РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 7

дисциплина: Компьютерный практикум

по математическому моделированию

Студент: Ли Тимофей Александрович

Группа: НФИбд-01-18

МОСКВА

2021 г.

Постановка задачи

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

Выполнение работы

Сначала выполнил все примеры к лабораторной работе №7:

1. Ознакомился со считыванием 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
```

```
P = CSV.File("programminglanguages.csv") |> DataFrame
```

8 rows × 2 columns

	year	language
	Int64	String7
1	1950	basic
2	1951	fortran
3	1952	pascal
4	1953	c
5	1954	java
6	1955	prolog
7	1956	r
8	1957	python

```
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)
```

```
language_created_year(P,"python")
```

```
1957
```

```
language_created_year(P,"prolog")
```

```
1955
```

```
language_created_year(P,"Prolog")
```

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

```
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)
```

```
language_created_year_v2(P,"Prolog")
```

```
1955
```

```
Tx = readdlm("programminglanguages.csv", ',')
```

```
9×1 Matrix{Any}:
 "year;language"
 "1950;basic"
 "1951;fortran"
 "1952;pascal"
 "1953;c"
 "1954;java"
 "1955;prolog"
 "1956;r"
 "1957;python"
```

2. Затем ознакомился с записью данных в файл.

```
In [11]: CSV.write("programming_languages_data2.csv", P)
Out[11]: "programming_languages_data2.csv"

In [12]: writedlm("programming_languages_data.txt", Tx, ',')

In [13]: # Пример записи данных в текстовый файл с разделителем '-':
         writedlm("programming_languages_data2.txt", Tx, '-')

In [14]: P_new_delim = readdlm("programming_languages_data2.txt", '-')
Out[14]: 9×1 Matrix{Any}:
          "year;language"
          "1950;basic"
          "1951;fortran"
          "1952;pascal"
          "1953;c"
          "1954;java"
          "1955;prolog"
          "1956;r"
          "1957;python"
```

3. Далее ознакомился со словарями.

```
In [15]: dict = Dict{Integer,Vector{String}}()
         dict2 = Dict()
```

```
Out[15]: Dict{Any, Any}()
```

```
In [16]: 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
```

```
In [18]: dict[1954]
```

```
Out[18]: 1-element Vector{String}:
          "java"
```

4. Ознакомился с DataFrames

```
using DataFrames
```

```
df = DataFrame(year = P[:,1], language = P[:,2])
```

8 rows × 2 columns

	year	language
	Int64	String7
1	1950	basic
2	1951	fortran
3	1952	pascal
4	1953	c
5	1954	java
6	1955	prolog
7	1956	r
8	1957	python

```
df[!,:year]
```

```
8-element Vector{Int64}:
 1950
 1951
 1952
 1953
 1954
 1955
 1956
 1957
```

```
describe(df)
```

2 rows × 7 columns

	variable	mean	min	median	max	nmissing	eltype
	Symbol	Union...	Any	Union...	Any	Int64	DataType
1	year	1953.5	1950	1953.5	1957	0	Int64
2	language		basic		r	0	String7

5. Ознакомился с RDatasets

```
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
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

```
typeof(iris)
```

```
DataFrame
```

```
describe(iris)
```

5 rows × 7 columns

	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}

6. Ознакомился с работой с missing переменными

```
a = missing
typeof(a)
```

```
Missing
```

```
a+1
```

```
missing
```

```
# 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
```

```
typeof(calories)
```

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

```
using Statistics
```

```
mean(calories)
```

```
missing
```

```
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 rows × 3 columns

	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

7. Ознакомился с FileIO

```
using FileIO
```

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

```
   Resolving package versions...
   Installed Imath_jll ──────── v3.1.2+0
   Installed AbstractFFTs ────── v1.0.1
```

```
X1 = load("julialogo.png")
```

```
776×1200 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)
 ⋮                            ⋱
```


```
@show typeof(X1);
@show size(X1);
```

```
typeof(X1) = Matrix{ColorTypes.RGBA{FixedPointNumbers.N0f8}}
size(X1) = (776, 1200)
```

8. Ознакомился с кластеризацией методом k-means

