Лабораторная работа №1. Julia. Установка и настройка. Основные принципы

Дисциплина: Компьютерный практикум по статистическому анализу данных

Манаева Варвара Евгеньевна.

11 ноября 2023

Российский университет дружбы народов, Москва, Россия

Цели и задачи работы

Цель лабораторной работы

Подготовить рабочее пространство и инструментарий для работы с языком программирования Julia, на простейших примерах познакомиться с основами синтаксиса Julia.

Задачи

1. Установите под свою операционную систему Julia, Jupyter (разделы 1.3.1 и 1.3.2).
2. Используя Jupyter Lab, повторите примеры из раздела 1.3.3.
3. Выполните задания для самостоятельной работы (раздел 1.3.4).

Выполнение

Повторение задания (1)

[1]:
```
typeof(3), typeof(3.5), typeof(3//3.55), typeof(sqrt(3+4im)), typeof(pi)
```

[1]:
```
(Int64, Float64, Float64, ComplexF64, Irrational{:π})
```

[2]:
```
1.0/0.0, 1.0/(-0.0), 0.0/0.0
```

[2]:
```
(Inf, -Inf, NaN)
```

[3]:
```
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]
```

[4]:
```
Int64(2.0), Char(2), typeof(Char(2))
```

[4]:
```
(2, '\x02', Char)
```

[5]:
```
convert(Int64, 2.0), convert(Char, 2)
```

[5]:
```
(2, '\x02')
```

[6]:
```
typeof(promote(Int8(1), Float16(4.5), Float32(4.1)))
```

[6]:
```
Tuple{Float32, Float32, Float32}
```

[7]:
```
function f(x)
    x^2
end
```

[7]:
```
f (generic function with 1 method)
```

[8]:
```
f(4)
```

[8]:
```
16
```

[9]:
```
g(x) = x^2
```

Повторение задания (2)

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

[9]: g (generic function with 1 method)

[10]: g(8)

[10]: 64

[11]: a = [4 7 6]
      b = [1, 2, 3]
      a[2], b[2]

[11]: (7, 2)

[12]: a = 1; b = 2; c = 3; d = 4
      Am = [a b; c d]

[12]: 2×2 Matrix{Int64}:
       1  2
       3  4

[13]: Am[1,1], Am[1,2], Am[2,1], Am[2,2]

[13]: (1, 2, 3, 4)

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

[16]: 1×1 Matrix{Int64}:
       27

[17]: aa, AA, aa'

[17]: ([1 2], [1 2; 3 4], [1; 2;;])
```

Рис. 2: Повторение (2)

Документация Julia, readdlm()

`DelimitedFiles.readdlm` — Method

```
readdlm(source, delim::AbstractChar, T::Type, eol::AbstractChar; header=false, skipstart=0,
```

Read a matrix from the source where each line (separated by `eol`) gives one row, with elements separated by the given delimiter. The source can be a text file, stream or byte array. Memory mapped files can be used by passing the byte array representation of the mapped segment as source.

If `T` is a numeric type, the result is an array of that type, with any non-numeric elements as `NaN` for floating-point types, or zero. Other useful values of `T` include `String`, `AbstractString`, and `Any`.

If `header` is `true`, the first row of data will be read as header and the tuple `(data_cells, header_cells)` is returned instead of only `data_cells`.

Specifying `skipstart` will ignore the corresponding number of initial lines from the input.

If `skipblanks` is `true`, blank lines in the input will be ignored.

If `use_mmap` is `true`, the file specified by `source` is memory mapped for potential speedups if the file is large. Default is `false`. On a Windows filesystem, `use_mmap` should not be set to `true` unless the file is only read once and is also not written to. Some edge cases exist where an OS is Unix-like but the filesystem is Windows-like.

If `quotes` is `true`, columns enclosed within double-quote (") characters are allowed to contain new lines and column delimiters. Double-quote characters within a quoted field must be escaped with another double-quote. Specifying `dims` as a tuple of the expected rows and columns (including header, if any) may speed up reading of large files. If `comments` is `true`, lines beginning with `comment_char` and text following `comment_char` in any line are ignored.

Прикладные применения (1)

[16]:
```
write("my_file.txt", "Удивительное рядом!\nДостаточно просто протянуть руку!")
read("my_file.txt", String)
```

[16]:
```
"Удивительное рядом!\nДостаточно просто протянуть руку!"
```

[17]:
```
readline("my_file.txt")
```

[17]:
```
"Удивительное рядом!"
```

[18]:
```
readlines("my_file.txt")
```

[18]:
```
2-element Vector{String}:
 "Удивительное рядом!"
 "Достаточно просто протянуть руку!"
```

[19]:
```
print("Julia is a programming language")
print("Julia is a programming language")
```

```
Julia is a programming languageJulia is a programming language
```

[20]:
```
println("Julia is a programming language")
println("Julia is a programming language")
```

```
Julia is a programming language
Julia is a programming language
```

[28]:
```
struct November
    n::Int
end

Base.show(io::IO, ::MIME"text/plain", d::November) = print(io, d.n, " ноября")
November(11)
```

[28]:
```
11 ноября
```

[22]:
```
open("delim_file.txt", "w") do f
        write(f, "1,2\n3,4\n5,6\n7,8")
end
```

[22]:
```
15
```

Рис. 4: Решения (1)

Прикладные применения (2)

```
[23]: using DelimitedFiles
      readdlm("delim_file.txt", ',', Float64)

[23]: 4×2 Matrix{Float64}:
       1.0  2.0
       3.0  4.0
       5.0  6.0
       7.0  8.0

[24]: parse(Int, "afc", base = 16)

[24]: 2812

[25]: 4+5, [1 2] + [2 3], [1, 2].*3, [1 2] * [1, 2], [10 5]./5

[25]: (9, [3 5], [3, 6], [5], [2.0 1.0])

[26]: [1 3]', [2, 4]', [1 2; 3 4]'

[26]: ([1; 3;;], [2 4], [1 3; 2 4])
```

Рис. 5: Решения (2)

Выводы по проделанной работе

Вывод

В результате выполнения работы мы на простейших примерах ознакомились с синтаксисом языка программирования Julia.

Были записаны скринкасты выполнения и защиты лабораторной работы.