

Компьютерный практикум по статистическому анализу данных

Лабораторная работа № 7. Введение в работу с данными

Сунгурова Мариян Мухсиновна

Содержание

1	Введение	4
2	Теоретическое введение	5
3	Выполнение лабораторной работы	6
4	Выводы	16
	Список литературы	17

Список иллюстраций

3.1	Примеры Считывание данных	6
3.2	Примеры Считывание данных	6
3.3	Примеры Запись данных в файл	7
3.4	Примеры Словари	7
3.5	Примеры DataFrames	8
3.6	Примеры RDatasets	8
3.7	Примеры Missing type	9
3.8	Примеры Fileio	9
3.9	Примеры Кластеризация	10
3.10	Примеры Кластеризация	10
3.11	Примеры Кластеризация	11
3.12	Примеры Кластеризация	11
3.13	Примеры Метод главных компонент	12
3.14	Примеры Линейная регрессия	12
3.15	Примеры Линейная регрессия	13
3.16	Задания 1	14
3.17	Задания 1	15
3.18	Задания 2	15

1 Введение

Цель работы

Основной целью работы является освоение специализированных пакетов Julia для обработки данных.

Задачи

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

2 Теоретическое введение

Julia — высокоуровневый свободный язык программирования с динамической типизацией, созданный для математических вычислений[1]. Эффективен также и для написания программ общего назначения. Синтаксис языка схож с синтаксисом других математических языков, однако имеет некоторые существенные отличия.

Для выполнения заданий была использована официальная документация Julia[2].

3 Выполнение лабораторной работы

Выполним примеры из лабораторной работы для изучения циклов и функций (рис. fig. 3.1 - fig. 3.15)

```
✓ Считывание данных

# Обновление окружения:
using Pkg
Pkg.update
# Установка пакетов:
using Pkg
for p in ["CSV", "DataFrames", "RDatasets", "FileIO"]
Pkg.add(p)
end
using CSV, DataFrames, DelimitedFiles

[1] ✓ 17m 19.0s
```

Рис. 3.1: Примеры Считывание данных

```
▷ ✓ # Считывание данных и их запись в структуру:
P = CSV.File("programminglanguages.csv") |> DataFrame

# Функция определения по названию языка программирования года его создания:
function language_created_year(P, language::String)
loc = findfirst(P[:,2].==language)
return P[loc,1]
end

# Пример вызова функции и определение даты создания языка Python:
language_created_year(P, "Python")
# Пример вызова функции и определение даты создания языка Julia:
language_created_year(P, "Julia")

[4] ✓ 0.3s
... 2012

# Функция определения по названию языка программирования
# года его создания (без учёта регистра):
function language_created_year_v2(P, language::String)
loc = findfirst(lowercase.(P[:,2]).==lowercase.(language))
return P[loc,1]
end

# Пример вызова функции и определение даты создания языка julia:
language_created_year_v2(P, "julia")

[5] ✓ 0.1s
```

Рис. 3.2: Примеры Считывание данных

Запись данных в файл

```
# Запись данных в CSV-файл:
CSV.write("programming_languages_data2.csv", P)
# Можно задать тип файла и разделитель данных:
# Пример записи данных в текстовый файл с разделителем ';':
writedlm("programming_languages_data.txt", Tx, ';')
# Пример записи данных в текстовый файл с разделителем '-':
writedlm("programming_languages_data2.txt", Tx, '-')

[8] ✓ 0.0s

# Построчное считывание данных с указанием разделителя:
P_new_delim = readldl("programming_languages_data2.txt", '-')

[1]
```

Рис. 3.3: Примеры Запись данных в файл

Словари

```
# Инициализация словаря:
dict = Dict{Integer,Vector{String}}()
# Инициализация словаря:
dict2 = Dict{Integer,Vector{String}}()

# Заполнение словаря данными:
for i = 1:size(P,1)
    year, lang = P[i,:]
    if year in keys(dict)
        dict[year] = push!(dict[year], lang)
    else
        dict[year] = [lang]
    end
end

# Пример определения в словаре языков программирования, созданных в 2003 году:
dict[2003]

[10] ✓ 0.7s

... 2-element Vector{String}:
     "Groovy"
     "Scala"
```

Рис. 3.4: Примеры Словари

```
✓ DataFrames

# Подгружаем пакет DataFrames:
using DataFrames
# Задаём переменную со структурой DataFrame:
df = DataFrame(year = P[:,1], language = P[:,2])

# Вывод всех значения столбца year:
df[:,year]

[12] ✓ 0.3s
... 73-element Vector{Int64}:
```

