R 语言基础：练习(三)

Table of Contents

1.列表练习

Note: Before proceeding, first read the help pages for the sum, length, strsplit, and setdiff functions.

Exercise 1

If: p <- c(2,7,8), q <- c("A", "B", "C") and x <- list(p, q), then what is the value of x[2]?

a. NULL
b. "A" "B" "C"
c. "7"

```
p <- c(2,7,8)
q <- c("A", "B", "C")
x <- list(p, q)
x[2]
## [[1]]
## [1] "A" "B" "C"
# (Answer: b)
```

Exercise 2

If: w <- c(2, 7, 8) v <- c("A", "B", "C") x <- list(w, v), then which R statement will replace "A" in x with "K".

a. x[[2]] <- "K"

b. x[[2]][1] <- "K"

c. x[[1]][2] <- "K"

```
w <- c(2, 7, 8)
v <- c("A", "B", "C")
x <- list(w, v)
x[[2]][1] <- "K"
x
## [[1]]
## [1] 2 7 8
##
## [[2]]
## [1] "K" "B" "C"
# (Answer: b)
```

Exercise 3

If a <- list ("x"=5, "y"=10, "z"=15), which R statement will give the sum of all elements in a?

a. sum(a)

b. sum(list(a))

c. sum(unlist(a))

```
a <- list ("x"=5, "y"=10, "z"=15)
sum(unlist(a))
## [1] 30
# (Answer: c)
```

Exercise 4

If `Newlist <- list(a=1:10, b="Good morning", c="Hi")`, write an R statement that will add 1 to each element of the first vector in `Newlist`.

```
Newlist <- list(a=1:10, b="Good morning", c="Hi")
Newlist$a <- Newlist$a + 1
Newlist
## $a
##  [1]  2  3  4  5  6  7  8  9 10 11
##
## $b
## [1] "Good morning"
##
## $c
## [1] "Hi"
```

Exercise 5

If `b <- list(a=1:10, c="Hello", d="AA")`, write an R expression that will give all elements, except the second of the first vector of b.

```
b <- list(a=1:10, c="Hello", d="AA")
b$a[-2]
## [1]  1  3  4  5  6  7  8  9 10
```

Exercise 6

Let `x <- list(a=5:10, c="Hello", d="AA")`, write an R statement to add a new item z = "NewItem" to the list x.

```
x <- list(a=5:10, c="Hello", d="AA")
x$z <-"New Item"
x
## $a
## [1]  5  6  7  8  9 10
##
## $c
## [1] "Hello"
##
## $d
## [1] "AA"
##
## $z
## [1] "New Item"
```

Exercise 7

Consider y <- list("a", "b", "c"), write an R statement that will assign new names "one", "two" and "three" to the elements of y.

```
y <- list("a", "b", "c")
names(y) <- c("one", "two", "three")
y
## $one
## [1] "a"
##
## $two
## [1] "b"
##
## $three
## [1] "c"
```

Exercise 8

If `x <- list(y=1:10, t="Hello", f="TT", r=5:20)`, write an R statement that will give the length of vector r of x.

```
x <- list(y=1:10, t="Hello", f="TT", r=5:20)
length(x$r)
```

Exercise 9

Let `string <- "Grand Opening"`, write an R statement to split this string into two and return the following output:

```
[[1]]
[1] "Grand"

[[2]]
[1] "Opening"
```

```
string <- "Grand Opening"
a <- strsplit(string," ")
list(a[[1]][1], a[[1]][2])
## [[1]]
## [1] "Grand"
##
## [[2]]
## [1] "Opening"
```

Exercise 10

Let: `y <- list("a", "b", "c")` and `q <- list("A", "B", "C", "a", "b", "c")`. Write an R statement that will return all elements of q that are not in y, with the following result:

```
[[1]]
[1] "A"

[[2]]
[1] "B"

[[3]]
[1] "C"
```

```
y <- list("a", "b", "c")
q <- list("A", "B", "C", "a", "b", "c")
setdiff(q, y)
## [[1]]
## [1] "A"
##
## [[2]]
## [1] "B"
##
## [[3]]
## [1] "C"
```

2. 条件执行练习

Exercise 1

Create an R script that returns the absolute value of an integer vector x of length one.

```
x <- -10
abs <- x
if (x < 0) {
    abs = -x
}
cat("The absolute value of ", x, " is ", abs , "\n" )
```

