

Data Mining / Extragerea cunoștințelor din date

Lab 1:

Seturi de date: caracteristici, formate, colecții Introducere în Rattle (R), Scikit-learn (Python) și Weka (Java)

I. Seturi de date

I.1. Caracteristici și formate

Datele de procesat pot fi de diferite tipuri

- structurate (e.g. matrice de date, tabele din baze de date relaționale)
- semi-structurate (e.g. fișiere XML, fișiere de tip log)
- nestructurate (e.g. documente text)

Date structurate:

- seturi de instanțe (înregistrări, articole)
- fiecare instanță conține valori corespunzătoare unor atribute (caracteristici, câmpuri)
- atributele pot fi de diferite tipuri:
 - calitative (valorile lor sunt obiecte simbolice, e.g. simboluri sau șiruri de caractere):
 - nominale (e.g. naționalitate, gen, religie, stare civilă etc)
 - logice/ binare (e.g. prezența sau absența unei caracteristici specifice)
 - ordinale (e.g. nivel de satisfacție (“scăzut”, “mediu”, “ridicat”), calificativ (“insuficient”, “suficient”, ”bine”, “foarte bine”, “excelent”))
 - cantitative (valorile acestora sunt numere care aparțin unor mulțimi discrete sau unor intervale de valori continue):
 - întregi (e.g. număr de copii, vârsta în ani, număr de accesări ale unei pagini web)
 - reale (e.g. temperatura, înălțime, greutate)
- operații posibile asupra valorilor atributelor:
 - verificarea egalității, numărul de apariții (frecvența): toate tipurile de atribute (nominale, logice, binare, ordinale, întregi, reale)
 - comparare, ierarhizare: ordinale, întregi, reale
 - comparare, ierarhizare, adunare, scădere: întregi, reale
 - comparare, ierarhizare, adunare, scădere, înmulțire, împărțire: reale

I.2. Colecții de date

UCI Machine Learning repository (<http://archive.ics.uci.edu/ml/>)

- peste 400 seturi de date grupate pe categorii
- un set de date conține de obicei un fișier descriptiv (“names”) și fișiere de tip csv care conțin datele propriu-zise (“data”)

Kaggle platform (<https://www.kaggle.com/>)

- date: peste 14500 seturi de date de dimensiuni diverse (de la câteva sute de kB la peste un GB), stocate în diverse formate (csv, json, sqLite, BigQuery) din diferite categorii
- soluții (kernels): Python notebooks, R scripts

- forum discuții
- competiții active

KDD competitions (<http://www.kdd.org/kdd-cup>)

- date aferente competițiilor anuale din domeniul Data Mining and Knowledge Discovery organizate de către ACM

Exercițiul 1: Descărcarea și analiza particularităților câte unui set de date de la UCI Machine Learning repository:

- Număr mic de atribute și de instanțe (e.g. Iris dataset)
- Număr mic de atribute și mare de instanțe (e.g. DBWorld emails dataset)
- Număr mare de atribute și de instanțe
- Specific pentru clasificare
- Specific pentru grupare
- Specific pentru regresie

[Bioinfo:] Analizați particularitățile următoarelor seturi de date

<http://archive.ics.uci.edu/ml/datasets/Mice+Protein+Expression>

<http://archive.ics.uci.edu/ml/datasets/Diabetic+Retinopathy+Debrecen+Data+Set>

<http://archive.ics.uci.edu/ml/datasets/E.+Coli+Genes>

<http://archive.ics.uci.edu/ml/datasets/Ecoli>

Exercițiul 2: Analiza unor seturi de date de la Kaggle:

- Titanic data set** (<https://www.kaggle.com/c/titanic>). Identificați: numărul de instanțe, numărul și tipul atributelor, tipul de prelucrare.
Exemple de prelucrări asupra setului:
<https://www.kaggle.com/hiteshp/head-start-for-data-scientist>
<https://www.kaggle.com/vin1234/best-titanic-survival-prediction-for-beginners>
- Google jobs** (<https://www.kaggle.com/niyamatalmass/google-job-skills>). Identificați: tipuri de prelucrări posibile
- [Bioinfo]: **Personalized Medicine:** <https://www.kaggle.com/c/msk-redefining-cancer-treatment> . Analizați conținutul fișierelor din setul de date
- Consultați competițiile curente de la Kaggle

