Лабораторная работа №7. Введение в работу с данными

Выполнила: Лебедева Ольга Андреевна

2024

Российский университет дружбы народов, Москва, Россия

Цель работы

Основной целью работы является специализированных пакетов Julia для обработки данных.

Задачи

- Освоить специализированные пакеты Julia для обработки данных.
- Научиться применять методы кластеризации, регрессии и анализа данных с использованием Julia.
- Ознакомиться с работой с файлами данных в формате CSV и структурой DataFrame.

Термины и условные обозначения

- DataFrame — структура данных, аналогичная таблицам в базах данных.
- Кластеризация — метод машинного обучения для группировки объектов на основе их характеристик.
- Линейная регрессия — метод для нахождения линейной зависимости между переменными.
- PCA (Principal Component Analysis) — метод снижения размерности данных.
- k-средние — алгоритм кластеризации данных на основе центроидов.

Объект и предмет исследования

Объект исследования: данные о недвижимости, включая цену, площадь, географическое расположение, а также данные о языках программирования.

Предмет исследования: применение алгоритмов обработки данных для анализа и визуализации.

Техническое оснащение

Программное обеспечение: Jupyter Notebook, язык программирования Julia с установленными библиотеками (CSV, DataFrames, Clustering, Plots и др.).

Методы: 1. Считывание данных из файлов CSV. Кластеризация данных методами k-средних и k ближайших соседей. 2. Применение PCA для снижения размерности. 3. Линейная регрессия для выявления зависимостей между переменными.

Теоретическое введение

Обработка и анализ данных, полученных в результате проведения исследований, — важная и неотъемлемая часть исследовательской деятельности. Большое значение имеет выявление определённых связей и закономерностей в имеющихся неструктурированных данных, особенно в данных больших размерностей. Выявленные в данных связей и закономерностей позволяет строить прогнозные модели с предполагаемым результатом. Для решения таких задач применяют методы из таких областей знаний как математическая статистика, программирование, искусственный интеллект, машинное обучение.

Задание лабораторной работы №7

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

Задание лабораторной работы №7

Задание См. рис. 1, См. рис. 2

7.4. Задания для самостоятельного выполнения

7.4.1. 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.

7.4.2. 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 `abline!`. Добавьте заголовок «График регрессии» и подпишите оси x и y.

Рис. 1: Задание_1

Задание лабораторной работы №7

7.4.3. Модель ценообразования биномиальных опционов

Описание модели ценообразования биномиальных опционов можно найти на стр. `https://en.wikipedia.org/wiki/Binomial_options_pricing_model`.

Постройте траекторию возможных цен на акции:

- S — начальная цена акции;
- T — длина биномиального дерева в годах;
- n — количество периодов;
- $h = Tn$ — длина одного периода;
- σ — волатильность акции;
- r — годовая процентная ставка;
- $u = \exp(rh + \sigma\sqrt{h})$;
- $d = \exp(rh - \sigma\sqrt{h})$;
- $p^* = \dfrac{\exp(rh) - d}{u - d}$.

a) Пусть $S = 100, T = 1, n = 10000, \sigma = 0.3$ и $r = 0.08$. Попробуйте построить траекторию курса акций. Функция `rand()` генерирует случайное число от 0 до 1. Вы можете использовать функцию построения графика из библиотеки графиков.

b) Создайте функцию `createPath(S :: Float64, r :: Float64, sigma :: Float64, T :: Float64, n :: Int64)`, которая создает траекторию цены акции с учетом начальных параметров. Используйте `createPath`, чтобы создать 10 разных траекторий и построить их все на одном графике.

c) Распараллельте генерацию траектории. Можете использовать `Threads.@threads`, `pmap` и `@parallel`.

d) Пусть $S = 100, T = 1, n = 10000, \sigma = 0.3$ и $r = 0.08$. Попробуйте построить траекторию курса акций. Функция `rand()` генерирует случайное число от 0 до 1. Вы можете использовать функцию построения графика из библиотеки графиков.

