Introduction to R

Chao-Jen Wong

Fred Hutchinson Cancer Research Center

29 July, 2010
Introduction

Atomic Vectors

Matrix and data.frame

Lists and Environments

Functions

Basic Lattice
Outline

Introduction

Atomic Vectors

Matrix and data.frame

Lists and Environments

Functions

Basic Lattice
Packages

R distributes software via *packages*.

- *CRAN* – primarily for statistics research and data analysis.
- *Bioconductor* – focus on analysis of high-throughput biological data.

Starting R

- Finding packages
- Installing packages
- Attaching packages.

> `library(HTSandGeneCentricLabs)`
Installing Packages

Install Bioconductor packages (and their dependencies)

> source("http://bioconductor.org/biocLite.R")
> biocLite()

Install from source archive

> pkg <- "myDir/HTSandGeneCentricLabs_1.0.0.tar.gz"
> install.packages(pkg, repos=NULL, type="source")
Getting Help in R

- help.start and HTML help button in the Windows GUI
- help and ?: `help('data.frame')`
- help.search, apropos
- browseVignettes - vignettes and corresponding R scripts
  - `browseVignettes("HTSandGeneCentricLabs")`
- R Mailing lists
> library(IRanges)
> ## what is on the search path?
> search()
> ls(2)
> ## package description of IRanges
> packageDescription("IRanges")

> ## What functionalities does IRanges provide?
> ls("package:IRanges")
> #help(package="IRanges")
> sessionInfo()
Outline

Introduction

Atomic Vectors

Matrix and data.frame

Lists and Environments

Functions

Basic Lattice
Atomic Vectors

Vector: one-dimensional array of items of the same type.

> # numeric
> L <- c(1.2, 4.3, 2.3, 4)
> W <- c(13.8, 22.4, 18, 18.9)
> # most of functions are vectorized
> length(L)

[1] 4

> area <- L * W
> area

[1] 16.56 96.32 41.40 75.60

Other basic data types:

> s <- "a string" # character
> t <- TRUE # logical
> i <- 1L # integer
> i <- 1+1i # complex
Functions for Creating Vectors

Functions

- `c` - concatenate
- `:` - integer sequences
- `rep` - repetitive patterns

```r
> 1:10
[1] 1 2 3 4 5 6 7 8 9 10
```

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

Exercise

1. Read the help page for `seq`
2. Use `seq` to generate a sequence of even integers between one to ten.
Subsetting Vectors

Naming

```r
> ## name the elements of a vector
> v <- c(a=1.1, b=2, c=100, d=50, e=60)
> v

 a    b    c    d    e
1.1  2.0 100.0  50.0  60.0
```

Subsetting with positive indices

```r
> v[c(1,3,4)]

 a    c    d
1.1 100.0  50.0
```

Subsetting with negative indices

```r
> v[-c(1:3)] # exclude elements

d    e
50  60
```
Subsetting Vectors

By Logical predicates

Vector subsets can be specified by logical TRUEs and FALSEs.

```r
> x <- 1:10
> x > 5

[1] FALSE FALSE FALSE FALSE FALSE TRUE
[7] TRUE TRUE TRUE TRUE

> x[x > 5]

[1] 6 7 8 9 10

NA as logical subscripts

> x[8:12]

[1] 8 9 10 NA NA```
Outline

Introduction

Atomic Vectors

Matrix and data.frame

Lists and Environments

Functions

Basic Lattice
Matrix

matrix - two-dimensional vector, all elements share a common type.

```r
> x <- matrix(1:25, ncol=5, dimnames=list(letters[1:5], LETTERS[1:5]))
> x

     A  B  C  D  E
a 1  6 11 16 21
b 2  7 12 17 22
c 3  8 13 18 23
d 4  9 14 19 24
e 5 10 15 20 25

> x[, 2]

     a  b  c  d  e
  6  7  8  9 10
```
Matrix

Exercise

1. Remove the second row and the fourth column from $x$
2. Subset $x$ to keep the 'D' column.
A special R structure.

Analogous to a table where each row represents a sample and each column an attribute of a sample.
```r
> df <- data.frame(type=c("case", "case", "control", "control"), time=rexp(4))
> df
type       time
1  case 0.8757854
2  case 0.8299656
3 control 1.1910723
4 control 2.2114602

> df$time
[1] 0.8757854 0.8299656 1.1910723
[4] 2.2114602

> names(df)
[1] "type" "time"
```
Outline

Introduction

Atomic Vectors

Matrix and data.frame

Lists and Environments

Functions

Basic Lattice
Lists

Recursive data structure – a list can contain other lists and other types of data structures.

