Лабораторная работа №2. Компьютерный практикум по статистическому анализу данных

Леснухин Даниил Дмитриевич
Российский университет дружбы народов
Москва

Цель работы

Основная цель работы – изучить несколько структур данных, реализованных в Julia, научиться применять их и операции над ними для решения задач.

Задание

1. Используя Jupyter Lab, повторите примеры.
2. Выполните задания для самостоятельной работы.

Теоретическое введение

Julia – высокоуровневый свободный язык программирования с динамической типизацией, созданный для математических вычислений [@julialang]. Эффективен также и для написания программ общего назначения. Синтаксис языка схож с синтаксисом других математических языков, однако имеет некоторые существенные отличия.

Для выполнения заданий была использована официальная документация Julia [@juliadoc].

Рассмотрим несколько структур данных, реализованных в Julia. Несколько функций (методов), общих для всех структур данных:

- isempty() – проверяет, пуста ли структура данных;
- length() – возвращает длину структуры данных;
- in() – проверяет принадлежность элемента к структуре;
- unique() – возвращает коллекцию уникальных элементов структуры,
- reduce() – свёртывает структуру данных в соответствии с заданным бинарным оператором;
- maximum() (или minimum()) – возвращает наибольший (или наименьший) результат вызова функции для каждого элемента структуры данных.

Выполнение лабораторной работы

Для начала выполним примеры из раздела про кортежи

Теперь выполним примеры из раздела про словари

```julia
# пустой кортеж:
()
```

```
()
```

```julia
# кортеж из элементов типа String:
favoritelang = ("Python","Julia","R")
```

```
("Python", "Julia", "R")
```

```julia
# кортеж из целых чисел:
x1 = (1, 2, 3)
```

```
(1, 2, 3)
```

```julia
# кортеж из элементов разных типов:
x2 = (1, 2.0, "tmp")
```

```
(1, 2.0, "tmp")
```

```julia
# именованный кортеж:
x3 = (a=2, b=1+2)
```

```
(a = 2, b = 3)
```

```julia
# длина кортежа x2:
length(x2)
```

```
3
```

```julia
# обратиться к элементам кортежа x2:
x2[1], x2[2], x2[3]
```

```
(1, 2.0, "tmp")
```

```julia
# произвести какую-либо операцию (сложение)
# с вторым и третьим элементами кортежа x1:
c = x1[2] + x1[3]
```

```
5
```

```julia
# обращение к элементам именованного кортежа x3:
x3.a, x3.b, x3[2]
```

Рис. 1: Примеры из раздела кортежи

Выполним примеры из раздела про множества

Выполним примеры из раздела про массивы

Теперь перейдем к выполнению заданий.

Задание №1

Даны множества: $A = 0, 3, 4, 9, B = 1, 3, 4, 7, C = 0, 1, 2, 4, 7, 8, 9$. Найдем $P = A \cap B \cup A \cap B \cup A \cap C \cup B \cap C$

Задание №2

Приведем свои примеры с выполнением операций над множествами элементов разных типов

