R Fundamentals: Exercises and Solutions

October 31, 2024

Exercises

Exercise 1. Write a function `mymax` which receives a vector `v` and return its greatest element. For example,

```
mymax(c(1, 5, -2, 3, -3))

[1] 5
```

Exercise 2. Write a function `optimum` that takes a vector `v` as input and returns a vector containing its minimum and maximum using built-in function `sort`.

Exercise 3. Write a function `mean2` that computes the mean of the squared values of a vector `v` using `mean`, and that takes additional arguments that it passes on to `mean` (e.g. `na.rm`).

Exercise 4. Fix the errors in the following code:

```
x <- c(1, 2, pi, 8)

# Only compute square roots if x exists and contains positive values:
if (exists(x)) {if (x > 0) {sqrt(x)}}
```

Exercise 5. The function `list.files` can be used to create a vector containing the names of all files in a folder. The `pattern` argument can be used to supply a regular expression describing a file name pattern. For instance, if `pattern = "\\.tex$"` is used, only `.tex` files will be listed. Create a function `printtex` that loops over all `.tex` files in `folder` and prints the name of each `.tex` file.

Exercise 6. An alternative to standardization is *normalization*, where all `numeric` variables are rescaled so that their smallest value is 0 and their largest value is 1. Write a function `normalize` that normalises the variables in a data frame containing `numeric` columns.

Exercise 7. Here we are working with the built-in dataset `Nile`.

1. What does the following mean?

   ```
   sum(Nile > 1200)
   ```

2. And this?

```
gt1200 <- which(Nile > 1200)
```

3. And this?

```
nile_gt_1200 <- Nile[which(Nile > 1200)]
mean(nile_gt_1200)
```

4. Write a function `mgt` that receive a vector `x` and a value `val` that returns the mean of the subset of `x` which are greater than `val`.
5. Write a function `n0` that receive a numeric vector `x` and report the number of 0's.
6. Write a function `rng` that receive a numeric vector `x` and report the difference between the maximal and the minimal elements of `x`.
7. Investigate the built-in function `range` when applying to `Nile`.

```
range(Nile)
```

Write a function `myrange` to replicate the result.

Exercise 8. Using `c`, `seq`, `rep`, `sequence`, `month.abb`, etc. to generate the following vectors:

```
1 1 1 1 1 2 2 2 2 3 3 3 4 4 5
1 4 7 10 13 16 19
A A A A A B B B B C C C D D E
b d f h j l n p r t v x z a c e g i k m o q s u w y
3 7 11 15 19 23 27 31 35 39
1 1 1 4 4
2 2 4 4 6 6 8 8 10 10 12 12 14 14 16 16 18 18 20 20
Jan Mar May Jul Sep Nov Feb Apr Jun Aug Oct Dec
8 7 6 5 7 6 5 4 6 5 4 3 5 4 3 2 4 3 2 1
1 2 3 4 5 6 2 3 4 5 6 3 4 5 6 4 5 6 5 6 6
```

Exercise 9. If we have the vector `colors`:

```
colors <- c('red', 'yellow', 'orange', 'beige')
```

Using `paste`, `paste0` with the vector `colors` to generate the following four string vectors

```
[1] "red flowers"    "yellow flowers" "orange flowers" "beige flowers"
[1] "redflowers"    "yellowflowers" "orangeflowers" "beigeflowers"
[1] "several red"    "several yellow" "several orange" "several beige"
[1] "I like red, yellow, orange, beige colors"
```

Exercise 10. Write a function `mysum` that receive `n`, `r` and compute $\sum_{k=0}^{n} r^k$. Compare with the exact result $\frac{1-r^{n+1}}{1-r}$ for $r = 1.08$ and $n = 10, 40, 100$.

