Отчёт по лабораторной работе №1
Статический анализ данных

Julia. Установка и настройка. Основные принципы

Выполнила: Коняева Марина Александровна,
НФИбд-01-21, 1032217044

Содержание

Список иллюстраций

Цель работы

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

Теоретическое введение

Julia — высокоуровневый свободный язык программирования с динамической типизацией, созданный для математических вычислений. Эффективен также и для написания программ общего назначения[1].

Задание

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 лабораторной работы

Устанавливаем инструментарий под ОС Windows: См. рис. 1, См. рис. 2, См. рис. 3? См. рис. 4, См. рис. 5, См. рис. 6

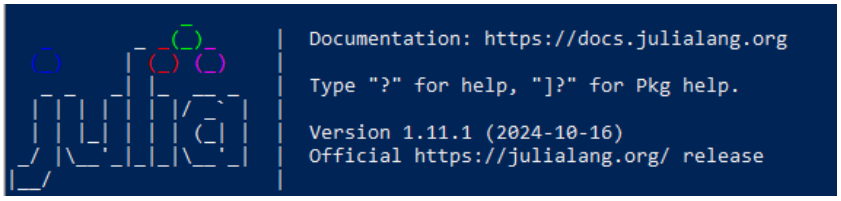

Рис. 1: Установка Julia (уже установлена)

Рис. 2: Установка IJulia

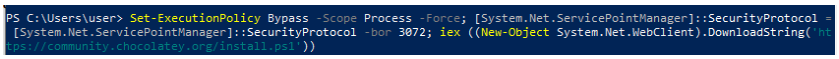

Рис. 3: Установка chocolatey

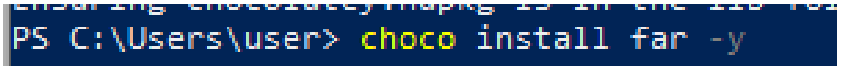

Рис. 4: Установка пакета far

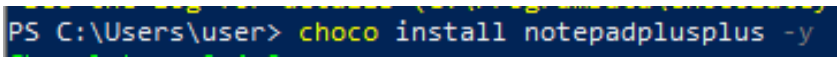

Рис. 5: Установка notepad++

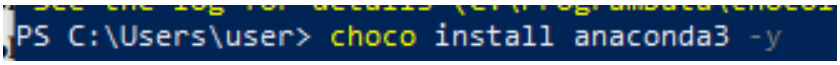

Рис. 6: Установка anaconda3

Выполняем примеры из раздела 1.3.3: См. рис. 7, См. рис. 8, См. рис. 9, См. рис. 10, См. рис. 11, См. рис. 12

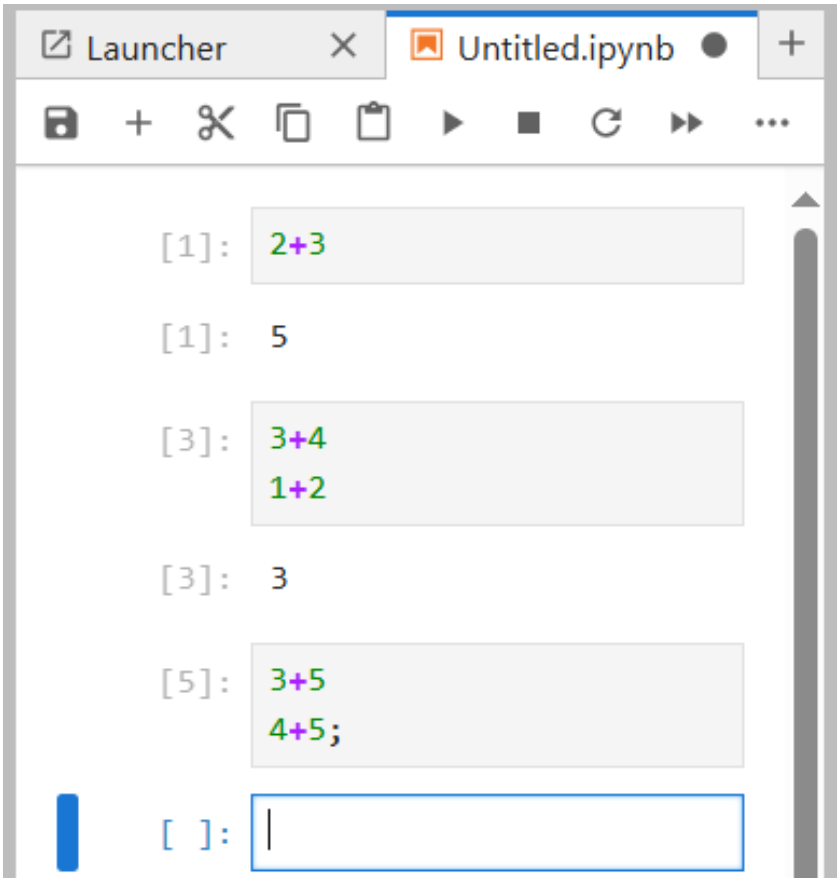

Рис. 7: Базовые математические операции

```
[7]: ?println
```

```
search: println print sprint pointer p
rintstyled
```

```
[7]: println([io::IO], xs...)
```

Print (using `print`) `xs` to `io` followed by a newline. If `io` is not supplied, prints to the default output stream `stdout`.

