

Компьютерный практикум по статистическому анализу данных

Лабораторная работа № 7. Введение в работу с данными

Сунгурова Мариян М.

Российский университет дружбы народов, Москва, Россия

Информация

- Сунгурова Мариян Мухсиновна
- студентка группы НКНбд-01-21
- Российский университет дружбы народов

Вводная часть

Цель работы

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

Задачи

1. Используя Jupyter Lab, повторите примеры.
2. Выполните задания для самостоятельной работы.

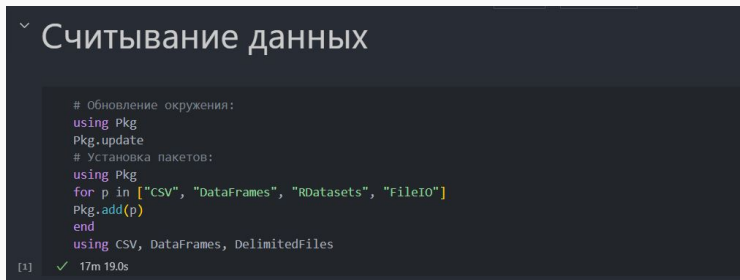
Теоретическое введение

Julia — высокоуровневый свободный язык программирования с динамической типизацией, созданный для математических вычислений. Эффективен также и для написания программ общего назначения. Синтаксис языка схож с синтаксисом других математических языков, однако имеет некоторые существенные отличия.

Для выполнения заданий была использована официальная документация Julia.

Выполнение лабораторной работы

Выполним примеры из лабораторной работы для изучения циклов и функций(рис. (fig:001?) - (fig:015?))



^ 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
```

[1] ✓ 17m 19.0s

Рис. 1: Примеры Считывание данных

Выполнение лабораторной работы



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

# 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

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

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

[4] ✓ 0.3s

... 2012

```
# 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

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

[5] ✓ 0.1s

✓ Запись данных в файл

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

[8] ✓ 0.0s

```
# Построчное считывание данных с указанием разделителя:  
P_new_delim = readlm("programming_languages_data2.txt", '-')  
[1]
```

Рис. 3: Примеры Запись данных в файл

Словари

```
# Инициализация словаря:
dict = Dict{Integer,Vector{String}}{()}
# Инициализация словаря:
dict2 = Dict{Integer,Vector{String}}{()}

# 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]
```

[10] ✓ 0.7s

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

Рис. 4: Примеры Словари

```
✓ DataFrames

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

# Вывод всех значения столбца year:
df[:,year]

[12] ✓ 0.3s
... 73-element Vector{Int64}:
```

Рис. 5: Примеры DataFrames

RDatasets

```
# С данными можно работать также как с наборами данных через пакет RDatasets
# языка R:
# Подгружаем пакет RDatasets:
using RDatasets
# Задаём структуру данных в виде набора данных:
iris = dataset("datasets", "iris")
# Определения типа переменной:
typeof(iris)
describe(iris)
```

[15] ✓ 1.0s

... 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}

Рис. 6: Примеры RDatasets

Работа с переменными отсутствующего типа (Missing Values)

```
# Пакет DataFrames позволяет использовать так называемый «отсутствующий» тип:  
# Отсутствующий тип:  
a = missing  
typeof(a)  
[16] ✓ 0.3s  
... Missing
```

```
# Пример операции с переменной отсутствующего типа:  
a + 1  
[17] ✓ 0.1s  
... missing
```

```
# Предположим есть перечень продуктов, для которых заданы калории:  
# Определение перечня продуктов:  
foods = ["apple", "cucumber", "tomato", "banana"]  
# Определение калорий:  
calories = [missing, 47, 22, 105]  
# Определение типа переменной:  
typeof(calories)  
[20] ✓ 0.0s  
... Vector{Union{Missing, Int64}} (alias for Array{Union{Missing, Int64}, 1})
```

```
# Подключаем пакет Statistics:  
using Statistics  
# Определение среднего значения:  
mean(calories)  
[21] ✓ 0.1s  
... missing
```

