GOVT PhD Math Camp Coding Lab Day 3

Theodore Landsman

Georgetown University

August 18 2021

Commands and Functions

- We use a command/function to perform some tasks on an object/objects

Commands and Functions

- We use a command/function to perform some tasks on an object/objects
- Argument: the definitions, directions, or objects that are passed to a command/function

Commands and Functions

- We use a command/function to perform some tasks on an object/objects
- Argument: the definitions, directions, or objects that are passed to a command/function
- When we specify multiple arguments,
  - separate the arguments by commas
  - it is desirable to specify them along with their names unless they are obvious
  - The code looks like `funcname(arg1 = input1, arg2 = input2)`

Commands and Functions Example

```
log(num.2)
```

```
## [1] 1.3862944 1.7917595 1.6094379 0.6931472 1.0986123
```

```
sqrt(num.2)
```

```
## [1] 2.000000 2.449490 2.236068 1.414214 1.732051
```

```
length(num.1)
```

```
## [1] 9
```

```
sum(num.1)
```

```
## [1] 13
```

Commands and Functions Example Continued

```
sort(num.1)
```

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

```
sort(num.1, decreasing = TRUE)
```

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

R Help Documentation

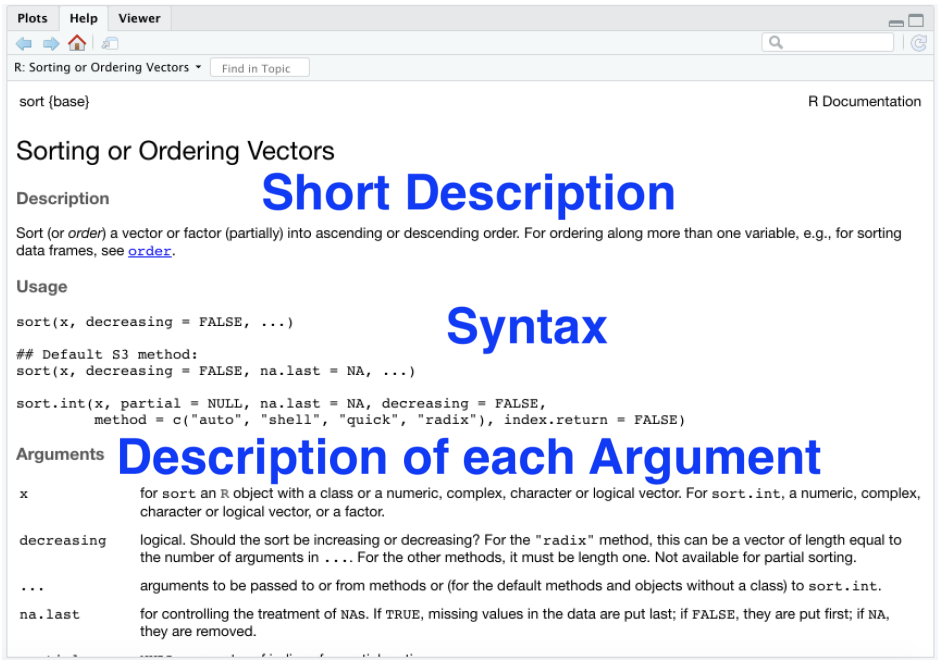
- We can access help files for functions by typing either `?funcname` or `help("funcname")` into the console.