```r
> lst <- list(a=1:4, b=c("X", "Y"),
+ uspaper=list(length=11, width=8.5))
> lst

$a
[1] 1 2 3 4

$b
[1] "X" "Y"

$uspaper
$uspaper$length
[1] 11

$uspaper$width
[1] 8.5
```
Subsetting Lists

- [[ – extracting a single element from a list
  > lst[[1]]
  [1] 1 2 3 4

- [ – extracting a sub-list of the list
  > lst[1]
  $a$
  [1] 1 2 3 4

- $ – accessing list elements by name.
  > lst[["b"]]
  [1] "X" "Y"
Environments

Implementation of a hash table – names are used to compute hash index and hash index is used to retrieve the value.

```r
> el <- new.env()
> el$a = 1:3
> assign("b", "ciao", el)
> ls(el)

[1] "a" "b"

> el[["a"]]

[1] 1 2 3

> el$b

[1] "ciao"
```
Creating a function

```r
> say <- function(name, greeting="hello")
+  {
+    paste(greeting, name)
+  }
> say("world")

[1] "hello world"
```

Function code can be viewed

```r
> colSums
```
Want to return more than one value? - make a list

```r
> circle <- function(radius) {
+   area <- pi * radius^2
+   circum <- 2 * pi * radius
+   return(list(area=area, cm=circum))
+ }
> circ <- circle(2)
> circ

$area
[1] 12.56637

$cm
[1] 12.56637
Exploring R object

*factor* - category

```r
> fac <- c(rep("normal", 2), rep("tumor", 3), "unknown")
> f <- factor(fac)
> class(f)

[1] "factor"
```

```r
> levels(f)

[1] "normal" "tumor" "unknown"
```

```r
> str(f)

Factor w/ 3 levels "normal","tumor",..: 1 1 2 2 2 3
```

Classes - arbitrary record type

```r
> class ? IRanges
```
Outline

Introduction

Atomic Vectors

Matrix and data.frame

Lists and Environments

Functions

Basic Lattice
Basics

▶ Provide high-level functions for visualization of multivariate data.
▶ Implements the Trellis graphics system - multiple panels.

> library(lattice)

<table>
<thead>
<tr>
<th>Function</th>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>xyplot()</td>
<td>Scatter Plot</td>
</tr>
<tr>
<td>dotplot()</td>
<td>Dot Plot</td>
</tr>
<tr>
<td>bwplot()</td>
<td>Box-and-Whisker Plots</td>
</tr>
<tr>
<td>densityplot()</td>
<td>Kernel Density Plot</td>
</tr>
<tr>
<td>histogram()</td>
<td>Histogram</td>
</tr>
<tr>
<td>contourplot()</td>
<td>Contour Plot of Surface</td>
</tr>
<tr>
<td>cloud()</td>
<td>3-D Scatter Plot</td>
</tr>
</tbody>
</table>

**Table:** High-level functions in lattice.
Basic Ideas

An example

> ## quakes: locations of earthquakes off Fiji
> Depth <- equal.count(quakes$depth, number=8, overlap=.1)
> xyplot(lat ~ long | Depth, data = quakes)
Basic Ideas

\[
> \text{xyplot}(y \sim x \mid c, \text{data}, \text{groups}=g)
\]

- formula: \(y \sim x \mid c\)
- primary variables: \(x\) and \(y\)
- conditional variable: \(c\) – a \texttt{factor} object, separate data into different panels
- group variable: \(g\), separate data into subgroups for superposition
- data: \texttt{data} – a \texttt{data.frame} object
Exercise

```r
> data(Indometh)
> df <- Indometh
> head(df)

Subject  time  conc
1       1  0.25  1.50
2       1  0.50  0.94
3       1  0.75  0.78
4       1  1.00  0.48
5       1  1.25  0.37
6       1  2.00  0.19

> class(df)

[1] "nfnGroupedData" "nfGroupedData"
[3] "groupedData"   "data.frame"
```
Selected Reference

- *Multivariate Data Visualization with R* by Deepayan Sarker.