.
(d) Υπολογιστικών προβλημάτων, που ικανοποιούν είτε το (b) είτε το (c).

2. Στο κατανεμημένο σύστημα αρχείων του Hadoop, τα πολύ μεγάλα αρχεία συνήθως σε τμήματα τι μεγέθους χωρίζονται και σε πόσους διαφορετικούς κόμβους δεδομένων αποθηκεύονται;

(a) 64MB και 5 κόμβους δεδομένων
(b) 64MB και 3 κόμβους δεδομένων
(c) 128MB και 5 κόμβους δεδομένων
(d) 128MB και 3 κόμβους δεδομένων

3. Σκοπός του κόμβου ονομάτων (namenode) στο Hadoop είναι:

(a) Να αναθέτει ένα ξεχωριστό όνομα σε κάθε πρόγραμμα Hadoop που ξεκινάει την εκτέλεση του, ώστε να μπορεί να αναγνωριστεί με μοναδικό τρόπο.
(b) Να αναθέτει ξεχωριστά ονόματα στους κόμβους που αποτελούν τη συστάδα Hadoop, κάθε φορά που αυτή τίθεται σε λειτουργία.
(c) Να διατηρεί μεταδεδομένα για τα αρχεία, τα οποία έχουν αποθηκευτεί στο σύστημα αρχείων του Hadoop.
(d) Να διατηρεί πληρφορίες για τα ονόματα των χρηστών, που εκτελούν κάποιο πρόγραμμα Hadoop.

4. Η προγραμματιστική διεπαφή (API) του Hadoop για την Java παρέχει, μεταξύ άλλων, τους τύπους δεδομένων "LongWritable", "IntWritable" και "Text", οι οποίοι, όμως, προσφέρουν παρόμοια λειτουργικότητα με τους βασικούς τύπους δεδομένων της Java "long", "int" και "String". Γιατί προσφέρονται αυτοί οι τύποι δεδομένων από το Hadoop;

(a) Η υλοποίησή τους είναι βελτιστοποιημένη για την αποτελεσματική αποθήκευση των αντίστοιχων δεδομένων στο σύστημα αρχείων HDFS.
(b) Η υλοποίησή τους είναι βελτιστοποιημένη για τη γρήγορη ανάκτηση των αντίστοιχων δεδομένων από το σύστημα αρχείων HDFS.
(c) Η υλοποίησή τους είναι βελτιστοποιημένη για τη μεταφορά των αντίστοιχων δεδομένων μέσω δικτύου.
(d) Η υλοποίησή τους είναι βελτιστοποιημένη για τη γρηγορότερη επεξεργασία των δεδομένων κατά την εκτέλεση των εργασιών Απεικόνισης και Συγχώνευσης.

5. Τι είναι το PIG;

(a) Μια επέκταση του προγραμματιστικού μοντέλου Hadoop σε μορφή γλώσσας προγραμματισμού, η οποία παρέχει τις λειτουργίες του μοντέλου Hadoop ως λειτουργίες της ίδιας της γλώσσας.
(b) Ένα υποσύνολο της προγραμματιστικής διεπαφής (API) του Hadoop για την επεξεργασία δεδομένων πολύ πιο συγκεκριμένων ομάδων προβλημάτων, αλλά με πιο αποδοτικό τρόπο.
(c) Το αντίστοιχο του Hadoop, αλλά για την επεξεργασία βάσεων δεδομένων που έχουν δημιουρ-

γηθεί με SQL.
(d) Κανένα από τα παραπάνω.

6. Για κάθε έργο που πρέπει να εκτελεστεί σε ένα πρόγραμμα Hadoop:

(a) Θα γίνει τουλάχιστον μία προσπάθεια εκτέλεσής του από έναν κόμβο της συστάδας Hadoop.
(b) Θα γίνει τουλάχιστον μία προσπάθεια εκτέλεσής του από κάθε κόμβο της συστάδας Hadoop.
(c) Θα πρέπει να τερματίσουν σωστά τουλάχιστον δύο εκτελέσεις του έργου προς επιβεβαίωση της ορθότητας των αποτελεσμάτων.
(d) Θα πρέπει ανά πάσα χρονική στιγμή να εκτελείται μόνο ένα αντίγραφο του έργου.

7. Δεδομένα εξόδου ενός έργου Απεικόνισης και δεδομένα εισόδου ενός έργου Συγχώνευσης:

(a) Μπορούν να είναι οποιουδήποτε τύπου δεδομένων. Το Hadoop αναλαμβάνει να κάνει τη μετατροπή μεταξύ τύπων, όταν αυτό χρειάζεται.
(b) Τα δεδομένα εξόδου ενός έργου Απεικόνισης πρέπει πάντα να έχουν τον ίδιο τύπο με τα δεδομένα εισόδου ενός έργου Συγχώνευσης.
(c) Έχουν πάντα συγκεκριμένους τύπους δεδομένων, σε οποιοδήποτε πρόγραμμα Hadoop, καθώς αυτό απαιτείται από το Hadoop.
(d) Κανένα από τα παραπάνω.

8. Ο Συνδυαστής (Combiner):

(a) Είναι μια συνάρτηση που πρέπει απαραίτητα να υλοποιηθεί, όπως η συνάρτηση Απεικόνισης και Συγχώνευσης.
(b) Είναι μια συνάρτηση Συγχώνευσης που μπορεί να εφαρμοστεί σε οποιοδήποτε είδος πράξης.
(c) Είναι μια συνάρτηση που συνδυάζει τις συναρτήσεις Απεικόνισης και Συγχώνευσης σε μια κοινή συνάρτηση.
(d) Είναι μια τοπική συνάρτηση συγχώνευσης για κλειδιά, που επαναλαμβάνονται στην ίδια συνάρτηση απεικόνισης.

9. Τι συμβαίνει σε περίπτωση σφάλματος εκτέλεσης ενός έργου;

(a) Τίποτα. Το πρόγραμμα απλά θα βγάλει λάθος αποτελέσματα.
(b) Θα ενημερωθεί ο χρήστης του προγράμματος ότι το πρόγραμμα θα βγάλει λάθος αποτελέσματα.
(c) Θα γίνει εκ νέου προσπάθεια εκτέλεσης του έργου.
(d) Θα τερματιστεί η εκτέλεση όλου του προγράμματος.

10. Ένα έργο Απεικόνισης ή Συγχώνευσης συνήθως εκτελείται:

(a) Στον κόμβο της συστάδας Hadoop, στον οποίο υπάρχουν τα δεδομένα που του έχουν ανατεθεί προς επεξεργασία.
(b) Σε τυχαίο κόμβο της συστάδας Hadoop, αφού πρώτα μεταφερθούν τα δεδομένα που του έχουν ανατεθεί προς επεξεργασία στον συγκεκριμένο κόμβο.
(c) Στον κόμβο της συστάδας Hadoop, που έχει τον λιγότερο φόρτο εργασίας την ώρα που πρέπει να εκτελεστεί το έργο.
(d) Βάσει του προγραμματιστή του προγράμματος Hadoop, που μπορεί να επιλέξει κάποια από τις παραπάνω μεθόδους μέσα από το πρόγραμμα του.

Απαντήσεις

Οι σωστές απαντήσεις είναι οι παρακάτω: 1b, 2d, 3c, 4c, 5a, 6a, 7b, 8d, 9c, 10a.

Βιβλιογραφία

Class. Ανακτήθηκε στις 10 Ιουλίου 2015 από: http://docs.oracle.com/javase/7/docs/api/java/Class.html

Generics. Ανακτήθηκε στις 10 Ιουλίου 2015 από: http://docs.oracle.com/javase/tutorial/java/generics/why.html

Ghemawat, S., Gobioff, H. & Leung S.T. (2003). The Google File System. *19th ACM Symposium on Operating Systems Principles*. Ανακτήθηκε στις 10 Ιουλίου 2015 από: http://research.google.com/archive/gfs.html

Google File System. Ανακτήθηκε στις 10 Ιουλίου 2015 από: http://en.wikipedia.org/wiki/GoogleFileSystem

Hadoop. Ανακτήθηκε στις 12 Ιουλίου 2015 από: http://hadoop.apache.org

HDFS. Ανακτήθηκε στις 12 Ιουλίου 2015 από: http://hadoop.apache.org/docs/stable1/hdfsdesign.html

MapReduce. Ανακτήθηκε στις 13 Ιουλίου 2015 από: http://en.wikipedia.org/wiki/MapReduce

Εκμάθηση του MapReduce. Ανακτήθηκε στις 13 Ιουλίου 2015 από: http://hadoop.apache.org/common/docs/current/mapredtutorial.html

Share-nothing. Ανακτήθηκε στις 12 Ιουλίου 2015 από: http://db.cs.berkeley.edu/papers/hpts85-nothing.pdf

Stringtokenizer. Ανακτήθηκε στις 10 Ιουλίου 2015 από: http://docs.oracle.com/javase/7/docs/api/java/util/StringTokenizer.html

Χρήση του Hadoop. Ανακτήθηκε στις 10 Ιουλίου 2015 από: http://hadoop.apache.org/hadoop/PoweredBy

Κεφάλαιο 9: Λυμένα Θέματα

Σύνοψη

Το κεφάλαιο αυτό περιέχει λυμένα θέματα μέσα από τα οποία γίνεται κατανοητή η χρήση της γλώσσας R και της ύλης που μελετήθηκε στα προηγούμενα κεφάλαια.