Рис. 7: Примеры Missing type

FileIO

```
# Подключаем пакет FileIO:
using FileIO
# Попробуем посмотреть, как Julia работает с изображениями.
# Подключим соответствующий пакет:
# Подключаем пакет ImageIO:
import Pkg
Pkg.add("ImageIO")
# Загрузим изображение (в данном случае логотип Julia):
# Загрузка изображения:

[ ]

X1 = load("1.JPG")
# Julia хранит изображение в виде множества цветов:
# Определение типа и размера данных:
@show typeof(X1);
@show size(X1);

[32] ✓ 1.7s

... typeof(X1) = Matrix{ColorTypes.RGB{FixedPointNumbers.N0f8}}
size(X1) = (865, 1163)
```

Рис. 8: Примеры Fileio

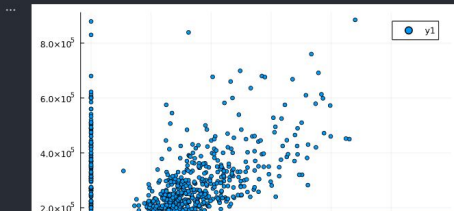
Кластеризация данных. Метод k-средних

```
# Загрузка пакетов:
import Pkg
Pkg.add("DataFrames")
Pkg.add("Statistics")
using DataFrames
using CSV
import Pkg
Pkg.add("Plots")

# Затем загрузим данные:
# Загрузка данных:
houses = CSV.File("houses.csv") |> DataFrame
# Построение графика:
using Plots
plot(size=(500,500),leg=false)
x = houses[:,sq_ft]
y = houses[:,price]
scatter(x,y,markersize=3)
```

[35] ✓ 9.7s

... WARNING: using Plots.Formatted in module Main conflicts with an existing identifier.



Выполнение лабораторной работы

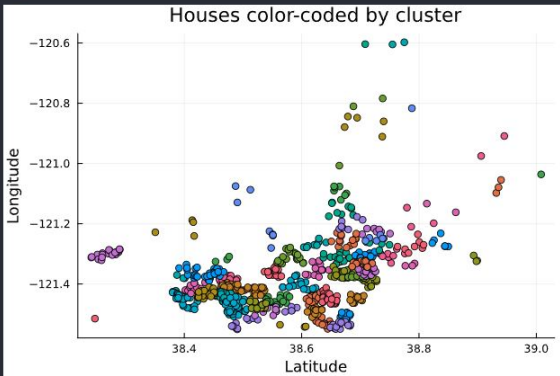


Выполнение лабораторной работы

```
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)
```

[49]

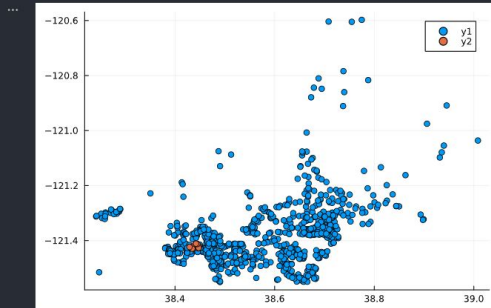
⊗ 0.9s



Выполнение лабораторной работы

```
▶
knearest = 10
id = 70
point = X[:,id]
# Поиск ближайших соседей:
kdtree = KDTree(X)
idxs, dists = knn(kdtree, point, knearest, true)
# Отобразим на графике соседей выбранного объекта недвижимости (рис. 7.5):
# 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)
```

[51] ✓ 0.3s



Обработка данных. Метод главных компонент

[+ Code](#)

```
# Фрейм с указанием площади и цены недвижимости:  
F = filter_houses[, [:sq_ft,:price]]  
# Конвертация данных в массив:  
F = Array(F)'
```

[]

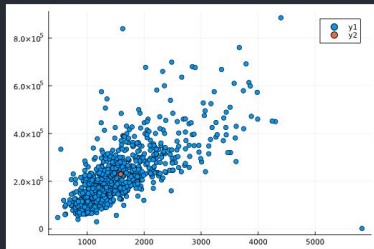
```
# Подключение пакета MultivariateStats:  
import Pkg  
Pkg.add("MultivariateStats")  
using MultivariateStats
```

[]

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

(66) ✓ 32s

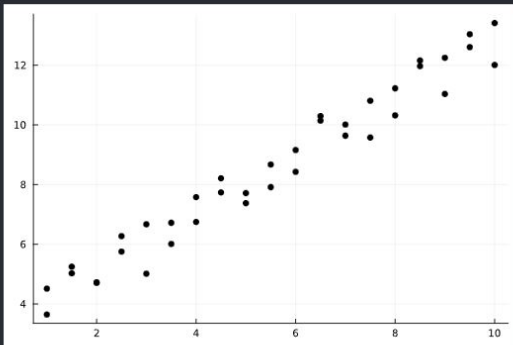
...



