## EEB Open Source for Open Science 2014

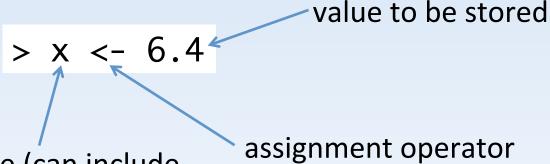
# Scientific Programming Using ( )



Thomas Olszewski Geology and Geophysics

# **(The Second of Second of**

### <u>Variable</u> = a 'place' in memory to store a value



variable name (can include any alphanumeric symbol, but must start with a letter)

An <u>operator</u> is a symbol that makes the computer <u>do</u> something – the assignment operator <u>assigns</u> the given value to the declared variable. This is <u>NOT</u> the same as an equal sign in a mathematical equation (as we will see).

# **Arithmetic Operators**

- > a+b Addition
- > a-b Subtraction
- > a\*b Multiplication
- > a/b Division
- > a^b Exponentiation

$$a = 6, b = 2$$
:

What does a\*b equal?

$$a = 6, b = 2, c = 1, d = 2$$
:  
What does  $a/b*(c+d)$  equal? 1 or 9? Why?

$$> a/b*(c+d)$$

$$(a = 6, b = 2, c = 1, d = 2)$$
  
What does  $a/b*(c+d)$  equal? 9.

The value of depends on the <u>order of operations</u>. In most computer languages, including R, the order of arithmetic operations from is:

- 1) left to right
- 2) parentheses
- 3) exponents and roots
- 4) multiplication and division
- 5) addition and subtraction

To get an answer of 1, an additional set of parentheses is needed:

<u>Functions</u> are used to package a series of commands. A function is a program or script that carries out a particular task. Prepackaged commands in R are examples of functions, but you can also create your own.

#### function name

assignment operator *parameter* – one or more values passed to the function (can be > foo <- function(x) {</pre> left blank if no parameters + a <- xseries of command in + b < - 3squiggly brackets + c <- a\*b + #unread comment ← lines beginning with "#" are ignored -+ return(c) useful for comments and instructions + } this command spits out the result of the function

```
> foo
function(x) {
a <- x
b <- 3
c <- a*b
return(c)
}</pre>
```

Typing the name of the function returns its contents (this works for prepackaged functions as well as your own).

Typing the name of the function() and passing it a parameter value results in the commands being carried out.



> source("foo.R")

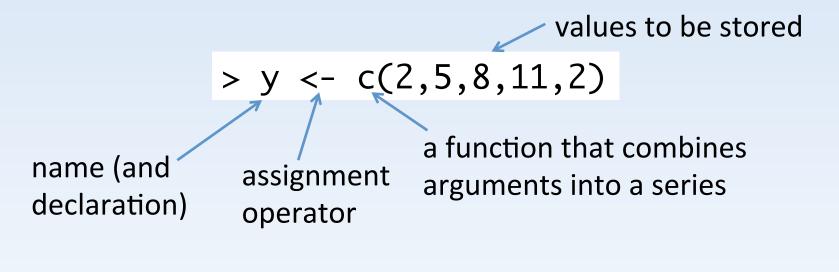
> foo2 <- edit(foo)</pre>

> X.2 <- edit(X)

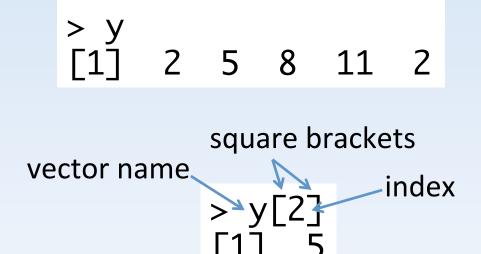
Functions written elsewhere and saved as text files can be loaded into a workspace using the source() command.

The edit() command will open any object – a function, a dataframe, a matrix, etc. – and allow you to change it (including prepackaged functions). The new version will be saved to the assigned name.

### **Vector** = a series of values



> y [1] 2 5 8 11 2 How does one access individual values? By using *index numbers*.



The critical difference between an index number and the actual value is that an index number refers to a particular slot in a vector (or other object), whereas the value is what is found in that slot.

Individual values can be reassigned:

Subsets of the vector can be drawn by referring to multiple indices:

":" is an operator that generates a sequence of integers *from:to* with a step size of 1.

<u>Matrix</u> = a table of values values to number number be stored of rows of columns

name (and assignment function that declaration) operator creates matrices

How does one access individual values? Again, by using *index numbers*.

Entire rows or columns can be referenced by leaving the index blank.

Note that each of these is a vector.



### Subsets can also be referenced and values can be changed:



**Loops** are used repeat a series of commands.

*i* is a variable that starts with the first value in the number sequence. Each command after "{" is carried out in succession. Every time the full succession is done (i.e., it hits "}"), *i* goes to the next value in the number sequence and each of the commands is repeated. Commands can change depending on the value of *i*. The loop will continue to repeat until *i* reaches the last value in the number sequence.



Nested loops are "loops within loops" – they provide a means of working with objects that have multiple indices.

```
> X = matrix(0,nrow=12,ncol=9)
> for (i in 1:12) {
+    for (j in 1:9) {
+      X[i,j] <- i*j
+    }
+ }</pre>
```

Indenting the body of a loop is regarded as good programming practice because it is a good way to keep track of the structure of the program.

What happens: first, i equals 1; j equals 1 and X[1,1] is assigned 1\*1; next j equals 2 and X[1,2] is assigned 1\*2; j equals 3 and X[1,3] is assigned 1\*3...until j equals 9. Now the first sweep through the i-loop is done and i becomes 2, but the j-loop starts again (the previous sweep is done and forgotten), so j equals 1 and X[2,1] is assigned 2\*1; j equals 2 and X[2,2] is assigned 2\*2...etc. Each time the j-loop is completed, the i-loop steps one value further and the whole set of commands (including the j-loop) within the i-loop is repeated.

#### Comparison Operators

- > a==b Equal
- > a!=b Not equal
- > a>b Greater than
- > a<b Less than
- > a>=b Greater than *or* equal
- > a<=b Less than or equal

# These operators result in a value of *TRUE* or *FALSE*.

Note that in typical use, an equal sign '=' can mean either assignment of a value OR a logical statement that is either true or false. These two roles have different operators: '<-' and '==,' respectively. In R, '=' is equivalent to assignment, but it is regarded as poor form (except when setting arguments in a function call).

### Conditional statements are used to compare values.

The *if-else* framework carries out one series of commands if a condition is TRUE and another if the condition is not TRUE. The () define the condition and the {} define the commands.

### Logical Operators (Boolean Algebra)

These operators compare the truth value of multiple comparisons and result in a value of TRUE or FALSE.

In R, logical values (i.e., comparisons resulting  $\lceil 1 \rceil = 3 + 3*1$  in TRUE or FALSE) are assigned numerical values: TRUE = 1 and FALSE = 0, allowing them  $\lceil 1 \rceil \mid 0 \mid \#3*0 \mid$  them to be manipulated as values.

Comparisons can be applied to vectors of values.

Vectors of logical values can be used to subset vectors, etc.