Basic R

Md Rasheduzzaman

2025-04-22

Data types, variables, vectors, data frame, functions

Table of contents

1 Using R as a Calculator

```
5+3
```

```
[1] 8
```

```
3+2
```

```
[1] 5
```

```
3-2
```

```
[1] 1
```

```
3*2
```

```
[1] 6
```

```
3/2 #normal division
```

```
[1] 1.5
```

```
7 %/% 2 #integer division, only the quotient
```

```
[1] 3
```

```
5 %% 3 #modulus division, the remainder
```

```
[1] 2
```

```
(10-5)*(2+4) #use of parentheses
```

```
[1] 30
```

```
10-5*2+4 #Noticed BODMAS?
```

```
[1] 4
```

```
(10-5)*(2+4) #Noticed BODMAS
```

```
[1] 30
```

```
7/(1+3); 7/1+3 #multi-line codes, separated with semi-colon
```

```
[1] 1.75
```

```
[1] 10
```

```
1+2; log(1); 1/10 #more multi-line codes
```

```
[1] 3
```

```
[1] 0
```

```
[1] 0.1
```

2 Variables and Assignments

```
a <- 5 #assign value 5 to a
b = 10
a
```

```
[1] 5
```

```
b
```

```
[1] 10
```

```
a <- a + 10
b = b + 15
```

```
a
```

```
[1] 15
```

```
a^2 #a squared
```

```
[1] 225
```

```
a**2 #a squared again, in a different way.
```

```
[1] 225
```

```
a^3 #a qubed
```

```
[1] 3375
```

> **Note**
>
> `<-` and `=` are used to assign values. It is not mathematical equality. `b <- b + 15` might make better sense than `b = b + 15`.

3 Integer and Modulus division again

```
7/3
```

```
[1] 2.333333
```

```
7%/%3
```

```
[1] 2
```

```
7%%3
```

```
[1] 1
```

4 Rounding

```
x <- 9/4
floor(x)
```

```
[1] 2
```

```
ceiling(x)
```

```
[1] 3
```

```
round(x)
```

```
[1] 2
```

```
round(x, 2) #round till 2 decimal points
```

```
[1] 2.25
```

5 Logical Operations

```
a = 5
b = 7
c = 10
d = 3
a == b #is a equal to b? Ans: No/FALSE
```

```
[1] FALSE
```

```
a != b #is a not equal to b? Ans: Yes/TRUE
```

```
[1] TRUE
```

```
a > b #is a greater than b? Ans: FALSE
```

```
[1] FALSE
```

```
a < b #is a less than b? Ans: TRUE
```

```
[1] TRUE
```

```
a >= b #is a greater than or equal to b? Ans: FALSE
```

```
[1] FALSE
```

```
a <= b #is a less than or equal to b? Ans: TRUE
```

```
[1] TRUE
```

```
a < b | d > b #is a less than b OR d greater than b?
```

```
[1] TRUE
```

```
#It's answer will be TRUE OR FALSE --> So, TRUE
a < b & c > d #is a less than b AND a greater than b? It's answer will be TRUE AND TRUE --
```

```
[1] TRUE
```

```
a < b & d > c #is a less than b AND a greater than b? It's answer will be TRUE AND FALSE -
```

```
[1] FALSE
```

6 Help and Documentation

```
?log
example(log)
```

```
log> log(exp(3))
[1] 3
```

```
log> log10(1e7) # = 7
[1] 7

log> x <- 10^-(1+2*1:9)

log> cbind(deparse.level=2, # to get nice column names
log+       x, log(1+x), log1p(x), exp(x)-1, expm1(x))
          x   log(1 + x)     log1p(x)   exp(x) - 1     expm1(x)
 [1,] 1e-03 9.995003e-04 9.995003e-04 1.000500e-03 1.000500e-03
 [2,] 1e-05 9.999950e-06 9.999950e-06 1.000005e-05 1.000005e-05
 [3,] 1e-07 1.000000e-07 1.000000e-07 1.000000e-07 1.000000e-07
 [4,] 1e-09 1.000000e-09 1.000000e-09 1.000000e-09 1.000000e-09
 [5,] 1e-11 1.000000e-11 1.000000e-11 1.000000e-11 1.000000e-11
 [6,] 1e-13 9.992007e-14 1.000000e-13 9.992007e-14 1.000000e-13
 [7,] 1e-15 1.110223e-15 1.000000e-15 1.110223e-15 1.000000e-15
 [8,] 1e-17 0.000000e+00 1.000000e-17 0.000000e+00 1.000000e-17
 [9,] 1e-19 0.000000e+00 1.000000e-19 0.000000e+00 1.000000e-19
```

```
?log()
```