Προαπαιτούμενη γνώση

Καλή γνώση της θεωρίας της Επιστήμης των Δεδομένων, που διδάχθηκε σε όλα τα προηγούμενα κεφάλαια του βιβλίου.

Λυμένα Θέματα

Θέμα 1

Γράψτε κώδικα που να υπολογίζει το άθροισμα των πολλαπλάσιων του 3 και του 4 στο διάστημα [0, 1000]. Συμπεριλάβετε μόνο μια φορά τα κοινά τους πολλαπλάσια. Για παράδειγμα, τα πολλαπλάσια του 3 και 4 στο διάστημα [0, 20] είναι: 3, 4, 6, 8, 9, 12, 15, 16, 18 και 20. Το 12 είναι πολλαπλάσιο και του 3 και του 4, ωστόσο θα πρέπει να προστεθεί μόνο μια φορά.

Απάντηση

```
> mul3 <- seq(3, 1000, 3)
> mul4 <- seq(4, 1000, 4)
> muls <- c(mul3, mul4)
> muls <- sum( unique(muls) )
```

Θέμα 2

Γράψτε όσο πιο συνοπτικό κώδικα γίνεται για τη δημιουργία των παρακάτω διανυσμάτων:

(a) **(1, 2, 3, ..., 19, 20)**
(b) **(20, 19, ..., 2, 1)**
(c) **(1, 2, 3, ...,19, 20, 19, ..., 2, 1)**
(d) **(4,6, 3) και αναθέστε στο αποτέλεσμα σε μια μεταβλητή με όνομα temp**
(e) **διάνυσμα, που να περιέχει το διάνυσμα του ερωτήματος (δ) 10 φορές (υπόδειξη: με χρήση της `rep`)**
(f) **(4, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 3, 3, 3, 3, 3, 3, 3, 3, 3)**

Απάντηση

(a)

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

(b)

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

(c)

```
> c(1:20, 19:1)
[1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
[20] 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2
[39]  1
```

(d)

```
> temp <- c(4, 6, 3)
```

(e)

```
> rep(temp, 10)
 [1] 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4
[29] 6 3
```

(f)

```
> rep(temp, times=c(5,7,9))
 [1] 4 4 4 4 4 6 6 6 6 6 6 6 3 3 3 3 3 3 3 3 3
```

Θέμα 3

Εάν το όνομα ενός data frame στο R είναι «data», ποιο από τα παρακάτω θα μου δώσει την τέταρτη γραμμή;

(a) **data[,4]**
(b) **data[4,]**
(c) **data(4,)**
(d) **data(,4)**

Απάντηση

Η σωστή απάντηση είναι το b.

Θέμα 4

Με δεδομένο ένα μέτρο ομοιότητας με τιμές στο διάστημα [0,1], επιλέξτε με ποιο τρόπο μπορείτε να το μετασχηματίσετε σε ένα μέτρο ανομοιότητας με τιμές στο διάστημα [0,∞].

(a) **d=1-s**
(b) **d=(1-s)/s**
(c) **d=-s**
(d) **d=1/(s+1)**

Απάντηση

Η σωστή απάντηση είναι το b.

Θέμα 5

Δημιουργήστε τις ακόλουθες συναρτήσεις:

(a) **συνάρτηση func1, η οποία δέχεται ως όρισμα διάνυσμα x = (x_1, x_2, ..., x_n) και επιστρέφει**

$$(x_1, x_2^2, \ldots, x_n^n)$$

(b) **συνάρτηση func2, η οποία δέχεται ως όρισμα διάνυσμα x = (x_1, x_2, ..., x_n) και επιστρέφει**

$$\left(x_1, \frac{x_2^2}{2}, \ldots, \frac{x_n^n}{n}\right)$$

(c) **συνάρτηση func3, η οποία δέχεται δυο ορίσματα, x και n. Το x μπορεί να είναι οποιοσδήποτε αριθμός, ενώ το n είναι αυστηρά θετικός αριθμός. Η συνάρτηση επιστρέφει το αποτέλεσμα της ακόλουθης έκφρασης:**

$$1 + \frac{x}{1} + \frac{x^2}{2} + \ldots + \frac{x^n}{n}$$

Απάντηση

(a)

```
> func1 <- function(x)
+ {
+    x^(1:length(x))
+ }
> func1(c(1:5))
[1]    1    4   27  256 3125
```

```
> func2 <- function(x)
+ {
+    n <- length(x)
+    (x^(1:n))/(1:n)
+ }
```

```
> func2(c(1:5))
[1]   1   2   9  64 625
> func3 <- function(x, n)
+ {
+     1 + sum((x^(1:n))/(1:n))
+ }
> func3(3,3)
[1] 17.5
```

Θέμα 6

(a) **Να γραφεί συνάρτηση με όνομα `listFun`, η οποία δέχεται ως όρισμα έναν αριθμό n και υλοποιεί τα παρακάτω:**

1. **Δημιουργεί n ανεξάρτητους αριθμούς, έστω x = (x_1, x_2, ..., x_n), οι οποίοι ακολουθούν την κατανομή N(0, 1). Υπόδειξη: ανατρέξτε στις οδηγίες χρήσης της συνάρτησης `rnorm`, π.χ. με την εντολή `?help(rnorm)` .**
2. **Υπολογίζει τη μέση τιμή του x, έστω .**
3. **Αν , τότε δημιουργεί n ανεξάρτητους αριθμός, έστω y = (y_1, y_2, ..., y_n), οι οποίοι ακολουθούν εκθετική κατανομή με μέση τιμή . Αν , τότε δημιουργεί n ανεξάρτητους αριθμός, έστω z = (z_1, z_2, ..., z_n), οι οποίοι ακολουθούν εκθετική κατανομή με μέση τιμή . Έπειτα θέτει y = (y_1, y_2, ..., y_n) = -z. Υπόδειξη: ανατρέξτε στις οδηγίες χρήσης των συναρτήσεων `sign` και `rexp`.**
4. **Υπολογίζει τον αριθμό των στοιχείων, έστω k, για τα οποία ισχύει $|y_j| > |x_j|$**
5. **Επιστρέφει τις λίστες x, y και τον αριθμό k.**

(b) **Εκτελέστε τις ακόλουθες εντολές και παρατηρήστε τον τρόπο με τον οποίο εκτυπώνονται τα αποτελέσματα.**

```
> lapply( rep(10, 4), listFun)
> sapply( rep(10, 4), listFun)
```

Απάντηση

(a)

```
> listFun <- function(n)
+ {
+     x <- rnorm(n)
+     xBar <- mean(x)
+     y <- sign(xBar)*rexp(n, rate=abs(1/xBar))
+     count <- sum( abs(y) > abs(x) )
+     list(x=x, y=y, k=count)
+ }
```

(b)

```
> lapply( rep(10, 4), listFun)
[[1]]
[[1]]$x
 [1]  0.15116299  0.05880405  0.53014976 -1.44514752
 [5]  1.31812337  1.34517134 -0.75170292 -0.02619527
 [9] -0.56419319 -0.43889426

[[1]]$y
 [1] 0.049285543 0.001993024 0.005348972 0.022135425
 [5] 0.009738019 0.005464418 0.015442826 0.006681277
 [9] 0.012475023 0.000586087

[[1]]$k
[1] 0
[[2]]
[[2]]$x
 [1]  0.7292918 -0.8750306  1.2590433 -0.4285283 -1.4339591
 [6] -1.0651494 -0.6020118 -1.0810530 -0.4071496 -0.0576051

[[2]]$y
 [1] -0.1015831 -1.0357603 -0.3063154 -0.2430346 -0.2742429
 [6] -0.8883897 -0.2813771 -1.0394824 -0.1879120 -0.2087744

[[2]]$k
[1] 2

[[3]]
[[3]]$x
 [1] -0.4711402  0.9639523  0.4908114  0.3257803 -1.0587862
 [6] -0.8948662 -0.5394175 -1.1200807 -0.7317642 -0.2353312

[[3]]$y
 [1] -0.02790416 -0.51041871 -0.02289331 -0.33233244
 [5] -0.13085901 -0.00698617 -0.26404962 -0.08911424
 [9] -1.34449028 -2.04041989
```

```
[[3]]$k
[1] 3

[[4]]
[[4]]$x
 [1] -0.5859908  0.1682858 -0.7233417 -1.6583072 -0.3628087
 [6] -0.5517844 -0.9563336 -0.1374157  0.1659169  0.9654025

[[4]]$y
 [1] -0.257688480 -0.127319713 -0.002418168 -0.451218564
 [5] -0.146414847 -1.003016860 -0.501697945 -0.003195404
 [9] -0.043886064 -0.100161879

[[4]]$k
[1] 1

> sapply( rep(10, 4), listFun)
  [,1]       [,2]       [,3]       [,4]
x Numeric,10 Numeric,10 Numeric,10 Numeric,10
y Numeric,10 Numeric,10 Numeric,10 Numeric,10
k 3          3          4          3
```

Σημείωση: Οι αριθμοί διαφέρουν από εκτέλεση σε εκτέλεση. Αυτό δεν είναι λάθος, είναι ο τρόπος λειτουργίας των συναρτήσεων που υλοποιούν τις κατανομές.