II. **Introducere în Rattle = R Analytical Tool To Learn Easily =A Graphical User Interface for Data Mining using R** (<https://rattle.togaware.com/>)

Rattle este o aplicație dezvoltată în R care oferă interfață grafică și încorporează mai multe tehnici și modele de analiză a datelor. Permite testarea rapidă a modelelor de analiză fără a necesita cunoașterea pachetelor și funcțiilor R corespunzătoare. Util pentru familiarizarea cu metodele de bază. Oferă funcții specifice etapelor principale de analiză a datelor:

- Încărcare set de date
- Selectare instanțe și atribute pentru analiză
- Explorarea datelor (sumarizare și vizualizare)
- Transformarea datelor
- Construirea modelului de analiză
- Evaluarea modelului

- Exportarea modelului

Instalare:

```
>install.packages("rattle")
```

Utilizare:

```
>library(rattle)
>rattle()
```

Prelucrări:

Incărcare date: **Data** -> completare Filename + upload -> Execute

Setare atribute: intrare/ target (atribut răspuns)

Obs: permite citire fișiere csv, arff (Weka), seturi de date din pachete R, baze de date (prin conexiune ODBC), corpus de texte

Explorare date: **Explore**

Summary – determinare caracteristici statistice

Distributions – vizualizare date

Correlation – analiza corelației dintre attribute (coeficienți de corelație: Pearson, Spearman, Kendall)

Principal Components – analiza compenentelor principale (identificarea direcțiilor de variabilitate maximă)

Test: teste statistice

Transform: scalare (**Rescale**) / completarea valorilor absente (**Impute**) / conversii de tipuri (**Recode**) / eliminare instanțe (**Cleanup**)

Model clasificare și regresie: arbori de decizie (**Tree**), colecții de arbori (**Forest**), clasificatori bazați pe vectori support (**SVM**), clasificatori bazați pe regresie logistică (**Linear**), rețele neuronale – doar pentru regresie (**Neural Net**)

Cluster: identificare grupuri (clustere) în date – algoritmi partiționali (**Kmeans**), algoritmi ierarhici aglomerativi (**Hierarchical**), algoritmi pentru biclustering (**BiCluster**)

Associate: construire reguli de asociere

Obs:

- după selectarea unei prelucrări și setarea atributelor se apasă **Execute**
- unele prelucrări necesită instalarea unor pachete R (Rattle realizează instalarea pachetelor)
- toate prelucrările efectuate pot fi exportate în script-uri R care pot fi utilizate ulterior

Exercițiul 3:

1. Se încarcă setul **Iris** in Rattle și se analizează sumarul:
 - a. Număr de instanțe
 - b. Număr de attribute
 - c. Pentru fiecare atribut: valori posibile și număr de apariții
2. Folosiți **View/Edit** pentru a vedea matricea de date și pentru a efectua modificări asupra datelor.
3. Utilizați **Explore** pt a analiza:
 - a. distribuția valorilor fiecărui atribut în fiecare dintre clase
 - b. corelația dintre valorile atributelor; Care dintre attribute sunt mai puternic corelate? Cum ar putea fi utilizată această informație?

Exercițiul 4:

- a. Pentru seturile de date din fișierul [carr.arff](#) și [autoMPG.arff](#) analizați:
 - Tipul atributelor
 - Distribuția datelor pe clase
 - Distribuția valorilor atributelor grupate pe clase
 - Corelația dintre atributele numerice (unde e cazul)
- b. **[Bioinfo]** Pentru seturile de date din fișierele [breast-cancer.arff](#), [Data_Cortex_Nuclear.csv](#), [train_GeneticMutation.csv](#) identificați:
 - Tipul atributelor
 - Distribuția datelor pe clase
 - Distribuția valorilor atributelor grupate pe clase

III. Introducere în Scikit-learn (<https://scikit-learn.org/>)

Scikit-learn este un pachet Python open-source (care se bazează pe [numpy](#), [scipy](#) și [matplotlib](#)) care oferă o serie de funcții utile în analiza datelor și învățarea automată.

