

Лабораторная работа №2

Компьютерный практикум по статистическому анализу данных

Амуничников Антон Игоревич

2025-09-27

1. Информация

2. Вводная часть

3. Выполнение лабораторной работы

1. Информация

1.1 Докладчик

- Амуничников Антон Игоревич
- Группа: НПИбд-01-22
- Российский университет дружбы народов им. П. Лумумбы
- 1132227133@pfur.ru

2. Вводная часть

2.1 Цель работы

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

2.2 Задание

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

3. Выполнение лабораторной работы

3.1 Кортеж

```
[1]: # пустой кортеж:
[1]: ()

[2]: # кортеж из элементов типа String:
      favoritelang = ('Python', 'Julia', 'R')
[2]: ('Python', 'Julia', 'R')

[3]: # кортеж из целых чисел:
      x1 = (1, 2, 3)
[3]: (1, 2, 3)

[4]: # кортеж из элементов разных типов:
      x2 = (1, 2.0, "tmp")
[4]: (1, 2.0, "tmp")

[5]: # именованный кортеж:
      x3 = (a=1, b=1+2)
[5]: (a = 1, b = 3)

[6]: # длина кортежа x2:
      length(x2)
[6]: 3

[7]: # обратиться к элементам кортежа x2:
      x2[1], x2[2], x2[3]
[7]: (1, 2.0, "tmp")

[8]: # произведем какую-либо операцию (сложение)
      # с вторым и третьим элементами кортежа x1:
      c = x1[1] + x1[2]
[8]: 5

[9]: # обращаем к элементу именованного кортежа x3:
      x3.a, x3.b, x3.c
[9]: (2, 3, 3)

[10]: # проверка наличия элементов tmp и 0 в кортеж x2
      # (два способа обращения к методу in()):
      in("tmp", x2), 0 in x2
[10]: (true, false)
```

Рисунок 1: Примеры использования кортежей

3.2 Словарь

```

[12]: # создать словарь с именем phonebook:
phonebook = Dict{"Иванов И.И." => ("867-5309", "333-5544"), "Бухантерия" => "555-2368"}

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

[13]: # вывести ключи словаря:
keys(phonebook)

[13]: KeySet for a Dict{String, Any} with 2 entries. Keys:
      "Бухантерия"
      "Иванов И.И."

[14]: # вывести значения элементов словаря:
values(phonebook)

[14]: ValueIterator for a Dict{String, Any} with 2 entries. Values:
      "555-2368"
      ("867-5309", "333-5544")

[15]: # вывести заданные в словаре пары "ключ - значение":
pairs(phonebook)

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

[16]: # проверка вхождения ключа в словарь:
haskey(phonebook, "Иванов И.И.")

[16]: true

[17]: # добавить элемент в словарь:
phonebook["Сидоров П.С."] = "555-3344"

[17]: "555-3344"

[19]: # удалить ключ и связанное с ним значение из словаря
pop!(phonebook, "Иванов И.И.")

[19]: ("867-5309", "333-5544")

[20]: # Объединение словарей (функция merge()):
a = Dict{"foo" => 0.0, "bar" => 42.0};
b = Dict{"baz" => 17, "bar" => 13.0};
merge(a, b, merge(b,a))

[20]: (Dict{String, Real}{"bar" => 13.0, "baz" => 17, "foo" => 0.0}, Dict{String, Real}{"bar" => 42.0, "baz" => 17, "foo" => 0.0})
```