Θέμα 7

Φορτώστε το σύνολο δεδομένων Epub του πακέτου `arules`. Αφού εκτυπώσετε και μελετήσετε τα χαρακτηριστικά του συνόλου δεδομένων, να βρείτε τον αριθμό των εγγραφών ανά έτος. Στη συνέχεια, φιλτράρετε και απεικονίστε με χρήση της συνάρτησης `image` τις εγγραφές του 2005. Τέλος, εκτυπώστε τις εγγραφές από το έτος 2005, στις οποίες οι δοσοληψίες περιέχουν πάνω από 15 στοιχεία. Για το τελευταίο ερώτημα, χρησιμοποιήστε τις συναρτήσεις `transactionInfo` και `size`.

Απάντηση

Αρχικά, φορτώνουμε το σύνολο δεδομένων `Epub` του πακέτου `arules`, και στη συνέχεια με τη συνάρτηση summary εκτυπώνουμε συνοπτικές, χρήσιμες πληροφορίες για το σύνολο δεδομένων:

```
> library(arules)
> data(Epub)
> Epub
transactions in sparse format with
 15729 transactions (rows) and
 936 items (columns)
>
> summary(Epub)
transactions as itemMatrix in sparse format with
 15729 rows (elements/itemsets/transactions) and
 936 columns (items) and a density of 0.001758755

most frequent items:
doc_11d doc_813 doc_4c6 doc_955 doc_698 (Other)
    356     329     288     282     245   24393

element (itemset/transaction) length distribution:
sizes
    1     2     3     4     5     6     7     8     9    10    11
12    13    14    15    16    17    18    19    20    21    22
11615  2189   854   409   198   121    93    50    42    34    26
12    10    10     6     8     6     5     8     2     2     3
   23    24    25    26    27    28    30    34    36    38    41
43    52    58
    2     3     4     5     1     1     1     2     1     2     1
1     1     1

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   1.000   1.000   1.646   2.000  58.000

includes extended item information - examples:
   labels
1 doc_11d
2 doc_13d
3 doc_14c

includes extended transaction information - examples:
      transactionID           TimeStamp
```

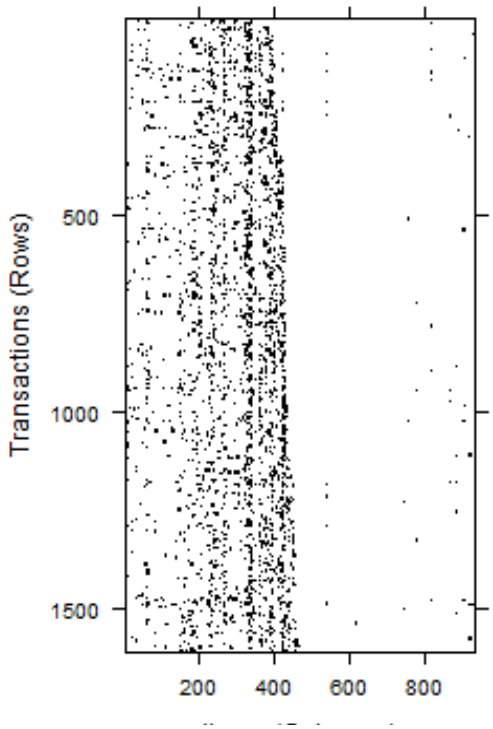
```
10792  session_4795 2003-01-02 03:59:00
10793  session_4797 2003-01-02 14:46:01
10794  session_479a 2003-01-02 17:50:38
```

Για να βρούμε τον αριθμό εγγραφών ανά έτος:

```
> year <- strftime(as.POSIXlt(transactionInfo(Epub)[["Time-
Stamp"]]), "%Y")
> table(year)
year
2003 2004 2005 2006 2007 2008 2009
 987 1375 1611 3012 4052 4690    2
>
```

Έπειτα, φιλτράρουμε τις εγγραφές του έτους 2005, και με χρήση της image κάνουμε την απεικόνιση:

```
> Epub2005 <- Epub[year == "2005"]
> length(Epub2005)
[1] 1611
> image(Epub2005)
```



Εικόνα *9.1* *Απεικόνιση εγγραφών συνόλου δεδομένων Epub για το έτος 2005.*

Τέλος, για να βρούμε τις δοσοληψίες με περισσότερα από 15 στοιχεία, για το έτος 2005, έχουμε:

```
> transactionInfo(Epub2005[size(Epub2005) > 15])
      transactionID           TimeStamp
13083  session_af4a 2005-01-27 23:48:23
13121  session_b0b7 2005-02-05 05:54:32
13195  session_b391 2005-02-26 17:34:21
13351  session_b9a8 2005-04-02 04:30:01
13687  session_c683 2005-06-16 06:50:01
>
```

Θέμα 8

Το χάλκινο, το ασημένιο και το χρυσό μετάλλιο που απονέμεται στους Ολυμπιακούς αγώνες, τι είδους γνώρισμα είναι;

(a) **ονομαστικό**
(b) **διατακτικό**
(c) **συνεχές**
(d) **διακριτό, αριθμητικό**

Απάντηση

Η σωστή απάντηση είναι το b.

Θέμα 9

Ο γιατρός ενός σχολείου μέτρησε το ύψος των μαθητών της Ε^{ης} τάξης. Τα αποτελέσματα των μετρήσεων σε cm ήταν τα ακόλουθα:

130	**132**	**139**	**136**	**131**	**153**	**133**	**129**	**133**	**119**
132	**135**	**132**	**134**	**130**	**135**	**130**	**136**	**134**	**137**

Αντίστοιχα, έγινε μέτρηση του βάρους των μαθητών και τα αποτελέσματα σε kg ήταν τα ακόλουθα:

35	**41**	**38**	**41**	**43**	**51**	**40.5**	**38.5**	**36**	**39**
40	**37**	**40**	**43**	**38**	30	**42.5**	**36**	**38**	**40**

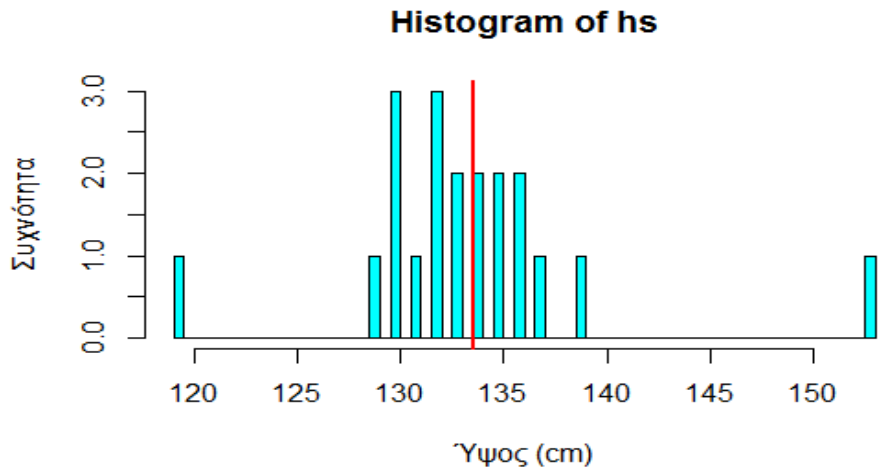
α) Ποιες από τις τιμές ύψους μπορούν να θεωρηθούν ακραίες τιμές και γιατί;
β) Να απεικονίσετε το βάρος των μαθητών με θηκόγραμμα.
γ) Να απεικονίσετε και τις δυο μεταβλητές μαζί (ύψος και βάρος).

Για την επίλυση των ερωτημάτων να γραφεί κώδικας R.

Απάντηση

α) Για να γίνει πιο κατανοητή η έννοια των ακραίων τιμών, φτιάχνουμε μια απεικόνιση των τιμών του ύψους. Παρατηρούμε ότι οι τιμές 119 και 153 απέχουν κατά πολύ και από τις υπόλοιπες τιμές, αλλά και από τη μέση τιμή, η οποία επισημαίνεται με μια κόκκινη γραμμή (Εικόνα 9.2).

