Лабораторная работа №1

Модель экспоненциального роста

София Черная

Содержание

# 1. Цель работы

Изучение модели экспоненциального роста и освоение методов имитационного моделирования с использованием языка Julia и пакета DrWatson.

# 2. Задание

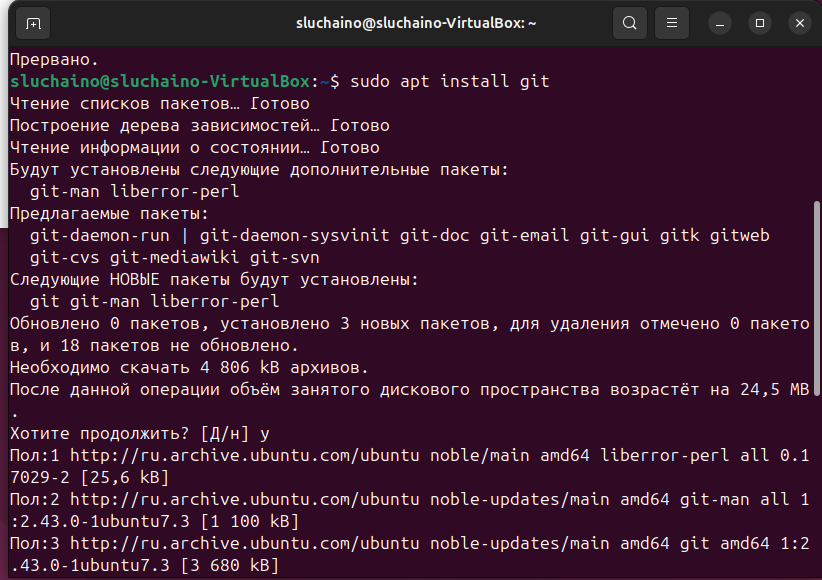
Создать рабочее пространство для курса, настроить Git и GitHub, реализовать модель экспоненциального роста в базовом и параметрическом вариантах, оформить отчёт с использованием литературного программирования.

# 3. Выполнение лабораторной работы

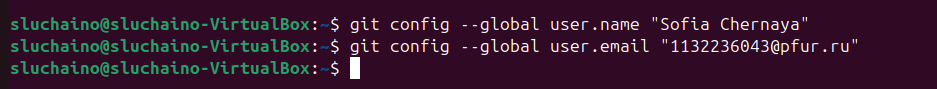
## 3.1 Настройка окружения

### 3.1.1 Установка Git и базовая настройка

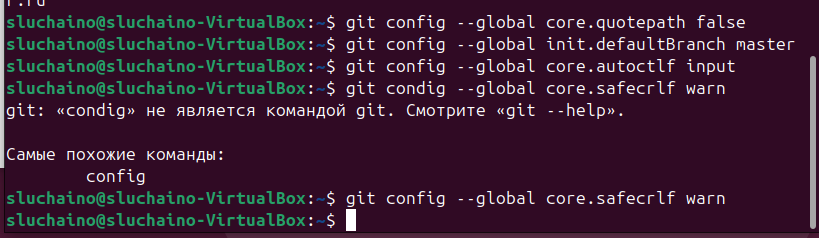
Был установлен Git и выполнена базовая настройка: указаны имя пользователя и email, настроена кодировка UTF-8, задано имя начальной ветки master, настроены параметры autocrlf и safecrlf (рис. 1-3).



Установка Git



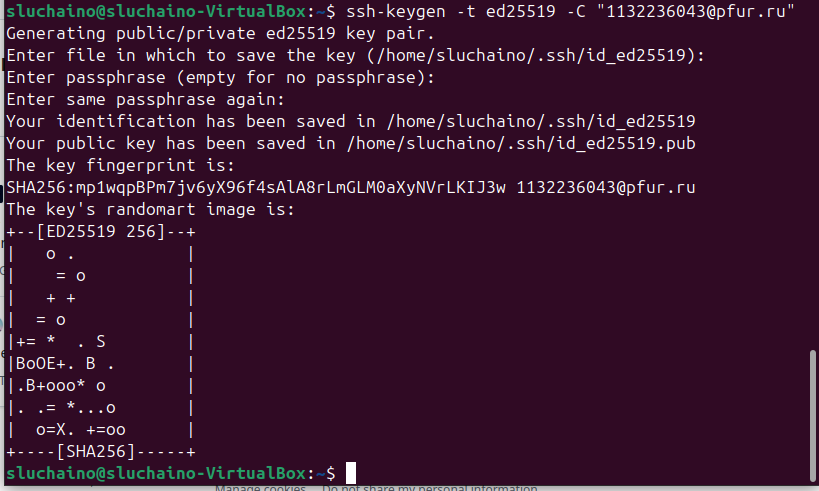
Настройка пользователя Git



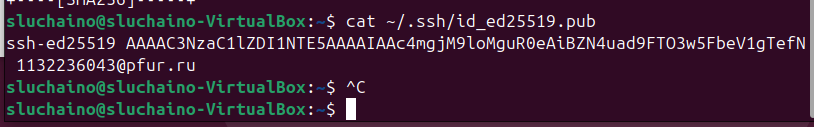
Настройка параметров Git

### 3.1.2 Создание SSH-ключа и добавление на GitHub

Сгенерирован SSH-ключ по алгоритму ed25519 для безопасного подключения к GitHub (рис. 4). Публичный ключ скопирован и добавлен в настройках аккаунта GitHub (рис. 5).



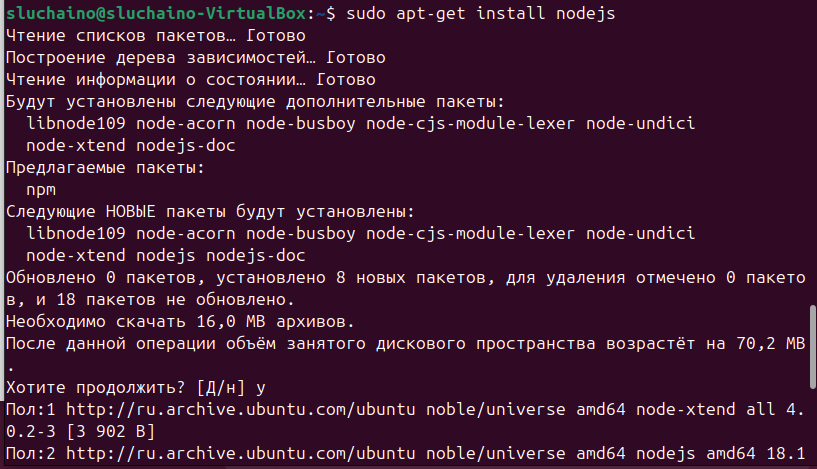
Генерация SSH-ключа



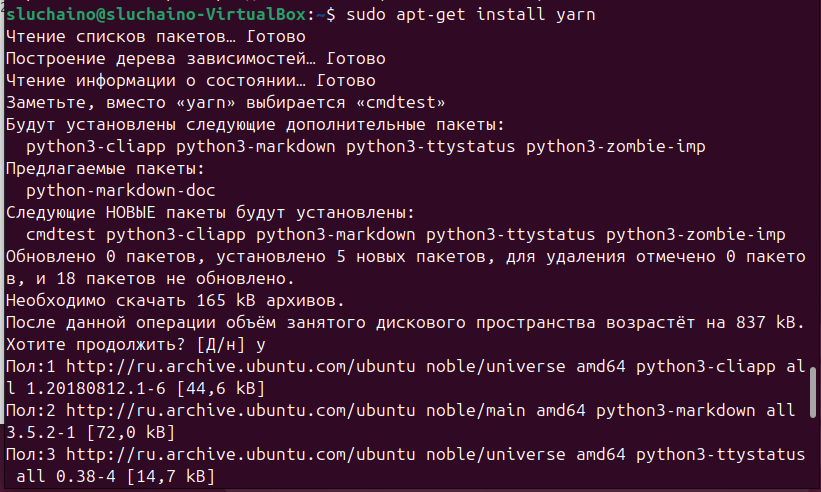
Просмотр публичного ключа

### 3.1.3 Установка Node.js, npm, pnpm и git-flow

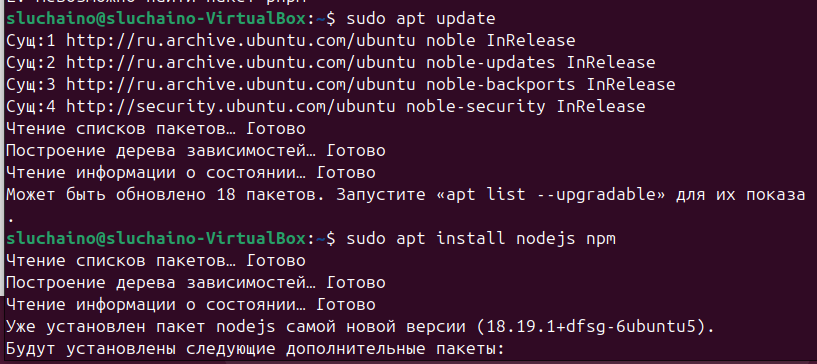
Установлены Node.js и npm (рис. 6). При попытке установить yarn через apt был установлен cmdtest (неверный пакет) (рис. 7). Позже Node.js и npm были переустановлены (рис. 8).



Установка Node.js

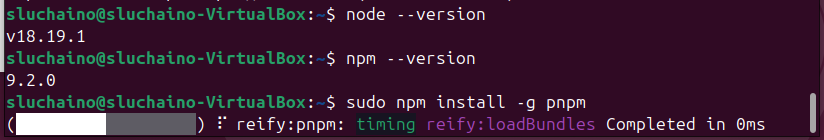


Ошибочная установка yarn

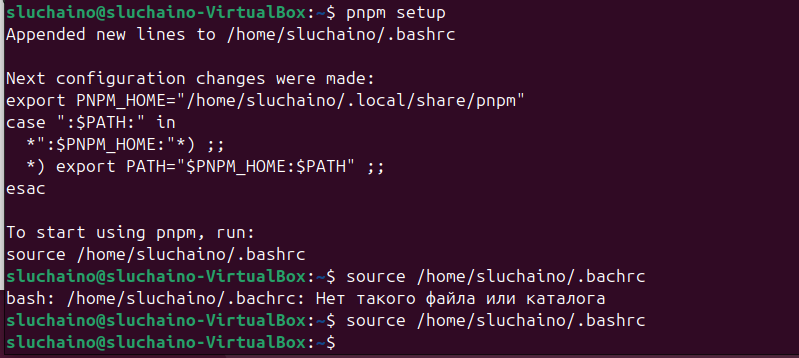


Обновление Node.js и npm

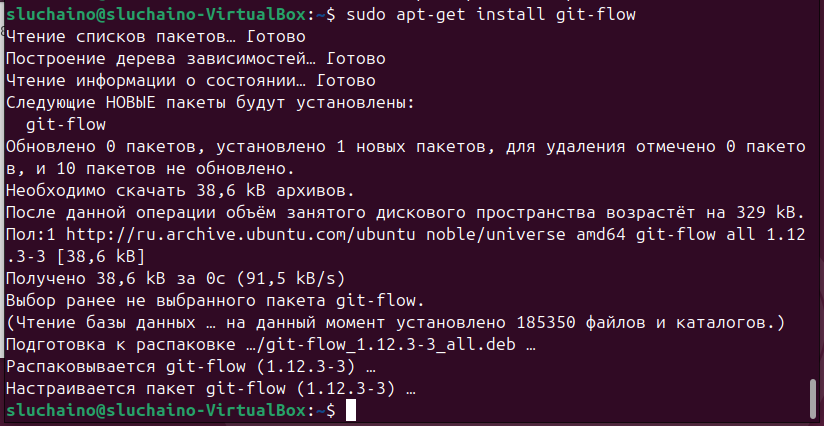
Проверены версии Node.js (v18.19.1) и npm (9.2.0), после чего через npm установлен pnpm (рис. 9). Выполнена настройка pnpm (рис. 10). Установлен git-flow (рис. 11).



