R for Bioinformatics

Introduction, Programming, Data Analysis and Visualization
R Programming

Gang Chen
chengang@bgitechsolutions.com

November 9, 2013

Outline

Next

Class, Type and Dimension

Class, Type and Dimension

Everything in R is a object, every object has class, type and dimension.

```
class(obj)
typeof(obj)
dim(obj)
```

Data Types

```
obj <- 1
class(obj)
```

```
## [1] "numeric"
```

```
obj <- "Gang Chen"
class(obj)
```

```
## [1] "character"
```

```
obj <- 1:3
class(obj)
```

```
## [1] "integer"
```

```
ranges <- GRanges(seqnames = c("chr1", "chr2"), ranges =
   4351), end = c(2314, NA), width = c(NA, 1)), strand =
class(ranges)
```

```
## [1] "GRanges"
## attr(,"package")
## [1] "GenomicRanges"
```

```
class(list(a = 1, b = 2))
```

```
## [1] "list"
```

```
class(matrix(1:16, ncol = 4))
```

```
## [1] "matrix"
```

```
class(array(1:64, c(4, 4, 4)))
```

```
## [1] "array"
```

```
obj <- as.data.frame(obj)
class(obj)
```

```
## [1] "data.frame"
```

```
obj <- as.factor(c("male", "female"))
class(obj)
```

```
## [1] "factor"
```

Types

```
 obj <- 1
 class(obj)
## [1] "numeric"
 obj <- 1:3
 class(obj)
## [1] "integer"
 obj <- 1+2i
 class(obj)
## [1] "complex"
```

Operations

Operators

- `+, -, *, /, ==, =, <-`
- `^`
- `exp(), log(), log10(), log2()`
- `sqrt(), abs(), sin(), cos()`
- `round(), floor(), ceriling()`
- `factorial()`

Character

A character object is used to represent string values in R.

```
fname <- "Gang"
lname <- "Chen"
class(fname)

## [1] "character"
```

```
myPI <- 3.14
class(myPI)

## [1] "numeric"

myPI <- as.character(myPI)
class(myPI)

## [1] "character"
```

Character Operators

```
paste(fname, lname)

## [1] "Gang Chen"

substr("I am learning R", start = 6, stop = 13)

## [1] "learning"

sub("I am", "We are", "I am learning R")

## [1] "We are learning R"
```

Regular Expression

Regular Expressions == Problem

Some people,
when confronted with a problem,
think "I know, I'll use **regular expressions.**"
Now they have **two** problems.

Regular Expression in R

Regular Expression Functions

```
help(regex)
grep(), grepl(), regexpr(), gregexpr(), sub(), gsub()
```

Example

```
grep("a.", c("Gang", "Chen", "aab", "Ag", "ga"))

## [1] 1 3
```

Logical

```
u = TRUE
v = FALSE
u & v   # u AND v

## [1] FALSE

u | v   # u OR v

## [1] TRUE

!u   # negation of u

## [1] FALSE
```

?

```
4.3 - 0.7
```

```
## [1] 3.6
```

```
4.3 - 0.7 == 3.6
```

```
## [1] FALSE
```

```
0.7 + 3.6 == 4.3
```

```
## [1] TRUE
```

```
4.2/6
```

```
## [1] 0.7
```

```
0.7 * 6
```

```
## [1] 4.2
```

```
4.2/6 == 0.7
```

```
## [1] FALSE
```

Vector

A **vector** is a sequence of data elements of the same basic type.

```
a = c(1, 2, 3)
b = c(T, F, F, T)
chars = c("Gang", "Chen", "AA", "Aa", "aB")
```

Arithmetic operations of vectors are performed memberwise.