Exercise 11. Write a function `fibon` that receive `n` and return the vector of the first `n` terms of the Fibonacci series: $1, 1, 2, 3, 5, 8, 13, \ldots$. For example,

```r
fibon(30)

 [1]      1      1      2      3      5      8     13     21     34     55
[11]     89    144    233    377    610    987   1597   2584   4181   6765
[21]  10946  17711  28657  46368  75025 121393 196418 317811 514229 832040
```

Exercise 12. Using the built-in function `nchar`, write a function `mycount` that receives a vector of words and return a vector consisting of 3 numbers: the length of the shortest word, the length of the longest word, and the average word length. For example,

```r
mycount(c('we', 'are', 'the', 'champions', 'right'))

[1] 2.0 9.0 4.4
```

Exercise 13. Write a function `testfreq` that receives a word and return a list of the number of occurences of each alphabet in the word. For example,

```r
testfreq('mississippi')

$m
[1] 1

$i
[1] 4

$s
[1] 4

$p
[1] 2
```

Exercise 14. Using loop and the built-in function `cat`, write a function `mytree` that receive `n` that produce the 'tree' in the console. For example, `mytree(5)` gives

```
*
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*
```

Exercise 15. Using double `for` loop to produce the following pattern:

```
1
1  2
1  2  3
1  2  3  4
1  2  3  4  5
```

Exercise 16. Using `for` loop to print the multplication table like

```
1x1= 1   1x2= 2   1x3= 3   1x4= 4   1x5= 5   1x6= 6   1x7= 7   1x8= 8   1x9= 9
2x1= 2   2x2= 4   2x3= 6   2x4= 8   2x5=10   2x6=12   2x7=14   2x8=16   2x9=18
3x1= 3   3x2= 6   3x3= 9   3x4=12   3x5=15   3x6=18   3x7=21   3x8=24   3x9=27
4x1= 4   4x2= 8   4x3=12   4x4=16   4x5=20   4x6=24   4x7=28   4x8=32   4x9=36
5x1= 5   5x2=10   5x3=15   5x4=20   5x5=25   5x6=30   5x7=35   5x8=40   5x9=45
6x1= 6   6x2=12   6x3=18   6x4=24   6x5=30   6x6=36   6x7=42   6x8=48   6x9=54
7x1= 7   7x2=14   7x3=21   7x4=28   7x5=35   7x6=42   7x7=49   7x8=56   7x9=63
8x1= 8   8x2=16   8x3=24   8x4=32   8x5=40   8x6=48   8x7=56   8x8=64   8x9=72
9x1= 9   9x2=18   9x3=27   9x4=36   9x5=45   9x6=54   9x7=63   9x8=72   9x9=81
```

Exercise 17. Sum

1. $\sum_{k=1}^{10000} k$
2. $\sum_{k=1}^{100} \min\{2^k, k^4\}$

with and without loops.

Exercise 18. Sum a vector `x` with loops.

Exercise 19. Write a function `myprod` to replicate the built-in function `prod`.

Exercise 20. Define the series $S_n = \sum_{i=1}^{n} \frac{(-1)^{i+1}}{2i-1} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots + \frac{(-1)^{n+1}}{2n-1}$, we know that $S_n \to \frac{\pi}{4}$. Verify this fact by computing $4S_{1000}$, $4S_{10000}$, $4S_{100000}$.

Exercise 21. Define $a_n = \frac{n+3}{n+8}$, $b_n = \frac{2n^2+3}{2n^2+8n}$, $c_n = \frac{\sqrt{n}}{\sqrt{n}+2}$; we know that

$$\lim_{n\to\infty} a_n = \lim_{n\to\infty} b_n = \lim_{n\to\infty} c_n = 1.$$

Reproduce the following:

```
       n        an        bn        cn
1      1 0.4444444 0.5000000 0.3333333
2     10 0.7222222 0.7250000 0.6125741
3     20 0.8214286 0.8364583 0.6909830
4     30 0.8684211 0.8838235 0.7325211
5     40 0.8958333 0.9099432 0.7597469
6     50 0.9137931 0.9264815 0.7795188
7     60 0.9264706 0.9378906 0.7947869
8     70 0.9358974 0.9462355 0.8070727
9     80 0.9431818 0.9526042 0.8172560
10    90 0.9489796 0.9576241 0.8258876
11   100 0.9537037 0.9616827 0.8333333
12 10000 0.9995004 0.9996002 0.9803922
```

Exercise 22. Generate an integer vector `number` of length 1000 as follows:

```r
set.seed(123456); number <- sample(0:100, 1000, replace = T)
```

Report the position of the 100th even integer by two different ways (`for` loop, `which`).