7 Working with Vectors

```
x <- c(1, 2, 3, 4, 5) #c means concatenate
z <- 1:5 #consecutively, from 1 through 5. A short-hand
y <- c(3, 6, 9, 12, 15, 20)
length(x)
```

```
[1] 5
```

```
mode(x)
```

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

```
is(x)
```

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

```r
x[1] #first entry in vector y
```

```
[1] 1
```

```r
x[2:5] #2nd to 5th entries in vector y
```

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

```r
DNA <- c("A", "T", "G", "C") #character vector. Notice the quotation marks.
dec <- c(10.0, 20.5, 30, 60, 80.9, 90, 100.7, 50, 40, 45, 48, 56, 55) #vector of floats. A
dec[c(1:3, 7:length(dec))] #1st to 3rd and then 7th till the end of vector `dec`. Output a
```

```
 [1]  10.0  20.5  30.0 100.7  50.0  40.0  45.0  48.0  56.0  55.0
```

8 Vector Operations

```r
x <- 1:10
y <- 2:11
#x and y are of same length
x + y
```

```
 [1]  3  5  7  9 11 13 15 17 19 21
```

```r
y / x
```

```
 [1] 2.000000 1.500000 1.333333 1.250000 1.200000 1.166667 1.142857 1.125000
 [9] 1.111111 1.100000
```

```r
log2(x)
```

```
 [1] 0.000000 1.000000 1.584963 2.000000 2.321928 2.584963 2.807355 3.000000
 [9] 3.169925 3.321928
```

```r
round(log2(x), 1) #log2 of all the values of `x`, 1 digit after decimal to round.
```

```
 [1] 0.0 1.0 1.6 2.0 2.3 2.6 2.8 3.0 3.2 3.3
```

```
round(log2(x), 3) #same logic
```

```
 [1] 0.000 1.000 1.585 2.000 2.322 2.585 2.807 3.000 3.170 3.322
```

Nested functions work inside out.

9 Data Frame Example

```
names <- c("Mina", "Raju", "Mithu", "Lali")
gender <- c("Female", "Male", "Female", "Female")
age <- c(15, 12, 2, 3)
is_human <- c(TRUE, TRUE, FALSE, FALSE)
cartoon <- data.frame(names, gender, age, is_human)
write.table(cartoon, "cartoon.csv", sep = ",", col.names = TRUE)
df <- read.table("cartoon.csv", header = TRUE, sep = ",")
dim(df) #`dim` means dimension. so, rows * columns
```

```
[1] 4 4
```

```
str(df) #structure of `df`
```

```
'data.frame':   4 obs. of  4 variables:
 $ names   : chr  "Mina" "Raju" "Mithu" "Lali"
 $ gender  : chr  "Female" "Male" "Female" "Female"
 $ age     : int  15 12 2 3
 $ is_human: logi  TRUE TRUE FALSE FALSE
```

We made the vectors first, and the used them to make the `cartton` data frame or table. We learned how to export the data frame using write.table function. Also, we learned to import or read back the table using `read.table` function. What are the `sep`, `col.names`, `header` arguments there? Why do we need them? Think. Try thinking of different properties of a data set.

10 Gene Expression Table

```
gene_expr <- data.frame(
  genes = c("TP53", "BRCA1", "MYC", "EGFR", "GAPDH", "CDC2"),
  sample1 = c(8.2, 6.1, 9.5, 7.0, 10.0, 12),
  Sample2 = c(5.9, 3.9, 7.2, 4.8, 7.9, 9),
  Sample3 = c(8.25, 6.15, 9.6, 7.1, 10.1, 11.9),
  pathways = c("Apoptosis", "DNA Repair", "Cell Cycle", "Signaling", "Housekeeping", "Cell
)
write.table(gene_expr, "gene_expr.csv", sep = ",", col.names = TRUE)
gene_set <- read.table("gene_expr.csv", header = TRUE, sep = ",")
```

Here, we directly used the vectors as different columns. Did you notice that? Also, the syntax is different here. We can't asign the vectors with the assignment operator (means we can't use `<-` sign. We have to use the `=` sign).

11 Homeworks

1. Compute the difference between this year (2025) and the year you started at the university and divide this by the difference between this year and the year you were born. Multiply this with 100 to get the percentage of your life you have spent at the university.

2. Make different kinds of variables and vectors with the data types we learned together.

3. What are the properties of a data frame?
   Hint: Open an excel/csv/txt file you have and try to "generalize".

4. Can you make logical questions on the 2 small data sets we used? Try. It will help you understanding the logical operations we tried on variables. Now we are going to apply them on vectors (columns) on the data sets. For example, in the `cartoon` data set, we can ask/try to subset the data set filtering for Females only. Or for both females age greater than 2 years.

5. Push the script with your solutions to one of your GitHub repo (and send me the repo link).