

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1

*дисциплина: Компьютерный практикум по статистическому
анализу данных*

Студент: Мухамедияр Адиль

Группа: НКНбд-01-20

МОСКВА

2023г.

Постановка задачи

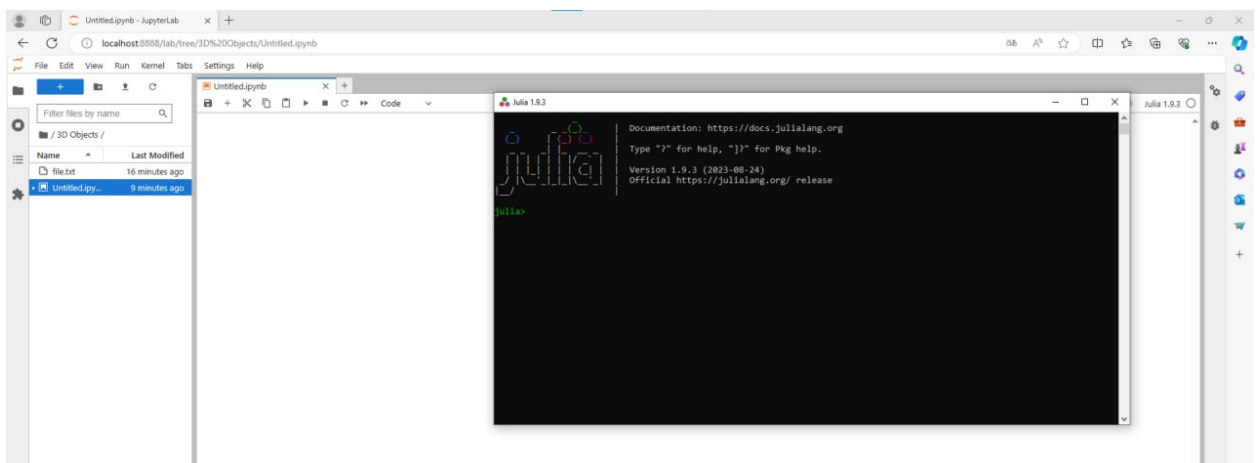
1. Установите под свою операционную систему Julia, Jupyter.
2. Используя Jupyter Lab, повторите примеры из раздела.
3. Выполните задания для самостоятельной работы.

Задача для самостоятельной работы.

1. Изучите документацию по основным функциям Julia для чтения / записи / вывода информации на экран: `read()`, `readline()`, `readlines()`, `readln()`, `print()`, `println()`, `show()`, `write()`. Приведите свои примеры их использования, поясняя особенности их применения.
2. Изучите документацию по функции `parse()`. Приведите свои примеры её использования, поясняя особенности её применения.
3. Изучите синтаксис Julia для базовых математических операций с разным типом переменных: сложение, вычитание, умножение, деление, возведение в степень, извлечение корня, сравнение, логические операции. Приведите свои примеры с пояснениями по особенностям их применения.
4. Приведите несколько своих примеров с пояснениями с операциями над матрицами и векторами: сложение, вычитание, скалярное произведение, транспонирование, умножение на скаляр.

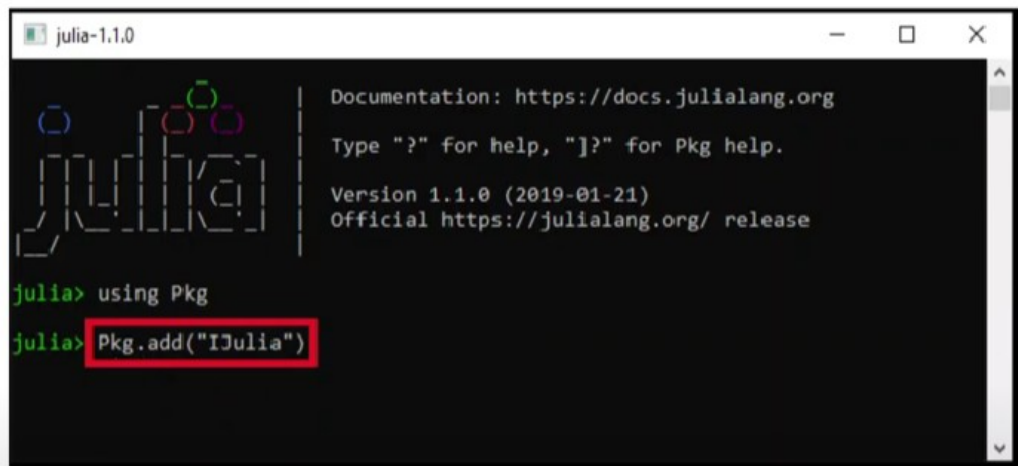
Выполнение работы

- 1.1. Установил под свою операционную систему Julia, Jupyter.



