РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3

дисциплина: Компьютерный практикум по статистическому анализу данных

Студент: Мухамедияр Адиль

Группа: НКНбд-01-20

МОСКВА

2023г.

Введение

В данном отчете представлены решения различных задач на языке программирования Julia. Задачи охватывают работу с циклами while и for, с условными выражениям, а также использование специализированных функций, таких как sayhi(), sort(), sort!(), map() и следующий пакетов: Colors, LinearAlgebra, Random.

Выполнение работы

1. Разбор материала и изучение тематики

В данном разделе были повторены примеры из данного раздела для изучения и понимание работы нового материала. Ниже представленных фотоматериалах мы можем увидеть что были разобраны циклы for и while без проблем, так как они похожи на синтаксис языка Python. Далее мы разобрали условные выражения и несколько функций, изучив их работу и вызов функций. Наконец, нами была вызвана библиотека Colors и использована, тем самым мы получили палитру из случайных(рандомных) 9 цветов в качестве таблицы размерностью 3x3.

Задание(повторяем примеры из раздела)

Циклы while и for

[6]:
```
# пока n<10 прибавить к n единицу и распечатать значение:
n = 0
while n < 10
    n += 1
    println(n)
end
```

[9]:
```
myfriends = ["Ted", "Robyn", "Barney", "Lily", "Marshall"]
for friend in myfriends
    println("Hi $friend, it's great to see you!")
end
```

```
Hi Ted, it's great to see you!
Hi Robyn, it's great to see you!
Hi Barney, it's great to see you!
Hi Lily, it's great to see you!
Hi Marshall, it's great to see you!
```

[13]:
```
# инициализация массива m x n из нулей:
m, n = 5, 5
A = fill(0, (m, n))

# формирование массива, в котором значение каждой записи
# является суммой индексов строки и столбца:
for i in 1:m
    for j in 1:n
        A[i, j] = i + j
    end
end
A
```

[13]:
```
5x5 Matrix{Int64}:
 2  3  4  5   6
 3  4  5  6   7
 4  5  6  7   8
 5  6  7  8   9
 6  7  8  9  10
```

[14]:
```
# инициализация массива m x n из нулей:
B = fill(0, (m, n))
for i in 1:m, j in 1:n
    B[i, j] = i + j
end
B
```

[14]:
```
5x5 Matrix{Int64}:
 2  3  4  5   6
 3  4  5  6   7
 4  5  6  7   8
 5  6  7  8   9
 6  7  8  9  10
```

[15]:
```
C = [i + j for i in 1:m, j in 1:n]
C
```

[15]:
```
5x5 Matrix{Int64}:
```

[15]:
```
C = [i + j for i in 1:m, j in 1:n]
C
```

[15]:
```
5x5 Matrix{Int64}:
 2  3  4  5   6
 3  4  5  6   7
 4  5  6  7   8
 5  6  7  8   9
 6  7  8  9  10
```

Условные выражения

[18]:
```
# используем `&&` для реализации операции "AND"
# операция % вычисляет остаток от деления
N = 15
if (N % 3 == 0) && (N % 5 == 0)
    println("FizzBuzz")
elseif N % 3 == 0
    println("Fizz")
elseif N % 5 == 0
    println("Buzz")
else
    println(N)
end
```

```
FizzBuzz
```

[20]:
```
# a ? b : c(ознаечает if (a): b; else: c)
x = 5
y = 10
(x > y) ? x : y
```

[20]:
```
10
```

Функции

[21]:
```
function sayhi(name)
    println("Hi $name, it's great to see you!")
end
# функция возведения в квадрат:
function f(x)
    x^2
end
```

[21]:
```
f (generic function with 1 method)
```

[21]:
```
f (generic function with 1 method)
```

[22]:
```
# Вызов функции
sayhi("C-3PO")
f(42)
```

```
Hi C-3PO, it's great to see you!
```

[22]:
```
1764
```

[24]:
```
# задаём массив v:
v = [3, 5, 2]
sort(v)
v
```

[24]:
```
3-element Vector{Int64}:
 3
 5
 2
```

Функция sort(v) возвращает отсортированный массив, который содержит те же элементы, что и массив v, но исходный массив v остаётся без изменений. Если же исполь- зовать sort!(v), то отсортировано будет содержимое исходного массива v.

