Лабораторная работа №7

Введение в работу с данными

Тазаева Анастасия Анатольевна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Основной целью работы является освоение специализированных пакетов для обработки данных.

2 Задание

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

3 Выполнение лабораторной работы

3.1 Примеры из раздела 6.2

Примеры представлены на рис. 3.1 - 3.21.

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) ●●●
```

Рис. 3.1: Считывание данных. Часть 1

[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"
```

Рис. 3.2: Считывание данных. Часть 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", '-')
```

```
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"
```

Рис. 3.3: Считывание данных. Часть 3

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"
```

Рис. 3.4: Словари

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

```
# Вывод 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

Рис. 3.5: DataFrames

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. 3.6: RDatasets

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
```

Рис. 3.7: Работа с переменными отсутствующего типа (Missing Values). Часть 1

```
# Определение среднего значения без значений с отсутствующим типом:
mean(skipmissing(calories))
```

```
58.0
```

```
# Задание сведений о ценах:
prices = [0.85,1.6,0.8,0.6]
# Формирование данных о калориях:
dataframe_calories = DataFrame(item=foods,calories=calories)
# Формирование данных о ценах:
dataframe_prices = DataFrame(item=foods,price=prices)
# Объединение данных о калориях и ценах:
DF = innerjoin(dataframe_calories,dataframe_prices,on=:item)
```

4×3 DataFrame

Row	item	calories	price
	String	Int64?	Float64
1	apple	*missing*	0.85
2	cucumber	47	1.6
3	tomato	22	0.8
4	banana	105	0.6

Рис. 3.8: Работа с переменными отсутствующего типа (Missing Values). Часть 2

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	38.6552	-121.076

Рис. 3.9: Кластеризация данных. Метод k-средних. Часть 1

```
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)
```

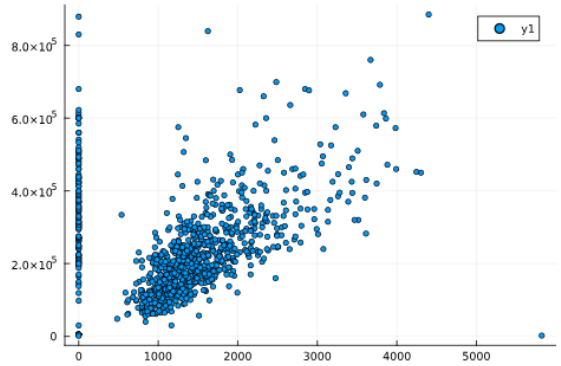


Рис. 3.10: Кластеризация данных. Метод k-средних. Часть 2

```
# 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)
```

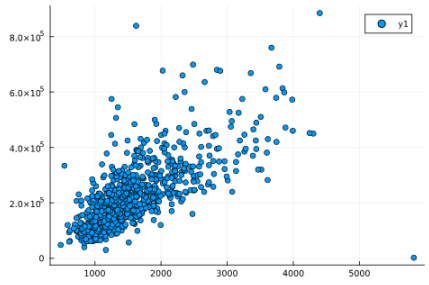
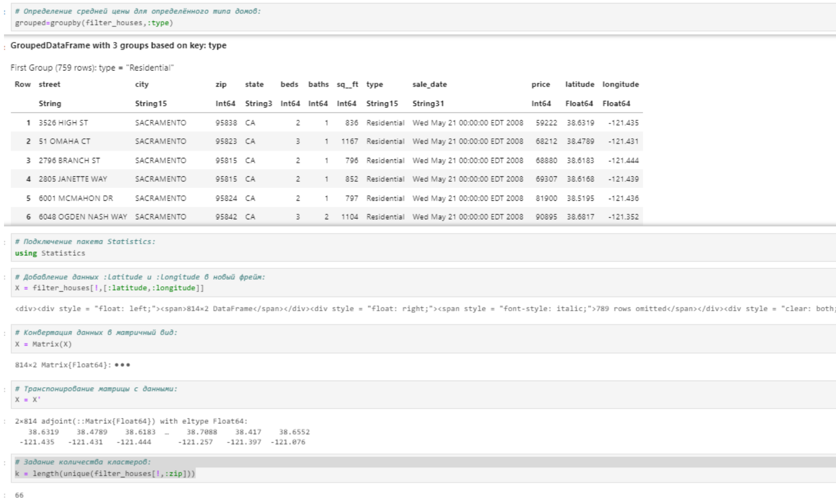


Рис. 3.11: Кластеризация данных. Метод k-средних. Часть 3



```
# 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}: •••
```

