

Лабораторная работа №6

Леснухин Даниил Дмитриевич Российский университет
дружбы народов Москва

Цель работы

Основной целью работы является освоение специализированных пакетов Julia для обработки данных

Задание

- 1 Используя JupyterLab, повторите примеры, дополнив графики обозначениями осей координат, легендой и названиями графиков
- 2 Выполните задания для самостоятельной работы

Теоретическое введение

Julia — высокоуровневый язык программирования с динамической типизацией для математических вычислений.
Эффективен для программ общего назначения.
Использовалась официальная документация Julia.

Задание №1 — Кластеризация

Загрузим данные Iris и построим диаграммы кластеров

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

```
using Pkg
Pkg.update()

# Установка пакетов (если не установлены)
for p in ["CSV", "DataFrames", "RDatasets", "FileIO",
          "DelimitedFiles", "Clustering", "Plots", "Statistics"]
    Pkg.add(p)
end

using RDatasets
using Clustering
using Plots
using DataFrames
using Statistics

# =====
# Загрузка данных Iris
# =====
iris = dataset("datasets", "iris")

println("Первые 5 строк набора данных Iris:")
println(first(iris, 5))

# Матрица признаков (4 признака × 150 объектов)
X_iris = Matrix(iris[:, 1:4])
println("Размерность матрицы признаков: ", size(X_iris))

# =====
✓ 11m 37.8s
```

```
plot(p1, p2, layout = (1, 2), size = (900, 400), legend = :topleft)

# =====
# Сравнение кластеров с реальными видами
# =====
println("Сравнение кластеров:")
ct = combine(groupby(iris_with_clusters, [:Species, :Cluster]), nrow => :Count)
println(ct)

println("Центры кластеров:")
display(result.centers')

# =====
# Определение доминирующего вида в каждом кластере
# =====
cluster_species = combine(groupby(iris_with_clusters, :Cluster)) do df
    counts = combine(groupby(df, :Species), nrow => :Count)
    sort!(counts, :Count, rev = true)

    DataFrame(
        Cluster = first(df.Cluster),
        DominantSpecies = counts.Species[1],
        Count = nrow(df)
    )
end
```

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

Размерность матрицы признаков: (4, 150)

Сравнение кластеров:

5×3 DataFrame

Row	Species	Cluster	Count
	Cat...	Int64	Int64
1	setosa	2	50
2	versicolor	1	3
3	versicolor	3	47
4	virginica	1	36
5	virginica	3	14

Центры кластеров:

```
... 3x4 adjoint(::Matrix{Float64}) with eltype Float64:  
    6.85385  3.07692  5.71538  2.05385  
    5.006      3.428      1.462      0.246  
    5.88361  2.74098  4.38852  1.43443
```

... Доминирующие виды в каждом кластере:

... 3x3 DataFrame

Row	Cluster	DominantSpecies	Count
	Int64	Cat...	Int64
1	1	virginica	39
2	2	setosa	50
3	3	versicolor	61



Задание №2 — Линейная регрессия

Матрица наблюдений факторов размером N на 3, массив результатов размером N на 1. Найдем МНК-оценку для линейной модели

- Сравним с функцией `l1sq` из `MultivariateStats`
- Сравним с использованием регулярной регрессии наименьших квадратов из `GLM`

Создадим матрицу данных с добавленным столбцом единиц и решим систему линейных уравнений

```
using Pkg
Pkg.update()

# Установка пакетов (если не установлены)
for p in ["CSV", "DataFrames", "RDatasets", "FileIO",
          "DelimitedFiles", "Clustering", "Plots", "Statistics"]
    Pkg.add(p)
end

using RDatasets
using Clustering
using Plots
using DataFrames
using Statistics

Pkg.add("GLM")

using GLM

N = 1000
X = rand(N, 3)
a0 = rand(3)

y = X * a0 + 0.1 * randn(N)
```

```

y = X * a0 + 0.1 * randn(N)

X_with_intercept = hcat(ones(N), X)
B_manual = (X_with_intercept' * X_with_intercept) \ (X_with_intercept' * y)

df = DataFrame(X1 = X[:, 1], X2 = X[:, 2], X3 = X[:, 3], Y = y)
model = lm(@formula(Y ~ X1 + X2 + X3), df)
B_glm = coef(model)

println("Коэффициенты, найденные вручную: ", B_manual)
println("Коэффициенты, найденные с помощью GLM: ", B_glm)
println("Разница между коэффициентами: ", B_glm[2:end])

y_pred_manual = X_with_intercept * B_manual
mse_manual = mean((y - y_pred_manual).^2)
println("MSE для ручного решения: ", mse_manual)