Рис. 2: Задание_2

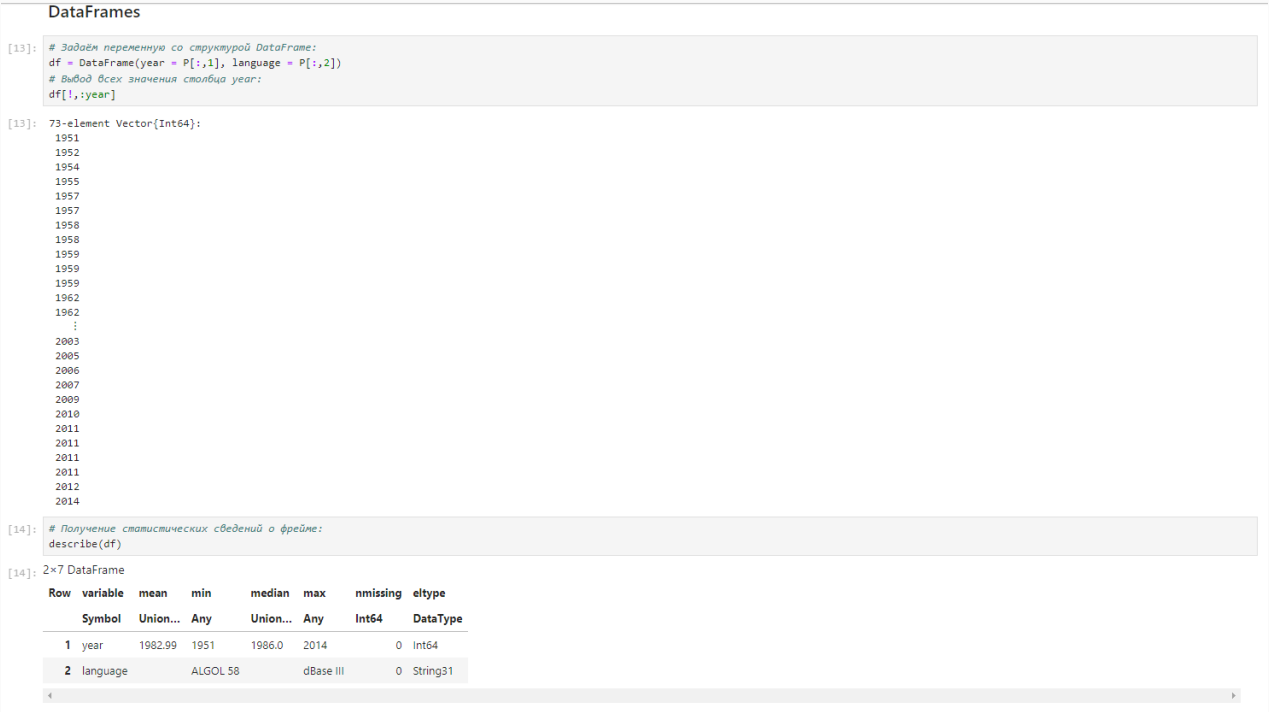
Выполнение лабораторной работы. Повторение примеров

Ниже приведены повторы примеров, данные в лабораторной работе для ознакомления: См. рис. 3, См. рис. 4, См. рис. 5, См. рис. 6.

```
1]: using CSV, DataFrames, DelimitedFiles

    # 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")

1]: 2012

2]: language_created_year(P,"julia")
```

Рис. 3: Повторение примеров_1)

Выполнение лабораторной работы. Повторение примеров



DataFrames

[13]:
```
# Задаём переменную со структурой DataFrame:
df = DataFrame(year = P[:,1], language = P[:,2])
# Вывод всех значения столбца year:
df[!,:year]
```

[13]:
```
73-element Vector{Int64}:
 1951
 1952
 1954
 1955
 1957
 1957
 1958
 1958
 1959
 1959
 1959
 1962
 1962
    ⋮
 2003
 2005
 2006
 2007
 2009
 2010
 2011
 2011
 2011
 2011
 2012
 2014
```