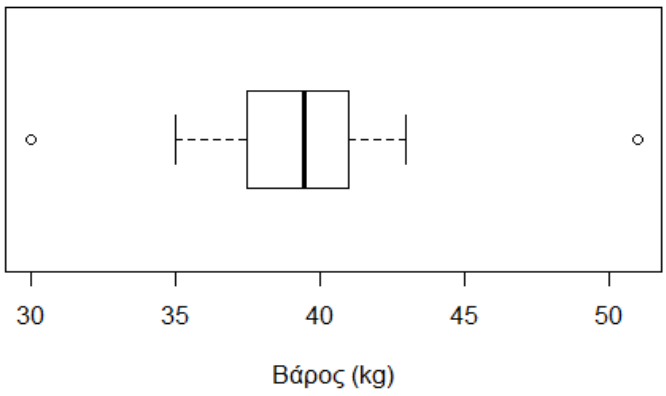
```
> hs <- c(130, 132, 139, 136, 131, 153, 133, 129, 133, 119,
132, 135, 132, 134, 130, 135, 130, 136, 134, 137)
> ws <- c(35, 41, 38, 41, 43, 51, 40.5, 38.5, 36, 39, 40, 37,
40, 43, 38, 30, 42.5, 36, 38, 40)
> hist(hs, breaks=50, col=5, xlab="Ύψος (cm)", ylab="Συχνότητα»)
> abline(b=90,v=mean(hs), col=2, lwd=2)
```



Εικόνα ***9.2*** *Απεικόνιση τιμών ύψους και μέσης τιμής (κόκκινη γραμμή).*

β)

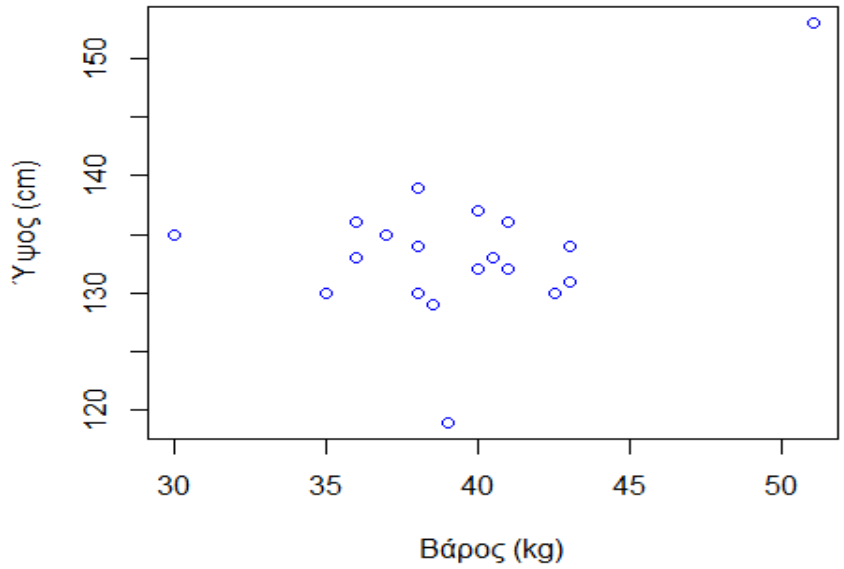
```
> boxplot(ws, horizontal=T, xlab=c("weight (kg)"))
>
```



Εικόνα ***9.3*** *Απεικόνιση των τιμών βάρους με θηκόγραμμα.*

γ)

```
> plot(ws, hs, col=4, xlab=»Βάρος (kg)», ylab=»Ύψος (cm)»)
```



Εικόνα *9.4* *Απεικόνιση των τιμών βάρους και ύψους.*

Θέμα 10

Εγκαταστήστε τα πακέτα `rattle`, `rpart.plot` και `RcolorBrewer`, χρησιμοποιώντας τις παρακάτω εντολές:

```
install.packages('rattle')
install.packages('rpart.plot')
install.packages('RColorBrewer')
```

και στη συνέχεια φορτώστε το σύνολο δεδομένων `weather`. Διαχωρίστε το σύνολο δεδομένων με τέτοιο τρόπο, ώστε το 70% να αποτελεί το σύνολο εκπαίδευσης και το υπόλοιπο 30% το σύνολο ελέγχου. Έπειτα, δημιουργήστε ένα δένδρο απόφασης με τη χρήση της συνάρτησης `rpart`, για πρόβλεψη των τιμών της κλάσης `RainTomorrow`. Για την εκπαίδευση του δένδρου απόφασης αγνοήστε τα χαρακτηριστικά `Date`, `Location` και `RISK_MM`. Για την αξιολόγηση του μοντέλου, θα πρέπει να αφαιρέσετε επιπλέον και την κλάση. Χρησιμοποιώντας τη συνάρτηση `fancyRpartPlot`, απεικονίστε το μοντέλο. Τέλος, αξιολογήστε το μοντέλο και βρείτε το ποσοστό των σωστά κατηγοριοποιημένων δειγμάτων.

Απάντηση

Αρχικά, φορτώνουμε τα πακέτα και το σύνολο δεδομένων που θα χρησιμοποιήσουμε.

```
> library(rpart)
> library(rattle)
> library(rpart.plot)
> library(RColorBrewer)
>
```

```
> # Φόρτωση δεδομένων
> data(weather)
> dsname <- «weather»
> ds <- weather
>
```

Στη συνέχεια, αφαιρούμε τα χαρακτηριστικά που δεν χρειάζονται για τη φάση εκπαίδευσης και τη φάση ελέγχου, και αποθηκεύουμε αντίγραφα του συνόλου δεδομένων στις μεταβλητές `vars` και `input_vars` αντίστοιχα. Δηλαδή, η `vars` περιέχει το σύνολο δεδομένων, χωρίς τα χαρακτηριστικά `Date`, `Location` και `RISK_MM`, ενώ η `input_vars` περιέχει το σύνολο δεδομένων, χωρίς τα χαρακτηριστικά `Date`, `Location`, `RISK_MM` και την κλάση `RainTomorrow`.

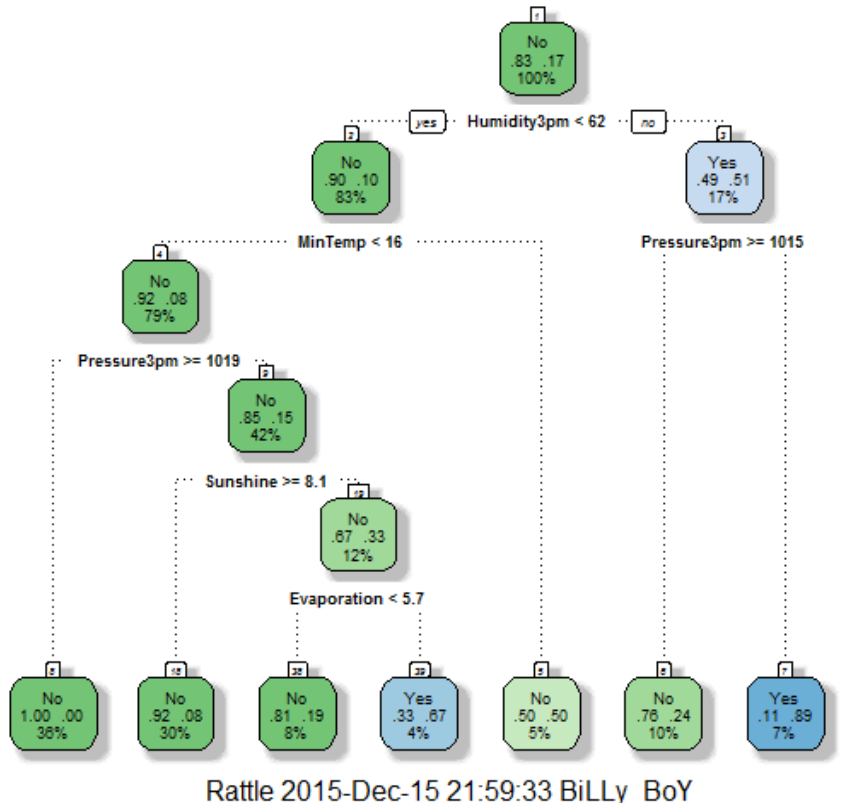
```
> # Διαγραφή χαρακτηριστικών χωρίς πληροφοριακό κέρδος
> id <- c(«Date», «Location»)
> target <- "RainTomorrow"
> risk <- "RISK_MM"
> ignore <- c(id, if (exists("risk")) risk)
> vars <- setdiff(names(ds), ignore)
>
> # Αφαίρεση κλάσης
> input_vars <- setdiff(vars, target)
>
```

Για τον διαχωρισμό του συνόλου δεδομένων χρησιμοποιούμε τη συνάρτηση sample. Για την παραγωγή ίδιων αποτελεσμάτων πρέπει να θέσουμε ένα συγκεκριμένο φύτρο (seed). Παράγοντας τους δείκτες για το σύνολο εκπαίδευσης, παίρνουμε τη διαφορά, για να υπολογίσουμε τους δείκτες του συνόλου ελέγχου.

```
> # Διαχωρισμός συνόλου εκπαίδευσης (70%) και ελέγχου (30%)
> set.seed(976)
> nobs <- nrow(ds)
> train <- sample(nobs, 0.7 * nobs)
> test <- setdiff(seq_len(nobs), train)
```

