Отчёт по лабораторной работе №1
Компьютерный практикум по статистическому анализу данных

Julia. Установка и настройка. Основные принципы.

Выполнила: Скандарова Полина Юрьевна,
НПИбд-01-22, 1132221815

Содержание

1 Цель работы

Основная цель работы — подготовить рабочее пространство и инструментарий для работы с языком программирования Julia, на простейших примерах познакомиться с основами синтаксиса Julia.

2 Выполнение лабораторной работы

2.1 Подготовка инструментария к работе

Так как мы используем ОС типа Windows для различных установок будем использовать терминал и уже установленную Julia. Далее посредством терминала установим JupiterLab(рис. 1):

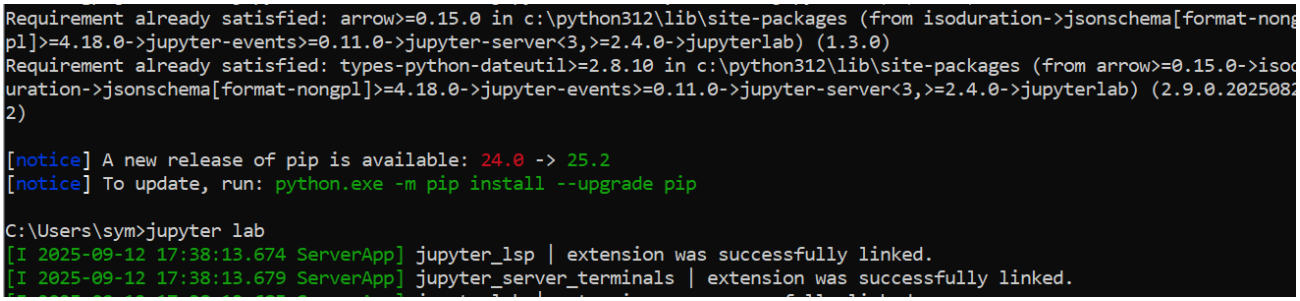

Рис. 1: Установка JupiterLab

Следующим шагом установим пакеты для работы с Jupyter. Для этого перейдём в пакетный режим Julia, нажав на клавиатуре знак закрывающейся квадратной скобки], затем введём add IJulia (рис. 2):