[14]:
```
# Получение статистических сведений о фрейме:
describe(df)
```

[14]: 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. 4: Повторение примеров_2

Выполнение лабораторной работы. Повторение примеров

RDatasets

[15]:
```
# Подгружаем пакет RDatasets:
using RDatasets
# Задаём структуру данных в виде набора данных:
iris = dataset("datasets", "iris")
```

[15]: 150×5 DataFrame — *125 rows omitted*

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. 5: Повторение примеров_3

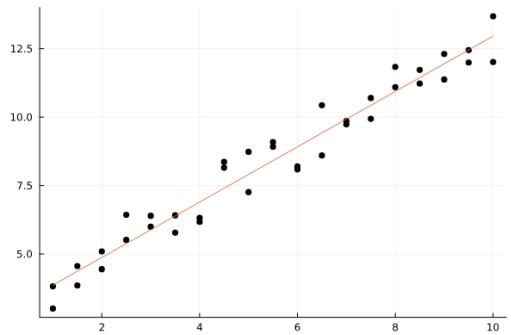
Выполнение лабораторной работы. Повторение примеров

[42]:
```
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
```

[42]:
```
find_best_fit (generic function with 1 method)
```

[43]:
```
a,b = find_best_fit(xvals,yvals)
ynew = a * xvals .+ b
plot!(xvals,ynew)
```

[43]:



[44]:
```
xvals = 1:100000;
xvals = repeat(xvals,inner=3);
yvals = 3 .+ xvals + 2*rand(length(xvals)) .- 1;
@show size(xvals)
@show size(yvals)
```

Рис. 6: Повторение примеров_4

Самостоятельная работа

Выполним задание 7.4.1. Кластеризация: См. рис. 7.

