Лабораторная работа 7

Петрушов Дмитрий, 1032212287

2024 г.

Российский университет дружбы народов, Москва, Россия

Цель работы

Цель работы

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

Выполнение лабораторной работы

Считывание данных

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

[2]: 73×2 DataFrame *48 rows omitted*

Row	year	language
	Int64	String31
1	1951	Regional Assembly Language
2	1952	Autocode
3	1954	IPL
4	1955	FLOW-MATIC
5	1957	FORTRAN
6	1957	COMTRAN
7	1958	LISP
8	1958	ALGOL 58
9	1959	FACT

Рис. 1: Считывание данных и запись в структуру

Пример

```
# 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
```

```
language_created_year (generic function with 1 method)
```

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

```
1991
2012
```

Рис. 2: Пример

Поиск “julia” со строчной буквы

```
# 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
```

```
language_created_year_v2 (generic function with 1 method)
```

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

```
2012
```

Рис. 3: Изменение исходной функции

Построчное считывание данных

```
# Построчное считывание данных с указанием разделителя:
Tx = readdlm("programminglanguages.csv", ',')
```

```
74×2 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"
 1959        "COBOL"
 1959        "RPG"
 1962        "APL"
    ⋮
```

```
# Запись данных в CSV-файл:
CSV.write("programming_languages_data2.csv", P)
```

```
"programming_languages_data2.csv"
```

Рис. 5: Запись данных в файл

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

```
Dict{Integer, Vector{String}}()
```

Рис. 6: Инициализация словаря

```
dict[2003]
```

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

Рис. 7: Пример работы словаря

DataFrames

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

73×2 DataFrame

Row	year	language
	Int64	String31
1	1951	Regional Assembly Language
2	1952	Autocode
3	1954	IPL
4	1955	FLOW-MATIC
5	1957	FORTRAN
6	1957	COMTRAN
7	1958	LISP
8	1958	ALGOL 58
9	1959	FACT
10	1959	COBOL
11	1959	RPG
12	1962	APL
13	1962	Simula
⋮	⋮	⋮

RDatasets

```
# Подгружаем пакет RDatasets:
using RDatasets
# Задаём структуру данных в виде набора данных:
iris = dataset("datasets", "iris")
```

150×5 DataFrame

Row	SepalLength	SepalWidth	PetalLength	PetalWidth	Species
	Float64	Float64	Float64	Float64	Cat…
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa
11	5.4	3.7	1.5	0.2	setosa
12	4.8	3.4	1.6	0.2	setosa
13	4.8	3.0	1.4	0.1	setosa

Получение основных статических сведений о каждом столбце в наборе данных

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

```
DataFrame
```

```
describe(iris)
```

5×7 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}

```
# Отсутствующий тип:
a = missing
typeof(a)
```

```
Missing
```

Рис. 11: Использование “отсутствующего” типа

```
# Определение перечня продуктов:
foods = ["apple", "cucumber", "tomato", "banana"]
# Определение калорий:
calories = [missing,47,22,105]
```

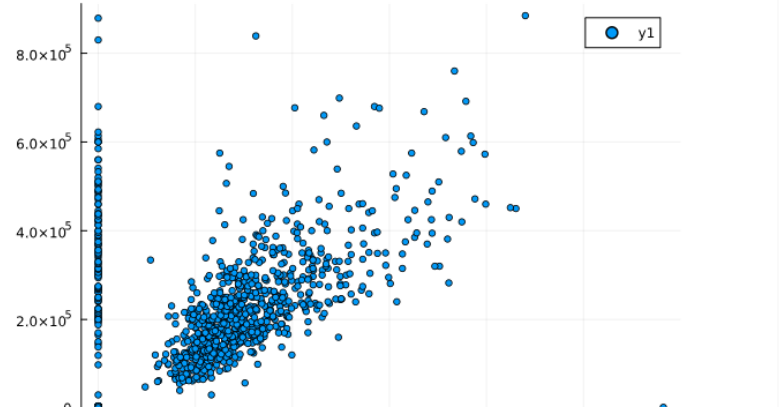
```
4-element Vector{Union{Missing, Int64}}:
     missing
  47
  22
 105
```

Рис. 12: Пример работы с данными, среди которых есть данные с отсутствующим типом

```
# Загрузка изображения:
X1 = load("julialogo.png")
```

