Statistical Computing with R
Masters in Data Science 503 (S4)
First Batch, SMS, TU, 2021

Shital Bhandary

Associate Professor

Statistics/Bio-statistics, Demography and Public Health Informatics

Patan Academy of Health Sciences, Lalitpur, Nepal

Faculty, Data Analysis and Decision Modeling, MBA, Pokhara University, Nepal

Faculty, FAIMER Fellowship in Health Professions Education, India/USA.

Review Preview

- Histogram
  - Correct use of central tendency and dispersion parameter
- Boxplot
  - Summary of skewed variable/s; comparison by groups
- Scatterplot
  - Linear or non-linear correlation and regression analysis
- Vector in R
  - Vector operations in R
- Function in R
  - Built-in function
  - User defined function

Vectorized operations in R:

- A key difference between R and many other languages is a topic known as vectorization
- We can write “total” function in R using “loop” but it will be much faster to use built-in “sum” function of R to get the “total”
- “Sum” is already coded in C to work with vector of numbers in R and the loop written in C is hidden from us in R
- Many other built-in functions also works in this way in R!

Vector operation in R:

- a <- 1:10 #seq(1,10) is equivalent to this
- b <- 1:10 #c(1,2,3,4,5,6,7,8,9,10) is same as this

- How to add a and b vectors (variables)?

- Do we need to write functions for this?

- What will happen if we type: a + b in R console?
- [1] 2 4 6 8 10 12 14 16 18 20 #Matrix addition?

Quick Think!

- What will happen when you type these two codes in R/R studio and run:

- y <- seq(1, 10, length.out = 5)

- (y <- seq(1, 10, length.out = 5))

Vector operation in R: Continued …

- a <- 1:10 #Vector length = 10
- b <- 5 #Vector length = 1

- What will happen if we type: a * b in R console?

- [1] 5 10 15 20 25 30 35 40 45 50

- Why? # R does not have scalars!

Exercise: Do this in R or R Studio!

- Define:
  - a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] #Check class of ‘a’
  - b = [5] #Check class of ‘b’
  - c = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11] #Check class of ‘c’
- Define:
  - ac = cbind(a,c) #Check class of ‘ac’
  - acb = ac * b #Check class of ‘acb’
- Get:
  - d = arithmetic mean by cases (row means) #Hint: Use “apply” function
  - acbd = add new variable d in the acb matrix #Check class of ‘acbd’ object
  - Summarize all the variables of ‘acbd’ object and interpret the results carefully!

Vector operation in R: Continued …

- a <- 1:10 #Vector length = 10
- b <- 1:5 #Vector length = 5

- What will happen if we type: a + b in R console?

- [1] 2 4 6 8 10 7 9 11 13 15 #Matrix addition!

- Why? #1+1, 2+2, 3+3, 4+4, 5+5, 6+1, 7+2, 8+3, 9+4, 10+5

- **#Vector recycling is done when we add vectors with unequal lengths, take a note of this!**

Vector operation in R: Continued …

- a <- 1:10 #Vector length = 10
- b <- 1:7 #Vector length = 7

- What will happen if we type: a + b in R console?

- What does the “error” message tells you?

- What is the key message?

Function in R: Continued …

best_practice <- c(“Let”, “the”, “computer”, “do”, “the”, “work”)

```
print_words <- function(sentence) {
        print(sentence[1])
        print(sentence[2])
        print(sentence[3])
        print(sentence[4])
        print(sentence[5])
        print(sentence[6])
}
```

What is wrong with this approach?

- print_words(best_practice) # [1] ”Let” [1] “the” [1] “computer” [1] “do” [1] “the” [1] “work”
- print_words(best_practice[-6]) # [1] ”Let” [1] “the” [1] “computer” [1] “do” [1] “the” [1] “NA”
- best_practice[-6] # [1] ”Let” “the” “computer” “do” “the”

Can we improve it in R?
We can use functions with “for” loop in R!

```
print_words <- function(sentence) {
    for (word in sentence) {
    print(word)
    }
}
print_words(best_practice)
```

[1] “Let” [1] “the” [1] “computer” [1] “do” [1] “the” [1] “work”

```
print_words(best_practice[-6])
```

[1] “Let” [1] “the” [1] “computure” [1] “do” [1] “the”

“for” loop

R:

```
for (variable in collection) {
    do things with variable
}
```

“for” and “while” loops can be very slow in R!