Проверка версий и установка pnpm



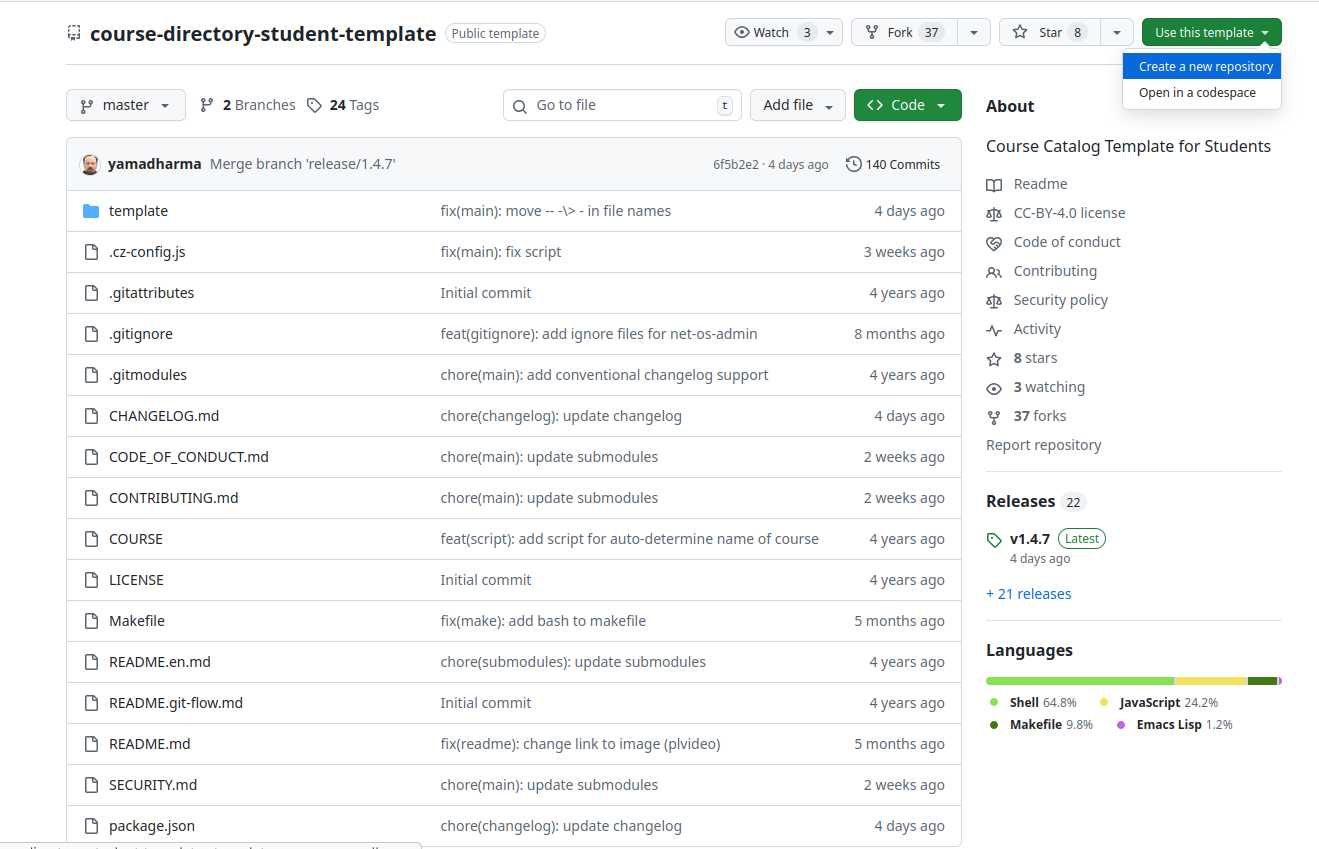
Настройка pnpm



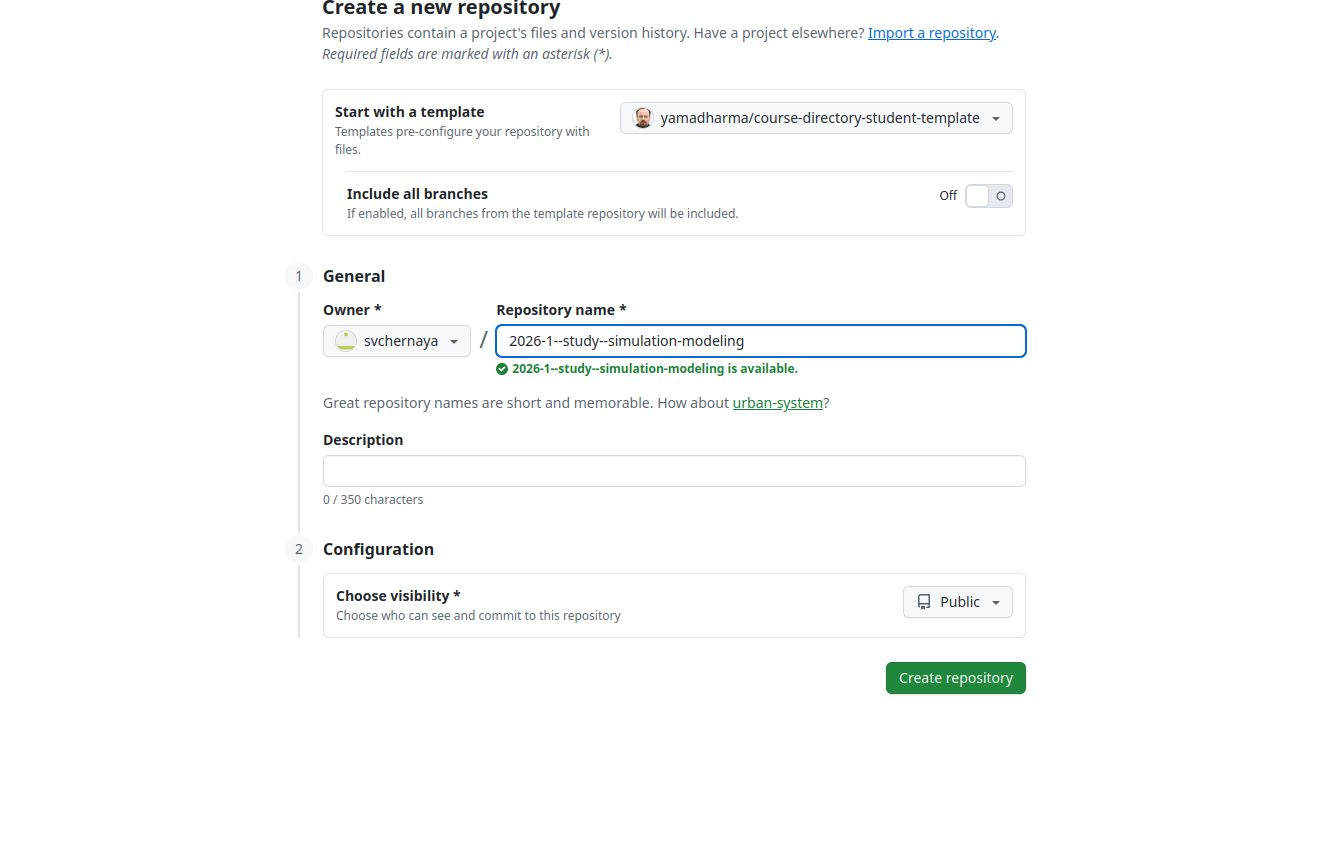
Установка git-flow

## 3.2 Создание репозитория курса

На GitHub из шаблона course-directory-student-template создан репозиторий 2026-1--study--simulation-modeling (рис. 12-14).



Шаблон курса на GitHub

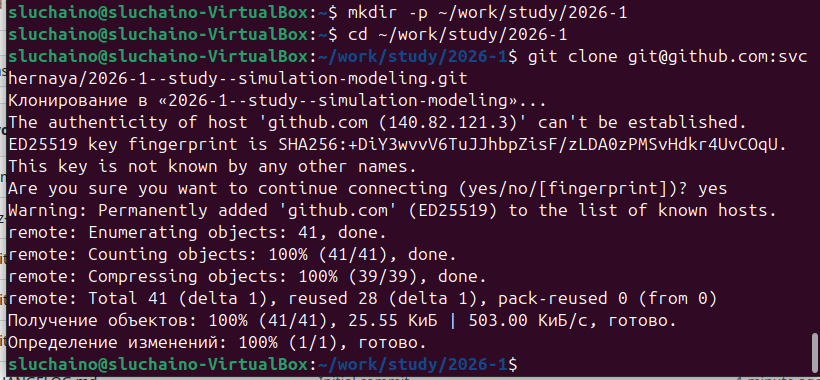


Создание нового репозитория

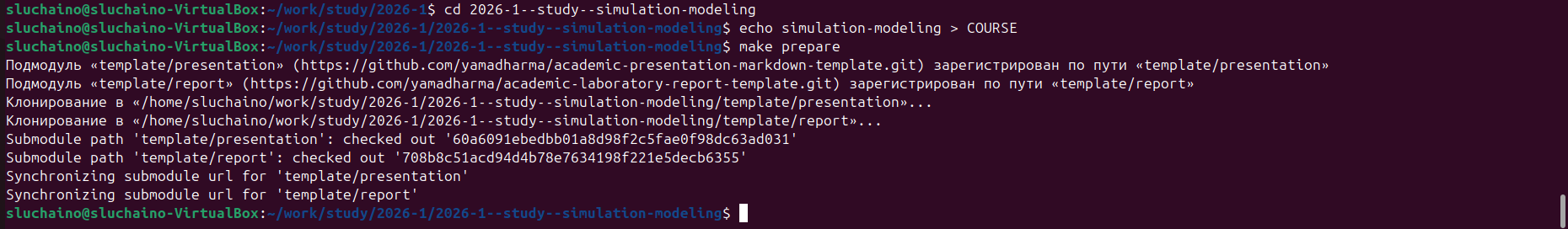


Созданный репозиторий

Репозиторий склонирован на локальную машину (рис. 15). Выполнена инициализация курса: создан файл COURSE и запущена команда make prepare для загрузки подмодулей (рис. 16).



Клонирование репозитория

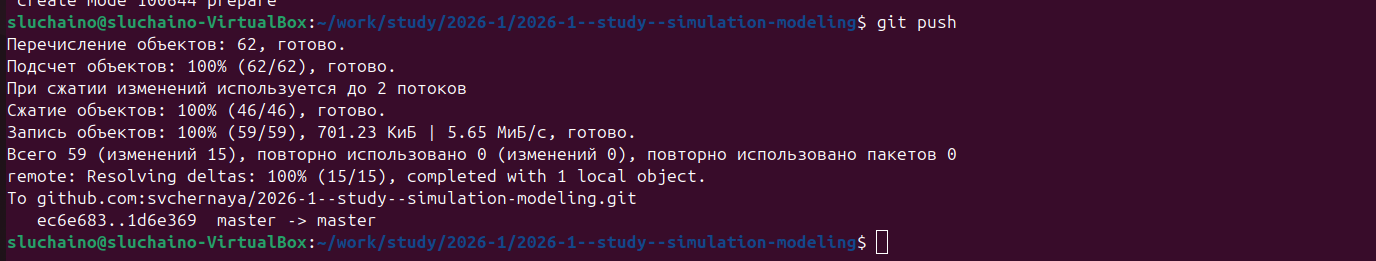


Инициализация курса

Изменения добавлены в Git и отправлены на GitHub (рис. 17-18).



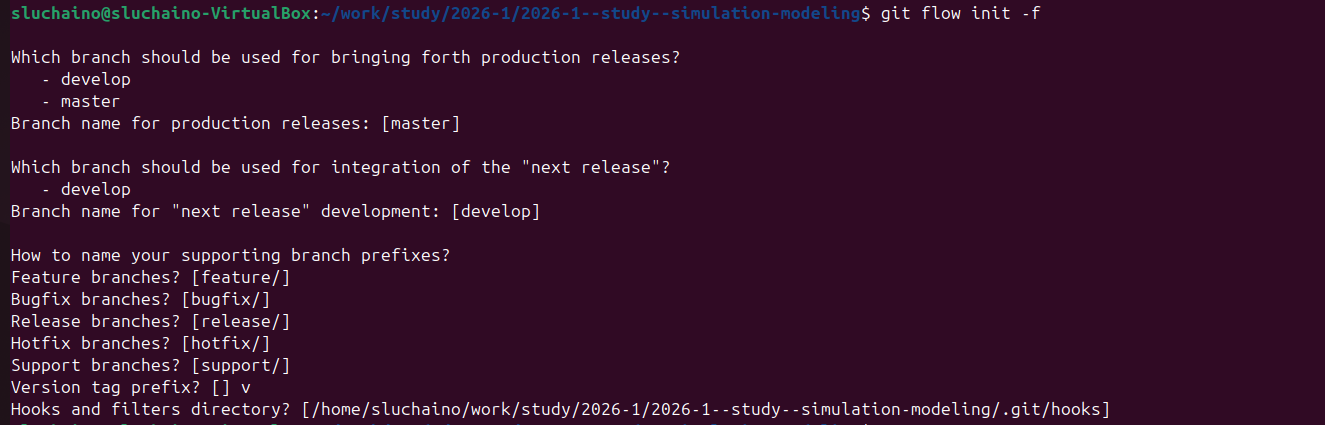
Добавление изменений в Git



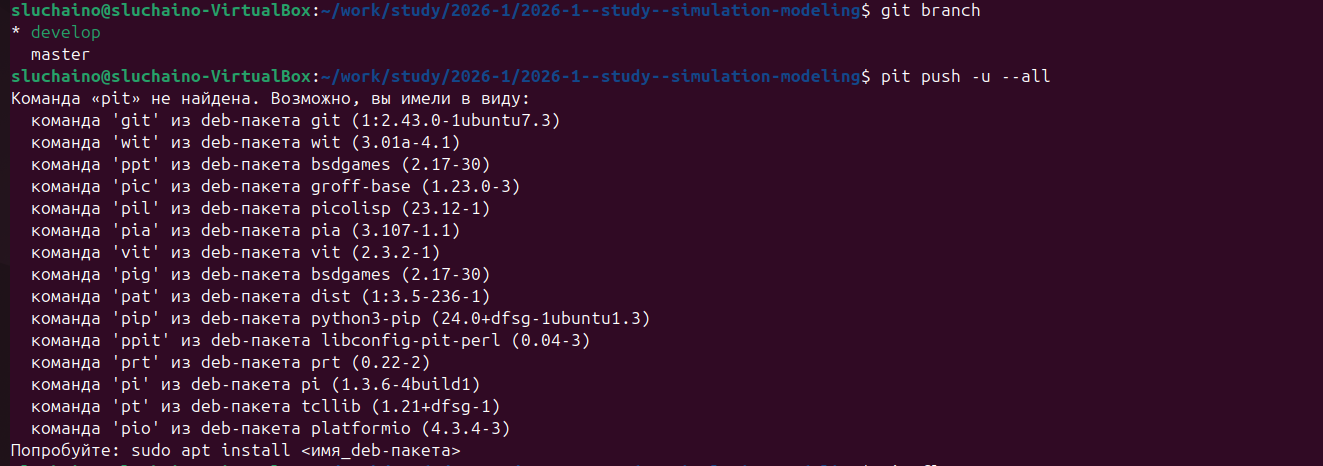
Отправка на GitHub

## 3.3 Настройка Git Flow

Выполнена инициализация Git Flow с префиксом тегов v (рис. 19). Созданы ветки master и develop (рис. 20).

