Basics of R Programming

Contents

1 R as an Advanced Calculator

- Arithmetic Operations in R

```
a = 3
b = 5
a+b
```

```
[1] 8
```

```
a*b
```

```
[1] 15
```

```
a/b
```

```
[1] 0.6
```

```
a^b
```

```
[1] 243
```

```
4^4
```

```
[1] 256
```

```
4^3 - 3^2
```

```
[1] 55
```

```
log(100, 10) #by default log gives natural log
```

```
[1] 2
```

- Logical Operations
  - & and | operators perform element-wise operations and provide the length of the longer operand.
  - && and || operators examine only the first element of the operands resulting in a single length logical vector.
  - The logical & operator returns true only if both the options are true, or else returns false.

```
5==5 # True
```

```
[1] TRUE
```

```
4 == 5 # False
```

```
[1] FALSE
```

```
a = (5==5)
a
```

```
[1] TRUE
```

```
F * 60 # False * 60 = 0
```

```
[1] 0
```

```
T * 50 # True * 50 = 50
```

```
[1] 50
```

```
5 > 2
```

```
[1] TRUE
```

```
4 != 5
```

```
[1] TRUE
```

```
5 >= 4
```

```
[1] TRUE
```

```
3 <= 4
```

```
[1] TRUE
```

```
TRUE & TRUE
```

```
[1] TRUE
```

```
T&T
```

```
[1] TRUE
```

```
T&FALSE
```

```
[1] FALSE
```

```
FALSE & FALSE
```

```
[1] FALSE
```

```
TRUE | FALSE
```

```
[1] TRUE
```

- Vectors

```
    # vector creation
    #c() seq()

    v = c(1,2,3,4)
    v
```

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

```
    head(v)
```

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

```
    tail(v)
```

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

```
    v1 = 5:13
    v1
```

```
[1]  5  6  7  8  9 10 11 12 13
```

```
    v = 3.8:11.4
    v
```

```
[1]  3.8  4.8  5.8  6.8  7.8  8.8  9.8 10.8
```

```
    seq(5,9, by = 0.4)
```

```
 [1] 5.0 5.4 5.8 6.2 6.6 7.0 7.4 7.8 8.2 8.6 9.0
```

```
    seq(0,1, length =11)
```

```
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

```
    #vector manipulation
    #Two vectors of the same length can be added

    v1 = c(3,8,4,5,0,11)
    v2 = c(4,5,2,5,6,5)
    v1+v2
```

```
[1]  7 13  6 10  6 16
```

```
    v1-v2
```

```
[1] -1  3  2  0 -6  6
```

```
    v1*v2 #element wise multiplication
```

```
[1] 12 40  8 25  0 55
```

```
    # vector element recycling
    v1 = c(3,4,5,2,2,4)
    v2 = c(4,11)

    v1 + v2 # v1 + c(4,11,4,11,4,11)
```

```
[1]  7 15  9 13  6 15
```

```r
    #vector element sorting
    v = c(3,4,5,2,5,1)
    sort(v)
```

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

```r
    sort(v, decreasing = T)
```

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

```r
    color = c("Red","Blue","White")
    sort(color)
```

```
[1] "Blue"  "Red"   "White"
```

```r
    #indexing
    v[3] # counts from 1
```

```
[1] 5
```

```r
    v[3:6]
```

```
[1] 5 2 5 1
```

```r
    v[c(1,3,5)]
```

```
[1] 3 5 5
```

```r
    v[-c(5,7)]
```

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

```r
fib <- c(0, 1, 1, 2, 3, 5, 8, 13, 21, 34)
cat("The first few Fibonacci numbers are:", fib, "...\n")
```

```
The first few Fibonacci numbers are: 0 1 1 2 3 5 8 13 21 34 ...
```

2 Data Types

R's basic data types are Numeric, Character, Integer, Factor, and Logical.