Exercise 2

Create an R script that calculates the square root of a given integer vector x of length one, if the value contained in x is negative it should return NA.

```
# Exercise 2
x <- 16
y <- ifelse(x >= 0, x, NA)
cat("The square root of",  x, "is", sqrt(y))
## The square root of 16 is 4
```

Exercise 3

Create an R script that returns the maximum value out of the elements of a numeric vector x of length 2.

```
x <- c(10, 1)
if(x[1] > x[2]) {
    cat("Max value is", x[1], "\n")
} else cat("Max value is", x[2], "\n")
## Max value is 10
```

Exercise 4

Create an R script that returns TRUE if the elements of a vector x, with length 3, are strictly increasing.

```
x <- c(10, 11, 12)
grow <- FALSE

ifelse ( ( (x[1] < x[3] & x[1] < x[2]) & x[2] < x[3]), grow <- TRUE, gr
ow)
## [1] TRUE
if (grow){
    cat ("Increasing strictly \n")
} else cat ("Not increasing strictly \n")
## Increasing strictly
```

Exercise 5

Create an R script that returns the max value of a vector x with length 3. Don't use the aid of an auxiliary variable.

```
x <- c(20, 10, 1)

if (x[1] > x[2] & x[1] > x[3] ) {
    cat (x[1], "\n" )
} else if (x[2] > x[3] ) {
    cat (x[2] , "\n" )
} else {
    cat (x[3] , "\n" )
}
## 20
```

Exercise 6

Create an R script that returns the amount of values that are larger than the mean of a vector. You are allowed to use mean().

```
x <- c(-100, 10, 20, 30, 50, 51, 52, 53, 54, 55)
counter <- 0
mean <- mean(x)

for (i in 1:length(x)){

    if(x[i] > mean){
        counter <- counter +1
    }
}

cat("The number of values that are bigger than the mean is", counter, "
\n")
## The number of values that are bigger than the mean is 7
```

Exercise 7

Create an R script that, given a numeric vector x with length 3, will print the elements by order from high to low.

```
x <- c(30, 120, 100)

if (x[1] > x[2]){
    fir <- x[1]
    sec <- x[2]
} else {
    fir <- x[2]
    sec <- x[1]
}
```

```
if ( x[3] > fir & x[3] > sec ) {
    thi <- sec
    sec <- fir
    fir <- x[3]
} else if ( x[3] < fir & x[3] < sec ) {
    thi <- x[3]
} else {
    thi <- sec
    sec <- x[3]
}
cat (fir, sec, thi, "\n")
## 120 100 30
```

3.函数练习

Note: For some exercises, the solution will be quite easy if you make clever use of some of R's built-in functions. For some exercises, you might want to create a vectorized solution (i.e., avoiding loops), and/or a (usually slower) non-vectorized solution. However, the exercises do not aim to practise vectorization and speed, but rather defining and calling functions.

Exercise 1

Create a function that will return the sum of 2 integers.

```
f.sum <- function (x, y) {
  r <- x + y
  r
}
f.sum(5, 10)
## [1] 15
```

Exercise 2

Create a function what will return TRUE if a given integer is inside a vector.

```
f.exists <- function (v, x) {
  exist <- FALSE
  i <- 1

  while (i <= length (v) & !exist) {

    if (v[i] == x) {
      exist <- TRUE
    }
  i <- 1 + i
  }
  exist
}
f.exists(c(1:10), 10)
```

```
## [1] TRUE
f.exists(c(9, 3, 1), 10)
## [1] FALSE
```

Exercise 3

Create a function that given a data frame will print by screen the name of the column and the class of data it contains (e.g. Variable1 is Numeric).

```
f.class <- function (df) {
  for (i in 1:ncol(df)) {
    cat(names(df)[i], "is", class(df[, i]), "\n")
  }
}
f.class(cars)
## speed is numeric
## dist is numeric
```

Exercise 4

Create the function unique, which given a vector will return a new vector with the elements of the first vector with duplicated elements removed.

```
f.uniq <- function (v) {
  s <- c()

  for(i in 1:length(v)) {
    if(sum(v[i] == s) == 0) {
      s <- c(s, v[i])
    }
  }
  s
}
f.uniq(c(9, 9, 1, 1, 1, 0))
## [1] 9 1 0
```

Exercise 5

Create a function that given a vector and an integer will return how many times the integer appears inside the vector.

```
f.count <- function (v, x) {
  count <- 0

  for (i in 1:length(v)) {
    if (v[i] == x) {
      count <- count + 1
    }
  }
  count
}
```

```
f.count(c(1:9, rep(10, 100)), 10)
## [1] 100
```

