

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра теории вероятностей и кибербезопасности

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ №7

*дисциплина: Компьютерный практикум по статистическому
анализу данных*

Студент: Быстров Глеб

Группа: НПИбд-01-20

МОСКВА

2023 г.

Цель работы

В данной лабораторной работе мне будет необходимо освоить специализированные пакеты Julia для обработки данных.

Описание процесса выполнения работы

Считывание данных

1. Перед тем, как начать проводить какие-либо операции над данными, необходимо их откуда-то считать и возможно сохранить в определённой структуре.

Довольно часто данные для обработки содержатся в csv-файле, имеющим текстовый формат, в котором данные в строке разделены, например, запятыми, и соответствуют ячейкам таблицы, а строки данных соответствуют строкам таблицы. Также данные могут быть представлены в виде фреймов или множеств.

В Julia для работы с такого рода структурами данных используют пакеты CSV, DataFrames, RDatasets, FileIO (рис. 7.1)

```
Ввод [8]: # 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

Out[8]: language_created_year_v2 (generic function with 1 method)

Ввод [9]: # Пример вызова функции и определение даты создания языка julia:
language_created_year_v2(P,"julia")

Out[9]: 2012

Ввод [*]: # Построчное считывание данных с указанием разделителя:
Tx = readlm("programminglanguages.csv", ',',')
```

Рис. 7.1. Считывание данных

Запись данных в файл

2. Предположим, что требуется записать имеющиеся данные в файл. Для записи данных в формате CSV можно воспользоваться следующим вызовом:

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, '-')
```

Можно проверить, используя `readdlm`, корректность считывания созданного текстового файла:

Построчное считывание данных с указанием разделителя:

```
P_new_delim = readdlm("programming_languages_data2.txt", '-') (рис. 7.2):
```

```
# Запись данных в CSV-файл:
```

```
CSV.write("programming_languages_data2.csv", P)
```

```
"programming_languages_data2.csv"
```

```
# 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, '-')
```

```
# Построчное считывание данных с указанием разделителя:
```

```
P_new_delim = readdlm("programming_languages_data2.txt", '-')
```

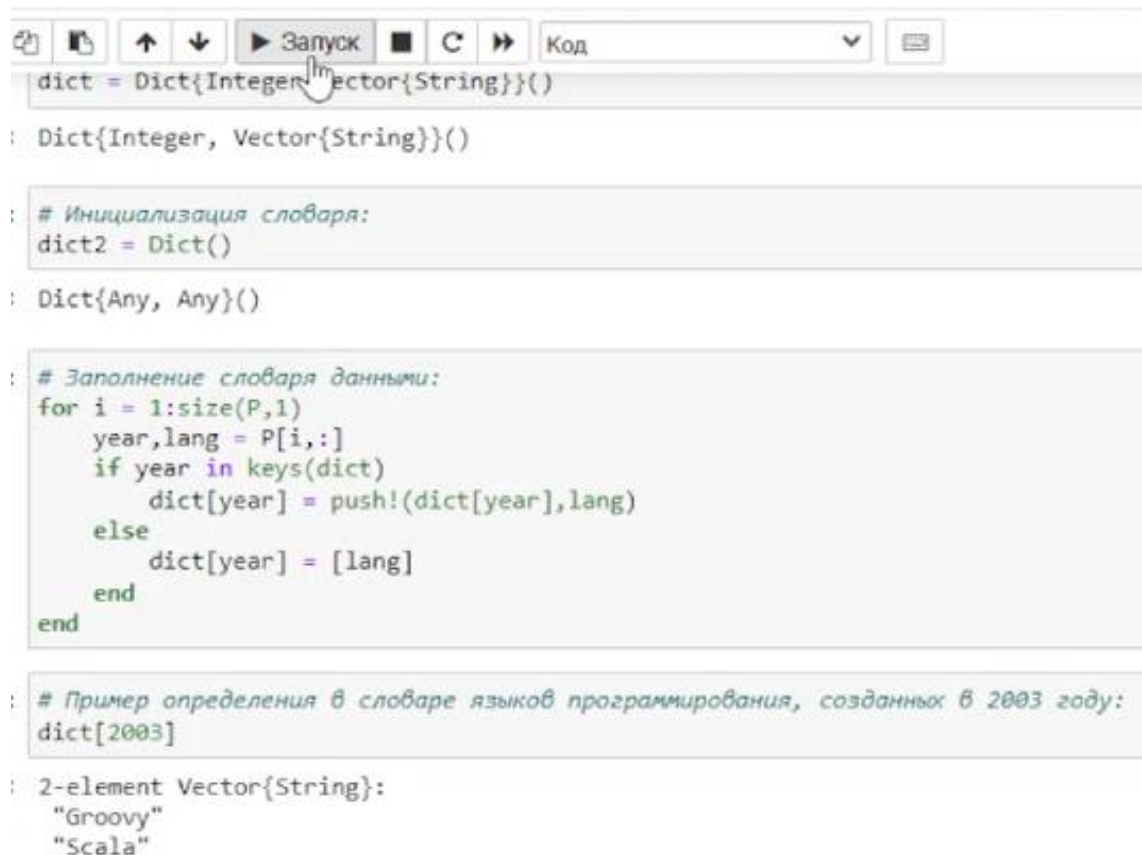
```
74x2 Matrix{Any}:  
  "year"  "language"  
1951      "Regional Assembly Language"  
1952      "Autocode"  
1954      "IPL"  
1955      "FLOW-MATIC"  
1957      "FORTRAN"  
1957      "COMTRAN"  
1958      "LISP"  
1958      "ALGOL 58"  
1959      "FACT"
```

Рис. 7.2. Запись данных в файл

3. При работе с данными бывает удобно записать их в формате словаря.

Предположим, что словарь должен содержать перечень всех языков программирования и года их создания, при этом при указании года выводить все языки программирования, созданные в этом году.

При инициализации словаря можно задать конкретные типы данных для ключей и значений (рис. 7.3):