Рисунок 2: Примеры использования словарей

3.3 Множество

```
[21]: # создать множество из четырёх целочисленных значений:
```

```
A = Set([1, 3, 4, 5])
```

```
[21]: Set(Int64) with 4 elements:
```

```
5
4
3
1
```

```
[22]: # создать множество из 11 символьных значений:
```

```
B = Set('abracadabra')
```

```
[22]: Set(Char) with 5 elements:
```

```
'a'
'd'
'r'
'k'
'b'
```

```
[23]: # проверка эквивалентности двух множеств:
```

```
S1 = Set([1,2]);
S2 = Set([3,4]);
issetequal(S1,S2)
S3 = Set([1,2,2,1,2,3,2,1]);
S4 = Set([2,3,1]);
issetequal(S3,S4)
```

```
[23]: true
```

```
[24]: # проверка эквивалентности двух множеств:
```

```
S1 = Set([1,2]);
S2 = Set([3,4]);
issetequal(S1,S2)
```

```
[24]: false
```

```
[25]: C=union(S1,S2)
```

```
[25]: Set(Int64) with 4 elements:
```

```
4
2
3
1
```

```
[26]: # пересечение множеств:
```

```
D = intersect(S1,S3)
```

```
[26]: Set(Int64) with 2 elements:
```

```
2
1
```

```
[27]: # разность множеств:
```

```
E = setdiff(S3,S1)
```

Рисунок 3: Примеры использования множеств

3.4 Массив

```
[30]: 2
[31]: # создание пустого массива с абстрактным типом:
empty_array_1 = []
[31]: Any[]
[32]: # создание пустого массива с конкретным типом:
empty_array_2 = (Int64[])
empty_array_3 = (Float64[])
[32]: Float64[]
[33]: # вектор-столбец:
a = [1, 2, 3]
[33]: 3-element Vector{Int64}:
 1
 2
 3
[34]: # вектор-строка:
b = [1 2 3]
[34]: 1x3 Matrix{Int64}:
 1 2 3
[35]: # многомерные массивы (матрицы):
A = [[1, 2, 3] [4, 5, 6] [7, 8, 9]]
B = [[1 2 3]; [4 5 6]; [7 8 9]]
[35]: 3x3 Matrix{Int64}:
 1 2 3
 4 5 6
 7 8 9
[36]: # одномерный массив из 8 элементов (массив $1 (times 8))
# со значениями, случайно распределенными на интервале [0, 1]:
c = rand(1,8)
[36]: 1x8 Matrix{Float64}:
 0.869706 0.752429 0.481719 0.36821 ... 0.109415 0.367586 0.00477669
[37]: # многомерный массив $2 (times 3) (2 строки, 3 столбца) элементов
# со значениями, случайно распределенными на интервале [0, 1]:
C = rand(2,3);
[37]: # трехмерный массив:
D = rand(4, 3, 2)
[38]: 4x3x2 Array{Float64, 3}:
[:, :, 1] =
 0.723352 0.80358367 0.720488
 0.237813 0.858544 0.178881
 0.540533 0.888536 0.8286531
[:, :, 2] =
```

Рисунок 4: Примеры использования массивов

3.5 Массив

```
C = rand(1,2)

[36]: 1x8 Matrix{Float64}:
      0.869706  0.752429  0.401719  0.36821  _  0.109415  0.367586  0.00477669

[37]: # многомерный массив {2 times 3} (2 строки, 3 столбца) элементов
      # со значениями, случайно распределёнными на интервале [0, 1]:
      C = rand(2,3);

[38]: # трёхмерный массив:
      D = rand(4, 3, 2)

[38]: 4x3x2 Array{Float64, 3}:
[:, :, 1] =
      0.723352  0.00350367  0.729408
      0.237013  0.858544   0.178081
      0.248532  0.808526   0.0206531
      0.436335  0.981371   0.50473

     [:, :, 2] =
      0.668084  0.015358  0.439154
      0.525835  0.825208  0.063783
      0.847101  0.310807  0.328788
      0.642303  0.974095  0.305427

[39]: # массив из квадратных корней всех целых чисел от 1 до 10:
      roots = [sqrt(i) for i in 1:10]

[39]: 10-element Vector{Float64}:
      1.0
      1.4142135623730951
      1.7320508075688772
      2.0
      2.23606797749979
      2.449489742783178
      2.6457513110645907
      2.8284271247461903
      3.0
      3.1622776601683795
```