```
import Pkg
Pkg.add("DataFrames")
Pkg.add("Statistics")
using DataFrames
using CSV
import Pkg
Pkg.add("Plots")
```

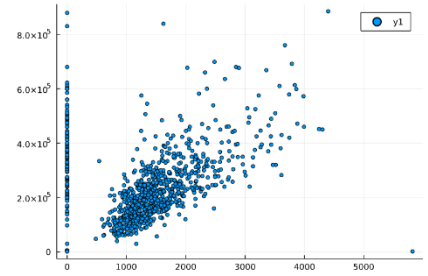
```
   Resolving package versions...
  No Changes to `C:\Users\Xiaomi\.julia\environments\v1.6\Project.toml`
  No Changes to `C:\Users\Xiaomi\.julia\environments\v1.6\Manifest.toml`
   Resolving package versions...
  No Changes to `C:\Users\Xiaomi\.julia\environments\v1.6\Project.toml`
  No Changes to `C:\Users\Xiaomi\.julia\environments\v1.6\Manifest.toml`
   Resolving package versions...
  No Changes to `C:\Users\Xiaomi\.julia\environments\v1.6\Project.toml`
  No Changes to `C:\Users\Xiaomi\.julia\environments\v1.6\Manifest.toml`
```

```
houses = CSV.File("housess.csv") |> DataFrame
```

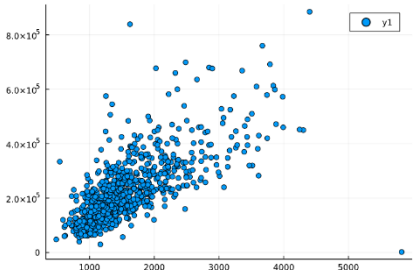
985 rows × 12 columns (omitted printing of 6 columns)

	street	city	zip	state	beds	baths
	String83	String15	Int64	String3	Int64	Int64
1	3526 HIGH ST	SACRAMENTO	95838	CA	2	1
2	51 OMAHA CT	SACRAMENTO	95823	CA	3	1
3	2796 BRANCH ST	SACRAMENTO	95815	CA	2	1
4	2805 JANETTE WAY	SACRAMENTO	95815	CA	2	1

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




```
filter_houses = houses[houses[!,:sq__ft].>0,:]
# Построение графика:
x = filter_houses[!,:sq__ft]
y = filter_houses[!,:price]
scatter(x,y)
```



```
using Statistics
# Определение средней цены для определённого типа домов:
combine(filter_houses,:type,filter_houses->mean(filter_houses[!,:price]))
```

814 rows × 2 columns

	type	x1
	String15	Float64
1	Residential	2.29448e5
2	Residential	2.29448e5
3	Residential	2.29448e5
4	Residential	2.29448e5
5	Residential	2.29448e5

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

814 rows × 5 columns

	cluster	city	latitude	longitude	zip
	Int64	String15	Float64	Float64	Int64
1	60	SACRAMENTO	38.6319	-121.435	95838
2	42	SACRAMENTO	38.4789	-121.431	95823
3	53	SACRAMENTO	38.6183	-121.444	95815
4	60	SACRAMENTO	38.6168	-121.439	95815
5	11	SACRAMENTO	38.5195	-121.436	95824

```
import Pkg
Pkg.add("Clustering")
using Clustering
```

```
   Resolving package versions...
```

```
# Добавление данных :Latitude и :longitude в новый фрейм:
X = filter_houses[!, [:latitude,:longitude]]
# Конвертация данных в матричный вид:
X = Matrix(X)
```

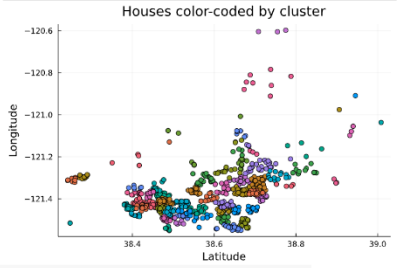
```
814×2 Matrix{Float64}:
 38.6319  -121.435
 38.4789  -121.431
 38.6183  -121.444
 38.6168  -121.439
 38.5195  -121.436
 38.6626  -121.328
 38.6817  -121.352
 38.5351  -121.481
 38.6212  -121.271
 38.7009  -121.443
 38.6377  -121.452
 38.4707  -121.459
 38.6187  -121.436
```

