Лабораторная работа № 7

Введение в работу сданными

Беличева Д. М.

Российский университет дружбы народов, Москва, Россия

Информация

Докладчик

- Беличева Дарья Михайловна
- студентка
- Российский университет дружбы народов
- 1032216453@pfur.ru
- https://dmbelicheva.github.io/ru/

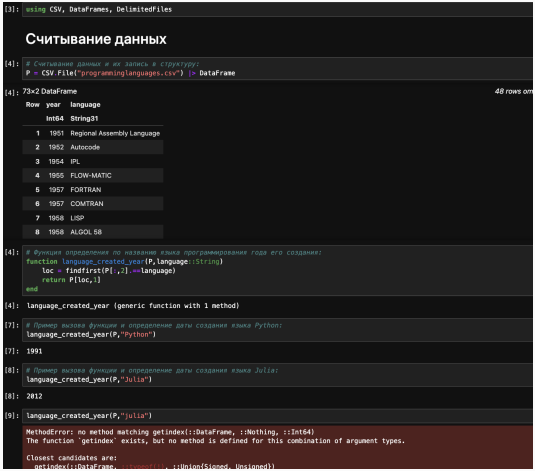
Цель работы

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

Задание

1. Используя JupyterLab, повторите примерыи. При этом дополните графики обозначениями осей координат, легендой с названиями траекторий, названиями графиков и т.п.
2. Выполните задания для самостоятельной работы.

Выполнение лабораторной работы



```
[3]: using CSV, DataFrames, DelimitedFiles
```

Считывание данных

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

[4]: 73×2 DataFrame — *48 rows om*

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
8	1958	ALGOL 58

```
[4]: # 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
```

```
[4]: language_created_year (generic function with 1 method)
```

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

```
[7]: 1991
```

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

```
[8]: 2012
```

```
[9]: language_created_year(P,"julia")
```

```
MethodError: no method matching getindex(::DataFrame, ::Nothing, ::Int64)
The function `getindex` exists, but no method is defined for this combination of argument types.

Closest candidates are:
  getindex(::DataFrame, ::typeof(!), ::Union{Signed, Unsigned})
```