All operators are applied to vectors

```
a^2
```

```
## [1] 1 4 9
```

```
!b
```

```
## [1] FALSE  TRUE  TRUE FALSE
```

```
grep("a.", chars)
```

```
## [1] 1 5
```

Vector Arithmetic

```
a = c(1, 2, 3, 4, 5)
b = c(5, 4, 3, 2, 1)
c(a, b)

##  [1] 1 2 3 4 5 5 4 3 2 1

a + b

## [1] 6 6 6 6 6

a * b

## [1] 5 8 9 8 5
```

Recycling Rule:

```
d = c(1, 2)
a + d

## Warning: longer object
length is not a multiple of
shorter object length

## [1] 2 4 4 6 6
```

Vector Index

```
a = c("one", "two", "three", "four", "five")
a[3]
```

```
## [1] "three"
```

```
a[2:4]
```

```
## [1] "two"   "three" "four"
```

```
a[-3]
```

```
## [1] "one"  "two"  "four" "five"
```

```
a[8]
```

```
## [1] NA
```

Matrix Construction

```
mat = matrix(1:24, ncol = 6, nrow = 4, byrow = T)
mat
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    2    3    4    5    6
## [2,]    7    8    9   10   11   12
## [3,]   13   14   15   16   17   18
## [4,]   19   20   21   22   23   24
```

Matrix Index

```
mat[3, 3]
```

```
## [1] 15
```

```
mat[2, ]
```

```
## [1]  7  8  9 10 11 12
```

```
mat[, 4]
```

```
## [1]  4 10 16 22
```

```
mat[2:3, 3:4]
```

```
##      [,1] [,2]
## [1,]    9   10
## [2,]   15   16
```

```
dim(mat)
```

```
## [1] 4 6
```

```
ncol(mat)
```

```
## [1] 6
```

```
nrow(mat)
```

Matrix Arithmetic

```
A
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    5    9   13
## [2,]    2    6   10   14
## [3,]    3    7   11   15
## [4,]    4    8   12   16
```

```
A * B
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1   25   81  169
## [2,]    4   36  100  196
## [3,]    9   49  121  225
## [4,]   16   64  144  256
```

```
B
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    5    9   13
## [2,]    2    6   10   14
## [3,]    3    7   11   15
## [4,]    4    8   12   16
```

```
A %*% B
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   90  202  314  426
## [2,]  100  228  356  484
## [3,]  110  254  398  542
## [4,]  120  280  440  600
```

List

A list is a generic vector containing other objects.

```
n = c(2, 3, 5)
s = c("aa", "bb", "cc", "dd", "e
b = c(TRUE, FALSE, TRUE, FALSE,
x = list(n, s, b, 3)
```

```
x

## [[1]]
## [1] 2 3 5
##
## [[2]]
## [1] "aa" "bb" "cc" "dd" "ee"
##
## [[3]]
## [1]  TRUE FALSE  TRUE FALSE FA
##
## [[4]]
## [1] 3
```

List Slice

```
x[1]

## [[1]]
## [1] 2 3 5

x[c(2, 4)]

## [[1]]
## [1] "aa" "bb" "cc" "dd" "ee"
##
## [[2]]
## [1] 3
```

List Member

```
x[[3]]

## [1]  TRUE FALSE  TRUE FALSE FALSE

x[3]

## [[1]]
## [1]  TRUE FALSE  TRUE FALSE FALSE
```

Data Frame

A data frame is used for storing data tables. It is a list of vectors of equal length.

```
head(mtcars)
```

```
##                   mpg cyl disp  hp drat    wt  qsec vs am gear car
## Mazda RX4        21.0   6  160 110 3.90 2.620 16.46  0  1    4
## Mazda RX4 Wag    21.0   6  160 110 3.90 2.875 17.02  0  1    4
## Datsun 710       22.8   4  108  93 3.85 2.320 18.61  1  1    4
## Hornet 4 Drive   21.4   6  258 110 3.08 3.215 19.44  1  0    3
## Hornet Sportabout 18.7  8  360 175 3.15 3.440 17.02  0  0
## Valiant          18.1   6  225 105 2.76 3.460 20.22  1  0    3
```