Instalare (necesită [numpy](#) și [scipy](#)) – folosind [pip](#) sau [conda](#):

```
pip install -U scikit-learn
```

sau

```
conda install scikit-learn
```

Funcții implementate pentru:

- Pre-procesare:
 - Conversii (discretizare)
 - Transformări (scalare, standardizare, normalizare)
 - Extragere caracteristici (analiza componentelor principale – Principal Component Analysis)
- Clasificare și regresie:
 - Arbori de decizie (Decision Trees)
 - Clasificatori bazați pe instanțe (k Nearest Neighbours)
 - Clasificator Bayesian (Naïve Bayes)
 - Clasificatori și modele de regresie bazate pe vectori support (Support Vector Machines)
 - Clasificatori și modele de regresie bazate pe rețele neuronale (Multilayer Perceptron)
 - Meta-modele
- Grupare (clustering)
 - Metode partiționale (kMeans)
 - Metode ierarhice (algoritmi aglomerativi)
 - Metode bazate pe densitate (DBSCAN)
 - Metode spectrale
- Evaluarea modelelor

IV. Introducere în Weka <http://www.cs.waikato.ac.nz/ml/weka/>

IV.1. Ce este Weka?

WEKA = Waikato Environment for Knowledge Analysis

Pachet open-source dezvoltat la Waikato University ce încorporează o serie de algoritmi folosiți în data mining:

- Vizualizarea datelor
- Pre-procesarea datelor (75 algoritmi implementați)
- Selecția atributelor (cca 25 algoritmi implementați)
- Clasificare (mai mult de 100 algoritmi implementați)
- Grupare (cca 20 algoritmi implementați)
- Extragerea regulilor de asociere

Weka este implementat în Java și rulează pe: Windows, Linux, Mac

IV.2. Cum poate fi instalat?

Descărcare de la <http://www.cs.waikato.ac.nz/ml/weka/downloading.html>

IV.3. Care sunt principalele componente ale Weka?

- **Graphical User Interface:**
 - **Explorer** – utilizat pt a aplica prelucrări specific
 - **Experimenter** – utilizat pt a efectua analize comparative între mai multe metode pe mai multe seturi de date
 - **KnowledgeFlow** – interfață grafică ce permite efectuarea unui flux de prelucrări
 - **Workbench**
- **Simple CLI** - utilizare prin linia de comandă
- **Java API**

IV.3.1. Explorer:

- Datele de prelucrat pot fi încărcate:
 - Din fișier (**Open File**) – formate uzuale: **arff** (Weka format), **csv** (comma separated values)
 - De la o adresă web (**Open URL**) dintr-o bază de date (**Open DB**)
- Pot fi generate date aleator folosind **Generate**
- Setul de date poate fi vizualizat și editat folosind **Edit**
- Categoriile de prelucrări:
 - Vizualizare
 - Preprocesarea datelor
 - Selecția atributelor
 - Clasificare
 - Grupare
 - Reguli de asociere
 - Predicție

IV.3.2. Experimenter:

- Permite compararea a mai multor metode asociate aceluiași tip de prelucrare aplicat unuia sau mai multor seturi de date (e.g. classification)
- Analiza statistică se bazează pe testul Student (cu corecții pentru comparații multiple)

IV.3.3. Knowledge Flow:

- **Workflow tipic:** “data source” ->”filter”->”classifier”->”evaluator”
- **Exemplu**
 - **Data sources: Arff loader** - **data set** connection to

- **Evaluation: Cross validation foldMaker** – [training set](#), [test set](#) connection to
- **Classifiers, Bayes, Naïve Bayes** – [batch classifier](#) connection to
- **Classifier Performance Evaluation** – [text](#) connection to
- **Text viewer**

Remark: conexiunile se selectează prin click dreapta pe icoana cu prelucrarea

- **Activare flux:**
 - Click buton dreapta pe Arff și [Start loading](#)

IV.3.4. Command line interface (exemplu pentru Weka-3-6):

