

# Компьютерный практикум по статистическому анализу данных

Лабораторная работа № 2. Julia. Структуры данных

Демидова Екатерина Алексеевна

# Содержание

1	Введение	4
2	Теоретическое введение	5
3	Выполнение лабораторной работы	6
4	Выводы	16
	Список литературы	17

## Список иллюстраций

3.1	Примеры. Кортежи . . . . .	6
3.2	Примеры. Множества . . . . .	7
3.3	Примеры. Массивы . . . . .	8
3.4	Примеры. Массивы . . . . .	9
3.5	Примеры. Массивы . . . . .	10
3.6	Задание 1,2 . . . . .	10
3.7	Задание 3 . . . . .	11
3.8	Задание 3 . . . . .	12
3.9	Задание 3 . . . . .	13
3.10	Задание 3 . . . . .	13
3.11	Задание 3 . . . . .	14
3.12	Задание 4 . . . . .	14
3.13	Задание 5,6 . . . . .	15

# 1 Введение

## Цель работы

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

## Задачи

1. Используя Jupyter Lab, повторите примеры из раздела 2.2.
2. Выполните задания для самостоятельной работы (раздел 2.4).

## 2 Теоретическое введение

Julia — высокоуровневый свободный язык программирования с динамической типизацией, созданный для математических вычислений.[1]. 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[2].

## 3 Выполнение лабораторной работы

Выполним примеры из лабораторной работы для действий над кортежами(рис. 3.1)

```
[6] a = [1 2; 3 4]
    b = [1 1]

... 1x2 Matrix{Int64}:
    1 1

[7] a*b'*2

... 2x1 Matrix{Int64}:
    6
   14

[8] x = (a = 2, b = "3")
    println(x.a, x[1])
    print(in("3", x))

... 22
    true

[9] phonebook = Dict{"Иванов И.И." => ("867-5309", "333-5544"), "Бухгалтерия" => "555-2368"}

... Dict{String, Any} with 2 entries:
    "Бухгалтерия" => "555-2368"
    "Иванов И.И." => ("867-5309", "333-5544")

[10] pop!(phonebook, "Иванов И.И.")

... ("867-5309", "333-5544")

[11] phonebook

... Dict{String, Any} with 1 entry:
    "Бухгалтерия" => "555-2368"

[12] a = Dict{"foo" => 0.0, "bar" => 42.0};
    b = Dict{"baz" => 17, "bar" => 13.0};
    print(merge(a, b), merge(b, a))

... Dict{String, Real}{"bar" => 13.0, "baz" => 17, "foo" => 0.0}Dict{String, Real}{"bar" => 42.0, "baz" => 17, "foo" => 0.0}
```

Рис. 3.1: Примеры. Кортежи

Также с множествами(рис. 3.2)

```
S1 = Set([1,2]);
S2 = Set([3,4]);
println("Равенство множеств")
println(issetequal(S1,S2))
S4 = Set([1,2,2,3,1,2,3,2,1]);
S3 = Set([2,3,1])
println("Объединение, пересечение и разность")
### объединение множеств:
C=union(S1,S2)
println(C)
### пересечение множеств:
D = intersect(S1,S3)
println(D)
### разность множеств:
E = setdiff(S3,S1)
println(E)
### проверка вхождения элементов одного множества в другое:
println(issubset(S1,S4))
### добавление элемента в множество:
println("Добавление и уадление элемента")
println(S4)
push!(S4, 99)
println(S4)
### удаление последнего элемента множества:
pop!(S4)
println(S4)
```

[12] ✓ 0.7s

```
... Равенство множеств
false
Объединение, пересечение и разность
Set([4, 2, 3, 1])
Set([2, 1])
Set([3])
true
Добавление и уадление элемента
Set([2, 3, 1])
Set([2, 99, 3, 1])
Set([99, 3, 1])
```