```
dict = Dict{Integer, Vector{String}}{()  
: Dict{Integer, Vector{String}}{()  
  
: # Инициализация словаря:  
dict2 = Dict()  
: Dict{Any, Any}()  
  
: # 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]  
: 2-element Vector{String}:  
  "Groovy"  
  "Scala"
```

Рис. 7.3. Словари

DataFrames

4. Работа с данными, записанными в структуре DataFrame, позволяет использовать индексацию и получить доступ к столбцам по заданному имени заголовка или по индексу столбца. На примере с данными о языках программирования и годах их создания зададим структуру DataFrame (рис. 7.4):

```
# Подгружаем пакет DataFrames:
using DataFrames
```

```
# Задаём переменную со структурой DataFrame:
df = DataFrame(year = P[:,1], language = P[:,2])
```

Рис. 7.4. DataFrames

RDatasets

- С данными можно работать также как с наборами данных через пакет RDatasets языка R (рис. 7.5):

```
# Определения типа переменной:
typeof(iris)
```

DataFrame

```
describe(iris)
```

5x7 DataFrame

Row	variable	mean	min	median	max	nmissing	eltype
	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}

4

Рис. 7.5. RDatasets

Задания для самостоятельного выполнения

- Кластеризация

Загрузите

```
using RDatasets iris = dataset("datasets", "iris")
```

Используйте Clustering.jl для кластеризации на основе k-средних.

Сделайте точечную диаграмму полученных кластеров.

Подсказка: вам нужно будет проиндексировать фрейм данных, преобразовать его в массив и транспонировать (рис. 7.6):

```
iris3 = iris[iris[:,Species].!=uspecies,:]  
x = iris3[:,SepalLength]  
y = iris3[:,Petallength]  
scatter!(species_figure,x,y)  
end  
xlabel!("SepalLength")  
ylabel!("Petallength")  
title!("Iris")  
display(species_figure)
```

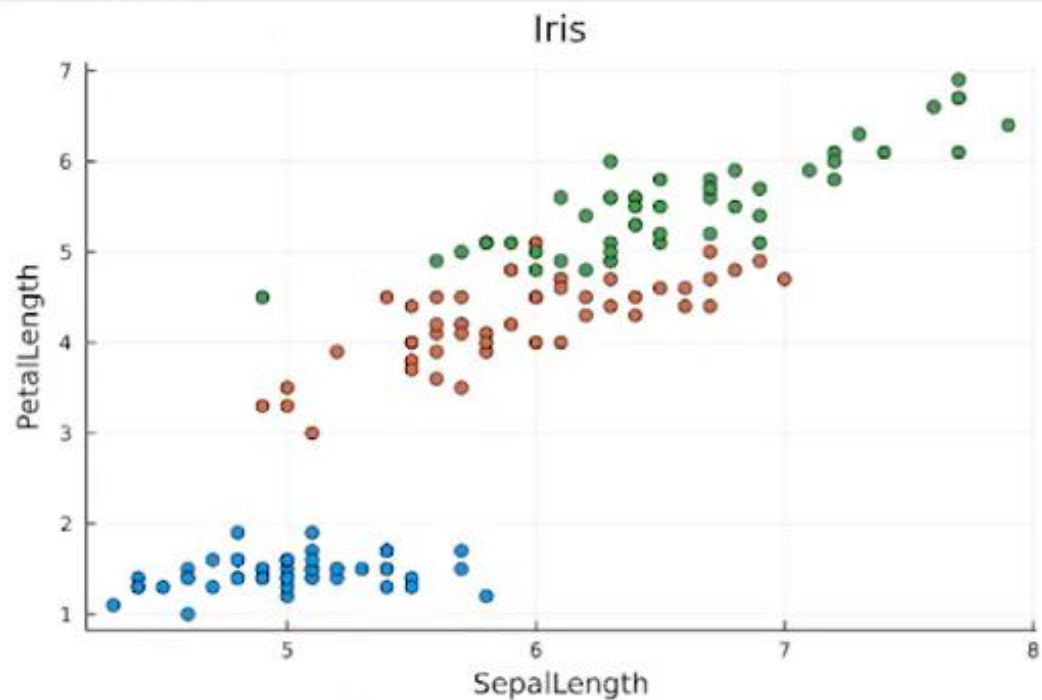


Рис. 7.6. Кластеризация

7. Регрессия (метод наименьших квадратов в случае линейной регрессии)
(рис. 7.7-7.8)

Часть 1

$X = \text{randn}(1000, 3)$

$a_0 = \text{rand}(3)$

$y = X * a_0 + 0.1 * \text{randn}(1000);$

Часть 2

$X = \text{rand}(100);$

$y = 2X + 0.1 * \text{randn}(100);$

Часть 1 Пусть регрессионная зависимость является линейной. Матрица наблюдений факторов X имеет размерность $N \times 3$ и $(N, 3)$, массив результатов $N \times 1$, регрессионная зависимость является линейной. Найдите МНК-оценку для линейной модели.

– Сравните свои результаты с результатами использования `ltsq` из `MultivariateStats.jl` (просмотрите документацию).

– Сравните свои результаты с результатами использования регулярной регрессии наименьших квадратов из `GLM.jl`.

Подсказка. Создайте матрицу данных X_2 , которая добавляет столбец единиц в начало матрицы данных, и решите систему линейных уравнений. Объясните с помощью теоретических выкладок.

Часть 2 Найдите линию регрессии, используя данные (X, y) . Постройте график (X, y) , используя точечный график. Добавьте линию регрессии, используя `abline!`. Добавьте заголовок «График регрессии» и подпишите оси x и y .