[25]:
```
sort!(v)
v
```

[25]:
```
3-element Vector{Int64}:
 2
 3
 5
```

[26]:
```
map(f, [1, 2, 3])
```

[26]:
```
3-element Vector{Int64}:
 1
 4
 9
```

[27]:
```
x -> x^3
map(x -> x^3, [1, 2, 3])
```

[27]:
```
3-element Vector{Int64}:
  1
  8
 27
```

[28]:
```
f(x) = x^2
broadcast(f, [1, 2, 3])
# Задаём матрицу A:
A = [i + 3*j for j in 0:2, i in 1:3]
```

[28]:
```
3×3 Matrix{Int64}:
 1  2  3
 4  5  6
 7  8  9
```

[29]:
```
# Вызываем функцию f возведения в квадрат
f(A)
```

[29]:
```
3×3 Matrix{Int64}:
  30   36   42
  66   81   96
 102  126  150
```

[30]:
```
B = f.(A)
```

[30]:
```
3×3 Matrix{Int64}:
   1   4   9
  16  25  36
  49  64  81
```

Точечный синтаксис для broadcast() позволяет записать относительно сложные со- ставные поэлементные выражения в форме, близкой к математической записи.

[31]:
```
A .+ 2 .* f.(A) ./ A
broadcast(x -> x + 2 * f(x) / x, A)
```

[31]:
```
3×3 Matrix{Float64}:
  3.0   6.0   9.0
 12.0  15.0  18.0
 21.0  24.0  27.0
```

Сторонние библиотеки (пакеты) в Julia

[33]:
```
import Pkg
Pkg.add("Example")
```

```
   Resolving package versions...
  No Changes to `C:\Users\adiks\.julia\environments\v1.9\Project.toml`
  No Changes to `C:\Users\adiks\.julia\environments\v1.9\Manifest.toml`
```

[33]:
```
import Pkg
Pkg.add("Example")
```

```
   Resolving package versions...
  No Changes to `C:\Users\adiks\.julia\environments\v1.9\Project.toml`
  No Changes to `C:\Users\adiks\.julia\environments\v1.9\Manifest.toml`
```

[34]:
```
Pkg.add("Colors")
using Colors
```

```
   Resolving package versions...
   Installed ColorTypes ──────── v0.11.4
   Installed FixedPointNumbers ─ v0.8.4
   Installed Reexport ────────── v1.2.2
   Installed Colors ──────────── v0.12.10
    Updating `C:\Users\adiks\.julia\environments\v1.9\Project.toml`
  [5ae59095] + Colors v0.12.10
    Updating `C:\Users\adiks\.julia\environments\v1.9\Manifest.toml`
  [3da002f7] + ColorTypes v0.11.4
  [5ae59095] + Colors v0.12.10
  [53c48c17] + FixedPointNumbers v0.8.4
  [189a3867] + Reexport v1.2.2
  [37e2e46d] + LinearAlgebra
  [2f01184e] + SparseArrays
  [10745b16] + Statistics v1.9.0
  [e66e0078] + CompilerSupportLibraries_jll v1.0.5+0
  [4536629a] + OpenBLAS_jll v0.3.21+4
  [bea87d4a] + SuiteSparse_jll v5.10.1+6
  [8e850b90] + libblastrampoline_jll v5.8.0+0
Precompiling project...
  ✓ Reexport
  ✓ CompilerSupportLibraries_jll
  ✓ FixedPointNumbers
  ✓ ColorTypes
  ✓ Colors
  5 dependencies successfully precompiled in 16 seconds. 21 already precompiled.
```

[35]:
```
palette = distinguishable_colors(100)
```

[35]:

[41]:
```
rand(palette, 3, 3)
```

[41]:

2. Самостоятельная работа

2.1. Работа с Циклами и Коллекциями

- Выведены целые числа от 1 до 100 и их квадраты.
- Создан словарь squares, где ключи - целые числа, а значения - их квадраты.
- Сформирован массив squares_arr с квадратами чисел от 1 до 100.

Самостоятельная работа

Задача 1

[42]:
```julia
# Часть 1: Вывод целых чисел от 1 до 100 и их квадратов
for i in 1:100
    println("$i, $(i^2)")
end

# Часть 2: Создание словаря с квадратами чисел
squares = Dict(i => i^2 for i in 1:100)

# Часть 3: Создание массива с квадратами чисел
squares_arr = [i^2 for i in 1:100]
```

```
1, 1
2, 4
3, 9
4, 16
5, 25
6, 36
7, 49
8, 64
9, 81
10, 100
11, 121
12, 144
13, 169
14, 196
15, 225
16, 256
17, 289
18, 324
19, 361
20, 400
21, 441
22, 484
23, 529
24, 576
25, 625
26, 676
27, 729
28, 784
29, 841
30, 900
31, 961
32, 1024
33, 1089
34, 1156
35, 1225
36, 1296
37, 1369
38, 1444
```

2.2. Условные Операторы

- Реализованы условные операторы для проверки четности числа. Использован как обычный if-else, так и тернарный оператор.