Рис. 2: Примеры использования массивов

Рис. 3: Примеры использования массивов

Рис. 4: Примеры использования массивов

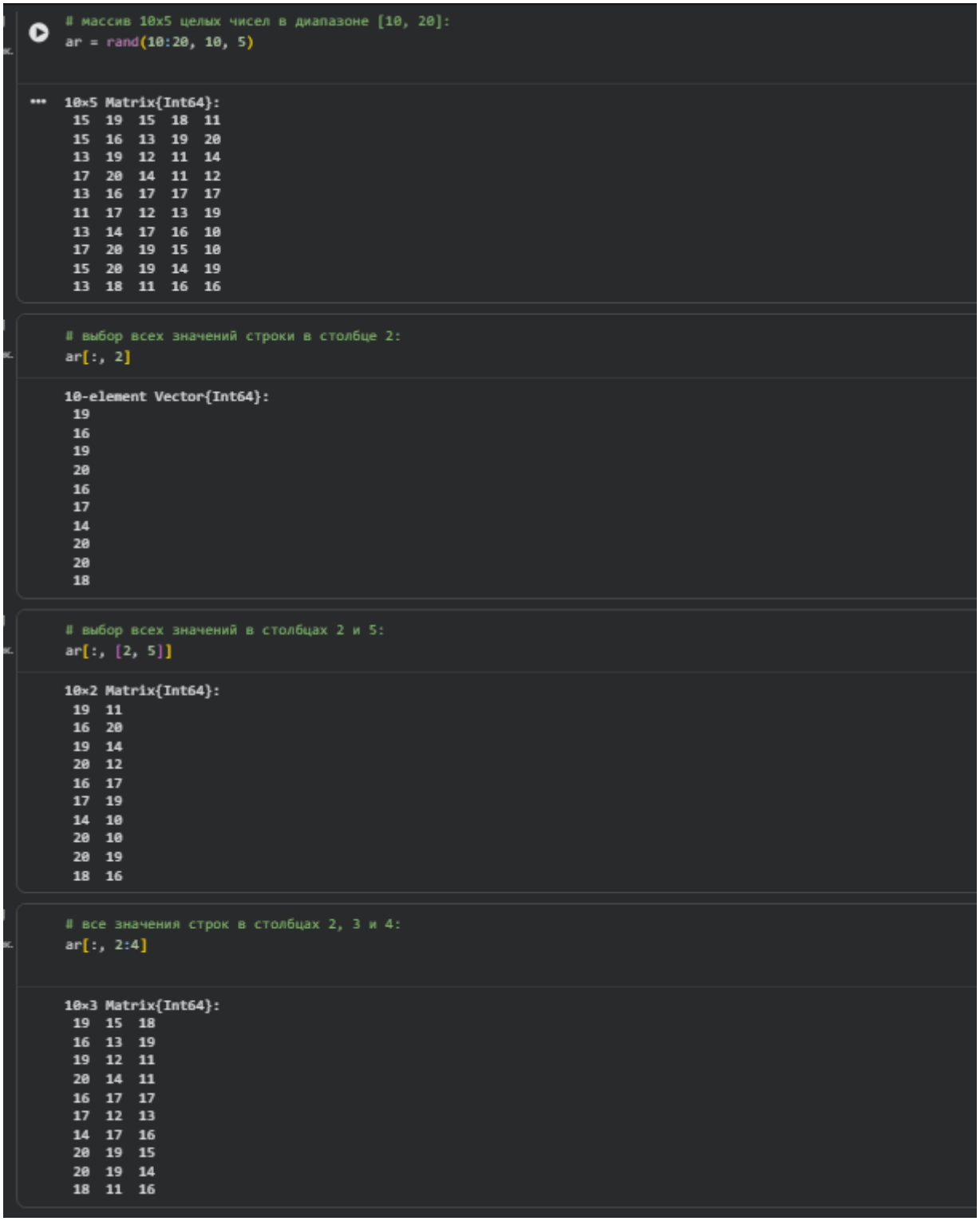

Рис. 5: Примеры использования массивов

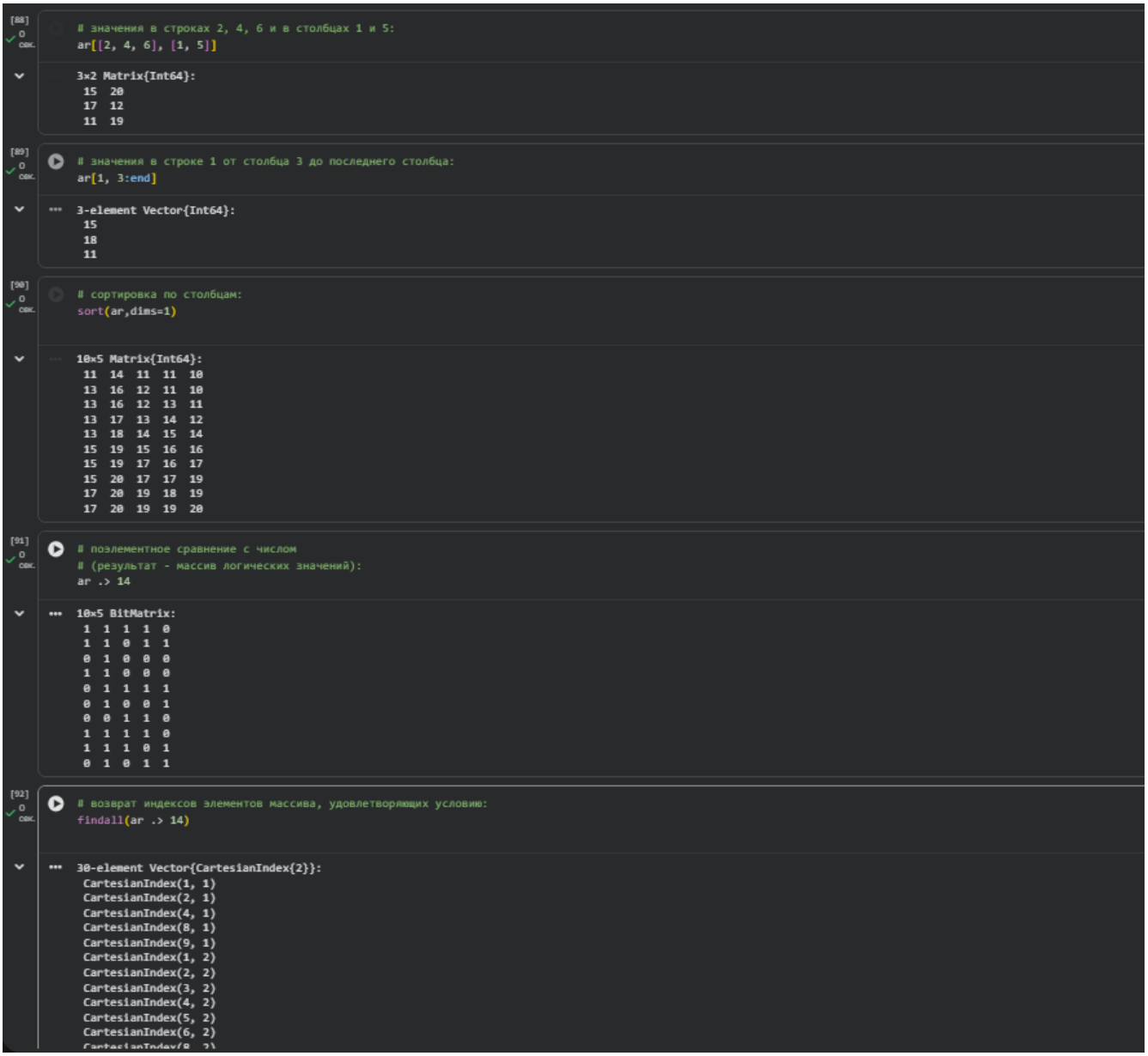

Рис. 6: Примеры использования массивов

```
A = Set([0, 3, 4, 9]);
B = Set([1, 3, 4, 7]);

C = Set([0, 1, 2, 4, 7, 8, 9]);
```

```
P = union(intersect(A, B), intersect(A, B), intersect(A, C), intersect(B, C))
```

```
Set{Int64} with 6 elements:
  0
  4
  7
  9
  3
  1
```

Рис. 7: Задание №1. Работа с множествами

Задание №3

Создадим массивы разными способами, используя циклы

3

```
[113]
d = [i for i in 1:21]
print(d, '\n')
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21]
```

```
[114]
c = []
c_1 = reverse([i for i in 1 :21])
print(c_1, '\n')
```

```
[21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

```
[115]
print(vcat(d, c_1), '\n')
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

```
[116]
tmp = [4 6 3]

println(vcat(fill.(tmp, [10 2 1])))
```

```
[[4, 4, 4, 4, 4, 4, 4, 4, 4, 4] [6, 6] [3]]
```

```
[117]
println(vcat(fill.(tmp, [10 10 10])...))
```

```
[4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
```

```
[118]
println(vcat(fill.(tmp, [11 10 10])...))
```

```
[4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
```

```
[119]
println(vcat(fill.(tmp, [10 20 30])...))
```

```
[4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
```

```
[121]
tmp_deg = []
for i in 1:3
  push!(tmp_deg, 2^tmp[i])
end

println(vcat(fill.(tmp_deg, [1, 1, 4])...))
count = 0
for i in tmp_deg
  if '6' in string(i)
    count = count + 1
  end
end
println(count)
```

```
[16, 64, 8, 8, 8, 8]
2
```