```
# as.integer()
# as.character()
# as.factor() # categorical

y = 5
class(y) # numeric
```

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

```
class(c("1","4")) #character
```

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

```
x = 4
y = 3
z = x>y
class(z) #logical
```

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

```
v2 = c("My","Name","is","David")
class(v2) #character
```

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

```
summary(v2)
```

```
   Length     Class      Mode
        4 character character
```

```
nchar(v2) # number of characters 2 4 2 5
```

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

```
#Factor Variables in R
v5 = c("M","F","F","M")
class(v5) #character
```

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

```
v5 = as.factor(v5)
class(v5) #
```

```
[1] "factor"
```

```
summary(v5)
```

```
F M
2 2
```

2.1 Computing Basic Statistics

```
x = c(0, 1, 1, 2, 3, 5, 8, 13, 21, 34)
mean(x)
```

```
[1] 8.8
```

```
median(x)
```

```
[1] 4
```

```
sd(x)
```

```
[1] 11.03328
```

```
var(x)
```

```
[1] 121.7333
```

```r
x = c(0, 1, 1, 2, 3, 5, 8, 13, 21, 34)
y = log(x + 1)
cor(x, y)
```

```
[1] 0.9068053
```

```r
cov(x, y)
```

```
[1] 11.49988
```

```r
# handling missing values...
x <- c(0, 1, 1, 2, 3, NA)
mean(x)
```

```
[1] NA
```

```r
sd(x)
```

```
[1] NA
```

```r
mean(x, na.rm = TRUE)
```

```
[1] 1.4
```

```r
sd(x, na.rm = TRUE)
```

```
[1] 1.140175
```

```r
data(cars)
head(cars)
```

```
  speed dist
1     4    2
2     4   10
3     7    4
4     7   22
```

```
5     8   16
6     9   10
```

```
var(cars)
```

```
          speed     dist
speed  27.95918 109.9469
dist  109.94694 664.0608
```

```
cor(cars)
```

```
          speed      dist
speed 1.0000000 0.8068949
dist  0.8068949 1.0000000
```

```
cov(cars)
```

```
          speed     dist
speed  27.95918 109.9469
dist  109.94694 664.0608
```

2.2 Creating Sequences

```
1:5
```

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

```
9:0
```

```
 [1] 9 8 7 6 5 4 3 2 1 0
```

```
seq(from = 1, to = 5, by = 2)
```

```
[1] 1 3 5
```

```
rep(1, times = 5)
```

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

```
seq(from = 0, to = 20, length.out = 5)
```

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

```
seq(from = 1.0, to = 2.0, length.out = 5)
```

```
[1] 1.00 1.25 1.50 1.75 2.00
```

2.3 Comparing Vectors

```
v <- c(3, pi, 4)
w <- c(pi, pi, pi)
v == w
```

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

```
v < w
```

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

```
v > w
```

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

```
v != w
```

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

```
v <- c(3, pi, 4)
v == pi # Compare a 3-element vector against one number
```

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

```r
v <- c(3, pi, 4)
any(v == pi) # Return TRUE if any element of v equals pi
```

```
[1] TRUE
```

```r
all(v == 0) # Return TRUE if all elements of v are zero
```

```
[1] FALSE
```

2.4 Vector Slicing (Indexing)

```r
a = c(0, 1, 1, 2, 3, 5, 8, 13, 21, 34)
a[1] # note that indexing starts at 1
```

```
[1] 0
```

```r
a[5]
```

```
[1] 3
```

```r
a[2:3] # also the last index is included, unlike Python
```

```
[1] 1 1
```

```r
a[-1]
```

```
[1]  1  1  2  3  5  8 13 21 34
```

```r
a[1:3]
```

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

```r
a[-(1:3)] # Invert sign of index to exclude instead of select
```

```
[1]  2  3  5  8 13 21 34
```

2.5 Vector Masking

```
a = c(0, 1, 1, 2, 3, 5, 8, 13, 21, 34)
a < 10
```

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