R Help Documentation

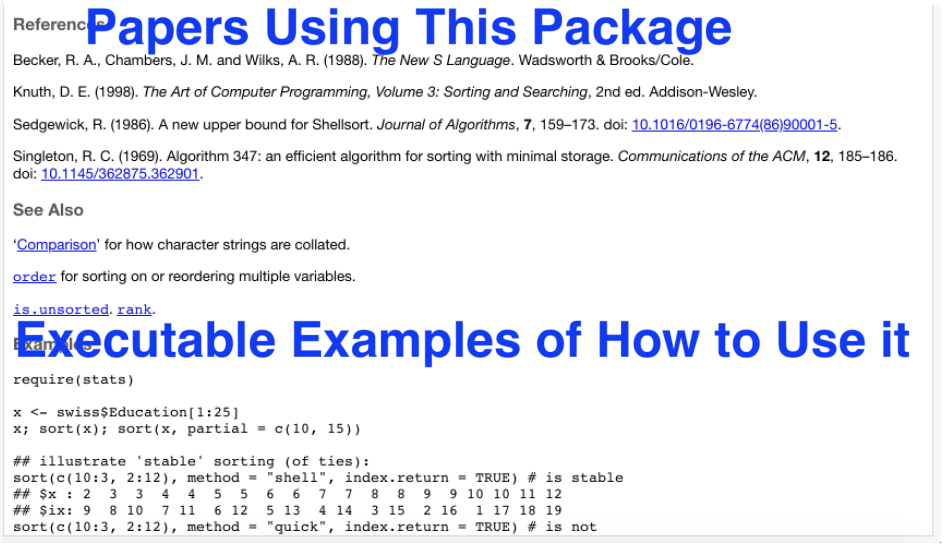
- We can access help files for functions by typing either `?funcname` or `help("funcname")` into the console.
- R help files are included for all base R functions *and* all package functions, coders are required to provide help documentation as part of uploading their packages to CRAN.

R Help Documentation Continued

- Output of `help("sort")`



(a) Top of help file



(b) Bottom of help file

- Don't understand what the help files are saying? Google search, ask others (including me!)

Function Exercises

- Calculate 10! and $\sqrt{16}$ *without* functions.
  Then do so using the functions `fact()` and `sqrt()`
- For the num.1 object we created earlier,
  - Calculate the product of all the numbers.
  - Count the number of elements larger than 0.
- Execute all five examples of `sum` under `help("sum")`

Creating Vectors and Matrices

- As we covered yesterday, in R we create vectors using c() command.

Creating Vectors and Matrices

- As we covered yesterday, in R we create vectors using c() command.
- We can create matrices with matrix() command

Usage

```
matrix(data, nrow, ncol, byrow)
```

'

Creating Vectors and Matrices

- As we covered yesterday, in R we create vectors using c() command.
- We can create matrices with matrix() command

Usage

```
matrix(data, nrow, ncol, byrow)
```

- Where:
  - `data:` vector of matrix element
  - `nrow, ncol:` number of rows/columns
  - `byrow:` if TRUE, the matrix is filled by rows; if FALSE, it is filled with columns

Creating Vectors and Matrices: Example

```r
# Creating matrices
A <- matrix(data = c(1, 4, 3, 5), nrow = 2, byrow = TRUE)
B <- matrix(data = c(1, 4, 3, 5), nrow = 2, byrow = FALSE)
C <- matrix(data = c(9, 7, 6, 2, 1, 3), nrow = 2,
            byrow = TRUE)
D <- matrix(data = c(2, 4, 5, 7, 1, 2), nrow = 3,
            byrow = TRUE)
# Print
A
```

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

```r
B
```

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

Creating Vectors and Matrices: Example 2

```
C
```

```
##      [,1] [,2] [,3]
## [1,]    9    7    6
## [2,]    2    1    3
```

```
D
```

```
##      [,1] [,2]
## [1,]    2    4
## [2,]    5    7
## [3,]    1    2
```

Vector Operations

```r
# Creating vectors
a <- c(1, 4, 5, 3, 7)
b <- c(3, 2, 4, 7, 1)
c <- c(8, -2, -4)
# Vector operations
a + b # vector addition
```

```
## [1]  4  6  9 10  8
```

```r
3 * a # scalar product
```

```
## [1]  3 12 15  9 21
```

```r
a + 3 # !?
```

```
## [1]  4  7  8  6 10
```

Vector Operations Continued

```
a %*% b # dot/inner product
```

```
##      [,1]
## [1,]   59
```

```
a * b # different from above!
```

```
## [1]  3  8 20 21  7
```

- Quick exercise: based on last class's matrix algebra lecture, show how to calculate the dot product of a and b.

