Data Types and Basic Operations

Gerardo Ferrara

Master in Economics and Complexity, Collegio Carlo Alberto

Fall 2013

Objects

R has five basic classes of objects:

- Character
- Numeric
- Integer
- Complex
- Logical (True/False).

Attributes

R objects can have attributes:

- Names
- Dimensions (e.g. matrices, arrays)
- Class
- Length
- Other user-defined attributes/metadata.

Attributes of an object can be accessed using the **attributes()** function.

Entering Input

The **c()** function can be used to create vectors of objects.

$$> x <-c(0.3, 0.2) \qquad \# numeric$$

$$> x <-c(TRUE, FALSE) \qquad \# logical$$

$$> x <-c(F, T) \qquad \# logical$$

$$> x <-c(''z'', ''y'', ''f'') \qquad \# character$$

$$> x <-11 : 17 \qquad \# integer$$

$$> x <-c(2 + 0i, 3 + 4i) \qquad \# complex$$

Mixing Objects

When different objects are mixed in a vector, coercion occurs so that every element in the vector is of the same class.

$$\begin{array}{ll} > y <\!-c(1.3,''b'') & \#character \\ > y <\!-c(FALSE, 3) & \#numeric \\ > y <\!-c(''z'', TRUE) & \#character \end{array}$$

Explicit Coercion

Objects can be explicitly coerced from one class to another using the **as.*** functions, if available.

```
>x <-0 : 4
>class(x)
  [1] "integer"
>as.numeric(x)
  [1] 0   1   2   3   4
>as.logical(x)
  [1] FALSE   TRUE   TRUE   TRUE   TRUE
>as.character(x)
  [1] "0""1""2""3""4"
>as.complex(x)
  [1] 0 + 0i1 + 0i2 + 0i3 + 0i4 + 0i
```

Explicit Coercion (cont.)

Nonsensical coercion results in *NA*.

```
>x <-c("a","b","c","d")
>as.numeric(x)
  [1]  NA   NA   NA   NA
>as.logical(x)
  [1]  NA   NA   NA   NA
```

Numbers

- Numbers in R a generally treated as numeric objects (i.e. double precision real numbers).
- If you explicitly want an integer, you need to specify the **L** suffix.
- The special number **Inf** represents infinity; e.g. 1 / 0; **Inf** can be used in ordinary calculations (e.g. 1 / **Inf** is 0).
- The value **NaN** represents an undefined value (e.g. 0 / 0); **NaN** can also be thought of as a missing value.

Matrices

Matrices are vectors with a dimension attribute. The dimension attribute is itself an integer vector of length 2 (nrow, ncol):

$$
\begin{aligned}
&{>}m <\!- matrix(1:6, nrow = 2, ncol = 3)\\
&{>}dim(m)\\
&\quad [1]\;\; 2 \quad 3
\end{aligned}
$$

Matrices can also be created directly from vectors by adding a dimension attribute:

$$
\begin{aligned}
&{>}m <\!- 1:6\\
&{>}dim(m) <\!- c(2,3)
\end{aligned}
$$

Operating on Matrices

We distinguish computation with data frames from computation with matrices. We have element-wise computations:

$$\begin{aligned} &>m <\!- matrix(1:10, ncol = 2)\\ &>m + 1\\ &>m + m \end{aligned}$$

We also have matrix multiplication from linear algebra:

$$\begin{aligned} &>m \,\%*\%\, t(m)\\ &>m \,\%*\%\, m \qquad \#Error:\ non\ conformable\ matrices \end{aligned}$$

where **t()** is the matrix transpose function. If the matrix and vector dimensions do not conform, an error message results.
Linear algebra functions are available: **eigen**, **det**, **solve**, ...

List

Lists are a special type of vector that can contain elements of different classes. Lists are a very important data type in R and you should get to know them well. The following code is an example:

$$>x <- list(1, ''a'', TRUE, 1+4i)$$

$$>x$$

$$[[1]]$$

$$[1]\;\; 1$$

$$[[2]]$$

$$[1]\;\; ''a''$$

$$[[3]]$$

$$[1]\;\; TRUE$$

$$[[4]]$$

$$[1]\;\; 1+4i$$

Missing Values

Missing values are denoted by **NA** or **NaN** for undefined mathematical operations:

- **is.na()** is used to test objects if they are **NA**.
- **is.nan()** is used to test for **NaN**.
- **NaN** values are also **NA** but the converse is not true.

For example:

```
>x <−c(1, 2, NaN, NA, 4)
>is.na(x)
  [1]  FALSE   FALSE   TRUE   TRUE   FALSE
>is.nan(x)
  [1]  FALSE   FALSE   TRUE   FALSE   FALSE
```

Removing NA Values

This trick is used to remove missing values:

$$>x <\!\!-c(2,3,NA,4,NA,5)$$

$$>bad <\!\!-is.na(x)$$

$$>x <\!\!-x[!bad]$$

$$[1]\ \ 2\quad 3\quad 4\quad 5$$

Data Frames

Missing values are denoted by **NA** or **NaN** for undefined mathematical operations:

- They are represented as a special type of list where every element of the list has to have the same length.
- Each element of the list can be thought of as a column and the length of each element of the list is the number of rows.
- Unlike matrices, data frames can store different classes of objects in each column (just like lists); matrices must have every element be the same class.
- Data frames are usually created by calling **read.table()** or **read.csv()**.
- Can be converted to a matrix by calling **data.matrix()**.
- Data frames also have the *row.names* attribute.

Data Frames (cont.)

```
>x <−data.frame(foo = 1 : 4, bar = c(F, T, T, F))
>x
        foo   bar
  [1]    1    FALSE
  [2]    2    TRUE
  [3]    3    TRUE
  [4]    4    FALSE
```