```
a[a<10] #select all numbers < 10
```

```
[1] 0 1 1 2 3 5 8
```

```
a[a %% 2 == 0] #select all even numbers.
```

```
[1]  0  2  8 34
```

```
v <- c(3, 6, 1, 9, 11, 16, 0, 3, 1, 45, 2, 8, 9, 6, -4)
v[ v > median(v)]
```

```
[1]  9 11 16 45  8  9
```

2.6 Vector Operations

```
v <- c(11, 12, 13, 14, 15)
w <- c(1, 2, 3, 4, 5)
v + w
```

```
[1] 12 14 16 18 20
```

```
v - w
```

```
[1] 10 10 10 10 10
```

```
v * w
```

```
[1] 11 24 39 56 75
```

```
v / w
```

```
[1] 11.000000  6.000000  4.333333  3.500000  3.000000
```

```
v^w
```

```
[1]     11    144   2197  38416 759375
```

```
sqrt(w)
```

```
[1] 1.000000 1.414214 1.732051 2.000000 2.236068
```

```
log(w)
```

```
[1] 0.0000000 0.6931472 1.0986123 1.3862944 1.6094379
```

2.7 Broadcasting

```
w
```

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

```
w+2
```

```
[1] 3 4 5 6 7
```

```
w*2
```

```
[1]  2  4  6  8 10
```

```
w/2
```

```
[1] 0.5 1.0 1.5 2.0 2.5
```

```
2^w
```

```
[1]  2  4  8 16 32
```

```
w^2
```

```
[1]  1  4  9 16 25
```

Example:

```
(w - mean(w)) / sd(w)
```

```
[1] -1.2649111 -0.6324555  0.0000000  0.6324555  1.2649111
```

2.8 Matrices

```
M = matrix(c(1, 2, 3, 4), 2, 2)
print(M)
```

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

3 User Defined Functions in R

```
x = c(3,4,4,5,6,9,20,3)
mean(x)
```

```
[1] 6.75
```

```
sum(x)/length(x)
```

```
[1] 6.75
```

In the absence of standard mean function, we could define our own function.

```
myMean = function(y){
  sum(y) / length(y)
}
myMean(x)
```

```
[1] 6.75
```

```
mySD = function(z){
  sqrt(sum(x-myMean(x))^2/ (length(x)-1))
}
```

4 Conditional Statements & Loops

```
# ifelse(condition, expr1, expr2)

gender = c("Male","Female","male","Female")

ifelse(gender == "Male",print("M"), print("F"))
```

```
[1] "M"
[1] "F"
```

```
[1] "M" "F" "F" "F"
```

```
length(gender)
```

```
[1] 4
```

```
x =3
if (x > 1) {
    1
  } else {
    "Less Than 1"
  }
```

```
[1] 1
```

For Loop

```
for (i in 1:length(gender)){
  ifelse(gender[i] == "Male",
  print("M"),
  print("F"))
}
```

```
[1] "M"
[1] "F"
[1] "F"
[1] "F"
```

5 Counting the Number of Combinations

```
# choose(n, k)
choose(5, 3)   # How many ways can we select 3 items from 5 items?
```

```
[1] 10
```

```
# generate all combinations of n items taken k at a time
items <- 2:5
k <- 2
combn(items, k)
```

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

```
combn(1:5, 3)
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    1    1    1    1    1    1    2    2    2     3
[2,]    2    2    2    3    3    4    3    3    4     4
[3,]    3    4    5    4    5    5    4    5    5     5
```

```
combn(c("T1", "T2", "T3", "T4", "T5"), 2)
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,] "T1" "T1" "T1" "T1" "T2" "T2" "T2" "T3" "T3" "T4"
[2,] "T2" "T3" "T4" "T5" "T3" "T4" "T5" "T4" "T5" "T5"
```

6 Generating Random Numbers

```
runif(1)
```

```
[1] 0.01695934
```

R can generate random variates from other distributions as well. For a given distribution, the name of the random number generator is "r" prefixed to the distribution's abbreviated name (e.g., `rnorm` for the normal distribution's random number generator). This example generates one random value from the standard normal distribution:

```
rnorm(1)
```

```
[1] 0.5300611
```

There are random number generators for all built-in distributions. Simply prefix the distribution name with "r" and you have the name of the corresponding random number generator. Here are some common ones:

```
runif(1, min = -3, max = 3)      # One uniform variate between -3 and +3
```

```
[1] -0.08476624
```

```
rnorm(1)                         # One standard normal variate
```

```
[1] 0.1828974
```

```
rnorm(1, mean = 100, sd = 15)    # One normal variate, mean 100 and SD 15
```

```
[1] 76.85778
```

```
rbinom(1, size = 10, prob = 0.5) # One binomial variate
```

```
[1] 5
```

```
rpois(1, lambda = 10)            # One Poisson variate
```

```
[1] 8
```

```
rexp(1, rate = 0.1)              # One exponential variate
```

```
[1] 3.57051
```

```
rgamma(1, shape = 2, rate = 0.1) # One gamma variate
```

```
[1] 10.48552
```

6.1 Generating Reproducible Random Numbers

Before running your R code, call the `set.seed` function to initialize the random number generator to a known state:

```
set.seed(42) # Or use any other positive integer...
runif(10)
```

```
 [1] 0.9148060 0.9370754 0.2861395 0.8304476 0.6417455 0.5190959 0.7365883
 [8] 0.1346666 0.6569923 0.7050648
```

6.2 Generating a Random Sample

The `sample` function will randomly select *n* items from a set:

```
# sample(set, n)
s = seq(1:10)
s
```

```
 [1]  1  2  3  4  5  6  7  8  9 10
```

```
sample(s,3)
```

```
[1] 8 7 4
```

```
medians <- numeric(1000)   # empty vector of 1000 numbers
x = 10
for (i in 1:1000) {
  medians[i] <- median(sample(x, replace = TRUE))
}
head(medians)
```

```
[1] 5.0 5.5 4.5 5.5 5.5 6.0
```

```
sample(c("H", "T"), 10, replace = TRUE)
```

```
 [1] "H" "T" "H" "T" "T" "H" "H" "H" "T" "H"
```

```
sample(c("H", "T"), 10, replace = TRUE, prob = c(0.2, 0.8))
```

```
 [1] "T" "T" "T" "T" "T" "T" "T" "H" "T" "H"
```

7 Data Cleaning Set the working directory.

```
data = read.csv('../data/Salary_Data.csv')
dim(data) #rows, cols
```

```
[1] 30  2
```

```
summary(data)
```

```
 YearsExperience      Salary
 Min.   : 1.100   Min.   : 37731
 1st Qu.: 3.200   1st Qu.: 56721
 Median : 4.700   Median : 65237
 Mean   : 5.313   Mean   : 76003
 3rd Qu.: 7.700   3rd Qu.:100545
 Max.   :10.500   Max.   :122391
```

```
head(data) #first 6
```

```
  YearsExperience Salary
1             1.1  39343
2             1.3  46205
3             1.5  37731
4             2.0  43525
5             2.2  39891
6             2.9  56642
```

```
head(data, 8) #first 8
```

```
  YearsExperience Salary
1             1.1  39343
2             1.3  46205
3             1.5  37731
```

```
4             2.0  43525
5             2.2  39891
6             2.9  56642
7             3.0  60150
8             3.2  54445
```

```
str(data)
```

```
'data.frame':   30 obs. of  2 variables:
 $ YearsExperience: num  1.1 1.3 1.5 2 2.2 2.9 3 3.2 3.2 3.7 ...
 $ Salary         : num  39343 46205 37731 43525 39891 ...
```

```
colnames(data)
```

```
[1] "YearsExperience" "Salary"
```

```
## change names of the column

# colnames(data)[1] = "Name"
# colnames(data)[2:4] = c("Title","ID","AgencyName")
```