- **Setare variabile de mediu pt java**
 - setenv WEKAHOME c:\Program Files\Weka-3-6
 - setenv CLASSPATH \$WEKAHOME/weka.jar:\$CLASSPATH
- **use a weka function**
 - java weka.classifiers.j48.J48 -t \$WEKAHOME/data/iris.arff

IV.3.5. Java API:

- Add weka.jar to class path
- Example of using the J48 classifier

```
import weka.core.Instances;
import weka.classifiers.trees.J48;
...
Instances data = ... // from somewhere
String[] options = new String[1];
options[0] = "-U"; // unpruned tree
J48 tree = new J48(); // new instance of tree
tree.setOptions(options); // set the options
tree.buildClassifier(data); // build classifier
```

IV.3.6. Formatul ARFF – Attribute Relation File Format (formatul standard al seturilor de date in Weka)

- Antet:
 - Comentarii (%)
 - Identificator set de date: [@relation dataset_name](#)
 - Lista de attribute (fiecare atribut este caracterizat prin nume si tip): [@attribute attr_name type](#)
Obs: [în cazul atributelor discrete](#) (care nu au multe valori) tipul este chiar setul de valori posibile; [în cazul atributelor numerice discrete](#) tipul se specific prin [integer](#); [în cazul atributelor numerice continue](#) tipul se specific prin [real](#)
- Matricea de date:
 - Specificate prin [@data](#)
 - Fiecare linie corespunde unei instanțe;valorile atributelor sunt separate prin virgule

Exemplu 1:

```
@relation car
@attribute buying {vhigh,high,med,low}
@attribute maint {vhigh,high,med,low}
@attribute doors {2,3,4,5more}
@attribute persons {2,4,more}
@attribute lug_boot {small,med,big}
```

```

@attribute safety {low,med,high}
@attribute class {unacc,acc,good,vgood}
@data
vhigh,vhigh,2,2,small,low,unacc
vhigh,vhigh,2,2,small,med,unacc
vhigh,vhigh,2,2,small,high,unacc
vhigh,vhigh,2,2,med,low,unacc
vhigh,vhigh,2,2,med,med,unacc

```

Exemplu 2:

```

@relation 'autoPrice.names'
@attribute symboling real
@attribute normalized-losses real
@attribute wheel-base real
@attribute length real
@attribute width real
@attribute height real
@attribute curb-weight real
@attribute engine-size real
@attribute bore real
@attribute stroke real
@attribute compression-ratio real
@attribute horsepower real
@attribute peak-rpm real
@attribute city-mpg real
@attribute highway-mpg real
@attribute class real
@data
2,164,99.8,176.6,66.2,54.3,2337,109,3.19,3.4,10,102,5500,24,30,13950
2,164,99.4,176.6,66.4,54.3,2824,136,3.19,3.4,8,115,5500,18,22,17450
1,158,105.8,192.7,71.4,55.7,2844,136,3.19,3.4,8.5,110,5500,19,25,17710
1,158,105.8,192.7,71.4,55.9,3086,131,3.13,3.4,8.3,140,5500,17,20,23875

```

Exercițiul 5:

4. Se încarcă setul [Iris](#) în Weka (click pe [Open file](#)) și se analizează sumarul:
 - a. Număr de instanțe
 - b. Număr de atribute
 - c. Pentru fiecare atribut: valori posibile și număr de apariții
5. Folosiți [Edit](#) pentru a vedea matricea de date și pentru a efectua modificări asupra datelor.
6. Utilizați [Visualize All](#) pt a vedea distribuția valorilor fiecărui atribut în fiecare dintre clase (ultimul atribut corespunde implicit clasei). Identificați perechea de attribute care ar permite separarea datelor pe clase (e.g. (sepal width, petal length)).
7. Utilizați filtrul adecvat din Weka pentru a elimina atributele care nu sunt considerate relevante pentru clasificare.