Рис. 3.2: Примеры. Множества

И с массивами(рис. 3.3, 3.4, 3.5)

```
A = rand(4,3,2)

[13]
... 4x3x2 Array{Float64, 3}:
[:, :, 1] =
 0.53715  0.0253307  0.557304
 0.392261  0.613429  0.321627
 0.230467  0.612415  0.786013
 0.975484  0.393498  0.595916

[:, :, 2] =
 0.122025  0.264899  0.171144
 0.965735  0.846615  0.826727
 0.00105212  0.443124  0.609205
 0.560781  0.983995  0.443232

A[1, 2, :]

[14]
... 2-element Vector{Float64}:
 0.025330716121884866
 0.26489877227643766

ar_2=[i^2 for i=1:10 if (i^2%5!=0 && i^2%4!=0)]

[15]
... 4-element Vector{Int64}:
 1
 9
 49
 81

ar = rand(10:20, 10, 5)

[16]
... 10x5 Matrix{Int64}:
 17 17 10 20 14
 12 13 10 18 20
 18 10 18 17 13
 11 11 15 13 10
 16 10 19 13 15
```

Рис. 3.3: Примеры. Массивы



```
[17] ar[[2, 4, 6], [1, 5]]

... 3x2 Matrix{Int64}:
12 20
11 10
18 14

[18] ar[1, 3:end]

... 3-element Vector{Int64}:
10
20
14

[19] sort(ar,dims=1)

... 10x5 Matrix{Int64}:
11 10 10 13 10
11 10 10 13 10
11 11 10 13 13
12 11 13 14 14
12 13 15 17 14
16 14 18 17 15
17 15 19 18 18
18 17 19 19 18
18 17 20 20 19
20 20 20 20 20

[20] sort(ar,dims=2)

... 10x5 Matrix{Int64}:
10 14 17 17 20
10 12 13 18 20
10 13 17 18 18
10 11 11 13 15
10 13 15 16 19
13 14 14 18 20
```

Рис. 3.4: Примеры. Массивы

```

findall(ar .> 14)
[21]
... 26-element Vector{CartesianIndex{2}}:
 CartesianIndex{1, 1}
 CartesianIndex{3, 1}
 CartesianIndex{5, 1}
 CartesianIndex{6, 1}
 CartesianIndex{10, 1}
 CartesianIndex{1, 2}
 CartesianIndex{7, 2}
 CartesianIndex{8, 2}
 CartesianIndex{10, 2}
 CartesianIndex{3, 3}
 CartesianIndex{4, 3}
 CartesianIndex{5, 3}
 CartesianIndex{7, 3}
 CartesianIndex{8, 3}
 CartesianIndex{9, 3}
 CartesianIndex{1, 4}
 CartesianIndex{2, 4}
 CartesianIndex{3, 4}
 CartesianIndex{6, 4}
 CartesianIndex{7, 4}
 CartesianIndex{9, 4}
 CartesianIndex{2, 5}
 CartesianIndex{5, 5}
 CartesianIndex{8, 5}
 CartesianIndex{9, 5}
 CartesianIndex{10, 5}

```

Рис. 3.5: Примеры. Массивы

Выполним задания для самостоятельной работы. Сначала найдем необходимое множество и рассмотрим разные действия над множествами(рис. 3.6)

```

#1,2
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, C), intersect(B, C))
[31] ✓ 0.0s
... Set{Int64} with 6 elements:
 0
 4
 7
 9
 3
 1

setdiff(C,B)
[32] ✓ 0.0s
... Set{Int64} with 4 elements:
 0
 2
 9
 8

```

Рис. 3.6: Задание 1,2

В третьем задании создадим нужные массивы, используя генераторы и циклы(рис. 3.7 - 3.11)