Рис. 1: Считывание данных

Выполнение лабораторной работы

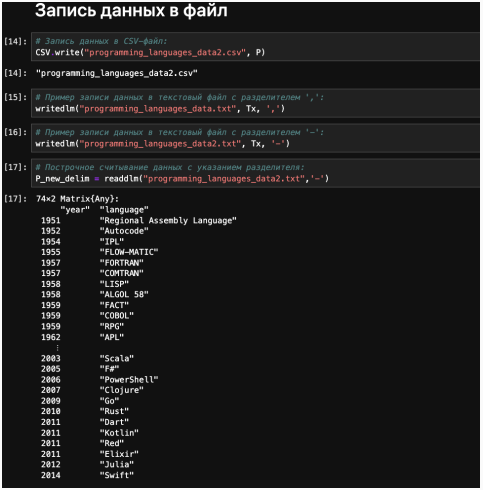


Рис. 2: Запись данных в файл

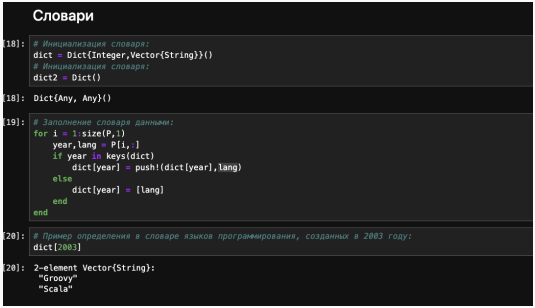


Рис. 3: Словари

Выполнение лабораторной работы

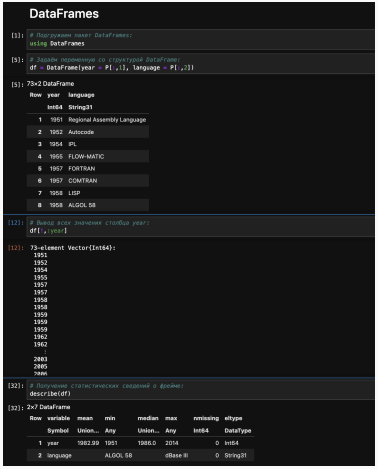
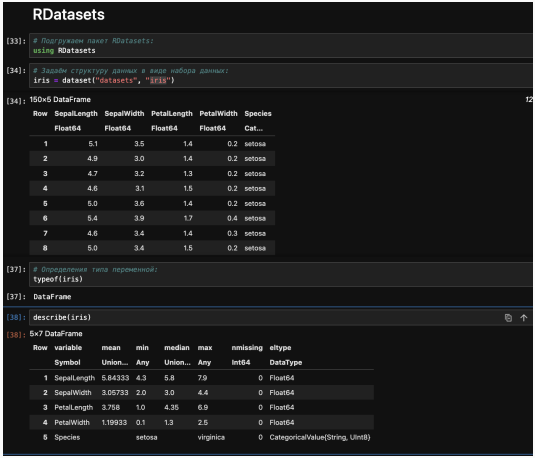


Рис. 4: DataFrames

Выполнение лабораторной работы



RDatasets

```
[33]: # Подгружаем пакет RDatasets:
      using RDatasets

[34]: # Задаём структуру данных в виде набора данных:
      iris = dataset("datasets", "iris")
```

[34]: 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

```
[37]: # Определения типа переменной:
      typeof(iris)
```

[37]: DataFrame

```
[38]: describe(iris)
```

[38]: 5×7 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. 5: RDatasets

Работа с переменными отсутствующего типа (Missing Values)

```
[40]: # Отсутствующий тип:
      a = missing
      typeof(a)
```

```
[40]: Missing
```

```
[41]: # Пример операции с переменной отсутствующего типа:
      a + 1
```

```
[41]: missing
```

```
[43]: # Определение перечня продуктов:
      foods = ["apple", "cucumber", "tomato", "banana"]
      # Определение калорий:
      calories = [missing,47,22,105]
      # Определение типа переменной:
      typeof(calories)
```

```
[43]: Vector{Union{Missing, Int64}}
```

```
[44]: # Подключаем пакет Statistics:
      using Statistics
      # Определение среднего значения:
      mean(calories)
```

```
[44]: missing
```

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

```
[45]: 58.0
```

```
[49]: # 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 = outerjoin(dataframe_calories,dataframe_prices,on=:item)
```

```
[49]: 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. 6: Работа с переменными отсутствующеготипа (MissingValues)

Выполнение лабораторной работы

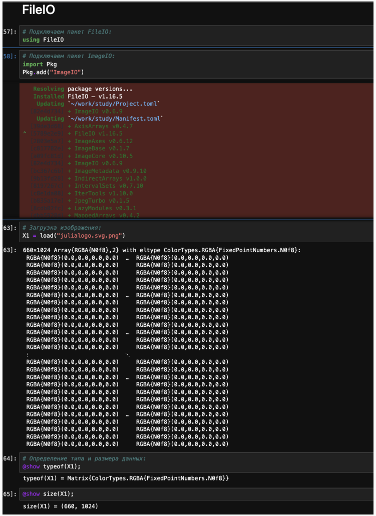


Рис. 7: FileIO

Выполнение лабораторной работы

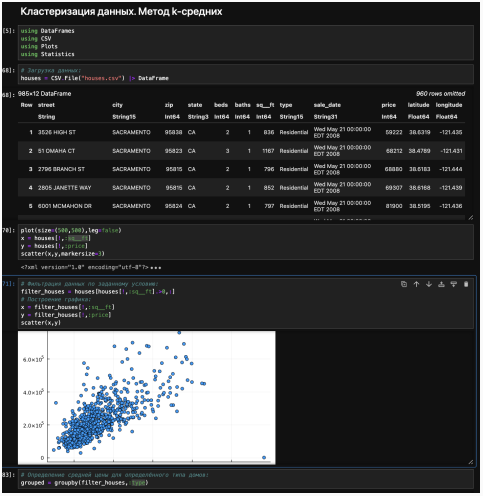


Рис. 8: Кластеризация данных. Метод k-средних

Выполнение лабораторной работы

```
85]: using Clustering

48]: # Добавление данных :latitude и :longitude в новый фрейм:
     X = filter_houses[:,[:latitude,:longitude]]
     # Конвертация данных в матричный вид:
     X = Matrix(X)
     # Транспонирование матрицы с данными:
     X = X'
     # Задание количества кластеров:
     k = length(unique(filter_houses[:,:zip]))
     # Определение k-средних:
     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])