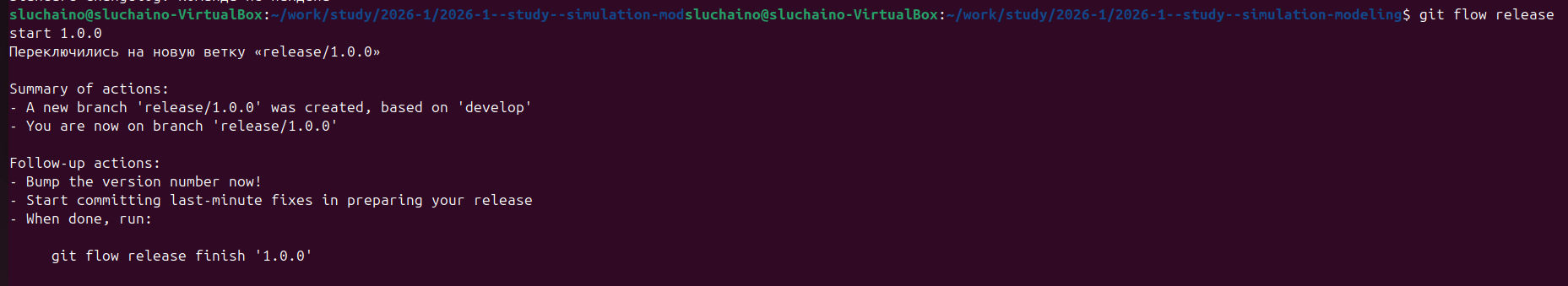


Инициализация Git Flow

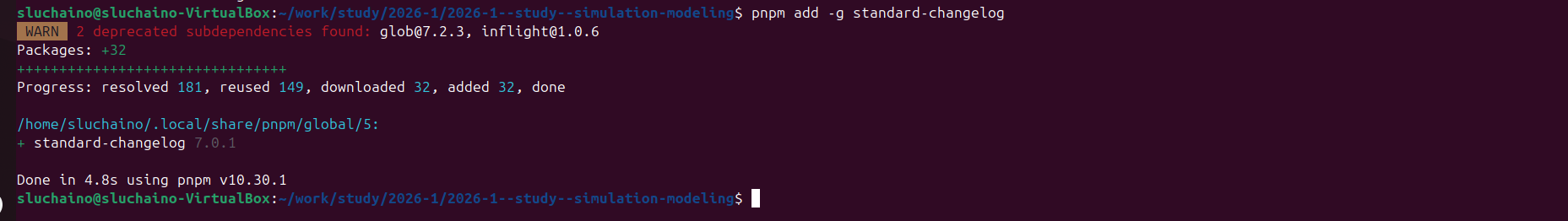


Ветки Git

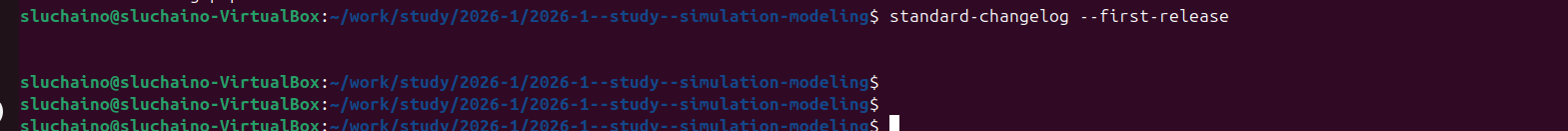
Создан релиз версии 1.0.0 (рис. 21). Установлен standard-changelog для генерации журнала изменений (рис. 22). Создан CHANGELOG.md (рис. 23) и добавлен в Git (рис. 24).



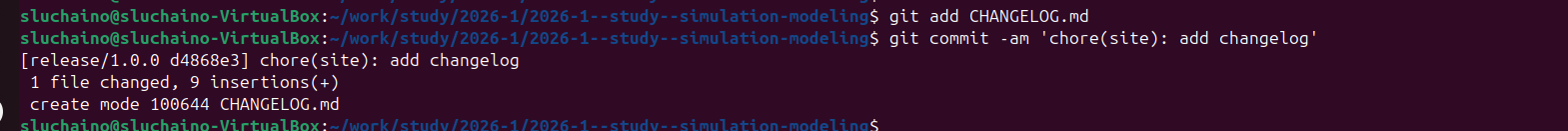
Создание релиза



Установка standard-changelog

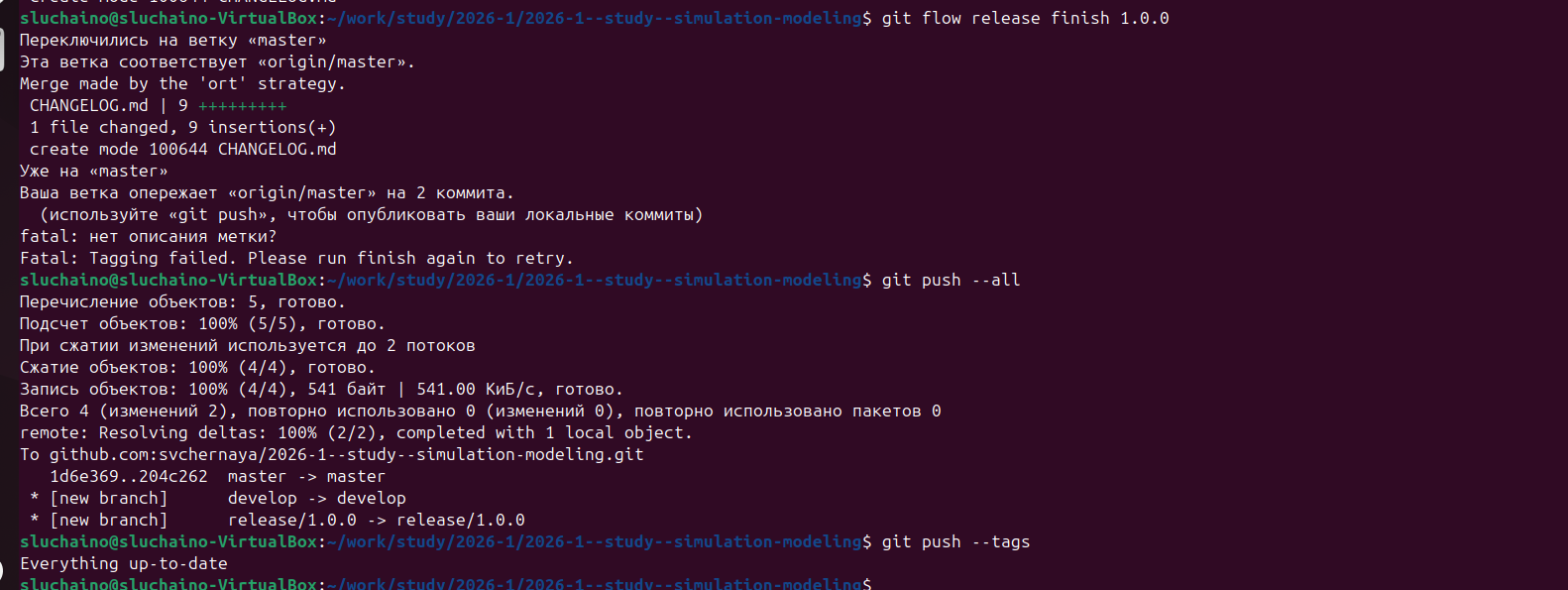


Создание CHANGELOG.md



Добавление CHANGELOG.md

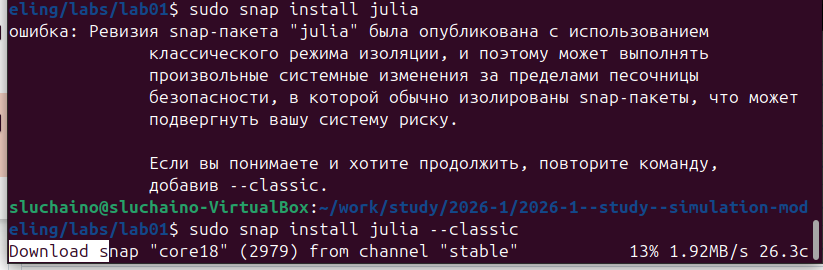
Релиз завершён, изменения отправлены на GitHub (рис. 25).



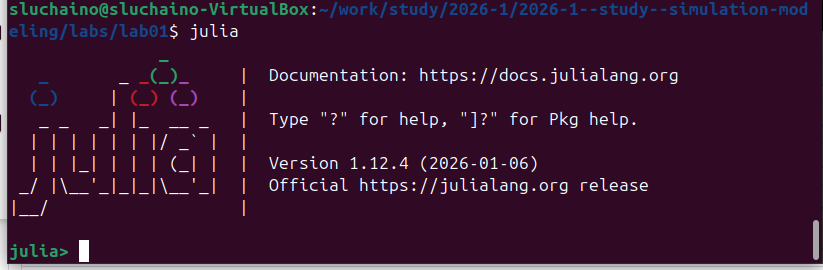
Завершение релиза и отправка

## 3.4 Установка Julia и создание проекта DrWatson

Установлена Julia через snap с флагом –classic (рис. 26-27).

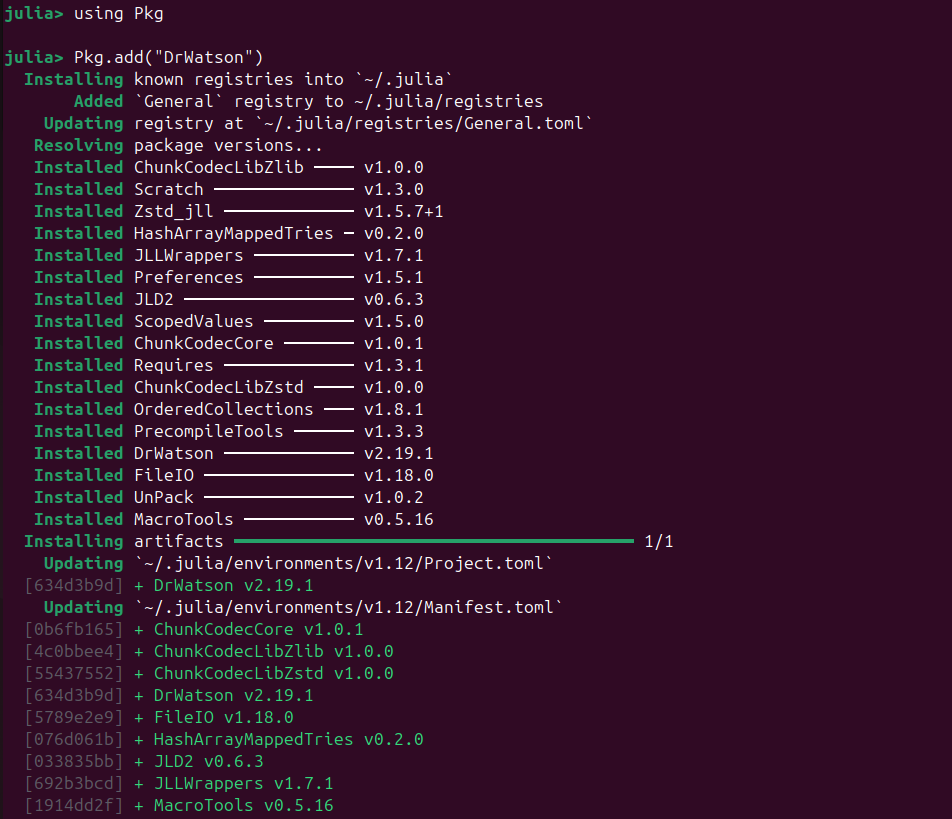


Установка Julia

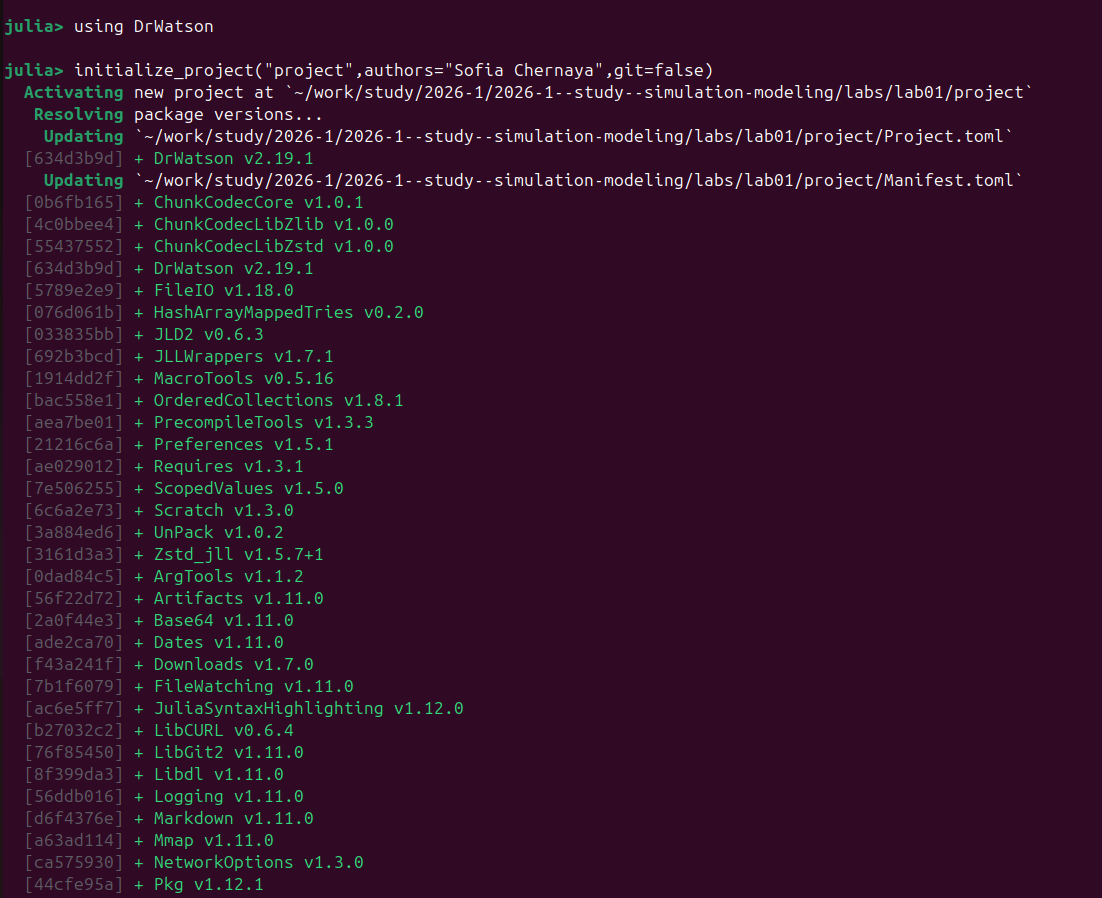


Запуск Julia

В Julia установлен пакет DrWatson (рис. 28) и создан проект project в папке labs/lab01/ (рис. 29).