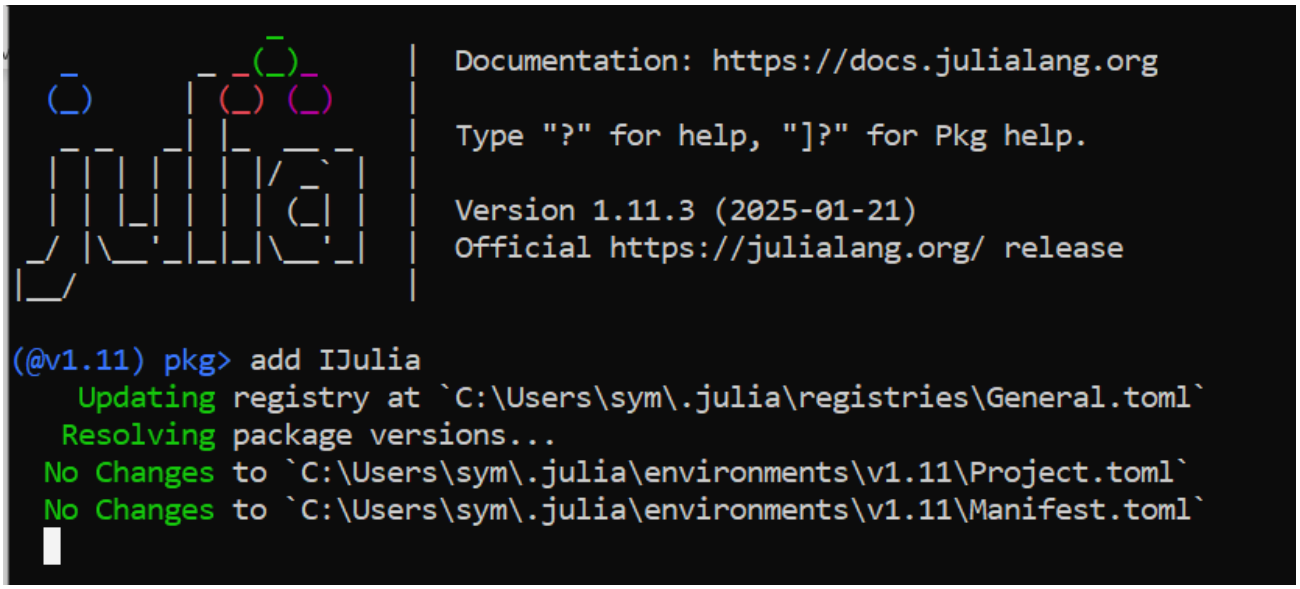

Рис. 2: Установка пакетов для работы с Jupyter

2.2 Основы синтаксиса Julia на примерах

Для начала потренируемся с определением типов числовых величин (рис. 3):

```
[60]: 1+2

[60]: 3

[61]: 3+4
      5+6

[61]: 11

[62]: 7+8
      9+10;

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

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

 [7]: 1.0/0.0, 1.0/(-0.0), 0.0/0.0

 [7]: (Inf, -Inf, NaN)

 [8]: typeof(1.0/0.0), typeof(1.0/(-0.0)), typeof(0.0/0.0)

 [8]: (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]
```

Рис. 3: Примеры определения типа числовых величин

После чего приступим к рассмотрению приведения аргументов к одному типу (рис. 8):

```
[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]: Bool(1), Bool(0)

[12]: (true, false)

[13]: promote(Int8(1), Float16(4.5), Float32(4.1))

[13]: (1.0f0, 4.5f0, 4.1f0)

[14]: typeof(promote(Int8(1), Float16(4.5), Float32(4.1)))

[14]: Tuple{Float32, Float32, Float32}
```

Рис. 4: Примеры приведения аргументов к одному типу

И рассмотрим примеры определения функций (рис. 5), а также работу с массивами (рис. 6):

```
[15]: function f(x)
      x^2
      end
```

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

```
[16]: f(4)
```

```
[16]: 16
```

```
[17]: g(x)=x^2
```

```
[17]: g (generic function with 1 method)
```

```
[18]: g(8)
```

```
[18]: 64
```

Рис. 5: Примеры определения функций

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

```
[19]: (7, 2)
```

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

```
[21]: 2×2 Matrix{Int64}:
       1  2
       3  4
```

```
[22]: Am[1,1], Am[1,2], Am[2,1], Am[2,2] # элементы матрицы
```

```
[22]: (1, 2, 3, 4)
```

```
[23]: aa = [1 2]
      AA = [1 2; 3 4]
      aa*AA*aa'
```

```
[23]: 1×1 Matrix{Int64}:
       27
```

```
[24]: aa, AA, aa'
```

```
[24]: ([1 2], [1 2; 3 4], [1; 2;;])
```

Рис. 6: Примеры работы с массивами

2.3 Самостоятельная работа

В первом задании ма рассмотрим основные функции для чтения / записи / вывода информации на экран. Для этого составим свои примеры (рис. 7):

[55]:
```
print("Просто вывод без переноса строки ")
println("А это вывод с переводом строки")
show("Hello, Julia!") # show() показывает представление объекта
```

```
Просто вывод без переноса строки А это вывод с переводом строки
"Hello, Julia!"
```

[44]:
```
open("example.txt", "w") do f
    write(f, "Это строка, записанная в файл.\n")
    println(f, "А это строка с println в файл.")
end
```

[56]:
```
open("example.txt", "r") do f
    println("read(): ", String(read(f))) # прочитать всё содержимое как строку
end
```

```
read(): Это строка, записанная в файл.
А это строка с println в файл.
```

[57]:
```
open("example.txt", "r") do f
    line = readline(f) # считывает первую строку
    println("readline(): ", line)
end
```

```
readline(): Это строка, записанная в файл.
```