Exercise 23. Set `tg` as the built-in data frame `ToothGrowth`.

1. Compute mean tooth length for supplement VC and OJ.
2. Extract the sub data frame from `tg` with supplement OJ and length < 8.8.
3. Extract the sub data frame from `tg` with length > 28 or dose $= 1.0$.

Exercise 24. In `mtcars`, split the miles-per-gallon (mpg) data according to the number of cylinders (cyl).

Exercise 25. Imaging you have (age, gender) pairs as follows:

```r
age <- c(20, 16, 38, 55, 25)
gender <- c('M', 'F', 'M', 'F', 'F')
```

Split `age` into groups according to the corresponding elements of `gender` and find the mean in each group.

Exercise 26. Using `tapply` in `tg` to compute the mean length for each supplement group.

Exercise 27. In the built-in data `mtcars`

1. Find how many cars there are in each cylinder category.
2. Find mean and standard deviation of miles per gallon in each cylinder category.

Solutions

Solution to Exercise 1

```
mymax <- function(v) {
  m <- -Inf
  for (i in seq(v)) {
      if (v[i] > m) {
          m <- v[i]
      }
  }
  m
}
```

Solution to Exercise 2

```
optimum <- function(v) {
    # Sort x so that the minimum becomes the first element
    #             and the maximum becomes the last element:
    sorted_x <- sort(v)
    min_x <- sorted_x[1]
    max_x <- sorted_x[length(sorted_x)]
    return(c(min_x, max_x))
}
```

Check that this works:

```
x <- c(3, 8, 1, 4, 5)
optimum(x) # Should be 1, 8

# [1] 1 8
```

Solution to Exercise 3

```
mean2 <- function(v, ...) {
    return(mean(v^2, ...))
}
```

Check that this works:

```
x <- c(3, 2, 1)
mean2(x) # Should be 14 / 3 = 4.666...

# [1] 4.666667

x <- c(3, 2, NA)  # With NA
mean2(x) # Should be NA

# [1] NA

mean2(x, na.rm = TRUE) # Should be 13 / 2 = 6.5

# [1] 6.5
```

Solution to Exercise 4 There are two errors: the variable name in `exists` is not between quotes and `x > 0` evaluates to a vector an not a single value. The goal is to check that all values in `x` are positive, so `all` can be used to collapse the logical vector `x > 0`:

```
x <- c(1, 2, pi, 8)

# Errors
if (exists(x)) {if (x > 0) {sqrt(x)}}

# Error in exists(x): invalid first argument

# Still not right
if (exists("x")) {if (x > 0) {sqrt(x)}}

# Error in if (x > 0) {: the condition has length > 1

# Only compute square roots if x exists and contains positive values:
if (exists("x")) {if (all(x > 0)) {sqrt(x)}}

# [1] 1.000000 1.414214 1.772454 2.828427
```

Alternatively, we can get a better looking solution by using `&&`:

```
if (exists("x") && all(x > 0)) {sqrt(x)}

# [1] 1.000000 1.414214 1.772454 2.828427
```

Solution to Exercise 5

```
printtex <- function(folder) {
    files <- list.files(folder, pattern = "\\.tex$")
    for (file in files) {
        cat(file, "\n")
    }
}
# Test example
#printtex('/home/cytu/Downloads/CLP1Slides/src/sections')
```