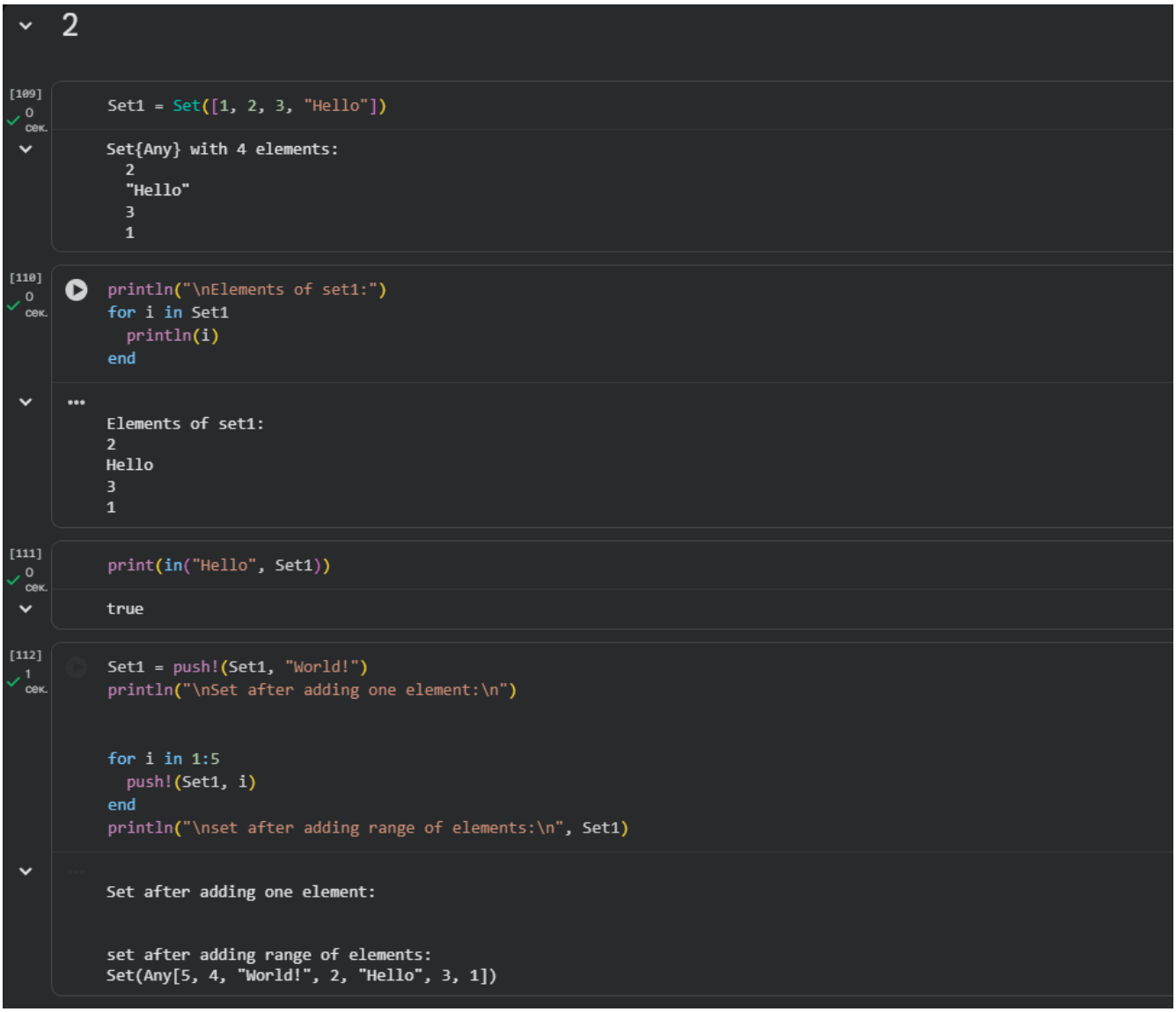

Рис. 8: Задание №2. Примеры операций над множествами элементов разных типов

Задание №4

Создадим массив squares, в котором будут храниться квадраты всех целых чисел от 1 до 100

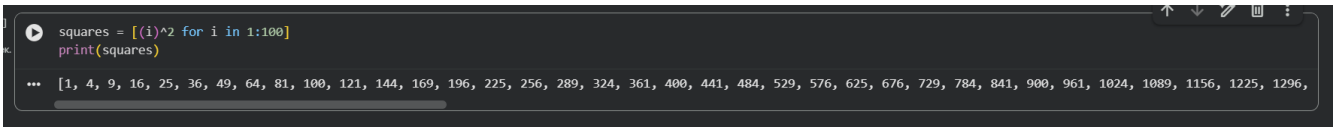

Рис. 9: Задание №4.

Задание №5

Подключим пакет Primes (функции для вычисления простых чисел). Сгенерируем массив myprimes, в котором будут храниться первые 168 простых чисел. Определим 89-е наименьшее простое число. Получии срез массива с 89-го до 99-го элемента включительно, содержащий наименьшие простые числа.

Задание №6

Рис. 10: Задание №5. Работа с пакетом Primes

Вычислим следующие выражения

$$\sum_{i=10}^{100}(i^3+4i^2);$$

$$\sum_{i=1}^{M}(\frac{2^i}{i}+\frac{3^i}{i^2}), M=25;$$

$$1+\frac{2}{3}+\left(\frac{2}{3}\frac{4}{5}\right)+\left(\frac{2}{3}\frac{4}{5}\frac{6}{7}\right)+\cdots+\left(\frac{2}{3}\frac{4}{5}\cdots\frac{39}{39}\right).$$

Выводы

В результате выполнения данной лабораторной работы я изучил несколько структур дыннх, реализованных в Julia, Научился применять их и операции над ними для решения задач.