1.2. При добавлении ядра Julia были затруднения, так что решил так же прикрепить этот пункт.

```
Pkg.add("IJulia")
```



2. Используя Jupyter Lab, повторил примеры из раздела.

2.1. Для начало было определено тип числовых величин.

```
[3]: typeof(3), typeof(3.5), typeof(3/3.55), typeof(sqrt(3+4im)), typeof(pi)
```

```
[3]: (Int64, Float64, Float64, ComplexF64, Irrational{::π})
```

```
[4]: 1.0/0.0, 1.0/(-0.0), 0.0/0.0
```

```
[4]: (Inf, -Inf, NaN)
```

```
[5]: typeof(1.0/0.0), typeof(1.0/(-0.0)), typeof(0.0/0.0)
```

```
[5]: (Float64, Float64, Float64)
```

```
[9]: for T in [Int8,Int16,Int32,Int64,Int128,UInt8,UInt16,UInt32,UInt64,UInt128]
println("${lpad(T,7)}: [$(typemin(T)),$(typemax(T))]" )
end
```

```
Int8: [-128,127]
Int16: [-32768,32767]
Int32: [-2147483648,2147483647]
Int64: [-9223372036854775808,9223372036854775807]
Int128: [-170141183460469231731687303715884105728,170141183460469231731687303715884105727]
UInt8: [0,255]
UInt16: [0,65535]
UInt32: [0,4294967295]
UInt64: [0,18446744073709551615]
UInt128: [0,340282366920938463463374607431768211455]
```

2.2. Далее приводим аргументы к одному типу.

```
[10]: Int64(2.0), Char(2), typeof(Char(2))  
[10]: (2, '\x02', Char)  
[11]: convert(Int64,2.0), convert(Char,2)  
[11]: (2, '\x02')  
[12]: typeof(promote(Int8(1), Float16(4.5), Float32(4.1)))  
[12]: Tuple{Float32, Float32, Float32}
```

2.3. Ниже мы пробуем определить значения функций при определенных значениях переменных.

```
[13]: function f(x)  
      x^2  
      end  
[13]: f (generic function with 1 method)  
[14]: f(4)  
[14]: 16  
[15]: g(x)=x^2  
[15]: g (generic function with 1 method)  
[16]: g(8)  
[16]: 64
```

2.4. И наконец, изучаем примеры работы с массивами.

```

[18]: a = [4 7 6] # вектор-строка
      b = [1, 2, 3] # вектор-столбец
      a[2], b[2] # вторые элементы векторов a и b соответственно

[18]: (7, 2)

[19]: a = 1; b = 2; c = 3; d = 4 # присвоение значений
      Am = [a b; c d] # матрица 2 x 2

[19]: 2x2 Matrix{Int64}:
      1  2
      3  4

[20]: Am[1,1], Am[1,2], Am[2,1], Am[2,2] # элементы матрицы Am

[20]: (1, 2, 3, 4)

[21]: aa = [1 2] # вектор-строка
      AA = [1 2; 3 4] # матрица 2 x 2
      aa*AA*aa' # умножение вектора-строки на матрицу и на вектор-столбец (операция транспонирования)

[21]: 1x1 Matrix{Int64}:
      27

[23]: aa, AA, aa'

[23]: ([1 2], [1 2; 3 4], [1; 2;:])

```

3. Выполнил задания для самостоятельной работы.