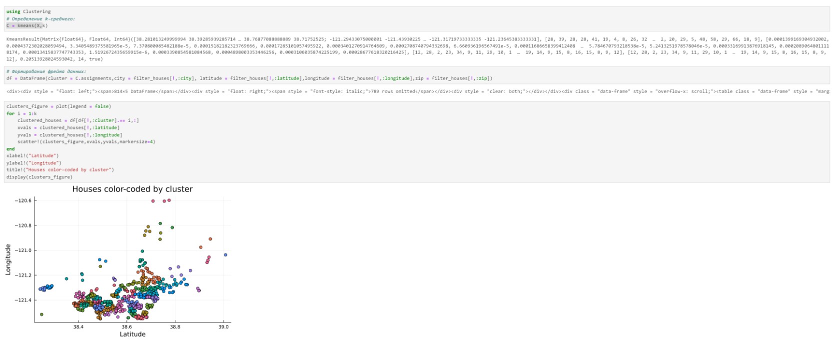
```
# Транспонирование матрицы с данными:
X = X'
```

```
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. 3.12: Кластеризация данных. Метод k-средних. Часть 4



```
using Clustering
# Определение k-среднего:
C = kmeans(X,k)
```

```
# Формирование фрейма данных:
df = DataFrame(cluster = C.assignments,city = filter_houses[!,:city], latitude = filter_houses[!,:latitude],longitude = filter_houses[!,:longitude],zip = filter_houses[!,:zip])
```

```
<div><div style = "float: left;"><span>814×5 DataFrame</span></div><div style = "float: right;"><span style = "font-style: italic;">789 rows omitted</span></div>
```

```
clusters_figure = plot(legend = false)
for i = 1:k
    clustered_houses = df[df[!,:cluster].== i,:]
    xvals = clustered_houses[!,:latitude]
    yvals = clustered_houses[!,:longitude]
    scatter!(clusters_figure,xvals,yvals,markersize=4)
end
xlabel!("Latitude")
ylabel!("Longitude")
title!("Houses color-coded by cluster")
display(clusters_figure)
```


Рис. 3.13: Кластеризация данных. Метод k-средних. Часть 5

```
unique_zips = unique(filter_houses[!,:zip])
zips_figure = plot(legend = false)
for uzip in unique_zips
    subs = filter_houses[filter_houses[!,:zip].==uzip,:]
    x = subs[!,:latitude]
    y = subs[!,:longitude]
    scatter!(zips_figure,x,y)
end
xlabel!("Latitude")
ylabel!("Longitude")
title!("Houses color-coded by zip code")
display(zips_figure)
```

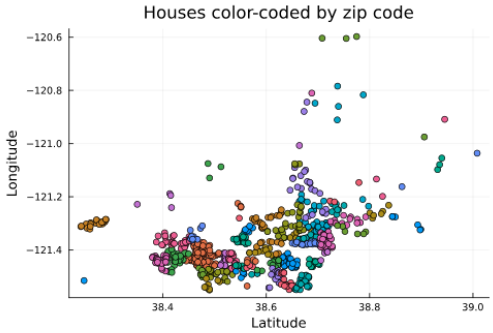


Рис. 3.14: Кластеризация данных. Метод k-средних. Часть 6

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`
```

```
kneарest = 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, kneарest, true)
```

```
([70, 764, 196, 125, 557, 368, 415, 92, 112, 683], [0.0, 0.006264891539364138, 0.008253202590504 62, 0.008473585132630057, 0.009164073548370188, 0.009405065124697706, 0.
```

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

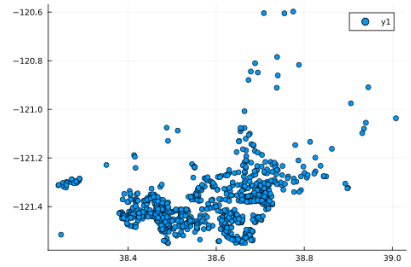
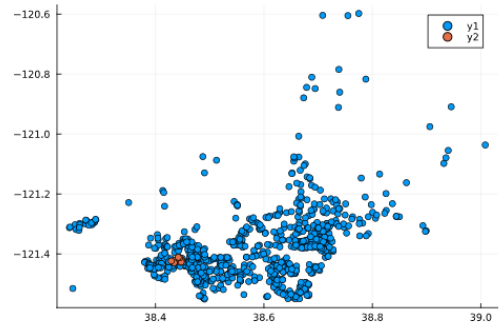


Рис. 3.15: Кластеризация данных. Метод k ближайших соседей. Часть 1

```
# 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"
```