Solution to Exercise 6

```
normalize <- function(df, ...) {
    for (i in seq_along(df)) {
        df[,i] <- (df[,i] - min(df[,i], ...)) /
                  (max(df[,i], ...) - min(df[,i], ...))
    }
    return(df)
}

aqn <- normalize(airquality, na.rm = TRUE)
summary(aqn)
```

```
#       Ozone            Solar.R            Wind              Temp
#  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
#  1st Qu.:0.1018   1st Qu.:0.3326   1st Qu.:0.3000   1st Qu.:0.3902
#  Median :0.1826   Median :0.6055   Median :0.4211   Median :0.5610
#  Mean   :0.2463   Mean   :0.5472   Mean   :0.4346   Mean   :0.5337
#  3rd Qu.:0.3728   3rd Qu.:0.7699   3rd Qu.:0.5158   3rd Qu.:0.7073
#  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
#  NA's   :37       NA's   :7
#       Month             Day
#  Min.   :0.0000   Min.   :0.0000
#  1st Qu.:0.2500   1st Qu.:0.2333
#  Median :0.5000   Median :0.5000
#  Mean   :0.4984   Mean   :0.4935
#  3rd Qu.:0.7500   3rd Qu.:0.7333
#  Max.   :1.0000   Max.   :1.0000
#
```

Solution to Exercise 7

```
mgt <- function(x, val) {
    return(mean(x[x > val]))
}

n0 <- function(x) {
    return(sum(x == 0))
}

rng <- function(x) {
    return(max(x) - min(x))
}

myrange <- function(x) {
    return(c(min(x), max(x)))
}
```

Solution to Exercise 8

```
cat(rep(1:5, 5:1))
cat(seq(1, 20, 3))
cat(rep(LETTERS[1:5], 5:1))
cat(c(letters[c(F, T)], letters[c(T, F)]))
cat(seq(3, 40, 4))
cat(rep(c(1, 4), c(3, 2)))
cat(rep(seq(2, 20, 2), each = 2))
cat(c(month.abb[c(T, F)], month.abb[c(F, T)]))
cat(sequence(rep(4, 5), from = 8:4, by = -1))
cat(sequence(6:1, from = 1:6))
```

Solution to Exercise 9

```
colors <- c('red', 'yellow', 'orange', 'beige')
paste(colors, 'flowers')
paste0(colors, 'flowers')
paste('several', colors)
paste('I like', paste(colors, collapse = ', '), 'colors')
```

Solution to Exercise 10

```
mysum <- function(n, r) {
    ans <- 0
    for (i in 0:n) {
        ans <- ans + r^i
    }
    ans
}
mysum(100, 1.08); (1 - 1.08^(100 + 1)) / (1 - 1.08)

[1] 29684.28
[1] 29684.28
```

Solution to Exercise 11

```
fibon <- function(n) {
    v <- vector(length=n)
    v[1] <- 1
    v[2] <- 1
    for (i in 3:n) {
        v[i] <- v[i - 1] + v[i - 2]
    }
    v
}
```

Solution to Exercise 12

```
mycount <- function(v) {
    x <- nchar(v)
    return(c(min(x), max(x), sum(x) / length(x)))
}
```

Solution to Exercise 13

```r
testfreq <- function(word) {
    l <- list()
    for (i in 1:nchar(word)) {
        w <- substr(word, i, i)
        if (is.null(l[[w]])) {
            l[[w]] <- 1
        } else {
            l[[w]] <- l[[w]] + 1
        }
    }
    l
}
```

Solution to Exercise 14

```r
mytree <- function(n) {
    for (i in c(1:n, (n-1):1)) {
        cat(rep('*', i), '\n')
    }
}
```

Solution to Exercise 15

```r
for (i in 1:5) {
    for (j in 1:i) {
        cat(j, ' ')
    }
    cat('\n')
}
```

Solution to Exercise 16

```r
for (i in 1:9) {
    for (j in 1:9) {
        cat(sprintf("%ix%i=%2i   ", i, j, i * j))
    }
    cat('\n')
}
```

