ECON457 lab01 R Basic

Jon

September 13, 2016

Some basic information related to R lab01.

Data type

R has five basic or "atomic" classes of objects:

character

numeric (real numbers)

integer

complex

logical (True/False)

```
x <- c(0.5, 0.6)       ## numeric
x <- c(TRUE, FALSE)    ## logical
x <- c(T, F)           ## logical
x <- c("a", "b", "c")  ## character
x <- c(1+0i, 2+4i)     ## complex
x <- 9:29              ## integer
```

Data Structure

Vector

The most basic type of R object is a vector. Empty vectors can be created with the vector() function. There is really only one rule about vectors in R, which is that A vector can only contain objects of the same class.

You can also use the vector() function to initialize vectors.

```
x <- vector("numeric", length = 10)
x
```

```
##  [1] 0 0 0 0 0 0 0 0 0 0
```

A vector can only contain objects of the same class.

Matrix

```
m <- matrix(nrow = 2, ncol = 3)
m
```

```
##      [,1] [,2] [,3]
## [1,]   NA   NA   NA
## [2,]   NA   NA   NA
```

Matrices can be created by column-binding or row-binding with the cbind() and rbind() functions.

```
x <- 1:3
y <- 10:12
cbind(x, y)
```

```
##      x  y
## [1,] 1 10
## [2,] 2 11
## [3,] 3 12
```

```
rbind(x, y)
```

```
##   [,1] [,2] [,3]
## x    1    2    3
## y   10   11   12
```

Data Frames

Data frames are used to store tabular data in R. They are an important type of object in R and are used in a variety of statistical modeling applications. Hadley Wickham's package dplyr has an optimized set of functions designed to work efficiently with data frames.

```
x <- data.frame(foo = 1:4, bar = c(T, T, F, F))
x
```

```
##   foo   bar
## 1   1  TRUE
## 2   2  TRUE
## 3   3 FALSE
## 4   4 FALSE
```

```
nrow(x)
```

```
## [1] 4
```

```
ncol(x)
```

```
## [1] 2
```

```
names(x)
```

```
## [1] "foo" "bar"
```

Object	Set column names	Set row names
data frame	names()	row.names()
matrix	colnames()	rownames()

Indexing

Indexing is used to specify the elements of an array. Indexing also allows you to get out certain bits of information from an array. To index into an array, type the name of the array, followed by the index of the element you want in brackets. Note that in R, indices start at 1.

For a multidimensional array, index by [row,column]

To index an entire row or column, use a colon.

Below we index into the named y to get out the element in the second row, third column, 6.

```
y <- c( 1,2,3, 4,5,6)
y <- matrix(y, nrow = 2, ncol = 3, byrow = T)
y[2,3]
```

```
## [1] 6
```

Below, we index the entire first row of the array named y.

```
y[1,]
```

```
## [1] 1 2 3
```

Below, we index the entire second column of the array named y.

```
y[,2]
```

```
## [1] 2 5
```

Load Data from Csv File

R works with many data formats.

csv file is the most convenient.

Methods on Objects

A method is a procedure associated with an object. Below is a list of common methods used on arrays.

```
length(y)
```

```
## [1] 6
```

```r
dim(y)
```

```
## [1] 2 3
```

```r
str(y)
```

```
##  num [1:2, 1:3] 1 4 2 5 3 6
```

```r
attributes(y)
```

```
## $dim
## [1] 2 3
```

```r
typeof(y)
```

```
## [1] "double"
```

```r
head(y, 1)
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
```

```r
tail(y,1)
```

```
##      [,1] [,2] [,3]
## [2,]    4    5    6
```

Mathematical Operations

Below is a list of common mathematical operations that you can perform on numerical types.

+,- ,/ , %, log, sin

x + y performs addition x - y performs subtraction x * y performs multiplication x / y performs division x ^ y raises x to the yth power x % y gives remainder when you divide x by y, same as rem(x,y)

x = y assigns the variable named "x" to the value y x == y evaluates to a Boolean, true if x equals y, false otherwise x != y evaluates to a Boolean, true if x does not equal y, false otherwise x > y evaluates to a Boolean, true if x is greater than y, false otherwise x < y evaluates to a Boolean, true if x is less than y, false otherwise x <= y evaluates to a Boolean, true if x is less than or equal to y, false otherwise x >= y evaluates to a Boolean, true if x is greater than or equal to y, false otherwise

x += y adds y to x, then reassigns x to the new value x -= y subtracts y from x, then reassigns x to the new value

Control Structure

Control Flow

Condition if/elseif/else

Conditional evaluation allows portions of code to be evaluated or not evaluated depending on the value of a Boolean expression. You do not need all if/elseif/else statements. You can have conditional evaluations with just an if, or just an if/else.

The general structure of conditional evaluation is as follows.

After assigning values to x and y and running the code we obtain the following output.

```
x =1
y=2
if (x<y){
  print(x)
}else{
  print(y)
}
```

```
## [1] 1
```

For Loops

A for loop allows you to specify the number of iterations for the repeated execution of a code block. They are great when you know how many iterations you want to run.

The general form of a for loop is shown below. The example shows a for loop that calculates the sum of the integers 1 through 10 and prints the final result.

Note that to obtain a range of integers, we use the colon : symbol.

```
sum = 0
for(num in 1:10){
  sum = sum + num
}
print(sum)
```

```
## [1] 55
```

A range of integers, we use the colon : symbol.

```
1:10
```

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

A series of video of R tutorial.

Each tutorial usually only is 2 minutes long.

http://www.twotorials.com/