Тюльпа Іван Юрійович

Звіт з лабораторної роботи 1

Завдання 1

(код + конспект без викликів можете подивитсь у директорії old/):

Конспект

Нижче буде фактично копіпаст з консолі R.
тобто заранений код позначається : `> ...`
а все інше, відповідно, без `>` – те що виводить код в консоль

```
> # print(<any object>) - displays the value of an object
> x ← c(1:3)
> print(c(1:3))
[1] 1 2 3
>
> # str(<any object>) - shows the structure of an object
> x ← c(1:3)
> str(x)
 int [1:3] 1 2 3
>
> # c(...) - combines values into a vector
> c(1:3)
[1] 1 2 3
>
> # seq(from, to, by, length.out) - generates sequences of numbers.
> seq(1, 5, 2)
[1] 1 3 5
```

```
>
> # rep(x, times, length.out, each) - repeats values
> rep((1:4), 8, , 8)
  [1] 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 3 3
3 3 3 3 3 3 4
 [58] 4 4 4 4 4 4 4 4 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 1 1 1 1 1 1 1 1 2 2 2
2 2 2 2 2 3 3
[115] 3 3 3 3 3 3 4 4 4 4 4 4 4 4 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 1 1 1 1
1 1 1 1 2 2 2
[172] 2 2 2 2 2 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 4 4 4 4 4
4 4 4 1 1 1 1
[229] 1 1 1 1 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4
>
> # numeric(length) - creates a numeric vector of length n filled with zeros
> n ← 5
> numeric(n)
[1] 0 0 0 0 0
>
> # names(x) / names(x) ← c(Names) - get or set element names for vector/list x
> v ← c("123123__" = 1, b = 2, c = 3)
> names(v)
[1] "123123__" "b"        "c"
> names(v) ← c("x", "y", "z")
> v
x y z
1 2 3
>
> # rbind(...) - binds vectors or matrices by rows
> rbind(c(1, 2), c(3, 4))
     [,1] [,2]
[1,]    1    2
[2,]    3    4
>
> # cbind(...) - binds vectors or matrices by columns
> cbind(c(1, 2), c(3, 4))
```

```
     [,1] [,2]
[1,]    1    3
[2,]    2    4
>
> # matrix(data, nrow, ncol, byrow, dimnames) - create a matrix from data
> matrix(1:6, nrow=2)
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> matrix(1:6, nrow=2, byrow = TRUE)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
>
> # dimnames(x) / dimnames(x) <- c(rowNames, colNames) - get/set both row and column names
> m <- matrix(1:4, nrow = 2)
> dimnames(m)
NULL
> dimnames(m) <- list(c("r1","r2"), c("c1","c2"))
> m
   c1 c2
r1  1  3
r2  2  4
>
> # rownames(x) / rownames(x) <- c(rowNames) - get/set row names
> m <- matrix(1:4, nrow = 2)
> dimnames(m) <- list(c("r1","r2"), c("c1","c2"))
> rownames(m)
[1] "r1" "r2"
> rownames(m) <- c("row1", "row2")
> rownames(m)
[1] "row1" "row2"
>
> # colnames(x) / colnames(x) <- value — get/set column names.
> m <- matrix(1:4, nrow = 2)
```

```
> dimnames(m) ← list(c("r1","r2"), c("c1","c2"))
> colnames(m)
[1] "c1" "c2"
> colnames(m) ← c("col1", "col2")
> colnames(m)
[1] "col1" "col2"
>
> # diag(x, nrow, ncol) - with numeric x as vector, create diagonal matrix; with matrix x extract
diagonal
> m ← matrix(1:4, nrow = 2)
> diag(m)
[1] 1 4
> diag(1:3)
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    2    0
[3,]    0    0    3
>
> # solve(A, b) solves Ax=b linear system. solve(A) returns inverse A
> A ← matrix(c(1, 4, 8, 8), 2, 2)
> b ← c(100, 2)
> solve(A, b)
[1] -32.66667  16.58333
> solve(A)
           [,1]        [,2]
[1,] -0.3333333  0.33333333
[2,]  0.1666667 -0.04166667
>
> # min(x, ...) - minimum value of x; also accepts multiple args
> min(c(2, 2, 8))
[1] 2
> min(c(2, 2, 8), -1, 123.45)
[1] -1
>
> # max(x, ...) - maximum value of x; also accepts multiple args
```

```
> max(c(2, 2, 8))
[1] 8
> max(c(2, 2, 8), -10000, 123.45)
[1] 123.45
>
> # max(x, ...) - sum of x values; also accepts multiple args
> sum(c(2, 2, 8))
[1] 12
> sum(c(2, 2, 8), -234, 123.45)
[1] -98.55
>
> # length(x) — number of elements in a vector (for matrix it returns total number of entries).
> length(1:5)
[1] 5
> length(matrix(1:6, nrow = 2))
[1] 6
>
> # t(x) — transpose of a matrix (or data frame-like object).
> x ← matrix(1:6, nrow = 2)
> x
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> t(x)
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
>
> # order(x) - returns the indices that would sort x.
> order(c(1, 100, 101230, 1, -1, -2, -543545))
[1] 7 6 5 1 4 2 3
>
> #sort(x) - returns sorted vector x
```

```
> sort(c(1, 100, 101230, 1, -1, -2, -543545))
[1] -543545      -2      -1       1       1     100  101230
```