Установка DrWatson

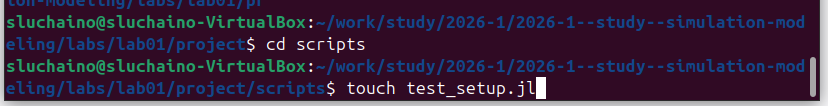


Создание проекта DrWatson

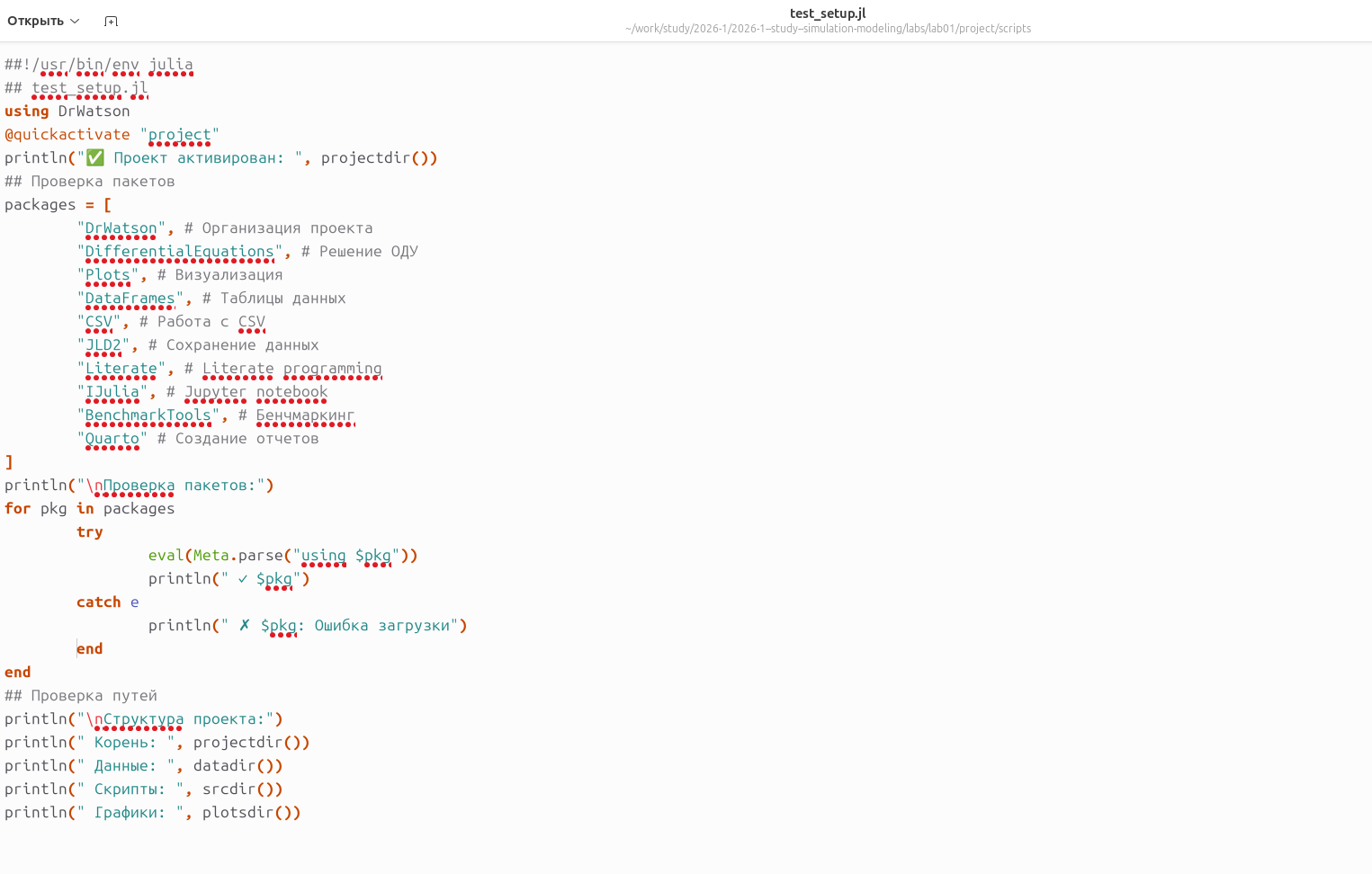
Установлены необходимые пакеты (DifferentialEquations, Plots, DataFrames и др.) (рис. 30). Создан тестовый скрипт test\_setup.jl для проверки установки (рис. 31-32).



Установка пакетов

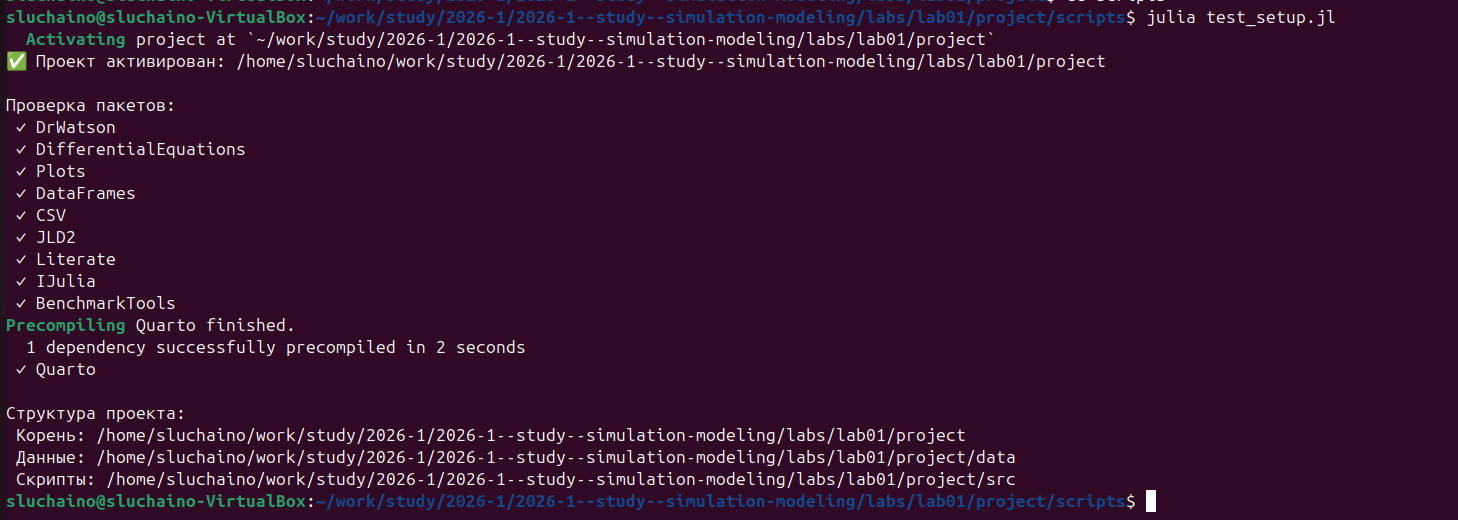


Создание test\_setup.jl



Код test\_setup.jl

Запуск тестового скрипта подтвердил корректную установку всех пакетов (рис. 33).



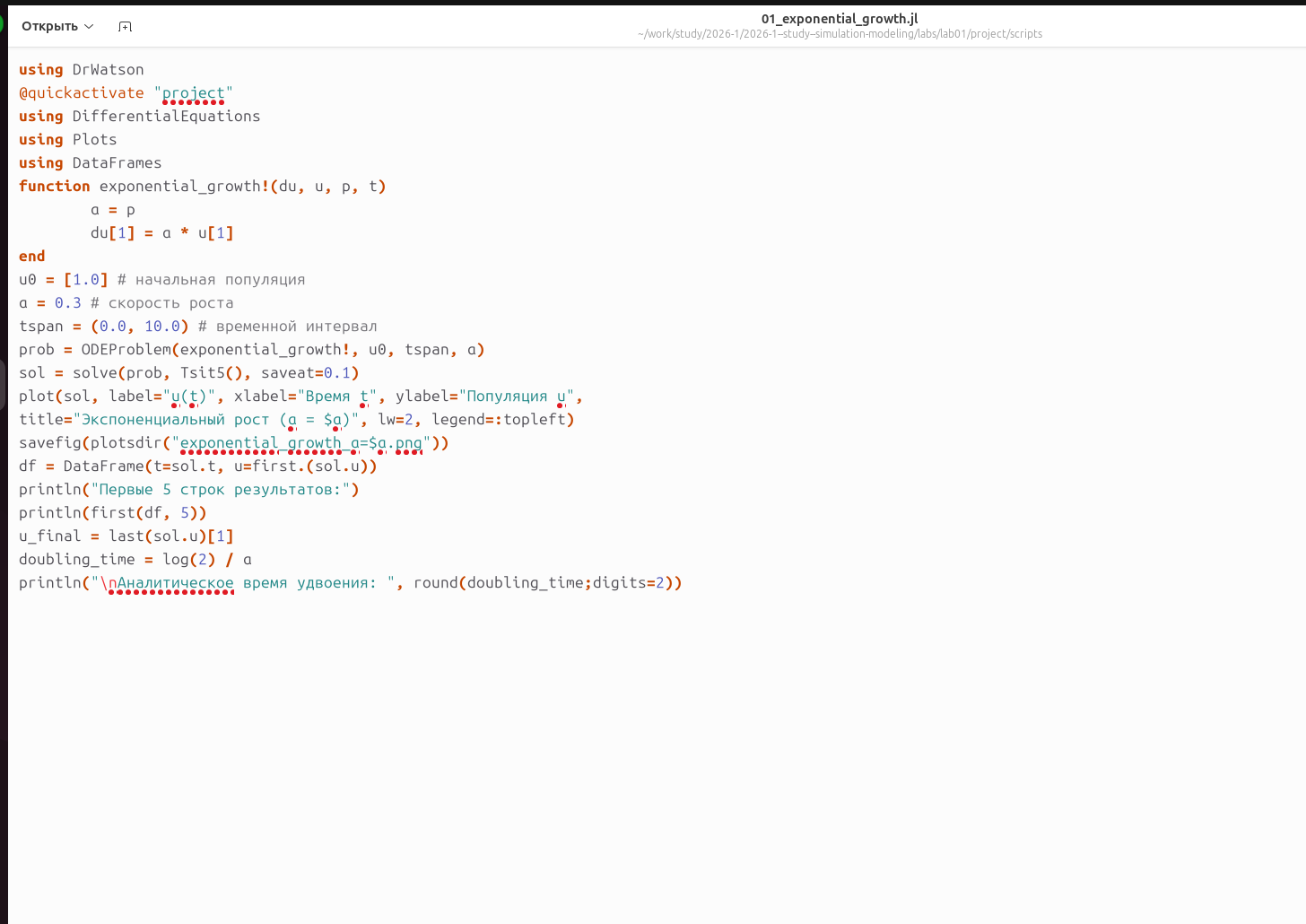
Проверка установки пакетов

## 3.5 Модель экспоненциального роста (базовый вариант)

Создан скрипт 01\_exponential\_growth.jl (рис. 34) с кодом модели (рис. 35). Скрипт реализует решение уравнения du/dt = α·u с начальными условиями u₀=1.0 и α=0.3.

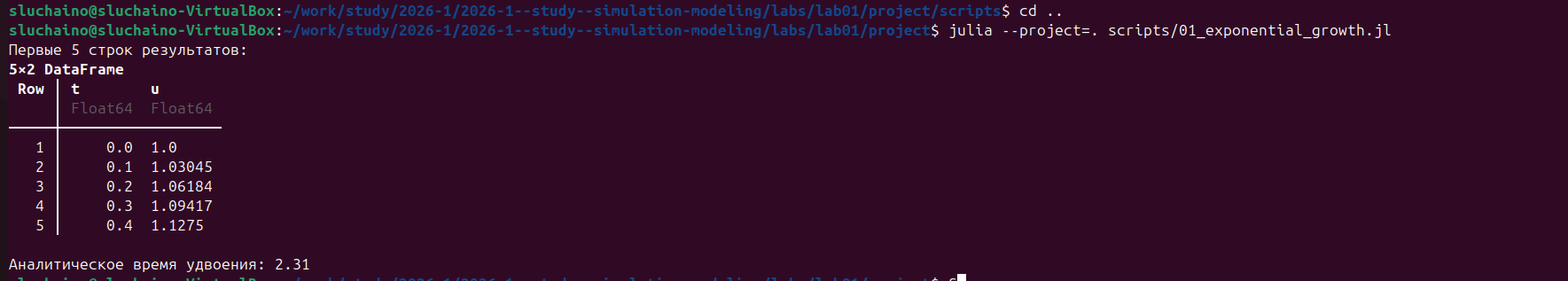
Создание скрипта

Создание скрипта



Код базовой модели

При запуске получены первые 5 строк результатов и вычислено аналитическое время удвоения (2.31) (рис. 36). Сгенерирован график экспоненциального роста (рис. 37).