Data Frame

```
mtcars[1, 2]
```

```
## [1] 6
```

```
mtcars["Mazda RX4", "wt"]
```

```
## [1] 2.62
```

```
ncol(mtcars)
```

```
## [1] 11
```

```
nrow(mtcars)
```

```
## [1] 32
```

Factor

```
gender <- c("male", "female")
class(gender)

## [1] "character"

gender <- as.factor(gender)
class(gender)

## [1] "factor"
```

Factor

```
group <- c(1, 2)
group[1] < group[2]

## [1] TRUE

class(group)

## [1] "numeric"

group <- as.factor(group)
group[1] < group[2]

## Warning: < not meaningful for factors

## [1] NA
```

Next

If else

```
if (something) {
    # do something
} else if (something) {
    # do something
} else {
    # do something
}
```

ifelse

```
ifelse(test, yes, no)
```

```
a <- c(2, 3, 4, 2, 5, 6, 7, 12)
ifelse(a%%2 == 0, a + 1, 0)

## [1]  3  0  5  3  0  7  0 13
```

Loop

```
for (var in seq) expr
while (cond) expr
repeat break
next
```

Loop

```
for (i in a) {
    if (i%%2 == 0) {
        print(i + 1)
    } else {
        print(0)
    }
}
```

```
## [1] 3
## [1] 0
## [1] 5
## [1] 3
## [1] 0
## [1] 7
## [1] 0
```

apply functions

```
apply()
lapply()
sapply()
tapply()
```

Function

```
add <- function(a, b) {
    a + b
}
add(1, 2)

## [1] 3

sapply(1:8, add, 3)

## [1]  4  5  6  7  8  9 10 11
```

Anonymous Function

```
sapply(1:8, function(a, b) {
    a + b
}, 3)

## [1]  4  5  6  7  8  9 10 11
```

Next

S4 Classes and methods

History

- 1976, Rick Becker and John Chambers, S on Honeywell OS
- Ported to UNIX, S2
- Around 1986, functional programming and object self-description, S3
- 1992, concept of classes and methods, S4
- 2010, Reference Classes (RC), R 2.12

appendix in Software for Data Analysis by Chambers

S4 Classes and methods

OO Systems in R

- S3
- S4
- RC
- Base Types

Best Reference: http://adv-r.had.co.nz/OO-essentials.html

S3

S4 Classes and methods

S4 in R

```
library(stats4)
library(pryr)

## Loading required package: Rcpp

y <- c(26, 17, 13, 12, 20, 5, 9, 8, 5, 4, 8)
nLL <- function(lambda) -sum(dpois(y, lambda, log = TRUE))
fit <- mle(nLL, start = list(lambda = 5), nobs = length(y))
isS4(fit)

## [1] TRUE

otype(fit)

## [1] "S4"

isS4(nobs)
```

S4 Classes and methods

Defining classes and creating objects

```
setClass("Person", slots = list(name = "character", age = "numeric"
setClass("Employee", slots = list(boss = "Person"), contains = "Per

alice <- new("Person", name = "Alice", age = 40)
john <- new("Employee", name = "John", age = 20, boss = alice)
```

S4 Classes and methods

access slots of an S4 object

```
alice@age
slot(john, "boss")
```

S4 Classes and methods

Creating new methods and generics

```
setGeneric("union")
setMethod("union", c(x = "data.frame", y = "data.frame"), function
    unique(rbind(x, y))
})
setGeneric("myGeneric", function(x) {
    standardGeneric("myGeneric")
})
```

Next

Standard I/O

```
scan()
print()
cat()
```

File I/O

Input

```
read.table()
readLines()
readChar()
readBin()
scan()
```

Output

```
write.table()
write()
```

Database I/O

```
library(RMySQL)   # for MySQL
library(RPostgreSQL)   # for PostgreSQL
library(XLConnect)   # for Excel
```