R Tutorial for Stat 331

Tae Hyun Kim

Introduction

R is a popular programming language among data analysts because of its intuitive syntax and its open-source nature: it is free, and anyone can contribute. In this tutorial, I will go over the following points.

1. Basic syntax and data structure
2. Data manipulation in R
3. R markdown documents

Basic Syntax and Data Structure

```
# You can write comment by using pound symbol.
# You can get help by using a question mark and command name
?mean
?var
?data.frame
```

Assign value to a variable using either '<-' or '='

```
x <- 5
x = 5
```

Display the value of the variable by 'print'

```
print(x)
```

```
[1] 5
```

The type of variable can be known by 'class'

```
class(x)
```

```
[1] "numeric"
```

Manipulation of scalars are fairly intuitive

```
y = 3
print(x+y)
```

```
[1] 8
```

```
print(x-y)
```

```
[1] 2
```

```
print(x/y)
```

```
[1] 1.666667
```

```
print(x*y)
```

```
[1] 15
```

Vectors

There are other types of data structure. Let’s create a vector. ‘rep’ command repeats the first value times the second value. rep(5, 3) prints three fives. ‘numeric’ returns a numeric vector of zeros with given length. ‘c’ command concatenates the values you assign.

```
y = rep(1, 5)
print(y)
```

```
[1] 1 1 1 1 1
```

```
z = numeric(5)
print(z)
```

```
[1] 0 0 0 0 0
```

```
w = c(1, 6, -4, 2, 3)
print(w)
```

```
[1]  1  6 -4  2  3
```

The type of a vector returns the type of one element in a vector. Naturally, a vector can only contain elements of the same type.

```
print(class(y))
```

```
[1] "numeric"
```

```
print(class(z))
```

```
[1] "numeric"
```

```
print(class(w));
```

```
[1] "numeric"
```

You can do element-wise arithmetics of the vectors.

```
print(y + w)
```

```
[1]  2  7 -3  3  4
```

```
print(w * z)
```

```
[1] 0 0 0 0 0
```

I can also create a sequence

```
y = -3:5
print(y)
```

```
[1] -3 -2 -1  0  1  2  3  4  5
```

Or, if you want the sequence to have different increment, you can use ‘seq’

```
y = seq(from = -15, to = 20, by = 5); print(y)
```

```
[1] -15 -10  -5   0   5  10  15  20
```

I can access a specific element of a vector using brackets.

```
print(w[3])
```

```
[1] -4
```

```
w[3] = 1000
print(w)
```

```
[1]    1    6 1000    2    3
```

Or you can access multiple elements of a vector

```
print(w[2:3])
```

```
[1]    6 1000
```

You can compute length, mean, variance, and sd of a vector with intuitivce function calls

```
length(w)
```

```
[1] 5
```

```
mean(w)
```

```
[1] 202.4
```

```
var(w)
```

```
[1] 198805.3
```

```
sd(w)
```

```
[1] 445.8759
```

Note that ‘var’ and ‘sd’ commands are for samples, not for population. It’s important to distinguish the two especially in this course. Take a look at the example below. I’m comparing two variance formulas.

$$var_{population}(x) = \sum_{i=1}^{n} \frac{(x_i - \bar{x})^2}{n}$$

$$var_{sample}(x) = \sum_{i=1}^{n} \frac{(x_i - \bar{x})^2}{n-1}$$

```
x = c(1,2,3,4,5)
m = mean(x)
sumsquares = (1-m)^2 + (2-m)^2 + (3-m)^2 + (4-m)^2 + (5-m)^2
sumsquares / length(x) #population variance formula
```

```
[1] 2
```

```
sumsquares / (length(x)-1) #sample variance formula
```

```
[1] 2.5
```

```
var(x) #R default variance computation
```

```
[1] 2.5
```

Matrices

Now, let’s create a matrix. I introduce two ways This repeats a certain value.

```
A = matrix(4, 3, 2); print(A)
```

