R Programming #2

JC Wang

Department of Statistics
Western Michigan University

October 8, 2007 / Talk #2
Objects

- Everything we deal with in R is object.
- Any object has two basic attributes: mode and length.
- The mode and length of an object \( \text{obj} \) can be accessed by function calls \text{mode}(\text{obj}) \text{ and } \text{length}(\text{obj}) \).
- With caution, the mode and length of an object can be set/changed by function calls \text{mode}(\text{obj}) \leftarrow \text{value} \text{ and } \text{length}(\text{obj}) \leftarrow \text{value} \).
- With few exceptions, each mode of objects have a constructor function \text{mode-name}(\text{length}=n, \ldots) \text{, mode-testing function } \text{is.mode-name}(\text{obj}) \text{, and mode-coercing function } \text{as.mode-name}(\text{obj}) \text{.}
Basic Objects

Listed in increasing order of complexity (listed in constructor, mode-testing, and mode-coercing triplets):

- **NULL, is.null(obj), as.null(obj,...)** (note: `c()` creates **NULL**)
- **logical(length=0), is.logical(obj), as.logical(obj,...)** (note: T, TRUE; and F, FALSE are logical values)
- **integer(length=0), as.integer(obj, ...), is.integer(obj)** (note: primarily used in .C and .FORTRAN, ...)
- **numeric(length=0), as.numeric(obj, ...), is.numeric(obj)**
- **complex(length.out=0, real=numeric(), imaginary=numeric(), modulus=1, argument=0), as.complex(obj, ...), is.complex(obj)** (examples for complex values: 1+0i, −2+3i)
Basic Objects
(continued)

- `character(length=0), as.character(obj, ...), is.character(obj)`
- `raw(length=0), as.raw(obj), is.raw(obj)"
Basic Objects
(continued)

- `character(length=0), as.character(obj, ...), is.character(obj)`
- `raw(length=0), as.raw(obj), is.raw(obj)`
- **Special constants**
  1. **NA** is a logical constant containing missing value indicator; it can be freely coerced to abovementioned modes except `raw`. `is.na(obj); use is.na(obj) <- index-values` to set elements to missing values. ("" and NA are different for character values)
  2. **NaN** is a numeric constant (Not a Number); `is.nan(obj). (example: log(-1) generates NaN)`
  3. **Inf** and **-Inf** are numeric constant (positive infinite and negative infinite); `is.finite(obj) & is.infinite(obj). (example: log(c(0/1,1/0)) generates numeric vector of elements −Inf and Inf)
• **Definition:** additional information associated with object. Some attributes force change to other data structures.

• **Access function:** `attr-name(obj)` or `attr(obj, "attr-name")`

• **Set/Replacement function:** `attr-name(obj) <-` or `attr(obj, "attr-name") <-`
Basic Objects

Attributes

• Definition: additional information associated with object. Some attributes force change to other data structures.

• Access function: `attr-name(obj)` or `attr(obj, "attr-name")`

• Set/Replacement function: `attr-name(obj) <-` or `attr(obj, "attr-name") <-`

• Example attributes that do not force change to other structures:
  1. `names` attribute associates elements with names (character strings)
  2. `comment` attribute add comments to an object.
dim attribute could force an object to change either matrix object or array object. Example:

```r
x <- 1:24; names(x) <- LETTERS[1:24]
y <- x; dim(y) <- c(4,6) # 4 by 6 matrix
z <- x; dim(z) <- c(4,3,2) # 4 by 3 by 2 array
w <- x; dim(w) <- c(5,5) # error
```
**Basic Objects**

Attributes (continued)

The `dim` attribute can force an object to change either a **matrix** object or an **array** object. Example:

```r
x <- 1:24; names(x) <- LETTERS[1:24]
y <- x; dim(y) <- c(4,6) # 4 by 6 matrix
z <- x; dim(z) <- c(4,3,2) # 4 by 3 by 2 array
w <- x; dim(w) <- c(5,5) # error
```