Рис. 3.5: Примеры DataFrames

```
RDatasets

# С данными можно работать также как с наборами данных через пакет RDatasets
# языка R:
# Подгружаем пакет RDatasets:
using RDatasets
# Задаём структуру данных в виде набора данных:
iris = dataset("datasets", "iris")
# Определения типа переменной:
typeof(iris)
describe(iris)

[15] ✓ 1.0s
... 5x7 DataFrame
```

Row	variable	mean	min	median	max	nmissing	dtype
	Symbol	Union...	Any	Union...	Any	Int64	DataType
1	Sepallength	5.84333	4.3	5.8	7.9	0	Float64
2	SepalWidth	3.05733	2.0	3.0	4.4	0	Float64
3	Petallength	3.758	1.0	4.35	6.9	0	Float64
4	PetalWidth	1.19933	0.1	1.3	2.5	0	Float64
5	Species		setosa		virginica	0	CategoricalValue{String, UInt8}

Рис. 3.6: Примеры RDatasets


```
Работа с переменными отсутствующего типа (Missing Values)

# Пакет DataFrames позволяет использовать так называемый «отсутствующий» тип:
# Отсутствующий тип:
a = missing
typeof(a)

[16] ✓ 0.3s
... Missing

# Пример операции с переменной отсутствующего типа:
a + 1

[17] ✓ 0.1s
... missing

# Предположим есть перечень продуктов, для которых заданы калории:
# Определение перечня продуктов:
foods = ["apple", "cucumber", "tomato", "banana"]
# Определение калорий:
calories = [missing, 47, 22, 105]
# Определение типа переменной:
typeof(calories)

[18] ✓ 0.0s
... Vector{Union{Missing, Int64}} (alias for Array{Union{Missing, Int64}, 1})

# Подключаем пакет Statistics:
using Statistics
# Определение среднего значения:
mean(calories)

[21] ✓ 0.1s
... missing
```

Рис. 3.7: Примеры Missing type

```
FileIO

# Подключаем пакет FileIO:
using FileIO
# Попробуем посмотреть, как Julia работает с изображениями.
# Подключим соответствующий пакет:
# Подключаем пакет ImageIO:
import Pkg
Pkg.add("ImageIO")
# Загрузим изображение (в данном случае логотип Julia):
# Загрузка изображения:

[ ]

X1 = load("~/1.JPG")
# Julia хранит изображение в виде множества цветов:
# Определение типа и размера данных:
@show typeof(X1);
@show size(X1);

[32] ✓ 1.7s
... typeof(X1) = Matrix{ColorTypes.RGB{FixedPointNumbers.N0f8}}
size(X1) = (865, 1163)
```