3.1. В данном пункте были разобраны следующие функции:

Если **read()** считывает все данные из файла или потока, то **readline()** читает лишь одну строку. А функция **readlines()** так же считывает все строки, но возвращает их в виде массива строк. **readdlm()** уже читает данные из файла, разделенные определенным разделителем(например, запятой или табуляцией), и возвращает массив. Если функция **print()** выводит данные без перевода строки, то функция **println()** может заняться добавлением перевода строки. Функция **show()** предназначена для вывода объектов в интерактивных сессиях. **Write()** записывает данные на файл или поток.


```
[45]: # Пункт 1
      read("file.txt", String) # считывает содержимое файла file.txt как строку

[45]: "Первая строка\nВторая строка\nТретья строка\nПоследняя строка(полный файл)"

[46]: readline("file.txt") # считывает первую строку из файла file.txt

[46]: "Первая строка"

[47]: readlines("file.txt") # считывает все строки из файла file.txt

[47]: 4-element Vector{String}:
      "Первая строка"
      "Вторая строка"
      "Третья строка"
      "Последняя строка(полный файл)"

[ ]: readlm("data.csv", ',') # считывает CSV-файл с разделителем-запятой

[50]: print("Hello, ")
      println("world!")
      # выводят "Hello, world!" с переводом строки после "world!"

      Hello, world!

[51]: show([1, 2, 3]) # отображает массив [1, 2, 3]

      [1, 2, 3]

[52]: write("file.txt", "Hello World") # записывает строку "Hello World" в файл file.txt

[52]: 11
```

3.2. Функция **parse()** преобразует строку в значение указанного типа.

```
[53]: # Пункт 2
      parse{Int, "123"} # преобразует строку "123" в целое число 123

[53]: 123
```

3.3. Изучение базовых переменных в Julia.

```
[59]: # Пункт3
      # Целочисленные переменные
      a = 5
      b = 2

      # Операции
      addition = a + b      # Сложение: 7
      subtraction = a - b  # Вычитание: 3
      multiplication = a * b # Умножение: 10
      division = a / b      # Деление: 2.5
```

[59]: 2.5

```
[61]: # Возведение в степень
      power = a ^ b      # 5 ^ 2 = 25
```

[61]: 25

```
[62]: # Извлечение корня
      sqrt_b = sqrt(b)   # Квадратный корень из 2 ≈ 1.41421
```

[62]: 1.4142135623730951

```
[66]: # Равенство
      is_equal = (a == b) # false

      # Неравенство
      is_not_equal = (a != b) # true

      # Больше, меньше
      is_greater = (a > b) # true
      is_less = (a < b)   # false
```

[66]: false

```
[67]: # Логическое И (AND)
      logical_and = (a > 1) && (b < 3) # true

      # Логическое ИЛИ (OR)
      logical_or = (a < 1) || (b < 3)  # true

      # Логическое НЕ (NOT)
      logical_not = !(a == b)          # true
```

[67]: true

3.4. Были разобраны несколько примеров операций над матрицами и векторами.

```
[68]: # Пункт 4
      # Векторы
      v1 = [1, 2, 3]
      v2 = [4, 5, 6]

      # Матрица
      m1 = [1 2; 3 4]

[68]: 2×2 Matrix{Int64}:
      1 2
      3 4

[69]: # Сложение векторов
      vector_add = v1 + v2 # [5, 7, 9]

      # Сложение матриц
      matrix_add = m1 + m1 # [2 4; 6 8]

      # Вычитание векторов
      vector_sub = v1 - v2 # [-3, -3, -3]

[69]: 3-element Vector{Int64}:
      -3
      -3
      -3

[ ]: # Скалярное произведение векторов
      dot_product = dot(v1, v2) # 32 (1*4 + 2*5 + 3*6)

[72]: # Транспонирование матрицы
      transpose_m1 = transpose(m1) # [1 3; 2 4]

[72]: 2×2 transpose{::Matrix{Int64}} with eltype Int64:
      1 3
      2 4

[73]: # Умножение вектора на скаляр
      scalar_mult_vector = 3 * v1 # [3, 6, 9]

      # Умножение матрицы на скаляр
      scalar_mult_matrix = 2 * m1 # [2 4; 6 8]

[73]: 2×2 Matrix{Int64}:
      2 4
      6 8

[*]: # Умножение матриц
      m2 = [5 6; 7 8]
      matrix_mult = m1 * m2 # Результат - матрица 2x2
```

Выводы

Были изучены основные понятия в Julia и Jupyter. Так же были применены основные операции на числа, матрицы и вектора.