```
X=X'
```

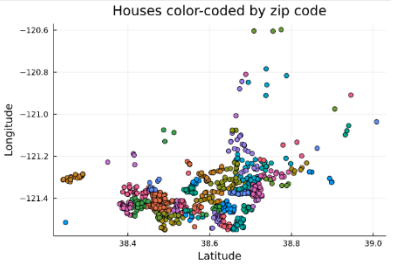
```
2×814 adjoint(::Matrix{Float64}) with eltype Float64:
   38.6319    38.4789    38.6183  …    38.7088    38.417    38.6552
 -121.435   -121.431   -121.444       -121.257   -121.397  -121.076
```

```
k = length(unique(filter_houses[!,:zip]))
C = kmeans(X,k)
```

```
KmeansResult(Matrix(Float64), Float64, Int64)([38.56742925 38.490447 … 38.53068435714287 38.69875490476191; -121.49139358333333
-121.129337 … -121.46543535714284 -121.37119928571428], [60, 42, 53, 60, 11, 4, 21, 29, 26, 33  …  6, 39, 42, 14, 43, 54, 59, 2
8, 30, 13], [6.734555427101441e-5, 0.0003736314138222951 4, 0.0001296234331675805 2, 0.0001724666290101595 2, 0.0001553982801851816
5, 0.0001611708285054192, 0.0002536611209507 1003, 0.0001470481474825646 7, 6.344488792819902e-5, 0.0001168665839941240 8  …  5.7
8467079321853 8e-5, 5.241325197857804 6e-5, 9.498398867435753e-6, 0.0001942098642757628, 0.0001341583774774353 3, 0.0001443395158 2
75962, 7.579611156849518e-5, 8.884414273779859e-5, 8.884414273779859e-5, 0.0003106035817845411, 0.0002109576598918783], [12, 1, 11, 18, 12, 28, 8, 1
9, 14, 28  …  6, 3, 17, 18, 6, 4, 14, 1, 14, 21], [12, 1, 11, 18, 12, 28, 8, 19, 14, 28  …  6, 3, 17, 18, 6, 4, 14, 1, 14, 21],
0.2031817762435821 4, 16, true)
```




```
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)
```



9. Ознакомился с кластеризацией методом k ближайших соседей

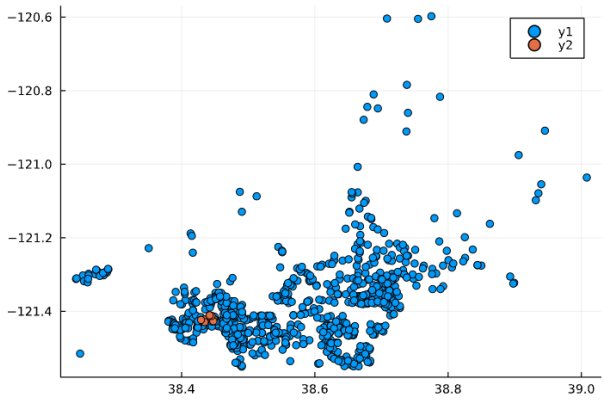
```
import Pkg
Pkg.add("NearestNeighbors")
using NearestNeighbors
```

```
   Resolving package versions...
    Updating `C:\Users\Xiaomi\.julia\environments\v1.6\Project.toml`
  [b8a86587] + NearestNeighbors v0.4.9
  No Changes to `C:\Users\Xiaomi\.julia\environments\v1.6\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)
# 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)
```




```
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"
```

10. Ознакомился с кластеризацией методом главных компонент

```
F = filter_houses[!,[:sq__ft,:price]]
F = Array(F)'
```

```
2×814 adjoint(::Matrix{Int64}) with eltype Int64:
   836   1167    796    852    797   1122  …   1477   1216   1685   1362
 59222  68212  68880  69307  81900  89921      234000  235000  235301  235738
```