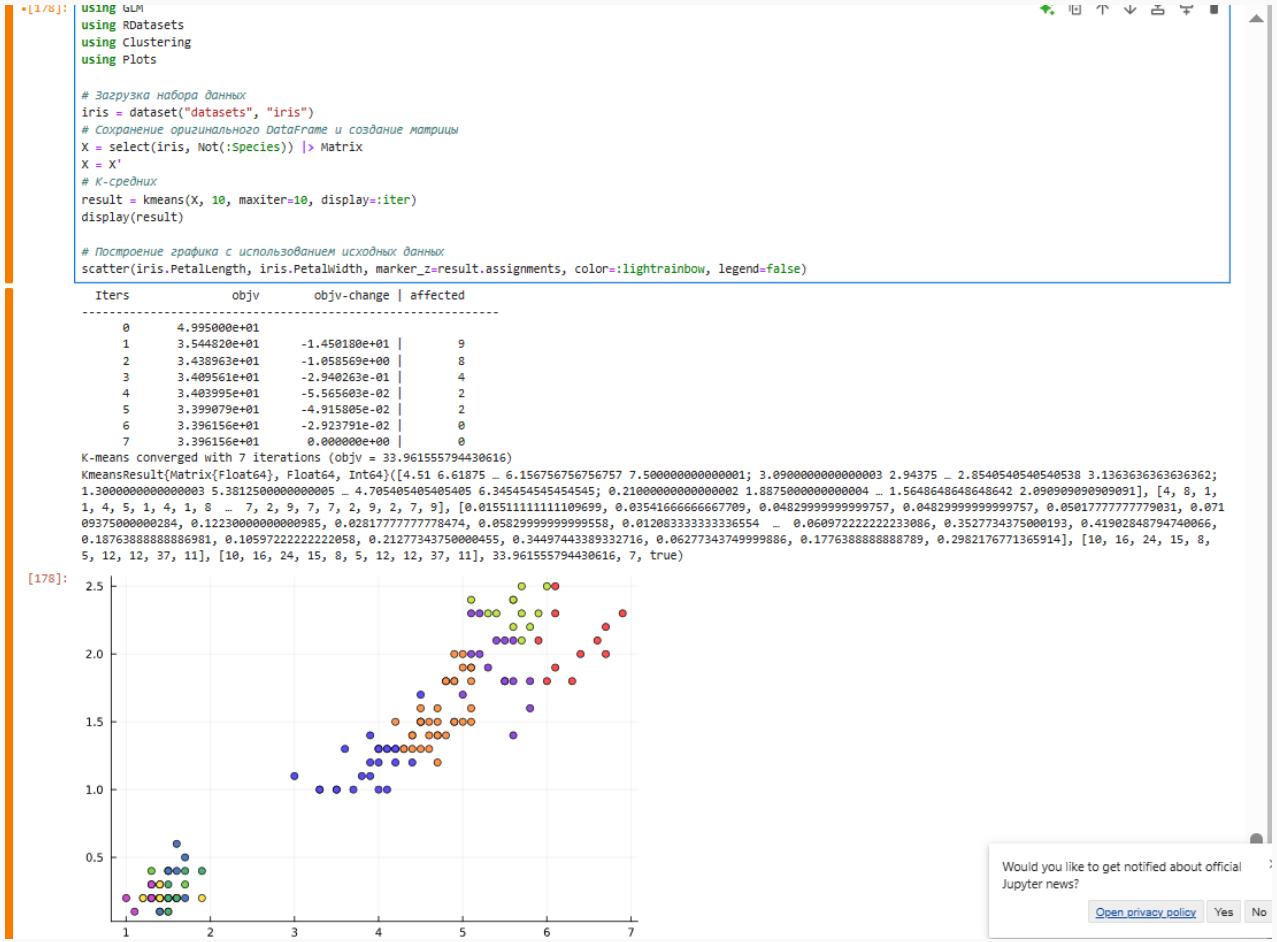


Рис. 7: Самостоятельная работа_1

Самостоятельная работа

Была выполнена кластеризация данных из набора iris с использованием языка Julia. Для этого данные были загружены с помощью библиотеки RDatasets, удалён столбец Species, а оставшиеся признаки преобразованы в матрицу и транспонированы для работы с алгоритмом k-средних.

С помощью библиотеки Clustering.jl данные были разделены на 10 кластеров. Результаты кластеризации визуализированы с использованием диаграммы рассеяния, где оси PetalLength и PetalWidth отражают размеры лепестков, а точки окрашены в зависимости от принадлежности к кластеру. В результате удалось продемонстрировать корректную работу алгоритма и его применение для анализа данных.

Самостоятельная работа

7.4.2. Регрессия (метод наименьших квадратов в случае линейной регрессии). См. рис. 8, См. рис. 9, См. рис. 10.

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

```
1000-element Vector{Float64}:
 -0.7945765229015872
 -0.6720296638517435
  1.3716244627873222
  2.1785938205075484
 -0.3260138255273183
  0.26170158513295094
  0.9557760816312258
 -2.304971670565372
 -0.6942984013717575
 -0.9267042171300182
 -1.2557384299492562
 -1.6445050915198212
  1.5532534557172666
  ⋮
 -1.6250530641171432
  0.6461356986934509
 -0.6007257958443772
 -0.010337923351421996
  1.328135396010722
 -1.983401200698946
 -0.14795949976913234
 -1.3949207950494904
  1.3520963911727373
 -1.8038372400597185
  0.6129175787211228
 -0.16580110961773115
```

```
N = 1000
X2 = hcat(ones(N), X)
```

```
1000x4 Matrix{Float64}:
 1.0  -1.11694     0.644413   -0.217648
 1.0   0.0169132  -3.37187    -0.46774
```

Самостоятельная работа

[51]:
```
betahat1 = X2 \ y
yp = X2 * betahat1
mse1 = sqrt(sum(abs2.(y - yp)) / N)
display(betahat1)
mse1
```

```
4-element Vector{Float64}:
 0.003675260146041499
 0.7485522672623919
 0.09827734019830424
 0.808080938633673
```

[51]:
```
0.09996843006061548
```

[52]:
```
betahat2 = llsq(X, y; bias=false)
yp = X * betahat2
mse2 = sqrt(sum(abs2.(y - yp)) / N)
display(betahat2)
mse2
```

```
3-element Vector{Float64}:
 0.7485913584036923
 0.0981758667102467
 0.8079549588733729
```

