R Introductory Course

Louis-Claude CANON and Emmanuel JEANNOT

Project-Team Runtime
Labri/INRIA/Université Henri Poincaré

Grid’5000 school

April, 2010
Outline

1. 5 Killer Features
2. Basics
3. Types
4. Common Functions
### Data analysis tools

- linear and nonlinear modeling (regression)
- classical statistical tests (*e.g.*, normality test)
- time-series analysis
- classification
- clustering
- principal component analysis
Statistical Functions

Data analysis tools

- linear and nonlinear modeling (regression)
- classical statistical tests (e.g., normality test)
- time-series analysis
- classification
- clustering
- principal component analysis

Large collection of user-submitted packages

Comprehensive R Archive Network (CRAN)
Review website: http://crantastic.org/
5 Killer Features

Graphics (1)

PCA 5 vars
princomp(x = data, cor = cor)

Clustering 4 groups

Factor 1 [41%]

Factor 3 [19%]
```r
> qplot(depth, ..density.., data = diamonds, geom = "freqpoly",
+     xlim = c(58, 68), binwidth = 0.1, colour = cut)
```
Example

```r
> X <- c(3, 4, 5)
> (Y <- X^2)
[1]  9 16 25
> X > 4 & Y > 4
[1] FALSE FALSE TRUE
> sqrt(Y)
[1] 3 4 5
```
Example

```r
> X <- c(3, 4, 5)
> (Y <- X^2)
[1] 9 16 25
> X > 4 & Y > 4
[1] FALSE FALSE TRUE
> sqrt(Y)
[1] 3 4 5
```

Indexes

```r
> X[c(2, 3)]
[1] 4 5
> X[-c(2, 3)]
[1] 3
> X[X > 4]
[1] 5
```
Vector (Speed)

Comparable to Numerical Analysis Software

Most internals are coded in C and Fortran. Benchmark put it on par with matlab.
Function (Higher-Order)

Example

```r
> X <- list(c(3, 4, 5), c(1, 10))
> length(X)
[1] 2
> sapply(X, length)
[1] 3 2
```
Function (Parameters)

Default value and arbitrary order

```r
> test <- function(X, Y = 1, V, W = sqrt(V)) {
  return (X + Y + V + W)
}
> test(1, 0, 9)
[1] 13
> test(V = 9, X = 1)
[1] 14
```
5 Killer Features

Integrated Help

Manual

> ?length
Integrated Help

Manual

> ?length

Show function code

> test

function(X, Y = 1, V, W = sqrt(V)) {
  return (X + Y + V + W)
}
Outline

1. 5 Killer Features
2. Basics
3. Types
4. Common Functions
Implementation of the S programming language (started in 1996).
Lexical scoping semantics inspired by Scheme.
Part of the GNU project (GPL).
Objectives: statistical and graphical techniques.
Command line interface & several graphical user interfaces
R Environment

Read-eval-print loop
Interactive computer programming environment. Possible to save session when quitting (creates 2 files: .RData, .Rhistory).
R Environment

Read-eval-print loop

Interactive computer programming environment. Possible to save session when quitting (creates 2 files: .RData, .Rhistory).

Functions

- `ls` list current objects names
- `rm` remove objects
Assignment

> 4 -> T
> T <- 4
> T = 4
Syntax (1)

**Assignment**

```r
> 4 -> T
> T <- 4
> T = 4
```

**Functions**

```r
> test <- function(X) { return(X) }
> test(5)
[1] 5
```
Instruction

One instruction per line or separation with ";".

> T <- 3 ; U <- 6
Instruction

One instruction per line or separation with ";".

> T <- 3 ; U <- 6

Special values

NA, Inf, NaN
Related functions: is.na, is.infinite, is.nan
Outline

1. 5 Killer Features
2. Basics
3. Types
4. Common Functions
Vector

Common structure for storing several basic values

> T <- c(4, 3)
### Vector

Common structure for storing several basic values

\[
> T <- c(4, 3)
\]

Related functions

- `sum`, `max/min`, `range`, `mean/var`, `length`, `sort/order`
- `summary`, `fivenum` (give summary of the repartition of the values)
Types

Vector

Common structure for storing several basic values

```r
> T <- c(4, 3)
```

Related functions

sum, max/min, range, mean/var, length, sort/order
summary, fivenum (give summary of the repartition of the values)

Generating functions

```r
> seq(from = 2, to = 3, by = 0.1)
[1] 2.0 2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9 3.0
> 1:6
[1] 1 2 3 4 5 6
> rep(5, 3)
[1] 5 5 5
```
Vector Operations

Vector arithmetic

\[ 3 \times X^2 + \sqrt{Y / Z} + 1 \]
Vector Operations

**Vector arithmetic**

\[3 \times X^2 + \sqrt{Y/Z} + 1\]

**Logical vector**

Boolean: TRUE or FALSE

- & vectorized and
- | vectorized or
- any test if any element of an array is TRUE
- all test if all elements of an array are TRUE
Vector Operations

Vector arithmetic

\[ 3 \times X^2 + \sqrt{Y / Z} + 1 \]

Logical vector

Boolean: TRUE or FALSE

- \& vectorized and
- | vectorized or

any test if any element of an array is TRUE

all test if all elements of an array are TRUE

Vector indexes

\[ > (X + 1)[3:5] \]
\[ [1] \quad 4 \quad 5 \quad 6 \]

\[ > X[is.na(X)] <- 0 \]
Manipulation functions

```r
> paste("X", "Y", sep = "")
[1] "XY"
> substr("abcde", start = 2, stop = 4)
[1] "bcd"
```
### String