```

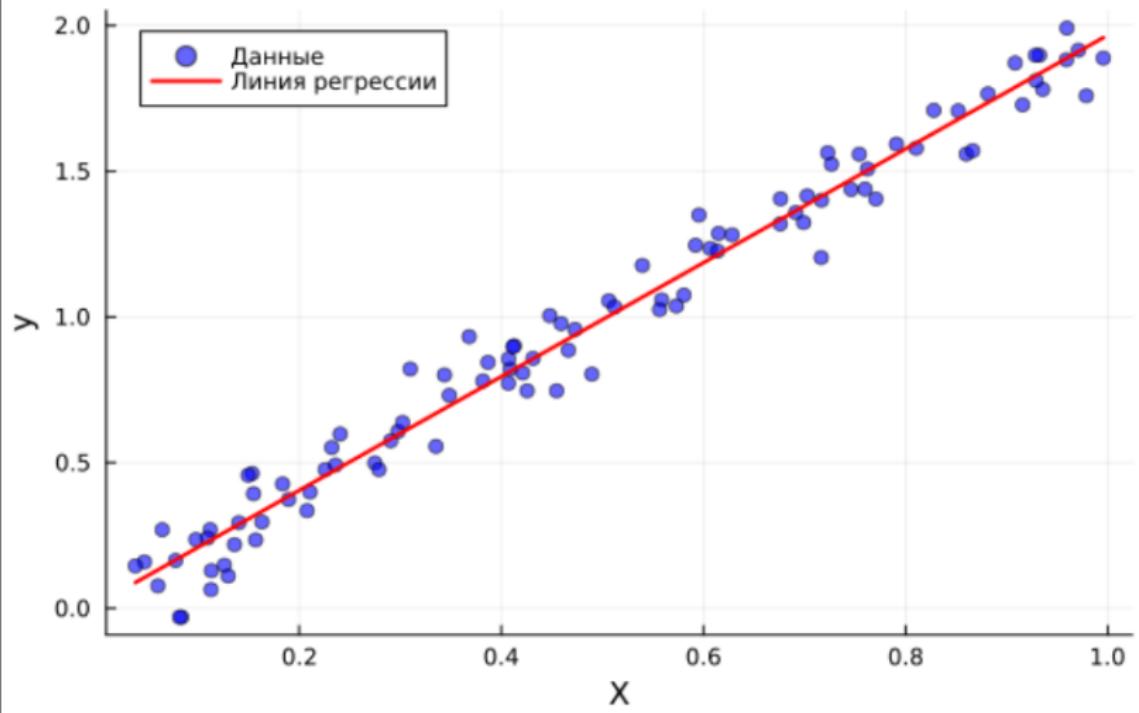
Output is truncated. View as a [scrollable element](#) or open in a [text editor](#). Adjust cell output [settings](#)...

Коэффициенты, найденные вручную: [0.016039051641520956, 0.2151320191113325, 0.13257289225109858, 0.18689555171543937]
Коэффициенты, найденные с помощью GLM: [0.01603905164152073, 0.2151320191113327, 0.1325728922510988, 0.18689555171543934]
Разница между коэффициентами: [0.2151320191113327, 0.1325728922510988, 0.18689555171543934]
MSE для ручного решения: 0.010556585871961083

