R programming for beginners

Ni Shuai

Computational Genome Biology
German Cancer Research Center (DKFZ)

November, 2016

Lists

List is a flexible data type which may contain any other class as each item, including vectors, matrices and even other lists. So you can have a list where the first element is a character vector, the second is a data frame, etc. To create a list, use list(Element1, Element2, ...):

```
a=matrix(1:10, nrow=2)
b='MyFirstList'
c=as.vector(a)
d=list('Item1'=a, 'Item2'=b,
       'Item3'=c)              #Put all defined objects into list d
                               #Items in the list can have names
e=list('Item1'=a, 'Item2'=b,
       'Item3'=c, 'Item4'=d) # e now contains another list d
```

Useful functions when manipulating a list:

unlist()	produce a vector with all the atomic components in the list
is.list()	Check if the argument is a list
as.list()	attempts to coerce its argument to a list

Lists

There are multiple ways to get access to items in an List:

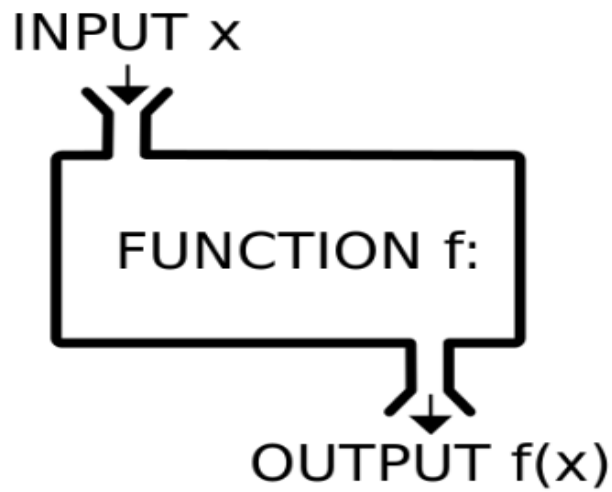
```
d[1] # It returns a 1-length list with the first item in d

## $Item1
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    3    5    7    9
## [2,]    2    4    6    8   10

d[[2]] # It directly returns the data type of the first item in d

## [1] "MyFirstList"

d$Item3 # Name of the item, similary to what does in data.frames

##  [1]  1  2  3  4  5  6  7  8  9 10
```

Because elements in lists can be every different, it is only possible to index one element in a list at a time

Functions

Functions in Mathematics *A function is a relation between a set of inputs and a set of permissible outputs with the property that each input is related to exactly one output*

A function in R is a piece of code written to carry out a specified task, most of the functions in R take argument(s) as the input and return something or do some tasks when they run



We know how to use a function and have encountered many built-in functions like sum() or matrix(), but we can also write our own functions

Create a function

A function definition takes the form:

```
name.of.function <- function(arguments) {
    statements
    return(something)
}
```

A function needs Usually, a function have a name, one or more arguments (the input), and a function body, which is everything in between the curly braces {}, where it does some computation. It may also return something as the function output:

```
MyFirstFunction=function(a,b){ #'MyFirstFunction' is the function name
                               #It takes two argument a and b
    x=a*b                      #Use a and b to calculate x
    return(x)                  #x is returned as the function output
}
MyFirstFunction(10,100)

## [1] 1000
```

Local vs global environment

The objects defined in the function are only stored in a 'local' environment and will not appear in the global environment, and will be erased after the function call

```
MyFun=function(a,b){
    y=a+b   #y is only valid when running the function
}
```

```
MyFun(2,3);  print(y)

## Error in print(y):  object 'y' not found

y=100;  MyFun(2,3)
print(y)

## [1] 100
```

Create a function

- A function can take no arguments or multiple arguments, with or without default
- It is optional but advisable to put the output object inside a return function
- If we have a function which performs multiple tasks and therefore has multiple results for output, we can put all the results in a list
- A function can call other functions in its body

```
MySecondFunction=function(a, b){ #Define function "MySecondFunction"
    x=MyFirstFunction(a, b)      #Call "MyFirstFunction" we just defined
    x1=x+(a+b)
    x2=x+(a-b)
    y=list(x1, x2)               #Put x1 and x2 into a list
    y                            #Just output y without using return
}
```

Function with optional arguments

If a function takes two arguments but only one is give, it will raise an error:

```
MyFirstFunction(4)

## Error in MyFirstFunction(4):  argument "b" is missing, with no
default
```

However if we give the second argument a default valule, the default value will be used when it's not specified.

```
MyFirstFunction=function(a,b=10){ #'MyFirstFunction' is the function name
                                  #It takes two argument a and b
    x=a*b                         #Use a and b to calculate x
    return(x)                     #x is returned as the function output
}
MyFirstFunction(4)

## [1] 40
```

Handling unexpected arguments

If the arguments passed to the function is missing not in the expected format, it will cause a error, but we can handle some exceptions and give a more informative feedback to the user by using stop() and warnings().

```
GreetMe=function(name){
    if (missing(name))                  # If the argument is given?
        stop('Please give me a name!')# No need the brackets
    if (class(name)!='character')       # If the argument in a right shape?
        warning("Sorry, I almost didn't recognize you!")
    print(paste0('Greetings, ', name, '!'))
}
GreetMe()

## Error in GreetMe():  Please give me a name!

GreetMe(108)

## Warning in GreetMe(108):  Sorry, I almost didn't recognize you!

## [1] "Greetings, 108!"

GreetMe('Ni Shuai')

## [1] "Greetings, Ni Shuai!"
```

Calling function from aother script

For matainability, it is always wise to abstract your code into many small functions. It helps expecially when one wants to reuse the code. source() function can read and parse the code from another file or a URL directly.

Exercises:

- Try to source `https://raw.githubusercontent.com/nishuai/bash/master/Greets.R` into R
- Save 'MyFirstFunction' into a separate called 'MyFirstFunction.R'
- Write a function called 'MySource' to read 'MyFirstFunction.R' into the local environment

Exercises

- Create a function that will return the sum of 2, 3, or 4 integers
- Create a function exist10() which returns TRUE if 10 is inside the input vector
- Modify the function exist10() so that it gives a warning when the input is a matrix
- Create a function that prints the name of the column and the type of data it is (e.g. Columnname1 is Numeric).
- Create a function that given a vector will print by screen the mean and the standard deviation, it will optionally also print the median.
- Create a function that outputs a list of all divisors (other than 1 and itself) of a given integer.