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)
```

61] ✓ 0.4s



Выполнение лабораторной работы

```
# Определим функцию линейной регрессии:
```

```
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
```

```
# Применим функцию линейной регрессии для построения соответствующего графика
```

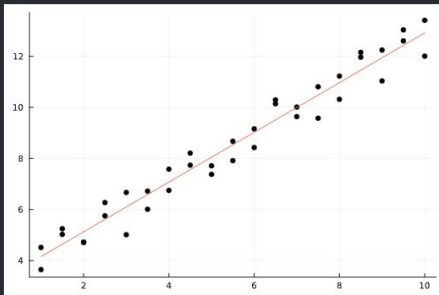
```
# значений (рис. 7.8):
```

```
a,b = find_best_fit(xvals,yvals)
```

```
ynew = a * xvals .+ b
```

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

[63] ✓ 0.1s



Выполнение лабораторной работы

Затем выполним задания(рис. (fig:016?) - (fig:015?))

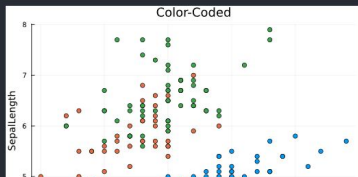
```
using RDatasets
iris = dataset("datasets", "iris") |> DataFrame
iris_spec = Dict{"setosa"=>1, "versicolor"=>2, "virginica"=>3}
iris[1,:5] = [iris_spec[item] for item in iris[1,:5]]

X = Matrix(iris)'

#Заделим число кластеров
k = length(unique(iris[1,:5]))
# Определение k-среднего
C = kmeans(X,k)

# DataFrame
df = DataFrame(cluster = C.assignments, SepalLength = iris[1,:SepalLength],
SepalWidth = iris[1,:SepalWidth], PetalLength = iris[1,:PetalLength],
PetalWidth = iris[1,:PetalWidth], Species = iris[1,:Species]);

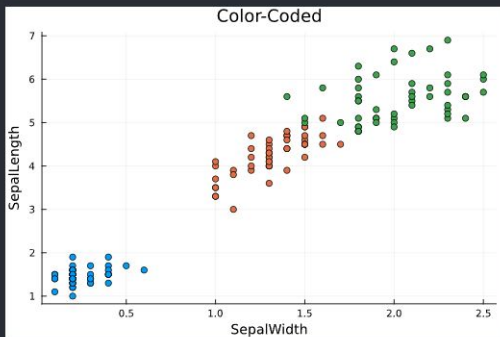
cluster_fig = plot(legend = false)
for i=1:k
    clustered_houses = df[df[1,:cluster].==i, :]
    xvals = clustered_houses[1,:SepalWidth]
    yvals = clustered_houses[1,:SepalLength]
    scatter!(cluster_fig, xvals,yvals,markersize=4)
end
xlabel!("SepalWidth")
ylabel!("SepalLength")
title!("Color-Coded")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



Выполнение лабораторной работы

```
cluster_fig = plot(legend = false)
for i=1:k
    clustered_houses = df[df[:, :cluster].==i, :]
    xvals = clustered_houses[:, :PetalWidth]
    yvals = clustered_houses[:, :PetalLength]
    scatter!(cluster_fig, xvals, yvals, markersize=4)
end
xlabel!("SepalWidth")
ylabel!("SepalLength")
title!("Color-Coded")
display(cluster_fig)
```

✓ 0.1s



Регрессия (метод наименьших квадратов в случае линейной регрессии)

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

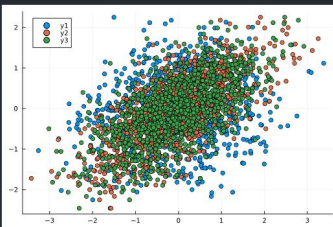


Рис. 18: Задания 2

Выводы

В результате выполнения данной лабораторной работы было освоено использование специализированных пакетов Julia для обработки данных.