See also `printstyled` to add colors etc.

Examples

```
julia> println("Hello, world")
Hello, world

julia> io = IOBuffer();

julia> println(io, "Hello", ',', "
world.")

julia> String(take!(io))
"Hello, world.\n"
```

Рис. 8: Справка по функции println

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

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

[13]: 1.0/0.0, 1.0/(-0.0), 0.0/0.0

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

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

[15]: (Float64, Float64, Float64)

[17]: 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, 170141183460469231731687303715
884105727]
  UInt8: [0, 255]
 UInt16: [0, 65535]
 UInt32: [0, 4294967295]
 UInt64: [0, 18446744073709551615]
UInt128: [0, 340282366920938463463374607431768211455]
```

Рис. 9: Определение типов переменных

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

[20]: (2, '\x02', Char)

[22]: convert(Int64, 2.0), convert(Char,2)

[22]: (2, '\x02')

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

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

Рис. 10: Определение типов переменных_2

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

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

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

```
[28]: 16
```

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

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

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

```
[32]: 64
```

Рис. 11: Работа с функциями

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

```
[34]: (7, 2)
```

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

```
[36]: 2×2 Matrix{Int64}:
       1  2
       3  4
```

```
[38]: Am[1,1], Am[1,2], Am[2,1], Am[2,2]
```

```
[38]: (1, 2, 3, 4)
```

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

```
[40]: 1×1 Matrix{Int64}:
       27
```

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

```
[42]: ([1 2], [1 2; 3 4], [1; 2;;])
```

Рис. 12: Работа с векторами и массивами

Задание для самостоятельного выполнения

1. Изучите документацию по основным функциям Julia для чтения / записи / вывода информации на экран: read(), readline(), readlines(), readdlm(), 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 лабораторной работы

Ознакомимся со справками к некоторым функциям: См. рис. 13, См. рис. 14, См. рис. 15, См. рис. 16, См. рис. 17, См. рис. 18, См. рис. 19, См. рис. 20, См. рис. 21

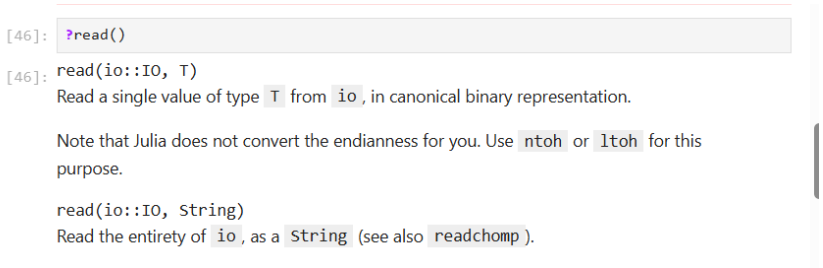

Рис. 1: Функция read()

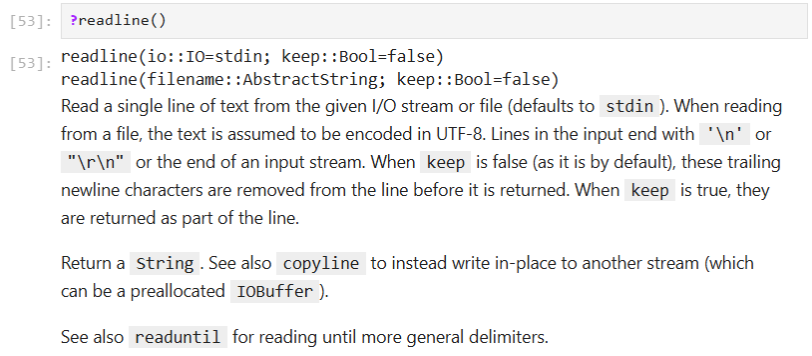

Рис. 2: Функция readline()

```
[55]: ?readlines()

[55]: readlines(io::IO=stdin; keep::Bool=false)
      readlines(filename::AbstractString; keep::Bool=false)
      Read all lines of an I/O stream or a file as a vector of strings. Behavior is equivalent to saving
      the result of reading readline repeatedly with the same arguments and saving the
      resulting lines as a vector of strings. See also eachline to iterate over the lines without
      reading them all at once.
```

Рис. 3: Функция readlines()

```
[57]: ?readdlm()

[57]: No documentation found.

      Binding readdlm does not exist.
```

Рис. 4: Функция readdlm()

```
[59]: ?print()

[59]: print([io::IO], xs...)
      Write to io (or to the default output stream stdout if io is not given) a canonical (un-
      decorated) text representation. The representation used by print includes minimal
      formatting and tries to avoid Julia-specific details.

      print falls back to calling show, so most types should just define show. Define print if
      your type has a separate "plain" representation. For example, show displays strings with
      quotes, and print displays strings without quotes.

