Лабораторная работа №7.

Введение в работу с данными

Тазаева А. А.

Российский университет дружбы народов, Москва, Россия

Цели работы

Цели работы

Основная цель работы — освоение специализированных пакетов для обработки данных.

Задание

Задание

1. Используя Jupyter Lab, повторите примеры из раздела 7.2.
2. Выполните задания для самостоятельной работы (раздел 7.4).

Считывание данных

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, FileIO
```

```
␛[32m␛[1m Updating␛[22m␛[39m registry at `C:\Users\noname\.julia\registries\General.toml` ●●●
```

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

```
<div><div style = "float: left;"><span>73×2 DataFrame</span></div><div style = "float: right;"><span st
```

```
# 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:
language_created_year(P,"Python")
```

```
1991
```

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

```
2012
```

```
language_created_year(P,"julia")
```

```
MethodError: no method matching getindex(::DataFrame, ::Nothing, ::Int64) ●●●
```

Считывание данных

[9]:
```
# 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
```

[9]:
```
language_created_year_v2 (generic function with 1 method)
```

[10]:
```
# Пример вызова функции и определение даты создания языка julia:
language_created_year_v2(P,"julia")
```

[10]:
```
2012
```

[11]:
```
# Построчное считывание данных с указанием разделителя:
Tx = readdlm("programminglanguages.csv", ',')
```

[11]:
```
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"
    ⋮
 2003        "Scala"
 2005        "F#"
 2006        "PowerShell"
 2007        "Clojure"
 2009        "Go"
 2010        "Rust"
 2011        "Dart"
 2011        "Kotlin"
 2011        "Red"
 2011        "Elixir"
 2012        "Julia"
 2014        "Swift"
```

Словари

```
# Инициализация словаря:
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"
```

DataFrames

DataFrames

```
# Подгружаем пакет 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

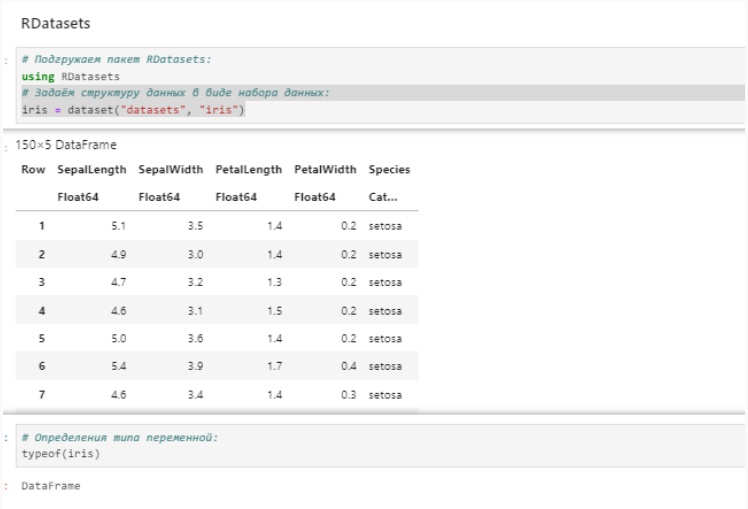
```
# Выбод всех значения столбца year:
df[!,:year]
```

```
73-element Vector{Int64}: ●●●
```

```
# Получение статистических сведений о фрейме:
describe(df)
```

2×7 DataFrame

Row	variable	mean	min	median	max	nmissing	eltype
	Symbol	Union...	Any	Union...	Any	Int64	DataType
1	year	1982.99	1951	1986.0	2014	0	Int64
2	language		ALGOL 58		dBase III	0	String31



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

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

```
DataFrame
```

Рис. 5: RDatasets

Missing Values

Работа с переменными отсутствующего типа (Missing Values)

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

```
Missing
```

```
# Пример операции с переменной отсутствующего типа:
a + 1
```

```
missing
```

```
# Определение перечня продуктов:
foods = ["apple", "cucumber", "tomato", "banana"]
```

```
4-element Vector{String}:
 "apple"
 "cucumber"
 "tomato"
 "banana"
```

```
# Определение калорий:
calories = [missing,47,22,105]
```

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

```
# Определение типа переменной:
typeof(calories)
```

```
Vector{Union{Missing, Int64}} (alias for Array{Union{Missing, Int64}, 1})
```

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

```
missing
```

Кластеризация данных. Метод k-средних.