Найдем линию регрессии и построим график с заголовком и подписями осей

Коэффициенты регрессии: $a = 1.9536027139429344$, $b = 0.013864760907581819$

График регрессии



Задание №3 — Модель ценообразования опционов

Построим траекторию возможных цен на акции с учетом начальных параметров

```
using Plots
using Random
using Distributed

function createPath(S::Float64, r::Float64, sigma::Float64, T::Float64
    h = T / n
    u = exp(r * h + sigma * sqrt(h))
    d = exp(r * h - sigma * sqrt(h))
    p_star = (exp(r * h) - d) / (u - d)

    path = zeros(n + 1)
    path[1] = S

    for i in 2:(n + 1)
        if rand() < p_star
            path[i] = path[i-1] * u
        else
            path[i] = path[i-1] * d
        end
    end

    return path
end

println("Задание а: Построение одной траектории")
S = 100.0
```

```
println("Задание а: Построение одной траектории")
S = 100.0
T = 1.0
n = 10000
sigma = 0.3
r = 0.08

path = createPath(S, r, sigma, T, n)
time_points = range(0, T, length = n + 1)

plot(time_points, path,
      title = "Биномиальная модель: траектория цены акции",
      xlabel = "Время (годы)",
      ylabel = "Цена акции",
      label = "Траектория цены",
      linewidth = 2,
      legend = :topleft)

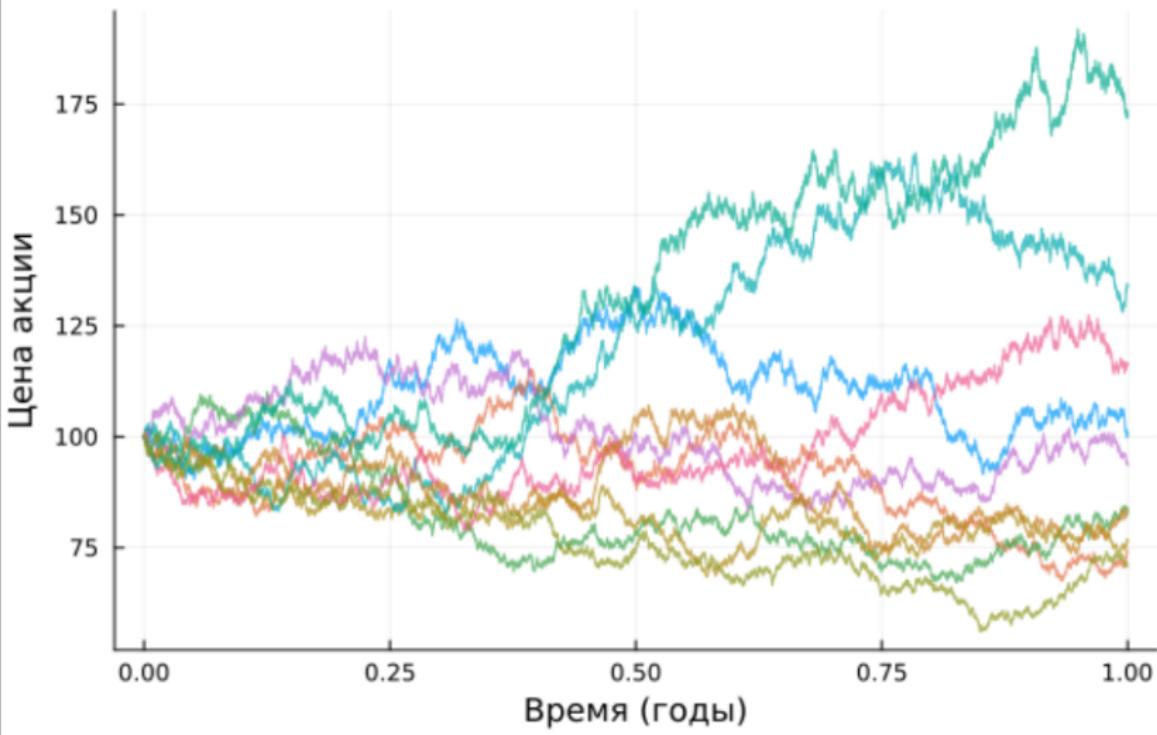
savefig("single_trajectory.png")
println("График сохранен как 'single_trajectory.png'")
```

```
println("\nЗадание б: Построение 10 траекторий")
time_points = range(0, T, length = n + 1)
plt = plot(title = "10 траекторий цен акций",
           xlabel = "Время (годы)",
           ylabel = "Цена акции",
           linewidth = 2)
```

7.5s

Создадим функцию `createPath` для генерации траектории цены акции. Построим 10 разных траекторий на одном графике

10 траекторий цен акций



Распараллелим генерацию траектории с использованием потоков и параллельных вычислений

Задание с: Тестирование параллельных методов

Доступное количество потоков: 1

Последовательная версия (50 траекторий):

0.035322 seconds (31.95 k allocations: 1.969 MiB, 94.51% compilation time)

Параллельная версия с Threads.@threads:

0.027088 seconds (12.13 k allocations: 1.002 MiB, 98.01% compilation time)

Задание d: Демонстрация модели

Демонстрационный график сохранен как 'demo_trajectory.png'

Статистика по сгенерированным траекториям:

Средняя конечная цена: 98.80017230409572

Минимальная конечная цена: 43.57284868796773

Максимальная конечная цена: 167.59737122187457

Стандартное отклонение: 32.06805411484571

Все задания выполнены успешно!

Выводы

- Освоены специализированные пакеты Julia для обработки данных
- Построены графики кластеризации, регрессии и финансовых моделей
- Получены навыки анализа и визуализации динамических систем