```
convert(Array{Float64,2}, F)
```

```
2×814 Matrix{Float64}:
   836.0   1167.0    796.0    852.0    797.0  …   1216.0   1685.0   1362.0
 59222.0  68212.0  68880.0  69307.0  81900.0     235000.0  235301.0  235738.0
```

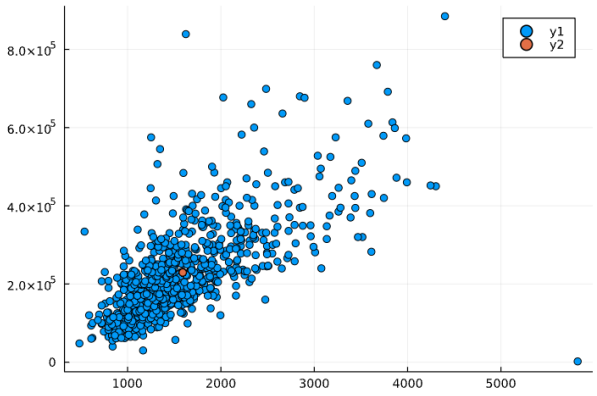
```
import Pkg
Pkg.add("MultivariateStats")
using MultivariateStats
```

```
   Resolving package versions...
  No Changes to `C:\Users\Xiaomi\.julia\environments\v1.6\Project.toml`
  No Changes to `C:\Users\Xiaomi\.julia\environments\v1.6\Manifest.toml`
```

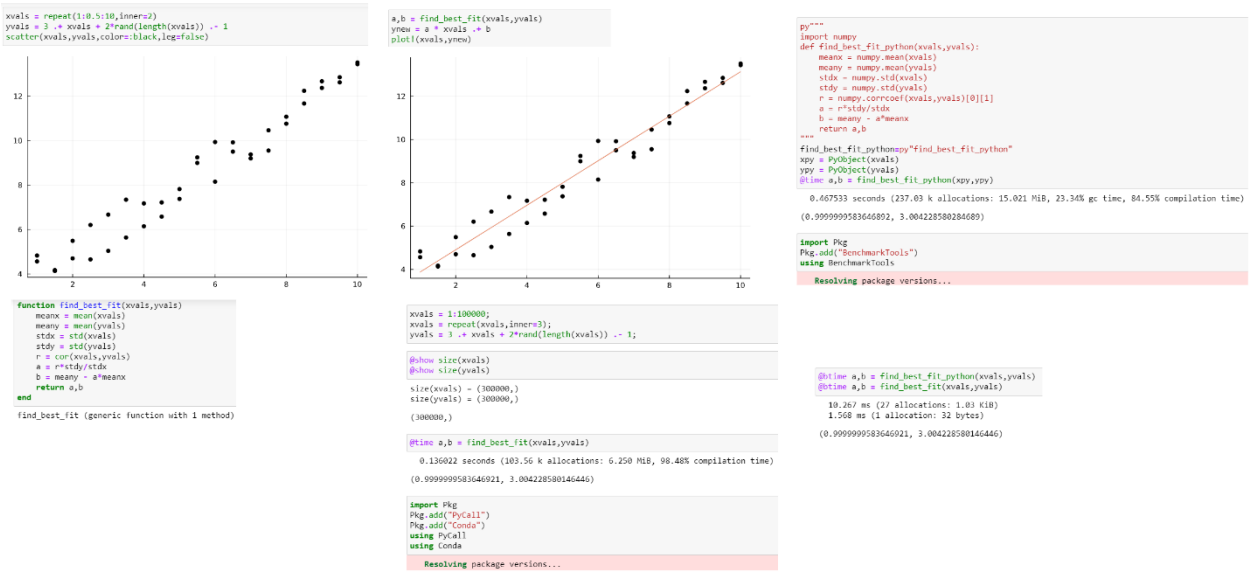
```
M = fit(PCA, F)
```

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

```
# 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,:])
```



11. Ознакомился с линейной регрессией



Далее выполнил поставленные задачи.

1. Кластеризовал набор данных iris методом k-means.

   Для этого загрузил датасет, проиндексировал его, преобразовал в массив и транспонировал. Определил количество кластеров и нашел k-среднее. Затем, создал новый датафрейм из изначального с добавлением столбца, в котором хранится метка кластера. Далее просто построил точечную диаграмму кластеров.