Рис. 3.7: Задание 3

```

№3.10
вектор значений  $y = e^x \cos x$  в точках  $x = 3, 3.1, 3.2, \dots, 6$ , найдите среднее значение  $y$ 

function f(x)
    exp(x)*cos(x)
end
x = 3:0.1:6
y_x = [f(i) for i in x]
sum(y_x)/sizeof(y_x)

6.639218243303714

№3.11
вектор вида  $(x^i, y^j)$ ,  $x = 0.1, i = 3, 6, 9, \dots, 36, y = 0.2, j = 1, 4, 7, \dots, 34$ 

p(3:3:36, 1:3:34) do i, j
    [0.1^i, 0.2^j]
end

12-element Vector{Vector{Float64}}:
 [0.0010000000000000002, 0.2]
 [1.0000000000000004e-6, 0.0016000000000000003]
 [1.0000000000000005e-9, 1.2800000000000006e-5]
 [1.0000000000000008e-12, 1.0240000000000006e-7]
 [1.0000000000000009e-15, 8.192000000000005e-10]
 [1.0000000000000008e-18, 6.553600000000005e-12]
 [1.0000000000000012e-21, 5.2428800000000056e-14]
 [1.0000000000000012e-24, 4.194304000000005e-16]
 [1.0000000000000015e-27, 3.3554432000000044e-18]
 [1.0000000000000017e-30, 2.684354560000004e-20]
 [1.0000000000000018e-33, 2.147483648000004e-22]
 [1.000000000000002e-36, 1.7179869184000035e-24]

first(zip(Vector{[3:3:36]}, Vector{[1:3:34]}))

(3:3:36, 1:3:34)

x1 = [i for i in 3:3:36]
x2 = [j for j in 1:3:34]
[(0.1^i, 0.2^j) for (i,j) in zip(x1,x2)]

12-element Vector{Vector{Float64}}:
 [0.0010000000000000002, 0.2]
 [1.0000000000000004e-6, 0.0016000000000000003]
 [1.0000000000000005e-9, 1.2800000000000006e-5]
 [1.0000000000000008e-12, 1.0240000000000006e-7]
 [1.0000000000000009e-15, 8.192000000000005e-10]
 [1.0000000000000008e-18, 6.553600000000005e-12]
 [1.0000000000000012e-21, 5.2428800000000056e-14]
 [1.0000000000000012e-24, 4.194304000000005e-16]
 [1.0000000000000015e-27, 3.3554432000000044e-18]
 [1.0000000000000017e-30, 2.684354560000004e-20]
 [1.0000000000000018e-33, 2.147483648000004e-22]
 [1.000000000000002e-36, 1.7179869184000035e-24]

```

Рис. 3.8: Задание 3

```

№ 3.12
вектор с элементами  $\frac{1}{i}$ ,  $i = 1, 2, \dots, M, M = 25$ .

[1] 1/1 for i in 1:25

25-element Vector{Float64}:
 1.0
 0.5
 0.3333333333333333
 0.25
 0.2
 0.16666666666666666
 0.14285714285714285
 0.125
 0.1111111111111111
 0.1
 0.09090909090909091
 0.08333333333333333
 0.07692307692307692
 0.07142857142857142
 0.06666666666666667
 0.0625
 0.05882352941176471
 0.05555555555555556
 0.05263157894736842
 0.05
 0.04761904761904762
 0.04545454545454546
 0.04347826086956522
 0.04166666666666667
 0.04
 0.03846153846153846
 0.03636363636363637
 0.03448275862068966
 0.03277777777777778

№ 3.13
вектор вида "f0a1", "f0a2", ..., "f0aN", N = 30

ans3[1] = 1
# 1x1 matrix
push!(ans3[1], "f0a1" * rand(1))
end
print(ans3[1])

Any{f0a1, "f0a2", "f0a3", "f0a4", "f0a5", "f0a6", "f0a7", "f0a8", "f0a9", "f0a10", "f0a11", "f0a12", "f0a13", "f0a14", "f0a15", "f0a16", "f0a17", "f0a18", "f0a19", "f0a20", "f0a21", "f0a22", "f0a23", "f0a24", "f0a25", "f0a26", "f0a27", "f0a28", "f0a29", "f0a30"}