#### Manipulation functions

```r
> paste("X", "Y", sep = "")
[1] "XY"
> substr("abcde", start = 2, stop = 4)
[1] "bcd"
```

#### Vector names

The function `names` returns the labels of the values of a vector.

```r
> X <- 1:10
> names(X) <- rep("ee", 10)
> X
ee ee ee ee ee ee ee ee ee ee
1 2 3 4 5 6 7 8 9 10
```
Data types

Primitive

- Numeric (integer and double) and complex
- Character
- Logical
- Factor (nominal value or level)
Data types

**Primitive**
- Numeric (integer and double) and complex
- Character
- Logical
- Factor (nominal value or level)

**Collections**
- Vector
- Array (multi-dimensional, same type)
- List (several elements of any type)
- Data frame (several collections having the same size and any type)
Data types

**Primitive**
- Numeric (integer and double) and complex
- Character
- Logical
- Factor (nominal value or level)

**Collections**
- Vector
- Array (multi-dimensional, same type)
- List (several elements of any type)
- Data frame (several collections having the same size and any type)

The functions *attributes, class* and *mode* gives information on the data.
### Arrays

#### Basic operations

- **dim** returns the dimensions as a vector (or `nrow` and `ncol`)
- **cbind** combines two elements by columns
- **rbind** combines two elements by rows
- **t** transpose of a matrix
Arrays

### Basic operations

- **dim** returns the dimensions as a vector (or `nrow` and `ncol`)
- **cbind** combines two elements by columns
- **rbind** combines two elements by rows
  - **t** transpose of a matrix

### Array indexes (on T, a matrix or data frame)

- `T[2,1]` is the element in the second row and first column
- `T[,1]` is the first column (also `T[-(2:ncol(T))]`)
- `T[,-1]` is `T` without the first column
- `T[-(2:3),]` is `T` without the second and third rows
Arrays

Basic operations

- `dim` returns the dimensions as a vector (or `nrow` and `ncol`)
- `cbind` combines two elements by columns
- `rbind` combines two elements by rows
- `t` transpose of a matrix

Array indexes (on T, a matrix or data frame)

- `T[2,1]` is the element in the second row and first column
- `T[,1]` is the first column (also `T[,-(2:ncol(T))]`)
- `T[,-1]` is `T` without the first column
- `T[-(2:3),]` is `T` without the second and third rows

Matrix multiplication

- `A %*% B`
**read.table** function

Read data (numeric and character) put by column in a file. Many parameters: separator between fields (**sep**), number of lines to skip (**skip**), number of lines to read (**nrows**), character comment (**comment.char**), ...
**read.table function**

Read data (numeric and character) put by column in a file. Many parameters: separator between fields (`sep`), number of lines to skip (`skip`), number of lines to read (`nrows`), character comment (`comment.char`), ... 

**Example**

```r
> read.table("t")
V1 V2
1 r 1
2 a 2
```
Plotting

High-level

Initialize a new plot with axis/scale, title, label, . . .

- **plot**  most generic function that draws list of points given in first argument(s) with lines, points, . . .
- **pairs**  draw scatter plots from a matrix of points: the plot on the $i$-th row and $j$-th column is the $i$-th column of the matrix drawn (y-axis) against the $j$-th column of the matrix (x-axis)

**hist, boxplot**  special graphical objects to denote the statistical dispersion of the points
### Plotting

#### High-level

Initialize a new plot with axis/scale, title, label, . . .

- **plot** most generic function that draws list of points given in first argument(s) with lines, points, . . .
- **pairs** draw scatter plots from a matrix of points: the plot on the $i$-th row and $j$-th column is the $i$-th column of the matrix drawn (y-axis) against the $j$-th column of the matrix (x-axis)

- **hist, boxplot** special graphical objects to denote the statistical dispersion of the points

#### Other plotting functions

- **lines, points, rug**: add lines or points
- **text, arrow, legend, axis, box**: add basic decorations
- **ecdf, density**: to be used with *plot*
## Distribution Functions

### Structure

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>d*</td>
<td>probability density function</td>
</tr>
<tr>
<td>p*</td>
<td>cumulative density function</td>
</tr>
<tr>
<td>q*</td>
<td>quantile function</td>
</tr>
<tr>
<td>r*</td>
<td>draw number randomly</td>
</tr>
</tbody>
</table>
Common Functions

Distribution Functions

Structure

- $d^*$: probability density function
- $p^*$: cumulative density function
- $q^*$: quantile function
- $r^*$: draw number randomly

List

- Beta (rbeta, dbeta, pbeta, qbeta), binomial, exponential, gamma, geometric, normal, uniform, Weibull, ...
Flow Control

Condition

```r
if (all(X > 5) && i == 1) {
    # Code
}
```
Flow Control

Condition

```r
if (all(X > 5) && i == 1) {
    # Code
}
```

Loop

```r
for (i in 1:10) {
    # Code
}
```

Better to use vectorized operations it allows to call compiled routines, otherwise everything is interpreted.
Within the environment

Search within manual pages: "??" before any string
Simply type the name of a function to show its internal.
Within the environment

Search within manual pages: "???" before any string
Simply type the name of a function to show its internal.

R project

Several documents on the R project website: introduction, language definition, installation/administration, data import/export, writing extensions, internals.