R cRash couRse

Chaochih Liu and Paul Hoffman

July 15th, 2016

Follow along at: http://z.umn.edu/rcrashcourse

basic aRithmetic

Addition: +

```
> 2 + 3
[1] 5
```

Subtraction: -

```
> 2 - 3
[1] -1
```

Multiplication: *

```
> 2 * 3
[1] 6
```

advanced basic aRithmetic

Integer Division: %/%

```
> 2 %/% 3
[1] 0
```

Remainders: %%

```
> 2 %% 3
[1] 2
```

Powers: ^ or **

```
> 2 ^ 3
[1] 8
> 2 ** 3
```

help

To pull up a manual page for R commands type:

```
#   Question mark followed by command
#
> ?setwd

#   Search within entire manual for this word
> ??set
```

variables and assignment

To store a value for future use:

```
#   Assign to variables
> x <- 5
> x
[1] 5

> myVariable <- data.frame(Values = c(1, 2, 3))
> myVariable
  Values
1      1
2      2
3      3
```

vectoRs

A vector is a collection of values of the same class.

```
#   Create a vector by combining values
> x <- c(1, 2, 3, 4, 5)
```

R quiRks

R intentionally recycles the values in shorter vectors:

```
> c(1, 2) + c(3, 3, 3, 3)
[1] 4 5 4 5

> c(1, 2) + c(3, 3, 3)
[1] 4 5 4
Warning message:
In c(1, 2) + c(3, 3, 3) :
  longer object length is not a multiple of shorter object length
```

R quiRks

Vectors in R are 1-indexed.

This means index 1 corresponds to the first element in a list.

Vectors must contain values of the same type.

subsetting

Subsetting in R means extracting more than one element simultaneously from a vector using indexing.

```
> myData <- data.frame(sample = c(1, 2, 3, 4, 5), height = c(3, 6, 4, 7, 2))
> myData
  sample height
1      1      3
2      2      6
3      3      4
4      4      7
5      5      2

#   Extract row one
> myData[1, ]
  sample height
1      1      3

#   Extract column 2
```

R quiRks

Special Value	Class	Description	Function to Test
NA	logical	Value to represent missing data	is.na()
NULL	NULL	Represents not having a value	is.null()
Inf/-Inf	Numeric	Positive and negative infinite values	is.finite(), is.infinite()
NaN	Numeric	“Not a Number”	is.nan()

R quiRks

R requires all values in a vector to have the same type.

Values are silently coerced so they have the same type.

Example:

```
> c(4.3, TRUE, 2.1, FALSE)
[1] 4.3 1.0 2.1 0.0
```

Reading in data

First, we'll set our working directory:

```
getwd() # view path to current working directory

setwd(dir = "~/bds-files/chapter-08-r") # path to directory
```

Reading in data

To read in file in table format and create data frame:

```
myDataset <- read.csv(file = "Dataset_S1.txt") # assign it to a variable name

bedFile <- read.delim(file = "noheader.bed", # assign it to a variable name
                      header = FALSE, # do not include headers
                      col.names = c("chrom", "start", "end")) # add column names
```

For more read.csv() and read.delim() arguments, reference Table 8-4.

R functions

Format of function:

```
myfunction <- function(argument1, argument2, ...) {
    commands
    return(object)
}
```

Example of function taken from random_sampler.R script:

```
#    Random sampler
randomSampler <- function(sampleList, numberOfDraws) {
    randomDraws <- sort(sample(x = as.vector(sampleList[, 1]), # sort samples from lowest to highest
number
                               size = numberOfDraws, # Number of samples we want to draw
                               replace = FALSE))
    return(randomDraws)
}
```

R objects and classes

Objects

- Representation of the data you have
- Example: myNumber <- 5

Class

- Limits the values of what your object can be
- Define how to interact with your object
- Example:

```
> class(myNumber)
[1] "numeric"
```

R functions

- Objects in function are local to the function
- Objects returned become available globally
- Objects returned can be any data type

R *enviRonment*

Environment

- Bag containing names that point to objects stored in memory

Global Environment (Big Bag)

- Variables are visible from within all functions

Local Environment (Sub Bag)

- Variables are only visible within a function
  - Unless returned with a return() function