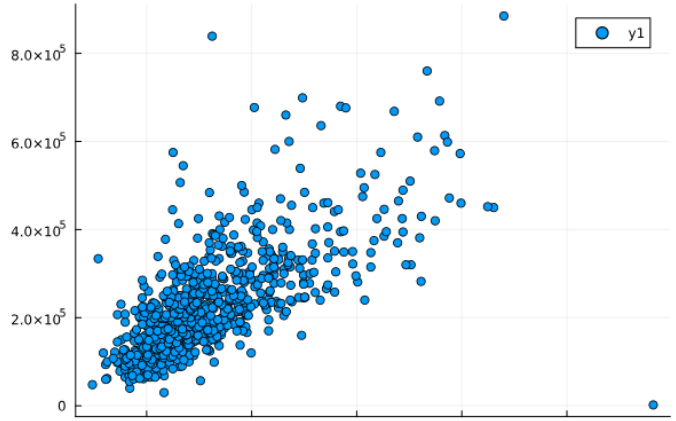
```
460×460 Array{RGBA{N0f8},2} with eltype ColorTypes.RGBA{FixedPointNumbers.N0f8}
 RGBA{N0f8}(0.0,0.0,0.0,0.0)  …  RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)  …  RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)  …  RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
 ⋮                            ⋱
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)  …  RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)  …  RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
```

```
# Построение графика:
using Plots
plot(size=(500,500),leg=false)
x = houses[!,:sqft]
y = houses[!,:price]
scatter(x,y,markersize=3)
```



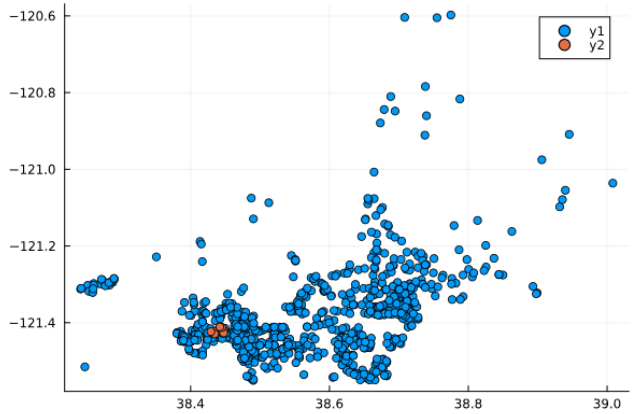
Построение графика без “артефактов”

```
# Фильтрация данных по заданному условию:
filter_houses = houses[houses[!,:sqft].>0,:]
# Построение графика:
x = filter_houses[!,:sqft]
y = filter_houses[!,:price]
scatter(x,y)
```



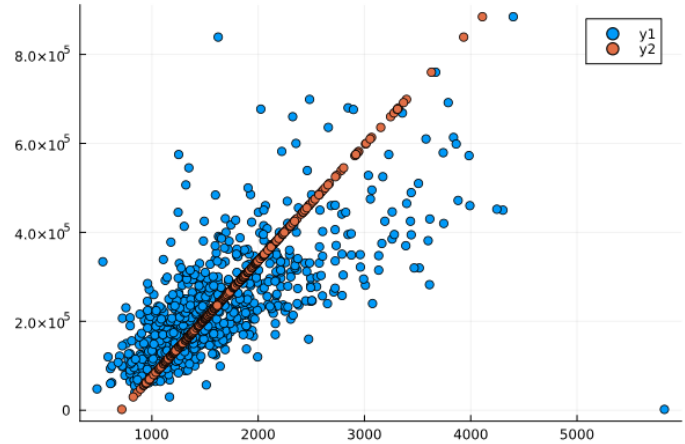
Кластеризация данных. Метод k ближайших соседей

```
# Все объекты недвижимости:
x = filter_houses[!,:latitude];
y = filter_houses[!,:longitude];
scatter(x,y)
# Соседи:
x = filter_houses[idxs,:latitude];
y = filter_houses[idxs,:longitude];
scatter!(x,y)
```