Exercițiul 6:

1. Parcurgeți aceleași etape pentru setul de date din fisierul [car.arff](#)

- Obs: Principala diferență între seturile iris și car este faptul că primul conține doar atribute numerice (cu excepția celui de clasă) pe când al doilea conține atribute nominale/ordinale.
2. Incărcați fișierul [autoMpg.arff](#) (conține informații privind consumul de combustibil (mpg=miles per gallon)) și analizați conținutul folosind [Edit](#).
Obs 1: Majoritatea atributelor (incluzând cel referitor la clasa) sunt numerice; dacă atributul de clasă este numeric atunci datele sunt adecvate pentru sarcini de predicție (estimează valoarea “miles per gallon” în funcție de caracteristicile mașinii).
Obs 2: Câmpurile vide (gray) corespund unor valori absente.
 - a) Utilizați [Visualize All](#) pentru a vedea care dintre atribute sunt corelate cu valoarea mpg value
 3. Incărcați fișierul [supermarket.arff](#) (conține date pentru analiza coșului de cumpărături)

Exercițiul 7: comparați performanțele mai multor tipuri de clasificatori pe două seturi de date (mai multe detalii privind metodele de clasificare vor fi prezentate în cursul 3 și lab 3)

1. Selectează [Experimenter](#) din panoul Weka
2. Click [New](#) pt a crea un nou experiment
3. Adaugă seturile de date: [iris.arff](#) și [breast-cancer.arff](#)
4. Adaugă algoritmi: [zeroR](#), [oneR](#), [naiveBayes](#), [J48](#) (oneR și zeroR sunt din grupul “Rules”, naiveBayes este din grupul “Bayes” și J48 este din grupul “Tree”)
5. [Run](#) (rulează experimentul)
6. [Analyze](#) (analizează rezultatele):
 - a. Click pe [Experiment](#)
 - b. [Perform](#) (efectuează testul statistic)
 - c. Interpretează testului statistic (folosind zeroR ca metodă de referință)
 - i. $v / *$ semnifică: $v =$ nr cazuri (seturi de date) pe care metoda curentă e mai bună ca cea de referință; $/ =$ nr cazuri (seturi de date) pe care metoda curentă nu diferă semnificativ de cea de referință; $*$ = nr cazuri (seturi de date) pe care metoda curentă e mai puțin bună ca cea de referință;

Anexa. Mediul R (<http://cran.r-project.org/>)

R este o suită integrată de instrumente software destinate prelucrării datelor, efectuării unor calcule și vizualizării grafice. R conține o colecție mare de pachete destinate diferitelor tipuri de prelucrări (în principal de natura statistică) și un limbaj care permite extinderea facilităților prin definirea de noi funcții și pachete.

Caracteristici ale limbajului R.

Aspecte generale:

- Este case-sensitive
- Cea mai simplă și frecvent utilizată comandă (de atribuire) se specifică prin
`nume_variabila <- valoare` sau
`nume_variabila = valoare` sau
`assign("nume_variabila",valoare)`
- Comenzile (instrucțiunile) se separă prin ; sau se plasează pe linii separate
- Comenzile elementare pot fi grupate prin specificare între { și }
- Comentariile se specifică prin #
- Revenirea la o comandă anterioară se poate face cu "săgeată sus"
- Execuția unei linii sau a unui grup de linii dintr-un fișier de comenzi se face prin:
 - Selecție linie/ grup
 - Tastare F5
- Execuția comenzilor stocate într-un fișier se face prin
`source("fișier_comenzi.R")`

Obiecte în R:

- **Scalari** (valori individuale care pot fi interpretate ca vectori cu un singur element):
 - Valori numerice (întregi, reale, complexe)
 - Conversii între tipuri: `var2=as.tip_nou(var1)` (ex: `as.character(2)` returnează "2")
 - Logice (constante predefinite: TRUE sau T, FALSE sau F)
 - Operatori logici: ! (negatie), | (disjunctie), & (conjunctie)
 - Operatori relationali: <, >, <=, >=, == (egalitate), != (diferit)
 - Simboluri
 - Constante uzuale:
 - NA (valoare nedefinită)
 - Pi
- **Șiruri de caractere:** se specifică între ghilimele sau apostroafe.
 - Concatenarea șirurilor se poate realiza cu funcția `paste`.
 - Conversii: `c2s`: vector de caractere -> șir de caractere; `s2c`: șir de caractere în vector de caractere
 - Specificare subsir: `substring(subsir, indiceStart, indiceFinal)`
 - Analiza:
 - `words.pos(subsir,sir)`: determina toate pozițiile de start ale subșirului în șirul dat
- **Vectori** (colecții omogene și ordonate de valori accesibile prin indice de poziție).