Exercise 6

Create a function that given a vector will print by screen the mean and the standard deviation, it will optionally also print the median.

```
desi <- function(x, med=FALSE) {

  mean <- round(mean(x), 1)
  stdv <- round(sd(x), 1)
  cat("Mean is:", mean, ", SD is:", stdv, "\n")

  if(med) {
    median <- median(x)
    cat("Median is:", median , "\n")
  }
}
desi(1:10, med=TRUE)
## Mean is: 5.5 , SD is: 3
## Median is: 5.5
```

Exercise 7

Create a function that given an integer will calculate how many divisors it has (other than 1 and itself). Make the divisors appear by screen.

```
f.div <- function(n) {
  i <- 2
  counter <- 0

  while(i <= n/2) {
    if(n%%i==0) {
      counter <- counter + 1
      cat (i ,"\n")
    }
    i <- i + 1
  }
  counter
}
f.div(13)
## [1] 0
f.div(16)
## 2
## 4
## 8
## [1] 3
```

Exercise 8

Create a function that given a data frame, and a number or character will return the data frame with the character or number changed to NA.

```
f.na <- function (df, otherna) {
  for(i in 1:ncol (df)) {
    for(j in 1:nrow (df)) {
      if(df[j,i] == otherna) {
        df[j,i] <- NA
      }
    }
  }
  df
}
carsnew <- f.na(cars, 10)
```

4.排序练习

注意：以下没有选择题！

Before proceeding, it might be helpful to look over the help pages for the `sort`, `order`, and `xtfrm` functions.

Exercise 1

Sort the vector x <- c(1, 3, 2, 5, 4) in:

a. ascending order
b. descending order

```
x <- c(1, 3, 2, 5, 4)
sort(x)
## [1] 1 2 3 4 5
sort(x, decreasing=T)
## [1] 5 4 3 2 1
```

Exercise 2

Sort the matrix x <- matrix(1:100, ncol=10):

a. in descending order by its second column (call the sorted matrix x1)
b. in descending order by its second row (call the sorted matrix x2)

```
x <- matrix(1:100, ncol=10)
x1 <- x[order(-x[,2]), ]
x2 <- x[, order(-x[2, ])]
```

Exercise 3

Sort only the first column of x in descending order.

```
x[, 1] <- sort(x[, 1])
```

Exercise 4

Consider the women data.

a. Confirm that the data are sorted in increasing order for both the height and weight variable, without looking at the data.
b. Create a new variable bmi, based on the following equation: BMI = (Weight in Pounds / (Height in inches) x (Height in inches)) x 703. Check, again without looking at the data, whether BMI increases monotonically with weight and height.
c. Sort the dataframe on bmi, and its variable names alphabetically

```
is.unsorted(women$height)
## [1] FALSE
is.unsorted(women$weight)
## [1] FALSE
women$bmi <- women$weight  / women$height^2 * 703
is.unsorted(women$bmi)
## [1] TRUE
women <- women[order(women$bmi), sort(names(women))]
women
```

Exercise 5

Consider the CO2 data.

a. Sort the data based on the Plant variable, alphabetically. (Note that Plant is a factor!). Check that the data are sorted correctly by printing the data on the screen.
b. Sort the data based on the uptake (increasing) and Plant (alphabetically) variables (in that order).
c. Sort again, based on uptake (increasing) and Plant (reversed alphabetically), in that order.

```
CO2 <- CO2[order(as.character(CO2$Plant)), ]
CO2 <- CO2[order(CO2$uptake, as.character(CO2$Plant)), ]
CO2 <- CO2[order(CO2$uptake, -xtfrm(as.character(CO2$Plant))), ]
```

Exercise 6

Create a dataframe df with 40 columns, as follows: df <- as.data.frame(matrix(sample(1:5, 2000, T), ncol=40))

a. Sort the dataframe on all 40 columns, from left to right, in increasing order.
a. Sort the dataframe on all 40 columns, from left to right, in decreasing order.

c. Sort the dataframe on all 40 columns, from right to left, in increasing order.

```r
df <- as.data.frame(matrix(sample(1:5, 2000, T), ncol=40))
df <- df[do.call(order, df), ]
df <- df[do.call(order, -df), ]
df <- df[do.call(order, rev(df)), ]
```

返回 课程主页 。