Результаты базовой модели

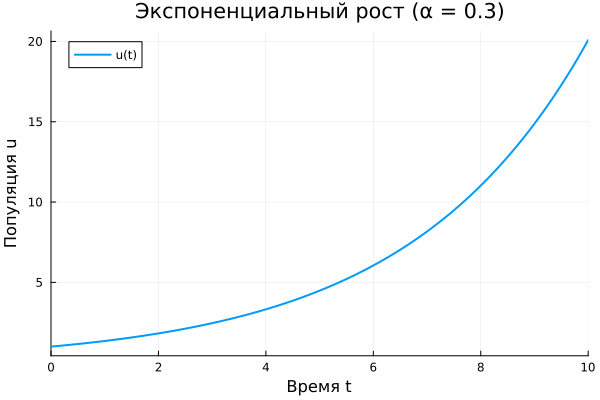
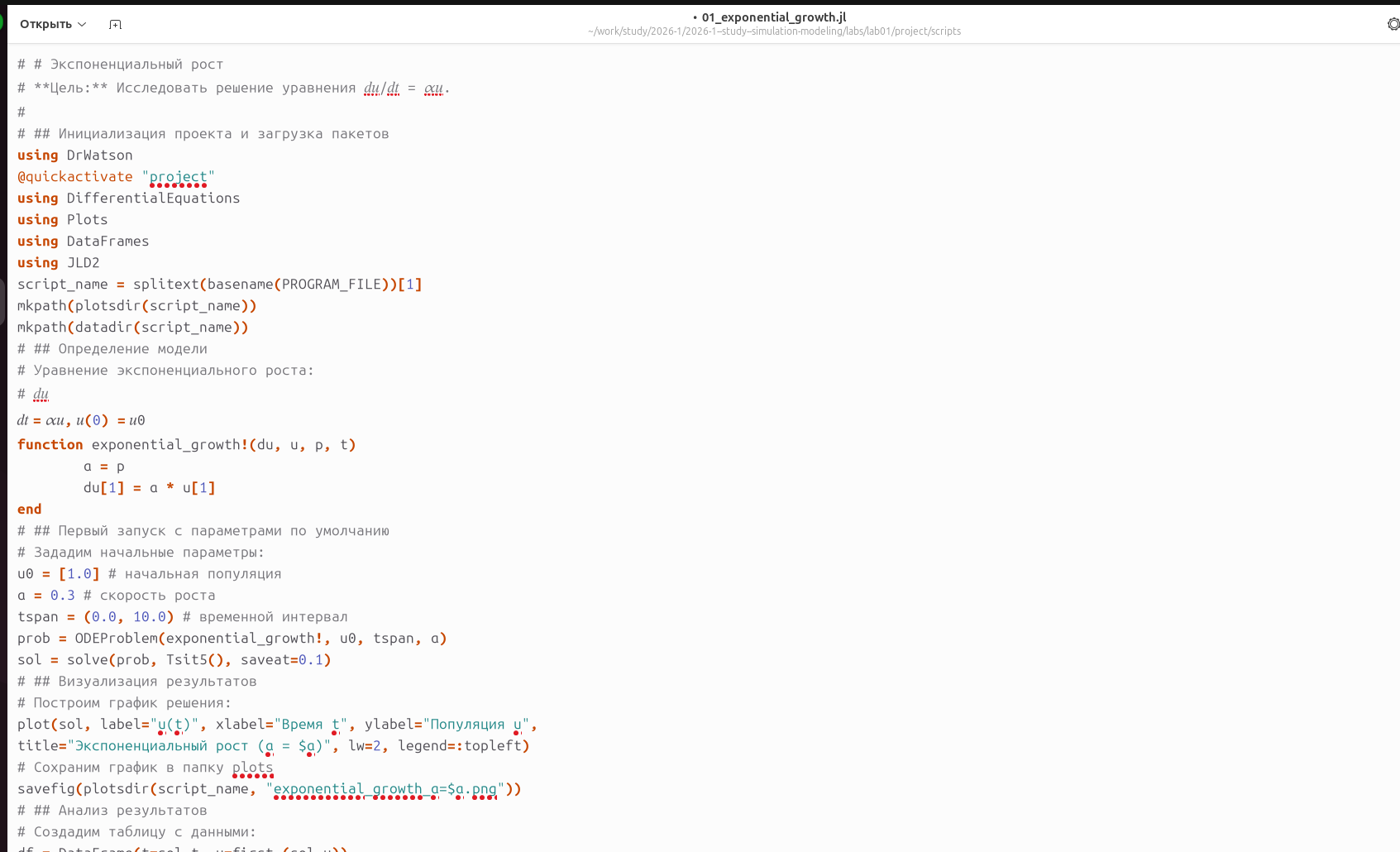


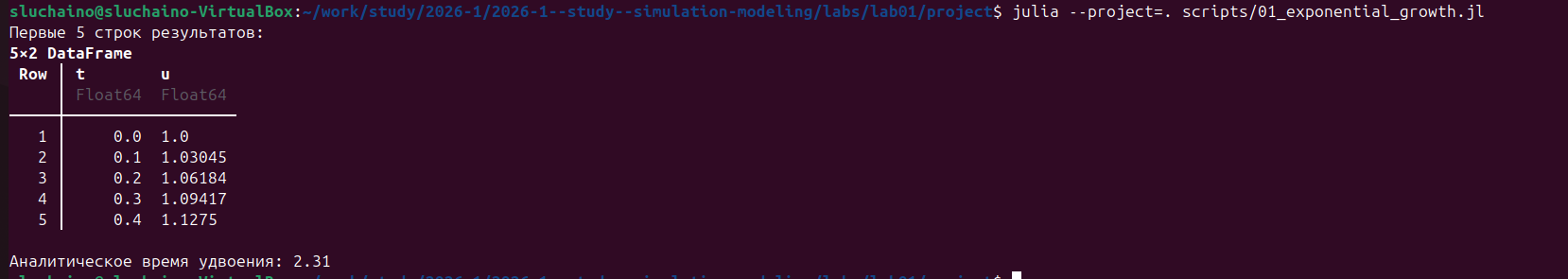
График экспоненциального роста

## 3.6 Литературный стиль

Скрипт преобразован в литературный стиль: добавлены комментарии с описанием модели, инициализацией проекта и анализом результатов (рис. 38). Запуск литературного скрипта дал аналогичные результаты (рис. 39-40).



Код в литературном стиле



Результаты литературного скрипта

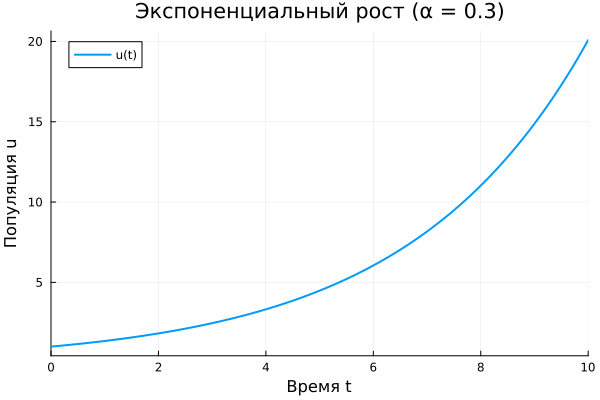


График из литературного скрипта

## 3.7 Генерация производных форматов

Создан скрипт tangle.jl для генерации чистого кода, Jupyter Notebook и Quarto-документации из литературных скриптов (рис. 41-42). Выполнена генерация для скрипта 01\_exponential\_growth.jl (рис. 43).

Создание tangle.jl

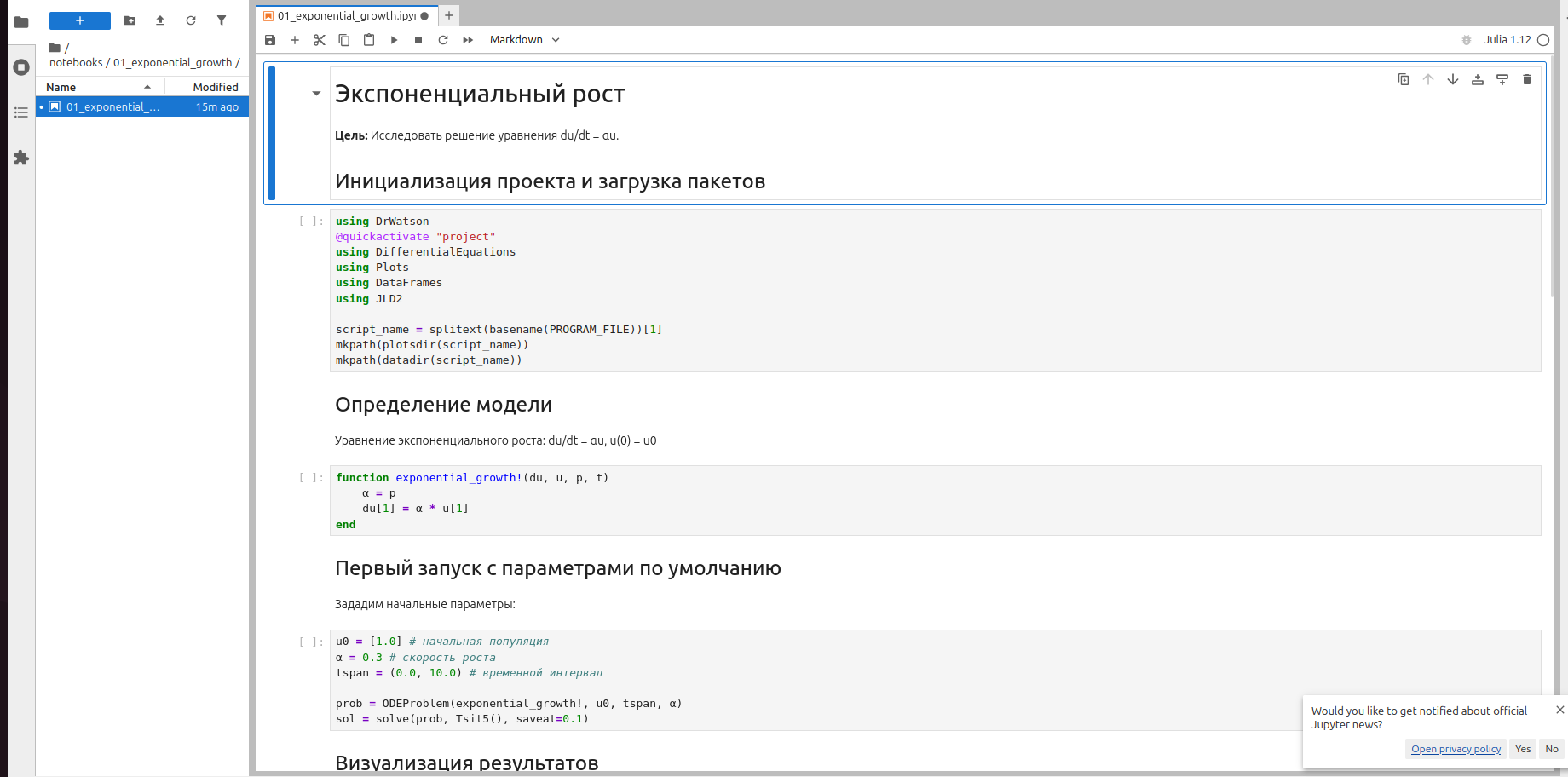
Создание tangle.jl



Код tangle.jl

## 3.8 Параметрическое исследование

Создан скрипт 02\_exponential\_growth.jl для исследования влияния параметра α (рис. 44). В файл \_quarto.yml добавлена поддержка Julia (рис. 45). Скрипт содержит параметрическое сканирование для значений α = 0.1, 0.3, 0.5, 0.8, 1.0 (рис. 46).