```
using Clustering
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
5	5.0	3.6	1.4	0.2	setosa

```
X=iris[!,[:SepalLength,:SepalWidth]]
X=Matrix{Float64}(X)
X=X'
```

```
2×150 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
```

```
k = length(unique(iris[!,:Species]))
C = kmeans(X,k)
```

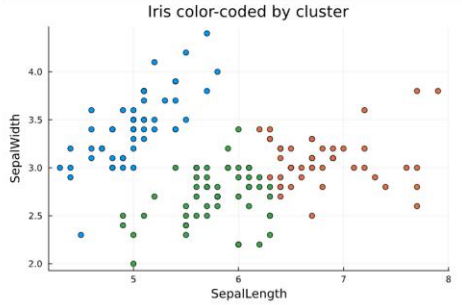
```
KmeansResult{Matrix{Float64}, Float64, Int64}([5.005999999999999 6.812765957446806 5.773584905660377; 3.428000000000001 3.07446
80851063824 2.692452830188679], [1, 1, 1, 1, 1, 1, 1, 1, 1, 1  …  2, 2, 3, 2, 2, 2, 3, 2, 2, 3], [0.01402000000000142, 0.19442
000000000803, 0.14562000000000808, 0.272420000000011, 0.029619999999994207, 0.37801999999999225, 0.1656199999999899, 0.00081999
9999990273, 0.64602, 0.11881999999999948  …  0.013368039837033052, 0.008261656858309152, 0.0007547169811203958, 0.0159212313263
87897, 0.06358080579444871, 0.018261656858300057, 0.3141509433962142, 0.10336803983703646, 0.48145314622001933, 0.11056603773583
618], [50, 47, 53], [50, 47, 53], 37.05070212765943 4, 10, true)
```

```
df=DataFrame(cluster=C.assignments, SepalLength=iris[!,:SepalLength],
    SepalWidth=iris[!,:SepalWidth], Species=iris[!,:Species])
```

150 rows × 4 columns

	cluster	SepalLength	SepalWidth	Species
	Int64	Float64	Float64	Cat...
1	1	5.1	3.5	setosa
2	1	4.9	3.0	setosa
3	1	4.7	3.2	setosa
4	1	4.6	3.1	setosa
5	1	5.0	3.6	setosa

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



2. Далее выполнил задание с регрессией (методом наименьших квадратов в случае линейной регрессии).

   Ввел начальные данные и написал функцию для метода наименьших квадратов в случае линейной регрессии. Вектор x – решение СУР. Далее нашел МНК-оценку модели и сравнил ее с результатами функций из пакетов MultivariateStats и GLM. Результаты сошлись.

   Для второй части нашел коэффициенты линейной регрессии с помощью моей функции и построил точечный график, добавив линию регрессии с помощью abline!

```
X = randn(1000, 3)
a0 = rand(3)
y = X * a0 + 0.1 * randn(1000);
```

```
function mnklin(X,y)
    x=fill(1.0,size(X)[1])
    X2=[x X]
    x=X2\y
    return x
end
```

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

```
a=mnklin(X,y)
```

```
4-element Vector{Float64}:
 -0.00319742817960049 03
  0.7020412149304233
  0.37818205181339587
  0.2488419694557077
```

```
using MultivariateStats
import Pkg
Pkg.add("GLM")
using GLM
```

```
  Resolving package versions...
```

```
MultivariateStats.llsq(X,y)
```

```
4-element Vector{Float64}:
  0.7020412149304235
  0.37818205181339587
  0.2488419694557076
 -0.0031974281796004703
```

```
X1=X[:,1]
X2=X[:,2]
X3=X[:,3]
reg=DataFrame(y=y,x1=X1,x2=X2,x3=X3)
lm(@formula(y~x1+x2+x3),reg)
```

```
StatsModels.TableRegressionModel{LinearModel{GLM.LmResp{Vector{Float64}}, GLM.D
ted{Float64, Matrix{Float64}}}}, Matrix{Float64}}

y ~ 1 + x1 + x2 + x3

Coefficients:
```