- Construire vector: `nume_vector <- c(el1,el2,...,eln)`
- Accesare element: `nume_vector[indice]` (indicii sunt numerotați începând de la 1)
- Operații:
 - operațiile aritmetice (+,-,*,/,^) și funcțiile matematice unare (exp, log, sqrt,sin,cos etc.) aplicate asupra vectorilor acționează la nivelul fiecărui element. Pentru operatorii binari nu e obligatoriu ca cei doi vectori să aibă același număr de elemente – rezultatul va avea lungimea celui mai lung vector iar operandul mai scurt va fi completat ciclic cu aceleași elemente. De exemplu `a=c(1,2,3); b=c(4,1); c=a+b` va conduce la rezultatul `5 3 7` (c este completat la (4,1,4))
 - determinarea numărului de elemente din vector: `length(vector)`
 - determinare minim respectiv maxim: `min(vector)` respectiv `max(nume_vector)`
 - calcul suma respectiv produs elemente: `sum(vector)` respectiv `prod(vector)`
 - sortare crescătoare: `sort(vector)`
 - selecție elemente: `nume_vector[secventa_indici]` (ex: `vector[c(1,3,5)]` returnează elementele de pe pozițiile 1, 3 și 5); secvența de indici poate fi creată prin orice funcție (ex: `vector[vector>0]` returnează vectorul conținând doar elementele pozitive din vectorul inițial.
 - Replicarea unui obiect: `rep(obiect,times=nr_repetari)`. De exemplu pentru `b=c(4,1)` prin `rep(b,times=3)` se va genera `4 1 4 1 4 1`
- Generarea unor secvențe de valori:
 - Valori consecutive: `valoare_initiala:valoare_finala` (ex: `1:5` generează vectorul corespunzător lui `c(1,2,3,4,5)`)
 - Valori separate printr-un pas specificat: folosind funcția `seq(valoare_initiala, valoare_finala, by=pas)` (ex: `seq(1, 5, by=1)`)
- **Liste** (colecții nu neapărat omogene de obiecte accesibile prin indice sau nume)
 - Construire lista: `lista=list(element1, element2,...)`; fiecare element al listei poate fi un obiect anonim sau cu nume (ex: `lista1=list("zi"=23,"luna"="februarie","an"=2014)`)
 - Accesarea valorilor elementelor:
 - Prin indici: `lista[[indice]]` (ex: `lista1[[1]]` va returna 23) (Obs: dacă se folosește `lista[indice]` se returnează atât valoarea cât și numele elementului.
 - Prin nume: `lista$nume_element` (ex: `lista1$an` va returna 2014)
 - Concatenarea listelor: `c(lista1,lista2,...)`
 - Modificarea unei liste: `lista[[indice]]=element_nou`
- **Tablouri** (uni și multi-dimensionale)
 - Construire: `tablou=array(vector_valori,dim=c(dim1,dim2, ...))`
ex: `matrice=array(1:6,c(2,3))`
 - Specificare elemente: `tablou[indice1,indice2,...]` (Obs: există multe variante de a accesa subseturi de elemente aflate pe diferite poziții în matrice)
 - Determinare dimensiuni matrici: `nrow` (numar de linii), `ncol` (numar de coloane)
 - Combinare tablouri: `cbind` (alipire orizontala – coloana dupa coloana), `rbind` (alipire verticala – linie dupa linie)
 - Operațiile aritmetice clasice (+,*,-,/) se pot aplica asupra unor tablouri fiind mapate la nivel de element (ca în cazul vectorilor). Pentru a calcula produsul a

două matrici (calculând produsul scalar al liniilor și coloanelor corespunzătoare) se poate folosi operatorul `%*%`