Έπειτα, δημιουργούμε το δένδρο απόφασης και την οπτικοποίησή του.

```
> # Δημιουργία και απεικόνιση μοντέλου
> form <- formula(paste(target, «~ .»))
> model <- rpart(formula=form, data=ds[train,vars])
> fancyRpartPlot(model)
>
```



Εικόνα *9.5* *Απεικόνιση δένδρου απόφασης για το σύνολο δεδομένων weather.*

Τέλος, μένει να αξιολογήσουμε το μοντέλο μας. Για τον σκοπό αυτό, θα χρησιμοποιήσουμε το σύνολο ελέγχου, του οποίου οι εγγραφές δεν έχουν χρησιμοποιηθεί στην εκπαίδευση του μοντέλου. Θα προβλέψουμε, δηλαδή, την τιμή κλάσης για τις εγγραφές του συνόλου ελέγχου, για το οποίο, όμως, γνωρίζουμε την πραγματική κλάση κάθε εγγραφής. Έτσι, θα αξιολογήσουμε το πόσο καλά λειτουργεί το μοντέλο μας. Με βάση τον παρακάτω κώδικα, έχουμε ποσοστό 79% σωστών προβλέψεων.

```
> # Απομόνωση πραγματικών τιμών κλάσης για το σύνολο ελέγχου
> actual <- ds[test, target]
>
> # Μετατροπή προβλέψεων σε ίδια μορφή
> pred <- predict(model, newdata=ds[test,input_vars])
> predicted <- vector(length=length(pred))
> predicted <- pred[, "No"] < pred[,"Yes"]
> predicted <- as.factor(predicted)
> levels(predicted) <- c("No", "Yes")
>
> # Ποσοστό σωστών τιμών που προβλέφτηκαν
> length(which(actual == predicted))/length(actual)
[1] 0.7909091
```

Θέμα 11

Φορτώστε το σύνολο δεδομένων `cats` από το πακέτο `MASS`. Τα χαρακτηριστικά `Bwt` και `Hwt` δίνουν το βάρος του σώματος (σε kg) και της καρδιάς (σε g), αντίστοιχα.

α) Στο σύνολο δεδομένων υπάρχουν δείγματα από αρσενικές και θηλυκές γάτες. Δημιουργήστε ένα σύνολο δεδομένων με εγγραφές μόνο από αρσενικές γάτες.

β) Δημιουργήστε μια απεικόνιση των μεταβλητών `Bwt` και `Hwt`. Από την απεικόνιση και μόνο, φαίνεται λογική η χρήση ενός μοντέλου γραμμικής παλινδρόμησης;

γ) Δημιουργήστε ένα μοντέλο γραμμικής παλινδρόμησης για το σύνολο εγγραφών που αφορά τις αρσενικές γάτες, για την πρόβλεψη του βάρους της καρδιάς, δοθέντος του βάρους του σώματος. Προσθέστε την απεικόνιση της παλινδρόμησης πάνω στην γραφική του ερωτήματος (β). Σχολιάστε το πόσο καλά προσεγγίζει το μοντέλο τα δεδομένα.

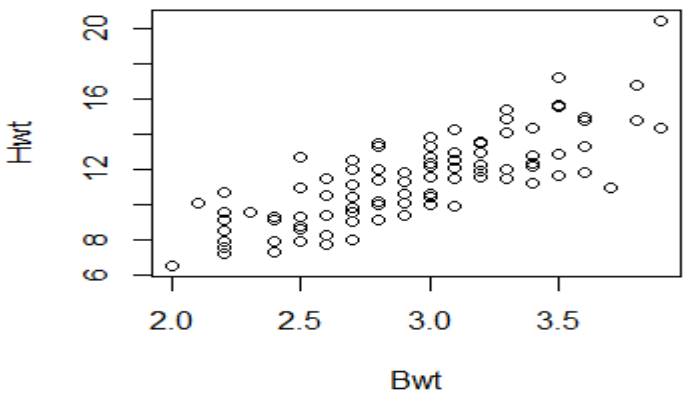
Απάντηση

α) Αφού φορτώσουμε τα δεδομένα, απομονώνουμε τις εγγραφές που αφορούν αρσενικές γάτες.

```
> library(MASS)
> library(dplyr)
> data(cats)
> cats <- tbl_df(cats)
>
> male <- filter(cats, Sex=="M")
>
```

β) Ο κώδικας για την απεικόνιση των δυο χαρακτηριστικών και η αντίστοιχη γραφική παράσταση (Εικόνα 9.6) φαίνονται παρακάτω. Παρατηρώντας όλα τα σημεία, μπορούμε εύκολα να διαπιστώσουμε ότι μπορούν να προσεγγιστούν από μια ευθεία. Επομένως, η χρήση ενός μοντέλου γραμμικής παλινδρόμησης είναι απολύτως λογική.

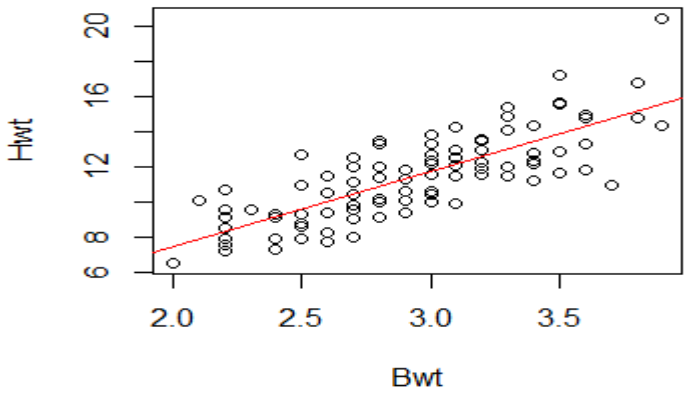
```
> plot(male$Bwt, male$Hwt, xlab="Bwt", ylab="Hwt")
>
```



Εικόνα 9.6 *Απεικόνιση των χαρακτηριστικών Bwt και Hwt για τις αρσενικές γάτες.*

γ) Η δημιουργία του μοντέλου γίνεται με χρήση της συνάρτησης `lm`. Για την απεικόνιση του μοντέλου χρειαζόμαστε τις συνιστώσες του. Ουσιαστικά, πρόκειται για μια γραμμή, η οποία προσεγγίζει όσο καλύτερα τα σημεία. Όπως κάθε ευθεία, έχει κλίση και σταθερό όρο, δηλαδή είναι της μορφής y = ax+ b. Τέλος, παρατηρώντας την Εικόνα 9.7, μπορούμε να διαπιστώσουμε ότι η ευθεία προσεγγίζει αρκετά καλά τα δεδομένα μας.

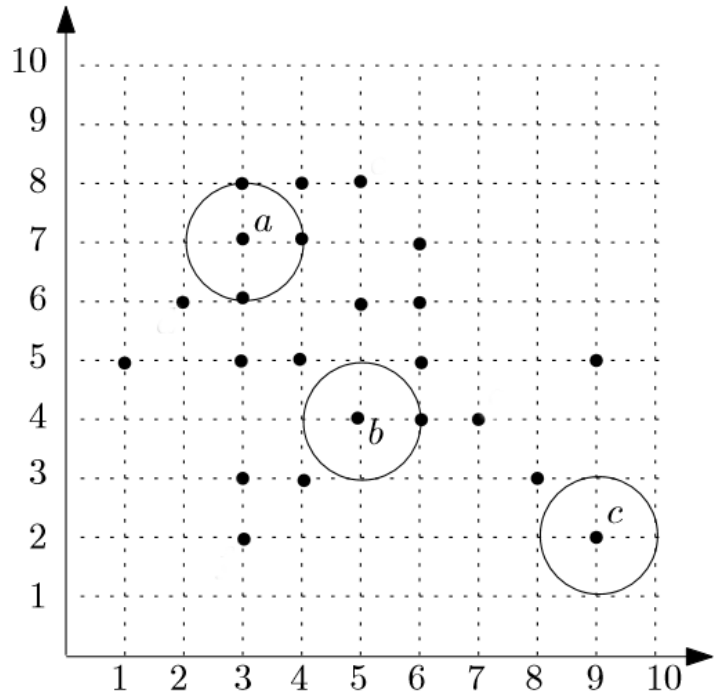
```
> fit <- lm(Hwt ~ Bwt, data=male)
> fit

Call:
lm(formula = Hwt ~ Bwt, data = male)

Coefficients:
(Intercept)          Bwt
     -1.184        4.313
> abline(a=fit$coefficients[1], b=fit$coefficients[2], col="red")
>
```



Εικόνα *9.7* *Απεικόνιση της προσέγγισης των δεδομένων από το μοντέλο γραμμικής παλινδρόμησης.*

Θέμα 12

Σας δίνεται ένα σύνολο σημείων, όπως φαίνεται στην Εικόνα 9.8.