[58]:
```
lines = readlines("example.txt") # считывает все строки в массив
println("readlines(): ", lines)
```

```
readlines(): ["Это строка, записанная в файл.", "А это строка с println в файл."]
```

[59]:
```
using DelimitedFiles
data = readdlm(IOBuffer("1 2 3\n4 5 6")) # читаем из буфера (как из файла)
println("readdlm(): ", data)
```

```
readdlm(): [1.0 2.0 3.0; 4.0 5.0 6.0]
```

Рис. 7: Примеры работы с функциями для чтения/записи/вывода информации на экран

Во втором задании состаивим пример для функции parse() (рис. 8):

[51]:
```
x = parse(Int, "42")         # строка -> целое число
y = parse(Float64, "3.14")   # строка -> число с плавающей точкой
b = parse(Bool, "true")      # строка -> логическое значение

println("parse(Int, \"42\") = ", x)
println("parse(Float64, \"3.14\") = ", y)
println("parse(Bool, \"true\") = ", b)
```

```
parse(Int, "42") = 42
parse(Float64, "3.14") = 3.14
parse(Bool, "true") = true
```

Рис. 8: Пример работы с функцией parse

Далее изучим синтаксис Julia для базовых математических операций с разным типом переменных (рис. 9):

[52]:
```
a = 10
b = 3.0

# Арифметика
println("a + b = ", a + b)
println("a - b = ", a - b)
println("a * b = ", a * b)
println("a / b = ", a / b)         # обычное деление
println("a ÷ 3 = ", a ÷ 3)         # целочисленное деление
println("a % 3 = ", a % 3)         # остаток
println("a ^ 2 = ", a ^ 2)         # возведение в степень
println("sqrt(a) = ", sqrt(a))     # квадратный корень

# Сравнение
println("a > b ? ", a > b)
println("a == 10 ? ", a == 10)

# Логические операции
x, y = true, false
println("x && y = ", x && y)   # логическое И
println("x || y = ", x || y)   # логическое ИЛИ
println("!x = ", !x)           # отрицание
```

```
a + b = 13.0
a - b = 7.0
a * b = 30.0
a / b = 3.3333333333333335
a ÷ 3 = 3
a % 3 = 1
a ^ 2 = 100
sqrt(a) = 3.1622776601683795
a > b ? true
a == 10 ? true
x && y = false
x || y = true
!x = false
```

Рис. 9: Примеры работы базовых математических операций

В конце работы приведём несколько примеров с операциями над матрицами (рис. 10):

[54]:
```julia
using LinearAlgebra

v1 = [1, 2, 3]
v2 = [4, 5, 6]

m1 = [1 2; 3 4]
m2 = [5 6; 7 8]

# Векторные операции
println("v1 + v2 = ", v1 + v2)
println("v1 - v2 = ", v1 - v2)
println("Скалярное произведение v1 · v2 = ", dot(v1, v2))
println("Транспонирование v1' = ", v1')

# Матрицы
println("m1 + m2 = ")
println(m1 + m2)

println("m1 - m2 = ")
println(m1 - m2)

println("Умножение матрицы на скаляр: 2 * m1 = ")
println(2 * m1)

println("Умножение матриц m1 * m2 = ")
println(m1 * m2)
```

```
v1 + v2 = [5, 7, 9]
v1 - v2 = [-3, -3, -3]
Скалярное произведение v1 · v2 = 32
Транспонирование v1' = [1 2 3]
m1 + m2 =
[6 8; 10 12]
m1 - m2 =
[-4 -4; -4 -4]
Умножение матрицы на скаляр: 2 * m1 =
[2 4; 6 8]
Умножение матриц m1 * m2 =
[19 22; 43 50]
```

Рис. 10: Примеры работы с операциями над матрицами

3 Вывод

В ходе выполнения лабораторной работы были получены навыки по подготовке рабочего пространства и инструментария для работы с языком программирования Julia, а также познакомились на простейших примерах с основами синтаксиса Julia.