CS 133 - Introduction to Computational and Data Science

Instructor: Renzhi Cao
Computer Science Department
Pacific Lutheran University
Spring 2017

Announcement

- *Read book to page 44.*
- *Final project*
- *Today we are going to learn how to get data In and Out of R and R control structure.*

Reading big data

Using the readr Package

The readr package is recently developed by Hadley Wickham to deal with

reading in large flat files quickly.

read.table()   => read_table()

read.csv()   => read_csv()

```
install.packages("readr")

>library(readr)
>read_csv(mtcars_path)
>write_csv(mtcars, mtcars_path)
```

dput() and dump()

One way to pass data around is by deparsing the R object with dput() and reading it back in (parsing it) using dget().

```
> ## Create a data frame
> y <- data.frame(a = 1, b = "a")
> ## Print 'dput' output to console
> dput(y)
```

dput() and dump()

The output of dput() can also be saved directly to a file.

```
> ## Create a data frame
> y<-data.frame(a=1,b="a")

> ## Print 'dput' output to console
> dput(y)


> ## Send 'dput' output to a file
> dput(y, file = "y.R")
> ## Read in 'dput' output from a file

> new.y <- dget("y.R")
> new.y
```

dput() and dump()

Multiple objects can be deparsed at once using the dump function and read back in using source.

```
> x <- "foo"
> y <- data.frame(a = 1L, b = “a")
```

We can dump() R objects to a file by passing a character vector of their names.

```
> dump(c("x", "y"), file = "data.R")
> rm(x, y)                    # this is going to remove the x and y object
```

The inverse of dump() is source().

```
> source("data.R")
> str(y)
> x
```

Interfaces to the Outside World

Data are read in using connection interfaces. Connections can be made to files (most common) or to other more exotic things.

- file, opens a connection to a file
- gzfile, opens a connection to a file compressed with gzip
- bzfile, opens a connection to a file compressed with bzip2
- url, opens a connection to a webpage

Connections

Connections to text files can be created with the file() function.

```
> str(file)
```

The open argument allows for the following options:

- “r” open file in read only mode
- “w” open a file for writing (and initializing a new file)
- “a” open a file for appending
- “rb”, “wb”, “ab” reading, writing, or appending in binary mode (Windows)

Connections

In practice, we often don’t need to deal with the connection interface directly as many functions for reading and writing data just deal with it in the background.

```
> ## Create a connection to 'foo.txt'
> con <- file("foo.txt")
>
> ## Open connection to 'foo.txt' in read-only mode

> open(con, "r")

>
> ## Read from the connection

> data <- read.csv(con)
>
> ## Close the connection
> close(con)
```

which is the same as

```
> data <- read.csv("foo.txt")
```

Reading lines from a text file

Text files can be read line by line using the readLines() function.

```
> ## Open connection to gz-compressed text file

> con <- gzfile("words.gz")

> x <- readLines(con, 10)
```

The above example used the gzfile() function which is used to create a connection to files compressed using the gzip algorithm.

There is a complementary function writeLines() that takes a character vector and writes each element of the vector one line at a time to a text file.

Reading lines from a URL

The readLines() function can be useful for reading in lines of webpages.

```
> ## Open a URL connection for reading
> con <- url("http://www.jhsph.edu", "r")
>
> ## Read the web page
> x <- readLines(con)
>
> ## Print out the first few lines
> head(x)
```

Exercises

1. Open a URL connection to this link: http://www.plu.edu

2. Read the webpage to data.

3. Check the first 50 rows of these data, what do you find?

4. Create a vector v with elements: “Weather”, “100”, and data frame d with two elements: element a is a integer 10, element b is vector v.

5. Use dump function to store v and d at file “dumpdata.R”. Remove object v and d

6. Use source function to read object v and d back to memory and print it out.

Relational Operators

Boolean Operators

Operator	Meaning
>	is greater than
<	is less than
>=	is greater than or equal to
<=	is less than or equal to
==	is equal to
!=	is not equal to

Boolean Expressions

Expression	Meaning
`x > y`	Is x greater than y?
`x < y`	Is x less than y?
`x >= y`	Is x greater than or equal to y?
`x <= y`	Is x less than or equal to y?
`x == y`	Is x equal to y?
`x != y`	Is x not equal to y?

Exercises

1. a <- 10, what should be a>20?

2. a <- 10, b <- 20

What should be a>b?

What should be a!=b?

How about a==b?

3. a <- 10, b <- a * 2 - 10

What should be a>b?

What should be a!=b?

How about a>=b?

Control structure of R

Control structures in R allow you to control the flow of execution of a series of R expressions.

Put some “logic” into your R code, rather than just always executing the same R code every time.

Control structures allow you to respond to inputs or to features of the data and execute different R expressions accordingly.

Control structure of R

Commonly used control structures are

- if and else: testing a condition and acting on it
- for: execute a loop a fixed number of times
- while: execute a loop while a condition is true
- break: break the execution of a loop
- next: skip an iteration of a loop

If-else

```
if(<condition>) {
## do something
}
## Continue with rest of code
```

```
if(<condition>) {
## do something
} else {
    ## do something else
}
```

If-else

You can have a series of tests by following the initial if with any number of **else ifs.**

```
if(<condition1>) {
## do something
} else if(<condition2>) {
## do something different
} else {
## do something different
}
```

If-else

Examples:

```
x <- 10
if(x > 3) {
  y <- 10
}
else {
   y <- 0
}

y <- if(x > 3) {

     10

} else {

    0

}
```

What's the value of x and y?

Exercises

- Create a integer i with value 10 and integer j with value 20. Write a if statement to calculate the difference of i and j only if i is less than j.
- Create a vector v with three elements: 10, 20, 30. Write if-else statement to print the summation of these three elements in v only when the summation is larger than 40, otherwise print 0.
- Write a if-else statement to print "Go swimming" and set variable cost to 5 if variable temperature is at least 80 degrees, otherwise, print "Do programming" and set cost to 0.

Exercises

- In R, we can use readline function to get user's input. Try:

  n<- readline(prompt = "Enter an numeric number:")

- Use readline function to get user's GPA input as gpa, and write if-else statement to print the message "Dean's list" if gpa is greater or equal to 3.8, print "Possible dean's list" is gpa is less than 3.8 but larger than 3.5, otherwise print "Does not qualify for dean's list".