Рисунок 5: Примеры использования массивов

3.6 Массив

```
[40]: ar_1 = [3*i^2 for i in 1:2:9]
[40]: 5-element Vector{Int64}:
      3
     27
     75
    147
    243
[41]: # массив квадратов элементов, если квадрат не делится на 5 или 4:
      ar_2=[i^2 for i=1:10 if (i^2%5!=0 && i^2%4!=0)]
[41]: 4-element Vector{Int64}:
      1
      9
     49
     81
[42]: # одномерный массив из пяти единиц:
      ones(5)
[42]: 5-element Vector{Float64}:
      1.0
      1.0
      1.0
      1.0
      1.0
[43]: # двумерный массив 2x3 из единиц:
      ones(2,3)
[43]: 2x3 Matrix{Float64}:
      1.0  1.0  1.0
      1.0  1.0  1.0
[44]: # одномерный массив из 4 нулей:
      zeros(4)
[44]: 4-element Vector{Float64}:
      0.0
      0.0
      0.0
      0.0
[45]: # заполнить массив 3x2 цифрами 3.5
      fill(3.5,(3,2))
[45]: 3x2 Matrix{Float64}:
      3.5  3.5
      3.5  3.5
      3.5  3.5
```

3.7 Массив

```
[48]: # транспонирование
      b'
```

```
[48]: 6x2 adjoint(::Matrix{Int64}) with eltype Int64:
      1  2
      3  4
      5  6
      7  8
      9 10
     11 12
```

```
[49]: # транспонирование
      c = transpose(b)
```

```
[49]: 6x2 transpose(::Matrix{Int64}) with eltype Int64:
      1  2
      3  4
      5  6
      7  8
      9 10
     11 12
```

```
[50]: # массив 10x5 целых чисел в диапазоне [10, 20]:
      ar = rand(10:20, 10, 5)
```

```
[50]: 10x5 Matrix{Int64}:
      12 14 14 12 12
      16 12 16 13 19
      17 15 20 12 16
      19 16 17 15 20
      12 13 20 20 10
      20 19 15 18 19
      20 13 10 11 14
      19 19 13 14 15
      15 13 17 17 20
      14 18 14 11 20
```

```
[51]: # выбор всех значений строки в столбце 2:
      ar[:, 2]
```

```
[51]: 10-element Vector{Int64}:
      14
      12
      15
      16
      13
      19
      13
      19
      13
      18
```

3.8 Массив

```
15 18 19 19 20
10 11 13 14 20
13 14 15 19 19
13 15 17 17 20
11 14 14 18 20

[58]: # поэлементное сравнение с числом
# (результат - массив логических значений):
ar .> 14

[58]: 10x5 BitMatrix:
0 0 0 0 0
1 0 1 0 1
1 1 1 0 1
1 1 1 1 1
0 0 1 1 0
1 1 1 1 1
1 0 0 0 0
1 1 0 0 1
1 0 1 1 1
0 1 0 0 1

[59]: # возврат индексов элементов массива, удовлетворяющих условию:
findall(ar .> 14)

[59]: 29-element Vector{CartesianIndex{2}}:
 CartesianIndex{2, 1}
 CartesianIndex{3, 1}
 CartesianIndex{4, 1}
 CartesianIndex{6, 1}
 CartesianIndex{7, 1}
 CartesianIndex{8, 1}
 CartesianIndex{9, 1}
 CartesianIndex{3, 2}
 CartesianIndex{4, 2}
 CartesianIndex{6, 2}
 CartesianIndex{8, 2}
 CartesianIndex{10, 2}
 CartesianIndex{2, 3}
 ⋮
 CartesianIndex{9, 3}
 CartesianIndex{4, 4}
 CartesianIndex{5, 4}
 CartesianIndex{6, 4}
 CartesianIndex{9, 4}
 CartesianIndex{2, 5}
 CartesianIndex{3, 5}
 CartesianIndex{4, 5}
 CartesianIndex{6, 5}
 CartesianIndex{8, 5}
 CartesianIndex{9, 5}
 CartesianIndex{10, 5}
```


3.9 Задание №1

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

```
[60]: #Задание 1
      A = Set([0,3,4,9])
      B = Set([1,3,4,7])
      C = Set([0,1,2,4,7,8,9])

      union(intersect(A,B), intersect(A,C), intersect(B,C))

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

Рисунок 9: Задание №1. Работа с множествами

3.10 Задание №2

```
[61]: #Задание 2
      A = Set(["Иванов", "Петров", "Сидоров"])
      B = Set(["Иванов", "Пушкин"])#объединение
      print(union(A,B))