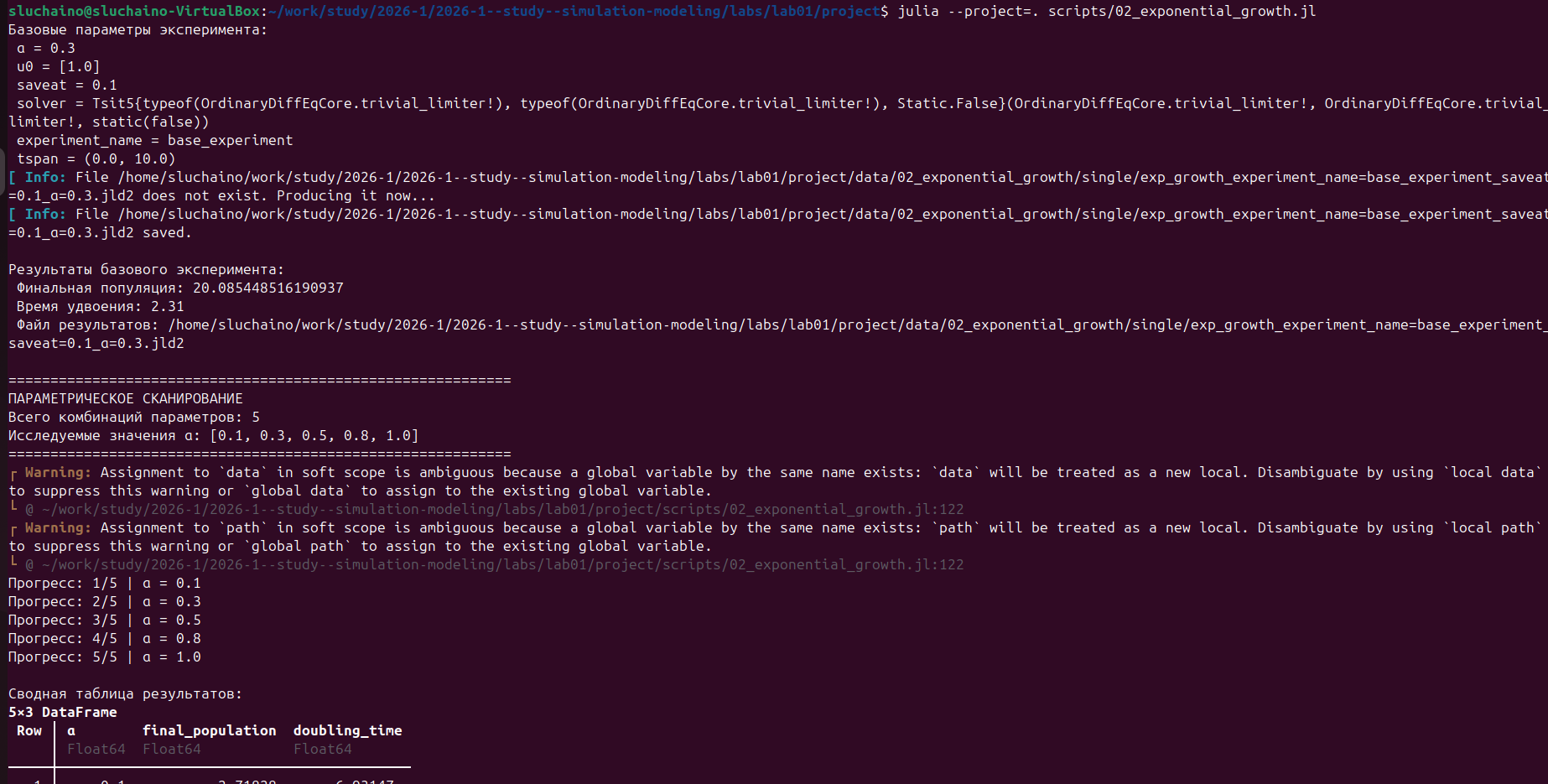


Код параметрического исследования

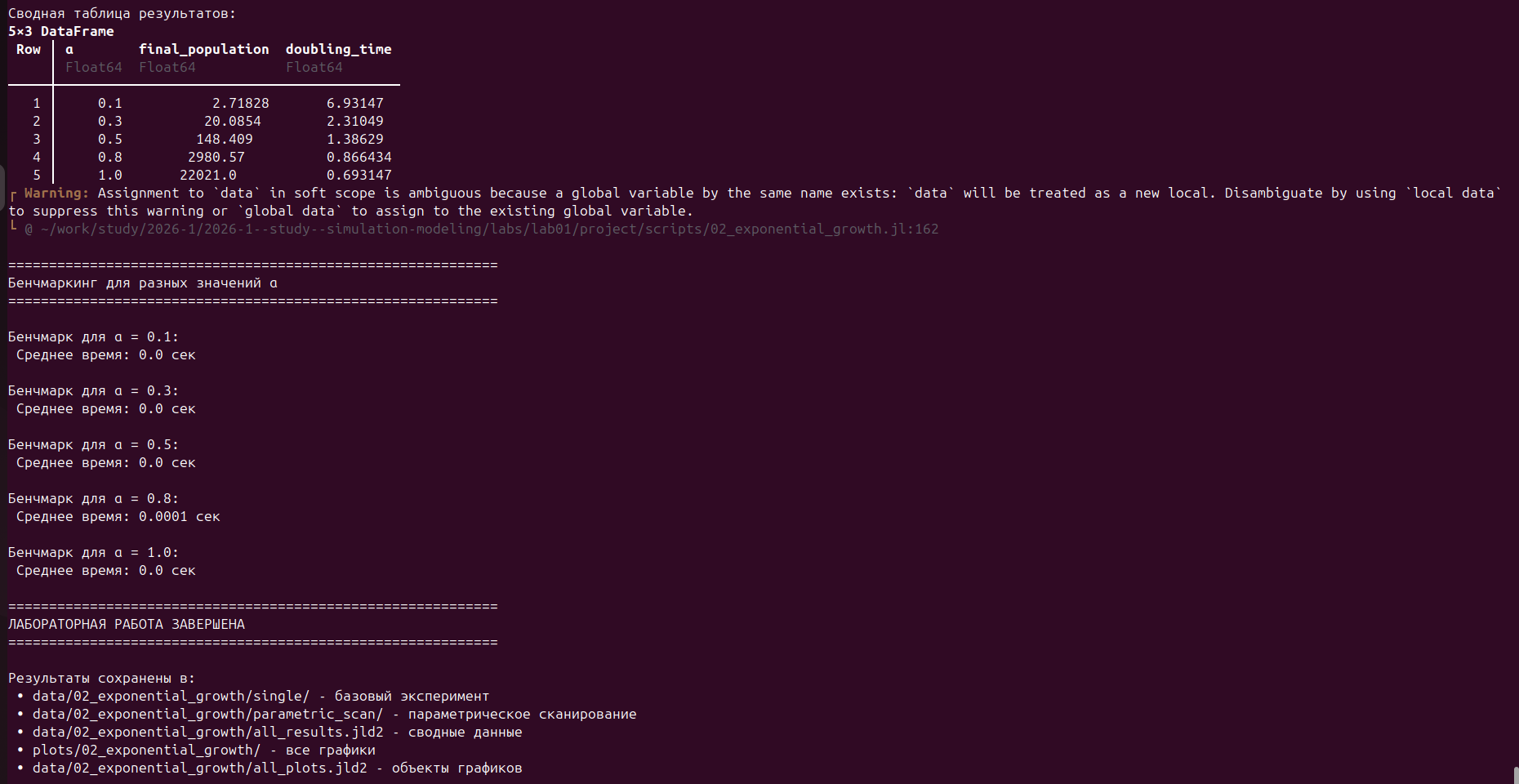


Конфигурация Quarto

Запуск параметрического скрипта показал процесс сканирования (рис. 47). Получена сводная таблица результатов: финальная популяция и время удвоения для каждого α (рис. 48).

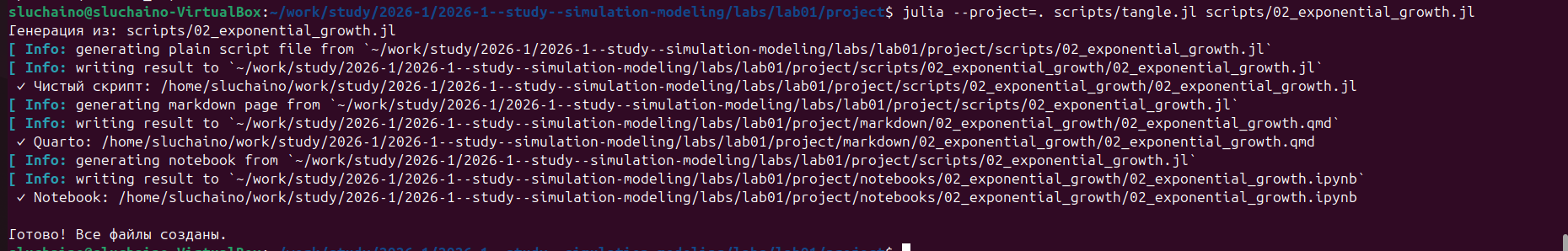


Процесс параметрического сканирования



Сводная таблица результатов

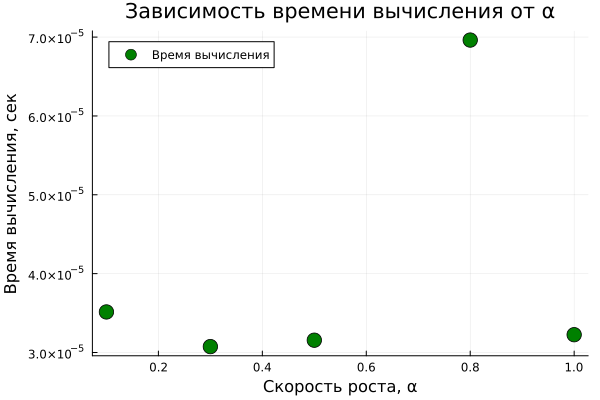
Выполнена генерация форматов для параметрического скрипта через tangle.jl (рис. 49).



Генерация форматов для параметрического скрипта

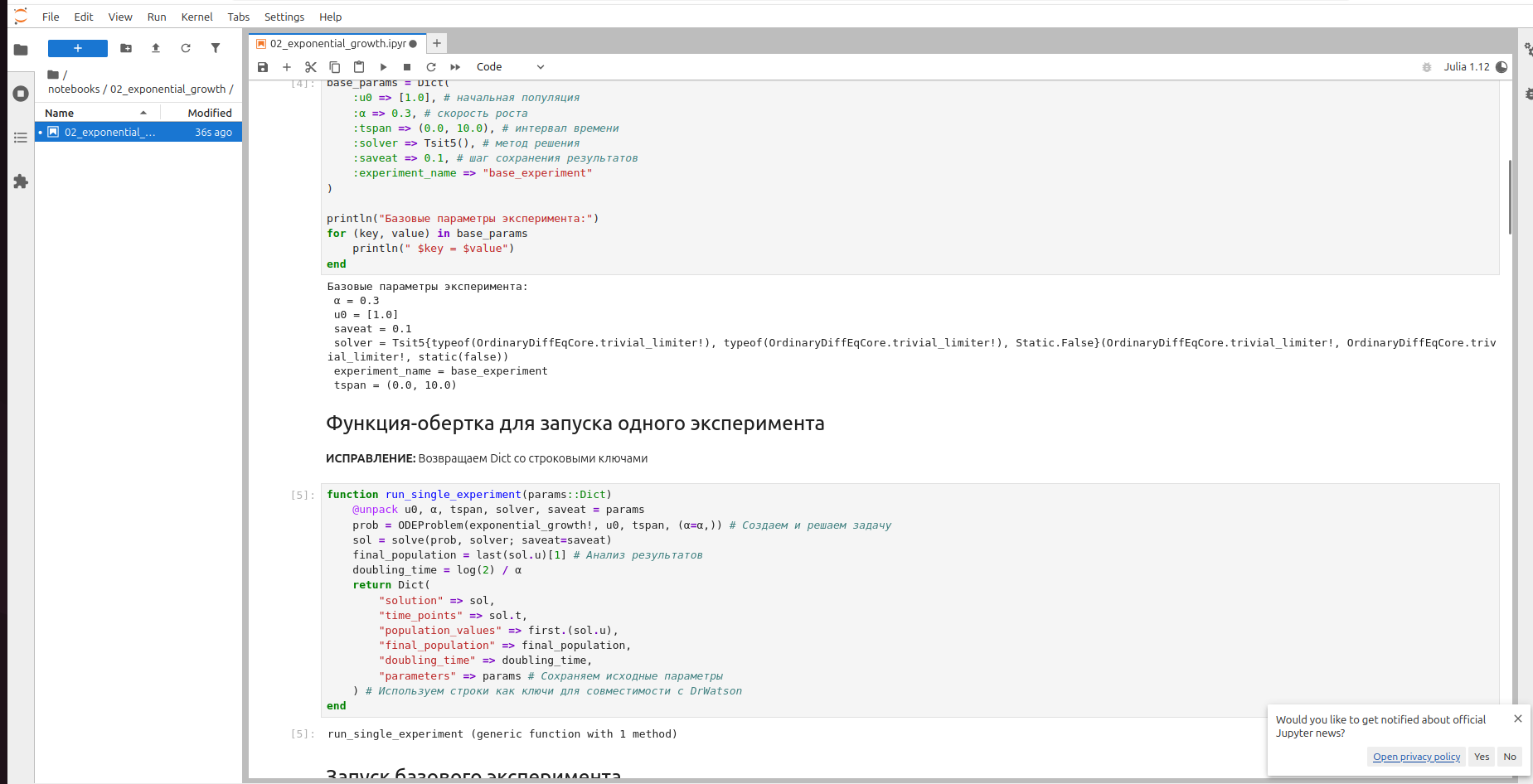
## 3.9 Визуализация результатов

Построены графики: зависимость времени вычисления от α (рис. 50), сравнительные данные для разных α (рис. 52), базовый эксперимент (рис. 53).