Рис. 3.16: Кластеризация данных. Метод k ближайших соседей. Часть 2

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
  90895
  91002
  94905
  98937
 100309
 106250
 106852
      ⋮
 224252
 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:
```

Рис. 3.17: Кластеризация данных. Метод главных компонент. Часть 1

```
# 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,:])
```

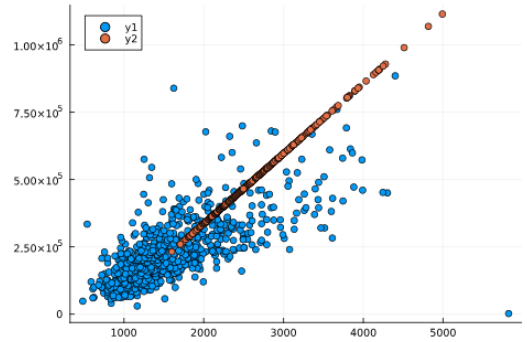
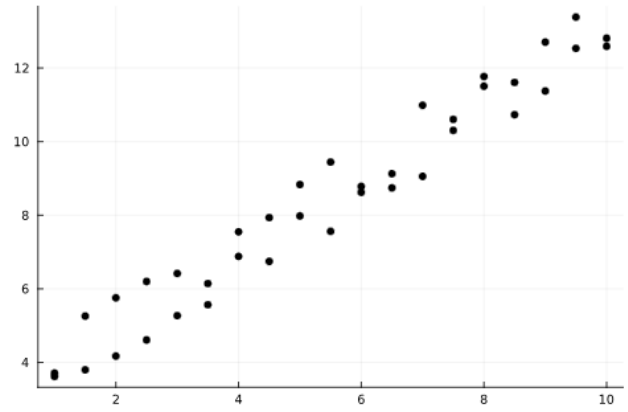


Рис. 3.18: Кластеризация данных. Метод главных компонент. Часть 2

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)
```

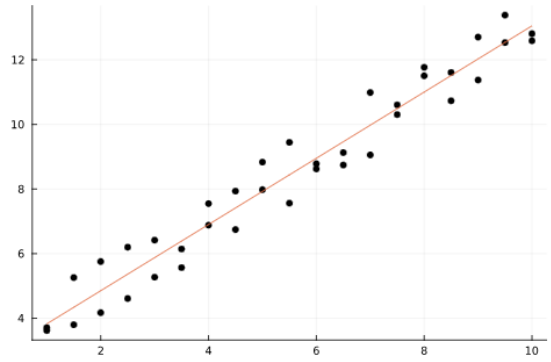



```
function find_best_fit(xvals,yvals)
meanx = mean(xvals)
meany = mean(yvals)
stdx = std(xvals)
stdy = std(yvals)
r = cor(xvals,yvals)
a = r*stdy/stdx
b = meany - a*meanx
return a,b
end
```

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

Рис. 3.19: Обработка данных. Линейная регрессия. Часть 1

```
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`
```

Рис. 3.20: Обработка данных. Линейная регрессия. Часть 2

```
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)
```

Рис. 3.21: Обработка данных. Линейная регрессия. Часть 3

3.2 Самостоятельная работа

Примеры представлены на рис. 3.22 - 3.26.

№1. 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.

```
using RDatasets
iris = dataset("datasets", "iris") |> DataFrame
```

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
⋮	⋮	⋮	⋮	⋮	⋮
139	6.0	3.0	4.8	1.8	virginica
140	6.9	3.1	5.4	2.1	virginica
141	6.7	3.1	5.6	2.4	virginica
142	6.9	3.1	5.1	2.3	virginica
143	5.8	2.7	5.1	1.9	virginica
144	6.8	3.2	5.9	2.3	virginica
145	6.7	3.3	5.7	2.5	virginica