Matrix Operations

```
A - B # matrix addition/subtraction
```

```
##      [,1] [,2]
## [1,]    0    1
## [2,]   -1    0
```

```
3 * C # scalar product
```

```
##      [,1] [,2] [,3]
## [1,]   27   21   18
## [2,]    6    3    9
```

```
A + 2
```

```
##      [,1] [,2]
## [1,]    3    6
## [2,]    5    7
```

Matrix Operations Continued

```
A %*% B # matrix product
```

```
##      [,1] [,2]
## [1,]   17   23
## [2,]   23   34
```

```
A * B # different from above!
```

```
##      [,1] [,2]
## [1,]    1   12
## [2,]   12   25
```

Matrix Operations 3

```
t(C) %*% B # t() to transpose matrices
```

```
##      [,1] [,2]
## [1,]   17   37
## [2,]   11   26
## [3,]   18   33
```

```
C %*% c # vectors are treated as the k by 1 matrices
```

```
##      [,1]
## [1,]   34
## [2,]    2
```

```
t(c) %*% D
```

```
##      [,1] [,2]
## [1,]    2   10
```

Determinants and Inverse

```r
det(A) # determinant
```

```
## [1] -7
```

```r
solve(B) # inverse
```

```
##            [,1]       [,2]
## [1,] -0.7142857  0.4285714
## [2,]  0.5714286 -0.1428571
```

- Quick exercise: What happens when we multiply `B` by `solve(b)`?

Solving Systems of Linear Equations

- We can also use the `solve()` command to find solutions to equations.
- Example: Let's solve the following system of equations with R.

$$x + 3y = 7$$
$$2x + 5y = 10$$

```
coefs <- matrix(c(1, 3, 2, 5), nrow = 2, byrow = TRUE)
rhs <- c(7, 10)
solve(coefs, rhs)
```

```
## [1] -5  4
```

Vector/Matrix Operations: Summary

Command	Meaning
`+`	Summation
`-`	Subtraction
`*`	Element-wise product (Hadamard product)
`%*%`	Matrix/Vector product
`length()`	(For vectors) Vector length
`dim()`	(For matrices) Matrix dimension
`t()`	Transpose
`det()`	Determinant
`solve()`	Inverse
`diag(A)`	($\boldsymbol{A}$ is a square matrix) Extract diagonal elements
`diag(k)`	(k is a scalar) Create a $k \times k$ identity matrix
`eigen()`	Compute eigenvalues and eigenvectors

Accessing Vector/Matrix Elements

- For vectors: `vectorname[i]` extracts the ith element of the vector
  - We can put in a vector within [ ] (e.g 1:3) to extract multiple elements
  - If we specify negative numbers within [ ], R deletes corresponding elements
- For matrices: `matrixname[i, j]` extracts the element in ith row and jth column
  - `matrixname[i,]` extracts all the elements in ith row as a vector.
  - `matrixname[, j]` extracts all the elements in jth column as a vector.

Accessing Vector/Matrix Elements: Example

```
b[3]
```

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

```
a[c(2, 4)]
```

```
## [1] 4 3
```

```
b[c(-1, -5)]
```

```
## [1] 2 4 7
```

```
C[2, 1]
```

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

```
D[-2,]
```

```
##      [,1] [,2]
## [1,]    2    4
## [2,]    1    2
```

Accessing Vector/Matrix Elements: Example Continued

```
B[1, 1] <- 9
B
```

```
##      [,1] [,2]
## [1,]    9    3
## [2,]    4    5
```

```
C[, 2] <- c(8, 3)
C
```

```
##      [,1] [,2] [,3]
## [1,]    9    8    6
## [2,]    2    3    3
```

Exercises!

For the following matrix

$$\boldsymbol{A} = \begin{pmatrix} 7 & -3 & 0 \\ -2 & 6 & 1 \\ 0 & -5 & 6 \end{pmatrix},$$

1. Find the determinant
2. Calculate the inverse
3. Replace the third row to (4, 2, 1) and recompute the determinant
4. Delete the first row and third column and find its inverse matrix