Brief Intro to R for Flow Packages Users

Chao-Jen Wong

Fred Hutchinson Cancer Research Center

30 July, 2010

Outline

Packages

Repository

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; and attaching packages.

```
> ## attaching packages
> library(flowCore)
```

Installing Packages

Install Bioconductor packages (and their dependencies)

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

Install from the flowTrack package

```
> pkg <- "myDir/flowTrack_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`
- `RSiteSearch`
- R Mailing lists

Outline

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

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

```
> ## 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

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

```
    a     c     d
  1.1 100.0  50.0
```

Subsetting with negative indices

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

```
 d   e
```

Outline

Matrix

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

```
> 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.

Outline

data.frame

- A special R structure.
- Analogous to a table where each row represents a sample and each column an attribute of a sample.

data.frame

```
> df <- data.frame(type=c("case", "case",
+                         "control", "control"), time=rexp(4))
> df
     type       time
1    case 0.77739394
2    case 1.95270944
3 control 0.91402175
4 control 0.02171282

> df$time

[1] 0.77739394 1.95270944 0.91402175
[4] 0.02171282

> names(df)

[1] "type" "time"
```

Outline

Lists

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

```
> 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"
  ```

Outline

Functions

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

```
[1] "hello world"
```

Outline

The flowFrame and flowSet Classes

- `flowFrame` - a class representing the data contained in a FCS file.
  1. raw measurement
  2. keywords in the FCS files
  3. annotation for parameters (stains, sample names, range)
- `flowSet` - a collection of `flowFrame`.

The flowFrame and flowSet Classes

```
> library(flowCore)
> data(GvHD)
> class(GvHD)
[1] "flowSet"
attr(,"package")
[1] "flowCore"
> GvHD
A flowSet with 35 experiments.

An object of class "AnnotatedDataFrame"
  rowNames: s5a01, s5a02, ..., s10a07  (35 total)
  varLabels and varMetadata description:
    Patient: Patient code
    Visit: Visit number
    ...: ...
    name: NA
    (5 total)
```

flowFrame

Subsetting

```
> f[, "FSC-H"]
```

```
flowFrame object 's5a01'
with 3420 cells and 1 observables:
     name        desc range minRange
$P1 FSC-H FSC-Height  1024        0
    maxRange
$P1     1023
119 keywords are stored in the 'description' slot
```

Extracting raw data

```
> head(exprs(f))
```

Some Methods for `flowFrame`

- `exprs`
- `colnames`, `featureNames` - names
- `keyword`, `identifier` - FCS keywords

```
> keyword(f, "FILENAME")

$FILENAME
[1] "s5a01"
```

- `parameters` - parameter annotation
- `range` - dynamic range
- `plot`, `xyplot` - visualization (flowViz)
- `spillover` - spillover matrix
- `transform`, `filter`, `Subset` and etc. - actions

Some Methods for `flowFrame`

xyplot

```
> library(flowViz)
> xyplot(`FSC-H` ~ `SSC-H`, f)
```

- accessing `flowViz::xyplot`.
- formula: `` `FSC-H` ~ `SSC-H` ``. Variables `FSC-H` (Y axis of the plot) and `SSC-H` (X axis of the plot) are the primary variables; sparated `~`.
- data: a `flowFrame`.

Some Methods for `flowSet`

Working with `flowSet`

- `[`, `[[`, `$` - subsetting
- `sampleNames`, `colnames` - names
- `phenoData`, `pData` - metadata
- `fsApply` - `apply` family, `flowSet`-specific iterator

Actions items

compensation, transformation, normalization, filtering and gating

Some Methods for `flowSet`

Examples

```
> head(pData(phenoData(GvHD)))
```

	Patient	Visit	Days	Grade	name
s5a01	5	1	-6	3	s5a01
s5a02	5	2	0	3	s5a02
s5a03	5	3	6	3	s5a03
s5a04	5	4	12	3	s5a04
s5a05	5	5	19	3	s5a05
s5a06	5	6	26	3	s5a06

```
> ## loop over a flowset to get the range for the
> ## first three flowFrames
> fsApply(GvHD[1:3], range)
```

Selected Reference

- *Software for Data Analysis: Programming with R* by John Chambers.
- *R Programming for Bioinformatics* by Robert Gentleman.
- *Multivariate Data Visualization with R* by Deepayan Sarker.