Рис. 3.22: Кластеризация. Часть 1

```
iris_spec = Dict("setosa"=>1, "versicolor"=>2, "virginica"=>3)
```

```
Dict{String, Int64} with 3 entries:
  "virginica"  => 3
  "setosa"     => 1
  "versicolor" => 2
```

```
iris[!,:5] = [iris_spec[item] for item in iris[!,:5]]
```

```
150-element Vector{Int64}:
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 ⋮
 3
 3
 3
 3
 3
 3
 3
 3
 3
 3
 3
 3
 3
```

```
X = Matrix(iris)'
```

```
5×150 adjoint(::Matrix{Float64}) with eltype Float64:
 5.1  4.9  4.7  4.6  5.0  5.4  4.6  5.0  …  6.8  6.7  6.7  6.3  6.5  6.2  5.9
 3.5  3.0  3.2  3.1  3.6  3.9  3.4  3.4     3.2  3.3  3.0  2.5  3.0  3.4  3.0
 1.4  1.4  1.3  1.5  1.4  1.7  1.4  1.5     5.9  5.7  5.2  5.0  5.2  5.4  5.1
 0.2  0.2  0.2  0.2  0.2  0.4  0.3  0.2     2.3  2.5  2.3  1.9  2.0  2.3  1.8
 1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0     3.0  3.0  3.0  3.0  3.0  3.0  3.0
```

```
# Задание количества кластеров:
k = length(unique(iris[!,:5]))
```

```
3
```

```
#Определение k - ср
C = kmeans(X,k)
```

```
KmeansResult{Matrix{Float64}, Float64, Int64}([5.915686274509803 5.005999999999999 6.622448979591835; 2.764705882352941 3.428000000000001 2.983673469387755; … ; 1.3333333333333333 0.2459999999999999 2.03
3, 3, 3, 3, 3], [0.01998000000001804, 0.2003800000000977, 0.1739800000001446, 0.2759800000000041, 0.03557999999999595, 0.4583800000000534, 0.1723800000000397, 0.0043800000011819, 0.6519799999999947,
815910034893, 0.3484997917534729, 0.21723448563899623, 0.6843773427738427, 0.15580591420243195, 0.4533569346105537, 0.8004997917534424], [51, 50, 49], [51, 50, 49], 87.22862785114079, 3, true)
```

```
# формирование фрейма
df = DataFrame(cluster = C.assignments,
    SepalLength = iris[!,:SepalLength],
    SepalWidth = iris[!,:SepalWidth],
    PetalWidth = iris[!,:PetalWidth],
```

Рис. 3.23: Кластеризация. Часть 2

```
clusters_f = plot(legend=false)
for i in 1:k
    clustered_houses = df[df[!,:cluster].==i,:]
    xvals = clustered_houses[!,:SepalWidth]
    yvals = clustered_houses[!,:SepalLength]
    scatter!(clusters_f, xvals,yvals,markersize=2)
end
```

```
xlabel!("SepalWidth")
xlabel!("SepalLength")
title!("Iris's color-coded by cluster")
display(clusters_f)
```

Рис. 3.24: Кластеризация. Часть 3

№2. Регрессия (метод наименьших квадратов в случае линейной регрессии)

```
# Часть 1
X = randn(1000, 3)
a0 = rand(3)
y = X * a0 + 0.1 * randn(1000);
# Часть 2
X = rand(100);
y = 2X + 0.1 * randn(100);
```

Часть 1 Пусть регрессионная зависимость является линейной. Матрица наблюдений факторов X имеет размерность $N \times 3$ randn (N, 3), массив результатов $N \times 1$, регрессионная зависимость является линейной. Найдите МНК-оценку для линейной модели. – Сравните свои результаты с результатами использования llsq из MultivariateStats.jl (просмотрите документацию). – Сравните свои результаты с результатами использования регулярной регрессии наименьших квадратов из GLM.jl. Подсказка. Создайте матрицу данных X2, 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 ablinе!. Добавьте заголовок «График регрессии» и подпишите оси x и y

```
# Часть 1
X = randn(1000, 3)
a0 = rand(3)
y = X * a0 + 0.1 * randn(1000);
scatter(X,y,vals=(1,2,3))
```

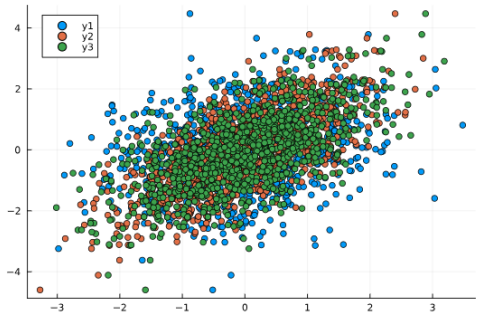


Рис. 3.25: Регрессия. Часть 1

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

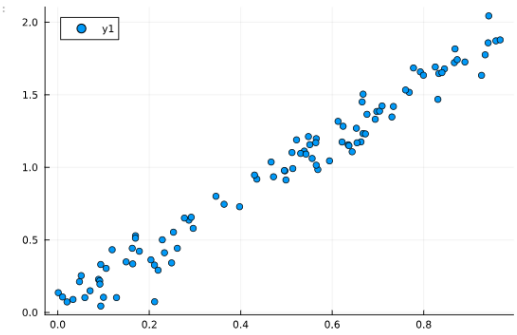


Рис. 3.26: Регрессия. Часть 2

4 Выводы

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