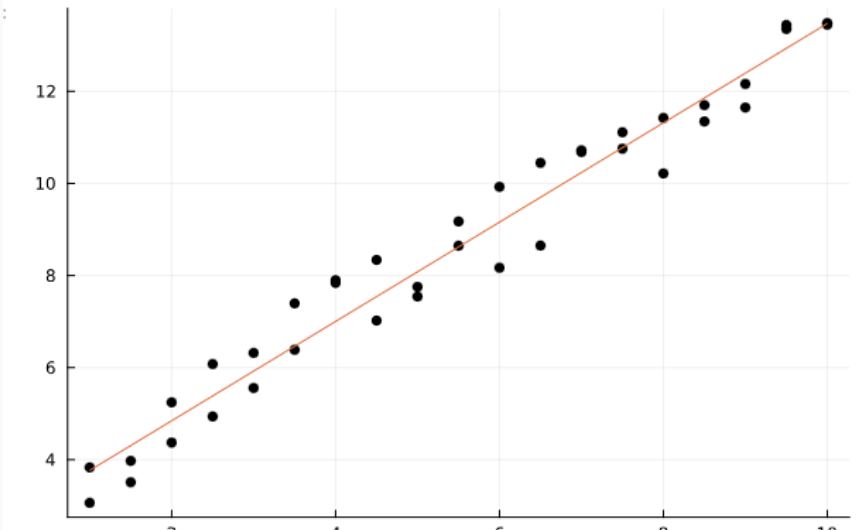


Обработка данных. Метод главных компонент

```
y= MultivariateStats.transform(M,F)
# Выделение значений главных компонент в отдельную переменную:
Xr = reconstruct(M, y)
# Построение графика с выделением главных компонент:
scatter(F[1,:],F[2,:])
scatter!(Xr[1,:],Xr[2,:])
```



```
plot!(xvals, ynew)
```



Сравнение

```
py"""
import numpy
def find_best_fit_python(xvals,yvals):
    meanx = numpy.mean(xvals)
    meany = numpy.mean(yvals)
    stdx = numpy.std(xvals)
    stdy = numpy.std(yvals)
    r = numpy.corrcoef(xvals,yvals)[0][1]
    a = r*stdy/stdx
    b = meany - a*meanx
    return a,b
"""
```

```
find_best_fit_python = py"find_best_fit_python"
xpy = PyObject(xvals)
ypy = PyObject(yvals)
@time a,b = find_best_fit_python(xpy,ypy)
```

```
  0.064046 seconds (74.75 k allocations: 5.179 MiB, 91.19% compilation time)
```

```
(0.9999999961460347, 3.000038831247366)
```

```
import Pkg
Pkg.add("BenchmarkTools")
using BenchmarkTools
@btime a,b = find_best_fit_python(xvals,yvals)
@btime a,b = find_best_fit(xvals,yvals)
```

```
  2.810 ms (25 allocations: 832 bytes)
  429.597 μs (1 allocation: 32 bytes)
   Resolving package versions...
  No Changes to `~/.julia/environments/v1.10/Project.toml`
  No Changes to `~/.julia/environments/v1.10/Manifest.toml`
```

```
(0.9999999961460344, 3.000038831269194)
```

Самостоятельное выполнение

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

150 rows × 5 columns

	SepalLength	SepalWidth	PetalLength	PetalWidth	Species
	Float64	Float64	Float64	Float64	Cat...
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa

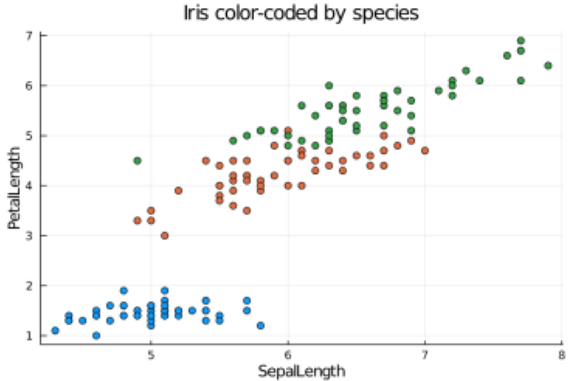
Решение задания №1

```
# Построение графика:
plot(size=(500,500),leg=false)

x = iris[!,:SepalLength]
y = iris[!,:PetalLength]
scatter(x, y, markersize=3, title="Распределение признаков SepalLength и PetalLength",
    xlabel="SepalLength", ylabel="PetalLength", leg=false)
```

Решение задания №1

```
unique_species = unique(iris[!,:Species])
species_figure = plot(legend = false)
for uspecies in unique_species
    iris_sp = iris[iris[!,:Species].==uspecies,:]
    x = iris_sp[!,:SepalLength]
    y = iris_sp[!,:PetalLength]
    scatter!(species_figure, x, y)
end
xlabel!("SepalLength")
ylabel!("PetalLength")
title!("Iris color-coded by species")
display(species_figure)
```



```
function linear_regression_model(X, y)
    X2 = ones(1000)
    X = hcat(X, X2)
    return  X \ y
end
```

```
linear_regression_model (generic function with 1 method)
```

```
a = linear_regression_model(X, y)
print(a)
```

```
[0.6468629061323115, 0.04253975444213305, 0.5580332821634576, -0.0014055938037774972]
```