      Set(["Иванов", "Петров", "Пушкин", "Сидоров"])

[62]: Set1 = Set([1, 2, 3, "Hello"])

[62]: Set{Any} with 4 elements:
      2
      "Hello"
      3
      1

[63]: println("\nElements of set:")
      for i in Set1
          println(i)
      end

      Elements of set:
      2
      Hello
      3
      1

[64]: print(in("Hello", Set1))

      true

[66]: Set1 = push!(Set1, "World")
      println("\nSet after adding one element: \n", Set1)

      Set after adding one element:
      Set{Any{2, "Hello", "World", 3, 1}}

[68]: for i in 1:5
      push!(Set1, i)
      end
      println("\nSet after adding range of elements:\n", Set1)

      Set after adding range of elements:
```

3.11 Задание №3

Рисунок 11: Задание №3. Работа с массивами

3.12 Задание №3

```
[86]: S8 = vcat(fill(2^tmp[1], 1), fill(2^tmp[2], 1), fill(2^tmp[3], 4))
      count_6 = count(x -> x == 6, S8)
      print(S8, '\n', count_6)

      [16, 64, 8, 8, 8, 8]
      8

[87]: # 3.10
      using Statistics
      f(x) = exp(x)*cos(x) #заменяю y на f так как такая функция уже создавалась
      Y = [f(x) for x in 3:0.1:6]
      mean(Y)

[87]: 53.11374594642971

[88]: # 3.11
      x_vals = [0.1^i for i in 3:3:36]
      y_vals = [0.2^j for j in 1:3:34]
      arr11 = vcat(x_vals, y_vals)

[88]: 24-element Vector{Float64}:
      0.0010000000000000002
      1.0000000000000004e-6
      1.0000000000000005e-9
      1.0000000000000006e-12
      1.0000000000000009e-15
      1.000000000000001e-18
      1.0000000000000012e-21
      1.0000000000000014e-24
      1.0000000000000015e-27
      1.0000000000000017e-30
      1.0000000000000018e-33
      1.000000000000002e-36
      0.2
      0.0016000000000000003
      1.2800000000000005e-5
      1.0240000000000006e-7
      8.192000000000005e-10
      6.553600000000005e-12
      5.2428800000000056e-14
      4.194304000000005e-16
      3.3554432000000048e-18
      2.684354560000004e-20
      2.1474836480000035e-22
      1.717986918400003e-24
```

3.13 Задание №3

```
4.776009184000000e-24
P9: 2 3 12
M = 25
arr12 = [2^i / i for i in 1:M]

P9: 25-element Vector{Float64}:
 2.0
 2.0
 2.6666666666666665
 4.0
 8.4
19.866666666666666
18.285714285714285
32.0
56.888888888888886
102.4
108.1818181818182
543.3333333333333
638.1538461538462
1170.2857142857142
2184.5233333333333
4096.0
7718.117647058823
14560.555555555555
27594.165263157893
52428.8
98864.38895238895
198058.18181818182
364722.8889565217
699058.8888888889
1.34217728e6

P9: 2 3 12
M = 30
arr13 = ["fn1" for i in 1:M]

P9: 30-element Vector{String}:
 "fn1"
 "fn2"
 "fn3"
 "fn4"
 "fn5"
 "fn6"
 "fn7"
 "fn8"
 "fn9"
 "fn10"
 "fn11"
 "fn12"
 "fn13"
  ⋮
 "fn19"
 "fn20"
 "fn21"
 "fn22"
 "fn23"
 "fn24"
```

Рисунок 13: Задание №3. Работа с массивами

3.14 Задание №3

```
[102]: # Задание №3. Работа с массивами
def f1(x):
    return x**2
arr1 = np.arange(100)
arr2 = arr1**2

[103]: # Задание №3. Работа с массивами
def f2(x):
    return x**2
arr1 = np.arange(100)
arr2 = arr1**2

[104]: # Задание №3. Работа с массивами
def f3(x):
    return x**2
arr1 = np.arange(100)
arr2 = arr1**2

[105]: # Задание №3. Работа с массивами
def f4(x):
    return x**2
arr1 = np.arange(100)
arr2 = arr1**2
```

Рисунок 14: Задание №3. Работа с массивами

3.15 Задание №3

[illegible]

Рисунок 15: Задание №3. Работа с массивами

3.16 Задание №3

```
UnboundError: 'vect_y' not defined in 'Main'
Suggestion: check for spelling errors or missing imports.