     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)

     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)
```

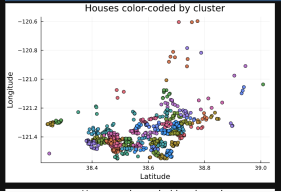


Рис. 9: Кластеризация данных. Метод k-средних

Выполнение лабораторной работы

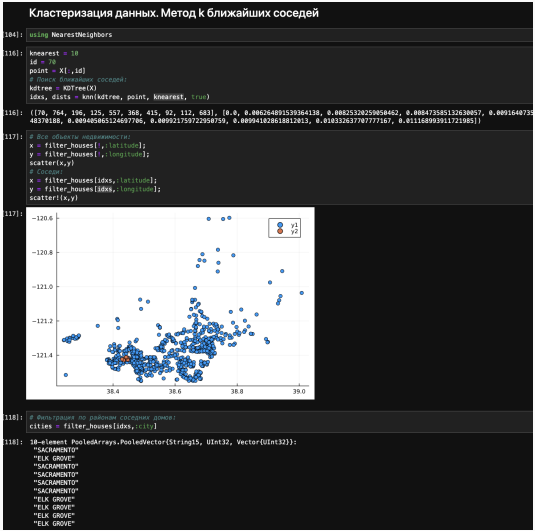


Рис. 10: Кластеризация данных. Метод k ближайших соседей

Выполнение лабораторной работы

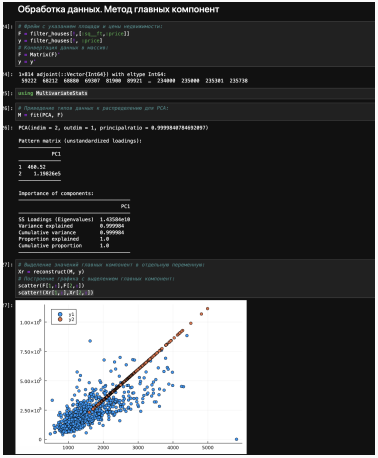


Рис. 11: Обработка данных. Метод главных компонент

Выполнение лабораторной работы

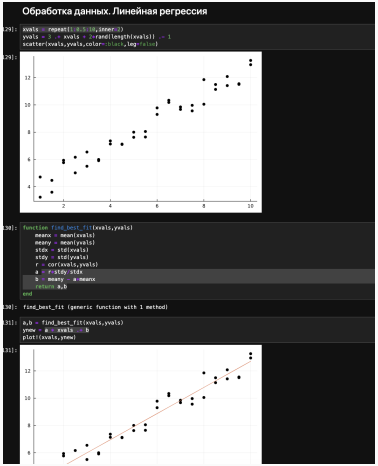


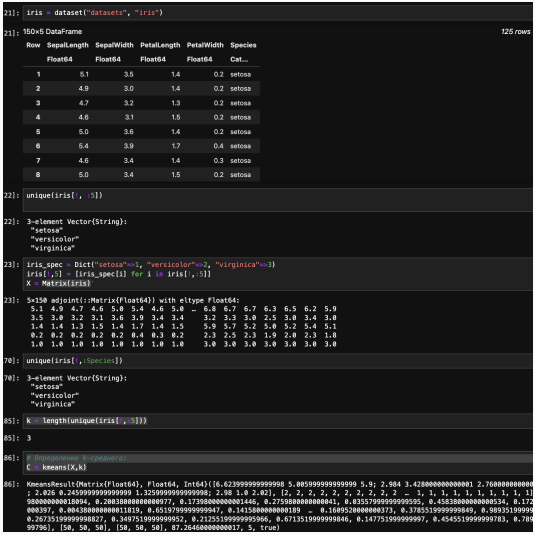
Рис. 12: Обработка данных. Линейная регрессия

Выполнение лабораторной работы

```
133]: xvals = 1:100000;
      xvals = repeat(xvals,inner=2);
      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,)
        0.016692 seconds (29.73 k allocations: 1.531 MiB, 96.41% compilation time)
133]: (1.0000000292669111, 2.9970284742303193)

134]: using PyCall
      using Conda

137]: 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)
        0.043921 seconds (112.61 k allocations: 5.636 MiB, 92.86% compilation time)
137]: (1.0000000292669111, 2.9970284742344871)