What to do?

https://swcarpentry.github.io/r-novice-inflammation/03-loops-R/

Loops in R will not be slow if we:

- Don’t use a loop when a vectorized alternative exists
- Don’t grow objects (via c, cbind, etc) during the loop – R has to create new object and copy across the information just to add new element or row/column
- Allocate an object to hold the result and fill it during the loop
- **Can we do even better in R? Alternative to “loop” in R??**

While working with data.frame in R:

- It is better to use:
  - apply
  - lapply
  - sapply
  - vapply

We will discuss this in detail while doing breakdown analysis session in R later!

- functions instead of “for loop” to run the script much faster in R!

- Same applies to the “while loop” too!

R script:

- It is a collection of codes and can be saved with an .R extension
- It contains #comments
- It can be used to create “markdown” file using knitr package
- R markdown files are ready to deploy files as an online or offline reports using bookdown package e.g. https://bookdown.org/

Use of R Markdown (*.rmd): Sharing!

What is R script? How to use it??

wp_websrapping_in_R.R

Source on Save | Run | Source

```r
library(rvest)
library(dplyr)

wiki_link = "https://en.wikipedia.org/wiki/COVID-19_pandemic_in_Nepal"
wiki_page = read_html(wiki_link)

covid_table = wiki_page %>% html_nodes("table") %>% .[14] %>%
  html_table() %>% .[[1]]

names(covid_table) = paste(names(covid_table), covid_table[1, ], sep = "_")
covid_table = covid_table[-1, ]

summary(covid_table)

colnames(covid_table)
names(covid_table)[names(covid_table) == "Date_Date"] = "Date"
names(covid_table)[names(covid_table) == "Confirmed cases_Total"] = "Confirmed_Cases_T
names(covid_table)[names(covid_table) == "Confirmed cases_New"] = "Confirmed_Cases_New
names(covid_table)[names(covid_table) == "Confirmed cases_Active"] = "Confirmed_Cases_
names(covid_table)[names(covid_table) == "RT-PCR tests_Total"] = "PCR_Total"
names(covid_table)[names(covid_table) == "RT-PCR tests_New"] = "PCR_New"
names(covid_table)[names(covid_table) == "TPR_TPR"] = "TPR"
names(covid_table)[names(covid_table) == "RR_RR"] = "RR"
names(covid_table)[names(covid_table) == "CFR_CFR"] = "CFR"
names(covid_table)[names(covid_table) == "Ref._Ref."] = "Ref"
colnames(covid_table)
```

1:1 (Top Level) R Script

R script: Why %>% was used?

- This is called “pipe” operator and described in chapter 14 of the course book provided to you
- Pipes are a powerful tool for clearly expressing a sequence of multiple operations
- The pipe, %>%, comes from **magrittr** package
- The pipe helps you to write code in a way that is easier to read and understand!

Example: Understand the run in R/R Studio:

```
rnorm(100) %>%
matrix(ncol = 2) %>%
plot() %>%
str()
```

Condition: if and else

```
if (condition) {
        #code executed when condition is TRUE
} else {
        #code executed when condition is FALSE
}
```

Can you think of an example?

What will be the output?

```
if (y < 20) {
   x <- "Too low"
} else {
  x <- "Too high"
}
```

Multiple conditions:

```
if (this) {
        # do that
} else if (that) {
        # do something else
} else {
        # remaining
}
```

Multiple Conditions: If, else if, else

```
if (temp <= 0) {
"freezing"
} else if (temp <= 10) {
"cold"
} else if (temp <= 20) {
"cool"
} else if (temp <= 30) {
"warm"
} else {
"hot"
}
```

Question/queries?

Thank you!

@shitalbhandary