- **Tabele de date (data frame):** sunt tabele în care fiecare linie corespunde unei înregistrări și fiecare coloană corespunde unei caracteristici. Valorile de pe aceeași coloană trebuie să fie de același tip dar cele de pe coloane diferite pot fi de tipuri diferite.
 - Construire: `data.frame(ume_caracteristica1= valori_caracteristica1, ume_caracteristica_2= valori_caracteristica_2,...)`. Exemplu: `df=data.frame(a1=c(1,2,3),a2=c("a","b","c"))`
 - Completare prin citire din fisier: `read.table("nume_fisier")`. Fisierul trebuie să conțină pe prima linie numele caracteristicilor și toate liniile trebuie să conțină același număr de valori
- **Tabele de frecvențe**
 - Construire: `tabel_frecv=table(vector)` (ex: `table(c(2,1,3,1,2))`)

Obs: obiectele din R au atribute specifice care pot fi vizualizate folosind funcția `attributes` și modificate folosind `attr`. De exemplu prin `attributes(df)` se obțin valorile corespunzătoare pentru atributele: `$names`, `$row.names` și `$class` iar prin `attr(df,"names")=c("b1","b2")` se modifica numele caracteristicilor din "a1" și "a2" în "b1" respectiv "b2".

Structuri de control

- Prelucrări decizionale: `if (conditie) instructiune1 else instructiune 2`
- Prelucrări repetitive:
 - **For:** `for (contor in domeniu) instructiune`
(obs: in varianta cea mai simpla domeniul se specifica prin valoarea_iniciala:valoarea_finala)
 - **While:** `while(conditie) instructiune`
 - **Repeat:** `repeat instructiune` (intreruperea ciclului repeat se poate face doar prin break).

Definirea funcțiilor

- Definirea unei funcții: `nume_funcție = function(arg1, arg2, ...){instructiuni; return(rez)}`
- Apelul unei funcții: `nume_funcție(val1,val2,...)` (Obs: la apel în loc de a specifica doar valoarea unui argument (val1) se poate specifica o pereche de forma `nume_argument=valoare` (ex: `arg1=val1`). In acest caz ordinea parametrilor de apel nu trebuie să coincidă neapărat cu ordinea specificată la definirea funcției.
- Particularități:
 - Variabilele utilizate în cadrul unei funcții sunt implicit considerate variabile locale astfel că efectele operațiilor de asignare specificate cu `<-` sau `=` au doar efect local
 - Pentru a efectua o asignare cu caracter global (al cărui efect să rămână după revenirea din apel) se poate folosi operatorul de "superasignare": `<<-`

Exemplu definire funcție:

```
suma=function(v)
{s<-0
for (i in 1:length(v)) s<-s+v[i]
return(s)}
```

}
Exemplu de apel: `suma(1:5)` (funcția trebuie încărcată în prealabil folosind în fereastra de comandă: `File->Source R code` urmat de selectarea fișierului care conține definirea funcției)

Prelucrări grafice simple

- Graficul unei funcții: `plot(ume_funcție, limita_inferioara, limita_superioara)` (ex: `plot(sin,-2*pi,2*pi)`)
- Grafic liste puncte: `plot(vector_abscise,vector_ordonate)` (ex: `plot(a,a^2)` cu a un vector cu valori numerice)

Citirea datelor din fișiere

- Citire sub forma de obiecte de tip “data frame” (tabele de valori în care prima linie conține denumiri ale câmpurilor iar prima coloană conține numărul de ordine al înregistrării). Este adecvată pentru citirea din fișierele ce conțin înregistrări specificate pe câte o linie.
 - Apel: `read.table(“nume_fisier”)`
- Citire sub forma de vector cu elemente de tip vector (în cazul în care fișierul conține linii cu același număr de valori separate prin spații dar nu conține o linie cu antet). Un element al vectorului returnat corespunde unei coloane din fișier (rezultatul nu e o structură de tip matrice astfel ca pentru a accesa a treia valoare din linia 2 trebuie specificat `rez[[3]][[2]]`).
 - Apel: `scan(“nume_fisier”,lista_tipuri)` (ex: `scan(“fisier.dat”,list(“”,0))` – citește un fișier cu două coloane în care pe prima coloană se află caractere iar pe a doua valori numerice.