139]: using BenchmarkTools

140]: @time a,b = py"find_best_fit_python"(xvals,yvals)
      @btime a,b = find_best_fit(xvals,yvals)
        0.011368 seconds (11.77 k allocations: 605.641 KiB, 70.18% compilation time)
        337.375 μs (1 allocation: 32 bytes)
140]: (1.0000000292669111, 2.9970284742303193)
```

Рис. 13: Обработка данных. Линейная регрессия

Выполнение лабораторной работы



```
21]: iris = dataset("datasets", "iris")
```

```
21]: 150×5 DataFrame                                                                                    125 rows
```

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

```
22]: unique(iris[!, :5])
```

```
22]: 3-element Vector{String}:
     "setosa"
     "versicolor"
     "virginica"
```

```
23]: iris_spec = Dict("setosa"=>1, "versicolor"=>2, "virginica"=>3)
     iris[!,5] = [iris_spec[i] for i in iris[!,5]]
     X = Matrix(iris)
```

```
23]: 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
```

```
70]: unique(iris[!,:Species])
```

```
70]: 3-element Vector{String}:
     "setosa"
     "versicolor"
     "virginica"
```

```
85]: k = length(unique(iris[!,5]))
```

```
85]: 3
```

```
86]: # Определение K-средних
     C = kmeans(X,k)
```

```
86]: KmeansResult{Matrix{Float64}, Float64, Int64}([6.623999999999998 5.005999999999999 5.9; 2.984 3.428000000000001 2.768000000000
     ; 2.026 0.2459999999999999 1.3259999999999998; 2.98 1.0 2.02], [2, 2, 2, 2, 2, 2, 2, 2, 2, 2  …  1, 1, 1, 1, 1, 1, 1, 1, 1, 1
     980000000018994, 0.20838888888888077, 0.17388000000001446, 0.27598888888888841, 0.8355799999999935, 0.4583888888888534, 0.172
     000397, 0.004388888888811819, 0.6519790999999947, 0.14158000000000189  …  0.1605520000000373, 0.3785519999999849, 0.9893519999
     0.2673519999999827, 0.3497519999999952, 4.213519999999956, 0.6713519999999846, 0.14775199999999997, 0.4545519999999783, 0.709
     99798], [50, 50, 50], [50, 50, 50], 87.26468000000017, 5, true)
```

Рис. 14: Кластеризация

Выполнение лабораторной работы

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

150×6 DataFrame — 125 rows o

Row	cluster	SepalLength	SepalWidth	PetalLength	PetalWidth	Species
	Int64	Float64	Float64	Float64	Float64	Int64
1	2	5.1	3.5	1.4	0.2	1
2	2	4.9	3.0	1.4	0.2	1
3	2	4.7	3.2	1.3	0.2	1
4	2	4.6	3.1	1.5	0.2	1
5	2	5.0	3.6	1.4	0.2	1
6	2	5.4	3.9	1.7	0.4	1
7	2	4.6	3.4	1.4	0.3	1
8	2	5.0	3.4	1.5	0.2	1

```
clusters_figure_2 = plot(legend = false)
for i = 1:k
    clustered_iris = df[df[:,:cluster].== i,:]
    xvals = clustered_iris[:,:PetalLength]
    yvals = clustered_iris[:,:PetalWidth]
    scatter!(clusters_figure_2,xvals,yvals,markersize=4)
end
xlabel!("PetalLength")
ylabel!("PetalWidth")
title!("Irises color-coded by cluster")
display(clusters_figure_2)
```

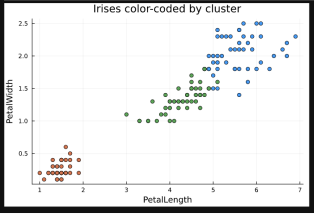


Рис. 15: Кластеризация

Выполнение лабораторной работы

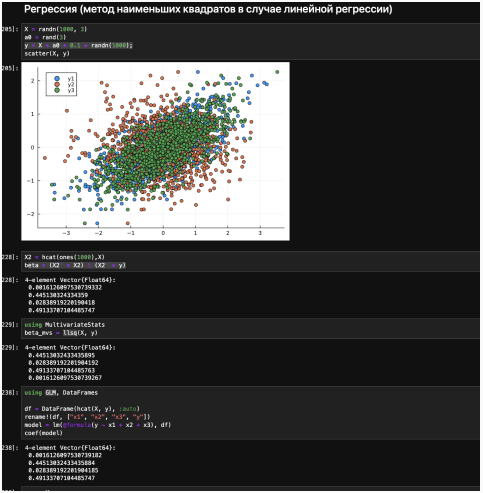


Рис. 16: Регрессия

Выполнение лабораторной работы

Рис. 17: Регрессия

Выполнение лабораторной работы