Εικόνα *9.8* *Σύνολο σημείων.*

α) Ο αλγόριθμος συσταδοποίησης DBSCAN κατηγοριοποιεί τα σημεία δεδομένων σε κεντρικά, προσεγγίσιμα, και ακραία. Η κατηγοριοποίησή τους γίνεται βάσει δυο παραμέτρων, των ε και MinPts. Έστω ότι ε =1 και MinPts = 3. Σε ποια κατηγορία ανήκουν τα σημεία a, b, και c;

β) Περάστε τα δεδομένα στην R και εφαρμόστε τον αλγόριθμο συσταδοποίησης DBSCAN, με ε =1 και MinPts = 3.

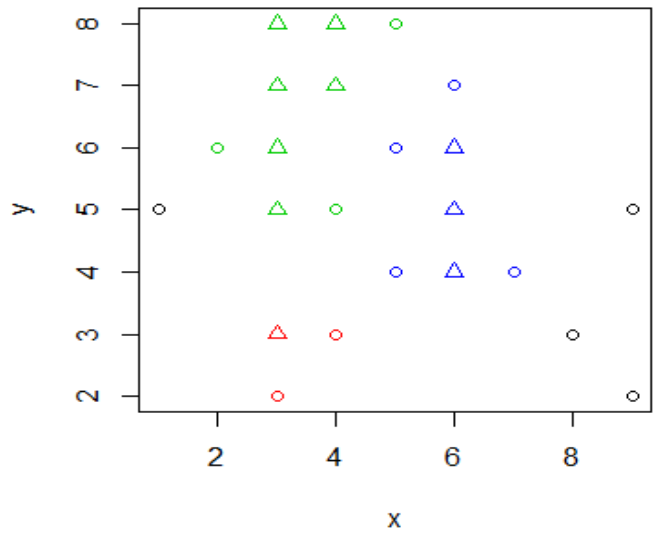
γ) Εφαρμόστε τον αλγόριθμο συσταδοποίησης k-means. Χρησιμοποιήστε όσα διδαχτήκατε, για να αποφασίσετε το καλύτερο δυνατό πλήθος συστάδων, k.

Απάντηση

α) Το σημείο a είναι κεντρικό σημείο, διότι στη γειτονιά του υπάρχουν 3 σημεία. Το σημείο b έχει στη γειτονιά του μόνο 1 (< 3) σημείο. Ωστόσο, είναι προσεγγίσιμο από το σημείο με συντεταγμένες (6,4). Τέλος, το σημείο c είναι ακραίο, αφού δεν υπάρχει κανένα άλλο σημείο στη γειτονιά του.

β) Αφού περάσουμε τα δεδομένα σε διανύσματα, καλούμε τον αλγόριθμο DBSCAN. Η απεικόνιση της συσταδοποίησης (Εικόνα 9.9) επαληθεύει την απάντηση που δώσαμε στο ερώτημα (α).

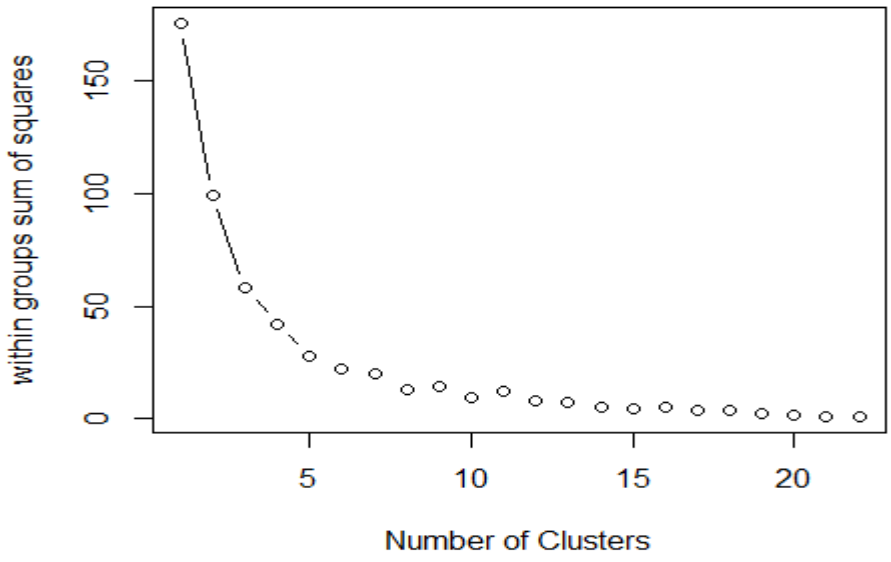
```
> x <- c(1, 2, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 6, 6, 6,
6, 7, 8, 9, 9)
> y <- c(5, 6, 2, 3, 5, 6, 7, 8, 3, 5, 7, 8, 4, 6, 8, 4, 5, 6,
7, 4, 3, 2, 5)
>
> data <- data.frame(x=x, y=y)
>
> dbscan(data, eps=1, MinPts=3, showplot=TRUE)
```



Εικόνα *9.9* *Οπτικοποίηση συσταδοποίησης με τον αλγόριθμο DBSCAN.*

γ) Για να αποφανθούμε για τον αριθμό των συστάδων, χρησιμοποιούμε τον κανόνα του αγκώνα (Εικόνα 9.10).

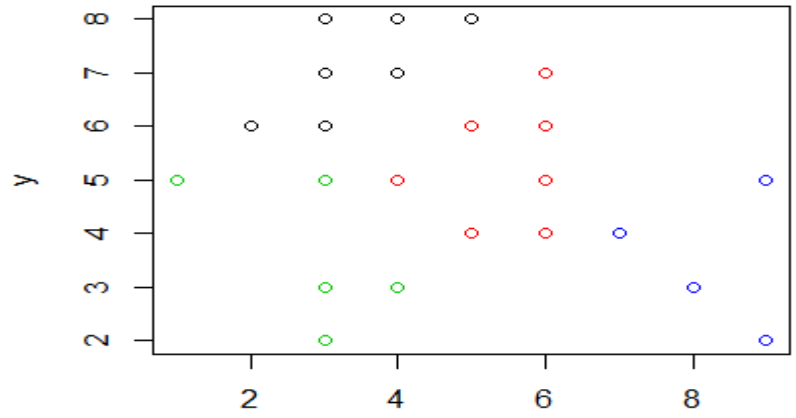
```
> wssplot <- function(data, nc){
+    wss <- (nrow(data)-1) * sum(apply(data,2,var))
+    for(i in 2:nc){
+       wss[i] <- sum(kmeans(data,centers=i)$withinss)
+    }
+    plot(1:nc, wss, type='b',xlab='Number of Clusters',
+         ylab='within groups sum of squares')
+ }
> wssplot(data, nrow(data)-1)
```



Εικόνα ***9.10*** *Εφαρμογή του κανόνα του αγκώνα για εντοπισμό του βέλτιστου αριθμού συστάδων.*

Παρατηρούμε ότι δεν είναι ξεκάθαρο ποιος αριθμός συστάδων είναι ο ιδανικός. Ξεκινάμε με τον αριθμό, στον οποίο βρίσκεται πλησιέστερα ο «αγκώνας» στη γραφική παράσταση, δηλαδή k = 4. Παρατηρούμε ότι οι δυο αλγόριθμοι δημιουργούν παρόμοιες συστάδες. Ωστόσο, ο k-means είναι περισσότερο επιρρεπής σε ακραίες τιμές από τον DBSCAN.

```
> set.seed(123)
> kc <- kmeans(data, 4)
> plot(data, col=kc$cluster)
```



Εικόνα ***9.11*** *Οπτικοποίηση συσταδοποίησης με τον αλγόριθμο k-means, για k = 4.*

Θέμα 13

Δίνονται τα έξι δισδιάστατα σημεία (Πίνακας 9.1). Βρείτε το δενδρόγραμμα και τις ακριβείς τιμές των αποστάσεων, στις οποίες δημιουργούνται οι αντίστοιχες συστάδες κατά την συσσωρευτική ιεραρχική συσταδοποίηση, όπως αυτά προκύπτουν από την εφαρμογή της Ευκλείδειας απόστασης σε συνδυασμό με την τεχνική του:

α) απλού συνδέσμου, και

β) πλήρους συνδέσμου

	X	Y
p1	0.4005	0.5306
p2	0.2148	0.3854
p3	0.3457	0.3156
p4	0.2652	0.1875
p5	0.0789	0.4139
p6	0.4548	0.3022