Кластеризация данных. Метод k-средних

```
using CSV
```

```
# Загрузка данных:
houses = CSV.File("houses.csv") |> DataFrame
```

985×12 DataFrame

Row	street	city	zip	state	beds	baths	sq_ft	type	sale_date	price	latitude	longitude
	String	String15	Int64	String3	Int64	Int64	Int64	String15	String31	Int64	Float64	Float64
1	3526 HIGH ST	SACRAMENTO	95838	CA	2	1	836	Residential	Wed May 21 00:00:00 EDT 2008	59222	38.6319	-121.435
2	51 OMAHA CT	SACRAMENTO	95823	CA	3	1	1167	Residential	Wed May 21 00:00:00 EDT 2008	68212	38.4789	-121.431
3	2796 BRANCH ST	SACRAMENTO	95815	CA	2	1	796	Residential	Wed May 21 00:00:00 EDT 2008	68880	38.6183	-121.444
4	2805 JANETTE WAY	SACRAMENTO	95815	CA	2	1	852	Residential	Wed May 21 00:00:00 EDT 2008	69307	38.6168	-121.439
5	6001 MCMAHON DR	SACRAMENTO	95824	CA	2	1	797	Residential	Wed May 21 00:00:00 EDT 2008	81900	38.5195	-121.436
6	5828 PEPPERMILL CT	SACRAMENTO	95841	CA	3	1	1122	Condo	Wed May 21 00:00:00 EDT 2008	89921	38.6626	-121.328
7	6048 OGDEN NASH WAY	SACRAMENTO	95842	CA	3	2	1104	Residential	Wed May 21 00:00:00 EDT 2008	90895	38.6817	-121.352
8	2561 19TH AVE	SACRAMENTO	95820	CA	3	1	1177	Residential	Wed May 21 00:00:00 EDT 2008	91002	38.5351	-121.481
9	11150 TRINITY RIVER DR Unit 114	RANCHO CORDOVA	95670	CA	2	2	941	Condo	Wed May 21 00:00:00 EDT 2008	94905	38.6212	-121.271
10	7325 10TH ST	RIO LINDA	95673	CA	3	2	1146	Residential	Wed May 21 00:00:00 EDT 2008	98937	38.7009	-121.443
11	645 MORRISON AVE	SACRAMENTO	95838	CA	3	2	909	Residential	Wed May 21 00:00:00 EDT 2008	100309	38.6377	-121.452
12	4085 FAWN CIR	SACRAMENTO	95823	CA	3	2	1289	Residential	Wed May 21 00:00:00 EDT 2008	106250	38.4707	-121.459
13	2930 LA ROSA RD	SACRAMENTO	95815	CA	1	1	871	Residential	Wed May 21 00:00:00 EDT 2008	106852	38.6187	-121.436
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
974	2181 WINTERHAVEN CIR	CAMERON PARK	95682	CA	3	2	0	Residential	Thu May 15 00:00:00 EDT 2008	224500	38.6976	-120.996
975	7540 HICKORY AVE	ORANGEVALE	95662	CA	3	1	1456	Residential	Thu May 15 00:00:00 EDT 2008	225000	38.7031	-121.235
976	5024 CHAMBERLIN CIR	ELK GROVE	95757	CA	3	2	1450	Residential	Thu May 15 00:00:00 EDT 2008	228000	38.3898	-121.446
977	2400 INVERNESS DR	LINCOLN	95648	CA	3	2	1358	Residential	Thu May 15 00:00:00 EDT 2008	229027	38.8978	-121.325
978	5 BISHOPGATE CT	SACRAMENTO	95823	CA	4	2	1329	Residential	Thu May 15 00:00:00 EDT 2008	229500	38.4679	-121.445
979	5601 REXLEIGH DR	SACRAMENTO	95823	CA	4	2	1715	Residential	Thu May 15 00:00:00 EDT 2008	230000	38.4453	-121.442
980	1909 YARNELL WAY	ELK GROVE	95758	CA	3	2	1262	Residential	Thu May 15 00:00:00 EDT 2008	230000	38.4174	-121.484
981	9169 GARLINGTON CT	SACRAMENTO	95829	CA	4	3	2280	Residential	Thu May 15 00:00:00 EDT 2008	232425	38.4577	-121.36
982	6932 RUSKUT WAY	SACRAMENTO	95823	CA	3	2	1477	Residential	Thu May 15 00:00:00 EDT 2008	234000	38.4999	-121.459
983	7933 DAFFODIL WAY	CITRUS HEIGHTS	95610	CA	3	2	1216	Residential	Thu May 15 00:00:00 EDT 2008	235000	38.7088	-121.257
984	8304 RED FOX WAY	ELK GROVE	95758	CA	4	2	1685	Residential	Thu May 15 00:00:00 EDT 2008	235301	38.417	-121.397
985	3882 YELLOWSTONE LN	EL DORADO HILLS	95762	CA	3	2	1362	Residential	Thu May 15 00:00:00 EDT 2008	235738	38.6562	-121.076

Кластеризация данных. Метод k-средних.