```
[9]: S = 100.0
     T = 1.0
     n = 10000
     σ = 0.3
     r = 0.08
     h = T/n

     u = exp(r*h + σ*sqrt(h))
     d = exp(r*h - σ*sqrt(h))
     p = (exp(r*h) - d)/(u - d);
```

```
[0]: prices = []
     price = S
     push!(prices, price)
     for i in 1:n
         if rand() < p
             price *= u
         else
             price *= d
         end
         push!(prices, price)
     end
```

```
[1]: X = collect(0:h:1)
     plot(X, prices, xlabel="Период", ylabel="Цена акции", title="Траектория цены акции")
```

[1]:

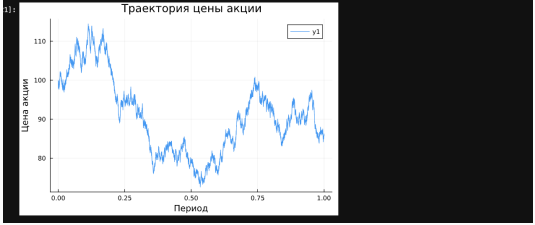


Рис. 18: Модель ценообразования биномиальных опционов

Выполнение лабораторной работы

```
[24]: function createPath(S::Float64, r::Float64, σ::Float64, T::Float64, n::Int64)
          h = T/n
          X = collect(0:h:1)
          u = exp(r*h + σ*sqrt(h))
          d = exp(r*h - σ*sqrt(h))
          p = (exp(r*h) - d)/(u - d);
          prices = []
          price = S
          push!(prices, price)
          for i in 1:n
              if rand() < p
                  price *= u
              else
                  price *= d
              end
              push!(prices, price)
          end
          return (X, prices)
      end

[24]: createPath (generic function with 1 method)

[25]: p = plot()
      for i=1:10
          plot!(p, createPath(S, r, σ, T, n))
      end
      p

[25]:
```

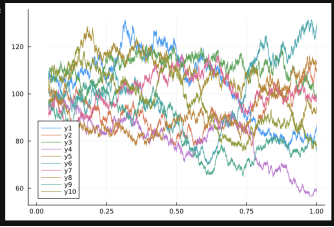


Рис. 19: Модель ценообразования биномиальных опционов

Выполнение лабораторной работы

```
[26]: function createPath_parall(S::Float64, r::Float64, σ::Float64, T::Float64, n::Int64)
          h = T/n
          u = exp(r*h + σ*sqrt(h))
          d = exp(r*h - σ*sqrt(h))
          p = (exp(r*h) - d)/(u - d);
          prices = []
          price = S
          push!(prices, price)
          Threads.@threads for i in 1:n
              if rand() < p
                  price *= u
              else
                  price *= d
              end
              push!(prices, price)
          end
          return prices
      end

[26]: createPath_parall (generic function with 1 method)

[28]: p = plot()
      for i=1:10
          plot!(p, createPath_parall(S, r, σ, T, n))
      end
      p

[28]:
```

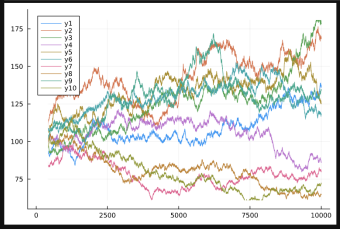


Рис. 20: Модель ценообразования биномиальных опционов

Выводы

В результате выполнения данной лабораторной работы я освоила специализированные пакеты Julia для обработки данных.

Список литературы

1. JuliaLang [Электронный ресурс]. 2024 JuliaLang.org contributors. URL:https://julialang.org/(дата обращения: 11.10.2024).
2. Julia 1.11 Documentation [Электронный ресурс]. 2024 JuliaLang.orgcontributors. URL:https://docs.julialang.org/en/v1/(дата обращения:11.10.2024).