Завдання 2

function:

```
> trim_sample ← function(x, alpha=0.05) {
+     n ← length(x)
+     trim_count ← floor(n * alpha / 2) # кількість елементів на видалення
+     sorted_idx ← order(x) # індекси відсортованого списку x
+     keep_idx ← sorted_idx[(trim_count + 1):(n - trim_count)] # беремо зріз, тобто повертаємо масив без
trim_count задніх елементів і trim_count передніх елементів
+     sorted_keep_idx ← sort(keep_idx) # відсотртовуємо індекси знову (щоб вони були по спадання, тобто
залишився попердній порядок елементів в x)
+     return(x[sorted_keep_idx]) # повертаємо список без trim_count мінімальних і trim_count максимальних
елементів
+ }
```

code:

```
> set.seed(123)
> x ← 1:5
> shuffled_x = sample(x)
> print(shuffled_x)
```

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

```
> trim_sample(shuffled_x, 0.5)
```

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

Завдання 3

function:
(Функція нижче нічого не повертає, просто print() матриці з атрибутами)

```
print_matrix ← function() {
  Matrix ← list(
    Title   = "Rocky",
    Creator = "Sylvestter Stallone",
    Type    = "Movie Franchise",
    Parts   = c("Rocky (1976",
                "Rock II (1979)",
                "Rocky III (1982)",
                "Rocky IV (1985)",
                "Rocky V (1990)",
                "Rocky Balboa (2006)"
    ),
    Characters = data.frame(
      Name = c("Rocky Balboa", "Apollo Creed", "Adrian Pennino", "Paulie Pennino",
               "Clubber Lang", "Ivan Drago"),
      Role = c("Protagonist", "Champion/Rival", "Partner", "Friend/Manager",
               "Antagonist", "Antagonist"),
      Age  = c(30, 33, 29, 40, 28, 26),
      Good = c(TRUE, TRUE, TRUE, TRUE, FALSE, FALSE),
      Wins        = c(57, 48, NA, NA, 23, 31),
      Losses      = c(23, 2, NA, NA, 1, 1),
      stringsAsFactors = FALSE
    ),
    Locations = c("Philadelphia", "Los Angeles", "Las Vegas", "Moscow")
  )
```

```
  print(Matrix)
}
```