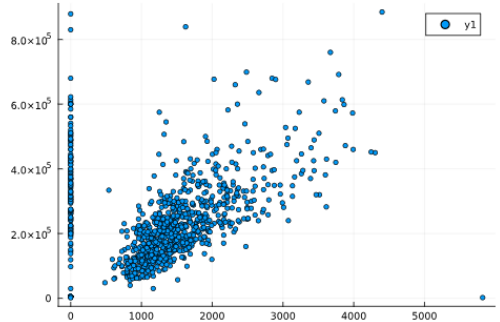
```
import Pkg
Pkg.add("Plots")
```

```
   Resolving package versions...
  No Changes to `C:\Users\noname\.julia\environments\v1.10\Project.toml`
  No Changes to `C:\Users\noname\.julia\environments\v1.10\Manifest.toml`
```

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

```
<?xml version="1.0" encoding="utf-8"?> ●●●
```

```
x = houses[!,:sq__ft]
y = houses[!,:price]
scatter(x,y,markersize=3)
```



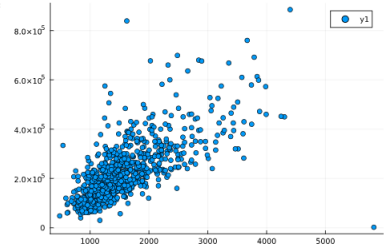
Кластеризация данных. Метод k-средних.

```
# фильтрация данных по заданному условию:
filter_houses = houses[houses[!,:sq__ft],>0,:]
```

814×12 DataFrame

Row	street	city	zip	state	beds	baths	sq__ft	type	sale_date	price	latitude	longitude
	String	String15	Int64	String3	Int64	Int64	Int64	String15	String31	Int64	Float64	Float64
1	3526 HIGH ST	SACRAMENTO	95838	CA	2	1	836	Residential	Wed May 21 00:00:00 EDT 2008	59222	38.6319	-121.435
2	51 OMAHA CT	SACRAMENTO	95823	CA	3	1	1167	Residential	Wed May 21 00:00:00 EDT 2008	68212	38.4789	-121.431
3	2796 BRANCH ST	SACRAMENTO	95815	CA	2	1	796	Residential	Wed May 21 00:00:00 EDT 2008	68880	38.6183	-121.444
4	2805 JANETTE WAY	SACRAMENTO	95815	CA	2	1	852	Residential	Wed May 21 00:00:00 EDT 2008	69307	38.6168	-121.439
5	6001 MCMAHON DR	SACRAMENTO	95824	CA	2	1	797	Residential	Wed May 21 00:00:00 EDT 2008	81900	38.5195	-121.436
6	5828 PEPPERMILL CT	SACRAMENTO	95841	CA	3	1	1122	Condo	Wed May 21 00:00:00 EDT 2008	89921	38.6626	-121.328
7	6048 OGDEN NASH WAY	SACRAMENTO	95842	CA	3	2	1104	Residential	Wed May 21 00:00:00 EDT 2008	90895	38.6817	-121.352

```
# Построение графика:
x = filter_houses[!,:sq__ft]
y = filter_houses[!,:price]
scatter(x,y)
```



Кластеризация данных. Метод k-средних.

```
# Определение средней цены для определённого типа домов:
grouped=groupby(filter_houses,:type)
```

GroupedDataFrame with 3 groups based on key: type

First Group (759 rows): type = "Residential"