```

Рис. 3.9: Задание 3

```

№ 3.14
векторы  $x = (x_1, x_2, \dots, x_n)$  и  $y = (y_1, y_2, \dots, y_n)$  целочисленного типа длины  $n = 250$  как случайные выборки из совокупности  $0, 1, \dots, 999$ , на его основе:

• сформируйте вектор  $(y_1 - x_1, \dots, y_n - x_n)$ ;
• сформируйте вектор  $(x_1 + 2x_2 - x_3, x_2 + 2x_3 - x_4, \dots, x_{n-1} - 2 + 2x_{n-1} - x_n)$ ;
• сформируйте вектор  $(\frac{\sin y_1}{\cos x_2}, \frac{\sin y_2}{\cos x_3}, \dots, \frac{\sin y_{n-1}}{\cos x_n})$ ;
• вычислите  $\sum_{i=1}^{n-1} \frac{e^{-x_{i+1}}}{x_i + 10}$ ;
• выберите элементы вектора  $y$ , значения которых больше 600, и выведите на экран, определите индексы этих элементов;
• определите значения вектора  $x$ , соответствующие значениям вектора  $y$ , значения которых больше 600 (под соответствием понимается расположение на аналогичных индексах позиций);
• сформируйте вектор  $(|x_1 - x_1^2|, |x_2 - x_1^2|, \dots, |x_n - x_1^2|)$ , где  $x$  обозначает среднее значение вектора  $x = (x_1, x_2, \dots, x_n)$ ;
• определите, сколько элементов вектора  $y$  отстоит от максимального значения не более, чем на 200;
• определите, сколько чётных и нечётных элементов вектора  $x$ ;
• определите, сколько элементов вектора  $x$  кратны 7;
• отсортируйте элементы вектора  $x$  в порядке возрастания элементов вектора  $y$ ;
• выведите элементы вектора  $x$ , которые входят в десятку наибольших (top-10)?
• сформируйте вектор, содержащий только уникальные (неповторяющиеся) элементы вектора  $x$ .

n = 250
# [rand(0:999) for i=1:n]
y = [rand(0:999) for i=1:n]
using Statistics

[1]

top1 = [y[i+1]-x[i] for i=1:n-1];
#p2 = [x[i]+2*x[i+1]-x[i+2] for i=1:n-2];
top3 = [sin(y[i])/cos(x[i+1]) for i=1:n-1];
top4 = sum([exp(-x[i+1])/x[i+10] for i=1:n-1]);
top5_ind = findall(y.>600);
top5_n = filter(x->mean(y);
top6_1 = [x[i] for i=1:250 if (y[i]>600)];
top6_2 = [x[i] for i in top5_ind];
mean_x = mean(x);
top7 = [abs(x[i]-mean(x)) for i=1:n];
top8 = sum(x.*maximum(x)/200);
top9_odd = sum(map(isodd, x));
top9_even = sum(map(iseven, x));
top10 = x[sortperm(y)];
top11 = last(sort(x),10);
top12 = collect(Set(x));