      See also println, string, printstyled.
```

Рис. 5: Функция print()

```
[61]: ?println()

[61]: println([io::IO], xs...)
      Print (using print) xs to io followed by a newline. If io is not supplied, prints to the
      default output stream stdout.

      See also printstyled to add colors etc.
```

Рис. 6: Функция println()

```
[63]: ?show()
```

```
[63]: show([io::IO = stdout], x)
```

Write a text representation of a value `x` to the output stream `io`. New types `T` should overload `show(io::IO, x::T)`. The representation used by `show` generally includes Julia-specific formatting and type information, and should be parseable Julia code when possible.

`repr` returns the output of `show` as a string.

For a more verbose human-readable text output for objects of type `T`, define `show(io::IO, ::MIME"text/plain", ::T)` in addition. Checking the `:compact` `IOContext` key (often checked as `get(io, :compact, false)::Bool`) of `io` in such methods is recommended, since some containers show their elements by calling this method with `:compact => true`.

See also `print`, which writes un-decorated representations.

Рис. 7: Функция show()

```
[65]: ?write()
```

```
[65]: write(io::IO, x)
```

Write the canonical binary representation of a value to the given I/O stream or file. Return the number of bytes written into the stream. See also `print` to write a text representation (with an encoding that may depend upon `io`).

The endianness of the written value depends on the endianness of the host system. Convert to/from a fixed endianness when writing/reading (e.g. using `htol` and `ltoh`) to get results that are consistent across platforms.

You can write multiple values with the same `write` call. i.e. the following are equivalent:

```
write(io, x, y...)
write(io, x) + write(io, y...)
```

Рис. 8: Функция write()

```
[69]: ?parse()
```

```
[69]: parse(type, str; base)
```

Parse a string as a number. For `Integer` types, a base can be specified (the default is 10). For floating-point types, the string is parsed as a decimal floating-point number. `Complex` types are parsed from decimal strings of the form `"R±Iim"` as a `Complex(R,I)` of the requested type; `"i"` or `"j"` can also be used instead of `"im"`, and `"R"` or `"Iim"` are also permitted. If the string does not contain a valid number, an error is raised.

!!! compat "Julia 1.1" `parse(Bool, str)` requires at least Julia 1.1.

Рис. 9: Функция parse()

Приведем примеры использования данных функций: См. рис. 22

```
[ ]: content = read("example.txt", String)
     println(content)

[ ]: file = open("example.txt", "r")
     line = readline(file)
     println(line)
     close(file)

[ ]: lines = readlines("example.txt")
     println(lines)

[ ]: using DelimitedFiles
     data = readdlm("data.csv", ',')
     println(data)

[ ]: print("Hello, World!")

[ ]: println("Hello, World!")

[ ]: show(3.34159)

[ ]: file = open("example.txt", "w")
     write(file, "Hello, World!")
     close(file)

[ ]: num = parse(Int, "123")
     println(num)
```

Рис. 10: Примеры использования функций

Пояснения к ним:

- read(): Читает весь контент из файла или потока. Используется для чтения данных целиком из файла или потока в виде строки или байтов.
- readline(): Читает одну строку из файла или потока. Используется для построчного чтения данных.
- readlines(): Читает все строки из файла или потока в виде массива строк. Полезно, если нужно сразу получить доступ ко всем строкам файла.
- readdlm(): Читает данные из файла, разделенные разделителями, и возвращает массив. Используется для работы с табличными данными, такими как CSV-файлы.
- print(): Выводит текст без переноса строки. Полезно для вывода текста на одной строке.
- println(): Выводит текст с переносом строки. Используется для вывода текста с автоматическим переходом на следующую строку.

- show(): Выводит объект в более техническом формате. Предназначен для отображения данных в формате, удобном для разработчиков.
- write(): Записывает данные в файл или поток. Используется для записи строк или байтов в файл.
- parse(): Преобразует строку в заданный тип данных. Удобен для конвертации строк в числа или другие типы, такие как Float64.

Приведем примеры базовых математических операций: См. рис. 23

Рис. 11: Примеры использования математических операций

Приведем примеры операций с матрицами и векторами: См. рис. 24

Рис. 12: Примеры использования математических операций с матрицами и векторами

Пояснения к ним:

- Сложение: Векторы складываются поэлементно.
- Вычитание: Вычитание выполняется поэлементно.
- Скалярное произведение: Используется функция dot() из библиотеки LinearAlgebra для вычисления суммы произведений соответствующих элементов двух векторов.
- Транспонирование: Функция transpose() возвращает транспонированную матрицу.
- Умножение на скаляр: Умножение вектора или матрицы на число выполняется поэлементно.

Заключение

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

Библиографическая справка

[1] Julia: https://ru.wikipedia.org/wiki/Julia