```

x1 = X[:,1]
x2 = X[:,2]
x3 = X[:,3]
data = DataFrame(y = y, x1 = x1, x2 = x2, x3 = x3);
lm(@formula(y ~ x1 + x2 + x3), data)

```

StatsModels.TableRegressionModel{LinearModel{GLM.LmResp{Vector{Float64}}, GLM.DensePredChol{Float64, LinearAlgebra.CholeskyPivoted{Float64, Matrix{Float64}}, Vector{Int64}}}}, Matrix{Float64}}

$y \sim 1 + x1 + x2 + x3$

Coefficients:

	Coef.	Std. Error	t	Pr(> t)	Lower 95%	Upper 95%
(Intercept)	-0.00176639	0.0031578	-0.56	0.5760	-0.0079631	0.00443032
x1	0.794747	0.00328422	241.99	<1e-99	0.788302	0.801192
x2	0.612432	0.00307744	199.01	<1e-99	0.606393	0.618471
x3	0.804289	0.00298659	269.30	<1e-99	0.798428	0.81015

Рис. 7.7. Часть 1


```
# Часть 2
X = rand(100);
y = 2X + 0.1 * randn(100);

a,b = find_best_fit(X,y)
ynew = a * X + b
scatter(X, y, title="График регрессии", xlabel="x", ylabel="y", leg=false, line=:scatter)
Plots.abline!(a,b)
```

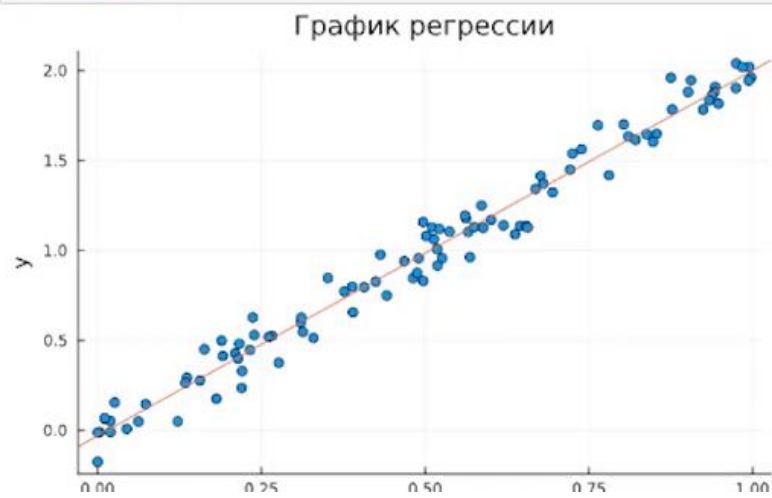


Рис. 7.8. Часть 2

Вывод

В данной лабораторной работе мне успешно удалось освоить специализированные пакеты Julia для обработки данных.