Solution to Exercise 17

```r
ans <- 0
for (i in 1:10000) {
    ans <- ans + i
}
ans
```

```r
# [1] 50005000

sum(1:10000)

# [1] 50005000

ans <- 0
for (k in 1:100) {
    ans <- ans + min(2^k, k^4)
}
ans

# [1] 2050220551

v = 1:100; sum(pmin(2^v, v^4))

# [1] 2050220551
```

Solution to Exercise 18

```r
my_sum <- function(x) {
    ans <- 0
    for (i in 1:length(x)) {
        ans <- ans + x[i]
    }
    ans
}
my_sum(1:100)

# [1] 5050
```

Solution to Exercise 19

```r
myprod <- function(x) {
    ans <- 1
    for (i in seq(x)) {
        ans <- ans * x[i]
    }
    ans
}
myprod(1:10)

# [1] 3628800
```

Solution to Exercise 20

```r
mypi <- function(m) {
    n <- 1:m
    sum((-1)^(n+1)/(2 * n - 1)) * 4
}
4 * mypi(1000); 4 * mypi(10000); 4 * mypi(100000)

[1] 12.56237
[1] 12.56597
[1] 12.56633
```

Solution to Exercise 21

```r
n <- c(1, seq(10, 100, 10), 10000)
an <- (n + 3) / (n + 8)
bn <- (2 * n^2 + 3) / (2 * n^2 + 8 * n)
cn <- sqrt(n) / (sqrt(n) + 2)
(l <- data.frame(n, an, bn, cn))
```

Solution to Exercise 22

```r
j <- 0
for (i in 1:length(number)) {
     if (number[i] %% 2 == 0) {
         j <- j + 1
         if (j == 100) {
             cat(sprintf('The 100th number is %d, at place %d', number[i], i))
             break
         }
     }
}

The 100th number is 2, at place 218

(nn <- which(number %% 2 == 0)[100]); number[nn]

[1] 218
[1] 2
```

Solution to Exercise 23

1.
```r
tg <- ToothGrowth
tg_vc <- tg[tg$supp == 'VC',]
tg_oj <- tg[tg$supp == 'OJ',]
mean(tg_vc$len)

# [1] 16.96333
```

```r
mean(tg_oj$len)

# [1] 20.66333
```

2. ```r
   tg[tg$supp == 'OJ' & tg$len < 8.8,]

   #    len supp dose
   # 37 8.2   OJ  0.5
   ```

3. ```r
   w <- tg[tg$len == 28 | tg$dose == 1.0,]; head(w)

   #     len supp dose
   # 11 16.5   VC    1
   # 12 16.5   VC    1
   # 13 15.2   VC    1
   # 14 17.3   VC    1
   # 15 22.5   VC    1
   # 16 17.3   VC    1
   ```

Solution to Exercise 24

```r
(mtl <- split(mtcars$mpg, mtcars$cyl))

# $`4`
#  [1] 22.8 24.4 22.8 32.4 30.4 33.9 21.5 27.3 26.0 30.4 21.4
#
# $`6`
# [1] 21.0 21.0 21.4 18.1 19.2 17.8 19.7
#
# $`8`
#  [1] 18.7 14.3 16.4 17.3 15.2 10.4 10.4 14.7 15.5 15.2 13.3 19.2 15.8 15.0
```

Solution to Exercise 25

```r
(z <- tapply(age, gender, mean))

#  F  M
# 32 29
```

Solution to Exercise 26

```r
tapply(tg$len, tg$supp, mean)

#       OJ       VC
# 20.66333 16.96333
```

Solution to Exercise 27

```r
tapply(mtcars$cyl, mtcars$cyl, length)

#  4  6  8
# 11  7 14

tapply(mtcars$mpg, mtcars$cyl, mean)

#        4        6        8
# 26.66364 19.74286 15.10000

tapply(mtcars$mpg, mtcars$cyl, sd)

#        4        6        8
# 4.509828 1.453567 2.560048
```