2.3. Функции

- Написана функция add_one, увеличивающая свой аргумент на 1.

Задача 2

[45]:
```
# Функция для проверки четности числа и вывода результата
function check_even_odd(n)
    if n % 2 == 0
        println(n)
    else
        println("нечётное")
    end
    # Тернарный оператор
    n % 2 == 0 ? println(n) : println("нечётное")
end

# Пример использования функции check_even_odd с числом 4 и 5
check_even_odd(4)
check_even_odd(5)
```

```
4
4
нечётное
нечётное
```

Задача 3

[48]:
```
# Функция add_one
add_one = x -> x + 1

# Пример использования функции add_one с числом 5
add_one_result = add_one(5)
```

[48]:
```
6
```

2.4. Работа с Матрицами

- Продемонстрировано использование map и broadcast для создания матрицы, где каждый элемент увеличен на 1.

2.5. Математические Операции с Матрицами

- Создана и преобразована матрица согласно заданным условиям, включая возведение в куб и изменение столбцов.

2.6. Создание Специфичных Матриц

- Сформирована матрица B по заданным правилам и вычислена матрица C как произведение B^T и B.

Задача 4

[49]:
```
# Создание матрицы A размером 5x5, где каждый элемент увеличивается на 1
A = reshape(1:25, 5, 5)

# Использование map или broadcast для увеличения каждого элемента на 1
A_map = map(x -> x + 1, A)
A_broadcast = A .+ 1
```

[49]:
```
5×5 Matrix{Int64}:
 2   7  12  17  22
 3   8  13  18  23
 4   9  14  19  24
 5  10  15  20  25
 6  11  16  21  26
```

Задача 5

[50]:
```
# Инициализация матрицы A
A = [1 1 3; 5 2 6; -2 -1 -3]

# Вычисление A^3
A_cubed = A^3

# Замена третьего столбца на сумму второго и третьего столбцов
A[:, 3] = A[:, 2] + A[:, 3]
```

[50]:
```
3-element Vector{Int64}:
  4
  8
 -4
```

Задача 6

[52]:
```
# Создание матрицы B размером 15x3
B = [10 -10 10; 10 -10 10; 10 -10 10; 10 -10 10; 10 -10 10;
     10 -10 10; 10 -10 10; 10 -10 10; 10 -10 10; 10 -10 10;
     10 -10 10; 10 -10 10; 10 -10 10; 10 -10 10; 10 -10 10]

# Вычисление матрицы C = B^T * B
C = transpose(B) * B
```

[52]:
```
3×3 Matrix{Int64}:
  1500  -1500   1500
 -1500   1500  -1500
  1500  -1500   1500
```

2.7. Алгоритмическое Заполнение Матриц

- Созданы и заполнены матрицы Z1, Z2, Z3 и Z4, используя циклы и специфичные правила заполнения.

Задача 7

[56]:
```
# Создание матриц Z, E размером 6x6
Z = zeros(Int, 6, 6)
E = ones(Int, 6, 6)

# Создание матриц Z1, Z2, Z3, Z4
Z1, Z2, Z3, Z4 = copy(Z), copy(Z), copy(Z), copy(Z)

# Заполнение матриц Z1 и Z2
for i in 1:6
    for j in 1:6
        if i == j || i == 7 - j
            Z1[i, j] = 1 - (i + j) % 2
            Z2[i, j] = (i + j) % 2
        end
    end
end

# Заполнение матриц Z3 и Z4
for i in 1:6
    for j in 1:6
        if (i + j) % 2 == 1
            Z3[i, j] = 1
            Z4[7 - i, 7 - j] = 1
        end
    end
end

println("Z1 = ", Z1)
println("Z2 = ", Z2)
println("Z3 = ", Z3)
println("Z4 = ", Z4)
```

```
Z1 = [1 0 0 0 0 0; 0 1 0 0 0 0; 0 0 1 0 0 0; 0 0 0 1 0 0; 0 0 0 0 1 0; 0 0 0 0 0 1]
Z2 = [0 0 0 0 0 1; 0 0 0 0 1 0; 0 0 0 1 0 0; 0 0 1 0 0 0; 0 1 0 0 0 0; 1 0 0 0 0 0]
Z3 = [0 1 0 1 0 1; 1 0 1 0 1 0; 0 1 0 1 0 1; 1 0 1 0 1 0; 0 1 0 1 0 1; 1 0 1 0 1 0]
Z4 = [0 1 0 1 0 1; 1 0 1 0 1 0; 0 1 0 1 0 1; 1 0 1 0 1 0; 0 1 0 1 0 1; 1 0 1 0 1 0]
```

2.8. Функциональное Программирование

- Реализована функция outer, аналогичная функции из языка R, для выполнения матричных операций с возможностью изменения операции.

