R Cheat Sheet: Brief Introduction to Language Elements and Control Structures

Comments

from the hash to the end of the line

Basic (underlying) data-types

1) logical – Boolean TRUE/FALSE
2) integer – 32 bit signed integer number
3) double – double precision real number
4) character – text in quotes – strings
5) complex – complex numbers (3+2i)

Note: integer and double of mode numeric

NULL v NA

Common R objects 1) NULL is an object, typically used to
1) atomic vector – 1-N, all of only one mean the variable contains no object.
basic data type, can be named. R does 2) NA is a value that means: missing data
not have a single value object. Single item here
values are held in a length=1 vector. x <- NULL; is.null(x); y <- NA; is.na(y)
2) list – 1-N of any R object (including length(NULL); length(NA) # -> 0, 1
lists), list elements can have different Trap: can have a list of NULLs but not a
types, list elements can be named vector of NULLs. Can have a vector of NAs.
3) factor – 1-N of ordinal (ordered) or
categorical (unordered) data (typically Other non-number numbers (NA the first)
character to integer coding) 1) Inf # positive infinity
4) data.frame 1-M rows by 1-N cols, cols is 2) -Inf # negative infinity
a named list, the data for each column 3) NaN # not a number
is a vector/factor, rows can be named 1/0; 0/0 # -> Inf, NaN
5) matrix – numeric vector with 2
dimensions, 1-M rows by 1-N cols, rows Operators
and cols can be named +, -, *, / # addition, subtraction,
6) array – essentially a matrix with # multiplication, division
(typically) 3 or more dimensions ^ or ** # exponentiation
Note: While these are the most common %% # modulus
objects used for analysis, most things in R %/% # integer division
are objects that can be manipulated. %in% # membership
Note: Some objects only contain certain : # sequence generation
types (eg. matrix), or everything in the <, <=, ==, >=, >, != # Boolean comparative
object is of the same type (eg. vector) |, || # (vectorised/not vec)
&, && # (vectorised/not vec)

Indexing objects

Note: with few exceptions (&&, || and :)
Because objects contain multiple values, operators take vectors and return vectors.
understanding indexing is critical to R:
1) x[i], x[r, c] – can select multiple
2) x[[i]], x[[r, c]] – select single
3) xi, x"i" – select single by name
a) by number: x[5]; x[1:10]; x[length(x)]
b) by logic: x[T,F,T,F]; x[!is.na(x)]
c) by name: x['me']; x$me; x[c('a', 'b')]
Note: 2-dimension indexes are x[row, col]
Trap: x[i] and x[[i]] can return very to the start of the loop with the next var
different results from the same object

Flow control structures

1) if (cond) expr
2) if (cond) expr1 else expr2
3) for (var in seq) expr
4) while (cond) expr
5) repeat expr
Note: break exits a loop, next moves flow
Note: expressions typically enclosed in {}
But single expressions do not need the {}

Classes

Multiple expression on a line ; separated
R has class mechanisms for creating more
complex data objects. Common classes
include Date, ts (time series data), lm
(the results of a regression linear model).
These are often used like other objects.

Flow control functions

1) the vectorised if statement:
result <- ifelse(cond, expr1, expr2)
2) the switch statement (not vectorised):
switch(expr.string,
case1 = expr1,
case2 = expr2,
default = expr
)

Objects and variables

Objects can be assigned to variables: <-
Note: objects have mode/type, not variables
Note: if an object has a rule your code
will be quietly coerced to meet the rule:
x <- c(1, "2"); cat(x) # -> "1", "2"

expr.string

3 # default optional
expr.string evaluates to a char string
Note: cases not enclosed in quotes.