```

Рис. 3.10: Задание 3

```

n = 250
x = [rand(0.999) for i=1:n]
y = [rand(0.999) for i=1:n]
using Statistics

[10] ✓ Ctrl+Enter

D
tmp1 = [y[i+1]-x[i] for i=1:n-1];
tmp2 = [x[i]+2*y[i+1]-x[i+2] for i=1:n-2];
tmp3 = [x[i+1]/cos(x[i+1]) for i=1:n-1];
tmp4 = sum(exp.-x[i+1])/x[i+10] for i=1:n-1];
tmp5_ind = findall(x.-0.001);
tmp5_val = [filter(x.-0.001, y);
            [x[i] for i=1:250 if y[i]!=0.001]];
tmp6_1 = [x[i] for i in tmp5_ind];
tmp6_2 = [x[i] for i in tmp5_ind];
mean_x = mean(x);
tmp7 = [abs(x[i]-mean(x)) for i=1:n];
tmp8 = sum(x.-abs(mean(x)-200));
tmp9_odd = sum(tmp1:odd:n, :);
tmp9_even = sum(tmp1:even:n, :);
tmp10 = [sortperm(tmp1)];
tmp11 = last(sort(x), 10);
tmp12 = collectSet(x);
println(tmp1, "\n", tmp2, "\n", tmp3, "\n", tmp4, "\n", tmp5_ind, "\n", tmp5_val);
println(tmp6_1, "\n", tmp6_2, "\n", mean_x, "\n", tmp7, "\n", tmp8, "\n", tmp9_odd, "\n", tmp9_even, "\n", tmp10, "\n", tmp11, "\n", tmp12)

[10] ✓ Ctrl+Enter

[112, 461, 279, 749, 906, 715, 649, 12, 562, 825, 715, 299, 588, 488, 696, 443, 68, 262, 387, 179, 49, 15, 527, 512, 721, -4, 183, 135, 210, 117, 432, 142, 86, 537, 239, 244, 159, -2, 0,
-4, 272, 1949, 1079, 633, 1147, 1876, 862, 1719, 318, 1697, 1324, 1168, 921, 1795, 1196, 1649, 386, 1264, 1266, 1859, 1664, 1916, 2652, 527, 471, 1372, 1482, 236, 1423, 1487, 1834, 1623, 1529, 1914, 1752, 13
1, 6294848846161771, -2.66255236423266, 1.958194294970534, -0.799929812592875, -2.637913861130003, 0.691384184119392, 6.686399998130587, -0.6427366387819793, -1.264691846742813, 1.8127856888460278, 0,
3.61469092506735e-9
[1, 4, 5, 6, 15, 17, 18, 22, 23, 24, 26, 29, 31, 33, 34, 35, 38, 39, 47, 55, 62, 64, 66, 76, 71, 82, 83, 84, 92, 93, 98, 99, 104, 105, 108, 108, 111, 112, 117, 118, 122, 125, 131, 131, 134, 134, 139, 140, 1,
184, 186, 184, 984, 889, 935, 832, 748, 939, 837, 815, 674, 988, 896, 946, 439, 626, 968, 723, 802, 878, 924, 969, 899, 932, 766, 734, 948, 665, 748, 769, 636, 717, 819, 857, 636, 723, 832, 712, 699, 631,
1383, 98, 84, 881, 627, 682, 247, 824, 844, 896, 537, 283, 598, 889, 795, 950, 781, 724, 383, 278, 525, 252, 394, 811, 766, 538, 287, 862, 924, 788, 412, 813, 872, 758, 725, 259, 390, 387, 937, 645, 251, 46
1293, 99, 44, 893, 627, 682, 247, 824, 844, 896, 537, 283, 598, 889, 795, 950, 781, 724, 383, 278, 525, 252, 394, 811, 766, 538, 287, 862, 924, 788, 412, 813, 872, 758, 725, 259, 390, 387, 937, 645, 251, 46
514, 268
[151.28799999999997, 442.28799999999997, 464.79200000000003, 424.28799999999997, 438.28799999999997, 376.79200000000003, 284.79200000000003, 61.28799999999997, 248.79200000000003, 254.28799999999997, 439.79
53
186
144
427, 398, 299, 318, 537, 688, 569, 884, 927, 246, 686, 963, 927, 74, 317, 82, 579, 689, 453, 642, 109, 979, 685, 954, 669, 243, 628, 167, 864, 487, 989, 285, 872, 38, 311, 453, 96, 945, 437, 366, 658, 322,
1398, 472, 878, 979, 382, 983, 984, 989, 991, 994]
1402, 875, 959, 689, 68, 719, 724, 561, 234, 983, 871, 687, 251, 164, 547, 394, 881, 80, 763, 525, 485, 86, 168, 993, 287, 422, 658, 318, 551, 328, 692, 177, 410, 669, 739, 264, 945, 539, 478, 487, 82, 838,
]