[52]:
```
0.10003581652171566
```

[53]:
```
X3 = DataFrame(a=y, b=X[:,1], c=X[:,2], d=X[:,3])
lmMSE = lm(@formula(a ~ b + c + d), X3)
betahat3 = GLM.coeftable(lmMSE).cols[1]
yp = X2 * betahat3
mse3 = sqrt(sum(abs2.(y - yp)) / N)
display(betahat3)
mse3
```

```
4-element Vector{Float64}:
 0.0036752601460415322
 0.7485522672623924
 0.09827734019830442
 0.8080809386336726
```

[53]:
```
0.09996843006061547
```

Часть 2.

[54]:
```
X = rand(100)
y = 2X + 0.1 * randn(100)
Xh = hcat(ones(100), X)
```

Рис. 9: Самостоятельная работа_2_2

Самостоятельная работа

```
y = 2X + 0.1 * randn(100)
Xh = hcat(ones(100), X)
lm2 = fit(LinearModel, Xh, y)
display(lm2)

Plots.plot(title="График регрессии", xlabel="x", ylabel="y", legend=:topleft)
Plots.plot!(X, predict(lm2), label="Линия регрессии")
Plots.scatter!(X, y, label="Данные")
```

```
LinearModel{GLM.LmResp{Vector{Float64}}, GLM.DensePredChol{Float64, LinearAlgebra.CholeskyPivoted{Float64, Matrix{Float64}, Vector{Int64}}}}:

Coefficients:
─────────────────────────────────────────────────────────────────────────
          Coef.  Std. Error      t  Pr(>|t|)   Lower 95%  Upper 95%
─────────────────────────────────────────────────────────────────────────
x1  -0.00481807   0.0181552  -0.27    0.7913  -0.0408465  0.0312103
x2   2.00541      0.0341101  58.79    <1e-77   1.93772    2.0731
─────────────────────────────────────────────────────────────────────────
```

[54]:

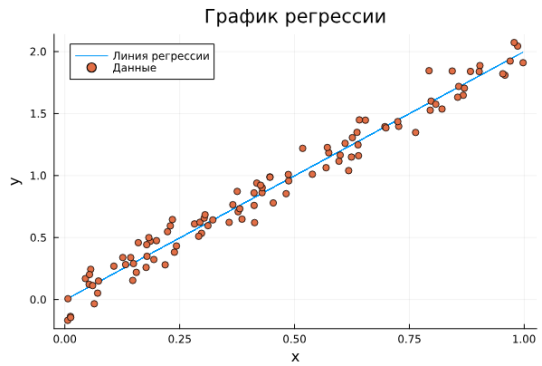


Рис. 10: Самостоятельная работа_2_3

Самостоятельная работа

Была выполнена линейная регрессия с использованием метода наименьших квадратов. Для этого были сгенерированы случайные данные и построена линейная зависимость между независимой переменной X и зависимой переменной y.

На графике представлена точечная диаграмма исходных данных (оранжевые точки) и линия регрессии (синяя линия), которая показывает оптимальное соответствие между переменными. Линия регрессии демонстрирует сильную положительную зависимость, что подтверждается коэффициентами регрессии, рассчитанными методом наименьших квадратов. Это визуализирует, как данный метод подходит для анализа линейных связей между переменными.

Самостоятельная работа

7.4.3. Модель ценообразования биномиальных опционов. См. рис. 11, См. рис. 12, См. рис. 13.

В этом задании была реализована модель ценообразования биномиальных опционов. Первоначально была написана функция binomial_stock_price, которая рассчитывает траекторию изменения цены акций, используя параметры начальной цены, волатильности, процентной ставки и длины временного периода. Функция возвращает массив цен, где каждая последующая цена зависит от случайного события (рост или падение), что моделируется с помощью случайных чисел.