Рис. 3.8: Примеры Fileio

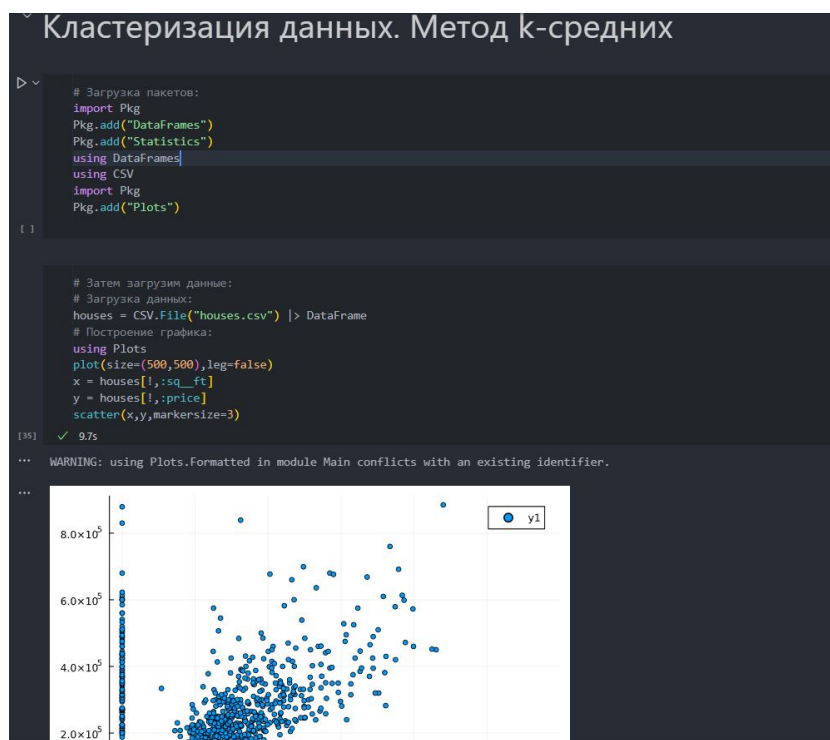


Рис. 3.9: Примеры Кластеризация

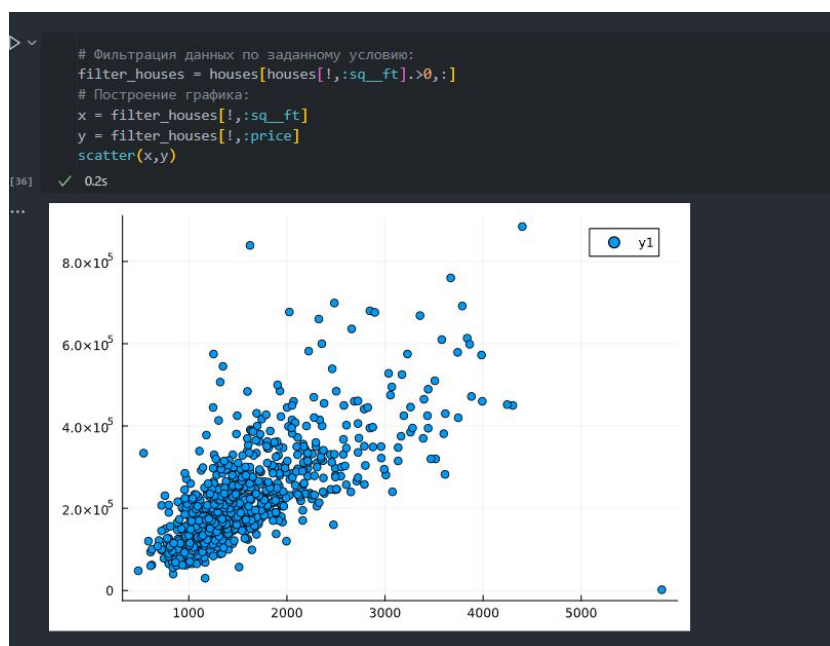


Рис. 3.10: Примеры Кластеризация



Рис. 3.11: Примеры Кластеризация



Рис. 3.12: Примеры Кластеризация

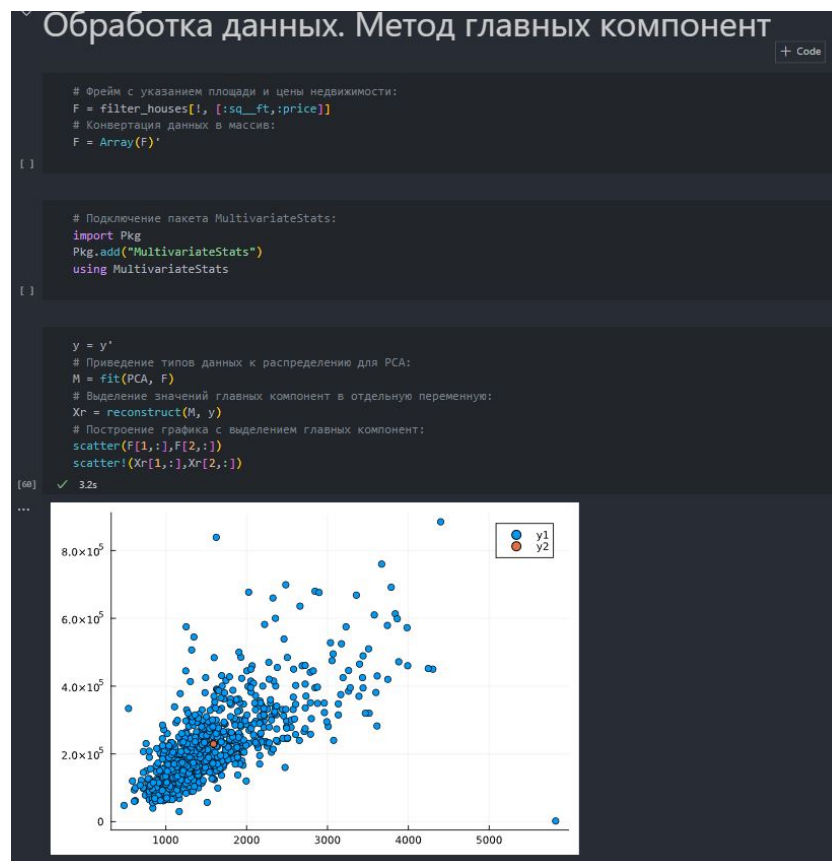


Рис. 3.13: Примеры Метод главных компонент

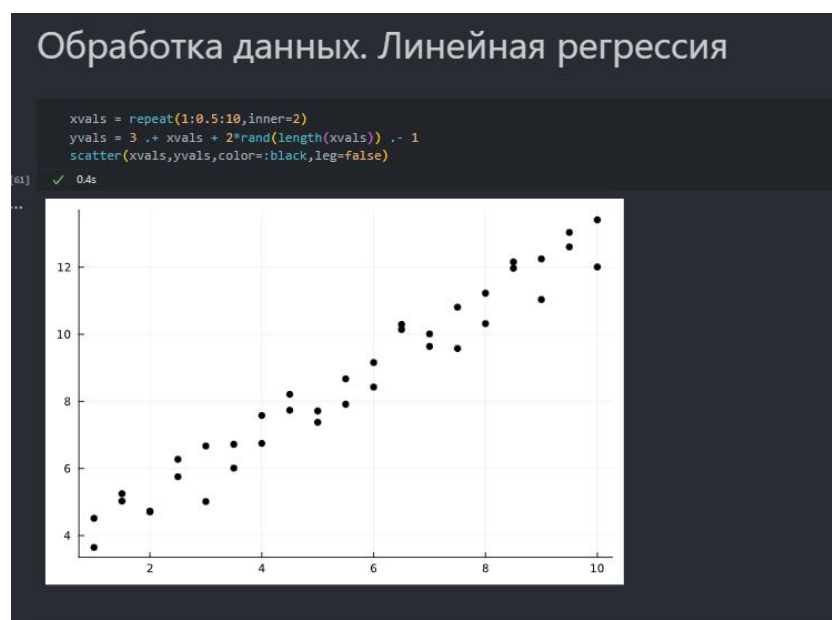


Рис. 3.14: Примеры Линейная регрессия

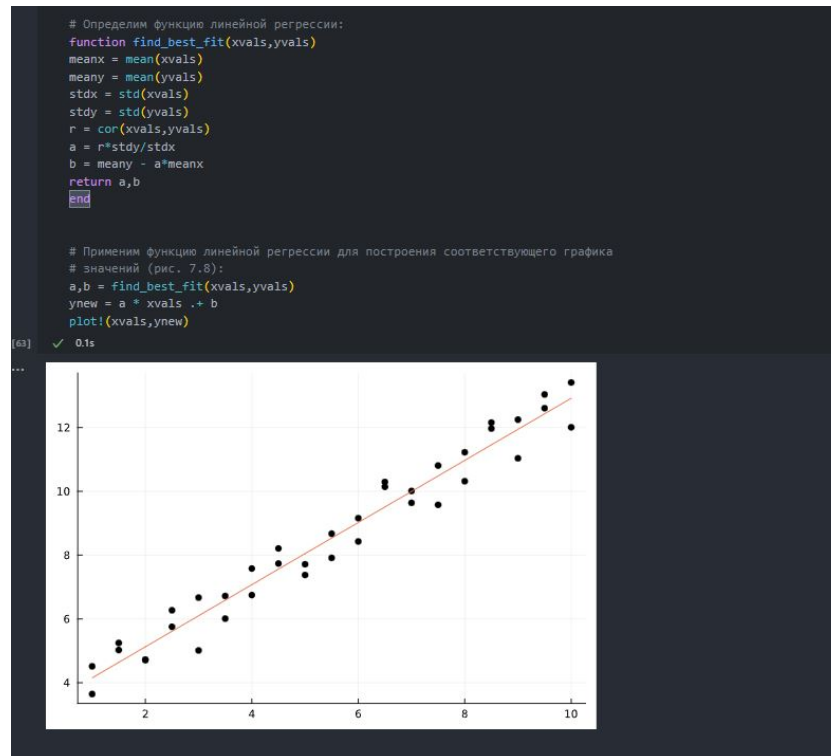


Рис. 3.15: Примеры Линейная регрессия

Затем выполним задания(рис. fig. 3.16 - fig. 3.15)

```

using RDatasets
iris = dataset("datasets", "iris") |> DataFrame
iris_spec = Dict{"setosa"=>1, "versicolor"=>2, "virginica"=>3}