Πίνακας *9.1* *Δισδιάστατα σημεία για το Θέμα 14.*

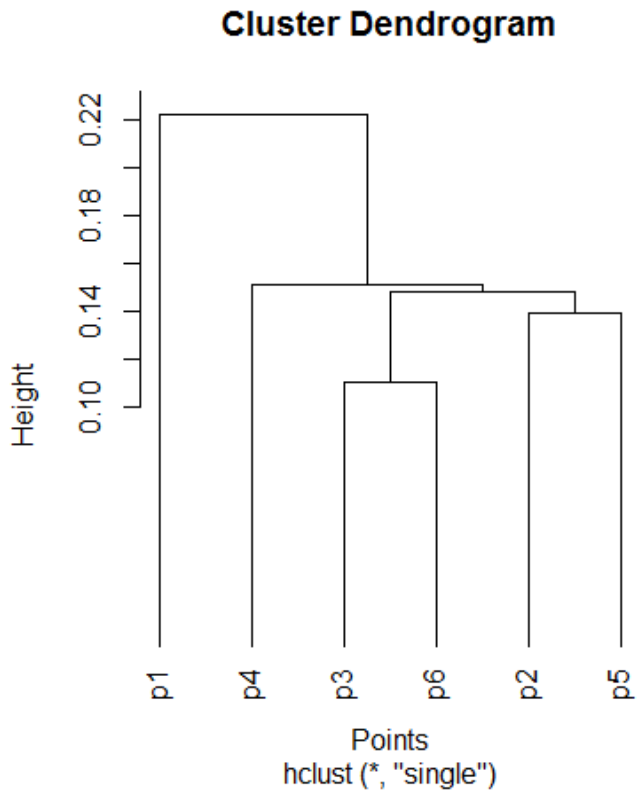
Απάντηση

Ο πίνακας Ευκλείδειας Απόστασης των έξι σημείων είναι ο παρακάτω:

	1	**2**	**3**	**4**	**5**	**6**
1	0.0000000	0.2357277	0.2218739	0.3688139	0.3421191	0.2347659
2	0.2357277	0.0000000	0.1483471	0.2042170	0.1388563	0.2540123
3	0.2218739	0.1483471	0.0000000	0.1512940	0.2843328	0.1099198
4	0.3688139	0.2042170	0.1512940	0.0000000	0.2931973	0.2215948
5	0.3421191	0.1388563	0.2843328	0.2931973	0.0000000	0.3921450
6	0.2347659	0.2540123	0.1099198	0.2215948	0.3921450	0.0000000

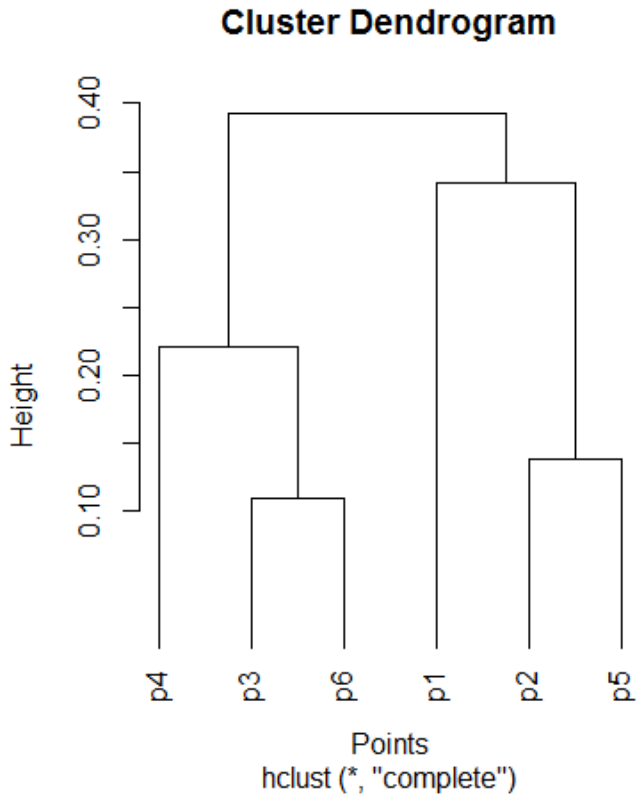
Πίνακας *9.2* *Πίνακας με την Ευκλείδεια απόσταση των 6 σημείων (Θέμα 14).*

1. α) Οι αποστάσεις, στις οποίες δημιουργούνται οι συστάδες του απλού συνδέσμου, είναι οι 0.11, 0.139, 0.148, 0.151 και 0.222, ενώ το δενδρόγραμμα απλού συνδέσμου δίνεται στη συνέχεια:



Εικόνα ***9.12*** *Δενδρόγραμμα απλού συνδέσμου.*

β) Οι αποστάσεις, στις οποίες δημιουργούνται οι συστάδες του πλήρους συνδέσμου, είναι οι 0.11, 0.139, 0.222, 0.342 και 0.392, ενώ το δενδρόγραμμα πλήρους συνδέσμου δίνεται στη συνέχεια:



Εικόνα ***9.13*** *Δενδρόγραμμα πλήρους συνδέσμου.*

Θέμα 14

Πώς μεταφράζεται το ότι ένας κανόνας συσχέτισης έχει υποστήριξη ίση με 0.02 και εμπιστοσύνη ίση με 1;

Απάντηση

Η υποστήριξη σχετίζεται με το πλήθος συναλλαγών που περιέχουν το αντίστοιχο στοιχειοσύνολο από το οποίο προέρχεται ο κανόνας συσχέτισης. Η εμπιστοσύνη έχει να κάνει με το ποσοστό των συναλλαγών, στις οποίες, αν περιέχονται τα στοιχεία του αριστερού μέλους, τότε περιέχεται και το στοιχείο του δεξιού μέλους. Έτσι, υποστήριξη ίση με 0.02 σημαίνει ότι υποστηρίζεται από το 2% των συναλλαγών, ενώ εμπιστοσύνη ίση με 1 σημαίνει ότι, αν μια συναλλαγή περιέχει τα στοιχεία του αριστερού μέλος, οπωσδήποτε θα περιέχει και τα στοιχεία του δεξιού μέλους.

Θέμα 15

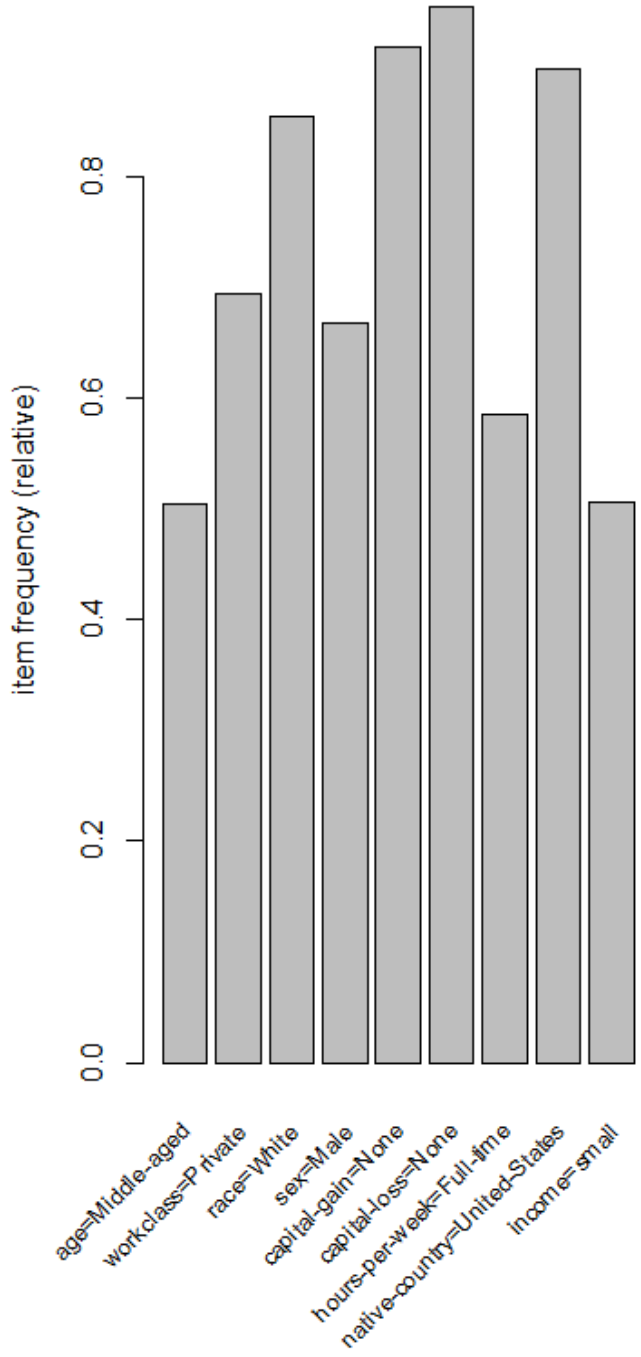
Φορτώστε το σύνολο δεδομένων `Adult` από το πακέτο `arules`. Χρησιμοποιώντας τη συνάρτηση `itemFrequencyPlot`, απεικονίστε ποια στοιχεία του συνόλου δεδομένων εμφανίζονται τουλάχιστον στο 50% των εγγραφών. Στη συνέχεια, εκτυπώστε το πλήθος των κανόνων συσχέτισης που εξάγονται για κατώφλι υποστήριξης 0.02 και κατώφλι εμπιστοσύνης 0.8.