(Нижче суто вивід коду, цей файл можна подивитись у old/t3.R)
output:

```
$Title
[1] "Rocky"

$Creator
[1] "Sylvestter Stallone"

$Type
[1] "Movie Franchise"

$Parts
[1] "Rocky (1976"         "Rock II (1979)"      "Rocky III (1982)"    "Rocky IV (1985)"     "Rocky V
(1990)"
[6] "Rocky Balboa (2006)"

$Characters
            Name           Role Age  Good Wins Losses
1   Rocky Balboa    Protagonist  30  TRUE   57     23
2   Apollo Creed Champion/Rival  33  TRUE   48      2
3 Adrian Pennino        Partner  29  TRUE   NA     NA
4 Paulie Pennino Friend/Manager  40  TRUE   NA     NA
5   Clubber Lang     Antagonist  28 FALSE   23      1
6     Ivan Drago     Antagonist  26 FALSE   31      1

$Locations
[1] "Philadelphia" "Los Angeles"  "Las Vegas"    "Moscow"
```

Завдання 4

у **R** я не знайшов функцію що рахує Garmonic Mean, тому зробив наступне:

$$\begin{aligned}\mathrm{GM}(x_1,\ldots,x_n) &= \left(\prod_{i=1}^{n} x_i\right)^{1/n} \\ &= \exp\left(\log\left(\prod_{i=1}^{n} x_i)\right)^{\frac{1}{n}}\right) \\ &= \exp\left(\frac{1}{n}\sum_{i=1}^{n}\log x_i\right) \\ &= \exp\big(\mathrm{mean}(\log x)\big),\end{aligned}$$

function:

```r
stats ← function(x) {
  result ← c(
    Mean  = mean(x),
    GM    = exp(mean(log(x))),              # геометричне
    HM    = length(x) / sum(1/x),           # гармонійне
    Med   = median(x),
    SV    = var(x),                         # дисперсія
    SD    = sd(x),
    MAD   = mean(abs(x - mean(x))),         # середнє абсолютне відхилення
    MADm  = mean(abs(x - median(x))),       # від медіани
    IQR   = IQR(x),
    Range = max(x) - min(x)                 # Розмах вибірки
  )
  return(result)
}
```

code:

```r
x ← c(6.7, 3.4, 0.8)
stats(x) # Виклик функції
```

```
 Mean       GM       HM      Med       SV       SD      MAD     MADm      IQR    Range
3.633333 2.631568 1.771614 3.400000 8.743333 2.956913 2.044444 1.966667 2.950000 5.900000
```

```
    Mean       GM       HM      Med       SV       SD      MAD     MADm      IQR    Range
3.633333 2.631568 1.771614 3.400000 8.743333 2.956913 2.044444 1.966667 2.950000 5.900000
```

output.png

(Вставив скіншот з консольі R, бо назви статистик криво відображаються в `output`)

Завдання 5

(У завданні було сказано повернути тільки суму, але я ще повертаю ітерації щоб більш явно бачити на якій ітерації $\sum_{n=1}^{m} n^{\alpha}$ зійшлось.)

function:

```r
sum_series ← function(alpha, delta = 1e-6, max_iter = 10000) {
  if (alpha ⩾ -1) {
    stop("Error: the conditions for the convergence of the series are not satisfied")
  }

  s ← 0.0
  for (n in 1:max_iter) {
    term ← exp(alpha * log(n))
    s_new ← s + term
    if (abs(s_new - s) < delta) {
      return(list(sum = s_new, iterations = n))
    }
    s ← s_new
```

```
  }
  stop("Error: iteration limit exceeded")
}
```

code:

```
sum_series(-2) # виклик функції
```

```
$sum
[1] 1.643935

$iterations
[1] 1000
```

```
tryCatch({
  sum_series(0) # виклик функції
}, error = function(e) {
  message("Recieved ", e$message)
})
```

```
Recieved Error: the conditions for the convergence of the series are not satisfied
```

```
> tryCatch({
+   sum_series(-2, 1e-8, 10) # Виклик функції
+ }, error = function(e) {
+   message("Recieved ", e$message)
+ })
```

```
Recieved Error: iteration limit exceeded
```