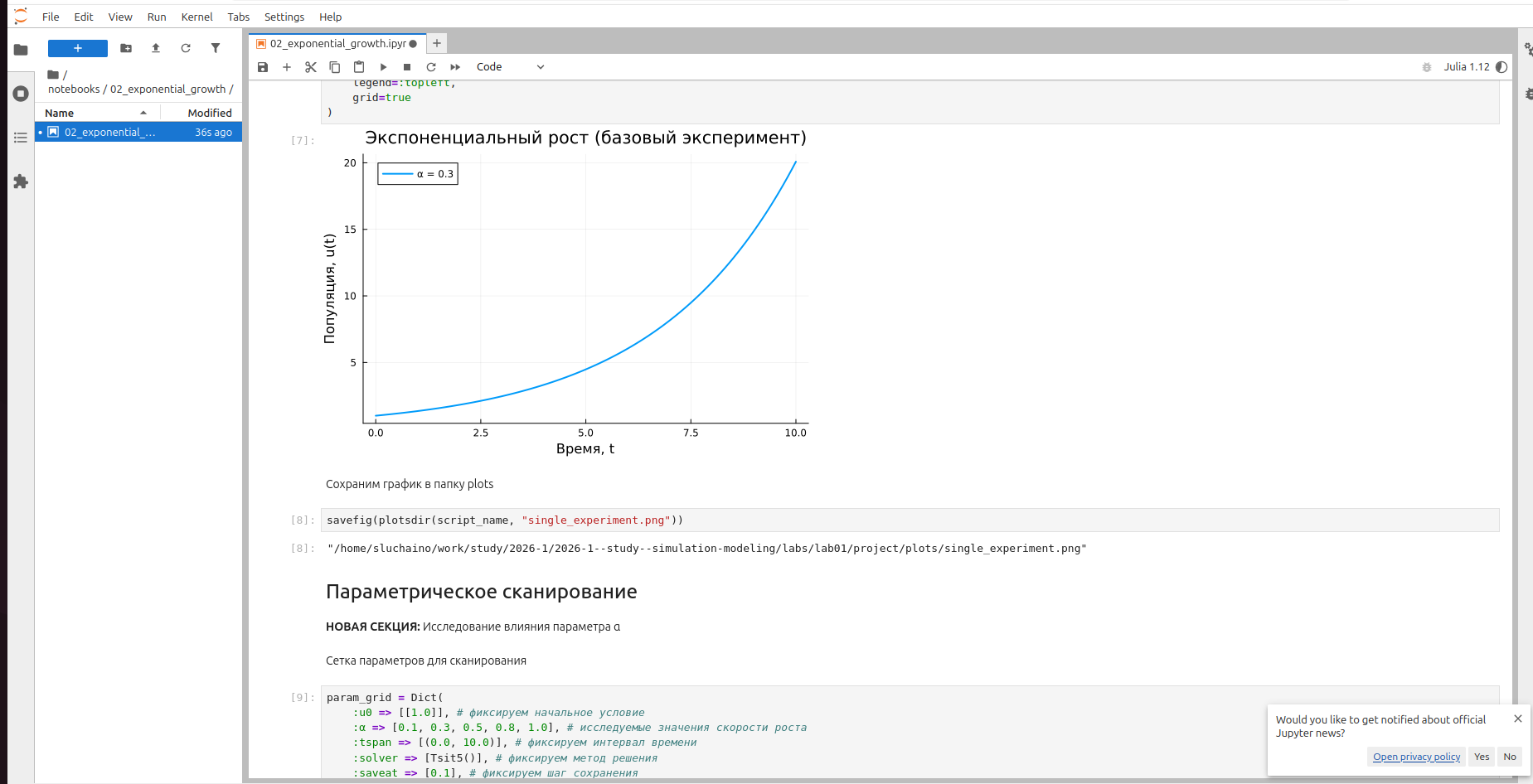


Зависимость времени вычисления от α

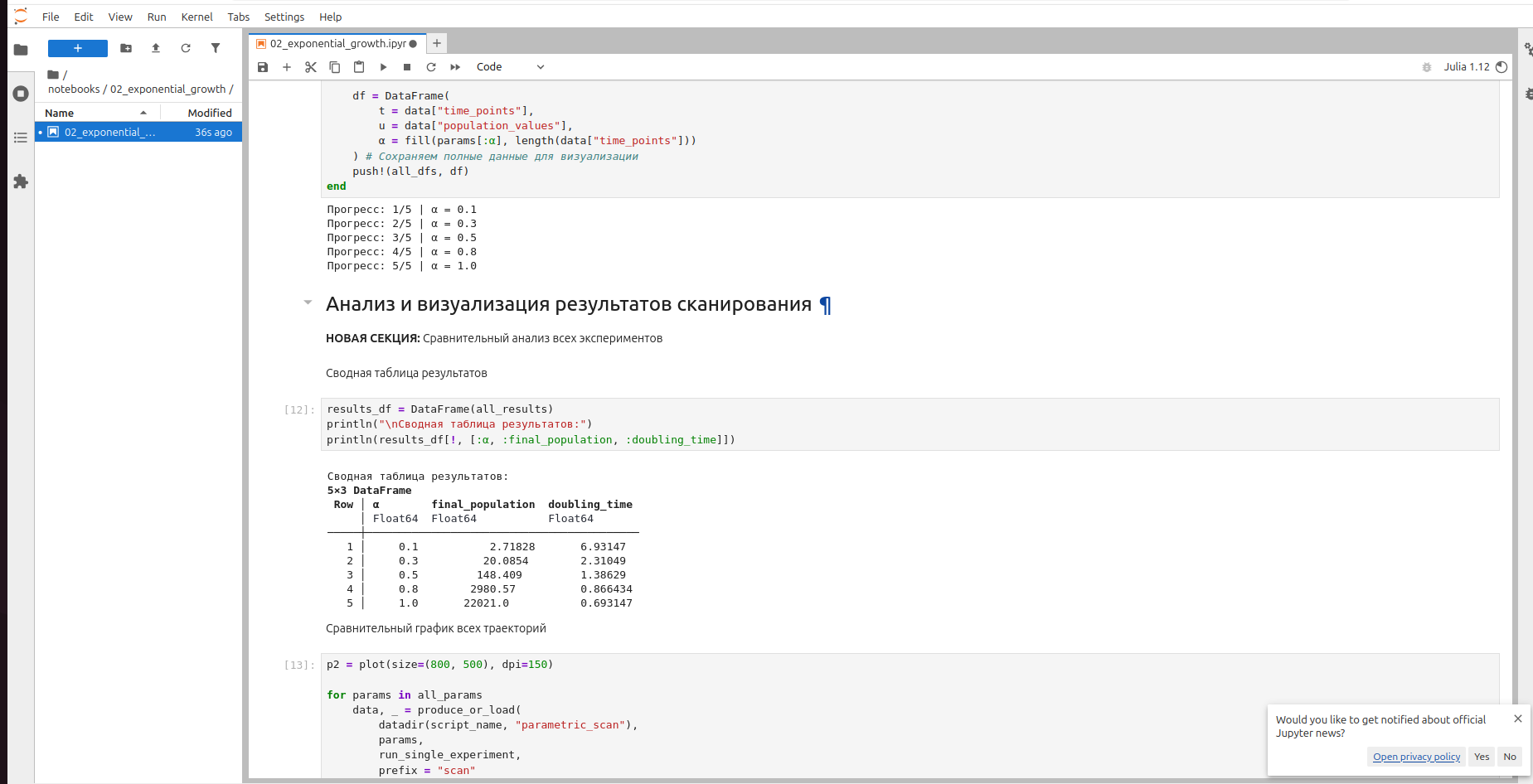
В Jupyter Notebook реализована функция-обёртка для запуска экспериментов (рис. 54-55). Выполнено сравнение всех траекторий для разных значений α (рис. 56).



Функция-обёртка в Jupyter

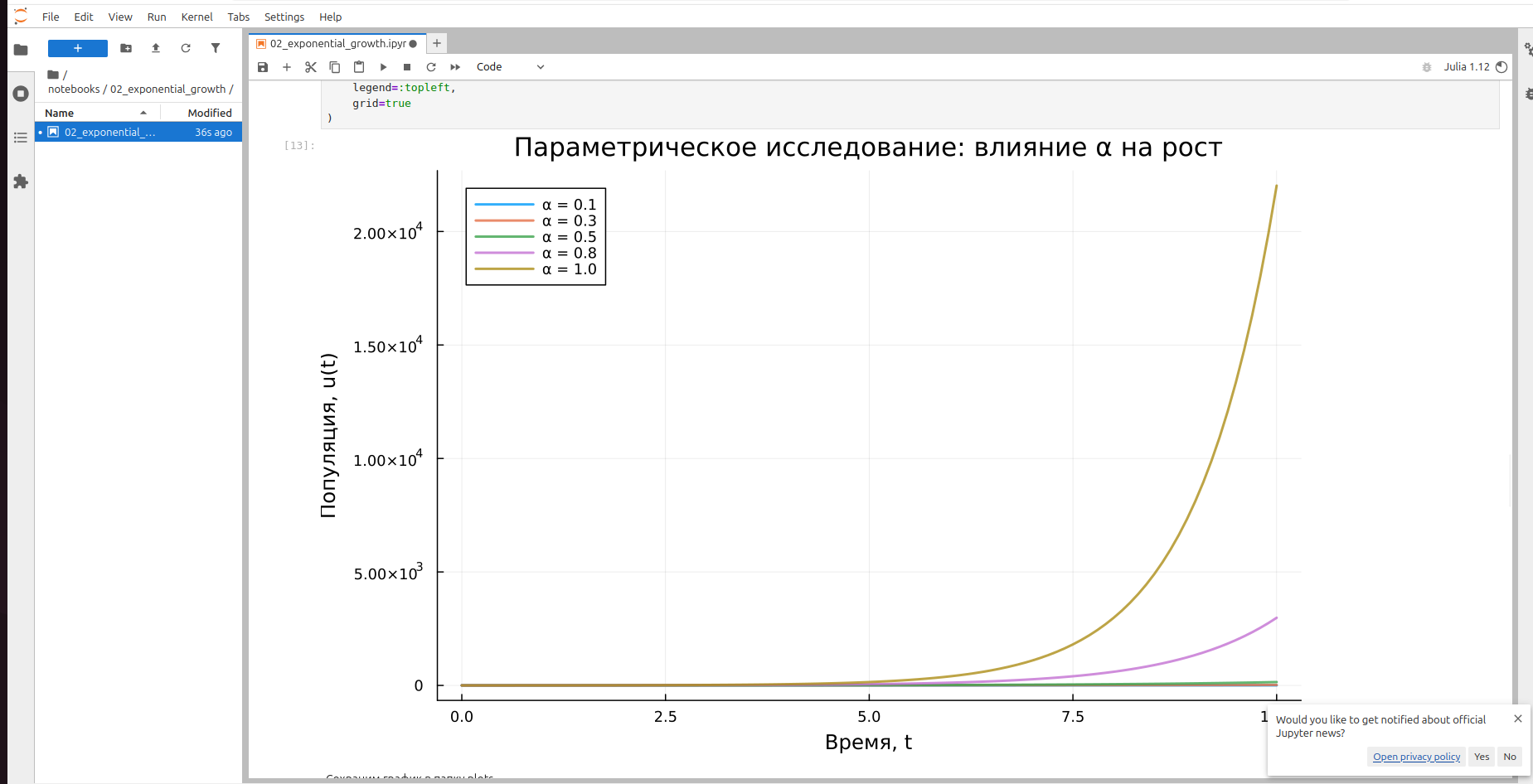


Параметрическое сканирование в Jupyter



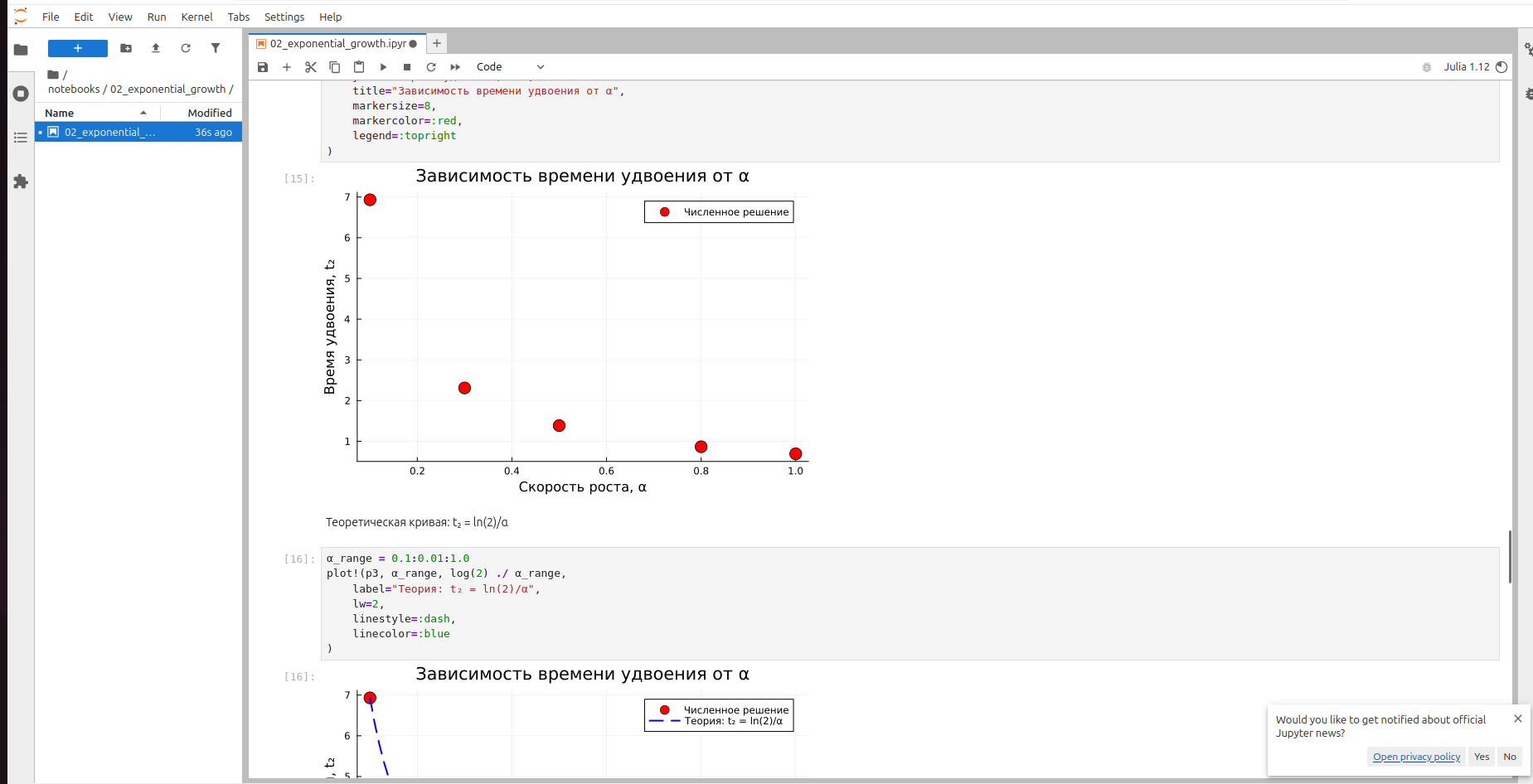
Сравнение траекторий

График параметрического исследования показывает, как меняется популяция при различных α (рис. 57). Чем больше α, тем быстрее рост.

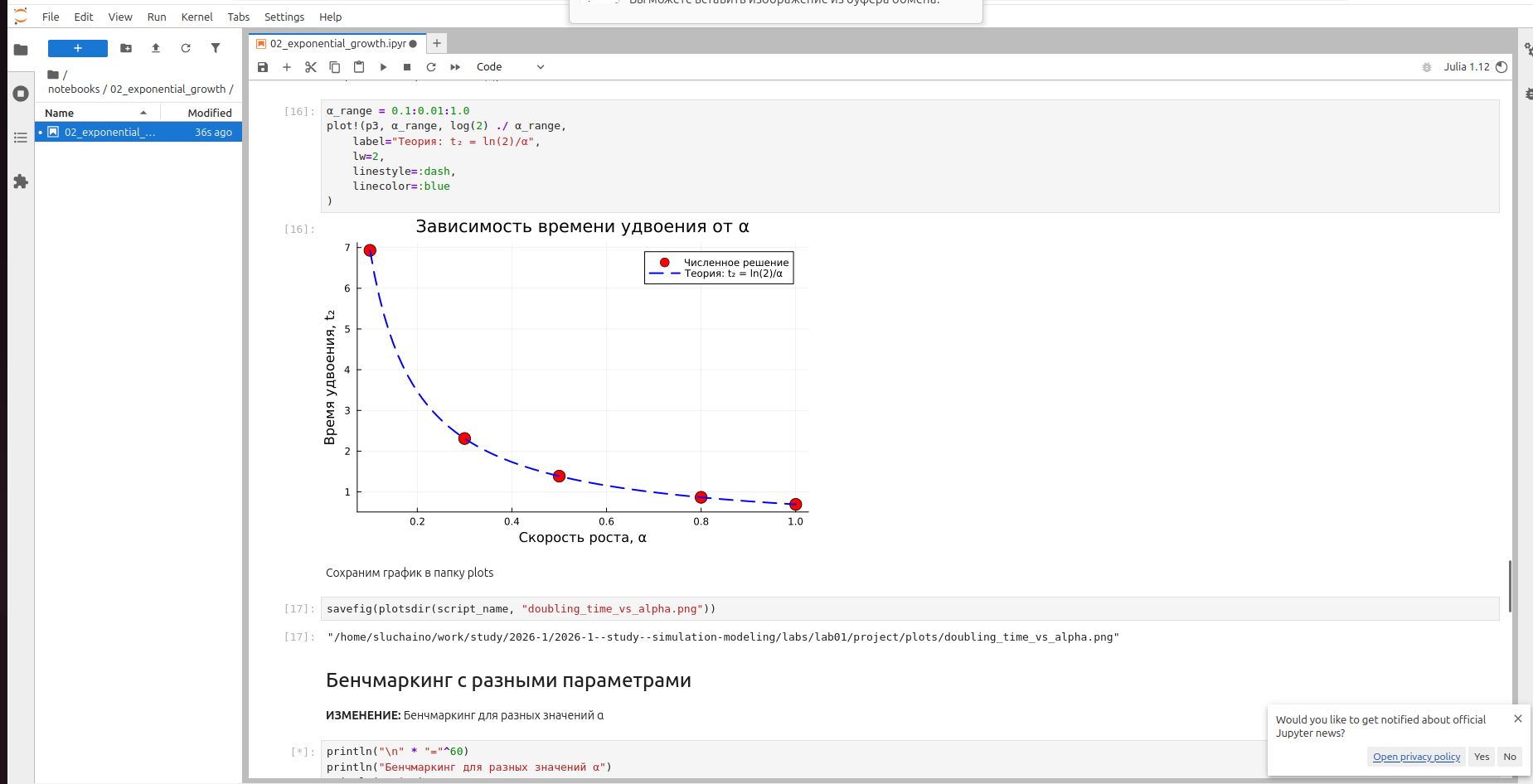


Параметрическое исследование

Построена зависимость времени удвоения от α (рис. 58-59). Точки численного решения совпадают с теоретической кривой t₂ = ln(2)/α.



