CWRU DSCI351-451: Rmd and For Loop Basics

Prof.:Roger French, TA:JiQi Liu

04 September, 2018

Contents

2.2.2.1 Some Simple Rmd Items

2.2.2.1.1 Rmd for Exploratory Data Analysis

- EDA is a foundation of Data Science
- Identify sources of data for your problem
- Need to acquire, assemble, clean, and explore your data
- An environment for Exploratory Data Analysis (EDA)

2.2.2.1.2 R markdown is tool for Open Science

- Reproducible data analysis
- Incorporating Data, Code, Presentation and Reporting
- Good coding practices are essential
- Comment your code, describe your data frames
- Make your data analyses a presentation and report.

2.2.2.1.3 Rcode in Rmd, delineated by three backticks{r}

```r
options("digits" = 5)
options("digits.secs" = 3)
```

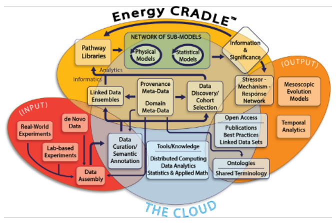

Figure 1: Caption

2.2.2.1.4 Code in Rmd is delinearted

- three backticks for code blocks
- one tick for inline code

2.2.2.1.5 YAML settings and commenting

- This is the top block of the Rmd file
  - set off by three dashes —

- In the YAML Header, you can comment lines with ' #### '
  - But in the body, ' #### ' is a second level header!

2.2.2.1.6 Commenting in the Rmd body

- To comment in the Rmd body, use html comment form
  - < ! - - Comment - - >

  - but with no spaces between the characters
    -
    - i.e.' "' ends a comment block

–>

2.2.2.1.7 Inserting Figs, 2 ways.

- Essential to use relative file paths
- Essential to use Posix compatible paths

2.2.2.2 Filenames and Paths

2.2.2.2.1 Filenames

- No Spaces
- No characters other than letters, numbers, - and underscore
- Better not to capitalize
- Or if you must, use CamelBacking

2.2.2.2.2 Paths

- Windows is not Posix compatible
  - is not understood, must be typed \
    - but should be /,
  - / always works on Linux, Mac, Windows
- Relative Paths
  - . i.e. dot, is the current folder
  - .. i.e. dot dot is the folder one above your current area
- setwd (setting working directory) is bad to rely on.

2.2.2.3 R Coding Training: For Loops

2.2.2.3.1 For loop basics

- For loops are an important part for almost any coding problem
- They work by applying an iterator that changes every time
  - i is the standard iterator over a code block
  - but the iterator can be named anything)

2.2.2.3.2 Common example, using a counter

```
# print out numbers upto a given num
num <- 5

for (i in 1:num) {
  print(i)
}
```

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

2.2.2.3.3 vectors or columns can also be iterated over

This can improve clarity in many cases

- if a counter is not needed

```
letters <- c('a','b','c')

for (i in letters) {
  print(i)
}
```

```
## [1] "a"
## [1] "b"
## [1] "c"
```

2.2.2.3.4 Collecting for loop outputs

Lets say we want to calculate the square root of every value in a vector

- using a for loop,
- what is the problem with the code below?

```r
num <- c(4, 8, 15, 16, 23, 42)

for (i in num) {
  result <- sqrt(i)
}

result
```

```
## [1] 6.4807
```

We only get 1 number when we wanted 6

- Every time the loop iterates it overwrites the 'result' variable,
  - leaving us with only the last value
- There are multiple was to save out results,
  - depending on what analysis you're running
- I've found this to be one of the most straight forward ways

```r
num <- c(4, 8, 15, 16, 23, 42)
# define a NULL variable to write into
all_results <- NULL

for (i in num) {
  # calculate the square root of i
  result <- sqrt(i)
  # concatinate the ith result onto the total result vector
  # rbind() is also useful if the results have multiple variables (columns)
  all_results <- c(all_results, result)
}

all_results
```

```
## [1] 2.0000 2.8284 3.8730 4.0000 4.7958 6.4807
```

- This gives us the answer we wanted

2.2.2.3.5 For loop drawbacks

For loops are highly fundamental

- But they have some problems
- As seen in the example above,
  - organizing results can be messy,
  - especially with complicated results
- They only run one process at a time,
  - making them slow and
  - unable to run parallel process
- Later on we will look at ways to avoid for loops
  - to improve code clarity and increase speed,
  - as well as allow for parallel processing

Dplyr, Pipes and the Tidyverse

- Help avoid the slow performance of For loops
- And streamline/clarify the code

2.2.2.4 Links

http://www.r-project.org

http://rmarkdown.rstudio.com/

<!– # Keep a complete change log history at bottom of file. # Complete Change Log History # v0.00.00 - 1405-07 - Nick Wheeler made the blank script ##########