Рис. 23: Решение задания №1

```
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)
```

```
y ~ 1 + x1 + x2 + x3

Coefficients:
```

	Coef.	Std. Error	t	Pr(>|t|)	Lower 95%	Upper 95%
(Intercept)	-0.00140559	0.00310024	-0.45	0.6504	-0.00748934	0.00467815
x1	0.646863	0.00306265	211.21	<1e-99	0.640853	0.652873
x2	0.0425398	0.00328213	12.96	<1e-34	0.0360991	0.0489804
x3	0.558033	0.00308713	180.76	<1e-99	0.551975	0.564091

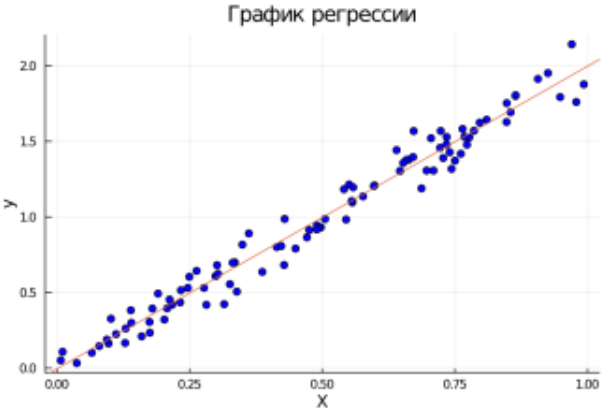
Рис. 24: Решение задания №1

Решение задания №2-2

```
X = rand(100);
y = 2*X + 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", color=:blue, leg=false, line=:scatter)
Plots.abline!(a, b, line=:solid)
```



```
using Plots
plot(stockTree, title="Траектория курса акций", xlabel="Длина биномиального дерева в годах", ylabel="Курс акций", leg=false)
```

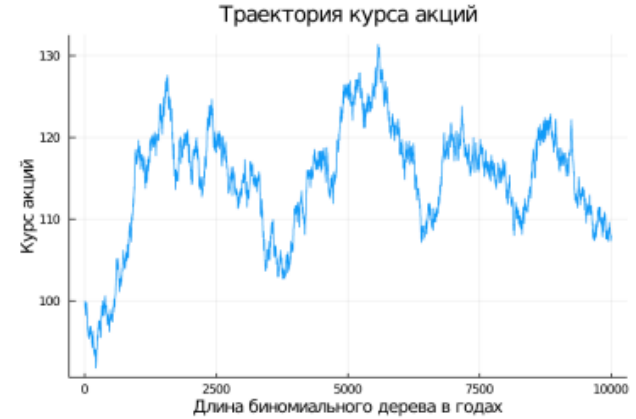
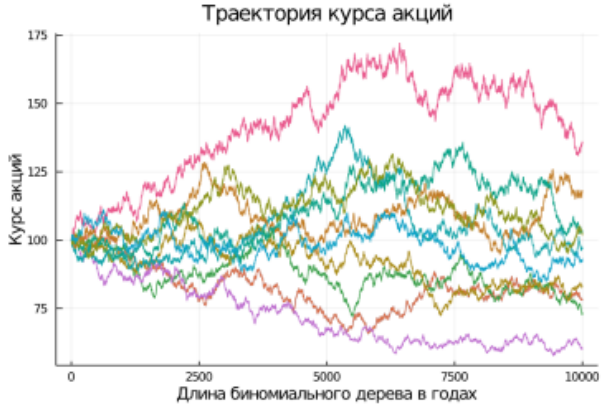


Рис. 26: Решение задания №1

Решение задания №2-2b

```
for i in 1:10
    IJulia.clear_output(true)
    traj = createPath(100, 1, 10000, 0.3, 0.08)
    if i == 1
        p = plot(traj, title="Траектория курса акций", xlabel="Длина биномиального дерева в годах",
            ylabel="Курс акций", leg=false)
    end
    p = plot!(traj)
    display(p)
end
```



Решение задания №2-2c

```
using Base.Threads

Threads.@threads for i in 1:10
    IJulia.clear_output(true)
    traj = createPath(100, 1, 10000, 0.3, 0.08)
    if i == 1
        g = plot(traj, title="Траектория курса акций", xlabel="Длина биномиального дерева в годах",
            ylabel="Курс акций", leg=false)
    end
    g = plot!(traj)
    display(g)
end
```

Выводы

Выводы

В ходе выполнения лабораторной работы были изучены специализированные пакеты Julia для обработки данных.