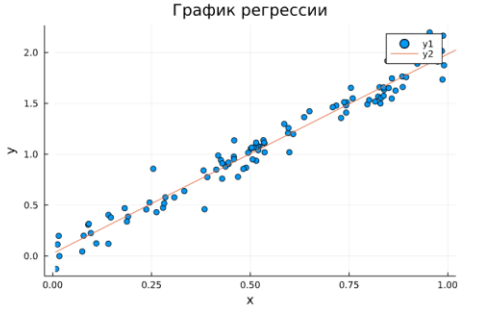
	Coef.	Std. Error	t	Pr(>|t|)	Lower 95%	Upper 95%
(Intercept)	-0.00319743	0.00313093	-1.02	0.3074	-0.0093414	0.00294655
x1	0.702041	0.00312824	224.42	<1e-99	0.695903	0.70818
x2	0.378182	0.0031445	120.27	<1e-99	0.372011	0.384353
x3	0.248842	0.00302775	82.19	<1e-99	0.2429	0.254783

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

```
a=mnklin(X,y)
b=a[2]
c=a[1]
b,c
```

```
(1.960615878718642, 0.02412528002865031)
```

```
scatter(X,y)
Plots.abline!(b,c, title="График регрессии", xlabel="x", ylabel="y")
```



3. Далее выполнил задания с моделью ценообразования биномиальных опционов.

   Ввел заданные начальный данные и построил траекторию курса акций.

   Затем, написанный цикл внес в функцию и получил функцию, которая создает траекторию курса с учетом начальных данных. Запустил ее 10 раз в цикле и вывел на один график все 10 траекторий.

Далее распараллелил цикл отрисовки траекторий с помощью Threads.@threads

```
S=100
T=1
n=10000
h=T/n
sigma=0.3
r=0.08
u=exp(r*h + sigma*sqrt(h))
d=exp(r*h - sigma*sqrt(h))
p=(exp(r*h)-d)/(u-d)
```

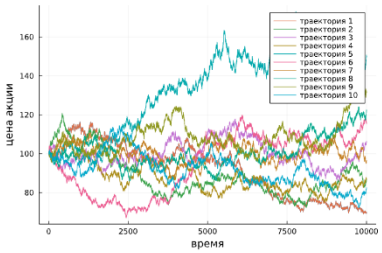
```
0.4992500005625153
```

```
line=[]
c=0
append!(line,S)
for i in 1:n
    k=rand()
    if k<p
        append!(line,S*u^(i-c)*d^c)
    else
        append!(line,S*u^(i-c-1)*d^(c+1))
        c+=1
    end
end
plot(line, title="траектория курса акций", xlabel="время", ylabel="цена акции")
```


```
function CreatePath(S,r,sigma,T,n)
    h=T/n
    u=exp(r*h + sigma*sqrt(h))
    d=exp(r*h - sigma*sqrt(h))
    p=(exp(r*h)-d)/(u-d)
    line=[]
    c=0
    append!(line,S)
    for i in 1:n
        k=rand()
        if k<p
            append!(line,S*u^(i-c)*d^c)
        else
            append!(line,S*u^(i-c-1)*d^(c+1))
            c+=1
        end
    end
    return line
end
```

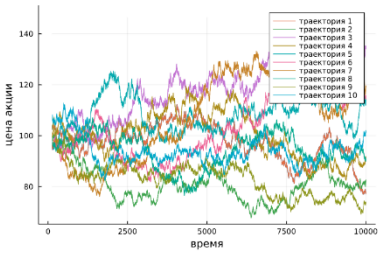
```
CreatePath (generic function with 1 method)
```

```
for i in 1:10
    IJulia.clear_output(true)
    path=CreatePath(100,0.08,0.3,1,10000)
    if i==1
        p=plot(path, label=false)
    end
    p=plot!(path, label="траектория $i", xlabel="время", ylabel="цена акции")
    display(p)
end
```




```
using Base.Threads
```

```
Threads.@threads for i in 1:10
    IJulia.clear_output(true)
    path=CreatePath(100,0.08,0.3,1,10000)
    if i==1
        p=plot(path, label=false)
    end
    p=plot!(path, label="траектория $i", xlabel="время", ylabel="цена акции")
    display(p)
end
```



Выводы

Изучил специализированные пакеты Julia для обработки данных.