2.9. Решение Системы Линейных Уравнений

- Решена система линейных уравнений с пятью неизвестными, используя соответствующее матричное уравнение.

Задача 8

[63]:
```
function custom_outer(x, y, operation)
    result = zeros(length(x), length(y))
    for i in 1:length(x)
        for j in 1:length(y)
            result[i, j] = operation(x[i], y[j])
        end
    end
    return result
end

# Пример использования функции custom_outer
x = [1, 2, 3]
y = [4, 5, 6]
result = custom_outer(x, y, (a, b) -> a * b)  # Пример с умножением
```

[63]:
```
3×3 Matrix{Float64}:
  4.0   5.0   6.0
  8.0  10.0  12.0
 12.0  15.0  18.0
```

Задача 9

[64]:
```
using LinearAlgebra

# Определение матрицы коэффициентов A и вектора свободных членов y
A = [1 2 3 4 5; 2 1 2 3 4; 3 2 1 2 3; 4 3 2 1 2; 5 4 3 2 1]
y = [7, -1, -3, 5, 17]

# Решение системы линейных уравнений
x = A \ y
```

[64]:
```
5-element Vector{Float64}:
 -2.0000000000000036
  3.0000000000000058
  4.999999999999998
  1.9999999999999991
 -3.999999999999999
```

2.10. Создание матрицы

Здесь были реализованы все полученные знания по этой лабораторной работы:

- Использование библиотеки, в данном случае Random;
- Благодоря циклу for были подсчитаны количества элементов;
- Использовали математические подсчеты над матрицами.

Задача 10

[73]:
```
using Random

# Создание матрицы M размером 6x10 со случайными элементами от 1 до 10
M = rand(1:10, 6, 10)

# N задано как 4
N = 4
# Подсчет количества элементов в каждой строке, больших N
count_greater_than_N = [sum(row .> N) for row in eachrow(M)]

# M_value задано как 7
M_value = 7
# Определение строк, где M_value встречается ровно 2 раза
rows_with_M_value_twice = [sum(row .== M_value) == 2 for row in eachrow(M)]

# K задано как 75
K = 75
# Определение пар столбцов, сумма элементов которых больше K
column_pairs_sum_greater_than_K = []
for i in 1:size(M, 2)
    for j in (i + 1):size(M, 2)
        if sum(M[:, i] + M[:, j]) > K
            push!(column_pairs_sum_greater_than_K, (i, j))
        end
    end
end

# Вывод результатов
println("Матрица M:\n", M)
println("Количество элементов в каждой строке больше N (", N, "): ", count_greater_than_N)
println("Строки, где значение ", M_value, " встречается ровно 2 раза: ", rows_with_M_value_twice)
println("Пары столбцов, сумма элементов которых больше K (", K, "): ", column_pairs_sum_greater_than_K)
```

```
Матрица M:
[3 5 1 9 2 7 7 8 4 7; 10 3 8 7 4 3 8 10 8 8; 8 5 1 6 3 9 3 5 6 8; 10 5 1 10 1 2 1 4 4 6; 8 1 5 7 4 9 4 3 3 4; 3 2 1 5 7 8 7 8 3 7]
Количество элементов в каждой строке больше N (4): [6, 7, 7, 4, 4, 6]
Строки, где значение 7 встречается ровно 2 раза: Bool[0, 0, 0, 0, 0, 0]
Пары столбцов, сумма элементов которых больше K (75): Any[(1, 4), (1, 6), (1, 8), (1, 10), (4, 6), (4, 8), (4, 10), (6, 8), (6, 10), (8, 10)]
```

2.11. Вычисления

- Выполнены вычисления используя функцию sum

Задача 11

[77]:
```julia
# Вычисление первой суммы
sum1 = sum([i^4 / (3 + j) for i in 1:20, j in 1:5])
```

[77]:
```
639215.2833333334
```

[78]:
```julia
# Вычисление второй суммы
sum2 = sum([i^4 / (3 + i * j) for i in 1:20, j in 1:5])
```

[78]:
```
89912.02146097137
```

Заключение

Решение представленных задач на языке Julia показало гибкость и мощь этого языка программирования в области численных и математических вычислений. Работа с разнообразными циклами, операциями над матрицами и использование специализированных пакетов демонстрируют широкие возможности Julia для решения широкого круга задач.