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

Or you can be more specific. The two lines below return the exact same matrices.

```
B1 = matrix(c(1, 3, 2, 6, 4, -5), nrow = 3)
B2 = matrix(c(1, 3, 2, 6, 4, -5), ncol = 2)
print(B1); print(B2)
```

```
     [,1] [,2]
[1,]    1    6
[2,]    3    4
[3,]    2   -5
```

```
     [,1] [,2]
[1,]    1    6
[2,]    3    4
[3,]    2   -5
```

Let's call this matrix B, instead of B1 and B2

```
B = B1
```

You can remove objects in your environment through 'rm'. This helps you manage your storage space when you work with large data sets. You won't be needing this in the scope of this class.

```
rm(B1); rm(B2)
```

You can access specific elements in matrices, too. The command below calls for the elements in the rows 1 and 3, and column 2.

```
B[c(1, 3), 2]
```

```
[1]  6 -5
```

Logicals

Logical values take either TRUE or FALSE. Let's check if x has value of 5. Note that we use two equal signs

```
x==5
```

```
[1] FALSE FALSE FALSE FALSE  TRUE
```

You can also assign this value to y

```
y = (x==5)
print(y)
```

```
[1] FALSE FALSE FALSE FALSE  TRUE
```

To check if something is NOT equal to something, use !

```
y = (x != 5)
print(y)
```

```
[1]  TRUE  TRUE  TRUE  TRUE FALSE
```

Lists

I mentioned earlier that vectors can only take elements of the same type. List, on the other hand, can handle objects of different types.

```
ListA = list('dog', c(2, 4, 5), -4, x==6)
class(ListA)
```

```
[1] "list"
```

```
print(ListA)
```

```
[[1]]
[1] "dog"

[[2]]
[1] 2 4 5

[[3]]
[1] -4

[[4]]
[1] FALSE FALSE FALSE FALSE FALSE
```

```
class(ListA[[1]]); class(ListA[[2]]); class(ListA[[3]]); class(ListA[[4]])
```

```
[1] "character"
```

```
[1] "numeric"
```

```
[1] "numeric"
```

```
[1] "logical"
```

Data Manipulation in R

You need to set your working directory first to access the data set stored in your computer.

You can check where you are by ‘getwd’

```
getwd()
```

```
[1] "/Users/tae/Desktop/S331"
```

You can check what files are currently in your directory like this

```
list.files()
```

```
[1] "0_Steps_in_a_Sample_Survey.pdf" "Course_Info_Autumn_2018.doc"
[3] "RMarkdown_Tutorial_tae.pdf"     "roster.numbers"
[5] "Rtutorial_tae.Rmd"              "schooldistrict.Rdata"
[7] "schools.txt"
```

If you are not in the correct working directory, you can set it up : example is my personal directory

```
setwd('~/Desktop/S331')
```

Now, I will read the data set in txt format and assign the name ‘school’ This data set is under ‘Schools Data’ on the course website Make sure you the data set is in your current directory

```r
school = read.table('schools.txt', header = TRUE)
```

Alternatively, you can specify the directory of the data

```r
school = read.table('~/Desktop/S331/schools.txt', header = TRUE)
```

'header' means that the first row of the txt file is the column names. You can also read csv files through read.csv. Check R manual to learn more about read.table and read.csv.

```r
class(school)
```

```
[1] "data.frame"
```

You can read data in RData format by 'load'. This data set is under 'School District Data' on the course website

```r
load('schooldistrict.RData')
class(schooldistrict) # Note that the object name is same as the file name in this case, but it could b
```

```
[1] "data.frame"
```