Самостоятельная работа

[242]:
```
function binomial_stock_price(S,T,n,sigma,r)
    h = T/n
    u = exp(r*h+sigma*sqrt(h))
    d= exp(r*h-sigma*sqrt(h))
    p = (exp(r*h)-d)/(u-d)
    stock_prices = zeros(n+1)
    stock_prices[1] = S
    for i in 2:n+1
        epsilon = rand()
        if epsilon > 0.5
            stock_prices[i] = stock_prices[i-1]*u
        else
            stock_prices[i] = stock_prices[i-1]*d
        end
    end
    return stock_prices
end

S,T,n, sigma, r = 100.0,1.0,10000,0.3, 0.08
stock_prices = binomial_stock_price(S,T,n,sigma,r)

plot(stock_prices, xlabel= "Время", ylabel = "Цена", label = "Стоимость акции")
```

[242]:

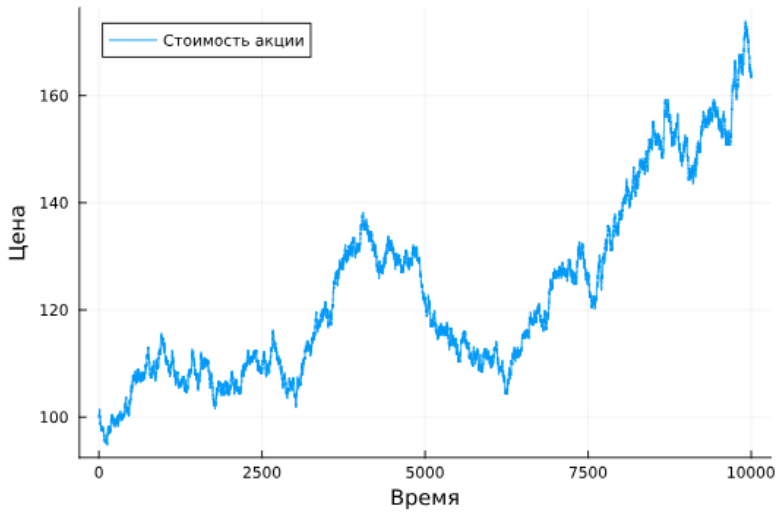


Рис. 11: Самостоятельная работа_3_1

Самостоятельная работа

На первом графике показана симуляция одной траектории изменения цены акции. Ось X представляет временные интервалы, а ось Y — стоимость акций. График демонстрирует динамику изменения цены, включая периоды роста и падения.

Самостоятельная работа

[244]:
```
function createPath(S::Float64, r::Float64, sigma::Float64, T::Float64, n::Int64)
    dt = T / n
    path = [S]

    for i in 1:n
        epsilon = randn()
        S = S * exp((r - 0.5 * sigma^2) * dt + sigma * sqrt(dt) * epsilon)
        push!(path, S)
    end

    return path
end

paths = [createPath(S, r, sigma, T, n) for i in 1:10]
plot(paths, title="Траектории", xlabel="Время", ylabel="Цена", legend=false)
```

[244]:

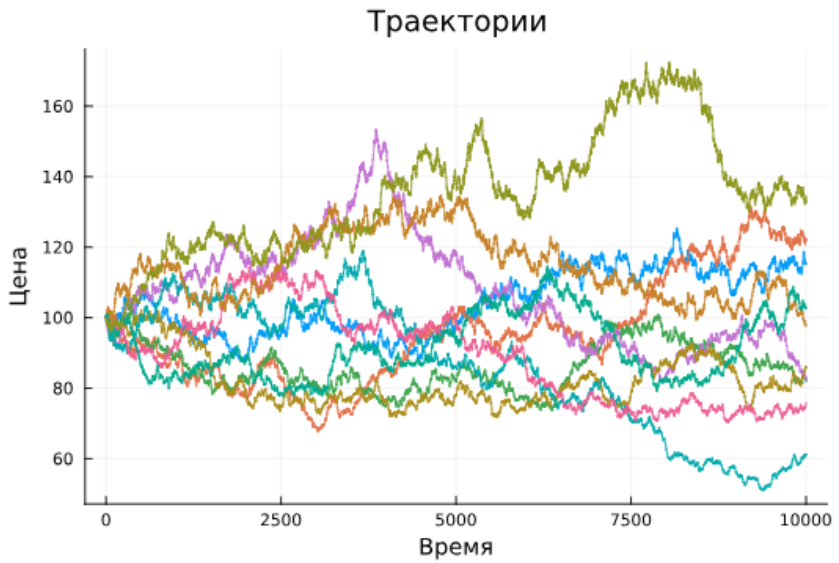


Рис. 12: Самостоятельная работа_3_2

Самостоятельная работа

Далее была реализована функция createPath, позволяющая сгенерировать несколько траекторий изменения цен. Результаты были визуализированы на втором графике, где представлены 10 различных траекторий. Каждая линия на графике — это отдельная симуляция изменения стоимости акции, что демонстрирует вариативность и влияние случайности на прогноз.

Самостоятельная работа

[252]:
```
using Random
using Base.Threads

paths2 = []
@threads for i = 1:10
    push!(paths2, createPath(S,r,sigma,T,n))
end
plot(paths2, title = "stocks price paths (threads)", xlabel = "Time", ylabel = "Price", legend = false)
```

[252]:

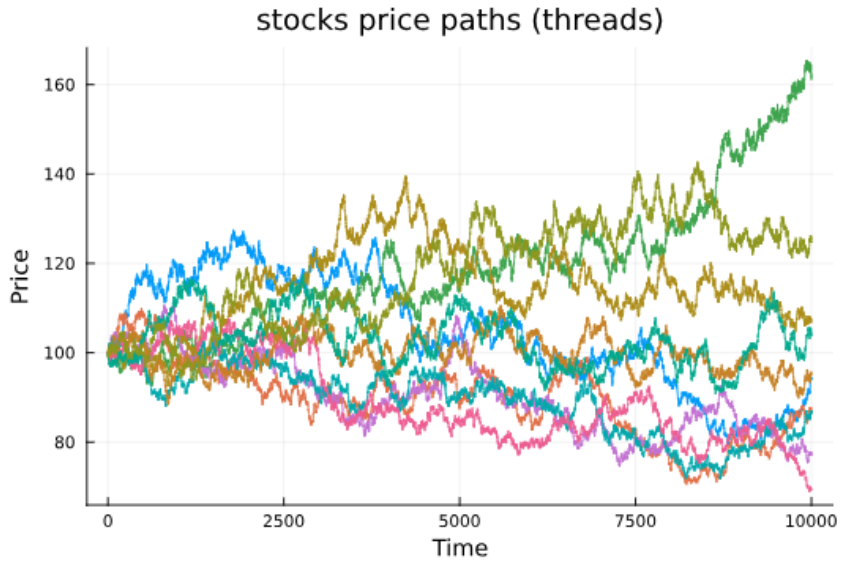


Рис. 13: Самостоятельная работа_3_3

Самостоятельная работа

Для оптимизации вычислений была использована параллельная обработка с помощью @threads, что позволило ускорить генерацию траекторий. На третьем графике представлены те же 10 траекторий, но они были сгенерированы с использованием многопоточности, что эффективно сокращает время расчётов при большом количестве симуляций.

Все графики отражают реалистичное поведение цен акций на рынке в условиях случайных изменений, что делает метод полезным для анализа и прогнозирования финансовых данных.

Полученные результаты

1. Считаны и обработаны данные в формате CSV.
2. Реализованы алгоритмы кластеризации методом k-средних и анализа методом главных компонент (PCA).
3. Построены графики для визуализации результатов анализа данных.
4. Выполнен анализ линейной регрессии с использованием больших наборов данных.
5. Полученные результаты подтвердили эффективность использования Julia для обработки и анализа данных.

Вывод

Научились работатать со специализированными пакетами Julia для обработки данных.

Список литературы

[1] Julia: https://ru.wikipedia.org/wiki/Julia