```

Рис. 3.11: Задание 3

В 4 задании создадим массив квадратов чисел от 0 до 100(рис. 3.12)

```

#4
Создайте массив чисел, в котором будут храниться квадраты всех целых чисел от 1 до 100.

D
squares = [i^2 for i = 1:100]

[10] ✓ Ctrl+Enter

11-element Vector{Vector{Int64}}:
 [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
 [100, 121, 144, 169, 196, 225, 256, 289, 324, 361]
 [400, 441, 484, 529, 576, 625, 676, 729, 784, 841]
 [900, 961, 1024, 1089, 1156, 1225, 1296, 1369, 1444, 1521]
 [1600, 1681, 1764, 1849, 1936, 2025, 2116, 2209, 2304, 2401]
 [2500, 2601, 2704, 2809, 2916, 3025, 3136, 3249, 3364, 3481]
 [3600, 3721, 3844, 3969, 4096, 4225, 4356, 4489, 4624, 4761]
 [4900, 5041, 5184, 5329, 5476, 5625, 5776, 5929, 6084, 6241]
 [6400, 6561, 6724, 6889, 7056, 7225, 7396, 7569, 7744, 7921]
 [8100, 8281, 8464, 8649, 8836, 9025, 9216, 9409, 9604, 9801]
 [10000, 10201, 10404, 10609, 10816, 11025, 11236, 11449, 11664, 11881]

#5
Подключите пакет Primes (функция для вычисления простых чисел). Сгенерируйте массив турпimes, в котором будут храниться первые 168 простых чисел. Определите 89-е наименьшее простое число. Получите сред
массива с 89-го до 99-го элемента включительно, содержащий наименьшие простые числа.

import Primes as pm

[10]

primes = [pm.prime(i) for i=1:168];
line 89 = pm.prime(89);
primes[89:100] = primes[89:99];

[10]

```

Рис. 3.12: Задание 4

В 5 рассмотрим возможности пакета простых чисел, а в 6 вычислим сложные математические функции с помощью генераторов массивов и и функции суммирования(рис. 3.13)

```

import Primes as pm

(1)  ✓ on.

D
    myprimes = [pm.prime(i) for i in 100]
    prime_89 = pm.prime(89)
    prime_89_to_99 = myprimes[89:99]
    print(myprimes, "\n", prime_89, "\n", prime_89_to_99)

(2)  ✓ on.

[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 401, 403, 407, 409, 421, 431, 439, 443, 449, 457, 461, 467, 479, 487, 491, 499, 503, 509, 521, 523]

Ne
Вычислите выражения

    tmp1 = sum(i**4*i**2 for i in 10:100)
    tmp2 = sum(2**i/i**3*i**1/2 for i in 1:25)
    tmp3 = 7
    print(tmp1, "\n", tmp2)
    for i in 2:10
        tmp=i*(i+1)
        tmp3=tmp
    end
    print(tmp, "\n", tmp3)

(3)  ✓ on.

20053725
2.2291784308133802e9
0.1294080334474402
6.976366117897618

```

Рис. 3.13: Задание 5,6

## 4 Выводы

В результате выполнения работы изучили несколько структур данных, реализованных в Julia, научились применять их и операции над ними для решения задач.



## Список литературы

1. JuliaLang [Электронный ресурс]. 2024 JuliaLang.org contributors. URL: <https://julialang.org/> (дата обращения: 11.10.2024).
2. Julia 1.11 Documentation [Электронный ресурс]. 2024 JuliaLang.org contributors. URL: <https://docs.julialang.org/en/v1/> (дата обращения: 11.10.2024).