Stacktrace:
[1] top-level scope

[110]: # Значения y в пределах 200 от максимума
      y_max = maximum(y)
      count_near_max = count(y .-> y_max - 200)

[110]: 50

[109]: # Найти k-неблизкие k x
      even_count = count(iseven, x)
      odd_count = count(isodd, x)

[109]: 112

[111]: # Найти k-неблизкие k x
      even_count = count(iseven, x)

[111]: 136

[108]: # Найти 7
      multiple_7 = count(x -> x % 7 == 0, x)

[108]: 39

[107]: # Сортировка x по y
      sorted_x_by_y = x[sortperm(y)]

[107]: 250-element Vector{Int64}:
      886
      855
      317
      294
      548
      478
      228
      653
      787
      283
      529
      378
      642
      |
      179
      184
      967
      601
      873
      736
      481
      532
      484
      457
      418
      820
```

Рисунок 16: Задание №3. Работа с массивами

3.17 Задание №3

```
[112]: # Top-10 x
top10_x = partialsort(x, 1:10, reverse)

[112]: 10-element view(::Vector{Int64}, 1:10) with eltype Int64:
 997
 994
 993
 991
 989
 976
 975
 971
 970
 967

[113]: # Уникальные x
unique_x = unique(x)

[113]: 222-element Vector{Int64}:
 283
 955
 717
 684
 554
 911
 623
 157
 928
 842
 458
 233
 418
  :
 629
 688
 953
 730
 993
 831
 588
 759
 743
 820
 964
 368
```

Рисунок 17: Задание №3. Работа с массивами

3.18 Задание №4

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

```
[115]: # Square of  
squares = [i**2 for i in 1:100]  
[115]: 100-element Vector{Int64}:  
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
```

Рисунок 18: Задание №4

3.19 Задание №5

```
[119]: import Pkg
      Pkg.add("Primes")

      #adding registry at "~/julia/registries/General.toml"
      #adding package versions...
      #adding IntegerMathUtils - v0.1.3
      #adding Primes - v0.5.7
      #adding "~/julia/environments/v1.11/Project.toml"
      #adding "~/julia/environments/v1.11/Manifest.toml"
      #adding project...
      499.5 ms
      578.7 ms Primes
      2 dependencies successfully precompiled in 2 seconds. 39 already precompiled.

[120]: #jupyter 5
      using Primes

      myprime = primes(1000)[1:100]
      print(myprime, "\n")
      print(myprime[50], "\n")
      slice = myprime[89:99]
      println(slice)

[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,
223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 46
1, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631, 641, 643, 647, 653, 659, 661, 673, 677, 683, 691, 701, 709, 719, 727, 733,
739, 743, 751, 757, 761, 769, 773, 787, 797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859, 863, 877, 881, 883, 887, 907, 911, 919, 929, 937, 941, 947, 953, 967, 971, 977, 983, 991, 997]
403
[461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523]
```

Рисунок 19: Задание №5. Работа с пакетом Primes

3.20 Задание №6

Вычислим следующие выражения.

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

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

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

3.21 Задание №6

```
[121]: #Задание 6  
  
#6.1  
sum1 = sum(i^3+4i^2 for i in 10:100)
```

```
[121]: 26852735
```

```
[122]: #6.2  
M = 25  
sum2 = sum(2i^i+3i^i^2 for i in 1:M)
```

```
[122]: 28971841895017365
```

```
[123]: #6.3  
M = 38  
sum3 = 1  
x_m = 1  
  
for i in 2:2:M  
    x_m += i/(i+1)  
    sum3 += x_m  
end  
print(sum3)  
  
189.1666991875271
```

Рисунок 20: Задание №6

3.22 Выводы

В результате выполнения данной лабораторной работы я изучил несколько структур данных, реализованных в Julia, Научился применять их и операции