Row	street	city	zip	state	beds	baths	sq_ft	type	sale_date	price	latitude	longitude
	String	String15	Int64	String3	Int64	Int64	Int64	String15	String31	Int64	Float64	Float64
1	3526 HIGH ST	SACRAMENTO	95838	CA	2	1	836	Residential	Wed May 21 00:00:00 EDT 2008	59222	38.6319	-121.435
2	51 OMAHA CT	SACRAMENTO	95823	CA	3	1	1167	Residential	Wed May 21 00:00:00 EDT 2008	68212	38.4789	-121.431
3	2796 BRANCH ST	SACRAMENTO	95815	CA	2	1	796	Residential	Wed May 21 00:00:00 EDT 2008	68880	38.6183	-121.444
4	2805 JANETTE WAY	SACRAMENTO	95815	CA	2	1	852	Residential	Wed May 21 00:00:00 EDT 2008	69307	38.6168	-121.439
5	6001 MCMAHON DR	SACRAMENTO	95824	CA	2	1	797	Residential	Wed May 21 00:00:00 EDT 2008	81900	38.5195	-121.436
6	6048 OGDEN NASH WAY	SACRAMENTO	95842	CA	3	2	1104	Residential	Wed May 21 00:00:00 EDT 2008	90895	38.6817	-121.352

```
# Подключение пакета Statistics:
using Statistics
```

```
# Добавление данных :Latitude и :Longitude в новый фрейм:
X = filter_houses[!,[:latitude,:longitude]]
```

```
<div><div style = "float: left;"><span>814×2 DataFrame</span></div><div style = "float: right;"><span style = "font-style: italic;">789 rows omitted</span></div><div style = "clear: both;
```

```
# Конвертация данных в матричный вид:
X = Matrix(X)
```

```
814×2 Matrix{Float64}: ...
```

```
# Транспонирование матрицы с данными:
X = X'
```

```
2×814 adjoint(::Matrix{Float64}) with eltype Float64:
   38.6319    38.4789    38.6183  …    38.7088    38.417    38.6552
 -121.435   -121.431   -121.444       -121.257   -121.397  -121.076
```

```
# Задание количества кластеров:
k = length(unique(filter_houses[!,:zip]))
```

```
66
```

Рис. 10: Кластеризация данных. Метод k-средних. Часть 4

Кластеризация данных. Метод k ближайших соседей.

Кластеризация данных. Метод k ближайших соседей

```
# Подключение пакета NearestNeighbors:
import Pkg
Pkg.add("NearestNeighbors")
using NearestNeighbors
```

```
  Resolving package versions...
  No Changes to `C:\Users\noname\.julia\environments\v1.10\Project.toml`
  No Changes to `C:\Users\noname\.julia\environments\v1.10\Manifest.toml`
```

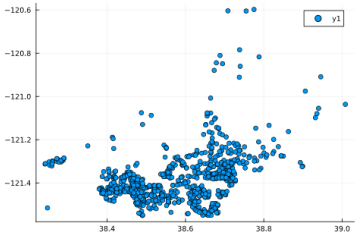
```
knearest = 10
id = 70
point = X[:,id]
```

```
2-element Vector{Float64}:
   38.44004
 -121.421012
```

```
# Поиск ближайших соседей:
kdtree = KDTree(X)
idxs, dists = knn(kdtree, point, knearest, true)
```

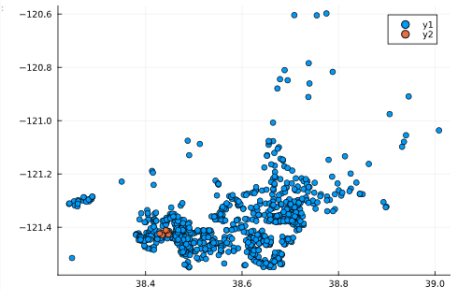
```
([70, 764, 196, 125, 557, 368, 415, 92, 112, 683], [0.0, 0.0026489153936138, 0.0082532025050462, 0.0084735851236057, 0.0091640735483701888, 0.0094050651246977706, 0.
```

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



Кластеризация данных. Метод k ближайших соседей.

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




```
# Фильтрация по районам соседних домов:
cities = filter_houses[idxs,:city]
```

```
10-element PooledArrays.PooledVector{String15, UInt32, Vector{UInt32}}:
 "SACRAMENTO"
 "ELK GROVE"
 "SACRAMENTO"
 "SACRAMENTO"
 "SACRAMENTO"
 "SACRAMENTO"
 "ELK GROVE"
 "ELK GROVE"
 "ELK GROVE"
 "ELK GROVE"
```

Кластеризация данных. Метод k ближайших соседей.

Обработка данных. Метод главных компонент

```
# Фрейм с указанием площади и цены недвижимости:
F = filter_houses[!,[:sq__ft_,:price]]
y = filter_houses[!,:price]
```

```
814-element Vector{Int64}:
 59222
 68212
 68880
 69307
 81900
 89921
 98895
 91002
 94985
 98937
100309
106250
106852

226252
225000
228000
229027
229500
230000
230000
232425
234000
235000
235301
235738
```

```
# Конвертация данных в массив:
F = Matrix(F)'
y=y'
```

```
1×814 adjoint(::Vector{Int64}) with eltype Int64:
 59222  68212  68880  69307  81900  89921  …  234000  235000  235301  235738
```

```
# Подключение пакета MultivariateStats:
import Pkg
Pkg.add("MultivariateStats")
using MultivariateStats
```

```
  Resolving package versions...
 No Changes to `C:\Users\noname\.julia\environments\v1.10\Project.toml`
 No Changes to `C:\Users\noname\.julia\environments\v1.10\Manifest.toml`
```

```
# Приведение типов данных к распределению для PCA:
M = fit(PCA, F)
```

```
PCA(indim = 2, outdim = 1, principalratio = 0.9999840784692097)

Pattern matrix (unstandardized loadings):
──────────────
            PC1
──────────────
1  460.52
2    1.19826e5
──────────────

Importance of components:
```

```
# 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,:])
```