iris[!,5] = [iris_spec[item] for item in iris[!,5]]

X = Matrix(iris)'

#Зададим число кластеров
k = length(unique(iris[!,5]))
# Определение k-среднего
C = kmeans(X,k)

# DataFrame
df = DataFrame(cluster = C.assignments, Sepallength = iris[!,:Sepallength],
SepalWidth = iris[!,:SepalWidth], Petallength = iris[!,:Petallength],
PetalWidth = iris[!,:PetalWidth], Species = iris[!,:Species]);

cluster_fig = plot(legend = false)
for i=1:k
    clustered_houses = df[df[!,:cluster].==i, :]
    xvals = clustered_houses[!,:SepalWidth]
    yvals = clustered_houses[!,:Sepallength]
    scatter!(cluster_fig, xvals,yvals,markersize=4)
end
xlabel!("SepalWidth")
ylabel!("Sepallength")
title!("Color-Coded")
display(cluster_fig)

```

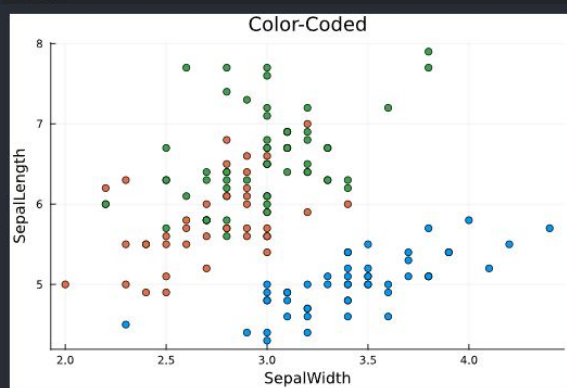


Рис. 3.16: Задания 1

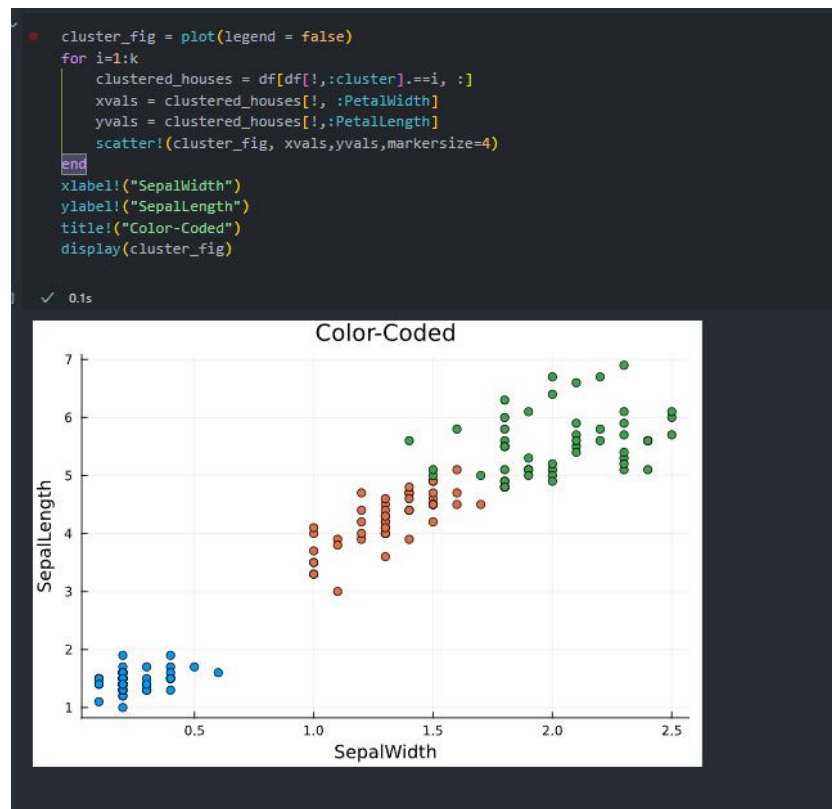


Рис. 3.17: Задания 1



Рис. 3.18: Задания 2

4 Выводы

В результате выполнения данной лабораторной работы было освоено использование специализированных пакетов Julia для обработки данных.

Список литературы

1. JuliaLang [Электронный ресурс]. 2024 JuliaLang.org contributors. URL: <https://julialang.org/> (дата обращения: 11.10.2024).
2. Julia 1.11 Documentation [Электронный ресурс]. 2024 JuliaLang.org contributors. URL: <https://docs.julialang.org/en/v1/> (дата обращения: 11.10.2024).