Απάντηση

Ο κώδικας επίλυσης του θέματος δίνεται παρακάτω. Το πλήθος των κανόνων συσχέτισης είναι 80211. Στην γραφική απεικόνιση (Εικόνα 9.14) φαίνονται και τα στοιχεία που εμφανίζονται σε παραπάνω από τις μισές εγγραφές του συνόλου δεδομένων.

```
> library(arules)
> data(Adult)
> itemFrequencyPlot(Adult, support = 0.5, cex.names=0.8)
> rules <- apriori(Adult, parameter = list(support = 0.02, con-
fidence = 0.8))
Parameter specification:
 confidence minval smax arem  aval originalSupport support
        0.8    0.1    1 none FALSE            TRUE    0.02
 minlen maxlen target   ext
      1     10  rules FALSE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

apriori - find association rules with the apriori algorithm
version 4.21 (2004.05.09)          (c) 1996-2004    Christian
Borgelt
set item appearances ...[0 item(s)] done [0.00s].
```

```
set transactions ...[115 item(s), 48842 transaction(s)] done
[0.10s].
sorting and recoding items ... [59 item(s)] done [0.01s].
creating transaction tree ... done [0.02s].
checking subsets of size 1 2 3 4 5 6 7 8 9 10 done [0.41s].
writing ... [80211 rule(s)] done [0.02s].
creating S4 object  ... done [0.04s].
> rules
set of 80211 rules
```




```
>
```

Εικόνα ***9.14*** *Απεικόνιση στοιχείων με συχνότητα πάνω από 0.5, για το σύνολο δεδομένων Adult.*

Βιβλιογραφία

Wickham, H. (2014). Tidy Data. *Journal of Statistical Sofrware, 59* (10). Ανακτήθηκε στις 17 Νοεμβρίου 2015, από: http://vita.had.co.nz/papers/tidy-data.pdf

Dunham, M. H. (2003). Data Mining: Introductory and Advanced Topics. *Pearson Education*, Upper Saddle River, N. J. Prentice Hall.

Hahsler, M., Grün, B. & Hornik, K. (2005). arules – A Computational Environment for Mining Association Rules and Frequent Item Sets. *Journal of Statistical Sofrware, 14* (15). Ανακτήθηκε στις 17 Νοεμβρίου 2015, από: http://epub.wu.ac.at/3976/1/arules.pdf

Ευρετήριο

Ελληνικός όρος	Αγγλικός όρος
Α	
αγωγός	pipeline
άδειασμα	flush
αθροίσματα ελέγχου	checksums
ακέραιος	integer
ακραίες τιμές	outliers
αληθής	true
αλγόριθμος εκτός μνήμης	out of core algorithm
αλγόριθμος σταδιακής καθόδου	gradient descent algorithm
ανάγνωσης-πολλές φορές	read-multiple times
αναγνώριση προτύπων	pattern recognition
αναγωγή	reduce
ανακάλυψη από βάσεις δεδομένων	knowledge discovery in databases
ανεξάρτητες μεταβλητές	independent variables
ανιχνευτής εργασίας	Jobtracker
αντικείμενο	object
απλός σύνδεσμος	single link
αποθετήρες δεδομένων	data warehouses
αποθήκες δεδομένων	data warehouses
απομακρυσμένη προσάρτηση NFS	remote NFS mount
από πριν	a priori
αραιά	sparse
αριθμητικός – πραγματικός αριθμός	numeric
αρχείο διαμόρφωσης	configuration file
ατομικός	atomic
αφέντης-σκλάβος	master-slave
Δ	
δημιουργός	constructor
διαβίβαση δεδομένων	throughput
διάγραμμα πίτας ή τομεογράφημα	pie chart
διαιρετικοί αλγόριθμοι	divisive algorithms
διάμεσος	Median
διάνυσμα	Vector
διανυσματοποίηση	vectorization
διασπορά	Variance
διάσταση	dimension
διαφορετικό ράφι από το πρώτο	off-rack
διερμηνεία και αξιολόγηση	interpretation/evaluation
δοχεία	bins
δυναμική σύνδεση/τοποθέτηση	pluggable
Ε	
εγγραφής-μια φορά	write-once
είδος	species
εικόνα ονομάτων	namespace image

εκπαίδευση	training
ελάχιστη	min
ελάχιστης απόσταση ή απλός σύνδεσμος	single link
ελεγκτής ανακατεύθυνσης	failover controller
έλεγχος	monitoring
εμπεριέχει	wraps
ελεύθερη μεταβλητή	free variable
εμπιστοσύνη	confidence
ενδοτεταρτημοριακό εύρος	interquartile range
ενδοχείαση	binning
ενοποίηση σχήματος	schema integration
εξαγωγή κανόνων συσχέτισης	mining association rules
εξαρτημένη μεταβλητή	dependent variable
Εξόρυξη Δεδομένων	Data Mining
αρχείο καταγραφής επεξεργασίας	edit log
επεκτείνει	extends
επιβλεπόμενη μάθηση	supervised learning
επίπεδα	level/ levelwise
Επιστήμη των Δεδομένων	Data Science
εργαλεία καθαρισμού δεδομένων	data scrubbing tools
εργαλεία λογιστικού ελέγχου	data auditing tools
εργασίες Αντιστοίχισης	Mappers
εργασίες Συγχώνευσης	Reducers
εύρος	range
εύρος ζώνης	bandwidth
Θ	
θηκόγραμμα	boxplot
Ι	
ιδιότητα	attribute
ιστογράμματα	histograms
Ιχνηλάτης Εργασιών	JobTracker
Κ	
καθολικό περιβάλλον	global environment
κανόνας συσχέτισης	association rule
κανονικοποίηση	regularization
κατάρα της διαστατικότητας	curse of dimensionality
κατηγορικός	nominal
κατηγοριοποιητής	classifier
κεντρικά σημεία	core points
κέντρο δεδομένων	data center
κλάση	class
κλάση δοσοληψίας	transaction
κλειδί	key
κλειδί αποτελέσματος	output key
κλειδί εισόδου	input key
κλιμακωσιμότητα	scalability
κόμβος	node
κόμβος ονομάτων	namenode

Λ	
λεξικολογική εμβέλεια	lexical scoping
λίστα	list
λογικός	logical– True/False
Μ	
μάθημα	lesson
μάθηση	learning
μεγάλα δεδομένα	big data
μέγιστη	max
μέγιστη απόσταση ή πλήρης σύνδεσμος	complete link
μέσα	mids
μέση τιμή	mean value
μέσος όρος της συστάδας	group average
μετασχηματισμός Δεδομένων	transformation
μη επιβλεπομένη μάθηση	unsupervised learning
μητρώ	matrix
μηχανική μάθηση	machine learning
μνήμη προσωρινής αποθήκευσης	buffer
μοντέλα πρόβλεψης	predictive models
Ο	
ομαδοποιημένο ραβδόγραμμα	grouped barplot
ονομαστικός	nominal
όσο	while
ουρά δεδομένων	data queue
ουρά επιβεβαίωσης	ack queue
Π	
πακέτο	package
παλινδρόμηση	regression
παράγοντες	factors
περιβάλλον	environment
περιγραφέας αρχείου	descriptor
περιγραφικά μοντέλα	descriptive models
περιθώριο	marginal
πίνακας συνάφειας	contingency matrix
πλήρης σύνδεσμος	complete link
πλαίσια δεδομένων	data frames
προεπεξεργασία Δεδομένων	preprocessing
προεπιλεγμένος	default
προσεγγίσιμα ή πυκνά-προσεγγίσιμα σημεία	density-reachable points
πρότυπα	patterns
πρότυπα προσπέλασης σε δεδομένα συνεχούς ροής	streaming data access patterns
Ρ	
ραβδόγραμμα	bar chart
ράφι	rack
ρεύμα εξόδου	output stream
Σ	
στηλοθέτης	tab

στοιχείο	item
στοιβαγμένο ραβδόγραμμα	stacked barplot
συλλογή δεδομένων	data collection
συνένωση	concatenate
συνεργατικό φιλτράρισμα	collaborative filtering
σύνθετος	complex
συντελεστής μεταβλητότητας	coefficient of variation
συσσωρευτικοί αλγόριθμοι	agglomerative algorithms
συστάδες	clusters
συσταδοποίηση	clustering
σχέσεις	relations
Τ	
ταυτότητα	identity
τιμή αποτελέσματος	output value
τυπική απόκλιση	standard deviation
Υ	
υπερκαλύπτω	override
υποψήφιοι	candidates
υψηλή διαθεσιμότητα	high availability
υψηλή παραγωγικότητα	throughput
ύψωση μιας εξαίρεση	raise of an exception
Χ	
χαρακτήρας	character
χαρακτηριστικά	features
χρόνος αναζήτησης	seek time
χρονοσειρές	time series
χτύπος καρδιάς	heartbeat
χώρος εργασίας	workspace
χώρος ονόματος πακέτου	package namespace
Ψ	
ψευδής	false