Зависимость времени удвоения от α (1)



Зависимость времени удвоения от α (2)

## 3.10 Интеграция кода в отчёт

В файл отчёта подключены сгенерированные Quarto-документы:

# 4. Экспоненциальный рост

**Цель:** Исследовать решение уравнения du/dt = αu.

## 4.1 Инициализация проекта и загрузка пакетов

using DrWatson  
@quickactivate "project"  
using DifferentialEquations  
using Plots  
using DataFrames  
using JLD2  
  
script\_name = splitext(basename(PROGRAM\_FILE))[1]  
mkpath(plotsdir(script\_name))  
mkpath(datadir(script\_name))

## 4.2 Определение модели

Уравнение экспоненциального роста: du/dt = αu, u(0) = u0

function exponential\_growth!(du, u, p, t)  
 α = p  
 du[1] = α \* u[1]  
end

## 4.3 Первый запуск с параметрами по умолчанию

Зададим начальные параметры:

u0 = [1.0] # начальная популяция  
α = 0.3 # скорость роста  
tspan = (0.0, 10.0) # временной интервал  
  
prob = ODEProblem(exponential\_growth!, u0, tspan, α)  
sol = solve(prob, Tsit5(), saveat=0.1)

## 4.4 Визуализация результатов

Построим график решения:

plot(sol, label="u(t)", xlabel="Время t", ylabel="Популяция u",  
title="Экспоненциальный рост (α = $α)", lw=2, legend=:topleft)

Сохраним график в папку plots

savefig(plotsdir(script\_name, "exponential\_growth\_α=$α.png"))

## 4.5 Анализ результатов

Создадим таблицу с данными:

df = DataFrame(t=sol.t, u=first.(sol.u))  
println("Первые 5 строк результатов:")  
println(first(df, 5))

Вычислим удвоение популяции:

u\_final = last(sol.u)[1]  
doubling\_time = log(2) / α  
println("\nАналитическое время удвоения: ", round(doubling\_time; digits=2))

## 4.6 Сохранение всех результатов

@save datadir(script\_name, "all\_results.jld2") df

# 5. Параметрическое исследование экспоненциального роста

## 5.1 Активация проекта и загрузка пакетов

**ИЗМЕНЕНИЕ:** Добавлен DrWatson для управления проектом и параметрами

using DrWatson  
@quickactivate "project" # Активация проекта DrWatson  
using DifferentialEquations  
using DataFrames  
using Plots  
using JLD2  
using BenchmarkTools

Установка каталогов

script\_name = splitext(basename(PROGRAM\_FILE))[1]  
mkpath(plotsdir(script\_name))  
mkpath(datadir(script\_name))

## 5.2 Определение модели

Модель: du/dt = α ⋅ u

function exponential\_growth!(du, u, p, t)  
 α = p.α # \*\*ИЗМЕНЕНИЕ:\*\* Параметры теперь передаются как именованный кортеж  
 du[1] = α \* u[1]  
end

## 5.3 Определение параметров в Dict

**ОСНОВНОЕ ИЗМЕНЕНИЕ:** Все параметры собраны в Dict для систематизации Базовый набор параметров (один эксперимент)

base\_params = Dict(  
 :u0 => [1.0], # начальная популяция  
 :α => 0.3, # скорость роста  
 :tspan => (0.0, 10.0), # интервал времени  
 :solver => Tsit5(), # метод решения  
 :saveat => 0.1, # шаг сохранения результатов  
 :experiment\_name => "base\_experiment"  
)  
  
println("Базовые параметры эксперимента:")  
for (key, value) in base\_params  
 println(" $key = $value")  
end

## 5.4 Функция-обертка для запуска одного эксперимента

**ИСПРАВЛЕНИЕ:** Возвращаем Dict со строковыми ключами

function run\_single\_experiment(params::Dict)  
 @unpack u0, α, tspan, solver, saveat = params  
 prob = ODEProblem(exponential\_growth!, u0, tspan, (α=α,)) # Создаем и решаем задачу  
 sol = solve(prob, solver; saveat=saveat)  
 final\_population = last(sol.u)[1] # Анализ результатов  
 doubling\_time = log(2) / α  
 return Dict(  
 "solution" => sol,  
 "time\_points" => sol.t,  
 "population\_values" => first.(sol.u),  
 "final\_population" => final\_population,  
 "doubling\_time" => doubling\_time,  
 "parameters" => params # Сохраняем исходные параметры  
 ) # Используем строки как ключи для совместимости с DrWatson  
end

## 5.5 Запуск базового эксперимента

**ИЗМЕНЕНИЕ:** Используем produce\_or\_load для автоматического кэширования

data, path = produce\_or\_load(  
 datadir(script\_name, "single"), # Папка для сохранения  
 base\_params, # Параметры эксперимента  
 run\_single\_experiment, # Функция для выполнения  
 prefix = "exp\_growth", # Префикс имени файла  
 tag = false, # Не добавлять git-тег  
 verbose = true  
)  
  
println("\nРезультаты базового эксперимента:")  
println(" Финальная популяция: ", data["final\_population"])  
println(" Время удвоения: ", round(data["doubling\_time"]; digits=2))  
println(" Файл результатов: ", path)

## 5.6 Визуализация базового эксперимента

p1 = plot(data["time\_points"], data["population\_values"],  
 label="α = $(base\_params[:α])",  
 xlabel="Время, t",  
 ylabel="Популяция, u(t)",  
 title="Экспоненциальный рост (базовый эксперимент)",  
 lw=2,  
 legend=:topleft,  
 grid=true  
)

Сохраним график в папку plots

savefig(plotsdir(script\_name, "single\_experiment.png"))

## 5.7 Параметрическое сканирование

**НОВАЯ СЕКЦИЯ:** Исследование влияния параметра α

Сетка параметров для сканирования

param\_grid = Dict(  
 :u0 => [[1.0]], # фиксируем начальное условие  
 :α => [0.1, 0.3, 0.5, 0.8, 1.0], # исследуемые значения скорости роста  
 :tspan => [(0.0, 10.0)], # фиксируем интервал времени  
 :solver => [Tsit5()], # фиксируем метод решения  
 :saveat => [0.1], # фиксируем шаг сохранения  
 :experiment\_name => ["parametric\_scan"]  
)

Генерация всех комбинаций параметров

all\_params = dict\_list(param\_grid)  
  
println("\n" \* "="^60)  
println("ПАРАМЕТРИЧЕСКОЕ СКАНИРОВАНИЕ")  
println("Всего комбинаций параметров: ", length(all\_params))  
println("Исследуемые значения α: ", param\_grid[:α])  
println("="^60)

## 5.8 Запуск всех экспериментов и сбор результатов

**НОВАЯ СЕКЦИЯ:** Автоматический запуск и сохранение всех вариантов

all\_results = []  
all\_dfs = []  
  
for (i, params) in enumerate(all\_params)  
 println("Прогресс: $i/$(length(all\_params)) | α = $(params[:α])")  
  
 data, path = produce\_or\_load(  
 datadir(script\_name, "parametric\_scan"), # Данные  
 params, # Текущий набор параметров  
 run\_single\_experiment, # Функция для выполнения  
 prefix = "scan", # Префикс имени файла  
 tag = false,  
 verbose = false # Не выводить подробности для каждого запуска  
 ) # Автоматическое сохранение/загрузка каждого эксперимента  
  
 result\_summary = merge(  
 params,  
 Dict(  
 :final\_population => data["final\_population"],  
 :doubling\_time => data["doubling\_time"],  
 :filepath => path # Путь к сохраненным данным  
 )  
 ) # Сохраняем сводные результаты (используем символы для параметров, но данные из data - строки)  
  
 push!(all\_results, result\_summary)  
  
 df = DataFrame(  
 t = data["time\_points"],  
 u = data["population\_values"],  
 α = fill(params[:α], length(data["time\_points"]))  
 ) # Сохраняем полные данные для визуализации  
 push!(all\_dfs, df)  
end

## 5.9 Анализ и визуализация результатов сканирования

**НОВАЯ СЕКЦИЯ:** Сравнительный анализ всех экспериментов

Сводная таблица результатов

results\_df = DataFrame(all\_results)  
println("\nСводная таблица результатов:")  
println(results\_df[!, [:α, :final\_population, :doubling\_time]])

Сравнительный график всех траекторий

p2 = plot(size=(800, 500), dpi=150)  
  
for params in all\_params  
 data, \_ = produce\_or\_load(  
 datadir(script\_name, "parametric\_scan"),  
 params,  
 run\_single\_experiment,  
 prefix = "scan"  
 ) # Загружаем данные (они уже есть на диске)  
  
 plot!(p2, data["time\_points"], data["population\_values"],  
 label="α = $(params[:α])",  
 lw=2,  
 alpha=0.8  
 )  
end  
  
plot!(p2,  
 xlabel="Время, t",  
 ylabel="Популяция, u(t)",  
 title="Параметрическое исследование: влияние α на рост",  
 legend=:topleft,  
 grid=true  
)

Сохраним график в папку plots

savefig(plotsdir(script\_name, "parametric\_scan\_comparison.png"))

График зависимости времени удвоения от α

p3 = plot(results\_df.α, results\_df.doubling\_time,  
 seriestype=:scatter,  
 label="Численное решение",  
 xlabel="Скорость роста, α",  
 ylabel="Время удвоения, t₂",  
 title="Зависимость времени удвоения от α",  
 markersize=8,  
 markercolor=:red,  
 legend=:topright  
)

Теоретическая кривая: t₂ = ln(2)/α

α\_range = 0.1:0.01:1.0  
plot!(p3, α\_range, log(2) ./ α\_range,  
 label="Теория: t₂ = ln(2)/α",  
 lw=2,  
 linestyle=:dash,  
 linecolor=:blue  
)

Сохраним график в папку plots

savefig(plotsdir(script\_name, "doubling\_time\_vs\_alpha.png"))

## 5.10 Бенчмаркинг с разными параметрами

**ИЗМЕНЕНИЕ:** Бенчмаркинг для разных значений α

println("\n" \* "="^60)  
println("Бенчмаркинг для разных значений α")  
println("="^60)  
  
benchmark\_results = []  
  
for α\_value in param\_grid[:α]  
 bench\_params = Dict(  
 :u0 => [1.0],  
 :α => α\_value,  
 :tspan => (0.0, 10.0),  
 :solver => Tsit5(),  
 :saveat => 0.1  
 ) # Подготавливаем параметры для бенчмарка  
  
 function benchmark\_run() # Функция для бенчмарка  
 prob = ODEProblem(exponential\_growth!,  
 bench\_params[:u0],  
 bench\_params[:tspan],  
 (α=bench\_params[:α],))  
 return solve(prob, bench\_params[:solver];  
 saveat=bench\_params[:saveat])  
 end  
  
 println("\nБенчмарк для α = $α\_value:")  
 b = @benchmark $benchmark\_run() samples=100 evals=1 # Запуск бенчмарка  
 push!(benchmark\_results, (α=α\_value, time=median(b).time/1e9)) # время в секундах  
 println(" Среднее время: ", round(median(b).time/1e9; digits=4), " сек")  
end

График зависимости времени вычисления от α

bench\_df = DataFrame(benchmark\_results)  
  
p4 = plot(bench\_df.α, bench\_df.time,  
 seriestype=:scatter,  
 label="Время вычисления",  
 xlabel="Скорость роста, α",  
 ylabel="Время вычисления, сек",  
 title="Зависимость времени вычисления от α",  
 markersize=8,  
 markercolor=:green,  
 legend=:topleft  
)

Сохраним график в папку plots

savefig(plotsdir(script\_name, "computation\_time\_vs\_alpha.png"))

## 5.11 Сохранение всех результатов

**НОВАЯ СЕКЦИЯ:** Сохранение сводных данных для последующего анализа

@save datadir(script\_name, "all\_results.jld2") base\_params param\_grid all\_params results\_df bench\_df  
@save datadir(script\_name, "all\_plots.jld2") p1 p2 p3 p4  
  
println("\n" \* "="^60)  
println("ЛАБОРАТОРНАЯ РАБОТА ЗАВЕРШЕНА")  
println("="^60)  
println("\nРезультаты сохранены в:")  
println(" • data/$(script\_name)/single/ - базовый эксперимент")  
println(" • data/$(script\_name)/parametric\_scan/ - параметрическое сканирование")  
println(" • data/$(script\_name)/all\_results.jld2 - сводные данные")  
println(" • plots/$(script\_name)/ - все графики")  
println(" • data/$(script\_name)/all\_plots.jld2 - объекты графиков")  
println("\nДля анализа результатов используйте:")  
println(" using JLD2, DataFrames")  
println(" @load \"data/$(script\_name)/all\_results.jld2\"")  
println(" println(results\_df)")

# 6. Выводы

В ходе лабораторной работы было создано рабочее пространство для курса, настроены Git, GitHub и Git Flow. Установлены Julia и необходимые пакеты, реализована модель экспоненциального роста в базовом и параметрическом вариантах. Код оформлен в литературном стиле, сгенерированы Jupyter Notebook и Quarto-документация. Проведён анализ зависимости времени удвоения от параметра α, подтверждено совпадение численных результатов с теоретической кривой t₂ = ln(2)/α.