Here are some basic commands useful in initial data analysis

```r
head(school)      #lets you see the first few rows of the data
```

```
    LEAID SCHNO LOCALE01 FTE01 LEVEL01 REDLCH01 ASIAN01 MEMBER01
1 3900009  1281        7   0.9       4       NA      NA       NA
2 3900011  1282        6  15.2       4       NA      NA       NA
3 3900011  3689        7   0.0       4        9       0       50
4 3900013  1283        7   8.0       4       NA      NA       NA
5 3900013  1299        6   1.0       4       NA       0       38
6 3900014  1260        7   6.0       4       NA      NA       NA
```

```r
tail(school)      #lets you see the last few rows of the data
```

```
        LEAID SCHNO LOCALE01 FTE01 LEVEL01 REDLCH01 ASIAN01 MEMBER01
32985 7200030  2078       NA    29       1       18       0      282
32986 7200030  2079       NA    36       1       34       0      673
32987 7200030  2082       NA    39       2       27       0      516
32988 7200030  2083       NA    24       2       30       0      448
32989 7200030  2084       NA    32       1       23       0      325
32990 7200030  2085       NA    39       3       35       0      870
```

```r
nrow(school)      #number of rows
```

```
[1] 32990
```

```r
ncol(school)      #number of columns
```

```
[1] 8
```

```r
dim(school)       #dimension of the data
```

```
[1] 32990     8
```

```r
colnames(school)  #column names
```

```
[1] "LEAID"    "SCHNO"    "LOCALE01" "FTE01"    "LEVEL01"  "REDLCH01"
[7] "ASIAN01"  "MEMBER01"
```

Just to make things simpler, I will take a subset of the data and work with that.

```
school2 = school[1:100, 1:3] #subset the data, rows from 1 to 100, columns from 1 to 3
summary(school2)             #lets you see the summary of each column
```

```
     LEAID             SCHNO         LOCALE01
 Min.   :3900009   Min.   :   2   Min.   :1.00
 1st Qu.:3900038   1st Qu.:2834   1st Qu.:1.00
 Median :3900072   Median :3323   Median :2.00
 Mean   :3900237   Mean   :2980   Mean   :2.13
 3rd Qu.:3900099   3rd Qu.:3821   3rd Qu.:2.00
 Max.   :3904348   Max.   :4167   Max.   :7.00
```

```
str(school2)                 #lets you see 'string' version of the data
```

```
'data.frame':   100 obs. of  3 variables:
 $ LEAID   : int  3900009 3900011 3900011 3900013 3900013 3900014 3900015 3900017 3900018 3900019 ...
 $ SCHNO   : int  1281 1282 3689 1283 1299 1260 1269 1509 1513 1514 ...
 $ LOCALE01: int  7 6 7 7 6 7 7 2 2 1 ...
```

As well as the column index, You can also access each column by the column name using dollar sign.

```
school2$SCHNO
```

```
  [1] 1281 1282 3689 1283 1299 1260 1269 1509 1513 1514 1515 1516 1520 1525
 [15] 1529 1543 1562 1572 1573 1578 2807 2833 2834 2837 2838 2844 2915 2939
 [29] 2975 2979 2984 2997 2998 3001 3011 3015 3027 3054 3067 3084 3090 3118
 [43] 3131 3159 3162 3248 3722 3315 3331 3346 3360 3361 3759 3368 3382 3387
 [57] 3407 3420 3428 3441 3445 3762 3786 3446 3447 3449 3463 3602 3607 3635
 [71] 4142 3913 4036 4167 4014 3820 3860 4160 4054 4157 4154 4067 4153 4121
 [85] 3876 3824 3949 3911 4150 4140 4155 4148 3854 4066 4131 3957    2   12
 [99]   13   14
```

```
mean(school2$SCHNO)
```

```
[1] 2979.68
```

I can also subset in more specific ways. I will take only rows with value '1' in LOCALE01 column. First, check how the LOCALE01 column looks like

```
school2$LOCALE01
```

```
  [1] 7 6 7 7 6 7 7 2 2 1 1 1 3 1 3 1 1 2 2 2 1 1 2 2 1 2 2 1 1 1 1 2 2 2 2
 [36] 1 1 2 1 1 2 1 1 1 1 1 2 3 3 2 1 2 1 2 2 1 1 1 1 1 1 2 2 1 1 4 1 3 3 3
 [71] 2 2 2 2 2 2 2 2 2 1 1 1 3 2 1 1 2 1 1 2 3 7 1 4 1 6 2 2 2 2
```

Then, take subset, and find out how many rows have value 1

```
newsubset = school2[school2$LOCALE01==1, ]
nrow(newsubset)
```

```
[1] 42
```