**Note:**

- both `y` & `z` are of numeric mode
- avoid using `dim` function to construct **matrix** or **array** objects.
Basic Objects
Subscripting

`obj` with names attributes; assume `x` is a numeric vector of indexes (positive integers), `a` is a character vector, `e` is a logical vector of the same length as `obj`

- `obj[x]` subscripts `obj` with indexes, the result is of length `length(x)`; note: NA for out-of-bound indexes
- `obj[−x]` subscripts all but elements indexed by `x`
- `obj[a]` subscripts elements named by `a`; note: NA for name not in `names(obj)`
- `obj[e]` subscripts elements where elements of `e` are TRUE
Basic Objects
Subscripting

`obj` with names attributes; assume `x` is a numeric vector of indexes (positive integers), `a` is a character vector, `e` is a logical vector of the same length as `obj`

- `obj[x]` subscripts `obj` with indexes, the result is of length `length(x)`; note: NA for out-of-bound indexes
- `obj[-x]` subscripts all but elements indexed by `x`
- `obj[a]` subscripts elements named by `a`; note: NA for name not in `names(obj)`
- `obj[e]` subscripts elements where elements of `e` are TRUE

Note: `obj[0]` returns `integer(0)`.
Basic Objects

matrix

\[
\text{matrix}(\text{data} = \text{NA}, \ nrow = 1, \ ncol = 1, \\
\text{byrow} = \text{FALSE}, \ \text{dimnames} = \text{NULL})
\]

\[
\text{as.matrix}(x, \ ...) \\
\text{is.matrix}(x)
\]
Basic Objects

**matrix**

```r
matrix(data = NA, nrow = 1, ncol = 1,
byrow = FALSE, dimnames = NULL)
```

```r
as.matrix(x, ...)
```

```r
is.matrix(x)
```

- `matrix(, 0, 0)` creates 0 by 0 matrix
- `matrix(F, 3, 4)` creates 3 by 4 matrix of logical mode
- `dimnames(a) <- list(letters[1:3], LETTERS[1:4])` names rows & columns
- `rownames(a) <- c("one", "two", "three")` names rows
- `colnames(a) <- c("x", "y", "z", "w")` names columns
Basic Objects
matrix (continued)

- `dim(a)` accesses dimension info
- `nrow(a)` gives number of rows
- `ncol(a)` gives number of columns
- **element-wise operation**: `A op B`; **operators**: `+ − ∗ / ^ %/% %%%; | &; > < >= <= == !=`
- matrix multiplication: `A %*% B`
- transpose: `t(A)`
- diagonal: `diag(A) <- value, diag(a)`
- **cross product**: `crossprod(A, B) gives A’B; crossprod(X) gives X’X`
- outer product (multiplication table): `a %o% b`
- kronecker product: `a %x% b`
Matrix subscripting

matrix \( A \) with dimnames attributes; \( x \) and \( y \) are numeric vectors of indexes; \( a \) and \( b \) are character vectors; \( e \) and \( f \) are logical vectors of respective lengths \( \text{nrow}(A) \) and \( \text{ncol}(A) \)
Basic Objects
Matrix subscripting

matrix \( A \) with dimnames attributes; \( x \) and \( y \) are numeric vectors of indexes; \( a \) and \( b \) are character vectors; \( e \) and \( f \) are logical vectors of respective lengths \( \text{NROW}(A) \) and \( \text{NCOL}(A) \)

- subscripting by indexes: \( A[x,], A[,y], A[x,y], A[-x,], A[-y], A[-x,-y] \)
- subscripting by names: \( A[a,], A[,b], A[a,b] \)
- subscripting by T/F: \( A[e,], A[,f], A[e,f] \)
- mix subscripting: \( A[-x,f] \)
- \( A[,1, \text{drop=F}] \) is a \( \text{NROW}(A) \) by 1 matrix
- all, except for last case, can appear at left side of \(<-\) for element replacement