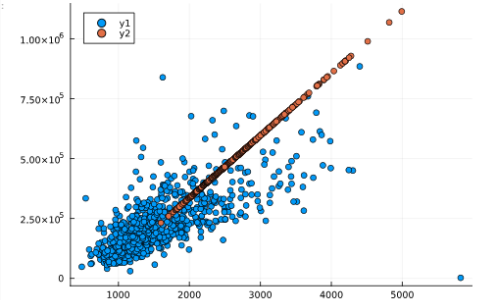
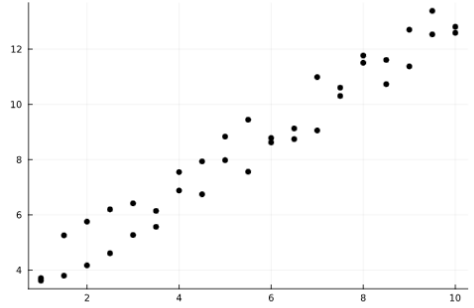


Рис. 14: Кластеризация данных. Метод главных компонент. Часть 2

Обработка данных. Линейная регрессия.

Обработка данных. Линейная регрессия

```
xvals = repeat(1:0.5:10,inner=2)
yvals = 3 .+ xvals + 2*rand(length(xvals)) .- 1
scatter(xvals,yvals,color=:black,leg=false)
```

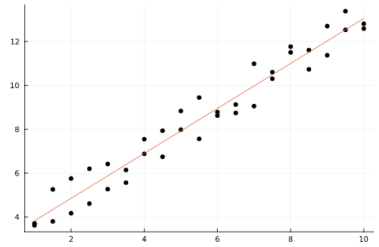



```
function find_best_fit(xvals,yvals)
meanx = mean(xvals)
meany = mean(yvals)
stdx = std(xvals)
stdy = std(yvals)
r = cor(xvals,yvals)
a = r*stdy/stdx
b = meany - a*meanx
return a,b
end
```

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

Обработка данных. Линейная регрессия.

```
a,b = find_best_fit(xvals,yvals)
ynew = a * xvals .+ b
plot!(xvals,ynew)
```




```
xvals = 1:100000;
xvals = repeat(xvals,inner=3);
yvals = 3 .+ xvals + 2*rand(length(xvals)) .- 1;
@show size(xvals)
@show size(yvals)
@time a,b = find_best_fit(xvals,yvals)
```

```
size(xvals) = (300000,)
size(yvals) = (300000,)
  0.035450 seconds (20.15 k allocations: 1.315 MiB, 96.12% compilation time)
(1.0000000574192405, 2.998644319261075)
```

```
import Pkg
Pkg.add("PyCall")
Pkg.add("Conda")
using PyCall
using Conda
```

```
   Resolving package versions...
  No Changes to `C:\Users\noname\.julia\environments\v1.10\Project.toml`
  No Changes to `C:\Users\noname\.julia\environments\v1.10\Manifest.toml`
   Resolving package versions...
  No Changes to `C:\Users\noname\.julia\environments\v1.10\Project.toml`
  No Changes to `C:\Users\noname\.julia\environments\v1.10\Manifest.toml`
```

Обработка данных. Линейная регрессия.

```
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
"""
```

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

```
  0.137126 seconds (65.78 k allocations: 4.622 MiB, 51.55% compilation time)
```

```
(1.0000000574192378, 2.998644319399318)
```

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

```
   Resolving package versions...
  No Changes to `C:\Users\noname\.julia\environments\v1.10\Project.toml`
  No Changes to `C:\Users\noname\.julia\environments\v1.10\Manifest.toml`
  4.701 ms (28 allocations: 976 bytes)
  434.600 μs (1 allocation: 32 bytes)
```

```
(1.0000000574192405, 2.998644319261075)
```

Выводы по проделанной работе

Выводы по проделанной работе

В ходе лабораторной работы мною были освоены специализированные пакеты для обработки данных.